Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Sobre o Modelo Neural RuleNet e suas Características Simbólica e Cooperativa

Lucas Baggio Figueira

Orientação
Maria do Carmo Nicoletti

São Carlos
Abril/2004

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Sobre o Modelo Neural RuleNet e suas Características Simbólica e Cooperativa

Lucas Baggio Figueira

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

São Carlos
Abril/2004

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

F475sm

Figueira, Lucas Baggio.
Sobre o modelo neural RuleNet e suas características simbólica e cooperativa / Lucas Baggio Figueira. -- São Carlos : UFSCar, 2005.
152 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2004.

1. Inteligência artificial. 2. Aprendizado do computador. 3. Redes neurais (computação). 4. Aprendizado simbólico. 5. Aprendizado cooperativo. I. Título.

CDD: 006.3 (20ª)

Aos meus pais
Washington e Conceição

Ao meu irmão
Rodrigo

A minha amada
Júnia

Agradecimentos

A Deus por me mostrar o Caminho, a Verdade e a Vida.

A Profª Maria do Carmo Nicoletti pela amizade, dedicação, incentivo e orientação na elaboração dessa dissertação, principalmente pelos preciosos ensinamentos transmitidos no decorrer desses dois anos de trabalho.

Aos meus pais, Washington e Conceição, por me ensinarem o caminho da verdade, pelo apoio moral e financeiro, pelo carinho, pela compreensão e pelo amor incondicional.

Ao meu irmão Rodrigo por ser um grande amigo e pelas inúmeras conversas durante as madrugadas.

A Júnia Carolina por me amar, me apoiar e me inspirar ao longo desses sete anos de namoro.

Ao Prof. Antonio Carlos Roque por ter me orientado quando eu ainda estava na graduação e pela colaboração dada a este trabalho de mestrado.

Aos meus amigos de república Matheus, Rodrigo e Thiago pela amizade e pelas horas de descontração.

Ao meu amigo Jean pelos trabalhos realizados em conjunto desde a graduação.

A todos os demais colegas e amigos da pós-graduação.

Aos professores e funcionários do Programa de Pós-Graduação do Departamento de Computação.

A Capes pelo apoio financeiro.

"O coração do homem pode fazer planos,
mas a resposta certa vem dos lábios do Senhor"
Provérbios 16:1

Resumo

Aprendizado de máquina é uma área da Inteligência Artificial que investe na pesquisa de métodos e técnicas para viabilizar o aprendizado automático em sistemas computacionais. Este trabalho de pesquisa investiga um modelo neural de aprendizado de máquina chamado RuleNet e sua extensão Fuzzy RuleNet, para domínios *fuzzy*. Dentre as vantagens da proposta RuleNet estão sua simplicidade, facilidade e rapidez no treinamento bem como a maneira como representa o conceito induzido, que pode ser caracterizada como simbólica. Esse aspecto torna o RuleNet adequado a ser incorporado a sistemas cooperativos de aprendizado. O trabalho de pesquisa investiga a contribuição do modelo RuleNet tanto como uma técnica de aprendizado *stand-alone* quanto como parte de um sistema cooperativo. O trabalho apresenta e discute os resultados obtidos em vários experimentos que avaliam o RuleNet como método de aprendizado *stand-alone* (versus dois outros métodos de aprendizado de máquina, o ID3 e o NGE) e como parte de um sistema cooperativo, articulado tanto ao ID3 quanto ao NGE.

Abstract

Machine learning is an area of Artificial Intelligence that deals with methods and techniques for implementing automatic learning in computational systems. This research work investigates a machine learning neural model called RuleNet and its extension for fuzzy domains named Fuzzy RuleNet. Among the advantages of the RuleNet proposal are its simplicity, easiness and fast training as well as the way it represents the induced concept, which can be characterized as symbolic. This aspect makes RuleNet suitable for participating in cooperative systems. This research work investigates both the contribution of the RuleNet model as a stand alone learning technique as well as part of a cooperative system. It presents and discusses the results obtained in several experiments, evaluating RuleNet as a stand alone machine learning (versus two other machine learning methods, the ID3 and the NGE) and as part of a cooperative system, articulated to ID3 and to NGE.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação e Justificativa

A área de pesquisa de Aprendizado de Máquina (AM) oferece uma grande variedade de modelos de aprendizado, algoritmos, resultados teóricos e aplicações. Ultimamente tem havido um avanço considerável nesta área evidenciado principalmente pelas inúmeras aplicações que fazem uso de técnicas de AM: programas que aprendem perfil de compradores potenciais, que aprendem a detectar fraudes em transações bancárias, sistemas de filtragem que aprendem as preferências dos usuários de vários sistemas, veículos autônomos que aprendem a se locomover em vias públicas, sistema de apoio ao diagnóstico médico etc.

As limitações de cada um dos vários modelos de AM e das técnicas disponibilizadas em cada um deles estão cada vez mais evidentes. Pode se verificar na literatura que a pesquisa em AM se direciona na busca de refinamentos de técnicas já consagradas e estabilizadas, com vistas a contornar e/ou eliminar algumas de suas fragilidades.

Apesar desse investimento, é fato que por mais refinada e ajustada que uma determinada técnica de AM seja, ela é, entre outros, fortemente determinada pelo domínio de conhecimento em que atua, pelos diferentes tipos de informação de que faz uso e pelas linguagens de representação de

conhecimento que emprega. Sob certos aspectos, tais fatores controlam e confinam a atuação da técnica não apenas a determinados domínios, mas também ao aprendizado de apenas determinadas classes de conceitos [Nicoletti 1998/2000].

Com o objetivo de ampliar o escopo de atuação de sistemas de aprendizado automático, é intuitivo e natural cogitar sistemas que buscam a promoção da diversidade de técnicas disponibilizando maneiras que permitam uma cooperação entre elas ou, então, viabilizando a integração de algumas delas.

A principal justificativa deste trabalho é a da investigação de um modelo cooperativo de aprendizado conhecido como RuleNet [Tschichold-Gurman 1995b], [Tschichold-Gurman 1997], tanto como método de aprendizado *stand alone* (eficiência de seu algoritmo de propagação, tempo de treinamento, adaptabilidade a diferentes domínios), quanto como parte de um sistema colaborativo, articulado a outro método de aprendizado.

O trabalho investiga também o aspecto dinâmico do modelo, com relação à construção da topologia da rede, durante o treinamento e enfatiza seu caráter fortemente simbólico. Além disso, analisa a generalização do modelo para domínios *fuzzy*.

1.2 Principais Objetivos

A pesquisa realizada focalizou:

- a investigação do sistema RuleNet;
- o desenvolvimento de dois sistemas computacionais: um que implementa o RuleNet e outro que implementa o mesmo sistema, para domínios *fuzzy*;
- a avaliação empírica do sistema construído, usando dados do Repositório da UCI [Blake e Merz 1998];
- a avaliação da contribuição da versão *fuzzy* do RuleNet, conhecida como Fuzzy RuleNet, e suas principais características. Experimentação do Fuzzy RuleNet versus o sistema Fuzzy NGE [Santos 1997];
- a implementação de dois métodos puramente simbólicos,

especificamente, o sistema NGE [Salzberg 1989], e o algoritmo ID3 [Quinlan 1986]. Experimentação do sistema RuleNet versus o NGE e RuleNet versus ID3. A escolha do NGE se deve ao fato que tanto o RuleNet quanto o NGE modelam o conhecimento aprendido usando uma representação de conhecimento baseada em hiper-retângulos. A escolha do ID3 está ligada ao fato deste algoritmo ser um dos mais utilizados e investigados no aprendizado simbólico e da relativa facilidade da conversão de regras em hiper-retângulos;

- a implementação de duas propostas de cooperação: a referenciada como cooperação NGE→RuleNet e a referenciada como ID3→RuleNet, bem como a avaliação dessas cooperações em vários domínios de conhecimento.

1.3 Organização do Trabalho

Esta dissertação está organizada da seguinte maneira:

- o Capítulo 2 aborda a área de pesquisa onde este trabalho se insere, apresentando e discutindo os principais conceitos e idéias relativas à área de Aprendizado de Máquina. São apresentadas características gerais de dois tipos de aprendizado que a pesquisa focaliza, a saber: o neural e o simbólico. Além disso, é feita uma revisão bibliográfica de dois sistemas cooperativos de aprendizado, os quais possuem abordagens bastante distintas da tratada neste trabalho;
- no Capítulo 3 são apresentadas e discutidas as principais idéias e conceitos relativos ao algoritmo ID3 e ao sistema NGE, os quais são relevantes à pesquisa conduzida;
- o Capítulo 4 descreve o sistema de aprendizado conhecido como RuleNet, apresentando seus principais aspectos em detalhes, além da caracterização do sistema que estende o RuleNet para domínios *fuzzy*;
- o Capítulo 5 apresenta os experimentos e discute os resultados

obtidos pelo RuleNet, pelo Fuzzy RuleNet e pelas colaborações NGE→RuleNet e ID3→RuleNet. Além disso, é apresentada uma análise comparativa dos resultados obtidos pelo RuleNet versus NGE e ID3, assim como dos resultados obtidos pelo Fuzzy RuleNet versus Fuzzy NGE;

- o Capítulo 6 apresenta as conclusões e futuros desdobramentos deste trabalho;
- o Apêndice A descreve o pseudocódigo detalhado do RuleNet;
- o Anexo A descreve o sistema Fuzzy NGE utilizado em experimentos descritos no Capítulo 5.

Capítulo 2

Considerações Sobre Aprendizado de Máquina e Modelos Cooperativos de Aprendizado

2.1 Introdução

O processo de aprendizado é fundamental para qualquer sistema inteligente, seja ele humano, animal ou computacional. Sem a incorporação de um processo de aprendizado esses sistemas não possuem condições para adquirir novos conhecimentos, reciclar o conhecimento adquirido ou, então, se adaptarem a eventuais mudanças das condições do ambiente em que atuam.

O aprendizado envolve a aquisição de conhecimento, desenvolvimento de habilidades sensoriais e motoras, organização, refinamento e adaptação do conhecimento adquirido, aperfeiçoamento das técnicas de aquisição e manipulação do conhecimento, de forma que o mesmo se torne mais rápido e preciso.

Há várias definições de aprendizado disponíveis na literatura. Simon em [Simon 1983] propõe que o aprendizado seja abordado como um processo por meio do qual um sistema melhora o seu desempenho. Esta definição assume que o sistema em questão está executando uma tarefa. Portanto, aprendizado, i.e., a melhoria de desempenho, está inerentemente ligado e confinado à

realização de uma determinada tarefa e pode acontecer por meio da aplicação de novos métodos e conhecimentos ou, então, por meio de melhoramento/refinamento de métodos e conhecimentos existentes, de maneira a torná-los mais rápidos e precisos.

Uma abordagem mais ampla de aprendizado é dada por Carbonell em [Carbonell 1989], que afirma que:

> *"Aprendizado pode ser definido operacionalmente como o processo de execução de tarefas que não eram executadas anteriormente, e a melhoria do desempenho na execução das tarefas já conhecidas".*

Quando o aprendizado é focalizado como um processo computacional geralmente é chamado de Aprendizado de Máquina (AM) e pode ser definido de inúmeras maneiras. Segue aquela apresentada em [Mitchell 1998]:

> *"Diz-se que um programa de computador aprende a partir da experiência E com relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho na realização das tarefas T, medido por P, melhora com a experiência E".*

Logo, como sugerido em [Briscoe e Caelli 1996], aprendizado de máquina pode ser abordado como um processo que pode realizar:

- a aquisição de novos conhecimentos a partir da experiência
- o aperfeiçoamento da representação do conhecimento já existente, de forma que o mesmo possa ser utilizado de forma mais eficiente.

De qualquer forma, quando aprendizado é abordado como um processo a ser automatizado e articulado a sistemas computacionais existentes, muitas das pesquisas na área focalizam as diferentes possibilidades de como definir, representar e executar esse processo. Pode se dizer que as pesquisas em AM se direcionam ao estudo dos diferentes mecanismos que habilitam máquinas com a capacidade de aprendizado, o que inclui, como identificado em [Nicoletti 1994]:

- exploração das formas alternativas do uso de indução,
- investigação da influência, no aprendizado, da representação do conhecimento adotada,
- determinação do alcance e limitação dos algoritmos existentes,

- experimentação com o volume e nível de abstração da informação inicialmente disponível ao sistema de aprendizado,
- tratamento dado a informações incompletas ou incertas,
- desenvolvimento de técnicas gerais aplicáveis a diferentes domínios de conhecimento,
- formas de viabilizar o aprendizado incremental e multiconceitual,
- métodos automáticos e métodos direcionados pelo usuário,
- articulação de diferentes estratégias de aprendizado e
- utilização da Teoria do Domínio em processo de aprendizado.

Via de regra, o que é disponibilizado a um sistema de aprendizado de máquina para a realização do aprendizado é o chamado conjunto de treinamento, ou conjunto de exemplos que, geralmente, é um conjunto de vetores de pares atributo-valor_de_atributo e um conceito (classe) associado. Cada um desses vetores é conhecido como instância (ou exemplo) de treinamento. Alguns métodos de aprendizado, contudo, além de instâncias de treinamento, requerem que o domínio de conhecimento (i.e. a área específica de conhecimento, na qual o aprendizado vai ocorrer) seja axiomatizado, representado formalmente e, que tal descrição formal seja também disponibilizada para o sistema de aprendizado. Essa representação formal de fatos e regras que representam o domínio do conhecimento é conhecida como Teoria do Domínio.

A área de pesquisa de AM oferece uma grande variedade de modelos e estratégias de aprendizado, algoritmos, resultados teóricos e aplicações. Uma das estratégias de AM mais investigadas e difundidas é o aprendizado indutivo, o qual é caracterizado como um processo que realiza generalizações a partir de novos fatos observados num determinado domínio de conhecimento.

Em outras palavras, o conjunto de treinamento é fornecido pelo instrutor ou pelo ambiente e pode ser abordado como um conjunto composto de exemplos positivos e exemplos negativos (contra-exemplos) do conceito[1] em

[1] O termo conceito indica uma classe de equivalência de entidades, que pode ser descrita por meio de um conjunto limitado de expressões e que deve ser suficiente para a diferenciação entre conceitos [Nicoletti 1994].

questão. A indução de um conceito corresponde a uma busca em um espaço de hipóteses (possíveis candidatas à representação do conceito), de forma a encontrar aquelas que melhor representam os exemplos. Neste contexto, "melhor" pode ser definido em termos de precisão e compreensibilidade. A tarefa de aprendizado consiste na construção da descrição do conceito (hipótese), e é denominado aprendizado de conceito[2].

A Figura 2.1 mostra um diagrama geral de um sistema de aprendizado indutivo, o qual generaliza o conjunto de treinamento em hipótese(s) (e, dependendo do algoritmo, a generalização é feita com a ajuda da Teoria do Domínio), onde o conjunto de treinamento é representado por instâncias do conceito, notadas por '+' (exemplos positivos) e por instâncias que não fazem parte do conceito em questão, notadas por '-' (exemplos negativos). Além disso, a Teoria do Domínio pode, ou não, ser utilizada pelo algoritmo de aprendizado indutivo, o qual apresenta como saída hipótese(s) que descreve(m) o conceito em questão.

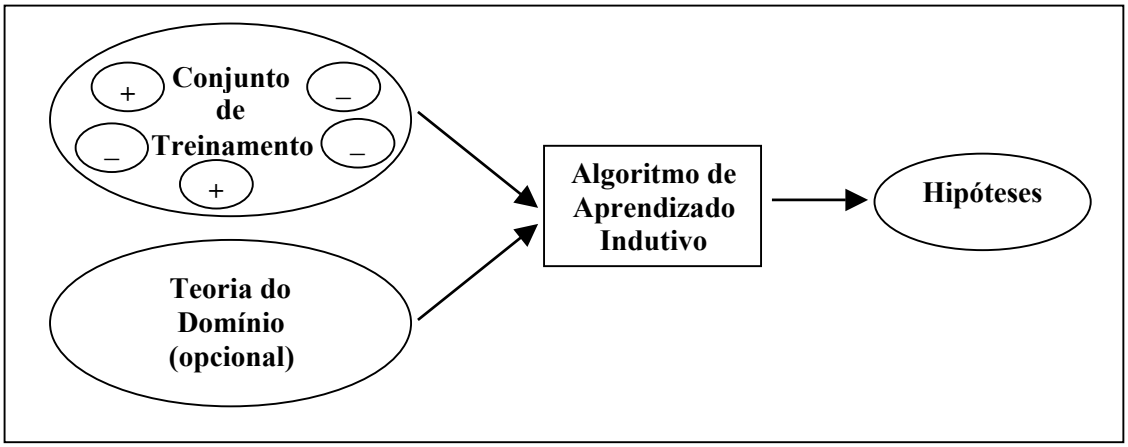


Figura 2.1: Diagrama geral de um sistema de aprendizado indutivo.

Dentre os muitos aspectos que caracterizam o aprendizado indutivo de máquina tem-se:

- **Aprendizado Supervisionado:** cada exemplo de treinamento E está associado a sua respectiva classe, dessa forma o sistema de aprendizado pode comparar a classe do exemplo E em questão com a saída da

[2] Aprender um conceito C significa adquirir habilidade de reconhecer se uma determinada instância x pertence a tal conceito.

classificação (i.e. a classe atribuída ao exemplo E pelo sistema de aprendizado) e, a partir daí corrigir eventuais erros de classificação.

- **Aprendizado Não Supervisionado:** os exemplos de treinamento não possuem classes associadas que, em geral, são desconhecidas; neste caso, o algoritmo divide o conjunto de exemplos *clusters* (i.e. agrupamentos) de exemplos similares (de acordo com alguma métrica como por exemplo, distância euclidiana) onde cada *cluster* de exemplos representa uma possível classe do domínio em questão.
- **Aprendizado Incremental (*On Line*):** neste tipo aprendizado o algoritmo mantém o conjunto de hipóteses correntes e apenas revisa/atualiza essas hipóteses, de acordo com os exemplos que são observados. O conceito vai sendo construído exemplo a exemplo; quando um novo exemplo é observado, um rearranjo do conceito induzido até o momento pode, ou não, se tornar necessário.
- **Aprendizado Não Incremental (*Batch*):** o algoritmo de aprendizado utiliza todo o conjunto de exemplos, de maneira coletiva, para construir o conjunto de hipóteses; o conceito é induzido considerando todos os exemplos de uma só vez.
- **Linguagem de Descrição:** na indução de conceitos é necessário que sejam usadas linguagens para a descrição dos exemplos, para Teoria do Domínio e para as hipóteses formuladas. Geralmente são usadas linguagens formais para esse fim. Há uma variedade de linguagens disponíveis, cada uma com suas vantagens e desvantagens na representação das hipóteses, exemplos e Teoria do Domínio. As linguagens são denominadas:

  linguagem de descrição de instâncias – usada para representar os exemplos do conjunto de treinamento.

  linguagens de descrição de conceitos – usada para descrever as hipóteses geradas.

  linguagem de descrição da Teoria do Domínio – usada para descrever o conhecimento disponível ao sistema de aprendizado.

Aprendizado indutivo pode também ser caracterizado por meio dos mais variados modelos que o implementam, com destaque a:

- Redes Neurais
- Programação Lógica Indutiva
- Árvores de Decisão/Regras
- Algoritmos Genéticos

Com o objetivo de fornecer subsídios para a descrição da investigação conduzida neste trabalho, o restante do capítulo está organizado da seguinte forma: a Seção 2.2 apresenta as principais idéias do que é conhecido como aprendizado usando árvores de decisão, uma vez que este é um dos métodos utilizados numa proposta cooperativa de aprendizado abordada no Capítulo 5, Seção 5.7.

A Seção 2.3 trata da apresentação dos principais conceitos pertinentes à área de aprendizado usando redes neurais *feedforward*, com o objetivo de padronizar a notação e estabelecer a terminologia básica, uma vez que o trabalho de pesquisa descrito nesta dissertação investiga um tipo particular de rede neural, presente no Capítulo 4.

Como a pesquisa conduzida focaliza, também, uma colaboração entre métodos de aprendizado, a Seção 2.4 discute dois modelos de colaboração que tiveram relativo impacto na área. A apresentação deste dois modelos se justifica não apenas pelo fato de terem sido estudados, mas também por serem abordagens bastante diferenciadas das implementadas neste trabalho.

2.2 Aprendizado Usando Árvores de Decisão

A indução de árvores de decisão a partir de um conjunto de exemplos é uma das técnicas mais bem sucedidas na área de aprendizado de máquina – o vasto número de aplicações que empregam árvores de decisão é evidência desse sucesso [Mitchell 1998], [Nicoletti 1994], [Quinlan 1979], [Quinlan 1986], [Quinlan 1993], [Quinlan 1996], [Gou et al. 1998] e [Yeang 2001] .

O algoritmo que viabilizou o uso de árvores de decisão como representação do conceito aprendido foi o ID3 [Quinlan 1979]. A grande contribuição desse algoritmo foi, além de propor o uso de árvores de decisão como uma linguagem de representação de conhecimento, a do uso do conceito de entropia (inspirado na Teoria de Decisão [Jeffrey 1965]) como uma medida da "contribuição" de um atributo na caracterização do conceito.

A partir da proposta original do ID3, muitos outros algoritmos foram propostos, em tentativas de melhorar/refinar o algoritmo original. A família de algoritmos conhecida como TDIDT (*Top Down Induction of Decision Trees*) é exemplo disso.

Os algoritmos e sistemas da família TDIDT possuem a característica de representar suas hipóteses como árvores de decisão. Esta forma de conhecimento, relativamente simples, não tem o poder de representação das redes semânticas ou de outras representações de primeira ordem. Entretanto, apesar da simplicidade, consegue-se expressar conceitos complexos por meio de árvores de decisão. Conseqüentemente, os algoritmos dessa família são menos complexos que outros algoritmos que fazem uso de representações mais poderosas.

Apesar das inúmeras propostas de refinamento e extensão do ID3, esse algoritmo continua sendo amplamente utilizado, e é uma referência quando a abordagem utilizada para o aprendizado é feita por meio do uso de árvores de decisão, razão pela qual é abordado de maneira geral (i.e. como parte integrante da família TDIDT) nas próximas subseções.

2.2.1 Árvores de Decisão como Linguagem de Descrição de Hipóteses

Em [Utgoff 1989] é apresentada uma definição formal de árvores de decisão como sendo uma árvore composta por n (n>1) nós, onde tais nós podem ser:

1. Um nó folha – ou nó resposta – que contém um nome de classe, ou
2. Um nó raiz – ou nó de decisão – que contém um atributo de teste que, para cada um dos possíveis valores deste atributo existe um ramo para uma outra árvore de decisão.

As árvores de decisão, na família TDIDT, são construídas de uma maneira *top-down* como o próprio nome já diz, ou seja, da raiz até as folhas. A indução *top-down* realiza uma busca *hill-climbing*, guiada por uma heurística estatística e sem *backtracking*.

O exemplo a seguir, extraído de [Mitchell 1998], mostra uma árvore de decisão (Figura 2.2) que caracteriza o conceito de “dia conveniente para jogar tênis” (ou o seu complementar, “dia não conveniente para jogar tênis”) induzida

usando o ID3 a partir do conjunto de treinamento mostrado na Tabela 2.1. Cada exemplo do conjunto é descrito por quatro valores de atributo e sua respectiva classe. Os atributos e seus possíveis valores são:

- Aparência – ensolarado, nublado, chuvoso
- Temperatura – baixa, média, alta
- Umidade – alta, normal
- Vento – fraco, forte
- Jogar Tênis (classe) – sim, não

Tabela 2.1: Exemplos de treinamento do conceito "Jogar Tênis".

Nº	Aparência	Temperatura	Umidade	Vento	Jogar Tênis
1	ensolarado	alta	alta	fraco	não
2	ensolarado	alta	alta	forte	não
3	nublado	alta	alta	fraco	sim
4	chuvoso	média	alta	fraco	sim
5	chuvoso	baixa	normal	fraco	sim
6	chuvoso	baixa	normal	forte	não
7	nublado	baixa	normal	forte	sim
8	ensolarado	média	alta	fraco	não
9	ensolarado	baixa	normal	fraco	sim
10	chuvoso	média	normal	fraco	sim
11	ensolarado	média	normal	forte	sim
12	nublado	média	alta	forte	sim
13	nublado	alta	normal	fraco	sim
14	chuvoso	média	alta	forte	não

É importante notar que muito embora o atributo Temperatura descreva instâncias do conjunto de treinamento, tal atributo foi comparece na árvore mostrada na Figura 2.2. O atributo Temperatura foi descartado pela heurística estatística, i.e. função de avaliação, durante a construção da árvore (mais detalhes são vistos nas próximas seções e no Capítulo 3).

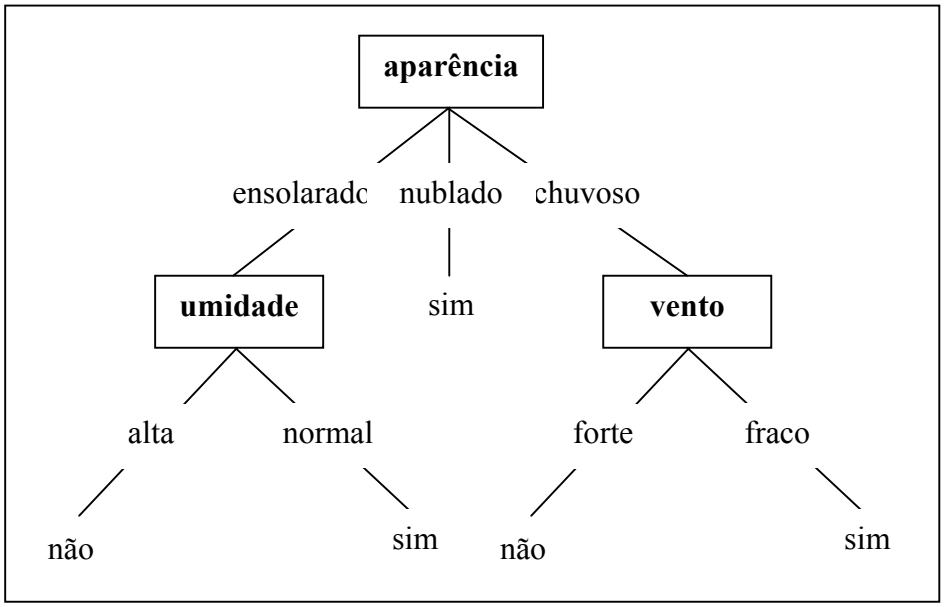


Figura 2.2: Árvore de decisão induzida a partir da Tabela 2.1.

2.2.2 Algoritmo Geral da Família TDIDT

Basicamente, a construção de árvores de decisão a partir de exemplos disponíveis é um processo iterativo, onde cada nó da árvore corresponde a um atributo, seja ele de classe ou não. Geralmente, as possíveis ramificações pertencentes a cada nó correspondem aos valores que o atributo associado ao nó em questão pode ter.

O objetivo do algoritmo de aprendizado de árvores de decisão é gerar a menor árvore que classifique corretamente todo o conjunto de exemplos. Para isso, é preciso escolher, a cada passo, o atributo mais relevante, ou seja, o atributo que melhor agrupa os exemplos segundo o valor de suas respectivas classes.

Para a escolha do atributo mais relevante é utilizada uma função denominada função de avaliação. Uma das diferenças presentes nos algoritmos de aprendizado de árvores de decisão está justamente na escolha da função de avaliação. Outra característica comumente encontrada nestes algoritmos é a condição de parada, responsável por verificar, a cada passo, as condições necessárias para interromper o processo de criação de nós. A condição de parada, geralmente, é a de que todos os exemplos pertençam à mesma classe,

ou pelo menos a maioria deles. Caso isto se verifique, então um nó folha é criado rotulado com o valor da classe predominante correspondente.

A Tabela 2.2 apresenta o algoritmo geral dos sistemas da família TDIDT, que se inicia com uma árvore de decisão vazia, e é construída até que todos os exemplos de treinamento possam ser classificados corretamente.

Tabela 2.2: Algoritmo geral dos sistemas da família TDIDT.

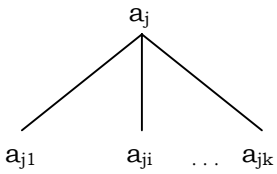
Dados um conjunto de exemplos de treinamento E
uma condição de parada t(E)
uma função de avaliação aval(E,atributo)

Se todos os exemplos em E satisfazem a condição de parada t(E)

Então retorne valor da classe

Caso contrário

1. Para cada atributo a_i determine o valor da função aval(E,a_i). Seja a_j o atributo que possui o melhor valor de aval(E,a_i), ou seja, a_j é o atributo mais relevante.
2. Se a_j possui valores a_{j1}, a_{j2}, ... , a_{jk} crie o seguinte nó da árvore



3. Particione os exemplos do conjunto E nos subconjuntos E_1, E_2, ... , E_k segundo os valores de a_j na árvore de decisão.
4. Aplique o algoritmo recursivamente para cada um dos subconjuntos E_i.

Como comentado anteriormente, é necessário que uma função de avaliação seja utilizada para determinar, em cada passo do processo de construção de uma árvore de decisão, qual atributo é mais relevante para a classificação dos exemplos disponíveis. Descobrir o atributo mais relevante é analisar a capacidade de cada um em separar os exemplos nas várias classes disponíveis. Entre as várias funções de avaliação utilizadas para descobrir o atributo mais relevante estão:

1. Medida de informação
2. Estatística χ^2 (Chi-quadrado)
3. Estatística G (baseada em medida de informação)

4. Índice de diversidade GINI
5. Medida Gain-Ratio (variante da medida de informação)

Experimentos realizados em [Mingers 1989] mostraram que a aplicação de qualquer uma das funções de avaliação anteriores faz com que as árvores obtidas tenham a metade do tamanho (número de nós folha) das árvores geradas utilizando um critério de seleção aleatória dos atributos.

Com relação à precisão na classificação de novas instâncias não há melhora significativa com o uso de funções de avaliação, ou seja, a seleção aleatória de atributos produz árvores tão precisas quanto às árvores construídas a partir do uso de alguma função de avaliação. Entretanto, vale lembrar que o uso da função de avaliação produz árvores bem menores, e o tamanho da árvore é um dos fatores, apesar de não ser o único, que tornam a árvore mais compreensível.

Uma outra característica importante a ser levada em consideração quando da indução de árvores de decisão, é a natureza dos atributos que descrevem o conjunto de treinamento. Uma possível definição da natureza de atributos com base nos seus valores pode ser [Wilson e Martinez 1997]:

- *linear e continuo* – atributos dessa natureza utilizam valores reais para expressar uma determinada característica da instância, como por exemplo, velocidade, comprimento, etc.
- *linear e discreto* – como o próprio nome já diz, estes atributos assumem apenas um conjunto linear e discreto de valores, como por exemplo, número de crianças, quantidade de produtos, etc.
- *nominal* – estes tipos de atributos apresentam valores discretos, mas que não são lineares. Por exemplo, o atributo cor pode assumir o valor verde, azul, preto, branco, etc.

2.3 Aprendizado Usando Redes Neurais *Feedforward*

As redes neurais foram inspiradas na observação e estudos de modelos biológicos de aprendizado; elas apresentam, entretanto, um grau de simplicidade e facilidade de compreensão maior que os modelos biológicos, que se caracterizam por apresentarem um alto grau de complexidade.

O histórico sobre o desenvolvimento da área de redes neurais

apresentado a seguir foi extraído de [Haykin 1994]. O trabalho pioneiro em redes neurais é de autoria de McCulloch e Pitts em [McCulloch e Pitts 1943]. Cerca de quinze anos depois Rosenblatt apresentou o perceptron em [Rosenblatt 1957], juntamente com a prova do teorema de convergência utilizada para o aprendizado. Em 1969, entretanto, Minsky e Papert em [Minsky e Papert 1969] demonstraram as limitações do *perceptron* quando da resolução de problemas não-lineares. A partir desta data as pesquisas na área de redes neurais não avançaram muito até que, em 1982, o artigo de Hopfield envolvendo redes recorrentes [Hopfield 1982] e o trabalho de Kohonen em mapas auto-organizáveis [Kohonen 1982] provocaram um ressurgimento das pesquisas na área. Em 1986 Rumelhart e McClelland [Rumelhart et al. 1986] criaram o algoritmo *backpropagation* e, com ele, a consolidação da pesquisa na área. Atualmente as redes neurais já estão consagradas e se tornaram objetos de grandes projetos de pesquisa e aplicações.

Basicamente, as redes neurais são caracterizadas pelo uso de unidades de processamento, chamadas neurônios ou nós. Os neurônios são interligados através de conexões associadas a valores reais, denominados pesos; a alteração dos pesos faz com que a rede neural se adapte a um determinado domínio descrito por um conjunto de treinamento.

Na literatura podem ser encontradas diferentes redes neurais. Entretanto, como comentado anteriormente, este trabalho aborda apenas as redes conhecidas como *feedforward*, uma vez que são relevantes a este trabalho de pesquisa.

As redes neurais *feedforward* são redes constituídas por neurônios dispostos em camadas. As camadas podem ser de entrada, intermediária e de saída, nessa ordem. Os neurônios de uma mesma camada não possuem conexões entre si; conexões existem apenas entre neurônios de camadas diferentes. O sinal propagado na rede neural *feedforward* começa a partir dos neurônios da camada de entrada, até os neurônios da camada de saída, passando consecutivamente pelos neurônios da(s) camada(s) intermediária(s) exitente(s). Uma outra característica das redes *feedforward* é que não existe retorno (*loop* ou *feedback*) do sinal propagado.

A Figura 2.3 mostra uma rede neural *feedforward* que possui duas

camadas (a camada de entrada não é contada). É possível notar que, não há conexões entre os nós de uma mesma camada, e o sinal é propagado da camada de entrada até a camada de saída passando pela(s) camada(s) intermediária(s) existente(s).

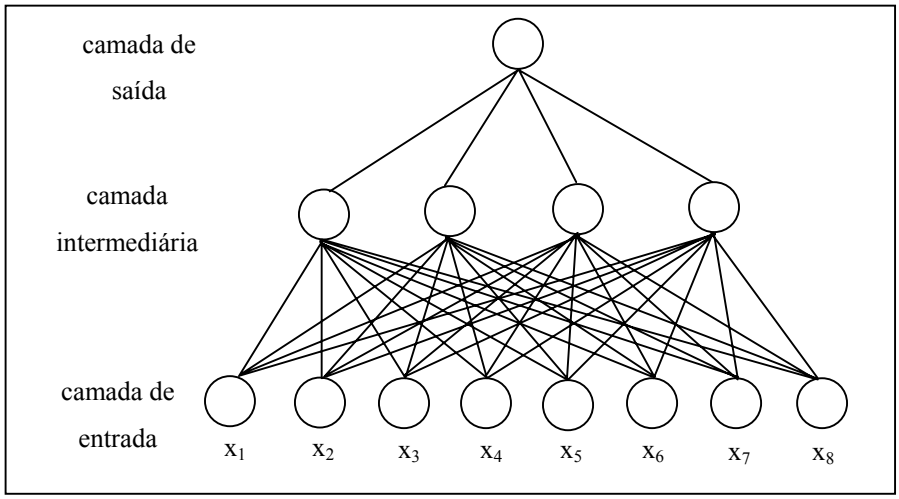


Figura 2.3: Rede neural *feedfoward*.

A seguir será apresentado o *perceptron* seguido do *perceptron* multicamadas, dada a relevância de ambas as arquiteturas para este trabalho.

2.3.1 O *Perceptron*

Um dos tipos de redes neurais mais conhecidos e difundidos é baseado em uma unidade chamada *perceptron*. Tal unidade tem como entrada um vetor de valores reais e calcula a combinação linear dessas entradas, podendo apresentar na saída o valor 1 ou -1. O *perceptron* foi proposto em [Rosenblatt 1957] e, embora tenha sido superado por outras arquiteturas mais poderosas, ainda é empregado devido à sua simplicidade, habilidade e rapidez no treinamento de classificação.

A Figura 2.4 mostra os componentes básicos de um *perceptron*, bem como seu processamento. Na figura, w representa vetor de pesos (w_0 faz o papel de *bias*), net armazena o valor do somatório (i.e. armazena a combinação linear ponderada das entradas), y é o sinal de ativação do neurônio, o é o valor de

saída do neurônio e x_1, ..., x_j representa o vetor de entrada para a rede.

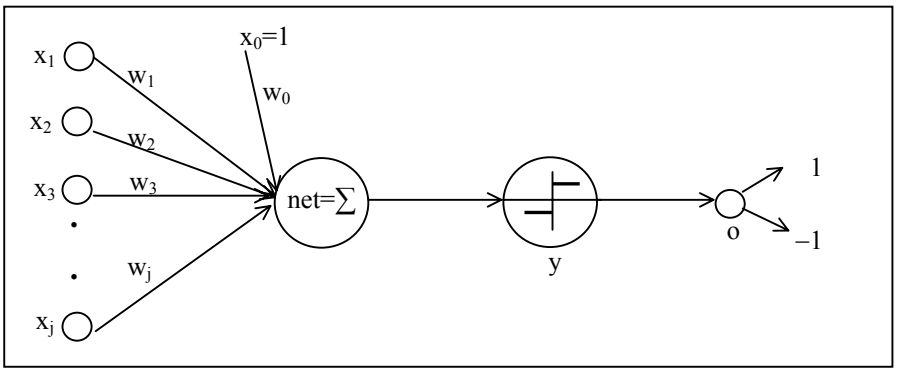


Figura 2.4: Diagrama de funcionamento do perceptron.

O processamento do *perceptron* pode ser escrito de acordo com as seguintes equações:

$$\text{net} = \sum_{j=0}^{n} w_j x_j \tag{2.1}$$

$$o = y(\text{net}) \tag{2.2}$$

A função de ativação y, denominada *hard-limiter*, é definida pela expressão (2.3):

$$y = \begin{cases} 1 & \text{se} \quad \text{net} > 0 \\ -1 & \text{caso} \quad \text{contrário} \end{cases} \tag{2.3}$$

Usando a equação (2.3), a saída da rede será 1 ou −1, dependendo do vetor de entrada. Assim a rede é capaz de classificar se uma instância apresentada como entrada pertence a uma das duas classes. Note que w_0 é o limiar (*bias*) que a combinação ponderada das entradas deve exceder.

O hiperplano separador entre as duas classes é dado por net=0, o qual, divide as duas classes no espaço n-dimensional em questão; dessa forma, a rede *perceptron* de uma camada representa uma função linear discriminante. A reescrita da equação (2.1) para um espaço bidimensional é:

$$w_0 + w_{11}y_1 + w_{21}y_2 = 0 \tag{2.4}$$

A equação (2.4) é reescrita em função de y_2 na equação (2.5), a representação geométrica dessa equação é apresentada na Figura 2.5, onde a linha em negrito divide o espaço dos elementos "+ " e "–", podendo discriminar linearmente tais elementos:

$$y_2 = -\frac{w_{11}}{w_{21}}y_1 - \frac{w_0}{w_{21}} \tag{2.5}$$

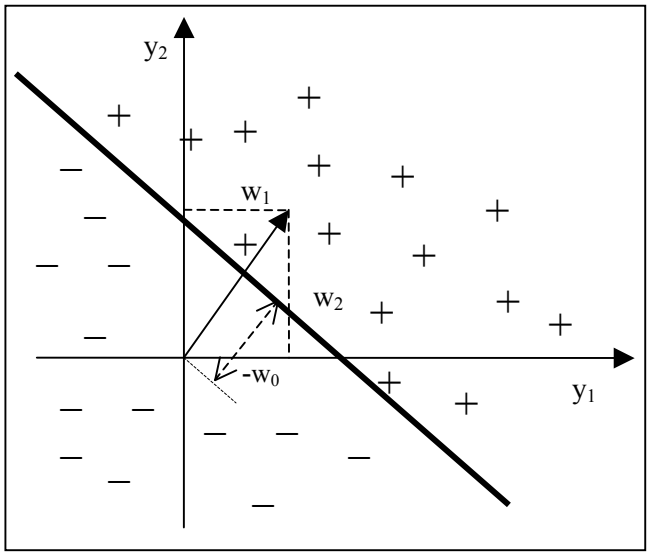


Figura 2.5: Representação geométrica da equação (2.5).

