Universidade Federal da Paraíba
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

Uso de Conhecimento Preliminar na Melhoria do Aprendizado em um Modelo Simbólico-Conexionista

Edmundo Tojal Donato Júnior

Campina Grande - PB
Março – 1994

Edmundo Tojal Donato Júnior

Uso de Conhecimento Preliminar na Melhoria do Aprendizado em um Modelo Simbólico-Conexionista

Dissertação apresentada ao curso de MESTRADO EM INFORMÁTICA da Universidade Federal da Paraíba, em cumprimento às exigências para a obtenção do Grau de Mestre.

Área de Concentração: Ciência da Computação

Giuseppe Mongiovi
(*Orientador*)

Campina Grande - PB
Março – 1994

D677u Donato Júnior, Edmundo Tojal.
Uso de conhecimento preliminar na melhoria do aprendizado em um modelo simbólico-conexionista / Edmundo Tojal Donato Júnior. - Campina Grande, 1994.
136 f.

Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1994.
"Orientação : Prof. Giuseppe Mongiovi".
Referências.

1. Rede Neural. 2. Modelo Neural Combinatório (MNC). 3. MNC - Aprendizagem - Melhoria. 4. Dissertação - Informática. I. Mongiovi, Giuseppe. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.032.26(043)

USO DE CONHECIMENTO PRELIMINAR NA MELHORIA DO APRENDIZADO EM UM MODELO SIMBÓLICO-CONEXIONISTA

EDMUNDO TOJAL DONATO JÚNIOR

DISSERTAÇÃO APROVADA EM 11.03.1994

GIUSEPPE MONGIOVI, M. Sc.
Orientador

GENTIL JOSÉ DE LUCENA FILHO, Ph. D.
Componente da Banca

BERNARDO LULA JÚNIOR, Docteur
Componente da Banca

EDILSON FERNEDA, Docteur
Componente da Banca

Campina Grande , 11 de março de 1994.

Agradecimentos

A Deus, pela vida, pelos dons, pela força de vontade, pela perseverança.

A meu pai, minha mãe, irmão e irmã, pela paciência, pelo apoio, carinho e participação.

A Giuseppe Mongiovi, pela confiança, serenidade e amizade.

Aos amigos Walfredo, Vasco e Marcos, pelo companheirismo e colaboração.

A Ricardo Machado (IBM / Centro Científico Rio), pela atenção e gentileza.

Aos colegas e professores do curso de graduação e mestrado, pela boa convivência.

À namorada, pelos momentos de alegria e tranquilidade.

Àquela que a falta de tempo, noutro tempo, não me deixou namorar, pela compreensão.

Ao Coração, por suportar tão bravamente tamanha ansiedade.

Ao Id e ao SuperEgo, pelas tréguas difíceis e muito necessárias; pelo adiamento da ruptura.

Aos meus alunos do Curso de Computação da UFAL, pelos momentos de descontração.

A Aninha, da COPIN, pela simpatia e presteza.

Ao CNPq, pelo apoio financeiro.

A todos que, de alguma forma, contribuíram para o desenvolvimento deste trabalho e para o crescimento que experimentei durante sua execução.

Resumo

O Modelo Neural Combinatório (**MNC**) é uma rede neural de ordem superior adequada para tarefas de classificação. O treinamento dessa rede é feito por um algoritmo baseado no *backpropagation*, através de punições e recompensas, de acordo com os exemplos do conjunto de treinamento. Este trabalho destaca a importância do conhecimento preliminar no aprendizado, de uma forma geral, e descreve como a relevância semântica e a valoração de atributos podem melhorar a qualidade do aprendizado realizado pelo MNC e o desempenho do algoritmo de treinamento. As modificações apresentadas atuam sobre a topologia da rede a ser treinada e são compatíveis com as extensões que vêm sendo propostas ao Modelo.

Abstract

The Combinatorial Neural Model (**MNC**) is a neural network of superior order that is adequate to classification tasks. The training of this network is performed by an algorithm based on backpropagation, through punishments and rewards, according to the examples of the training set. This work detaches the importance of the background knowledge to learning process in a general form and describes how semantic relevance and attribute valoration can improve both the quality of learning accomplished by MNC and the performance of the training algorithm. The presented alterations actuate on the topology of the network to be trained and are compatible with the extensions that have been proposed to the Model.

Sumário

Sumário

Sumário

1 - Introdução

A Inteligência Artificial (**IA**) é uma disciplina relativamente recente, nascida no final da década de 50, que procura reproduzir em máquinas o comportamento inteligente de seres humanos [Rich 83].

1.1 - Sistemas Baseados em Conhecimento

Atualmente, as pesquisas em IA têm se direcionado no sentido de procurar métodos específicos de solução de problemas, dependentes do domínio do conhecimento e do tipo do problema. Esta direção se deve ao insucesso da busca por métodos gerais, ocorrida nas décadas de 60 e 70 [Jackson 86], quando se buscava desenvolver um resolvedor universal de problemas (GPS - *General Problem Solver*). Dessa experiência, percebeu-se que um grande poder de inferência não seria bastante para se desenvolver um sistema "inteligente" de utilidade prática, pois a quantidade e qualidade do conhecimento manipulado pelo sistema eram pelo menos tão importante quanto o poder de inferência. Devido a isso, os programas passaram a ser desenvolvidos com ênfase no conhecimento, o que deu origem aos Sistemas Baseados em Conhecimento (**SBC**). Os SBC são programas construídos com o cuidado de se separar claramente o conhecimento do código que manipula esse conhecimento, i.e., que realiza inferências sobre o mesmo. Essa separação é a principal característica de um SBC; ela evidencia a importância do conhecimento dentro do sistema e facilita a sua representação, validação e manutenção.

1.2 - Sistemas Especialistas

O uso da abordagem de SBC para a construção de programas de IA representou um avanço importante na técnica de construção de programas que apresentam comportamento "inteligente". Com o surgimento dos primeiros SBC, chegou-se a pensar na possibilidade de se construir um sistema que simulasse o comportamento de um ou de vários seres humanos. Uma boa máquina de inferência,

baseada em técnicas formais de dedução e indução, tendo a sua disposição um grande banco de conhecimento, poderia se comportar como um generalista, i.e., um "sabe-tudo". Contudo, essa empreitada logo mostrou-se de difícil realização. O tamanho de um banco de conhecimento generalista e o tamanho do espaço de busca enfrentado pela máquina de inferência assumiriam proporções intratáveis. Isso pode ser percebido sem muito esforço.

Assim, mesmo com o uso da técnica de SBC, notou-se uma dificuldade tremenda para se construir sistemas que executassem algumas atividades aparentemente banais para o ser humano. Na verdade, qualquer raciocínio que envolvesse o "bom senso" das pessoas ficava impossibilitado de ser realizado devido à impossibilidade de se prover um banco de conhecimento generalista, que englobasse todo o conhecimento de senso comum. Um banco de conhecimento desses exigiria uma capacidade de armazenamento e um poder de inferência incomensuráveis.

Devido a essas dificuldades, optou-se por construir sistemas baseados em conhecimento para trabalhar em domínios menores e bem delimitados, como em áreas específicas da Medicina, Agronomia e Geologia, entre outras. A esses sistemas, de implementação factível e utilidade efetiva em aplicações reais, convencionou-se chamar Sistemas Especialistas (**SE**) [Waterman 86].

Um sistema especialista é formado por três componentes principais: a máquina de inferência, a base de conhecimento e a interface com o usuário. O desenvolvimento de um sistema especialista pode ser dividido em fases, assim como outros tipos de programas. McGraw [McGraw 89] sugere as seguintes fases: identificação das características do problema, conceituação do domínio, organização e formalização do conhecimento, implementação, testes, validação e manutenção do sistema.

De acordo com o tipo de problema que resolvem, os sistemas especialistas podem ser divididos em dois grandes grupos, quais sejam: sistemas de análise e de síntese [Boose 90]. Geralmente, os problemas de análise envolvem a identificação de

conjuntos de objetos baseada em suas características. Uma característica importante dos problemas de análise é que se pode enumerar um conjunto com todas as soluções possíveis e inclui-lo no sistema. Já os problemas de síntese requerem que a solução seja construída a partir de componentes ou de soluções de subproblemas. Nos problemas de síntese, há muitas soluções em potencial, o que impossibilita a sua enumeração e inclusão explícita no sistema. Neste trabalho, nos deteremos nos aspectos ligados à implementação de sistemas que resolvem um tipo particular de problema de análise: os sistemas classificatórios.

A semelhança existente entre os sistemas de um mesmo grupo possibilita a existência de ferramentas de apoio ao seu desenvolvimento. A ajuda proporcionada por essas ferramentas, que são conhecidas como *shells*, é muito importante para as fases relacionadas à implementação, possibilitando, inclusive, a adoção da forma de desenvolvimento baseada em prototipação e versões sucessivas [Fairley 85]. A utilização de um *shell* dispensa maiores preocupações com a implementação de dois dos três componentes do sistema especialista: a máquina de inferência e a interface com o usuário. Isto permite a concentração das atenções na confecção da base de conhecimento, que é um trabalho bastante complexo, que engloba a análise do domínio, a análise do problema e a formalização do conhecimento necessário para resolver os problemas desejados.

1.3 - Aquisição de Conhecimento

Os sistemas especialistas são SBC construídos para trabalhar na solução de problemas específicos, em domínios bem delimitados. Pelo fato de também ser um SBC, um SE tem sua construção fortemente dependente do processo de identificação e transferência do conhecimento especializado para um formalismo de representação de conhecimento adequado, "compreensível" – ou manipulável – pela máquina. Este processo de construção da base de conhecimento, conhecido como aquisição de conhecimento, passou a ser um dos mais importantes alvos de pesquisa em IA. Isto ocorreu devido ao fato de a aquisição de conhecimento ser uma tarefa difícil, dispendiosa, mas de extrema importância na qualidade final dos SE. De fato, essas

características são tão marcantes que fazem dela o gargalo ou ponto de estrangulamento do processo de construção de sistemas especialistas [Feigenbaum 82].

Uma alternativa promissora para se amenizar os problemas inerentes à aquisição de conhecimento, tornando-a mais simples e menos onerosa, é a sua automatização, que pode se dar em vários níveis. De acordo com o nível de automatização empregado, a aquisição de conhecimento pode ser: cognitiva, semi-automática ou automática (figura 1.1).

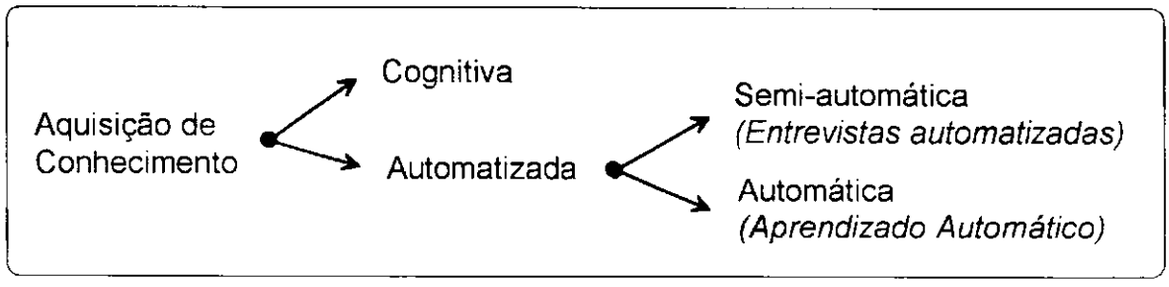


Figura 1.1 – Formas de aquisição de conhecimento, de acordo com o nível de presença do homem e da máquina no processo.

A aquisição de conhecimento é dita **cognitiva** quando é desempenhada de uma forma intrinsecamente manual por um engenheiro de conhecimento, através de pesquisa a documentos, observação de procedimentos e entrevistas com especialistas. Existem várias técnicas que procuram facilitar a eliciação do conhecimento de um especialista ou de um grupo deles. Entre essas técnicas, podemos citar: a análise de protocolos, as entrevistas estruturadas, não estruturadas e semi-estruturadas, as entrevistas com *teachback* e o *brainstorming* [McGraw 89].

Apesar dessas técnicas, a aquisição de conhecimento cognitiva depende muito do relacionamento pessoal entre o engenheiro de conhecimento e o(s) especialista(s) envolvido(s). A importância desse relacionamento é tão grande que alguns autores procuram estabelecer os tipos psicológicos ideais de especialistas e engenheiros de conhecimento, de forma a otimizar o processo de aquisição [Leão 88].

Mas, não é somente a relação pessoal entre os profissionais envolvidos que determina o grau de facilidade do processo. Há ainda a barreira da linguagem e da natureza do conhecimento especialista. A linguagem, recheada de termos técnicos específicos a cada especialidade, tem que ser assimilada pelo engenheiro de conhecimento. Além disso, a natureza compacta do conhecimento heurístico do especialista, ao mesmo tempo que lhe confere grande eficiência no dia-a-dia, também prejudica a sua comunicação. Sobre isso, [Leão 88] afirma que "parece haver uma relação inversa entre o grau de experiência e a conscientização dos níveis intermediários de um processo cognitivo".

O engenheiro de conhecimento se coloca como um intermediário entre a fonte do conhecimento e a base de conhecimento. A representação do conhecimento no formalismo adequado ao sistema especialista é realizada pelo engenheiro de conhecimento. O problema dessa intermediação é que ela pode introduzir erros no banco de conhecimento porque, em última instância, o conhecimento lá representado é o conhecimento do especialista de acordo com o que entendeu o engenheiro de conhecimento.

É justamente esse problema que a aquisição de conhecimento **semi-automática** procura resolver. Nela, a participação do engenheiro de conhecimento é diminuída, pois as entrevistas com os especialistas são realizadas por programas de computador, que guiam o especialista em sessões de exposição e formalização de conhecimento. Exemplos de ambientes que implementam entrevistas automatizadas são ETS [Boose 86] e AQUINAS [Boose 88]. Nesses ambientes, o conhecimento é eliciado na forma de Redes de Repertório, técnica baseada na Psicologia de Construções Pessoais [Kelly 55]. A idéia por detrás desses entrevistadores automáticos é solicitar do especialista características que possam fazer a distinção entre as classes de objetos dentro do domínio. Após essa fase inicial, o especialista é convidado a preencher uma Rede de Repertório com pesos de forma a identificar as características que cada classe apresenta.

As diferentes formas de aquisição de conhecimento maciçamente dependentes da presença de especialistas esbarram em problemas como a dificuldade de caracterização e identificação de especialistas e a escassez de tempo dos mesmos, além daqueles relacionados à transferência do conhecimento do especialista para outro agente.

O maior nível de automatização da aquisição de conhecimento é o aprendizado por máquina ou **aprendizado automático (AA)** (*Machine Learning*) [Michalski 87]. Esse nível de automatização é obtido com o uso de algoritmos que extraem conhecimento a partir de exemplos, de textos, etc. O objetivo aqui é o de refazer o aprendizado e não o de transferir conhecimento que já está armazenado de forma compacta, boa para ser usada e não para ser transmitida. Cabe ao algoritmo de aprendizado automático refazer o aprendizado a partir da observação de casos reais, podendo contar com a ajuda de um mestre que lhe fornece casos mais representativos e lhe guia pelo vasto espaço de possibilidades.

Uma questão que surge para a alternativa de aprendizado automático refere-se à representação do conhecimento adquirido: ela deve ser inteligível também para o ser humano ou apenas para a máquina? A resposta a isto depende da finalidade do aprendizado. Se o objetivo for construir um sistema especialista, deve-se lembrar que o mesmo deve ser capaz de apresentar explanações sobre a inferência realizada. Isso geralmente leva à necessidade de se ter o conhecimento representado numa forma inteligível para o ser humano. Neste trabalho, nos concentraremos na aquisição de conhecimento para a construção de sistemas especialistas. Assim, os métodos de aprendizado automático abordados devem gerar conhecimento inteligível para o ser humano.

A automatização da aquisição de conhecimento não descarta a presença de seres humanos na construção de sistemas especialistas. Mesmo no nível de maior automatização apresentado, o ambiente de aquisição deve ser operado por alguém. Além disso, na maioria das vezes, o mundo real precisa ser modelado numa forma manipulável pela máquina e que satisfaça às exigências do algoritmo de aprendizado

[Vasco 92b]. Na verdade, o homem e a máquina constituem um ótimo conjunto para a realização de tarefas complexas. Assim, o aprendizado automático não deve ser visto como uma técnica que substitui totalmente a aquisição de conhecimento cognitiva. A melhor forma de se realizar a aquisição de conhecimento, se existe, certamente é um misto de técnicas manuais e automatizadas.

1.4 - Paradigmas de Aprendizado Automático

Existem diversas taxonomias para as pesquisas em aprendizado automático. Os critérios de classificação consideram, em geral, as seguintes características: a estratégia de aprendizado adotada, o tipo de conhecimento adquirido e o domínio de aplicação [Carbonell 83]. Em um trabalho posterior, Carbonell passa a considerar a existência de quatro grandes paradigmas de aprendizado automático: Indutivo, Analítico, Genético e Conexionista [Carbonell 89] [Denis 91].

Um paradigma apresenta uma forma consistente de abordar um problema através da escolha de aspectos relevantes do mesmo, baseada numa linha mestra de raciocínio e atuação. Para nossos propósitos, o estudo desses paradigmas é util sob o aspecto das facilidades e dificuldades proporcionadas por cada um na aquisição de conhecimento com vistas à construção de sistemas especialistas. Comentaremos, a seguir, os paradigmas relacionados por Carbonell, ressaltando, em cada caso, a idéia principal que norteia a sua aplicação.

Os paradigmas Indutivo e Analítico são duas formas distintas de aprendizado de conceitos. O aprendizado de conceitos, dentro da IA, "denota a aquisição de descrições estruturais a partir de exemplos sobre aquilo que está sendo descrito" [MacDonald 89]. Ambos os paradigmas procuram adquirir descrições estruturais. A partir de exemplos e de algum conhecimento já disponível (ao qual nos referiremos como conhecimento preliminar), os métodos baseados nestes paradigmas geram conceitos que descrevem o domínio em questão. Adotam, contudo, estratégias diferentes, o que os fazem úteis para aplicações distintas.

No paradigma Indutivo, busca-se a descrição de um conceito geral a partir de exemplos empíricos e conhecimento preliminar sobre esse conceito. O objetivo principal é induzir descrições gerais para as instâncias apresentadas na entrada. Nesse processo, garante-se que os conceitos produzidos não são inconsistentes com os exemplos nem com o conhecimento preliminar.

No paradigma Analítico, métodos dedutivos utilizam poucos exemplos sobre um domínio de base teórica. Neste paradigma, o conhecimento existente é utilizado para resolver problemas, servindo de guia nas deduções dos novos problemas, e para formular regras de controle de busca que tornem mais eficiente a aplicação do conhecimento no domínio. Em outras palavras, o conceito operacionaliza partes relevantes do conhecimento preliminar. O aprendizado desempenhado por esse paradigma é também denominado aprendizado dedutivo ou "baseado em explanações".

O conhecimento aprendido através dos paradigmas Indutivo e Analítico é usualmente representado através de lógica formal, composta por símbolos que possuem interpretação semântica. As definições formais para indução e dedução podem ser encontradas no capítulo 3.

O paradigma Genético é inspirado na analogia com o processo de evolução biológica das espécies e na seleção natural de Darwin. Variações das descrições dos conceitos correspondem a indivíduos de uma espécie, e combinações destes indivíduos são testadas contra uma função objetivo (critério de seleção). Os algoritmos genéticos codificam uma busca paralela através do espaço de conceitos, onde cada processo tenta maximizar a função objetivo. Neste paradigma, o ambiente é geralmente representado, através de suas entradas e saídas, por cadeias de símbolos de comprimento fixo, em um alfabeto binário (0, 1). Esta cadeia, comumente chamada de cromossomo, serve de material genético que sofre mutações resultantes da aplicação de operadores genéticos [Austin 90].

No paradigma Conexionista, inspirado no cérebro humano, o aprendizado é realizado mediante o ajuste de pesos em uma rede neuronal. Este paradigma

apresenta bons resultados em domínios com uma grande massa de dados à disposição. Os modelos conexionistas são também chamados de redes neuronais artificiais (RNA), redes neurais ou sistemas de processamento paralelo distribuídos. Há três aspectos imprescindíveis na descrição de uma rede neural: a sua topologia, as características dos nós e as regras de treinamento ou aprendizado. Além disso, são considerados elementos importantes na caracterização da rede neural a representação distribuída, as operações locais e o processamento não-linear [Simpson 90].

1.5 - Justificativa do trabalho

Segundo [Gallant 88], não há uma concordância geral do que define um sistema especialista, mas as seguintes características devem ser encontradas em um programa desenvolvido para resolver (ou ajudar a resolver) problemas num pequeno domínio em particular:

- Inferência, que possibilita traçar conclusões sem ter que consultar todas as informações possíveis.
- Aquisição interativa de dados, que direciona a aquisição para as informações mais relevantes, buscando a forma mais eficiente possível.
- Estrutura modular, com a base de conhecimento bem distinta da máquina de inferência e da interface com o usuário.
- Justificação de conclusões, que permite a verificação de sua lógica interna e pode dar a usuários novatos dicas sobre como resolver o problema.

Queremos implementar sistemas especialistas de classificação. Para a aquisição de conhecimento, podemos lançar mão de um paradigma de aprendizado automático. Como vimos, existem diversos paradigmas de aprendizado a que podemos recorrer. Cada um apresenta algumas vantagens sobre os demais em algumas tarefas específicas.

O paradigma Conexionista, por exemplo, apresenta muitas características desejadas, como flexibilidade, boa capacidade de generalização e robustez quanto aos dados de treinamento, saindo-se razoavelmente bem com ruído, buracos[1] e

contra-exemplos. Contudo, a representação de conhecimento distribuída não favorece a emissão de justificações aproveitáveis. Isto ocorre porque as células da camada intermediária não são sensíveis a um único padrão de entrada: sua função de ativação pode apresentar valores acima do limiar de aceitação a partir de diversos padrões de entrada diferentes. Essa "indeterminação" se repete em todas as células possivelmente encadeadas na camada escondida. Isto inviabiliza a identificação exata de quais dos valores atualmente presentes na entrada desempenham papel decisivo no resultado apresentado pela camada de saída.

Já o paradigma Indutivo garante a classificação de todos os exemplos do conjunto submetido ao treinamento, além de viabilizar a justificação das conclusões, já que todos os conceitos estão expressos de uma forma simbólica, inteligível aos seres humanos. Contudo, o paradigma Indutivo não garante uma boa generalização e não consegue lidar de forma satisfatória com ruídos, buracos e contra-exemplos no conjunto de treinamento.

Assim, é natural que se procure a integração cada vez maior das diversas técnicas e paradigmas de aprendizado automático. É nessa linha que Kandel [Kandel 91] apresenta a proposta dos Sistemas Inteligentes Híbridos. Esses sistemas, construídos a partir da integração de diversos paradigmas, podem se aproveitar das vantagens de cada um deles. É de se esperar, portanto, que os sistemas especialistas da próxima geração adotem essa estratégia.

Outra possibilidade para se reunir as vantagens de diversos paradigmas em um único modelo está na definição de um paradigma híbrido, "customizado" para a aplicação. Este paradigma pode apresentar algumas desvantagens que não sejam muito importantes para a aplicação em questão, com tanto que reuna as características desejadas dos diversos paradigmas para a aplicação que se tem em mente. Convém ressaltar que a busca por esses paradigmas especialmente projetados ("customizados") serve como um ótimo exercício para se identificar as decisões

---

[1] O termo "buraco" será usado para representar a ausência de dados na descrição dos casos (exemplos) do conjunto de treinamento.

responsáveis pelas vantagens e desvantagens obtidas. Além disso, permite que se indique soluções híbridas boas para determinados tipos de problemas.

Devido às características relacionadas por Gallant e também por razões históricas, o conhecimento vem sendo representado, na construção de sistemas especialistas, de uma forma simbólica, especialmente a forma de regras de produção, o que permite leitura e interpretação direta por seres humanos.

Por outro lado, sabemos que uma das principais características dos modelos conexionistas é a sua capacidade de aprendizado e, como vimos, este é o principal problema para a construção de sistemas especialistas. Assim, não é surpresa a existência de muitos trabalhos que integram sistemas especialistas e modelos conexionistas. Gallant [Gallant 88] criou o termo Sistema Especialista Conexionista para classificar esta categoria de sistemas especialistas. Exemplos de sistemas que se enquadram nesta categoria são: o SEC, com o motor de inferência MACIE [Gallant 88]; o RUBICON, usado para a implementação de sistemas especialistas baseados em regras [Samad 88]; o FUZZNET, que utiliza a teoria dos conjuntos difusos para fazer raciocínio aproximado em sistemas especialistas baseados em regras [Romaniuk 89]; o SDPDP, que é um sistema para diagnóstico médico baseado no processamento paralelo distribuído [Saito 88]; e o Modelo Neural Combinatório (**MNC**), um modelo eminentemente conexionista voltado para tarefas de classificação [Machado 89].

O MNC é um modelo híbrido que representa uma ponte importante entre os campos das redes neurais subsimbólicas e o de sistemas especialistas, permitindo a construção de sistemas especialistas conexionistas [Gallant 88] capazes de herdar propriedades desejáveis de ambos os campos. Contudo, este Modelo apresenta alguns problemas. Entre eles, podemos citar o problema da explosão combinatorial e o problema semântico, intrínseco a todo algoritmo de aprendizado baseado apenas em aspectos sintáticos do conjunto de treinamento.

O objetivo deste trabalho é analisar o MNC sob a ótica desses dois problemas e propor o uso de conhecimento preliminar como forma de minimizá-los.

1.6 - Estrutura da dissertação

Para atingir o objetivo proposto, estruturamos o presente trabalho da seguinte forma: o capítulo 2 apresenta o Modelo Neural Combinatório, abrangendo desde alguns antecedentes até a direção atual das pesquisas sobre o mesmo. O capítulo 3 discorre sobre a importância do conhecimento preliminar para o aprendizado, mostrando alguns exemplos. O capítulo 4 mostra a aplicação da relevância semântica ao MNC e o capítulo 5 introduz a valoração de atributos no Modelo. As alterações introduzidas nos capítulos anteriores fazem surgir vários parâmetros importantes para o desempenho geral do aprendizado no Modelo. O capítulo 6 mostra um estudo desses parâmetros e a forma como interferem no resultado. O capítulo 7 mostra como é possível implementar-se as alterações propostas de forma a usufruir melhor de suas vantagens. O capítulo 8 contém a conclusão e as sugestões para os trabalhos futuros. Finalmente, os apêndices apresentam detalhes sobre os algoritmos e módulos do protótipo, dados de experimentos e resultados preliminares.

2 - O Modelo Neural Combinatório

2.1 - Motivação e Antecedentes

O Modelo Neural Combinatório (**MNC**) "é uma poderosa rede neural de ordem superior adequada para tarefas de classificação" [Machado 89]. Este modelo inspira-se nas ciências neurais, na lógica difusa e em pesquisas sobre a análise do conhecimento de especialistas humanos [Machado 90b].

Das ciências neurais, a neurofisiologia é tida como a fonte primária para os modelos conexionistas, que tentam emular em um maior ou menor grau as propriedades do sistema nervoso. O objetivo dessa emulação é obter a mesma efetividade obtida pelos modelos biológicos na solução de uma grande variedade de problemas difíceis, usando, para tanto, dispositivos projetados segundo os mesmos princípios dos sistemas naturais. A psicologia forneceu idéias interessantes adotadas no modelo, tais como o paradigma de aprendizado baseado no condicionamento operante, e a limitação de ordem inspirada nas limitações cognitivas apresentadas pelos seres humanos.

A lógica difusa é derivada da teoria dos conjuntos difusos, que foi introduzida por Zadeh em 1965 [Zadeh 65], como um modo adequado para formalizar os processos associados ao raciocínio humano, que é caracterizado pela incerteza, incompletude, imprecisão e inconsistência. Fenômenos do sistema nervoso, como excitabilidade celular, conectividade celular e distribuição e processamento de mensagens podem ser adequadamente descritos pela teoria dos conjuntos difusos.

Contudo, ainda segundo [Machado 89], a principal fonte de inspiração para a definição do modelo foi a engenharia do conhecimento. A análise do conhecimento heurístico, eliciado de especialistas de diferentes áreas de perícia humana para a construção de bases de conhecimento de sistemas especialistas [Leão 87], e os resultados obtidos da aplicação do método desenvolvido por Rocha e outros para

aquisição e análise do conhecimento especialista [Leão 88] [Machado 88], foram muito importantes para a definição do modelo.

2.1.1 - Os Grafos de Conhecimento

Nessas pesquisas de Rocha, Machado e Leão, sobre engenharia do conhecimento, chegou-se a uma ferramenta de eliciação e representação de conhecimento simples e flexível, o grafo de conhecimento. Vários estudos de campo foram desenvolvidos com esta ferramenta, onde especialistas humanos em diversas áreas expressaram seu conhecimento heurístico em um dado domínio de problema através do seu uso.

O grafo de conhecimento (figura 2.1) é um grafo E/OU acíclico, com três camadas: a camada de entrada, onde os nós representam evidências sobre o domínio; a camada de saída, formada por nós OU que representam as classes ou hipóteses do domínio; e a camada intermediária, que fica entre as duas anteriores e pode ter um ou mais níveis. A camada intermediária é formada por nós E, que agregam logicamente informações, rotulados com um grau de correlação à hipótese correspondente. Esses nós associam diferentes combinações ou padrões de entrada, representando abstrações que podem ser estruturadas de forma hierárquica em diversos níveis. As abstrações correspondentes aos nós E podem representar conceitos que possuem nomes no mundo real ou existir apenas nas estruturas mentais do especialista. Os aspectos importantes de um grafo de conhecimento são [Leão 88]:

a) sua topologia, identificada pela disposição dos nós nas diversas camadas e pelas ligações entre os mesmos, sendo que os nós da camada inferior representam as evidências importantes para a classe representada pelo nó da camada superior;
b) o grau de importância de cada evidência no aglomerado, de acordo com a ordem de disposição da mesma na camada de entrada; e
c) o peso de cada arco.

Os grafos de conhecimento foram usados até no estudo da integração de conhecimento eliciado de especialistas diferentes. Machado descreve um algoritmo

para gerar o grafo de conhecimento médio, a partir de diversos grafos de conhecimento diferentes [Machado 88]. Foram identificados parâmetros que permitem distinguir entre grafos de conhecimento eliciados de especialistas e grafos eliciados de não especialistas [Machado 90].

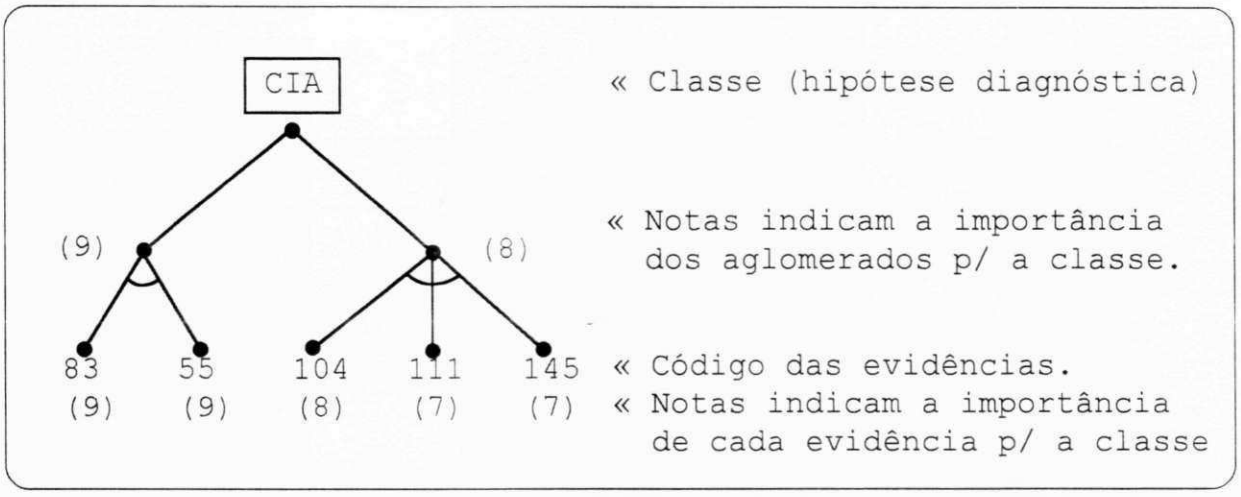


Figura 2.1 – Grafo de Conhecimento sobre Comunicação Interatrial (CIA), retirado de [Leão 88].

Como as pesquisas em engenharia do conhecimento obtiveram muito sucesso com o uso dos grafos de conhecimento, estes acabaram se tornando uma das principais fontes de inspiração para a definição do Modelo Neural Combinatório. Os grafos de conhecimento demonstraram ser um meio de representação de conhecimento bastante eficaz para tarefas de classificação, proporcionando facilidade para eliciação de conhecimento de especialistas, boa legibilidade e integração de conhecimento eliciado de fontes diferentes.

O grafo de conhecimento é uma forma de representação de conhecimento declarativa e inteligível para seres humanos. Esta é uma qualidade desejável para que um formalismo de representação de conhecimento seja usado na confecção de sistemas especialistas. Outra necessidade é que este formalismo seja manipulável com facilidade também pela máquina, para que ela possa realizar inferências, apresentar justificações sobre as conclusões e também, idealmente, realizar aprendizado automático. Esta última facilidade visa a construir sistemas novos ou refinar conhecimento já existente em um sistema, de forma a melhorar seu desempenho. O grafo de conhecimento contempla este requisito de maneiras diferentes, a depender

do enfoque adotado, que pode buscar suporte na teoria dos grafos ou em técnicas conexionistas, como veremos.

A manipulação automática do grafo de conhecimento pode aproveitar-se da solidez e estabilidade da teoria dos grafos. Como se trata de um grafo acíclico, os algoritmos de caminhamento não encontram dificuldades especiais. A classificação de casos reais (inferência) pode ocorrer através da comparação das características do caso real com os nós da camada inferior do grafo. A partir daí, o significado dos nós E e OU, das camadas intermediária e de saída, indicam a classe a que mais provavelmente deve pertencer o caso.Vemos que, a partir do momento em que as características do caso são apresentadas à camada de entrada, o significado e a interligação dos nós superiores é que vai determinar a classe do caso.

Sabemos que os significados dos nós E e OU podem ser plenamente determinados por funções lógicas, pois cada nó assume um estado de ativação dependendo unicamente de suas entradas. A possibilidade de processamento localizado, definido por funções de variáveis disponíveis localmente, abre as portas para um novo enfoque. Cada nó pode conter "inteligência" suficiente para calcular seu valor de ativação e propagá-lo para os nós seguintes. É assim que ocorre a inferência numa rede neural. Este enfoque permite o aproveitamento de várias técnicas conexionistas usadas para inferência e aprendizado sobre redes neurais.

Encarar o grafo de conhecimento como uma rede neural em que os neurônios computam funções diferentes, dependendo da camada em que se encontram – esta é a base do Modelo Neural Combinatório. A rede obtida a partir do grafo de conhecimento tem uma topologia bem definida, onde cada nó implementa localmente uma função bem definida, de comportamento não linear. Assim, ela apresenta as características comumente encontradas nas redes neurais, com uma importante exceção, que veremos a seguir.

