**UNIVERSIDADE FEDERAL DE SÃO CARLOS**
**Centro de Ciências Exatas e de Tecnologia**
**Programa de Pós-Graduação em Ciência da Computação**

# Extração de Conhecimento Simbólico de Redes Neurais

**Fábio Seitoku Nagamine**

**Orientação:**
Profa. Dra. Maria do Carmo Nicoletti

*Exame de Defesa apresentado ao Programa de Pós-Graduação em Ciência da Computação, como parte dos requisitos exigidos para obtenção do grau de Mestre em Ciência da Computação*

São Carlos
Janeiro/2005

**Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar**

N384ec

Nagamine, Fábio Seitoku.
Extração de conhecimento simbólico de redes neurais / Fábio Seitoku Nagamine. -- São Carlos : UFSCar, 2005.
149 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Redes neurais. 2. Inteligência artificial. 3. ADT. 4. MACIE. I. Título.

CDD: 006.32 (20ª)

# AGRADECIMENTOS


- a Robert Andrews por ter fornecido o software ELSY, utilizado durante os testes do método RULEX, e a J. Besada-Juez pelos esclarecimentos sobre seu método de extração de regras fuzzy a partir de redes neurais.
- a Maria do Carmo Nicoletti pela orientação, incentivo e paciência.
- aos colegas do DC-UFSCar pelo apoio e amizade, e aos colegas da Motorola, pela compreensão quando esta foi necessária.
- a minha família e, sobretudo, a meus pais, cujo apoio tem sido essencial em todos os momentos.

# RESUMO

O fato das Redes Neurais (RNs) serem incapazes de explicar, de maneira simbólica, as decisões fornecidas ou o conhecimento embutido em suas conexões e arquitetura é uma limitação já bastante conhecida. Este trabalho investiga diversos métodos de extração de conhecimento de RNs propostos na literatura. Mais especificamente, o trabalho concentra-se em quatro diferentes abordagens para extração de conhecimento que são detalhadas, criticadas e, para cada uma delas, discutida uma possível implementação. Adicionalmente, uma taxonomia para classificação de métodos de extração de regras de RNs, encontrada na literatura, é detalhada. A partir dessa taxonomia, são sugeridos refinamentos que visam tornar a taxonomia mais útil, refinada e versátil. O objetivo principal do trabalho é abordar a extração do conhecimento de RNs de uma maneira crítica, analisando cada uma dos quatro métodos, principalmente com relação ao escopo de atuação, limitações e contribuição efetiva para a melhoria da legibilidade e compreensão das RNs.

# *ABSTRACT*

*The fact that Artificial Neural Networks (ANNs) are not able to explain, in a symbolic way, neither their decisions or the knowledge embedded in its connections and architecture is a well-known limitation. This work investigates several methods of knowledge extraction from ANNs proposed in the literature. More specifically, it focuses on four different approaches for knowledge extraction that are detailed and criticized and, for each of them, discusses a possible implementation. Also, a taxonomy for methods of rule extraction from ANNs, found in the literature, is detailed. An extension of this taxonomy aiming at a more useful, refined and versatile version is proposed. The main goal of the work, however, is to approach knowledge extraction from ANN in a critical way, analyzing each of the four methods concerning, mainly, their scopes, limitations and effective contribution to improving readability and easy understanding.*

## SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# CAPÍTULO 1. INTRODUÇÃO

## 1.1 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial (IA) que investiga maneiras de incorporar aprendizado a sistemas computacionais. Existe já um número bastante grande de diferentes modelos e técnicas que permitem que sistemas computacionais incorporem alguma forma de aprendizado (ver [Mitchell–1997], por exemplo).

O modelo de AM conhecido como *aprendizado indutivo a partir de exemplos* é baseado em um processo indutivo que constrói uma expressão que representa o conceito, a partir de um conjunto exemplos do conceito, referenciado como *conjunto de treinamento*. Geralmente as instâncias do conjunto de treinamento são descritas por vetores de pares *atributo-valor_atributo* e de uma classe associada, que representa o conceito ao qual a instância em questão pertence.

O processo de indução do conceito por um sistema automático de aprendizado de máquina pode ser abordado como uma busca em um espaço de hipóteses, visando encontrar aquela(s) que melhor classifica(m) as instâncias do conjunto de treinamento. Essa(s) hipótese(s) são consideradas a representação do conceito sendo aprendido. São chamadas hipóteses porque foram induzidas a partir de conhecimento empírico.

Essas hipóteses podem ser representadas das mais variadas maneiras usando diferentes linguagens de representação (cláusulas de Horn, regras de decisão, árvores de decisão, hiper-retângulos, redes neurais, *clusters* de pontos, etc…). Algumas dessas representações são facilmente expressas em outras, como é o caso, por exemplo, da tradução trivial da representação de hiper-retângulos ou árvores de decisão em regras. A facilidade é conseqüência, também, do fato de que hiper-retângulos, árvores de decisão e regras serem representações simbólicas. A situação é diferente quando representações simbólicas devem ser consideradas para serem 'traduzidas' em representações neurais e vice-versa. O processo nem sempre pode ser realizado de maneira satisfatória, pode ser bastante complicado e é dependente do tipo de rede neural considerado.

Este trabalho de pesquisa investiga o processo de 'tradução' de uma rede neural (RN) para uma representação simbólica, analisando várias propostas existentes na literatura, sob o enfoque das características elencadas pela taxonomia ADT [Andrews et

al.–1995]. O trabalho descreve, analisa, implementa e compara as propostas e os resultados de quatro sistemas que se propõem a extrair conhecimento simbólico a partir de redes neurais, com base na investigação dos quatro métodos e propõe um refinamento da ADT com base nas investigações realizadas.

## 1.2 OBJETIVOS E JUSTIFICATIVAS DA PESQUISA

Dependendo da aplicação, redes neurais podem apresentar vantagens com relação aos métodos de AM simbólicos. Além de permitirem processamento massivamente paralelo, as redes neurais têm demonstrado maior capacidade de generalização e tolerância a falhas [Browne & Sun–1999]. Embora totalmente numérica, a forma de representação de uma RN é compacta e o conhecimento aprendido faz parte da própria rede; esse conhecimento pode ser acessado e usado com facilidade e rapidez.

É fato, entretanto, que RNs são incapazes de explicar, de maneira semelhante aos métodos simbólicos, o processo por meio do qual uma dada decisão ou saída foi obtida, dado que o conhecimento aprendido por uma rede neural está embutido na própria representação da rede, estando codificada:

- na própria arquitetura da rede;
- nas funções de ativação de cada unidade intermediária e de saída e
- no conjunto de pesos numéricos associados às conexões

Isso de certa forma se torna uma deficiência marcante à medida que capacidades humanas para raciocínio algorítmico e abstrações evidenciam a importância do processamento baseado em símbolos [Buchheit–1999]. Experimentos sugerem que mesmo crianças realizam processamento mental baseado em regras [Marcus et al.–1999].

Em alguns domínios e certas aplicações, para que redes neurais ganhem uma aceitação ainda maior e para que se consolidem como uma técnica fundamental de aprendizado, é essencial que uma capacidade 'explicativa' se torne parte integrante de sua funcionalidade. Geralmente essa capacidade 'explicativa' tem sido implementada por meio de técnicas que 'traduzem' a rede neural em um conjunto de regras permitindo, com isso, sua validação por seres humanos.

Existem inúmeras técnicas que buscam explicar o comportamento de redes neurais por meio da extração de regras, árvores de decisão ou relações da rede. A área de pesquisa de extração de regras está expandindo para incluir técnicas para a

representação de conhecimento neural e explicação conexionista baseada em generalização e métodos para a descrição do comportamento de redes neurais recorrentes.

Uma das principais justificativas que suportam o investimento na combinação e/ou integração dos modelos simbólico e conexionista é que, aparentemente, sem que redes neurais incorporem alguma forma de explicação, dificilmente esse modelo de aprendizado poderá ser explorado em todo o seu potencial. Em muitas tarefas de aprendizado é importante obter classificadores que, além de precisos sejam também facilmente inteligíveis por humanos. Redes neurais são limitadas nesse sentido, uma vez que, após o treinamento, são difíceis de serem interpretadas. Contrastando com redes neurais, as soluções encontradas por sistemas de aprendizado simbólico são, na maioria das vezes, mais passíveis de serem compreendidas por humanos.

A pesquisa realizada nesta dissertação de mestrado focaliza a investigação de quatro diferentes abordagens à extração de conhecimento simbólico de redes neurais:

1. a implementada pelo Sistema MACIE (MAtrix Controlled Inference Engine) [Gallant–1988], que é um engenho de inferência independente que interpreta uma base de conhecimento neural;
2. a subsidiada pelo algoritmo FERNN (Fast Extraction of Rules from Neural Networks) [Setiono & Leow–2000] que extrai regras a partir redes neurais podadas sem que seja necessário um retreinamento da rede;
3. a implementada pelo RULEX [Andrews & Geva–2002], que focaliza redes neurais do tipo LC (Local Clusters);
4. a proposta pela TBNN (Tree Based Neural Network) [Ivanova & Kubat–1995], que faz o processo inverso, ou seja, transforma uma representação simbólica em uma rede neural.

A principal justificativa para a escolha do MACIE se deve ao fato desse sistema ter sido usado com sucesso e ter se tornado disponível comercialmente; a escolha do FERNN se deve ao fato de tal proposta ter como principal característica a rapidez com que regras são extraídas; a do RULEX devido à sua abordagem relativamente nova que faz uso de um novo modelo de redes neurais que pode ser considerado uma generalização do modelo RuleNet, proposto em [Tschichold-Gurman–1997] e a da TBNN por abordar o problema da maneira inversa.

Outro critério que subsidiou a escolha das propostas foi a caracterização do tipo de conhecimento simbólico que tais métodos extraem, de acordo com a taxonomia ADT utilizada neste trabalho, que foi proposta para classificar os métodos de extração de regras de redes neurais.

O MACIE extrai regras de produção a partir dos pesos das conexões sinápticas de cada unidade individual (neurônio). O RULEX extrai regras de intervalo a partir das funções de ativação de cada agrupamento de neurônios. O FERNN extrai regras M-de-N a partir da ativação dos neurônios da camada intermediária e dos exemplos de treinamento corretamente classificados. O TBNN gera uma árvore de decisão.

Pela taxonomia ADT, apresentada e discutida no Capítulo 2, os quatro métodos têm translucidez decomposicional[1] e regras no formato proposicional sendo, segundo tais critérios, classificados no mesmo grupo. Os quatro métodos, porém, funcionam de modo bastante diferenciado um do outro e geram regras bastante diferentes. Esses fatos sugerem a existência de critérios para caracterização de tais sistemas outros que os propostos pela ADT. A determinação e estabelecimento desses outros critérios foram também objetivos da pesquisa desenvolvida neste trabalho.

O estudo detalhado dos métodos elencados anteriormente pode, portanto, evidenciar que critérios adicionais (tais como o critério "origem das regras extraídas" discutido no Capítulo 2) podem conferir à taxonomia ADT um maior grau de refinamento e, conseqüentemente, fornecer um esquema de classificação que permita uma melhor comparação de métodos de extração de regras de redes neurais.

## 1.3 ORGANIZAÇÃO DO TRABALHO

Esse trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta e discute a taxonomia de caracterização de técnicas de extração de conhecimento simbólico a partir de redes neurais, conhecida como ADT [Andrews et al.–1995], uma vez que um dos objetivos deste trabalho é evidenciar a necessidade da incorporação de critérios adicionais à ADT, de maneira a refiná-la. Os quatro capítulos seguintes tratam da apresentação e discussão das principais idéias e conceitos que subsidiam o MACIE, o FERNN, o RULEX e o TBNN, respectivamente. Em cada capítulo é apresentado, discutido e avaliado o desempenho do método correspondente em dois domínios. Foram

[1] Translucidez é um dos critérios de avaliação da ADT. Decomposicional significa que o método analisa a RN a partir de suas unidades individuais. Ver Capítulo 2.

preservadas as notações originais usadas nas propostas de cada método com o objetivo de facilitar a referência às suas propostas originais. O Capítulo 7 propõe alterações à taxonomia ADT, visando torná-la mais adequada à tarefa de descrição e classificação dos métodos de extração de regras de RNs.

O objetivo do trabalho é apresentar, em cada um desses capítulos, uma implementação do sistema correspondente, resultados experimentais em domínios de dados e possíveis refinamentos a serem incorporados à ADT derivados da investigação/caracterização das quatro abordagens.

A proposta inicial deste trabalho era a de utilizar um mesmo exemplo para ilustrar a idéia/implementação de cada método pesquisado e, assim, viabilizar elementos de comparação entre eles. Alguns métodos, entretanto, são projetados para resolver um tipo específico de problema, o que torna difícil propor um exemplo neutro, onde nenhum método possa ser favorecido ou prejudicado. Este é o caso do MACIE, por exemplo, que apresenta diversas vantagens de uso quando articulado a sistemas especialistas com valores discretos mas que, entretanto, é bem menos interessante em problemas que apresentam valores contínuos.

Devido a esse problema decidiu-se pela apresentação de quatro exemplos distintos, de maneira a contemplar as funcionalidades e peculiaridades para cada método. Adicionalmente, entretanto, a idéia original será mantida, por meio da discussão de um exemplo bem simples, será utilizado para comparação direta dos quatro métodos.

A idéia é de, inicialmente, apresentar um exemplo específico para explorar a funcionalidade de cada método no domínio de problema para o qual foi proposto e, a seguir, discutir o seu uso no tratamento de um problema simples. É óbvio que, para o tratamento do problema comum simples, será necessário fazer alterações no formato das entradas e saídas relativas à implementação de cada um dos métodos. Por esse motivo, na comparação será dada ênfase ao processo central de extração de regras em detrimento de outras características do método.

# CAPÍTULO 2. A TAXONOMIA ADT E SEU USO NESTE TRABALHO

Pode ser evidenciado na literatura um número crescente de métodos que, implementando as mais diversas estratégias, buscam explicar redes neurais, de forma simbólica. Neste trabalho, para contextualizar a pesquisa sendo desenvolvida, optamos por discutir as principais propostas na área, usando uma metodologia de caracterização de tais técnicas, com o objetivo de poder abordá-las sistematicamente e estabelecer um conjunto de critérios que podem ser usados para avaliá-las e compará-las.

A primeira seção deste capítulo descreve detalhadamente a metodologia ADT e as informações foram extraídas de [Andrews et al.–1995] e [Andrews & Tickle–1998]; a segunda seção discute como a ADT foi utilizada no trabalho.

## 2.1 A TAXONOMIA ADT

A ADT (Andrews-Diederich-Tickle) é uma taxonomia de caracterização de técnicas de extração de conhecimento simbólico a partir de redes neurais que evoluiu a partir da proposta descrita em [Andrews et al.–1995]. Essa proposta descreve uma taxonomia para categorização de uma ampla variedade de técnicas desenvolvidas para extração de regras a partir de redes neurais. Essa taxonomia estabelece cinco critérios principais para a classificação de técnicas de extração, que são descritos em detalhes nas próximas subseções. São eles:

- poder de expressão das regras extraídas (também referenciado como formato das regras);
- a qualidade das regras extraídas;
- a translucidez com que o método aborda as unidades da RN;
- a complexidade algorítmica da técnica de extração/refinamento de regras;
- a medida de portabilidade do método de extração de regras a várias arquiteturas de RN.

### 2.1.1 O PODER DE EXPRESSÃO DAS REGRAS EXTRAÍDAS

Esse critério caracteriza a representatividade da expressão simbólica do conceito extraído da RN. Para a especificação deste critério, a ADT sugere o agrupamento das

técnicas em três diferentes grupos. As técnicas são então agrupadas em função da representação do conceito extraído ser apresentado como regras:

(1) Simbólicas convencionais ou seja, proposicionais;

(2) Baseadas em conjuntos e lógica *fuzzy*; ou

(3) Simbólicas de primeira ordem (com quantificadores e variáveis).

### 2.1.2 A QUALIDADE DAS REGRAS EXTRAÍDAS

O conjunto de critérios adotado pela ADT para avaliar a qualidade das regras é uma extensão do conjunto proposto em [Towell & Shavlik–1993] e inclui:

(1) Acurácia;

(2) Fidelidade;

(3) Consistência e

(4) Compreensibilidade das regras extraídas.

Em [Towell & Shavlik–1993] qualidade é avaliada apenas em termos de acurácia e fidelidade. A característica de compreensibilidade é tratada como uma característica individual. Muito embora a discussão sobre a qualidade das regras extraídas contida em [Towell & Shavlik–1993] focaliza apenas o sistema KBANN, os comentários dos autores são gerais o suficiente para serem aplicados a qualquer outro sistema de extração/tradução de conhecimento:

> "Como uma medida, qualidade é, pelo menos, bidimensional. Primeiro, as regras devem caracterizar precisamente exemplos que não participaram do treinamento. Se as regras perdem a vantagem em acurácia que o KBANN tem sobre métodos de aprendizado simbólico, então extração de regras tem pouco valor. Seria mais simples usar um método totalmente simbólico para criar as regras e usar o KBANN apenas para criar classificadores altamente precisos e não inteligíveis (no sentido de não serem passíveis de tradução simbólica[2]). Segundo, as regras extraídas devem expressar a informação contida na RN. Isso é necessário se as regras extraídas vão ser usadas para o entendimento do que a RN aprendeu durante o treinamento."

[2] A informação entre parênteses não faz parte da citação e foi introduzida para esclarecer melhor o comentário dos autores.

Segue uma breve descrição de cada uma das cinco características que, juntas, expressam a qualidade das regras extraídas.

(1) Acurácia

Um conjunto de regras é considerado acurado se consegue classificar corretamente um conjunto de exemplos do domínio do problema que não foram usados para o treinamento da RN. Nos experimentos descritos em [Towell & Shavlik–1993], para a avaliação da acurácia do sistema KBANN foi usada a média dos resultados obtidos como dez repetições de uma 10-validação cruzada.

(2) Fidelidade

Um conjunto de regras tem um alto grau de fidelidade se consegue imitar o comportamento da RN da qual foi extraído, fornecendo as mesmas respostas que a RN para um mesmo conjunto de exemplos. De acordo com [Towell & Shavlik–1993], fidelidade é importante em duas circunstâncias: (a) se as regras extraídas vão ser abordadas como uma ferramenta para o entendimento do comportamento da rede e (b) se as regras extraídas vão ser usadas como um método de transferência do conhecimento contido na rede.

(3) Consistência

Um conjunto de regras é considerado consistente se, mesmo em diferentes sessões de treinamento, a RN gera conjuntos de regras que produzem as mesmas classificações, em conjuntos de exemplos não usados no treinamento.

(4) Compreensibilidade

Um conjunto de regras extraídas de uma RN deve não apenas ser acurado, mas também inteligível. Embora seja um conceito relativamente vago, existem várias maneiras de 'medir' compreensibilidade. Uma abordagem é olhar para medidas estatísticas que descrevem todo o conjunto de regra; outra alternativa é examinar se cada uma das regras é inteligível. Compreensibilidade de um conjunto de regras pode também ser abordada como sendo o número de regras do conjunto e o número de antecedentes por regra. É óbvio que quanto maior o número de regras extraídas, menor é compreensibilidade de todo o conjunto. A mesma observação vale para a avaliação de regras individuais, no que diz respeito ao número de

condições que devem ser consideradas para o estabelecimento (ou não) da conclusão da regra.

### 2.1.3 A TRANSLUCIDEZ (GRANULARIDADE) DAS REGRAS EXTRAÍDAS

O terceiro critério de classificação é baseado na granularidade com que a RN é analisada pela técnica de extração de regras, ou seja, o quanto a técnica de extração leva em consideração os elementos da RN explicitamente (analisando unidades, pesos, etc) ou implicitamente (considerando apenas as saídas determinadas por esses elementos).

Os autores de [Andrews et al.–1995] usam o termo 'translucidez' para descrever o grau de granularidade e propõem três abordagens chamadas de (a) decomposicional, (b) pedagógica e (c) eclética, como pontos de referência no conjunto de granularidades perceptíveis.

(a) Decomposicional

Técnicas que se posicionam em uma das extremidades do espectro de possíveis granularidades. Essas técnicas abordam a RN em seu nível mais refinado de granularidade, ou seja, no nível de unidades individuais (neurônios intermediários e de saída). Primeiro extraem regras no nível das unidades individuais da rede e, então, agregam essas regras para formar relações globais (sendo relações globais, o conjunto de regras locais que se relacionam para determinar o comportamento da RN).

(b) Pedagógico

Técnicas que se posicionam na extremidade oposta à das técnicas decomposicionais. Tratam a RN como uma ‘caixa preta’, extraindo as relações globais entre entradas e saídas da rede diretamente, sem analisar as características que levam a rede a apresentar uma solução.

(c) Eclético

Esse ponto de referência foi criado, inicialmente, para acomodar a técnica DEDEC [Tickle et al.–1997] devido às suas características híbridas. Pode ser usado, entretanto, para caracterizar técnicas híbridas que analisam a RN em nível de unidade individual, mas que extraem regras em nível global.

Em [Andrews & Tickle–1998], os autores propõem a extensão do critério de translucidez com a adição de uma classificação adicional, denominada "Composicional", para designar métodos que trabalham sobre grupos de neurônios da RN. Essa classificação engloba técnicas como o RULEX, apresentado em [Andrews & Geva–2002], que não analisam unidades individuais nem o comportamento global, não podendo ser agrupadas, portanto, em nenhuma das três classificações originais.

### 2.1.4 A COMPLEXIDADE ALGORÍTMICA

O quarto critério da taxonomia ADT para categorização de técnicas de extração de regras de RNs é a complexidade do algoritmo central que extrai as regras. Essa medida é levada em consideração em vários métodos (ver por exemplo [Fu–1991] [Fu–1994] [Craven & Shavlik–1994] [Thrun–1995]). Na proposta do método KT [Fu–1994], por exemplo, é feito um estudo de complexidade sustentando a necessidade da aplicação de heurísticas para redução do espaço de busca. Na maioria dos trabalhos, porém, os autores não mencionam qualquer preocupação com a complexidade.

### 2.1.5 A PORTABILIDADE DA TÉCNICA DE EXTRAÇÃO

Existem diversas técnicas de extração de regras de RNs que funcionam apenas em arquiteturas específicas ou, então, que necessitam que a rede tenha um treinamento diferenciado de modo a facilitar o trabalho de extração das regras [Andrews & Geva–2002] [Gallant–1988] [Hatzilygeroudis & Prentzas–2001] [Setiono & Leow–2000]. Por outro lado, técnicas como TREPAN [Craven & Shavlik–1996] foram desenvolvidas levando em consideração a portabilidade.

Considerando a relevância dessa característica, o quinto critério da ADT para categorização de técnicas de extração de regras de RNs é a descrição da portabilidade ou seja, se a técnica é ou não portável e para quais arquiteturas.

## 2.2 O USO DA ADT NESTE TRABALHO

A ADT tem sido descrita e usada em diversos trabalhos (ver por exemplo [Andrews & Tickle–1998] [Tickle et al.–1997] [Darbari–2000]) como uma taxonomia para classificação de técnicas de extração de regras de RNs.

Os trabalhos que usam a ADT para sistematizar a análise/avaliação de técnicas de extração de regras a partir de RN não utilizam uma abordagem padrão para o seu uso. Em [Tickle et al.–1997] é sugerida uma extensão para ampliar o escopo da taxonomia, permitindo classificar técnicas que extraem representações diferentes de regras (tais como árvores de decisão e autômatos finitos), além de modelos híbridos que utilizam RNs para refinar conhecimento simbólico pré-existente.

Porém, ao ser apenas estendida no sentido de se tornar mais abrangente, a ADT se mostra uma taxonomia generalista, que agrupa em uma mesma categoria técnicas que, na prática, apresentam muitas diferenças.

O uso da ADT na caracterização das quatro abordagens de extração de conhecimento tratadas neste trabalho focaliza, principalmente, a identificação das características da metodologia que precisam ser refinadas e estendidas de maneira a torná-la mais abrangente e, ao mesmo tempo, mais precisa, realista e útil para a caracterização das técnicas de extração de conhecimento a partir de RNs.

# CAPÍTULO 3. O SISTEMA MACIE

A investigação do sistema MACIE (MAtrix Controlled Inference Engine) [Gallant−1988] foi motivada pelo fato do sistema apresentar uma abordagem mais simples que a de outros sistemas simbólico-conexionistas, resolvendo problemas no domínio ao qual se aplica com pouco acréscimo de custo computacional. Além disso, também, por ser um sistema que subsidiou o desenvolvimento de um produto seu similar, chamado KnowledgeNet, disponível comercialmente (HNC Inc. – San Diego – California). Segundo Gallant, o MACIE foi construído como um engenho de inferência independente, com o objetivo de interpretar a base de conhecimento de um sistema especialista conexionista. Um sistema especialista conexionista é um sistema especialista cuja base de conhecimento é uma rede neural [Gallant−1988].

A interpretação fornecida pelo MACIE é dada na forma de regras, extraídas a partir da rede neural, que justificam uma inferência feita pelo sistema especialista conexionista. A técnica empregada pelo MACIE é bastante particular e aplicável a apenas uma situação específica. Além disso, trabalha com um modelo conexionista simples, conhecido como rede discriminante linear, que não permite *feedback* e cujos pesos são inteiros positivos. Ainda assim, é uma proposta pioneira que apresenta grande contribuição para o desenvolvimento de abordagens colaboracionistas [Nicoletti−1998/2000]. Propostas mais modernas, como a do sistema HYMES [Hatzilygeroudis & Prentzas−2001], têm sido apresentadas, inspiradas no sistema MACIE.

## 3.1 INTRODUÇÃO

Gallant [Gallant−1988] desenvolveu um sistema que permite a construção de sistemas especialistas conexionistas a partir de exemplos de treinamento. O uso de métodos de aprendizado a partir de exemplos de treinamento facilita a fase de construção da base de conhecimento do sistema especialista; a capacidade de generalização da RN resolve de modo elegante o problema de tratamento de informações incompletas, ambos problemas difíceis de serem tratados em sistemas especialistas tradicionais (aqueles com base de conhecimento formada por regras simbólicas adquiridas de um especialista humano).

O sistema proposto por Gallant é constituído por dois subsistemas; o primeiro deles é um gerador de base de conhecimento neural, que disponibiliza várias técnicas de aprendizado neural. O segundo, identificado como MACIE, é um engenho de inferência independente que interpreta tais bases de conhecimento. Como a rede neural é representada internamente por uma matriz de pesos, esse fato serviu para batizar o engenho de inferência como MACIE – MAtrix Controlled Inference Engine. O MACIE usa a rede neural para:

- realizar inferências baseadas em informação parcial;
- descobrir, dentre as variáveis de entrada com valores desconhecidos, as mais promissoras para realizar determinadas inferências;
- criar justificativas para as inferências que o engenho realiza.

É justamente quando da criação de justificativas para as inferências, que um processo de 'tradução' da inferência realizada pela rede, para regras, acontece. Como é um processo que traduz informação neural em simbólica, o algoritmo que subsidia esse processo é o foco da análise realizada neste trabalho.

A seguir são apresentados os dois subsistemas: o gerador de base de conhecimento neural, e o MACIE, respectivamente.

## 3.2 O GERADOR DE BASE DE CONHECIMENTO NEURAL

O gerador de base de conhecimento neural é o subsistema que gera a RN utilizada como base de conhecimento pelo sistema conexionista. Esse subsistema assume como entrada diversas informações sobre o domínio do problema a ser tratado, tais como os nomes das variáveis de interesse e as relações entre elas, além de um conjunto de exemplos de treinamento.

A partir das informações sobre o domínio do problema é definida a arquitetura inicial da RN. Definida a arquitetura, o sistema disponibiliza uma variedade de algoritmos de aprendizado para que a RN inicial seja treinada de modo a 'aprender' a informação contida nos exemplos de treinamento. Após o treinamento, a rede, pesos, nomes e interrogações relativas às variáveis constituem a base de conhecimento do sistema especialista conexionista, a partir da qual o MACIE realizará as inferências e

demais operações[3]. A seguir são detalhados o modelo e o processo de geração da RN que servem de base de conhecimento do sistema.

### 3.2.1 O MODELO CONEXIONISTA

Em [Gallant–1993] o autor discute a possibilidade de serem usadas redes do tipo "o vencedor leva tudo" e redes com valores contínuos com função de ativação sigmoidal, como base de conhecimento de sistemas especialistas conexionistas. São citadas, entretanto, diversas restrições ao uso desses modelos, como a determinação de limiares para determinação de valor verdadeiro/falso para uso de redes com valores contínuos e o uso de cálculos adicionais para determinação da saída para uso de redes "o vencedor leva tudo". Por essa razão, o gerador de base do conhecimento neural e, conseqüentemente, o MACIE, trabalha, idealmente, com modelos conexionistas simples (Figura 3.1), que têm as seguintes propriedades:

> **rede**: *feedforward* (ciclos não são permitidos). Pesos ($w_{ij}$) são inteiros positivos ou negativos. Desde que não existem ciclos, unidades podem ser indexadas de maneira que se um arco direcionado sai da unidade $u_j$ e chega à unidade $u_i$, então $i > j$. A numeração de unidades adotada é consistente com essa propriedade. A rede pode ser convenientemente expressa usando uma matriz de pesos W, onde $w_{ij} = 0$ se não existir arco conectando $u_j$ com $u_i$. Os valores $w_{i,0}$ representam os valores dos *biases*. O esquema de numeração adotado para redes *feedforward* implica $w_{ij} = 0$ para $j \geq i$ (conexões somente com neurônios da próxima camada).
>
> É importante notar que essa notação é usada apenas nos algoritmos envolvidos no MACIE. Como comentado anteriormente, este trabalho preserva a notação original de cada um dos métodos com o objetivo de facilitar a referência aos documentos originais.
>
> **unidades**: ativações de unidades são discretas, com os possíveis valores de +1, -1 ou 0. Esses valores correspondem aos valores lógicos de *verdade*,

[3] Como será visto na Seção 3.3, não apenas inferências, mas também diversas outras operações são realizadas com base na informação contida na RN.

*falso* ou *desconhecido*, respectivamente. A unidade $u_i$ calcula sua nova ativação $u_i$' como a função discriminante linear:

$$S_i = \sum_{j \geq 0} w_{ij} u_j$$

$$u_i' = \begin{cases} +1 \text{ ou verdade se } S_i > 0 \\ -1 \text{ ou falso se } S_i < 0 \\ 0 \text{ ou desconhecido se } S_i = 0 \end{cases}$$

Tais unidades são também chamadas de unidades lógicas de limiar ou células McCulloch-Pitts. Por convenção, a unidade $u_0$ serve de *bias* e sempre tem ativação +1 e unidades as $u_1,\ldots,u_p$ que são identificadas como entradas, têm seus valores estabelecidos externamente.

**pesos**: inteiros positivos ou negativos atuam reforçando (pesos positivos) ou inibindo (pesos negativos) a ativação dos nós.

**dinâmica**: uma iteração da rede consiste na reavaliação de cada unidade, na ordem de seus índices e mudança de seu valor de ativação antes que a próxima unidade seja reavaliada. Uma iteração apenas é suficiente para conduzir a rede a um estado estável, devido à ausência de ciclos e ao esquema de indexação.

O modelo de rede apresentado é um dos modelos conexionistas mais simples possíveis, uma vez que não permite *feedback* e os pesos da rede são valores inteiros.

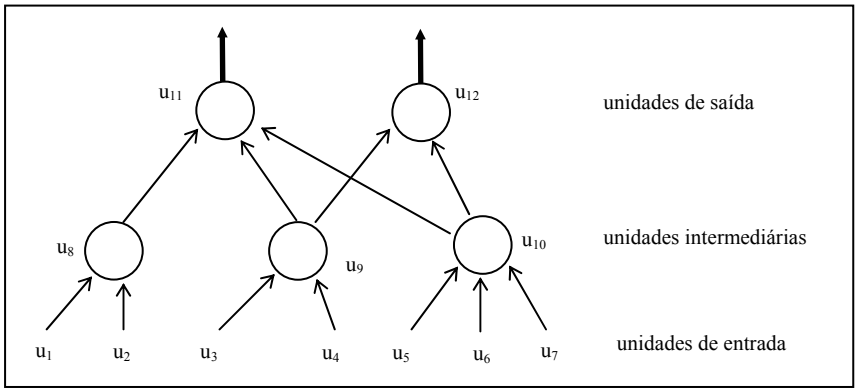



Figura 3.1 Rede neural *discriminante linear*

### 3.2.2 A GERAÇÃO DA REDE: ENTRADAS

Para a geração de uma base de conhecimento conexionista, as seguintes informações devem ser fornecidas:

- nome de cada unidade que corresponde às variáveis de interesse;
- como deve ser a pergunta (qual o formato) que o sistema deve fazer ao usuário, para 'coletar' os valores relativos a cada uma das variáveis de entrada;
- informação de dependência para variáveis intermediárias e variáveis de saída – a informação de dependência é opcional dado que toda unidade de saída pode ser especificada como dependente de cada uma das unidades de entrada. A informação de dependência especifica a rede de dependências, consistindo de um arco de $u_j$ a $u_i$, para todo nó $u_j$ na lista de dependências de $u_{i;}$
- conjunto de exemplos de treinamento.

A partir dessas informações, a arquitetura inicial da RN pode ser definida segundo o algoritmo apresentado na Figura 3.2.

É importante observar aqui que a rede formada a partir da informação de dependência pode não ser capaz de modelar perfeitamente um dado conjunto de exemplos de treinamento uma vez que não é toda função booleana que pode ser representada por meio de uma simples discriminação linear. Se uma função booleana pode ser calculada por uma única unidade, é chamada de *função separável*; caso contrário, é uma *função não separável*. Entretanto, é possível adicionar unidades intermediárias para formar uma rede final que seja capaz de modelar qualquer conjunto consistente de exemplos de treinamento. Uma maneira de fazer isso é adicionar unidades com pesos aleatórios (por exemplo, inteiros entre –5 e +5), como proposto em [Gallant–1993]. Uma vez que unidades adicionadas têm os pesos das entradas fixos, escolhidos aleatoriamente, a adição de unidades se torna uma tarefa bastante fácil.

```
Procedimento: DefiniçãoDaArquiteturaDaRN
entrada:
   ϑ = Conjunto de variáveis do problema
   Δ = Lista de dependências
início
   para cada variável v_i ∈ ϑ faça
   inicio
      se (v_i é variável a ser questionada ao usuário ) então
         Crie um nó N na camada de entrada da RN
      senão
         se (v_i é uma saída a ser apresentada ao usuário ) então
            Crie um nó N na camada de saída da RN
         senão
            Crie um nó N na camada intermediária da RN
         fimSe
      fimSe
      N recebe o nome de v_i
   fim

   para cada dependência (v_i => v_j ) em Δ faça
   inicio
      u_i ← unidade da RN com o nome da variável v_i
      u_j ← unidade da RN com o nome da variável v_j
      Crie uma conexão γ entre u_i e u_j
   fim
fim
```

Figura 3.2 Pseudocódigo do algoritmo para definição da arquitetura inicial da RN

### 3.2.3 A GERAÇÃO DA REDE: APRENDIZADO

Após a configuração final da RN ter sido definida via matriz de dependências (com a adição possível de unidades intermediárias), métodos conexionistas de aprendizado são usados para a geração dos pesos e dos *biases* correspondentes. O sistema disponibiliza uma variedade de algoritmos, tais como o *Pocket Algorithm* [Gallant–1988], *Backpropagation* [Rumelhart et al.–1986] e *Boltzmann Machines* [Rumelhart et al.–1986].

Para treinamento de redes cuja arquitetura possa ser definida com sucesso a partir da matriz de dependências, Gallant recomenda o uso do *Pocket Algorithm* para treinamento de nós individuais que representem funções booleanas linearmente separáveis, e o uso de algoritmos de aprendizado para redes MLP (Perceptron Multicamadas [Rumelhart et al.–1986]), como o *Backpropagation,* para treinamento de grupos de nós que representam funções booleanas não-separáveis.

O treinamento de nós individuais da RN é possível porque, caso a arquitetura seja definida com sucesso por meio da matriz de dependências, a saída esperada para cada

nó pode ser obtida dos exemplos de treinamento e, portanto, pode-se considerar a RN como sendo um conjunto de várias redes Perceptrons e MLP interligadas. Mesmo quando a informação de dependência não permite definir perfeitamente a arquitetura da RN, seu uso facilita o trabalho de aprendizado, pois na medida em que as conexões da RN são restritas de modo a conectar somente nós que representam conceitos dependentes, reduz-se o número de conexões a serem treinadas.

## 3.3 O MACIE

O subsistema MACIE é o engenho de inferência do sistema especialista conexionista proposto por Gallant. Esse subsistema é responsável pela inferência das respostas, determinação das perguntas a serem feitas ao usuário e justificativa das perguntas e inferências realizadas. A seguir são apresentados os algoritmos empregados pelo MACIE, em especial, o algoritmo para justificativa da inferência, que realiza a extração de regras da RN.

### 3.3.1 O PROCESSO DE INFERÊNCIA DO MACIE

O processo de inferência realizado pelo MACIE consiste basicamente em deixar a RN processar as entradas do usuário e, então, apresentar ao usuário a resposta da rede. O modelo conexionista utilizado permite que as entradas no formato "sim/não" sejam diretamente convertidas em valores de entrada 1 ou –1 da RN e, da mesma forma, as saídas 1 ou –1 da RN sejam diretamente mapeadas em confirmações ou negações de conceitos do problema, respectivamente.