Algoritmo de Treinamento do *Perceptron* (Regra Delta)

O *perceptron* realiza o aprendizado por meio do ajuste do hiperplano de decisão no espaço de maneira a separar as instâncias de acordo com as duas classes às quais elas pertencem. Uma maneira de ajustar esse hiperplano é por meio da atualização de um vetor de pesos. O vetor, geralmente inicializado de forma aleatória, é modificado toda a vez que o *perceptron* classificar uma instância erradamente. Este processo é repetido até que o *perceptron* classifique corretamente as instâncias de treinamento.

Sob a perspectiva de aprendizado, a regra de treinamento envolve a

escolha do vetor de pesos mais apropriado, logo o espaço de hipóteses considerado no aprendizado do *perceptron* é o conjunto de todos os valores reais, e é dado por:

$$H = \{ \mathbf{w} \mid \mathbf{w} \in \Re^{(n+1)} \}$$

onde **w** é o vetor de pesos do *perceptron*, e (n+1) é a quantidade de elementos no vetor mais o *bias*.

Para que o vetor de pesos seja encontrado no espaço de hipóteses acima, é utilizada a regra delta de treinamento do *perceptron* dada por:

$$w_i = w_i + \Delta w_i$$

onde (2.6)

$$\Delta w_i = \eta(t - o)x_i$$

As variáveis t e o representam a classe do exemplo em questão e a saída da rede relativa ao exemplo, respectivamente. Além disso, η é uma constante positiva denominada taxa de aprendizado. O objetivo da taxa de aprendizado é regular a intensidade da alteração dos pesos.

É possível provar utilizando um número finito de passos que a regra de treinamento converge, i.e. encontra um vetor de pesos que classifica corretamente todo o conjunto de treinamento, desde que tal conjunto seja linearmente separável (veja [Minsky e Papert 1969]). Obviamente, se o conjunto de treinamento não for linearmente separável, esse vetor de pesos é impossível de ser encontrado. Mais detalhes sobre o *perceptron* podem ser encontrados em várias referências (ver [Haykin 1994] e [Gallant 1994] por exemplo).

2.3.2 *Perceptron* Multicamadas

O *perceptron* multicamadas foi proposto para contornar uma limitação do *perceptron*, que é a de aprender apenas a partir de conjuntos de treinamento cujas instâncias têm classes linearmente separáveis. O *perceptron* multicamadas é capaz de aprender a classificar instâncias de acordo com suas respectivas classes, a partir de um conjunto de treinamento cujas instâncias têm, ou não, classes não linearmente separáveis. A Figura 2.6 (lado esquerdo)

apresenta as regiões de decisão determinadas pelo *perceptron* multicamadas (lado direito) da Figura 2.6.

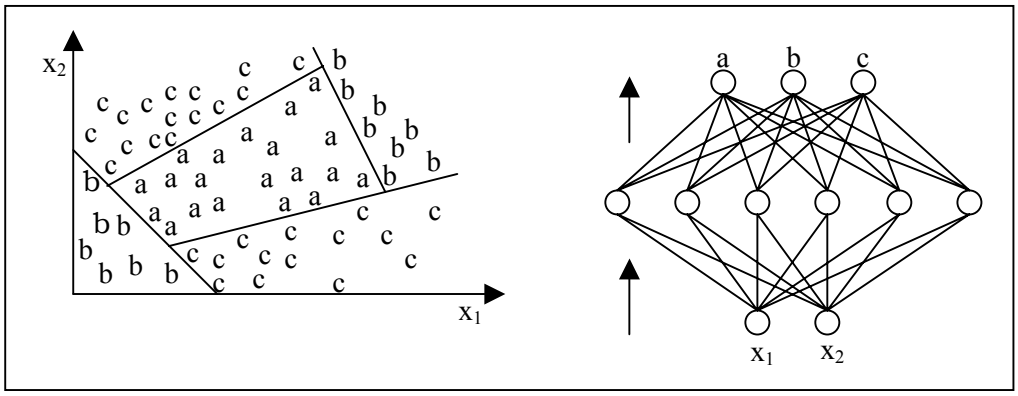


Figura 2.6: Regiões de decisão mapeadas por um perceptron multicamadas.

Os neurônios usualmente utilizados em *perceptron* multicamadas são os mesmos utilizados no *perceptron*, entretanto, possuem como função de ativação algum tipo de curva sigmoide, como por exemplo, a função denotada por:

$$y_k = \frac{1}{1 + e^{-net_k / q}} \tag{2.7}$$

Outra função sigmoide bem conhecida é a tangente hiperbólica, dada por:

$$y_k = \frac{1 - e^{-net_k / q}}{1 + e^{-net_k / q}} \tag{2.8}$$

Em ambas as funções acima, q é uma constante positiva que determina a inclinação da curva sigmoid e k representam o k-ésimo neurônio da rede. A Figura 2.7 (a) e (b) mostra a representações geométricas das funções (2.7) e (2.8), respectivamente. Para que possam ser usadas pelo algoritmo *backpropagation*, que é um dos algoritmos mais investigados e utilizados em *perceptron* multicamadas, as funções de ativação devem ser diferenciáveis.

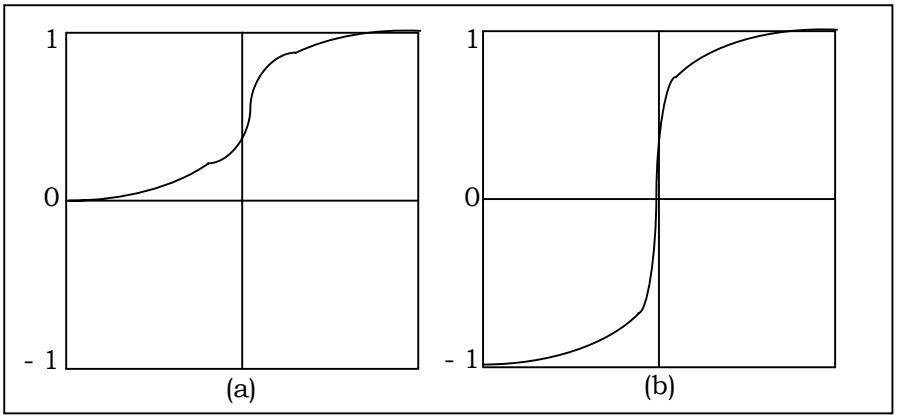


Figura 2.7: Função sigmoid (a) logística e (b) tangente hiperbólica.

2.4 Considerações sobre a Colaboração Simbólico-Conexionista

Existem várias maneiras de viabilizar a combinação dos modelos simbólico e conexionista de aprendizado. Os primeiros trabalhos que abordam esses dois modelos de aprendizado tiveram como objetivo comparar o desempenho de ambos (representados pelo ID3 [Quinlan 1986] e *backpropagation* [Rumelhart et al. 1986] respectivamente), em vários domínios de conhecimento. Os resultados obtidos, entretanto, foram razoavelmente vagos e não conclusivos, como comprovam os dados publicados em [Fisher e McKusick 1989], [Mooney et al. 1989] e [Weiss e Kapouleas 1989].

O conceito aprendido por uma rede neural é a própria rede obtida após a fase de treinamento, representada por meio de:

- sua tolopogia
- das funções de transferência usadas entre as unidades
- dos pesos das conexões
- dos valores da função sigmóide definida nas unidades

A expressão do conceito é dada, pois, por esse conjunto de características que expressam a rede. A compreensão de um conceito assim representado implica o entendimento, por parte do ser humano, do que esse grupo de características representa. É óbvio que essa compreensão é bastante difícil de ser conseguida. Dentre as várias razões que contribuem para essa

dificuldade, segundo [Craven 1996], estão:

- Redes neurais típicas têm centenas ou milhares de parâmetros com valores reais que, essencialmente, codificam relações entre valores de entrada, x e classe, y. Embora a codificação de parâmetros não seja complicada de ser entendida, o alto número de tais parâmetros contribui para tornar bastante difícil o seu entendimento.
- Em redes neurais multicamadas parâmetros podem representar relações não lineares e não monotônicas entre os valores de entrada e de saída. Assim sendo, usualmente não é possível determinar, isoladamente, os efeitos que uma determinada característica tem, na determinação de uma classe, uma vez que tais efeitos podem estar mascarados pela atuação de outras características.
- Tais relações não-linerares e não-monotônicas são representadas pelas unidades intermediárias da rede, que combinam as entradas de múltiplas características, permitindo que o modelo se beneficie das dependências entre elas. O entendimento das unidades intermediárias é bastante difícil porque freqüentemente essas unidades aprendem representações distribuídas [Hinton 1990].

Unidades em camadas intermediárias podem ser pensadas como representações de atributos "derivados". Em representações distribuídas, entretanto, essas características intermediárias podem não corresponder a características do domínio do problema. Características que são significativas no contexto do domínio do problema são freqüentemente codificadas como padrões de ativação que acontecem entre as unidades intermediárias. De maneira similar, cada unidade intermediária desempenha uma parte na representação de várias características derivadas.

Uma das principais justificativas que suportam o investimento na combinação e/ou integração dos modelos simbólico e conexionista é a 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 é difícil de serem interpretadas após o treinamento. Contrastando com redes neurais, as soluções encontradas por sistemas de aprendizado simbólico são, na maioria das vezes, mais passíveis de serem compreendidas por humanos.

Como comentado em [Nicoletti 1998/2000]. "A colaboração simbólico-conexionista pode acontecer em diferentes níveis e sob diferentes formas. As pesquisas na área investem principalmente em técnicas que permitem a tradução de uma representação em outra. Tal tradução pode acontecer como resultado de um processo específico que mapeia uma representação em outra (por exemplo, algoritmos SUBSET (implementado no KBANN [Towell 1991]) e MofN [Towell e Shavlik 1993], algoritmo KT [Fu 1991], [Fu 1994] e, particularmente, a técnica de tradução implementada pelo MACIE [Gallant 1988] e [Gallant 1994], que subsidia a explicação das soluções encontradas por sistemas especialistas conexionistas Pode também acontecer como conseqüência de um processo de aprendizado (por exemplo, o sistema TREPAN [Craven 1996]).

Todas essas abordagens, entretanto, podem ser classificadas como cooperativas no sentido que abordam distintamente uma das duas representações (simbólica ou conexionista) e, de maneira cooperativa, buscam formas de viabilizar a expressão de um mesmo conhecimento (expresso em uma delas), em ambas representações. Já a abordagem de árvores perceptron [Utgoff 1988a], [Utgoff 1988b] aborda especificamente um algoritmo de aprendizado que busca tirar vantagens de uma representação de conhecimento híbrida. A "cooperação" neste caso acontece como uma maneira híbrida (simbólico-conexionista) de representar o conhecimento induzido por um método de aprendizado".

Serão apresentadas duas possíveis formas de colaboração entre modelos simbólicos e conexionistas de aprendizado de máquina, existentes na literatura, o sistema KBANN e o TREPAN.

2.4.1 O KBANN

O modelo KBANN (*Knowledge-Based Artificial Neural Networks*) [Towell 1991] integra dois modelos de aprendizado, o aprendizado baseado em explicação e o aprendizado indutivo usando redes neurais, buscando suprir as falhas destes paradigmas quando usados separadamente.

Como comentado em [Towell 1991], o sistema KBANN por ele proposto e implementado demonstra a hipótese que sistemas podem aprender usando tanto dados quanto teoria combinando, desta maneira, aspectos do aprendizado indutivo e de sistemas baseados em explicação.

A seguir serão descritas as principais idéias que subsidiam a proposta KBANN, de maneira a contextualizar qual o tipo de colaboração simbólico-conexionista este sistema implementa, razão pela qual não serão apresentados os algoritmos específicos de alguns dos módulos do sistema. Uma descrição detalhada do sistema pode ser encontrada em [Towell 1991], [Towell e Shavlik 1993], [Shavlik 1994] e [Nicoletti e Rocha 2000a]. O sistema KBANN mostrado na Figura 2.8 é constituído por três módulos que têm por nome a função que desempenham no sistema. São eles:

- **Módulo de Inserção:** este módulo faz a "tradução" de regras simbólicas, que é a entrada para o sistema, em uma rede neural que tem o mesmo "comportamento de classificação" que o conjunto de regras fornecido.
- **Módulo de Refinamento:** a rede neural criada pelo módulo de Inserção é refinada, usando o algoritmo *backpropagation*. É importante notar que, enquanto o mecanismo de aprendizado é essencialmente o *backpropagation* padrão[3], a rede na qual ele opera não é padrão. É a rede que "reflete o comportamento" do conjunto de regras simbólicas.
- **Módulo de Extração:** extrai um conjunto de regras refinadas a

[3] *Backpropagation* padrão é um nome dado à regra delta generalizada (*generalized delta rule*), a qual, é um algoritmo de treinamento que foi proposto por Rhumelhart e outros [Rhumelhart et al. 1986]. A regra delta generalizada (incluindo o termo momento) é chamada de método da bola pesada (*heavy ball method*) [Polyak 1987].

partir da rede treinada. Como resultado, a informação aprendida pela rede se torna disponível para a avaliação por um especialista humano. Com isso o KBANN permite também que possa acontecer uma realimentação, i.e., o conhecimento refinado obtido pelo sistema pode voltar a lhe ser apresentado, como entrada, e com isso o ciclo voltar a se repetir.

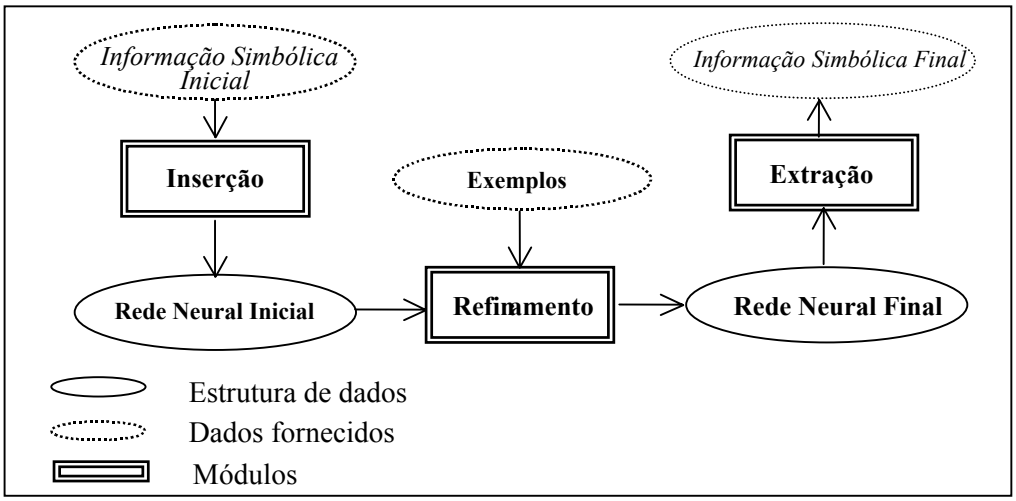


Figura 2.8: O modelo KBANN.

O Módulo de Inserção

Este módulo faz a tradução das regras que são entrada para o sistema, para uma rede neural, utilizando a correspondência apresentada na Tabela 2.3.

Tabela 2.3: Correspondências entre bases de conhecimento e redes neurais.

Base de Conhecimento	Rede Neural
Conclusões Finais	Unidades de Saída
Fatos	Unidades de Entrada
Conclusões Intermediárias	Unidades Intermediárias
Dependências	Conexões Ponderadas

A informação simbólica inicial que é entrada para o módulo de Inserção e que subsidia a criação da rede KBANN é fornecida como um conjunto de

cláusulas de Horn[4] que obedecem a uma determinada sintaxe. Tais cláusulas têm as seguintes restrições:

- devem ser proposicionais - esta restrição se deve ao posterior uso do algoritmo *backpropagation* que não manipula variáveis
- devem ser acíclicas - não pode haver regras, ou combinações de regras, em que o conseqüente aparece também como seu antecedente.

O módulo de Inserção traduz o conjunto de regras em uma rede neural fazendo inicialmente a tradução individual de cada regra em uma pequena subrede; cada uma dessas subredes deve reproduzir, com precisão, o comportamento da regra da qual foi derivada. Feito isso, o algoritmo organiza essas pequenas subredes em uma rede única que reflete o comportamento de todo o conjunto de regras.

O algoritmo que o módulo de Inserção implementa traduz o conjunto de regras de entrada em uma rede e, então, expande a rede obtida, de maneira que ela possa ser capaz de aprender conceitos não cobertos pelo conjunto inicial de regras. Após a tradução de cada regra as subredes resultantes são organizadas em níveis de acordo com a Tabela 2.3 compondo, assim, uma única rede, sendo possível à inserção de unidades, e conexões na rede por parte do usuário com o objetivo de buscar um maior desempenho por parte da rede.

A Figura 2.9 mostra a tradução de um conjunto inicial de regras disjuntivas em uma rede neural, onde w é o peso para as conexões e os valores anexados às unidades representam os *bias*.

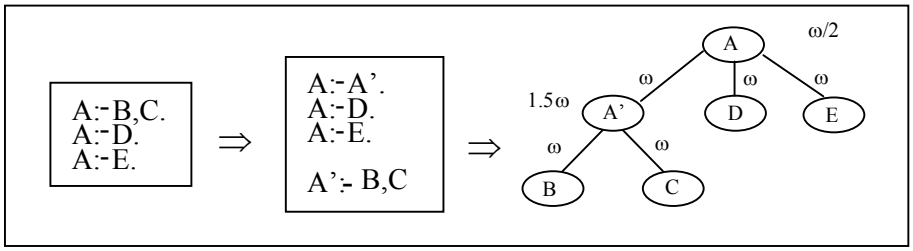


Figura 2.9: Exemplo de tradução de um conjunto de regras em uma rede neural.

[4] Uma cláusula de Horn é uma cláusula que contém no máximo um literal positivo, por exemplo, {~P, ~Q,~, ~R, S} onde os literais negativos são as hipóteses e o literal positivo é a conclusão, tal sentença é equivalente a (P∧Q∧R)⇒S, i.e. S:-P,Q,R.

O Módulo de Refinamento

Após a criação da rede, pelo módulo de Inserção, o módulo de Refinamento utiliza o algoritmo *backpropagation* [Rumelhart et al. 1986] juntamente com o conjunto de treinamento para treinar a rede. O uso do *backpropagation* é problemático devido ao modo como a rede é gerada pelo módulo de inserção, possuindo assim um alto grau de confiança nas respostas, ou seja, possuem saída de ativação próxima de 0 (zero) ou 1. O modo com que o *backpropagation* é especificado, faz com que haja poucas alterações na rede quando há um alto grau de confiança na resposta independente de estar correta ou não, apresentando assim "resistência" na alteração dos pesos para que a rede classifique corretamente os novos exemplos. A solução adotada neste caso é a utilização da função de erro *cross-entropy* [Hinton 1990] em lugar da função de erro padrão (erro quadrático médio).

O Módulo de Extração

Após o treinamento pelo módulo de refinamento, a rede pode ser usada como um classificador e espera-se um desempenho melhor do que as regras que serviram de base para a criação da rede; ainda não é possível, entretanto, estabelecer um mecanismo de explicação das respostas dadas pela rede. Por essa razão se faz necessário um método para extrair regras da rede; esse método está embutido no módulo de extração.

O KBANN disponibiliza dois métodos de extração de regras a partir de uma rede neural. O algoritmo SubSet tem funcionamento idêntico ao algoritmo KT [Fu 1991], [Fu 1994] e o MofN, que é considerado uma evolução do SubSet. O MofN parte do pressuposto de que as regras geradas pelo SubSet freqüentemente geravam expressões m-de-n, ou seja, expressões da forma *if (M dos seguintes N antecedentes são verdade) then conclusão*. Ambos os algoritmos se propõem a buscar combinações de valores de entrada que deixe o valor de ativação da unidade próximo de 1. A suposição de que as unidades possuem ativação perto de 0 ou 1 simplifica essa tarefa. Isto faz com que os métodos se preocupem apenas com o peso das conexões e ignorem a ativação da unidade de onde se origina a conexão.

2.4.2 O TREPAN

As informações contidas nesta seção foram compiladas das referências [Craven e Shavlik 1994], [Craven e Shavlik 1995], [Craven 1996], [Craven e Shavlik 1996] e [Nicoletti e Rocha 2000b]. O algoritmo TREPAN foi originalmente proposto por Craven em sua tese de doutorado [Craven 1996] e teve como justificativa para seu desenvolvimento a mesma utilizada por métodos de colaboração simbólico-conexionista, que é a extração de hipóteses de redes neurais de forma que a mesma possa ser entendida por seres humanos.

O TREPAN, particularmente, não "cuida" da precisão do conceito induzido, que fica a cargo do próprio treinamento da rede neural. O TREPAN busca tornar uma rede compreensível a seres humanos. Sob esse enfoque pode ser tratado como um algoritmo de aprendizado, que aprende a "tradução" de uma rede neural para uma árvore de decisão, num processo que se assemelha bastante àquele usado por vários algoritmos simbólicos de aprendizado indutivo de máquina.

Nesta tarefa de aprendizado, o conceito a ser aprendido é a função representada pela rede neural, e as hipóteses produzidas nessa tarefa são representadas por meio de uma árvore de decisão que se aproxima da rede neural. A tarefa de aprendizado à qual o sistema TREPAN se propõe [Craven e Shavlik 1996] é descrita na Figura 2.10.

dados:
RN: uma rede neural treinada
CE: o conjunto de exemplos usados para treinar a rede
produzir:
uma descrição do conceito representado por RN, que seja compreensível e que também classifique CE de maneira similar à rede.

Figura 2.10: TREPAN como sistema de aprendizado.

É possível diferenciar o processo de extração de regras do TREPAN dos demais métodos existentes, uma vez que é abordado como um processo de

aprendizado e não como um algoritmo que faz uma busca na arquitetura da rede com o objetivo de gerar regras. Entre as vantagens que este sistema tem em relação aos demais que se propõem à mesma tarefa estão [Craven 1996]:

- não pressupõe que a rede neural tenha uma arquitetura específica,
- não pressupõe que a rede neural tenha sido treinada de acordo com um método específico,
- é geral o suficiente para ser aplicado a vários métodos de aprendizado, e não apenas em redes neurais,
- tem boa escalabilidade em tarefas que exigem grande espaço de instâncias, bem como grandes redes.

Entre as principais características do TREPAN estão:

- um sistema que oferece suporte a todo o processo, denominado, oráculo.
- criação de árvores de decisão utilizando a expansão "melhor-primeiro".
- divisão de tipos,
- divisão de seleção.

A Figura 2.11 mostra a arquitetura geral do sistema TREPAN. Note que o uso do (*backpropagation*) no treinamento da rede é um processo à parte. O TREPAN espera como entradas tanto a rede treinada quanto o conjunto de instâncias usadas no treinamento. Uma das funções do Oráculo é a de providenciar mais exemplos de treinamento, de maneira a viabilizar uma melhor “tradução” da rede em uma árvore de decisão m-de-n.

O TREPAN constrói a árvore de decisão recursivamente, particionando o espaço de instâncias usando uma abordagem do "melhor-primeiro". O TREPAN mantém uma fila de nós-folha que vão sendo expandidos em subárvores, à medida que vão sendo retirados da fila. Cada nó na fila é armazenado com as seguintes informações:

- *um subconjunto de exemplos de treinamento* - são exemplos que "alcançam" o nó.
- *um conjunto de exemplos interrogados* - criados com a ajuda do Oráculo e são usados, juntamente com o conjunto de treinamento,

para a seleção do teste, caso seja um nó interno ou, então, para determinar o rótulo da classe, caso seja um nó folha.

- *um conjunto de restrições* - que descrevem as condições que os exemplos devem satisfazer para alcançar o nó, esta informação é usada quando da criação de um conjunto de exemplos interrogados, que deve acompanhar um novo nó criado.

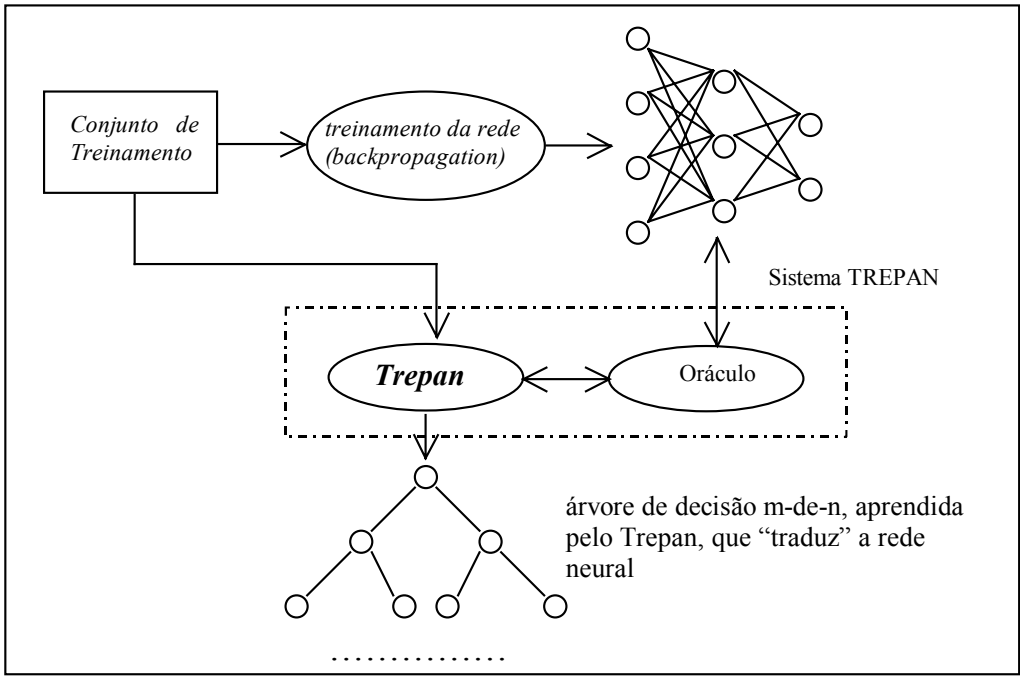


Figura 2.11: Arquitetura do sistema TREPAN.

O processo de expansão de um nó no TREPAN é semelhante aos usados pelos algoritmos convencionais de árvores de decisão: um teste de particionamento é selecionado para o nó, e um nó filho é criado para cada decisão feita a partir do teste. Cada nó filho se torna um nó folha, ou um nó interno.

O TREPAN inicialmente reescreve cada exemplo do conjunto de treinamento com a classe que lhe é atribuída pela rede neural (via Oráculo). A classe original à qual o exemplo pertence, então, é ignorada. Ela foi usada apenas no treinameno da rede.

Segundo [Craven e Shavlik 1996] a grande vantagem do aprendizado usando perguntas é a de que a informação pode ser coletada no momento em

que for necessária, durante o aprendizado. O TREPAN adota essa estratégia de aprendizado (mas não só), operacionalizada pelo uso do Oráculo.

A interação do TREPAN com a rede neural se dá por meio do Oráculo; especificamente, a interação acontece via "interrogação de pertinência" de uma dada instância do espaço de instâncias, com o objetivo de determinar a qual classe essa instância pertence. Dada essa interrogação de pertinência, o Oráculo retorna a classe da instância. Na realidade, a classe é retornada pela rede neural, via Oráculo, uma vez que a resposta a uma "interrogação de pertinência" envolve o uso da rede para classificar a instância. O Oráculo pode desempenhar as seguintes funções:

- determinar a classe dos exemplos usados no treinamento da rede. É importante notar que essa classe não é a classe que comparece na descrição das instâncias, dado que o aprendizado neural é supervisionado. Essa classe é a classe que a rede atribui à instância. Uma vez que o que se pretende é induzir uma árvore de decisão que "traduza" a rede, as instâncias de treinamento devidamente classificadas pela rede fazem parte do conjunto de treinamento usado pelo TREPAN.
- criar mais instâncias de treinamento. Como comentado em [Craven 1996], uma limitação importante em algoritmos convencionais para a indução de árvores de decisão é que a quantidade de instâncias de treinamento usadas para selecionar o teste de ramificação e rotular folhas decresce à medida que a profundidade da árvore aumenta. Assim sendo, testes de rótulos de classes perto da franja da árvore freqüentemente não são escolhidos com muito critério, dado que não existem muitos dados disponíveis para subsidiar uma escolha mais criteriosa. Como o TREPAN tem o Oráculo disponível, ele pode usar tantas instâncias de treinamento quantas achar necessário e, assim, aumentar o conjunto de dados que tem disponível, para avaliar/escolher os testes associados ao nó, ou então, associar uma classe a um nó. Particularmente, o TREPAN assegura que existem pelo menos um número mínimo de exemplos (definido pelo usuário) em um nó,

antes de atribuir a ele um rótulo ou então, nomeá-lo como classe.

O Oráculo é versátil o suficiente para lidar com instâncias incompletas, que podem lhe ser apresentadas por meio de restrições de valores que determinados atributos podem ter. Neste caso, ele gera uma instância completa, selecionando randomicamente valores para cada um daqueles atributos, ao mesmo tempo em que garante a observância das restrições. Para a geração dos valores randômicos sob restrições, usa:

- as instâncias de treinamento para modelar as distribuições de valores dos atributos discretos, e
- método de estimativa de densidade proposto por [Silverman 1986] para modelar os valores de atributos contínuos.

2.5 Considerações Finais

Alguns tópicos abordados neste capítulo foram:

- Visão geral dos aspectos do aprendizado de máquina, particularmente o aprendizado indutivo de máquina, bem como as estratégias empregadas neste tipo de aprendizado.
- A indução de árvores de decisão como um método simples de aprendizado, entretanto, capaz de aprender conceitos relativamente complexos. Sistemas que empregam a indução árvores de decisão utilizam o algoritmo geral da família TDIDT alterando apenas a heurística estatística, a condição de parada e o tratamento dado aos diversos tipos de atributos, bem como a manipulação da árvore de decisão obtida.
- As redes neurais *feedforward* juntamente com sua notação padrão e conceitos básicos. O *perceptron* com seus componentes básicos e seu algoritmo de aprendizado (regra delta de treinamento) ressaltando que o *perceptron* é capaz apenas de aprender problemas linearmente separáveis. Como solução para a limitação do *perceptron* surgiu o *perceptron* multicamadas.
- Justificativa para a abordagem de colaboração simbólico-conexionista e uma revisão de dois modelos distintos que implementam uma colaboração simbólica-neural:

- O modelo simbólico-conexionista KBANN, um sistema composto por três módulos (inserção, refinamento e extração), o qual, é capaz de traduzir regras numa rede neural, e a partir dos exemplos disponíveis fazer o treinamento da rede adquirida, e por fim traduzir a rede treinada num conjunto de regras proposicionais.
- O modelo TREPAN, o qual, encara a tradução de uma rede neural em regras como um processo de aprendizado. Este modelo não pressupõe uma arquitetura de rede definida e não trata da precisão apresentada pela rede, ou seja, este modelo apenas tenta reproduzir numa árvore de decisão o conhecimento da rede neural.

No próximo capítulo serão abordados dois sistemas simbólicos de aprendizado, são eles: o ID3, o qual é parte da família TDIDT, e o NGE que realiza a indução e representa as hipóteses induzidas por meio de hiper-retângulos e possui estreita relação com o sistema RuleNet (foco central de pesquisa neste trabalho).

Capítulo 3

Aprendizado Simbólico – NGE e ID3

3.1 Introdução

Uma das principais características dos sistemas/algoritmos que fazem uso do aprendizado simbólico é a possibilidade de utilizar algum tipo de linguagem para representar o conhecimento adquirido. Grande parte dos sistemas/algoritmos simbólicos é capaz de representar o conhecimento por meio de regras. Entretanto, existem outras linguagens de representação do conceito, que não a de regras.

Este capítulo apresenta um sistema e um algoritmo de aprendizado simbólico, os quais utilizam duas linguagens distintas para a representação do conceito. Primeiramente, é apresentado o algoritmo ID3, o qual faz uso da entropia do conjunto de exemplos para construir uma árvore de decisão (que podem ser traduzidas em regras de decisão). Além disso, dois métodos de discretização de atributos contínuos necessários ao ID3 são abordados. Na seqüência é apresentado o sistema NGE, o qual faz uso do aprendizado baseado em exemplares e expressa o conceito induzido através de hiper-retângulos no espaço euclidiano n-dimensional. Tais hiper-retângulos podem ser traduzidos em regras proposicionais. A apresentação tanto do ID3 quanto NGE são necessárias, uma vez que ambos são comparados ao RuleNet (Capítulo 5), bem como participam de uma abordagem cooperativa associada ao RuleNet,

proposta neste trabalho (ver Seção 5.7).

3.2 O Algoritmo ID3

O ID3 [Quinlan 1986] é um dos sistemas de indução de árvores de decisão mais conhecidos e difundidos, razão pela qual se tornou fonte de inspiração para muitos sistemas.

A idéia básica do ID3 é a mesma apresentada na Seção 2.2 para a construção de árvores de decisão, ou seja, o ID3 constrói árvores de decisão avaliando a contribuição de cada atributo na caracterização das classes.

Para determinar qual atributo particiona melhor os exemplos considerando as diversas classes, o ID3 utiliza uma função de avaliação estatística, denominada ganho de informação, a qual mede o quanto um atributo é capaz de agrupar os exemplos em suas respectivas classes.

Com o intuito de definir o ganho de informação, é necessário apresentar uma medida, denominada entropia, uma vez que o ganho de informação se baseia nela.

3.2.1 Entropia

Se um objeto pode ser classificado em

- N classes diferentes C_1, C_2, ..., C_N, e
- a probabilidade de ocorrência do objeto na classe C_i é $p(C_i)$, $\forall i$, $1 \leq i \leq N$

então a classificação – H(C) – é dada por:

$$H(C) = -\sum_{i=1}^{N} p(C_i) \log_2 p(C_i) \qquad (3.1)$$

Entropia é uma medida de incerteza na classificação do objeto. H(C) é utilizada em teoria da informação é interpretada como a quantidade de informação[1] contida numa mensagem. Quanto maior for o valor de H(C), maior a incerteza com relação ao conteúdo da mensagem.

Nesse contexto, uma árvore de decisão para a classificação de exemplos

[1] A quantidade de informação é especificada em bits.

pode ser vista como uma fonte de informação que, dado um exemplo, gera uma mensagem indicando a classificação do exemplo. Quando o nó da árvore contém apenas exemplos da mesma classe, a entropia é igual a 0 – significando que a decisão de classificação é definitiva para exemplos pertencentes àquele nó [Shaw e Gentry 1990].

Considere novamente a situação de aprendizado cujo conjunto de exemplos é descrito na Tabela 2.1 e representado, novamente, na Tabela 3.1. Este conjunto caracteriza os conceitos "dia bom para se jogar tênis" e "dia ruim para se jogar tênis". O conjunto de atributos é descrito a seguir:

- Aparência – ensolarado, nublado, chuvoso
- Temperatura – baixa, média, alta
- Umidade – alta, normal
- Vento – fraco, forte
- Jogar Tênis (classe) – sim, não

Tabela 3.1: Exemplos de treinamento do conceito "Jogar Tênis".

Nº	Aparência	Temperatura	Umidade	Vento	Jogar Tênis
1	ensolarado	alta	alta	fraco	não
2	ensolarado	alta	alta	forte	não
3	nublado	alta	alta	fraco	sim
4	chuvoso	média	alta	fraco	sim
5	chuvoso	baixa	normal	fraco	sim
6	chuvoso	baixa	normal	forte	não
7	nublado	baixa	normal	forte	sim
8	ensolarado	média	alta	fraco	não
9	ensolarado	baixa	normal	fraco	sim
10	chuvoso	média	normal	fraco	sim
11	ensolarado	média	normal	forte	sim
12	nublado	média	alta	forte	sim
13	nublado	alta	normal	fraco	sim
14	chuvoso	média	alta	forte	não

O algoritmo de indução de árvores de decisão, que usa a entropia como função de avaliação, particiona o conjunto de exemplos em subconjunto com o

intuito de minimizar ao máximo a entropia e continua esse processo, recursivamente, até que a entropia seja 0.

Basicamente, para a construção da árvore de decisão, o que se busca evidenciar, dentre os atributos existentes, é aquele que possui o menor valor de entropia. Para calcular a entropia de classificação relativa a um determinado atributo A, notada por H(C|A), o conjunto de exemplos (Tabela 3.1) é dividido em subconjuntos, agrupando os exemplos que têm em comum o mesmo valor de A. A Tabela 3.2 mostra a partição do conjunto de treinamento, considerando o atributo Vento e seus possíveis valores.

Tabela 3.2: Conjunto de exemplos agrupado por valores do atributo Vento.

Vento	**Jogar Tênis**
forte	não
forte	não
forte	sim
forte	sim
forte	sim
forte	não
fraco	sim
fraco	não
fraco	sim
fraco	sim
fraco	sim
fraco	sim
fraco	sim
fraco	não

A entropia de cada um dos subconjuntos $H(C|A=a_j)$ é calculada para cada valor a_j e é dada pela expressão:

$$H(C \mid A = a_j) = -\sum_{i=1}^{N} p(C_i \mid A = a_j) \log_2 p(C_i \mid A = a_j) \qquad (3.2)$$

onde a função $p(C_i \mid a_j)$ é a probabilidade de que o valor da classe seja C_i quando o valor do atributo A for a_j. Calculando o valor da entropia para cada subconjunto, tem-se:

$$\begin{aligned} H(C \mid \text{Vento} = \text{forte}) = & -p(\text{sim} \mid \text{Vento} = \text{forte}) * \\ & \log_2 p(\text{sim} \mid \text{Vento} = \text{forte}) - \\ & p(\text{não} \mid \text{Vento} = \text{forte}) * \\ & \log_2 p(\text{não} \mid \text{Vento} = \text{forte}) \\ = & (3/6)\log_2(3/6) - (3/6)\log_2(3/6) \\ = & 1 \end{aligned} \tag{3.3}$$

$$\begin{aligned} H(C \mid \text{Vento} = \text{fraco}) = & -p(\text{sim} \mid \text{Vento} = \text{fraco}) * \\ & \log_2 p(\text{sim} \mid \text{Vento} = \text{fraco}) - \\ & p(\text{não} \mid \text{Vento} = \text{fraco}) * \\ & \log_2 p(\text{não} \mid \text{Vento} = \text{fraco}) \\ = & (6/8)\log_2(6/8) - (2/8)\log_2(2/8) \\ = & 0{,}811 \end{aligned} \tag{3.4}$$

Para encontrar a entropia de todo o conjunto – $H(C \mid \text{Vento})$ – deve ser feita a soma ponderada das entropias associadas a cada um dos possíveis valores do atributo em questão. Se um atributo A tem por possíveis valores a_i, i=1, ...,M a entropia de classificação de A é a média ponderada da entropia de cada um dos seus possíveis valores a_i, i=1, ...,M, expressa por:

$$H(C \mid A) = \sum_{i=1}^{M} p(a_i) H(A \mid a_i) \tag{3.5}$$

No caso do atributo Vento tem-se:

$$H(C \mid \text{Vento}) = (6/14) * 1 + (8/14) * 0{,}811 = 0{,}891 \tag{3.6}$$

3.2.2 Ganho de Informação (Information Gain)

Dada a entropia como uma medida de incerteza com relação ao conteúdo da informação contida num conjunto de exemplos, é possível avaliar cada atributo do conjunto na classificação dos exemplos, compondo assim, uma função de avaliação. Esta função, denominada ganho de informação, mede a redução da entropia causada pela partição do conjunto de exemplos a partir do atributo em questão. O ganho de informação, Ganho(C|A), de um atributo A com relação a um ao atributo de classe C é dado por:

$$\text{Ganho}(C \mid A) = H(C) - H(C \mid A) \tag{3.7}$$

onde H(C|A) é descrito pela equação (3.5), e H(C) é a entropia do conjunto de exemplos dada pela equação (3.1). A entropia, H(C), do conjunto de exemplos da Tabela 3.1 é mostrada a seguir:

$$H(C) = -p(\text{sim})\log_2 p(\text{sim}) - p(\text{não})\log_2 p(\text{não}) = 0{,}940 \tag{3.8}$$

Para calcular o ganho de informação do atributo Vento é necessário utilizar os resultados obtidos nas equações (3.8) e (3.6) na equação (3.7). Logo, o ganho de informação do atributo Vento, é dado por:

$$\text{Ganho}(C \mid \text{Vento}) = 0{,}940 - 0{,}891 = 0{,}049 \tag{3.9}$$

3.2.3 Pseudocódigo do ID3

O pseudo-código do algoritmo ID3 é semelhante aquele mostrado na Tabela 2.2 e pode ser visto na Tabela 3.3.