A representação de conhecimento nas redes neurais é comumente distribuída, o que significa que não se pode identificar as regiões da rede que guardam o conhecimento sobre um determinado conceito. Assim, essas redes não podem

oferecer nenhum tipo de explanação ou justificação sobre os resultados obtidos. Isto só pode ser conseguido naquelas redes em que cada unidade da camada escondida é sensível a um único padrão de entrada. Este é o caso da rede definida no Modelo Neural Combinatório. Nela, a representação de conhecimento é localizada, e não distribuída. Cada nó da camada intermediária é ativado por uma determinada combinação na camada de entrada. Essa combinação representa um único conceito. Essa característica é fundamental para a confecção de SEs conexionistas, pois torna a Base de Conhecimento inteligível para o ser humano, o que possibilita a sua avaliação por especialistas humanos e a emissão de explanação pelo sistema para justificar as respostas geradas pelo processo de inferência.

2.2 - Descrição do Modelo

Depois de conhecermos as fontes de inspiração do Modelo Neural Combinatório, descrevê-lo tornou-se uma tarefa bem mais simples. O MNC é uma rede neural modificada, com topologia bastante semelhante àquela dos grafos de conhecimento, onde os neurônios (unidades de processamento) são capazes de processar as funções lógicas correspondentes àquelas dos nós encontrados nos grafos de conhecimento. O MNC representa uma ponte importante entre os campos das redes neurais subsimbólicas e o de sistemas especialistas, permitindo a construção de sistemas especialistas conexionistas capazes de herdarem propriedades desejáveis de ambos os campos.

Uma descrição com enfoque mais conexionista ressaltaria que o modelo é formado por um conjunto de unidades de processamento paralelas – os "neurônios" –, dispostos em três camadas – entrada, intermediária ou escondida e saída –, que processam diferentes funções não lineares, dependendo da camada em que se encontram, e são interconectados de forma a que cada neurônio da camada intermediária seja ativado por uma determinada família de padrões de ativação dos neurônios da camada de entrada.

O objetivo do Modelo Neural Combinatório é computar o grau de possibilidade de cada hipótese de um problema de classificação e apresentar aquelas que ultrapassam o **limiar de aceitação** previamente definido como sendo a solução do problema. Isso é feito através da propagação pela rede das evidências disponíveis no problema. O MNC pode expressar sua indecisão declarando que um objeto é similar a várias classes, como as pessoas fazem frequentemente em situações ambíguas. A figura 2.2 mostra um exemplo de uma rede neural combinatória difusa de acordo com o MNC.

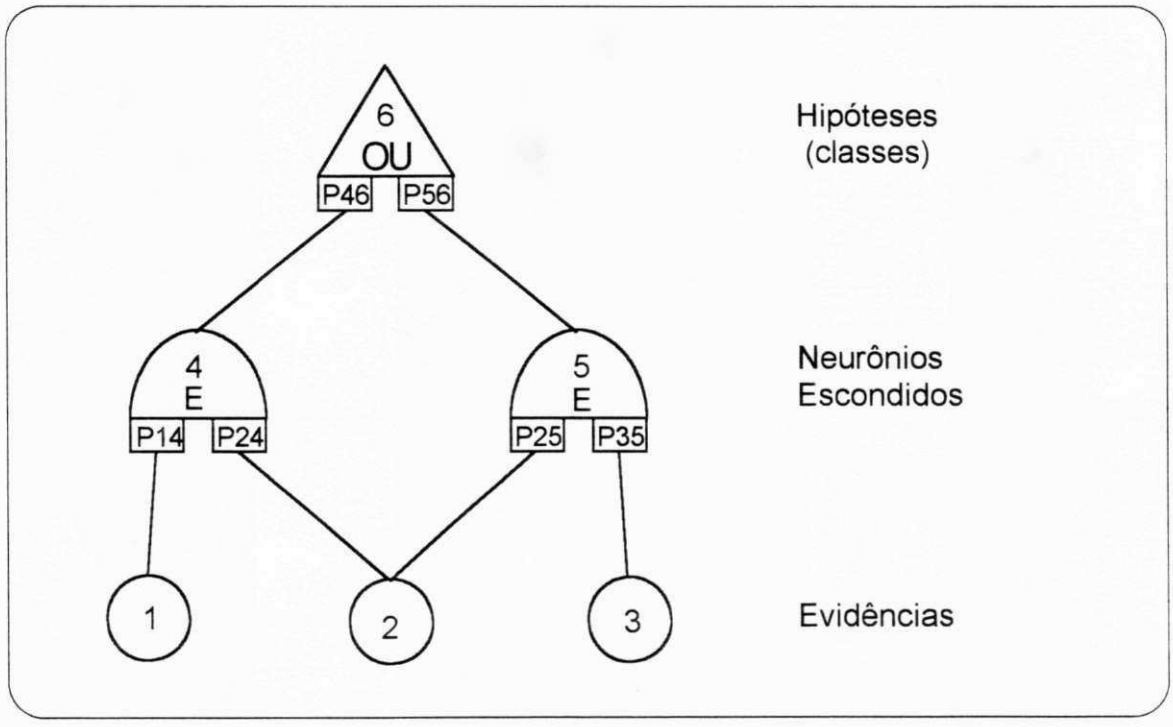


Figura 2.2 – Rede Neural Difusa baseada no Modelo Neural Combinatório.

No MNC, cada célula representa um conceito simbólico do domínio do problema, como evidências, hipóteses, estados do problema, etc. Cada célula calcula um valor numérico pertencente ao intervalo [0,1], chamado de **estado de ativação**, baseado na sua informação de entrada. O estado de ativação, que também representa o valor de saída da célula, pode ser interpretado como o grau de crença ou aceitação que o sistema coloca no conceito representado pela célula específica. Quando a célula trabalha no modo booleano, seu estado de ativação só pode assumir os valores 0 e 1; já no modo difuso, pode assumir qualquer valor no intervalo [0,1].

As unidades de processamento são conectadas através de arcos ("sinapses"), formando "redes neurais". Essas sinapses podem ser classificadas como **excitatórias** ou **inibitórias** e são caracterizadas por um valor que expressa sua força (peso), variando entre 0 (não conectada) e 1 (sinapse plenamente conectada). As sinapses excitatórias atuam diretamente sobre o sinal de entrada. As sinapses inibitórias atuam sobre a negação difusa do sinal de entrada. Em ambos os tipos de sinapses, o peso é usado como um fator de atenuação aplicado ao sinal de entrada (ou a sua negação). O sinal transmitido através da sinapse age como uma mensagem que é transmitida de um neurônio para outro. Essa mensagem é chamada de **Fluxo Evidencial** do arco ou sinapse, e é calculada da seguinte forma:

$$\text{Fluxo Evidencial na Sinapse} = w \cdot E \cdot t + w \cdot (1-E) \cdot (1-t)$$

onde:

- E é o sinal de entrada na sinapse;
- w é o peso da sinapse;
- t é o tipo da sinapse (0 se inibitória; 1 se excitatória).

Apesar da sinapse poder ser excitatória ou inibitória, usualmente apenas as excitatórias são usadas. Isto advém das experiências práticas com os grafos de conhecimento.

Voltando aos aspectos importantes para a descrição de uma rede neural, verificamos que falta apenas a definição de um algoritmo de treinamento para a rede, que permita refinar o conhecimento de uma rede ou realizar aprendizado em uma rede não treinada. O algoritmo de treinamento definido para o MNC é baseado no *backpropagation* (retropropagação) usado nos Perceptrons Multicamadas [Rumelhart 86].

Veremos, nas seções seguintes, detalhes sobre vários aspectos do MNC.

2.3 - As Unidades de Processamento

Como pode ser visto na figura 2.2, existem basicamente três tipos de neurônios, determinados pela camada que ocupam na rede. Esses tipos de neurônios são caracterizados por diferentes funções de agregação: *OU-difuso*, na camada de saída; *E-difuso*, na camada intermediária; e *Número-difuso*, na camada de entrada. O modelo de neurônio difuso aqui empregado é baseado na lógica difusa de Zadeh [Zadeh 65].

Os neurônios *OU-difuso* e *E-difuso*, mostrados na figura 2.3, implementam as funções difusas E e OU, definidas por Zadeh. Na figura, temos:

- Xj, j = 1, 2, ..., n representa as entradas da célula. A entrada para uma célula pode ser uma informação proveniente do meio ou a saída de uma outra célula.
- y é o estado de ativação calculado pela célula. Expressa o grau de possibilidade do conceito representado pelo neurônio e pode assumir valores entre 0 e 1.
- Pj representa a força (peso) da sinapse da entrada *j* para a célula. Os pesos podem ser interpretados como grau de pertinência do conceito representado na entrada para o conceito representado na saída da célula.

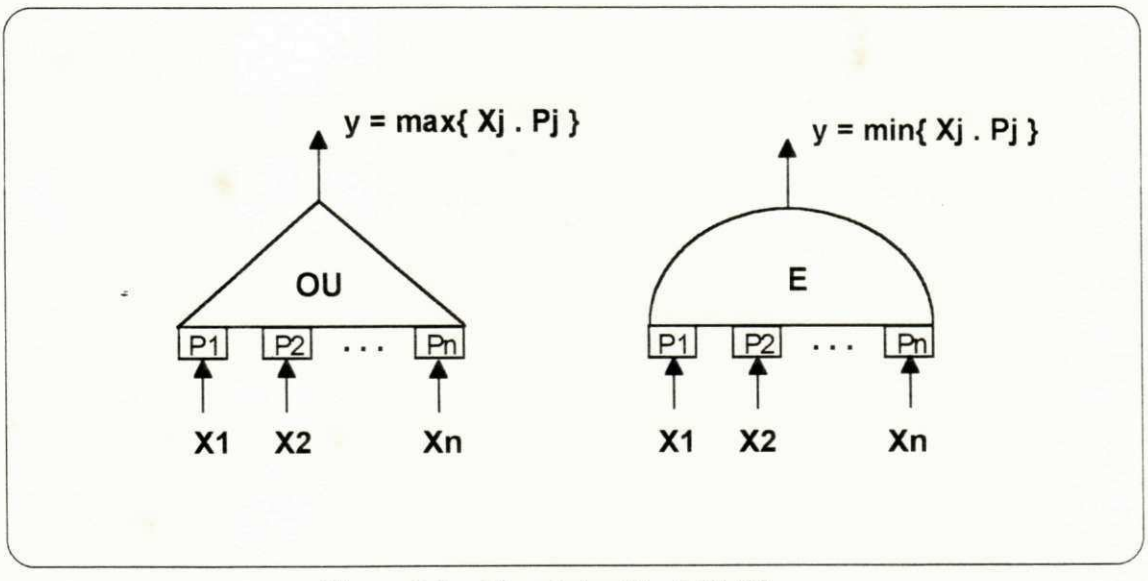


Figura 2.3 – Neurônios E e OU difusos

Os neurônios do tipo *Número-difuso* implementam funções difusas que mapeam características do ambiente em valores no intervalo [0, 1], de acordo com a aderência ou similaridade da característica do ambiente inserida pela camada de entrada com o conceito linguístico representado pelo neurônio.

Assim, esses neurônios não implementam uma mesma função difusa, como os anteriores. Cada neurônio *Número-difuso* implementa uma função diferente, de acordo com o conceito linguístico a ele associado. Na teoria de Zadeh, essa função de pertinência deve ser aplicada a um conjunto universo para definir um conjunto difuso, contido no primeiro, que representa um conceito linguístico.

Para um melhor entendimento sobre esse tipo de neurônio, mostramos na figura 2.4 um exemplo de neurônio *Número-difuso* para o conceito "alto". Nessa figura, podemos ver o neurônio e a função que ele implementa.

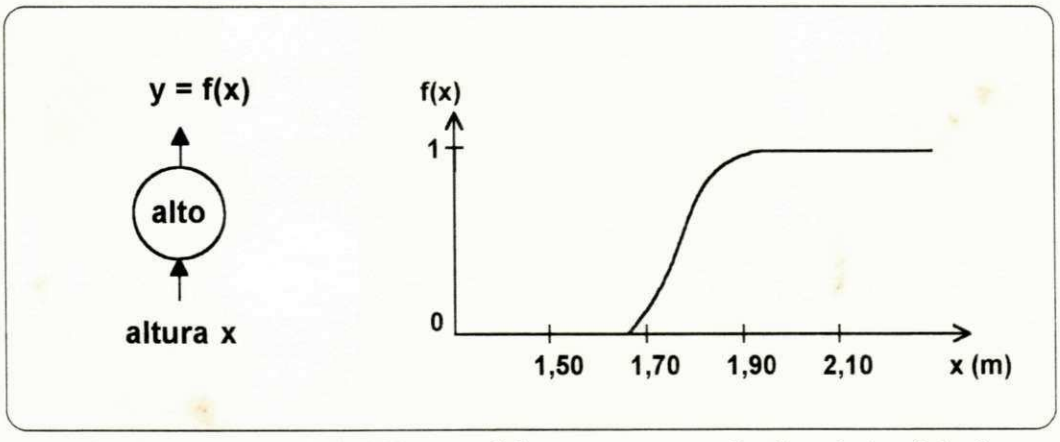


Figura 2.4 – Neurônio *Número-difuso* para o conceito linguístico "alto".

Os atributos do caso aplicado à entrada da rede podem ser de dois tipos: nominais e ordinais. Os atributos nominais assumem valores simbólicos de um conjunto não-ordenado. Os atributos ordinais assumem valores contínuos dentro de um intervalo. No primeiro caso, cada neurônio *Número-difuso* indica apenas se o atributo do ambiente assumiu um determinado valor entre um conjunto de valores simbólicos possíveis. Nesse caso, necessita-se de um neurônio na camada de entrada para cada valor que o atributo possa assumir.

Os conceitos linguísticos relacionados aos atributos nominais são, em geral, aqueles representados pelos próprios valores simbólicos que estes podem assumir. Já para os atributos ordinais, a quantidade de conceitos linguísticos deriváveis é infinita e nenhum deles está explícito na definição do atributo. Assim, deve-se estabelecer, entre os conceito linguísticos deriváveis do atributo, quais são aqueles efetivamente importantes para a aplicação. Cada um desses conceitos escolhidos fica representado no Modelo por um neurônio *Número-difuso*.

2.4 - A Topologia da Rede

Como já vimos, a topologia da rede do Modelo Neural Combinatório é Inspirada na topologia dos grafos de conhecimento; mas apresenta algumas diferenças. Em primeiro lugar, a ordem das evidências na camada de entrada, que é importante no grafo de conhecimento, não tem nenhuma importânica no MNC.

A segunda diferença diz respeito à camada intermediária. A camada intermediária dos grafos de conhecimento eliciados de seres humanos podem ter vários níveis. Isto permite a representação da hierarquia existente entre os conceitos. Algumas evidências podem ser agrupadas em um nó E que por sua vez pode estar agrupado a outros formando um conceito mais complexo, representado por um nó E de nível superior, que está finalmente ligado a um nó OU da camada de saída. Esta hierarquia não é representada no MNC porque sua camada intermediária apresenta apenas um nível. Esta restrição não diminui a generalidade do modelo, pois sempre é possível produzir uma rede com apenas um nível na camada combinatória que seja equivalente a outra com um número maior de níveis. Esta propriedade advém do uso do operador *min* para modelagem das células E [Machado 89]. Por ora, admitamos que a existência desta restrição simplifica o algoritmo de inferência e é imprescindível para a introdução do aprendizado automático no modelo, que é baseado no treinamento de uma rede pré-definida, como veremos adiante.

A figura 2.5 mostra uma rede combinatória simples, composta pelas três camadas já conhecidas, as quais chamaremos doravante Camada de Entrada, Camada Combinatória e Camada de Saída.

A rede é altamente modular. Cada classe do problema de classificação tem uma rede independente, chamada *sub-rede da hipótese*, como pode ser visto na figura 2.5. Em cada rede de três camadas, cada neurônio *E-difuso* pode ser visto como um módulo individual independente, chamado caminho ou aglomerado, que compete com os outros aglomerados para estabelecer a classe do caso. Como cada aglomerado pode ser visto como uma regra heurística difusa, é bem simples prover uma explanação sobre a resposta dada pelo sistema, apenas mostrando a cadeia de regras derivadas dos neurônios responsáveis pela ativação da hipótese vencedora.

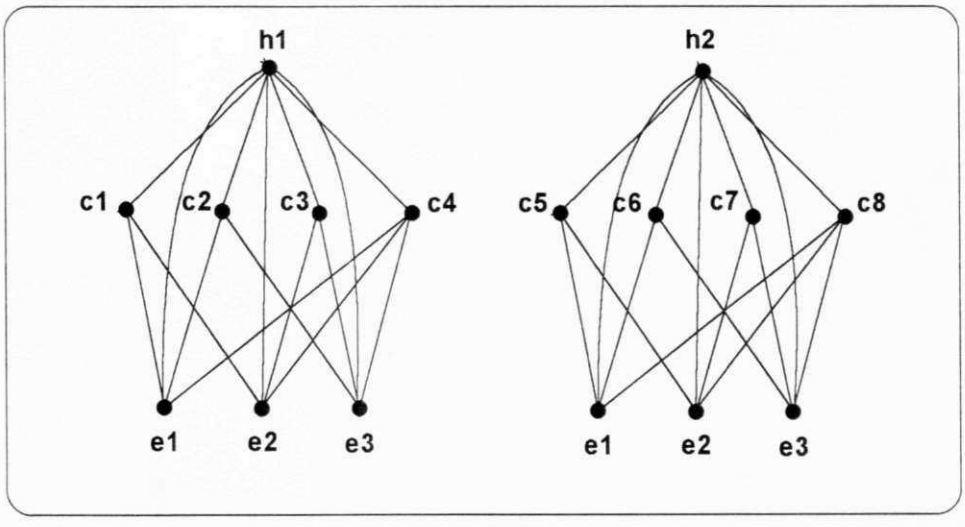


Figura 2.5 – Versão completa da rede neural combinatória para três evidências de entrada e duas hipóteses. Por questão de legibilidade da figura, as evidências de entrada e1, e2 e e3 estão repetidas e os neurônios estão representados por pontos. Isto pode ser feito porque não existe ambiguidade sobre o tipo de cada neurônio: os da camada de saída, rotulados com h1 e h2, são do tipo *OU-difuso*; os da camada combinatória, rotulados com c1, ..., c8, são do tipo *E-difuso*; e aqueles da camada de entrada, rotulados com e1, e2 e e3, são neurônios *Número-difuso*.

2.4.1 - A Camada de Entrada

Cada neurônio da primeira camada representa um conceito de entrada sob a forma de uma tripla: objeto-atributo-valor, como por exemplo: "a temperatura de João é alta". Nesta situação, cada elemento do conjunto F_k de todos os possíveis

conceitos linguísticos associados a um atributo *k* é representado por um neurônio diferente. O grau de confiança em cada uma das proposições corresponde ao estado de ativação dos neurônios de entrada.

Dados simbólicos e numéricos, provenientes do meio ambiente ou do usuário, são introduzidos na camada de entrada da rede diretamente sob a forma de um grau de possibilidade para uma proposição ou na forma de um número, representando o resultado de uma medição. No primeiro caso, o neurônio da camada de entrada apenas repassa o grau de possibilidade presente em sua entrada, implementando uma função identidade. No segundo caso, o neurônio *Número-difuso* converte o dado de entrada para um grau de possibilidade empregando uma função de pertinência especificada pelo especialista no domínio do problema. Em qualquer dos dois casos, a informação propagada para a camada superior reflete o grau de adesão do caso real com o conceito linguístico representado pelo neurônio.

Esta abordagem apresenta as seguintes propriedades interessantes:

- Representação da situação de ignorância (evidência desconhecida ou não disponível). Para tanto, é suficiente fazer igual a zero todas as entradas correpondentes aos achados do conjunto F_k do atributo k.
- Codificação de atributos complexos envolvendo múltiplos achados, com a capacidade de simultaneamente atribuir um grau de crença a cada um desses achados. Ex: Considere o atributo Cor da Pele = (branco, negro, amarelo). Três células permitem representar esses achados, bem como todos os casos de mistura de raças.

2.4.2 - A Camada Combinatória

Uma característica fundamental no raciocínio de especialistas humanos é associar evidências da entrada, compondo aglomerados de informação. Isso é feito usando o conectivo lógico *E*. Com o objetivo de representar essas associações, foi incluído no modelo da rede uma camada oculta, que deve conter as diferentes combinações dos dados de entrada. Essa camada oculta foi batizada de Camada Combinatória.

Na versão completa do modelo, a Camada Combinatória inclui todas as combinações das evidências de entrada, desde $C_{n,1}$ até $C_{n,n}$. A presença de todas essas combinações, caminhos ou aglomerados é importante por causa do aprendizado, que é realizado no modelo através do treinamento de uma rede previamente definida, como veremos na próxima seção.

2.5 - O Aprendizado no Modelo

O aprendizado no MNC é realizado em duas fases distintas: o treinamento e a poda da rede (figura 2.6). A primeira fase consiste no treinamento de uma rede inicial a partir de um conjunto de exemplos. Essa fase gera uma rede intermediária, com acumuladores e dados de controle. A segunda fase consiste na poda e normalização dessa rede intermediária. Essa fase é baseada em um limiar de poda e em algumas técnicas para identificar e preservar aglomerados importantes. A rede gerada por essa fase é considerada o resultado do aprendizado, podendo ser usada pelo algoritmo de inferência para classificar casos reais.

O algoritmo de treinamento possui duas versões: a versão de arranque treina uma REDE ACÍCLICA "nova", gerada por ele mesmo; a versão incremental treina uma REDE ACÍCLICA que já foi anteriormente treinada, preferencialmente através de um outro conjunto de treinamento.

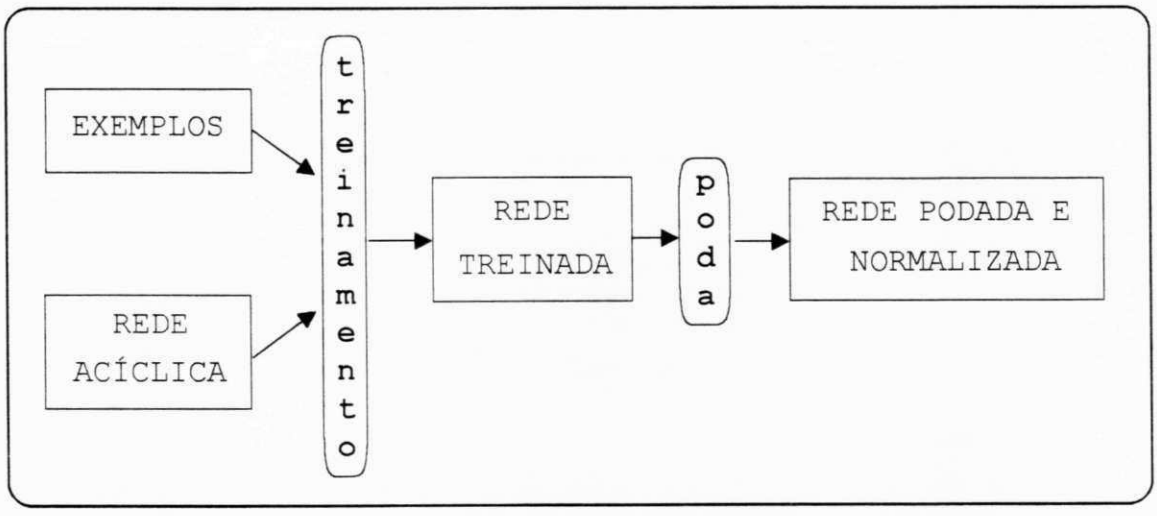


Figura 2.6 – Fases do aprendizado no Modelo Neural Combinatório.

O algoritmo de treinamento é baseado no algoritmo de retropropagação *(backpropagation)*, usado nos Perceptrons Multicamadas [Rumelhart 86]. Ele ajusta os pesos da rede acíclica através de um mecanismo de punição das "asserções" (caminhos ou aglomerados da rede) contraditas pelos exemplos e recompensa daquelas reforçadas pelos mesmos. Isso lhe permite identificar, dentre os aglomerados da rede, aqueles que estão de acordo com os exemplos do conjunto de treinamento. O algoritmo é capaz de trabalhar com dados de treinamento booleanos [Machado 89] ou difusos [Denis 91].

O algoritmo de treinamento não cria novas vias de raciocínio. Ele apenas ajusta os pesos das vias existentes na Camada Combinatória da topologia que lhe é apresentada. É por isso que a topologia inicial de trabalho desempenha um papel tão importante para o aprendizado no modelo. Fazendo uma analogia com o paradigma simbolista, a topologia inicial funciona como um conjunto com todas as possíveis asserções que se possa fazer sobre o domínio.

Percebe-se então que a escolha da topologia de trabalho assume importância crucial para a eficiência computacional do algoritmo e para a qualidade do aprendizado desempenhado por ele. Quanto maior a topologia, maior o esforço computacional gasto pelo algoritmo; quanto menor, mais pobre o leque de conceitos a treinar e aprender.

2.5.1 - A Topologia Usada no Modelo

Como vimos, o aprendizado no MNC é realizado através do treinamento de uma topologia pré-concebida. A topologia "mais segura" para esse aprendizado seria aquela que contivesse todas as combinações possíveis das evidências do domínio. Chamemos essa topologia "ideal" de Topologia Combinatorial Completa (TCC)[2]. Nessa topologia, nenhuma combinação (ou asserção), por mais esdrúxula que fosse, escaparia ao crivo do algoritmo de aprendizado. Uma topologia com essa

[2] Para facilitar a referência às topologias, resolvemos batizá-las, desde as topologias originalmente presentes no Modelo até aquelas que nós definimos neste trabalho. Os nomes de batismo foram escolhidos com a preocupação de identificarem bem as topologias, ressaltando as diferenças entre as mesmas.

característica teria, num dominio com n evidências, um total de

$$NC(n,n) = \Sigma_{i=1}^{n} C_{n,i} = 2^n - 1$$

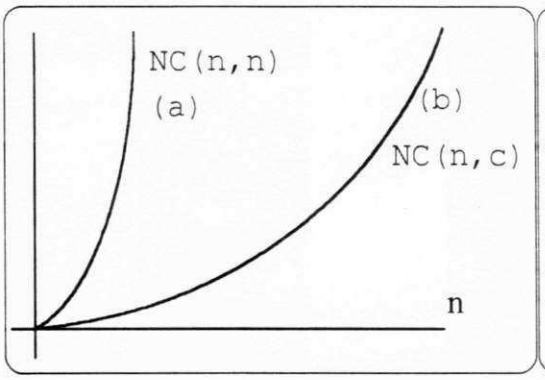
combinações, o que seria infactível para problemas reais, porque o tamanho da topologia cresceria exponencialmente com o aumento do número de evidências.

A alternativa adotada para diminuir este problema foi limitar o tamanho das combinações da topologia. Esta limitação é implementada na definição da Topologia Combinatorial de Ordem Superior (TCOS), que comporta apenas as combinações com tamanho máximo igual à ordem da topologia. A TCOS é a topologia originalmente usada no MNC. Ela possui

$$NC(n,m) = \Sigma_{i=1}^{m} C_{n,i}$$

células na camada combinatória. Em nome da simplicidade e uniformidade, consideramos que as combinações de evidências tomadas uma a uma ($C_{n,1}$) também são representadas por células na camada intermediária, apesar de aparecerem como caminhos diretos entre a camada de entrada e a camada de saída, na figura 2.5.

Devido ao expurgo de muitos aglomerados, a TCOS possui sempre tamanho bem menor que a TCC, sendo que essa diferença aumenta à medida que n aumenta. Um estudo numérico mostra que o tamanho da topologia continua com crescimento exponencial em função da quantidade de evidências do domínio. Isto significa que a limitação imposta pela ordem apenas possibilita o uso de valores um pouco maiores para n (figura 2.7); de forma nenhuma resolve definitivamente o problema de qual topologia usar para o aprendizado.



n	NC(n,5)	NC(n,7)	NC(n,9)
5	3,10e+01	3,10e+01	3,10e+01
10	6,37e+02	9,67e+02	1,02e+03
20	2,17e+04	1,38e+05	4,32e+05
30	1,74e+05	2,80e+06	2,30e+07
40	7,60e+05	2,32e+07	3,74e+08
50	2,37e+06	1,18e+08	3,16e+09
100	7,94e+07	1,73e+10	2,11e+12
150	6,12e+08	3,09e+11	8,85e+13
200	2,60e+09	2,37e+12	1,23e+15

Figura 2.7 – As curvas (a) e (b) mostram que a limitação do tamanho das combinações, através da introdução do conceito de ordem, apenas retarda o crescimento da topologia. A tabela mostra o tamanho da TCOS de ordem 5, 7 e 9, para diversos números de evidências. Nela, podemos ver que a TCOS assume tamanho muito grande nos casos reais ($n \geq 40$). Para efeito de comparação, lembramos que o cérebro humano possui algo em torno de 10^{10} neurônios.

De agora em diante, passaremos a encarar a expressão que representa o número de caminhos ou aglomerados da Camada Combinatória como o tamanho da rede e, por conseguinte, da topologia. Ela servirá de base para a estimativa do esforço computacional requerido por todos os algoritmos que manipulam a rede, especialmente o algoritmo de aprendizado.

2.5.2 - Escolhendo a Ordem da Topologia

A TCOS introduz um novo parâmetro na definição da topologia: a sua ordem *m*. Mas como se deve agir para determinar o valor da ordem em cada caso? Para tentar responder a essa pergunta, podemos nos reportar a alguns estudos que discutem os valores que *'m'* deve assumir. Norusis e Jacques [Norusis 75] trabalharam em um ambiente de classificação bayesiano bastante poderoso e flexível. A partir desse trabalho, recomendam que os aglomerados formados tenham ordem igual à metade ou mais do número de atributos. Isto para permitir um bom desempenho discriminatório, especialmente quando as classes diagnósticas não são facilmente diferenciáveis. Uma rápida análise da tabela da figura 2.7 mostra que essa recomendação não é viável em aplicações reais.

Outra recomendação, bem mais praticável, pode ser encontrada em [Machado 89]. Apoiados nos argumentos elencados a seguir, os autores recomendam o uso da ordem entre 5 e 9 (números mágicos de Miller). Os argumentos usados baseam-se:

a) no número mágico de Miller (7 ± 2) [Miller 56], que denota o limite de entidades mentais que os seres humanos podem processar ao mesmo tempo;
b) no trabalho de Leão, onde os grafos de conhecimento eliciados de uma população de cardiologistas apresentam aglomerados com uma média de 2,61 elementos e desvio-padrão de 0,068, não ultrapassando o tamanho máximo de 8 [Leão 88]; e
c) no tamanho do antecedente de regras de sistemas especialistas (por exemplo: em um sistema especialista de previsão meteorológica, média de 3,94 e desvio-padrão de 0,038 [Duarte 88]).

Um outro estudo bastante substanciado sobre esta matéria pode ser encontrado em [Machado 92]. Ele propõe um critério baseado na entropia do domínio[3] para selecionar a ordem mais adequada à topologia de um problema específico. Segundo o estudo, a rede neural difusa usada para treinamento deve possuir entropia maior ou igual à entropia do domínio. Alguns estudos de caso mostram que, à medida que se aumenta a ordem da rede a ser treinada, consegue-se uma melhoria no grau de acerto da inferência realizada sobre a rede gerada. Isto ocorre com muita intensidade enquanto a entropia da rede inicial cresce em direção à entropia do domínio do problema.

Os valores de ordem sugeridos por essa abordagem não são muito animadores. Tomemos como exemplo um domínio com 150 evidências, cada uma com probabilidade de ocorrência de 0,10. A entropia deste domínio seria:

$$h(W) = -\sum_{i=1}^{n} p(w_i) \cdot \log p(w_i) = -\sum_{i=1}^{150} (0.1) \cdot \log(0.1) =$$

$$= -150 \cdot 0.1 \cdot \log(0.1) \approx 49.8$$

---

[3] Usaremos a expressão "entropia do domínio" para nos referirmos à entropia do mundo a ser modelo e a representaremos por h(W).

Segundo [Machado 92], a entropia máxima de uma rede é obtida quando todos os caminhos r_i possuem a mesma probabilidade igual a 1/n. Isto nos leva à fórmula:

$$\max(h(rede)) = -\sum_{i=1}^{n} p(r_i) \cdot \log p(r_i) = -\sum_{i=1}^{n} \frac{1}{n} \cdot \log \frac{1}{n} = -\log \frac{1}{n} = \log n$$

Para descobrir o tamanho da rede cuja entropia seja aproximadamente 49.8, aplicamos este valor à fórmula acima. Aí temos:

$$49.8 = \log n \quad \Rightarrow \quad n = 2^{49.8} \approx 9.8 \cdot 10^{14}$$

Consultando a tabela da figura 2.7, vemos que uma rede deste tamanho, com 150 evidências, teria ordem entre 9 e 10. Como o cérebro humano possui algo em torno de $3*10^{10}$ neurônios [Machado 89], a rede sugerida para este domínio teria um tamanho maior que o cérebro humano em 4 ordens de grandeza.

Apesar de desanimadores neste caso, os números obtidos por este método não devem ser motivo para a negação do Modelo. Além desse problema hipotético ser de dimensão razoavelmente grande, testes feitos em alguns outros domínios encontraram resultados bastante animadores com redes iniciais bem menores.

Como acabamos de ver pelos estudos aqui apresentados, o problema da escolha da arquitetura inicial para a rede a ser treinada ainda não foi resolvido. A rede a ser treinada tem um tamanho exageradamente grande. A definição da TCOS, baseada no conceito de ordem da rede, apenas ameniza este problema. Além disso, a escolha da ordem passa a ser uma tarefa delicada. Os estudos que procuram estabelecer a ordem ideal possuem alguns ou todos os seguintes inconvenientes:

- baseados em princípios *ad-hoc*;
- não indicam uma ordem com precisão;
- indicam valores de ordem que resultam em redes demasiadamente grandes;
- não garantem a qualidade do aprendizado resultante do uso daquela ordem sugerida.

O método apresentado em [Machado 92] indica o tamanho e, por conseguinte, a ordem tal que garante um aprendizado útil. A partir de estudos de casos reais, a ordem sugerida parece ser maior que o realmente necessária. Portanto, devemos

procurar por métodos que indiquem ordens menores com alguma "garantia de qualidade" e também por métodos que definam redes menores que ainda apresentem aglomerados do tamanho da ordem sugerida. Este último poderia ser feito através da exclusão de aglomerados de forma a não diminuir a "qualidade" do aprendizado. Como veremos adiante, esta é a estratégia adotada neste trabalho.

2.6 - A Inferência no Modelo

A rede gerada durante o aprendizado contém conhecimento acerca do domínio. Veremos agora como esse conhecimento pode ser usado. O principal interesse do aprendizado automático é gerar uma base de conhecimento que possa ser manipulável por uma máquina para resolver problemas do domínio. No MNC, em especial, esses problemas resolvíveis pelo algoritmo de inferência são de classificação. Contudo, o conhecimento armazenado numa rede do MNC pode ser encarado de três formas diferentes:

1. Um banco de conhecimento simbólico, expresso em regras;
2. Uma rede neural a ser manipulada por um algoritmo de inferência;
3. Uma rede neural que possui o conhecimento e implementa a inferência através do processamento local de suas células.

Devido às características da topologia – principalmente a representação de conhecimento localizada –, a rede pode ser transformada em um conjunto de regras que pode ser acoplado a um *shell* que, por sua vez, se encarregará do processo de inferência. Isto é expresso pela primeira forma apresentada acima.