Além disso, a capacidade da RN de tolerância à falhas permite que variáveis desconhecidas ou indeterminadas[4], representadas por meio da atribuição de valores 0 (zero) às entradas, sejam processadas juntamente com as variáveis com valores conhecidos. Gallant caracteriza a atribuição dada às variáveis desconhecidas/indeterminadas como sendo um *default* elegante já que, diferente do que ocorre em sistemas especialistas tradicionais, nenhum processamento adicional é necessário para o tratamento dessas variáveis.

Como o sistema trabalha com informação parcial, nem sempre as entradas fornecidas podem ser suficientes para permitir a inferência de alguma resposta. Quando

[4] Variável desconhecida é aquela para a qual o usuário ainda não forneceu um valor, mas que poderá fazê-lo, quando solicitado pelo sistema; uma variável é indeterminada quando o usuário não sabe ou não quer informar seu valor.

esse for o caso, o MACIE evidencia qual, dentre as variáveis desconhecidas, precisa saber o valor e, então, solicita essa informação do usuário. O MACIE usa o conceito de *valor de confiança* (equação 3.1) para determinar a variável cujo valor irá (mais provavelmente) permitir que uma inferência possa ser realizada. Existem várias heurísticas para a determinação da confiança de nó $u_i$ ($conf(u_i)$). Uma das mais simples é:

- se a ativação de $u_i$ é conhecida, $conf(u_i)$ = ativação de $u_i$
- se $u_i$ é unidade de entrada com valor desconhecido, $conf(u_i) = 0$
- para qualquer outro nó cuja ativação tem valor desconhecido, $conf(u_i)$ é calculada por ordem do índice do nó como:

$$conf(u_i) = \frac{\sum_{j=0}^{i-1} w_{ij} conf(u_j)}{\sum_{j:u_j \text{ desconhecido}} |w_{ij}|} \qquad (3.1)$$

O algoritmo apresentado na Figura 3.3 utiliza o cálculo do valor de confiança para determinação da variável que mais contribui para a ativação de alguma saída.

**Procedimento:** DefiniçãoDaEntradaMaisPromissora
**Entradas:**
E = conjunto das unidades de entrada da RN
Σ = conjunto das unidades de saída da RN
$\Phi = u_i \mid u_i \in \Sigma \text{ e } \sigma(u_i) = 0 \}$ /*o conjunto das unidades de saída não ativadas*/
$S_{max} = u_i \in \Phi \mid \forall\ u_j \in \Phi, conf(u_i) > conf(u_j)\ \ j \neq i$ /* a unidade de saída com maior conf */
**Inicio**
$u_{corrente} \leftarrow S_{max}$
**Faça**
$\Psi \leftarrow$ conjunto das unidades de entrada de $u_{corrente}$
$w_x \leftarrow$ peso da conexão entre as unidades $u_x$ e $u_{corrente}$
$\psi_{max} \leftarrow u_i \in \Psi / w_i > w_j\ \ \forall\ j \neq i$ /* entrada de $u_{corrente}$ que mais contribui na ativação */
$u_{corrente} \leftarrow \psi_{max}$
**Ate que** ($u_{corrente} \in E$ )
**retorne** $u_{corrente}$
**Fim**

Figura 3.3 Pseudocódigo do procedimento para definição da entrada mais promissora à conclusão de uma inferência

Além da inferência e definição das perguntas a serem feitas ao usuário, a informação contida na RN pode ser utilizada para fornecer justificativas ao usuário. O MACIE é capaz de fornecer justificativas sobre as inferências realizadas e sobre as perguntas feitas ao usuário. O algoritmo para justificativa das perguntas feitas ao usuário (Figura

3.4) é muito simples e aproveita os valores de confiança, calculados para determinação da pergunta efetuada.

```
Procedimento: JustifiquePergunta(u_q)
Entradas:
   Ξ = conjunto dos nós u_corrente utilizados no procedimento
         "DefiniçãoDaEntradaMaisPromissora"

   Σ = conjunto das unidades de saída da RN
   u_q = nó da RN relativo à pergunta feita ao usuário

Início
   Escreva "u_q é um fator:"
   Ξ ← Ξ – {u_q}
   u_t ← u_q
   Enquanto  Ξ – Σ <> ∅ faça   /* enquanto existem variáveis que não são de saída  */
   Inicio
      Seja u_i | u_i ∈ Ξ  e  w_ti <> 0
      Se (w_t,i > 0) então
         escreva "que dá suporte positivo à determinação de u_i"
      senão
         escreva "que dá suporte negativo à determinação de u_i"
      FimSe
      Ξ ← Ξ – {u_i}
      u_t ← u_i
   Fim

   Para u_s | u_s ∈ Ξ  faça
      escreva "e u_s é a saída mais promissora a ser deduzida"
   FimPara
Fim
```

Figura 3.4 Pseudocódigo do procedimento para justificativa das perguntas feitas ao usuário

Na Figura 3.5 é apresentado um trecho traduzido e adaptado de um exemplo apresentado por Gallant para mostrar o formato da justificativa fornecida.

O exemplo original é um sistema especialista para diagnóstico de doenças. Biramibio e Posiboost estão entre os possíveis diagnósticos sugeridos pelo sistema (mais detalhes sobre esse exemplo podem ser obtidos em [Gallant–1993]).

Segundo Gallant, a justificativa dada ao usuário não é muito esclarecedora, mas ela vem 'de graça', já que os valores utilizados no procedimento de justificativa são obrigatoriamente calculados na escolha das perguntas. Um outro tipo de justificativa fornecida pelo MACIE é a justificativa sobre a inferência realizada, cujo algoritmo é apresentado na próxima seção.

```
O paciente sofre de queda de cabelo?
   ➔ s)im
   ➔ n)ao
   ➔ d)esconhecido
   ➔ ?)explique
   ➔ i)informação sobre as variáveis
?

'queda de cabelo' é um fator
que dá suporte negativo a 'Biramibio'
que dá suporte positivo à 'variável intermediária 2 de Posiboost'
que dá suporte negativo à 'Posiboost'
e 'Posiboost' é a saída mais promissora a ser deduzida"
```

Figura 3.5 Exemplo que mostra justificativa dada pelo MACIE para uma pergunta feita ao usuário

### 3.3.2 O ALGORITMO PARA A JUSTIFICATIVA DA INFERÊNCIA

Dado um conjunto de unidades que estão conectadas a uma determinada unidade, o algoritmo de justificativa da inferência produz como saída uma regra que busca refletir a inferência subsidiada por aquelas unidades. O primeiro nó passado para o algoritmo deve ser um nó de saída e, a partir de sua justificativa, outras justificativas seguem recursivamente até que se chegue a um nó de entrada.

Para a extração da regra o algoritmo usa uma ordenação dos atributos (associados às unidades) que é estabelecida tendo como critério à contribuição que os atributos e o valor absoluto dos pesos de suas conexões têm na inferência. A Figura 3.6 apresenta o pseudocódigo do algoritmo.

O algoritmo descrito na Figura 3.6 está baseado no conceito de *variável contribuinte*; uma variável contribuinte é caracterizada como uma variável que não modifica a 'direção' da soma ponderada. Isto pode ser visto mais facilmente considerando as possíveis situações exibidas na Tabela 3.1. Note que, quando da criação da regra que reflete a ativação +1 ($C^i = +1$), ou –1 ($C^i = -1$) deverão ser considerados apenas aqueles produtos $w_{i,j}u_j$ tais que $C^i w_{i,j} u_j \geq 0$.

**Procedimento:** Justifique($u_j$) /* justificativa da inferência para o nó $u_j$ da RN */

**Entradas:**

$u_j$ = nó com ativação 1 ou -1

E = conjunto das unidades de entrada da RN

$E_{relevante}$ = {} /* conjunto das entradas necessárias para ativação de $u_j$ */

$\varepsilon = \{u_i \mid \exists$ conexão $u_i => u_j\}$ */ as unidades de entrada de $u_j$ */

$\Omega = \{w_{ij} \mid u_i \in \varepsilon\}$ /* conjunto dos pesos de entrada da unidade $u_j$ */

**Inicio**

**Se** ($u_j \in$ E) **então Fim**

**senão**

**Início**

$S_{conh} \leftarrow u_{0i}$ /*soma das entradas conhecidas de $u_j$ é iniciada com o *bias* */

$S_{desc} \leftarrow \sum_{u_i \in \varepsilon} | w_{ij} |$ /* valor máximo das entradas desconhecidas de $u_j$ */

**Enquanto** ($S_{conh} < S_{desc}$) **faça**

**Inicio**

$u_m \leftarrow u_x \mid$ contrib($u_x$) > 0 e contrib($u_x$) > contrib($u_y$) $\forall\ u_y \in \varepsilon$

$V_{max} \leftarrow$ saída($u_m$) * $w_{ij}$ /* valor da entrada que mais contribui */

$S_{conh} \leftarrow S_{conh} + V_{max}$

$E_{relevante} \leftarrow E_{relevante} \cup \{u_m\}$

$S_{desc} \leftarrow S_{desc} - w_{mj}$

$\varepsilon \leftarrow \{u_i \mid \exists$ conexão $u_i => u_j$ e $u_i \neq u_m\}$ /* retira elemento da lista */

**Fim**

R $\leftarrow$ "SE $\{u_i \mid u_i \in E_{relevante}\}$ ENTÃO $u_j$"

Substitua as unidades $u_i$ em R pelos respectivos nomes das variáveis, negando a variável quando a entrada da unidade for –1

/* a condição da regra é um AND entre todos os $U_i$'s */

**Para cada** $u_i \in E_{relevante}$ **faça**

Justifique($u_i$)

**Fim**

**FimSe**

**Fim**

**Função** contrib($u_i$) /* contribuição da entradas ponderadas do nó $u_i$ para $u_j$ */

**Inicio**

/* $V_X$ = valor de ativação do nó $u_x$ */

**Retorne** $V_i * w_{ij} * V_j$

**Fim**

Figura 3.6 Pseudocódigo do algoritmo que justifica as inferências do MACIE

Tabela 3.1 Identificando variáveis contribuintes

| $C^i$ | $w_{i,j}$ | $u_j$ | $w_{i,j} * u_j$ | modifica $C^i$ | $C^i w_{i,j} u_j$ |
|---|---|---|---|---|---|
| +1 | + | – | – | Sim | ≤ 0 |
| +1 | – | + | – | Sim | ≤ 0 |
| **+1** | **–** | **–** | + | **Não** | **≥ 0** |
| **+1** | + | + | + | **Não** | **≥ 0** |
| **–1** | + | **–** | **–** | **Não** | **≥ 0** |
| **–1** | **–** | + | **–** | **Não** | **≥ 0** |
| –1 | – | – | + | Sim | ≤ 0 |
| –1 | + | + | + | Sim | ≤ 0 |

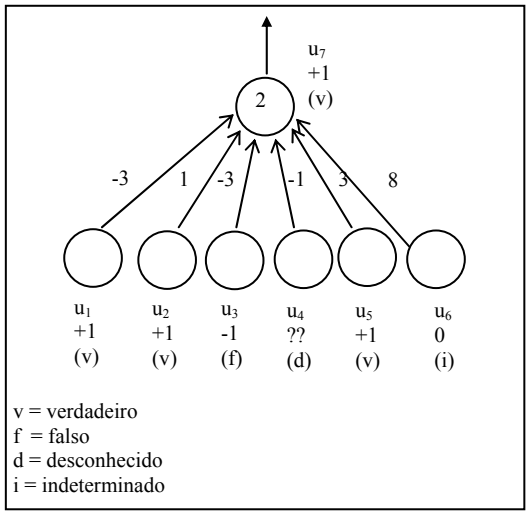



Figura 3.7 Justificando que a inferência de $u_7$ é verdade

Conforme descrito no algoritmo da Figura 3.6, as variáveis do algoritmo MACIE são iniciadas da seguinte forma:

- a variável $S_{conh}$ é iniciada com o *bias* da unidade focalizada na inferência
- a variável ε é iniciada com o conjunto dos índices relativos às conexões que ainda não foram tratadas pelo algoritmo (ou seja, inicialmente, todas as conexões, com exceção daquelas cujo peso é zero e daquela que representa o *bias*).
- a variável $S_{desc}$ é iniciada com a soma dos pesos do conjunto ε. Essa variável contém o valor máximo de variação da variável $S_{conh}$.

Os passos do algoritmo descrevem um procedimento iterativo que basicamente:

- verifica se o procedimento terminou, i.e., verifica se o valor acumulado em $S_{conh}$ é maior do que o valor ainda por examinar. Se isso for verdade, a regra sendo criada é finalizada, caso contrário,
- 'escolhe' um atributo (representado por um índice associado a uma unidade) que irá fazer parte da regra. Essa escolha é baseada em dois critérios:

  - o atributo escolhido não pode alterar a soma ponderada, isto é,

    $C^i w_{ij} u_j \geq 0$

  - o valor $|w_{ij}|$ deve ser o maior possível.

  Esses dois critérios são avaliados no algoritmo da Figura 3.3 através da função contrib, que calcula o produto das ativações de $u_i$ e $u_j$ e do peso $w_{i,j}$ entre as duas unidades.
- acumula na variável $S_{conh}$, o produto do peso da conexão do atributo eleito pelo seu correspondente valor de ativação
- atualiza a variável $S_{desc}$, retirando dela o peso da conexão do atributo eleito
- atualiza o conjunto de índices armazenado em ε, retirando dele o índice correspondente à conexão utilizada nos passos anteriores

Segue um trecho da execução do algoritmo, para a rede mostrada na Figura 3.7

$u_j$ = nó $u_7$ (nó de saída desse exemplo)

$E = \{u_1, u_2, u_3, u_4, u_5, u_6\}$

$\varepsilon = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ (nesse exemplo as entradas do nó coincidem com as entradas da rede)

$\Omega = \{-3, 1, -3, -1, 3, 8\}$

$S_{conh} \leftarrow$ 2 (valor do *bias*)

$$S_{desc} \leftarrow \sum_{U_i \in \varepsilon} \left| W_{ij} \right| = |-3| + |1| + |-3| + |-1| + |3| + |8| = 19$$

$|2| < 19$ ?

$u_m \leftarrow u_r \mid contrib(u_r) > contrib(u_s) \quad \forall \; u_r, u_s \in \varepsilon$

| j | contrib | $\lvert w_{7,j} \rvert$ |
|---|---|---|
| 1 | (+1)(−3)(+1) ≤ 0 | 3 |
| 2 | (+1)(+1)(+1) ≥ 0 | 1 |
| 3 | (+1)(−3)(−1) ≥ 0 | 3 |
| 4 | (+1)(−1)(??) | 1 |
| 5 | (+1)(+3)(+1) ≥ 0 | 3 |
| 6 | (+1)(0)(+8) ≥ 0 | 8 (*) |

$contrib(u_6) = 8$ "nó que mais contribui com a ativação"

$V_{max} \leftarrow 8 * 0 = 0$ "valor da entrada que mais contribui"

$S_{conh} \leftarrow 2 + 0$

$E_{relevante} \leftarrow \{\} \cup \{u_6\}$

$S_{desc} \leftarrow 19 - 8 = 11$

$\varepsilon \leftarrow \{u_1, u_2, u_3, u_4, u_5\}$

$|2| < 11$?

$u_m \leftarrow u_r \mid contrib(u_r) > contrib(u_s) \quad \forall \; u_r, u_s \in \varepsilon$

| j | contrib | $\lvert w_{7,j} \rvert$ |
|---|---|---|
| 1 | (+1)(−3)(+1) ≤ 0 | 3 |
| 2 | (+1)(+1)(+1) ≥ 0 | 1 |
| 3 | (+1)(−3)(−1) ≥ 0 | 3 |
| 4 | (+1)(−1)(??) | 1 |
| 5 | (+1)(+3)(+1) ≥ 0 | 3 |

$contrib(U_3) = 3$

$V_{max} \leftarrow -3 * -1 = 3$

$S_{conh} \leftarrow 2 + 3 = 5$

$E_{relevante} \leftarrow \{u_6\} \cup \{u_3\} = \{u_3, u_6\}$

$S_{desc} \leftarrow 11 - 3 = 8$

$\varepsilon \leftarrow \{u_1, u_2, u_4, u_5\}$

$|5| < 8$ ?

$u_m \leftarrow u_r \mid contrib(u_r) > contrib(u_s) \quad \forall \; u_r, u_s \in \varepsilon$

| j | contrib | $\lvert w_{7,j} \rvert$ |
|---|---|---|
| 1 | (+1)(−3)(+1) ≤ 0 | 3 |
| 2 | (+1)(+1)(+1) ≥ 0 | 1 |

| | | |
|---|---|---|
| 4 | (+1)(−1)(??) | 1 |
| 5 | (+1)(+3)(+1) ≥ 0 | 3 |

contrib($U_5$) = 3

$V_{max} \leftarrow 3 * 1 = 3$

$S_{conh} \leftarrow 5 + 3 = 8$

$E_{relevante} \leftarrow \{u_3,u_6\} \cup \{u_5\} = \{u_3, u_5, u_6\}$

$S_{desc} \leftarrow 8 - 5 = 3$

$\xi \leftarrow \{u_1, u_2, u_4\}$

|8| < 5 ? Não => sai da estrutura de iteração

R ← "SE $\{u_3, u_5, u_6\}$ ENTÃO $u_7$"

Substituindo os nomes das variáveis:

"SE ($u_3$ é falso) E ($u_5$ é verdade) E ($u_6$ é desconhecido) ENTÃO $u_7$ é verdade"

Note que se o *bias* for grande o suficiente é possível ter uma unidade que é sempre verdade (ou falsa), independentemente de suas entradas. Por exemplo, se mudarmos apenas o *bias* associado à unidade $u_7$ de 2 para 100, na rede mostrada na Figura 3.4, aquela unidade terá ativação +1 permanentemente.

Como a variável $S_{conh}$ é iniciada com o valor do *bias* da unidade, ela será, neste caso, iniciada com 100. Como $S_{conh}$ é maior do que $S_{desc}$ o algoritmo passa direto sem efetuar nenhuma iteração, deixando as variáveis com os valores com os quais foram iniciadas. Como o resultado, a lista $E_{relevante,}$ que armazena as variáveis necessárias à ativação do nó, ficará vazia, e a regra gerada será "$u_7$ é sempre verdade".

## 3.4 O CONJUNTO DE REGRAS

Esta seção apresenta o algoritmo que gera um conjunto de regras a partir de uma rede neural, fazendo uso do engenho de inferência do MACIE. A abordagem proposta em [Gallant−1993] limita o número de condições das regras, de maneira a poder limitar o número de possíveis regras geradas (caso contrário, o problema pode ser intratável).

Suponha, por exemplo, que dadas p variáveis booleanas de entrada, se busque por regras que tenham no máximo 3 delas (assumindo que cada variável booleana possa apenas ser verdadeira ou falsa, i.e., o valor desconhecido não é considerado). Podemos então verificar todas as regras com 0, 1, 2 e 3 termos, o que leva ao exame de no máximo $\sum_{t=0}^{3}\binom{p}{t}2^t$ casos.

Quando da verificação se um determinado conjunto de 3 das p variáveis constitui uma regra válida, o algoritmo evita a verificação das $2^{p-3}$ variáveis restantes, objetivando desempenho. A Figura 3.8 apresenta o pseudocódigo do algoritmo, como descrito em [Gallant–1993] e a Figura 3.9 apresenta uma adaptação do algoritmo mais fiel à implementação realizada.

Esse algoritmo apenas evidencia possíveis subconjuntos de variáveis booleanas, constrói uma regra e deixa a cargo do engenho de inferência a verificação da validade ou não da regra construída. A versão adaptada do algoritmo e descrita na Figura 3.9 foi colocada para permitir a visualização das estruturas de repetição aninhadas, que são necessárias para 'varrer' o espaço de variáveis.

```
Let T = maximum number of terms allowed in a rule.
1. RULES_SET = ∅.
2. For number of terms t = 0,1,…, T:
  2a. For each choice of t variables, and for each setting of these variables to
      true (+1) or false (-1):
    2aa. If MACIE's inferencing indicates a valid rule, and if no subset of
         t-1 out of the t variables gives a valid rule, then add this rule to
         RULES_SET.
3. Output RULES_SET.
```

Figura 3.8 Geração de um conjunto de regras com um número limitado de condições, usando enumeração e o sistema de inferência do MACIE – pseudocódigo original proposto em [Gallant-1993]

```
Procedimento: Extração(R)  /* para extração de regras da RN usando o MACIE */
Entradas
   T = número máximo de antecedentes em uma regra
   R = ∅                    /* conjunto de regras     */
Início
   Para t variando de 0 a T faça
      Para cada α ← agrupamento de t variáveis faça
         Para cada combinação de valores verdadeiro (+1) ou falso (-1) nas variáveis de α faça
            Para cada unidade de saída u_s faça
            Início
               r ← Justifique(u_s)
               Se (r é uma regra válida e não existe regra mais geral que r em R) então
                  R ← R ∪ {r}
               FimSe
            Fim
         FimPara
      FimPara
   FimPara
   Retorne R
Fim
```

Figura 3.9 Pseudocódigo do algoritmo adaptado para a geração de um conjunto de regras com um número limitado de condições, usando enumeração e o sistema de inferência do MACIE

## 3.5 EXPERIMENTOS USANDO O MACIE

Essa seção diz respeito ao uso do MACIE em um problema comum, para fornecer subsídios para uma comparação entre os quatro métodos pesquisados (Seção 3.5.1), bem como do seu uso no problema específico, para explorar suas funcionalidades (Seção 3.5.2).

### 3.5.1 EXEMPLO COMUM

Nessa seção é apresentado o exemplo comum que será utilizado para discutir detalhes do projeto e da implementação dos quatro métodos pesquisados.

Para que o exemplo seja mais facilmente adaptado para cada um dos métodos, é preferível que o exemplo seja simples e contenha poucas restrições. Um exemplo clássico que atende a esses requisitos é o OU-Exclusivo ou XOR.

O problema OU-Exclusivo mais simples, com duas entradas e uma saída, é uma função binária, conforme apresentado na Tabela 3.4. Esse problema apresenta as seguintes características interessantes para este trabalho:

- pode ser implementado para qualquer dos métodos pesquisados;
- tem entradas binárias, podendo ser implementado por uma rede simples como a utilizada pelo MACIE;
- necessita de apenas duas entradas, podendo ser visualizado em um gráfico, no caso do RULEX;

- é um problema linearmente não separável, permitindo testar o poder de representação da RN utilizada em cada método;
- é um problema simples e didático, facilitando o entendimento, bem como a comparação entre os vários métodos.

Tabela 3.4 Função OU-Exclusivo

| Entrada a | Entrada b | Saída c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

A arquitetura da RN a ser utilizada para o problema OU-Exclusivo é apresentada na Figura 3.11.

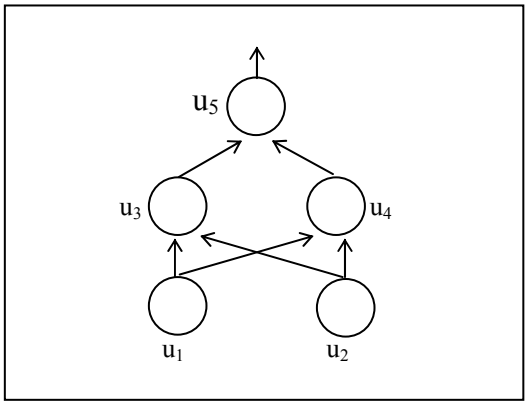



Figura 3.11 RN do OU-Exclusivo

### 3.5.2 EXEMPLO ESPECÍFICO

Para o MACIE, o problema abordado no exemplo específico é um sistema especialista. Mais especificamente, um sistema especialista baseado em dados reais[5] que recomenda a aceitação ou recusa de uma Ordem de Serviço em uma fábrica de software.

Algumas fábricas de software realizam o trabalho de desenvolvimento de software de modo geograficamente distribuído, com programadores em diferentes escritórios trabalhando no mesmo projeto. Devido a essa distribuição, existe a necessidade de um grupo central que avalie os trechos de código desenvolvidos em cada escritório para, finalmente, agregá-los em um único sistema. Os trechos de código devem assim ser

[5] Nomes foram alterados para evitar divulgação de informação confidencial.

enviados ao grupo central por meio de um registro denominado Ordem de Serviço, que contém diversos dados sobre o trecho de código escrito.

Existe uma série de verificações que o grupo central deve fazer antes de aceitar uma Ordem de Serviço, caso contrário, corre-se o risco de integrar um código problemático ao sistema final. A experiência particular de cada membro do grupo central permite que as Ordens de Serviço problemáticas sejam identificadas sem maiores dificuldades, por meio da análise de campos específicos de cada Ordem de Serviço. Porém, o modo exato como essa decisão é tomada não é formalmente definido.

Nesse contexto, o sistema para avaliação de Ordens de Serviço pode ser empregado para facilitar a tomada de decisão do grupo central, bem como tornar mais viável a possibilidade de outros indivíduos menos experientes realizarem essa tarefa.

Os dados utilizados para geração da RN são de duas origens: nomes e dependências de variáveis de entrada, fornecidas pelo grupo central, e registro histórico de Ordens de Serviço problemáticas, armazenado ao longo de vários meses de trabalho da fábrica de software. As dependências são usadas na definição da arquitetura da RN (limitando as conexões entre os nós que representam as variáveis do problema), enquanto que o registro histórico é utilizado como conjunto de padrões de treinamento da RN.

Os dados presentes em uma Ordem de Serviço são apresentados na Tabela 3.2.

As dependências existentes, determinadas pelos membros do grupo central, são apresentadas na Tabela 3.3. As variáveis na coluna da esquerda (*Variável dependência*) são dependências das variáveis na coluna da direita (*Variável dependente*). Na arquitetura de RN, isso significa que a ativação do neurônio equivalente à variável dependente só ocorre mediante a ativação do neurônio equivalente à variável dependência.

Tabela 3.2 Variáveis de entrada do sistema de ordens de serviço

| Nome da Variável | Descrição da variável | Formato de Entrada |
|---|---|---|
| IR2 | Taxa de inspeção maior que 300 | sim/não |
| MFD< | Número de falhas operacionais menor que 5 | sim/não |
| MFD> | Densidade de falhas operacionais maior que 20 | sim/não |
| PR2 | Taxa de preparação maior que 300 | sim/não |
| PT0 | Tempo de preparação igual a 0 | sim/não |
| SZ | Tamanho da inspeção maior que 300 | sim/não |
| SZ0 | Tamanho da inspeção igual a 0 | sim/não |
| TF0 | Número total de falhas declaradas igual a 0 | sim/não |

| | | |
|---|---|---|
| TFD< | Densidade total de falhas menor que 30 | sim/não |
| TFD> | Densidade total de falhas maior que 80 | sim/não |
| IP | Interessa a preparação | sim/não |
| INE | Interessa o número de erros | sim/não |
| ITT | Interessa o tamanho do trabalho | sim/não |
| ITE | Interessa o tipo de erro | sim/não |

Tabela 3.3 Lista de dependências do sistema de ordens de serviço

| Variável dependência | Variável dependente |
|---|---|
| IR2 | ITT |
| MFD< | ITE |
| MFD> | ITE |
| PR2 | IP |
| PT0 | IP |
| SZ | ITT |
| SZ0 | ITT |
| TF0 | INE |
| TFD< | INE |
| TFD> | INE |

A partir da lista de nomes de variáveis e dependências, é obtida uma arquitetura de RN apresentada na Figura 3.10.

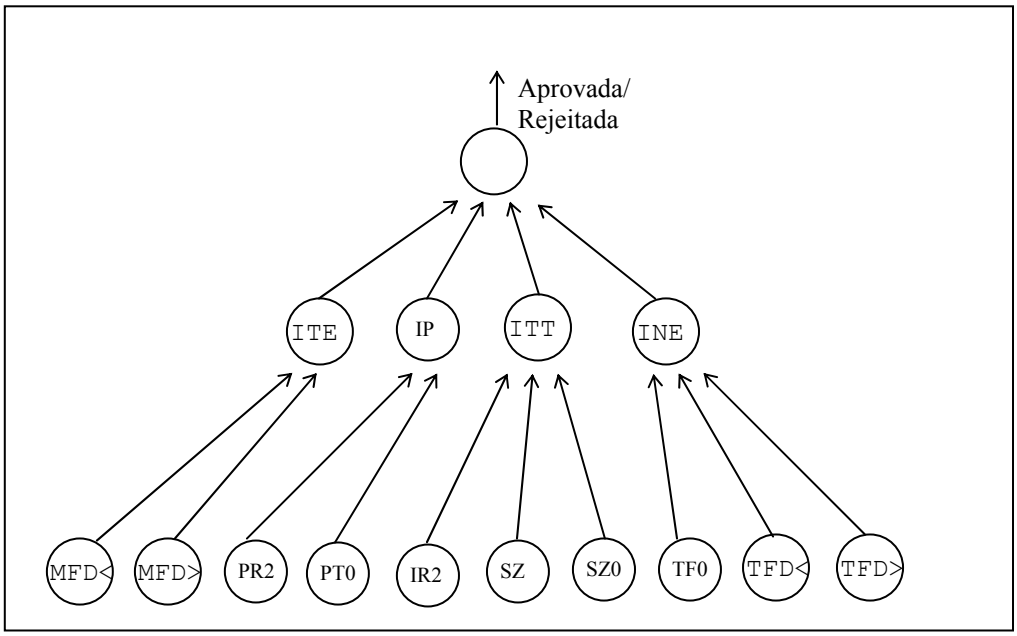



Figura 3.10 Arquitetura da RN do Problema das Ordens de Serviço utilizando lista de dependências

Os exemplos sugeridos para o MACIE não serão tratados como problemas de Sistema Especialista. O intuito é apresentar o algoritmo de extração de regras do MACIE. Para tanto, será utilizada como entrada uma RN treinada com algoritmo *Backpropagation* padrão (sem termo-momento), pois a rede discriminante linear não é capaz de aprender problemas não-linearmente-separáveis. A RN, porém, não será padrão, sendo modificada de modo que as funções de ativação (*thresholds*) tenham *ativação conservadora*, ou seja, o neurônio só propaga sinal quando a soma dos valores conhecidos supera a soma máxima dos valores desconhecidos. Os experimentos realizados mostram que a ativação conservadora é um pré-requisito essencial para o correto funcionamento do MACIE (Ver Seção 3.6.2.1).

## 3.6 RESULTADOS EXPERIMENTAIS USANDO O MACIE

Foram executados 13 conjuntos de testes usando o sistema MACIE. Seis testes foram realizados utilizando o conjunto de treinamento para o exemplo comum ("OU-Exclusivo") e sete testes foram realizados utilizando o conjunto de treinamento para o exemplo específico para o MACIE ("Ordens de serviço"). A seguir são apresentados a configuração e os resultados dos testes.

### 3.6.1 CONFIGURAÇÃO DOS TESTES

Todos os testes foram realizados com os conjuntos de treinamento descritos na Seção 3.5 deste trabalho. Todos os testes foram realizados com os mesmos valores de configuração do algoritmo de aprendizado, definidos na Tabela 3.5. Todos os testes utilizaram RN com pesos iniciais definidos com valores aleatórios variando entre – 0.002 e 0.002[6].

Nos testes realizados para o exemplo específico, a verificação do aprendizado neural é realizada usando *5*-validação cruzada. Nos testes relativos ao exemplo comum (OU-Exclusivo), a verificação é realizada utilizando o próprio conjunto de treinamento como conjunto de teste, pois não é possível gerar mais de um conjunto de padrões para o problema "OU-Exclusivo" com duas entradas binárias.

As RNs utilizadas apresentam imposição na função de ativação necessária ao funcionamento do MACIE conforme apresentado na Seção 3.2.1.

[6] Neste trabalho foi adotado "." como separador de casa decimal.

Tabela 3.5 Valores para o algoritmo de aprendizado

| Limite de épocas | Erro mínimo | Taxa de Aprendizado |
|---|---|---|
| 50 000 | 0.1 | 0.1 |

### 3.6.2 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO COMUM

Para o exemplo comum, descrito na Seção 3.5, foram realizados seis conjuntos de testes agrupados da seguinte forma: no primeiro grupo, identificado como *Capacidade*, estão os testes que avaliam a eficácia da rede usada pelo método; no segundo grupo, identificado como *Qualidade*, estão os testes que avaliam a qualidade das regras via critérios da ADT.

Capacidade:

1) Aprendizado com restrição da função de ativação da RN;
2) Aprendizado sem restrição da função de ativação da RN;

Qualidade:

3) Consistência das regras extraídas para diferentes RNs;
4) Consistência das regras extraídas para a mesma RN;
5) Acurácia e Fidelidade das regras extraídas;
6) Compreensibilidade das regras extraídas.

Os dois primeiros conjuntos de testes têm por objetivo verificar a capacidade de aprendizado da RN, sobretudo com intuito de verificar se o aprendizado não é prejudicado pelas alterações necessárias para a extração de regras.

Para esse exemplo, bem como para o exemplo específico, não foram realizadas medições de tempo de execução. Para os testes de aprendizado, assume-se que quanto maior o número de conexões entre nós e quanto maior o número de épocas necessárias para convergência dos pesos das conexões, maior será o tempo do aprendizado. Para os testes de extração de regras, a escala do tempo de execução (menos de um segundo para qualquer caso) tornou qualquer comparação de tempo irrelevante. Essa mesma observação é válida para todos os demais métodos implementados neste trabalho.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 3.6.2.1 e Qualidade – Subseção 3.6.2.2).

#### 3.6.2.1 TESTES PARA VERIFICAÇÃO DA CAPACIDADE DE APRENDIZADO

Para que o sistema MACIE funcione, é necessária a imposição de restrições à função de ativação da RN (Seção 3.2.1). Os resultados experimentais apresentados na Tabela 3.6 mostram que tais restrições comprometem o aprendizado neural. A RN cuja arquitetura foi restrita com funções de ativação conservadora e com saída discreta não apresentou convergência dos pesos, ou seja, foi incapaz de aprender o conceito embutido no conjunto de treinamento. Testes parciais mostram que a RN não converge mesmo com alterações na taxa de aprendizado.

A Tabela 3.7 apresenta os resultados experimentais para a RN cuja arquitetura é a "Perceptron Multicamadas" sem qualquer alteração para comportar o MACIE. Houve convergência dos pesos em 85% dos casos. Em testes realizados com o aumento da taxa de aprendizado houve convergência dos pesos em 100% dos casos.

Tabela 3.6 Resultados experimentais para "OU-Exclusivo" e RN com função de ativação restrita

| Número médio de épocas até a convergência[7] | Erro médio | Desvio Padrão do Erro |
|---|---|---|
| 50000 | 53% | 0.14 |

Tabela 3.7 Resultados de teste para "OU-Exclusivo" e RN sem ativação restrita

| Taxa de Aprendizado | Número médio de épocas até a convergência | Erro médio | Desvio Padrão do Erro |
|---|---|---|---|
| 0.1 | 10748 | 4% | 0.09 |
| 0.5 | 356 | 0 | 0 |

Desse modo, os resultados apresentados mostram que a restrição na função de ativação compromete o aprendizado neural. Os testes seguintes avaliam o MACIE quanto à qualidade das regras extraídas.

[7] 50 000 foi o limite de épocas do treinamento, qualquer RN cuja sessão de treinamento chegou a esse número de épocas provavelmente não conseguiu aprender satisfatoriamente o conceito representado pelo conjunto de treinamento.

#### 3.6.2.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

Uma das dimensões propostas pela taxonomia ADT é a "Qualidade das Regras Extraídas", que por sua vez possui quatro subgrupos (Capítulo 2). A seguir são apresentados os resultados dos testes de Consistência, Acurácia, Fidelidade e Compreensibilidade das regras. São sugeridas medidas para avaliação dos critérios, as quais são discutidas com mais detalhes no Capítulo 7.

- CONSISTÊNCIA DAS REGRAS EXTRAÍDAS: característica de "Qualidade das Regras Extraídas" que, apesar de não possuir uma medida definida (Capítulo 7), pode ser tomada como o percentual de conjuntos de regras comuns obtido em várias execuções do processo de extração de regras, sobre a mesma RN ou sobre RNs originadas de diferentes sessões de treinamento[8].

  Os resultados mostram que as regras geradas variam de acordo com a configuração dos pesos e, por isso, diferentes sessões de treinamento geram regras diferentes. Ainda assim, houve coincidência de 40% dos conjuntos de regras extraídos.

  A mesma seqüência de testes foi realizada sem alteração na RN, ou seja, foi executado o algoritmo de extração de regras diversas vezes sobre um mesmo conjunto de pesos. Nesse caso, houve coincidência de 100% dos conjuntos de regras extraídos, ou seja, os conjuntos são idênticos para todas as execuções realizadas. A Figura 3.12 apresenta o conjunto de regras extraído a partir do teste de Consistência sobre a mesma RN. Uma regra "SE(BIAS_0 Não_Entrada1 Entrada2) ENTÃO Não_Saida" é lida como "Se Entrada1 é negativa e Entrada2 é positiva, então a saída é negativa".