Tabela 3.3: Pseudo-código do algoritmo utilizado pelo ID3 na indução de conceitos que representam a classe positiva (sim) e negativa (não) do conjunto de treinamento.

ID3(Exemplos, Atributo_Alvo,Atributos)
{ Exemplos é o conjunto de treinamento. Atributo_Alvo é o atributo de classe (por exemplo "Jogar Tênis"), ou pode ser qualquer outro atributo que participa da definição das instâncias de treinamento. Atributos é a lista de atributos existentes no conjunto de treinamento. }

Se todos os Exemplos possuem o valor de Atributo_Alvo = sim
Então Retorne um nó com o rótulo = sim
Se todos os Exemplos possuem o valor de Atributo_Alvo = não
Então Retorne um nó com o rótulo = não
Se Atributos está vazio
Então Retorne um nó com o rótulo = valor de Atributo_Alvo mais comum em Exemplos
Senão
A ← o atributo pertencente a Atributos que melhor[2] classifica Exemplos
Crie um nó raiz Node ← A { Node é um nó de decisão para o atributo A }
Para cada possível valor, a_i, de A
Crie um novo ramo a partir de Node com o rótulo = a_i
Seja $Exemplos_{a_i}$ o subconjunto de Exemplos que têm o A=a_i
Se $Exemplos_{a_i}$ está vazio
Então adicione ao ramo em questão um nó com o rótulo = valor de Atributo_Alvo mais comum em Exemplos.
Senão adicione ao novo ramo a seguinte árvore
ID3($Exemplos_{a_i}$,Atributo_Alvo,Atributos-{A})
Retorne Node

É possível notar que a condição de parada da recursão é composta por uma disjunção dada por: o conjunto Exemplos deve possuir o mesmo valor em Atributo_Alvo para todas as instâncias, ou o conjunto Atributos deve estar vazio, ou então não haver instâncias em Exemplos com o valor desejado de

[2] O melhor atributo é escolhido de acordo com a equação (3.7), ou seja, o atributo que apresenta o maior ganho de informação é atribuído ao nó em questão.

Atributo_Alvo.

Além disso, no algoritmo do ID3 o atributo escolhido para fazer o teste de decisão é retirado de Atributos para a próxima chamada do ID3, ou seja, o atributo escolhido não comparece nos ramos a partir do seu próprio nó de decisão.

O algoritmo do ID3 realiza uma busca *hill-climbing* sem *backtracking*, ou seja, uma vez que o ganho de informação determinou o melhor atributo, o algoritmo segue em frente com a recursão e nunca retorna ao nó anterior para avaliar outras opções. Esta estratégia, entretanto, está sujeita a convergir numa solução ótima local, a qual, não corresponde a solução ótima global. Existem, portanto, algumas técnicas utilizadas para evitar este tipo de situação (veja [Nicoletti 1994] e [Mitchell 1998]).

3.2.4 Tratamento de Atributos Contínuos

Na proposta original do ID3 não existe uma estratégia de tratamento de atributos contínuos. O processo de discretização é uma das maneiras de tratar atributos com valores contínuos, presentes nos exemplos de treinamento e fundamental para a pesquisa conduzida neste trabalho, dada a natureza dos domínios utilizados. Tal processo envolve a partição do escopo de valores do atributo categorias. Existem várias estratégias de discretização propostas na literatura. Algumas apresentam característica local versus global, supervisionada versus não-supervisionada e estática versus dinâmica [Dougherty et al. 1995].

Métodos locais, tais como o usado pelo C4.5 [Quinlan 1993], realizam partições em locais específicos, i.e. subconjuntos do conjunto de treinamento em questão. Já os métodos globais, tais como *binning* [Chmielewski e Grzymala-Busse 1994], produzem as partições considerando todos os exemplos do conjunto de treinamento de uma vez só.

Métodos de discretização que não fazem uso do rótulo das classes são chamados métodos não-supervisionados, tais como o método *equal width interval binning*. Já métodos que utilizam os rótulos das classes são chamados de supervisionados.

Por fim, alguns métodos, tais como o *equal width interval* e o *entropy-*

based partitioning [Fayyad e Irani 1993], são classificados como estáticos, pois fazem a discretização de cada atributo individualmente, sem levar em consideração o restante dos atributos. Em contrapartida, os métodos dinâmicos consideram, ao mesmo tempo, todos os atributos do conjunto no processo de discretização.

Dois métodos, um não-supervisionado e outro supervisionado, são abordados nesta subseção, a seguir. A descrição destes métodos é importante, uma vez que ambos serão utilizados na implementação do ID3 realizada para este trabalho.

Equal Width Interval Binning

O *Equal Width Interval Binning* é, provavelmente, o método mais simples de partição de atributos contínuos, produzindo assim, atributos nominais. A idéia básica deste método consiste em ordenar os exemplos de treinamento, usando possíveis valores do atributo em questão, e dividir o conjunto de exemplos em k subconjuntos de mesmo tamanho.

Considere o atributo x, e sejam x_{min} e x_{max} os seus valores mínimo e máximo no conjunto de treinamento, respectivamente. Portanto, a medida de cada uma das k partições é determinada pela seguinte equação:

$$\delta = \frac{x_{max} - x_{min}}{k} \qquad (3.10)$$

Assim é possível construir as partições, $x_{min}+(i*\delta)$, onde i = 1, ... , k-1. Este método é aplicado separadamente para cada atributo contínuo.

Recursive Minimal Entropy Partitioning

Este método supervisionado, apresentado em [Fayyad e Irani 1993], usa a entropia com relação às classes para selecionar entre as possíveis partições aquela que irá determinar os limites dos intervalos.

Seja C um conjunto de instâncias, A um atributo contínuo qualquer, e T um limite entre duas partições, a entropia com relação às classes, E(A,T;C), é dada por:

$$E(A,T;C) = \frac{|C_1|}{|C|} H(C_1) + \frac{|C_2|}{|C|} H(C_2) \qquad (3.11)$$

onde $C_1 \subset C$ e contém os exemplos que possuem o valor do atributo $A \leq T$, $C_2 = C - C_1$, e H(C) é a medida de entropia.

Para um atributo qualquer A, o limite T_{min} escolhido será aquele que minimiza a medida de entropia. Geralmente, os candidatos a T_{min} são limites estabelecidos entre dois exemplos pertencentes a classes diferentes e, para que tais candidatos sejam estabelecidos, é necessário que os exemplos de treinamento sejam ordenados de acordo com os possíveis valores do atributo A em questão.

O princípio apresentado acima pode ser aplicado recursivamente dividindo as duas partições, gerando assim, várias outras partições de diferentes tamanhos para o atributo A. Porém, é necessário que seja especificada uma condição de parada, a qual, é determinada pelo uso da MLDP (*Minimal Description Length Principle*). Logo, o processo recursivo para se:

$$G(A,T;C) < \frac{\log_2(N-1)}{N} + \frac{\Delta(A,T;C)}{N} \qquad (3.12)$$

onde

- N é a quantidade de exemplos no conjunto C,
- $G(A,T;C) = H(C) - E(A,T;C)$,
- $\Delta(A,T;C) = \log_2(3^k - 2) - [k \times H(C) - k_1 \times H(C_1) - k_2 \times H(C_2)]$ sendo que, k_i é a quantidade de rótulos da classe presentes na partição C_i. Como as partições são processadas independentemente, o tamanho de cada partição é variável.

3.3 O Sistema NGE

O sistema de aprendizado NGE (*Nested Generalized Exemplar*), proposto por [Salzberg 1989] e [Salzberg 1991], é baseado numa estratégia de aprendizado

denominada aprendizado baseado em exemplares[3]. Basicamente, em sistemas que implementam tal estratégia de aprendizado, como, por exemplo, o aprendizado baseado em instâncias, todos os exemplos de treinamento são armazenados como pontos no espaço n-dimensional e, a partir daí uma métrica de similaridade é utilizada na classificação de novos exemplos. O conceito é representado por todos os exemplos armazenados no sistema.

O NGE, entretanto, aprende por meio da generalização dos exemplos iniciais (*seeds*), transformando-os em hiper-retângulos. O conceito induzido pelo NGE tem a forma de um conjunto de pontos e hiper-retângulos no espaço Euclidiano. A Figura 3.1 mostra um espaço tridimensional onde a expressão dos conceitos C, C1 e C2 foram induzidas como um conjunto de três hiper-retângulos tridimensionais (dois hiper-retângulos e um ponto). Além disso, esses hiper-retângulos podem ser expressos pelo seguinte conjunto de regras:

SE $((x \geq x_4)$ E $(x \leq x_5))$ **E**
$((y \geq y_4)$ E $(y \leq y_5))$ **E**
$((z \geq z_3)$ E $(z \leq z_5))$ **ENTÃO** C

SE $((x \geq x_1)$ E $(x \leq x_2))$ **E**
$((y \geq y_1)$ E $(y \leq y_3))$ **E**
$((z \geq z_1)$ E $(z \leq z_2))$ **ENTÃO** C1

SE $(x = x_3)$ **E**
$(y = y_2)$ **E**
$(z = z_4)$ **ENTÃO** C2

[3] Um exemplar representa um exemplo armazenado por um sistema de aprendizado.

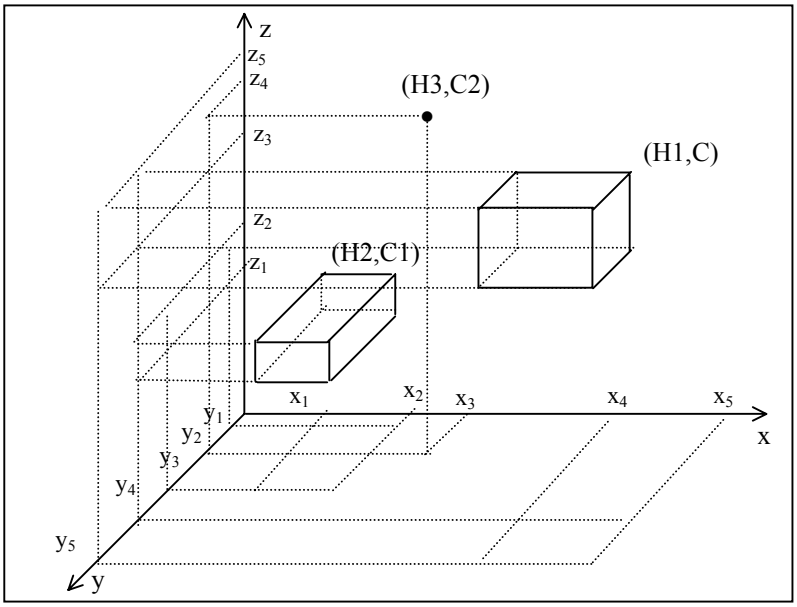


Figura 3.1: Expressão do conceito induzido pelo NGE. A classe C é representada pelo exemplar generalizado H1, a classe C1 pelo exemplar generalizado H2 e a classe C2 pelo exemplar trivial H3.

Como comentado em [Santos 1997] " ... algoritmos simbólicos de aprendizado de máquina implementam o processo de generalização através da substituição de formas simbólicas que expressam o conceito, por fórmulas mais gerais. O algoritmo NGE, entretanto, implementa tal processo modificando os hiper-retângulos (i.e. exemplares) que expressam o conceito, através de sua(s) reestruturação(ões)".

O processo de aprendizado do NGE é composto por duas fases, a saber: a fase de aprendizado, onde a expressão do conceito é induzida e a fase de classificação, onde o conceito induzido é utilizado para classificar novos exemplos.

Como o modelo de aprendizado RuleNet tem paralelos com o modelo do NGE, a seguir são discutidas as principais características do NGE, detalhando a fase de aprendizado e de classificação, de maneira a fornecer informações que irão subsidiar comparações e uma cooperação entre os dois modelos (ver Seções 5.5 e 5.7, respectivamente).

3.3.1 Fase de Aprendizado

O NGE, como todo os algoritmo indutivo, aprende a partir de um conjunto de

treinamento. Tal conjunto é formado por vetores n-dimensionais, onde cada vetor é um conjunto de n-pares atributo/valor de atributo e uma classe associada.

A fase de aprendizado é responsável por induzir o conceito no NGE a partir dos exemplos disponíveis e, é composta por duas partes, inicialização e treinamento. A partir de um conjunto de exemplares criados na inicialização, o NGE compara cada exemplo de treinamento com os exemplares e, usando uma métrica de similaridade (neste caso a similaridade é representada por uma métrica de distância), o NGE realiza a generalizações e especializações nos exemplares usando o exemplo em questão.

Inicialização

O processo de inicialização do NGE consiste na escolha aleatória de um número s (determinado pelo usuário) de exemplos de treinamento, denominados *seeds*, que se tornam exemplares. Estes exemplares já possuem sua classe associada e formam o conceito inicial do NGE como hiper-retângulos triviais, ou seja, pontos no espaço n-dimensional. A partir daí, o NGE está pronto para utilizar o restante dos exemplos de treinamento na generalização/reestruturação dos exemplares existentes e na criação de novos exemplares.

As *seeds* escolhidas possuem considerável influência sobre o desempenho do NGE, ou seja, a disposição dos exemplares iniciais influencia o processo de aprendizado do NGE e, principalmente, a quantidade de hiper-retângulos criados. Sendo assim, o processo de escolha de *seeds* pode ser otimizado com o intuito de melhorar o desempenho do NGE. Em [Figueira e Nicoletti 2004] são propostas três estratégias para a escolha de *seeds*, em adição a estratégia de seleção aleatória, que é parte do NGE. Segue a discussão das quatro estratégias:

1) **Aleatória:** esta é a estratégia padrão utilizada pelo NGE, ou seja, consiste na escolha aleatória de s exemplos de treinamento, os quais se tornam exemplares.
2) **Média:** inicialmente o conjunto de treinamento é processado buscando identificar, para cada atributo que descreve as instâncias de uma determinada classe, o seu intervalo de variação. Uma vez

determinado o intervalo de variação de cada atributo, é achado o ponto médio do intervalo e, com os valores dos pontos-médios dos intervalos associados a cada um dos atributos, é construído um exemplo protótipo da classe em questão. Ao final desta fase existem tantos exemplos-protótipo quanto classes. O conjunto de treinamento então é processado novamente, buscando identificar, para cada classe, a instância mais próxima do exemplo protótipo daquela classe. O exemplo mais próximo é escolhido como *seed*. Neste método o número de *seeds* é igual ao número de classes descritas no conjunto de treinamento. O conjunto inicial de *seeds* tem uma instância de cada classe. Para o cálculo de proximidade é usada a distância euclidiana.

3) **Média e Máxima:** esta estratégia é semelhante ao anterior. Para cada classe, escolhe-se como *seed* tanto a instância do conjunto de treinamento mais próxima ao exemplo protótipo quanto a mais distante. Este método, portanto, escolhe duas *seeds* para cada classe.
4) **Clusterização:** implementa uma colaboração entre métodos de aprendizado. O método de aprendizado não supervisionado conhecido como clusterização é utilizado para evidenciar, no conjunto de treinamento, os *clusters* de exemplos. Os centros de tais *clusters* são utilizados como *seeds* para o NGE. O método de clusterização usado nesse trabalho foi o k-means [Ball 1967]. Uma das vantagens deste método é sua simplicidade. É necessário o uso do k-means algumas vezes no sentido de determinar o melhor número de *clusters* para um domínio específico.

Como comentado anteriormente, as três últimas estratégias são propostas no trabalho em questão como uma alternativa à estratégia aleatória adotada pelo NGE. São comparados os resultados (precisão de classificação e quantidade de hiper-retângulos existentes ao final do treinamento) apresentados pelo NGE utilizando cada uma das estratégias em vários domínios de conhecimento. Embora a precisão de classificação do NGE seja influenciada pela utilização de cada uma das estratégias, a maior contribuição do uso das estratégias está relacionada à quantidade de hiper-retângulos

existentes após o treinamento, ou seja, a descrição do conceito induzido. Em geral, o uso da estratégia da media faz com que o número de hiper-retângulos existentes ao final do treinamento seja menor se comparado ao número obtido pelas outras estratégias. Contudo, a estratégia da média traz algumas desvantagens, tais como a generalização excessiva dos hiper-retângulos. O uso da estratégia de clusterização apresentou um equilíbrio razoável entre precisão e número de hiper-retângulos com relação às demais estratégias.

Treinamento

O treinamento começa efetivamente à medida que novos exemplos do conjunto de treinamento são processados pelo NGE. Os exemplos são processados de forma incremental, ou seja, a cada novo exemplo E, o NGE busca entre todos os hiper-retângulos existentes até então o hiper-retângulo mais próximo $H_{próximo1}$, e o segundo mais próximo $H_{próximo2}$ do exemplo E.

Para que $H_{próximo1}$ e $H_{próximo2}$ sejam encontrados é necessário que o NGE utilize uma métrica de similaridade entre o exemplo E e cada um dos hiper-retângulos H existentes. A métrica utilizada pelo NGE é a distância Euclidiana ponderada, e é definida a seguir:

- Se H é um hiper-retângulo trivial (i.e. um ponto no espaço euclidiano), então

$$D(E,H) = w_H \sqrt{\sum_{i=1}^{n} \left(w_i \frac{E_{f_i} - H_{f_i}}{max_i - min_i} \right)^2} \qquad (3.13)$$

  onde
  - w_H é o peso do exemplar H,
  - w_i é o peso do atributo f_i,
  - E_{fi} é o valor do i-ésimo atributo de E,
  - H_{fi} é é o valor do i-ésimo atributo de H,
  - min_i e max_i são os valores mínimo e máximo, respectivamente, do atributo f_i,
  - n é a quantidade de atributos.

- Se H é um hiper-retângulo generalizado, então

$$D(E,H) = w_H \sqrt{\sum_{i=o}^{n} \left(w_i \frac{dif_i}{max_i - min_i} \right)^2}$$

onde (3.14)

$$dif_i = \begin{cases} E_{f_i} - H_{superior} \text{ se } E_{f_i} > H_{superior} \\ H_{inferior} - E_{f_i} \text{ se } E_{f_i} < H_{inferior} \\ 0 \text{ caso contrário} \end{cases}$$

No NGE, a métrica de distância pode ser entendida da seguinte maneira:

- Se o exemplo E estiver "contido" no hiper-retângulo H, então distancia D(E,H) é zero.
- Se o exemplo E estiver fora do hiper-retângulo H, então a distância é medida ponto-a-ponto, ou ponto-a-hiper-retângulo de acordo com o tipo do hiper-retângulo H.
- Se o exemplo E estiver eqüidistante de dois ou mais hiper-retângulos, então o hiper-retângulo de menor área é o hiper-retângulo mais próximo.

A utilização de pesos nas equações 3.13 e 3.14 tenta melhorar o desempenho do NGE e permitir sua tolerância a ruídos. Cada exemplar H possui um peso w_H, que "mede" a freqüência de uso do exemplar H na predição correta do exemplo E (classe de H=classe de E). Quanto maior w_H, menos confiável é o exemplar em questão. Tal peso é inicializado com 1 e, a partir daí é atualizado de maneira incremental e a cada iteração de acordo com a equação a seguir:

$$w_H = \frac{m}{c} \quad (3.15)$$

onde

- m é o número de vezes que H foi usado, e
- c é o número de vezes que H fez uma predição correta.

Outro peso, w_{f_i}, faz referência a cada i-ésimo atributo que descreve o exemplar. Tal ajuste é uma maneira de evidenciar a relevância de um atributo,

tanto positivamente quanto negativamente, na determinação de $H_{próximo1}$ e $H_{próximo2}$. O NGE adota a estratégia de correção de pesos, w_{fi}, descrita na Tabela 3.4.

Tabela 3.4: Esquema de ajuste dos pesos dos atributos.

Se classe(E) = classe(H), ou seja, H prediz corretamente.
Para cada f_i faça
Se $E_{f_i} = H_{f_i}$ ou $E_{f_i} \in [H_{inferior}, H_{superior}]$ então
$w_{f_i} = w_{f_i} (1-\eta f)$
Senão
$w_{f_i} = w_{f_i} (1+\eta f)$
onde ηf a taxa global de ajuste de atributo (valor default $\eta f=0,2$)
Se classe(E) ≠ classe(H), ou seja, H prediz incorretamente.
Para cada f_i faça
Se $E_{f_i} = H_{f_i}$ ou $E_{f_i} \in [H_{inferior}, H_{superior}]$ então
$w_{f_i} = w_{f_i} (1+\eta f)$
Senão
$w_{f_i} = w_{f_i} (1-\eta f)$

Após a identificação dos hiper-retângulos $H_{próximo1}$ e $H_{próximo2}$, o NGE pode, de acordo com os resultados obtidos, executar um dos três procedimentos apresentados a seguir:

1) Se a classe de E for igual à classe de $H_{próximo1}$, então $H_{próximo1}$ é generalizado de forma a "cobrir" o exemplo E. Cada atributo de $H_{próximo1}$ é ajustado, incluindo assim, o correspondente valor de atributo de E. A Figura 3.2 mostra o processo de generalização de H duas situações: (a) H é um hiper-retângulo generalizado, e (b) H é um hiper-retângulo trivial. A linha tracejada indica a expansão de H como conseqüência do processo de generalização.

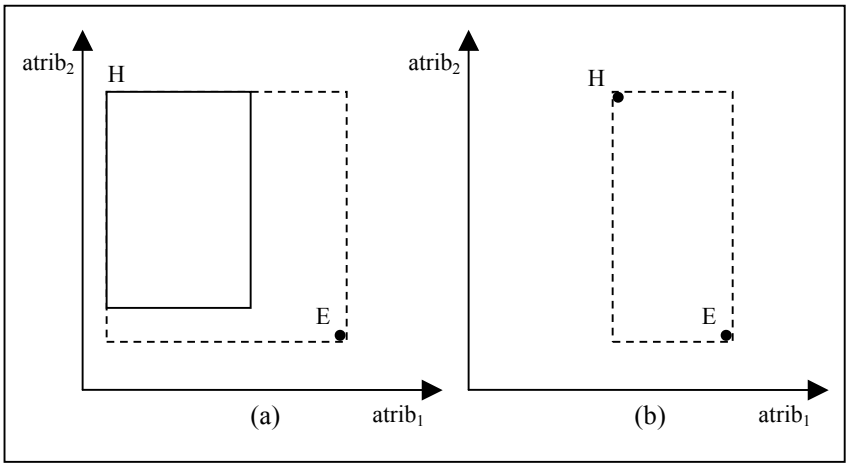


Figura 3.2: Processo de generalização de hiper-retângulos: (a) H é um hiper-retângulo generalizado e (b) H é um hiper-retângulo trivial.

2) Se a classe de E for diferente da classe de $H_{próximo1}$, então o NGE compara a classe de E com a classe de $H_{próximo2}$. Caso a classe de E seja igual à classe de $H_{próximo2}$, então o NGE generaliza $H_{próximo2}$ de acordo com o procedimento descrito em 1), e especializa $H_{próximo1}$ reduzindo-o com o intuito de distanciá-lo de E. O objetivo deste procedimento é aumentar a precisão do sistema evitando a criação excessiva de exemplares. Entretanto, é importante lembrar que a especialização de $H_{próximo1}$ pode causar perda de informação, uma vez que exemplos pertencentes à $H_{próximo1}$ podem ficar "descobertos" após a especialização. O processo de especialização é ilustrado na Figura 3.3, onde o espaço bidimensional é composto por três hiper-retângulos (sendo um hiper-retângulo trivial) e um novo exemplo (Figura 3.3 (a)), notados respectivamente por (H_1,c_1), (H_2,c_2), (H_3,c_2) e (E,c_1), onde c_1 e c_2 representam classes. Na figura o exemplar (H_2,c_2) é especializado por ser o exemplar mais próximo de (E,c_1) e não fazer a classificação correta (Figura 3.3 (b)). O segundo exemplar mais próximo (H_1,c_1) é, então, generalizado pois tem a mesma classe de (E,c_1). Ele é expandido para incluir (E_1,c_1).

3) Se nem $H_{próximo1}$ e nem $H_{próximo2}$ têm a mesma classe do exemplo E, então E se torna um exemplar na forma de hiper-retângulo trivial.

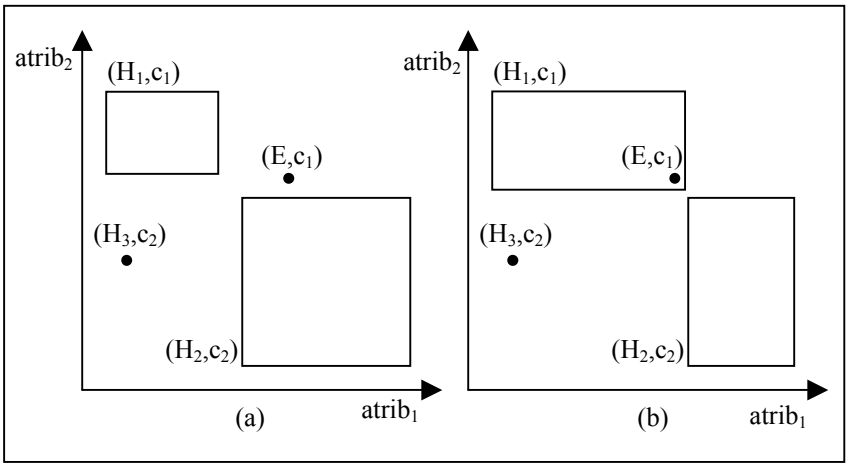


Figura 3.3: Generalização e especialização de H_1 e H_2, respectivamente.

Hiper-retângulos Sobrepostos e Aninhados

Uma das conseqüências da generalização é que o aumento de qualquer hiper-retângulo H_1 pode fazer com que este se sobreponha parcialmente/totalmente a um hiper-retângulo qualquer H_2, mesmo que ambos pertençam a classes diferentes. Se isso acontecer, o NGE atribui a região sobreposta ao hiper-retângulo de menor área. A Figura 3.4(a) mostra dois hiper-retângulos (H_1,c_1), (H_2,c_2) e o exemplo (E,c_1). Com o processo de generalização (H_1,c_1) é expandido, se sobrepondo ao (H_2,c_2) como mostra a Figura 3.4(b).

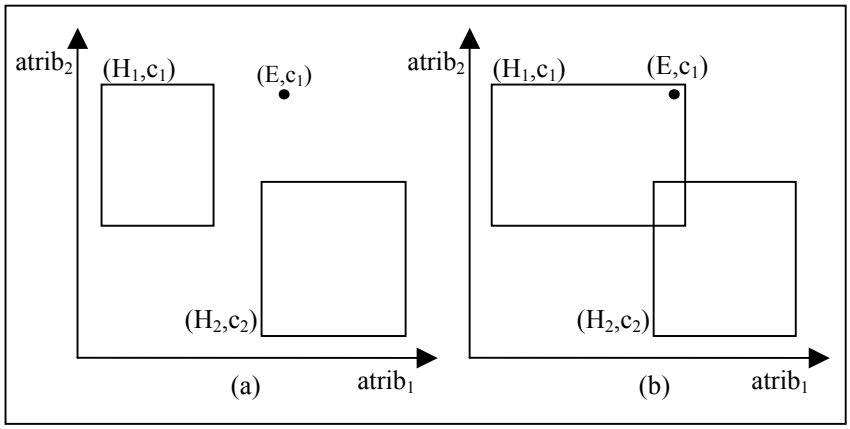


Figura 3.4: Sobreposição de hiper-retângulos de diferentes classe como conseqüência do processo de generalização.

Além da sobreposição parcial de hiper-retângulos, existe a possibilidade de uma sobreposição total entre hiper-retângulos. Esta é uma característica do

sistema NGE, denominada exceção. A Figura 3.5 mostra a criação de exceção a partir da generalização de (H_1,c_1) que, na Figura 3.5(a), é o hiper-retângulo mais próximo de (E,c_1). Após o processo de generalização, (H_1,c_1) se sobrepõe totalmente a (H_2,c_2) criando uma exceção, como mostra a Figura 3.5(b).

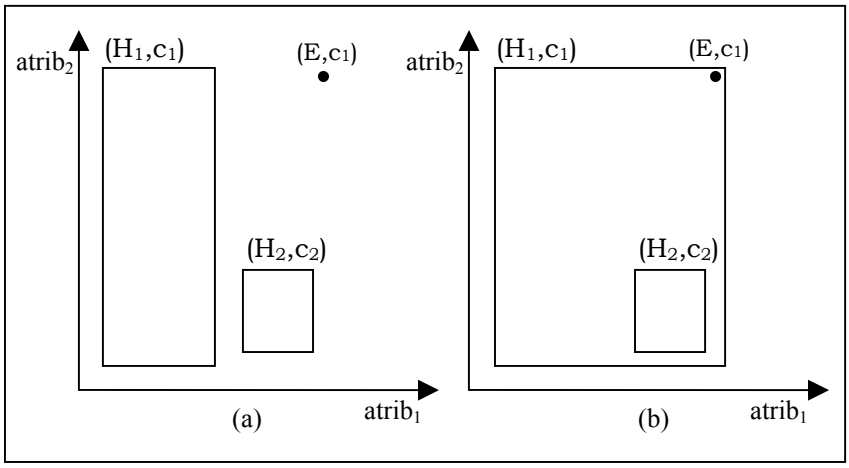


Figura 3.5: Geração de exceção por meio da generalização do hiper-retângulo mais próximo.

Outra situação que gera exceções acontece por meio da generalização do segundo hiper-retângulo mais próximo ao novo exemplo de treinamento, como mostra a Figura 3.6, o exemplo (E,c_2) encontra-se dentro de (H_1,c_1), ou seja, a distâncias entre eles é 0 (Figura 3.6(a)). Como ambos possuem classes diferentes, o segundo hiper-retângulo mais próximo de E é o (H_2,c_2), que é generalizado, pois possui a mesma classe do exemplo de treinamento em questão (Figura 3.6(b)).

Apesar da especialização de H_1, na Figura 3.6, ser esperada, isto não acontece, uma vez que o valor distância entre E e H_1 é zero, não há necessidade de especialização de H_1, pois o uso da exceção impede a perda de informação por parte de H_1.

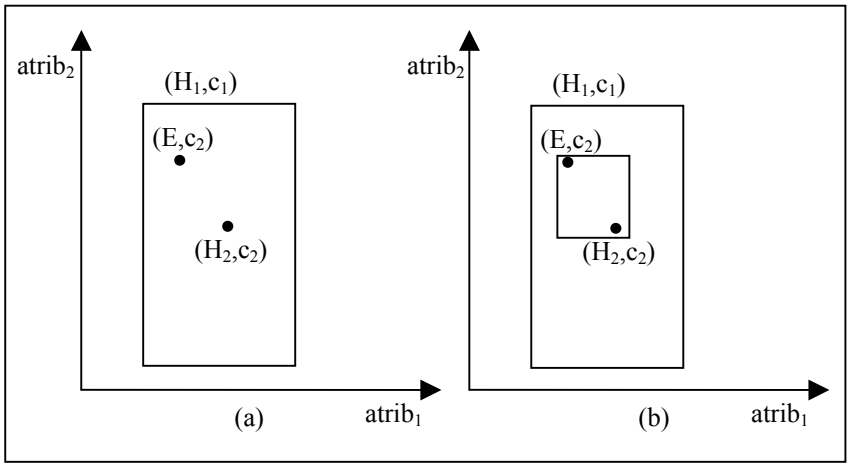


Figura 3.6: Geração de exceção através da generalização do segundo hiper-retângulo mais próximo.

3.3.2 Fase de Classificação

Após a fase de aprendizado, o conceito consiste em todos os hiper-retângulos construídos (triviais ou não) e são utilizados para a classificação de novos exemplos. A classificação de um novo exemplo E é feita através da busca do hiper-retângulo mais próximo de E.

A Figura 3.7 apresenta três situações de classificação do exemplo de teste (E,?) num espaço bidimensional, com cinco exemplares (H_1,c_1), (H_2,c_2), (H_3,c_3), (H_4,c_3) e (H_5,c_2), onde estão representados os conceito que descrevem três classe num domínio com dois atributos e três classes. Na Figura 3.7(a), (E,?) é classificado como sendo da classe c_1, pois o exemplo está mais próximo de (H_1,c_1). Na Figura 3.7(b), (E,?) é classificado como sendo da classe c_2, apesar da sobreposição dos exemplares, a classe do hiper-retângulo com menor área é escolhida para o exemplo em questão. Na Figura 3.7(c), (E,?) é classificado como sendo da classe c_2 por razão idêntica à anterior.

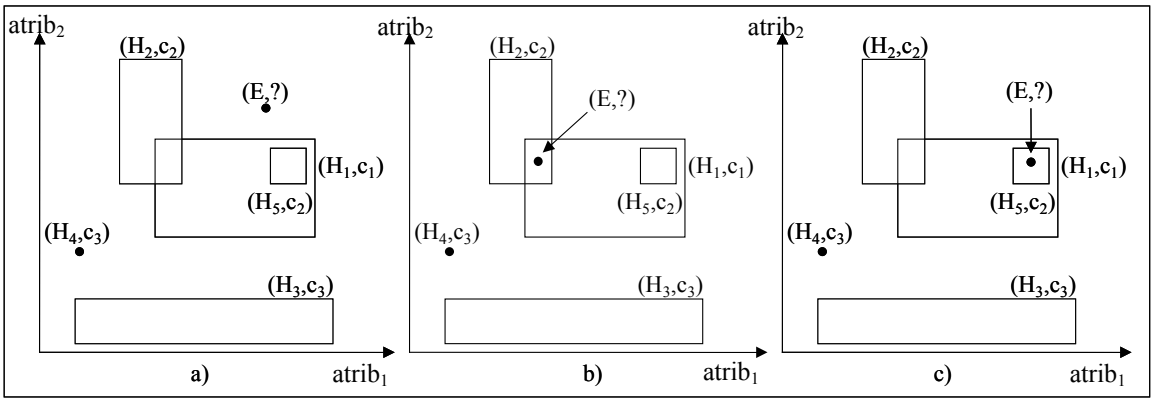


Figura 3.7: Classificação da instância E em três situações distintas.

3.3.3 Algoritmo do NGE

A Tabela 3.5 apresenta o pseudocódigo do NGE, no qual são omitidas as descrições de algumas funções dada a obviedade de seu papel. A estratégia de tratamento de pesos não foi considerada, uma vez que o sistema NGE utilizado neste trabalho não faz uso de qualquer esquema de pesos.

Tabela 3.5: Pseudocódigo do sistema NGE.

```
NGE(Exemplos,S)
Início
{ Exemplos é o conjunto de treinamento.
    S é o número de seeds }
    Para i ← 1 até S faça
        Início
            E ← exemplo_escolhido_aleatoriamente_de(Exemplos)
            Exemplos ← Exemplos – {E}
            criaExemplar(E)
        Fim
    Para cada exemplo E ∈ Exemplos faça
        Início
            H_próximo1 ← hiper-retângulo_H_mais_próximo_de(E)
            H_próximo2 ← segundo_hiper-retângulo_H_mais_próximo_de(E)
            Se classe(H_próximo1)=classe(E) então generalizaExemplar(H_próximo1,E)
            Senão
```

```
                Se classe(H_próximo2)=classe(E) então
                    Início
                        generalizaExemplar(H_próximo2,E)
                        especializaExemplar(H_próximo1,E)
                    Fim
                Senão criaExemplar(E)
        Fim
Fim

criaExemplar(E)
{ E é um exemplo de treinamento
   atributo(i,E) representa o valor do i-ésimo atributo de E  }
Início
    Para cada atributo i ∈ E faça
        Início
            superior(atributo(i,H)) ← atributo(i,E)
            inferior(atributo(i,H)) ← atributo(i,E)
        Fim
Fim

generalizaExemplar(H,E)
Início
    Para cada atributo i ∈ E faça
        Início
            Se atributo(i,E) < inferior(atributo(i,H)) então
                inferior(atributo(i,H)) ← atributo(i,E)
            Senão
                Se atributo(i,E) > superior(atributo(i,H)) então
                    superior(atributo(i,H)) ← atributo(i,E)
        Fim
Fim

especializaExemplar(H,E)
Início
    Para cada atributo i ∈ E faça
        Início
```

```
            Se atributo(i,E) ≥ inferior(atributo(i,H)) e
              atributo(i,E) ≤ superior(atributo(i,H)) então
                  Início
                    Se (atributo(i,H) – inferior(atributo(i,H))) ≤
                        (superior(atributo(i,H)) – atributo(i,H)) então
                            inferior(atributo(i,H)) ← atributo(i,E)
                    Senão superior(atributo(i,H)) ← atributo(i,E)
                  Fim
        Fim
Fim
```

3.4 Considerações Finais

Este capítulo focalizou a apresentação e discussão das principais idéias do algoritmo para construção de árvores de decisão ID3 e do sistema de aprendizado NGE. Esses dois métodos de aprendizado desempenham um papel importante na pesquisa conduzida, pois são usados comparativamente, na avaliação de desempenho do RuleNet, que é o principal foco do trabalho. Além disso, ambos são participantes de um esquema cooperativo com o RuleNet.

O próximo capítulo trata da apresentação e investigação do modelo RuleNet, que pode ser caracterizado como uma rede neural com característica simbólica, adequado a participar de esquemas cooperativos de aprendizado, com outros métodos.

Capítulo 4

O Sistema RuleNet

4.1 Introdução

Muitas das informações apresentadas nesta seção foram extraídas das referências que propõem e descrevem o modelo RuleNet e sua extensão para domínios *fuzzy*, a saber [Dabija e Tschichold-Gurman 1993], [Tschichold-Gurman 1995b] e [Tschichold-Gurman 1997]. As descrições do modelo, entretanto, são omissas em vários detalhes funcionais que possibilitariam sua tradução precisa em um procedimento algorítmico (particularmente aquele do redimensionamento das regiões influência, tratado na Seção 4.5). Isto fez com que parte do trabalho de pesquisa investisse na especificação detalhada em pseudocódigo do algoritmo, descrito no Apêndice A.

O modelo RuleNet é uma rede neural *feedforward* com um algoritmo de aprendizado supervisionado, uma arquitetura dinâmica e saídas discretas. O princípio básico do RuleNet é o de aproximar regiões pertencentes a diferentes classes, no espaço de atributos, usando hiper-retângulos. Entre as principais características deste sistema apresentadas em [Tschichold-Gurman 1997] que motivaram sua escolha para investigação estão:

- o sistema tem uma estrutura dinâmica, dado que inicia a construção da rede apenas com os nós de entrada. Os nós da camada intermediária e de saída são adicionados pelo algoritmo à medida que

o treinamento acontece, sendo assim, a arquitetura final da rede é definida pelo domínio a ser aprendido;

- outras características importantes do sistema RuleNet são sua simplicidade, o curto tempo de treinamento e a eficiência de seu algoritmo de propagação. O algoritmo de treinamento do RuleNet não têm parâmetros que possam afetar a sua convergência e o aprendizado sempre acontece em poucas épocas. Além disso, o algoritmo de propagação é bastante eficiente dado que não implementa operações computacionalmente pesadas;
- tal arquitetura permite também uma propagação paralela, com baixa comunicação entre os nós da camada intermediária, o que facilita a implementação desse modelo em computadores paralelos com um grande número de processadores;
- o conhecimento da rede neural pode ser descrito via regras *if-then* e, conseqüentemente, tanto a extração de regras a partir da rede quanto à tradução de regras em uma rede podem ser realizadas sem perda de informação. O RuleNet é, portanto, adequado para uso em sistemas híbridos onde a cooperação neural-simbólica acontece, geralmente, com o objetivo de refinar o conhecimento simbólico.