A diferença entre a segunda e a terceira formas é muito sutil. Na segunda possibilidade tem-se implementada uma simulação da terceira. Ou seja, o comportamento da rede neural gerada pode ser simulado num computador tradicional. Nesse caso, o trabalho de inferência é realizado por um algoritmo que é executado nesse computador. Essa forma é muito usada durante as pesquisas sobre os modelos e durante os testes de uma rede específica, onde se procura verificar sua utilidade, medida na forma de grau de acerto e confiabilidade, por exemplo. Na

terceira forma, a rede é realmente implementada em hardware. Isto só é feito quando uma rede ideal é encontrada e vai ser produzida em escala para ser usada no campo, em aplicações reais. A vantagem dessa implementação é o aumento da eficiência. Entre as desvantagens, podemos colocar a falta de flexibilidade.

Pelas razões colocadas, usamos neste trabalho a segunda forma. Assim, podemos fazer as alterações necessárias tanto na rede usada como nos algoritmos que a manipulam. Porém, no caso do algoritmo de inferência, deve-se ter o cuidado de garantir que o mesmo também possa ser implementado diretamente em hardware, através do processamento localizado realizado pelas células que compõem a rede. Essa restrição é importante para a utilização real dos produtos da pesquisa.

A restrição sugerida acima vem sendo obedecida pelos autores do Modelo. Os algoritmos de inferência e escolha de perguntas descritos em [Machado 90c], além de incorporarem diversas características interessantes, que veremos a seguir, podem ser implementados através de processamento localizado.

As características interessantes dos "algoritmos" referem-se às perguntas feitas durante o processo de inferência. Existe o cuidado de se escolher a melhor pergunta a ser realizada em cada instante. Essa escolha é baseada no estado da rede e no custo de cada informação. O estado da rede é definido pelo grau de possibilidade corrente de cada célula, pelos valores que esses graus podem alcançar e pelo seu limiar de aceitação – esses dados podem indicar se as células já foram aprovadas, descartadas ou ainda estão "na briga". O custo de cada informação é indicado pela célula correspondente, devendo ter sido estabelecido antecipadamente, com a ajuda de um especialista.

A implementação do processo de inferência e inquisição através de decisões locais é possível devido à propagação simultânea de três informações pela rede: o fluxo evidencial corrente, o fluxo evidencial potencial e o fluxo de custo.

Segundo o próprio autor, o processo de inquisição apresentado permite diminuir significativamente o custo e risco da consulta e pode dar ao sistema a

propriedade de senso comum apresentada por especialistas ao selecionarem testes para serem executados [Machado 90c].

2.7 - Direção das Pesquisas sobre o Modelo

Como vimos neste capítulo, o MNC é um modelo inspirado nos paradigmas simbólico e conexionista. Abordamos aqui as partes que constituem o modelo, algumas de suas características e a forma como se implementa o aprendizado e a inferência sobre o mesmo.

A definição do MNC foi mostrada como um processo evolutivo, baseado em experiências cognitivas, calçadas inicialmente em ferramentas do paradigma simbolista que sofreram, posteriormente, um tratamento inspirado no paradigma conexionista.

Muitas qualidades podem ser encontradas no MNC, juntamente com alguns problemas. Um dos problemas está ligado ao aprendizado. Como pudemos ver, existe uma dificuldade grande para se estabelecer a arquitetura ideal da rede a ser treinada. Mesmo depois de se definir a TCOS, o problema da escolha da arquitetura continua existindo sob a face de um número – a ordem da rede. Redes de tamanho maior proporcionam uma diversidade maior de caminhos a treinar, mas apresentam tamanho muito maior, o que aumenta sobremaneira o esforço computacional despendido pelo algoritmo de aprendizado. Por outro lado, redes de ordem menor, apesar de exigirem um esforço menor dos algoritmos de aprendizado, têm a grave deficiência de proporcionarem um conjunto de conceitos demasiadamente pobre para o treinamento. Ou seja, a qualidade do aprendizado está intimamente ligada ao tamanho da rede que, por sua vez, exerce absoluta influência no esforço computacional do aprendizado, que pode facilmente se tornar impraticável.

A rede a ser treinada funciona como o espaço de busca do algoritmo de aprendizado do Modelo. O espaço de busca grande garante uma maior diversidade de conceitos a serem testados contra os exemplos. Contudo, sabemos que, no meio dessa grande quantidade de conceitos, a esmagadora maioria deve ser descartada.

Assim, podemos procurar formas de descartar previamente aglomerados da rede, antes de submetê-la ao algoritmo de aprendizado. Isto pode ser feito através do uso de conhecimento sobre o domínio, ao qual temos nos referido como conhecimento preliminar (*background knowledge*). Neste trabalho exploramos esta alternativa, fazendo estudos com o uso de dois tipos diferentes de conhecimento preliminar.

Os autores do MNC apresentam uma outra proposta para contornar o problema da qualidade e do esforço computacional do aprendizado no Modelo. A idéia consiste em dotar o Modelo de um *aprendizado evolutivo*, que permita preencher os vazios existentes no conhecimento atual da rede através da geração de novos caminhos ou aglomerados plausíveis, sem que ocorra a destruição das capacidades já estabelecidas. Este aprendizado evolutivo é obtido mediante o emprego de algoritmos genéticos [Denis 91].

A combinação do Modelo Neural Combinatório com algoritmos genéticos é implementada na proposta do Modelo Conexionista Evolutivo (MCE). O uso de algoritmos genéticos permite modificar a topologia da rede durante o aprendizado com o objetivo de explorar parte do espaço de busca não contemplado pelo MNC.

O processo de aprendizado no MCE apresenta três ângulos [Denis 91]:

- Ajuste de pesos, efetuado pelo algoritmo de punição e recompensa do MNC;
- Poda de sinapses irrelevantes, executada pelo algoritmo de poda do MNC;
- Alteração da topologia da rede, com a criação de novos caminhos realizada pelo algoritmo genético.

A alteração da topologia da rede é realizada pelo emprego de operadores genéticos, através de um algoritmo genético. Esses operadores, armazenados na base de operadores genéticos, agem sobre uma população de aglomerados da rede para gerar novos aglomerados. A escolha de quais operadores usar é muito importante nesse contexto. Assim, visando determinar qual a população ideal de operadores genéticos, existe na arquitetura um algoritmo genético de segundo nível, chamado de algoritmo meta-genético, que se responsabiliza pela evolução nesta população.

O MCE é um modelo mais complexo que o MNC. Sua utilização também exige um cuidado maior devido aos parâmetros que se deve ajustar para o seu funcionamento. Além da escolha da ordem, também presente no MNC, deve-se ajustar, através de testes, alguns outros parâmetros. Entre eles, podemos citar o número de iterações dos algoritmos genéticos utilizados e a presença ou ausência de poda durante o treinamento, variando com a iteração.

3 - O papel do Conhecimento Preliminar no Aprendizado

> "O mundo real é incerto e mutável. Todavia, pessoas podem reconhecer seu ambiente, tomar decisões e agir com flexibilidade através do processamento de uma variedade de informações incompletas e ambíguas. Isto é o processamento flexível da informação ou computação para o mundo real".
>
> IEEE Expert, v.7, n.2, p.56.

3.1 - Considerações sobre o Aprendizado por Máquina

A noção do que é aprendizado é muito ampla: aprender denota um ganho de conhecimento, habilidade ou entendimento. Este ganho pode se dar a partir de instrução, experiência ou reflexão. É importante ressaltar que esta noção de aprendizado está intimamente ligada à "aquisição de conhecimento" realizada por seres humanos.

A interpretação de aprendizado na Inteligência Artificial é muito mais restrita e específica. Os sistemas de aprendizado por máquina são firmemente baseados na realidade dos algoritmos práticos. A essência de seu funcionamento é uma busca restrita por um espaço invariavelmente muito grande de descrições possíveis.

De acordo com o tipo de entrada que usam, os algoritmos de aprendizado podem ser baseados em dados (*data driven*) ou em modelos (*model driven*) [Dietterich 83]. Os algoritmos baseados em dados são úteis quando se tem disponível uma grande quantidade de exemplos, mas pouca teoria a respeito do domínio onde se quer aprender. Esses algoritmos devem construir generalizações basicamente a partir desses exemplos empíricos. Já os algoritmos baseados em modelos fazem uso intensivo de uma teoria sobre o domínio já conhecida de antemão. Neste caso, apenas alguns exemplos de casos reais são usados para ajudar a concluir novas assertivas sobre a teoria apresentada.

Sob a ótica da lógica formal, esses algoritmos implementam indução e dedução, respectivamente. Definimos a seguir essas estratégias de inferência. Na indução, busca-se a descrição de um conceito geral a partir de exemplos desse conceito. Isto pode ser formalizado da seguinte maneira: a partir de um conhecimento preliminar[4] **CP** e um conjunto de exemplos **E**, diz-se que o conceito **C** é induzido se e somente se:

1. $CP \nvdash E$ (os exemplos não são dedutíveis do conhecimento preliminar somente);
2. $CP \cup E \nvdash \sim C$ (o conceito não é inconsistente com o conhecimento preliminar e os exemplos);
3. $CP \cup C \vdash E$ (os exemplos são dedutíveis do conceito e do conhecimento preliminar juntos).

Normalmente, $CP \cup E \nvdash C$ (i.e., **C** é induzido e não deduzido). Contudo, se **E** for sabidamente exaustivo, tem-se um tipo especial de indução, conhecida como *summative induction*, que situa-se na linha divisória com a dedução.

Agora suponha um domínio onde exista uma vasta teoria disponível, e para o qual se deseje estabelecer conceitos e regras de controle de busca que tornem mais eficiente a aplicação do conhecimento no dominio. Neste caso, os exemplos e conceitos são ambos dedutíveis do conhecimento preliminar, apesar de não estarem explicitamente presentes neste último. É para este cenário que se presta o aprendizado dedutivo, que pode ser formalizado como segue:

1. $CP \vdash E$ (os exemplos são dedutíveis do conhecimento preliminar);
2. $C \notin CP$ (o conceito não é uma parte explícita do conhecimento preliminar);
3. $CP \vdash C$ (o conceito é dedutível do conhecimento preliminar);
4. $C \vdash E$ (os exemplos são dedutíveis do conceito).

Uma enorme quantidade de conceitos são dedutíveis do conhecimento preliminar; o problema é selecionar aqueles apropriados para os exemplos. Em outras palavras, o conceito aprendido operacionaliza a parte relevante do conhecimento

---

[4] A expressão inglesa é *background knowledge*. Poderíamos traduzir como conhecimento de fundo ou conhecimento sobre o dominio. Consiste no conhecimento disponivel sobre o domínio antes do uso do algoritmo de aprendizado. Pelo fato de representar uma teoria sobre o domínio previamente conhecida, usaremos, em seu lugar, a expressão conhecimento preliminar.

preliminar. Este tipo de aprendizado é também conhecido como baseado em explanação [MacDonald 89].

A dedução é uma inferência que produz sentenças garantidamente verdadeiras, a partir de uma teoria. Contudo, não apresenta um comportamento criador, limitando-se à análise da teoria em face dos exemplos. Como vimos, pode ser usada para destacar aspectos relevantes com base nos exemplos fornecidos.

Já a indução não produz, na maioria dos casos, sentenças garantidamente verdadeiras. As generalizações produzidas a partir de instâncias têm apenas a garantia de não serem inconsistentes com essas instâncias e com o conhecimento preliminarmente fornecido. Por outro lado, a indução realiza um trabalho "criador", de síntese. Isto é imprescindível ao processo de aprendizagem.

Um dos problemas geralmente encontrados nos algoritmos de aprendizado está relacionado à produção de generalizações a partir de dados, com pouca ou frequentemente nenhuma referência ao conhecimento preliminar [Holland 86]. O conhecimento preliminar tem um caráter de complementariedade aos exemplos (conjunto de treinamento), e pode contribuir de duas formas no aprendizado: melhorando a qualidade e utilidade do conhecimento gerado e facilitando o trabalho do algoritmo de aprendizado, através da diminuição do seu espaço de busca.

3.2 - O Aprendizado como Busca

A busca é fundamental para os dois tipos de aprendizado. A busca envolvida tanto no aprendizado indutivo quanto no dedutivo é intratável, a não ser que seja fortemente tendenciada[5], pois o espaço de possibilidades é muito grande. Na verdade, toda busca é tendenciada para algumas descobertas em detrimento de outras através da forma como o espaço é construído. É particularmente importante reconhecermos

---

[5] A palavra inglesa referente a esse conceito é *bias*. Entre as traduções possíveis para este termo está: tendência, propensão, inclinação. O sentido da aplicação de *bias* é guiar a busca restringindo-se o espaço de possibilidades. A busca deve tender para as sentenças ou conceitos de melhor qualidade. Neste trabalho usaremos tendência, guia e limitação do espaço de busca para denotar *bias*.

as tendências inatas, muitas vezes implícitas. A figura 3.1, extraída de [MacDonald 89], mostra duas categorias principais.

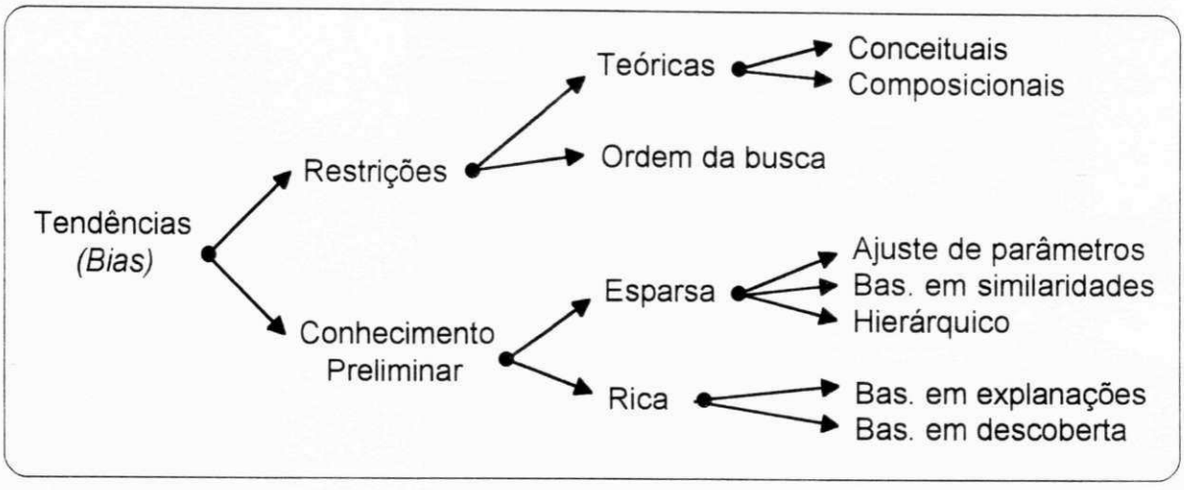


Figura 3.1 – Principais categorias de tendências no espaço de busca. Retirado de [MacDonald 89].

As primeiras restrições que devemos investigar são aquelas impostas pela linguagem usada. Indicadas na figura como restrições teóricas, elas podem ser: conceitual ou composicional. A restrição **conceitual** limita o vocabulário usado para expressar os conceitos. Esta restrição está implicitamente presente em qualquer formulação de um problema. A restrição **composicional** determina como o vocabulário pode ser usado para construir conceitos. Essa restrição limita os conectivos usados na formação de expressões a partir dos termos do vocabulário. As restrições quanto à **ordem da busca** são aplicadas quando não se consegue decidir entre conceitos que competem entre si. Devido à dificuldade de se capturar formalmente a qualidade subjetiva, pode-se optar pelos conceitos que possuem descrição mais simples.

Outra forma de tendenciar a busca consiste na aplicação de conhecimento preliminar. O montante de conhecimento já possuído sobre o domínio afeta de forma crítica o resultado do aprendizado. As técnicas de aprendizado automático podem ser classificadas, de acordo com o grau de utilização de conhecimento preliminar, em ricas e esparsas (figura 3.1). Essa classificação não se baseia tanto no montante de

conhecimento incorporado, mas em até que ponto esse conhecimento é representado explicitamente.

Como veremos adiante, o conhecimento preliminar pode ser representado de diversas formas diferentes. Isto dificulta a caracterização do seu papel no tendenciamento da busca. A classificação apresentada na figura 3.1 é baseada no montante de conhecimento preliminar representado explicitamente. Essa classificação é particularmente relevante para as aplicações de aquisição de conhecimento pois, apesar do nível de utilização de conhecimento preliminar poder ser uma escala contínua, na prática uma classe diferente de métodos é usada quando uma quantidade substancial de conhecimento preliminar é representada explicitamente. Maiores detalhes sobre as técnicas da figura 3.1 podem ser encontrados em [MacDonald 89].

Para ilustrar a importância do conhecimento preliminar na indução, reproduzimos a seguir uma citação do filósofo C. S. Pierce, extraída de [Holland 86]:

> Suponha que um alienígena fosse apresentado ao censo de um país. Ele compararia dados que não necessariamente levariam a generalizações importantes, como a quantidade de mortes em relação à primeira letra do nome dos falecidos, etc. A natureza é um repertório de fatos muito mais vasto e menos claramente disposto que um relatório de censo; e se os homens não forem a ela com aptidões especiais para supor ou adivinhar de forma acertada, talvez em dez ou vinte mil anos sua maior mente possa ter obtido a quantidade de conhecimento atualmente possuída pelo mais simples idiota.

A referência de Pierce às aptidões especias para adivinhar certo reflete sua crença de que as restrições ao que se induzir são baseadas em conhecimento inato, disponível antes do aprendizado.

3.3 - Usando Conhecimento Preliminar

Como vimos no capítulo 1, a aquisição de conhecimento é a tarefa mais importante e mais dispendiosa da confecção de um sistema baseado em conhecimento. O aprendizado automático se coloca como uma possibilidade real para facilitar e baratear essa tarefa. O aprendizado automático consiste em técnicas capazes de captar o conhecimento desejado em um domínio qualquer e representá-lo na forma adequada para que ele possa ser usado por algoritmos computacionais.

Contudo, para que esse conhecimento seja efetivamente usado em aplicações reais, ele deve antes passar por um processo cuidadoso de validação. Isso pode ser feito de duas maneiras diferentes: o teste do conhecimento contra uma bateria de exemplos ou casos reais, de forma a medir o grau de acerto; e a análise direta por especialistas humanos no domínio. A validação pode descobrir alguns problemas com o conhecimento adquirido: a classificação da bateria de teste e a análise cognitiva podem indicar aspectos do domínio não considerados durante o aprendizado. A correção desses defeitos pode ser feita diretamente no conhecimento analisado ou indiretamente, através da alteração das fontes e posterior reaplicação dos algoritmos de aprendizado.

Porém, segundo a filosofia de que construir certo é bem melhor (leia-se menos dispendioso) do que consertar depois, devemos buscar o uso de maior diversidade de informação pelo algoritmo de aprendizado, ao invés de despender maiores esforços na análise e, principalmente, correção do conhecimento resultante. Esse aumento na diversidade da fonte de informação pode ser conseguido de duas formas diferentes: o uso de conjuntos de treinamento maiores (mais próximos de serem completos) proporciona a análise de uma quantidade maior de características, que podem vir a ser consideradas como importantes. Por outro lado, o uso de outras fontes pode indicar alguns outros aspectos relevantes sobre o domínio de uma forma mais direta, quando seriam até difíceis de serem captados a partir de exemplos. Neste trabalho, temos nos referido a essas outras fontes genericamente como conhecimento

preliminar. Como veremos adiante, o conhecimento preliminar pode se apresentar de diversas formas diferentes, podendo ou não ser eliciado de especialistas.

Poderia-se argumentar que a participação de especialistas na formalização de alguns tipos de conhecimento preliminar seria uma volta aos métodos manuais de aquisição de conhecimento (aquisição de conhecimento cognitiva), com todos os seus inconvenientes. Mas esse argumento pode ser facilmente refutado. Os maiores inconvenientes da aquisição de conhecimento manual estão na dificuldade de se estabelecer técnicas genéricas e eficazes de eliciação de conhecimento. O que se propõe aqui é considerar fontes alternativas que disponibilizem mais subsídios para o aprendizado. O aproveitamento de toda a ajuda que puder ser dada a priori (antes da execução do algoritmo) possibilita a diminuição do trabalho necessário a posteriori. Além disso, devido ao fato de serem bastante específicos, os diversos tipos de conhecimento preliminar podem ser eliciados de forma objetiva, através de entrevistas focadas [Vasco 92b].

Essa ajuda a priori pode ser disponibilizada de diversas formas diferentes, como: relevância dos pares atributo-valor para cada hipótese (classe), grafos de hierarquia IS-A, informação quanto ao custo dos atributos, grafos de conhecimento simplificados, árvores de decisão, conjuntos de regras modulares, etc. De acordo com o tipo de conhecimento preliminar usado e com o algoritmo que o utiliza, o conhecimento preliminar pode exercer um papel muito importante ou apenas de apoio durante o aprendizado.

O uso do conhecimento preliminar, juntamente com os dados clássicos de treinamento (os exemplos), permite a geração de uma base de conhecimento melhor, mais próxima da realidade. Isto aumenta a eficácia do aprendizado automático e permite a redução dos esforços pós-algorítmicos.

3.4 - Alguns Tipos de Conhecimento Preliminar

Encaramos aqui o conhecimento preliminar como todo conhecimento que pode ser usado para ajudar o processo de aprendizado. Assim, o conhecimento preliminar

pode se apresentar em diversas formas diferentes. Cada uma delas trata de aspectos específicos do domínio e pode ajudar, à sua maneira, na melhoria da qualidade ou da eficência do aprendizado. Cabe ressaltar que esses diversos tipos de conhecimento preliminar podem ser usados de forma isolada ou simultaneamente.

Veremos a seguir apenas algumas das possivelmente infinitas formas que o conhecimento preliminar pode assumir. Algumas delas são mostradas a nível conceitual; outras são mostradas mais a fundo, através da discussão de como podem ser eliciadas e como podem ser efetivamente úteis em algoritmos de aprendizado.

3.4.1 - A variabilidade das características em função da classe

A observação da maneira como as pessoas aprendem e generalizam conceitos pode exercer um papel muito importante no aprimoramento das técnicas de aprendizado automático. Em seu trabalho sobre os processos indutivos de inferência, aprendizado e descoberta, Holland et al [Holland 86] apontam o uso maciço, pelos seres humanos, de informação sobre a variabilidade durante a generalização. Isto é corroborado por estudos empíricos que mostram que as pessoas são capazes de avaliar o grau de variabilidade de muitas propriedades de forma acurada. Um estudo sobre os aspectos estatísticos da generalização, conduzido por Nisbett, Krantz, Jepson e Kunda, mostra a importânica e a influência das suposições sobre a variabilidade das características de uma classe. Esse estudo, descrito a seguir, foi retirado de [Holland 86], pags. 241 a 243.

Algumas informações sobre novos tipos de objetos foram dadas a pessoas que foram, em seguida, convidadas a fazer generalizações sobre aqueles objetos:

> Imagine que você é um explorador em uma ilha desconhecida do sudeste do Pacífico. Lá, encontra uma série de novos animais, pessoas e objetos. Você observa as propriedades dessas "amostras" e precisa fazer suposições a respeito do quanto essas propriedades são comuns em outros animais, pessoas ou objetos do mesmo tipo daqueles observados.

Suponha que você encontra um novo pássaro de cor azul, que chama de *shrebble*. Qual a percentagem dos *shrebbles* da ilha que você espera que sejam azuis? Por que você supõe essa porcentagem?

Também foi dito às pessoas que o *shrebble* foi visto aninhado em uma árvore de eucalipto e perguntado a elas qual a porcentagem dos *shrebbles* da ilha elas esperavam que estivessem aninhados em árvores de eucalipto.

Em seguida, as pessoas foram convidadas a imaginar que tivessem encontrado um membro da tribo dos "Barratos", que possuia pele morena e era obeso. Perguntou-se então a essas pessoas qual a porcentagem dos Barratos machos que elas esperavam ser morenos e qual a porcentagem que elas esperavam ser obesos.

Finalmente, foi-lhes dito que haviam encontrado uma amostra de um elemento raro chamado "florídio". Foi constatado que ele conduzia eletricidade e queimava produzindo uma chama verde quando esticado como um filamento e aquecido a uma temperatura muito alta. Foi perguntado às pessoas qual porcentagem de florídio elas esperavam que conduzisse eletricidade e que queimasse com uma chama verde.

Duas outras condições na mesma experiênica foram colocadas. Na primeira, foi dito às pessoas que três amostras de cada objeto haviam sido encontradas, ao invés de apenas uma. Na segunda, foi dito que vinte amostras de cada objeto haviam sido encontradas.

As razões dadas pelas pessoas para as estimativas apresentadas recairam em três tipos básicos:

1. referências à homogeneidade do tipo de objeto com respeito ao tipo de propriedade;
2. referências à heterogeneidade ou variabilidade do tipo de objeto com respeito ao tipo de propriedade, atribuída à existência de subtipos com propriedades diferentes (por exemplo, macho e fêmea) ou a algum mecanismo causal que produza propriedades diferentes (como um erro genético) ou puramente à

variabilidade estatística ("o lugar onde os *shrebbles* aninham é muitas vezes uma questão de oportunidade");

3. outras razões, em sua maioria meras tautologias.

Os resultados dos experimentos foram bastante claros. A figura 3.2 mostra as estimativas apresentadas pelas pessoas sobre a porcentagem de objetos de cada tipo que elas esperavam que tivessem os atributos das amostras. Pode-se ver que as pessoas esperaram essencialmente que todo florídio tivesse as propriedades de condutividade e de queimar com uma chama verde. Eles também esperaram que essencialmente todos os Barratos tivessem pele morena. Essas expectativas ocorreram a despeito do número de casos na amostra.

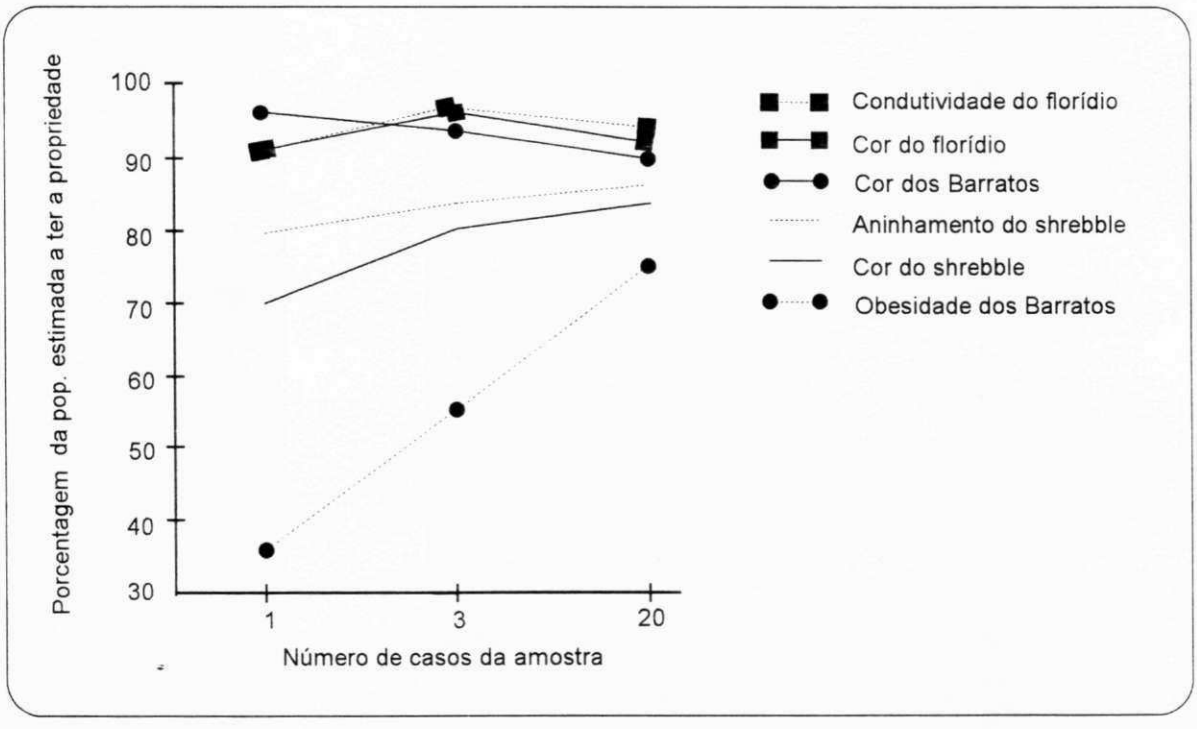


Figura 3.2 – Porcentagem estimada da presença da propriedade em função do número de casos da amostra.

Por outro lado, as pessoas não esperaram que todos os *shrebbles* fossem azuis ou que aninhassem em árvores de eucalipto, mesmo que vinte *shrebbles* tivessem sido observados. As pessoas ainda esperaram uma porcentagem ainda menor de Barratos obesos do que *shrebbles* azuis ou *shrebbles* aninhados em árvores de eucalipto, e suas crenças sobre a porcentagem de Barratos obesos mostrou-se altamente

dependente do número de casos da amostra. Isto ocorreu porque, até onde se sabe, essas ilhas são habitadas por povo de um mesmo grupo étnico, caracterizados pela cor, pela estatura média, etc, mas não pelo peso corporal. Assim, para que essa generalização fosse feita, seria necessária a observação de uma grande quantidade de exemplares e que a característica se repetisse neles.

As crenças das pessoas sobre a variabilidade dos objetos em questão com respeito às propriedades citadas claramente determinaram esse padrão de respostas. As pessoas reportaram acreditar que as propriedades dos elementos são invariantes através das amostras e que os componentes de uma tribo são homogêneos com respeito à cor. Elas acreditaram que os tipos de pássaros são homogêneos apenas ocasionalmente com respeito à cor e local preferido de aninhamento; e eles acreditaram que mesmo grupos humanos isolados são muito heterogêneos com respeito ao peso corporal. Em cada ponto do gráfico, as diferenças individuais na estimativa de percentagens foi dependente de diferenças individuais na crença sobre variabilidade. As pessoas que estimaram altas porcentagens foram aquelas que pensaram que o tipo do objeto não era muito variável com respeito ao tipo de propriedade.

Assim, para este tipo simples de generalização, está claro que as pessoas são capazes de temperar a força de suas conclusões sobre tipos de objetos de acordo com o grau de homogeneidade que elas esperam. Em um extremo, uma generalização muito forte pode ser formada com base em um único caso, se homogeneidade for presumida. No outro extremo, onde grande heterogeneidade é presumida, mesmo um grande número de casos não resultará em uma generalização forte. Como podemos ver, a propensão para generalizar depende fortemente da expectativa de variabilidade da característica observada. Quanto maior for a variabilidade esperada, menor é propensão de generalizar, ou seja, maior precisa ser a quantidade de casos observados para que a generalização aconteça.

O conhecimento sobre a variabilidade de cada característica, com respeito a cada classe, é fundamental na decisão sobre que generalizações se pode fazer a partir

de uma determinada quantidade de exemplos observados. Se esse conhecimento estiver disponível, pode ajudar muito no aprendizado automático, causando uma melhoria na qualidade do conhecimento produzido.

3.4.2 - Hierarquia e custo

Os algoritmos da família TDIDT[6] não fazem uso de conhecimento preliminar. Por essa razão, geram árvores de decisão que eventualmente não parecem lógicas nem inteligíveis para os especialistas. Esses algoritmos não são capazes de construir generalizações em um nível de abstração maior que aquele dos atributos que descrevem os exemplos usados para a indução. Além disso, também não se preocupam em reduzir o custo da classificação realizada com base no uso da árvore por ele gerada.

Esses inconvenientes apontados por Núñez [Núñez 91] não são exclusivos dos algoritmos da família TDIDT. Na verdade, nenhum algoritmo de aprendizado que se baseie apenas em exemplos sobre o domínio pode sequer captar o custo inerente à aquisição das informações do ambiente. Também não pode trabalhar em níveis de abstração maiores sem lançar mão de algum conhecimento que indique como as características se agrupam nesses níveis de abstração. Os jargãos conhecidos pelo algoritmo são somente aqueles em que os exemplos são expressos: pares atributo-valor.

A consequência direta desses inconvenientes apontados por Núñez é a baixa qualidade do conhecimento gerado pelos algoritmos de aprendizado. O aprendizado a partir de exemplos que não leva em consideração o custo das informações corre o risco de ter pouca utilidade prática. Como exemplo, imagine um perito em tumores cerebrais atendendo um paciente que se queixa de dor de cabeça. O exame mais efetivo, neste caso, seria a tomografia computadorizada; mas o perito não a solicita no primeiro instante, devido ao seu alto custo. Este tipo de exame só é utilizado depois de se esgotar os testes mais comuns, que permitem a solução da maioria dos casos a um custo bem inferior. Apesar desse exemplo, o custo de uma informação

[6] Top Down Induction of Decision Trees. TDIDT é uma família de algoritmos de indução que geram árvores de decisão. O representante mais conhecido desta família é o ID3.

não é expressa sempre em valores monetários. Esse custo pode também ser derivado do risco inerente ao exame, como uma biópsia, da dor causada ou de qualquer outra dificuldade ou facilidade para obter a informação desejada.

O fato de não se usar informações hierárquicas sobre o domínio pode provocar a geração de bancos de conhecimento grandes e pouco expressivos. Em muitos casos, dependendo do domínio e do conjunto de treinamento, o baixo nível de abstração da linguagem usada para representar o domínio, aliado à inexistência de conhecimento hierárquico sobre o mesmo, faz com que a base de conhecimento gerada contenha muitas asserções particulares ao invés de generalizações aproveitáveis.

Para resolver esse problema, Núñez [Núñez 91] propõe o uso de um grafo IS-A, que estabelece uma hierarquia através do agrupamento de valores dos atributos, formando níveis de abstração mais elevados. Como exemplo, considere um domínio de figuras geométricas que devem ser classificadas de acordo com um critério qualquer. Os exemplos submetidos ao algoritmo de aprendizado são formados por pares atributo-valor. Um dos atributos é o *formato* da figura geométrica, que pode assumir os valores: quadrado, triângulo, pentágono, círculo e elipse. O conhecimento hierárquico proposto por Núñez divide os valores do atributo *formato* em dois grupos: o dos polígonos (quadrado, triângulo e pentágono) e o das cônicas (círculo e elipse). Assim, num nível de abstração superior, o atributo *formato* assume os valores polígono e cônica. Esses atributos associados a uma hierarquia são batizados de *atributos estruturados*. Os valores das folhas são os únicos observáveis; os valores dos níveis de abstração superiores são chamados de *valores abstratos*.

Os dois tipos de conhecimento preliminar relacionados aqui – custo e grafo de hierarquia – são usados pelo algoritmo EG2 (*Economic Generalizer 2*) [Núñez 88] [Núñez 91] para realizar aprendizado de conhecimento econômico com alto nível de generalização. O conhecimento gerado pelo EG2 está na forma de árvore de decisão. Além dos dois tipos de conhecimento preliminar e do conjunto de exemplos, o EG2 usa ainda dois parâmetros definidos pelo usuário: o fator de economia e o limiar de completude. O fator de economia deve estar entre 0 e 1, quando determina economia

máxima. O limiar de completude gradua a liberdade do algoritmo para generalizar sem que todos os valores observáveis participem do conceito a ser generalizado.