[8] RNs originadas de diferentes seções de treinamento fornecem classificação semelhante, mas possuem os pesos das ligações entre os nós potencialmente diferentes.

| A) | B) |
|---|---|
| 0<br>SE ( BIAS_1 Não_Interno1) ENTÃO Não_Saida<br>SE ( BIAS_0 Não_Entrada2 Não_Entrada1) ENTÃO Não_Interno1<br><br>1<br>SE ( BIAS_1 Interno2 Interno1) ENTÃO Saida<br>SE ( BIAS_0 Não_Entrada1) ENTÃO Interno2<br>SE ( BIAS_0 Entrada2) ENTÃO Interno1<br><br>2<br>SE ( BIAS_1 Interno2 Interno1) ENTÃO Saida<br>SE ( BIAS_0 Não_Entrada2) ENTÃO Interno2<br>SE ( BIAS_0 Entrada1) ENTÃO Interno1<br><br>3<br>SE ( BIAS_1 Não_Interno2) ENTÃO Não_Saida<br>SE ( BIAS_0 Entrada1 Entrada2) ENTÃO Não_Interno2 | 0<br>SE (BIAS_0 Não_Entrada2 Não_Entrada1)<br>ENTÃO Não_Saida<br><br>1<br>SE (BIAS_0 Não_Entrada1 Entrada2)<br>ENTÃO Saida<br><br>2<br>SE (BIAS_0 Não_Entrada2 Entrada1)<br>ENTÃO Saida<br><br>3<br>SE (BIAS_0 Entrada1 Entrada2)<br>ENTÃO Não_Saida |

Figura 3.12 A) Regras do exemplo "OU-Exclusivo" extraídas para o teste de Consistência do MACIE. B) Versão simplificada das regras.

- FIDELIDADE DAS REGRAS EXTRAÍDAS: Para o critério "Fidelidade das Regras Extraídas" a medida adotada foi a porcentagem de padrões classificados da mesma maneira que a RN no conjunto de treinamento. Nesse experimento, 100% dos padrões utilizados foram classificados da mesma maneira tanto pela regra quanto pela RN.

- ACURÁCIA DAS REGRAS EXTRAÍDAS: A medida adotada para Acurácia é a porcentagem de padrões corretamente classificados em um conjunto de padrões diferente daquele usado no treino da RN. Não há como obter mais de um conjunto "OU-Exclusivo", por isso a Acurácia foi calculada utilizando-se os mesmos conjuntos para treinamento e teste da RN. O resultado do teste de Acurácia foi 100% dos padrões corretamente classificados.

- COMPREENSIBILIDADE DAS REGRAS EXTRAÍDAS: a medida utilizada foi o produto do número médio de regras pelo número médio de antecedentes por regra. Essa medida foi criada neste trabalho a partir do comentário sobre Compreensibilidade apresentado em [Towell & Shavlik–1993] reproduzido na Seção 2.1.2 deste trabalho. Para o exemplo "OU-Exclusivo", todos os conjuntos

de regras gerados apresentaram duas regras com um antecedente cada, resultando em uma medida de Compreensibilidade igual a 2.

### 3.6.3 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO ESPECÍFICO

Para o exemplo específico, descrito na Seção 3.5.2, foram realizados sete conjuntos de testes, organizados da seguinte maneira:

Capacidade:

1) Aprendizado com ativação conservadora e arquitetura de RN definida por matriz de dependências;
2) Aprendizado sem ativação conservadora e arquitetura de RN definida por matriz de dependências;
3) Aprendizado sem ativação conservadora e arquitetura de RN totalmente conectada;
4) Aprendizado de RN com nós adicionais sem ativação conservadora e arquitetura de RN definida por matriz de dependências;

Qualidade:

5) Qualidade das regras extraídas (Compreensibilidade) de RN com arquitetura definida por matriz de dependências;
6) Qualidade das regras extraídas (Acurácia e Fidelidade) de RN com restrições de arquitetura e função de ativação;
7) Qualidade das regras extraídas (Acurácia e Fidelidade) de RN sem restrições;

É importante notar que, diferente do que foi adotado para o exemplo comum, os testes realizados para o exemplo específico variam dependendo do método. Como comentado anteriormente, o principal objetivo do exemplo específico é avaliar as características particulares de cada método.

Os quatro primeiros conjuntos de testes têm por objetivo verificar o impacto das restrições impostas pelo MACIE na função de ativação e arquitetura de RN sobre a capacidade de aprendizado do sistema. Os demais testes têm por objetivo avaliar a variação da qualidade das regras, com relação aos subgrupos "Compreensibilidade", "Acurácia" e "Fidelidade" da taxonomia ADT, frente às restrições impostas para uso do

MACIE. O principal intuito dessas avaliações é verificar se há ganho real na definição de arquitetura de RN por matriz de dependências (Seção 3.2 deste trabalho) e qual o custo da restrição à função de ativação na qualidade das regras geradas pelo sistema.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 3.6.3.1 e Qualidade – Subseção 3.6.3.2).

#### 3.6.3.1 TESTES PARA AVALIAÇÃO DA CAPACIDADE DE APRENDIZADO

O conjunto de treinamento do "OU-Exclusivo" é um conjunto não linearmente separável, porém autoconsistente, o que permite ao algoritmo de aprendizado realizar a convergência dos pesos da RN. O conjunto de treinamento "Ordens de Serviço", porém, apresenta dados contraditórios, ou seja, instâncias com mesmas descrições pertencentes a classes diferentes.

Apesar da inconsistência dificultar o aprendizado neural, os dados foram mantidos, pois em aplicações reais a identificação de tais inconsistências pode ser muito custosa, ou mesmo impossível. Os resultados dos testes para avaliação da capacidade de aprendizado são apresentados na Tabela 3.8.

Com relação à restrição da função de ativação, o resultado é semelhante ao apresentado para o exemplo "OU-Exclusivo", ou seja, a RN é incapaz de aprender o conceito embutido no conjunto de treinamento, apresentando uma média de acertos próxima de 50%. Sem a restrição da função de ativação, a RN apresenta média bem maior de padrões corretamente classificados (83%).

A restrição imposta à arquitetura, por meio de uma matriz de dependências (Seção 3.1), também afeta negativamente a capacidade de aprendizado da RN. Os resultados do teste em RN com as camadas adjacentes totalmente conectadas, ou seja, sem restrição de arquitetura, apresentam em média 88% dos padrões corretamente classificados.

O último teste realizado para avaliação da capacidade de aprendizado (teste 4) tem por objetivo verificar se a adição de nós entre a camada de entrada e intermediária melhora a classificação da RN, no caso de arquitetura definida por matriz de ativação. O resultado, com média de 77% dos padrões corretamente classificados, mostra que a simples adição de nós não melhora o aprendizado da RN. Para que houvesse aumento da capacidade de aprendizado a partir da adição de nós à RN, seria necessário prover os valores de ativação esperados para cada nó intermediário da RN como parte do conjunto

de treinamento. Porém, tais valores não estão disponíveis no exemplo de treinamento adotado, e em nenhum dos exemplos estudados para este trabalho.

Tabela 3.8 Avaliação da capacidade de aprendizado no exemplo “Ordens de serviço”

| Restrição na função de ativação? | Arquitetura por Matriz de Dependência? | Nós adicionais? | Épocas para convergência | Erro médio | Desvio Padrão do Erro |
|---|---|---|---|---|---|
| Sim | Sim | Não | 50000 | 48.61% | 13.17 |
| Não | Sim | Não | 50000 | 17.28% | 1.39 |
| Não | Não | Não | 50000 | 11.88% | 1.39 |
| Não | Sim | Sim | 50000 | 23.24% | 7.11 |

#### 3.6.3.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

Três dentre as quatro características do critério "Qualidade das Regras Extraídas" da ADT são verificadas nos cinco testes realizados e cujos resultados são apresentados na Tabela 3.9 e Tabela 3.10. A característica “Consistência”, avaliada para o exemplo comum não é reavaliada para o exemplo específico, por ser independente do exemplo utilizado.

A primeira observação a ser feita sobre os resultados é com relação ao número de regras necessárias para classificar um padrão e o número de antecedentes por regra, que fornecem uma medida para o critério "Compreensibilidade das Regras Extraídas". Por essa medida, o conjunto de regras gerado a partir de uma RN cuja arquitetura é definida por uma matriz de dependências é mais passível de ser compreendido que um conjunto gerado a partir de RN com arquitetura sem restrição. Em média, o conjunto de regras gerado a partir de RN sem restrição utiliza 4.4 regras com três antecedentes em cada regra para classificar um padrão, enquanto o conjunto gerado a partir de RN com restrição apresenta uma média de três regras com apenas um antecedente por regra. Adotando o produto dos dois valores como uma medida única de compreensibilidade, temos que o primeiro conjunto tem uma "medida de Compreensibilidade" de 13.2 (4.4 regras × 3 antecedentes por regra), enquanto que a "medida de Compreensibilidade" do segundo conjunto é de apenas 3 (3 regras × 1 antecedente por regra). Nesse caso, quanto menor o valor, mais compreensível é o conjunto de regras.

Tabela 3.9 Compreensibilidade para o exemplo "Ordens de serviço"

| RN definida por matriz de dependências? | Média de regras para classificar um padrão | Média de antecedentes por regra | Compreensibilidade |
|---|---|---|---|
| Sim | 3 | 1 | 3 |
| Não | 4.4 | 3 | 13.2 |

Além da Compreensibilidade, foram verificadas a Acurácia e a Fidelidade das regras extraídas.

O critério "Acurácia das Regras Extraídas" é adotado como sendo o percentual de padrões de teste classificados corretamente.

Para o conjunto de regras gerado a partir de RN cuja arquitetura foi definida por matriz de dependências, a Acurácia média é de 78.98%, ou seja, 78.98% dos padrões são corretamente classificados pelo conjunto de regras extraído da RN. O conjunto de regras extraído a partir de uma RN sem restrições de arquitetura apresenta uma média de Acurácia ligeiramente superior, com 81.89% dos padrões corretamente classificados.

A "Fidelidade das Regras Extraídas" é medida como o percentual de padrões classificados pelas regras da mesma maneira que pela RN, ou seja, o quanto o conjunto de regras gerado é similar à RN que o originou. Para todos os testes sem valores indeterminados, a Fidelidade foi de 100%. Os testes com valores indeterminados foram feitos alterando-se para zero o valor de um atributo escolhido aleatoriamente em cada instância de treinamento. Esse teste visa avaliar a capacidade de generalização atribuída ao MACIE.

Tabela 3.10 Acurácia e Fidelidade para o exemplo "Ordens de serviço"

| RN definida por matriz de dependências? | Valores Indeterminados no teste de Acurácia? | Ativação Conservadora | Média / DesvioPadrão da Acurácia | Média / DP da Acurácia da RN[9] | Média / DesvioPadrão da Fidelidade |
|---|---|---|---|---|---|
| Não | Não | Sim | 81.89 / 2.56 | 81.89 / 2.56 | 100 / 0 |
| Sim | Não | Sim | 78.98 / 3.47 | 78.98 / 3.47 | 100 / 0 |
| Não | Não | Não | 81.38 / 2.48 | 81.38 / 2.48 | 100 / 0 |
| Não | Sim | Não | 72.16 / 15.13 | 88.65 / 1.13 | 80.17 / 19.15 |

[9] Resultados da RN para o conjunto de teste. Como a RN pode apresentar saída diferente, a classificação da RN também é colocada para permitir comparação.

Nos testes realizados sem a função de ativação conservadora e com entradas incompletas (cada padrão apresentava uma entrada indeterminada), as regras extraídas apresentaram valores médios de Acurácia e Fidelidade cerca de 10% e 20%, respectivamente, abaixo do teste com entradas completas. Nesse mesmo teste, a RN sem ativação conservadora apresentou resultado superior ao das regras extraídas, com uma diferença de apenas 0.27% nas médias dos erros de classificação entre os conjuntos completo e incompleto.

Os resultados mostram que a definição da arquitetura por meio de uma matriz de dependências melhora a Compreensibilidade, porém piora a Acurácia e a Fidelidade das regras extraídas. Também pelos resultados dos testes, a restrição da função de ativação conservadora mostrou-se fundamental para garantir que a Fidelidade das regras extraídas seja sempre 100% no caso de valores de entrada indeterminados.

## 3.7 CONCLUSÕES SOBRE O MACIE

Conforme pode ser observado na apresentação feita nas seções anteriores, o MACIE apresenta uma série de restrições, atribuídas tanto ao modelo de RN que utiliza, quanto ao processo de extração de regras. Tais restrições são apresentadas em [Gallant–1988] e [Gallant–1993], porém sem muito detalhes e/ou discussão sobre seu impacto no sistema.

As conclusões apresentadas nesta seção têm por objetivo explicitar algumas restrições do MACIE, bem como levantar a questão do significado dessas restrições ao uso do sistema.

O uso de informação de dependência reduz a dimensionalidade do problema de treinamento da RN. Em contrapartida, uma RN não totalmente interconectada tem menos chances de aprender novos conceitos possivelmente ocultos no conjunto de treinamento. Uma questão relevante é que uma das vantagens do sistema especialista conexionista é justamente não depender do conhecimento de um especialista humano. A vantagem obtida com o uso de informação de independência é claramente demonstrada em [Gallant–1988], porém deve-se destacar que, apesar de desejável, nem sempre será possível obter esse recurso em problemas do mundo real. Nenhum dos exemplos utilizados neste trabalho apresenta todas as informações necessárias para aplicação efetiva da matriz de dependências.

É possível utilizar outros modelos além da RN discriminante linear. Porém, qualquer modelo utilizado apresenta uma severa restrição na função de ativação, a qual necessariamente será de *ativação conservadora*. Isso significa que os neurônios da RN disparam uma saída somente quando suas entradas conhecidas superam o valor, em módulo, dos pesos relacionados às entradas desconhecidas. Assim, é possível utilizar modelos de RN que tenham pesos com valores reais com função sigmoidal com saída entre –1 e 1; a função de ativação, entretanto, precisará de um teste extra, para resultar em 0 (zero) caso as entradas conhecidas não superem o valor máximo possível das entradas desconhecidas.

Caso essa restrição não seja satisfeita, a fidelidade das regras extraídas é comprometida devido ao modo como o algoritmo trabalha. Para que o algoritmo de extração de regras funcione, valores (1,-1 ou 0) precisam ser atribuídos às entradas e propagados através da RN para que cada nó intermediário e de saída tenha suas *variáveis contribuintes* determinadas. Porém, pode acontecer de nem todos os valores de entrada sejam conhecidos (ou seja, seja aplicado valor 0).

Suponha que um nó da RN tenha apenas um valor de entrada conhecido e seu valor ponderado somado ao *bias* seja maior que o limiar de ativação do nó. Se esse nó não tiver ativação conservadora, ele irá disparar uma saída positiva, independente dos pesos de suas demais conexões. O MACIE, porém, não irá gerar uma regra que reflita essa ativação caso a soma em módulo dos pesos das entradas desconhecidas supere o valor ponderado da entrada conhecida. Nesse caso, a saída gerada por esse nó não seria corretamente representada pela regra gerada e a regra gerada não estaria de acordo com o resultado obtido da RN. Como o algoritmo de extração de regras do MACIE depende dessa condição, a solução para evitar inconsistências entre a RN e as regras geradas acaba sendo restringir a rede. Essa restrição reduz bastante o escopo de aplicação do MACIE. Caso a função de ativação não tivesse tal restrição, MLPs já treinadas e em uso poderiam ser usadas como base de conhecimento de sistemas especialistas conexionistas, sem que fosse necessário o trabalho extra de definição e treinamento de outra RN.

O algoritmo de extração de regras do MACIE foi proposto para fornecer explicação simbólica para justificar as inferências realizadas, por meio de um conjunto de regras específico para um determinado conjunto de valores de entrada. O uso desse algoritmo visando extrair todo o conhecimento embutido na RN implica uma abordagem exaustiva indiscriminada, possivelmente buscando regras em regiões do espaço de pesos da RN

que não foram influenciadas pelo treinamento e gerando um conjunto de regras muito grande. A restrição do número de antecedentes, proposta por Gallant, reduz o espaço de busca, mas pode prejudicar a fidelidade do conjunto de regras, pois as regras geradas poderão não representar todo o conhecimento embutido na rede.

A alternativa utilizada neste trabalho para reduzir a dimensão do problema foi extrair as regras apenas a partir dos valores do conjunto de treinamento. Com essa abordagem, além de se evitar a busca exaustiva, gerou-se um conjunto de regras que reflete mais precisamente o comportamento esperado da RN. O problema decorrente é que, com essa abordagem, perde-se duas vantagens normalmente associadas à pesquisa em extração de regras de RN: 1)Ao extrair regras somente a partir do conjunto de treinamento, reduz-se a possibilidade de encontrar regiões do espaço de pesos da RN em que o treinamento tenha sido menos eficiente. 2) As regras geradas não irão refletir todo o conhecimento existente na RN, de modo que não será possível prever a resposta dada a valores de entrada que não estejam presentes no conjunto de treinamento. Mas a despeito dos problemas decorrentes dessa solução alternativa, outras abordagens apresentadas para extração de regras de RN (como o FERNN [Setiono & Leow–2000], apresentado no Capítulo 4 deste trabalho) utilizam solução semelhante.

# CAPÍTULO 4. O FERNN

O método FERNN (Fast Extraction of Rules from Neural Networks) [Setiono & Leow–2000] se propõe à rápida extração de regras a partir de RNs MLP com uma camada intermediária, o que, segundo os autores, é possível por meio de uma técnica que dispensa o uso de algoritmos de poda tradicionais (com retreinamento da RN).

A aplicação de algoritmos de poda a uma rede, após o seu treinamento, visa a obtenção de uma RN mais enxuta, sem nós redundantes ou irrelevantes. Para a extração de regras simbólicas, a remoção de nós desnecessários significa a possibilidade de obter um conjunto de regras menor e mais conciso e, portanto, passível de ser melhor entendido por seres humanos. Porém, diversos algoritmos de poda ([Hagiwara–1994], [Setiono–1997], [Castellano et al.–1997], entre outros) realizam um processo iterativo que exige que a RN seja retreinada diversas vezes até a retirada de todos os nós supérfluos, o que se torna inconveniente uma vez que o retreinamento após a poda pode consumir mais tempo que o treinamento inicial da rede.

A escolha do FERNN como objeto de investigação deste trabalho foi motivada pelo fato de este ser, dentre os métodos analisados, o único cuja proposta mostra preocupação com a performance (nesse caso, tempo de execução) do processo de extração de regras.

## 4.1 INTRODUÇÃO

O funcionamento do FERNN é altamente dependente do regime de treinamento da RN da qual se pretende extrair as regras. A função de erro utilizada deve ser a função erro de entropia-cruzada aumentada[10], e o algoritmo de treinamento deve ser o BFGS (Broyden-Fletcher-Goldfarb-Shanno) [Battiti–1992], desenvolvidos para minimizar os pesos irrelevantes de modo que possam ser facilmente identificados na RN treinada.

Por meio da análise de ativação dos nós da RN treinada, o FERNN constrói uma árvore de decisão, a partir da qual são extraídas as regras simbólicas. O processo de construção da árvore de decisão garante que qualquer nó que não faça parte da árvore é

[10] A função de erro de entropia-cruzada é estendida com a adição de uma função de penalidade de maneira que conexões relevantes e irrelevantes da rede possam ser diferenciadas por meio de seus pesos, quando o treinamento termina.

um nó irrelevante e pode ser removido da rede, sem qualquer necessidade de re-treinamento.

O método FERNN apresenta uma proposta interessante pela importância dada à performance e pelo rigor matemático na prova da viabilidade da remoção das conexões de nós de entrada irrelevantes e extração das regras. É necessário um estudo mais detalhado de seu funcionamento, entretanto, pois existe uma questão importante com relação ao escopo do método, conforme apontado nas conclusões sobre o FERNN (Seção 4.5).

## 4.2 O FUNCIONAMENTO DO FERNN

O método FERNN consiste de quatro passos:

1. treinamento da RN de maneira a minimizar a função de erro de entropia-cruzada aumentada;
2. uso da medida de *ganho de informação* [Setiono & Leow–2000] para identificação de unidades relevantes e construção de uma árvore de decisão que classifica os padrões de treinamento em termos de ativação de nós da camada intermediária;
3. para cada nó intermediário cujos valores de ativação são usados para a divisão de um nó na árvore de decisão, remover as conexões de entrada irrelevantes a esse nó intermediário. Uma conexão é irrelevante se sua remoção não afeta a performance de classificação da árvore;
4. para conjuntos de dados com atributos discretos, substituição de cada nó de decisão da árvore de decisão por um conjunto de regras simbólicas equivalente.

Os dois componentes principais do FERNN são o algoritmo de treinamento e o algoritmo de geração da árvore de decisão.

O algoritmo de treinamento minimiza a função de erro de entropia-cruzada aumentada usando uma função de penalidade. A minimização da função de erro garante que os pesos das conexões aos nós da camada de entrada que correspondem a entradas irrelevantes tenham valores próximos a zero na RN treinada. Tais conexões podem ser removidas sem que a precisão da classificação da rede seja afetada.

O algoritmo de geração da árvore de decisão constrói uma árvore de decisão usando os valores de ativação das unidades intermediárias da rede.

Após a árvore de decisão ter sido criada, as entradas relevantes e irrelevantes da rede são diferenciadas. Setiono e Leow criaram um critério para remoção das conexões irrelevantes entre as camadas de entrada e intermediária, que não afeta a acurácia da rede. Um grupo de tais conexões pode ser removido de uma vez se o critério for satisfeito.

A extração de regras é feita reescrevendo as condições dos nós da árvore de decisão em termos das entradas da RN que não foram removidas. Uma condição da árvore de decisão pode ser reescrita como regras na Forma Normal Disjuntiva (FND) ou como regras M-de-N (apresentadas e discutidas na Seção 4.2.4). As seções a seguir detalham, de acordo com [Setiono & Leow–2000], os quatro passos que constituem o FERNN.

### 4.2.1 O TREINAMENTO DA REDE NEURAL

Considere P instâncias de treinamento e seja p uma delas. O valor $S_{ip}$ da unidade de saída $S_i$ da RN e o valor de ativação $H_{jp}$ da unidade intermediária $H_j$ são calculados da seguinte forma:

$$S_{ip} = \sigma\left(\sum_{j=1}^{J} v_{ij} H_{jp}\right) \tag{4.1}$$

$$H_{jp} = \sigma\left(w_j x_p\right) = \sigma\left(\sum_{k=1}^{K} w_{jk} x_{kp}\right) \tag{4.2}$$

onde:

$x_{kp} \in [0,1]$ : valor da unidade de entrada k para o padrão de treinamento p,

$w_{jk}$ : peso da conexão entre a unidade de entrada k e a unidade intermediária j,

$v_{ij}$ : peso da conexão da unidade intermediária j e a unidade de saída i,

$\sigma(\xi)$ : função sigmóide $1/\left(1+e^{-\xi}\right)$,

J : número de unidades intermediárias,

K : número de unidades de entrada.

Cada instância $\mathbf{x_p}$ pertence a uma de C possíveis classes $C_1$, $C_2$, …,$C_C$. O valor esperado da unidade de saída i para o padrão de treinamento p é denotado $t_{ip}$. Para problemas de classificação com duas classes é usada apenas uma unidade de saída. Para problemas de classificação com C > 2 classes, C unidades de saída são usadas na RN, de

modo que, se um padrão p é classificado como pertencente à classe $C_x$, então $t_{xp} = 1$ e $t_{ip} = 0$, $\forall\, i \neq x$.

Para o treinamento da RN é utilizado o algoritmo BFGS [Battiti–1992] , desenvolvido a partir do algoritmo quasi-Newton [Dennis & Schnabel–1983], que busca minimizar a função erro de entropia-cruzada aumentada θ(w,v), descrita pela equação (4.3):

$$\theta(w,v) = F(w,v) - \sum_{i=1}^{C}\sum_{p=1}^{P}\left[t_{ip}\,\log(S_{ip}) + (1-t_{ip})\times\log(1-S_{ip})\right] \tag{4.3}$$

onde F(w,v) é a função de penalidade, definida pela equação:

$$F(w,v) = E_1\sum_{j=1}^{J}\left(\sum_{i=1}^{C}\frac{\beta v_{ij}^2}{1+\beta v_{ij}^2} + \sum_{k=1}^{K}\frac{\beta w_{jk}^2}{\beta w_{jk}^2}\right) + E_2\sum_{j=1}^{J}\left(\sum_{i=1}^{C} v_{ij}^2 + \sum_{k=1}^{K} w_{jk}^2\right) \tag{4.4}$$

com parâmetros $E_1$, $E_2$ e β positivos arbitrários.

A função de erro de entropia-cruzada é utilizada por melhorar a convergência do treinamento da RN em relação à função de menor erro quadrático[11] [Ooyen & Nienhuis–1992]. A função de penalidade é adicionada para incentivar a diminuição dos pesos, por meio da imposição de uma penalidade a pesos diferentes de zero. Cada conexão da rede com peso diferente de zero é penalizada. Minimizando a função de erro aumentada, espera-se que aquelas conexões que não sejam úteis para a classificação de padrões tenham pesos pequenos.

Setiono e Leow afirmam que o uso da função de erro de entropia-cruzada aumentada se mostrou efetivo na obtenção de RNs cujas conexões relevantes e irrelevantes podem ser facilmente identificadas pela magnitude de seus pesos. Em [Setiono–1997] é provado que o algoritmo BFGS converge mais rapidamente que o algoritmo *Backpropagation*.

[11] Função LMS (*Least-Mean-Square*), utilizada, entre outros, pelo algoritmo *Backpropagation.*

### 4.2.2 A CONSTRUÇÃO DA ÁRVORE DE DECISÃO

A partir da RN treinada, as unidades intermediárias relevantes são identificadas usando o método de ganho de informação e uma árvore de decisão com as unidades identificadas é construída.

Para construção da árvore de decisão, Setiono e Leow utilizam o C4.5 [Quinlan–1993], um algoritmo de aprendizado simbólico baseado no algoritmo ID3 [Quinlan–1986], que constrói uma árvore de decisão a partir de um conjunto de instâncias de treinamento. O algoritmo C4.5 (Figura 4.1) avalia, para escolha dos atributos dos nós da árvore, o ganho de informação associado a cada possível atributo, escolhendo aqueles que dividem o conjunto de dados da forma a minimizar a entropia.

```
Procedimento: C45(P, D)
Entradas:
   D = conjunto de dados para o nó corrente
   P = nó pai (se houver) do nó corrente

Início
   /* gera a árvore de decisão recursivamente */

   Se( D contém um ou mais exemplos, todos de uma mesma classe Cc) então
      Crie um nó N com conteúdo "Classe Cc"
   FimSe.

   Se( D não contém nenhum exemplo) então
      Crie um nó N com conteúdo "Classe X".
      Sendo X a classe mais freqüente no nó pai deste nó.
   FimSe

   Se( D contém exemplos pertencentes a várias classes) então
      Use o ganho de informação como heurística para dividir D em partições P1 e P2
      usando como critério os valores V de um único atributo T.
      F1 ← C45(N, P1)
      F2 ← C45(N, P2)
      Crie um no N com filhos F1 e F2 e conteúdo "Se (T < V)".
   FimSe

   Retorne N
Fim
```

Figura 4.1 Algoritmo C4.5

No caso do FERNN, a árvore de decisão é construída usando como critérios de decisão valores de ativação dos nós intermediários para as instâncias de treinamento corretamente classificadas pela RN. Esses valores de ativação são contínuos no intervalo $[0,1]$ pois são resultado da aplicação da função sigmóide sobre o somatório das entradas ponderadas de cada nó intermediário (equação 4.2).

Uma vez que as ativações dos nós intermediários apenas daquelas instâncias corretamente classificadas são usadas para criar a árvore de decisão, o número de instâncias em D usualmente é menor que o número de instâncias P do conjunto de treinamento.

Seja $\overline{P}$ o número de instâncias de treinamento corretamente classificadas. Para o nó intermediário j, seus valores de ativação $H_{jp}$ em resposta a cada instância de treinamento p, p=1,2,… $\overline{P}$ são ordenados em ordem crescente e divididos em dois grupos, $D_1=\{H_{j1},H_{j2},\ldots,H_{jq}\}$ e $D_2 = \{H_{jq+1},H_{jq+2},\ldots, H_{j\overline{P}}\}$ onde $H_{jq} < H_{jq+1}$. O ganho de informação é então calculado para cada possível combinação de $D_1$ e $D_2$, ou seja, para q variando de 1 a $\overline{P}$.

Para o cálculo do ganho de informação, inicialmente suponha que cada instância de treinamento de D seja classificada em uma das C classes e que $n_c$ é o número de instâncias na classe $C_C$. Calcula-se a informação esperada I(D) como:

$$I(D) = -\sum_{c=1}^{C} \frac{n_c}{N} \log_2 \frac{n_c}{N}$$

onde $N = \sum_{c=1}^{C} n_c$ é o número de instâncias do conjunto D.

O cálculo da informação esperada para os subconjuntos $D_1$ e $D_2$ é feito de forma equivalente:

$$I(D_1) = -\sum_{c=1}^{C} \frac{n_{c1}}{N_1} \log_2 \frac{n_{c1}}{N_1}$$

$$I(D_2) = -\sum_{c=1}^{C} \frac{n_{c2}}{N_2} \log_2 \frac{n_{c2}}{N_2}$$

onde $n_{cj}$ é o número de instâncias de $D_j$ que pertencem à classe $C_C$ e $N_j = \sum_{i=1}^{C} n_{ci}$ , j = 1, 2.

O ganho de informação conseguido com a divisão do conjunto de instâncias de treinamento D em dois conjuntos $D_1$ e $D_2$ é então calculado da seguinte forma:

$$\text{Ganho}(H_{jq}) = I(D) - [I(D_1) + I(D_2)]$$

O ganho normalizado GanhoN($H_{jq}$) é calculado como:

$$\text{GanhoN}(H_{jq}) = \left[ \text{Ganho}(H_{jq}) \Big/ -\sum_{j=1}^{2} (N_j/N) \log_2 (N_j/N) \right]$$

O C4.5 escolhe como raiz da árvore de decisão o teste que envolve o nó intermediário cuja ativação maximiza o ganho normalizado. Os critérios dos testes subseqüentes são definidos de modo similar, recursivamente, até que todos as instâncias de treinamento sejam classificadas.

A partir da árvore construída, a identificação das unidades irrelevantes é trivial: as unidades intermediárias que participam do teste de algum nó da árvore são relevantes. Qualquer outra unidade intermediária será irrelevante, podendo ser descartada.

A Figura 4.2 apresenta uma versão do algoritmo C4.5 utilizando a função de ganho de informação GanhoN($H_{jq}$). O algoritmo assume como entradas a RN treinada e o conjunto de dados utilizado no treinamento.

**Procedimento:** C4.5(D,P,Θ)

***Entradas:***

D /* o conjunto de instâncias de treinamento corretamente classificadas */
P /* o número de instâncias corretamente classificadas */
Θ /* o conjunto das unidades intermediárias da RN */

**Seja:**
Classe(p) /* classe associada à instância p no conjunto de treinamento */
Noh, RamoEsq, RamoDir /* nós da árvore, com Noh apontando para RamoEsq e RamoDir */

**Início**
**Se** (P=0) **então**
Noh ← "Classe mais freqüente no nó pai"
**senão Se** (Classe(p) = C, ∀ p ∈ D) **então**
Noh ← "Classe C"
s**enão**

$H_{max} \leftarrow H_{jp} \mid GanhoN(H_{jp}) > GanhoN(H_{xy}), \forall x \in \Theta, y \in [1,P]$
Noh ← "$H_{jp} > H_{max}$"
$D_1 \leftarrow \{x \mid x \leq p, \forall x \in D\}$ /* Conjunto de instâncias que não atendem à condição */
$D_2 \leftarrow \{x \mid x > p, \forall x \in D\}$ /* Conjunto de instâncias que atendem à condição */
$P_1 \leftarrow |D_1|$
$P_2 \leftarrow |D_2|$
RamoEsq ← C4.5($D_1$,$P_1$, Θ)
RamoDir ← C4.5($D_2$,$P_2$, Θ)
*FimSe*
Retorne Noh
**Fim**

*Sendo:*

$$GanhoN(H_{jq}) = \left[ Ganho(H_{jq}) \Big/ -\sum_{j=1}^{2} (N_j/N) \log_2 (N_j/N) \right]$$

Figura 4.2 Algoritmo C4.5 utilizando ganho de informação

### 4.2.3 IDENTIFICAÇÃO E REMOÇÃO DAS CONEXÕES IRRELEVANTES AOS NÓS DE ENTRADA

Além das unidades intermediárias irrelevantes, as conexões de entrada que tiverem pouca influência na decisão das unidades intermediárias relevantes poderão ser removidas, sem que isso afete o comportamento da RN. Devido ao fato do treinamento minimizar uma função de erro que penaliza pesos maiores que zero, pode-se esperar que conexões irrelevantes da RN tenham pesos relativamente pequenos.

Mais especificamente, para cada unidade intermediária j, um ou mais de seus pesos $w_{jk}$ de conexões das unidades de entrada podem ser suficientemente pequenos para poderem ser removidos sem afetar a acurácia geral da classificação. Setiono & Leow propõem duas proposições que caracterizam as situações que permitem a exclusão de conexões de entrada.

**PROPOSIÇÃO 1.** Seja $H_{jp} \leq H_{jt}$ para algum t, a condição para um nó da árvore de decisão. Defina $L = H_{jt}$, $U = H_{j\ t+1}$ (o menor valor de ativação maior que L), $D_L = \{\mathbf{x}_p \mid H_{jp} = \sigma(\mathbf{w}_j\mathbf{x}_p) \leq L\}$, $D_U = \{\mathbf{x}_p \mid H_{jp} = \sigma(\mathbf{w}_j\mathbf{x}_p) \geq U\}$. Seja S o conjunto de unidades de entrada cujas conexões para a unidade intermediária j satisfaçam a seguinte condição:

$$\sum_{k \in S} \left| w_{jk} \right| < 2(U - L) \tag{4.5}$$

e seja S' o complemento de S. Então, mudando a condição para:

$$H_{jp} \leq (L + U)/2 \tag{4.6}$$

as conexões das unidades em S para a unidade intermediária j podem ser removidas sem alterar a pertinência aos conjuntos $D_L$ e $D_U$.

**PROPOSIÇÃO 2.** Seja $H_{jp} > H_{jt}$ para algum t, a condição para um nó da árvore de decisão. Defina $L = H_{jt}$, $U = H_{j\ t+1}$ (o menor valor de ativação maior que L), $D_L = \{\mathbf{x}_p \mid H_{jp} = \sigma(\mathbf{w}_j\mathbf{x}_p) \leq L\}$, $D_U = \{\mathbf{x}_p \mid H_{jp} = \sigma(\mathbf{w}_j\mathbf{x}_p) \geq U\}$. Seja S o conjunto de unidades de entrada cujas conexões para a unidade intermediária j satisfaçam a seguinte condição:

$$\sum_{k \in S} \left| w_{jk} \right| \leq 2(U - L) \tag{4.7}$$

e seja S' o complemento de S. Então, mudando a condição para:

$$H_{jp} > (L + U)/2 \tag{4.8}$$

as conexões das unidades em S para a unidade intermediária j podem ser removidas sem alterar a pertinência aos conjuntos $D_L$ e $D_U$.

As duas proposições são semelhantes e resumem um resultado obtido do fato das unidades irrelevantes terem pesos pequenos: qualquer conjunto de conexões que seja tal que a soma total dos seus valores seja menor que o valor necessário para alterar a classificação de uma condição da árvore poderá ser removido. Para que a classificação da árvore não seja alterada, o valor do atributo usado no teste é alterado para a média entre o valor original $H_{jk}$ e o valor de $H_{j\,k+1}$.

A prova das proposições 1 e 2 estão no Apêndice A deste trabalho, nas seções A.1 e A.2, respectivamente.

### 4.2.4 A GERAÇÃO DE REGRAS SIMBÓLICAS

Por meio do cálculo da inversa da função sigmóide $\sigma^{-1}((L+U)/2)$ para cada nó não terminal da árvore de decisão, obtém-se condições que são combinações lineares dos atributos de entrada da RN.

Para um conjunto de dados com atributos contínuos, essas regras de decisão são apropriadas para classificar os padrões. Para um conjunto de dados com atributos discretos, é possível aplicar um passo adicional e extrair regras simbólicas, por meio da substituição das condições dos nós por regras na FND (Forma Normal Disjuntiva) ou M-de-N. Regras M-de-N são regras no formato:

*Se {pelo menos/exatamente/no máximo} M dentre N condições {C1, C2,..., CN} são satisfeitas, então...*

Regras M-de-N são utilizadas em outros trabalhos ([Towell & Shavlik–1993], [Craven & Shavlik–1996], [Setiono–2000], entre outros) e segundo Setiono & Leow,

podem ser mais concisas e mais gerais que regras na FND. No FERNN, regras M-de-N são geradas sempre que possível.

As proposições 3 e 4, apresentadas a seguir estabelecem o processo de geração das regras M-de-N a partir das condições da árvore de decisão.

**PROPOSIÇÃO 3.** Seja a condição em um nó de decisão dada por

$$w_1x_1 + w_2x_2 + \ldots + w_Kx_K \leq U \tag{4.9}$$

Considerando que a função $\lfloor U \rfloor$ representa o maior inteiro menor ou igual a U, seja $F = U - \lfloor U \rfloor$ e assuma como válido que:

1. Os pesos $w_i$ são positivos e podem ser expressos como $w_i = 1 + f_i$, com $f_i \in [0,1)$ ou seja, $w_i \in [1,2)$;
2. Os possíveis valores de $x_i$ são 0 ou 1;
3. A soma dos $\lfloor U \rfloor$ maiores $f_i$, i=1,2,…,K é menor ou igual a F.