4.2 Pré-Requisitos

O RuleNet adota a estrutura geral de um neurônio mostrada na Figura 4.1.

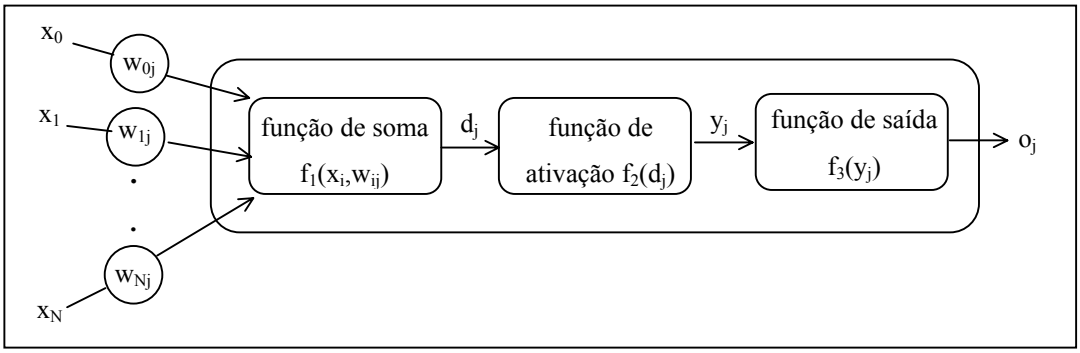


Figura 4.1: Modelo genérico de neurônio adotado pelo RuleNet.

Duas funções de soma freqüentemente utilizadas são o produto interno (usada, por exemplo, no *perceptron*):

$$d_j = \sum_{i=1}^{N} w_{ij} x_i \qquad (4.1)$$

e a distância euclidiana (usada, por exemplo, em redes de Kohonen).

$$d_j = \sqrt{\sum_{i=1}^{N} \left(w_{ij} - x_i \right)^2} \qquad (4.2)$$

O estado de um nó é calculado pela função de ativação. Existe uma grande variedade de funções que podem ser usadas (e.g. sigmóide, gaussiana, linear, *step*, etc.). Todo o modelo de rede neural é constituído de nós como mostrado na Figura 4.1, onde diferentes funções de soma, ativação e saída são combinadas. Por exemplo, os nós da versão original do *perceptron* têm a função de soma com a equação (4.1), uma função de ativação *step*, e uma função identidade como saída.

4.3 Características do RuleNet

A seguir serão apresentados os principais aspectos da arquitetura do modelo RuleNet, que são: o algoritmo de propagação, ou seja, como o RuleNet classifica os exemplos do conjunto de treinamento e é utilizado tanto na fase de treinamento quanto de classificação, e o algoritmo de treinamento, o qual, é responsável por alterar os hiper-retângulos, descritos pelos pesos dos nós intermediários presentes na rede.

4.3.1 Algoritmo de Propagação

No sistema RuleNet a rede consiste de três camadas: camada de entrada, intermediária e de saída. A camada de entrada é completamente conectada à camada intermediária. Cada nó da camada intermediária está conectado a um único nó de saída através de uma conexão com peso constante, como mostrado na Figura 4.2.

Na propagação são adotados os procedimentos padrões do modelo

feedforward: as funções de soma, ativação e saída de todos os nós são calculadas, começando com os nós da camada de entrada e terminando com os nós da camada de saída. O algoritmo de propagação é baseado na estratégia o "vencedor_leva_todas" e cada neurônio intermediário H_k descreve um hiper-retângulo n-dimensional, no espaço de atributos definido pelo vetor de peso w^{H_k} e pelos parâmetros λ_L e λ_R, os quais definem a região de influência do hiper-retângulo representado pelo neurônio H_k. Basicamente, os neurônios de saída competem entre si e o neurônio que contém o exemplo x, é então ativado e determina a saída geral da rede.

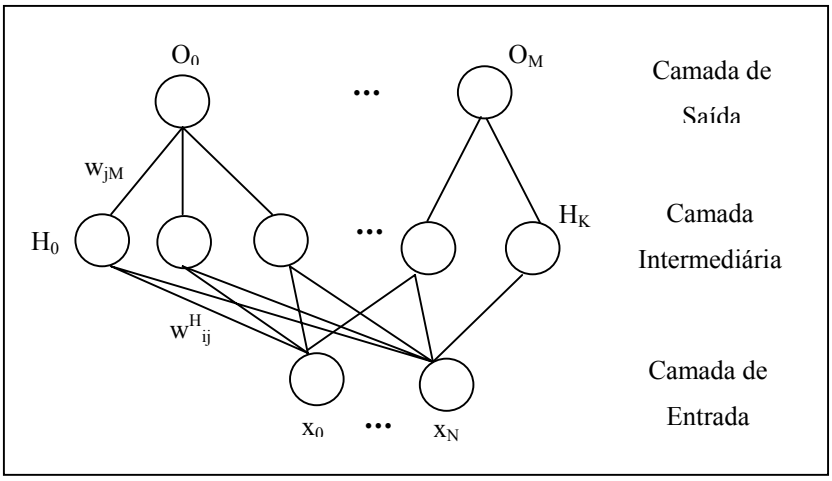


Figura 4.2: Arquitetura do RuleNet.

Pode acontecer de um exemplo x estar contido em mais do que um hiper-retângulo e, conseqüentemente, mais do que um nó intermediário pode estar ativo no mesmo passo de propagação. O algoritmo de treinamento, entretanto, garante que todos os nós ativos pertencem à mesma classe, i.e., estão conectados ao mesmo neurônio de saída. Durante a classificação, se um exemplo x não estiver em qualquer dos hiper-retângulos n-dimensionais, a rede responde com o resultado "desconhecido".

No RuleNet os nós de cada camada têm funções de soma, ativação e saída diferentes. Nos nós de entrada, a função identidade é usada como função de soma, de ativação e de saída. Isto significa que os nós de entrada têm apenas a função de "buferização". Cada nó de entrada representa um único atributo do domínio. Os nós intermediários são *perceptrons* modificados com

raio limitado. O significado geométrico desta modificação é a de que as regiões de influência, descrita pelo vetor de pesos, dos nós intermediários descrevem hiper-retângulos (Figura 4.3) ao invés de hiper-esferas, como tradicionalmente ocorre em *perceptrons*.

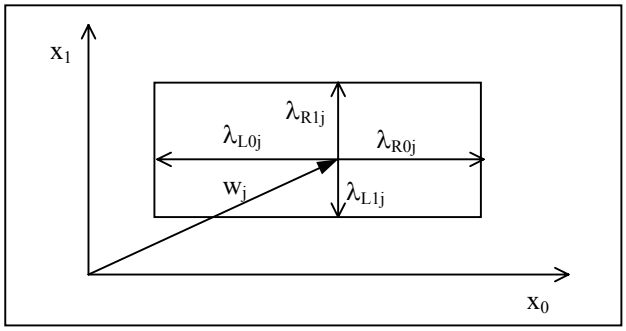
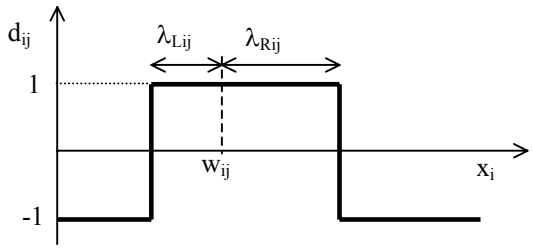


Figura 4.3: Interpretação geométrica dos parâmetros que definem as regiões de influência dos nós intermediários do RuleNet num espaço bidimensional.

No RuleNet a função de soma nos nós intermediários, que distingue se um vetor de entrada está na região de influência daquele nó ou não, é especificada por três parâmetros: um ponto de referência definindo a posição da região e dois parâmetros descrevendo a largura à esquerda e à direita do ponto de referência da região. Se o vetor de entrada estiver contido na região de influência do nó intermediário, então a função de soma devolve um valor positivo, caso contrário, devolve um valor negativo. A Figura 4.4 mostra este comportamento para uma das dimensões em questão.



i: 0…N, N = número de nós de entrada

j:0…M, M=número de nós intermediários

x_i: valor de entrada no nó de entrada i

w_j: ponto de referência da região de influência do nó intermediário j

λ_L: largura à esquerda da região de influência do nó intermediário

λ_R: largura à direita da região de influência do nó intermediário

Figura 4.4: Função de soma: $d_j = \min(d_{ij})$.

A Figura 4.5 apresenta um ponto de referência e sua região de influência num espaço tridimensional.

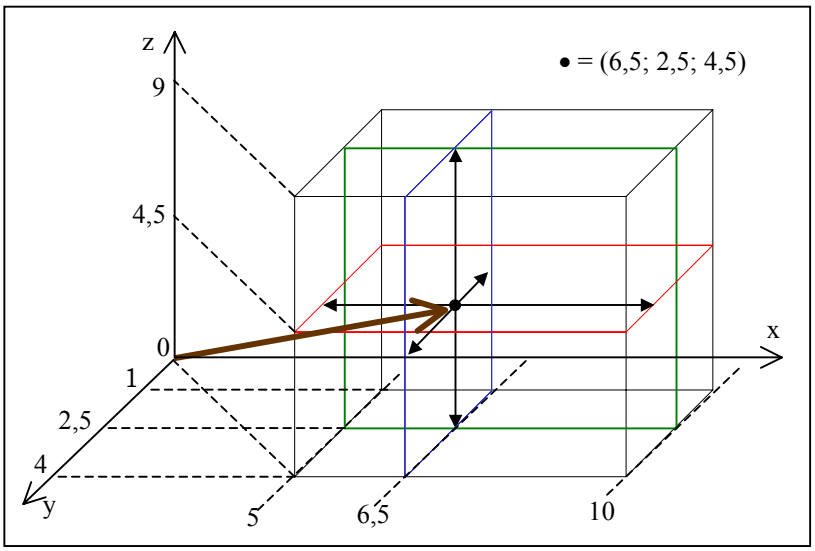


Figura 4.5: Interpretação geométrica e significado dos parâmetros que definem as regiões de influência de um nó intermediário (no caso o nó (6,5, 2,5, 4,5)) do RuleNet num espaço tridimensional.

No espaço tridimensional mostrado na Figura 4.5, note que o ponto de referência pode ser abordado como o ponto resultante da intersecção dos três planos, cada um deles paralelo a um dos eixos. O ponto de referência (6,5, 2,5, 4,5) define uma região de influência cujo contorno geométrico é um hiper-retângulo tridimensional.

Num espaço n-dimensional, o ponto de referência de cada nó intermediário j define a região de influência desse nó que, geometricamente, é modelada como um hiper-retângulo n-dimensional. Dizer que uma exemplo de treinamento está na região de influência de um nó é dizer que o exemplo está contido no hiper-retângulo que tal nó representa; conseqüentemente tal exemplo de treinamento está sujeito à função de soma (Figura 4.4) do nó em questão, a qual retorna 1 ou –1 dependendo do exemplo estar ou não na região de influência de tal nó, respectivamente.

Supondo que no exemplo da Figura 4.5 o nó intermediário representado seja um nó intermediário j e $e=(e_1,e_2,e_3)$ descreve um exemplo de treinamento, a definição da função de soma $d_j(e)$ é dada por: $d_j(e)=\min\{d_{xj}(e), d_{yj}(e), d_{zj}(e)\}$, onde as representações geométricas das funções d_{xj}, d_{yj}, d_{zj} são apresentadas na

Figura 4.6 (a), (b) e (c) respectivamente.

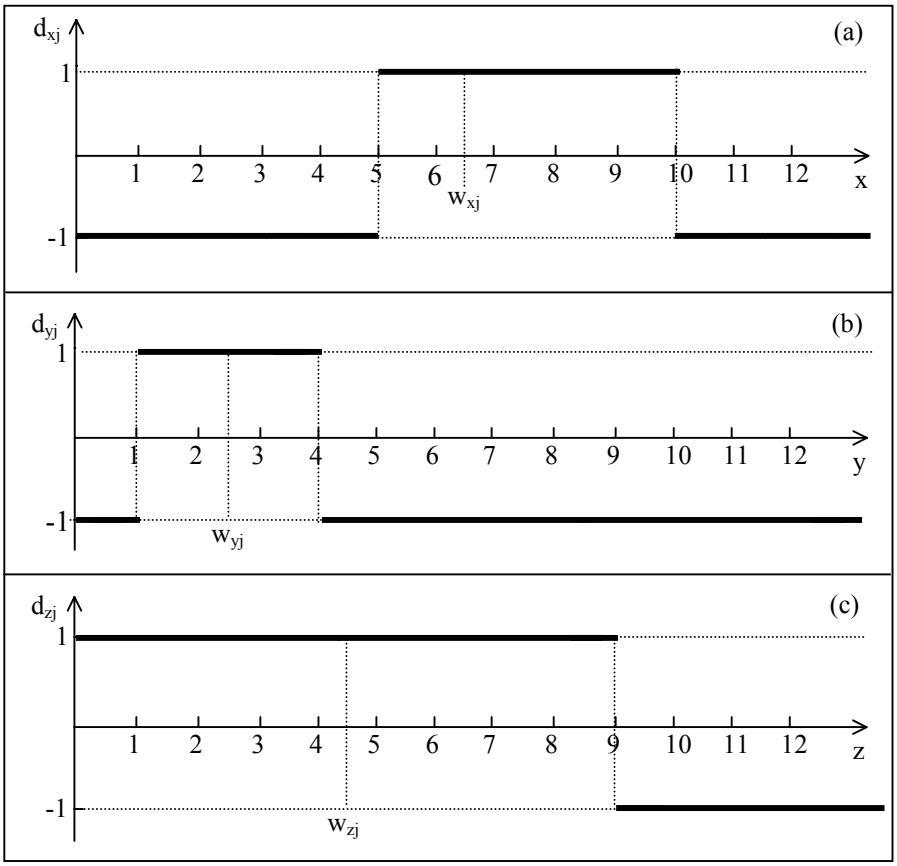


Figura 4.6: Interpretação geométrica da função de soma $d_j(e)=\min\{d_{xj}(e), d_{yj}(e), d_{zj}(e)\}$.

Considere um exemplo dado por e=(8, 2 ,7,5). Como $d_j(e) = \min\{d_{xj}(e),d_{yj}(e),d_{zj}(e)\}$, tem-se $d_{xj}(e) = d_{yj}(e) = d_{zj}(e) = 1$ e, portanto, o resultado da função de soma é $d_j^{(e)} = 1$, ou seja, o exemplo e está na região de influência de j. Por outro lado, se e=(11, 2, 7,5), tem-se $d_{xj}(e) = -1$, $d_{yj}(e) = 1$, e $d_{zj}(e) = 1$ e, portanto, a função de soma retorna o valor –1, configurando o fato do exemplo não estar na região de influência de j.

A função de ativação aplicada aos nós intermediários do RuleNet é a norma multiplicada pelo valor calculado pela função soma, descrita pela equação (4.3).

$$y_i = d_i * \left\| w_j - x \right\|_\infty = d_i * \max\left\{ \left| w_{ij} - x_i \right| \right\} \qquad (4.3)$$

Cada nó intermediário está conectado a apenas um nó de saída. A função saída é a função identidade. Assim, se o vetor estiver no hiper-retângulo representado por um neurônio intermediário, o correspondente nó intermediário é ativado e envia um sinal ao nó de saída a que está conectado. O sinal corresponde à distância entre o exemplo x e o ponto de referência, quando a entrada estiver dentro da região de influência do nó intermediário. Quando não, o sinal transmitido à camada de saída é a distância entre o exemplo x e o ponto de referência, negada.

A funcionalidade da camada de saída é baseada no princípio do "vencedor_leva_todas"' assim sendo, os nós de saída competem entre si e o valor de saída do nó vencedor é a saída da rede como um todo. O nó vencedor é calculado com a função de soma, correspondente à função-max, descrita pela equação (4.4) e, portanto, o nó de saída com valor de entrada máxima é definido como o nó saída vencedor. A função de ativação dos nós de saída é a função identidade e a função de saída atribui a cada nó de saída a classe correspondente.

$$d_j = \max\{(o_i\, w_{ij}),.....,(o_M\, w_{Mj})\} \qquad (4.4)$$

onde

- i: 0…M, M = número de nós intermediários,
- j: 0…N, N=número de nós de saída,
- d_j: valor da soma no nó de saída j,
- w_{ij}: peso da conexão entre o nó intermediário i e nó de saída j, e,
- o_i: valor de saída do nó intermediário i.

4.3.2 Formalização do Algoritmo de Propagação do RuleNet

O modelo RuleNet consiste num *perceptron* modificado de 3 camadas compondo uma rede neural *feedforward* (U,W,D,Y,O,x) com as seguintes especificações:

1. $U = \bigcup_{i\in\{1,2,3\}} U_i$ é um conjunto não vazio de unidades, onde U_1 é a camada de entrada, U_2 é a camada intermediária e U_3 é a camada se saída. $\forall i, j \in \{1,2,3\}, U_i \neq \varnothing$ e $U_i \cap U_j = \varnothing$ se $i \neq j$.

2. A estrutura da rede (conexões) é definida como:

   2.1 $W : U \times U \rightarrow R^3$ para conexões W(u,v), onde u ∈ U_1 e v ∈ U_2. Além disso, W(u,v)=[λ_L,w,λ_R] faz referência às regiões de influência de v.

   2.2 $W : U \times U \rightarrow R$ para conexões W(u,v) ≠ ∅ , onde u ∈ U_2, v ∈ U_3 e $W(u, v')= \varnothing, \forall v' \in U_3 - \{v\}$.

3. x: $U_1 \rightarrow R$ define para cada unidade de entrada u ∈ U_1 uma entrada externa referente ao vetor de entrada.

4. Para as unidades de entrada u ∈ U_1 existe D, Y, O, tal que:

   4.1 D define uma função de soma $D_u : R \rightarrow R$ que calcula a soma d_u, dada por

   $$d_u = D_u(x_u) = x_u.$$

   4.2 Y define uma função de ativação $Y_u : R \rightarrow R$ que calcula a ativação y_u, dada por:

   $$y_u = Y_u(d_u) = x_u.$$

   4.3 O define uma função de saída $O_u : R \rightarrow R$, que calcula a saída o_u, dada por:

   $$o_u = O_u(y_u) = x_u.$$

5. Para as unidades intermediárias u ∈ U_2 existe D, Y, O, tal que:

   5.1 D define uma função de soma $D_u : R \rightarrow R$ que calcula a soma d_u, dada por:

   $d_u = D_u(h_u) = \min(h_u)$, onde $H : R \rightarrow \{-1,1\}^n$ é dado por

   $$h_u = f(W(v,u), o_v) = \begin{cases} 1 & \text{se } (w - \lambda_L) \le o_v \le (w + \lambda_R) \\ -1 & \text{caso contrário} \end{cases}, \forall v \in U_1$$

   5.2 Y define uma função de ativação $Y_u : R \rightarrow R$ que calcula a ativação y_u, dada por:

   $$y_u = Y_u(d_u) = d_u * \|w - o_v\|_\infty, \ \forall v \in U_1 .$$

   5.3 O define uma função de saída $O_u : R \rightarrow R$, que calcula a saída o_u, dada por:

   $$o_u = O_u(y_u) = y_u.$$

6. Para as unidades de saída u ∈ U_2 existe D, Y, O, tal que:

6.1 D define uma função de soma $D_u : R \rightarrow R$ que calcula a soma d_u, dada por:

$d_u = D_u(t_u) = \max(t_u)$, onde $H : R \rightarrow R^n$ é dado por

$$t_u = f\left(W(v,u), o_v\right) = \left(o_v w\right), \forall v \in U_2 .$$

6.2Y define uma função de ativação $Y_u : R \rightarrow R$ que calcula a ativação y_u, dada por:

$$y_u = Y_u(d_u) = d_u.$$

6.3 O define uma função de saída $O_u : R \rightarrow STRING$, que calcula a saída o_u, dada por:

$$o_u = O_u(y_u) = \begin{cases} \text{CLASSE} & \text{se} \quad y_u \geq 0 \\ \text{'S/A'} & \text{caso contrário} \end{cases}.$$

onde CLASSE é o nome da classe associada à unidade de saída $u \in U_3$ em questão, e 'S/A' é a saída das unidades $u \in U_3$ que não foram ativadas. Apenas uma unidade se saída u pode ser ativada numa única propagação. Caso todas as unidades $u \in U_3$ apresentem 'S/A' como saída então o exemplo x é considerado "DESCONHECIDO".

4.3.3 Algoritmo de Treinamento

Inicialmente, a rede RuleNet consiste apenas dos nós de entrada que correspondem aos atributos que descrevem o domínio; na inicialização a rede RuleNet não contém ainda qualquer nó intermediário ou de saída. Durante o treinamento, nós intermediários e de saída são adicionados à rede pelo algoritmo de aprendizado. Cada exemplo de treinamento consistindo de um vetor de entrada x (que é uma seqüência de valores de atributos e uma classe C associada) é "propagado" pela rede. Durante a fase de treinamento, dado um exemplo de treinamento x, três situações podem ocorrer, dependendo da resposta da rede:

1) A saída da rede é C, i.e. a rede classifica x corretamente; neste caso a rede não sofre qualquer alteração.

2) A rede não classifica x o que implica x não estar contido em qualquer dos hiper-retângulos representados pelos nós intermediários. Neste caso um novo nó intermediário é criado e seu peso (i.e., ponto de referência) estabelecido como x. Os parâmetros λ_L e λ_R são definidos pelas regiões de decisão aproximadas das outras classes e por $\lambda_{Default}$. O limite $\lambda_{Default}$ é definido pelo escopo do valor dos atributos, sendo grande o suficiente para permitir que o nó intermediário em questão cubra todo o espaço de atributos, desde que a região de influência de tal nó não sobreponha regiões de influência de outros nós pertencentes a outras classes do domínio. A justificativa para essa definição se deve ao fato que, durante o treinamento, os valores de λ sempre diminuem e nunca aumentam. Se já existe um nó de saída associado à classe C, o novo nó intermediário criado é conectado a esse nó de saída. Se a classe C aparece pela primeira vez no conjunto de treinamento, um novo nó de saída rotulado C é criado e conectado ao novo nó intermediário.

3) A rede classifica x incorretamente, i.e. x está na região de decisão de algum(s) nó(s) intermediário(s), pertencente(s) a uma classe diferente de C. Neste caso, os λs dos nós intermediários "vencedores" são ajustados numa dimensão de maneira a remover x de suas regiões de influência. Feito isso, um novo nó é criado como no caso anterior.

Pseudocódigo do Algoritmo de Treinamento

Considere uma rede RuleNet com n unidade de entrada $u_1,\ldots,u_n \in U_1$, $U_2 = \varnothing$ e $U_3 = \varnothing$, um número de épocas e uma tarefa de aprendizado representada pelo conjunto $S=\{s_1,s_2,\ldots,s_k\}$, onde $\forall\ s \in S$, $s=\{x_1,x_2,\ldots,x_n,c\}$ é um par vetor de atributos p e sua respectiva classe c, o pseudocódigo do algoritmo de treinamento do RuleNet pode ser visto na Tabela 4.1. No Apêndice A deste trabalho é possível encontrar um pseudocódigo referente ao algoritmo de treinamento detalhado do RuleNet.

Tabela 4.1: Pseudocódigo do algoritmo de treinamento utilizado pelo RuleNet.

1) Enquanto número de épocas não for alcançado faça
2) $\forall$ $s_i \in S$ faça
   - 2.1) Propague s_i na rede RuleNet e armazene a saída da rede em c
   - 2.2) Se c = 'DESCONHECIDO'
     - 2.2.1) Cria unidade intermediária $v \in U_2$
     - 2.2.2) $\forall$ $x \in s_i$ e sua unidade de entrada referente $u \in U_1$ crie uma conexão $W(u,v) = \{\lambda_{L\_default}, w, \lambda_{R\_default}\}$ onde w=x.
     - 2.2.3) Ajuste as regiões de influência da unidade intermediária $v \in U_2$, de forma que v não sobreponha nenhuma outra unidade intermediária de U_2 que represente uma classe diferente.
     - 2.2.4) Se a unidade de saída $o \in U_3$ associada à classe de s_i existir, então conecte W(v,o) = 4, caso contrário crie unidade de saída $o \in U_3$ associe o à classe s_i e conecte W(v,o) = 4.
   - 2.3) Se c $\neq$ classe de s_i
     - 2.3.1) Ajuste os λ's de todos as unidades intermediárias $v \in U_2$ cuja saída foi > 0.
     - 2.3.2) Repita os passos 2.2.1 até 2.2.4.
3) Incremente o número de épocas e retorne ao passo 1.

4.3.4 Exemplo Utilizando o RuleNet

Um exemplo do funcionamento do RuleNet, no aprendizado da função booleana AND, é mostrado nas próximas figuras (Figura 4.7 a Figura 4.10). Na Figura 4.7 o exemplo (1,1,1) é usado como exemplo de treinamento. Como a rede está vazia, o primeiro nó intermediário é criado, de maneira que a rede possa aprender esse primeiro exemplo. O ponto de referência no nó intermediário corresponde aos valores dos atributos de entrada (1,1) e seu tamanho é definido com o valor *default* de λ grande o suficiente para cobrir todo o espaço de atributo.

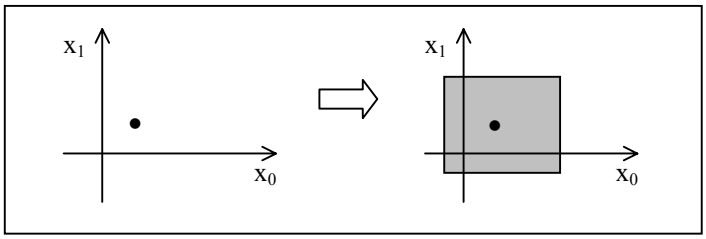


Figura 4.7: Início de treinamento com o exemplo (1,1,1).

O próximo exemplo de treinamento (1,0,0) está contido no retângulo que representa a classe 1 (diagrama à esquerda da Figura 4.8) e, portanto, esse exemplo é classificado incorretamente pela rede. Neste caso, primeiro a região de influência do nó que está fazendo a classificação incorreta é ajustada em uma dimensão (como será visto na próxima seção) e um novo neurônio é criado para representar a nova classe (Figura 4.8 diagrama à direita). É importante notar que não existe sobreposição dos nós que representam classes diferentes.

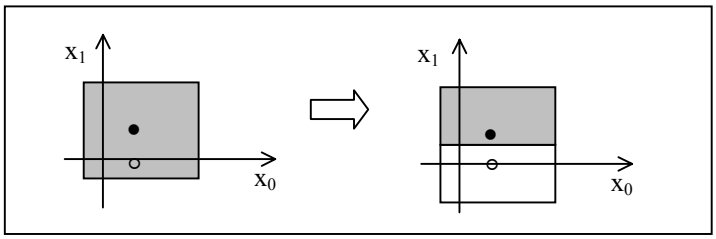


Figura 4.8: Continuação do treinamento, com o exemplo (1,0,0).

O terceiro exemplo é (0,0,0) e é classificado corretamente pela rede. Neste caso, o algoritmo básico de treinamento não faz qualquer mudança na rede (parte superior da Figura 4.9). Por outro lado, o algoritmo estendido de treinamento (descrito posteriormente), move o ponto de referência do nó intermediário na direção do novo exemplo (parte inferior da mesma figura).

O último exemplo que define a função AND é o (0,1,0) (Figura 4.10) que é classificado incorretamente. O algoritmo de aprendizado ajusta o nó que fez a classificação errada em uma dimensão e gera um novo nó intermediário, de maneira semelhante à situação apresentada na Figura 4.8.

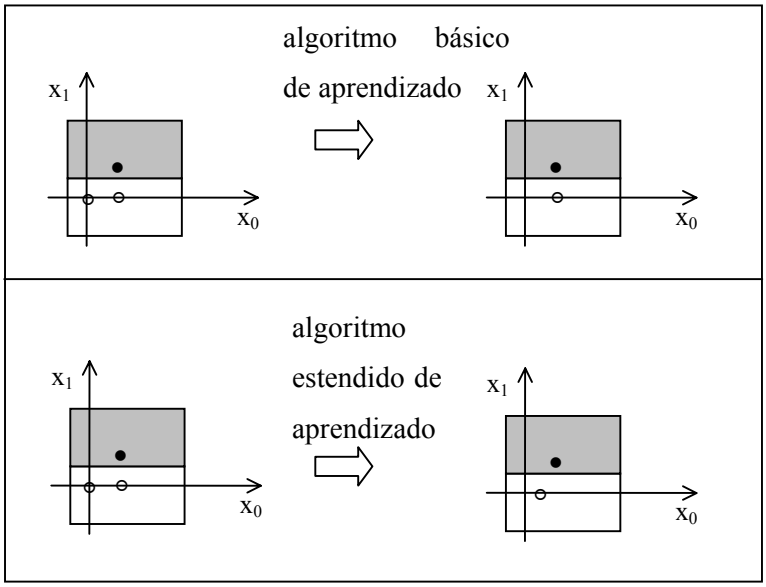


Figura 4.9: Continuação do treinamento, com o exemplo (0,0,0), considerando os algoritmos de aprendizado básico e estendido.

Se a entrada x é classificada incorretamente, ou seja, se x estiver na região de decisão de um nó intermediário associado a uma classe diferente daquela de x, um dos parâmetros λ tem que ser redimensionado. Teoricamente existem duas possibilidades, que estão mostradas na Figura 4.11. No RuleNet, o método correspondente ao diagrama à esquerda na figura é aplicado, onde $d_1 > d_2$. A razão para esta seleção é a maior probabilidade de uma generalização errada para pontos distantes do ponto de referência.

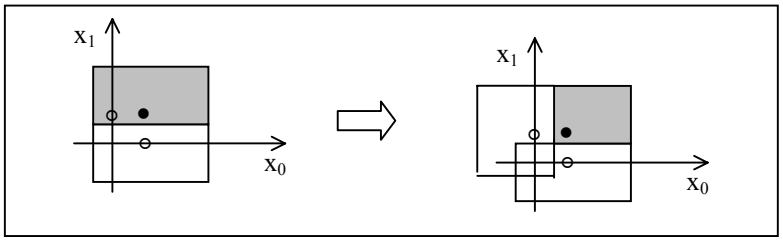


Figura 4.10: Continuação do treinamento, com o exemplo (1,0,0).

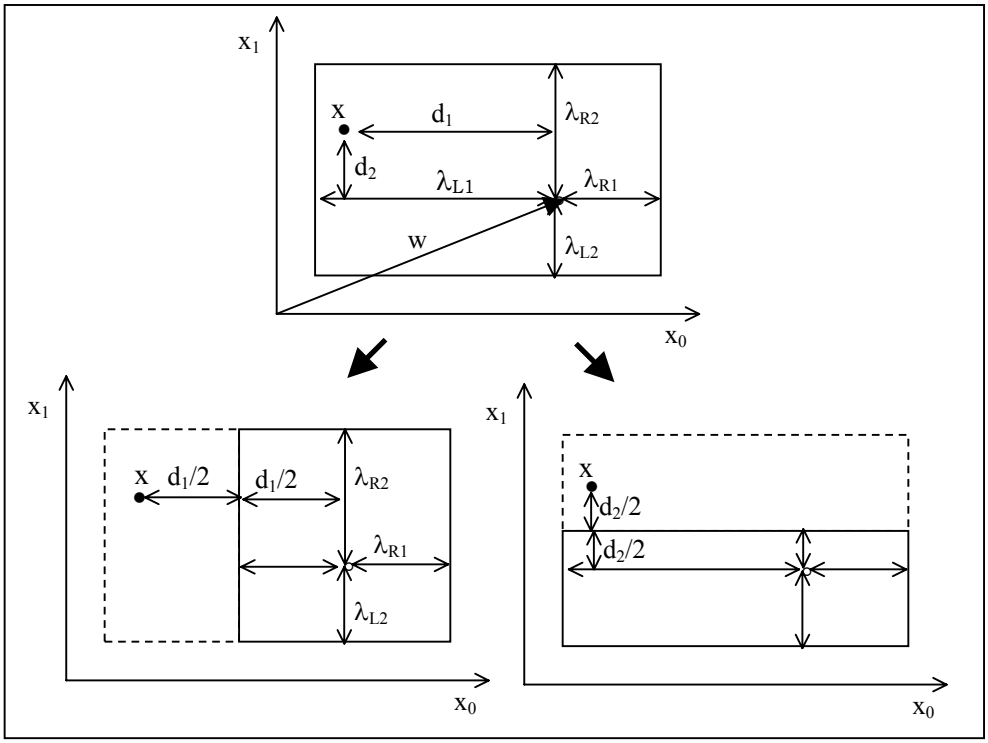


Figura 4.11: Redimensionando o parâmetro λ.

Dependendo da ordem com que os exemplos de treinamento são apresentados, o número de épocas necessárias para o aprendizado do conceito pode variar. A Figura 4.12 e a Figura 4.13 mostram como a ordem de apresentação dos exemplos pode interferir no aprendizado de um conceito, no caso a função booleana XOR. O tempo de convergência do algoritmo, o tamanho da rede e a qualidade da classificação dependem da ordem de apresentação dos exemplos de treinamento.

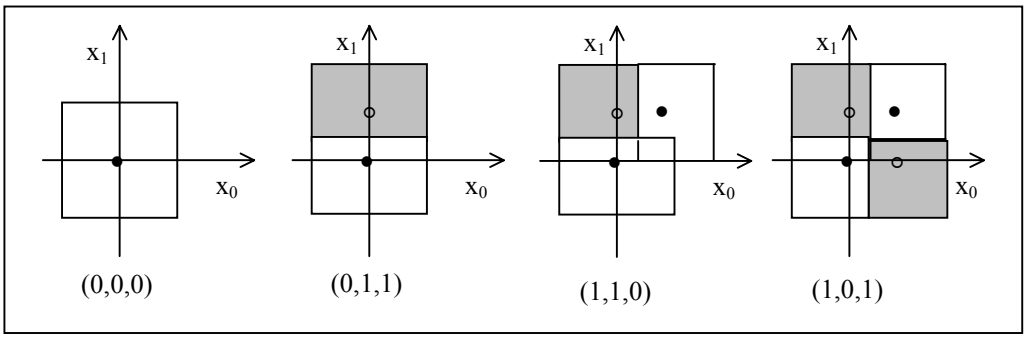


Figura 4.12: Aprendendo a função XOR em apenas 1 epoch, via apresentação dos exemplos na ordem apropriada i.e., exemplos pertencentes a classes diferentes são apresentados sucessivamente.

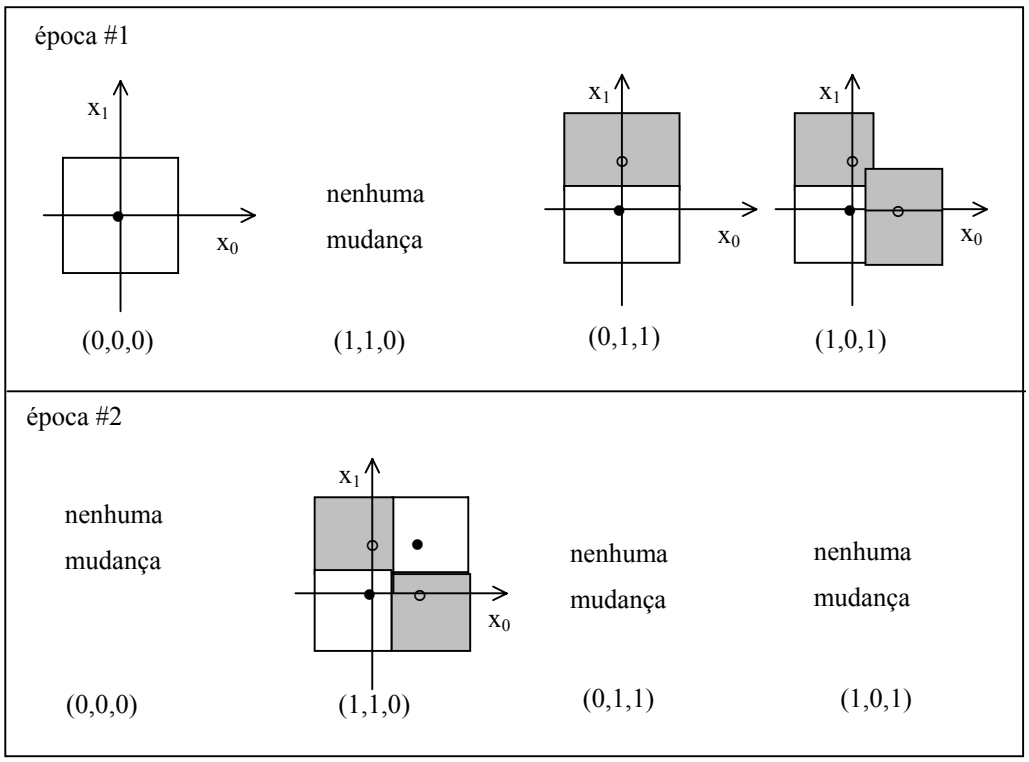


Figura 4.13: Aprendendo a função XOR em 2 *epochs*.

4.4 XRuleNet – A Versão Estendida do RuleNet

A versão estendida do RuleNet, denominada XRuleNet, é o próprio RuleNet com duas alterações. A primeira delas diz respeito à classificação de um exemplo x que não está contido em qualquer dos hiper-retângulos que representam os nós intermediários, e é classificado pelo RuleNet "desconhecido". O XRuleNet modifica esse comportamento determinando, dentre as classes existentes, aquela mais plausível de ser a classe do exemplo x.

Para fazer isso o XRuleNet busca determinar a classe dos exemplos "mais próximos" de x e, para tanto, uma distância precisa ser definida (no contexto, distância representa "similaridade"). Existem duas possibilidades. A primeira delas é a de calcular a distância euclidiana (ou vetor norma-∞) entre x e os pontos de referências dos nós intermediários, como mostra a Figura 4.14 (distâncias r11 e r12). Isto corresponde a adotar os princípios do aprendizado nearest-neighbour. A segunda possibilidade é calcular a distância de x à fronteira de cada hiper-retângulo que limita a classe (Figura 4.14, distâncias

e11 e e12) e eleger como classe de x a classe que é representada pelo hiper-retângulo mais próximo de x.

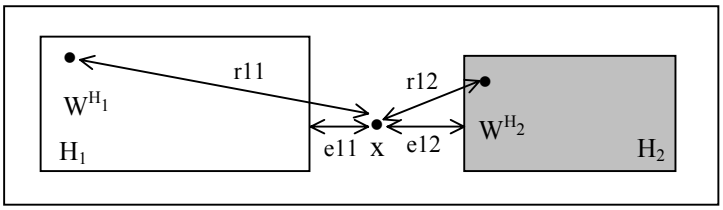


Figura 4.14: Exemplo de atuação do algoritmo de propagação estendido do RuleNet. Para o vetor de entrada x, o RuleNet ativa o neurônio intermediário H_1, e11 < e12. As distâncias r11 e r12 não são adotadas pelo XRuleNet.

No caso do RuleNet, a segunda métrica é usada pois, segundo [Tschichold-Gurman 1995b], os hiper-retângulos podem ser interpretados como sendo regiões onde a probabilidade de uma generalização correta é muito alta. Por essa razão é mais significativo o cálculo da distância entre a fronteira do hiper-retângulo e x do que o cálculo da distância entre o ponto de referência do hiper-retângulo e x. Essa alteração diz respeito apenas à fase de classificação dos exemplos; a estratégia de propagação, durante a o treinamento é igual à proposta do RuleNet.

