Extração de conhecimento de redes neurais artificiais utilizando sistemas de aprendizado simbólico e algoritmos genéticos

Claudia Regina Milaré

Extração de conhecimento de redes neurais artificiais utilizando sistemas de aprendizado simbólico e algoritmos genéticos

Claudia Regina Milaré

Orientador: *Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho*
Co-Orientadora: *Prof^a. Dr^a. Maria Carolina Monard*

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

USP - São Carlos
Abril/2003

Resumo

Em Aprendizado de Máquina — AM — não existe um único algoritmo que é sempre melhor para todos os domínios de aplicação. Na prática, diversas pesquisas mostram que Redes Neurais Artificiais — RNAs — têm um *bias* indutivo apropriado para diversos domínios. Em razão disso, RNAs têm sido aplicadas na resolução de vários problemas com desempenho satisfatório.

Sistemas de AM simbólico possuem um *bias* indutivo menos flexível do que as RNAs. Enquanto que as RNAs são capazes de aprender qualquer função, sistemas de AM simbólico geralmente aprendem conceitos que podem ser descritos na forma de hiperplanos. Por outro lado, sistemas de AM simbólico representam o conceito induzido por meio de estruturas simbólicas, as quais são geralmente compreensíveis pelos seres humanos. Assim, sistemas de AM simbólico são preferíveis quando é essencial a compreensibilidade do conceito induzido. RNAs carecem da capacidade de explicar suas decisões, uma vez que o conhecimento é codificado na forma de valores de seus pesos e *thresholds*. Essa codificação é difícil de ser interpretada por seres humanos.

Em diversos domínios de aplicação, tal como aprovação de crédito e diagnóstico médico, prover uma explicação sobre a classificação dada a um determinado caso é de crucial importância. De um modo similar, diversos usuários de sistemas de AM simbólico desejam validar o conhecimento induzido, com o objetivo de assegurar que a generalização feita pelo algoritmo é correta. Para que RNAs sejam aplicadas em um maior número de domínios, diversos pesquisadores têm proposto métodos para extrair conhecimento compreensível de RNAs.

As principais contribuições desta tese são dois métodos que extraem conhecimento simbólico de RNAs. Os métodos propostos possuem diversas vantagens sobre outros métodos propostos previamente, tal como ser aplicáveis a qualquer arquitetura ou algoritmo de aprendizado de RNAs supervisionadas. O primeiro método proposto utiliza sistemas de AM simbólico para extrair conhecimento de RNAs, e o segundo método proposto estende o primeiro, combinado o conhecimento induzido por diversos sistemas de AM simbólico por meio de um Algoritmo Genético — AG.

Os métodos propostos são analisados experimentalmente em diversos domínios de aplicação. Ambos os métodos são capazes de extrair conhecimento simbólico com alta fidelidade em relação à RNA treinada. Os métodos propostos são comparados com o método TREPAN, apresentando resultados promissores. TREPAN é um método bastante conhecido para extrair conhecimento de RNAs.

Abstract

In Machine Learning — ML — there is not a single algorithm that is the best for all application domains. In practice, several research works have shown that Artificial Neural Networks — ANNs — have an appropriate inductive bias for several domains. Thus, ANNs have been applied to a number of data sets with high predictive accuracy.

Symbolic ML algorithms have a less flexible inductive bias than ANNs. While ANNs can learn any input-output mapping, *i.e.* ANNs have the universal approximation property, symbolic ML algorithms frequently learn concepts describing them using hyperplanes. On the other hand, symbolic algorithms are needed when a good understating of the decision process is essential, since symbolic ML algorithms express the knowledge induced using symbolic structures that can be interpreted and understood by humans. ANNs lack the capability of explaining their decisions since the knowledge is encoded as real-valued weights and biases of the network. This encoding is difficult to be interpreted by humans.

In several application domains, such as credit approval and medical diagnosis, providing an explanation related to the classification given to a certain case is of crucial importance. In a similar way, several users of ML algorithms desire to validate the knowledge induced, in order to assure that the generalization made by the algorithm is correct. In order to apply ANNs to a larger number of application domains, several researches have proposed methods to extract comprehensible knowledge from ANNs.