As mudanças na árvore de decisão provocadas pelo uso das informações sobre o custo são sentidas pela tendência de distanciar os atributos mais caros da raiz da árvore de decisão, mesmo que eles sejam mais efetivos para a classificação. Esse distanciamento faz com que o valor do atributo caro somente seja solicitado depois de esgotadas as tentativas de classificação com o uso dos atributos mais baratos. Note que a ênfase das preocupações com relação à árvore gerada não está mais no seu tamanho. A árvore de menor custo pode não ser a menor árvore.

Já o uso das informações hierárquicas tende a produzir uma árvore de decisão menor e com maior nível de generalidade. Isto ocorre quando há uma certa relação entre alguns valores abstratos e classes. Neste caso, os vários caminhos que levam a uma mesma classe, cada um deles representando um valor diferente para um mesmo atributo, podem ser unificados pelo assinalamento de um valor abstrato ao atributo, se existir um tal valor que generalize todos os valores substituídos.

Os tipos de conhecimento preliminar sobre hierarquia e custo são independentes entre si. Eles foram mostrados aqui de forma conjunta para ilustrar como aspectos diferentes do domínio podem influenciar o conhecimento gerado de maneiras até conflitantes, mas que acabam por propiciar uma melhoria real no resultado do aprendizado.

Mostramos a aplicação desses tipos de conhecimento em um determinado algoritmo de aprendizado, que representa o conhecimento captado em árvores de decisão. Isso foi útil para salientar a importância do seu uso para a qualidade da aquisição e utilidade do conhecimento adquirido, mas não quer dizer que o uso dos mesmos seja privativo a esse algoritmo ou a essa família de algoritmos. Quanto a isso, há um aspecto a ressaltar: a árvore de decisão embute uma estratégia de inferência ao determinar a sequência em que as informações devem ser buscadas no ambiente. Outras formas de representação de conhecimento não possuem essa característica. Nesses casos, as informações sobre custo devem estar disponíveis para

serem usadas durante o aprendizado (construção da base de conhecimento) e a inferência (uso da base de conhecimento).

3.4.3 - Relevância Semântica

Como vimos, os algoritmos de aprendizado automático baseados em exemplos têm negligenciado o uso do conhecimento preliminar, deixando de aproveitar as vantagens que o mesmo pode oferecer. O uso exclusivo dos exemplos do conjunto de treinamento como subsídio para o aprendizado pode acarretar problemas, diminuindo a qualidade do conhecimento adquirido. Um desses problemas foi identificado por Mongiovi e Cirne [Mongiovi 91], que o chamaram de **problema semântico**.

O Problema Semântico

O problema semântico foi identificado durante o trabalho de validação de um banco de conhecimento para diagnóstico em ginecologia, gerado por uma implementação do algoritmo ID3 [Quinlan 83] a partir de casos de um fichário médico. Durante a validação, resolveu-se fazer uma análise semântica do banco de conhecimento, submetendo-o à apreciação de um especialista. Essa análise apontou regras semanticamente incorretas (chamadas de regras inúteis). Entre as regras assinaladas, a que mais chamou atenção foi uma que considerava suficiente a condição *IDADE=criança* para concluir o diagnóstico *VAGINITE*.

A investigação para identificar os motivos da geração dessas regras revelou que o problema não está no algoritmo de aprendizado em si, mas na pobreza do conjunto de treinamento, que é formado exclusivamente por exemplos. Em problemas reais, o conjunto de treinamento só contém uma minúscula parte de todos os exemplos possíveis (todas as combinações de atributos e seus valores). Conjuntos de treinamento completos, i. e., que contêm todos os casos possíveis, só são administráveis em pequenos problemas de brinquedo. Em problemas reais, a completude do conjunto de treinamento não é sequer cogitada, devido à dificuldade de sua construção e ao custo de sua manipulação. Mesmo que ele esteja facilmente disponível, não pode ser completamente usado. Na verdade, devido a problemas de

esforço computacional, a maioria dos algoritmos de indução acabam usando apenas uma amostra do conjunto de treinamento, nos casos em que este é muito grande.

A incompletude do conjunto de treinamento apresenta efeitos diferentes em algoritmos de aprendizado diferentes. O ID3 escolhe o próximo atributo a usar na confecção da árvore de decisão através do cálculo da entropia. Em um determinado caso, pode acontecer que, por uma simples coincidência, uma classe esteja sendo plenamente determinada por uma condição semanticamente irrelevante para a mesma. Neste caso, será introduzido na árvore um atributo irrelevante para a classe. A introdução do atributo pode até permitir a classificação de todos os exemplos restantes daquela classe. Assim, a árvore de decisão gerada classifica corretamente os exemplos constantes no conjunto de treinamento, mas poderá falhar fragorosamente na classificação de outros casos.

Segundo Cirne [Cirne 91], uma circunstância em que a geração de regras inúteis frequentemente ocorre é quando o conjunto de exemplos contém casos raros. Consideram-se casos raros aqueles caracterizados por condições que ocorrem apenas em pouquíssimos exemplos. Esta é a explicação para a geração da regra *SE IDADE=criança ENTÃO VAGINITE*. Esta regra foi gerada porque todas as crianças do conjunto de treinamento estavam com vaginite. Isto é uma coincidência que talvez possa ser explicada pela origem do conjunto de treinamento: um fichário médico. Ao escolher o atributo IDADE para compor a árvore de decisão, o ID3 encerra a construção do ramo correspondente ao caso, pois todos os exemplos relacionados ao caso ficam classificados apenas com *IDADE=criança*. Os exemplos do caso raro ficam, então, mapeados numa regra inútil, uma vez que a premissa da regra acima não é suficiente, nem importante, para concluir se a paciente está ou não com vaginite.

A ocorrência de casos raros forma uma situação propícia para a geração de regras inúteis, pois as chances de erro numa pequena amostragem são maiores do que em uma mais significativa. Porém, embora mais fácil de ocorrer em casos raros, esse

problema pode aparecer também em outras situações, sob condições diferentes. Para um melhor estudo dessas condições, veja [Cirne 91] e [Cirne 92].

A Relevância Semântica

Sabemos que nem todas as características de um objeto são relevantes para a determinação de sua classe. Apenas uma parcela de todas as possíveis condições são efetivamente relevantes para a determinação da classe. O papel de um algoritmo de aprendizado é justamente descobrir quais são os aspectos relevantes para se poder afirmar que um objeto pertence a uma determinada classe. Informalmente, "a relevância se refere ao fato de que certas premissas são mais importantes que outras para uma conclusão particular" [Dutta 88].

Os algoritmos procuram as regularidades no domínio do problema através do estudo de aspectos sintáticos dos elementos do conjunto de treinamento. Contudo, devido à incompletude do conjunto, não existe uma garantia real de que todas as regularidades do conjunto se repitam nos casos reais. Isto é, em última análise, o que quer dizer o problema semântico. É a falta de semântica no conjunto de treinamento; é a diferença que sempre haverá entre o conjunto completo e seu melhor representante possível.

O problema semântico não pode ser resolvido através do simples aumento do conjunto de treinamento, pois já vimos que, em casos reais, a utilização de conjuntos completos é inviável. Além disso, conjuntos pequenos, bem escolhidos, podem satisfazer à maioria das necessidades. Assim, convém estudarmos uma forma alternativa de suprir as deficiências ligadas ao problema semântico; alguma forma que nos permita captar uma relevância que complemente aquela captada através do estudo sintático do conjunto de treinamento.

O especialista no domínio pode ser uma boa fonte complementar de relevância. A participação ativa de um agente humano no processo de aprendizado lhe empresta características cognitivas. Por causa disto e para realçar o objetivo de atacar o problema semântico, a relevância eliciada de especialistas humanos é chamada de relevância semântica.

Eliciação – A Matriz de Relevância

Para se garantir a utilidade da contribuição de um especialista, sem recair em alguns inconvenientes da aquisição de conhecimento cognitiva, a forma de participação do perito humano deve ser bem definida, de modo a ser o mais objetiva possível. Essa objetividade proporciona maior produtividade que as entrevistas conduzidas durante a aquisição de conhecimento cognitiva. A maior objetividade na eliciação de um tipo de conhecimento é conseguida através da criação de ferramentas de eliciação específicas para o tipo de conhecimento desejado. A relevância semântica conta com uma ferramenta dessas. Trata-se da matriz de relevância [Mongiovi 90] [Cirne 91] [Cirne 92].

A relevância semântica pode ser eliciada diretamente de um especialista no domínio através do preenchimento de uma matriz de relevância. A matriz de relevância representa uma forma de se especificar quais são os pares atributo-valor relevantes para a determinação de cada classe. A seguir, reproduzimos uma definição de matriz de relevância (**MR**), retirada de [Cirne 92]:

> Uma MR para um dado domínio é o relacionamento de relevância entre os elementos de classificação e os atributos deste domínio. Em uma MR, a linha i representa o atributo a_i e a coluna j o elemento de classificação E_j. Um elemento r_{ij} de uma MR indica o grau de relevância do atributo a_i para a classificação de E_j. Este grau é dado pelo conjunto de valores do atributo a_i que são relevantes para a conclusão de E_j. O elemento r_{ij} é, na verdade, um subconjunto do conjunto de valores do atributo a_i. Se r_{ij} for vazio, significa que o atributo a_i é irrelevante na classificação de E_j. Ao passo que, r_{ij} igual ao conjunto de valores de a_i expressa que a_i é totalmente relevante na classificação de E_j. Uma MR é dita completa se todos os atributos forem totalmente relevantes para todas as classes. Uma MR é dita vazia se todos os seus elementos forem iguais ao conjunto vazio.

As experiências de eliciação de relevância semântica através do uso da matriz de relevância não apresentaram problemas. Sua forma tabular tem sido facilmente entendida pelo público-alvo e seu preenchimento tem se dado sem maiores dificuldades, em uma boa velocidade. Isto é possível por causa das características da relevância semântica. Durante o preenchimento da matriz de relevância, o especialista não precisa estabelecer implicações completas entre a característica, ou grupo de características, e classes, mas somente indicar a possível existência de relação entre características individuais e cada uma das classes do domínio. A figura 3.3, extraída de [Cirne 91], apresenta um exemplo de matriz de relevância bastante simples. Dela, pode-se concluir:

a) O atributo *COME* é relevante para classificar *MAGRO* no valor *pouco*, e é relevante, no valor *muito*, para classificar *GORDO*;
b) O atributo *DIABÉTICO* é relevante para classificar *GORDO* no valor *sim* e *MAGRO* no valor *não*;
c) O atributo *IDADE* é irrelevante para classificar *MAGRO* e *GORDO*;
d) O atributo *VEGETARIANO* é relevante para classificar *GORDO* no valor *não* e *MAGRO* no valor *sim*.

	MAGRO	GORDO
COME	{pouco}	{muito}
DIABÉTICO	{não}	{sim}
IDADE	Ø	Ø
VEGETARIANO	{sim}	{não}

Figura 3.3 – Exemplo simples de uma matriz de relevância, extraída de [Cirne 91].

Depois de devidamente preenchida pelo especialista, a matriz de relevância pode apresentar diferentes graus de preenchimento, que determinam sua intensidade de relevância. Uma matriz de relevância completa indica que todos os pares atributo-valor válidos formados pelos atributos nela contidos são importantes (relevantes) para a determinação da classe de um caso (classificação de uma instância). Uma matriz de relevância vazia indica que nenhuma informação referente àqueles atributos é relevante para a determinação da classe de um caso. Entre uma

matriz de relevância completa e uma matriz de relevância vazia, podemos identificar uma grande quantidade de matrizes de relevância possíveis, com diferentes graus de preenchimento.

Diminuição do Espaço de Busca

Nas seções anteriores, vimos que o propósito inicial do uso da relevância semântica foi a melhoria da qualidade do conhecimento adquirido pelos algoritmos de aprendizado, através da resolução do problema semântico.

Por outro lado, ao estudarmos algumas definições sintáticas de aprendizado indutivo e dedutivo, vimos que o aprendizado pode ser entendido como uma busca por um espaço de possibilidades que descrevem o domínio. Esta busca deve identificar quais são aquelas descrições válidas de acordo com os subsídios apresentados ao algoritmo de aprendizado – conjunto de treinamento e conhecimento preliminar.

Pois além do benefício semântico, fortemente ligado à utilidade das asserções produzidas pelo aprendizado, a relevância semântica apresenta um benefício colateral, mais ligado a aspectos sintáticos do domínio. Trata-se da diminuição do espaço de busca que, por sua vez, pode acarretar uma grande diminuição do esforço computacional do algoritmo de aprendizado.

A diminuição do espaço de busca é uma consequência direta do uso da relevância semântica. Uma rápida inspeção na matriz de relevância permite apontar uma grande quantidade de evidências completamente irrelevantes para a conclusão de diversas classes. Assim, devemos poupar o esforço computacional de testar essas evidências contra aquelas classes, pois, mesmo que elas se saiam bem (o que é difícil), deverão ser eliminados no futuro, quer seja através do refinamento do conhecimento, graças à submissão de outros exemplos, ou mesmo através da sua inspeção por um especialista. Assim, no início do processo de aprendizado, ao invés de termos um desconhecimento geral do domínio, situação em que qualquer combinação entre todas as possíveis condições é potencialmente importante para a conclusão de qualquer classe, temos um espaço de possibilidades potencialmente

importantes bem menor. Este novo espaço de possibilidades é formado pela combinação das condições semanticamente relevantes para cada classe.

A diminuição do espaço de busca está diretamente ligada ao grau de preenchimento da matriz de relevância. Se a matriz de relevância for completa, o espaço de busca continua o mesmo, sem qualquer diminuição. O motivo disto é que todas as condições continuam potencialmente válidas para a conclusão de todas as classes. Por outro lado, se a matriz de relevância for vazia, o espaço de busca também se torna vazio, pois nenhuma condição entre aquelas possivelmente formadas pelos atributos relacionados durante a modelagem do domínio se mostra importante para a conclusão de nenhuma classe. A prática mostra que nenhuma das possibilidades anteriores se verifica. A matriz de relevância usualmente possui um grau de preenchimento intermediário.

Vimos que a matriz de relevância apresenta a definição de um espaço de atributos relevantes, menor que o original, e cuja investigação merece o esforço despendido. Contudo, o aproveitamento dessa diminuição do espaço de possibilidades depende muito da estratégia adotada pelo algoritmo de aprendizado. Veremos, no próximo capítulo, como o Modelo Neural Combinatório pode aproveitar-se dessa facilidade.

4 - Aplicando Semântica ao MNC

No capítulo 3, mostramos alguns tipos de conhecimento preliminar, bem como as vantagens que seu uso pode trazer ao aprendizado por máquina. Entre os tipos apresentados, está a relevância semântica, que pode contribuir no processo de aprendizado basicamente de duas formas:

1. Minimização do problema semântico, resultante da incompletude do conjunto de treinamento, aumentando a qualidade do aprendizado desenvolvido pelo Modelo;
2. Diminuição do espaço de busca, podendo proporcionar a diminuição da complexidade computacional do algoritmo de aprendizado.

As contribuições da relevância semântica podem ser usufruídas a um custo baixo, devido à objetividade e simplicidade da ferramenta de eliciação e armazenamento – a matriz de relevância.

Contudo, o aproveitamento das vantagens oferecidas pela relevância semântica depende do algoritmo de aprendizado, mais especificamente da estratégia usada para percorrer o espaço de conceitos possíveis. Em especial, a existência do segundo benefício depende de alguns parâmetros, como o grau de preenchimento da matriz de relevância, o número de classes do domínio e o tipo de implementação que se faça do algoritmo.

Veremos, neste capítulo, uma forma de agregar a relevância semântica ao Modelo Neural Combinatório. Em primeiro lugar, apresentaremos a modificação sugerida. Em seguida, faremos análises da modificação e, finalmente, comparações entre a situação original do Modelo e a modificada. Devido às características da modificação proposta, lançaremos mão de um exemplo representativo para fazer a comparação.

4.1 - A relevância semântica no Modelo Neural Combinatório

Como vimos, o aprendizado automático pode ser visto como uma busca em um espaço de possibilidades. No MNC, essa busca ocorre através do percorrimento da rede inicial, que se dá durante o seu treinamento, feito com o fito de ajustar-lhe os pesos. Assim, para o MNC, o espaço de possibilidades é a rede a ser treinada, visto que a mesma contém todos os conceitos que podem ser aprendidos pelo Modelo.

Uma consulta à matriz de relevância, antes mesmo do início do treinamento da rede, já pode identificar vários caminhos ou aglomerados como fadados ao insucesso, por representarem conceitos irrelevantes (sem importância semântica para a classe). A maioria desses conceitos será eliminada da rede durante o treinamento. Contudo, alguns deles podem permanecer na rede gerada pelo algoritmo de aprendizado. A partir dos trabalhos de Mongiovi e Cirne, podemos concluir que a presença desses conceitos na base de conhecimento é uma anomalia derivada da incompletude do conjunto de treinamento. Como vimos, esta anomalia pode ser sanada através do uso da relevância semântica.

Podemos melhorar a qualidade da rede gerada e diminuir o esforço computacional do algoritmo de aprendizado através da eliminação prévia de caminhos irrelevantes da rede original, usada pelo algoritmo de aprendizado do MNC. Devemos ressaltar que, mesmo que esses caminhos "se saissem bem" no treinamento, eles acabariam por ser eliminados durante um posterior refinamento da rede. Esse refinamento, que tende a eliminar os caminhos da rede correspondentes às regras inúteis, poderia ocorrer através da submissão de outros exemplos (aprendizado incremental) ou mesmo através da inspeção da rede por um especialista.

Assim, a relevância semântica deve ser aplicada para podar previamente a rede inicial. Isto é feito através da definição de uma nova topologia de rede para o Modelo. Trata-se da Topologia Combinatorial Relevante de Ordem Superior, que passa a definir a rede que deve ser usada para treinamento no Modelo.

4.2 - A Topologia Combinatorial Relevante de Ordem Superior (TCROS)

Os algoritmos de aprendizado são muito dependentes de uma atividade conhecida como busca. Na verdade, a busca, inerente tanto ao aprendizado indutivo quanto ao dedutivo, torna-se intratável a não ser que seja fortemente restringida ou podada através do uso de tendências (*bias*), devido ao enorme tamanho do espaço de possibilidades [MacDonald 89]. O conhecimento preliminar pode ser um forte instrumento na poda do espaço de busca. E, de fato, a relevância semântica cumpre muito bem este papel. Conheceremos, a seguir, a Topologia Combinatorial Relevante de Ordem Superior (TCROS).

A TCROS é formada pelos aglomerados (combinações) relevantes que compõem a Topologia Combinatorial de Ordem Superior. Diz-se que um aglomerado é relevante quando todas as suas evidências (células de entrada, que representam pares atributo-valor) são relevantes. Ou seja, a TCROS pode ser obtida a partir da TCOS através da supressão dos aglomerados não relevantes (i.e., onde pelo menos uma das evidências que o compõem seja formada por um par atributo-valor que não pertença à matriz de relevância eliciada do especialista).

A partir de sua definição, podemos ver que o tamanho da TCROS depende de dois parâmetros:

1. O tamanho da TCOS que o origina, dado por sua ordem e pelo número de evidências e classes do domínio.
2. O grau de relevância e a forma como está distribuída, entre as evidências e as classes do domínio, dados pela quantidade relativa de evidências relevantes e sua distribuição entre as classes, na matriz de relevância.

Esses parâmetros podem ser resumidos em: ordem da topologia e composição da matriz de relevância. Geralmente, o que se deseja, em problemas reais, é a possibilidade de se usar a maior ordem possível, dentro de certos limites, mas que ainda defina uma rede computacionalmente tratável. Devido ao alto crescimento da TCOS, o limite superior ainda está longe de ser atingido. Posteriormente, voltaremos a esse assunto. Por ora, desejamos estudar o comportamento da influência desses

dois parâmetros no esforço computacional despendido pelo algoritmo de aprendizado.

Numa análise inicial, as perspectivas que surgem com o uso da TCROS são muito promissoras. Mostraremos aqui os dados colhidos a partir de um estudo de caso em ginecologia [Mongiovi 90b]. O domínio foi modelado com 46 atributos, formando 158 evidências (pares atributo-valor), e 18 classes. Na matriz de relevância eliciada para esse domínio, a classe com o maior número de evidências relevantes aparece com 44, enquanto que a de menor número possui 3 evidências relevantes, fazendo uma média de cerca de 19 evidências relevantes por classe, com um desvio-padrão de 10,78.

Como somente as evidências relevantes entram na composição da rede, vê-se que a quantidade de evidências a combinar caiu de forma impressionante, passando de 158 para, no máximo, 44 (veja na tabela da figura 2.7 os tamanhos das topologias de ordem 5, para as quantidades de evidências citadas). Como NC(n, m), que mede o tamanho da rede de ordem *'m'* para *'n'* evidências, varia exponencialmente com a variação de *'n'*, qualquer acréscimo ou decréscimo no parâmetro *'n'* repercute significativamente no tamanho da topologia.

De acordo com essa análise inicial, espera-se uma grande diminuição do esforço computacional do algoritmo para treinar redes dessa topologia. Contudo, há uma característica da TCROS que abranda a certeza da diminuição do esforço computacional, colocando-se como um forte obstáculo ao usufruto desta diminuição da topologia. Trata-se da sua diferenciação por classe, que resulta em subredes heterogêneas. Esta característica se contrapõe à homogeneidade da TCOS.

Quando o algoritmo de aprendizado usa a TCOS, ele se beneficia da homogeneidade dessa topologia entre as diversas classes, fazendo o treinamento de todas as subredes em "paralelo". Durante o aprendizado, as combinações são geradas e testadas contra os exemplos da seguinte forma: para cada combinação gerada, todos os exemplos são testados, à procura daqueles que satisfazem a combinação em questão. Um exemplo satisfaz a uma combinação quando contém todas as evidências

que formam a combinação. Na terminologia de redes neurais, isto equivale a dizer que o exemplo consegue propagar-se pelo caminho formado pela combinação em questão.

Assim, quando é encontrado um exemplo que satisfaz a combinação em questão, os pesos daquela combinação correspondentes a todas as classes são alterados da seguinte forma: a classe a qual pertence o exemplo tem o peso daquela combinação de sua subrede aumentado (recompensa) e as demais classes têm aquele peso diminuído (punição). É por isso que dizemos que as subredes relativas a todas as classes são geradas "simultaneamente", numa única consulta à subrede padrão da topologia, que serve de guia para o aprendizado. Dessa forma, economiza-se o esforço de gerar várias vezes as mesmas combinações e testá-las contra todos os exemplos, para todas as classes, visto que a subrede original a ser treinada é idêntica para todas as classes.

O uso da TCROS, contudo, inviabiliza esse "paralelismo". Como cada classe possui um conjunto de evidências relevantes diferente, as diferentes subredes devem ser geradas separadamente, cada uma para o treinamento de sua classe específica. Cada subrede deve ser submetida a todos os exemplos do conjunto de treinamento para calcular os seus pesos, relativos a uma única classe. Em seguida, outra subrede, relativa a outra classe, é gerada para o treinamento e assim sucessivamente.

4.3 - Análise comparativa entre TCROS e demais topologias

A partir do que foi exposto até aqui, não se torna óbvio que o uso da TCROS proporcione uma diminuição do esforço computacional como um todo, pois, apesar do tamanho de cada subrede ser menor ou igual ao que seria na TCOS, a heterogeneidade entre as subredes da TCROS obriga que as mesmas sejam geradas separadamente. O espaço de busca total a ser pesquisado pelo algoritmo é dado pela soma das diversas subredes da TCROS. Assim, para que o uso da TCROS seja firmemente favorável, a diminuição do tamanho de cada subrede tem que ser suficiente para que a soma delas seja menor que a subrede padrão da TCOS.

O tamanho da Topologia

A seguir, faremos uma análise sobre o comportamento do tamanho das topologias, medido em quantidade de aglomerados ou caminhos ou combinações, que é igual ao número de nós da camada intermediária. Esta medida de tamanho é derivada do fato das topologias em questão serem todas "achatadas", i.e., grafos com apenas três camadas, sendo que a intermediária é a camada combinatória e possui apenas um nível. O esforço computacional despendido pelo algoritmo de treinamento tem correspondência direta com o tamanho da topologia treinada.

O tamanho de cada subrede da TCROS é diretamente relacionado ao grau de preenchimento e à disposição dos valores na matriz de relevância. A expressão para o tamanho da subrede padrão da TCOS é:

$$NC(n, m) = \sum_{i=1}^{m} C_{n,i}$$

onde *'n'* é o número de evidências do domínio e *'m'* a ordem da topologia. Já no caso da TCROS, o tamanho da subrede relativa à classe *'c_i'* é:

$$NCR(i, m) = NC(nerc_i, m) = \sum_{j=1}^{m} C_{nerc_i,j}$$

onde *'$nerc_i$'* é o número de evidências relevantes para a classe *'c_i'*, ou seja, o número de elementos (representando pares atributo-valor) presentes na coluna *'i'* da matriz de relevância.

Assim, o tamanho da rede TCROS é:

$$\sum_{i=1}^{nc} NCR(i, m) = \sum_{i=1}^{nc} NC(nerc_i, m) = \sum_{i=1}^{nc} \sum_{j=1}^{m} C_{nerc_i,j}$$

onde *'nc'* é o número de classes do domínio.

O esforço computacional

Para uma matriz de relevância completa, as diversas subredes da TCROS são iguais entre si e iguais à subrede padrão da TCOS. Nesse caso, que na prática é irreal, o uso da TCOS é mais conveniente porque a soma dos tamanhos das subredes da TCROS é muito maior que o tamanho da subrede padrão da TCOS. À medida que

se altera a matriz de relevância, diminuindo-se seu grau de preenchimento, as subredes da TCROS vão diminuindo de tamanho e se diferenciando. O ponto em que a redes TCOS e TCROS despendem o mesmo esforço computacional para o aprendizado pode ser descrito pela equação

$$\sum_{i=1}^{nc} NCR(i, m) = NC(n, m) \cdot k \qquad \text{(E1)}$$

onde:

- *'n'* é o número de evidências do domínio;
- *'m'* é a ordem da topologia (tamanho máximo dos aglomerados);
- *'nc'* é o número de classes do domínio;
- NC(n, m) é o tamanho da TCOS de ordem *'m'* para um domínio de *'n'* evidências, dado em quantidade de aglomerados;
- NCR(i, m) é o tamanho da TCROS de ordem *'m'* para a classe *'c_i'*, com *'$nerc_i$'* evidências relevantes. Esse tamanho é dado em número de aglomerados. NCR(i, m) = NC($nerc_i$, m).
- *'k'* é uma constante que deve assumir um valor no intervalo (0, *nc*], dependendo da implementação.

Pela expressão acima, é possível notar que, apesar do esforço computacional ser diretamente proporcional ao tamanho da topologia, essa proporção difere entre a TCROS e a TCOS. Assim, a equação que indica o ponto onde os esforços se igualam não é a igualdade entre os tamanhos das topologias. Isto se deve ao "paralelismo" resultante da homogeneidade da TCOS, discutido anteriormente.

Quando o primeiro membro da equação passar a ser menor que o segundo membro, o uso da TCROS passará a ser preferível ao uso da TCOS, com relação ao esforço computacional despendido. Convém deixar claro aqui que o ponto exato em que essa transição ocorre depende também de características próprias da implementação e do ambiente em que o algoritmo é executado. Isto é expresso pela constante *'k'*, cujo cálculo depende da relação entre o esforço despendido para gerar uma combinação e procurar pelos exemplos que a satisfazem, contra o esforço de gerenciar o ajuste de pesos para todas as classes, com relação a uma determinada

combinação. Como esse gerenciamento é bastante simples, espera-se que essa constante assuma valores um pouco maiores, porém muito próximos de 1. Há ainda a possibilidade de termos constantes somadas a qualquer dos lados da equação. Isto tudo depende de particularidades da implementação e da arquitetura do ambiente em que a mesma é executada.

Voltando ao exemplo do Ginecol, se trabalharmos com aglomerados de tamanho máximo 5, teremos:

$$\sum_{i=1}^{nc} NCR(i,5) = NC(3,5) + \ldots + NC(39,5) + NC(44,5)$$

$$= 3.350.066 \qquad \text{(N1)}$$

$$NC(158,5) = 795.404.548 \qquad \text{(N2)}$$

Comparando o número N1, que representa a soma dos tamanhos das subredes da TCROS, criadas de acordo com a matriz de relevância eliciada do especialista, com o número N2, que representa o tamanho da subrede padrão da TCOS que seria treinada para todas as classes do domínio, concluimos que o uso da matriz de relevância e da TCROS apresenta resultados melhores com muito menos esforço computacional, pelo menos no caso do Ginecol.

Esta afirmação é corroborada pela figura 4.1, que mostra a razão entre o tamanho da TCROS e da TCOS para redes com ordens entre 1 e 7, referentes ao domínio desse estudo de caso. O tamanho considerado para a TCROS é calculado como a soma dos tamanhos das subredes referentes a todas as 18 classes do domínio. No entanto, o tamanho considerado para a TCOS é o tamanho de uma única subrede, a subrede padrão da TCOS. Sabemos que esta comparação favorece um pouco a TCOS mas, mesmo assim, de acordo com a figura, a diferença entre os tamanhos cresce à medida que a ordem aumenta.

ordem m	tam(TCOS)	tam(TCROS)	$\frac{\text{tam(TCROS)}}{\text{tam(TCOS)}}$
1	1,6 e+02	3,4 e+02	2,1 e+00
2	1,3 e+04	4,8 e+03	3,8 e-01
3	6,6 e+05	5,2 e+04	7,9 e-02
4	2,6 e+07	4,6 e+05	1,8 e-02
5	8,0 e+08	3,4 e+06	4,2 e-03
6	2,0 e+10	2,1 e+07	1,0 e-03
7	4,5 e+11	1,1 e+08	2,5 e-04

Figura 4.1 – Compara os tamanhos da TCROS e TCOS para redes com ordens entre 1 e 7, referentes ao domínio do estudo de caso em ginecologia. O tamanho da TCROS é calculado como a soma dos tamanhos das subredes referentes a todas as 18 classes do domínio. Já o tamanho considerado para a TCOS é o tamanho de sua subrede padrão.

Até aqui nos detivemos em um aspecto da contribuição proporcionada pelo uso da relevância semântica, que é a diminuição do tamanho da topologia a ser treinada. O segundo aspecto diz respeito à melhoria na qualidade dos resultados, através da eliminação de conceitos semanticamente irrelevantes. A importância da eliminação desses conceitos é apresentada por Cirne e Mongiovi [Cirne 91].

Generalizando os resultados

O resultado acima diz respeito a um estudo de caso. Mas será que em outras situações a vantagem da TCROS se apresenta sempre e com tanta clareza? A essa questão não cabe uma resposta absoluta. A vantagem da TCROS depende fundamentalmente de três pontos:

a) A variação de $NC(n,m)$ com relação à variação de $'n'$, que determina a diminuição do tamanho de cada subrede da TCROS com relação à diminuição da quantidade de evidências a combinar;
b) A proporção de evidências relevantes com relação ao total de evidências, que depende do grau de preenchimento das colunas da matriz de relevância;
c) A quantidade de classes do domínio, que determina o número de subredes da TCROS que devem ser somadas para a obtenção da topologia final.

Com relação a essas preocupações, podemos afirmar o seguinte:

a) Como pode ser visto na figura 2.7, pequenas variações no valor de *'n'* causam grandes variações no valor de *NC(n,m)*, principalmente nos casos em que *'n'* é maior. Isto significa que diminuições significativas no tamanho das subredes são obtidas a partir da identificação de uma pequena quantidade de evidências irrelevantes por classe. Devido a essas características, convém salientar que a diminuição no tamanho da topologia depende mais da classes com a maior quantidade de evidências relevantes que da média de evidências relevantes por classe.

b) A experiência tem mostrado que as matrizes de relevância tendem a ser esparsas. Quanto à disposição da matriz de relevância, temos observado a existência de dois tipos de atributos, que poderíamos chamar de classificadores e particulares. Os classificadores são relevantes para uma grande quantidade de classes, porém com valores diferentes para cada classe ou grupo de classes. Dessa forma, os valores acabam se distribuindo pelas colunas da matriz de relevância. Os atributos particulares são aqueles relevantes para poucas classes. Sua utilidade está em ajudar a qualificar uma classe ou distingui-las em seu grupo. Ambos os tipos de atributos contribuem para diminuir o grau de preenchimento da matriz de relevância.

c) O número de classes de cada domínio não é geralmente muito grande, tendendo a ser menor que dez. Devemos lembrar que, quanto maior esse número, maior deve ser a quantidade de exemplos oferecidos ao algoritmo para que ele possa identificar as características que determinam cada classe.

Assim, apesar de não podermos dar uma resposta absoluta em favor da TCROS, podemos sustentar que seu uso é vantajoso na esmagadora maioria dos casos.

4.4 - Conclusão

A topologia a ser usada no Modelo Neural Combinatório exerce grande influência sobre o desempenho do algoritmo de aprendizado e sobre a qualidade dos resultados. Neste capítulo, abordamos uma forma de melhorar o resultado do

aprendizado, não através de mudanças nos algoritmos de treinamento e poda, mas sim através de alterações na topologia originalmente gerada para ser treinada. Essas alterações decorrem do uso de um tipo de conhecimento preliminar: a relevância semântica. As topologias obtidas dessa forma foram comparadas entre si e com a original, levando-se em consideração as melhorias decorrentes na qualidade do aprendizado e no desempenho do algoritmo.

A Topologia Combinatorial Relevante de Ordem Superior (TCROS) proporciona uma melhoria no resultado gerado pelo algoritmo e mantém as vantagens originais do MNC, como: robustez (admissão de contra-exemplos, buracos e atributos multivalorados no conjunto de treinamento), conhecimento expresso numa forma inteligível e verificável diretamente por um especialista humano, e facilidade na modelagem do domínio.

O uso da matriz de relevância para podar os caminhos irrelevantes da Topologia Combinatorial de Ordem Superior apresenta dois grandes benefícios. O primeiro deles diz respeito à qualidade do conhecimento, na medida em que este é livrado de caminhos (asserções) sem importância, que equivaleriam às regras inúteis identificadas por Mongiovi e Cirne [Mongiovi 90]. O segundo benefício diz respeito à eficiência computacional do algoritmo, que é um fator importante, do qual depende a sua utilidade. A poda antecipada de caminhos irrelevantes pode diminuir o *overhead* na geração da topologia e a quantidade de espaço necessária para armazenar a rede em memória secundária.

Em comparação aos grafos de conhecimento, a matriz de relevância é, decididamente, uma forma mais segura e produtiva de "eliciar uma topologia inicial". Primeiro, a sua construção não requer a eliciação de informação que não seja usada. Segundo, ela permite ao especialista especificar de forma mais sucinta uma topologia maior. Isto ocorre porque o especialista não precisa relacionar todas as combinações interessantes para a topologia. Ele apenas indica quais as evidências são relevantes para cada classe. Cabe ao algoritmo gerar as combinações pertinentes àquelas evidências indicadas. Assim, a inclusão de algumas evidências corresponde à inclusão

implícita de todas as combinações (aglomerados) possíveis de serem formadas com aquelas evidências. Essas características tornam a descrição da topologia a ser treinada uma tarefa mais rápida e fácil, permitindo ainda ao especialista incluir na topologia a ser treinada todas as evidências que apresentam alguma relevância, mesmo que parcial.