Considere a situação mostrada na Figura 4.14. O algoritmo de propagação básico (RuleNet) retorna "desconhecido" para a classificação de x, porque x não pertence a qualquer das duas regiões de influência dos neurônios intermediários. No caso do algoritmo de propagação estendido (XRuleNet), entretanto, as distâncias e11 e e12 de x às fronteiras dos hiper-retângulos que representam os nós intermediários são calculadas e a classe representada pelo hiper-retângulo com a menor distância (e11) a x é eleita como classe de x.

Como comentado na seção anterior, a seleção dos pontos de referência dos nós intermediários é dependente da ordem de apresentação dos exemplos de treinamento. O exemplo responsável pela criação de um novo nó intermediário define o ponto de referência do nó criado. Esse fato faz com que tanto do tamanho da rede quanto sua precisão de classificação sejam dependentes da ordem dos exemplos de treinamento. Devido a esse problema, a segunda alteração implementada pelo XRuleNet viabiliza a modificação dos pesos (i.e. pontos de referência) durante a fase de treinamento. Cada vez que a rede produz um resultado correto (i.e., x está na região de decisão do nó

intermediário correto), os pesos dos nós "vencedores" são modificados usando a equação (4.5).

$$w_{new} = w + \varepsilon_t (x - w) \tag{4.5}$$

onde

- $\varepsilon_t = \exp\left(\frac{t}{T * P}\ln(\theta)\right)$, que possui os seguintes parâmetros:
  - θ : parâmetro da função geralmente é igual a 0.01,
  - t : época corrente,
  - T : número máximo de época, e
  - P : número total de exemplos de treinamento.

Os parâmetros λ devem ser ajustados de forma que a região de influência do nó em questão se mantenha inalterada, pois a idéia da alteração do ponto de referência é movê-lo para o centro de *cluster* dos exemplos contidos do hiper-retângulo sem modificar a região de influência do nó, pois qualquer modificação na região de influência afeta diretamente a classificação de exemplos.

Em comparação com a proposta inicial do RuleNet, a alteração do ponto de referência não afeta diretamente o algoritmo de propagação (na fase de classificação), uma vez que a classificação é feita em função dos limiares dos hiper-retângulos e, como o algoritmo de modificação do ponto de referência mantém constante a região de influência, não há impacto de tal modificação na classificação.

Apesar disso, o algoritmo de ajuste das regiões de influência dos hiper-retângulos leva em consideração alguns fatores e, dentre eles, a posição do ponto de referência no hiper-retângulo. Isto significa que existem diferenças no ajuste das regiões de influência produzida pelo método de ajuste usando a proposta inicial (RuleNet) e a proposta estendida (XRuleNet) com a estratégia de alteração dos pontos de referência.

Ao final do treinamento os nós gerados por ambas as propostas apresentam pontos de referência em posições diferentes no espaço, mesmo que a ordem de apresentação dos exemplos, durante o treinamento, tenha sido mantida. Tal influência na especialização afeta indiretamente o algoritmo de

propagação (fase de classificação), pois as regiões de influência dos nós são alteradas de maneiras diferentes em ambas as propostas e a classificação, como se sabe, é feita usando as regiões de influência.

4.5 Ajuste das Regiões de Influência

Como comentado anteriormente, se durante o treinamento a rede classificar incorretamente um determinado exemplo, é feito o redimensionamento da região de influência do(s) nó(s) responsável(eis) por tal classificação.

A idéia do redimensionamento é fazer com que um exemplo que tenha sido classificado incorretamente, deixe de fazer parte da região de influência do nó intermediário que colaborou para tal classificação. Isso é feito diminuindo alguma dimensão da região de influência do nó intermediário responsável. O procedimento de diminuição deve ser bastante cuidadoso, uma vez que com a diminuição da região de influência com o objetivo de deixar fora dela o exemplo classificado incorretamente, outros exemplos (classificados corretamente) podem também ficar fora.

As referências bibliográficas sobre o RuleNet não definem ou discutem maneiras de implementar o redimensionamento dos nós intermediários. Este trabalho propõe e implementa duas heurísticas de redimensionamento. As duas heurísticas diferem com relação à escolha da dimensão a ser redimensionada e ao próprio processo de redimensionamento.

A primeira tem como objetivo deixar o exemplo classificado incorretamente fora da região de influência reduzindo a região o mínimo possível. Para fazer isso é necessário, primeiro, determinar a distância do exemplo aos limites das regiões que definem cada uma das dimensões para, então, identificar a dimensão correspondente à menor distância e reduzi-la.

O procedimento que implementa essa heurística usa o parâmetro (α, $0<\alpha<1$) para representar o grau de "penalização" do nó que fez a classificação incorreta. Quanto maior o valor de α, maior a penalização do nó, traduzida por uma redução "mais drástica" da dimensão escolhida. Formalmente, tal procedimento pode ser descrito como:

- Dimensão associada à menor distância entre w e x dada por:

$$d = \min\left(\left|\lim(M)_1 - x_1\right|,\ldots,\left|\lim(M)_n - x_n\right|\right) \tag{4.6}$$

onde $\lim(M)_i$ (i=1,…n) denota o limite da região de influência de w considerando x dado por:

$$\lim(M)_i = \begin{cases} w_i + \lambda_{Ri} \text{ se } x_i > w_i \\ \lambda_{Li} - w_i \text{ se } x_i < w_i \end{cases}$$

Os parâmetros λ que definem o hiper-retângulo correspondente à dimensão d são então reajustados para:

$$\begin{aligned} \lambda_{Rd} &= (x_d - w_d) * (1 - \alpha) \text{ se } x_d > w_d \\ \lambda_{Ld} &= (w_d - x_d) * (1 - \alpha) \text{ se } x_d < w_d \end{aligned} \tag{4.7}$$

A Figura 4.15 mostra o ajuste da região de influência de um nó usando a primeira heurística de redimensionamento. O redimensionamento é mostrado para os valores α = 0,5 e α = 0,2. O valor de α determina a distância entre o exemplo classificado incorretamente e a fronteira da região de influência, após a redução. Quanto maior for o valor de α, maior será essa distância. Em geral, um valor de α próximo de 0 permite retirar o exemplo x da região de influência do nó minimizando o número de exemplos classificados corretamente que também serão retiradas, como um efeito colateral da redução.

A segunda heurística de redimensionamento tem como objetivo selecionar a dimensão com a maior distância entre o exemplo classificado incorretamente (x) e o ponto de referência do nó em questão (w). Tal heurística procura manter as fronteiras da região de influência do ponto de referência, o mais distante possível do ponto de referência. A idéia que subsidia essa heurística é a de que o ponto de referência de um nó (o qual é um exemplo da classe que o nó representa) delimita uma região com exemplos da mesma

classe. Para a implementação desta heurística o parâmetro α é também usado desempenhando idêntico papel.

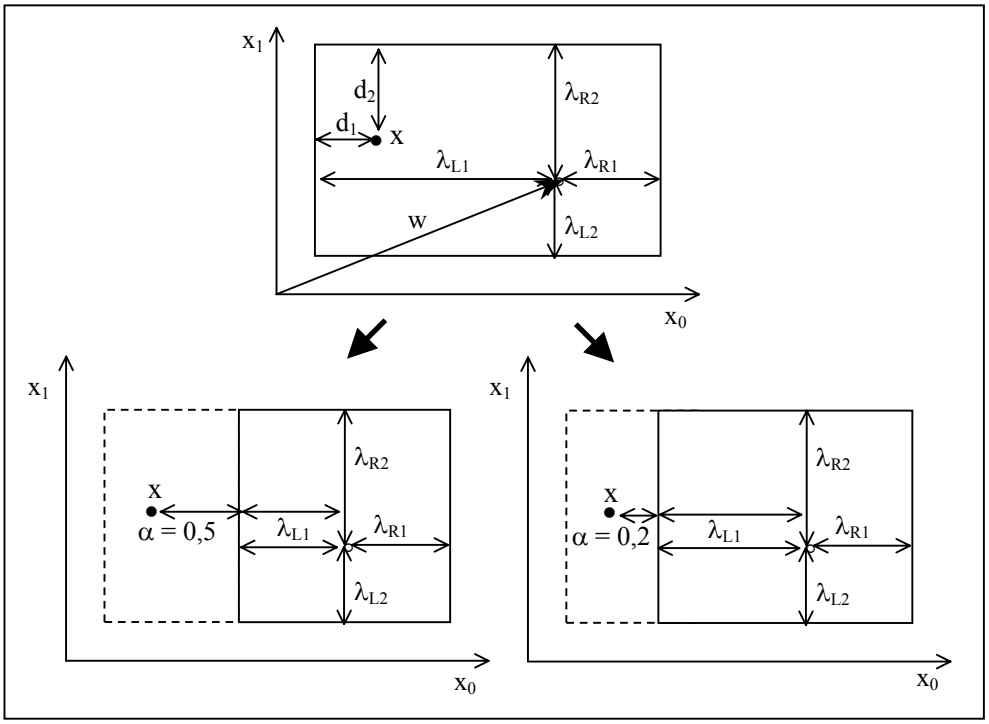


Figura 4.15: Representação gráfica do uso da primeira heurística no redimensionamento da região de influência de um nó intermediário.

Seja M o nó intermediário representado pelo vetor de referência n-dimensional w, responsável pela classificação incorreta de um exemplo x. Para o redimensionamento os seguintes valores devem ser calculados:

- Dimensão associada à maior distância entre w e x dada por:

$$d = \max\left(\left|w_1 - x_1\right|,\ldots,\left|w_n - x_n\right|\right) \tag{4.8}$$

Os parâmetros que definem o retângulo correspondente à dimensão d são então reajustados de acordo com a equação (4.7). A Figura 4.16 apresenta o ajuste da região de influência de um nó.

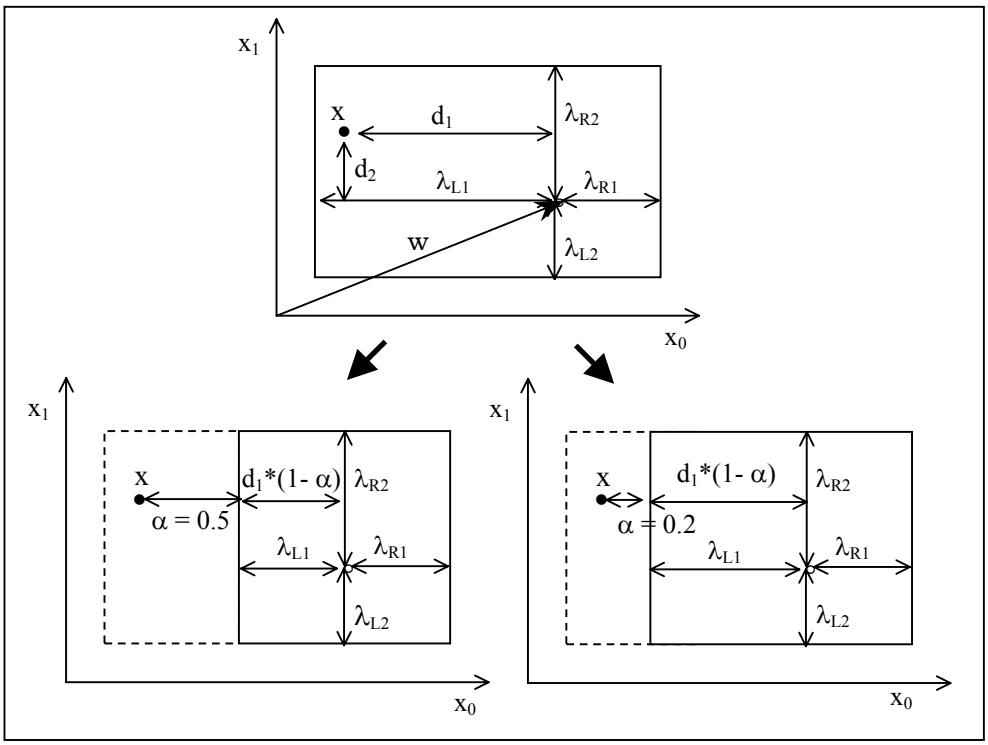


Figura 4.16: Representação gráfica do uso da segunda heurística no redimensionamento da região de influência de um nó intermediário.

As duas heurísticas de redimensionamento de nós intermediários podem ser usadas como uma forma de otimizar a indução de conceito usando exemplos de treinamento de um determinado domínio. A escolha da heurística mais indicada é fortemente determinada pelas características do domínio utilizado. O parâmetro α também é dado em função do domínio – seu valor pode ser calibrado com o objetivo de explorar uma melhor representação do conceito sendo aprendido. Tanto a definição da heurística quanto à escolha do valor para o parâmetro α podem ser encarados como problemas de otimização.

4.6 Representação do Conhecimento no RuleNet

O RuleNet usa uma representação local de conhecimento; a principal motivação para a escolha desta representação foi a de viabilizar a construção de uma rede que, uma vez pronta, pudesse ser "traduzida" para uma representação simbólica. Foi determinante na escolha da representação, também, prover condições para que a construção da rede pudesse ser feita a partir de conhecimento simbólico. Tal versatilidade torna o RuleNet um modelo bastante

conveniente para sistemas cooperativos. O conhecimento representado por uma rede RuleNet pode ser descrito como um conjunto de regras *if-then*: os nós intermediários caracterizam a parte condicional do *if* e o valor associado ao nó de saída ao qual o nó intermediário está conectado representa a conclusão *then*.

É também possível traduzir um conjunto de regras em uma rede RuleNet. A parte condicional da regra *if* define o tamanho do hiper-retângulo associado a um nó intermediário. O ponto de referência do nó é tomado como o centro de gravidade do hiper-retângulo. Os parâmetros λ_L e λ_R são derivados do tamanho do hiper-retângulo e do ponto de referência. A conclusão da regra define o nó de saída a ser conectado ao nó intermediário criado. Se a variação de um atributo estiver definida em apenas uma direção (e.g. diagrama à direita na), o correspondente valor λ é assumido ser $\lambda_{Default}$.

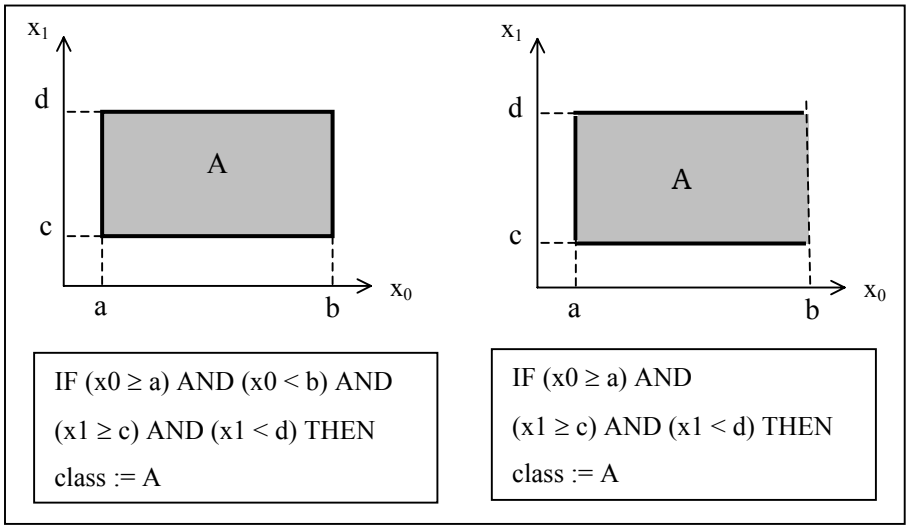


Figura 4.17: Correspondência entre uma regra e um nó intermediário do RuleNet.

Diferentemente dos sistemas KBANN e TREPAN, no RuleNet a "tradução" dos hiper-retângulos em regras não requer quaisquer tipos de suposições e/ou procedimentos simbólicos para tornar o conceito induzido inteligível, uma vez que o RuleNet possui a principal característica dos sistemas simbólicos como parte integrante e inerente do seu sistema, ou seja, a sua forma de representação de conhecimento como hiper-retângulos. Hiper-retângulos podem ser traduzidos em diretamente em regras, permitindo que o conhecimento possa ser representado simbolicamente. Isso já não acontece

com uma rede neural, cuja representação de conhecimento é a própria rede.

4.7 O RuleNet para Domínios Fuzzy

Esta subseção apresenta as principais idéias que subsidiam o sistema neuro-fuzzy denominado Fuzzy RuleNet, o qual é uma extensão do sistema RuleNet para domínios *fuzzy*.

O sistema Fuzzy RuleNet [Tschichold-Gurman 1995b], [Tschichold-Gurman 1995a] e [Tschichold-Gurman 1997] é um sistema neuro-fuzzy usado para a construção de sistemas *fuzzy*, principalmente sistemas que envolvam classificação *fuzzy*. Uma característica importante do Fuzzy RuleNet é a capacidade de ser inicializado com regras *fuzzy*. Conseqüentemente, ele pode ser empregado no aperfeiçoamento de sistemas *fuzzy*. A grande quantidade de conjuntos e regras *fuzzy* geradas, entretanto, pode fazer com a que compreensibilidade do sistema fique prejudicada.

No Fuzzy RuleNet são mantidas as características básicas presentes no RuleNet, como, por exemplo, a arquitetura composta por 3 camadas. Além disso, o algoritmo de treinamento do Fuzzy RuleNet e do RuleNet são similares. As principais diferenças entre estes dois sistemas podem ser vistas no princípio de classificação de exemplos, no vetor de pesos w, o qual define as regiões de influência para cada um dos nós intermediários, e na função de ativação dos nós de saída.

Em ambos os sistemas o conceito é representado por meio de hiper-retângulos definidos no espaço euclidiano n-dimensional. Entretanto, no RuleNet, hiper-retângulos (i.e. nós intermediários) pertencentes a classes diferentes não podem estar sobrepostos. Já no Fuzzy RuleNet, tal sobreposição é válida e representa a possibilidade de sobreposição entre os conjuntos *fuzzy* utilizados pelo sistema. O tamanho da sobreposição é definido pelo usuário do sistema (parâmetro Ω). A Figura 4.18 mostra a relação entre a sobreposição de dois hiper-retângulos, os quais representam os conceitos C1 e C2, respectivamente, e seus conjuntos *fuzzy*.

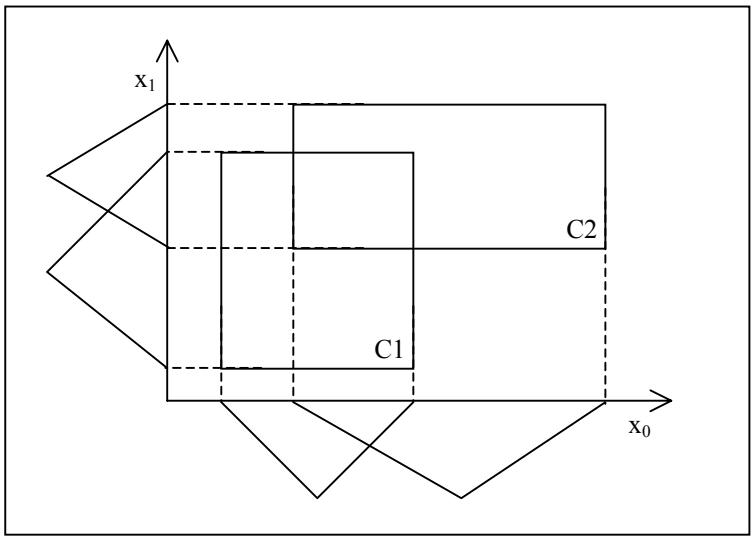


Figura 4.18: Relação entre hiper-retângulos e conjuntos fuzzy.

O vetor de pesos w, entre a camada de entrada e a camada intermediária, é composto por valores reais contínuos representando as regiões de influência do nó. No RuleNet, este vetor define a posição e o tamanho dos hiper-retângulos correspondentes as regiões de influência dos nós intermediários, e são utilizadas para determinar se o exemplo em questão está ou não contido nestas regiões de influência. No Fuzzy RuleNet este vetor representa as funções de pertinência do vetor de entrada juntamente com os valores que definem a região de influência das funções, ou seja, os pontos de referência dos hiper-retângulos representam os exemplares perfeitos das classes. Quanto mais próximo um vetor de entrada estiver de um ponto de referência, maior será o valor da função de pertinência associada à classe representada pelo hiper-retângulo. A Figura 4.19 apresenta três possíveis regiões, na qual, um vetor de entrada pode estar contido e seus valores de pertinência correspondentes a uma função triangular (Figura 4.19(a)) e trapezoidal (Figura 4.19(b)).

O uso deste tipo de fuzzificação implica uma relação direta entre as funções de pertinência de entrada e de saída. Assim, as funções da Figura 4.19 são definidas no intervalo variando de 0 à saída da classe desejada, ou seja, os conjuntos *fuzzy* utilizados no Fuzzy RuleNet não são normalizados como usualmente acontece em outros sistemas de classificação *fuzzy*.

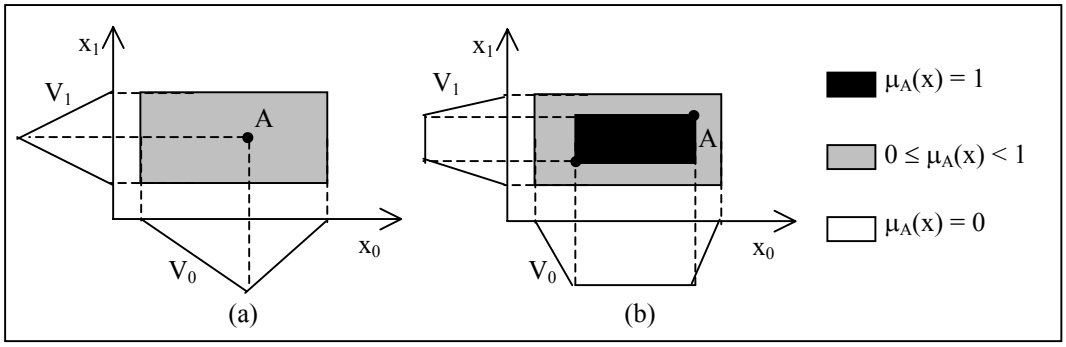


Figura 4.19: Funções de pertinência (a) triangular e (b) trapezoidal.

A função de ativação do sistema RuleNet apresenta sempre um valor discreto, geralmente definido pelo nome da classe associada ao nó de saída em questão. No Fuzzy RuleNet a função de saída pode ser discreta ou contínua de acordo com a aplicação em questão.

Outra questão a ser abordada é o resultado final apresentado pela rede, o qual, corresponde ao resultado de classificação de um exemplo em questão. No RuleNet, como não há sobreposição de hiper-retângulos de classes diferentes, apenas um nó de saída é ativado na classificação, o qual é tido como saída final da rede na classificação em questão.

Porém, no Fuzzy RuleNet é possível que um exemplo possa estar contido numa região coberta por hiper-retângulos de classes diferentes. Sendo assim, dois ou mais nós de saída serão ativados e, conseqüentemente, será necessário o uso de alguma estratégia para determinar a saída final da rede. Em alguns casos, a saída da rede pode ser definida pelo nó de saída que apresenta o maior valor de ativação, ou então, pode ser utilizado algo mais sofisticado como, por exemplo, a soma ponderada de todos os valores de ativação, i.e., controlador Sugeno. A estratégia utilizada para determinar a saída da rede pode ser escolhida com base no domínio de conhecimento em questão. As características restantes são as mesmas tanto para o RuleNet quanto para Fuzzy RuleNet, ou seja, as demais funções presentes nas unidades de entrada, intermediária e de saída permanecem inalteradas. A Figura 4.20 mostra uma rede Fuzzy RuleNet contendo dois nós intermediários, n_1 e n_2, ligados aos nós de saída c_1 e c_2. Esta rede é capaz de classificar exemplos que estão contidos no espaço bidimensional, em duas classes c_1 e c_2, respectivamente. Cada uma das

conexões entre os nós de entrada e os nós intermediários possui um conjunto *fuzzy* associado, determinados pelos parâmetros w, λ_L e λ_R, além da função de pertinência (neste caso foi utilizada a função triangular).

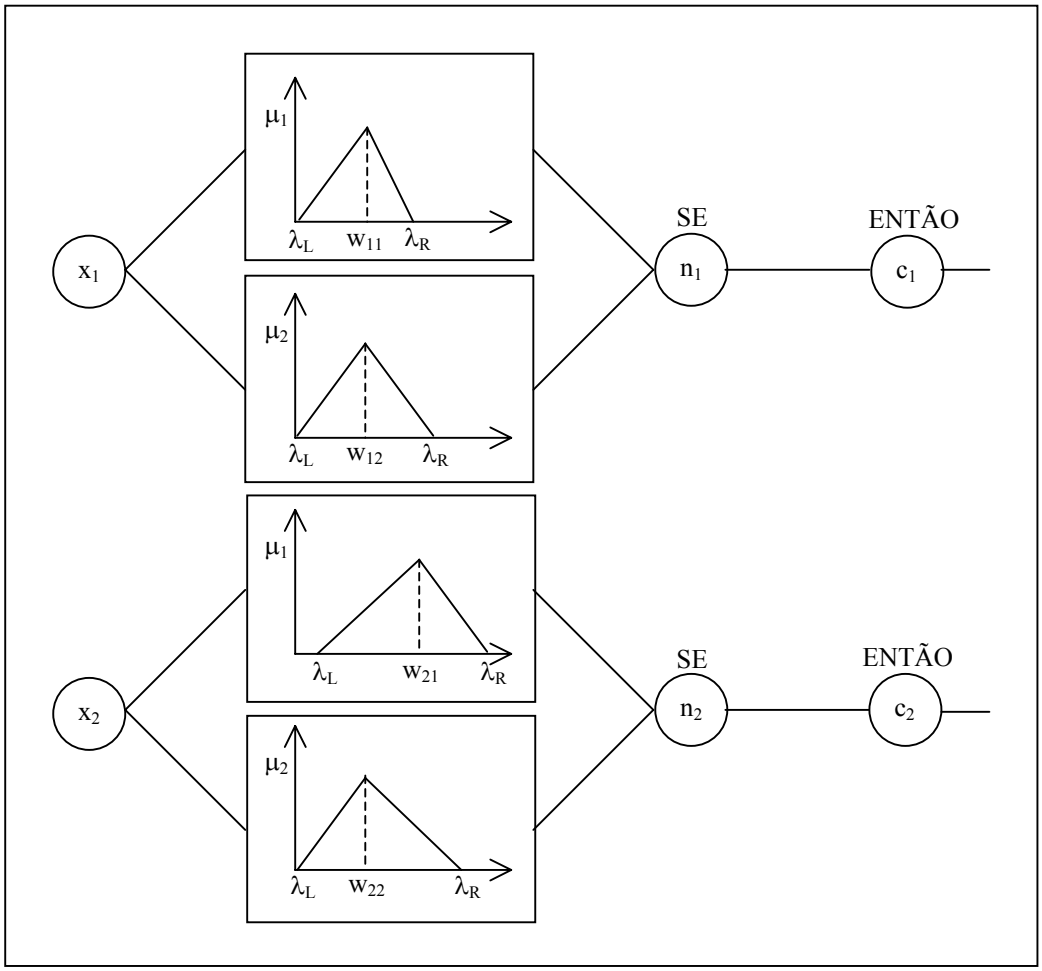


Figura 4.20: Arquitetura do Fuzzy RuleNet.

Outras modificações podem ser realizadas com o intuito de adaptar o Fuzzy RuleNet às características do conceito a ser induzido. Apesar das diferentes maneiras para classificação do exemplo em questão pela rede, três situações podem acontecer:

1. O vetor de entrada está contido na região de influência "coberta" por apenas um nó intermediário, ou todos os nós intermediários ativados estão conectados ao mesmo nó de saída, i.e., pertencem à mesma classe (caso *crisp*).
2. O vetor de entrada está contido na região de influência "coberta" por

vários nós intermediários pertencentes a diferentes classes (caso *fuzzy*).

3. O vetor de entrada está contido numa região que não é "coberta" por nenhum nó intermediário.

O caso *crisp* acontece quando o vetor de entrada é o ponto de referência (função de pertinência seja triangular), ou então quando o vetor de entrada está na região definida por dois pontos de referência (função de pertinência trapezoidal). É provável que o vetor de entrada seja semelhante a um exemplo aprendido anteriormente. Neste caso, a probabilidade que o Fuzzy RuleNet tem de fazer a classificação correta é alta. O caso *fuzzy* acontece quando o vetor de entrada está perto da fronteira da região de influência de vários nós, existindo, assim, a chance da saída da rede estar errada. Com relação ao último caso a rede pode apresentar saída "desconhecida", ou então, forçar a classificação de acordo com alguma métrica de distância (este procedimento é semelhante à classificação "forçada" do XRuleNet).

O algoritmo de treinamento do Fuzzy RuleNet é o mesmo do XRuleNet, incluindo, a equação (4.5) de ajuste do ponto de referência utilizado no caso 3 do algoritmo, i.e., quando a classificação é feita corretamente.

Uma das características mais importantes do Fuzzy RuleNet é a possibilidade de inicializar a rede com regras *fuzzy*. A idéia básica é definir os pontos de referência dos nós intermediários e o tamanho da sua região de influência utilizando a parte IF da regras *fuzzy* disponíveis. Assim, o vetor de pesos de cada nó intermediário é definido utilizando alguma função de pertinência. A parte THEN das regras é utilizada para a criação dos nós de saída e para conexão desse nós com seus respectivos nós intermediários. É importante lembrar que o Fuzzy RuleNet não leva em consideração a semântica dos conjuntos *fuzzy*, por exemplo, um conjunto *fuzzy* do atributo altura, denominado muito baixo, está contido nas regras r_1 e r_2, as quais, são traduzidas em hiper-retângulos H_1 e H_2. À medida que o treinamento acontece, as alterações feitas nestes hiper-retângulos podem fazer com o conjunto muito baixo associado ao hiper-retângulos H_1 seja diferente do conjunto muito baixo associado a H_2. Logo na tradução destes hiper-retângulos em regras, a semântica estará totalmente prejudicada.

A extração de regras e conjuntos *fuzzy* do Fuzzy RuleNet é possível, mas é difícil para o especialista humano entender tais regras porque o número de regras e de conjuntos apenas aumenta à medida que o treinamento acontece, ou seja, esse número pode ser grande o suficiente para prejudicar o entendimento do especialista.

4.8 Considerações Finais

Os principais tópicos abordados neste capítulo foram:

- abordagem do sistema RuleNet como tipo de rede neural capaz de refinar/construir regras na forma de hiper-retângulos no espaço n-dinensional, juntamente com todas as suas características discutidas em detalhes.
- o algoritmo de treinamento nas versões básica e estendida, assim como, a capacidade de classificação de exemplos através do uso da distância euclidiana. Além disso, duas propostas para ajuste das regiões de influência foram apresentadas.
- o sistema RuleNet para domínios *fuzzy* foi apresentado, que se caracteriza por permitir que regiões de influência de nós intermediários pertencentes a diferentes classes se sobreponham. Esta característica está associada ao uso da teoria dos conjuntos *fuzzy* nas regiões de influência.

O próximo capítulo apresenta os experimentos realizados com o sistema RuleNet, sua versão estendida, e sua versão para domínios *fuzzy*, discutindo os resultados obtidos na indução de conceitos em vários domínios de conhecimento.O desempenho apresentado pelo RuleNet é comparado ao desempenho do ID3 do NGE em vários domínios. Além disso, esquemas cooperativos ID3→RuleNet e NGE→RuleNet são apresentados e discutidos.

Capítulo 5

Experimentos e Resultados

5.1 Introdução

Este capítulo descreve os experimentos realizados durante o desenvolvimento deste trabalho de pesquisa e analisa/discute os resultados obtidos. O capítulo está organizado da seguinte maneira. Na Seção 5.2 são descritas as principais características dos domínios de conhecimento utilizados nos experimentos. A Seção 5.3 discute a técnica usada para a avaliação comparativa dos métodos e estabelece terminologia e *defaults* utilizados. A Seção 5.4 analisa comparativamente os resultados obtidos pelo RuleNet e XRuleNet. A Seção 5.5 analisa comparativamente os resultados obtidos pelo RuleNet e NGE. A Seção 5.6 analisa comparativamente os resultados obtidos pelo RuleNet e ID3. A Seção 5.7 analisa a cooperação NGE→RuleNet e ID3→RuleNet, a Seção 5.8 apresenta os resultados obtidos pelo Fuzzy RuleNet e, finalmente, a Seção 5.9 apresenta e discute resultados obtidos da comparação entre o Fuzzy NGE e o Fuzzy RuleNet.

5.2 Domínios de Conhecimento Utilizados nos Experimentos

Os experimentos realizados e descritos neste capítulo utilizaram dados de treze domínios de conhecimento. Três deles são domínios artificiais, oito foram

extraídos do UC-Irvine Repository [Blake e Merz 1998], um foi obtido junto ao Setor de Otoneurologia do Hospital das Clínicas da Faculdade de Medicina de Ribeirão Preto da Universidade de São Paulo (HCRP-FMRP-USP), e o último é um domínio farmacotécnico relacionado à fabricação de medicamentos.

- **Domínios A, B, C:** os domínios artificiais [Wettschereck e Dietterich 1995] foram construídos com o intuito de experimentar em domínio sem ruídos e com diferentes contornos entre as classes. Cada um desses domínios possui 500 instâncias divididas igualmente entre as classes. Os domínios A e B possuem duas classes e, em ambos, as classes (+ e –) indicam a posição dos exemplos positivos e negativos respectivamente, no espaço euclidiano (veja Figura 5.1). No domínio C as 500 instâncias estão divididas igualmente em dez classes; os números de 0 a 9 na Figura 5.1 representam a classe e suas posições no gráfico indicam onde se localizam as respectivas instâncias no plano cartesiano. A Figura 5.1 apresenta uma representação pictórica dos domínios artificiais com o intuito apenas de mostrar a divisão entre as classes; o número de exemplos é mostrado, em negrito, no canto de cada região que define as classes.

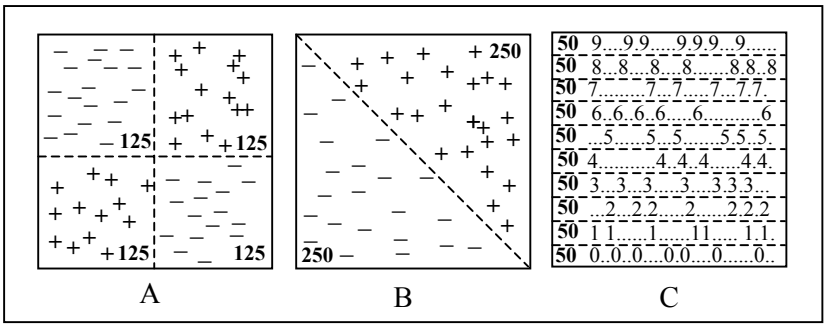


Figura 5.1: Os domínios artificiais A, B, e C cada um deles contendo 500 instâncias. Os domínios A e B estão divididos igualmente em duas classes distintas (+) e (–); C está dividido igualmente em dez classes (0 ... 9).

Os oito domínios extraídos do UC-Irvine Repository são descritos a seguir:

- **Íris:** este é um dos domínios mais conhecidos e utilizados em aprendizado indutivo de máquina [Fisher 1936] e [Duda e Hart 1974], razão pela qual seu uso neste trabalho se justifica. Este domínio possui três classes, cada uma delas representada por 50 instâncias de

treinamento, caracterizando um tipo de planta da espécie íris. A classe Íris-Setosa é lineramente separável das classes Íris-Versicolor e Íris-Virgínica, entretanto, estas duas últimas não são linearmente separáveis entre si. A Tabela 5.1 apresenta a freqüência das classes. Este domínio é descrito por quatro atributos numéricos e contínuos referentes a medidas (cm) feitas nas plantas pertencentes a cada uma das classes.

Tabela 5.1: Distribuição de Classes – Domínio Íris.

Classes	Freqüência	%
Íris-Setosa	50	33,3
Íris-Versicolor	50	33,3
Íris-Virgínica	50	33,3

- **Breast Câncer Wisconsin:** este domínio possui duas classes que representam tipos de câncer de mama (benigno ou malígno). Como 16 instâncias deste domínio possuem atributos ausentes, estas foram retiradas do conjunto de dados para os experimentos realizados neste trabalho. Cada uma das instâncias deste domínio possui dez atributos que descrevem características das células a partir de biópsias realizadas em 683 pacientes. A quantidade de instâncias deste domínio justifica seu uso neste trabalho, além disso este domínio é bastante conhecido. A Tabela 5.2 apresenta a freqüências das classes. Em [Mangasarian e Wolberg 1990] e [Bennett e Mangasarian 1992] podem ser encontradas informações adicionais.

Tabela 5.2: Distribuição de Classes – Domínio Breast Câncer.

Classes	Freqüência	%
Benigno	444	65,0
Maligno	239	35,0

- **Liver Disorders (Bupa):** as 345 instâncias desse domínio são compostas por 6 atributos, sendo que, 5 deles são índices de análises sanguíneas e um representa a quantidade média de álcool ingerida

por dia. Tais instâncias são divididas em duas classes, positiva (indica a presença de doenças no fígado), e negativa (indica a ausência de doenças no fígado). Uma das razões pelas quais este domínio foi escolhido, se deve a quantidade ruído presente nas suas instâncias. A Tabela 5.3 mostra as freqüências das classes.

Tabela 5.3: Distribuição de Classes – Domínio Liver Disorders.

Classes	Freqüência	%
Positivo	200	58,0
Negativo	145	42,0

- **E.coli:** este domínio, com 336 instâncias, está dividido em 8 classes que determinam a localização de sítios de seqüências de proteínas, os quais, são determinados pela seqüência de aminoácidos nas células, caracterizados por 7 atributos. A escolha deste domínio se justifica pela quantidade de classes presentes, e pela quantidade variada de instâncias que descreve cada classe. A Tabela 5.4 apresenta a distribuição de classes, note que, algumas classes possuem poucas instâncias. (Mais informações podem ser encontradas em [Horton e Nakai 1996] e [Horton e Nakai 1997]).

Tabela 5.4: Distribuição de Classes – Domínio E.coli.

Classes	Freqüência	%
cp	143	42,6
im	77	22,9
pp	52	15,5
imU	35	10,4
om	20	6,0
omL	5	1,5
imL	2	0,6
imS	2	0,6

- **Balance Scale:** este domínio foi gerado a partir de resultados experimentais de um modelo psicológico denominado *balance scale phenomena*. Cada exemplo é classificado de acordo com uma escala de

balanceamento, que pode ser direita, esquerda ou balanceado (a Tabela 5.5 mostra a distribuição das classes). Os 4 atributos existentes são baseados em medidas de peso e distância tanto para a direita, quanto para a esquerda. Mais detalhes podem ser vistos em [Shultz et al. 1994].

Tabela 5.5: Distribuição de Classes – Domínio Balance Scale.

Classes	Freqüência	%
Right	288	46,08
Left	288	46,08
Balanced	49	7,84

- **Ionosphere:** este conjunto se refere a leituras feitas por um sistema de radar de alta freqüência. A classificação pode ser "good", ou seja, o radar detectou algum tipo de estrutura na ionosfera ou a classificação pode ser "bad", ou seja, o radar não detectou a estrutura na ionosfera. A Tabela 5.6 mostra a distribuição de classe deste domínio descrito por 34 atributos.

Tabela 5.6: Distribuição de Classes – Domínio Ionosphere.

Classes	Freqüência	%
Good	225	64,10
Bad	126	35,90

- **Wine:** os exemplos deste domínio são descritos por 13 atributos numéricos que são resultados de análises químicas de diferentes vinhos; o conjunto de valores caracteriza a classe do exemplo, que corresponde à região de cultivo do vinho. Esse domínio representa instâncias pertencentes a três classes diferentes distribuídas na Tabela 5.7.

Tabela 5.7: Distribuição de Classes – Domínio Wine.

Classes	Freqüência	%
Região 1	59	33,14
Região 2	71	39,88
Região 3	48	26,97

- **Hayes Roth:** não foi encontrada uma descrição deste domínio nas referências consultadas. Entretanto, a distribuição das classes pode ser vista na Tabela 5.8.