The primary contribution of this thesis consists of two methods that extract symbolic knowledge, expressed as decision rules, from ANNs. The proposed methods have several advantages over previous methods, such as being applicable to any architecture and supervised learning algorithm of ANNs. The first method uses standard symbolic ML algorithm to extract knowledge from ANNs, and the second method extends the first method by combining the knowledge induced by several symbolic ML algorithms through the application of a Genetic Algorithm — GA.

The proposed methods are experimentally analyzed in a number of application domains. Results show that both methods are capable to extract symbolic knowledge having high fidelity with trained ANNs. The proposed methods are compared with TREPAN, showing promising results. TREPAN is a well known method to extract knowledge from ANNs.

Aos meus pais,
Idio e Armelinda.

Aos meus irmãos,
Márcia e Idinho.

À minha avó,
Idalina.

Agradecimentos

Faz nove anos que vim para São Carlos estudar na USP. Durante esse tempo convivi com muitas pessoas, fiz muitas amizades, aprendi muitas coisas. Agradeço a Deus por todos esses anos e por essa oportunidade.

As minhas palavras de agradecimento às pessoas que me ajudaram e apoiaram neste trabalho não conseguirão expressar todo o meu sentimento de gratidão, mas eu vou tentar!

O André, recém chegado da Inglaterra, foi meu professor de IA, meu orientador de mestrado e meu orientador de doutorado. Agradeço ao André por me orientar todos esses anos com muita paciência e compreensão. Confesso que nem sempre eu entendia sua letra e acompanhava seu raciocínio, mas o André, como grande mestre que é, não se importava em repetir uma, duas, três vezes.

Gostaria de agradecer também à Carolina por me co-orientar neste trabalho. Suas explicações e opiniões foram muito valiosas.

Aprendi muitas coisas com o André e com a Carolina, além do conhecimento técnico. O que mais me encanta, é a simplicidade e a humildade dessas duas pessoas. Foi um privilégio ter convivido e tê-los como orientadores durante esses anos. Muitíssimo obrigada!

O André, a Carolina e a professora Solange fazem com que o LABIC seja um laboratório especial. As pessoas que trabalham lá criam vínculos muito fortes, e quando o trabalho termina é difícil ir embora. Nunca me esquecerei do esforço destes três professores para que seus alunos sempre participem dos congressos e eventos da área.

Quero agradecer aos meus queridos pais, Idio e Armelinda, que sempre nos proporcionaram, a mim e aos meus irmãos, oportunidades que eles nunca tiveram. Os nossos estudos sempre foram prioridade para eles. Obrigada pai e mãe.

Agradeço toda minha família pelo enorme apoio que me deram, em especial à Vó Idalina, Márcia, Idinho, Osvaldo, Grazi, Elisa, Bruno, Maria e Tia Nilce.

Dentre tantas coisas boas que aconteceram durante o doutorado, encontrar o Gustavo foi muito especial. Gostaria de agradecer ao Gustavo por toda a ajuda. Suas contribuições enriqueceram demais este trabalho. Obrigada também por todos os momentos que passamos juntos.

Quero agradecer também os pais do Gustavo, Joselito e Margarida, pelo carinho com que me tratam e por torcerem para que tudo desse certo.

Conheci a Andréia em 1990 na UEL no dia da matrícula, desde então somos amigas, acho que irmãs. Agradeço à Andréia por nossa amizade e pela força que ela sempre me deu nos momentos difíceis.

Quero agradecer ao pessoal da biblioteca e em especial às meninas da pós-graduação Beth, Laura, Ana Paula e também à Marília que sempre me atenderam com a maior boa vontade e fazem com que tudo funcione direitinho.

Às amigas do LABIC Claudinha, Jaque, Patrícia, Cris, Huei e Katti pela amizade e por tantos momentos legais que passamos juntas. E também aos amigos Marcos Geromini, Walter, Edson, Valmir, Daniel, Kaminski, José Flávio, Ronaldo, Pila, Gedson, Marcos Paula e José Augusto.

Um agradecimento especial ao Ronaldo que forneceu a biblioteca da sintaxe padrão $\mathcal{P}BM$ para que eu utilizasse no trabalho.