5 - Usando Valoração de Atributos

5.1 - A modelagem do Domínio e a Valoração de Atributos

Os métodos de aprendizado automático realizam seu trabalho a partir de dados do mundo real, conhecidos como exemplos, casos ou instâncias. Na maioria das vezes, esses exemplos precisam sofrer um tratamento prévio antes de serem submetidos ao algoritmo de aprendizado. O tratamento dado aos dados brutos, com o objetivo de se obter exemplos utilizáveis, é conhecido como modelagem do domínio [Vasco 92b]. A modelagem se preocupa em estabelecer a estrutura ideal para os exemplos (quais os atributos e seus respectivos valores), bem como prover uma quantidade suficiente de exemplos tratados, i.e., que satisfazem às restrições sintáticas do algoritmo, são livres de males como ruídos e ambiguidades e, na medida do possível, representam de forma adequada os casos a partir dos quais se deseja adquirir conhecimento.

A modelagem do domínio deve ter em vista o algoritmo de aprendizado que será usado. Isso deve ser feito para que se chegue ao melhor resultado possível, captando o máximo da semântica do domínio e satisfazendo as restrições do algoritmo. Dentre as restrições comumente encontradas, está a não aceitação de contra-exemplos, atributos não valorados (buracos ou *missing values*) e atributos multivalorados (que assumem mais de um valor simultaneamente).

Durante a modelagem, alguns artifícios podem ser usados para adaptar os dados brutos do mundo real às restrições impostas pelo algoritmo. Por exemplo, um atributo multivalorado pode ser decomposto em dois ou mais atributos e seus valores distribuídos entre os mesmos, de forma que nenhum dos atributos criados admita mais de um valor simultaneamente. Vasco et al [Vasco 92] apresentam um ambiente de apoio à aquisição automática de conhecimento, que assiste o usuário na modelagem do domínio, na aplicação de um algoritmo de aprendizado e na validação do conhecimento adquirido. Para obter maiores detalhes sobre a modelagem do

domínio, inclusive sobre algumas técnicas de estruturação, captação e crítica de exemplos, consulte [Vasco 92b] e [Mongiovi 90b].

A distância entre o modelo e a realidade modelada exerce um papel fundamental no sucesso do aprendizado automático: quanto menor essa distância, melhor será considerado o modelo, porque representará o mundo com maior fidelidade. Para que essa distância seja a menor possível, é conveniente que a linguagem utilizada no modelo seja a mais próxima possível daquela utilizada no mundo real. Ora, quanto menos restrições o algoritmo impor à forma como o modelo deve ser representado, mais próximo do mundo real poderá estar o modelo produzido. Assim, apesar da existência de diversos artifícios para modelar o domínio de acordo com algumas exigências dos algoritmos, deve-se procurar por um algoritmo que imponha a menor quantidade possível de restrições à forma como o mundo real é modelado para a sua aplicação.

5.1.1 - A Modelagem para o Modelo Neural Combinatório

O Modelo Neural Combinatório e seu algoritmo de aprendizado formam uma dupla bastante robusta quanto aos dados de entrada. A baixa qualidade dos exemplos pode no máximo diminuir a qualidade do aprendizado, mas não pode causar confusões na execução do algoritmo, como, por exemplo, não conseguir terminar (alguns outros algoritmos podem apresentar comportamento estranho nesses casos). O MNC aceita conjuntos de treinamento com ruídos, contra-exemplos, atributos não valorados e multivalorados. Essa robustez o credencia para atuar em domínios não-determinísticos, facilita a modelagem e permite a representação e uso de exemplos mais próximos da realidade.

A aceitação de atributos multivalorados facilita a modelagem do domínio e permite que o modelo produzido seja mais fiel ao mundo real modelado. Essa flexibilidade apresentada permite liberar o usuário de recorrer ao uso de técnicas que transformam atributos multivalorados em atributos monovalorados. O uso dessas técnicas deve ser evitado, pois elas apresentam diversos efeitos colaterais, como: aumento da dificuldade para realizar a modelagem, distanciamento entre a realidade e

o modelo, e aumento na complexidade do modelo produzido (crescimento do número de atributos e valores). Este último ocorre porque, ao se distribuir os valores do atributo multivalorado original entre os atributos monovalorados dele derivados, tem-se que prever cada combinação válida de valores para o atributo original e representá-las como valores diferentes nos atributos derivados.

Assim, vemos que aceitar atributos multivalorados apresenta muitas vantagens, entre elas: a maior facilidade da modelagem e a obtenção de um modelo mais enxuto e mais fiel ao mundo real modelado. Isto indica que essa flexibilidade deve ser mantida em caso de se propor alterações ao MNC.

Contudo, o suporte nativo a atributos multivalorados acarreta, na maioria dos casos, um esforço computacional desnecessário. Isto ocorre porque a topologia usada pelo MNC prevê que todos os atributos sejam multivalorados. No entanto, a quantidade relativa de atributos multivalorados em um modelo usual é, na verdade, bem pequena. Assim, convém tentarmos diminuir o esforço computacional quando o algoritmo lida com atributos monovalorados. Para isso, devemos saber, em primeiro lugar, quais são esses atributos.

5.1.2 - A Valoração dos Atributos

A principal atividade da modelagem do domínio é identificar sua estrutura. Isso é feito através da identificação dos atributos necessários para representar adequadamente os exemplos daquele domínio. A identificação desses atributos consiste em descobrir seu significado e os valores que podem assumir.

Quanto aos valores que podem assumir, os atributos são divididos em nominais e ordinais. Os atributos nominais assumem valores que possuem significado individual, não havendo ordenação entre os mesmos. Os atributos ordinais assumem um valor de um conjunto de valores ordenados entre si.

A modelagem de atributos nominais deve identificar os valores que o atributo pode assumir, o significado de cada um, e verificar se dois ou mais desses valores podem ocorrer ao mesmo tempo. Se o atributo só puder assumir um único valor a

cada tempo, diz-se que ele é monovalorado. Caso possa assumir mais de um valor simultaneamente, ele é multivalorado.

O conhecimento sobre a valoração de atributos é eliciado durante a modelagem do domínio. A valoração de atributos refere-se à quantidade de valores que cada atributo pode assumir simultaneamente. Para referências futuras, consideraremos que esse conhecimento está armazenado na tabela de valoração de atributos.

A Natureza da Valoração de Atributos

No capítulo 3, examinamos alguns tipos de conhecimento preliminar. Entre os tipos examinados, não incluimos a Valoração de Atributos. O motivo desta omissão está na dificuldade de caracterizá-la como tal. A discussão sobre a natureza deste conhecimento deve contemplar dois aspectos: o conceitual e o de utilização.

Do ponto de vista conceitual, a concepção da Valoração de Atributos como um tipo de conhecimento preliminar é questionável. A Valoração de Atributos é um conhecimento acerca da estrutura do modelo do domínio; não a respeito do domínio em si. Por um lado, podemos argumentar que se trata de um conhecimento eliciável durante a modelagem do domínio e que está fora do conjunto de treinamento. Por outro lado, sabemos que está implicitamente contido no conjunto de treinamento, se não por completo ao menos na sua maior parte.

Do ponto de vista da utilização, vemos que a porção deste conhecimento que é derivável do conjunto de treinamento satisfaz plenamente os requisitos para a sua aplicação no MNC. Para explicar isto melhor, cabe antes o esclarecimento de quais são as partes deriváveis do conjunto de treinamento. Sabemos que a Valoração de Atributos diz respeito à estrutura do modelo idealizado para o domínio. Ora, o conjunto de treinamento é construído com base nessa estrutura: os exemplos são representados por meio de atributos e seus valores. Contudo, devido à incompletude inerente a todo conjunto de treinamento real (vide capítulo 4), nada garante que o mesmo utilize todos os atributos e valores levantados durante a modelagem. Menos certa ainda é a valoração de atributos dele derivada: um atributo multivalorado, i.e.,

que pode assumir mais de um valor no mesmo exemplo, pode ter ocorrências unitárias ou nem as ter, num conjunto de treinamento. Dessa forma, a Valoração de Atributos derivada do conjunto de treinamento pode ser incompleta quando comparada com aquela obtida externamente.

Contudo, devido à estreita relação entre os exemplos e a topologia usada, a incompletude da Valoração de Atributos derivada do conjunto de treinamento não é empecilho para seu uso no MNC. A construção da topologia com base na estrutura de domínio derivada do conjunto de treinamento traz dois benefícios: a garantia de que não há nenhum valor estranho na rede a ser treinada; e de que não há nenhum aglomerado com atributo de valoração superior àquela que ele possui no conjunto de treinamento. Este último ponto impede a geração de aglomerados com múltiplos valores de um atributo quando este só ocorre no conjunto de treinamento com, no máximo, um valor.

Do exposto, podemos concluir que, para o uso no MNC, a estrutura de domínio derivada diretamente do conjunto de treinamento acaba se tornando uma solução melhor. A razão para isto é a ocorrência da poda imediatamente após o treinamento. Isto torna desinteressante a existência de aglomerados para os quais se pode prever peso nulo como resultado do treinamento, pois sua presença apenas aumenta o custo computacional do aprendizado, sem oferecer qualquer benefício em contrapartida.

5.2 - A Topologia Construída com Ajuda da Valoração de Atributos

Como vimos anteriormente, muitas são as vantagens de se usar um algoritmo que permita o uso de atributos multivalorados. No MNC, essa facilidade é proporcionada pela forma como os exemplos são armazenados e pelo tipo da topologia treinada: a topologia definida no MNC contém aglomerados formados por todos os possíveis pares atributo-valor, inclusive dois ou mais pares formados por diferentes valores de um mesmo atributo, no mesmo aglomerado. A restrição imposta pela Topologia Combinatorial de Ordem Superior (TCOS) se refere ao tamanho dos

aglomerados, que não pode passar de um certo valor, determinado pela ordem da topologia.

Essa topologia presume que todos os atributos do domínio são multivalorados, podendo assumir diversos valores ao mesmo tempo. Contudo, uma ligeira análise de alguns domínios modelados nos mostra que, na realidade, a maioria dos atributos encontrados são monovalorados. Isso não diminui a necessidade do algoritmo trabalhar com atributos multivalorados, tratando-os como tal. Mas, sugere a tentativa de suprimir-se o esforço computacional desnecessariamente despendido para treinar uma topologia que trata todos os atributos como multivalorados, quando apenas alguns deles o são.

O tratamento diferenciado para atributos monovalorados pode ser feito através da utilização do conhecimento sobre a valoração de atributos, que permite eliminar da topologia os aglomerados formados por mais de uma ocorrência desses atributos. A exclusão precoce desses aglomerados, que de outra forma seriam expurgados da rede somente após o treinamento, aumenta a eficiência computacional do aprendizado sem alterar a sua qualidade. Esta poda antecipada, calçada na valoração de atributos, é implementada na definição da TCOSES.

A Topologia Combinatorial de Ordem Superior com Exclusividade Seletiva (TCOSES), de ordem m, é formada por todas as combinações possíveis de pares atributo-valor, com tamanho máximo igual a m, exceto aquelas em que atributo(s) monovalorado(s) aparece(m) com mais de um valor simultaneamente. Dessa forma, a TCOSES garante que os atributos monovalorados, identificados a partir da tabela de valoração de atributos, assumam valores mutuamente exclusivos. Chamaremos essa condição de exclusividade seletiva.

A exclusividade seletiva implementada pela TCOSES permite a convivência de atributos monovalorados e multivalorados em um mesmo domínio. Isso permite mantermos toda a flexibilidade original do Modelo e ainda conseguirmos uma diminuição do tamanho da topologia: para os atributos realmente monovalorados, a topologia não gera aglomerados com mais de um valor simultaneamente; para os

atributos multivalorados, não se faz necessária a aplicação de artifícios que terminam por aumentar a quantidade de atributos e evidências do modelo e distanciá-lo da realidade.

5.2.1 - Construção da TCOSES e seu tamanho

A TCOSES é construída a partir da combinação de evidências, que representam pares atributo-valor. Durante a formação dos aglomerados de evidências que compõem a topologia, as evidências relativas a atributos multivalorados são tratadas indistintamente, sem considerar a que atributo correspondem. Isso quer dizer que elas podem ocorrer em qualquer aglomerado, sem nenhuma preocupação com relação às demais evidências. Já as evidências relativas a atributos monovalorados não podem ser tratadas indistintamente: durante a construção das combinações, deve-se tomar cuidado para que duas delas, referentes a um mesmo atributo, não apareçam simultaneamente em um mesmo aglomerado.

Para analisarmos essas diferenças de tratamento, consideremos um domínio com três atributos – **a**, **b** e **c** –, que podem assumir qualquer um entre 2, 4 e 3 valores, respectivamente. Se todos os atributos forem multivalorados, teremos 9 evidências (soma de 2, 4 e 3) para combinar, formando aglomerados de tamanho 1, 2, 3 ... até a ordem desejada. Isso pode ser visto na figura 5.1-a.

No caso de um atributo ser monovalorado, a estratégia adotada sofre modificação. Como as evidências relativas a esse atributo não podem aparecer simultaneamente em um mesmo aglomerado, elas são substituídas por uma evidência fictícia, que pode ser considerada como uma macro ou um símbolo não-terminal. Durante a formação dos aglomerados, aqueles que possuírem a evidência fictícia são reproduzidos, substituindo-se, em cada reprodução, a evidência fictícia por evidências formadas por cada um dos valores que o atributo possa assumir. A figura 5.1-b mostra, para o caso do atributo **b** ser monovalorado, a rearrumação das evidências, a criação da evidência fictícia e a "explosão" de um "aglomerado fictício" originando 4 aglomerados reais.

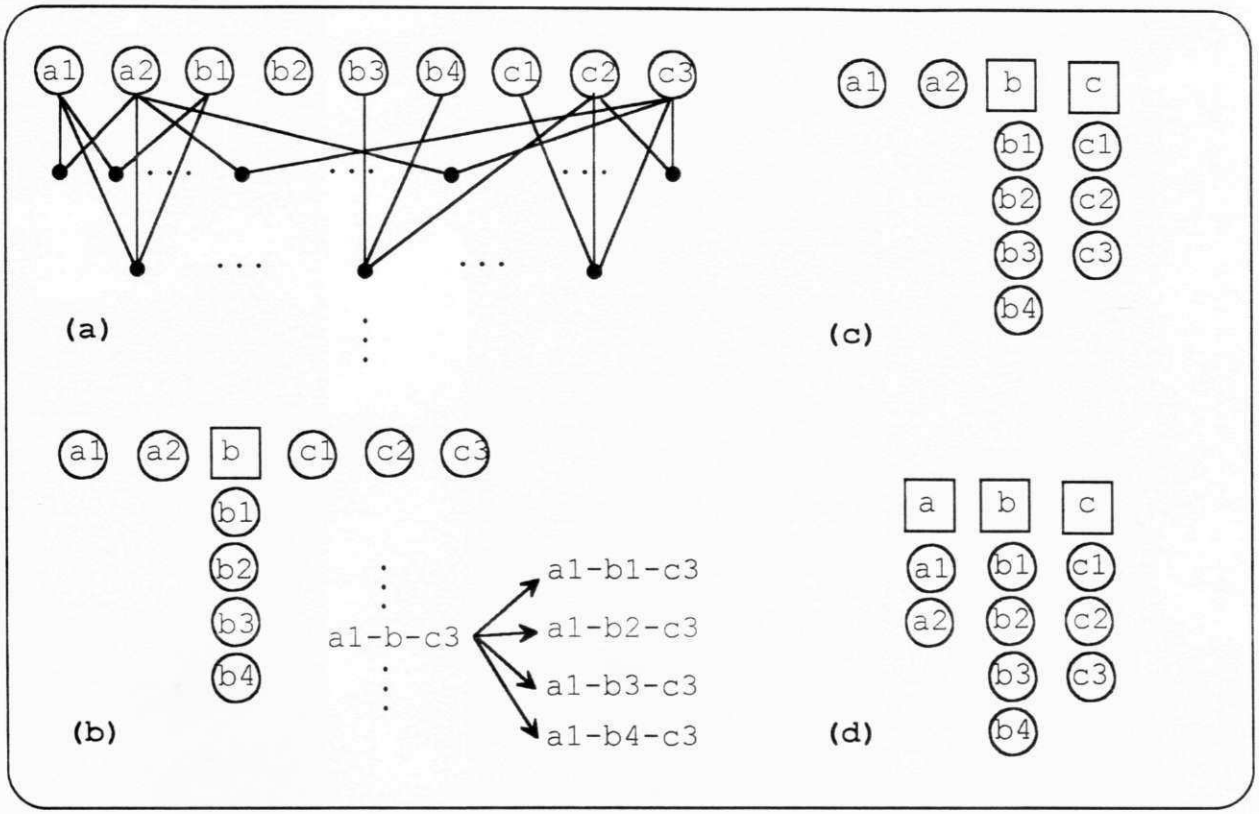


Figura 5.1 – Domínio com atributos **a**, **b** e **c**, com 2, 4 e 3 valores respectivamente. A formação dos aglomerados depende da valoração dos atributos. Em **(a)** todos os atributos são multivalorados; em **(b)** o atributo **b** é monovalorado; em **(c)** os atributos **b** e **c** são monovalorados; em **(d)** os atributos **a**, **b** e **c** são monovalorados.

No caso de mais atributos serem monovalorados, a técnica é semelhante: as evidências relativas a cada atributo monovalorado são substituídas por evidências fictícias que, quando presentes em um aglomerado, causam a sua reprodução até esgotarem todas as possibilidades de substituição das evidências fictícias com evidências formadas com os valores dos atributos que elas representam. A figura 5.1-c mostra como devem ser encaradas as evidências no caso do atributo **c** também ser monovalorado e a figura 5.1-d apresenta a situação onde os 3 atributos do domínio são monovalorados.

Em um domínio real, há uma mistura de atributos multivalorados e monovalorados. A expressão genérica que representa o tamanho da TCOSES, com atributos multivalorados e monovalorados, é a seguinte:

$$NCE(m) = \sum_{i=1}^{m} \sum_{j=1}^{Cn,i} \prod_{k=1}^{i} nf_{j,k}$$

onde:

- $nf_{j,k}$ = número de faces da **k**-ésima evidência da **j**-ésima combinação. Assume valor 1 para as evidências reais, formadas por atributos multivalorados. Já no caso de evidências fictícias, que representam atributos monovalorados, assume a quantidade de valores que o atributo pode assumir;
- n = número de evidências formadas por atributos multivalorados + número de atributos monovalorados (evidências fictícias);
- m = ordem da topologia.

Essa expressão pode ser entendida através do acompanhamento da maneira como os aglomerados são construídos. Porém, ela não se mostra de grande utilidade para uma análise do tamanho da topologia.

Vejamos uma outra expressão que, apesar de mais específica, "ataca o problema" de uma forma mais interessante. Considere um domínio em que cada atributo esteja relacionado a um conjunto de v valores possíveis. A expressão:

$$NCE(m) = \sum_{i=1}^{m} \sum_{j=0}^{i} (C_{v \cdot tu, i-j} \cdot C_{to,j} \cdot v^{j})$$

onde:

- tu = número de atributos multivalorados;
- to = número de atributos monovalorados;
- v = cardinalidade do conjunto de valores que cada atributo pode assumir;
- m = ordem da topologia;

representa o tamanho da TCOSES para esse domínio, considerando $C_{n,0} = 1$, p/ qualquer n e $C_{n,m} = 0$, p/ $n < m$. A construção dessa expressão baseia-se na idéia de que, entre os aglomerados de um certo tamanho, existem aqueles formados somente por atributos multivalorados, aqueles com um atributo monovalorado, aqueles com dois atributos monovalorados ... até aqueles formados completamente por atributos monovalorados.

5.2.2 - Avaliação da diminuição proporcionada pela Valoração de Atributos

A diminuição da topologia proporcionada pela valoração de atributos depende de diversos fatores. Entre eles, está a quantidade de valores que cada atributo pode assumir: quanto maior a quantidade de valores por atributo e a quantidade relativa de atributos monovalorados, maior será a diminuição do tamanho da TCOSES. Supondo um caso extremo, em que todas as evidências do domínio forem formadas por diversos valores de um só atributo monovalorado, a TCOS pode ser enorme, de acordo com a quantidade de evidências, enquanto que a TCOSES tem apenas aglomerados de uma célula na camada intermediária, em número igual à quantidade de evidências. Como veremos adiante, um outro parâmetro que também influencia na magnitude da diminuição relativa do tamanho da TCOSES é a ordem escolhida para a topologia.

As expressões mostradas na seção anterior não são de análise fácil, pois misturam fatoriais, exponenciais, somatórios e produtórios. Assim, estudaremos, a seguir, o comportamento do tamanho da topologia para domínios hipotéticos, mas de dimensões próximas do real.

As tabelas da figura 5.2 comparam o tamanho da TCOS com o tamanho da TCOSES para dois domínios diferentes. O primeiro deles possui 15 atributos e o segundo possui 30. Em ambos, todos os atributos envolvidos são monovalorados, podendo assumir um dentre 4 valores diferentes cada. As colunas (A) e (B) apresentam o tamanho da TCOS e da TCOSES, respectivamente, para cada domínio, considerando diversas ordens diferentes. A coluna (C) mostra a proporção entre os tamanhos da TCOSES e da TCOS. Nela, podemos ver que o tamanho relativo da TCOSES com relação à TCOS diminui à medida que a ordem m aumenta. Além disso, vemos que quanto mais complexo é o domínio, menor se torna a diminuição relativa proporcionada pela TCOSES, para uma mesma ordem. No caso de domínios mais complexos, somente para ordens maiores é que a relação atinge valores menores.

(a)	15 atributos de 4 valores			(b)	30 atributos de 4 valores		
Ordem	(A) TCOS	(B) TCOSES	$\frac{(B)}{(A)} * 100$	Ordem	(A) TCOS	(B) TCOSES	$\frac{(B)}{(A)} * 100$
5	6,0e+06	3,5e+06	57,7	5	2,0e+08	1,5e+08	76,9
6	5,6e+07	2,4e+07	42,9	7	6,3e+10	3,6e+10	56,7
7	4,4e+08	1,3e+08	29,2	9	1,1e+13	4,2e+12	36,6
8	3,0e+09	5,5e+08	18,4	11	1,3e+15	2,7e+14	20,5
9	1,8e+10	1,9e+09	10,4	13	9,9e+16	9,8e+15	9,8
11	4,4e+11	1,1e+10	2,5	15	5,5e+18	2,2e+17	3,9

Figura 5.2 – Mostra o tamanho da TCOS (coluna A) e da TCOSES (coluna B) para dois domínios diferentes. O domínio considerado em (a) possui 15 atributos de 4 valores cada; o domínio de (b) possui 30 atributos de 4 valores cada. Em ambos, os atributos são monovalorados. Em cada tabela, a última coluna apresenta a relação entre os tamanhos da TCOSES e da TCOS.

O resultado mostrado acima confirma a expectativa de que a valoração de atributos é um instrumento importante para diminuir o custo computacional envolvido no aprendizado, tanto com relação ao tempo quanto ao espaço.

A diferença relativa entre os tamanhos das duas topologias fica maior à medida que a ordem aumenta. Isto é bom, pois todo distanciamento entre o tamanho das topologias indica uma diminuição em potencial do esforço requerido para o aprendizado. Na figura 5.2-a, vemos que o tamanho da TCOSES de ordem 11 é 2,5 por cento do tamanho da TCOS de mesma ordem. Ou seja, nesse caso, a TCOSES é 40 vezes menor que a TCOS correspondente. A grosso modo, isto significa que o mesmo aprendizado pode ser feito até 40 vezes mais rápido.

Por outro lado, podemos ver que o distanciamento proporcional entre os tamanhos das topologias começa a ser mais significativo a partir de ordens grandes, tanto maiores quanto maiores forem os domínios (comparar 5.2-a e 5.2-b). Ao mesmo tempo que isto proporciona a poda de uma grande quantidade de aglomerados, observa-se que ela ocorre quando a rede já atinge tamanhos intratáveis. Para se ter uma idéia do que esses números significam na prática, tomemos os custos computacionais envolvidos em um caso real: o aprendizado usando uma topologia de

tamanho aproximadamente igual a $6,6*10^5$, por 81 exemplos, levou 124 s para ser concluído em uma Sun SPARCSTATION II sem carga. Isto permite acreditar que o aprendizado em um domínio como o da figura 5.2-b, usando-se ordem 7, também com 81 exemplos envolvidos, levaria aproximadamente:

$$\frac{6,3\cdot10^{10}}{6,6\cdot10^{5}}\cdot 124\,s \approx 118\cdot 10^{5}\,s \quad \text{ou} \quad \frac{3,6\cdot10^{10}}{6,6\cdot10^{5}}\cdot 124\,s \approx 68\cdot 10^{5}\,s$$

caso se usasse a TCOS ou a TCOSES, respectivamente; considerando todos os atributos como monovalorados. Estes valores representam cerca de 137 e 79 dias, respectivamente. Assim, apesar da TCOSES possiblitar a economia de 58 dias de processamento, consome ainda cerca de 79 dias, o que decididamente não é pouco. Na verdade, está muito longe do razoável.

A partir dos números expostos, concluimos que a valoração de atributos permite realizar uma boa poda na topologia original, mas não consegue resolver sozinha o problema da explosão combinatorial.

Devemos alertar aqui que a redução de tamanho proporcionada pela monovaloridade do atributo não compensa o aumento da complexidade necessária para forçá-la. Isto quer dizer que, se um atributo for originalmente multivalorado, não compensa transformá-lo "à força" em monovalorado, com o objetivo de obter uma diminuição na topologia. Se isto for feito, o resultado será um aumento no tamanho da topologia, além do distanciamento entre o modelo e a realidade modelada.

Mostraremos isso através de um exemplo: em um determinado domínio, existe o atributo *dor*, que é multivalorado e assume valores no conjunto {*cabeça*, *tórax*, *abdômem*, *membros*, *nenhuma*}. Se quisermos trabalhar somente com atributos monovalorados, devemos decompor o atributo *dor* em quatro outros: *dor_cabeça*, *dor_tórax*, *dor_abdomem* e *dor_membros*, cada um podendo assumir um valor no conjunto {*sim*, *não*}. Na situação original, temos 5 evidências; na situação modificada, existem 8 evidências. A rede TCOSES, de ordem 4, formada somente por este(s) atributo(s) e considerando a valoração dos mesmos, apresenta 30 nós na

situação original e 80 nós na situação modificada. Este exemplo serve para realçar a importância da seletividade ao considerar valores exclusivos.

5.3 - Uso de Relevância Semântica e Valoração de Atributos

Como vimos neste capítulo, o uso da valoração de atributos nos permite construir a TCOSES – uma topologia menor que a TCOS, mas ainda muito grande para problemas reais. No capítulo 4, discutimos a aplicação da relevância semântica que, através da alteração da topologia inicial, pode proporcionar um aprendizado de menor custo e maior qualidade [Donato 93].

Na realidade, como veremos adiante, o uso da relevância semântica costuma ser bem mais eficaz que o uso da valoração de atributos, i. e., a TCROS tende a ser bem menor que a TCOSES. Logo, o uso da TCOSES não é em geral justificável. A rápida comparação entre os benefícios das duas, entretanto, não objetiva a escolha desta ou daquela. Na verdade, a definição da TCOSES teve apenas o intuito de estudar o potencial da valoração de atributos isoladamente. O que nos interessa, neste trabalho, é combinar os benefícios de ambas as alterações estudadas.

Assim, introduzimos a Topologia Combinatorial Relevante de Ordem Superior com Exclusividade Seletiva (TCROSES), criada a partir da aplicação da relevância semântica e da valoração de atributos sobre a TCOS. O uso combinado dessas duas fontes de conhecimento adicional sobre a TCOS surte um efeito bem maior que aquele observado a partir de seu uso individual, tanto em relação ao custo computacional quanto em relação à qualidade do aprendizado. Isto será analisado a partir do estudo de um caso real, de dimensões razoáveis.

A observação de um estudo de caso diretamente, antes da apresentação de qualquer expressão analítica que represente o comportamento do modelo, deve-se, principalmente, à grande quantidade de variáveis envolvidas e à pouca utilidade das espressões analíticas, subproduto de sua razoável complexidade. Como já tivemos oportunidade de ver, o estudo da relevância semântica e da valoração de atributos,

isoladamente, já apresenta essas dificuldades. O estudo analítico de seu comportamento em conjunto apresenta dificuldades redobradas.

O estudo de caso utilizado para analisar comparativamente a TCROSES com as demais topologias será o Ginecol [Mongiovi 90b], já comentado neste trabalho. Trata-se de um conjunto de treinamento no domínio ginecológico, com 81 exemplos, 18 classes diagnósticas, 46 atributos e 158 evidências (pares atributo-valor), dando uma média de 3,4 valores por atributo. Esse conjunto de treinamento já foi previamente modelado para satisfazer às restrições do ID3, que é um algoritmo muito exigente quanto ao dados de entrada. Dessa forma, várias técnicas já foram usadas para transformar os atributos multivalorados em monovalorados e eliminar os buracos e contra-exemplos. Essas alterações no domínio não são necessárias para o MNC, sendo, inclusive, prejudiciais à sua execução. Por outro lado, a quantidade de exemplos é pequena em relação à quantidade de classes do domínio. Há muitas classes com um exemplo somente. Isto inviabiliza uma boa discriminação das mesmas. Contudo, a utilização desse conjunto de treinamento é aceitável por permitir uma comparação futura entre os resultados gerados pelo MNC, antes e após as modificações, com outros algoritmos de indução.

O conjunto de treinamento usado também inclui conhecimento sobre relevância semântica, eliciado de um especialista na forma de uma Matriz de Relevância. Quanto à valoração de atributos, sabe-se de antemão que todos eles são monovalorados.

Analisaremos o tamanho das topologias em duas etapas. A figura 5.3 mostra, em valores aproximados, o tamanho de cada topologia para o domínio em questão. Já a figura 5.4 mostra uma análise comparativa entre os tamanhos das topologias, com relação à topologia inicial. Antes de tirarmos conclusões a respeito dos números, devemos fazer alguns esclarecimentos sobre a natureza das topologias e sobre o significado desses números.

Ordem	TCOS	TCOSES	TCROS	TCROSES
1	1,6e+02	1,6e+02	7,4e+02	3,4e+02
2	1,3e+04	1,2e+04	4,8e+03	4,4e+03
3	6,6e+05	6,2e+05	5,2e+04	4,1e+04
4	2,6e+07	2,3e+07	4,6e+05	2,8e+05
5	8,0e+08	6,6e+08	3,4e+06	1,5e+06
6	2,0e+10	1,6e+10	2,1e+07	6,2e+06
7	4,5e+11	3,0e+11	1,1e+08	2,0e+07

Figura 5.3 – Tamanho das topologias para o Ginecol. O tamanho considerado para as topologias homogêneas (TCOS e TCOSES) é o tamanho de uma única subrede, chamada subrede padrão. Já para as topologias heterogêneas (TCROS e TCROSES), o tamanho considerado é a soma dos tamanhos das 18 subredes diferenciadas.

Já vimos anteriormente que as topologias podem ser homogêneas ou heterogêneas. Nas topologias homogêneas, as diferentes subredes são inicialmente iguais entre si e vão diferenciando-se durante o treinamento. Este é o caso da TCOS e da TCOSES. Nas topologias heterogêneas, as diversas subredes já são originalmente diferentes entre si. Este é o caso da TCROS e da TCROSES. Isto acontece porque a valoração de atributos trata das características dos atributos, independentemente das classes. Já a relevância semântica descreve a importância de cada par atributo-valor para a determinação de cada uma das classes separadamente.

Os números apresentados nas figuras 5.3 e 5.4 tentam expressar, com a maior fidelidade possível, o custo computacional do treinamento da rede referente à cada topologia. Assim, o tamanho considerado para as topologias homogêneas (TCOS e TCOSES) é o tamanho da subrede padrão, que é igual para todas as classes. Já o tamanho das topologias heterogêneas (TCROS e TCROSES) é calculado como a soma dos tamanhos das subredes referentes a cada uma das 18 classes do domínio.

Devemos chamar atenção sobre o fato de que o tamanho correto para as topologias homogêneas seria o produto do tamanho da subrede padrão pelo número de classes do domínio. A escolha tomada aqui deve-se basicamente a três razões, relacionadas a seguir:

1. O custo computacional de gerar as topologias e comparar cada aglomerado gerado contra os exemplos é bem mais significativo que o de ajustar os pesos do aglomerado em cada uma das subredes (classes);
2. O algoritmo de treinamento é otimizado de maneira a gerar a topologia homogênea somente uma vez;
3. Como a topologia final, que emprega as duas alterações propostas, é heterogênea e a topologia original é homogênea, preferimos privilegiar, na análise, as topologias homogêneas em detrimento das topologias heterogêneas.

Vê-se que as topologias que usam relevância (TCROS e TCROSES) apresentam valores menores que a TCOS e a TCOSES, com exceção dos valores referentes à ordem 1, que são maiores. Isto ocorre mesmo sob a condição desvantajosa descrita acima.

A figura 5.4 mostra o tamanho relativo entre cada topologia apresentada e a topologia original. As colunas (A) e (B) mostram a diminuição proporcional dos tamanhos das topologias causada pela aplicação da relevância semântica e da valoração de atributos, repectivamente, para o caso em estudo. A coluna (C) contém o produto entre os dois tamanhos relativos apresentados nas colunas anteriores. Este produto aparece com o objetivo de estimar a diminuição relativa provocada pela aplicação da relevância semântica e da valoração de atributos simultaneamente. Esta estimativa, que nos parece intuitivamente razoável, basea-se na aplicação sequencial das duas reduções e na independência entre elas.

A coluna (D) apresenta o tamanho da TCROSES em relação à TCOS. Deste número pode-se deduzir a diminuição relativa real provocada pela aplicação dos dois redutores estudados. A princípio, admitia-se que a diminuição real fosse menor que aquela calculada na coluna (C); ou seja, a TCROSES poderia ter um tamanho um pouco maior que o estimado. Contudo, o resultado foi completamente contrário: os valores obtidos na coluna (D) são menores que aqueles estimados na coluna (C). Além disso, a diferença entre eles se acentua à medida que a ordem aumenta. Ou seja, quanto maior a ordem, maior é a diferença entre o tamanho esperado e o tamanho real da TCROSES.

Ordem	(A) tam(TCROS) / tam(TCOS)	(B) tam(TCOSES) / tam(TCOS)	(C) (A) * (B)	(D) tam(TCROSES) / tam(TCOS)
1	2,1 e +00	1,0 e +00	2,1 e +00	2,1 e +00
2	3,8 e -01	9,8 e -01	3,7 e -01	3,5 e -01
3	7,9 e -02	9,5 e -01	7,5 e -02	6,2 e -02
4	1,8 e -02	9,0 e -01	1,6 e -02	1,1 e -02
5	4,2 e -03	8,3 e -01	3,5 e -03	1,9 e -03
6	1,0 e -03	7,6 e -01	7,8 e -04	3,0 e -04
7	2,5 e -04	6,8 e -01	1,7 e -04	4,5 e -05

Figura 5.4 – Tamanho relativo das topologias para o Ginecol. Os números indicam a razão entre o tamanho da topologia da coluna e o tamanho da TCOS. O tamanho considerado para as topologias homogêneas (TCOS e TCOSES) é o tamanho de uma única subrede, chamada subrede padrão. Já para as topologias heterogêneas (TCROS e TCROSES), o tamanho considerado é a soma dos tamanhos das 18 subredes diferenciadas

Vemos, dessa forma, que a união das alterações em uma só topologia provoca uma melhoria maior que aquela esperada a partir da aplicação isolada das alterações. Ou seja, existe uma sinergia entre as duas alterações propostas de mode que, quando aplicadas em conjunto, provocam um efeito bem maior do que seria esperado. Isto explica a validade de se estudar e aplicar a valoração de atributos, pois mesmo não sendo esta muito atrativa isoladamente, assume grande importância quando usada em conjunto com a relevância semântica.