Tabela 5.8: Distribuição de Classes – Domínio Hayes Roth.

Classes	Freqüência	%
Classe 1	51	38,63
Classe 2	51	38,63
Classe 3	30	22,72

- **Vestibular:** este domínio foi obtido junto ao Setor de Otoneurologia[1] do Hospital das Clínicas (HCRP) da Faculdade de Medicina de Ribeirão Preto (FMRP) da Universidade de São Paulo (USP). As instâncias são descritas por seis valores de atributos que são extraídos de um exame denominado sacádico fixo. Tal exame tem como objetivo medir os movimentos do globo ocular em tarefas de rastreio de alvos. Dependendo dos valores de atributos, a instância correspondente caracteriza uma situação de normalidade ou anormalidade, dividindo o conjunto de dados em duas classes (veja Tabela 5.9). (mais detalhes sobre esse domínio podem ser encontrados em [Volpini et al. 2002], [Figueira et al. 2003]. Três atributos são referentes à medidas feitas no olho esquerdo e, os outros três referentes ao olho direito, a saber:
    - **Latência:** intervalo de tempo de deslocamento do alvo e o início do movimento ocular para sua captura.
    - **Amplitude:** extensão do movimento ocular para atingir a posição final de localização exata do alvo. A amplitude

[1] Através do Prof. Dr. José Fernando Colafêmina.

determina a precisão da busca.

- **Acurácia:** precisão na localização do alvo.

Tabela 5.9: Distribuição de Classes – Domínio Vestibular.

Classes	Freqüência	%
Normal	98	49,0
Anormal	101	51,0

- **Excipientes:** este é um domínio de conhecimento farmacotécnico com 170 instâncias relacionadas à produção industrial de medicamentos e, consiste em 14 atributos que descrevem as características dos excipientes. Um excipiente é um composto, no qual a substância ativa presente no medicamento está embutida. As 170 instâncias estão divididas em 17 classes (10 instâncias por classe), cada classe representa um tipo de excipiente. Maiores detalhes podem ser encontrado em [Nicoletti et al. 2000]. A Tabela 5.10 mostra a distribuição de classes do domínio Excipientes.

Tabela 5.10: Distribuição de Classes – Domínio Excipientes.

Classes	Freqüência	%
Amido de Milho	10	5,88
Avicel PH 102	10	5,88
Avicel PH 200	10	5,88
Cellactose	10	5,88
Cloreto de Sódio	10	5,88
Emcompress	10	5,88
Lactose Monoidratada	10	5,88
Ludipress	10	5,88
Manitol Pó	10	5,88
Manitol Granular	10	5,88
Microcel – MC101	10	5,88
Microcel – MC102	10	5,88
Microcel – MC250	10	5,88
Microcel – MC301	10	5,88
Microcel – MC302	10	5,88
Vivapur type 101	10	5,88
Vivapur type 102	10	5,88

A Tabela 5.11 mostra um sumário dos domínios utilizados neste trabalho, as instâncias que apresentam atributos ausente ou não classificadas foram retiradas do conjunto que representa o domínio.

Tabela 5.11: Principais características dos domínios utilizados.

Domínio	Total de Instâncias no Domínio	Total de Instâncias Eliminadas	Total de Instâncias Treinamento	Total de Instâncias de Teste	Nro. Atributos	Nro. Classes	Número de Instâncias por Classe
A	500	0	450	50	2	2	250
B	500	0	450	50	2	2	250
C	500	0	450	50	2	2	250
Íris	150	0	120	30	4	3	50
Breast Câncer	699	16	615	68	10	2	444 (Benigno) 239 (Maligno)
Liver Disorders	345	0	310	35	6	2	200 (Positivos) 145 (Negativos)
E.coli	336	0	302	34	8	2	143 (cp) 77 (im) 52 (pp) 35 (imU) 20 (om) 5 (omL) 2 (imL) 2 (imS)
Balance Scale	625	0	562	63	4	3	288 (Right) 288 (Left) 49 (Balanced)
Ionosphere	351	0	316	35	14	1	225 (Good) 126 (Bad)
Wine	178	0	160	18	13	3	59 (Região 1) 71 (Região 2) 48 (Região 3)
Hayes Roth	132	0	119	13	5	3	51 (Classe 1) 51 (Classe 2) 30 (Classe 3)
Vestibular	199	0	159	40	6	2	98 (Normais) 101 (Anormais)
Excipientes	170	0	153	34	14	17	10

Os testes de comparação dos sistemas RuleNet, NGE, ID3 e Fuzzy RuleNet foram realizados utilizando oito desses domínios, a saber: A, B, C, Íris, Breast Câncer, Liver Disorders, E.coli e Vestibular. Os cinco domínios restantes foram escolhidos posteriormente para a condução de experimentos envolvendo as cooperações ID3→RuleNet e NGE→RuleNet, bem como para a comparação entre Fuzzy RuleNet e Fuzzy NGE.

É importante notar que todos os domínios possuem atributos contínuos, uma vez que o RuleNet, bem como sua versão *fuzzy*, trabalham apenas com atributos dessa natureza.

5.3 Metodologia e *Defaults* Empregados

Teste de hipóteses é usado com o intuito de avaliar a performance de algoritmos de aprendizado de máquina, mais especificamente avaliar se dois algoritmos de aprendizado distintos possuem performances significativamente diferentes.

Nos diversos experimentos apresentados neste capítulo serão testadas hipóteses de que o conceito induzido por um determinado algoritmo de aprendizado A possui diferenças significativas com relação ao conceito induzido por outro algoritmo de aprendizado B. Uma das diversas técnicas utilizadas para comparar algoritmos de aprendizado é a *k-fold cross-validated paired t-test* [Dietterich 1998] onde o conjunto de dados S é dividido em k conjuntos disjuntos de igual tamanho, $T_1,T_2,\ldots,T_k$, e são conduzidas k repetições. Em cada repetição o conjunto de teste é T_i é o conjunto de treinamento é a união de todos os T_j, onde $j \neq i$. Para cada repetição é armazenado a acurácia de A e de B, e ao final de k repetições é realizado o "teste-t pareado" [Sachs 1984], onde a média das diferenças nas medidas de acurácia para os algoritmos é calculada de acordo com a equação:

$$\overline{\text{diff}} = \frac{1}{k}\sum_{i=0}^{k+1}\left(\text{acuracia}_A^i - \text{acuracia}_B^i\right) \qquad (5.1)$$

onde a variável acuracia_A^i representa a taxa de acertos apresentada pelo algoritmo *A* na repetição i, e acuracia_B^i é a taxa de acertos apresentada pelo algoritmo *B* na mesma repetição i. O desvio padrão, denotado por s, também é computado, além disso, para que o teste seja válido é necessário que ambos os algoritmos sejam submetidos às mesmas partições da validação cruzada. Após cálculo das medidas acima é possível aplicar o "teste-*t* pareado" dado por:

$$t = \frac{\overline{\text{diff}}}{s/\sqrt{k}} \qquad (5.2)$$

Dois algoritmos de aprendizado apresentam diferenças significativas de acurácia com confiança em $100(1-\alpha)\%$, se:

$$|t| \geq t_{\alpha/2,k-1}$$

onde $t_{\alpha/2,k-1}$ define a região de rejeição do teste. Por exemplo, caso o número de repetições seja k=10 (*10-fold cross-validated paired t-test*), o valor de α=0,05 (95% de confiança), e $|t| \geq t_{0.975,9}$, onde $t_{0.975,9}$ = 2.23, então há diferença entre a acurácia dos algoritmos A e B.

Existem outros parâmetros de comparação além da acurácia, como, por exemplo, a quantidade de regras existentes ao final do treinamento, o tempo de duração do treinamento, entre outros.

Para a obtenção dos resultados apresentados nas próximas seções foram realizados, basicamente, os seguintes passos:

(a) Os procedimentos de indução do conceito a partir do conjunto de treinamento e sua avaliação num conjunto de teste foi repetido para dez pares (conjunto de treinamento/conjunto de teste), a fim de reduzir a variação estatística. Os dados apresentados como resultados indicam a média dos valores obtidos. Além disso, os resultados obtidos em cada uma das 10 repetições foram utilizados para realizar o teste-*t* pareado, formando assim o teste *10-fold cross-validated paired t-test* com o intuito de comparar os resultados dos sistemas entre si. É importante lembrar que todos os testes de hipóteses foram realizados com 95% de confiança.

(b) Os dez pares (conjunto de treinamento/conjunto de teste) são fixos para todos os experimentos.

(c) Além da média de acertos, denominada acurácia, outros dados foram coletados após cada experimento de acordo com o sistema de aprendizado em questão, são eles:

   a. Para o RuleNet e XRuleNet foi armazenado o número de nós existentes ao final do treinamento (que constituem a expressão induzida do conceito). Além disso, o número de épocas foi

computado, sendo que o número máximo é 10, se a rede convergir antes, então o treinamento é finalizado.

b. Para o NGE foi armazenado o número de hiper-retângulos existentes ao final do treinamento, os quais, são as regras que representam o conceito induzido.

c. Para o ID3, o número de folhas da árvore de decisão induzida foi contado, ou seja, o número de regras geradas pela indução do conceito foi armazenado.

(d) No caso do RuleNet (XRuleNet) e NGE foi necessário determinar empiricamente o valor do parâmetro α (parâmetro que determina o valor de ajuste da região de influência do nó intermediário) e o valor do número de *seeds*, respectivamente. No caso do RuleNet e XRuleNet o valor do parâmetro α variou entre 0,2 e 0,8 e, para cada domínio o valor que proporcionou a melhor acurácia foi computado. Já no caso do NGE, o número de *seeds* variou entre 3 e 25 e, para cada domínio o número de *seeds* que forneceu a melhor acurácia foi armazenado.

5.4 RuleNet *versus* XRuleNet

A Tabela 5.12 mostra os resultados obtidos com o uso do RuleNet e do XRuleNet em vários domínios de conhecimento. A coluna Sistema informa o sistema, e fornece o valor do parâmetro α (valor entre parênteses). A coluna Acurácia fornece a média percentual de acertos e seu respectivo desvio padrão. É importante observar que a acurácia do XRuleNet refere-se às instâncias que o mesmo classificou corretamente, ou seja, instâncias que estavam contidas nas regiões de influência que representam suas respectivas classes, bem como as instâncias que não estavam contidas nas suas respectivas regiões de influência, mas foram classificadas corretamente pelo algoritmo estendido (i.e., classificação "forçada").

A coluna Nós Intermediários informa o número de nós intermediários existentes ao final do treinamento; o número de nós intermediários pode ser entendido como o número de regras geradas durante o treinamento. A coluna Épocas indica a quantidade de épocas utilizadas no treinamento e, por fim a

coluna Convergência indica a taxa de convergência da rede em dez repetições de treinamento. Por exemplo, se a rede convergiu apenas duas vezes, então a taxa de convergência será de 20%.

Tabela 5.12: RuleNet *versus* XRuleNet, '♣' indica os resultados onde a acurácia apresentou diferenças significativas, igualmente, '♦' indica os resultados onde o número de nós ocultos apresentou diferenças significativas.

Domínio	Sistema	Acurácia(%)	Nós Intermediários	Épocas	Convergência (%)
A	RuleNet (0,8)	99,2 ± 1	35,5 ± 5,4 ♦	3 ± 0,5	100
	XRuleNet (0,8)	98,6 ± 2,1	45,5 ± 6,4	3,9 ± 2,2	90
B	RuleNet (0,6)	94,6 ± 2,8	54 ± 3,7 ♦	3,2 ± 0,4	100
	XRuleNet (0,6)	93,6 ± 2,5	235,3 ± 23,2	10 ± 0	0
C	RuleNet (0,8)	92,4 ± 4,2	118,5 ± 4,4 ♦	3 ± 0	100
	XRuleNet (0,8)	91,0 ± 3,7	140,9 ± 11,3	6,5 ± 3,7	50
Íris	RuleNet (0,2)	96,0 ± 3,4	16,1 ± 2,7 ♦	2,8 ± 0,4	100
	XRuleNet (0,2)	94,0 ± 4,9	43,4 ± 18,3	8,1 ± 3,1	30
Breast Cancer	RuleNet (0,6)	94,6 ± 3,2	54,7 ± 6,9 ♦	3 ± 0	100
	XRuleNet (0,6)	94,3 ± 2,4	119,7 ± 29,7	7,9 ± 2,7	40
Liver Disorders	RuleNet (0,8)	50,2 ± 9,7	195 ± 12,3 ♦	3,1 ± 0,3	100
	XRuleNet (0,8)	61,7 ± 7,0 ♣	240,5 ± 14,6	6,8 ± 3,4	50
E.coli	RuleNet (0,2)	70,5 ± 10,7	114,6 ± 5,5 ♦	3 ± 0	100
	XRuleNet (0,2)	69,0 ± 6,3	129,5 ± 6,6	3,2 ± 0,4	100
Vestibular	RuleNet (0,4)	88,2 ± 8,3	30,9 ± 1,3 ♦	3 ± 0	100
	XRuleNet (0,4)	87,3 ± 5,1	84,9 ± 22,5	9,3 ± 2,2	10

A Figura 5.2 apresenta um gráfico comparativo do desempenho do RuleNet versus XRuleNet considerando a acurácia do conceito induzido pelas redes, na classificação de instâncias do conjunto de testes. Apesar do RuleNet obter uma pequena vantagem na acurácia na maioria dos domínios, tal vantagem não foi estatisticamente significante, ou seja, ambos apresentaram, basicamente, o mesmo nível de acurácia, com exceção do domínio Liver Disorders em que o XRuleNet foi, significativamente, superior ao RuleNet. A razão disso se dá pelo fato do XRuleNet fazer uso da classificação "forçada" utilizando a distância Euclidiana para classificar instâncias "descobertas". Tal

fator pode ser observado na Tabela 5.13, onde a classificação "forçada" aumentou substancialmente a acurácia do XRuleNet no domínio Liver Disorders, apesar disso, o desempenho de ambos os sistemas neste domínio de conhecimento é precário. É importante notar que e o XRuleNet apresentou apenas 50% de convergência neste domínio, enquanto que o RuleNet convergiu em todas as repetições (100%), e utilizou menos épocas de treinamento (veja Tabela 5.12).

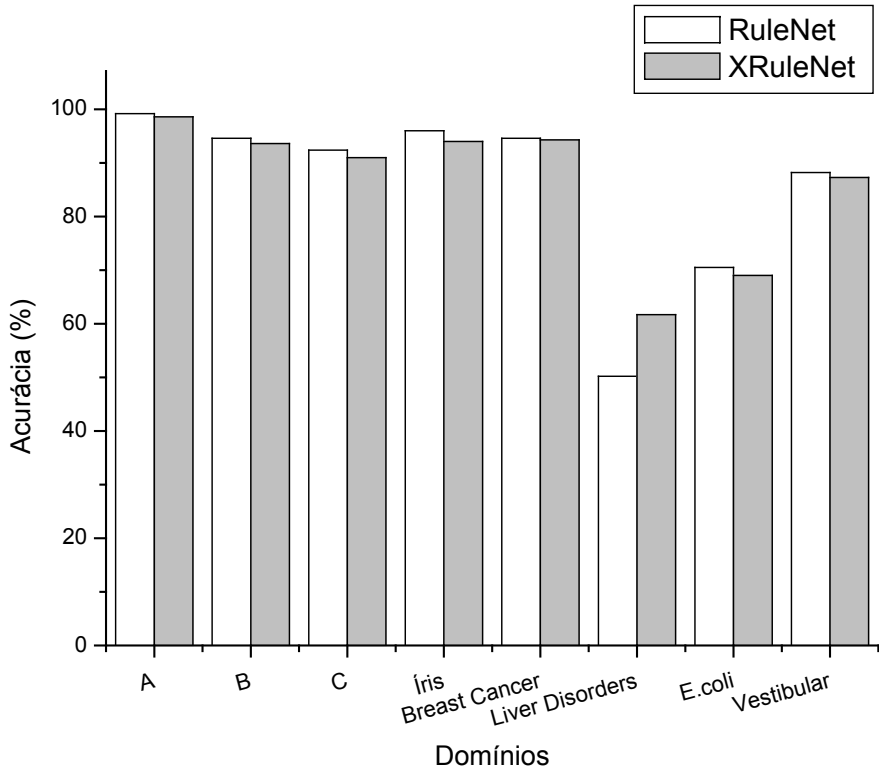


Figura 5.2: Gráfico de comparação de acurácia entre o RuleNet e o XRuleNet.

Tabela 5.13: Comparação da acurácia do RuleNet *versus* o XRuleNet no domínio Liver Disorders.

Sistema	Acurácia(%)	Acurácia Forçada(%)	Acurácia Total (%)
RuleNet (0,8)	50,2 ± 9,7	—	50,2 ± 9,7
XRuleNet (0,8)	42,1 ± 10,6	19,6 ± 10,2	61,7 ± 7

Outros dois fatores importantes a serem analisados são o número de nós intermediários e a quantidade de épocas usadas no treinamento. Na Figura 5.3 é possível observar que, para todos os domínio, a medida que o número de épocas aumenta, número de nós intermediários existentes também aumenta. Isso ocorre porque cada vez que o conjunto de treinamento é submetido à rede, as regiões de influência se ajustam e podem, eventualmente, "descobrir" instâncias antes "cobertas". Assim sendo, nas épocas subseqüentes, será necessária a criação de novos nós que possam "cobrir" essas instâncias "descobertas". Outra característica a considerar é que o RuleNet (e também o XRuleNet) sempre diminuem as regiões de influência dos nós, entretanto, isso não é padrão.

É possível observar na Figura 5.3 que o XRuleNet gerou um maior número de nós que o RuleNet como conseqüência do número de épocas usados. A Tabela 5.12 mostra que o XRuleNet com exceção dos domínios A e E.coli apresentou uma taxa de convergência muito baixa, ou seja, algumas vezes o número de máximo de épocas (nesse caso 10) não foi suficiente para o XRuleNet convergir e quando houve convergência o número de épocas utilizadas era relativamente maior que o número de épocas utilizadas pelo RuleNet para qualquer domínio de conhecimento.

Como XRuleNet apresentou um número bem maior de nós (regras) do que o RuleNet (veja Tabela 5.12 e Figura 5.3), e a acurácia apresentada por ambos em todos os domínios, com exceção do Liver Disorders, foi praticamente a mesma, isto faz com que o RuleNet seja indicado para estes domínios.

A razão principal da acurácia e do número de nós do XRuleNet serem diferentes do RuleNet é a aplicação da fórmula de movimentação dos pontos de referência durante o treinamento, uma vez que esta é a única modificação feita no algoritmo de treinamento do XRuleNet com relação ao RuleNet.

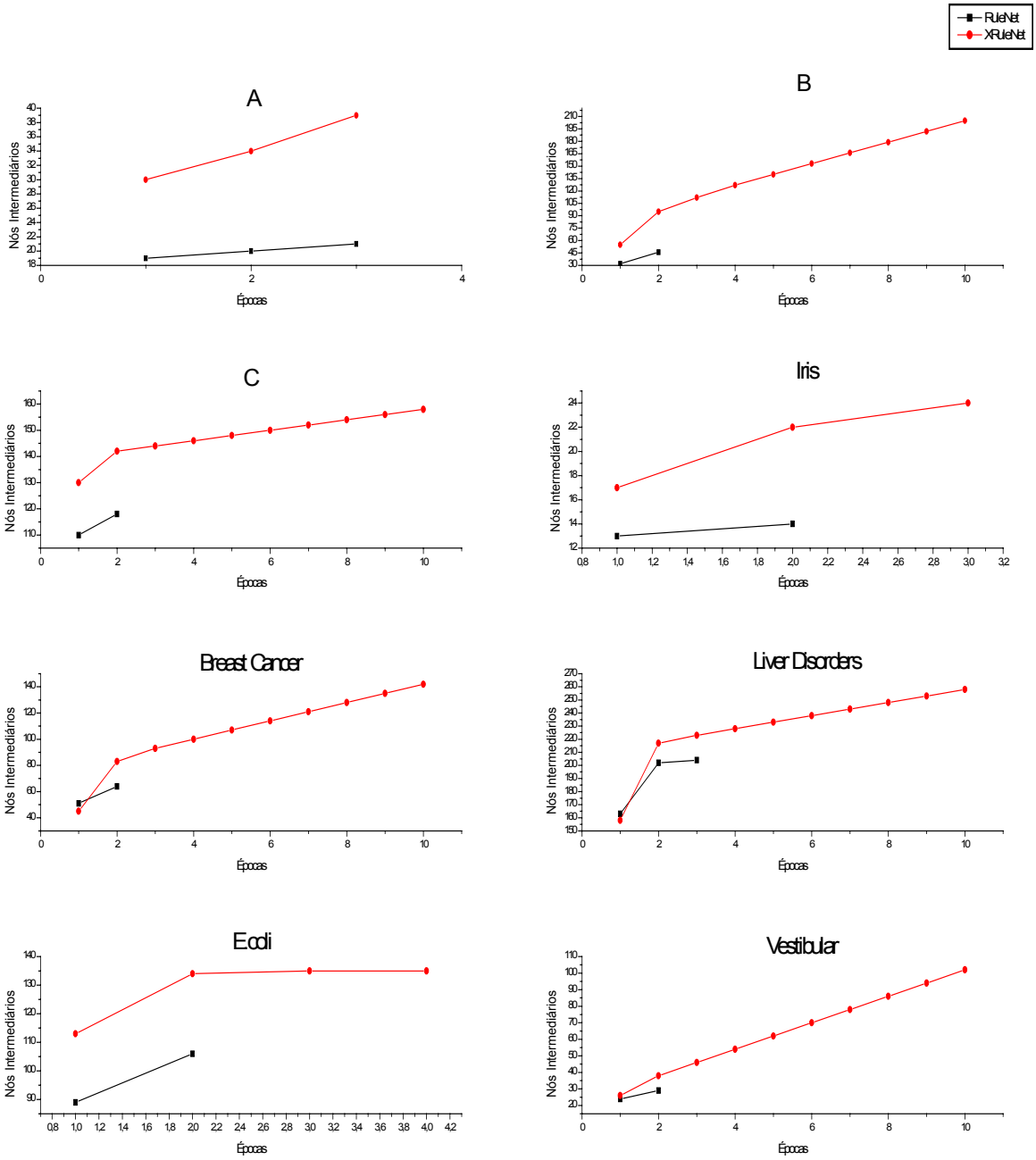


Figura 5.3: Gráfico de comparação entre RuleNet e o XRuleNet, com relação ao número de épocas por número de nós intermediários.

A Figura 5.4 mostra graficamente o desempenho do RuleNet nos domínios Íris e B para quatro valores do parâmetro α: 0,2, 0,4, 0,6 e 0,8 em diferentes processos de treinamento/teste. A alteração do parâmetro α não causou muito impacto nos resultados; tanto a média de acertos (Figura 5.4 (a)), quanto o número de nós intermediários (Figura 5.4 (b)), quanto ao número de

épocas (Figura 5.4 (c)), embora variem ligeiramente com a variação de α não apresentaram uma variação escalar grande. Métodos de otimização podem ser aplicados com o objetivo de encontrar o melhor valor para α; tal otimização, entretanto, é dependente do domínio em questão – não existe um valor de α que proporcione o desempenho desejado para todos os domínios.

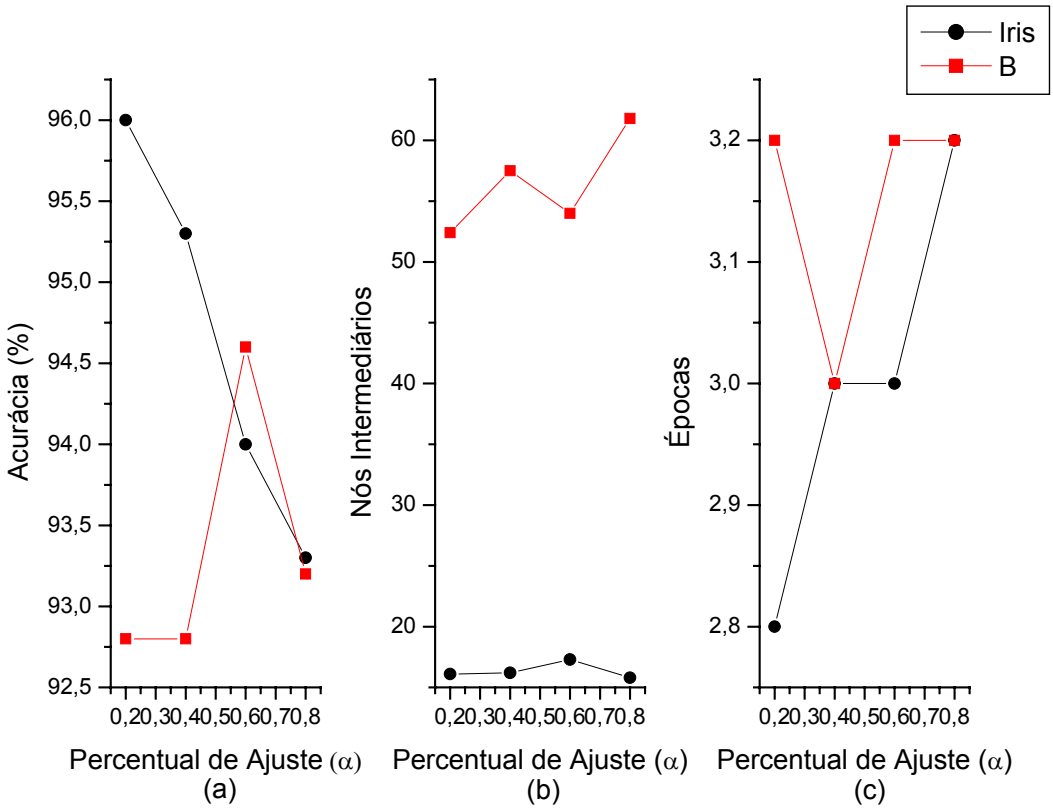


Figura 5.4: Gráfico de desempenho do RuleNet nos domínios Íris e B.

5.5 RuleNet *versus* NGE

Como discutido anteriormente, os sistemas NGE e RuleNet representam o conceito induzido por meio de hiper-retângulos no espaço n-dimensional. Os sistemas se diferenciam, entretanto, na maneira como a indução é feita.

O RuleNet parte da expressão (i.e., hiper-retângulo) mais geral e, à medida que o aprendizado progride, o RuleNet especializa essa expressão. O NGE faz exatamente o oposto, ou seja, parte da expressão mais especifica do conceito (i.e., uma instância de treinamento) e vai generalizando essa expressão à medida que o aprendizado progride.

O objetivo desta seção, portanto, é avaliar se os processos de especialização e generalização empregados pelo RuleNet e pelo NGE, respectivamente, interferem na acurácia do conceito induzido.

A Tabela 5.14 apresenta os resultados obtidos pelo RuleNet e pelo NGE na indução de conceitos em vários domínios de conhecimento. A coluna Domínio indica o domínio utilizado, a segunda coluna Sistema informa o sistema utilizado, sendo que os parênteses informam o valor do parâmetro α e do número de *seeds* utilizadas, no RuleNet e no NGE, respectivamente. A coluna Acurácia mostra a acurácia e, por fim a coluna Nós Intermediários/HR's indica a quantidade de nós intermediários, no caso do RuleNet, e a quantidade de hiper-retângulos (HR's), no caso do NGE.

Tabela 5.14: RuleNet *versus* NGE, '♣' indica os resultados onde a acurácia apresentou diferenças significativas, igualmente, '♦' indica os resultados onde o número de nós/HR's apresentou diferenças significativas.

Domínio	Sistema	Acurácia(%)	Nós Intermediários/HR's
A	RuleNet (0,8)	99,2 ± 1,0 ♣	35,5 ± 5,4
	NGE(5)	96,8 ± 3,0	24,5 ± 16,1 ♦
B	RuleNet (0,6)	94,6 ± 2,8	54,0 ± 3,7
	NGE(3)	92,4 ± 2,6	11,1 ± 5,3 ♦
C	RuleNet (0,8)	92,4 ± 4,2 ♣	118,5 ± 4,4 ♦
	NGE(3)	71,0 ± 7,8	191,4 ± 19,2
Íris	RuleNet (0,2)	96,0 ± 3,4	16,1 ± 2,7
	NGE(3)	94,7 ± 6,1	10,6 ± 2,0 ♦
Breast Cancer	RuleNet (0,6)	94,6 ± 3,2	54,7 ± 6,9
	NGE(15)	96,1 ± 1,7	20,9 ± 3,7 ♦
Liver Disorders	RuleNet (0,8)	50,2 ± 9,7	195,0 ± 12,3
	NGE(5)	61,2 ± 6,9 ♣	28,5 ± 7,7 ♦
E.coli	RuleNet (0,2)	70,5 ± 10,7	114,6 ± 5,5
	NGE(10)	85,2 ± 7,0 ♣	48,9 ± 6,9 ♦
Vestibular	RuleNet (0,4)	88,2 ± 8,3	30,9 ± 1,3
	NGE(5)	88,4 ± 7,4	19,0 ± 2,1 ♦

A Figura 5.5 apresenta o percentual de acurácia de ambos os sistemas, e pode ser visto que o RuleNet foi superior ao NGE em quatro domínios de

conhecimento (A, B, C, Íris e Breast Cancer), todavia, essa superioridade não foi significativa com exceção do domínio A e C, onde o RuleNet apresentou acurácia superior a 2% e 12,4%, respectivamente, com relação a acurácia apresentada pelo NGE. Já nos domínios Liver Disorders e E.coli, o NGE foi significativamente superior ao RuleNet apresentando acurácia superior a 10% e 14%, respectivamente. Estas diferenças favorecendo tanto o RuleNet quanto o NGE podem ser explicadas pelo fato de cada um dos processos de geração/reestruturação dos hiper-retângulos (nós intermediário representam hiper-retângulos) de ambos os modelos se adaptarem melhor num domínio do que em outro. No caso do domínio Liver Disorders, onde o RuleNet teve péssimo desempenho, o XRuleNet apresentou acurácia semelhante ao NGE de 61,7 ± 7,0 (veja Tabela 5.12), ou seja, o uso da métrica de distância na fase de classificação tanto pelo XRuleNet quanto pelo NGE, pode ter sido a causa do aumento de acurácia com relação do RuleNet.

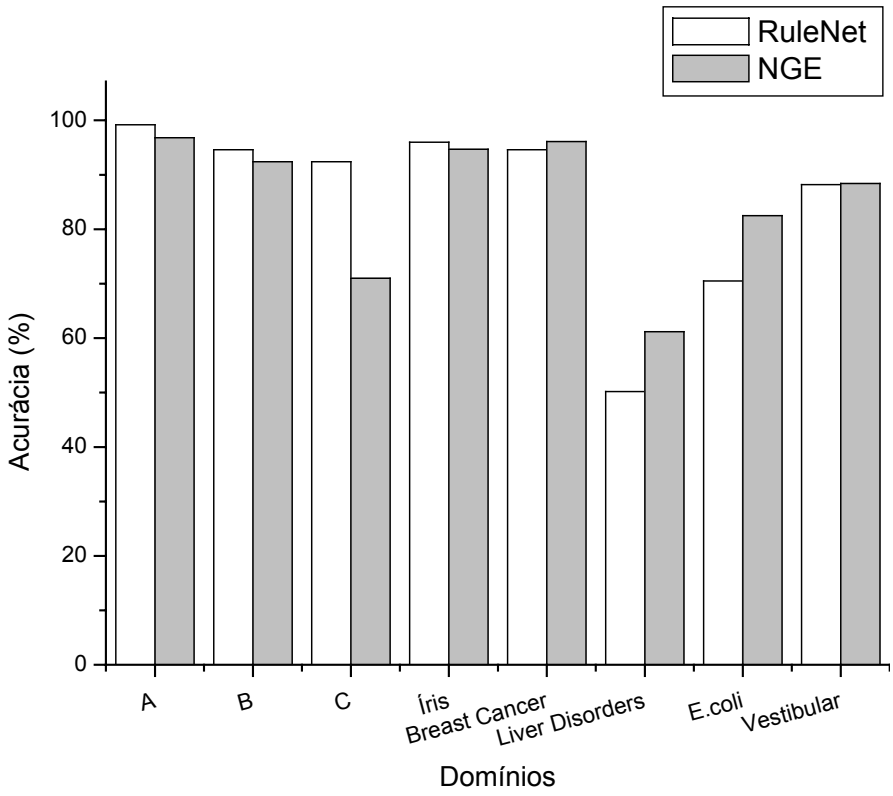


Figura 5.5: Gráfico de comparação de acurácia entre o RuleNet e o NGE.

O número de nós (i.e. hiper-retângulos), gerados pelo RuleNet, e o

número de HR's, gerados pelo NGE, podem ser comparados, uma vez que ambos podem ser traduzidos em regras, as quais representam o conceito induzido. Foram necessários menos hiper-retângulos para que o NGE apresentasse praticamente o mesmo desempenho na maioria dos domínios. Apenas no domínio C o NGE gerou um número maior hiper-retângulos e não conseguiu alcançar a acurácia do RuleNet, mas nos domínio Liver Disorders e E.coli o NGE usou menos hiper-retângulos que o RuleNet e obteve maior acurácia.

A Figura 5.6 apresenta o gráfico de comparação entre o RuleNet e o NGE com relação ao número de nós/hiper-retângulos gerados durante a indução de conceitos utilizando os domínios de conhecimento disponíveis.

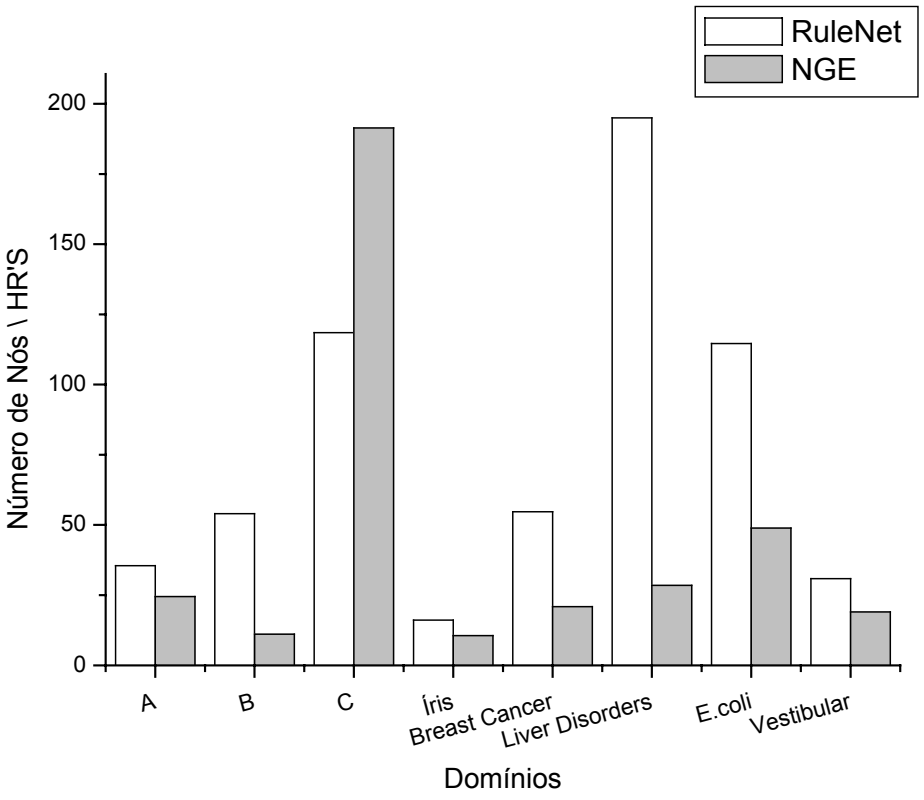


Figura 5.6: Números de nós e HR's gerados pelo RuleNet e NGE, respectivamente.

Como comentado anteriormente, tanto o RuleNet quanto o NGE possuem a capacidade de transformarem seus nós em regras, entretanto, existem algumas diferenças na obtenção das regras a partir desses sistemas. No NGE existem hiper-retângulos triviais e hiper-retângulos generalizados, os quais são

traduzidos em regras. Entretanto, não é usual utilizar os hiper-retângulos triviais como regras, uma vez que eles cobrem apenas um exemplo de treinamento e, portanto, não estão generalizados. Logo, a obtenção de regras pelo NGE se dá apenas pela tradução dos hiper-retângulos generalizados. Em contra partida, o RuleNet fornece todos os hiper-retângulos generalizados, isto acontece porque, cada vez que o RuleNet cria um nó intermediário, ele o cria o mais geral possível, e à medida que o treinamento acontece progride, pode diminuir (ou não) o nó em questão.

A Tabela 5.15 mostra na terceira coluna a quantidade de regras geradas pelo RuleNet (i.e. quantidade de nós intermediários) e pelo NGE (i.e. hiper-retângulos generalizados) respectivamente, e na quarta coluna a acurácia obtida pelo uso das regras geradas por cada um dos modelos na classificação de instâncias do domínio questão.

Tabela 5.15: RuleNet *versus* NGE, '♦' indica os resultados onde o número de regras apresentou diferenças significativas, igualmente, '♣' indica os resultados onde a acurácia apresentou diferenças significativas.

Domínios	**Sistema**	**Regras**	**Acurácia (%)**
A	RuleNet (0,8)	35,5 ± 5,4	99,2 ± 1,0
	NGE(5)	14,9 ± 4,6 ♦	91,8 ± 13,2
B	RuleNet (0,6)	54,0 ± 3,7	94,6 ± 2,8 ♣
	NGE(3)	9,3 ± 3,7 ♦	75,0 ± 11,2
C	RuleNet (0,8)	118,5 ± 4,4 ♦	92,4 ± 4,2 ♣
	NGE(3)	124,4 ± 3,7	43,8 ± 5,8
Íris	RuleNet (0,2)	16,1 ± 2,7	96,0 ± 3,4 ♣
	NGE(3)	10,2 ± 1,6 ♦	83,3 ± 11,9
Breast Cancer	RuleNet (0,6)	54,7 ± 6,9	94,6 ± 3,2 ♣
	NGE(15)	18,6 ± 2,2 ♦	88,2 ± 6,1
Liver Disorders	RuleNet (0,8)	195,0 ± 12,3	50,2 ± 9,7 ♣
	NGE(5)	23,7 ± 6,8 ♦	31,0 ± 7,5
E.coli	RuleNet (0,2)	114,6 ± 5,5	70,5 ± 10,7 ♣
	NGE(10)	38,1 ± 7,2 ♦	57,4 ± 10,3
Vestibular	RuleNet (0,4)	30,9 ± 1,3	88,2 ± 8,3 ♣
	NGE(5)	16,2 ± 2,3 ♦	69,6 ± 16,2

Apesar do número menor de regras geradas, com exceção do domínio C, o NGE gerou regras que apresentaram acurácia significativamente menor que o RuleNet, ou seja, as regras obtidas através do RuleNet se mostraram mais fiéis, do que as regras do NGE com relação ao domínio em questão. A Figura 5.7 mostra os índices de acurácia obtidos pelas regras geradas pelo RuleNet e pelo NGE, onde é possível observar a superioridade da acurácia das regras RuleNet com relação ao NGE.