Não poderia deixar de lembrar aqui da Alice e da Sofia, elas são muito fofas.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

AGs . . . . . Algoritmos Genéticos
ANNs . . . . . *Artificial Neural Networks*
AM . . . . . Aprendizado de Máquina
DLE . . . . . *Discover Learning Environment*
DOL . . . . . *Discover Object Library*
DSX . . . . . *Discover Dataset Sintax*
GA . . . . . *Genetic Algorithm*
IA . . . . . Inteligência Artificial
LABIC . . . . . Laboratório de Inteligência Computacional
ML . . . . . *Machine Learning*
MLP . . . . . *Multi-Layer Perceptron*
Perl . . . . . *Practical Extraction and Report Language*
RNAs . . . . . Redes Neurais Artificiais
SNNS . . . . . *Stuttgart Neural Network Simulator*
SVM . . . . . *Support Vector Machines*
TREPAN . . . . . *TREes PArroting Networks*
UCI . . . . . Universidade da Califórnia em Irvine

Capítulo 1

Introdução

1.1 Considerações Iniciais

Redes Neurais Artificiais — RNAs — têm sido aplicadas na solução de problemas de diversas áreas, como comércio, indústria, engenharia, medicina, economia, genética, entre outras, o que confirma sua relevância na resolução de problemas práticos. De fato, RNAs possuem algumas características que justificam esse sucesso, entre elas destacam-se:

- a aquisição automática de conhecimento sobre um determinado domínio de problema, por meio de um processo de treinamento;
- o armazenamento do conhecimento adquirido, o qual é armazenado em uma RNA de forma compacta, ou seja, na forma numérica, e pode ser rapidamente acessado e utilizado quando necessário;
- a consistência do funcionamento das RNAs na presença de informações incorretas.

Juntamente com essas características, RNAs têm geralmente obtido bom desempenho quando utilizadas em uma grande variedade de aplicações (Devogelaere, Rijckaert, Leon & Lemus 2002; Hall 1992; LeCun, Boser, Denker, Henderson, Howard, Hubbard & Jackel 1990; Moreno, Sebastian, Fernandez & Caballero 1998; Thrun 1994). Entretanto, para várias dessas aplicações é importante não apenas o desempenho obtido, mas também a facilidade do usuário compreender como a rede chega às suas decisões. Essa é justamente uma das principais críticas às RNAs: a sua incapacidade de explicar como e porque a rede gera suas respostas.

É cada vez mais claro que o potencial das RNAs não poderá ser completamente explorado enquanto não houver um mecanismo que explique suas decisões, ou um mecanismo que faça com que uma RNA treinada seja compreensível aos usuários para que possa ser validada. Essa necessidade é essencial se as redes forem utilizadas em sistemas em que a segurança na operação é um aspecto importante. Esse é o caso de um grande número de problemas como, por exemplo, sistemas de:

- controle de usinas nucleares;
- navegação de aeronaves;
- diagnóstico médico;
- auxílio a cirurgias médicas;
- diagnóstico de falhas mecânicas.

O tema deste trabalho está centrado nesse aspecto de RNAs, propondo soluções para o seguinte problema:

> *Dado um modelo produzido por um sistema de aprendizado, no caso* RNAs*, e descrito em uma linguagem de difícil compreensão para a maioria dos usuários, como reapresentá-lo em uma linguagem que facilite sua compreensão para o usuário?*

Neste capítulo é fornecida uma visão geral do trabalho. O capítulo está organizado da seguinte forma: na Seção 1.2 são descritos alguns conceitos sobre Aprendizado de Máquina Indutivo que serão utilizados neste e nos próximos capítulos; na Seção 1.3 é descrito o objetivo do trabalho; na Seção 1.4 são apresentados brevemente os métodos propostos neste trabalho para extração de conhecimento compreensível de RNAs; na Seção 1.5 é descrito o ambiente de extração de conhecimento — *NNRules* — que, além de implementar os métodos propostos, dá suporte à realização de experimentos para avaliar esses métodos; e por fim, na Seção 1.6 é descrita a organização do trabalho.

1.2 Aprendizado de Máquina Indutivo

Aprendizado de Máquina — AM — é uma subárea de Inteligência Artificial — IA — que estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e

novos meios de organizar o conhecimento já existente. O principal objetivo de AM é o estudo e a modelagem computacional dos processos de aprendizado. Processos de aprendizado incluem a aquisição de novos conhecimentos, o desenvolvimento das habilidades motoras e cognitivas por meio da instrução e prática, a organização de novos conhecimentos em geral ou formas efetivas de representar o conhecimento, e a descoberta de novos fatos e teorias por meio de observações e experimentações (Mitchell 1997b; Monard & Baranauskas 2003).