A partir da figura 5.3, podemos ver o resultado prático de se usar as alterações aqui propostas. Tomando como referência o tempo necessário para gerar, treinar e podar uma rede de determinado tamanho com uma certa quantidade de exemplos, podemos estimar o tempo necessário para o uso de topologias diferentes, com a mesma quantidade de exemplos. Como vimos anteriormente, rodando em uma Sun SparcStation II, o algoritmo de aprendizado leva 124 segundos (2 min e 4 s) para submeter 81 exemplos a uma rede original (TCOS) de ordem 3. Consultando a figura 5.3, vemos que o tamanho da TCOS de ordem 3 é $6,6*10^5$. A partir destes dados, podemos estimar os seguintes tempos:

1. Usando a TCROSES, em menos da metade desse tempo (52,6 s), podemos gerar e treinar uma rede de ordem 4.
2. O tempo necessário para gerar e treinar uma rede original (TCOS) de ordem 4 (tamanho $2,6*10^7$) é de aproximadamente 4200 segundos (1 h e 10 min). Usando a TCROSES, podemos gerar e treinar uma rede de ordem 7 (tamanho $2,0*10^7$) em menos tempo (3800 s ou aprox. 1 h e 3 min).
3. Se fôssemos usar a TCOS com ordem 7 (tamanho $4,5*10^{11}$), o tempo necessário seria de $8,5*10^7$ s, aproximadamente 978,5 dias ou 2 anos e 8 meses. Este tempo é $2,2*10^4$ vezes maior que o tempo necessário para se usar a TCROSES de ordem 7 (tamanho $2,0*10^7$, tempo de 3800 s ou 1 h e 10 min).

5.4 - Conclusão

Sabemos que o aprendizado no MNC é baseado no treinamento de uma rede acíclica, a partir de um conjunto de exemplos. A topologia dessa rede exerce grande influência sobre o desempenho do algoritmo e sobre a qualidade do aprendizado. Apresentamos, neste capítulo, uma forma de melhorar o resultado do aprendizado através da alteração da topologia originalmente gerada para ser treinada.

A alteração aqui sugerida é baseada na quantidade de valores simultâneos que cada atributo pode assumir. Essa informação, batizada de valoração de atributos, pode ser obtida do próprio conjunto de treinamento. As topologias obtidas a partir da valoração de atributos e da relevância semântica, discutida no capítulo anterior, foram comparadas entre si e com a original, levando-se em consideração as mudanças necessárias na forma de modelar o domínio, e as melhorias decorrentes na qualidade do aprendizado e no desempenho do algoritmo.

A TCROS (Relevante) e a TCOSES (Exclusividade Seletiva) permitiram a confecção de estudos isolados sobre a contribuição dada por cada uma das alterações apresentadas. A TCROSES mostra que essas contribuições podem ser unidas em uma só topologia, proporcionando um resultado bem melhor que aquele esperado a partir do estudo de suas contribuições individuais.

6 - Estudo dos parâmetros que influenciam o tamanho das topologias

Nos capítulos anteriores, mostramos que o uso da valoração de atributos e da relevância semântica pode aumentar a qualidade do aprendizado e diminuir o tamanho da rede a ser treinada sem prejuízo das vantagens oferecidas pelo Modelo Neural Combinatório – tais como flexibilidade, robustez e proximidade entre o mundo real e o modelo produzido. Esta possibilidade foi mostrada através da análise conceitual de cada topologia individualmente e da análise numérica de seu tamanho. Como precisávamos justificar conceitualmente o uso dos redutores de topologia, principalmente no caso da relevância semântica, tivemos a preocupação de basear a análise em um caso real. Esta escolha teve sua importância, mas prejudicou a análise e previsão do comportamento do tamanho das topologias para casos genéricos. Mesmo no capítulo sobre a valoração de atributos, onde alguns domínios artificiais foram usados, o estudo do comportamento do tamanho das topologias apresentadas não foi feito de uma forma sistemática.

Neste capítulo, procuramos preencher esta lacuna através da análise dos parâmetros que influenciam o tamanho das topologias. Esta análise numérica é baseada no cálculo sistemático do tamanho de todas as topologias para um determinado domínio e pequenas variações do mesmo; visa a estudar o grau de influência de cada parâmetro sobre o tamanho das topologias, bem como identificar as condições que fazem variar este grau de influência.

O objetivo final da análise aqui realizada é conhecer melhor o comportamento do tamanho das topologias face à variação de certos parâmetros. A importância disto está em poder prever o comportamento do tamanho das topologias para diferentes situações, identificar qual a melhor topologia para um determinado caso e estimar a alteração deste quadro diante de variações em cada parâmetro. A principal preocupação na escolha da topologia é devida à relevância semântica: a diminuição da topologia resultante de sua aplicação deve compensar a diferenciação entre as

subredes, que torna a topologia heterogênea e obriga a geração e verificação de cada subrede separadamente.

6.1 - Identificação dos parâmetros

Para identificar os parâmetros relevantes a esta análise, devemos verificar todas as topologias, partindo da mais simples e avançando até alcançar a mais complexa delas.

Comecemos então pela Topologia Combinatorial Completa (TCC), comentada no capítulo 2. A TCC nunca foi efetivamente recomendada; apenas serviu de base para a definição de topologias mais complexas. Ela contém todas as combinações possíveis para todas as evidências do domínio. Seu tamanho é determinado por um único parâmetro: o *número de evidências* do domínio.

A primeira topologia realmente usada é a Topologia Combinatorial de Ordem Superior (TCOS), definida no capítulo 2. Sua definição introduz um novo parâmetro: a *ordem*.

A TCOS é a topologia originalmente usada pelo MNC. A partir dela, definimos três outras, através da aplicação de dois redutores, a princípio de forma individual e em seguida de forma conjunta: a valoração de atributos e a relevância semântica.

A Topologia Combinatorial de Ordem Superior e Exclusividade Seletiva (TCOSES) foi definida no capítulo 5 através da aplicação da valoração de atributos. A valoração de atributos introduz o conceito de atributos e determina quais deles podem assumir mais de um valor simultaneamente e quais não podem. Dois tipos de parâmetros são introduzidos: os que dizem respeito à estrutura do domínio e aqueles relacionados à valoração em si. Os parâmetros do primeiro tipo são: o *número de atributos* do domínio que, juntamente com o número de evidências, determina a quantidade média de valores por atributo; e a *distribuição do número de valores por atributo*, que determina a concentração de atributos com diferentes quantidades de valores. Os parâmetros do segundo tipo são: o *grau de valoração*, que determina a

quantidade de atributos monovalorados e multivalorados; e a *distribuição da valoração*, que determina como a monovaloração e a multivaloração se distribuem entre os atributos de diferentes quantidades de valores.

No capítulo 4, apresentamos a Topologia Combinatorial Relevante de Ordem Superior (TCROS), definida pela aplicação da relevância semântica sobre a TCOS. Os parâmetros introduzidos pela relevância semântica são: *grau de relevância*, que determina o grau de preenchimento da matriz de relevância; e a *distribuição da relevância*, determinada pela forma como as evidências relevantes são distribuídas entre as classes diferentes. A ação da relevância semântica difere entre as classes do domínio. Assim, a aplicação da relevância semântica causa uma diferenciação entre as subredes, fazendo com que a topologia se torne heterogênea. Esta heterogeneidade, já discutida no capítulo 4, traz à cena um outro parâmetro: o *número de classes*. Este parâmetro, que diz respeito à estrutura do domínio, já estava presente desde a TCC. Contudo, pelas razões apresentadas no capítulo 5, ele somente exerce influência significativa quando a topologia é heterogênea.

A Topologia Combinatorial Relevante de Ordem Superior e Exclusividade Seletiva (TCROSES) resulta da aplicação da valoração de atributos e da relevância semântica, simultaneamente. Portanto, seu tamanho depende de todos os parâmetros discutidos até aqui. Para simplificar futuras referências aos mesmos, reproduzimo-os a seguir:

1. Número de evidências;
2. Número de atributos;
3. Distribuição do número de valores por atributo;
4. Número de classes;
5. Ordem da rede;
6. Grau de valoração;
7. Distribuição da valoração;
8. Grau de relevância;
9. Distribuição da relevância.

Os parâmetros 1,2,3 e 4 referem-se à estrutura do domínio. Os parâmetros 6 e 7 dizem respeito à valoração de atributos e os parâmetros 8 e 9 são determinados pela relevância semântica.

6.2 - A metodologia do estudo

Como já foi colocado acima, o objetivo deste capítulo é apresentar uma análise numérica dos parâmetros que influenciam o tamanho das topologias. Esta análise visa a estudar o grau de influência de cada parâmetro sobre o tamanho das topologias, bem como identificar as condições que fazem variar este grau de influência. Essas condições são expressas pelos valores assumidos pelos demais parâmetros, e traduzem a influência ou interferência entre os mesmos.

Para ser possível analisar a influência individual de cada parâmetro, lançamos mão de um domínio artificial, que pode ser ajustado de modo a variar um único parâmetro de cada vez. Isto possibilita o estudo da influência de cada parâmetro isoladamente e facilita o estudo da influência (interferência) entre alguns deles. Este último pode ser realizado através da comparação das curvas obtidas pela variação de um mesmo parâmetro em circunstâncias diferentes, isto é, quando os demais parâmetros assumem um outro conjunto de valores.

Assim, a tônica de nosso estudo consiste em analisar as consequências da variação de um só parâmetro enquanto mantemos os demais em valores constantes. Para que isto tenha algum significado real, precisamos encontrar valores significativos para atribuir aos parâmetros fixos. Chamaremos essas constantes de **valor típico** de cada parâmetro. Quando voltarmos à especificação mais detalhada dos parâmetros, na próxima seção, definiremos um valor típico para cada.

A escolha do valor típico de cada parâmetro enseja algumas preocupações, além da necessidade óbvia de ser o mais coerente possível com a realidade. Essas preocupações estão ligadas ao controle do experimento como, por exemplo, possibilitar a manutenção do parâmetro em valor constante quando da alteração dos demais parâmetros ou mesmo facilitar a sua própria variação. Para atender ao

primeiro caso, principalmente para os parâmetros de relação ou distribuição de valores, devem ser evitados valores típicos extremos, privilegiando-se valores médios ou uniformes. Já para facilitar a variação do próprio parâmetro, a escolha de seu valor típico deve prever a existência de múltiplos e submúltiplos válidos, de acordo com as relações e distribuições "típicas" previamente estipuladas.

6.3 - Apresentação dos parâmetros

Nesta seção, os parâmetros relacionados anteriormente são apresentados de forma mais minunciosa. Para cada parâmetro são considerados, pelo menos: o tipo dos valores que assume, a faixa de valores válidos e razoáveis e o valor típico.

Número de evidências: assume valores numéricos inteiros, maiores que 1. A faixa de valores razoáveis está, grosseiramente, entre 30 e 200. Com o objetivo de facilitar a manipulação dos domínios experimentais e o cálculo do tamanho das topologias, consideraremos domínios pequenos, com 40 evidências. Este é, portanto, o valor típico deste parâmetro.

Número de atributos: assume valores numéricos inteiros. A faixa de valores razoáveis situa-se entre 5 e 50. Juntamente com o número de evidências, determina a quantidade média de valores por atributo. Consideraremos a média típica de 3,6 valores por atributo, o que nos dá uma quantidade de 11 atributos no domínio.

Distribuição do número de valores por atributo: indica a maneira como os valores estão distribuídos por entre os atributos. Dois domínios com a mesma relação entre o número de evidências e o número de atributos podem ter distribuições radicalmente diferentes, variando o desvio-padrão da população de valores por atributo. Neste estudo, consideramos três cenários de distribuição diferentes, descritos na figura 6.1. Os três cenários apresentam a mesma média de 3,6 valores por atributo. O cenário considerado típico será o segundo (B).

Cenários	Atributos											desvio-padrão populacional
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	
A	3	3	3	3	4	4	4	4	4	4	4	0,48
B	2	2	3	3	3	4	4	4	5	5	5	1,07
C	2	2	2	2	2	5	5	5	5	5	5	1,49

Figura 6.1 – Distribuição da quantidade de valores entre os atributos.

Número de classes: assume valores numéricos inteiros, maiores que 1. A faixa de valores razoáveis situa-se entre 2 e 15. Nosso domínio experimental possui 6 classes.

Ordem da rede: assume valores numéricos inteiros, sendo razoáveis aqueles entre 3 e 9. Escolhemos como valor típico para este estudo a ordem 6.

Grau de valoração: representa a quantidade relativa de atributos multivalorados. Assume valores numéricos reais, no intervalo entre 0 e 1. Consideraremos os cenários apresentados na figura 6.2; o caso típico será o cenário B, com 3 atributos multivalorados, que representam 27% do total de 11 atributos.

Cenários		Atributos e quantidade de valores que assumem										
		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
A	0 %											
B	27 %					u	u	u				
C	64 %			u	u	u	u	u	u	u		
D	100 %	u	u	u	u	u	u	u	u	u	u	u

Figura 6.2 – Cenários com diversos graus de valoração de atributos. A quantidade relativa de atributos multivalorados varia de 0% a 100% do total de atributos do domínio. Os atributos marcados com "u" são multivalorados e os demais são monovalorados.

Distribuição da valoração: reflete a forma como a multivaloração e a monovaloração estão distribuídas entre os diversos atributos do domínio, que possuem quantidades diferentes de valores. O estudo deste parâmetro mostra que não somente a quantidade de atributos monovalorados influencia no tamanho da topologia; como veremos, o perfil destes atributos também exerce influência

marcante. O domínio artificial de estudo trabalha com três cenários diferentes para este parâmetro, indicados na figura 6.3, sendo o cenário B considerado típico para efeito da variação de outros parâmetros.

Cenários	Atributos e quantidade de valores que assumem										
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
A	u	u	u								
B					u	u	u				
C									u	u	u

Figura 6.3 – Cenários para distribuição de valoração entre os atributos. Os atributos marcados com "u" são multivalorados e os demais são monovalorados. Variamos este parâmetro respeitando o valor típico estabelecido para o grau de valoração – 27% dos atributos são multivalorados. No cenário A, os atributos multivalorados são aqueles com a menor quantidade de valores. No cenário C, são multivalorados os atributos que possuem a maior quantidade de valores.

Grau de relevância: determinada pelo grau de preenchimento da matriz de relevância. Assume valores numéricos contínuos, entre 0 e 1. O valor deste parâmetro é muito importante para o tamanho da TCROS e TCROSES, sendo decisivo para a existência ou não de vantagem ao se usar a relevância semântica. Para o estudo ora desenvolvido, consideramos típico o valor de 25%.

Distribuição da relevância: reflete a forma como as evidências relevantes se distribuem entre as classes. É tão importante para o tamanho da TCROS e TCROSES como o parâmetro anterior. Se, por exemplo, todas as evidências forem relevantes para uma determinada classe, a subrede referente àquela iguala-se em tamanho à subrede padrão da TCOS e TCOSES. A figura 6.4 mostra os três cenários de distribuição de relevância considerados neste trabalho, que partem de uma distribuição uniforme (A) até uma distribuição concentrada (C). Consideramos típica a distribuição dispersa (B).

Cenários	Classes					
	C_1	C_2	C_3	C_4	C_5	C_6
A	10	10	10	10	10	10
B	20	16	10	5	5	4
C	35	20	5	0	0	0

Figura 6.4 – Distribuição da relevância semântica entre as classes.

6.4 - Os resultados do estudo

Uma análise numérica exaustiva de todos os parâmetros identificados constitui-se numa tarefa de grande complexidade. Com o intuito de controlar esta complexidade, estabelecemos uma restrição à variação dos parâmetros: variamos apenas um deles de cada vez. Podemos abrir mão parcialmente desta restrição em alguns casos, quando variamos um parâmetro para circuntâncias diferentes, isto é, quando os demais parâmetros assumem conjuntos de valores diferentes.

Baseados no objetivo deste estudo, privilegiaremos aqui a apresentação dos resultados referentes aos parâmetros que exercem maior influência no tamanho das topologias; a ordem em que são apresentados, contudo, não é importante. A apresentação dos estudos relativos aos parâmetros escolhidos será sucinta, baseada nos gráficos que mostram o tamanho das topologias para diferentes valores do parâmetro. As tabelas com os valores mostrados nos gráficos estão no apêndice C. Lembramos que em cada caso apenas um parâmetro é alterado; os demais assumem seus respectivos valores típicos.

Quanto à ordem da rede:

A figura 6.5 mostra o gráfico para a variação do parâmetro *ordem da rede*. Os tamanhos das quatro topologias apresentam comportamento exponencial, sendo que as topologias definidas neste trabalho apresentam curvas mais suaves.

Uma observação importante pode ser feita com relação ao comportamento da TCROSES: além de apresentar um comportamento mais suave, ela apresenta uma

tendência à estabilização, a partir da ordem 9. Isto pode ser facilmente explicado pelos valores típicos adotados para os parâmetros *grau de relevância* e *distribuição da relevância*, que atribuem às duas classes "mais relevantes" – i. e., com maior quantidade de evidências relevantes – respectivamente 20 e 16 evidências relevantes. Essas classes influem decisivamente no tamanho da TCROSES, que é calculado pela soma do tamanho de suas subredes. Cada subrede, por sua vez, tem o tamanho calculado como uma soma de parcelas $C_{n,i}$, sendo n o número de evidências relevantes e i um índice variando de 1 até a ordem m, como vimos nos capítulos anteriores.

Pode ser demonstrado que essas parcelas crescem enquanto $i < n/2$ (para maiores detalhes, consulte bibliografia sobre o Triângulo de Pascal e o Binômio de Newton). A partir de n/2, as parcelas $C_{n,i}$ começam a decrescer. Assim, à medida que a ordem se aproxima, pela esquerda, de n/2, o crescimento do tamanho da topologia tende a diminuir. A ordem 9 ultrapassa a metade do número de evidências relevantes da segunda classe mais relevante e se aproxima da metade do número de evidências relevantes da classe mais relevante. Esta é a explicação para o comportamento da TCROSES na figura 6.5.

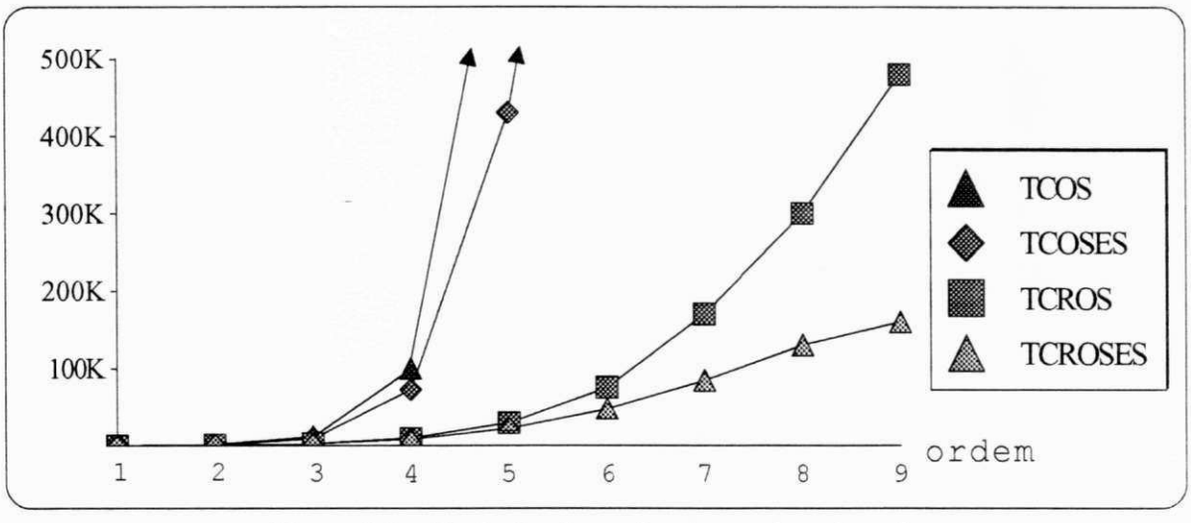


Figura 6.5 – Variação do parâmetro *ordem da rede*.

Quanto à distribuição do número de valores por atributo:

As figuras 6.6-A e 6.6-B mostram o tamanho das topologias para diferentes valores do parâmetro *distribuição do número de valores por atributo*. Vê-se que o parâmetro não exerce grande influência no tamanho das topologias, visto que eles variam pouco para os diferentes cenários. Na figura 6.6-A, vemos que a TCOSES cresce um pouco quando se aumenta o desvio-padrão da população de valores por atributo (veja figura 6.1), enquanto que a TCOS se mantém com tamanho constante.

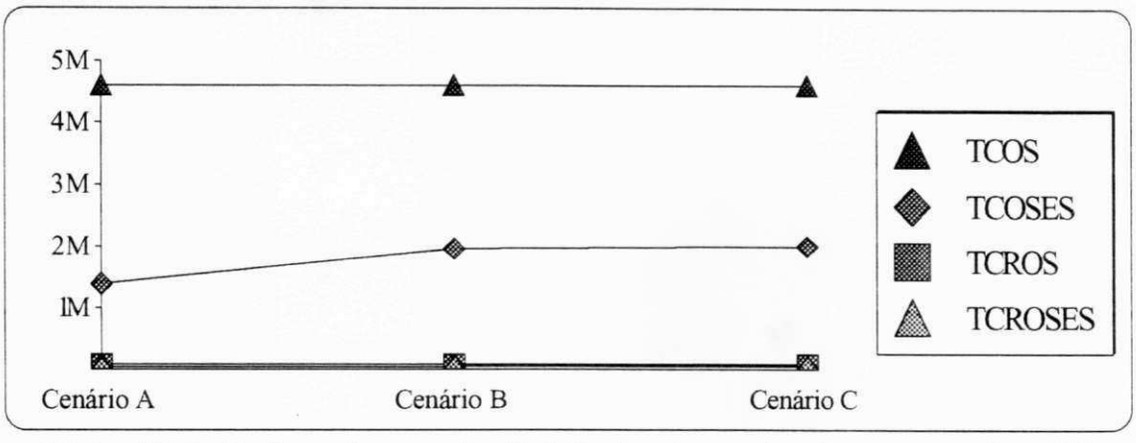


Figura 6.6-A – Variação do parâmetro *distribuição do número de valores por atributo*. Mostra os tamanhos entre 0 e $5*10^6$, usando uma escala em que a unidade representa $1*10^6$ aglomerados.

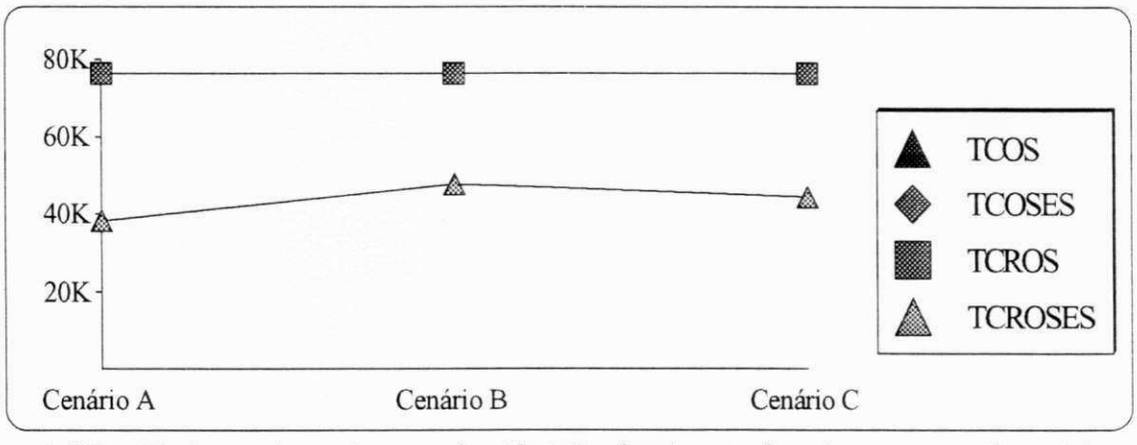


Figura 6.6-B – Variação do parâmetro *distribuição do número de valores por atributo*. Mostra os tamanhos entre 0 e $80*10^3$, usando uma escala em que a unidade representa $20*10^3$ aglomerados.

Como as topologias com relevância semântica apresentam tamanhos muito menores que as outras, com uma distância de duas ordens de grandeza, a figura 6.6-A não permite analisar seu comportamento. A figura 6.6-B mostra o tamanho

dessas duas topologias em uma escala apropriada, o que nos permite detectar uma variação também no tamanho da TCROSES. Enquanto as demais topologias mantêm-se constantes ou apenas crescem, a TCROSES cresce entre os cenários A e B e decresce entre este e o cenário C. É óbvio que atributos com maior quantidade de valores contribuem para o aumento da topologia e atributos com menor quantidade para seu decréscimo. Contudo, o tamanho da TCROSES não é função exclusiva da variância da população de valores por atributo, como se poderia supor.

Das duas figuras, podemos ver que o parâmetro não influencia o tamanho das topologias definidas a partir de evidências sem considerar a existência de atributos. Este é um resultado já esperado. Quanto às topologias que consideram a existência de atributos, e têm seu tamanho influenciado pela valoração de atributos, podemos detectar uma variação de tamanho inferior a 30%, o que dá uma idéia da importância deste parâmetro.

Quanto ao grau de valoração de atributos

As figuras 6.7-A e 6.7-B mostram o comportamento do tamanho das topologias diante da variação do parâmetro *grau de valoração*. No cenário A, todos os atributos são monovalorados. No cenário D, todos são multivalorados (veja figura 6.2). As figuras 6.7-A e 6.7-B mostram que as topologias que implementam a exclusividade seletiva apresentam tamanho bem inferior (menos da metade) ao das demais para o cenário de atributos monovalorados (cenário A), sendo que esta diferença diminui à medida que surgem atributos multivalorados no domínio e se anula ao alcançar o cenário onde todos os atributos são multivalorados (cenário B).

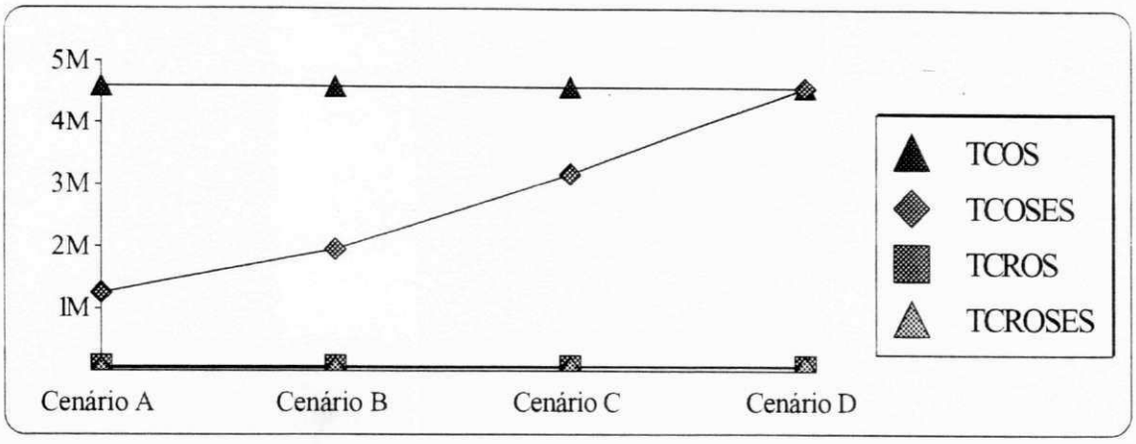


Figura 6.7-A – Variação do parâmetro *grau de valoração dos atributos*. Mostra os tamanhos entre 0 e $5*10^6$, usando uma escala em que a unidade representa $1*10^6$ aglomerados.

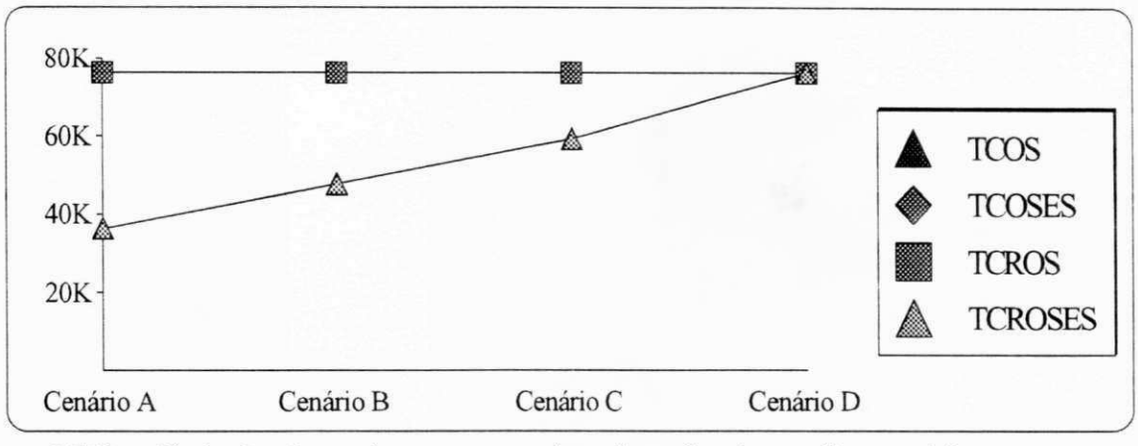


Figura 6.7-B – Variação do parâmetro *grau de valoração dos atributos*. Mostra os tamanhos entre 0 e $80*10^3$, usando uma escala em que a unidade representa $20*10^3$ aglomerados.

Quanto à distribuição da valoração de atributos:

As figuras 6.8-A e 6.8-B mostram o tamanho das topologias para diferentes valores do parâmetro *distribuição da valoração de atributos*. Os três cenários, descritos na figura 6.3, elegem atributos diferentes para serem multivalorados. No cenário A, os atributos multivalorados são aqueles com a menor quantidade de valores. O cenário C simula o extremo oposto: os atributos multivalorados são aqueles com a maior quantidade de valores.

Na figura 6.8-A, vemos que, para uma mesma quantidade de atributos multivalorados, quanto maior a quantidade de valores dos atributos multivalorados, maior é o tamanho da TCOSES. Já a TCOS não tem seu tamanho alterado.

Como o tamanho da TCROS e TCROSES é menor que o das demais em duas ordens de grandeza, recorremos a um outro gráfico para perceber o seu comportamento. A figura 6.8-B mostra esse outro gráfico. Nele, podemos ver que o parâmetro em estudo não influi no tamanho da TCROS; apenas altera o tamanho da TCROSES da mesma forma que o faz com a TCOSES.

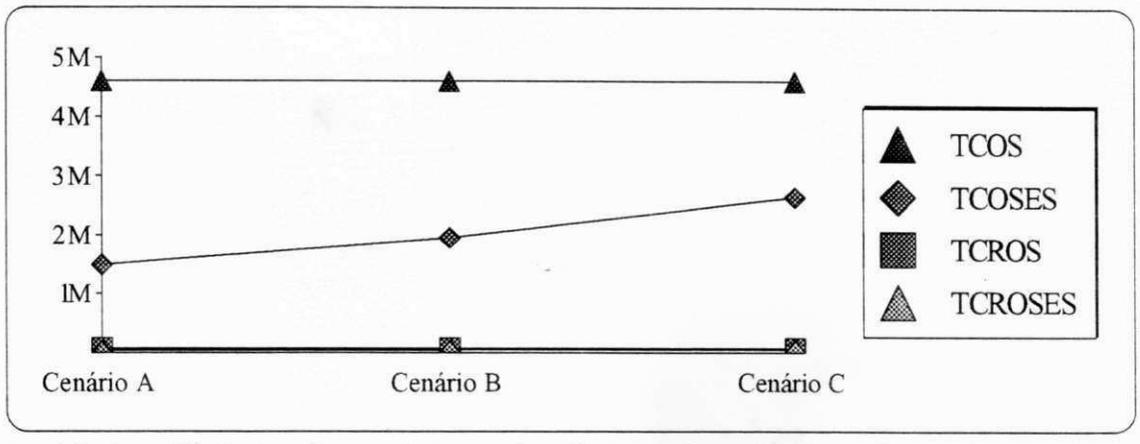


Figura 6.8-A – Variação do parâmetro *distribuição da valoração de atributos*. Mostra os tamanhos entre 0 e $5*10^6$, usando uma escala em que a unidade representa $1*10^6$ aglomerados.

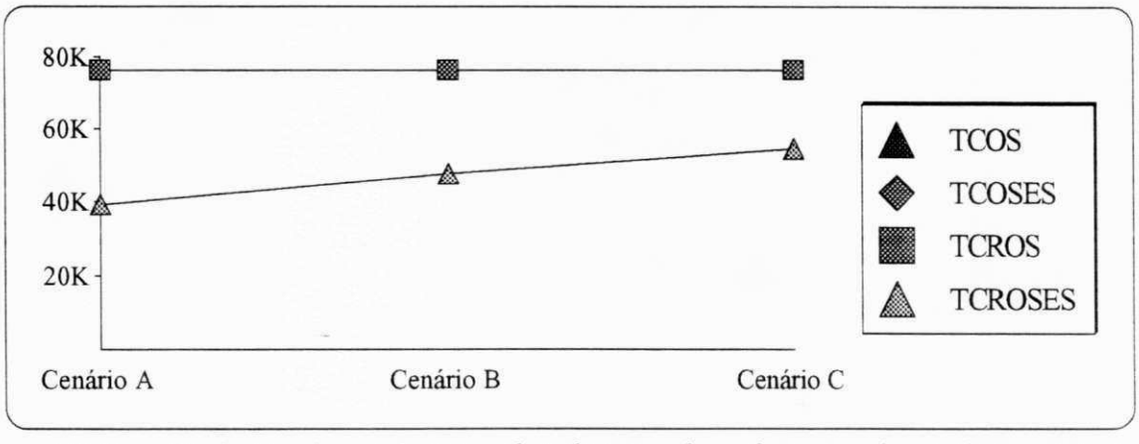


Figura 6.8-B – Variação do parâmetro *distribuição da valoração de atributos*. Mostra os tamanhos entre 0 e $80*10^3$, usando uma escala em que a unidade representa $20*10^3$ aglomerados.

Quanto à distribuição da relevância entre as classes:

Como já sabemos, a relevância semântica provoca a diferenciação das subredes da topologia porque as evidências relevantes são diferentes para cada classe. A figura 6.9 mostra o comportamento do tamanho das topologias em função da *distribuição da relevância* entre as classes. A figura 6.4 mostra a definição dos três cenários considerados. No cenário A, a distribuição é homogênea (as evidências relevantes são distribuídas igualmente entre as classes). No cenário B a distribuição já é heterogênea, mas ainda dispersa. No cenário C, a evidências relevantes estão concentradas em poucas classes.

De volta à figura 6.9, podemos ver que as topologias que não consideram a relevância semântica não possuem seu tamanho afetado pela variação deste parâmetro. Já a TCROS e TCROSES apresentam uma expressiva variação de tamanho, sendo que a primeira mantém-se sempre maior que a segunda. Podemos considerar o parâmetro *distribuição da relevância* de grande importância para o tamanho das topologias que consideram a relevância semântica. Sua influência é tal que, sozinho, pode determinar a vantagem de se usar ou não uma topologia baseada na relevância semântica, como a TCROS ou TCROSES.