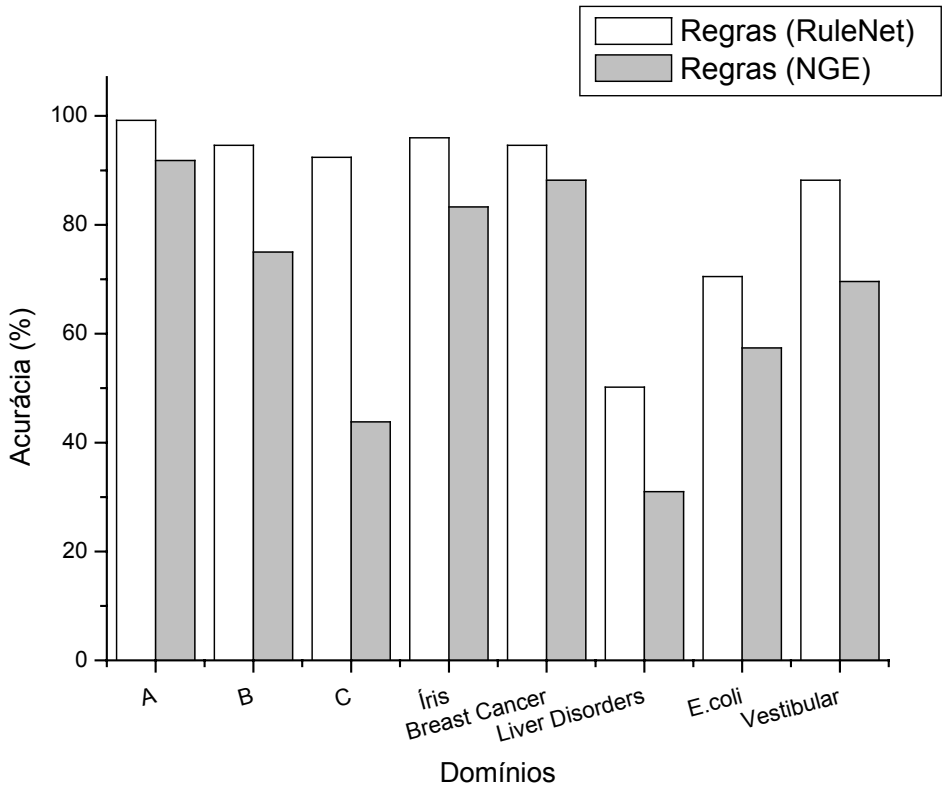


Figura 5.7: Gráfico de comparação de acurácia entre as regras geradas pelo RuleNet e as regras geradas pelo NGE.

Tal superioridade pode ser explicada pelo fato de que o NGE utiliza a métrica de similaridade para classificar exemplos que podem ou não estar contidos nos hiper-retângulos existentes e, é usada em todo o processo de aprendizado e classificação. O uso da métrica baseada na distância euclidiana colabora para que uma quantidade menor regras seja gerada, além, é claro, da permissividade do NGE na geração de hiper-retângulos aninhados e sobrepostos mesmo com classes diferentes. Já o RuleNet classifica apenas

instâncias contidas nas regiões de influência dos nós intermediários, sendo assim, pode ser necessário vários nós intermediários para obter um nível de acurácia satisfatório. Logo, as regras obtidas pelo RuleNet apresentam acurácia idêntica ao do próprio modelo, além disso, tais regras são as mais gerais possíveis graças fato do RuleNet criar nós intermediários já generalizados.

5.6 RuleNet *versus* ID3

A comparação de resultados entre RuleNet e ID3 pode ser vista na Tabela 5.16 onde, a coluna Domínio indica o domínio de conhecimento, a coluna Sistema faz referência aos modelos utilizados, a coluna Acurácia e a coluna Nós Ocultos/Regras indicam a acurácia e o números de nós intermediários/regras, respectivamente.

Tabela 5.16: RuleNet *versus* ID3, '♣' indica os resultados onde a acurácia apresentou diferenças significativas, igualmente, '♦' indica os resultados onde o número de nós intermediários/regras apresentou diferenças significativas.

Domínio	Sistema	Acurácia(%)	Nós Intermediários/Regras
A	RuleNet (0,8)	99,2 ± 1,0	35,5 ± 5,4♦
	ID3	96,6 ± 5,9	47,8 ± 3,4
B	RuleNet (0,6)	94,6 ± 2,8	54,0 ± 3,7
	ID3	91,6 ± 5,1	15,7 ± 0,9 ♦
C	RuleNet (0,8)	92,4 ± 4,2	118,5 ± 4,4
	ID3	100,0 ± 0,0 ♣	10,0 ± 0,0 ♦
Íris	RuleNet (0,2)	96,0 ± 3,4	16,1 ± 2,7 ♦
	ID3	95,3 ± 6,3	21,8 ± 2,6
Breast Cancer	RuleNet (0,6)	94,6 ± 3,2	54,7 ± 6,9 ♦
	ID3	94,3 ± 1,9	95,8 ± 7,3
Liver Disorders	RuleNet (0,8)	50,2 ± 9,7	195 ± 12,3
	ID3	63,2 ± 7,1 ♣	3,0 ± 0,0 ♦
E.coli	RuleNet (0,2)	70,5 ± 10,7	114,6 ± 5,5
	ID3	83,0 ± 9,7 ♣	68,9 ± 2,3 ♦
Vestibular	RuleNet (0,4)	88,2 ± 8,3	30,9 ± 1,3
	ID3	92,4 ± 6,8	25,7 ± 1,5 ♦

É importante lembrar que foi necessário utilizar algum processo de discretização dos atributos contínuos para que o ID3 pudesse induzir o conceito nestes domínios. O método de discretização utilizado para todos os domínios foi o *Recursive Minimal Entropy Partitioning* (RMEP) com exceção do domínio A onde foi utilizado o método *Equal Width Interval Binning* , uma vez que o método RMEP não conseguiu discretizar o domínio A devido à maneira como as classes estão dispostas neste domínio. Ambos os métodos de discretização são discutidos na Seção 3.2.4.

Nos domínios C, Liver Disorders e Ecoli, o ID3 apresentou acurácia superior ao RuleNet. Entretanto, o RuleNet apresentou bons resultados com relação ao ID3, mostrando a sua capacidade de classificação, uma vez que o ID3 é um algoritmo amplamente estudado e possui relativo sucesso na resolução de problemas de classificação. A Figura 5.8 mostra o gráfico de comparação de acurácia entre os domínios.

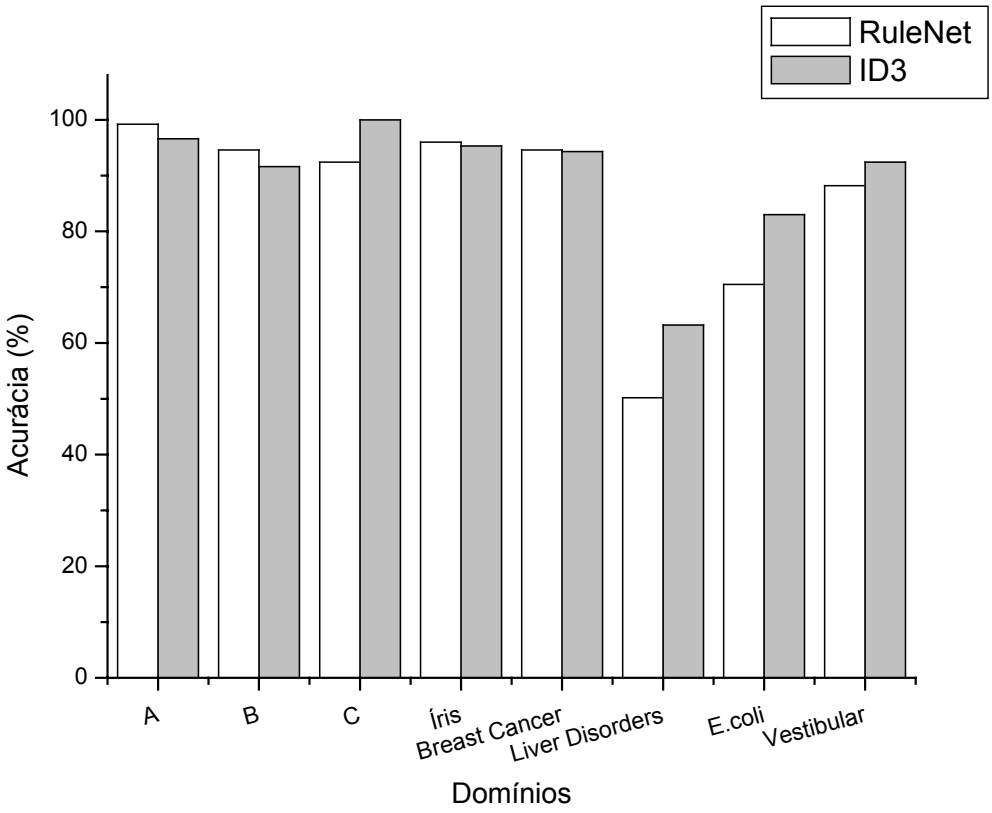


Figura 5.8: Gráfico de comparação de acurácia entre o RuleNet e o ID3.

Com relação à quantidade de nós intermediários (i.e. regras), geradas pelo RuleNet, em comparação com a quantidade de regras geradas pelo ID3, o RuleNet conseguiu com um número inferior de regras e acurácia semelhante ao ID3 nos domínios A e Íris (veja Tabela 5.16 e Figura 5.9). No restante dos domínios, entretanto, o ID3 descreveu o conceito em menos regras que o RuleNet, com destaque para o domínio Liver Disorders, onde o ID3 utiliza sempre 3 regras para descrever o conceito aprendido (note que o desvio padrão é 0) e apresenta um índice de acurácia maior que o RuleNet. Isto pode ser atribuído a abordagem baseada na entropia que o ID3 usa para induzir o conceito.

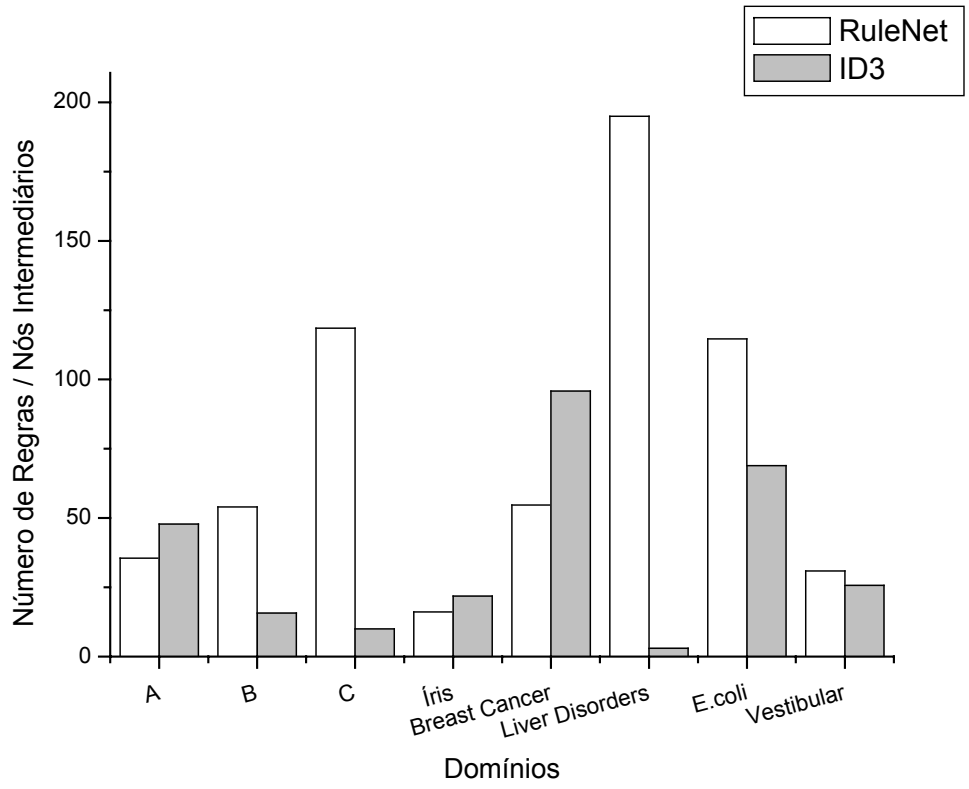


Figura 5.9: Número de regras e nós intermediários gerados pelo ID3 e RuleNet, respectivemente.

5.7 As Abordagens Cooperativas NGE→RuleNet e ID3→RuleNet

O RuleNet possui uma forte característica simbólica devido ao fato de utilizarem hiper-retângulos na representação do conhecimento adquirido. Logo, é possível extrair regras do conceito induzido, bem como inserir regras no RuleNet na forma de hiper-retângulos.

A idéia básica desta seção é investigar a cooperação entre sistemas por meio da transferência de conhecimento adquirido pelo ID3 ou NGE para o RuleNet. Ao invés de iniciar o treinamento com a rede vazia, o sistema RuleNet usará o conhecimento aprendido por estes dois métodos de aprendizado para estabelecer o conjunto inicial de hipóteses e, a partir daí iniciar o treinamento propriamente dito. Está não é uma tarefa difícil, uma vez que o NGE e o ID3 utilizam representações de conhecimento (hiper-retângulos e árvores de decisão, respectivamente) que podem ser facilmente traduzidas para o RuleNet. Entretanto, é importante lembrar que, os hiper-retângulos gerados pelo NGE devem que passar por um processo de adaptação, pois o RuleNet não permite que hiper-retângulos de classes diferentes estejam sobrepostos ou aninhados, situação que é comumente encontrada em hiper-retângulos gerados pelo NGE.

Para cada domínio, após a aplicação do *10-fold cross-validation* usando o RuleNet e o XRuleNet com o conjunto completo de exemplos, tal conjunto foi dividido em dois subconjuntos: um conjunto contendo 80% dos exemplos e outro contendo os 20% restantes. Para avaliar a cooperação NGE→RuleNet e ID3→RuleNet (e também sua extensão NGE→XRuleNet e ID3→XRuleNet), o subconjunto contendo 20% foi submetido apenas uma vez no ID3 (e no NGE). As regras obtidas pelo ID3 ou pelo NGE foram inseridas no RuleNet (e XRuleNet) e iniciado o treinamento utilizando *10-fold cross-validation* no conjunto contendo 80%.

A idéia por trás do uso de vários esquemas de aprendizado é uma tentativa de reproduzir a seguinte situação de aprendizado. A expressão do conceito induzido pelo ID3 ou NGE está disponível e corresponde a um conjunto de dados (neste caso o subconjunto contendo 20%) que não está mais disponível (hipoteticamente). É importante descobrir como esta expressão do

conceito pode ser usada pelo RuleNet na indução do conceito utilizando o conjunto de dados disponível (neste caso o conjunto contendo 80% do conjunto original). Os resultados obtidos pelo RuleNet na indução de conceitos utilizando 100% do conjunto são apresentados apenas como referência.

Os experimentos com a cooperação NGE→RuleNet e ID3→RuleNet foram realizados utilizando oito domínios de conhecimento. Os resultados são apresentados na Tabela 5.17 até a Tabela 5.24. O número de *seeds* usadas pelo NGE foi 5.

A primeira coluna das oito tabelas mostradas seguir apresenta cada um dos esquemas de aprendizado adotado neste experimento. RuleNet(100%) – é o sistema RuleNet utilizando 100% do conjunto de exemplos; RuleNet(80%) – é o sistema RuleNet utilizando apenas 80% do conjunto; ID3→RuleNet – ID3 aprende utilizando o conjunto com 20% dos exemplos e as regras obtidas são traduzidas em hiper-retângulos e são usadas para inicializar o RuleNet que inicia o processo de aprendizado o conjunto com 80% dos exemplos; NGE→RuleNet – é o mesmo que o ID3→RuleNet, mas desta vez, é o NGE que aprende usando 20% dos exemplos. Os esquemas também são utilizados envolvendo o XRuleNet ao invés do RuleNet.

Tabela 5.17: Resultados obtidos utilizando o domínio A.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	99,2 ± 1,0	35,5 ± 5,4	3,0 ± 0,5
RuleNet (80%)	95,0 ± 3,9	43,3 ± 10,8	2,9 ± 0,3
ID3 (20%) → RuleNet (80%)	91,7 ± 5,9	147,6 ± 1,8	2,0 ± 0,0
NGE (20%) → RuleNet (80%)	99,5 ± 1,2	14,6 ± 1,3	2,9 ± 0,3
XRuleNet (100%)	98,6 ± 2,1	45,5 ± 6,4	3,9 ± 2,2
XRuleNet (80%)	96,1 ± 3,7	34,6 ± 8,2	2,9 ± 0,3
ID3 (20) → XRuleNet (80%)	91,7 ± 5,9	147,6 ± 1,8	2,0 ± 0,0
NGE (20) → XRuleNet (80%)	99,5 ± 1,2	14,6 ± 1,3	2,0 ± 0,0

Tabela 5.18: Resultados obtidos utilizando o domínio B.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	94,6 ± 2,8	54,0 ± 3,7	3,2 ± 0,4
RuleNet (80%)	93,6 ± 5,7	39,8 ± 3,5	3,0 ± 0,0
ID3 (20%) → RuleNet (80%)	93,9 ± 3,9	49,6 ± 4,5	3,0 ± 0,0
NGE (20%) → RuleNet (80%)	94,7 ± 4,0	49,6 ± 6,3	3,1 ± 0,3
XRuleNet (100%)	93,6 ± 2,5	235,3 ± 23,2	10 ± 0,0
XRuleNet (80%)	94,7 ± 4,2	42,8 ± 3,6	3,0 ± 0,0
ID3 (20) → XRuleNet (80%)	93,9 ± 3,9	49,6 ± 4,5	2,2 ± 0,4
NGE (20) → XRuleNet (80%)	93,3 ± 5,7	43,6 ± 3,5	3,0 ± 0,0

Tabela 5.19: Resultados obtidos utilizando o domínio C.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	92,4 ± 4,2	118,5 ± 4,4	3,0 ± 0,0
RuleNet (80%)	83,7 ± 3,4	121,2 ± 9,6	3,0 ± 0,0
ID3 (20%) → RuleNet (80%)	100 ± 0,0	10,0 ± 0,0	1,0 ± 0,0
NGE (20%) → RuleNet (80%)	65,4 ± 2,3	132,1 ± 3,3	3,5 ± 0,2
XRuleNet (100%)	91,0 ± 3,7	140,9 ± 11,3	2,5 ± 3,7
XRuleNet (80%)	80,7 ± 8,6	140,1 ± 5,6	2,1 ± 0,3
ID3 (20) → XRuleNet (80%)	100 ± 0,0	10,0 ± 0,0	1,0 ± 0,0
NGE (20) → XRuleNet (80%)	69,5 ± 3,4	139,1 ± 4,3	4,1 ± 1,1

Tabela 5.20: Resultados obtidos utilizando o domínio Breast Cancer.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	94,6 ± 3,2	54,7 ± 6,9	3,0 ± 0,0
RuleNet (80%)	95,0 ± 3,4	37,7 ± 4,4	3,0 ± 0,0
ID3 (20%) → RuleNet (80%)	92,5 ± 3,4	60,7 ± 6,2	3,0 ± 0,0
NGE (20%) → RuleNet (80%)	93,1 ± 3,1	48,7 ± 3,8	3,0 ± 0,0
XRuleNet (100%)	94,3 ± 2,4	119,7 ± 29,7	3,9 ± 2,7
XRuleNet (80%)	95,0 ± 2,0	38,7 ± 3,0	3,0 ± 0,0
ID3 (20) → XRuleNet (80%)	92,5 ± 3,4	60,7 ± 6,2	3,0 ± 0,0
NGE (20) → XRuleNet (80%)	92,7 ± 3,8	56,7 ± 3,2	3,0 ± 0,0

Tabela 5.21: Resultados obtidos utilizando o domínio Liver Disorders.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	50,2 ± 9,7	195,0 ± 12,3	3,1 ± 0,3
RuleNet (80%)	45,8 ± 10,6	162,1 ± 4,7	3,3 ± 0,5
ID3 (20%) → RuleNet (80%)	42,8 ± 9,0	167,0 ± 6,3	3,7 ± 0,5
NGE (20%) → RuleNet (80%)	36,8 ± 6,8	188,5 ± 6	3,0 ± 0,0
XRuleNet (100%)	61,7 ± 7,0	240,5 ± 14,6	6,8 ± 3,4
XRuleNet (80%)	45,5 ± 7,3	160,7 ± 7,7	3,2 ± 0,4
ID3 (20) → XRuleNet (80%)	42,8 ± 9,0	167,0 ± 6,3	3,4 ± 0,5
NGE (20) → XRuleNet (80%)	36,8 ± 6,8	188,5 ± 6,0	3,0 ± 0,0

Tabela 5.22: Resultados obtidos utilizando o domínio E.coli.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	70,5 ± 10,7	114,6 ± 5,5	3,0 ± 0,0
RuleNet (80%)	70,0 ± 9,5	96,3 ± 3,9	3,0 ± 0,0
ID3 (20%) → RuleNet (80%)	72,3 ± 8,8	118,3 ± 7,9	3,0 ± 0,0
NGE (20%) → RuleNet (80%)	54,8 ± 11,1	142 ± 5,1	3,0 ± 0,0
XRuleNet (100%)	69,0 ± 6,3	129,5 ± 6,6	3,2 ± 0,4
XRuleNet (80%)	64,2 ± 8,5	102,9 ± 5,5	3,0 ± 0,0
ID3 (20) → XRuleNet (80%)	72,2 ± 8,8	118,3 ± 7,9	3,0 ± 0,0
NGE (20) → XRuleNet (80%)	54,8 ± 11,1	142,0 ± 5,1	3,0 ± 0,0

Tabela 5.23: Resultados obtidos utilizando o domínio Vestibular.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	88,2 ± 8,3	30,9 ± 1,3	3,0 ± 0,0
RuleNet (80%)	85,5 ± 15	22,7 ± 1,9	2,9 ± 0,3
ID3 (20%) → RuleNet (80%)	82,9 ± 17,2	37,4 ± 2,9	2,7 ± 0,5
NGE (20%) → RuleNet (80%)	82,9 ± 10,3	37,1 ± 3,4	3,0 ± 0,0
XRuleNet (100%)	87,3 ± 5,1	84,9 ± 22,5	4,3 ± 2,2
XRuleNet (80%)	83,0 ± 15,8	29,0 ± 3,7	2,9 ± 0,3
ID3 (20) → XRuleNet (80%)	82,9 ± 17,2	37,4 ± 2,9	3,2 ± 0,4
NGE (20) → XRuleNet (80%)	82,9 ± 10,3	37,1 ± 3,4	3,0 ± 0,0

Tabela 5.24: Resultados obtidos utilizando o domínio Excipientes.

Sistema	Acurácia(%)	Nós Intermediários	Época
RuleNet (100%)	92,9 ± 18,6	18,7 ± 0,7	2 ± 0
RuleNet (80%)	98,1 ± 5,9	23 ± 1,2	2,1 ± 0,3
ID3 (20%) → RuleNet (80%)	96,8 ± 6,1	62,5 ± 1,2	2 ± 0
NGE (20%) → RuleNet (80%)	95,5 ± 5,3	41,1 ± 1,1	2 ± 0
XRuleNet (100%)	88,8 ± 19,3	30 ± 4,4	2 ± 0
XRuleNet (80%)	91,6 ± 5,9	46,4 ± 1,4	2,7 ± 0,5
ID3 (20) → XRuleNet (80%)	96,2 ± 4,4	65,5 ± 1,8	3,7 ± 0,9
NGE (20) → XRuleNet (80%)	90,9 ± 6,4	43,5 ± 1,8	2,8 ± 0,4

Para comparação, a Tabela 5.25 apresenta os resultados obtidos com os conceitos induzidos pelo ID3 e NGE, os quais foram utilizados para inicializar o RuleNet, a partir do subconjunto contendo 20% dos exemplos. O subconjunto contendo 80% foi utilizado como conjunto de teste. Tais resultados são úteis para a verificação do potencial da cooperação realizada.

Os resultados apresentados nas Tabelas 5.16 a 5.23 mostram que a cooperação entre ID3→RuleNet e NGE→RuleNet é possível, praticável e de fácil implementação. Em quatro domínios (A,B,C e E.coli) pelo menos um esquema de cooperação apresentou resultados ligeiramente melhores que o RuleNet(80%) ou XRuleNet(80%). Nos outros quatro domínios (Vestibular, Liver Disorders, Breast Câncer e Excipientes), os resultados apresentados pelos esquemas de cooperação não foram tão bons quanto os resultados apresentados pelo RuleNet(80%) ou XRuleNet(80%). É importante ressaltar que o baixo desempenho apresentado pelas colaborações (principalmente pela cooperação NGE→RuleNet e NGE→XRuleNet) no domínio Liver Disorders é causado pela natureza deste domínio, a qual possui atributos que contém muitos ruídos. Ainda com relação ao Liver Disorders, a Tabela 5.25 mostra o ID3, que apresenta o melhor resultado obtido neste domínio com apenas 3 regras geradas. Os esquemas de cooperação envolvendo tanto o RuleNet quanto o XRulenet apresentaram resultados semelhantes.

Tabela 5.25: Resultados obtidos utilizando o ID3 e NGE (20% do conjunto para treinamento e 80% para teste).

Domínio	Sistema	Acurácia(%)	No. de Regras
A	ID3	69,8	91
	NGE	96,7	6
B	ID3	91,5	13
	NGE	84,8	3
C	ID3	100	10
	NGE	44,3	65
Breast Cancer	ID3	93,2	17
	NGE	96,0	5
Liver Disorders	ID3	60,5	3
	NGE	60,0	13
E.coli	ID3	78,2	25
	NGE	66,5	26
Vestibular	ID3	88,6	8
	NGE	90,5	7
Excipients	ID3	75,0	35
	NGE	42,2	15

Com relação à cooperação ID3→RuleNet e ID3→XRuleNet, os resultados da Tabela 5.19 e Tabela 5.25 referentes ao domínio C, quando confrontados, mostram que o ID3 é o responsável pelo desempenho obtido, ou seja, não há cooperação por parte do RuleNet e do XRuleNet. O domínio C não é apropriado para sistema que induzem o conceito por meio de hiper-retângulos.

O número de nós intermediários obtido pelo RuleNet(80%) é menor do que o obtido por qualquer um dos esquemas de cooperação, com exceção dos domínios A e C.

Outros fatores, porém, podem ter uma leve influência nos resultados obtidos com os esquemas de cooperação. O método de discretização utilizado pelo ID3 pode ter uma pequena influência no desempenho apresentado. Já o NGE é sensível a ordem de apresentação do conjunto de treinamento, e ao número de *seeds* escolhido pelo usuário. Além disso, o desempenho apresenta pelas colaborações envolvendo o NGE podem ser explicadas pelo fato de que os

hiper-retângulos gerados pelo NGE são alterados para satisfazer as restrições impostas pelo RuleNet.

De qualquer forma os experimentos evidenciam que a cooperações NGE→RuleNet e ID3→RuleNet são possíveis e apresentam bons resultados. Logo, o RuleNet é capaz de adaptar o conhecimento prévio ao conhecimento induzido a partir de um conjunto de exemplos. Não pode ser esquecido, entretanto, que a cooperação só é possível devido às linguagens de representação de conhecimento utilizadas pelos três sistemas, que permitem a tradução de uma para outra de maneira relativamente simples.

5.8 Fuzzy RuleNet

A Tabela 5.26 apresenta os resultados obtidos pelo Fuzzy RuleNet na indução de conceitos em oito domínios de conhecimento. A coluna "Função de Pertinência" indica, como o próprio nome diz, a função de pertinência utilizada pelo sistema Fuzzy RuleNet. O parâmetro α (determina o percentual de redução das regiões de influência) e, o parâmetro Ω (determina o grau de sobreposição entre os conjuntos fuzzy), foram estabelecidos em 0,2 e 50%, respectivamente. Além disso, o número máximo de épocas de treinamento foi fixado em 5.

Pode ser visto que, não houve diferenças significativas entre o Fuzzy RuleNet utilizando a função triangular, e o Fuzzy RuleNet usando a função trapezoidal. O Fuzzy RuleNet não convergiu nenhuma vez na indução de conceitos nos domínios utilizados. As mudanças feitas nos pontos de referência e o método de classificação utilizado podem ter sido responsáveis pelo não convergência da rede antes de 5 épocas.

Outro fator que deve ser observado é que o número de nós intermediários é relativamente alto, ou seja, é difícil para o especialista entender uma quantidade relativamente grande de regras fuzzy. Logo, o refinamento de sistemas fuzzy já existentes utilizando o Fuzzy RuleNet pode não ser uma solução adequada.

Tabela 5.26: Desempenho do Fuzzy RuleNet utilizando as funções de pertinência triangular e trapezoidal.

Domínio	Função de Pertinência	Acurácia(%)	Nós Intermediários	Épocas
A	Triangular	80,8 ± 5,6	572,1 ± 28,1	5 ± 0
	Trapezoidal	78,0 ± 8,2	576,2 ± 26,6	5 ± 0
B	Triangular	81,8 ± 10,3	547,5 ± 33,4	5 ± 0
	Trapezoidal	77,2 ± 9,1	548,6 ± 36,9	5 ± 0
C	Triangular	40,4 ± 12,8	1448,8 ± 61,3	5 ± 0
	Trapezoidal	40,4 ± 8,9	1389,5 ± 53,9	5 ± 0
Íris	Triangular	90,0 ± 7,9	106,1 ± 13,6	5 ± 0
	Trapezoidal	91,3 ± 7,7	104,9 ± 16,9	5 ± 0
Breast Cancer	Triangular	89,0 ± 4,5	518,3 ± 16,2	5 ± 0
	Trapezoidal	89,2 ± 4,6	519,6 ± 21,8	5 ± 0
Liver Disorders	Triangular	56,6 ± 10,5	710,5 ± 26,5	5 ± 0
	Trapezoidal	55,9 ± 9,8	721,0 ± 24,2	5 ± 0
E.coli	Triangular	66,7 ± 6,6	599,9 ± 29,3	5 ± 0
	Trapezoidal	65,5 ± 8,9	603,9 ± 29,6	5 ± 0
Vestibular	Triangular	74,2 ± 16,9	223,6 ± 19,5	5 ± 0
	Trapezoidal	77,0 ± 14,4	228,2 ± 20,6	5 ± 0

5.9 Fuzzy RuleNet *versus* Fuzzy NGE

Esta seção apresenta e discute os resultados obtidos de experimentos utilizando o Fuzzy NGE [Santos 1997] e o Fuzzy RuleNet, comparativamente. Como o Fuzzy NGE é utilizado apenas para confrontar o Fuzzy RuleNet, a descrição deste sistema é apresentada no Anexo A. Os resultados apresentados pelo Fuzzy NGE foram extraídos de [Lisboa et al. 2003]. Além disso, os experimentos com o Fuzzy NGE e com o Fuzzy RuleNet foram conduzidos utilizando *5-fold cross-validation*.

Os parâmetros δ e τ, utilizados pelo Fuzzy NGE, foram determinados empiricamente para cada um dos domínios utilizados. A Tabela 5.27 mostra os valores considerados ideais para cada um dos domínios em questão.

Tabela 5.27: Parâmetros δ e τ utilizados no Fuzzy NGE.

Domínio	**δ**	**τ**
Balance Scale	1	0,5
Hayes Roth	0,5	0,2
Ionosphere	0,1	0,5
Iris	0,5	1
Wine	1	1

A Tabela 5.28 apresenta os resultados utilizando a métrica de similaridade similar(A,B)=c(A∪B)/c(A∩B) (veja Anexo A). O número de *seeds* foi fixado em 3 para todos os domínios. É importante ressaltar que a proposta do Fuzzy NGE não trata situações onde funções de pertinência referentes a um determinado atributo estão aninhadas (i.e. totalmente sobrepostas) e os respectivos hiper-retângulos pertencem a classes diferentes. Na Tabela 5.28 a coluna N.C. apresenta a quantidade de exemplos não classificados. O domínio Ionosphere é o único domínio onde o valor desta coluna é representativo. A coluna HR's representa a quantidade de hiper-retângulos criados, o qual descreve o conceito aprendido, e Conj. Fuzzy representa o número de generalizações fuzzy que ocorreram durante a fase de treinamento.

Assim como o sistema NGE, o sistema Fuzzy NGE é sensível à ordem de apresentação dos exemplos de treinamento. Segundo [Salzberg 1989] é difícil determinar qual a ordem de apresentação mais apropriada. Logo, nos resultados obtidos a ordem de apresentação foi determinada aleatoriamente.

Tabela 5.28: Resultados obtidos pelo Fuzzy NGE.

Domain	**Acurácia(%)**	**N.C**	**HR's**	**Conj. Fuzzy**
Balance Scale	79,7 ± 3,95	0,0	65,4	10,2
Hayes Roth	49,7 ± 3,32	0,0	22,0	13,4
Ionosphere	91,8 ± 1,96	6,2	112,0	1052,4
Iris	87,1 ± 3,60	0,0	3,4	65,8
Wine	93,2 ± 0,8	2,2	20,8	416,8

O Fuzzy NGE apresentou um desempenho muito fraco com relação ao domínio Hayes Roth. Uma das possíveis causas do baixo desempenho pode ser atribuído ao próprio domínio, uma vez que todos os atributos possuem valores de 1 a 4, os quais representam categorias, e a generalização é baseada nestes valores.

Os resultados apresentados na Tabela 5.29 foram obtidos pelo Fuzzy RuleNet utilizando funções de pertinência triangular e trapezoidal, respectivamente. O método MAX foi utilizado para ajustar as regiões de influência de acordo com o parâmetro α=0,2. Além disso, o parâmetro Ω foi fixado 50%. O número de épocas foi fixado em 5.

Tabela 5.29: Desempenho do Fuzzy RuleNet.

Domínio	**Função de Pertinência**	**Acurácia(%)**	**Nós Intermediários**
Balance Scale	Triangular	77,6 ± 11,2	687,3 ± 25,9
	Trapezoidal	77,2 ± 5,5	678,3 ± 40,8
Hayes Roth	Triangular	67,0 ± 5,1	111,1 ± 9,8
	Trapezoidal	66,3 ± 5,6	107,3 ± 6,4
Ionosphere	Triangular	65,9 ± 3,8	490,8 ± 15,5
	Trapezoidal	65,9 ± 4,6	491 ± 17,5
Íris	Triangular	90,7 ± 12,3	101,7 ± 14,1
	Trapezoidal	86,0 ± 11,5	127,1 ± 27,7
Wine	Triangular	70,7 ± 11,2	250,6 ± 27,5
	Trapezoidal	71,2 ± 10,4	249,5 ± 27,2

Pode ser visto que as diferenças entre acurácia e quantidade de nós apresentados pelo Fuzzy RuleNet usando a função triangular, e usando a função trapezoidal, são estatisticamente insignificantes.

Comparando a acurácia obtida pelo Fuzzy NGE (Tabela 5.28) e pelo Fuzzy RuleNet (Tabela 5.29), é possível perceber que os resultados são levemente favoráveis ao Fuzzy NGE. Porém, quando a comparação é feita com base na quantidade de hiper-retângulos induzidos, o Fuzzy NGE mostra-se bem superior do Fuzzy RuleNet.

5.10 Considerações Finais

Alguns dos principais tópicos abordados neste capítulo foram:

- Experimentos envolvendo comparações entre o RuleNet e o XRuleNet, entre o RuleNet e o NGE e, entre o RuleNet e o ID3, abordando as questões relacionadas a estes quatro sistemas, bem como suas vantagens e desvantagens.
- A cooperação envolvendo os sistemas RuleNet, XRuleNet, ID3 e NGE na indução de conceitos em vários domínios de conhecimento, apresentando e discutindo os resultados obtidos do uso de alguns esquemas de cooperação.
- Alguns experimentos com o Fuzzy RuleNet mostrando sua capacidade de utilizar a teoria dos conjuntos fuzzy em problemas de classificação, bem como sua desvantagem com relação ao número de regras geradas ao final do treinamento.

O próximo capítulo apresenta as conclusões deste trabalho, abordando as principais questões envolvendo o sistema RuleNet, XRuleNet e Fuzzy RuleNet. Além disso, serão apresentados os possíveis desdobramentos desta pesquisas, bem como os trabalhos futuros.

Capítulo 6

Conclusões

O trabalho de pesquisa realizado nesta dissertação teve como principal objetivo a investigação e análise do modelo neural RuleNet e sua versão estendida para domínios *fuzzy*, com ênfase nas características simbólica e cooperativa desse modelo.

A pesquisa desenvolvida envolveu, primeiramente, familiarização e aquisição dos principais conceitos de aprendizado de máquina e, mais especificamente, o aprendizado usando árvores de decisão (focalizando principalmente o ID3), e o aprendizado usando redes neurais *feedforward*. Foi feito um levantamento bibliográfico de métodos que integram modelos distintos de aprendizado, com estudo detalhado dos modelos implementados pelo RuleNet, o KBANN e o TREPAN. Além disso, o modelo de aprendizado baseado em exemplares chamado NGE foi investigado, devido às suas características similares ao RuleNet, principalmente no que diz respeito à linguagem de representação de conceitos utilizada por ambos, com vistas a uma possível cooperação entre os dois modelos.

O modelo RuleNet foi o foco principal da pesquisa devido às suas características incremental, simbólica e facilidade potencial de ser integrado a um sistema de aprendizado, de maneira cooperativa. Apesar de proposto e caracterizado como um modelo neural, esse trabalho evidencia que o RuleNet pode perfeitamente ser considerado um sistema de aprendizado simbólico ao

qual foram incorporados alguns conceitos relacionados a aprendizado neural.

As referências bibliografias relativas ao RuleNet são vagas e não descrevem completamente o algoritmo. Como um dos objetivos da pesquisa foi o desenvolvimento de um sistema computacional RuleNet, com vistas à experimentação, uma das contribuições deste trabalho foi a especificação completa do pseudocódigo do algoritmo, o que permitirá que outros sistemas RuleNet possam ser implementados usando exatamente o mesmo algoritmo. Além disso, o trabalho propõe dois métodos de ajuste das regiões de influência (i.e. hiper-retângulos).

Com base nos experimentos realizados e descritos neste trabalho, pode-se dizer que:

- os experimentos com o RuleNet e com o XRuleNet mostraram que o uso da movimentação dos pontos de referência afeta a convergência do XRuleNet fazendo com que o número de nós intermediários criados seja, no geral, maior que o do RuleNet. Entretanto, o XRuleNet, por meio do uso da classificação "forçada", mostrou ser capaz de classificar instâncias antes consideradas pelo RuleNet como "desconhecidas". Com relação ao desempenho, ambos se mostraram como soluções úteis e praticáveis em problemas de classificação, principalmente onde a obtenção de regras se faz necessária.
- os resultados obtidos por meio da comparação do RuleNet versus o NGE mostraram que a acurácia de ambos os sistemas são parecidas, o RuleNet foi melhor em alguns domínios enquanto que o NGE foi melhor em outros. Os resultados dos experimentos evidenciaram que a diferença significativa entre esses dois sistemas está na quantidade de hiper-retângulos existentes ao final do treinamento. O NGE, na maior parte dos domínios, induziu o conceito utilizando menos hiper-retângulos que o RuleNet. Entretanto, a acurácia das regras traduzidas do RuleNet é maior que a acurácia das regras traduzidas do NGE.
- na comparação entre RuleNet e ID3 foi possível perceber que a quantidade de regras geradas pelo ID3 foi menor que o RuleNet, apenas com algumas exceções. O desempenho de cada um foi

fortemente determinado pelo domínio em questão. O RuleNet mostrou equilíbrio com relação ao ID3, o qual é um algoritmo consagrado e conhecido.

- o RuleNet pode ser integrado a outros sistemas de aprendizado, constituindo um ambiente de aprendizado cooperativo. Essa integração é particularmente fácil de ser feita, se os outros sistemas envolvidos tiverem uma linguagem de representação de conceitos que possa ser "traduzida" em hiper-retângulos, como é o caso do NGE e do ID3. As cooperações propostas e investigadas neste trabalho foram NGE→RuleNet e ID3→RuleNet. Os resultados obtidos foram bons, dado a facilidade de implementação das cooperações, e mostraram que o RuleNet é capaz de utilizar conhecimento prévio na indução de conceitos. Entretanto, as regras devem ser adaptadas para satisfazer às restrições do RuleNet.
- os resultados obtidos com o Fuzzy RuleNet não foram conclusivos com relação a acurácia, entretanto, o Fuzzy RuleNet apresentou um número de nós intermediários (i.e. regras) relativamente alto, o que torna difícil o entendimento do conceito induzido por parte do especialista.
- na comparação entre Fuzzy RuleNet e Fuzzy NGE pôde-se observar que os resultados de acurácia são levemente favoráveis ao Fuzzy NGE. Na comparação do número de hiper-retângulos induzidos, ficou evidenciada a desvantagem do Fuzzy RuleNet, ou seja, o Fuzzy NGE apresentou um número de hiper-retângulos significativamente menor que o Fuzzy RuleNet.