Em um processo de aprendizado, conhecimentos prévios são utilizados para obter novos conhecimentos. Esses novos conhecimentos são então lembrados para serem utilizados posteriormente. O aprendizado de um novo conceito pode ser realizado de várias maneiras, uma delas é a *indução*. Indução é a forma de inferência lógica que permite que conclusões gerais sejam obtidas a partir de exemplos particulares. Ela é caracterizada como o raciocínio que parte do específico para o geral, do particular para o universal, da parte para o todo. No aprendizado por indução, um conceito é adquirido por meio de inferências indutivas sobre os fatos apresentados, desse modo, os modelos gerados por inferência indutiva podem ou não preservar a verdade.

O aprendizado indutivo é efetuado a partir do raciocínio sobre exemplos fornecidos por um processo externo ao aprendiz. Em AM, o aprendiz é um sistema computacional freqüentemente denominado *sistema de aprendizado*, ou *algoritmo de aprendizado*, ou simplesmente *indutor*. Um sistema de aprendizado é um sistema computacional que toma decisões baseado em experiências acumuladas contidas em casos resolvidos com sucesso (Weiss & Kulikowski 1991).

O aprendizado indutivo pode ser dividido em aprendizado supervisionado e não supervisionado, descritos a seguir.

Aprendizado Supervisionado

No aprendizado supervisionado é fornecido ao sistema de aprendizado um conjunto de exemplos $E = \{E_1, E_2, \ldots E_N\}$, sendo que cada exemplo $E_i \in E$ possui um rótulo associado. Esse rótulo define a *classe* a qual o exemplo pertence. Mais formalmente, pode-se dizer que cada exemplo $E_i \in E$ é uma tupla

$$E_i = (\vec{x_i}, y_i) \tag{1.1}$$

na qual $\vec{x_i}$ é um vetor de valores que representam as características, ou *atributos* do exemplo E_i, e y_i é o valor da classe desse exemplo. O objetivo do aprendizado

supervisionado é induzir um mapeamento geral dos vetores $\vec{x}$ para valores y. Portanto, o sistema de aprendizado deve construir um modelo, $y = f(\vec{x})$, de uma função desconhecida, f, também chamada de *função conceito*[1], que permite predizer valores y para exemplos previamente não vistos.

Entretanto, o número de exemplos utilizados para a criação do modelo não é, na maioria dos casos, suficiente para caracterizar completamente essa função f. Na realidade, os sistemas de aprendizado são capazes de induzir uma função **h** que aproxima f, ou seja, $\mathbf{h}(\vec{x}) \approx f(\vec{x})$. Nesse caso, **h** é chamada de *hipótese* ou *modelo* sobre a função conceito f.

Aprendizado Não Supervisionado

No aprendizado não supervisionado é fornecido ao sistema de aprendizado um conjunto de exemplos E, no qual cada exemplo consiste somente de vetores $\vec{x}$, não incluindo a informação sobre a classe y. O objetivo no aprendizado não supervisionado é construir um modelo que procura por regularidades nos exemplos, formando agrupamentos ou *clusters* de exemplos com características similares.

Assim, um *conjunto de exemplos* ou *conjunto de dados* $E = \{E_1, E_2, \ldots E_N\}$ é um conjunto de vetores $\vec{x_1}, \ldots \vec{x_N}$, com ou sem a classe associada y. Na Tabela 1.1 na próxima página é mostrada a forma geral de um conjunto de exemplos E com N exemplos e M atributos. Essa tabela está no formato *atributo-valor*[2], o qual é utilizado como entrada pela maioria dos algoritmos de aprendizado. Na forma atributo-valor as colunas A_1, $\ldots A_M$ da tabela representam os diferentes atributos, e as linhas $E_1, \ldots E_N$ os diferentes exemplos. Assim, a linha i na Tabela 1.1 refere-se ao *i-ésimo* exemplo e a entrada x_{ij} refere-se ao valor do *j-ésimo* atributo A_j do exemplo i, ou seja, $\vec{x_i} = (x_{i1}, x_{i2}, \ldots x_{iM})$. O atributo Y é o atributo que assume os valores da classe de cada exemplo E_i. Esse atributo é freqüentemente denominado de *atributo classe* e, como já mencionado, ele deve estar presente em conjuntos de dados para aprendizado supervisionado.