Então a condição da equação 4.9 pode ser substituída pela regra equivalente:

Se no máximo $\lfloor U \rfloor$ de K entradas $\{x_1, x_2, \ldots, x_K\}$ for igual a 1

**PROPOSIÇÃO 4.** Seja a condição de divisão em um nó:

$$w_1x_1 + w_2x_2 + \ldots + w_Kx_K > U \tag{4.10}$$

Considerando que a função $\lceil L \rceil$ representa o menor inteiro maior ou igual a L, seja $F = L - \lceil L \rceil$ e assuma como válido que:

1. Os pesos $w_i$ são positivos e podem ser expressos como $w_i = 1 + f_i$, com $f_i \in [0,1)$ ou seja, $w_i \in [1,2)$;
2. Os possíveis valores de $x_i$ são 0 ou 1;
3. A soma dos $\lceil L \rceil$ maiores $f_i$, i=1,2,…,K é menor ou igual a 1 – F.

Então a condição (4.10) pode ser substituída pela regra equivalente:

Se pelo menos $\lceil L \rceil$ de K entradas $\{x_1, x_2, \ldots, x_K\}$ for igual a 1

As provas das proposições 3 e 4 são apresentadas no Apêndice A deste trabalho, nas seções A.3 e A.4, respectivamente.

Os pesos relevantes em uma rede treinada podem ter qualquer valor real. Setiono & Leow descrevem um procedimento de manipulação dos pesos da RN que facilita a aplicação das proposições anteriores, para a geração de regras M-de-N. O procedimento envolve dois passos simples: remover pesos negativos, e normalizar todos os pesos.

A remoção dos pesos negativos é feita por meio da substituição de uma entrada $x_i$ por seu complemento, sempre quando esta entrada estiver ligada a um peso negativo. Considere, por exemplo, o atributo "is_smiling" do problema dos Monges (*Monks* [Blake & Merz–1998]). Esse atributo pode assumir um de dois valores {yes,no} e é representado por um par de entradas binárias $\{x_1,x_2\}$, de modo que is_smiling = yes $\leftrightarrow$ $(x_1,x_2) = (1,0)$ e is_smiling = no $\leftrightarrow$ $(x_1,x_2) = (0,1)$. Assim, se $w_i$ é negativo, $x_1$ pode ser substituído pelo seu complemento que, nesse caso, é $x_2$.

O passo seguinte, a normalização, é feito simplesmente dividindo todos os pesos da unidade intermediária pelo menor deles.

Por exemplo, para o domínio Monk2, como mostrado em [Setiono & Leow–2000], uma regra M-de-N pode ser gerada como segue. Uma instância é classificada como um monge se a condição:

$$0.023 < \sigma(4.6x_3 + 4.2x_6 - 4.5x_7 + 4.4x_{11} + 4.6x_{15} - 2.6) \leq 0.43$$

é satisfeita.

Os pesos de $x_7$ e $x_{16}$ são negativos. Uma vez que ambos atributos têm apenas dois possíveis valores, 0 ou 1, os pesos negativos podem ser removidos, substituindo $x_7$ e $x_{16}$ por seus complementos: $x_7 = 1 - x_8$ e $x_{16} = 1 - x_{17}$, obtendo-se a regra:

$$0.023 < \sigma(1.1x_3 + x_6 + 1.1x_8 + x_{11} + 1.1x_{15} + 1.1x_{17} - 2.6) \leq 0.43$$

Aplicando a sigmóide inversa tem-se $\sigma^{-1}(0.023) = -3.75$ e $\sigma^{-1}(0.43) = -0.28$. Subtraindo o valor dos *bias* na desigualdade, obtém-se a regra simplificada:

$$1.9 < 1.1x_3 + x_6 + 1.1x_8 + x_{11} + 1.1x_{15} + 1.1x_{17} \leq 2.7$$

que pode ser transformada na seguinte regra M-de-N:

Se pelo menos 2 de 6 entradas {$x_3, x_6, x_8, x_{11}, x_{15}, x_{17}$} for igual a 1, então Monge = yes

usando os resultados das proposições 3 e 4.

Quando não é possível extrair regras M-de-N, o FERNN utiliza um outro algoritmo, o X2R [Liu & Tan–1995] para extrair regras na FND. O algoritmo X2R é caracterizado por seu autor como um algoritmo simples e rápido para ser aplicado a problemas com dados tanto numéricos quanto discretos. O pseudo-código do X2R, extraído de [Liu & Tan–1995], é apresentado na Figura 4.3.

```
Procedimento: X2R
Entradas:
   D = conjunto de dados /* sem duplicação */
   R = ∅ /* conjunto de regras geradas  inicialmente vazio */
Início

   /* Geração de regras */
   OrdenePorFreq(D)      /* ordena cjto de dados por freqüência */
   i ← 0
   Enquanto(D ≠∅)
   Início
      Gere a regra Ri para cobrir o padrão mais freqüente
      D ← D - todos os padrões cobertos por Ri
      R ← R + Ri
      i ← i + 1
   Fim

   /* Agrupamentos das regras */
   Agrupe_as_regras_de_acordo_suas_classes(R)

   /* Poda das regras  */
   Elimine as regras redundantes(R)
   Determine uma regra padrão(R)
Fim
```

Figura 4.3 Algoritmo X2R

## 4.3 EXPERIMENTOS USANDO O FERNN

Essa seção diz respeito ao uso do FERNN em um problema específico, para explorar suas funcionalidades, bem como em um problema comum, para fornecer subsídios para uma comparação entre os quatro métodos pesquisados.

### 4.3.1 EXEMPLO COMUM

A RN utilizada para a implementação do problema OU-Exclusivo (Figura 4.4) é idêntica à sugerida para a implementação do MACIE (Seção 3.5.2). Também foi utilizado o mesmo conjunto de instâncias de treinamento.

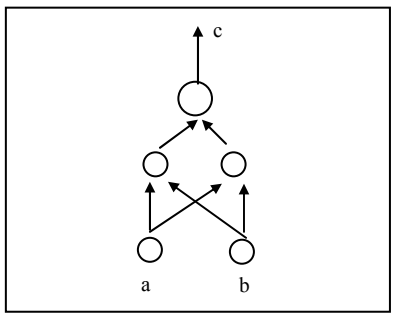



Figura 4.4 RN do OU-Exclusivo

A RN utilizada na implementação do FERNN tem duas diferenças com relação à RN utilizada para o MACIE: primeiro, as funções de ativação são sigmóides com saída entre 0 e 1, sem nenhum teste adicional; segundo, o algoritmo de treinamento aplicado utiliza função de penalidade para pesos diferentes de zero.

### 4.3.2 EXEMPLO ESPECÍFICO

O exemplo específico para a experimentação com o FERNN é um problema da área de Biologia Molecular. Esse exemplo foi escolhido para implementação do FERNN devido à dimensão da entrada: após adaptação para o FERNN, o problema é descrito por um total de 240 atributos binários de entrada.

A principal característica do FERNN é permitir a poda da RN sem necessidade de retreinamento. Com a aplicação do método a um problema de maiores proporções, principalmente em termos de número de atributos de entrada, espera-se evidenciar o ganho efetivo da capacidade de remoção de nós e conexões irrelevantes atribuída ao FERNN.

O conjunto de treinamento de *pontos de divisão em seqüências de genes de primatas* [Blake & Merz–1998] foi originalmente utilizado para auxiliar a avaliação do algoritmo de aprendizado KBANN [Towell & Shavlik–1993]. Cada instância de treinamento representa uma seqüência de DNA caracterizada por 60 nucleotídeos. A classe associada à instância caracteriza se a instância apresenta fronteira do tipo

a) "intron -> exon" (IE)

b) "exon -> intron" (EI)

c) Nenhum dos dois tipos (Nenhum)

Pontos de divisão são pontos em uma seqüência de DNA na qual DNA 'supérfluo' é removido durante o processo de criação das proteínas em organismos superiores.

O problema é a identificação dos limites entre exons (as partes da seqüência de DNA retidas após da divisão) e introns (as partes do DNA que são descartadas), dada uma seqüência de DNA. Esse problema pode ser dividido em dois subproblemas: o reconhecimento dos limites exon/intron (referidos como borda EI), e o reconhecimento dos limites intron/exon (borda IE).

Estudos apresentados em [Towell et al.–1991] indicaram que técnicas de aprendizado de máquina (redes neurais, *nearest neighbor*, KBANN) tiveram uma classificação igual ou melhor que a classificação baseada em comparação de padrões canônicos (apresentada em [Watson et al.–1987]). Os valores de todos os atributos de entrada são não numéricos e nominais: A, G, T, C, onde as letras A, G, T e C indicam nucleotídeos identificados na seqüência.

O conjunto de dados contém um total de 3190 instâncias, cada uma delas descrita com 60 atributos referentes a uma seqüência de DNA com 60 nucleotídeos, e uma classe associada. Como cada elemento da seqüência possui quatro valores possíveis (A, G, T, C), os atributos são convertidos em 240 entradas binárias.

A Tabela 4.1 apresenta as classes do problema, bem como o número de elementos em cada classe e a Tabela 4.2 apresenta a freqüência de cada nucleotídeo no conjunto de treinamento.

Espera-se que o método seja capaz de identificar, dentre os atributos de entrada inicialmente definidos, quais são relevantes e quais podem ser ignorados no tratamento do problema.

Os exemplos apresentados em [Setiono & Leow–2000] sugerem que apenas um pequeno número de unidades de entrada é suficiente para que uma RN classifique com sucesso um determinado padrão.

Tabela 4.1 Freqüência de classes

| Classe | Freqüência |
|---|---|
| EI | 767 (25%) |
| IE | 768 (25%) |
| Nenhum | 1655 (50%) |

Tabela 4.2. Freqüência de cada nucleotídeo no conjunto de treinamento

| Nucleotídeo/Classe | EI | IE | Nenhum |
|---|---|---|---|
| A | 22.153% | 20.577% | 24.984% |
| G | 31.415% | 22.383% | 25.653% |
| T | 21.771% | 26.445% | 24.273% |
| C | 24.561% | 30.588% | 25.077% |

## 4.4 RESULTADOS EXPERIMENTAIS USANDO O FERNN

Foram executados nove conjuntos de testes usando o FERNN. Cinco testes foram realizados utilizando o conjunto de treinamento para o exemplo comum ("OU-Exclusivo") e quatro testes foram realizados utilizando o conjunto de treinamento do exemplo específico para o FERNN ("Pontos de divisão em seqüências de genes de primatas"). A seguir são apresentados a configuração e os resultados dos testes.

### 4.4.1 CONFIGURAÇÃO DOS TESTES

Todos os testes foram realizados com os conjuntos de treinamento descritos na Seção 4.3 deste trabalho. Todos os testes foram realizados com os mesmos valores de configuração do algoritmo de aprendizado (Seção 4.2.1), definidos na Tabela 4.3. Os valores $\beta$, $E_1$ e $E_2$ são parâmetros utilizados na equação (4.4) definida na Seção 4.2.1.

Todos os testes utilizaram RN "Perceptron Multicamadas" com pesos iniciais definidos com valores aleatórios variando entre –0.002 e 0.002. O treinamento inicial para todos os exemplos foi realizado com as camadas adjacentes da RN totalmente conectadas. O algoritmo de aprendizado é diferente do proposto em [Setiono & Leow–2000]. A mudança realizada no algoritmo e sua justificativa são detalhadas no Apêndice B. A verificação do aprendizado neural nos testes de Capacidade é realizada usando 5-validação cruzada.

Tabela 4.3 Valores para o algoritmo de aprendizado

| Limite de épocas | Erro mínimo | Taxa de Aprendizado | $\beta$ | $E_1$ | $E_2$ |
|---|---|---|---|---|---|
| 50 000 | 0.1 | 0.1 | 0.001 | 0.000001 | 0.000001 |

### 4.4.2 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO COMUM

Para o exemplo comum, descrito na Seção 4.3.2, foram realizados cinco conjuntos de testes, agrupados como Capacidade e Qualidade:

Capacidade:

1) Aprendizado com função de erro entropia-cruzada;
2) Aprendizado sem função de erro entropia-cruzada;

Qualidade:

3) Qualidade das regras extraídas (Consistência) para diferentes RNs;
4) Qualidade das regras extraídas (Consistência) para mesma RN;
5) Qualidade das regras extraídas (Acurácia, Fidelidade e Compreensibilidade).

Os dois primeiros conjuntos de testes têm por objetivo verificar a capacidade de aprendizado do sistema, sobretudo com intuito de verificar se o aprendizado não é prejudicado pelas alterações da RN necessárias para a extração de regras. Os três conjuntos de testes restantes têm por objetivo verificar a qualidade das regras extraídas.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 4.4.2.1 e Qualidade – Subseção 4.4.2.2).

#### 4.4.2.1 TESTES PARA VERIFICAÇÃO DA CAPACIDADE DE APRENDIZADO

O FERNN impõe o uso da função erro de entropia-cruzada aumentada (Seção 4.1) para que os pesos irrelevantes das conexões da RN sejam reduzidos ao longo do treinamento.

Os resultados de testes apresentados na Tabela 4.4 mostram um pequeno aumento no erro médio e no número de épocas necessárias para a RN convergir com a função de erro entropia-cruzada mas, considerando o elevado valor de desvio padrão[12], a diferença não é significativa. A restrição da função de erro não chega a comprometer o aprendizado neural.

[12] O valor de desvio padrão, tanto para o número de épocas necessárias para convergência quanto para o erro médio obtido, é elevado devido a casos em que não houve convergência e o algoritmo de treinamento parou ao atingir o limite de 50 000 épocas.

Tabela 4.4 Resultados de teste para aprendizado do "OU-Exclusivo" no FERNN

| Função de entropia-cruzada aumentada? | Média /DesvioPadrão do número de épocas | Média / DesvioPadrão do Erro médio |
|---|---|---|
| Sim | 14 953.85 / 14 608.87 | 2.5% / 7.69 |
| Não | 12 581.9 / 14 020.33 | 1.25% / 5.59 |

#### 4.4.2.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

A Consistência das regras extraídas pelo FERNN é calculada utilizando o mesmo procedimento utilizado para o MACIE, apresentado na Seção 3.6.2.2.

Os resultados mostram que as regras geradas variam de acordo com a configuração dos pesos e, por isso, diferentes treinamentos geram regras diferentes. Não houve nenhuma coincidência entre os conjuntos de regras extraídos durante o experimento.

A mesma seqüência de testes foi realizada sem alteração na RN, ou seja, foi executado o processo de extração de regras diversas vezes sobre um mesmo conjunto de pesos. Nesse caso, houve coincidência de 100% dos conjuntos de regras gerados, ou seja, os conjuntos gerados são idênticos para todas as execuções realizadas. A Figura 4.5 apresenta a árvore gerada neste experimento.

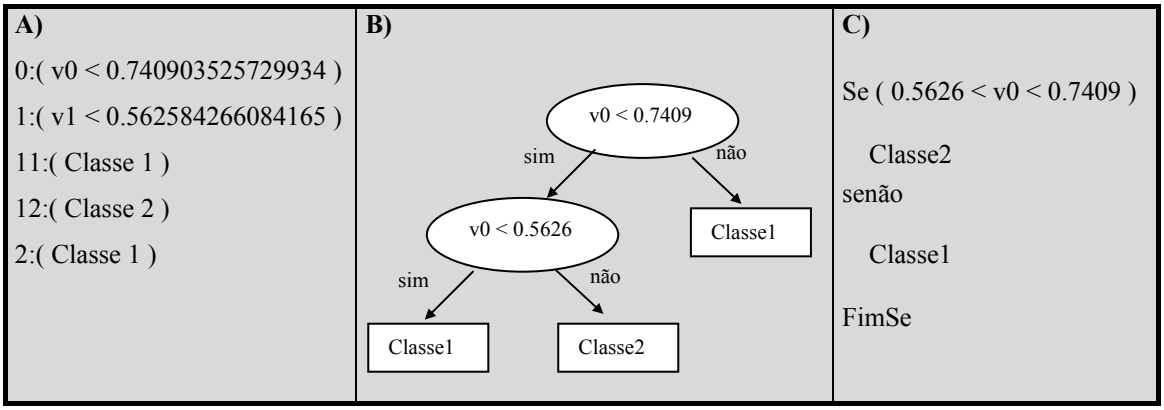



Figura 4.5 A) Saída do teste de Consistência; B) Representação gráfica equivalente; C) Regra de intervalo equivalente (v0 é o valor de ativação do primeiro nó da camada intermediária da RN utilizada no experimento).

Para o problema "OU-Exclusivo" com valores discretos, não é possível gerar conjuntos de teste diferentes do conjunto de treinamento. Por isso, a Acurácia e Fidelidade das regras extraídas são calculadas com o próprio conjunto de treinamento. Ambos os valores tiveram Acurácia e Fidelidade de 100% no experimento executado.

A Compreensibilidade das regras extraídas pelo FERNN foi avaliada considerando cada folha da árvore de decisão como sendo uma regra. O resultado do teste de

Compreensibilidade das árvores extraídas para o exemplo "OU-Exclusivo" é apresentado na Tabela 4.5.

A árvore de decisão utiliza como teste de decisão a ativação de nós da camada intermediária. A ativação de um nó da camada intermediária pode ser determinada a partir das entradas. A árvore expandida utiliza as entradas da RN, ao invés da ativação da camada intermediária, como teste nos nós da árvore.

Tabela 4.5 Compreensibilidade das árvores extraídas para o exemplo "OU-Exclusivo"

| **Média/DesvioPadrão número de folhas** | **Média/DesvioPadrão antecedentes por folha** | **Média/DesvioPadrão antecedentes por folha na árvore expandida** | **Compreensibilidade para árvore / árvore expandida** |
|---|---|---|---|
| 4 / 0 | 1 / 0 | 4 / 0 | 4 / 16 |

Os resultados obtidos apresentam valor de desvio padrão igual a zero, pois a árvore obtida apresentou o mesmo tamanho em todas as execuções do processo de extração. A medida de Compreensibilidade apresentada é calculada conforme apresentada na Seção 3.6.3.2.

### 4.4.3 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO ESPECÍFICO

Para o exemplo específico, descrito na Seção 4.3.1, foram realizados cinco conjuntos de testes, agrupados da mesma maneira que os testes para o exemplo comum:

Capacidade:

1) Aprendizado com e sem função de erro entropia-cruzada;
2) Verificação do aprendizado após a poda da RN;

Qualidade:

3) Qualidade das regras extraídas (Compreensibilidade);
4) Qualidade das regras extraídas (Acurácia e Fidelidade);
5) Qualidade das regras extraídas (Compreensibilidade, Acurácia e Fidelidade) após a poda da RN.

Os dois primeiros conjuntos de testes têm por objetivo verificar o impacto do uso da função de erro entropia-cruzada e do algoritmo de poda sobre a resposta da RN. Os três

testes restantes têm por objetivo avaliar a variação da qualidade das regras, com relação aos subgrupos "Compreensibilidade", "Acurácia" e "Fidelidade" da taxonomia ADT, perante a aplicação ou não da função de erro entropia-cruzada e do algoritmo de poda (Seção 4.1).

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 4.4.3.1 e Qualidade – Subseção 4.4.3.2).

#### 4.4.3.1 TESTES PARA AVALIAÇÃO DA CAPACIDADE DE APRENDIZADO

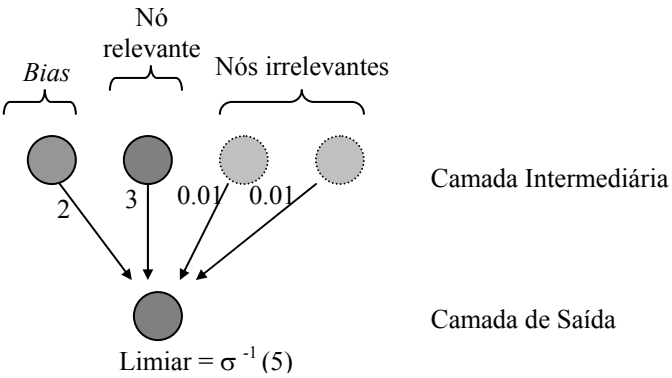
Os resultados dos testes para avaliação da capacidade de aprendizado da RN com e sem a restrição do FERNN são apresentados na Tabela 4.6.

Tabela 4.6 Aprendizado para o exemplo
"Pontos de divisão em seqüências de genes de primatas"

| Número médio de Épocas | Usa função de erro entropia-cruzada? | Poda da camada intermediária? | Poda da camada de entrada? | Média/DesvioPadrão do Erro da RN |
|---|---|---|---|---|
| 433 | Sim | Não | Não | 7.73% / 3.96 |
| 308 | Não | Não | Não | 7.17% / 3.2 |
| 433 | Sim | Sim | Não | 22.8% / 17.9 |
| 433 | Sim | Sim | Sim | 22.8% / 17.9 |

Os dois primeiros resultados da tabela mostram que não há variação significativa na capacidade de aprendizado da RN devido ao uso da função de erro de entropia-cruzada. Quanto aos algoritmos de poda, não há alteração após aplicação da poda dos valores irrelevantes da camada de entrada. O algoritmo de poda da camada intermediária, porém, altera a resposta fornecida pela RN.

Em [Setiono & Leow-2000] não é explicado em detalhes como deve ser utilizado o algoritmo de poda. Além dos resultados experimentais apresentados, uma avaliação simples, apresentada na Figura 4.6 mostra que a remoção dos nós irrelevantes da camada intermediária pode prejudicar a saída da RN.




Nesse exemplo, a remoção dos nós irrelevantes impede que a saída seja disparada, pois a soma do *bias* com o peso do nó relevante não excede o limiar de ativação.

A função $\sigma^{-1}(X)$ é o inverso da função de ativação do nó. Se o limiar é igual a $\sigma^{-1}(5)$, isso significa que a soma dos pesos de entrada deve ser maior que 5.

Figura 4.6 Remoção de nós da camada intermediária

Neste trabalho é sugerido que a poda pode ser realizada dessa maneira desde que, após a poda, as saídas da RN sejam determinadas por meio da árvore de decisão extraída da RN original. Os resultados apresentados na Tabela 4.7 mostram que a RN podada por meio do procedimento proposto neste trabalho apresenta mesma classificação que a RN original (com média de erro de 6.13%), porém com um número muito menor de conexões entre nós (286 conexões contra 14 400 conexões da RN original, uma redução de 98% do número de conexões).

Tabela 4.7 Avaliação da poda utilizando árvore de decisão extraída da RN

| **Média/DesvioPadrão do erro da RN original** | **Média/DesvioPadrão do erro da RN podada** | **Média/DesvioPadrão do número de pesos da RN original** | **Média/DesvioPadrão do número de pesos da RN podada** |
|---|---|---|---|
| 6.13% / 4.36 | 6.13% / 4.36 | 14 400 / 0 | 286.3 / 181.1 |

#### 4.4.3.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

Assim como no exemplo "OU-Exclusivo", assume-se, para verificação da Compreensibilidade, que cada folha da árvore de decisão gerada é equivalente a uma regra de um conjunto de regras. A Tabela 4.8 resume os resultados de teste para Compreensibilidade das regras extraídas para o exemplo "Pontos de divisão em

seqüências de genes de primatas". A medida de Compreensibilidade usada é a mesma definida na Seção 3.6.3.2.

Tabela 4.8 Compreensibilidade para o exemplo
"Pontos de divisão em seqüências de genes de primatas"

| Média/DesvioPadrão do número de folhas da árvore | Média/DesvioPadrão dos antecedentes por folha na árvore | Média/DesvioPadrão dos antecedentes por folha na árvore expandida | Média de Compreensibilidade da árvore / árvore expandida |
|---|---|---|---|
| 2.4 / 1.5 | 1 / 0 | 118.3 / 1.9 | 2.4 / 283.92 |

Os resultados da 5-validação cruzada realizada para verificação da Acurácia e Fidelidade das regras extraídas são apresentados na Tabela 4.9.

Tabela 4.9 Acurácia e Fidelidade para o exemplo
"Pontos de divisão em seqüências de genes de primatas"

| Conjunto verificado | Média/DP da Acurácia | Média/DP da Fidelidade |
|---|---|---|
| Conjunto de Treino | 90.47% / 8.25 | 100% / 0 |
| Conjuntos de Teste | 66.67% / 12.73 | 65.15% / 13.97 |

A acentuada redução dos valores médios de Acurácia e sobretudo de Fidelidade apresentados nos resultados do teste evidencia que as árvores de decisão extraídas são altamente dependentes do conjunto de padrões usado para o treinamento da RN.

A partir dos resultados apresentados na Seção 4.4.3, já foi possível concluir que a função de erro entropia-cruzada e o algoritmo de poda não têm influência significativa no aprendizado e na geração de regras do FERNN. O último teste realizado verifica a influência da função de entropia-cruzada na poda e, conseqüentemente, na medida de Compreensibilidade das regras extraídas. Os resultados são apresentados na Tabela 4.10.

Tabela 4.10 Compreensibilidade com uso da função de erro entropia-cruzada aumentada

| Função de erro entropia-cruzada aumentada? | Média/DesvioPadrão de neurônios intermediários podados | Média/DesvioPadrão de neurônios de entrada podados | Compreensibilidade da árvore expandida |
|---|---|---|---|
| Sim | 14 040 / 0 | 1.8 / 1.3 | 358.2 |
| Não | 13 716 / 344 | 0.3 / 0.6 | 683.7 |

A partir dos resultados apresentados para os testes sobre o FERNN, pode-se concluir que a poda da RN melhora a Compreensibilidade das árvores de decisão extraídas, e a utilização da função de erro entropia-cruzada aumentada facilita a poda, sobretudo dos nós da camada de entrada da RN, sem comprometer o aprendizado neural.

Os testes mostram ainda que as árvores de decisão extraídas são pouco compreensíveis e altamente dependentes da RN que as originou. Além disso, o algoritmo de poda é bastante eficiente, mas a RN fica dependente da árvore de decisão para gerar classificações sem perda de acurácia.

## 4.5 CONCLUSÕES SOBRE O FERNN

As críticas apresentadas com relação à abordagem do FERNN para extração de regras concentram-se nas restrições impostas pelo método, principalmente com relação ao espaço de busca do algoritmo que constrói a árvore, o qual é restrito às instâncias de treinamento corretamente classificadas.

O FERNN exige o uso de algoritmo de aprendizado e função de erro especiais para que os pesos irrelevantes sejam facilmente identificados [Setiono & Leow–2000]. Essa restrição torna o FERNN inadequado para qualquer aplicação em que já exista uma RN treinada com outra arquitetura ou outro algoritmo de treinamento. Porém não chega a ser uma restrição das mais severas, na medida em que a maioria dos métodos de extração de regras exige arquiteturas de RN específicas, e a alta performance atribuída ao método incentiva o treinamento de uma nova RN.

A maior crítica em relação ao FERNN refere-se ao espaço de busca do método. O processo de construção da árvore de decisão, assim como a viabilidade da eliminação de conexões irrelevantes, é altamente dependente do conjunto das instâncias de treinamento da RN. Por exemplo, a eliminação de conexões irrelevantes ocorre quando os pesos das conexões são menores que a diferença entre a ativação de dois padrões determinados como parte do critério de decisão da árvore de decisão. Porém, como o conjunto de padrões utilizados para a determinação deste critério é restrito ao conjunto de treinamento, nada garante que duas RNs com e sem as conexões 'irrelevantes' apresentarão a mesma classificação para um padrão novo, que não esteja no conjunto de treinamento.

Outro problema decorrente dessa restrição é risco do conjunto de regras representar o comportamento do conjunto de treinamento ao invés do comportamento total da RN.

Isso porque, devido ao fato da construção da árvore de decisão utilizar apenas os padrões corretamente classificados pela RN, a árvore e as regras geradas irão refletir somente o comportamento da RN nas regiões em que a classificação é correta. Desse modo, perde-se uma das vantagens da extração de regras, que é a possibilidade de se descobrir regiões da RN onde o treinamento está deficiente.

Essa restrição, porém, torna-se menos severa, caso o método seja adotado desde o início com o intuito de usar apenas um dado conjunto de treinamento.

# CAPÍTULO 5. O RULEX

## 5.1 INTRODUÇÃO

Como antecipado no Capítulo 1, uma das razões para a escolha do método RULEX [Andrews & Geva–2002] para a investigação conduzida neste trabalho foi o fato deste método ser uma abordagem relativamente recente e fazer uso de um novo modelo de redes neurais.

O RULEX trabalha com um modelo de RN diferenciado que, segundo seus autores, tem diversas vantagens sobre o modelo de redes *Perceptron Multicamadas* (MLP). A motivação do RULEX é ser um método de extração de regras que trabalha com um tipo de rede chamada de *Local Cluster* (LC).

O modelo de rede LC [Geva & Sitte–1993] é apresentado como uma alternativa ao modelo de rede MLP para implementação de aproximadores de função em hardware de baixo custo. Entre as características de uma rede LC, apresentadas em [Geva et al.–1998], está a possibilidade de extrair explicação simbólica de uma rede treinada; nesta referência, entretanto, não está mostrado como isso pode ser efetivamente realizado. A proposta do método RULEX, algum tempo depois ([Andrews & Geva–2002]), teve por objetivo viabilizar a capacidade explicativa (isto é, possibilidade de representação do conhecimento da rede como regras simbólicas) previamente conferida pelos autores à rede LC.

Apesar disso, as redes LC usadas com o método RULEX devem satisfazer algumas restrições que, conforme será discutido na Seção 5.5 deste trabalho, comprometem tanto a capacidade de aprendizado quanto o escopo de atuação da rede. A seguir, é apresentado o modelo de redes *Local Cluster*, com o intuito de facilitar a discussão posterior sobre o RULEX.

## 5.2 AS REDES LOCAL CLUSTER

Em [Geva & Sitte–1993] e posteriormente em [Geva et al.–1998] é proposta uma classe especial de redes MLP que utiliza grupos de funções localizadas, denominada de rede *Local Cluster* (LC), como uma alternativa a redes MLP tradicionais, para implementação de tarefas de aproximação de funções e classificação em hardware de baixo custo.

### 5.2.1 DESCRIÇÃO DA REDE LOCAL CLUSTER

Segundo seus autores, uma rede LC pode ser treinada tão rapidamente quanto redes de função de base radial (RBF) [Park & Sandberg–1991] sendo, porém, mais gerais que estas. As referências que propõem a rede LC comparam o desempenho dessa rede a uma rede MLP tradicional treinada com *Backpropagation* e evidenciam a maior capacidade de generalização da rede LC.

Geva denomina 'MLP tradicional' uma RN acíclica com uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída, com cada camada totalmente conectada à seguinte por conexões ponderadas com valores reais, função de ativação sigmóide ou similar e treinamento supervisionado.

A rede LC apresenta uma camada de entrada semelhante à camada de entrada da rede MLP. A camada de saída, porém, geralmente apresenta apenas um nó, e a camada intermediária é dividida em agrupamentos. Diferente da MLP tradicional, na qual todos os nós são semelhantes (executando o somatório das entradas, subtração do limiar de ativação e sigmóide), os nós da rede LC são diferenciados, havendo nós que efetuam exclusivamente funções sigmóides, somatórios, produtórios ou subtrações.

Em geral, os agrupamentos que compõem uma rede LC apresentam mais nós e são organizados de forma mais complexa que a camada intermediária de uma MLP projetada para um mesmo problema.

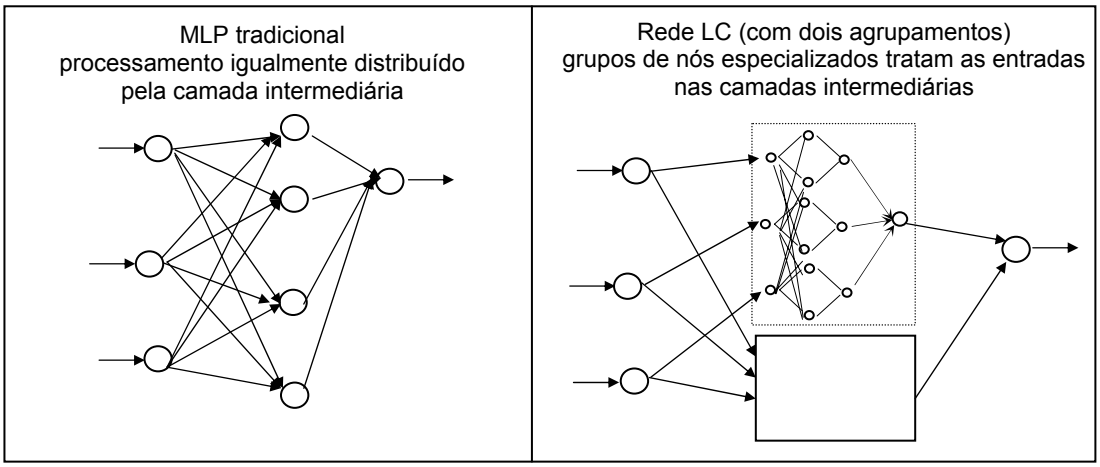



Figura 5.1 Comparação entre as arquiteturas de redes MLP e LC

A principal característica de uma rede LC, entretanto, está no modo como representa o domínio do problema. Em [Andrews & Geva–1996] é mostrado que tanto o problema de classificação quanto o de aproximação de funções por uma RN podem ser tratados como uma função no domínio do problema.

Considerando que todo problema tratado será a representação de uma função matemática, na MLP tradicional, cada nó da camada intermediária contribui, a priori, na representação de todo o espaço do domínio da função. Ou seja, antes de qualquer definição dos pesos das conexões (por treinamento ou construção a partir de conhecimento inicial), qualquer variação em uma das entradas acarreta uma variação em todos os nós da camada intermediária e, como conseqüência, acarreta uma variação na saída da rede.

Durante o treinamento de uma MLP tradicional, os nós relacionados a entradas irrelevantes podem ter os pesos de suas conexões diminuídos de modo a reduzir sua interferência na geração da saída da RN. Porém, esse é um processo bastante complexo, já que a correção na representação por uma determinada região do domínio pode implicar na deterioração da representação de uma outra região; por essa razão nem sempre o treinamento consegue modelar corretamente a função original do problema

Por outro lado, cada agrupamento da rede LC é uma função independente que representa uma região específica do domínio do problema. Desse modo, se uma determinada região do domínio da função não é corretamente representada por um agrupamento, apenas este agrupamento precisará ser treinado, e seu treinamento não afetará a resposta dada pela rede para as demais regiões.

Como resultado, Geva et al. mostram que uma rede LC apresentou uma representação muito mais fiel à função original que a MLP treinada com o mesmo exemplo, mesmo para regiões não cobertas pelos exemplos de treinamento. Outro argumento apresentado em [Geva et al.–1998] a favor da rede LC é que neurônios com resposta local são encontrados em diversas partes do sistema nervoso humano e, além disso, uma arquitetura de agrupamentos locais permite que circuitos eletrônicos simples representem os nós da RN, reduzindo o custo de uma possível implementação em hardware.

### 5.2.2 O FUNCIONAMENTO DA REDE LOCAL CLUSTER

A função sigmóide (Figura 5.2) é uma função clássica utilizada como função de ativação de redes MLP por ser uma função contínua que pode ser facilmente diferenciável e, principalmente, por apresentar um comportamento bastante característico, permitindo que valores reais de qualquer magnitude sejam mapeados para uma faixa bem menor de valores (por exemplo, valores entre $-\infty$ e $+\infty$ são levados, pela

função, a valores que estão em uma faixa mais facilmente tratável de valores, por exemplo aqueles entre –1 e 1).

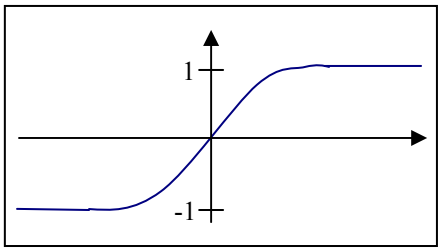


Figura 5.2 Gráfico da função sigmóide

A função de ativação sigmóide, na notação utilizada em [Geva et al.–1998], é dada pela equação 5.1.

$$\sigma(h,k) = \frac{1}{1+e^{-hk}} \tag{5.1}$$

$$h = \mathbf{w}^T\mathbf{x} - \theta = \mathbf{w}^T(\mathbf{x} - \mathbf{r})$$

Nesta função, h é o produto do vetor de pesos **w** pelo vetor de entradas **x**, menos o limiar de ativação θ; e k é um fator que define o ângulo da curva da função (se a curva será uma inclinação suave ou abrupta). A função h pode ainda ser reescrita para substituir o limiar de ativação θ por um vetor de referência **r**, que indica o centro do cluster, uma posição arbitrária no espaço de valores de **x**.

Na arquitetura de uma rede LC, duas sigmóides são combinadas de modo a formarem uma função semelhante à curva Gaussiana (Figura 5.3). Para isso, duas sigmóides idênticas têm seu centro deslocado por um fator d (por meio da adição ou subtração do valor da função h) e é calculada a diferença entre as sigmóides por meio de uma subtração simples, apresentada na equação 5.2. A equação 5.3 é uma simplificação da equação 5.2 para o caso de d = 1.

$$h^{\pm} = \mathbf{w}^T(\mathbf{x} - \mathbf{r}) \pm d \tag{5.2}$$
$$l(\mathbf{w}, \mathbf{r}, d, k, \mathbf{x}) = \sigma(h^+,k) - \sigma(h^-,k)$$

$$l(\mathbf{w}, \mathbf{r}, k, \mathbf{x}) = \sigma(k, \mathbf{w}^T(\mathbf{x} - \mathbf{r}) + 1) - \sigma(k, \mathbf{w}^T(\mathbf{x} - \mathbf{r}) - 1) \tag{5.3}$$

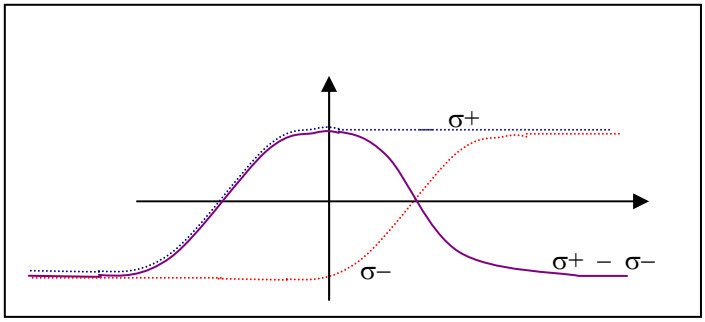



Figura 5.3 Diferença de duas sigmóides deslocadas em sentidos opostos

O RULEX permite que seja utilizado um vetor de entrada de tamanho arbitrário. Nos gráficos das Figuras 5.2 e 5.3 as sigmóides desenham linhas no espaço bidimensional, representando uma função de uma entrada em um espaço bidimensional. No caso de **x** ser um vetor de N-dimensional, entretanto, o gráfico da função l(**w**, **r**, k, **x**) é uma curva no hiperplano (N+1)dimensional. Por exemplo, na Figura 5.4 é apresentado o gráfico de l(**w**, **r**, d, k, **x**) para um vetor **x** bidimensional, no qual a curva gerada pela função é uma saliência no plano (espaço tridimensional).