Um possível elenco de atividades para dar continuidade às pesquisas realizadas e descritas neste trabalho pode ser:

- adaptar ao RuleNet, um procedimento de controle com relação à criação de hiper-retângulos durante o treinamento, utilizando algum tipo de métrica, visto que a grande desvantagem do RuleNet e de sua versão *fuzzy* é o número de nós intermediários criados.
- propor um algoritmo de treinamento não-supervisionado para o

RuleNet e, conseqüentemente, adaptá-lo ao NGE.

- investigar o uso do Fuzzy RuleNet em problemas de controle, devido à possível adaptação do Fuzzy RuleNet de acordo com o controlador Mandani.
- adaptar o Fuzzy RuleNet, de forma que o mesmo mantenha a semântica dos conjuntos *fuzzy*, caso ela exista.
- refinar e melhorar as cooperações entre NGE→RuleNet e ID3→RuleNet, principalmente, com relação à adaptação das regras geradas pelo NGE.
- propor novas cooperações entre o RuleNet e outros métodos de aprendizado.
- adicionar ao RuleNet uma proposta de tratamento de pesos de atributos e de hiper-retângulos.
- implementar a colaboração RuleNet → NGE.

Referências Bibliográficas

[Bennett e Mangasarian 1992]

Bennett, K. P. e Mangasarian, O. L. *Robust linear programming discrimination of two linearly inseparable sets*. Optimization Methods and Software, vol. 1, pp. 23-34,1992.

[Blake e Merz 1998]

Blake, C. L. e Merz, C. J. UCI Repository of machine learning databases. Dept of Information and Computer Science, University of California, Irvine, CA, http://www.ics.uci.edu/~mlearn/MLRepository.html,1998.

[Briscoe e Caelli 1996]

Briscoe, G. e Caelli, T. A Compendium of Machine Learning Volume 1: Symbolic Machine Learning. New Jersey, Ablex Publishing Corp,1996.

[Carbonell 1989]

Carbonell, J. E. *Introduction: Paradigms for machine learning*. Artificial Intelligence, vol. 40, pp. 1-9,1989.

[Chmielewski e Grzymala-Busse 1994]

Chmielewski, M. R. e Grzymala-Busse, J. W. Global Discretization of Continuous Attributes as Preprocessing for Machine Learning. In Lin, T. Y. (ed.) *3th International Workshop on Rough Sets and Soft Computing*, San Jose, CA, AAAI Press, pp. 294-301,1994.

[Craven e Shavlik 1994]

Craven, M. e Shavlik, J. Using sampling and queries to extract rules from trained neural networks. In *Proceedings on 11th International Conference on Machine Learning*, New Brunswick, NJ, Morgan Kaufmann, pp. 37-45,1994.

[Craven e Shavlik 1995]

Craven, M. e Shavlik, J. Extracting comprehensible concept representations from trained neural networks. Montreal, Quebec, Canada,1995.

[Craven e Shavlik 1996]

Craven, M. e Shavlik, J. Extracting tree-structured representations of trained networks. In Touretzky, D. S., Mozer, M. C. e Hasselmo, M. E. (eds.) *Advances in Neural Information Processing Systems*, Denver, CO, The MIT Press, pp. 24-30,1996.

[Craven 1996]

Craven, M. W. *Extracting Comprehensible Models from Trained Neural Networks*. Tese (Doutorado), Department of Computer Science, University of Winsconsin, Madison, p. 186,1996.

[Dabija e Tschichold-Gurman 1993]

Dabija, V. G. e Tschichold-Gurman, N. A framework for combining symbolic and connectionist learning with equivalent concept descriptions. In *Proceedings on International Joint Conference on Neural Networks*, pp. 790-793,1993.

[Dietterich 1998]

Dietterich, T. G. *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms*. Neural Computation, vol. 10(7), pp. 1895-1923,1998.

[Dougherty et al. 1995]

Dougherty, J., Kohavi, R. e Sahami, M. Supervised and Unsupervised Discretization of Continuous Features. In Prieditis, A. e Russel, S. (eds.) *Proceedings of 12th Internation Conference on Machine Learning*, San Fransisco, CA, Morgan Kaufmann,1995.

[Duda e Hart 1974]

Duda, R. O. e Hart, P. E. Pattern Classification and Scene Analysis. New York, Jonh Wiley and Sons,1974.

[Fayyad e Irani 1993]

Fayyad, U. M. e Irani, K. B. Multi-Interval Discretization for Continuous-Valued Attributes for Classification Learning. In *Proceeding of the 13th*

Internation Joint Conference on Artificial Intelligence, Sydney, Morgan Kauffmann, pp. 1022-1027,1993.

[Figueira et al. 2003]

Figueira, L. B., Colafemina, J. F. e Roque, A. C. Diagnóstico de Patologias do Sistema Vestibular utilizando Redes Neurais na Análise dos Movimentos Sacádicos. In Anido, R. O. e Masiero, P. C. (eds.) *Anais do XXIII Congresso da Sociedade Brasileira de Computação*, Campinas, São Paulo, pp. 55-62,2003.

[Figueira e Nicoletti 2004]

Figueira, L. B. e Nicoletti, M. C. Choosing the initial set of exemplars when learning with an NGE-based system. In Valafar, F. (ed.) *IEEE International Conference on Information in Technology: Coding and Computing*, Las Vegas, IEEE Press,2004.

[Fisher e McKusick 1989]

Fisher, D. H. e McKusick, K. B. An empirical comparison of ID3 and back-propagation. In Sridharan, N. S. (ed.) *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, USA, Morgan Kaufmann, pp. 788-793,1989.

[Fisher 1936]

Fisher, R. A. *The use of multiple measurements in taxonomic problems*. Annual Eugenics, vol. 7(2), pp. 179-188,1936.

[Fu 1991]

Fu, L. M. Rule learning by searching on adapted nets. In McKeown, K. (ed.) *Proceedings of the 9th National Conference on Artificial Intelligence*, MIT Press, pp. 590-595,1991.

[Fu 1994]

Fu, L. M. *Rule generation from neural networks*. IEEE Transactions on Systems, Man and Cybernetics, vol. 8(24), pp. 1114-1124,1994.

[Gallant 1988]

Gallant, S. I. *Connectionist expert systems*. Communications of the ACM - CACM, vol. 32(2), pp. 152-169,1988.

[Gallant 1994]

Gallant, S. I. Neural network learning and expert systems. London, The MIT Press,1994.

[Gou et al. 1998]

Gou, M., Norihiro, S. e Kazuo, U. Applying Design Patterns to Decision Tree Learning System. In *Proceedings of the ACM SIGSOFT 6th International Symposium on the Foundations of Software Engineering (FSE-98)*, New York, ACM Press, pp. 111-120,1998.

[Haykin 1994]

Haykin, S. Neural Networks : A Comprehensive Foundation. New York, Macmillan,1994.

[Hinton 1990]

Hinton, G. E. Connectionist learning procedures. Machine Learning: An Artificial Intelligence Approach. Kodratoff, Y. e Michalski, R. S. (eds.), vol. 3,1990.

[Hopfield 1982]

Hopfield, J. J. Neural networks and physical systems with emergent collective behavior. In *Proc. Nat. Acad. Sci*, USA, pp. 2554-2558,1982.

[Horton e Nakai 1996]

Horton, P. e Nakai, K. A Probablistic Classification System for Predicting the Cellular Localization Sites of Proteins. In States, D. J., Agarwal, P., Gaasterland, T., Hunter, L. e Smith, R. (eds.) *Proceedings of the Fourth International Conference on Intelligent Systems in Molecular Biology*, Menlo Park, AAAI Press, pp. 109-115,1996.

[Horton e Nakai 1997]

Horton, P. e Nakai, K. Better Prediction of Protein Cellular Localization Sites with the *k* Nearest Neighbors Classifier. In Gaasterland, T., Karp, P., Karplus, K.et al (eds.) *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, AAAI Press, pp. 147-152,1997.

[Jeffrey 1965]

Jeffrey, R. C. The Logic of Decision. New York, McGraw-Hill,1965.

[Kohonen 1982]

Kohonen, T. *Self-organized formation of topological feature maps*. Biological Cybernetics, vol. 43, pp. 59-69,1982.

[Lisboa et al. 2003]

Lisboa, F. O. S., Nicoletti, M. C. e Ramer, A. Experiencing Fuzzy Exemplar-Based Classifier Systems. In *Proceeding of the 12th IEEE International Conference on Fuzzy Systems*, St. Louis, USA, IEEE Press, pp. 90-95,2003.

[Mangasarian e Wolberg 1990]

Mangasarian, O. L. e Wolberg, W. H. *Cancer diagnosis via linear programming*. SIAM News, vol. 23(5), pp. 1-18,1990.

[McCulloch e Pitts 1943]

McCulloch, W. S. e Pitts, W. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics, vol. 5, pp. 115-137,1943.

[Mingers 1989]

Mingers, J. *An Empirical Comparison of Selection Measures for Decision-Tree Induction*. Machine Learning, vol. 3, pp. 319-342,1989.

[Minsky e Papert 1969]

Minsky, M. e Papert, S. Perceptrons. Cambridge, MA, MIT Press,1969.

[Mitchell 1998]

Mitchell, T. M. Machine Learning. New York, McGraw-Hill Companies Inc,1998.

[Mooney et al. 1989]

Mooney, R., Shavlik, J., Towell, G. e Gove, A. An experimental comparison of symbolic and connectionist learning algorithms. In Sridharan, N. S. (ed.) *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Miami, USA, Morgan Kaufmann, pp. 775-780,1989.

[Nicoletti 1994]

Nicoletti, M. C. *Ampliando o Aprendizado Indutivo de Máquina Através da Abordagens Proposicional e Relacional*. Tese (Doutorado), Instituto de Física de São Carlos, USP, São Carlos,1994.

[Nicoletti 1998/2000]

Nicoletti, M. C. *Investigação de modelos cooperativos simbólico-conexionistas de aprendizado indutivo de máquina*, Relatório Científico referente ao Projeto de Pesquisa - Fapesp,1998/2000.

[Nicoletti et al. 2000]

Nicoletti, M. C., Ramer, A. e Nicoletti, M. A. A symbolic neuro-fuzzy collaborative approach for inducing knowledge in a pharmacological domain. In *Proceedings of the 3th International Conference on Information Fusion*, Paris, France, pp. 4-10,2000.

[Nicoletti e Rocha 2000a]

Nicoletti, M. C. e Rocha, M. G. B. O Modelo KBANN. Relatório Técnico, UFSCar, São Carlos,2000a.

[Nicoletti e Rocha 2000b]

Nicoletti, M. C. e Rocha, M. G. B. O Modelo TREPAN. Relatório Técnico, UFSCar, São Carlos,2000b.

[Quinlan 1979]

Quinlan, J. R. Discovering rules by induction from large collections of examples. Expert systems in micro eletronic age. Michie, D. (ed.), Edinburgh University Press,1979.

[Quinlan 1986]

Quinlan, J. R. *Induction of decision trees*. Machine Learning, vol. 1(1), pp. 81-106,1986.

[Quinlan 1993]

Quinlan, J. R. C4.5: Programs for Machine Learning. Los Altos, CA, Morgan Kauffmann,1993.

[Quinlan 1996]

Quinlan, J. R. *Learning Decision Tree Classifiers*. ACM Computing Surveys, vol. 28(1), pp. 71-72,1996.

[Rosenblatt 1957]

Rosenblatt, F. The perceptron: a perceiving and recognizing automaton. Relatório Técnico, Ithaca, NY,1957.

[Rumelhart et al. 1986]

Rumelhart, D. E., Hinton, G. E. e Willians, R. J. Learning internal representation by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Rumelhart, D. E. e McClelland, J. L. (eds.). Cambridge, MIT Press, vol. 1, pp. 318-362,1986.

[Sachs 1984]

Sachs, L. Apllied Statistics: A Handbook of Techniques. New York, NY, Springer-Verlag,1984.

[Salzberg 1989]

Salzberg, S. L. *Learning with nested generalized exemplars*. Tese (Doutorado), Departament of Computer Science, Harvard University, Cambridge, MA, p. 194,1989.

[Salzberg 1991]

Salzberg, S. L. *A nearest hyperrectangle learning method*. Machine Learning, vol. 6, pp. 252-276,1991.

[Santos 1997]

Santos, F. O. *O uso de exemplares generalizados no aprendizado indutivo e sua extensão para domínios fuzzy*. Dissertação (Mestrado), Departamento de Computação, Universidade Federal de São Carlos, São Carlos, p. 109,1997.

[Shavlik 1994]

Shavlik, J. *Combining symbolic and neural learning*. Machine Learning, vol. 14, pp. 321-331,1994.

[Shaw e Gentry 1990]

Shaw, M. J. e Gentry, J. A. *Inductive Learning for Risk Classification*. IEEE Expert, pp. 47-53,1990.

[Shultz et al. 1994]

Shultz, T., Mareschal, D. e Schmidt, W. *Modeling Cognitive Development on Balance Scale Phenomena*. Machine Learning, vol. 16, pp. 59-88,1994.

[Silverman 1986]

Silverman, B. W. Density Estimation for Statistics and Data Analysis. London, Chapman and Hall,1986.

[Simon 1983]

Simon, H. A. *Why should machine learn?* Machine Learning, vol. 1,1983.

[Towell 1991]

Towell, G. *Symbolic knowledge and neural networks: insertion, refinement and extraction*. Tese (Doutorado), Computer Science Department, University of Wisconsin, Madison, p. 182,1991.

[Towell e Shavlik 1993]

Towell, G. e Shavlik, J. *Extracting refined rules from knowledge-based neural networks*. Machine Learning, vol. 13, pp. 71-101,1993.

[Tschichold-Gurman 1995a]

Tschichold-Gurman, N. Generation and Improvement of Fuzzy Classifiers with Incremental Learning using Fuzzy RuleNet. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, Nashville, Tennessee, pp. 466-470,1995a.

[Tschichold-Gurman 1995b]

Tschichold-Gurman, N. *RuleNet: a new knowledge based artificial neural network model with application examples in robotics*. Tese (Doutorado), ETH, Zurich, Switzerland,1995b.

[Tschichold-Gurman 1997]

Tschichold-Gurman, N. *The neural network model rulenet and its application to mobile robot navigation*. Fuzzy Sets and Systems, vol. 1(85), pp. 287-303,1997.

[Utgoff 1988a]

Utgoff, P. Perceptron Tree: A Case Study in Hybrid Concept Representations. Relatório Técnico,1988a.

[Utgoff 1988b]

Utgoff, P. Perceptron trees: A case study in hybrid concept representations. In Smith, R. G. e Mitchell, T. M. (eds.) *Proceedings of the 7th National Conference on Artificial Intelligence*, St. Paul, MN, Morgan Kaufmann, pp. 601-606,1988b.

[Utgoff 1989]

Utgoff, P. *Incremental Induction of Decision Trees*. Machine Learning, vol. 4, pp. 161-186,1989.

[Volpini et al. 2002]

Volpini, P., Figueira, L. B., Colafemina, J. F. e Roque, A. C. A neural network-based system for the diagnosis of central vestibular lesion. In Valafar, F. (ed.) *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, Las Vegas, Nevada, CSREA Press, pp. 29-32,2002.

[Weiss e Kapouleas 1989]

Weiss, S. M. e Kapouleas, I. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In Sridharan, N. S. (ed.) *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, USA, Morgan Kaufmann, pp. 781-787,1989.

[Wettschereck e Dietterich 1995]

Wettschereck, D. e Dietterich, T. G. *An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms*. Machine Learning, vol. 19(1), pp. 5-27,1995.

[Wilson e Martinez 1997]

Wilson, D. R. e Martinez, T. R. *Improved Heterogeneous Distance Functions*. Journal of Intelligence Research, vol. 6, pp. 1-34,1997.

[Yeang 2001]

Yeang, C. *Molecular Classification of Multiple Tumor Types*. Bioinformatics Discovery Notes, vol. 1(1), pp. 1-7,2001.

[Zadeh 1999]

Zadeh, L. *Advances in Fuzzy Theory*. Artificial Intelligence in Medicine, vol. 15, pp. 90-95,1999.

Apêndice A

Pseudocódigo do Sistema RuleNet

O pseudocódigo do procedimento RuleNet descrito neste anexo é uma proposta deste trabalho, uma vez que, as referências bibliográficas que propõem o modelo se limitam apenas a descrevê-lo em linguagem natural, sem qualquer preocupação em descreve-lo na forma algorítmica. Na descrição que segue usada a seguinte notação:

- um conjunto de N atributos Atrib: $\{Atrib_1,Atrib_2,...,Atrib_N\}$. Cada atributo está associado a uma dimensão do espaço onde as instâncias estão representadas.
- um conjunto de treinamento CT = $\{E_1,E_2,...,E_M\}$ com M instâncias de treinamento. Cada instância E_i, $1 \leq i \leq M$ é descrita por um vetor de dimensão N+1, de valores de atributos e uma classe associada, como segue:

  $E_i = [E_{i\_1}, E_{i\_2},..., E_{i\_N}, CLASSE]$ tal que $E_{i\_1}, E_{i\_2},..., E_{i\_N}$ são valores dos atributos $Atrib_1, Atrib_2, ..., Atrib_N$ respectivamente e CLASSE é a classe de E_i

A rede RuleNet construída pelo algoritmo RuleNet é representada pela lista [NE,NH,NO], onde:

1) NE = $[NE_1,NE_2,...,NE_N]$ representa a camada de entrada da rede. Cada

nó NE_i, $1 \leq i \leq N$ representa o nó que propaga o valor do i-ésimo atributo ($Atrib_i$) que descreve uma instância de treinamento. A função de cada um desses nós é apenas de servir como um *buffer* para um determinado valor de atributo.

2) $NH = [NH_1, NH_2, \ldots, NH_K]$ representa a camada intermediária a ser criada pelo algoritmo RuleNet. Inicialmente esse conjunto é vazio. A estrutura que representa cada um dos nós intermediários, notados por NH_i, $1 \leq i \leq K$, é:

NH_i = [[[NH_{i_1}, $NH_{i\_\lambda_{R_1}}$, $NH_{i\_\lambda_{L_1}}$],
[NH_{i_2}, $NH_{i\_\lambda_{R_2}}$, $NH_{i\_\lambda_{L_2}}$],
,. . . ,
[NH_{i_N}, $NH_{i\_\lambda_{R_N}}$, $NH_{i\_\lambda_{L_N}}$]],
NO_OUTPUT,
ATIVACAO
]

2.1) Cada uma das N listas com três elementos da forma [NH_{i_j}, $NH_{i\_\lambda_{R_j}}$, $NH_{i\_\lambda_{L_j}}$] $1 \leq j \leq N$ representa um estrutura unidimensional com as informações sobre os limites da dimensão j.

2.1.1) O primeiro valor da lista, NH_{i_j}, representa o valor de referência, que corresponde ao valor da dimensão j que é extraído da instância de treinamento.

2.1.2) O valor $NH_{i\_\lambda_{R_j}}$ corresponde à distância à direita do valor de referência.

2.1.3) O valor $NH_{i\_\lambda_{L_j}}$ corresponde à distância à esquerda do valor de referência

2.2) NO_OUTPUT é o nó de saída associado ao nó intermediário em questão, no caso NH_i.

2.3) ATIVACAO é o valor de ativação corrente.

3) $NO = [NO_1, NO_2, \ldots, NO_P]$ representa a camada de saída da rede. Cada nó NO_i, $1 \leq i \leq P$, caracteriza o i-ésimo nó de saída da rede. A estrutura

de cada um desses nós é:

> NO_i = [CLASSE,LISTA_NOS_CONECTADOS], onde
> LISTA_NOS_CONECTADOS=[[NH_d,w_d], [NH_f,w_f], ... ,[NH_q,w_q]].
> Cada elemento é uma lista com dois outros elementos. O primeiro indica o nó intermediário conectado ao nó de saída e o segundo, o peso dessaa conexão.
> CLASSE é a classe representada pelo nó.

Além disso, o pseudocódigo do algoritmo faz uso das seguintes funções pré-definidas:

- *classe*(NO_i): recupera a classe associada a um nó de saída, seu parâmetro;
- *classe*(E_i): recupera a classe associada à uma instância de treinamento, seu parâmetro;
- *no_output*(NH_i): recupera o nó de saída ao qual o nó intermediário NH_i está conectado. Essa informação faz parte da estrutura que define o nó intermediário NH_i;
- *ativacao*(NH_i): recupera o valor de ativação corrente do nó intermediário NH_i;
- *lista_nos_conectados*(NO_i): recupera a lista dos nós intermediários conectados a um nó de saída. Essa informação é parte da descrição de cada nó de saída;
- *add*(L,E) adiciona o elemento E à lista L;
- *valor_atributo*($Atrib_j$,E): recupera o valor do j-ésimo atributo do exemplo E.

```
procedure RuleNet(CT,N,M,[NE,NH,NO])
{CT: conjunto de treinamento
  N: número de atributos que descrevem as instâncias de treinamento
  M: número de instâncias presentes em CT }
begin
   NE := NH := NO := [ ]
   inicializa(N,NE)
   treina(CT,N,M,[NE,NH,NO])
end
```

```
procedure inicializa(N,NE)
{este procedimento cria a camada de entrada da rede RuleNet}
begin
   for  i := 1 to N do
     add(NE,NE_i)  {cria um nó de entrada referente ao i-ésimo atributo que
                    descreve o conjunto de treinamento}
end
```

```
procedure cria_intermediario(E_i,NH_k)
begin
   for j := 1 to N do
    begin
       NH_k_j := valor_atributo(Atrib_j,E_i)
       NH_k_λR_i := λ_default
       NH_k_λL_i := λ_default
    end
     ajusta_no_espaco(NH_k,classe(E_i))
    add(NH,NH_k)
end
```

```
procedure cria_output(E_i,NO_p)
begin
  NO_p := [ classe(E_i), [ ] ]
  add(NO,NO_p)
end
```

```
procedure treina(CT,N,M,[NE,NH,NO])
begin
  i := 1          {variavel que representa o indice de uma instancia de treinamento}
  k := 1          {variavel que representa o indice de um no intermediario}
  p := 1          {variavel que representa o indice de um no output}
  while i <= M do
    begin
      Lista_Nos_Ativados := [ ]
      propaga(Ei,Lista_Nos_Ativados,Classe)
      if Classe = null
         then  begin
                  cria_intermediario(Ei,NHk)
                  if existe_output(Ei)  then conecta(NHk,Ei)
                                        else begin
                                               cria_output(Ei,NOp)
                                               conecta(NHk,Ei)
                                               p := p + 1
                                             end
                  k := k + 1
                end
         else if classe(Ei) <> Classe
                 then begin
                        ajusta(Lista_Nos_Ativados,Ei)
                        cria_intermediario(Ei,NHk)
                        if existe_output(Ei)  then conecta(NHk,Ei)
                                              else begin
                                                     cria_output(Ei,NOp)
                                                     conecta(NHk,Ei)
                                                     p := p + 1
                                                   end
                        k := k + 1
                      end
      i := i + 1
    end
```

```
procedure propaga(Ei,Lista_Nos_Ativados,Classe)
begin
   Lista_Nos_Ativados := [ ]
   for j := 1 to N do
        NEj := valor_atributo(Atribj,Ei)
   for j := 1 to length(NH) do
        begin
           d := F_Soma_Intermediário(Ei,NHj)
          ativacao(NHj) := F_Ativação_Intermediário(d,NHj,Ei)
        end
   j := 1
   no_ativado := false
   while (j <= length(NO)) and not (no_ativado) do
        begin
            if F_Soma_Output(NOj) >= 0
             then begin
                  no_ativado := true
                        NH := lista_nos_conectados(NOj)
                        for t := 1 to length(NH) do
                              if ativacao(NHt) > 0
                              then add(Lista_Nos_Ativados,NHt)
                        Classe := classe(NOj)
                     end
            j := j + 1
        end
   if not no_ativado
     then Classe := null
end
```

```
fuction existe_output(Ei) : boolean
begin
   j := 1
   existe_output := false
   while (j <= length(NO)) and (not existe_output) do
        begin
            if classe(NOj) = classe(Ei)
```

```
            then existe_output := true
           j := j + 1
        end
end
```

```
procedure conecta(NH_k,E_i)
begin
  j := 1
 conectou_no := false
  while (j <= length(NO)) and (not conectou_no) do
      begin
        if classe(NO_j) = classe(E_i)
         then begin
                  no_output(NH_k) := NO_j
                 add(lista_nos_conectados(NO_j),[NH_k,4])
                 conectou_no := true
               end
        j := j + 1
      end
end
```

```
function F_Soma_Intermediario(E_i,NH_j) : integer
{Verifica se o exemplo E_i está "dentro" do hiper-retângulo definido pelo nó intermediário NH_j}
begin
    t := 1
    F_Soma_Intermediario := 1
    while (t <= N) and (F_Soma_Intermediario ≠ -1) do
        begin
            if (valor_atributo(Atrib_t,E_i) < (NH_j_t - NH_j_λ_L_t)) or (valor_atributo(Atrib_t,E_i) >
              (NH_j_w_t + NH_j_λ_R_t))
             then F_Soma_Intermediário := -1
           t := t + 1
        end
end
```

```
function F_Ativação_Intermediario(d,Ei,NHj) : real
begin
    max := |NHj_1 – valor_atributo(Atrib1,Ei_1)|
    for t := 2 to N do
        begin
            if |NHj_t – valor_atributo(Atribt,Ei_t)| > max
              then max := |NHj_t – valor_atributo(Atribt,Ei)|
        end
    F_Ativação_Intermediario := d * max
end
```

```
function F_Soma_Output(NOj) : integer
begin
    NH := lista_nos_conectados(NOj)
    max := ativacao(LH1) * w1
    for t := 2 to lenght(NH) do
        begin
            if (ativacao(NHt) * NOj_t) > max
              then max := ativacao(NHt) * wt
        end
    F_Soma_Output := max
end
```

```
procedure ajusta(Lista_Nos_Ativados,Ei)
{Lista_Nos_Ativados = [NH1,NH2,...,NHNE]    }
begin
  for j := 1 to length(Lista_Nos_Ativados) do
    begin
     determina_dimensao_ajuste(NHj,Ei,dim,param)
      if param = R
        then NHj_λRdim := valor_atributo(Atribdim,Ei) – NHj_dim – 0,1
        else NHj_λLdim := NHj_dim – valor_atributo(Atribdim,Ei) – 0,1
     end
end
```

```
procedure determina_dimensao_ajuste(NH_j,E_i,dim,param)
begin
  dim=0
  encontrou_distancia := false
  while (not encontrou_distancia)
    begin
      dim := dim + 1
      if valor_atributo(Atrib_dim,E_i) > NH_j_dim
        then begin
               param := R
               distancia := (NH_j_dim + NH_j_λ_R_dim) – valor_atributo(Atrib_dim,E_i)
               encontrou_distancia := true
             end
      else if valor_atributo(Atrib_dim,E_i) < NH_j_dim
             then begin
                    param := L
                    distancia := valor_atributo(Atrib_dim,E_i) – (NH_j_dim – NH_j_λ_L_dim)
                    encontrou_distancia := true
                  end
    end
  for d := 2 to N do
    begin
      if valor_atributo(Atrib_d,E_i) > NH_j_d
        then begin
               distd := (NH_j_d + NH_j_λ_R_d) – valor_atributo(Atrib_d,E_i)
               if distd < distancia
                 then begin
                        param := R
                        distancia :=distd
                        dim := d
                      end
             end
      else if valor_atributo(Atrib_d,E_i) < NH_j_d
             then begin
```

```
                    distd := valor_atributo(Atrib_d,E_i) – (NH_j_d – NH_j_λL_d)
                    if distd<distancia
                       then begin
                                param := L
                            distancia := distd
                            dim := d
                            end
                end
    end
end
```

```
procedure ajusta_no_espaco(NH_k,classe(E_i))
begin
   for s := 1 to length(NH) do
     begin
        if classe(no_output(NH_s)) = classe(E_i)
          then begin
                dim := 0
                sobreposto := false
                while (not sobreposto) and (dim<M) do
                    begin
                      dim := dim + 1
                       if (NH_k_dim < NH_s_dim – NH_s_λL_dim) and (NH_k_dim + NH_k_λR_dim >=
                          NH_s_dim – NH_s_λL_dim)
                          then begin
                                  lambda := R
                               distancia := (NH_s_dim – NH_s_λL_dim) – NH_k_dim
                               sobreposto := true
                               end
                        else if (NH_k_dim > NH_s_dim – NH_s_λR_dim) and (NH_k_dim –
                                  NH_k_λL_dim <= NH_s_dim + NH_s_λR_dim)
                               then begin
                                       lambda := L
                                    distancia := NH_k_dim – (NH_s_dim + NH_s_λR_dim)
```

```
                                    sobreposto  := true
                                    end
                end
            d := dim + 1
            while (d < M) and (sobreposto) do
                begin
                sobreposto := false
                  if (NH_k_d < NH_s_d – NH_s_λL_d) and (NH_k_d + NH_k_λR_d >= NH_s_d –
                     NH_s_λL_d)
                     then begin
                          sobreposto  := true
                             dist := (NH_s_d – NH_s_λL_d) – NH_k_d
                   if dist < distancia
                          then begin
                                   lambda := R
                               distancia :=  dist
                               dim := d
                              end
                       end
                 else if (NH_k_d > NH_s_d – NH_s_λR_d) and (NH_k_d – NH_k_λL_d <= NH_s_d
                        + NH_s_λR_d)
                        then begin
                                 sobreposto  := true
                                    dist := NH_k_dim – (NH_s_dim + NH_s_λR_dim)
                                 if dist < distancia
                                    then begin
                                             lambda := L
                                         distancia :=  dist
                                         dim := d
                                        end
                                 end
                 d := d + 1
                end
        if sobreposto
```

then begin

if lambda = R

then $NH_{k\_\lambda_{R_{dim}}}$:= distancia – 0,01

else $NH_{k\_\lambda_{L_{dim}}}$:= distancia – 0,01

end

Anexo A

O Sistema Fuzzy NGE

O algoritmo Fuzzy NGE é uma versão *fuzzy* do algoritmo NGE original que aceita exemplos de treinamento descritos por atributos que possuem valores *fuzzy* (i.e. cada valor de atributo é tratado como um conjunto *fuzzy*) e uma classe crisp associada. Cada valor *fuzzy* pertencente ao conjunto de treinamento está associado a uma função de pertinência triangular. Geralmente, após o inicio do processo de aprendizado, a tendência é que, com a generalização, os valores fuzzy se tornem intervalos *fuzzy* (representados pela função de pertinência trapezoidal).

O processo de inicialização do Fuzzy NGE é o mesmo do NGE original, onde um número de *s* exemplos especificado pelo usuário é transformado em exemplares *fuzzy*. Após a inicialização, e com a expressão inicial do conceito formada pelos *s* exemplares, o sistema inicia o treinamento. Cada novo exemplo *fuzzy* é, então, comparado com cada um dos exemplares *fuzzy* existentes até então, buscando encontrar os dois exemplares mais próximos do novo exemplo em questão. Feito isso, acontece o processo de escolha de qual exemplar será generalizado. Se nenhum dos dois exemplares se qualifica, então o novo exemplo *fuzzy* se torna um exemplar *fuzzy*.

Durante o treinamento, uma métrica de similaridade é utilizada para determinar o exemplar *fuzzy* "mais próximo" do novo exemplo de treinamento. Uma das funções de similaridade, extraída de [Zadeh 1999] utilizadas pelo

Fuzzy NGE emprega o conceito de cardinalidade definida como:

Se $A(\mu_A(x_1), \mu_A(x_2), \ldots, \mu_A(x_n))$ é um conjunto *fuzzy* representado por meio de um vetor, sua cardinalidade c(A) é simplesmente a soma de seus valores de pertinência: $c(A)=\Sigma\mu_A(x_i)$. Em [Zadeh 1999] é apresentado um teorema que se A e B são dois conjuntos *fuzzy* então

$$\text{similar}(A,B) = \frac{c(A \cup B)}{c(A \cap B)} \qquad (A.1)$$

i.e. o grau de similaridade entre A e B é representado por um valor real entre 0 e 1.

Note que a similaridade será 0 quando as funções de pertinência *fuzzy* envolvidas são disjuntas entre si. Com o intuito de construir uma métrica de similaridade mais flexível, a equação (A.1) é estendida de forma a "tolerar" conjuntos disjuntos até um certo ponto. Logo, o sistema Fuzzy NGE permite que o usuário determine, a partir do parâmetro τ, a distância máxima "tolerável" entre conjuntos disjuntos. A Figura A.1 mostra a funcionalidade do parâmetro τ na métrica de similaridade.

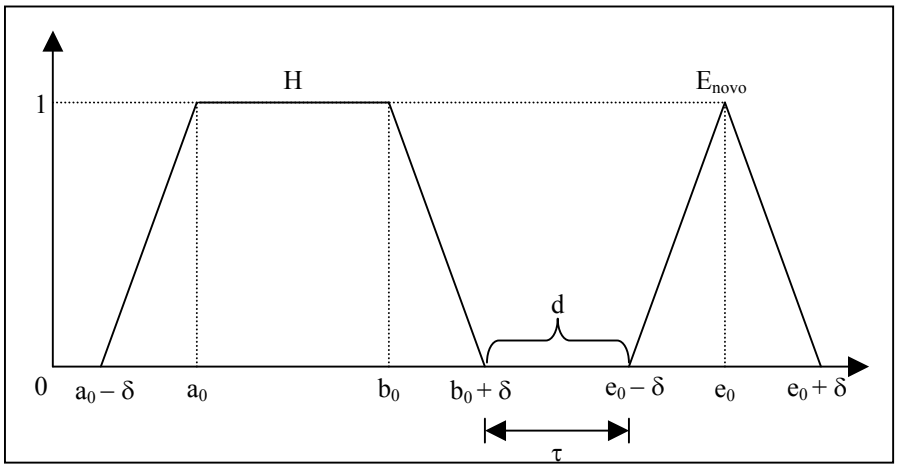


Figura A.1: Métrica de similaridade estendida para conjuntos disjuntos.

Conjuntos disjuntos que apresentam distância maior que τ tem seu valor de similaridade igual a 0. Ao contrário, conjuntos disjuntos que apresentam distância, no máximo, τ, o sistema a similaridade como sendo pequeno valor

positivo, dado pelo inverso da sua distância.

Se, durante a avaliação de similaridade entre os conjuntos do exemplo de treinamento e de cada exemplar, algum atributo apresentar similaridade 0, então o exemplar em questão não será considerado um potencial candidato para a generalização. Caso esta mesma situação ocorrer durante a fase de classificação, então o exemplo em questão é considerado não-classificado.

O próximo passo após o cálculo da similaridade entre E_{novo} e H é combinar, individualmente, todas as similaridades atributo-para-atributo num único valor. A função que retorna a similaridade atributo-para-atributo entre E_{novo} e H, é dada por:

$$\text{atas}(E_{novo}, H) = \frac{\sum_{i=1}^{n} \text{similar}(\text{valor\_atrib}_i(E_{novo}), \text{valor\_atrib}_i(H))}{n} \qquad \text{(A.2)}$$

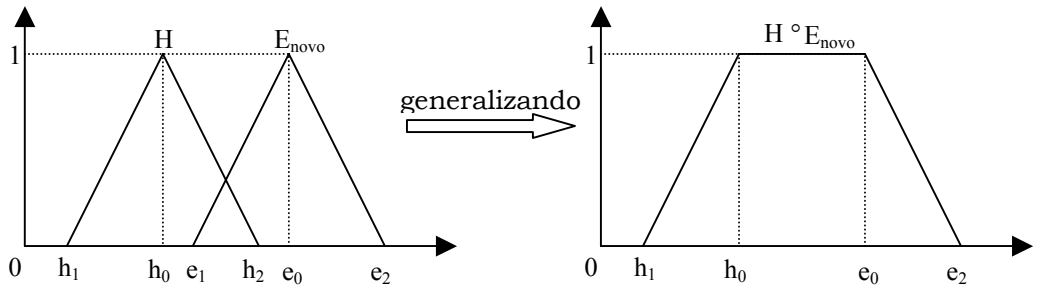
onde a função similar pode ser calculada a partir da equação (A.1).

Finalmente, é necessário ponderar o valor de atas com o peso do exemplar H em questão utilizando a seguinte função:

$$\text{tws}(E_{novo}, H) = \text{atas}(E_{novo}, H) \times \text{weight}_H \qquad \text{(A.3)}$$

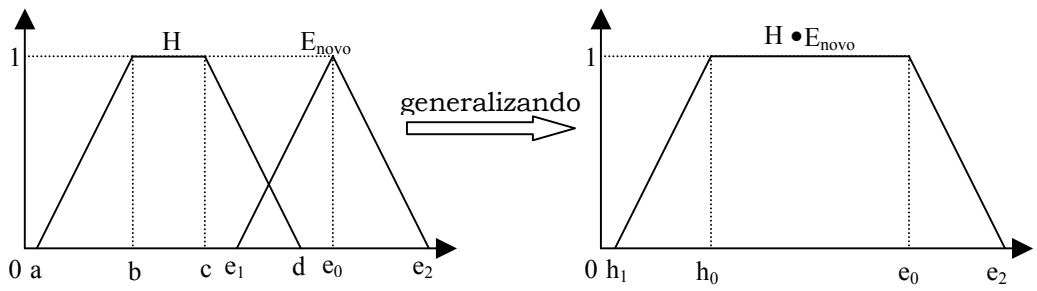
que é calculada para cada exemplar existente.

O processo de generalização acontece após a escolha dos dois exemplares mais similares a E_{novo}, e identificados como $H_{prósimo1}$ e $H_{próximo2}$. Dependendo do resultado obtido a partir da comparação das classes crisp, $H_{prósimo1}$ ou $H_{próximo2}$ podem ser generalizados, ou então E_{novo} se torna um novo exemplar *fuzzy*. Dependendo do tipo de função de pertinência (triangular ou trapezoidal) que representa cada atributo, dois tipos de generalização podem ocorrer, a partir da aplicação dos operadores ° (Figura A.2) e • (Figura A.3), respectivamente.



% definição do operador ° entre dois conjuntos fuzzy
% valores em H e E_{novo}
% situação onde $e_0 > h_0$
$h_1 \leftarrow h_0 - \delta$
$h_2 \leftarrow h_0 + \delta$
$e_1 \leftarrow e_0 - \delta$
$e_2 \leftarrow e_0 + \delta$
if $x < h_0$ então $\mu_{H \circ E_{novo}}(x) = x + y(h_1 - h_0) - h_1$
if $x > e_0$ então $\mu_{H \circ E_{novo}}(x) = x - y(e_0 - e_2) - e_2$
if $h_0 \leq x \leq e_0$ então $\mu_{H \circ E_{novo}}(x) = 1$

Figura A.2: Operador ° para generalização de conjuntos fuzzy.



% definição do operador • entre dois conjuntos fuzzy
% valores em H e E_{novo}
% situação onde $e_0 > c$ ($c > b$)
$e_1 \leftarrow e_0 - \delta$
$e_2 \leftarrow e_0 + \delta$
if $x < b$ então $\mu_{H \bullet E_{novo}}(x) = x + y(a - b) - a$
if $x > e_0$ então $\mu_{H \bullet E_{novo}}(x) = x - y(e_0 - e_2) - e_2$
if $h_0 \leq x \leq e_0$ então $\mu_{H \circ E_{novo}}(x) = 1$

Figura A.3: Operador • para generalização de conjuntos fuzzy.

A expressão do conceito após a fase de aprendizado é representada por

todos os exemplares existentes e pode ser utilizada para a classificação de novos exemplos. Na fase de classificação, a classe do exemplar mais similar ao exemplo é tida como a classe do exemplo.