Em aprendizado supervisionado, o atributo classe Y pode ser um atributo qualitativo que assume um conjunto de valores discretos $C = \{C_1, C_2, \ldots C_{Ncl}\}$, ou um atributo quantitativo que assume um conjunto de valores reais.

No primeiro caso, assumindo que os vetores $\vec{x}$ correspondem a pontos em um espaço n-dimensional $\mathbb{R}^n$, o objetivo do aprendizado é encontrar uma função **h** que aproxima a

[1] *Concept function.*
[2] Também chamada de tabela *flat*.

	A_1	A_2	$\cdots$	A_M	Y
E_1	x_{11}	x_{12}	$\cdots$	x_{1M}	y_1
E_2	x_{21}	x_{22}	$\cdots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\ldots$	x_{NM}	y_N

Tabela 1.1: Conjunto de exemplos no formato atributo-valor.

função $f : \mathbb{R}^n \rightarrow C$. Nesse caso, a hipótese **h** é denominada *classificador*, e a tarefa de aprendizado é denominada *classificação*.

No segundo caso, o atributo classe é quantitativo, o qual pode assumir um conjunto de valores reais. O objetivo do aprendizado é encontrar uma função **h** que aproxima a função $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Nesse caso, a hipótese **h** é denominada *regressor*, e a tarefa de aprendizado é denominada *regressão*.

Um fator importante em aprendizado indutivo é que a hipótese **h**, induzida por um sistema de aprendizado, seja a mais precisa possível. Em aprendizado indutivo supervisionado isso significa que essa hipótese seja capaz de predizer corretamente o valor y_i para um vetor $\vec{x_i}$. Essa tarefa é relativamente simples para exemplos que foram utilizados durante a indução da hipótese. Entretanto, o interesse está no desempenho da hipótese induzida para exemplos nunca vistos.

Um outro fator importante a ser considerado é a compreensibilidade da hipótese induzida, ou seja, se a linguagem em que a hipótese **h** é representada é de fácil compreensão para os seres humanos.

Na Figura 1.1 na página seguinte é mostrada a hierarquia do aprendizado indutivo previamente descrita, sendo que o foco deste trabalho é sobre aprendizado indutivo supervisionado para problemas de classificação.

Dentro da área de AM foram propostos diferentes paradigmas capazes de aprender a partir de um conjunto de exemplos. Um requisito básico para todos os paradigmas de AM supervisionado é que o conceito a ser aprendido deve estar relacionado com casos observados, isto é, exemplos, e cada exemplo deve estar rotulado com a classe a qual pertence. Os paradigmas de aprendizado mais conhecidos são brevemente descritos a seguir.

Paradigma Simbólico

Os sistemas de AM simbólico buscam aprender construindo representações simbólicas de um conceito por meio da análise de exemplos e contra-exemplos desse conceito.

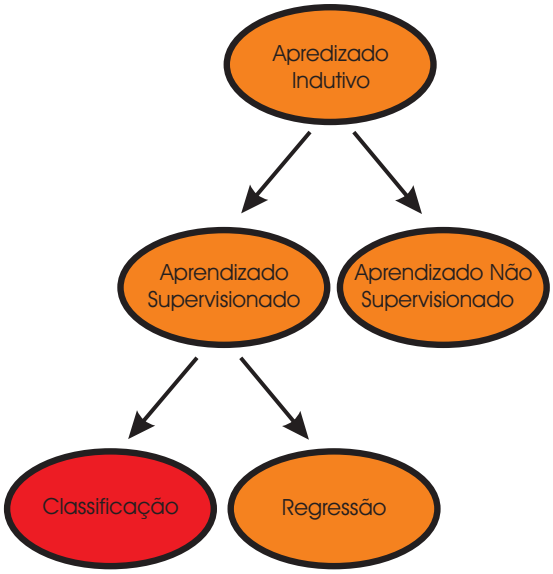

Figura 1.1: Hierarquia do aprendizado indutivo.

As representações simbólicas assumem tipicamente a forma de alguma expressão lógica: árvores de decisão, regras de decisão ou redes semânticas. Uma característica importante das representações simbólicas é que elas geralmente são compreensíveis pelos humanos.