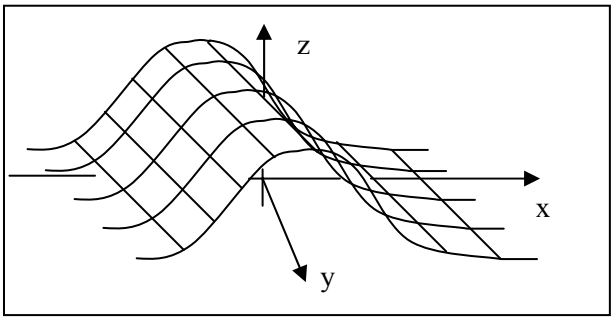



Figura 5.4 Saliência obtida pela diferença de sigmóides em função de 2 variáveis

Para facilitar a visualização, os próximos exemplos apresentados considerarão, sem perda de generalidade, vetores bidimensionais. Desse modo, o valor de l(**w**, **r**, d, k, **x**) varia em função de duas entradas (x, y).

Fazendo variar os valores de x e y, o reflexo no gráfico de l(**w**, **r**, d, k, **x**) é uma variação na orientação da saliência no plano XY. Por exemplo, gerando-se duas funções, l' e l" a partir de l, sendo l' = l(**w**, **r**, d, k, **x**) com **x** = (0,y) e l" = l(**w**, **r**, d, k, **x**) com **x** = (x,0), o resultado gráfico são duas saliências, sendo a saliência desenhada por l' orientadas sobre o eixo Y e a saliência desenhada por l" orientada sobre o eixo X.

Na rede LC, diversas funções l com um centro comum são somadas para formar uma nova função f que se aproxima de uma função de base radial. Por exemplo, somando as

funções l' e l" citadas, a função f resultante é a soma de duas saliências perpendiculares, cada uma alinhada a um dos eixos (X ou Y), resultando um gráfico semelhante ao apresentado na Figura 5.5.

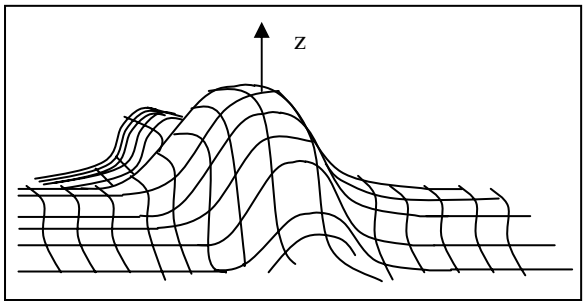



Figura 5.5 Intersecção de duas saliências perpendiculares

Desse modo, define-se a função f(**w**, **r**, k, **x**) (equação 5.4) como o somatório de N saliências que têm intersecção em um centro comum definido por **r**. O parâmetro **w** é uma matriz NxN construída a partir de N vetores de entrada $\mathbf{w}^i$, que geram N saliências no hiperplano (N+1)dimensional.

$$f(\mathbf{w}, \mathbf{r}, k, \mathbf{x}) = \Sigma\, l(\mathbf{w}^i, \mathbf{r}, k, \mathbf{x}) \tag{5.4}$$

A rede LC utiliza funções locais. Para que a função f seja local, ela deve ser manipulada de modo que apenas a região da intersecção tenha valores diferentes de zero. A função gerada não deve ter saliências laterais, apenas a intersecção no centro, sendo uma função local semelhante à função de base radial (Figura 5.6). Geva et al. sugerem duas maneiras de isolar a saliência central:

1. Utilizando, na equação 5.4, produtório ao invés de somatório para intersecção das saliências:

$$\mathcal{L}(\mathbf{w}, \mathbf{r}, k, \mathbf{x}) = \Pi\, l(\mathbf{w}^i, \mathbf{r}, k, \mathbf{x}) \tag{5.5}$$

2. Subtraindo da função um valor b e aplicando uma nova sigmóide:

$$\mathcal{L}(\mathbf{w}, \mathbf{r}, k, \mathbf{x}) = \sigma(\, f(\mathbf{w}, \mathbf{r}, k, \mathbf{x}) - b\,) \tag{5.6}$$

O valor b é selecionado para assegurar que o máximo valor da função f(**w**, **r**, k, **x**) coincida com o centro da região linear da sigmóide de saída $\sigma$:

$$b = n\left(\frac{1}{1+e^{-k1}} - \frac{1}{1+e^{k1}}\right) \tag{5.7}$$

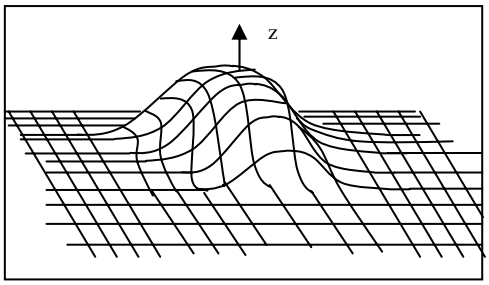



Figura 5.6 Figura formada pela intersecção de funções l(**w**$^i$, **r**, k, **x**) sem as saliências laterais

Em [Andrews & Geva–2002], é escolhida a segunda opção por ser a mais adequada à implementação em hardware[13].

As operações realizadas com as funções das equações 5.1, 5.3 e 5.5 podem ser sistematicamente traduzidas em arquiteturas de RN, como mostra a Figura 5.7.

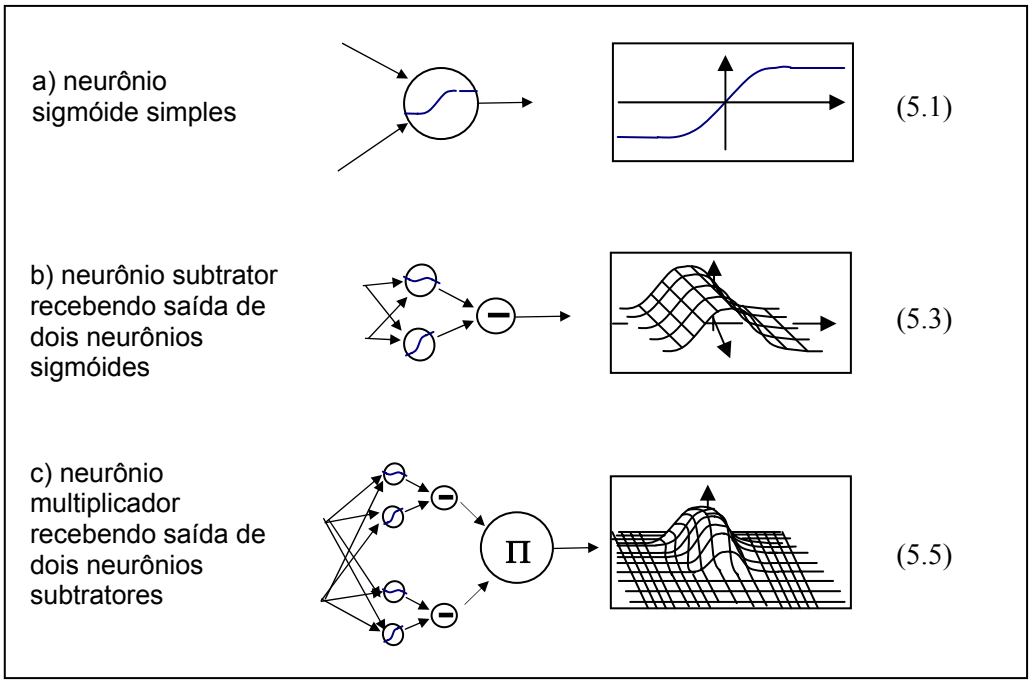



Figura 5.7 Arquitetura de RN para funções sigmóide, diferença de par de sigmóides, agrupamento de saliências

[13] Segundo Andrews & Geva, funções de subtração e sigmóide são mais facilmente implementáveis em hardware porque existem circuitos simples que realizam essas funções, enquanto que produtórios exigiriam a montagem de circuitos mais complexos. Para uma implementação em software, porém, é muito mais simples fazer uma única iteração multiplicando os valores.

A rede LC é, na realidade, um conjunto M funções $\mathcal{L}$ (**w**, **r**, k, **x**) ligadas a um único nó de saída por meio de conexões ponderadas com pesos $\mathcal{V}i$, i = 1, 2, .., M. Cada função $\mathcal{L}$ (**w**, **r**, k, **x**) é gerada por um agrupamento de nós e, durante o treinamento, novos agrupamentos podem ser adicionados. As 'saliências' formadas por cada função podem variar de forma (de acordo com o valor de k), de posição no espaço (de acordo com o valor de **r**) e de tamanho (de acordo com o valor do peso $\mathcal{V}i$, que representa a contribuição da i-ésima função $\mathcal{L}$ (**w**, **r**, k, **x**) na saída final da rede). A Figura 5.8 ilustra uma rede LC e o gráfico da função equivalente, com várias 'saliências' representando o domínio do problema.

A representação da função que expressa um problema é feita via o treinamento da rede, por meio do qual cada uma das 'saliências' são moldadas para adquirir o formato da função do problema em uma determinada região do espaço de busca. Em [Geva et al.–1998], é destacado que a capacidade que essas funções locais possuem de assumir praticamente qualquer formato (um plano, uma ondulação, ou mesmo um hiper-trapézio) permite à rede LC mapear facilmente a maioria dos problemas que podem ser representados por uma função.

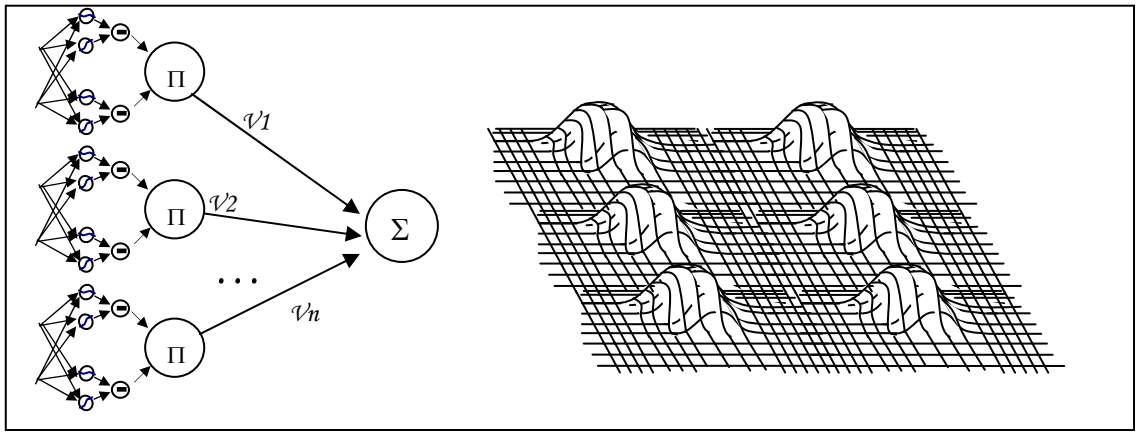



Figura 5.8 Arquitetura e gráfico da rede LC

### 5.2.3 O TREINAMENTO DA REDE LOCAL CLUSTER

Para o treinamento da rede LC, inicialmente é definida a quantidade e o posicionamento das funções $\mathcal{L}$ (**w**, **r**, k, **x**). Em seguida, cada função é individualmente treinada para adquirir o formato da função implícita nos exemplos de treinamento na região onde foi posicionada. O algoritmo utilizado para treinamento, mais complexo que o algoritmo de treinamento *Backpropagation*, é uma variação do algoritmo gradiente descendente com

taxas de aprendizado individuais adaptativas para cada parâmetro ajustável, apresentado em [Silva & Almeida−1990]. A Figura 5.9 apresenta o pseudocódigo do algoritmo de treinamento da rede LC.

---

**Algoritmo de treinamento da rede LC**

**Entradas:**

$\mathcal{V}_k i$ = peso da conexão de cada *cluster* com o nó de saída, na época k

$\mathbf{W}_k$ = matriz de pesos interna do *cluster*, no momento k

$\mathbf{r}_k$ = centro do *cluster*, no momento k

$\eta_v$ = taxa de aprendizado inicial do fator $\mathcal{V}_k i$

$\eta_w$ = taxa de aprendizado inicial do fator $\mathbf{W}_k$

$\eta_r$ = taxa de aprendizado inicial do fator $\mathbf{r}_k$

$\varepsilon$ = fator de comparação entre erros de classificação em duas épocas consecutivas, valor arbitrário

$m$ = fator de avaliação do erro de classificação em um conjunto de épocas, valor arbitrário

**Para cada** fator ajustável $\mathcal{V}$, **W** e **r faça**

**Início**

$y_p \leftarrow \mathbf{\Sigma}\, \mathcal{V}i * \mathcal{L}(\mathbf{w}, \mathbf{r}, k, \mathbf{x}_p)$ /* Compute a saída da rede para todas as instâncias de entrada $\mathbf{x}_p$ */

/* Calcule o ajuste do fator ajustável de acordo com a derivada parcial da função de erro sobre **x** */

Cálculo do erro:

$e_p \leftarrow (y_p - y_p^*)$, onde $y_p$ é a saída da RN para a entrada $\mathbf{x}_p$, e $y_p^*$ é classe associada a $\mathbf{x}_p$.

$erro_t \leftarrow \frac{1}{2}\sum_p e_p^2$ /* somatório dos erros de todas as instâncias de treinamento na época t */

Cálculo das derivadas parciais:

$$\frac{\partial V}{\partial x} \leftarrow \mathbf{\Sigma}\, erro_t [\, \mathcal{L}(X)' \,]$$

$$\frac{\partial W}{\partial x} \leftarrow \mathbf{\Sigma}\, erro_t [\mathcal{V}\, \sigma'(\Sigma\, l(X,Wn) - k)\, (\sigma'(X,d^+)) - (\sigma'(X,d^-))\, (X-r)\,]$$

$$\frac{\partial r}{\partial x} \leftarrow \mathbf{\Sigma}\, erro_t [\mathcal{V}\, \sigma'(\Sigma\, l(X,Wn) - k)\, \Sigma - w(\sigma'(X,d^-) - \sigma'(X,d^+))\,]$$

/* Aplique o ajuste de cada fator, ponderado pelas taxas de aprendizado */

$\Delta v_t \leftarrow -\eta_v * \mathcal{V}'$

$\Delta w_t \leftarrow -\eta_w * \mathbf{W}'$

$\Delta r_t \leftarrow -\eta_r * \mathbf{r}'$

$\mathcal{V}_{t+1} \leftarrow \mathcal{V}_t + \Delta v_t$

$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t + \Delta w_t$

$\mathbf{r}_{t+1} \leftarrow \mathbf{r}_t + \Delta r_t$

/* Atualize a taxa de aprendizado */

$y^+ \leftarrow$ fator de correção positiva do ajuste /* valor maior que 1 definido pelo usuário*/

$y^- \leftarrow$ fator de correção negativa do ajuste /* valor entre 0 e 1 definido pelo usuário */

**Para cada** fator de aprendizado ($\eta_v$, $\eta_w$ e $\eta_r$) **faça**

**Início**

**Se** (sinal($\Delta v_t$) = sinal($\Delta v_{t-1}$) ) **então** $\eta_t \leftarrow \eta_{t-1} * y^+$ **senão** $\eta_t \leftarrow \eta_{t-1} * y^-$

/* Verifique novamente o erro da rede */

saída_da_rede $\leftarrow \sum_{\text{clusters}} \mathcal{V}i * \mathcal{L}(\mathbf{w}, \boldsymbol{r}, k, \mathbf{x})$

$e_p \leftarrow (y_p - y_p^*)$

$erro_t \leftarrow \frac{1}{2}\sum_p e_p^2$ /* somatório dos erros de cada instância de treinamento */

/* Se o erro aumentou em relação ao anterior mais que o fator ε, reduza a taxa de aprendizado e retorne os valores anteriores dos fatores ajustáveis*/

**Se** ($erro_t > (erro_{t-1} * (1+\varepsilon))$) **então**

$\eta_t \leftarrow \eta_{t-1} * y$

$\mathcal{V}_{t+1} \leftarrow \mathcal{V}_t$

$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t$

$\mathbf{r}_{t+1} \leftarrow \mathbf{r}_t$

**FimSe**

**FimPara**

**Se** o erro não diminui mais que o fator *m* nas M últimas execuções **então**

aumente a capacidade de aprendizado naquela região:

- Adicione um novo agrupamento;
- Ajuste os fatores do novo agrupamento para cobrir a região de maior erro;

e continue o treinamento da rede.

**FimSe**

**Fim** /* fim do procedimento*/

Figura 5.9 – Pseudocódigo do algoritmo de treinamento da rede LC

## 5.3 DESCRIÇÃO DO ALGORITMO RULEX

O RULEX é proposto em [Andrews & Geva–2002] como um método que fornece capacidade explicativa a uma rede LC. Na realidade, em [Geva et al.–1998] já é citado o fato de uma rede LC permitir extração de regras sem, porém, detalhar o modo como isso poderia ser feito.

Segundo os autores, as regras extraídas pelo RULEX são simples e, portanto, de fácil entendimento por humanos. Ainda segundo os autores, o fato das regras poderem ser calculadas diretamente torna o processo de extração mais rápido que outras técnicas que empregam buscas exaustivas. Em [Andrews et al.–1995] é mencionado que, diferente do demais métodos de extração de regras de RN, o RULEX não necessita de heurísticas para restrição do espaço de busca.

O funcionamento do método RULEX é relativamente simples, porém pressupõe duas limitações à rede LC:

1) A rede deve ser do tipo *rede LC restrita* [Geva et al.–1998], sem conexões entre entradas de diferentes agrupamentos. Desse modo, os pesos das conexões da camada intermediária formam uma matriz em que apenas a diagonal principal é diferente de zero. Por exemplo, no caso de um espaço de busca bidimensional, o vetor **w** seria (0,x), (y,0), sendo x e y valores arbitrários a serem modificados durante o aprendizado neural. O resultado dessa restrição na arquitetura da rede LC é a redução do número de conexões, que pode ser visualizada na Figura 5.10. O resultado na função de ativação é o alinhamento da função l(**w**, **r**, d, k, **x**) com os eixos que definem as dimensões do espaço de busca.

2) Os agrupamentos da rede devem ter os valores ajustados de modo que as funções de ativação induzam 'saliências' quase retangulares, que se parecem com hiper-retângulos. Isso é conseguido por meio do ajuste da constante k na equação 5.4, de modo que as funções sigmóides individuais (equação 5.1) tenham um ângulo de curvatura próximo de $90^o$, aproximando-se de uma função binária, conforme particularizado na Figura 5.11, para o caso tridimensional. Com isso, a saída da rede pode ser tratada como uma variável binária com estados "maior que zero" ou "próximo de zero", permitindo o uso da rede LC como um classificador.

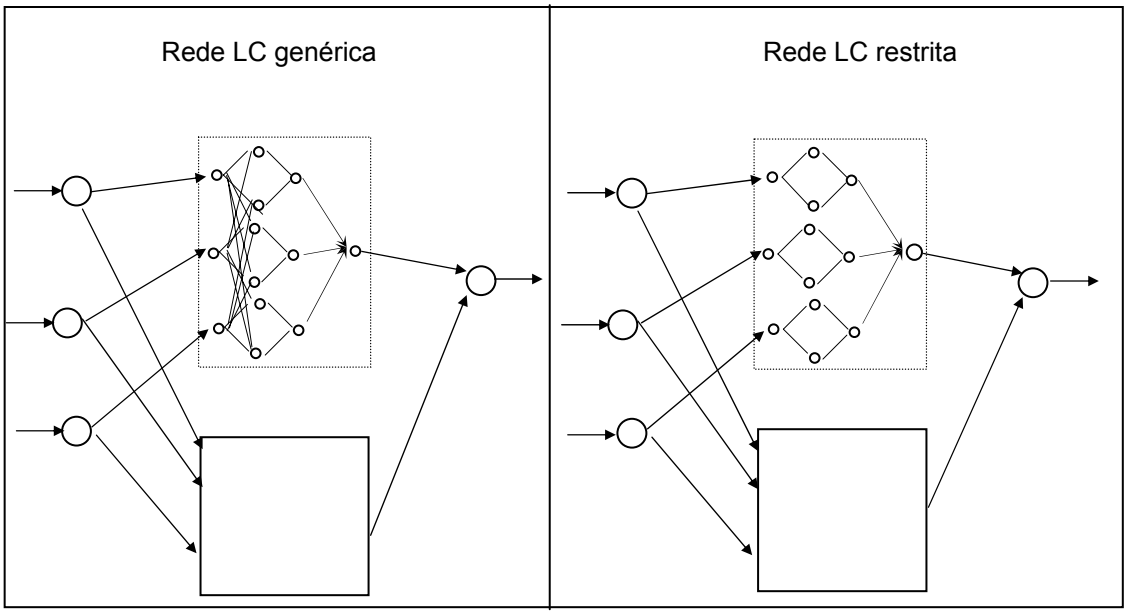



Figura 5.10 Redes LC genérica e restrita

As duas restrições em conjunto fazem com que a função de ativação seja alinhada aos eixos das dimensões e apresente um valor de saída regular maior que zero em um determinado intervalo e um valor regular próximo de zero em todo o espaço fora desse intervalo. Desse modo, pode-se estabelecer que a saída da rede LC será "maior que zero" ou "próximo de zero" por meio de intervalos nos eixos de N-1 dimensões de um espaço N-dimensional, do mesmo modo que poderia ser feito com hiper-retângulos. Por exemplo, na Figura 5.11, o espaço de saída "maior que zero" pode ser descrito através dos intervalos [x1,x2] e [y1, y2].

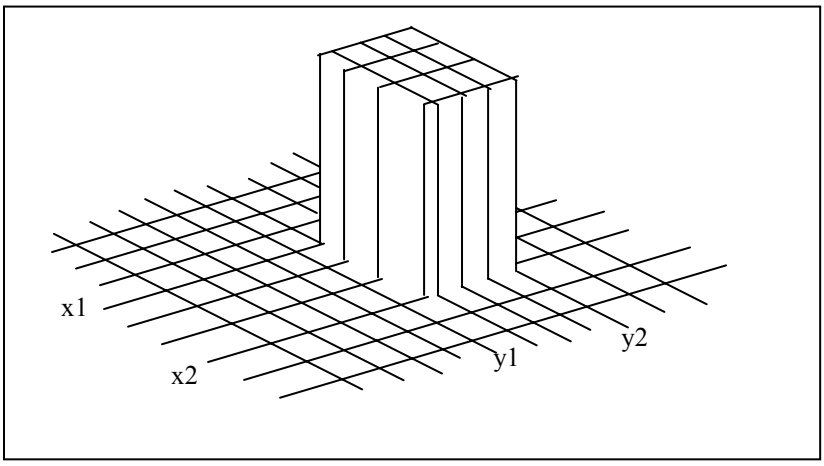



Figura 5.11 Gráfico de função de ativação retangular

### 5.3.1 O FUNCIONAMENTO DO RULEX

Dada uma rede LC restrita com função de ativação próxima a uma função retangular (como a particularizada na Figura 5.11 em um espaço tridimensional), a extração de uma regra no formato apresentado na Figura 5.12 pode ser feita por meio de um cálculo direto.

**SE** ( entrada1 ∈ [a1, b1]
**E** entrada2 ∈ [a2, b2]
...
**E** entradaN ∈ [aN, bN] )
**ENTÃO** valor identifica classe X

Figura 5.12 Formato de regra gerada pelo RULEX

Para isso, Andrews & Geva manipulam a função $\mathcal{L}(\mathbf{w}, \mathbf{r}, k, \mathbf{x})$, apresentada na equação 5.5, gerando a equação 5.7 que permite obter o valor do vetor de entrada **x** em função dos valores de **r**, b e k:

$$\mathbf{x} = \mathbf{r} - \ln\left(\frac{p - q \pm \sqrt{p^2 + q^2 - 2(x^2 + 1)}}{2\alpha}\right) * k^{-1} \tag{5.7}$$

onde

$p = (1 - \alpha)e^{b\,k}$

$q = (\alpha + 1)e^{-b\,k}$

$\alpha = \min(\sigma(k, (\mathbf{x} - \mathbf{r} - b)))$

$\min(\sigma(k, (\mathbf{x} - \mathbf{r} - b))) = \max(\sigma(k, (\mathbf{x} - \mathbf{r} - b))) - \ln\left(\frac{1}{O_T} - 1\right) * k_2$

max( . ) = a máxima ativação possível para qualquer função l "diferença de sigmóides"

$O_T$ = constante, limiar de ativação da função l

$k_2$ = constante, inclinação da função que determina $O_T$

A partir da equação 5.7, obtém-se os valores máximo e mínimo de x para cada agrupamento:

$$x_{máximo} = r - \ln\left(\frac{p - q - \sqrt{p^2 + q^2 - 2(x^2 + 1)}}{2\alpha}\right) * k^{-1}$$

$$x_{mínimo} = r - \ln\left(\frac{p - q + \sqrt{p^2 + q^2 - 2(x^2 + 1)}}{2\alpha}\right) * k^{-1}$$

Graças à restrição imposta à rede LC, a qual permite considerar cada agrupamento representando uma classe, é possível utilizar a informação do cálculo do $x_{mínimo}$ e $x_{máximo}$ de cada agrupamento $A_j$ para gerar regras no formato:

"para cada dimensão i : **SE** entrada $x_i \in [X_{i_{mínimo}}, X_{i_{máximo}}]$ **ENTÃO** $(x, y) \in$ classe $A_j$".

São geradas M regras, uma regra para cada agrupamento, com cada regra contendo N condições, uma para cada entrada da rede. Um exemplo simples, com M=2 e N=2 poderia gerar um conjunto de regras como:

SE ( $2 < x_1 < 4$ ) E ( $-1 < x_2 < 0$ ) ENTÃO Classe $A_1$
SE ( $2 < x_1 < 4$ ) E ( $2 < x_2 < 3$ ) ENTÃO Classe $A_2$

Porém, para funções mais complexas, mesmo com poucas entradas, o treinamento incremental pode gerar redes contendo muitos agrupamentos, que resultariam em um número excessivo de regras.

### 5.3.2 A SIMPLIFICAÇÃO DO CONJUNTO DE REGRAS

Dependendo da complexidade do problema, o conjunto de regras gerado pelo RULEX pode ser muito grande e apresentar diversas redundâncias que, possivelmente, vão reduzir sua compreensibilidade. O RULEX inclui um mecanismo que simplifica o conjunto de regras gerado, removendo regras redundantes, removendo condições antecedentes redundantes, e unindo pares de regras cujas condições antecedentes possam ser combinadas.

O procedimento para simplificação sugerido em [Andrews & Geva–2002] é detalhado a seguir:

a) Uma regra R é considerada redundante e pode ser removida se, para qualquer dimensão do espaço de entradas, o intervalo que define a condição de R está contido ou é igual ao intervalo que define a condição de uma outra regra S para a dimensão equivalente:

$$\forall\ 1 \le i \le n : [\ X_{Ri_{mínimo}}, X_{Ri_{máximo}}\ ] \subseteq [\ X_{Si_{mínimo}}, X_{Si_{máximo}}\ ]$$

Também é considerada redundante uma regra que possua uma condição fora do domínio de entrada para alguma dimensão de entrada:

$$\exists\ 1 \le i \le n : [\ i_{mínimo}, i_{máximo}\ ] \cap [\ X_{i_{mínimo}}, X_{i_{máximo}}\ ] = \varnothing$$

b) Uma condição de entrada é considerada redundante e pode ser removida da regra se tal condição engloba o domínio de entrada na dimensão para qual foi definida:

$$\exists\ 1 \le i \le n : [\ i_{mínimo}, i_{máximo}\ ] \subseteq [\ X_{i_{mínimo}}, X_{i_{máximo}}\ ]$$

c) Duas regras R e S podem ser combinadas se as condições forem idênticas para cada dimensão de entrada, exceto uma:

$$\forall\ 1 \le i \le n : (i \neq j) \wedge ([\ X_{Ri_{mínimo}}, X_{Ri_{máximo}}\ ] = [\ X_{Si_{mínimo}}, X_{Si_{máximo}}\ ])$$

Por exemplo, duas regras:

SE ( 2 < x < 4 ) E ( -1 < y < 0 ) ENTÃO ClasseA
SE ( 2 < x < 4 ) E ( 2 < y < 3 ) ENTÃO ClasseB

Seriam combinadas:

SE ( 2 < x < 4 ) ENTÃO
SE ( -1 < y < 0) ENTÃO ClasseA
SE ( 2 < y < 3) ENTÃO ClasseB

A terceira simplificação é mais específica e, aparentemente, é destinada a problemas de muitas entradas em que haja regras com várias condições antecedentes em comum.

## 5.4 EXPERIMENTOS USANDO O RULEX

Essa seção diz respeito ao uso do RULEX em um problema específico, para explorar suas funcionalidades, bem como do seu uso no problema comum, para fornecer subsídios para uma comparação entre os quatro métodos pesquisados.

### 5.4.1 EXEMPLO COMUM

Devido ao fato da rede LC tratar melhor problemas que podem ser representados como funções, o treinamento para o problema comum OU-Exclusivo foi feito de modo que a rede aprenda uma função *z* = f(x,y) onde x e y são entradas binárias e z é igual a 0 (zero) se o ambas as entradas (x,y) são iguais, e z é igual a 1 (um) se as entradas (x,y) são diferentes:

$$
\begin{aligned}
z = f(x,y): \quad & x = 0 \ \wedge \ y = 0 : \quad 0 \\
& x = 0 \ \wedge \ y = 1 : \quad 1 \\
& x = 1 \ \wedge \ y = 0 : \quad 1 \\
& x = 1 \ \wedge \ y = 1 : \quad 0
\end{aligned}
$$

Para evitar descontinuidades, devido a pontos sem valor definido pela função, é necessário estender a função da seguinte maneira:

$$
\begin{aligned}
z = f(x,y): \quad & x \in [0, 0.5] \ \wedge \ y \in [0, 0.5] : \quad 0 \\
& x \in [0, 0.5] \ \wedge \ y \in (0.5, 1] : \quad 1 \\
& x \in (0.5, 1] \ \wedge \ y \in [0, 0.5] : \quad 1 \\
& x \in (0.5, 1] \ \wedge \ y \in (0.5, 1] : \quad 0
\end{aligned}
\tag{5.8}
$$

Por meio do gráfico mostrado na Figura 5.12 é possível verificar que a função apresentada na equação 5.8 não apresenta descontinuidades.

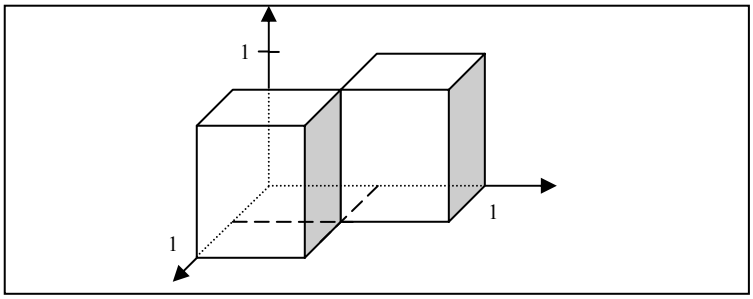


Figura 5.13 Gráfico da função OU-Exclusivo estendida

### 5.4.2 EXEMPLO ESPECÍFICO

O exemplo específico implementado para explorar e avaliar o funcionamento do RULEX é o problema de classificação de flores do tipo Íris. Os dados que caracterizam este problema estão disponíveis no repositório de dados da UCI [Blake & Merz–1998]. Esse domínio de dados, particularmente, tem sido utilizado como *benchmark* para avaliação de diversos sistemas de aprendizado, tais como os descritos em [Dasarathy–1980] [Neumann–1998] [Duch et al.–2001].

O conjunto de treinamento para classificação de flores Íris é composto por 150 instâncias de treinamento, divididas igualmente em três diferentes classes. Cada instância de treinamento é descrita por quatro atributos numéricos reais que representam as medidas da flor, bem como por uma classe associada.

Os atributos que descrevem as instâncias são:

- comprimento da sépala em centímetros,
- largura da sépala em centímetros,
- comprimento da pétala em centímetros,
- largura da pétala em centímetros.

As três classes, que caracterizam três subespécies de Íris são:

- Íris Setosa
- Íris Versicolour
- Íris Virginica

O conjunto de treinamento não possui instâncias de treinamento sem classe associada, nem atributos sem valor definido. Os atributos apresentam a estatística de correlação descrita na Tabela 5.1.

Tabela 5.1 Características dos atributos que definem o domínio de dados Íris

| Atributo em centímetros | Min | Max | Média | Desvio Padrão | Correlação de Classe |
|---|---|---|---|---|---|
| Comprimento da sépala | 4.3 | 7.9 | 5.84 | 0.83 | 0.7826 |
| Largura da sépala | 2.0 | 4.4 | 3.05 | 0.43 | -0.4194 |
| Comprimento da pétala | 1.0 | 6.9 | 3.76 | 1.76 | 0.9490 |
| Largura da pétala | 0.1 | 2.5 | 1.20 | 0.76 | 0.9565 |

Para facilitar a implementação do RULEX, duas alterações são feitas no exemplo original. Ambas as alterações são simplificações que objetivam facilitar o estudo do RULEX:

1) Ao invés de classificar três flores, a rede LC será construída para classificar apenas se uma flor pertence à classe "Íris Versicolour" ou não.
2) Ao invés dos quatro atributos disponíveis, serão utilizados apenas dois: "Comprimento da pétala" e "Largura da pétala", que são os atributos que apresentam a maior correlação de classes.

A primeira alteração se deve ao fato da rede LC apresentar apenas um neurônio de saída. A classificação em três classes exigiria três neurônios e, conseqüentemente, a construção de três redes distintas com arquiteturas similares. Como as três redes seriam independentes, o problema de classificação de três classes ou de uma classe apresentaria a mesma complexidade, e a construção de duas redes adicionais apenas acarretaria maior dispêndio de recurso computacional (principalmente memória).

A segunda alteração tem por objetivo reduzir o número de entradas para que o gráfico da função possa ser desenhado em um plano tridimensional, facilitando a visualização dos resultados. Foram escolhidos os dois atributos com maior grau de correlação de classes, de modo que a redução do número de atributos não reduza ou reduza o mínimo possível a informação necessária para a tarefa de classificação.

A definição da arquitetura inicial da rede LC é uma tarefa mais complexa que a definição da arquitetura de redes MLP tradicionais. A rede LC exige que as funções locais tenham os valores de **r** (que indicam a posição da função no espaço do problema) alterados de modo que as funções tenham seus centros localizados sobre pontos de algumas instâncias de treinamento, escolhidas ao acaso.

## 5.5 RESULTADOS EXPERIMENTAIS USANDO O RULEX

Foram executados nove conjuntos de testes sobre o RULEX. Cinco testes foram realizados utilizando o conjunto de treinamento para o exemplo comum ("OU-Exclusivo") e quatro testes foram realizados utilizando o conjunto de treinamento do exemplo específico para o RULEX ("Classificação de flores Íris"). A seguir são apresentados a configuração e os resultados dos testes.

### 5.5.1 CONFIGURAÇÃO DOS TESTES

Todos os testes foram realizados com os conjuntos de treinamento descritos na Seção 5.4 deste trabalho. Todos os testes foram realizados com os mesmos valores de configuração[14] do algoritmo de aprendizado LC, definidos na Tabela 5.2. Para treinamento da rede LC foi utilizado o software ELSY cedido pelo autor [Geva et al.–1995]. Detalhes sobre a implementação do RULEX são apresentados no Apêndice B, Seção B.3. A verificação do aprendizado neural nos testes de Capacidade é realizada usando 5-validação cruzada.

Tabela 5.2 Valores para o algoritmo de aprendizado

| Limite de épocas | Erro mínimo | Taxa de Aprendizado Inicial | k | K |
|---|---|---|---|---|
| 1 000 | 0.001 | 0.05 | 4 | 10 |

### 5.5.2 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO COMUM

Para o exemplo comum "OU-Exclusivo", descrito na Seção 5.4.2, foram realizados três conjuntos de testes agrupados da seguinte forma: no primeiro grupo, identificado como *Capacidade*, estão os testes que avaliam a eficácia da rede usada pelo método; no segundo grupo, identificado como *Qualidade*, estão os testes que avaliam a eficácia dos métodos via critérios da ADT.

Capacidade:

1) Aprendizado com arquitetura genérica;
2) Aprendizado com arquitetura restrita;

Qualidade:

3) Qualidade das regras extraídas (Consistência);
4) Qualidade das regras extraídas (Compreensibilidade);
5) Qualidade das regras extraídas (Acurácia e Fidelidade).