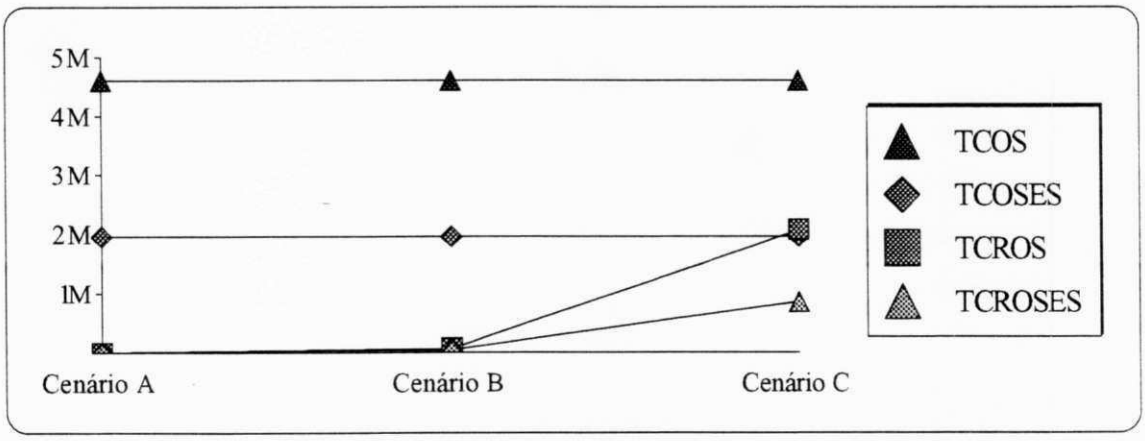


Figura 6.9 – Variando a distribuição da relevância entre as classes, mantendo o grau de relevância no valor típico de 25%.

Quanto ao grau de relevância:

O grau de relevância das evidências do domínio é determinado pelo grau de preenchimento da matriz de relevância. Nas figuras 6.10-A e 6.10-B, apresentamos o tamanho das topologias para diferentes valores deste parâmetro, considerando duas circunstâncias que diferem no valor congelado para o parâmetro *distribuição da relevância*.

Na figura 6.10-A, a distribuição de relevância adotada é a **uniforme**, o que significa que as evidências relevantes estão distribuídas igualmente por todas as classes do domínio. O parâmetro *grau de relevância* varia de 15% a 85%, em incrementos de 10%. Já na figura 6.10-B, a distribuição de relevância adotada é a **dispersa**, definida como típica na figura 6.4. O parâmetro *grau de relevância* varia de 20% a 50%, com incrementos de 5%. A razão para o menor intervalo de variação do parâmetro neste último caso está na estrutura do domínio do experimento, que não permite maior liberdade de variação do *grau de relevância* quando adotamos a distribuição de relevância dispersa.

O comportamento dos gráficos das figuras 6.10-A e 6.10-B mostra a grande importância do *grau de relevância* para o tamanho das topologias que usam a relevância semântica. A variação deste parâmetro provoca grandes reflexos nos tamanhos da TCROS e TCROSES, fazendo-os comportarem-se como curvas exponenciais, que partem de valores muito inferiores aos das demais topologias, chegando a igualá-los e ultrapassá-los facilmente. Dessa forma, o valor assumido por este parâmetro assume grande importância na escolha da topologia a ser usada.

Como vimos, os gráficos das figuras 6.10-A e 6.10-B indicam que o valor assumido pela parâmetro *grau de relevância* pode até invalidar o uso das topologias TCROS e TCROSES, dependendo do valor que assuma. Contudo, claro também está que, se este parâmetro se mantiver próximo ao seu valor típico, o uso da TCROS ou TCROSES será vantajoso, posto que, neste caso, essas topologias apresentarão tamanho bem inferior ao das demais. Como exemplo, vamos comparar o tamanho relativo entre a TCROSES e a TCOS (tamanho da TCROSES dividido pelo tamanho

da TCOS) nas duas circunstâncias apresentadas: para a distribuição uniforme, a variação do parâmetro entre 15% e 35% provoca a variação do tamanho relativo de $8{,}2*10^{-5}$ a $6{,}1*10^{-3}$; para a distribuição dispersa, a variação do parâmetro entre 20% e 35% provoca variação de $3{,}0*10^{-3}$ a $6{,}7*10^{-2}$ no tamanho relativo mencionado. De fato, no experimento usado, o tamanho da TCROSES só se iguala ao tamanho da TCOS quando o *grau de relevância* atinge valores próximos de 85% ou 55%, para distribuição de relevância uniforme ou dispersa, respectivamente.

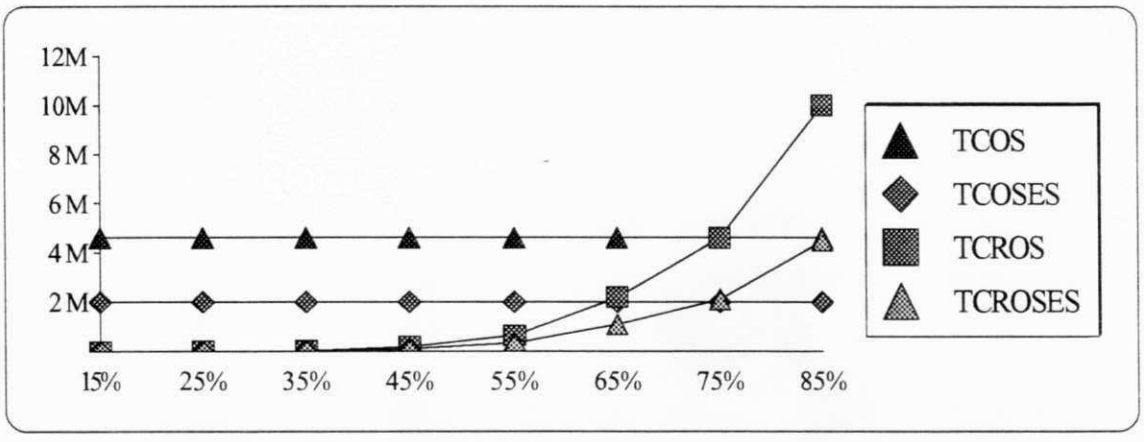


Figura 6.10-A – Variação do grau de relevância, mantendo a distribuição uniforme (A).

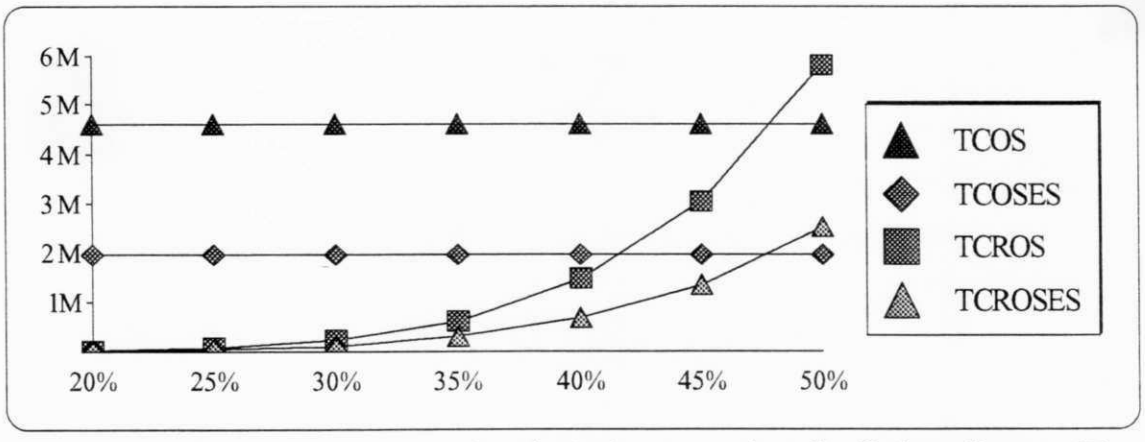


Figura 6.10-B – Variação do grau de relevância, mantendo a distribuição dispersa (B).

A partir do que foi exposto, podemos concluir que, quando o parâmetro *grau de relevância* situa-se próximo ao seu valor típico, a TCROSES assume tamanho muito inferior ao tamanho das demais topologias. Assim, esperamos que, na maioria dos casos, a TCROSES proporcione um aprendizado de melhor qualidade com menor esforço computacional.

Contudo, apesar da importância do *grau de relevância* para o tamanho das topologias, a escolha pelo uso da relevância semântica não pode se basear somente neste parâmetro. Comparando as figuras, percebemos a influência do parâmetro *distribuição de relevância*, que proporciona grande variação no tamanho da TCROS e TCROSES para o mesmo valor do parâmetro *grau de relevância*. Isto se torna nítido ao se confrontar os graus de relevância vigentes quando o tamanho da TCROS e TCROSES atinge o tamanho da TCOS e TCOSES, nas figuras 6.10-A e 6.10-B.

Assim, vemos que a decisão sobre usar ou não a relevância semântica depende decisivamente de pelo menos dois parâmetros em conjunto: o grau e a distribuição da relevância. Vemos também que, quanto mais uniforme for a distribuição da relevância e menor o grau de relevância, mais propício é o uso das topologias que consideram a relevância semântica na sua construção.

6.5 - Conclusão

Neste capítulo, vimos que existem vários parâmetros que influenciam o tamanho das diversas topologias apresentadas neste trabalho. Os parâmetros identificados retratam diferentes aspectos do experimento, desde a estrutura do domínio até características do conhecimento preliminar eliciado. Dentre os parâmetros apresentados, há aqueles que influenciam o tamanho de todas as topologias e aqueles que influenciam apenas o tamanho de algumas, dependendo das alterações implementadas pelas mesmas.

É importante ressaltar aqui que esses parâmetros são funções do domínio e não o contrário. Assim, devemos esquecer a tentação de manipulá-los com o único objetivo de diminuir o tamanho da topologia da rede a ser usada. O estudo desses parâmetros deve servir a dois objetivos básicos: orientar as decisões tomadas durante a modelagem do domínio; e estimar a topologia mais indicada para um determinado domínio, depois de modelado.

Para que o segundo objetivo seja alcançado, outros estudos precisam ser realizados. Estimar a topologia mais indicada para um determinado caso com base

somente nesses parâmetros não é uma tarefa trivial. Para que isso seja feito com exatidão, precisamos estabelecer limites precisos para o valor dos parâmetros, de forma a indicar a topologia correta. A grande interferência entre os parâmetros faz com que esses limites não possam ser absolutos. Assim, vemos que essa estimativa se constitui um trabalho complexo. Dessa forma, a sua automatização só poderia ser plenamente realizada através da construção de um sistema baseado em conhecimento, a partir de estudos exaustivos para diversos tipos de domínio.

7 - Considerações sobre a implementação

As atividades de desenvolvimento de programas e análise de seus resultados foram parte significativa deste trabalho. Não somente os programas que implementam a geração, o treinamento e a poda da rede foram desenvolvidos, mas também diversos outros, com objetivos bem variados, como: filtros para tradução do conjunto de treinamento e conhecimento preliminar dos formatos disponíveis para o formato usado por nossa implementação do MNC; cálculo do tamanho das diversas topologias aqui apresentadas, considerando domínios reais ou hipotéticos, variando a ordem ou a quantidade de evidências do domínio, para possibilitar a análise do comportamento do tamanho dessas topologias sob diversas condições; automatização da geração das topologias para diversos cenários para possibilitar o estudo da influência de todos os parâmetros através de domínios experimentais; e máquina de inferência classificadora para verificar o comportamento da rede gerada pelo aprendizado na classificação de casos de teste. À exceção de um filtro, que foi escrito em Prolog, todos os demais programas estão escritos em C.

O processo de desenvolvimento desses programas levou em consideração princípios como: inteligibilidade, expandibilidade, flexibilidade e eficiência.

Neste capítulo, apresentaremos a estrutura geral do protótipo implementado, que denominaremos APR-X, enfocando apenas os detalhes mais relevantes. Mostraremos também algumas alterações básicas sobre o algoritmo original, decididas no início do projeto e implementação do APR-X.

7.1 - A evolução dos algoritmos

Durante o desenvolvimento deste trabalho, o Modelo Neural Combinatório continuou sofrendo modificações propostas por seus próprios autores. Algumas dessas modificações podem ser vistas nos apêndices A e B, que mostram versões diferentes dos algoritmos de treinamento e poda, respectivamente, encontradas em trabalhos publicados pelos autores do Modelo.

Esta contemporaneidade entre este trabalho e aqueles desenvolvidos pelos autores garante um valor científico maior ao nosso esforço, mas também aumenta a sua complexidade. Como este trabalho começou baseado nas primeiras versões dos algoritmos, tivemos algum trabalho extra para identificar e corrigir problemas que acabaram sendo sanados em versões posteriores. Como resutado disto temos que, apesar de identificarmos detalhes em comum, nossa implementação não se encaixa exatamente em nenhuma das versões produzidas pelos autores do MNC. Mas, com certeza, adequa-se perfeitamente à filosofia básica que perpassa todas as versões.

7.1.1 - Análise das alterações propostas ao algoritmo de treinamento

Apresentaremos a seguir duas alterações propostas à versão de arranque do algoritmo de treinamento da rede. A apresentação de cada alteração proposta seguirá a seguinte ordem: primeiro a situação original é apresentada, com enfoque nítido sobre os problemas que as alterações pretendem resolver ou amenizar. Em seguida, a alteração é proposta. Finalmente, alguns comentários sobre a nova situação são colocados, bem como uma análise comparativa entre a situação antes e depois da alteração.

Alteração 1: Mudança dos laços.

A análise do espaço em memória que EXEMPLOS e REDE TREINADA da figura 2.6 podem ocupar leva à conclusão de que a segunda deve ser armazenada em disco e o primeiro deve ficar na memória principal, para casos típicos. O armazenamento da REDE TREINADA em disco é necessário devido ao grande crescimento da mesma (explosão combinatória). Além disso, isto facilita a separação entre os algoritmos de treinamento e poda, bem como possibilita desenvolver-se testes com limiares de poda diferentes, sem que seja necessário retreinar a rede. Finalmente, isto abre a possibilidade de implementarmos uma versão incremental do algoritmo de treinamento, proporcionando assim a possibilidade de se realizar refinamentos sucessivos à REDE TREINADA.

O algoritmo de treinamento apresenta dois laços aninhados (ver apêndice A1): "p/ cada exemplo da base de treinamento, faça ..." e "p/ cada arco que alcança um nó

hipótese, faça ...". Ou seja, cada exemplo é propagado por toda a rede, comparando suas evidências com todos os aglomerados da rede e ajustando os pesos daqueles satisfeitos pelo exemplo. Assim, cada exemplo pode provocar mudanças em vários aglomerados espalhados pela rede. Isto provoca uma grande quantidade de acessos ao disco, onde a REDE TREINADA está armazenada. Este número de acessos é igual a:

$$n\_ex * 2 * NC(n\_evid, m) * nc$$

onde:

- n_ex é o número de exemplos do conjunto de treinamento;
- n_evid é o número de evidências ou pares atributo-valor do domínio;
- nc é o número de classes do dominio;
- m é a ordem da rede;
- 2 representa um acesso para leitura e outro para gravação;
- NC(n_evid, m) é o número de células da rede.

A alteração aqui proposta consiste em mudar a ordem de aninhamento dos laços para: "p/ cada arco que alcança um nó hipótese, faça ..." seguido de "p/ cada exemplo da base de treinamento, faça ...". Assim, em vez de se verificar a influência de cada exemplo separadamente para todos os aglomerados, verifica-se a influência de todos os exemplos sobre cada aglomerado. Ou seja, o treinamento não é mais feito sobre a rede toda, simultaneamente, mas sobre cada aglomerado da rede, sequencialmente. Dessa forma, pode-se construir a REDE TREINADA, um aglomerado por vez, com valores já definitivos, pelo menos em se tratando do conjunto de treinamento atual. Isto diminui a quantidade de acessos para:

$$NC(n\_evid, m) * nc$$

visto que a rede é construída pelo cálculo completo de uma combinação de cada vez. Essa mudança não altera o espaço ocupado pela REDE TREINADA nem a quantidade de operações envolvendo exemplos e combinações (aglomerados da rede). Apesar disso, o algoritmo tem sua carga de E/S substancialmente diminuída. Isto

propicia um ganho na velocidade de execução do algoritmo. Mas, além do benefício na eficiência, essa aleração "abre caminho" para outra alteração, descrita a seguir.

Alteração 2: Armazenamento dos aglomerados com pesos positivos somente.

O aprendizado no MNC é realizado em duas fases: a primeira constrói uma rede intermediária a partir do treinamento da rede definida na topologia utilizada; a segunda gera a rede final a partir da poda da rede intermediária. Como sabemos, a rede intermediária pode assumir tamanho bastante grande. A presente alteração consiste em embutir uma poda "grosseira" já na primeira etapa do aprendizado, de forma a diminuir consideravelmente o tamanho da rede intermediária. A segunda etapa do aprendizado continua encarregada de realizar a poda de ajuste fino.

Devido à incompletude do conjunto de treinamento, podemos dividir os aglomerados presentes na rede intermediária em dois conjuntos: aquele dos aglomerados não encontrados em nenhum exemplo do conjunto de treinamento e aquele formado pelos aglomerados presentes em pelo menos um exemplo do conjunto de treinamento. Analisemos então o comportamento de cada conjunto durante a poda.

Os aglomerados não encontrados em nenhum exemplo do conjunto de treinamento não são afetados pelo algoritmo de treinamento da rede. Assim, comparecem na rede intermediária com uma quantidade nula de punições e recompensas. Sua poda na fase seguinte é certa. Portanto, em nome da economia de espaço e tempo, não precisam ser incluídos na rede intermediária.

Já entre os aglomerados representados no conjunto de treinamento, a situação é diferente. Sabemos que cada aglomerado está presente em todas as subredes. Pela construção do algoritmo de treinamento, somente uma subrede pode apresentar peso positivo (n^{o} de recompensas > n^{o} de punições) para cada tipo de aglomerado. Pode acontecer também de todas as subredes apresentarem peso nulo para o aglomerado (n^{o} de recompensas = n^{o} de punições). No primeiro caso, apenas o aglomerado de peso positivo tem alguma chance de sobreviver à poda da segunda fase. Os demais podem ser eliminados já na primeira fase, o que não acarretará nenhum prejuízo ao

aprendizado. Para o segundo caso, pode-se tomar duas decisões: eliminar o aglomerado de todas as subredes ou mantê-lo em todas elas. A primeira opção sustenta-se nas razões já elencadas. A segunda basea-se na terceira versão do algoritmo de poda (apêndice B.3), que preserva este tipo de aglomerado tendo em vista sua alta potencialidade para aprendizado incremental, i. e., aplicação de outros exemplos em outro treinamento da rede.

Esta alteração permite diminuir bastante o espaço em disco requerido pelo algoritmo. O espaço exato a ser ocupado depende da topologia e dos exemplos do conjunto de treinamento. No máximo, ele será igual ao tamanho da topologia multiplicado pelo número de bytes ocupados por cada aglomerado. Mas, na média, deve ser bem menor.

Para a implementação dessa alteração, foi necessário a modificação da estrutura dos arquivos que armazenam a REDE TREINADA: como nem todas as combinações são armazenadas, identificamos aquelas que o são através de um campo com o índice da combinação de acordo com um algoritmo que gera as combinações numa ordem pré-determinada. Como sabemos, o MNC gera uma subrede para cada classe do domínio. Como essas subredes são geradas "paralelamente", à medida que cada combinação da topologia é submetida ao conjunto de treinamento, cada subrede é armazenada em um arquivo distinto, um para cada classe. Esses arquivos são colocados em um mesmo diretório, juntamente com o arquivo de controle da REDE TREINADA.

Essa alteração faz com que se armazene apenas as combinações sintaticamente relevantes, i.e., que apresentam peso positivo. Isto só é possível devido à alteração A1, que permite ao algoritmo calcular "definitivamente" o peso de cada combinação antes de armazená-la.

Uma desvantagem dessa alteração é que ela insere dificuldades na implementação da versão incremental do algoritmo de aprendizado, visto que fornece uma rede já desfalcada de alguns aglomerados, que podem vir a fazer falta. Quanto a isso, alguns cuidados podem ser tomados, como a preservação daqueles aglomerados

que, apesar de possuírem peso nulo, possuem número de recompensas não nulo (ver algoritmo de poda no apêndice B.3).

7.2 - A estrutura do APR-X: os módulos

O tamanho das topologias envolvidas pode ser muito grande. Assim, a implementação do algoritmo deve prover uma forma eficaz de gerar, armazenar e recuperar cada aglomerado da rede.

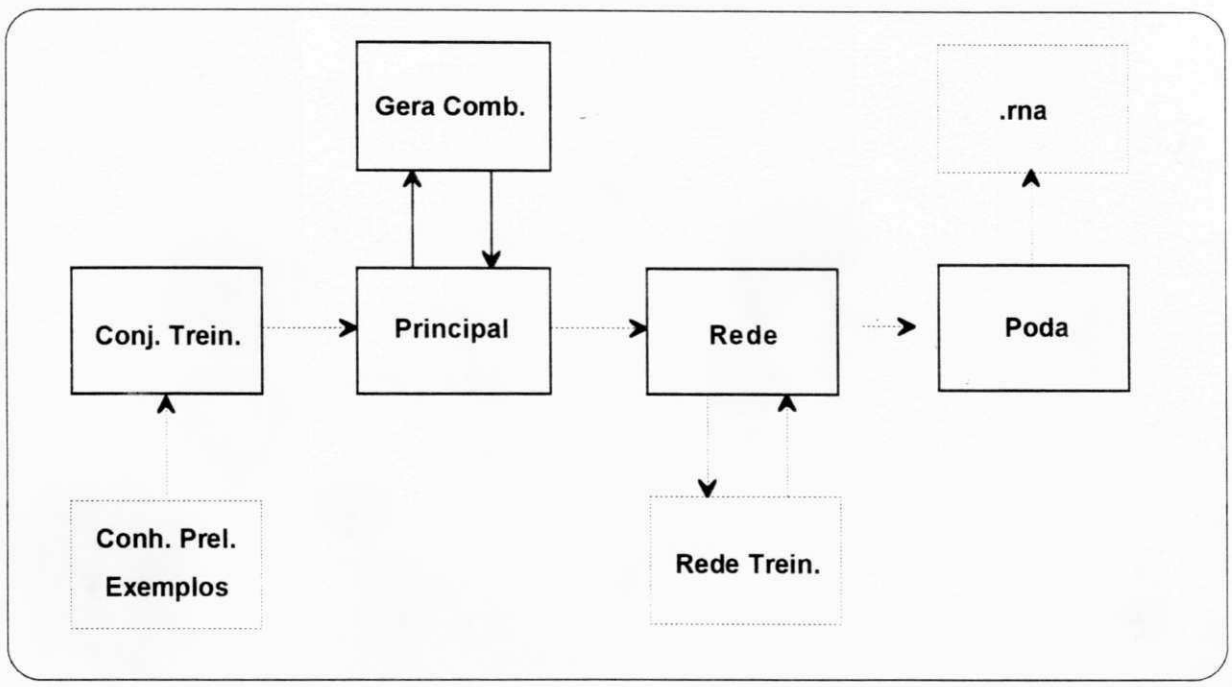


Figura 7.1 – Módulos do protótipo APR-X e arquivos de dados envolvidos.

A definição dos módulos seguiu aqueles princípios gerais que foram considerados em toda a implementação. Os principais módulos do protótipo são (figura 7.1):

- Principal: Responsável pela interface com o usuário, inicialização dos outros módulos e execução do algoritmo de treinamento propriamente dito – punições e recompensas sobre os aglomerados e decisão sobre sua gravação ou não na REDE TREINADA.

- Conjunto de Treinamento: Lê as caracteristicas do dominio, os exemplos e o conhecimento preliminar e os armazena de forma a proporcionar a maior eficiência ao aprendizado. Constrói as matrizes utilizadas na geração da topologia, descritas adiante.
- Gera Combinações: Implementa um gerador genérico de combinações. Para cada combinação gerada, ele chama uma função indicada para o tratamento de aglomerados, que pode variar de acordo com o que se esteja fazendo na rede. Por exemplo, esta função pode realizar o treinamento do aglomerado gerado submetendo-o a todos os exemplos do conjunto de treinamento, ou apenas gravá-lo em disco num determinado formato. O presente módulo utiliza a estrutura de dados gerada pelo módulo *Conjunto de Treinamento* para gerar de forma mais eficiente as topologias definidas neste trabalho.
- Rede: Grava a REDE TREINADA no disco, separando as subredes em arquivos diferentes. Gera ainda um arquivo de controle, com dados sobre as características das topologias e dados complementares calculados durante o treinamento, como maior peso de cada subrede, maior peso total, número de exemplos que satisfizeram cada aglomerado, etc. Essas informações são úteis para a poda, para um futuro aprendizado incremental e para a pesquisa de parâmetros possivelmente importantes

7.3 - Cuidados com a implementação de restrições à topologia

A topologia originalmente usada pelo MNC possui definição simples e crescimento complexo. Neste trabalho, definimos topologias alternativas, que podem ser construídas pela introdução de restrições à topologia original. As topologias alternativas apresentam definição mais complexa e crescimento um pouco mais suave que a original. Esta relação inversa entre a complexidade de construção e o tamanho da topologia nos põe à procura de uma solução de compromisso: devemos encontrar restrições que promovam uma diminuição substancial no tamanho com um aumento de complexidade de construção suportável.

As restrições derivadas do uso da relevância semântica e da valoração de atributos são compensatórias sob este aspecto. Contudo, isto só ocorre graças à técnica empregada para sua implementação. As dificuldades ligadas à implementação de geradores de topologias definidas através de restrições à topologia original estão no esforço computacional necessário para testar se cada aglomerado produzido satisfaz às ditas restrições. Como todos os aglomerados da topologia original precisam ser testados, o tempo gasto na geração da topologia modificada acaba sendo maior que aquele da topologia original.

Uma abordagem mais eficiente para a implementação de geradores para essas topologias é implementar a sua definição diretamente, libertando-se da definição da topologia original. Nos casos das topologias abordadas aqui, isto pode ser feito através de pequenas alterações no algoritmo gerador. Como a definição da topologia é mais detalhada, o custo para a geração de cada aglomerado se torna maior que no caso da topologia original. Contudo, como apenas os aglomerados pertencentes à topologia são gerados, o custo é bem menor que o da solução anterior.

O problema dessa última abordagem é que um conjunto adicional de decisões precisam ser tomadas para a geração de cada aglomerado da rede. Como até mesmo a TCROSES apresenta crescimento exponencial, o custo dessas decisões assume uma proporção considerável do custo total do aprendizado, consumindo bastante tempo na geração de redes para domínios reais. Assim, com o objetivo de diminuir o tempo necessário à geração da toplogia, recorremos a uma outra solução para o problema.

A solução implementada consiste em eliminar a maior parte das decisões necessárias à geração de cada aglomerado. Isto pode ser feito colocando-se essas decisões em uma estrutura de dados, calculada antes do início do treinamento, com base na definição da topologia a ser gerada. Esta solução funciona perfeitamente para as topologias definidas neste trabalho, mas não é universal. Portanto, a sua possibilidade de uso deve atuar como um novo parâmetro no cálculo da solução de compromisso a que nos referimos anteriormente.

A grande vantagem dessa solução é que o cálculo da estrutura de dados com as decisões codificadas apresenta esforço apenas linear com relação ao número de evidências e classes do domínio. Essa estrutura de dados é formada por duas matrizes para cada tipo de topologia possível. Essas matrizes possuem uma linha para cada classe e uma coluna para cada evidência do domínio.

O algoritmo original constrói a topologia de aglomerado em aglomerado, gerando sempre um novo aglomerado a partir de modificações no anterior: cada aglomerado é construído a partir do anterior, pela troca de uma ou mais evidências. A escolha da próxima evidência é feita de forma sequencial, baseada no código das evidências que compõem o aglomerado.

O gerador de topologia que implementamos escolhe a próxima evidência a usar não pelo simples incremento de seu código, mas de acordo com as matrizes calculadas anteriormente, a partir do tipo da topologia e da estrutura do domínio. No caso da TCROSES, as matrizes usadas são *prox_rel* e *prox_va_rel*, conforme se esteja construindo uma nova família de aglomerados ou derivando novos aglomerados da mesma família – consideramos da mesma família aglomerados que começam com a mesma evidência. O conteúdo da matriz *prox_rel* leva em consideração a relevância semântica, enquanto que a *prox_va rel* considera a relevância semântica e a valoração de atributos, simultaneamente.

O uso dessas matrizes, juntamente com um algoritmo genérico modificado de geração de combinações, permite a criação de aglomerados para as topologias de forma tão eficiente quanto o algoritmo gerador da topologia original. O apêndice D reproduz o trecho de código que implementa o gerador de topologias. Este código faz parte do módulo *Gera Combinações*, descrito anteriormente. O apêndice E reproduz o trecho de código que constrói as matrizes. Este código faz parte do módulo *Conjunto de Treinamento*.

Como alertamos anteriormente, nem todas as alterações podem ser implementadas através desta técnica. Mostraremos aqui um exemplo de alteração que chegou a ser considerada, mas que foi abandonada, em nome daquela solução de

compromisso, por não poder usufruir da eficiência deste tipo de implementação. Trata-se de um refinamento da valoração de atributos: a alteração consiste em considerar, dentro de cada atributo, grupos de valores não exclusivos internamente mas exclusivos em relação a outros grupos. Assim, em vez de considerar o atributo como monovalorado ou multivalorado, podemos fazer uma investigação mais acurada para identificar quais dentre seus valores podem ser combinados entre si. Uma análise desta idéia mostra que essa informação é muito particular, de difícil eliciação e implementação, e que não apresenta uma redução significativamente maior que a pura valoração de atributos, que já nem é tão eficaz quando usada separadamente.

7.4 - Dados sobre o desempenho do protótipo

Desejamos finalizar a discussão sobre a implementação do algoritmo de geração e treinamento da topologia apresentando alguns números sobre o desempenho desta implementação, no que concerne ao tempo e espaço requeridos.

7.4.1 - Quanto ao tempo

Muitos fatores influenciam no tempo necessário para se gerar uma topologia e treiná-la com uma determinada quantidade de exemplos. Contudo, podemos calcular, a grosso modo, o tempo gasto por um certo computador para gerar um aglomerado e submetê-lo a uma determinada quantidade de exemplos. Este tempo básico, obtido de forma experimental, pode ser muito importante para se estimar o tempo a ser despendido ao se variar o tamanho da topologia para o mesmo domínio ou um domínio semelhante.

Os testes de desempenho realizados em dois ambientes computacionais diferentes mostram que o tempo necessário para gerar **um** aglomerado e submetê-lo a todos os exemplos de um conjunto de treinamento com 200 casos é de:

- $2{,}2*10^{-3}$ segundo, para um microcomputador 386 DX ISA com 33 MHz, sem co-processador aritmético, rodando DOS; e
- $3{,}5*10^{-4}$ segundo, para uma estação de trabalho Sun Sparcstation II, rodando SunOs e Openwindows.

As duas máquinas testadas apresentam desempenho geral bastante diferenciado. A pequena diferença de desempenho encontrada na execução do algoritmo de geração e treinamento da rede pode ser parcialmente explicada pelo fato da implementação não usar aritmética de ponto flutuante; em todos os pontos críticos do algoritmo, apenas a aritmética inteira é usada.

Para um mesmo domínio e conjunto de treinamento, o tempo gasto para se gerar e treinar uma rede é diretamente proporcional ao seu tamanho, sendo que a constante de proporcionalidade varia de acordo com a plataforma computacional e assume os valores mencionados acima.

Para se ter uma idéia do que esses números, em conjunto com os números que indicam os tamanhos das diversas topologias apresentadas neste trabalho, significam, vejamos as seguintes estimativas.

Considerando o domínio hipotético do capítulo 6, com uma base de treinamento composta por 200 casos e usando topologias de ordem 7, temos que, em um 386 DX, a TCOS despenderia 14,1 horas e a TCROSES 3,1 minutos. Caso a ordem seja 9, esses números vão para 9,4 dias e 5,9 minutos, respectivamente.

Considerando ainda o mesmo domínio e a mesma quantidade de casos, se a máquina usada for uma Sun SparcStation II, a ordem 7 leva 2,2 horas para a TCOS e 29,8 segundos para a TCROSES. Já a ordem 9 vai a 1,5 dia para a TCOS e 56 segundos para a TCROSES.

Para um domínio mais complexo (maior quantidade de evidências), esses números são ainda mais discrepantes. Se submetermos 200 casos de um domínio semelhante ao Ginecol para treinamento numa Sun SparcSation II, o tempo para gerar e treinar uma rede de ordem 7 será de 5 anos para a TCOS e 1,9 hora para a TCROSES. Se a máquina usada for um 386 DX, esses tempos vão para 31,4 anos e 12,2 horas, respectivamente. Vale lembrar que esses tempos são estimados.

7.4.2 - Quanto ao espaço

Cada aglomerado gravado na REDE TREINADA ocupa 8 bytes (4 bytes para um inteiro que determina a combinação e 4 bytes para dois inteiros curtos que guardam a quantidade de punições e recompensas). Além disso, há um arquivo de controle que contém informações globais sobre o domínio, como: nome do domínio, nome das classes, atributos e evidências, ordem da rede, tipo da topologia usada, etc. O tamanho deste arquivo de controle não é significativo, pois cresce apenas linearmente com a quantidade de classes, atributos e evidências do domínio e não é afetado pelo tamanho da topologia em si.

Posto isso, vemos que o montante de espaço requerido pelo algoritmo depende basicamente da quantidade de aglomerados gravados na REDE TREINADA. Contudo, como vimos na alteração A.2 (seção 7.1.1), apenas os aglomerados com peso positivo são gravados na REDE TREINADA. Acontece que o peso dos aglomerados só é conhecido após o treinamento, pois depende fundamentalmente dos exemplos do conjunto de treinamento. Desta forma, a quantidade de espaço requerida pelo algoritmo depende mais do conjunto de treinamento do que da topologia em si.

Assim, fica difícil prever a quantidade de espaço requerida para um determinado caso. A quantidade de aglomerados da REDE TREINADA será uma parcela do tamanho da topologia. A título ilustrativo, vejamos o espaço ocupado pela REDE TREINADA de um subconjunto útil do Ginecol (5 classes, 153 evidências e 59 exemplos), onde foram eliminadas classes com pequena quantidade de exemplos representativos. Para a TCROSES de ordem 3, a quantidade de espaço requerida é de 13 Kbytes (7 Kbytes do arquivo de controle e 6 Kbytes correspondentes a 768 aglomerados). Para a TCROSES de ordem 4, a quatidade de espaço consumida passa para cerca de 22 Kbytes (7 Kbytes para o arquivo de controle e 15 Kbytes de dados, correpondentes a 1920 aglomeraados). Podemos ver que a quantidade de aglomerados é muito inferior ao tamanho das topologias. Isto se deve à alteração A.2.

7.5 - A poda

Como pode ser visto na figura 2.6, a poda analisa a REDE TREINADA, que contém aglomerados com acumuladores de punição e recompensa e gera a rede final podada e com pesos normalizados no intervalo [0,1] (REDE PODADA E NORMALIZADA).