Entre as representações simbólicas mais estudadas atualmente podem ser citadas árvores e regras de decisão. É atribuído a Morgan & Messenger (1973) o desenvolvimento original do programa para a indução de árvores de decisão. O método de indução de árvores de decisão a partir de dados empíricos, conhecido como *particionamento recursivo*, foi estudado por pesquisadores das áreas de IA e Estatística. Os sistemas ID3 (Quinlan 1986) e C4.5 (Quinlan 1987b) para indução de árvores de decisão representaram uma importante contribuição para as pesquisas em IA. É interessante observar que sistemas de árvores de classificação e regressão (Breiman, Friedman, Olshen & Stone 1984) foram desenvolvidos independentemente por estatísticos durante praticamente o mesmo período que o ID3, no final dos anos 70.

Os trabalhos com indução de regras de decisão surgiram com a simples tradução das árvores de decisão para regras, com a poda realizada sobre as regras. Tal abordagem foi proposta no trabalho de Quinlan (1987a). Posteriormente, foram criados métodos que induziam regras diretamente a partir dos dados. Um exemplo desse trabalho pode ser encontrado em (Michalski, Mozetic, Hong & Lavrac 1986). Um excelente levantamento dos principais sistemas indutores de regras de decisão pode ser encontrado em (Fürnkranz 1999).

No Capítulo 2 são descritos em maiores detalhes os sistemas de AM simbólico que induzem árvores de decisão e regras de decisão, por serem utilizados no desenvolvimento deste trabalho.

Paradigma Estatístico

Pesquisadores da área de estatística têm criado diversos métodos de classificação e regressão, muitos deles semelhantes aos métodos empregados pela comunidade de AM. Por exemplo, o método CART (Breiman, Friedman, Olshen & Stone 1984), um sistema muito conhecido para construir árvores de decisão, foi desenvolvido por estatísticos. Como regra geral, técnicas estatísticas tendem a focar tarefas em que todos os atributos têm valores contínuos ou ordinais. Muitos deles também são paramétricos, assumindo alguma forma de modelo ou distribuição, para posteriormente encontrar valores apropriados para os parâmetros do modelo a partir de um conjunto de dados. Por exemplo, um classificador linear assume que as classes podem ser expressas como combinação linear dos valores dos atributos, e então procura uma combinação linear particular que forneça a melhor aproximação sobre o conjunto de dados. Os classificadores estatísticos freqüentemente assumem que os valores de atributos estão normalmente distribuídos, e então usam os dados fornecidos para determinar média, variância e co-variância da distribuição.

Paradigma Baseado em Exemplos

Uma forma de classificar um caso é lembrar de um caso similar cuja classe é conhecida e assumir que o novo caso pertencerá à mesma classe. Essa filosofia exemplifica os sistemas baseados em exemplos[3], os quais classificam casos nunca vistos utilizando casos similares conhecidos (Aha, Kibler & Albert 1991).

Uma das características principais dos sistemas baseados em exemplos é que os casos de treinamento devem ser lembrados. Entretanto, se todos os casos forem memorizados, o classificador pode se tornar lento e difícil de manusear. O ideal é reter casos prototípicos que juntos consigam sintetizar toda a informação importante. Essa abordagem pode ser observada em livros médicos e legais. Aha, Kibler & Albert (1991) descrevem algumas estratégias para decidir quando um novo caso deve ser memorizado. Uma segunda solução reside em construir estruturas capazes de indexar os exemplos e responder consultas sobre os exemplos mais semelhantes de forma mais rápida. Exemplos dessas estruturas são as M-trees (Ciaccia, Patella & Zezula 1997) e as Slim-trees (Jr., Traina, Seeger & Faloutsos 2000).

[3] *Instance based.*

Dado um novo caso a ser classificado por meio de sistemas baseados em exemplos, é necessário recuperar um caso memorizado similar. Para isso, os sistemas baseados em exemplos utilizam uma medida de similaridade entre os casos. Se todos os atributos forem quantitativos, pode-se calcular a distância entre dois casos utilizando a distância euclidiana, por exemplo. Quando alguns atributos são qualitativos, essa interpretação de distância se torna mais problemática. Além do mais, se existem muitos atributos irrelevantes, dois casos similares podem aparentar serem muito diferentes pois eles podem possuir valores diferentes em atributos sem importância. Stanfill & Waltz (1986) desenvolveram um método sensível ao contexto para alterar a escala dos atributos de forma que as medidas de distância fiquem mais robustas.