[14] k é o valor que define o ângulo da função l(**w**,**r**,k,**x**)
K é o valor que define o ângulo da função $\angle$ (**w**,**r**,k,**x**)

Os dois primeiros conjuntos de testes têm por objetivo verificar a capacidade de aprendizado do sistema. Os demais conjuntos de testes têm por objetivo verificar a qualidade das regras extraídas.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 6.5.2.1 e Qualidade – Subseção 6.5.2.2).

#### 5.5.2.1 TESTES PARA VERIFICAÇÃO DA CAPACIDADE DE APRENDIZADO

Para que o uso do RULEX seja possível, a rede LC deve ter arquitetura restrita (Seção 5.3). Os testes cujos resultados são apresentados na Tabela 5.3 avaliam o impacto do uso de arquitetura restrita no aprendizado neural.

Tabela 5.3 Resultados de teste para "OU-Exclusivo" e RN com arquitetura genérica e restrita

| Arquitetura | Média/DesvioPadrão de épocas até a convergência | Média/DesvioPadrão do erro | Média/DesvioPadrão do número de agrupamentos |
|---|---|---|---|
| Restrita | 1 000 / 0 | 0.017 / 0.003 | 18.6 / 1.96 |
| Genérica | 1 000 / 0 | 0.028 / 0.006 | 16.4 / 1.34 |

Os resultados mostram que, para o exemplo "OU-Exclusivo" houve pequena diferença entre as capacidade de aprendizado e número de agrupamentos de redes LC com arquitetura genérica e restrita. A rede LC com arquitetura restrita foi capaz de aprender o conceito com média de erro menor, porém precisou de um número ligeiramente maior de agrupamentos que a rede LC com arquitetura genérica.

De modo geral, pode-se observar que a restrição de arquitetura não apresenta grande impacto na capacidade de aprendizado da rede LC.

#### 5.5.2.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

A seguir são apresentados os resultados dos testes avaliando a Consistência, Acurácia, Fidelidade e Compreensibilidade das regras extraídas. As medidas utilizadas para avaliação dos critérios são apresentadas na Seção 3.6 e discutidas com mais detalhes no Capítulo 7.

- CONSISTÊNCIA DAS REGRAS EXTRAÍDAS: as regras extraídas pelo RULEX são dependentes dos pesos gerados durante o treinamento da rede LC. Diferentes seções de treinamento geram conjunto de regras potencialmente diferentes. Nos testes realizados com retreinamento da rede LC não houve nenhuma coincidência entre os conjuntos de regras gerados, ou seja, o valor de Consistência é igual a zero. Para conjuntos gerados a partir de uma mesma rede LC em várias execuções do processo de extração o valor de Consistência é igual a 100%.

- ACURÁCIA DAS REGRAS EXTRAÍDAS: as regras extraídas classificam corretamente 82.65 % (em média) dos padrões em um conjunto de padrões diferente do conjunto utilizado para treinamento.

- FIDELIDADE DAS REGRAS EXTRAÍDAS: 82.65 % (em média) dos padrões dos conjuntos de teste foram classificados da mesma maneira tanto pela regra quanto pela RN. Esse valor é igual ao valor de Acurácia porque o resultado gerado pela rede LC é idêntico ao conjunto de teste (ou seja, a rede apresenta 100% de acurácia no conjunto de teste).

- COMPREENSIBILIDADE DAS REGRAS EXTRAÍDAS: as regras avaliadas são as regras geradas a partir da árvore de decisão. Em média foram geradas 98 regras com 17.8 atributos cada. A medida de compreensibilidade (Regras × Atributos) resultante é 1744.4.

### 5.5.3 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO ESPECÍFICO

Para o exemplo específico, descrito na Seção 5.4.2, foram realizados quatro conjuntos de testes:

Capacidade:

1) Aprendizado com arquitetura genérica;
2) Aprendizado com arquitetura restrita;

Qualidade:

3) Geração de regras com/sem simplificação: Qualidade das regras (Compreensibilidade);

4) Geração de regras com/sem simplificação: Qualidade das regras (Acurácia e Fidelidade);

Os dois primeiros testes têm por objetivo avaliar a influência da restrição da arquitetura sobre a capacidade de aprendizado da rede LC. Os demais testes têm por objetivo avaliar a influência da simplificação na qualidade das regras, com relação aos subgrupos "Compreensibilidade", "Acurácia" e "Fidelidade" da taxonomia ADT.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 5.5.3.1 e Qualidade – Subseção 5.5.3.2).

#### 5.5.3.1 TESTES PARA AVALIAÇÃO DA CAPACIDADE DE APRENDIZADO

Os resultados apresentados na Tabela 5.4 mostram que a restrição da arquitetura não apresenta influência significativa sobre a capacidade de aprendizado da RN, com relação ao exemplo "Classificação de flores Íris". Tanto a média de erro de classificação quanto o número médio de agrupamentos da rede LC gerada apresentam diferença insignificante entre as arquiteturas restrita e genérica.

Tabela 5.4 Resultados de teste para o exemplo "Classificação de flores Íris"

| Arquitetura | Épocas para convergência | Média/DesvioPadrão do Erro | Média/DesvioPadrão do número de agrupamentos |
|---|---|---|---|
| Restrita | 1 000 | 0.009 / 0.0009 | 10.6 / 1.5 |
| Genérica | 1 000 | 0.005 / 0.0004 | 10 / 1.41 |

#### 5.5.3.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

As regras foram avaliadas com relação aos subgrupos "Compreensibilidade", "Acurácia" e "Fidelidade" da ADT, visando avaliar a influência da simplificação sobre as regras extraídas. As medidas utilizadas são as mesmas introduzidas na Seção 3.6 deste trabalho.

Tabela 5.5 Compreensibilidade, Acurácia e Fidelidade para o exemplo "Classificação de flores Íris"

| Simplificação das Regras | Média / DesvioPadrão da Compreensibilidade | Média / DesvioPadrão da Acurácia | Média / DesvioPadrão da Fidelidade |
|---|---|---|---|
| Sim | 836 / 122 | 60.2 / 8.47 | 60.2 / 8.47 |
| Não | 2240 / 261 | 62.2 / 6.76 | 62.2 / 6.76 |

Os resultados, resumidos na Tabela 5.5, mostram que a simplificação das regras não apresenta influência sobre a Acurácia e Fidelidade, mas influencia fortemente a Compreensibilidade das regras. O conjuntos de regras simplificados possuem, em média, 37% do tamanho do conjunto original.

## 5.6 CONCLUSÕES SOBRE O RULEX

O RULEX apresenta uma importante contribuição à pesquisa sobre extração de regras de redes neurais. O método de extração de regras a partir da função de ativação dos agrupamentos é uma idéia que foge completamente do paradigma de associação de neurônios a elementos da lógica simbólica. Isso permite que, diferente de outros métodos, o RULEX extraia as regras diretamente dos valores embutidos na rede sem que seja necessário utilizar processamento da rede[15].

Desse modo, as regras extraídas são completamente independentes da atribuição de entradas à rede, o que é positivo em dois aspectos: primeiro, as regras extraídas refletem o comportamento total da rede, e não apenas o comportamento para um dado conjunto de valores de entrada; segundo, elimina-se o problema de explosão combinatória, pioneiramente citado em [Fu–1991], na geração de entradas para extração das regras. Outros métodos apresentam apenas soluções paliativas para esse problema (como heurísticas para redução do espaço de busca [Fu–1991], ordenação de atributos [Krishnan–1996], uso do conjunto de treinamento [Setiono & Leow–2000], uso de informação de dependência funcional [Tickle et al.–1996]).

Em contrapartida, devido à sua característica local, a rede LC apresenta dificuldade para aprender conceitos que tenham dependências globais[16] [Geva et al.–1998]. Além

[15] O MACIE e o FERNN, por exemplo precisam que valores sejam aplicados à entrada da RN e saídas sejam computadas para que as regras sejam extraídas. Diversos outros métodos funcionam dessa maneira.
[16] A despeito disso, os autores argumentam que a maioria dos problemas reais não apresentam tais dependências.

disso, a complexidade da rede e do algoritmo de treinamento tornam o modelo pouco atraente para tratamento de problemas mais simples, que constituem muitos dos casos reais.

O maior problema do RULEX, porém, está nas restrições impostas à rede LC para a extração das regras. Apesar de não acarretar perda da capacidade de aprendizado, ao ter o formato da função local restrito de modo a se tornar semelhante a um hiper-cubo, a rede LC tem sua capacidade de representação reduzida. Em [Geva et al.–1998], é argumentado que a rede LC não sofre da 'maldição da dimensionalidade'[17] devido, principalmente, à flexibilidade do formato das funções locais. Com a restrição imposta pelo RULEX, entretanto, as funções perdem esta flexibilidade. A rede LC se degenera em um modelo semelhante ao utilizado pelo RuleNet [Tschichold & Gurman–1997] tornando-se assim sensível à 'maldição'.

Essa última observação leva à comparação do RULEX a métodos não conexionistas que trabalham com hiper-retângulos, e à suspeita de que um estudo mais detalhado talvez possa mostrar que o RULEX é uma forma de aplicação de aprendizado neural em um classificador não conexionista.

[17] A "maldição da dimensionalidade" (*curse of dimensionality*) é o problema decorrente da dificuldade em representar classes com distribuição complexa através de formas simples como hipercubos. Em termos grosseiros significa que, para representar um conjunto de treinamento com precisão absoluta, seriam necessários tantos hipercubos quantos fossem os exemplos do conjunto de treinamento.

# CAPÍTULO 6. A PROPOSTA TBNN

A abordagem TBNN (Tree Based Neural Network) [Ivanova & Kubat–1995] parte da premissa que uma descrição lógica aproximada de um determinado conceito pode ser aprendida por meio de um método simbólico de AM (particularmente árvore de decisão) e, então, facilmente transformada em uma arquitetura de rede neural, para ser subseqüentemente treinada pelo algoritmo *Backpropagation.* O objetivo é ter uma rede cuja precisão de classificação seja melhor que a da árvore de decisão que lhe deu origem.

## 6.1 INTRODUÇÃO

Diferente dos outros métodos apresentados e discutidos anteriormente neste trabalho, o método TBNN propõe o uso de métodos de aprendizado conexionista para refinamento de conhecimento aprendido por um método simbólico de aprendizado. O uso de uma árvore de decisão para definição da arquitetura de uma RN é proposto em outros trabalhos, como [Towell & Shavlik–1993] por exemplo, e levanta a questão da comparação entre as capacidades de representação de modelos simbólicos e conexionistas. Na Seção 6.5 é discutida a premissa assumida pelos autores de que árvores de decisão são incapazes de capturar conceitos que tenham granularidade refinada, enquanto que redes neurais fazem isso de maneira eficiente.

## 6.2 O FUNCIONAMENTO DO TBNN

### 6.2.1 CONSIDERAÇÕES INICIAIS

A Figura 6.1 mostra neurônios com função *step* modelando diretamente operações lógicas AND e OR. Se todas as entradas binárias do neurônio da esquerda forem iguais a 1, a soma $\sum_{i=1}^{N} w_i x_i = 2N\psi$ excede o valor do limiar de ativação $t_{AND} = \psi(2N - 1)$, o que produz valor 1 como saída do neurônio. De maneira similar, a saída do neurônio do lado direito é 1 sempre que pelo menos uma de suas entradas binárias for 1. Negação é modelada por pesos negativos.

Quando o valor do limiar de ativação for $t = \psi(2M - 1)$, com $M \leq N$, o neurônio modela um conceito M-de-N que é verdade sempre que pelo menos M das N entradas

binárias estão ativas (=1). Além disso, a cada entrada pode ser atribuído um peso diferente, de acordo com a relevância do atributo em questão. Por exemplo, se os pesos forem $w_1 = 1$, $w_2 = 1$, $w_3 = 3$ e o valor de limiar de ativação t = 1.5, o neurônio ficará ativo não apenas com a presença combinada de $x_1$ e $x_2$, mas também com a presença de $x_3$, mesmo se $x_1$ e $x_2$ estiverem desativadas.

Na realidade as entradas freqüentemente são numéricas (ao invés de binárias) e, conseqüentemente, um alto valor de entrada com um pequeno peso pode ter um efeito na saída de um neurônio maior do que aquele causado por um valor pequeno de entrada, com um peso alto. A organização de neurônios em uma rede e a substituição de limiares binários por funções diferenciáveis, tais como a função sigmóide, permite a modelagem de um número imenso de funções. Uma árvore de decisão pode ser representada por meio de diferentes redes neurais. Entretanto, redes neurais podem também capturar conceitos que são mais 'refinados' do que aqueles que podem ser expressos por expressões da lógica tradicional e, portanto, uma árvore de decisão pode apenas aproximá-los.

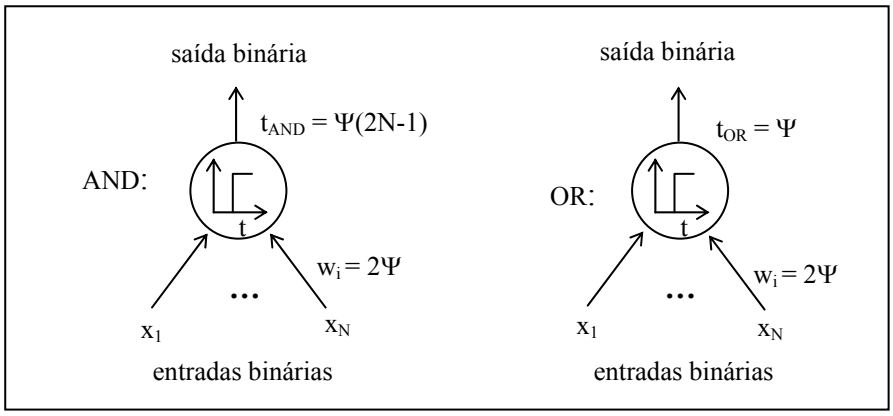



Figura 6.1 Codificação de funções booleanas em um neurônio

O principal objetivo da proposta TBNN é o de aumentar significativamente o desempenho de classificação de árvores de decisão, por meio de sua 'tradução' em uma rede neural, com um treinamento subseqüente via *Backpropagation* [Nicoletti–1998/2000]. A Figura 6.2 a seguir descreve o algoritmo TBNN. No início, a árvore de decisão é induzida e mapeada em uma rede neural parcialmente conectada. A seguir, a flexibilidade da rede é melhorada por meio da adição de conexões e da fuzificação de testes de decisão. Finalmente, a rede é treinada pelo *Backpropagation* com a função *momentum*.

**Procedimento:** TBNN
**Entradas:**
C = conjunto dos exemplos de treinamento
S = subconjunto dos exemplos de treinamento
ε = peso inicial pequeno
**Início**
/* Gera árvore de decisão a partir de S e converte a árvore em regras*/
A ← gera_arvore_de_decisão(S)
R ← converte_em_regras(A)

/* Transforma as regras em uma rede neural cujas únicas ligações são aquelas derivadas das regras. Neste passo os pesos das conexões não estão ainda definidos e a rede está apenas parcialmente conectada.*/
N ← converte_em_RN(R)

/* interconectar totalmente as camadas adjacentes da rede e atribuir a elas um peso inicial ε */
Adiciona_ligações(N, ε)

/* calcular os pesos ao longo das ligações regulares de maneira que a classificação realizada pela rede se aproxime da classificação realizada pela árvore */
Ajusta_pesos(N)

/* perturbar ligeiramente os pesos com o objetivo de facilitar seu ajuste */
Perturba_pesos(N)

/* transformar as funções *step* de todos os neurônios em funções sigmóides. Transformar os valores de atributos em valores de funções de pertinência a intervalos fuzzy */
Converte_Funções_Em_Sigmóides(N)
$C_{novo}$ ← Gera_Intervalos_Fuzzy(C)

**Fim**

Figura 6.2 Algoritmo em alto nível do TBNN

### 6.2.2 A INICIALIZAÇÃO E A CRIAÇÃO DA ÁRVORE

O algoritmo usado para a construção da árvore de decisão é uma variante do ID3 [Quinlan−1986] referenciada como NID3 [Ivanova & Kubat−1995], uma vez que representa uma versão numérica do ID3. A Figura 6.3 mostra um exemplo de árvore de decisão binária e o correspondente conjunto de regras extraído a partir dela.

Para cada atributo, a classificação atribuída aos exemplos (positiva e negativa, neste caso) decompõe seus valores em várias regiões e os candidatos a 'pontos de corte' estão nos limites entre as regiões. Para selecionar entre os vários candidatos, o conteúdo de informação de cada ponto de corte é determinado separadamente, como se cada um deles definisse um atributo binário. A Figura 6.3 mostra também como representar uma árvore de decisão em termos de descrições lógicas: cada conceito é representado por uma fórmula lógica (em FND), onde átomos são testes de valores de atributos. É

importante notar que alguns atributos comparecem mais do que uma vez na árvore e, a cada vez, são testados com relação a diferentes valores. Isso particiona os valores do atributo em intervalos; o único requisito para a classificação de novos exemplos é identificar a qual dos intervalos o valor pertence. Por exemplo, o escopo do atributo at1 é dividido nos intervalos: [0,0.2), [0.2,0.6) e [0.6,1].

Para determinar qual o teste de atributo em cada nó, o NID3 implementa a estratégia para tratamento de atributos numéricos proposta em [Fayyad & Irani–1992]. De acordo com essa estratégia os exemplos são ordenados por valores de atributos, como mostra a Figura 6.4.

Este fato sugere um método para a simplificação da regra. A cada um dos intervalos é associado um nome e o intervalo correspondente é tratado como uma variável booleana: o valor do atributo de um exemplo a ser classificado pertence (ou não) ao intervalo. A árvore da Figura 6.3 é então reformulada em termos das regras mostradas na Figura 6.5.

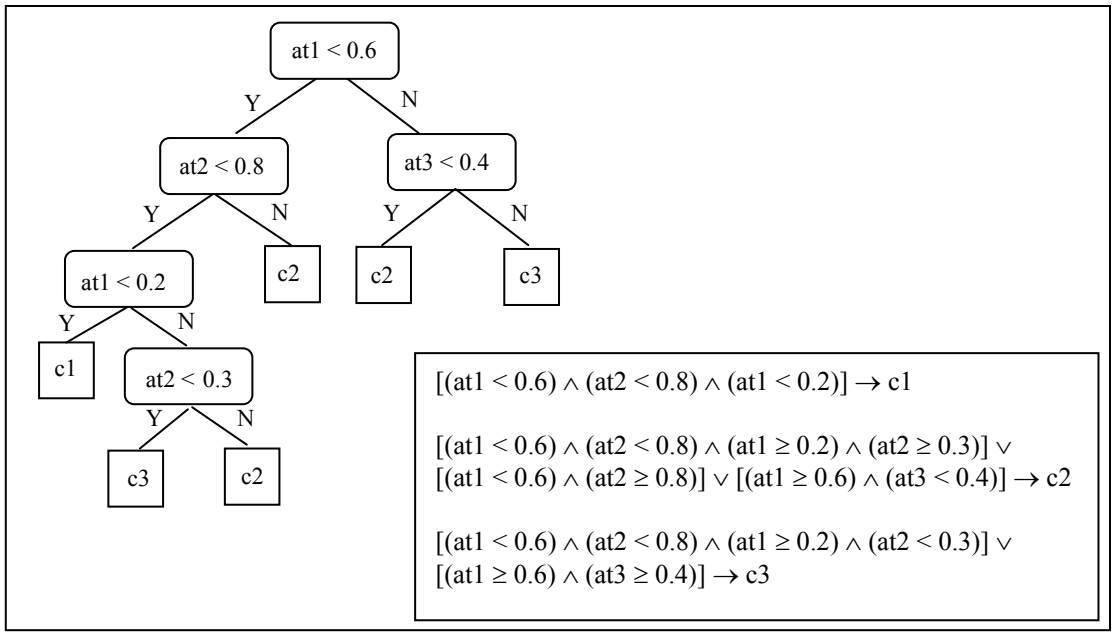



Figura 6.3 Árvore de decisão binária numérica, com profundidade 4 e com 5 nós de decisão

As regras contendo testes redundantes são simplificadas posteriormente. Por exemplo, a expressão (at1 < 0.6) ∧ (at2 < 0.8) ∧ (at1 < 0.2) é simplificada em (at1 < 0.2) ∧ (at2 < 0.8), que é reescrita como (a ∧ ¬f).

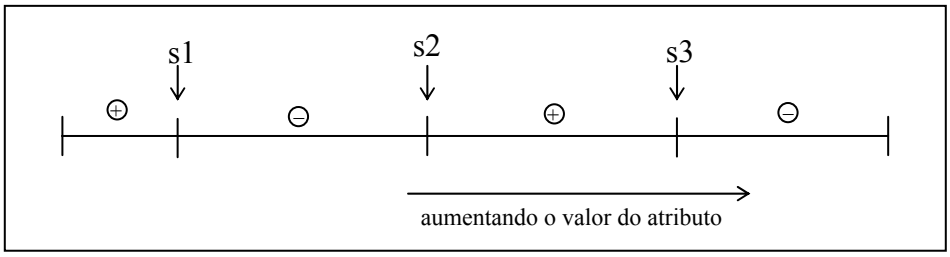



Figura 6.4 Exemplos ordenados por valor de atributo – quatro regiões são criadas, cada uma contendo exemplos com a mesma classe. Os candidatos a pontos de divisão são s1, s2 e s3

Uma outra simplificação que pode acontecer deriva do fato que, se um atributo é testado em dois ou mais ramos da árvore com relação a valores bem próximos (ver por exemplo at1 < 0.51 e at1 < 0.52), então apenas um dos testes é considerado e o outro é descartado. Entretanto, se tais 'testes semelhantes' comparecem num mesmo ramo nenhum é descartado.

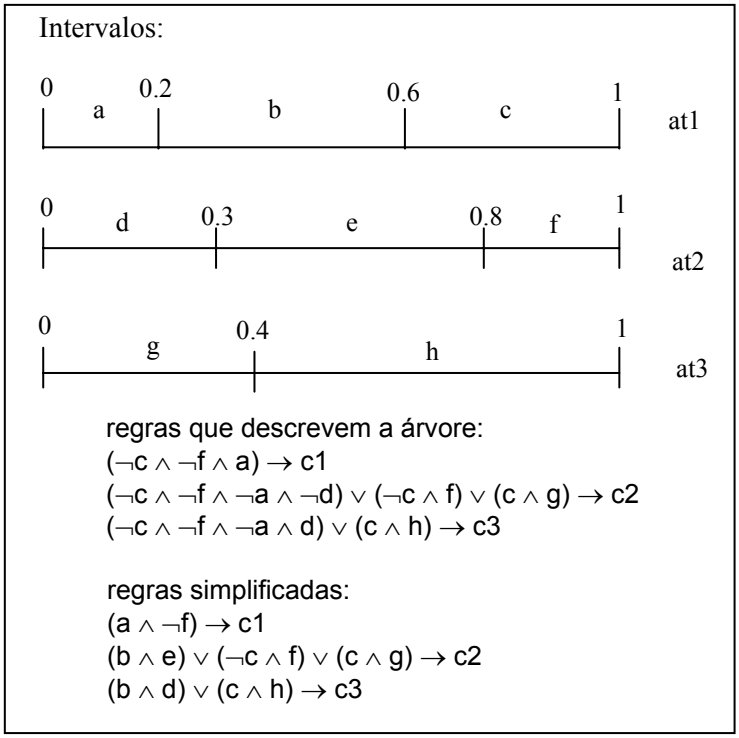



Figura 6.5 Reescrita das regras da Figura 6.3 usando variáveis booleanas

A Figura 6.6 mostra como as regras são mapeadas em uma rede neural com três camadas.

Cada camada é nomeada de acordo com a função que desempenha: camada de intervalos, camada-AND e camada-OR. A principal parte do algoritmo TBNN trata da construção de uma rede parcialmente conectada que modela o comportamento da árvore de decisão. Os neurônios da camada de intervalos distribuem os valores de pertinência-

a-intervalos aos respectivos neurônios da camada seguinte. É importante lembrar que as entradas da rede não são as mesmas que as entradas da árvore de decisão original. Os três atributos da Figura 6.3 'se transformam' em oito intervalos representados por oito variáveis booleanas de pertinência-a-intervalo, ou seja *a*, *b*, *c*, *d*, *e*, *f*, *g* e *h*.

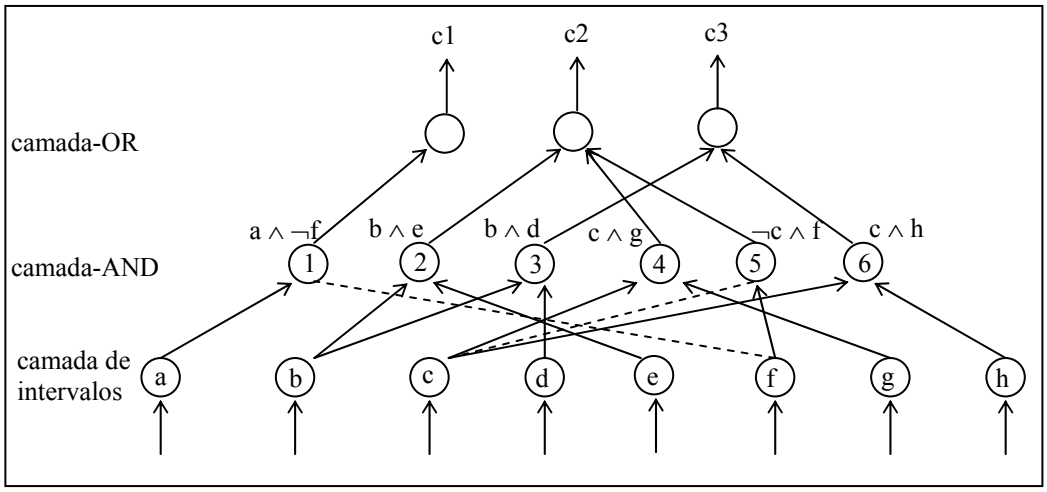



Figura 6.6 Rede neural criada a partir da árvore de decisão da Figura 6.2

Um exemplo descrito por at1 = 0.8 ∧ at2 = 0.4 ∧ at3 = 0.6 implicará valor 1 para os atributos *c*, *e* e *h* e valor 0 para os demais. Portanto, o número de entradas da rede será, usualmente, diferente do número original de atributos, e dado por

$$\sum_{i=1}^{N}(W_i+1)$$

onde N é o número de atributos que aparecem na árvore e $W_i$ é o número de testes na árvore de decisão, que envolvem o i-ésimo atributo (no exemplo sendo considerado, $W_1 = 2$, $W_2 = 2$ e $W_3 = 1$). A transformação dos atributos numéricos em variáveis booleanas de pertinência-a-intervalo é realizada como um passo anterior à construção da rede.

A camada-AND modela conjunções. Por exemplo, (a ∧ ¬f) é modelada pelo neurônio identificado por '1' (na Figura 6.6), o qual tem duas entradas: uma que se origina no neurônio que representa o intervalo identificado por 'a' e outra que se origina no neurônio que representa o intervalo *f*. Linha pontilhadas representam pesos negativos e, no exemplo, modelam a negação de *f* e de *c*.

No exemplo mostrado nos parágrafos anteriores, as entradas *c* e *h* irão ativar o neurônio '6', enquanto que todos os outros neurônios nesta camada ficarão desativados. Em geral, cada unidade da camada intermediária (camada-AND) corresponde a um

caminho da raiz da árvore a uma folha. Portanto, a rede neural terá T+1 unidades intermediárias, onde T é o número de nós não-terminais (ou seja, nós de decisão) da árvore. A camada-OR contém um neurônio por conceito. As entradas para esses neurônios são também derivadas das regras obtidas a partir da árvore de decisão. Pode ser facilmente verificado que o exemplo at1 = 0.8 $\wedge$ at2 = 0.4 $\wedge$ at3 = 0.6 irá ativar o neurônio de saída c3 e todos os outros neurônios de saída ficarão desativados.

No caso especial em que cada atributo aparece na árvore apenas uma vez, N = T e a rede neural correspondente terá complexidade mínima. Tal rede terá 2N neurônios de entrada, N+1 neurônios intermediários e K neurônios de saída, onde K é o número de classes.

Após o mapeamento da árvore na rede, para facilitar o refinamento posterior pelo método de gradiente-descendente, as funções de transferência de todos os neurônios (incluindo aqueles na camada de intervalos) são transformadas em funções sigmóides, ou seja, $f_i(v_i) = 1/(1+e^{-hvi})$, onde $v_i$ é a entrada do neurônio e h é um parâmetro definido pelo usuário. A fase de classificação adota a estratégia "o vencedor leva tudo". O exemplo é classificado como uma instância do conceito i, onde i é o índice do neurônio OR com a maior saída.

### 6.2.3 O REFINAMENTO DA REDE

Numa abordagem mais simplificada, as operações lógicas de conjunção e disjunção podem ser modeladas como sugerido na Figura 6.1. Essa simplificação, entretanto, faz com que a rede apenas emule as classificações da árvore de decisão original. Para fazer uso do potencial de redes neurais, um novo grau de liberdade deve ser adicionado (é importante notar que a rede está conectada apenas parcialmente e que todas as conexões a um neurônio têm o mesmo peso). Primeiro, os pesos são perturbados ligeiramente e, então, a rede é refinada usando algum método baseado em gradiente-descendente. Para fazer isso o TBNN usa o *Backpropagation* com a função *momentum*. Suponha que um exemplo do i-ésimo conceito é usado para treinamento. Os pesos são atualizados de acordo com a seguinte fórmula:

$$w_{t+1} = w_t + \eta.\Delta.x + \mu(w_t - w_{t-1}) \qquad (3.1)$$

onde:

$w_t$ : peso no instante t

$\Delta$ : erro de classificação

x : saída anterior do neurônio

$\eta$ : velocidade de aprendizado

$\mu$ : constante momentum.

Para prevenir um excesso de treinamento, o *Backpropagation* do TBNN usa um mecanismo simples: o conjunto de treinamento é dividido em dois conjuntos, um para o treinamento da rede e outro para o teste *online*. Após cada 'varrida' dos exemplos de treinamento (ou seja, após cada época), o sistema testa o desempenho no conjunto de teste. O treinamento pára quando a precisão no conjunto de teste diminui. Um passo importante para aumentar a flexibilidade da rede é conectar completamente as camadas adjacentes em uma maneira *feedforward*[18], o que facilita a descoberta de relações entre atributos que não foram identificadas pelo NID3. A adição de conexões adicionais, entretanto, complica o mecanismo que estabelece os pesos iniciais e os valores de limiares de ativação.

Considere o i-ésimo neurônio que não seja de entrada, com n conexões positivas regulares e m conexões negativas regulares. Suponha que k conexões tenham sido adicionadas, como mostra a Figura 6.7. Pesos associados às conexões positivas regulares ao i-ésimo neurônio AND são inicializados como $w_{i,\alpha.}$ Pesos associados às conexões negativas regulares ao i-ésimo neurônio AND são inicializados como $-w_{i,\alpha}$. Pesos associados às conexões positivas regulares ao i-ésimo neurônio OR são inicializados como $w_{i,\beta.}$ Pesos associados às conexões negativas regulares ao i-ésimo neurônio OR são inicializados como $-w_{i,\beta}$. Todos os pesos adicionais são estabelecidos como $\varepsilon$ (um valor bem pequeno). Os limiares de ativação dos respectivos neurônios são notados por $t_{i\alpha}$ e $t_{i\beta}$. Note que, para um número finito de entradas, o valor da sigmóide nunca atinge 0 ou 1. Portanto, a noção de estado ativo (equivalente à saída 1, no caso da função-step), deve ser definida por limiares de ativação. Sob essa perspectiva, um neurônio é considerado ativo se sua saída for $f(x) > A$, onde x é a soma ponderada de suas entradas e $A \in (0.5,1]$ é uma constante definida pelo usuário. De maneira similar, o neurônio está inativo se $f(x) < 1-A$. Todos os outros estados são considerados indefinidos. O neurônio estará ativo se a soma ponderada de suas entradas exceder um valor crítico $x = \psi$. Resolvendo a equação $A = \frac{1}{1+e^{h\Psi}}$ para $\psi$, obtém-se $\psi = (1/h)$

[18] Com conexões entre neurônios de cada camada para a camada seguinte, de modo similar ao apresentado para a arquitetura de Rede Discriminante Linear, descrita na Seção 3.2.1 deste trabalho.

ln(A/(1-A)). Devido à simetria da função sigmóide, o neurônio estará inativo se a soma ponderada de suas entradas for menor que $-\psi$.

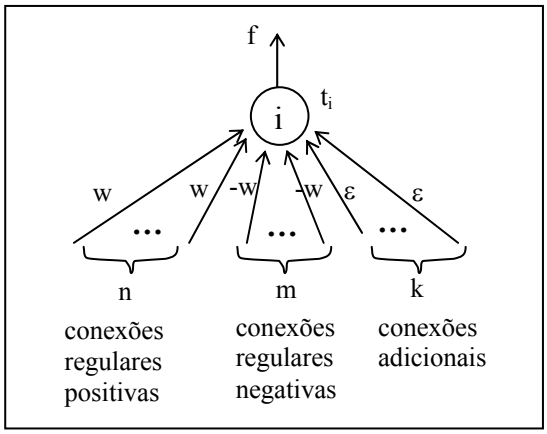



Figura 6.7 Entradas de um neurônio intermediário ou de saída, no TBNN

### 6.2.4 AS CONEXÕES AND-OR

Devido à lógica codificada na árvore de decisão, todas as entradas na camada OR são positivas e, portanto, m = 0. Este fato simplifica as respectivas equações. Se a camada OR for modelar a disjunção original de folhas na presença de conexões adicionais, então as seguintes condições devem ser satisfeitas:

- o i-ésimo neurônio OR deve se tornar ativo quando a árvore original produz a respectiva classificação (correspondente à classe representada pelo neurônio OR). Isso significa que suas entradas, decrementadas pelo limiar $t_{i,\beta}$ devem ser maior do que $\psi$ se pelo menos um dos neurônios AND ao longo das conexões regulares está ativo (alimentando o neurônio OR com entrada $(1*A*w_{i\beta})$), mesmo se todas outras conexões regulares fornecem entradas zero $((n-1)*0*w_{i\beta})$ e se as conexões adicionais fornecem entradas negativas máximas $(-k*\varepsilon)$
- o i-ésimo neurônio OR deve estar inativo sob as condições que a árvore original produz a respectiva classificação. Isto significa que suas entradas, decrementadas pelo limiar $t_{i,\beta}$ deve ser menor do que $-\psi$, se todos os neurônios AND ao longo das conexões regulares têm como saída o máximo valor para os quais eles são ainda considerados

inativos ($n*(1-A)*w_{i\beta}$), mesmo se as conexões adicionais fornecem entradas positivas máximas ($k*\varepsilon$)

Essas duas exigências são traduzidas nas duas desigualdades:

$$(1*A*w_{i\beta}) + ((n-1)*0*w_{i\beta}) - k\varepsilon - t_{i,\beta} \geq \psi \tag{6.2}$$

$$n*(1-A)*w_{i\beta} + k\varepsilon - t_{i,\beta} \leq \psi$$

Transformando as desigualdades em equações e resolvendo para $\omega_{i\beta}$ e $t_{i\beta}$ obtemos:

$$w_{i,\beta} = \frac{2(\Psi + k\varepsilon)}{A(n+1) - n} \tag{6.3}$$

$$t_{i,\beta} = (\psi + k\varepsilon)\frac{n - A(n-1)}{A(n+1) - n} \tag{6.4}$$

### 6.2.5 AS CONEXÕES INTERVALO-AND

A notação usada nesta seção é a mesma da usada na seção anterior. É importante lembrar que as conexões de entrada aos neurônios AND podem ter pesos negativos e, portanto, $m \geq 0$. Se a camada AND for modelar a conjunção original de intervalos na presença de conexões adicionais, então as seguintes condições devem ser satisfeitas:

- o i-ésimo neurônio AND deve se tornar ativo quando o exemplo for propagado através do respectivo ramo na árvore. Isso significa que suas entradas, decrementadas pelo limiar $t_{i,\beta}$ devem ser maior do que $\psi$ se todos os neurônios prévios, ao longo das conexões regulares positivas têm como saída o valor mínimo para o qual são considerados ativos ($n*A*w_{i\alpha}$), mesmo se todos os neurônios prévios ao longo das conexões negativas regulares têm como saída o valor máximo para os quais eles ainda são considerados inativos ($m*(1 - A)*(-w_{i\alpha})$), e mesmo se todas as conexões adicionais fornecem entradas negativas máximas ($-k*\varepsilon$)

- o i-ésimo neurônio AND deve estar inativo se pelo menos uma entrada regular positiva está inativa ($1*(1-A)*w_{i\alpha}$), mesmo se todas as outras entradas regulares positivas tenham valores máximos ($(n-1)*1*w_{i\alpha}$) e mesmo se todas as entradas regulares negativas sejam zero ($m*0*(-w_{i\alpha})$), e mesmo se todas as conexões adicionais forneçam entradas positivas máximas ($k*\varepsilon$)
- o i-ésimo neurônio AND deve estar inativo se pelo menos uma entrada regular negativa fornece o valor mínimo para o qual será considerado ativo ($1*A*(-w_{i\alpha})$), mesmo se todas as entradas positivas regulares estão maximamente ativas ($n*1*w_{i\alpha}$), e mesmo se todas as outras entradas regulares negativas forem zero ($(m-1)*0*(-w_{i\alpha})$) e mesmo se todas as conexões adicionais forneçam entradas positivas máximas ($k*\varepsilon$)

As três exigências anteriores são traduzidas nas desigualdades:

$$\begin{aligned} &(n*A*w_{i\alpha}) + m(1-A)(-w_{i,\alpha}) - k*\varepsilon - t_{i\alpha} \geq \psi \\ &1*(1-A)*w_{i\alpha} + (n-1)*1*w_{i\alpha} + m*0*(-w_{i\alpha}) + k*\varepsilon - t_{i\alpha} \leq -\psi \\ &n*1*w_{i\alpha} + 1*A*(-w_{i\alpha}) + (m-1)*0*(-w_{i\alpha}) + k*\varepsilon - t_{i\alpha} \leq -\psi \end{aligned} \tag{6.5}$$

Transformando as desigualdades em equações e resolvendo para $w_{i\alpha}$ e $t_{i\alpha}$ obtemos:

$$w_{i,\alpha} = \frac{2(\Psi + k\varepsilon)}{A(n+m+1)-(n+m)} \tag{6.6}$$

$$t_{i,\alpha} = (\psi + k\varepsilon)\frac{A(n+m-1)+(n-m)}{A(n+m+1)-(n+m)} \tag{6.7}$$

Se os pesos e valores de limiares de ativação forem estabelecidos com os valores dados pelas equações anteriores, a rede emula o comportamento da árvore original, mesmo na presença de conexões adicionais. Note que, no caso da função *step*, para k =

0 (ausência de conexões adicionais) e A = 1, a fórmula se degenera naquela mostrada na Figura 6.1 ($t_{AND} = \Psi(2N-1)$, $w_i = 2\Psi$).