A preservação da REDE TREINADA, pela poda, possibilita seu retreinamento (aprendizado incremental) e a possibilidade de testar vários tipos de poda diferentes, com outros critérios e limiares de poda.

O denominador de normalização usado no apêndice B.1 leva em consideração o maior peso da rede como um todo. Isto não parece adequado. Passamos a usar o maior peso de cada classe. Esta modificação também apareceu nas versões seguintes, propostas pelos autores.

O estudo para a escolha de um denominador de normalização levou à consideração do uso de diversas outras informações deriváveis do aprendizado, como, por exemplo, informações locais a cada hipótese (número de exemplos para cada hipótese, maior peso e peso máximo entre os caminhos que levam à hipótese) e globais, referentes ao domínio, como maior peso entre os caminhos da rede .

A poda que implementamos preserva os caminhos ou aglomerados da REDE TREINADA que se encaixam em uma das seguintes situações: primeira, os caminhos cujo peso (diferença positiva entre as quantidades de recompensas e punições) ultrapassa um determinado limiar (0,40), conhecido como limiar de poda; e segunda, os caminhos patognomônicos, que são aqueles em que o número de recompensas é maior que zero e o número de punições é igual a zero.

A implementação do algoritmo de poda aproveita as mesmas ferramentas usadas pelo algoritmo de aprendizado. A REDE TREINADA está contida em vários arquivos, sendo um para cada classe e mais um de controle, que contém informações sobre a estrutura do domínio, e os tipos de conhecimento preliminar utilizados, bem como os dados globais resultantes do treinamento.

O protótipo implementado lê os dados desses arquivos e solicita o limiar de poda a ser aplicado. Em seguida, realiza a poda e normalização da REDE TREINADA e apresenta a REDE PODADA E NORMALIZADA em dois formatos distintos: um relatório legível para seres humanos na saída padrão e um aquivo com a descrição da rede a ser usada pelo algoritmo de classificação, de nome "espec.rns", onde espec é o nome do domínio em questão.

O relatório da saída padrão contém um cabeçalho que dá informações globais sobre o domínio e o aprendizado em si, como nome do domínio, quantidade de atributos, evidências e exemplos, maior peso da rede, etc. Em seguida, aparecem as informações sobre cada classe, separadamente. Para cada uma, aparece o nome da classe, a quantidade de exemplos pertencentes àquela classe, o maior peso entre os caminhos daquela classe e os caminhos propriamente ditos, um por linha, em formato legível. Os caminhos patognomônicos são marcados com um asterisco.

O arquivo com a descrição da rede não contém cabeçalhos: apenas os caminhos, um por linha, onde cada linha contém os códigos, em formato ASCII, da classe e das evidências que levam a ela, em conjunção.

8 - Conclusão e Trabalhos Futuros

8.1 - Conclusão

O aprendizado no MNC é baseado no treinamento de uma rede aciclica, a partir de um conjunto de exemplos. A topologia dessa rede exerce grande influência sobre o desempenho do algoritmo e sobre a qualidade do aprendizado. Abordamos aqui formas de melhorar o resultado do aprendizado, não através de mudanças nos algoritmos de treinamento e poda, mas sim através de alterações na topologia originalmente gerada para ser treinada. Essas alterações decorrem do uso de dois tipos diferentes de conhecimento preliminar (*background knowledge*): a relevância semântica e a quantidade de valores simultâneos que cada atributo pode assumir. As topologias obtidas dessa forma foram comparadas entre si e com a original, levando-se em consideração as mudanças necessárias na forma de modelar o domínio, e as melhorias decorrentes na qualidade do aprendizado e no desempenho do algoritmo.

A TCROS (Relevante) e a TCOSES (Exclusividade Seletiva) permitiram a confecção de estudos isolados sobre a contribuição dada por cada tipo de conhecimento preliminar. A TCROSES mostra que essas contribuições podem ser unidas em uma só topologia, proporcionando um resultado bem melhor que aquele esperado a partir do estudo de suas contribuições individuais.

A Topologia Combinatorial Relevante de Ordem Superior e Exclusividade Seletiva (TCROSES) proporciona uma melhoria no resultado gerado pelo algoritmo e mantém as vantagens originais do MNC, como: robustez (admissão de contra-exemplos, buracos e atributos multivalorados no conjunto de treinamento), conhecimento expresso numa forma inteligivel e verificável diretamente por um especialista humano e facilidade na modelagem do domínio. Contudo, alguns dos problemas originais são mantidos, como: a não captura de conhecimento hierárquico (estruturado), que se deve ao "achatamento" da camada intermediária das topologias estudadas; e a inexistência de garantia de que o conhecimento gerado classifique

todos os exemplos do conjunto de treinamento. Esta última "desvantagem" está diretamente relacionada à possibilidade de aceitar contra-exemplos, visto ser impossível que a rede gerada classifique corretamente um exemplo e seu contra-exemplo.

O aprendizado automático não pode se restringir ao uso de uma única fonte de conhecimento; deve sim lançar mão de toda ajuda que esteja à sua disposição, representada nas mais diversas formas possíveis (exemplos, matrizes de relevância, características dos atributos do domínio, etc). A fidelidade a um paradigma não lhe garante bons resultados, mas o aproveitamento das melhores características de vários paradigmas pode lhe garantir as melhores armas disponíveis para que possa realizar seu trabalho, que é fixar conhecimento de qualidade, da melhor forma possível. As alterações propostas neste trabalho contribuem para a melhoria dos resultados obtidos pelo MNC, e se encaixam perfeitamente às demais extensões que vêm sendo propostas ao modelo.

8.2 - Sugestões para trabalhos futuros

Este trabalho está longe de esgotar o estudo dos reflexos causados pelo uso da relevância semântica e da valoração de atributos no Modelo Neural Combinatório. Apresentamos, a seguir, algumas sugestões de trabalhos que podem ser desenvolvidos como uma continuação natural deste.

- Validar as idéias apresentadas neste trabalho usando um domínio real. A validação deve iniciar obrigatoriamente pela modelagem do domínio, não devendo ser aproveitada uma modelagem feita para outro algoritmo ou ambiente. Isto garantirá a reflexão nos resultados das vantagens resultantes da flexibilidade de modelagem oferecida pelo MNC.
- Comparar o desempenho geral entre o aprendizado no MNC estendido pela alterações aqui propostas e em algoritmos indutivos, fazendo modelagens específicas para cada paradigma de aprendizado. Comparar a facilidade de

modelagem, grau de generalização, grau de acerto, inteligibilidade, incrementabilidade, etc.

- Sugerir estudos e alterações na poda: mudança do quociente de normalização e possibilidade de expurgo de aglomerados específicos que levam à mesma conclusão de aglomerados genéricos de alto peso. Ou seja, eliminar aglomerados cujos subconjuntos estejam na rede com bom peso.
- Conceber um analisador de matriz de relevância para extrair informações como: quantidade de valores relevantes por atributo e por classe, número de evidências relevantes por classe, médias, desvio-padrão, etc. Gerar também um relatório com as quantidades de evidências relevantes por classe.
- Construir um módulo gerenciador que escolha a topologia a ser usada em cada caso. Este módulo deve conter algum conhecimento heurístico, visto que o simples cálculo do tamanho da topologia pode requerer um esforço computacional significativo.
- Comparar o MNC com as listas de decisão apresentadas em [Gomes 93].
- Estudar a aplicação de Conhecimento Preliminar aos algoritmos genéticos do Modelo Conexionista Evolutivo (MCE) [Denis 91].

Apêndice A: O Algoritmo de Treinamento do MNC

A.1) Versão apresentada em [Machado 89]:

REGRA_DE_APRENDIZADO
- Criar para cada arco da rede um acumulador com valor inicial 0
- Para cada exemplo da base de dados de treinamento, faça:
    - Propague as crenças nas evidências dos nós de entrada até a camada de hipóteses.
    - Para cada arco que alcança um nó hipótese, faça:
        - Se o nó hipótese alcançado corresponde à classe correta do caso
        então: propague retroativamente a partir deste nó até os nós de entrada, incrementado os acumuladores de cada arco percorrido pelo valor de seu fluxo evidencial (Recompensa)
        senão propague retroativamente a partir deste nó até os nós de entrada, decrementando os acumuladores de cada arco percorrido pelo valor de seu fluxo evidencial (Punição)

A.2) Versão apresentada em [Denis 91]

- Criar para cada arco da rede um acumulador para recompensar e um acumulador para punições, ambos com valor inicial 0
- Fazer todos os pesos sinápticos da rede iguais a 1
- Para cada exemplo da base de dados de treinamento, faça:
    - Propagar as crenças dos nós de entrada até a camada de hipóteses.
    - Para cada arco que alcança um nó-hipótese faça:
    - Se o nó-hipótese alcançado corresponde à classe correta do caso
        então: propague retroativamente a partir deste nó até os nós de entrada, incrementado o acumulador de recompensas de cada arco percorrido pelo produto (Recompensa):
        Fluxo evidencial * ativação do neurônio de destino * importância do exemplo
        senão: propague retroativamente a partir deste nó até os nós de entrada, decrementando o acumulador de punições de cada arco percorrido pelo produto (Punição):
        Fluxo evidencial * ativação do neurônio de destino * importância do exemplo
        - Fim-Se
    - Fim-Para
- Fim-Para
- Fim

A.3) Versão apresentada em [Machado 92]:

- Estabeleça para cada sinapse da rede um acumulador para recompensas e um acumulador para punições, ambos com valores iniciais iguais a zero.
- Inicialize todos os pesos de arcos da rede com 1.
- Para cada exemplo do conjunto de treinamento, faça:
    Propague as crenças nas evidências a partir dos nós de entrada até a camada de saída
    Para cada sinapse que alcança um nó de saída, faça:
        Se o nó alcançado corresponde à classificação correta do caso,
        então: Navegue retroativamente a partir deste nó até os nós de entrada,
          incrementando o acumulador de recompensa de cada arco visitado pelo produto:
            fluxo evidencial*ativação do neurônio de destino*importância do exemplo
        senão: Navegue retroativamente a partir deste nó até os nós de entrada,
          incrementando o acumulador de punição de cada arco visitado pelo produto:
            fluxo evidencial*ativação do neurônio de destino*importância do exemplo

Apêndice B: O Algoritmo de Poda do MNC

B.1) Versão retirada de [Machado 89]

REGRA_DE_PODA_DA_REDE

- Remover da rede todos arcos cujos valores de acumuladores sejam mais baixos que o limiar de poda.
- Remover todas as células das camadas de entrada e combinatória que se tornem desconectadas de todos os nós-hipótese.
- Fazer os pesos dos arcos iguais aos valores obtidos dividindo os acumuladores de arcos pelo maior valor de acumulador que ocorre na rede.

B.2) Versão retirada de [Denis 91]

Início - Para cada arco da rede, faça:
- Calcule o valor líquido dos acumuladores:
  ACUMLIQ = acumulador de recompensas - acumulador de punições
- Se ACUMLIQ ≤ 0
  então: Remova o arco da rede; Fim-Para
  Fim-Se.
- Se acumulador de punições do arco > 0
  então: Compute o peso do arco como: ACUMLIQ/MAXLIQ
  onde MAXLIQ = máximo ACUMLIQ na subrede da hipótese a qual pertence o arco.
  senão: Compute o peso do arco como:
  SQR(Taceit) + (1-SQR(Taceit))*ACUMLIQ/MAXLIQ
  Fim-Se.
- Se peso do arco < limiar de poda
  então: Remova o arco da rede
  Fim-Se.
  Fim-Para
- Remover todos os neurônios das camadas de entrada e combinatória que percam conectividade com os nós-hipótese.

Fim.

B.3) Versão retirada de [Machado 92] (traduzida do Inglês):

- Para cada sinapse da rede, faça:
    - Compute o valor do acumulador de rede:
    NETACC = valor do acumulador de recompensas - valor do acumulador de punições
    - Se NETACC ≤ 0 então remova a sinapse da rede; Fim-Para.
    - Se os acumuladores de punição e recompensa do arco tiverem o mesmo valor
    então: peso do arco = 0; Fim-Para (é conservado na rede).
    - Se o valor do acumulador de punições do arco > 0
    então - Compute o peso do arco como: NETACC/MAXNET
    senão - Compute o peso do arco como:
        SQR(Tacc) + (1-SQR(Tacc))*NETACC/MAXNET
    - Se o peso do arco < limiar de poda então exclua o arco
- Remova todos os neurônios que perdem acesso aos nós de saída.

Obs:

MAXNET é a maior NETACC existente na subrede da categoria à qual pertence este arco.

Apêndice C: Dados sobre o estudo numérico artificial

Este apêndice contém os dados resultantes dos experimentos descritos no capítulo 6. Estes dados são os tamanhos das topologias para cada experimento; são a versão numérica dos gráficos apresentados no capítulo.

Ordem	TCOS	TCOSES	TCROS	TCROSES
1	4,0e+01	4,0e+01	6,0e+01	6,0e+01
2	8,2e+02	1,8e+02	4,4e+02	4,3e+02
3	1,1e+04	9,1e+03	2,3e+03	2,1e+03
4	1,0e+05	7,3e+04	9,2e+03	7,7e+03
5	7,6e+05	4,3e+05	2,9e+04	2,2e+04
6	4,6e+06	2,0e+06	7,6e+04	4,8e+04
7	2,3e+07	7,0e+06	1,7e+05	8,5e+04
8	1,0e+08	2,0e+07	3,0e+05	1,3e+05
9	3,7e+08	4,7e+07	4,8e+05	1,6e+05

Figura C.1 – Parâmetro *ordem*. Refere-se à figura 6.5.

	Cenário A	Cenário B	Cenário C
TCOS	4,60e+06	4,60e+06	4,60e+06
TCOSES	1,39e+06	1,96e+06	2,00e+06
TCROS	7,63e+04	7,63e+04	7,63e+04
TCROSES	3,81e+04	4,77e+04	4,43e+04

Figura C.2 – Parâmetro *distribuição do número de valores por atributo*. Refere-se às figuras 6.6-A e 6.6-B.

	Cenário A	Cenário B	Cenário C	Cenário D
TCOS	4,60e+06	4,60e+06	4,60e+06	4,60e+06
TCOSES	1,26e+06	1,96e+06	3,17e+06	4,60e+06
TCROS	7,63e+04	7,63e+04	7,63e+04	7,63e+04
TCROSES	3,63e+04	4,77e+04	5,93e+04	7,63e+04

Figura C.3 – Parâmetro *grau de valoração*. Refere-se às figuras 6.7-A e 6.7-B.

	Cenário A	Cenário B	Cenário C
TCOS	4,60e+06	4,60e+06	4,60e+06
TCOSES	1,50e+06	1,96e+06	2,65e+06
TCROS	7,63e+04	7,63e+04	7,63e+04
TCROSES	3,92e+04	4,77e+04	5,45e+04

Figura C.4 – Parâmetro *distribuição da valoração de atributos*. Refere-se às figura 6.8-A e 6.8-B.

	Cenário A	Cenário B	Cenário C
TCOS	4,60e+06	4,60e+06	4,60e+06
TCOSES	1,96e+06	1,96e+06	1,96e+06
TCROS	5,08e+03	7,63e+04	2,07e+06
TCROSES	4,43e+03	4,77e+04	8,52e+05

Figura C.5 – Parâmetro distribuição da relevância entre as classes. Refere-se às figuras 6.9-A e 6.9-B.

	15%	25%	35%	45%	55%	65%	75%	85%	95%
TCOS	4,6e+6	4,6e+6	4,6e+6	4,6e+6	4,6e+6	4,6e+6	4,6e+6	4,6e+6	4,6e+6
TCOSES	2,0e+6	2,0e+6	2,0e+6	2,0e+6	2,0e+6	2,0e+6	2,0e+6	2,0e+6	2,0e+6
TCROS	3,8e+2	5,1e+3	3,9e+4	1,9e+5	6,6e+5	2,2e+6	4,6e+6	1,0e+7	2,0e+7
TCROSES	3,8e+2	4,4e+3	2,8e+4	9,9e+4	3,4e+5	1,1e+6	2,1e+6	4,5e+6	8,6e+6

Figura C.6 – Parâmetro grau de relevância, mantendo distribuição uniforme. Refere-se à figura 6.10-A.

	20%	25%	30%	35%	40%	45%	50%
TCOS	4,60e+06	4,60e+06	4,60e+06	4,60e+06	4,60e+06	4,60e+06	4,60e+06
TCOSES	1,96e+06	1,96e+06	1,96e+06	1,96e+06	1,96e+06	1,96e+06	1,96e+06
TCROS	1,93e+04	7,63e+04	2,37e+05	6,16e+05	1,48e+06	3,05e+06	5,81e+06
TCROSES	1,40e+04	4,77e+04	1,07e+05	3,10+e05	6,86e+05	1,35e+06	2,52e+06

Figura C.7 – Parâmetro grau de relevância, mantendo distribuição dispersa. Refere-se à figura 6.10-B.

Apêndice D: Gerador de combinações

Este apêndice apresenta o trecho de código responsável pela geração dos aglomerados para as diversas topologias. O código apresentado faz parte do módulo *Gera Combinações*.

```
...
static void (*processa_comb)( int*, int, int );
static int  vetor_comb[MAXMCOMB + 2];
...
static int  cod_clas;
static void (*gera_comb_aux[])(int, int) = {
            gera_comb_tcos_aux,
            gera_comb_tcoses_aux,
            gera_comb_tcros_aux,
            gera_comb_tcroses_aux };

/***
      Gera as combinacoes para uma subrede, de acordo com a
      topologia e a classe indicada.
      As combinacoes sao geradas em uma sequencia bem determinada.
***/
void  gera_comb( topol, lprocessa_comb, lcod_clas)
int   topol;
void  (*lprocessa_comb)(int*, int, int);
int   lcod_clas;
{
      register int      i_ord;

      processa_comb = lprocessa_comb;
      cod_clas = lcod_clas;
      if( topol < 0 || topol > 3 ){
            fprintf( stderr, "\nErro em gera_comb: topologia inexistente." );
            exit( 1 );
      }
      if( tot_evid < ordem_rede || ordem_rede < 1 || tot_evid < 1 ||
            ordem_rede > MAXMCOMB ){
            fprintf( stderr, "\nErro nos parametros de gera_comb." );
            exit( 1 );
      }

      vetor_comb[0] = 0;
      for( i_ord = 1 ; i_ord <= ordem_rede ; i_ord++ ){
            (gera_comb_aux[topol])( 1, i_ord );
      }
}
```

```
/***
      Gera as combinacoes da TCOS a partir da posicao 'i' do 'vetor_comb'.
***/
static void gera_comb_tcos_aux( i, ord )
int   i;
int   ord;
{
      for(  vetor_comb[i] = vetor_comb[i-1] + 1;
            vetor_comb[i] <= tot_evid - ord + i;
            vetor_comb[i] ++  ){
            if( i == ord ){
                  (*processa_comb)( vetor_comb, ord, cod_clas );
            } else {
                  gera_comb_tcos_aux( i+1, ord );
            }
      }
}

/***
      Gera as combinacoes da TCOSES a partir da posicao 'i' do 'vetor_comb'.
***/
static void gera_comb_tcoses_aux( i, ord )
int   i;
int   ord;
{
      for(  vetor_comb[i] = prox_evid_va[0][vetor_comb[i-1]];
            vetor_comb[i] <= tot_evid - ord + i;
            vetor_comb[i] ++  ){
            if( i == ord ){
                  (*processa_comb)( vetor_comb, ord, cod_clas );
            } else {
                  gera_comb_tcoses_aux( i+1, ord );
            }
      }
}

/***
      Gera as combinacoes da TCROS a partir da posicao 'i' do 'vetor_comb'.
***/
static void gera_comb_tcros_aux( i, ord )
int   i;
int   ord;
{
      for(  vetor_comb[i] = prox_evid_rel[cod_clas][ vetor_comb[i-1] ];
            vetor_comb[i] <= tot_evid - ord + i;      /* Ou ... <= tot_evid */
            vetor_comb[i] = prox_evid_rel[cod_clas][vetor_comb[i]]      ){
            if( i == ord ){
                  (*processa_comb)( vetor_comb, ord, cod_clas );
            } else {
                  gera_comb_tcros_aux( i+1, ord );
            }
      }
}
```

```
/***
      Gera as combinacoes da TCROSES a partir da posicao 'i' do 'vetor_comb'.
***/
static void gera_comb_tcroses_aux( i, ord )
int   i;
int   ord;
{
      for(  vetor_comb[i] = prox_evid_varel[cod_clas][ vetor_comb[i-1] ];
            vetor_comb[i] <= tot_evid - ord + i;      /* Ou ... <= tot_evid */
            vetor_comb[i] = prox_evid_rel[cod_clas][vetor_comb[i]]      ){
            if( i == ord ){
                  (*processa_comb)( vetor_comb, ord, cod_clas );
            } else {
                  gera_comb_tcroses_aux( i+1, ord );
            }
      }
}
```

Apêndice E: Construção da estrutura de auxílio à geração das topologias

Este apêndice apresenta o trecho de código responsável pela construção da estrutura de dados usada pela rotina de geração da topologia. Essa estrutura é composta por três matrizes e contém informações sobre a relevância semântica, a valoração de atributos e a estrutura do domínio numa forma otimizada para a geração das topologias. O código aqui apresentado foi extraído do módulo *Dados de Treinamento*.

```
/*
 *    Cria matrizes de trabalho a partir de informacoes sobre relevancia
 *    semantica (matr_rel[][]) e valoracao de atributos tab_atrib[].valoracao)
 *    As matrizes criadas sao usadas por gera_comb para gerar a topologia a
 *    ser visitada, com o objetivo de treinamento, analise ou impressao.
 *
 *    Usa fato de que as evidencias sao agrupadas por atributos (sequencia
 *    crescente de codigos).
 *    No caso de se ter uma MR com pesos nebulosos, basta estabelecer um
 *    limiar de relevancia e mudar a comparacao abaixo.
 */

/*
 *    prox_evid_va[0][cod_evid]. Igual para todas as classes.
 *    Usada pela TCOSES.
 */

      if( ( *approx_evid_va = prox_evid_va = (Tprox_evid*) calloc( 1,
            sizeof(Tprox_evid) ) ) == NULL ){
            fprintf( stderr, "\nNao consegui alocar memoria p/ trabalho.\n" );
            exit( 1 );
      }
      prox_evid_va[0][0] = 1;
      for( i = 1 ; i <= *aptot_evid ; i++ ){
            if( tab_atrib[tab_evid[i].cod_atrib].valoracao > 1 ){
                  prox_evid_va[0][i] = i+1;
            } else {
                  prox_evid_va[0][i] =
                  tab_atrib[tab_evid[i].cod_atrib+1].prim_evid;
            }
      }
```

```
/*
 *    prox_evid_rel[cod_clas][cod_evid]
 *    Usada pela TCROS e TCROSES.
 */
      if( ( *approx_evid_rel = prox_evid_rel = (Tprox_evid*) calloc(
            *aptot_clas + 1, sizeof(Tprox_evid) ) ) == NULL ){
            fprintf( stderr, "\nNao consegui alocar memoria p/ trabalho.\n" );
            exit( 1 );
      }
      for( i = 1 ; i <= *aptot_clas ; i++ ){
            int   evid_rel;
            evid_rel = *aptot_evid + 1;   /* Nenhuma -- infinito */
            prox_evid_rel[i][*aptot_evid] = evid_rel;
            for( j = *aptot_evid ; j >= 1 ; j-- ){
                  if( matr_rel[i][j] != 0 ){
                        evid_rel = j;
                  }
                  prox_evid_rel[i][j-1] = evid_rel;
            }
      }
/*
 *    prox_evid_varel[cod_clas][cod_evid] e' criada a partir de prox_evid_rel,
 *    alterando os casos em que os atributos sao monovalorados
 */
      if( ( *approx_evid_varel = prox_evid_varel = (Tprox_evid*) calloc(
            *aptot_clas + 1, sizeof(Tprox_evid) ) ) == NULL ){
            fprintf( stderr, "\nNao consegui alocar memoria p/ trabalho.\n" );
            exit( 1 );
      }
      memcpy( prox_evid_varel, prox_evid_rel, (*aptot_clas+1) *
      sizeof(Tprox_evid ) );
      for( i = 1 ; i <= *aptot_clas ; i++ ){
            for( j = *aptot_evid-1 ; j >= 1 ; j-- ){
                  if( tab_atrib[ tab_evid[j].cod_atrib ].valoracao <= 1 ){
                        prox_evid_varel[i][j] =
      prox_evid_varel[i][tab_atrib[tab_evid[j].cod_atrib+1].prim_evid-1];
                  }
            }
      }
```

Referências

[Austin 90] AUSTIN, S., "An Introduction to Genetic Algorithms", AI Expert, p. 48-53, 1990.

[Boose 86] BOOSE, John H. "Rapid Acquisition and Combination of Knowledge from Multiple Experts in the Same Domain", Future Computing Systems, vol.1, no.2, 1986.

[Boose 88] BOOSE, John H., SCHEMA, D.B. & BRADSHAW, J.M. Recent progress in Aquinas: a knowledge acquisition workbench. In: Proceedings of the European Knowledge Acquisition Workshop (EKAW'88), June, 1988.

[Boose 90] BOOSE, John H. "Knowledge Acquisition Tools, Methods and Mediating Representations". In MUTODA, H., MIZOGUCHI, R., BOOSE, J. & GAINES, B. (Eds.), Knowledge Acquisition for Knowledge-Based Systems, IOS Press, Ohmsha Ltd., Tokyo, 1990.

[Carbonell 83] CARBONELL, J.G., MICHALSKY, R.S. & MITCHELL, T.M., "An Overview of Machine Learning". In: Machine Learning I, Eds: MICHALSKY, R.S., CARBONELL, J.G. & MITCHELL, T.M., Tioga, Paulo Alto, Calif., 1983.

[Carbonell 89] CARBONELL, J.G., "Introduction: Paradigms for Machine Learning", Artificial Intelligence, v.40, p.1-9, 1989.

[Cirne 91] CIRNE F., W.C. & MONGIOVI, G. "Indução Semântica de Regras Modulares". Anais do VIII Simpósio Brasileiro de Inteligência Artificial, p. 143-148. Brasília, 1991.

[Cirne 92] CIRNE F., W.C. "Uso de Semântica na Melhoria dos Métodos Automáticos de Aquisição de Conhecimento". Dissertação apresentada ao curso de Mestrado em Informática, Universidade Federal da Paraíba, Brasil, 1992.

[Denis 91] DENIS, F.A.R.M. & MACHADO, R.J. "O Modelo Conexionista Evolutivo". Relatório Técnico, IBM, Centro Científico Rio, CCR-128, Setembro, 1991.

[Dietterich 83] DIETTERICH, T.G. & MICHALSKY, R.S., "A Comparative Review of Selected Methods for Learning from Examples". In: Machine Learnig I, Eds: MICHALSKY, R.S., CARBONELL, J.G. & MITCHELL, T.M., Tioga, Paulo Alto, Calif., 1983.

[Donato 93] DONATO Jr., E.T., "Uso de Semântica na Melhoria do Aprendizado em um Modelo Simbólico-Conexionista", Anais do X Simpósio Brasileiro de Inteligência Artificial, Porto Alegre, Brasil, Out., 1993.

[Duarte 88] DUARTE, V.H.A., "An Expert System for Weather Forecasting", Technical Report CCR058, IBM, Rio Scientific Center, June, 1988.

[Dutta 88] DUTTA, S. "Domain Independent Inductive Learning of Relevance". Proceedings of The European Knowledge Acquisition Workshop – EKAW'88, 1988.

[Fairley 85] FAIRLEY, R.E., "Software Engineering Concepts", McGraw-Hill, 1985.

[Feigenbaum 82] FEIGENBAUM, E. A., "The Handbook of Artificial Intelligence", vol. 3, William Caufmann ed., Los Altos, CA, 1982.

[Gallant 88] GALLANT, S.I., "Conexionist Expert Systems", Communications of the ACM, v.31, n.2, p. 152-168, Feb, 1988.

[Gomes 93] GOMES, F. C. & CASCUEL, O. "Learning Stochastic Decision Lists with Limited Complexity", Anais do X Simósio Brasileiro de Inteligência Artificial, Porto Alegre, Brasil, Out., 1993.

[Holland 86] HOLLAND, J.H., HOLYOAK, K.S., NISBETT, R.E. & THAGARD, P.R., "Induction Process of Inference, Learning and Discovery". Eds: FELDMAN, J.A., HAYES, P.J. & RUMELHART, D.E. The Massachusetts Institute of Technology, 1986.

[Jackson 86] JACKSON, P., "Introduction to Expert Systems", Addison-Wesley Publishing, 1986.

[Kandel 91] KANDEL, A. & LANGHOLZ, G. "Intelligent Hybrid Systems", Department of Computer Science of Florida State University, Tallahassee, outline do livro, 1991.

[Kelly 55] KELLY, G. The Psichology of Personal Constructs (Norton, New York), 1955.

[Leão 87] LEÃO, B.F., LUCCHESI, S.A. & ROCHA, A.F., "A Methodology Proposal for Knowledge Acquisition", Seventh International Congress of Medical Informatics, p. 1042-50, 1987.

[Leão 88] LEÃO, B.F., "Construção da Base de Conhecimento de um Sistema Especialista de Apoio ao Diagnóstico de Cardiopatias Congênitas". Tese de Doutorado em Cardiologia da Escola Paulista de Medicina, São Paulo, Brasil, 1988.

[MacDonald 89] MACDONALD, B.A. & WITTEN, I.H., "A Framework for Knowledge Acquisition through Techniques of Concept Learning". IEEE Transactions on Systems, Man, and Cybernetics, v.19, n.3, May/Jun 1989.

[Machado 88] MACHADO, R.J. & ROCHA, A.F. "Cálculo da Representação Média do Conhecimento de Múltiplos Especialistas", Anais do V Simpósio Brasileiro de Inteligência Artificial, 1988.

[Machado 89] MACHADO, R.J. & ROCHA, A.F., "Redes Neurais Combinatórias – Um Modelo Conexionista para Sistemas Baseados em Conhecimento", Anais do VI Simpósio Brasileiro de Inteligência Artificial, p. 344-358, 1989.

[Machado 90] MACHADO, R.J., ROCHA, A.F. & LEÃO, B.F. "Calculating the Mean Knowledge Representation from Multiple Experts", in the book Multiperson Decision Making Models Using Fuzzy Sets and Possibility Theory, editors: FEDRIZZI, M. & KACPRZYK, J., Kluwer Academic Publishers, 1990.

[Machado 90b] MACHADO, R.J., ROCHA, A.F., "The Combinatorial Neural Network: Connectionist Model for Knowledge Based Systems", Proceedings of the conference IPMU – Information Processing and Management of Uncertainty in Knowledge Based Systems, p. 9 - 11, Paris, 1990.

[Machado 90c] MACHADO, R.J., ROCHA, A.F., RAMOS, M.P. & GUILHERME, I.R. "Inférence et Interrogation Dans Systèms Experts Connexionists Flous – Inference and Inquiry in Fuzzy Connectionist Expert Systems", Proceedings of COGNITIVA-90, Madrid, Nov., 1990.

[Machado 92] MACHADO, R.J., ROCHA, A.F. & DENIS, F.A.R.M. "Evolutive Learning in Neural Networks", Anais do IX Simpósio Brasileiro de Inteligência Artificial, Rio de Janeiro, 1992.

[McGraw 89] MCGRAW, Karen L. & HARBISON-BRIGGS, Karan. Knowledge Acquisition: principles and guidelines. Prentice-Hall International Editions, 1989. ISBN 0-13-517095-8.

[Michalski 87] MICHALSKI, R.S., "Learning Strategies and Automated Knowledge Acquisition: An Overview". Computacional Models of Learning, Leonard Boole Ed., Springer-Verlag, p. 1-20, 1987.

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

[Mongiovi 90] MONGIOVI, G. & CIRNE F., W.C., "Um Algoritmo Baseado em Conhecimento para Ampliar a Potencialidade do ID3", Anais do VII Simpósio Brasileiro de Inteligência Artificial, Campina Grande, Paraíba, 1990.

[Mongiovi 90b] MONGIOVI, G., ALENCAR, W. & BAENY, X. "Ginecol: Um Sistema de Diagnóstico Ginecológico", Relatório Técnico, Universidade Federal da Paraíba, Depto. de Sistemas e Computação, 1990.

[Mongiovi 91] MONGIOVI, G. & CIRNE F., W.C., "O Uso de Semântica na Construção e Expansão de Árvores de Decisão Indutivas", Anais da XVII Conferencia Latinoamericana de Informática. Caracas, Vanezuela, 1991.

[Norusis 75] NORUSIS, M.J., JACQUES, J.A., "Diagnosis II. Diagnostic Models Based on Attribute Clusters: A Proposal and Comparisons", Computers and Biomedical Research, 8, p.173, 1975.

[Núñes 88] NÚÑES, M. "El Metodo de aprendizaje EG2: Una Aplicacion de Conocimiento de Base a Ejemplos Estructurados", Master Thesis in Knowledge Engineering, Universidad Politecnica de Madrid, 1988.

[Núñes 91] NÚÑES, Marlon. "The Use of Background Knowledge in Decision Tree Induction". Machine Learning 6, p. 231-250, Kluwer Academic Publishers, Boston, 1991.

[Quinlan 83] QUINLAN, J.R., "Learning Efficient Classification Procedures and Their Application to Chess End Games", Machine Learning I: An Artificial Intelligence Approach. Michalsky, Carbonell e Mitchell (eds). Morgan Kauffmann, 1983.

[Rich 83] RICH, E. "Artificial Intelligence", McGraw-Hill, 1983.

[Romaniuk 89] ROMANIUK, S.G. & HALL, L.O., "FUZZNET, A Fuzzy Connectionist Expert System", Technical Report CSE-89-07, Dept. of Computer Science and Engineering, University of South Florida, FL, 1989.

[Rumelhart 86] RUMELHART, D.E., HINTON, G.E. & WILLIAMS, R.J., "Learning Internal Representations by Error Propagation", Parallel Distributed Processing, v. I (MIT Press, Cambridge, MA). D. RUMELHART & J. MCCLELLAND eds., p. 318-362, 1986.

[Saito 88] SAITO, K., & NAKANO, R., "Medical Diagnostic Expert System Based on PDP Model", NTT Communications and Information Processing Laboratories, Yokosuka-shi, Kanagana, Japan, p. 255-262, 1988.

[Samad 88] SAMAD, T., "Towards Connectionist Rule-Based Systems", Proceedings of the Conference on Neural Information Processing Systems, vol. II, p. 525-531, 1988.

[Simpson 90] SIMPSON, P., "Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations", Pergamon Press, 1990.

[Vasco 92] VASCO, J.J.P.F., MONGIOVI, G., CIRNE F., W.C. & DONATO Jr., E.T. "A4 – Ambiente para Auxílio à Aquisição Automática de Conhecimento". Anais do IX Simpósio Brasileiro de Inteligência Artificial, Rio de Janeiro, 1992.

[Vasco 92b] VASCO, J.J.P.F., MONGIOVI, G., CIRNE F., W.C. & DONATO Jr., E.T. "Modelagem do Domínio para Algoritmos Indutivos de Aquisição de Conhecimento". Anais do IX Simpósio Brasileiro de Inteligência Artificial, Rio de Janeiro, 1992.

[Waterman 86] WATERMAN, D.A., "A Guide to Expert Systems", Addison-Wesley, 1986.

[Zadeh 65] ZADEH, L.A., "Fuzzy Sets", Information and Control, 8, p. 338-353, 1965.