A presença de conexões adicionais aumenta significativamente a dimensionalidade do espaço de busca, tornando-o mais plausível a conter a solução. Além disso, a inicialização de pesos e limiares com valores sugeridos pela árvore de decisão significa que o *Backpropagation* deve ser inicializado a partir de uma posição que está já bem perto do mínimo global ou de um bom mínimo local.

### 6.2.6 A FUZIFICAÇÃO INTERVALAR

O módulo de construção da árvore de decisão do NID3 considera testes de decisão na forma $x_i < t_j$, o que implica o uso de uma função de pertinência intervalar. Isto leva a um comportamento um pouco 'dogmático' da rede. Um valor no meio do intervalo e um valor perto de seus limites são tratados da mesma forma, o que limita a flexibilidade da rede. Em TBNN isto é resolvido usando um método simples de graduação de pertinência a intervalo. Para a fuzificação dos limites do intervalo, são impostas as seguintes restrições no valor resultante da função de pertinência $a_i$. O maior valor de $a_i$ está no meio do intervalo e decresce na direção dos extremos do intervalo. Quanto mais distante o valor estiver do meio do intervalo, menor o valor de $a_i$. TBNN primeiro determina a proximidade ao centro do intervalo:

$$v_i = \frac{R_i - 2\,|\,\mu_i - x_j\,|}{2R_i}$$

onde:

$\mu_i$: centro do i-ésimo intervalo

$R_i$: tamanho do intervalo

$x_j$: valor atual do atributo

A proximidade $v_i$ ao centro do intervalo é então sujeita à função sigmóide $a_i = /(1+e^{-hvi})$ dos neurônios da camada de intervalos. Este mecanismo torna possível, mesmo se o valor de um atributo estiver aquém do limite do intervalo, que o exemplo possa ter ainda uma 'segunda chance' na próxima camada de neurônios.

## 6.3 EXPERIMENTOS USANDO O TBNN

Essa seção diz respeito ao uso do TBNN em um problema específico, para explorar suas funcionalidades, bem como para fornecer subsídios para uma comparação entre os quatro métodos pesquisados.

### 6.3.1 EXEMPLO COMUM

Com o intuito de demonstrar a fuzificação intervalar (Seção 6.2.6), para a aplicação do TBNN no exemplo comum, será utilizado o mesmo conjunto de padrões gerado para a aplicação do método RULEX. Ou seja, um conjunto que não conterá somente valores discretos 0 e 1, mas valores reais quaisquer contidos no intervalo [0,1].

O conjunto de treinamento foi gerado da seguinte maneira: foram gerados 500 pares de valores aleatórios reais com duas casas decimais de precisão, no intervalo [0,1]. Os valores gerados foram então classificados, de acordo com a função apresentada na equação 6.8 (apresentada inicialmente na Seção 5.4.1):

$$
\begin{aligned}
z = f(x,y): \quad & x \in [0, 0.5] \ \wedge \ y \in [0, 0.5] : \quad 0 \\
& x \in [0, 0.5] \ \wedge \ y \in (0.5, 1] : \quad 1 \\
& x \in (0.5, 1] \ \wedge \ y \in [0, 0.5] : \quad 1 \\
& x \in (0.5, 1] \ \wedge \ y \in (0.5, 1] : \quad 0
\end{aligned}
\tag{6.8}
$$

A arquitetura da RN deverá ser definida pelo próprio TBNN durante a aplicação.

### 6.3.2 EXEMPLO ESPECÍFICO

O exemplo específico implementado para mostrar e avaliar o funcionamento do TBNN é um problema de classificação de veículos a partir de suas silhuetas [Siebert–1987]. Os dados desse problema foram originalmente coletados entre 1986 e 1987 por J. P. Siebert. Seu propósito original era encontrar um método de distinguir objetos tridimensionais em uma imagem bidimensional através da aplicação de um conjunto de características dos perfis das silhuetas bidimensionais dos objetos.

Medidas de características das formas extraídas de exemplos de silhuetas de objetos a serem discriminados foram usadas para gerar uma árvore de decisão. Essa estratégia foi usada com sucesso na discriminação entre silhuetas de modelos de carros, vans e ônibus, vistos de um único ângulo de elevação, mas de todos os ângulos de rotação (360

graus). A performance de classificação da árvore foi favoravelmente comparada aos classificadores estatísticos MDC (*Minimum Distance Classifier*) [Kim–1998] e k-NN (*k-Nearest Neighbour*) [Laaksonen & Oja–1996] em termos de taxa de erro e eficiência computacional.

Uma investigação da árvore de decisão gerada indicou que a estrutura da árvore é fortemente influenciada pela orientação dos objetos, e que imagens de objetos similares são agrupadas em uma única decisão.

Quatro modelos de veículos foram utilizados para o experimento: um ônibus de dois andares, uma van Chevrolet, e os carros Saab 9000 e Opel Manta 400. Essa combinação particular de veículos foi escolhida na expectativa de que o ônibus, a van, e qualquer dos carros fossem identificados sem dificuldades, mas que fosse mais difícil distinguir entre os dois carros.

As imagens foram obtidas de uma câmera posicionada a 34.2 graus a partir do chão, e os veículos foram rotacionados de modo que 0 e 180 graus correspondessem à frente e traseira do veículo, respectivamente, enquanto que 90 e 270 graus correspondessem à visão de perfil de lados opostos.

A Tabela 6.1 apresenta os 18 atributos utilizados para classificação das silhuetas.

Tabela 6.1 Atributos para classificação de silhuetas de veículos

| # | Atributo | Cálculo do atributo |
|---|---|---|
| 1 | Compacticidade | (perímetro médio)$^2$ / área |
| 2 | Circularidade | (raio médio)$^2$ / área |
| 3 | Distância de circularidade | área / (distância média da borda)$^2$ |
| 4 | Taxa de raio | (raio máximo – raio mínimo) / raio médio |
| 5 | Taxa dos eixos | (eixo menor) / (eixo maior) |
| 6 | Taxa do comprimento máximo | (comprim.perpendicular * comprim.máximo)/(comprim.máximo) |
| 7 | Taxa de espalhamento | (inércia sobre menor eixo)/(inércia sobre maior eixo) |
| 8 | Alongacidade | área / (largura)$^2$ |
| 9 | Retangularidade do eixo perpendicular | área / (comprimento do eixo * largura do eixo) |
| 10 | Comprim. Máx. de retangularidade | área / (comprim.máximo * comprim.perpendicular ao máximo) |
| 11 | Variância escalada sobre maior eixo | (momento de 2$^a$ ordem sobre menor eixo)/área |
| 12 | Variância escalada sobre menor | (momento de 2$^a$ ordem sobre maior eixo)/área |

| | | |
|---|---|---|
| | eixo | |
| 13 | Raio de giro escalado | (máx. variância + min.variância)/área |
| 14 | Grau de assimetria sobre maior eixo | (momento de 3ª ordem sobre maior eixo)/sigma_min$^3$ |
| 15 | Grau de assimetria sobre menor eixo | (momento de 3ª ordem sobre menor eixo)/sigma_max$^3$ |
| 16 | Grau de desnível sobre menor eixo | (momento de 4ª ordem sobre maior eixo)/sigma_min$^4$ |
| 17 | Grau de desnível sobre maior eixo | (momento de 4ª ordem sobre menor eixo)/sigma_max$^4$ |
| 18 | Taxa de visibilidade | (área visível)/(área do menor polígono que engloba a forma) |
| ( sigma_max$^2$ é a variância ao longo do maior eixo, sigma_min$^2$ é a variância ao longo do menor eixo ) | | |

A Tabela 6.2 apresenta as 4 classes para o problema. Para cada classe, há cerca de 240 instâncias de treinamento. A bibliografia encontrada não especifica o motivo pelo qual não foram utilizados todas as 360 possíveis instâncias para cada classe.

Tabela 6.2 Veículos a serem identificados através das características das silhuetas

| **Classe** | **Número de exemplos em cada classe (de um total de 946)** |
|---|---|
| Opel | 240 |
| Saab | 240 |
| Ônibus | 240 |
| Van | 226 |

Esse exemplo foi escolhido por dois motivos: primeiro porque apresenta todos os atributos de entrada numéricos, os quais podem ser usados para demonstrar a fuzificação intervalar (apresentada na Seção 6.2.6); segundo, porque esse exemplo foi mostrado ser mais passível de ser aprendido por uma árvore de decisão do que por outros métodos simbólicos, o que é interessante pois o primeiro passo do TBNN é justamente a geração de uma árvore de decisão a partir do conjunto de treinamento. A arquitetura inicial da RN é definida pelo TBNN como parte da aplicação do método.

## 6.4 RESULTADOS EXPERIMENTAIS USANDO O TBNN

Foram executados nove conjuntos de testes sobre o sistema TBNN. Quatro testes foram realizados utilizando o conjunto de treinamento para o exemplo comum (“OU-Exclusivo”) e cinco testes foram realizados utilizando o conjunto de treinamento do

exemplo específico para o TBNN ("Identificação de silhueta de veículos"). A seguir são apresentados a configuração e os resultados dos testes.

### 6.4.1 CONFIGURAÇÃO DOS TESTES

Todos os testes foram realizados com os conjuntos de treinamento descritos na Seção 6.3 deste trabalho. Todos os testes foram realizados com os mesmos valores de configuração do algoritmo de aprendizado *Backpropagation* com função *momentum*, definidos na Tabela 6.3. Todos os testes utilizaram RNs com pesos iniciais definidos com valores aleatórios variando entre –0.002 e 0.002, valor de limiar de ativação A = 0.7, e valor dos pesos adicionais $\varepsilon = 0.0001$. Em todos os testes realizados, a verificação do aprendizado neural é realizada usando 5-validação cruzada.

Tabela 6.3 Valores para o algoritmo de aprendizado

| Limite de épocas | Erro mínimo | Taxa de Aprendizado | Momentum |
|---|---|---|---|
| 50 000 | 0.2 | 0.2 | 0.1 |

### 6.4.2 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO COMUM

Para o exemplo comum, descrito na Seção 6.3.2, foram realizados dois conjuntos de testes agrupados da seguinte forma: no primeiro grupo, identificado como *Capacidade*, estão os testes que avaliam a eficácia da rede usada pelo método; no segundo grupo, identificado como *Qualidade*, estão os testes que avaliam a eficácia dos métodos via critérios da ADT.

Capacidade:

1) Aprendizado com fuzificação intervalar;
2) Aprendizado sem fuzificação intervalar;

Qualidade:

3) Qualidade das regras extraídas (Compreensibilidade e Consistência);
4) Qualidade das regras extraídas (Acurácia e Fidelidade).

Os dois primeiros conjuntos de testes têm por objetivo verificar a capacidade de aprendizado do sistema. Os dois últimos conjuntos de testes têm por objetivo verificar a qualidade das regras extraídas.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 6.4.2.1 e Qualidade – Subseção 6.4.2.2).

#### 6.4.2.1 TESTES PARA VERIFICAÇÃO DA CAPACIDADE DE APRENDIZADO

A RN utilizada pelo TBNN não apresenta restrições com intuito de facilitar a extração de regras, pois as regras são extraídas diretamente a partir conjunto de dados. A única diferença entre a arquitetura da RN definida pelo TBNN e a arquitetura do Perceptron Multicamadas é o uso de fuzificação intervalar. Nos testes apresentados a seguir é avaliado o impacto da fuzificação intervalar no aprendizado neural.

A Tabela 6.4 apresenta os resultados dos testes realizados. No caso do problema "OU-Exclusivo", os testes mostram que a fuzificação intervalar não melhora a capacidade de aprendizado da RN.

Tabela 6.4 Resultados de teste para "OU-Exclusivo" e RN com e sem fuzificação intervalar

| Fuzificação intervalar | Número médio de épocas até a convergência | Erro médio | Desvio Padrão do Erro |
|---|---|---|---|
| SIM | 36 589 | 14.375 | 4.19 |
| NÃO | 14 357 | 11.875 | 8.09 |

#### 6.4.2.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

A seguir são apresentados os resultados dos testes avaliando a Consistência, Acurácia, Fidelidade e Compreensibilidade das regras extraídas. As medidas utilizadas para avaliação dos critérios são apresentadas na Seção 3.6.2.2 e discutidas com mais detalhes no Capítulo 7.

- CONSISTÊNCIA DAS REGRAS EXTRAÍDAS: no caso do TBNN, no qual as regras não são extraídas da RN, não há como executar um teste de consistência nos moldes dos demais métodos. Porém, pode-se constatar que a arquitetura inicial de RN gerada não varia em diferentes execuções do TBNN usando um mesmo conjunto de dados.

- ACURÁCIA DAS REGRAS EXTRAÍDAS: a árvore de decisão classifica corretamente 65% dos padrões em um conjunto de padrões diferente do conjunto utilizado para treinamento.

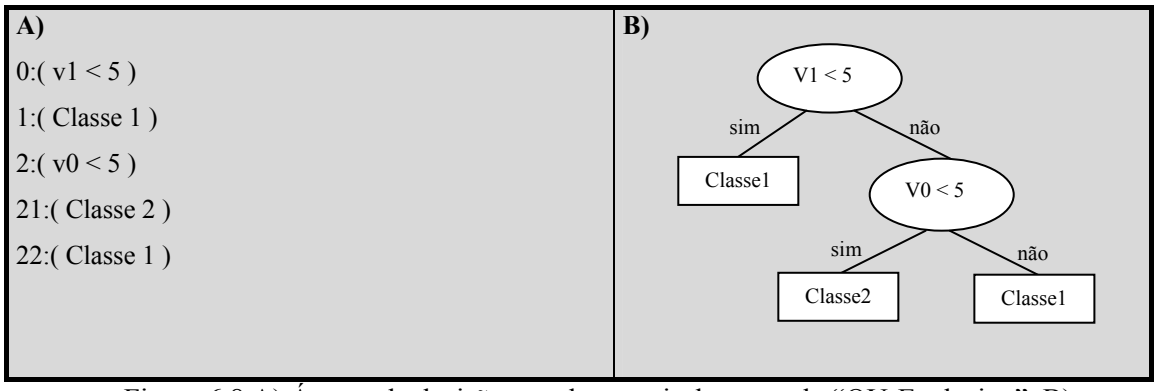



Figura 6.8 A) Árvore de decisão gerada a partir do exemplo "OU-Exclusivo". B) Representação gráfica da árvore

- FIDELIDADE DAS REGRAS EXTRAÍDAS: 67.5% dos padrões dos conjuntos de teste foram classificados da mesma maneira tanto pela regra quanto pela RN.

- COMPREENSIBILIDADE DAS REGRAS EXTRAÍDAS: as regras avaliadas são as regras geradas a partir da árvore de decisão. Em média foram geradas 3.6 regras com 5.8 atributos cada. A medida de compreensibilidade (Regras × Atributos) resultante é 20.88.

### 6.4.3 RESULTADOS EXPERIMENTAIS RELATIVOS AO EXEMPLO ESPECÍFICO

Para o exemplo específico, descrito na Seção 6.3.2, foram realizados cinco conjuntos de testes:

Capacidade:

1) Aprendizado com fuzificação intervalar e com simplificação das regras;
2) Aprendizado sem fuzificação intervalar e com simplificação das regras;
3) Aprendizado sem fuzificação intervalar e sem simplificação das regras;

Qualidade:

4) Qualidade das regras extraídas (Compreensibilidade) com/sem simplificação;

5) Qualidade das regras extraídas (Acurácia e Fidelidade) com/sem fuzificação intervalar;

Os três primeiros testes têm por objetivo avaliar a influência da fuzificação intervalar e da simplificação das regras sobre a capacidade de aprendizado da RN gerada. Os demais testes têm por objetivo avaliar tal influência na qualidade das regras, com relação às características "Compreensibilidade", "Acurácia" e "Fidelidade" da taxonomia ADT.

A seguir é apresentada uma descrição mais detalhada, os resultados e as conclusões de cada teste realizado (Capacidade – Subseção 6.4.3.1 e Qualidade – Subseção 6.4.3.2).

#### 6.4.3.1 TESTES PARA AVALIAÇÃO DA CAPACIDADE DE APRENDIZADO

Os testes aplicados avaliam a influência da fuzificação intervalar e da simplificação das regras sobre a capacidade de aprendizado da RN. Os resultados, apresentados na Tabela 6.5 mostram que, para o exemplo específico utilizado, a fuzificação intervalar influencia negativamente a capacidade de aprendizado da RN, e a simplificação das regras não apresenta influência significativa.

Tabela 6.5 Resultados relativos a "Identificação de silhueta de veículos"

| Fuzificação Intervalar | Simplificação das Regras | Média / DesvioPadrão de Épocas para convergência | Média / DesvioPadrão do Erro |
|---|---|---|---|
| Sim | Sim | 50 000 / 0 | 29.25% / 2.13 |
| Sim | Não | 50 000 / 0 | 29.43% / 5.74 |
| Não | Sim | 14 579 / 11942 | 14% / 0.61 |
| Não | Não | 10 031 / 7191 | 15.07% / 0.81 |

#### 6.4.3.2 TESTES PARA VERIFICAÇÃO DA QUALIDADE DAS REGRAS EXTRAÍDAS

As regras foram avaliadas com relação às características "Compreensibilidade", "Acurácia" e "Fidelidade" da ADT, frente à variação no uso de simplificação de regras e fuzificação intervalar. As medidas utilizadas são as mesmas introduzidas na Seção 3.6.2.2 deste trabalho e detalhadas no Capítulo 7.

O primeiro resultado a ser observado na Tabela 6.8 é que as regras geradas pelo TBNN apresentam baixa qualidade, notadamente com relação à Acurácia e Fidelidade. Enquanto a RN treinada apresenta valores de Acurácia em torno de 70% e 85%, as regras utilizadas para gerar as RNs apresentam valores em torno de 30%. O baixo valor de Fidelidade mostra ainda que a classificação das regras, além de serem incorretas com relação ao conceito a ser aprendido, são diferentes da classificação obtida pela RN para os mesmos padrões.

Tabela 6.8 Compreensibilidade, Acurácia e Fidelidade para o exemplo "Identificação de silhuetas de veículos"

| Fuzificação Intervalar | Simplificação das Regras | Média / DesvioPadrão do número de nós da árvore | Média / DesvioPadrão da Compreensibilidade | Média / DesvioPadrão da Acurácia | Média / DesvioPadrão da Fidelidade |
|---|---|---|---|---|---|
| Sim | Sim | 47 / 6.9 | 1360 / 496 | 28.72 / 0 | 26.59 / 8.12 |
| Sim | Não | 49.67 / 6.1 | 3791 / 863 | 30.14 / 2.46 | 30.26 / 5.98 |
| Não | Sim | 47 / 6.9 | 1360 / 496 | 28.72 / 0 | 28.72 / 3.25 |
| Não | Não | 49.67 / 6.1 | 3791 / 863 | 30.14 / 2.46 | 28.01 / 4.3 |

A fuzificação intervalar não influencia a qualidade das regras. Esse resultado se justifica pelo fato da fuzificação intervalar ser aplicada na construção da RN, que ocorre após a geração das regras.

A simplificação das regras não influencia de modo significativo a Acurácia e Fidelidade das regras, mas influencia fortemente a Compreensibilidade das regras. O valor de Compreensibilidade de conjuntos de regras sem simplificação é cerca de três vezes maior do que para conjuntos de regras simplificados.

## 6.5 CONCLUSÕES SOBRE O TBNN

Conforme apresentam os autores em [Ivanova & Kubat–1995], o TBNN é, grosso modo, o aperfeiçoamento de métodos já existentes de geração de arquitetura de RN ([Sethi–1990] [Brent–1991]) por meio da adição de dois novos recursos: a adição de conexões adicionais interligando completamente as camadas da RN, e a fuzificação intervalar, resultando em um método mais eficaz na determinação da arquitetura inicial de uma RN.

No início da pesquisa realizada para este trabalho, cogitou-se a possibilidade do TBNN ser um método de extração de regras no qual as regras extraídas seriam aquelas embutidas na árvore de decisão utilizada para definição da arquitetura inicial da RN. Porém, o baixo valor de Fidelidade obtido nos experimentos sobre o TBNN, sobretudo para o exemplo específico, mostra que o método não pode ser considerado um método de extração de regras. Apesar da árvore de decisão representar o comportamento inicial da RN, as alterações nos pesos neurais decorrentes do refinamento da RN durante o aprendizado podem alterar a resposta fornecida pela RN de modo a tornar a árvore de decisão pouco confiável como representante do comportamento da RN após o treinamento.

Ainda segundo os experimentos realizados, a elevada taxa de erro de classificação apresentada pela RN treinada, sobretudo para o exemplo comum, mostra que a definição da arquitetura inicial da RN por uma árvore de decisão não necessariamente produz RNs mais precisas. Especificamente no caso estudado, a árvore de decisão gerada não foi capaz de generalizar o conceito do conjunto de treinamento, dificultando o aprendizado neural subseqüente.

O uso de fuzificação intervalar resultou em redução da capacidade de aprendizado neural em todos os experimentos realizados. Aparentemente, isso é devido à necessidade de se estabelecer limites superior e inferior para os conjuntos *fuzzy*. Principalmente no exemplo específico implementado, os diferentes atributos apresentam-se distribuídos em faixas distintas de valores. Por exemplo, a "Taxa de visibilidade" apresenta valores variando de 1 a 4, enquanto que a "Variância escalada sobre o maior eixo" apresenta valores variando de 206 a 998. Desse modo, não é possível definir um único par de limites inferior e superior que abranja satisfatoriamente todos os atributos do problema. Uma possível solução seria estabelecer limites dinâmicos para cada atributo, porém essa solução não é apresentada na proposta original do método e foi deixada fora do escopo deste trabalho.

Os experimentos realizados permitem concluir que o TBNN não é interessante como um método de extração de regras de RNs. O uso do método para facilitar o aprendizado neural por meio da geração de uma arquitetura inicial também não apresentou resultados satisfatórios. Contudo, a aplicação de uma técnica de aprendizado neural (no caso, *Backpropagation*) mostrou-se eficaz como refinamento da classificação fornecida por uma árvore de decisão.

# CAPÍTULO 7. UMA PROPOSTA DE REFINAMENTO E EXTENSÃO DA ADT

As técnicas apresentadas em [Gallant–1988], [Setiono & Leow–2000], [Andrews & Geva–2002], [Ivanova & Kubat–1995] e investigadas neste trabalho são classificadas no mesmo grupo, de acordo com os critérios "Translucidez" e "Poder Expressivo das Regras Extraídas" da metodologia ADT. Apesar disso, como mostrado nos capítulos anteriores, tais técnicas têm comportamento e resultados substancialmente variados. Neste capítulo é apresentada uma proposta de alteração da ADT, com o objetivo de permitir que a taxonomia forneça uma descrição mais detalhada do método avaliado. A proposta de alteração inclui:

A) Criação de uma subdivisão do grupo "Regras Simbólicas Convencionais" para o critério "Poder Expressivo das Regras Extraídas";

B) Adição de grupos de classificação para os critérios "Portabilidade" e "Complexidade";

C) Estabelecimento de medidas de classificação para as características que compõem o critério "Qualidade das Regras Extraídas";

D) Criação do critério "Origem das Regras Extraídas";

A primeira alteração à ADT proposta neste trabalho tem por objetivo facilitar uma rápida consulta a algumas das características dos métodos, bem como promover uma comparação entre eles. A classificação dos métodos apresentada de forma textual em [Andrews & Tickle–1998] foi compilada em forma de tabela (Tabela 7.1).

Tabela 7.1 Métodos classificados em [Andrews & Tickle–1998]

| Método | Formato das regras | Qualidade das regras | Translucidez | Complexidade | Portabilidade |
|---|---|---|---|---|---|
| Algoritmo COMBO [Krishnan–1996] | Proposicional com | Regras extraídas mostram alta | Decomposicional | Exponencial no pior caso | Genericamente aplicáveis a redes |

| | | | | | |
|---|---|---|---|---|---|
| | Entrada Booleana | fidelidade, acurácia e compreensibilidade | | | retroalimentadas com entradas booleanas |
| Algorítmo RF5 (Rule Extraction From Facts) [ Saito & Nakano– 1996] | Leis científicas | Leis extraídas mostram alto grau de acurácia e fidelidade | Decomposi-cional | Linear no número de unidades intermediárias e de entrada | Não portável |
| RX [Setiono-1997] | Proposicio-nal | Bom | Decomposi-cional | Não disponível | Genericamente aplicáveis a redes retroalimentadas supervisionadas com entradas booleanas |
| Knowledge Extraction and Recurrent Neural Networks [Schellhammer et al.-1997] | MEF/Regras de transição de estados | Bom | Composicio-nal | Não disponível | Genericamente aplicáveis a redes recorrentes com entradas booleanas |
| Algoritmo de análise de intervalo [Filer et al.-1996] | Proposicio-nal com entrada contínua ou discreta | Conjuntos de regras extraídas mostram alto grau de acurácia, mas baixa compreensibilidade | Pedagógica | Não disponível | Portável |
| TREPAN [Craven & Shavlik-1996] | Árvore de decisão com testes de decisão por regra M-de-N nos nós | Árvores de decisão extraídas mostram alto grau de acurácia, fidelidade e compreensibilidade | Pedagógica | Polinomial no tamanho da amostra, dimensionalidade do espaço de entrada e máximo número de valores para cada característica discreta | Portável para qualquer modelo de treinamento/aprendizado |
| TopGen [Opitz & Shavlik-1996] | Proposicio-nal/regas M-de-N | Melhor que KBANN (‘‘conjuntos de regras mais esparsos’’) | Decomposi-cional | Determinada por múltiplos treinamentos de redes neurais | Arquitetura e algoritmo de treinamento especializado |
| Extraction of Finite-State Automata from Recurrent Networks [Giles & Omlin-1993] | MEF/Regras de transição de estado | MEFs extraídas exibem alto grau de acurácia e fidelidade | Composicio-nal | Não disponível | Portável para qualquer rede com restrições do espaço de estados e entradas externas |
| Cascade ARTMAP Integrating Neural Computation and Symbolic Knowledge Processing [Tan-1997] | Proposicio-nal com entrada discreta | Bom | Decomposi-cional. | Linear no número de entradas de categorias reconhecidas | Específico para redes ARTMAP (adaptive resonance theory mapping) |

Com exceção do critério "Translucidez", os valores descritos em [Andrews & Tickle–1998] estão praticamente na forma de comentários e, conseqüentemente, não apresentam padrão algum. Para os critérios "Formato das regras extraídas" e "Qualidade das regras extraídas" não são sequer seguidos os grupos sugeridos pela ADT. Desse modo, a simples coleção dos valores em uma tabela não é suficiente para permitir uma comparação direta entre os métodos.

As adições A), B) e C) feitas para refinar os critérios da ADT permitem que grupos padronizados sejam definidos e utilizados, possibilitando a construção de uma tabela comparativa simples e direta, com uma padronização de valores de características. Além disso, as adições permitem um maior detalhamento da classificação dos métodos estudados.

Esse maior detalhamento é fundamental uma vez que os métodos existentes e, particularmente aqueles investigados nesse trabalho, não podem ser diferenciados pelos critérios originalmente definidos na ADT. Como comentado anteriormente, usando a ADT como originalmente proposta, os métodos descritos nos capítulos anteriores são classificados no mesmo grupo, apesar de apresentarem resultados e comportamento substancialmente variados. Assim sendo, os detalhamentos propostos em A), B), C), e D) justificam-se e são detalhados a seguir.

**A) Subdivisão do grupo "Regras Simbólicas Convencionais" para o critério "Poder Expressivo das Regras Extraídas"**

Dos grupos sugeridos pela ADT para o critério "Poder Expressivo das Regras Extraídas", a maioria dos métodos existentes enquadra-se no grupo "Regras Simbólicas Convencionais". Dos métodos candidatos a investigação neste trabalho, apenas um se enquadra no grupo "Regras Baseadas em Conjuntos Fuzzy" e nenhum se enquadra no grupo "Regras Expressas em Lógica de Primeira Ordem". Segundo [Tickle et al.–1997], o terceiro grupo foi criado prevendo a criação de métodos de extração sobre um tipo de RN ainda não existente.

Assim sendo, dos grupos sugeridos para o critério "Poder Expressivo das Regras", na prática somente são utilizados os grupos "Regras Baseadas em Conjuntos Fuzzy", e "Regras Simbólicas Convencionais", este último incluindo praticamente todos os métodos existentes. Essa divisão fornece uma classificação que, presentemente, não é de grande interesse para uso em casos reais.

Neste trabalho, são utilizados os seguintes subgrupos para o grupo "Regras Simbólicas Convencionais":

a) Regras de Produção com Valores Booleanos Simples
b) Regras de Produção com Condições de Igualdade
c) Regras de Produção com Condições de Intervalo
d) Regras de Produção M-de-N
e) Árvores de Decisão

As regras de qualquer desses grupos podem ser facilmente traduzidas como "Regras de Produção com Valores Booleanos Simples". Os autores de cada método estudado, porém, apresentam as regras geradas por seu método em um subgrupo específico, seja por ser tal formato de regra a linguagem de representação mais adequada ao domínio do problema ao qual o método se destina, seja porque o autor considera tal formato de regra o mais fácil de extrair ou mais fácil de ser compreendido.

Uma consideração a ser feita com relação ao critério de "Poder Expressivo das Regras Extraídas" é que os grupos sugeridos em [Andrews et al.–1995] não têm sido utilizados em trabalhos posteriores [Andrews & Tickle–1998], sendo substituídos por uma descrição mais refinada do formato da regra semelhante à sugerida neste trabalho. Em outros trabalhos estudados ([Boz–1995] [Darbari–2000] [Setiono & Leow–2000] [Tickle et al.–1997]) pouca importância é dada a esse critério, talvez pelo fato de todos métodos estudados nesses trabalhos pertencerem ao mesmo grupo (regras simbólicas convencionais). Espera-se que, com as alterações sugeridas neste capítulo, esse critério possa ser refinado e, conseqüentemente, ter uma maior representatividade na caracterização das técnicas.

**B) Criação de grupos de classificação para os critérios "Portabilidade" e "Complexidade"**

Nenhum dos trabalhos estudados utiliza grupos para os critérios "Portabilidade" e "Complexidade". Por exemplo, em [Andrews & Tickle–1998], cada método tem uma descrição textual particular para esses critérios. Desse modo, entretanto, não há como agrupar os métodos a partir desses critérios de modo objetivo, pois o agrupamento depende de uma descrição em texto informal e, portanto, sujeito a interpretações subjetivas (conforme pode ser observado na Tabela 7.1).

Para o critério “Portabilidade”, são sugeridos os seguintes grupos:

- Dependente da RN (arquitetura, função de ativação, tipo dos pesos, etc)
- Dependente do algoritmo de treinamento (*Backpropagation*, *Pocket*, etc)
- Dependente do formato de entrada ou saída (valores booleanos, reais, etc)
- Dependente do paradigma de treinamento (supervisionado, etc)
- Dependente do domínio do problema
- Totalmente Portável

Esses grupos foram criados com base no mesmo estudo citado para a adição D., onde são identificados os elementos a partir dos quais os métodos extraem as regras.

O critério “Portabilidade” pode ser organizado de maneira aninhada, como mostra a Figura 7.1.

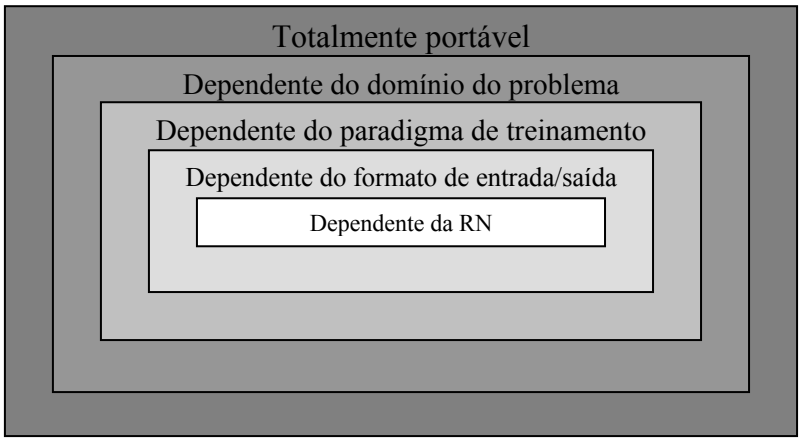



Figura 7.1 Organização aninhada do critério “Portabilidade”

A hierarquia mostrada na Figura 7.1 deve ser interpretada da seguinte maneira: Um método que pertence a um determinado grupo P é mais portável que um método que pertence a um grupo interno a P. Isso não significa, por exemplo, que um método do grupo “Dependente da RN” seja, necessariamente, dependente do formato de entrada ou saída. Mas como a dependência de algum elemento da RN é mais severa, em termos de portabilidade, que uma dependência do formato da entrada ou saída, essa segunda dependência pode ser ignorada. O que importa nessa classificação é definir qual o gargalo à portabilidade do método para outros sistemas.

Para o critério "Complexidade", é utilizada a notação "*O*" ou o texto equivalente, de acordo com a Tabela 7.2.

Tabela 7.2 Notação "*O"*

| Notação O | Complexidade | Exemplo |
|---|---|---|
| O(1) | tempo constante | Consulta a tabela Hash |
| O(log n) | (geralmente $\log_2 n$) tempo logarítmico | Busca binária |
| O(n) | tempo linear | Busca por comparação de cada elemento |
| $O(n^2)$ | tempo n quadrado | Ordenação de elementos (Bubblesort) |
| $O(n^m)$ | tempo polinomial | Multiplicação de matrizes cúbicas |
| $O(e^n)$ | tempo exponencial | Particionamento de conjuntos |

A notação "*O"* é utilizada por ser uma notação clássica adotada na análise de complexidade de algoritmos. A análise assintótica associada à notação "*O*" é utilizada por não se importar com detalhes do algoritmo, dando importância somente grau aproximado de aumento do tempo de processamento em relação ao tamanho da entrada. Neste trabalho, o critério "Complexidade do algoritmo de extração" é utilizado apenas para avaliar a viabilidade da implementação.

**C) Estabelecimento de medidas de classificação para as características que compõem o critério "Qualidade das Regras Extraídas"**

Em [Andrews & Tickle–1998] a avaliação do critério "Qualidade das Regras Extraídas" apresenta valores subjetivos, conforme apresentado na Tabela 7.1. Na Seção 3.6.2.2 deste trabalho são introduzidas medidas de classificação para as características "Acurácia", "Fidelidade", "Consistência" e "Compreensibilidade" do critério "Qualidade das Regras Extraídas", que permite classificar quantitativamente a qualidade das regras extraídas pelos métodos. As medidas sugeridas são:

CONSISTÊNCIA DAS REGRAS EXTRAÍDAS: A medida adotada para Consistência é o percentual de conjuntos de regras comuns obtidas a partir de diversas execuções do processo de extração de regras, sobre a mesma RN ou sobre RNs originadas de diferentes sessões de treinamento. RNs originadas de diferentes seções de treinamento fornecem classificações semelhantes, mas possuem os pesos das ligações entre os nós

potencialmente diferentes. Um método de extração de regras que apresenta 100% de Consistência sobre RNs originadas de diferentes sessões de treinamento provavelmente extrai o conceito sem se prender a detalhes da arquitetura interna da RN.

COMPREENSIBILIDADE DAS REGRAS EXTRAÍDAS: A medida adotada para Compreensibilidade é o produto do número de regras geradas pelo número médio de antecedentes por regra. Presume-se que quanto menor o valor de Compreensibilidade de um conjunto de regras, mais esse conjunto é passível de ser compreendido por seres humanos. A Compreensibilidade não pode adotar uma medida percentual como as demais características deste grupo da ADT pois seu valor não se restringe a um intervalo fechado.

ACURÁCIA DAS REGRAS EXTRAÍDAS: A medida adotada para Acurácia é a porcentagem de padrões corretamente classificados em um conjunto de padrões diferente daquele usado no treinamento da RN. Quanto maior o tamanho do conjunto utilizado para avaliação da Acurácia, mais preciso será o valor obtido. Porém, dependendo do conjunto de padrões utilizado, a Acurácia nunca será 100% devido à presença de dados incompletos ou inconsistentes. Na prática, um baixo valor de Acurácia pode significar que: a) as regras não capturaram o conceito embutido no conjunto de treinamento, seja por deficiência no aprendizado da RN, seja por perda de informação durante a tradução da informação da RN em regras ou por falta de capacidade de representação das regras; ou b) o conjunto de treinamento não contém dados suficientes para representar o conceito e, nesse caso, poderia ser interessante incluir o conjunto de padrões utilizado para verificação da Acurácia no conjunto de treinamento.

FIDELIDADE DAS REGRAS EXTRAÍDAS: A medida adotada para Fidelidade é a porcentagem de padrões classificados pelas regras da mesma maneira que a RN para um dado conjunto de padrões (como por exemplo, o conjunto de teste). Um conjunto de regras com 100% de Fidelidade classifica o dado conjunto de padrões exatamente da mesma maneira que a RN que originou tais regras.

**D) Criação do critério "Origem das Regras Extraídas"**

A última adição feita à ADT na pesquisa realizada até o momento é o acréscimo de um critério descritivo que permite agrupar os métodos de extração de regras de RNs a partir das bases de seu modo de funcionamento.

O critério "Origem das Regras Extraídas" é um critério descritivo porque, diferente de critérios como "Complexidade" e "Qualidade das Regras Extraídas", não fornece um critério de avaliação, mas sim uma descrição cujo objetivo principal é facilitar a caracterização do método.

Os diversos métodos de extração de regras de RNs estudados ([Andrews & Geva–2002] [Besada & Sanz–2002] [Craven & Shavlik–1996] [Fu–1994] [Gallant–1988] [Hatzilygeroudis & Prentzas–2001] [Setiono & Leow–2000] [Towell & Shavlik–1993] [Utgoff–1988], entre outros) extraem as regras das RNs de modos variados. Porém, independente do modo de operação, todos os métodos obrigatoriamente extraem as regras por meio de análises e operações sobre a RN ou sobre o conjunto de treinamento, uma vez que é na RN e nos dados que se encontra, mesmo que implicitamente, o conhecimento que se deseja obter.

Por exemplo, alguns métodos extraem as regras analisando e operando sobre os pesos das conexões entre as unidades da rede, outros se concentram na organização da RN, outros ainda, na função de ativação das unidades.

A utilização dessa característica dos métodos de extração de regras de RN para seu agrupamento permite uma caracterização semelhante à fornecida pelo critério "Translucidez", porém de modo mais detalhado e com objetivo diferente. Enquanto a "Translucidez" representa o grau de granularidade da visão do método sobre a RN como um valor para classificação, a "Origem das Regras Extraídas" é, sobretudo, uma representação do modo de funcionamento do método, descrevendo-o e auxiliando a interpretação dos demais critérios.

São propostos neste trabalho, para o critério "Origem das Regras Extraídas", que os métodos sejam agrupados dependendo do uso (ou não) das seguintes características para extração das regras:

- Função de ativação
- Pesos das conexões
- Organização das unidades
- Padrões de treinamento

Um método pode ser classificado por mais de um grupo, permitindo 15 conjuntos de grupos distintos, conforme ilustrado na Tabela 7.3. Os nomes sugeridos para os grupos têm intuito de facilitar a referência a grupos com mais de uma característica, mas não são utilizados na tabela comparativa apresentada no item E).

Tabela 7.3 Proposta de grupos para o critério "Origem das regras extraídas"

| **Nome do grupo** | **Características utilizadas** |
|---|---|
| FA | Função de ativação |
| PC | Pesos das conexões |
| OU | Organização das unidades |
| PT | Padrões de treinamento |
| FA-PC | Função de ativação, Pesos das conexões |
| FA-OU | Função de ativação, Organização das unidades |
| FA-PT | Função de ativação, Padrões de treinamento |
| PC-OU | Pesos das conexões, Organização das unidades |
| PC-PT | Pesos das conexões, Padrões de treinamento |
| OU-PT | Organização das unidades, Padrões de treinamento |
| FA-PC-OU | Função de ativação, Pesos das conexões , Organização das unidades |
| FA-PC-PT | Função de ativação, Pesos das conexões , Padrões de treinamento |
| FA-OU-PT | Função de ativação, Organização das unidades , Padrões de treinamento |
| PC-OU-PT | Pesos das conexões, Organização das unidades , Padrões de treinamento |
| FA-PC-OU-PT | Função de ativação, Pesos das conexões, Organização das unidades, Padrões de treinamento |

Algumas considerações podem ser feitas a partir dos grupos apresentados. Os conjuntos com mais elementos são os que, potencialmente, apresentam menor portabilidade. E um método que seja classificado apenas no grupo "Padrões de treinamento" provavelmente se enquadra no grupo "Eclético" do critério "Translucidez" e no grupo "Dependente de modelo de treinamento" do critério "Portabilidade".

As adições A), B), C) e D) permitem que a criação de uma tabela comparativa para diversos métodos de extração de regras de RNs, seja possível. Tal tabela não pode ser construída sem as adições feitas aos critérios porque, sem uma padronização, uma tabela que reúna diversos métodos é uma mera coleção de dados sem possibilidade de comparação objetiva.

**Construção de uma tabela com os métodos implementados neste trabalho**

A Tabela 7.4 apresenta a classificação dos quatro métodos estudados neste trabalho, segundo os critérios da taxonomia ADT com as extensões propostas.

A tabela deve ser lida da seguinte maneira: Uma célula na intersecção entre a coluna X e a linha Y é preenchida com um "×" quando o método de extração de regra da coluna X (cujo nome é listado na primeira cela da coluna) pertence ao grupo da linha Y (cujo nome é listado na cela mais à esquerda da linha). As células equivalentes a características quantitativas são preenchidas com o valor numérico resultante do teste aplicado para avaliação da característica ou com o texto "Não se aplica".

O uso de sombreamento em algumas células tem como único objetivo facilitar a leitura da tabela. As células que representam critérios ou características que possuem subdivisões não são preenchidas e servem apenas para agrupar visualmente os grupos que a compõem.

Os valores quantitativos utilizados no preenchimento da tabela são os resultados de teste apresentados nos capítulos que tratam da aplicação dos métodos no tratamento do problema "OU-Exclusivo". (Capítulos 3.6.2, 4.4.2, 5.5.2 e 6.4.2 deste trabalho).

Tabela 7.4 Tabela comparativa utilizando a taxonomia ADT com as extensões propostas

| **Método** / **Critério ADT** | MACIE | FERNN | RULEX | TBNN |
|---|---|---|---|---|
| **FORMATO DAS REGRAS** | | | | |
| **Simbólicas Convencionais** | | | | |
| Valores Booleanos Simples | × | × | | |
| Condições de Igualdade | | | | |
| Condições de Intervalo | | | × | |
| M-de-N | | × | | |
| Árvores de Decisão | | × | | × |
| **Fuzzy** | | | | |
| **Lógica de Segunda Ordem** | | | | |
| **QUALIDADE DAS REGRAS** | | | | |
| **Consistência** | 40% / 100% | 0% / 100% | 0% / 100% | Não Aplicável |
| **Compreensibilidade** | 2 | 4 | 1744 | 20.88 |
| **Acurácia** | 100% | 100% | 82.65% | 65% |
| **Fidelidade** | 100% | 100% | 82.65% | 67.5% |

| | | | | |
|---|---|---|---|---|
| **TRANSLUCIDEZ** | | | | |
| **Decomposicional** | × | × | | |
| **Pedagógico** | | | | × |
| **Eclético** | | | | |
| **Composicional** | | | × | |
| **COMPLEXIDADE**[19] | $O(m.n^3)$ | $O(m^2.n^2)$ | $O(n^2)$ | $O(n)$ |
| **PORTABILIDADE** | | | | |
| **Depende da RN** | × | × | × | |
| **Depende regime treinamento** | | | | |
| **Depende entrada/saída** | | | | |
| **Depende modelo treinamento** | | | | |
| **Depende domínio problema** | | | | × |
| **Totalmente Portável** | | | | |
| **ORIGEM DAS REGRAS EXTRAÍDAS** | | | | |
| **Função de Ativação** | | × | × | |
| **Pesos das Conexões** | × | | × | |
| **Organização das Unidades** | × | | | |
| **Padrões de treinamento** | × | × | | × |

A tabela comparativa utilizando as extensões propostas para a ADT permite que os métodos sejam comparados objetivamente. Por exemplo, com relação à característica "Compreensibilidade" do critério "Qualidade das regras extraídas", é fácil perceber que os métodos MACIE e FERNN apresentam conjuntos de regras bem menores que os demais métodos.

[19] n = número de agrupamentos no caso do RULEX; n = número de nós da árvore no caso do TBNN; m = número de padrões de treinamento e n = número de nós de uma camada da RN, no caso do MACIE e do FERNN.

# CAPÍTULO 8. CONCLUSÕES

Neste trabalho de pesquisa investigou-se o processo de 'tradução' de uma rede neural para uma representação simbólica, analisando várias propostas existentes na literatura. Quatro dessas propostas foram implementadas, analisadas e comparadas sob o enfoque das características elencadas pela taxonomia ADT.

A partir do estudo comparativo de cada uma das propostas implementadas, várias conclusões puderam ser obtidas sobre o processo de extração de regras e sobre as vantagens e desvantagens das diferentes abordagens.

O sistema MACIE, desenvolvido para ser utilizado como sistema especialista conexionista, apresenta um bom desempenho na extração de regras simbólicas para justificativa das inferências realizadas pela RN que compõe o engenho de inferência. Porém exige o uso de uma arquitetura de RN muito simples que só é capaz de aprender conceitos não linearmente separáveis se estiverem disponíveis informações sobre o comportamento dos dados de treinamento (matriz de dependência e resultados esperados para inferências internas da RN).

O método FERNN apresenta um algoritmo de poda bastante eficiente, que é capaz de identificar o conjunto mínimo de unidades da camada intermediária da RN necessário para a classificação dos padrões de treinamento, mas exige que os pesos das conexões sejam forçados a valores próximos de 0 ou 1, por meio de uma função de penalidade aplicada durante o treinamento da RN. Além disso, as regras extraídas pelo FERNN são totalmente dependentes do conjunto de treinamento utilizado.

O método RULEX é estritamente ligado a um modelo de RN particular, a rede LC, que apresenta algumas vantagens sobre as redes Perceptron Multi-Camadas tradicionais. As restrições impostas pelo RULEX sobre a rede LC não causam perda da capacidade de aprendizado e as regras são extraídas com baixo custo computacional. Porém, conforme observado nos experimentos realizados, as regras extraídas apresentam baixo grau de compreensibilidade.

O TBNN, inicialmente abordado neste trabalho como um método de extração de regras, mostrou-se inadequado para uso com objetivo de explicar o comportamento de

uma RN através de regras simbólicas, pois verificou-se que a informação contida na árvore de decisão utilizada na geração de uma RN pode ser alterada durante o aprendizado neural.

Além das conclusões sobre os diferentes métodos de extração de regras de RN, o trabalho de pesquisa realizado possibilitou a formação de uma proposta de extensão à taxonomia ADT, utilizada como ferramenta para classificação e comparação dos métodos implementados.

Foi evidenciado que a taxonomia ADT não apresenta recursos suficientes para comparação objetiva entre os métodos classificados. O método não sugere, por exemplo, unidades de medida nem grupos de classificação para os critérios de classificação. Adições foram propostas e utilizadas para a construção de uma tabela que permitisse comparar os quatro métodos implementados de modo objetivo, através da definição de conjuntos de grupos e medidas quantitativas para avaliação das características definidas pela taxonomia.

Finalmente conclui-se que, apesar do número crescente de trabalhos relacionados à proposta de extração de regras de RNs, os métodos existentes apresentam várias deficiências. A quase totalidade dos métodos estudados exige que a RN tenha determinadas características especificamente para permitir ou facilitar o trabalho de extração das regras e, na maioria dos casos, tais modificações reduzem a capacidade de aprendizado da RN. As regras extraídas apresentam, em maior ou menor grau, comportamento diferente da RN original, ou são aplicáveis somente a um conjunto restrito de padrões de treinamento. Enfim, não há como realizar a 'tradução' de uma RN em regras simbólicas sem que isso acarrete um custo que, na melhor das hipóteses, pode ser escolhido entre perda do poder expressivo da RN, baixa fidelidade das regras, restrição do domínio de aplicação ou outro ainda menos interessante.

A partir dessa conclusão, cabe observar que, na Tabela 7.4 do Capítulo 7, o critério "Portabilidade" é o mesmo ("Dependente da RN") para os quatro métodos implementados. O fato de RNs com características específicas serem necessárias para o funcionamento dos quatro métodos de extração de regras avaliados torna esse critério desnecessário para comparação entre tais métodos. Essa observação sugere que, conforme novos métodos de extração de regras de RN sejam criados, determinadas abordagens possam tornar-se comuns à maioria dos métodos, enquanto outras deixem de ser utilizadas, fazendo com que a avaliação de determinadas características dos

métodos torne-se desnecessária. Perante tal situação, uma taxonomia como a ADT precisaria de atualizações constantes, ou tornar-se-ia obsoleta com critérios inúteis para uma comparação objetiva entre os novos métodos de extração de regras de RNs.

É possível que a existência de uma taxonomia como a ADT mostre-se desnecessária caso a variedade de métodos seja reduzida a poucos métodos consagrados, ou caso a pesquisa chegue a resultados que provem a impossibilidade prática da tarefa de extração de regras de RN. Apesar disso, a atual existência de grande quantidade de métodos para extração de regras e a pesquisa crescente na área encorajam o desenvolvimento de ferramentas que permitam analisar, classificar e comparar tais métodos, na tentativa de tornar mais organizada e sistemática a busca por uma maneira de traduzir em regras o conhecimento implícito adquirido pelas RNs.

# APÊNDICE A

## A.1 PROVA DA PROPOSIÇÃO 1 DO FERNN (Seção 4.2.3, pg. 67):

Sejam $\psi = \sum_{k \in S'} w_{j,k} x_{k,p}$ e $\psi = \sum_{k \in S} w_{j,k} x_{k,p}$, considere os dois casos seguintes:

**Caso 1**: $x_p \in D_L$, ou seja, para essa amostra $H_{jp} = \sigma(w_j x_p) = \sigma(\psi + \psi) \leq L$. Pelo Teorema de Taylor:

$$\sigma(\psi) = \sigma(\psi + \psi) - \psi\sigma'(\xi)$$

onde $\xi$ é um ponto situado entre $\psi$ e $\psi + \psi$

A derivada da função sigmóide é sempre positiva e menor ou igual a 1/4. Existem dois casos possíveis:

1. $\psi \leq 0$: $\sigma(\Psi) \leq L - \frac{1}{4}\hat{\Psi} \leq L + \frac{1}{4}\sum_{k \in S} |w_{j,k}| < (L + U)/2$
2. $\psi > 0$: $\sigma(\Psi) \leq \sigma(\Psi + \hat{\Psi}) \leq L < (L + U)/2$

**Caso 2:** $x_p \in D_U$, ou seja, para essa amostra $H_{jp} = \sigma(w_j x_p) = \sigma(\psi + \psi) \geq U$. Existem também dois casos possíveis:

1. $\psi \leq 0$: $\sigma(\Psi) \geq \sigma(\Psi + \psi) \geq U > (L + U)/2$
2. $\psi > 0$: $\sigma(\Psi) \geq U - \frac{1}{4}\psi \geq U - \frac{1}{4}\sum_{k \in S} |w_{j,k}| > (L + U)/2$

Desse modo, pode-se concluir que após mudar a condição do nó da árvore para o valor da equação 4.6, as conexões de entrada para a unidade intermediária cujos pesos satisfaçam o critério da equação 4.5, podem ser removidas sem afetar a classificação dos conjuntos DL e DU.

## A.2 PROVA DA PROPOSIÇÃO 2 DO FERNN (Seção 4.2.3, pg. 67):

Sejam $\Psi = \sum_{k \in S'} w_{j,k} x_{k,p}$ e $\psi = \sum_{k \in S} w_{j,k} x_{k,p}$, considere os dois casos seguintes:

**Caso 1**: $x_p \in D_U$, ou seja, para essa amostra $H_{jp} = \sigma(w_j x_p) = \sigma(\psi + \psi') \geq U$. Existem também dois casos possíveis:

1. $\psi' \leq 0$: $\sigma(\Psi) \geq \sigma(\Psi + \psi) \geq U > (L + U)/2$
2. $\psi' > 0$: $\sigma(\Psi) \geq U - \frac{1}{4}\psi' \geq U - \frac{1}{4}\sum_{k \in S} |w_{j,k}| > (L + U)/2$

**Caso 2:** $x_p \in D_L$, ou seja, para essa amostra $H_{jp} = \sigma(w_j x_p) = \sigma(\psi + \psi') \leq L$. Pelo Teorema de Taylor:

$$\sigma(\psi) = \sigma(\psi + \psi') - \psi'\sigma'(\xi)$$

onde $\xi$ é um ponto situado entre $\psi$ e $\psi + \psi'$

A derivada da função sigmóide é sempre positiva e menor ou igual a 1/4. Existem dois casos possíveis:

1. $\psi' \leq 0$: $\sigma(\Psi) \leq L - \frac{1}{4}\psi' \leq L + \frac{1}{4}\sum_{k \in S} |w_{j,k}| < (L + U)/2$
2. $\psi' > 0$: $\sigma(\Psi) \leq \sigma(\Psi + \psi) \leq L < (L + U)/2$

Desse modo, pode-se concluir que após mudar a condição do nó da árvore para o valor da equação 4.8, as conexões de entrada para a unidade intermediária cujos pesos satisfaçam o critério da equação 4.7 podem ser removidas sem afetar a classificação dos conjuntos DL e DU.

## A.3 PROVA DA PROPOSIÇÃO 3 DO FERNN (seção 4.2.4, pg. 68):

1. Assumindo-se as três as suposições apresentadas (vide seção 4.1.4, proposição 4), se pelo menos $\lceil U \rceil$ das entradas $\{x_1, x_2, ..., x_K\}$ for igual a 1, então:

$$\sum_{i=1}^{K} w_i x_i = \sum_{i=1}^{K} (1 + f_i) x_i > \lceil U \rceil + F = U$$

2. Assumindo as três suposições apresentadas e tendo a condição da equação 4.9 satisfeita pelas variáveis $\hat{x}_1, \hat{x}_2, ..., \hat{x}_K$, podemos dizer que no máximo $\lfloor U \rfloor$ dessas $K$ entradas têm valor 1. Mas suponha que $\lfloor U \rfloor + \mu, (\mu \geq 1)$ entradas tenham valor 1. Nesse caso:

$$\sum_{i=1}^{K} w_i \hat{x}_i = \sum_{i=1}^{K} (1 + f_i) \hat{x}_i = \lfloor U \rfloor + \mu + \sum_{i=1}^{K} f_i \hat{x}_i \geq \lfloor U \rfloor + \mu > \lfloor U \rfloor + F = U$$

O que é uma contradição (enquanto a equação 4.9 diz que a soma ponderada é menor ou igual a U, a equação imediatamente anterior diz que a soma ponderada é maior que U).

## A.4 PROVA DA PROPOSIÇÃO 4 DO FERNN (seção 4.2.4, pg. 69):

1. Assumindo-se as três as suposições apresentadas (vide seção 4.1.4, proposição 3), se no máximo $\lfloor L \rfloor$ das entradas $\{x_1, x_2, ..., x_K\}$ for igual a 1, então:

$$\sum_{i=1}^{K} w_i x_i = \sum_{i=1}^{K} (1 + f_i) x_i > \lfloor L \rfloor - F = L$$

2. Assumindo as três suposições apresentadas, e tendo a condição da equação 4.10 satisfeita pelas variáveis $\hat{x}_1, \hat{x}_2, ..., \hat{x}_K$, podemos dizer que no mínimo $\lceil L \rceil$ dessas $K$ entradas têm valor 1. Mas suponha que $\lceil L \rceil - \mu, (\mu \geq 1)$ entradas tenham valor 1. Nesse caso:

$$\sum_{i=1}^{K} w_i \hat{x}_i = \sum_{i=1}^{K} (1 + f_i)\hat{x}_i = \lceil L \rceil - \mu + \sum_{i=1}^{K} f_i \hat{x}_i \leq \lceil L \rceil - \mu < \lceil L \rceil - F = L$$

O que é uma contradição (enquanto a equação 4.10 diz que a soma ponderada é maior que L, a equação imediatamente anterior diz que a soma ponderada é menor que L).

# APÊNDICE B

## CONSIDERAÇÕES SOBRE A IMPLEMENTAÇÃO DOS MÉTODOS

Todos os métodos foram implementados em C++, com uso do ambiente de desenvolvimento *Borland C++ Builder* ®. Cada método foi implementado como uma aplicação separada. Todos eles, entretanto, compartilham algumas características. Dentre as características comuns, estão a possibilidade de carregar e salvar arquivos texto contendo padrões de treinamento, definição de arquitetura e pesos de RN, configuração do algoritmo de extração de regras e regras geradas.

A seguir são apresentadas observações sobre a implementação de cada método, principalmente com relação a alterações feitas sobre a proposta original dos autores.

### B.1 SOBRE A IMPLEMENTAÇÃO DO MACIE

A implementação do MACIE utilizada neste trabalho apresenta algumas variações em relação ao sistema apresentado em [Gallant–1988].

A primeira variação é com relação ao universo de regras extraídas. Ao invés de extrair todas as regras possíveis no espaço do conjunto de entradas, o sistema implementado neste trabalho obtém regras apenas para os padrões de treinamento usados no treinamento da RN. Essa alteração foi feita para redução do espaço de busca do problema de extração de regras.

Este sistema implementa apenas o algoritmo de consulta (inferência neural) e justificativa das respostas, deixando de lado os mecanismos de justificativa das perguntas e consulta da entrada mais promissora, existentes no sistema proposto por Gallant. A implementação dessas funcionalidades foi deixada de lado por não serem necessárias à tarefa de extração de regras de RN.

### B.3 SOBRE A IMPLEMENTAÇÃO DO FERNN

A implementação do FERNN utilizada neste trabalho emprega, para o treinamento da RN, o algoritmo *Backpropagation* com função de erro de entropia-cruzada aumentada por meio de uma função de penalidade para pesos diferentes de zero (Seção 4.2.1). A substituição do algoritmo BFGS pelo *Backpropagation* foi feita com objetivo de

facilitar a comparação com o MACIE e TBNN, que utilizam o mesmo algoritmo de treinamento, e leva em consideração o fato de que o algoritmo de treinamento utilizado não interfere na tarefa de extração de regras, desde que seja aplicada uma função que encoraje os pesos das conexões da RN a convergirem para valores próximos de zero.

Não foi utilizada nesta implementação a conversão da árvore de decisão em regras M-de-N. Mesmo após a manipulação dos pesos neurais (Seção 4.2.4), as RNs obtidas nos experimentos realizados não apresentaram os requisitos necessários para conversão em regras M-de-N.

## B.3 IMPLEMENTAÇÃO DO RULEX

A implementação do RULEX desenvolvida neste trabalho foi feita para receber como entrada uma RN treinada pelo ELSY [Geva et al.–1995], um software que implementa os algoritmos de geração e treinamento de redes LC. A versão do ELSY utilizada neste trabalho foi fornecida por Robert Andrews.

O algoritmo de extração de regras inclui uma simplificação adicional não descrita em [Andrews & Geva–2002]. A simplificação consiste em remover conjuntos de regras idênticos gerados a partir de padrões de entrada distintos. A necessidade dessa simplificação ocorre devido à limitação da precisão das variáveis de ponto flutuante, que ocasionam geração de regras idênticas a partir de padrões de entrada muito semelhantes.

## B.4 IMPLEMENTAÇÃO DO TBNN

A implementação do TBNN desenvolvida neste trabalho segue a descrição do método apresentada em [Ivanova & Kubat–1995], exceto pelo algoritmo utilizado para construção da árvore de decisão. Em [Ivanova & Kubat–1995] é sugerido o uso do algoritmo NID3, porém na implementação realizada para este trabalho utilizou-se o algoritmo C4.5, que também foi utilizado na implementação do método FERNN.

O uso do mesmo algoritmo de construção de árvore de decisão para o FERNN e o TBNN teve por objetivo facilitar uma possível comparação entre os dois métodos.

# REFERÊNCIAS


**[Andrews et al.–1995]** Andrews, R.; Diederich, J.; Tickle, A. B., A survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge Based Systems, vol. 8, 1995, págs. 373-389. 15,18,20,23,94,134

**[Andrews & Geva–1996]** Andrews, R.; Geva, S., Rules and local function networks. Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, Universidade Sussex, Brighton, ReinoUnido, 1996, págs. 1-15. 85

**[Andrews & Geva–2002]** Andrews, S.; Geva, S., Rule extraction from local cluster neural nets. Neurocomputing, vol. 47, 2002, págs. 1-20. 16,24,83,89,94,95,97,130,138,150

**[Andrews & Tickle–1998]** Tickle, A.B.; Andrews R., The truth will come to light: directions and challenges in extracting the knowledge embedded within mined artificial neural networks. IEEE Transactions on Neural Networks, vol. 9, nro. 6, 1998, págs. 1057-1068. 20,24,25,131,132,134,136

**[Battiti–1992]** Battiti, R., First and second-order methods for learning: Between steepest descent and Newton's method. Neural Computation, vol. 4, 1992, págs. 141-166. 59,62

**[Besada & Sanz–2002]** Besada-Juez, J.M.; Sanz-Bobi, M.A., Extraction of fuzzy rules using sensibility analysis in a neural network. International Conference Artificial Neural Networks, Madrid, Espanha, 2002, págs. 395-400. 138

**[Blake & Merz–1998]** Blake, C.L.; Merz, C.J., UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: Universidade da California, Departmento de Informação e Ciência da Computação, 1998 (Download realizado em 10/2003). 70,72,100

**[Boz–1995]** Boz, O., Knowledge integration and rule extraction in neural networks. Relatório Técnico, Universidade Lehigh, PA, EUA, 1995, 18 págs. 134

**[Brent–1991]** Brent, R.P., Fast training algorithms for multilayer neural nets. IEEE Transactions on Neural Networks, vol. 2, 1991, págs. 346–354. 128

**[Browne & Sun–1999]** Browne, A.; Sun, R., Conectionist variable binding. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, 1999, págs.189-207. 15

**[Buchheit–1999]** Buchheit, P., A neuro-propositional model of language processing. International Journal of Intelligent Systems, 1999, págs. 585-601. 15

**[Castellano et al.–1997]** Castellano, G.; Faneli, A.M.; Pelilo M., An iteractive pruning algorithm for feedforward networks. IEEE Transactions on Neural Networks, vol. 8, nro. 3, 1997, págs. 519-531. 59

**[Craven & Shavlik–1994]** Craven, M. W.; Shavlik, J. W., Using sampling and queries to extract rules from trained neural networks. Machine Learning: Proceedings of the Eleventh International Conference, São Francisco, CA, EUA, 1994, págs. 37-45. 24

**[Craven & Shavlik–1996]** Craven, M.W.; Shavlik, J.W., Extracting tree-structured representations of trained networks. Advances in Neural Information Processing Systems, vol. 8, MIT Press, Cambridge, MA, EUA, 1996, págs. 24-30. 25,68,131,138

**[Darbari–2000]** Darbari, A., Rule extraction from trained ANN: A survey. Relatório Técnico, Grupo de Raciocínio e Representação do Conhecimento, Departmento de Ciência da Computação, Universidade de Tecnologia de Dresden, Alemanha, 2000, 61 págs. 25,134

**[Dasarathy–1980]** Dasarathy, B.V., Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, nro. 1, 1980, págs. 67-71. 100

**[Dennis & Schnabel–1983]** Dennis, J.E. Jr.; Schnabel R.B., Numerical methods for unconstrained optimization and nonlinear equations. Prentice Hall: Englewood Cliffs, NJ, EUA, 1983, 378 págs. 62

**[Duch et al.–2001]** Duch, W.; Adamczak, R.; Grabczewski K., A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks, vol. 12, 2001, págs. 277-306. 100

**[Fayyad & Irani–1992]** Fayyad, U.M.; Irani, K.B. On the handling of continuous-valued attributes in decision tree generation. Machine Learning, vol. 8, 1992, págs. 87-102. 111

**[Filer et al.–1996]** Filer, R.; Sethi, I.; Austin, J., A comparisson between two rule extraction methods for continuous input data. NIPS'97 Rule Extraction From Trained Artificial Neural Network Workshop, Queensland University of Technology, 1996, págs. 38-95. 131

**[Fu–1991]** Fu, L.M., Rule learning by searching on adapted nets. Proceedings of the 9th National Conference on Artificial Intelligence, EUA, 1991, págs. 590-595. 24,106

**[Fu–1994]** Fu, L.M., Rule generation from neural networks. IEEE Transactions on Systems, Man and Cybernetics, vol. 8, nro. 24, 1994, págs. 1114-1124. 24,138

**[Gallant–1988]** Gallant, S., Connectionist expert systems. Communications of the ACM, vol. 4, 1988, págs. 152-169. 16,25,26,31,56,130,138,149

**[Gallant–1993]** Gallant, S., Neural network learning and expert systems. MIT Press, Cambridge, MA, EUA, 1993, 365 págs. 28,30,41,56

**[Geva & Sitte–1993]** Geva, S.; Sitte, J., Local response neural networks and fuzzy logic for control. 2nd IEEE International Workshop on Emerging Technologies and Factory Automation (ETFA'93), Cairns, Australia, 1993, págs. 51-57. 83

**[Geva et al.–1995]** Geva, S.; Malmstrom, K.; Sitte, J., ELSY release 1.0 user manual. Queensland University of Technology, Australia, 1995, 26 págs. 102,150

**[Geva et al.–1998]** Geva, S.; Malmstrom, K.; Sitte, J., Local cluster neural net: Architeture, training and applications. Neurocomputing, vol. 20, 1998, págs. 35-56. 83,85,86,90,94,107

**[Giles & Omlin–1993]** Giles, C.L.; Omlin, C.W., Extraction, insertion, and refinement of simbolic rules in dinamically driven recurrent networks. Connection Sci., vol. 5, nro. 3, 1993, págs. 307-328. 132

**[Hagiwara–1994]** Hagiwara, M. A simple and effective method for removal of hidden units and weights. Neurocomputing, vol. 6, 1994, págs. 207-218. 59

**[Hatzilygeroudis & Prentzas–2001]** Hatzilygeroudis, I; Prentzas J. HYMES: A hybrid modular expert system with efficient inference and explanation. 8th Panhellenic Conference on Informatics, Nicosia, Cyprus, vol. 1, 2001, págs. 422-431. 25,26,138

**[Ivanova & Kubat–1995]** Ivanova, I.; Kubat, M., Initialization of neural networks by means of decision trees. Knowledge-Based Systems, vol. 8, nro. 6, 1995, págs. 333-344. 16,108,110,128,130,150

**[Kim–1998]** Kim, H. S., Recognition of Korean Address Strings by Tight-Coupling of Minimum Distance Classification and Dictionary–Based Post–Processing. International Journal of Computer Processing of Oriental Languages, vol. 12, nro. 2, 1998, págs. 207-221. 121

**[Krishnan–1996]** Krishnan, R., A systematic method for decompositional rule extraction from neural networks. NIPS*96 Rule Extraction from Trained Artificial Neural Network Workshop, Sowmass, CO, EUA, 1996, págs. 38-45. 106,131

**[Laaksonen & Oja–1996]** Laaksonen J.; Oja, E., Classification with learning k-nearest neighbors. The 1996 IEEE International Conference on Neural Networks, vol. 3, Nova York, NY, EUA, 1996, págs. 1480-1483. 121

**[Liu & Tan–1995]** Lui, H.; Tan, S. T., X2R: A fast rule generator. IEEE International Conference on Systems, Man and Cybernetics, Vancouver, Canadá, 1995, págs. 631-635. 71

**[Marcus et al.–1999]** Marcus, G. F.; Vijayan, S.; Rao, S. B.; Vishton, P., Rule learning in seven month old infants. Science, 1999, págs. 77-80. 15

**[Mitchell–1997]** Mitchell, T.M., Machine Learning. The McGraw-Hill Companies Inc. NY, 1997. 14

**[Neumann–1998]** Neumann, J., Classification and evaluation of algorithms for rule extraction from artificial neural networks. Relatório Técnico, ICCS, Division of Informatics, Universidade de Edinburgo, Escócia, 1998, 54 págs. 100

**[Nicoletti–1998/2000]** Nicoletti, M.C., Investigação de modelos cooperativos simbólico-conexionistas de aprendizado indutivo de máquina. Relatório de Pesquisa de Pós-doutorado, FAPESP, 1998-2000, 59 págs. 26,109

**[Ooyen & Nienhuis–1992]** van Ooyen, A.; Nienhuis, A., Improving the convergence of the backpropagation algorithm. Machine Learning, vol. 13, nro. 1, 1993, págs. 71-101. 62

**[Park & Sandberg–1991]** Park, J., Sandberg, I.W., Universal approximation using radial-basis-function networks, Neural Computation, 1991, págs. 246-257. 84

**[Quinlan–1986]** Quinlan, J.R., Induction of decision trees. Machine Learning, vol. 1, Kluwer Academic Publishers, Boston, MA, EUA, 1986, págs. 81-106. 63,110

**[Quinlan–1993]** Quinlan, J.R., C4.5 Programs. Machine Learning, 1993, 302 págs. 63

**[Rumelhart et al.–1986]** Rumelhart, D.E.; Hinton, G.E.; Williams, R.J., Learning internal representations by error propagation. Parallel distributed Processing: explorations in the microstructure of cognition. vol. 1, MIT Press, Cambridge, MA, EUA, págs. 318-363. 31

**[Saito & Nakano–1996]** Saito K.; Nakano, R., Law discovery using neural networks. NIPS'96 Rule Extraction From Trained Artificial Neural Networks Workshop, Queensland University of Technology, 1996, págs. 62-69. 131

**[Schellhammer et al.–1997]** Schellhammer, I.; Diederich, J.; Towsey, M.; Brugman, C., Knowledge extraction and recurrent neural networks: An analysis of an Elman network trained on a natural language learning task. Relatório Técnico, QUT-NRC 97-IS1, 1997. 131

**[Sethi–1990]** Sethi, I.K., Entropy nets: From decision trees to neural networks. Proceedings of the IEEE, vol. 78, 1990, págs. 1605-1613. 128

**[Setiono–1997]** Setiono, R., Extracting rules from neural networks by pruning and hidden-unit splitting. Neural Computation, vol. 9, no. 1, 1997, págs. 205-225. 59,62,131

**[Setiono–2000]** Setiono, R., Extracting *M* of *N* rules from trained neural networks. Transactions on Neural Networks, vol. 11, 2000, 23 págs. 68

**[Setiono & Leow–2000]** Setiono, R.; Leow W. K., FERNN: An algorithm for fast extraction of rules from neural networks. Applied Intelligence, vol. 12, 2000, págs. 15-25. 16,25,58-61,70,74,79,81,106,130,134,138

**[Siebert–1987]** – Siebert, J. P., Vehicle recognition using rule based methods. Turing Institute Research Memorandum TIRM-87-018, 1987. 121

**[Silva & Almeida–1990]** Silva, F.M.; Almeida, L.B., Speeding up backpropagation. Advanced Neurocomputers, Ed. R.Eckmiller, Amsterdã, Holanda, 1990, págs. 151-158. 91

**[Tan–1997]** Tan, A.W., Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. IEEE Transactions on Neural Networks, vol.8, nro. 2, 1997, págs. 237-250. 132

**[Tickle at al.–1996]** Tickle, A.B.; Orlowski, M.; Diederich, J., DEDEC: a methodology for extracting rules from trained artificial neural networks. Rules and Networks, QUT, Neurocomputing Research Centre, Queensland, Australia, 1996, págs. 96-102. 106

**[Tickle et al.–1997]** Tickle A. B.; Golea M.; Hayward R.; Diederich J., The truth is in there: current issues in extracting rules from trained feedforward artificial neural networks. Proceedings of the 1997 IEEE International Conference on Neural Networks, IEEE Press, Houston, TX, EUA, 1997, págs. 2530-2534. 24-5,133-4

**[Thrun–1995]** Thurn, S.B., Extracting rules from artificial neural networks with distributed representations. Advances in Neural Information Processing Systems, vol. 7, G. Tesauro, D. Touretzky and T. Leen (eds.), 1995, págs. 505-512. 24

**[Towell & Shavlik–1993]** Towell, G.; Shavlik, J., Extracting refined rules from knowledge-based neural networks. Machine Learning, vol. 13, 1993, págs. 71-101. 21-2,68,108,138

**[Towell et al.–1991]** Towell, G.; Craven, M.; Shavlik, J., Constructive induction in knowledge-based neural networks. Machine Learning: Proceedings of the Eighth International Workshop, Birnbaum, L. and Collins, G., Morgan Kaufmann (eds.), San Mateo, CA, EUA, 1991, págs. 213-217. 73

**[Tschichold-Gurman–1997]** Tschichold-Gurman, N., The neural network model rulenet and its application to mobile robot navigation. Fuzzy Sets and Systems, vol. 85, 1997, págs. 287-303. 17

**[Utgoff–1988]** Utgoff, P.E., Perceptron tree: A case study in hybrid concept representation. Proceedings of the 7th Nat. Conf. Artificial Intelligence, 1988, págs. 601–605. 138

**[Watson et al.–1987]** Watson, J. D.; Hopkins, N. H.; Roberts, J.W.; Steitz,J. A.; Weiner, A. M., Molecular Biology of the Gene. Weiner Benjamin/Cummings (eds.), 1987, págs. 634–647. 73