Para classificar um novo caso existem diversas alternativas. Uma delas consiste em usar um único caso, o qual é o mais próximo do novo caso para classificar esse novo caso. Uma segunda alternativa consiste em usar vários casos. Nessa alternativa, pode-se levar em consideração os diferentes graus de similaridade entre cada caso e o novo caso na determinação da classe do novo caso. A segunda alternativa é geralmente mais utilizada por ser mais robusta a erros nos dados.

Paradigma Conexionista

Redes Neurais Artificiais são construções matemáticas relativamente simples que foram inspiradas no modelo biológico do sistema nervoso. Sua representação envolve unidades altamente inter-conectadas, e o nome *conexionismo* é também utilizado para descrever a área de estudo. Uma característica de RNAs é que elas podem ser muito difíceis de serem compreendidas pelos seres humanos.

A metáfora biológica com as conexões neurais do sistema nervoso tem interessado muitos pesquisadores, e tem fornecido diversas discussões sobre os méritos e as limitações dessa abordagem de aprendizado. Em particular, as analogias com a biologia têm levado muitos pesquisadores a acreditar que as RNAs possuem um grande potencial na resolução de problemas que requerem intenso processamento sensorial humano, tal como visão e reconhecimento de voz.

As pesquisas em RNAs foram iniciadas com o trabalho pioneiro de McCulloch & Pitts (1943). McCulloch era um psiquiatra e pesquisou por 20 anos uma forma de representar um evento no sistema nervoso. Pitts era um jovem pesquisador e começou a trabalhar com McCulloch em 1942. Praticamente 15 anos após a publicação de McCulloch e Pitts, Rosenblatt (1958) apresentou o *Perceptron*, cuja grande contribuição foi a prova do teorema de convergência. Contudo, no livro *Perceptrons*, Minsky &

Papert (1969) demonstraram a existência de limites fundamentais nos *perceptrons* de uma única camada. A pesquisa na área ficou praticamente adormecida até que Hopfield (1982) utilizou a idéia de uma função de energia para formular uma nova forma de compreender os cálculos realizados em redes recorrentes com conexões sinápticas simétricas.

Talvez mais do que qualquer outra publicação, o artigo de Hopfield em 1982 e o livro de Rumelhart, Hilton & Williams (1986), foram as publicações que mais influenciaram o ressurgimento no interesse em RNAs na década de 80.

No Capítulo 2, RNAs são descritas em maiores detalhes.

Paradigma Evolutivo

Esse paradigma tem sido freqüentemente utilizado como método de busca para encontrar uma hipótese que aproxima o conceito a ser aprendido. O paradigma evolutivo possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os mais bem adaptados ao ambiente. Esse paradigma, descrito em detalhes no Capítulo 2, é também utilizado neste trabalho.

1.3 Objetivo do Trabalho

RNAs têm sido utilizadas na resolução de problemas de diversos domínios, obtendo geralmente um bom desempenho. Entretanto, para várias aplicações não é suficiente obter bom desempenho, é necessário também que a hipótese induzida seja compreensível pelos seres humanos. A compreensibilidade da hipótese induzida é uma das principais características dos sistemas de aprendizado de máquina simbólico.

Neste trabalho é proposta a utilização de sistemas de AM simbólico para a extração de conhecimento compreensível de RNAs. O principal objetivo é criar uma descrição simbólica de uma RNA que possua alta fidelidade com o conhecimento aprendido pela RNA. Dessa forma, são propostos dois métodos para extração de conhecimento simbólico compreensível de RNAs, tal que, a partir de um conjunto de dados E, os métodos propostos utilizam uma RNA para induzir uma hipótese $\mathbf{h}$ que aproxima a função conceito desconhecida f. Em um segundo passo, são utilizados sistemas de AM simbólico que visam induzir uma hipótese $\mathbf{h}'$ que aproxima a hipótese $\mathbf{h}$ aprendida pela RNA.

Os métodos propostos possuem como uma de suas principais qualidades o fato de poderem ser aplicados a qualquer tipo ou arquitetura de RNAs para problemas de clas-

sificação. Para utilizá-los, não é necessário que a RNA tenha uma arquitetura específica, nem que ela seja treinada com um determinado algoritmo. Outro aspecto positivo dos métodos propostos a ser destacado é que a indução das representações simbólicas não é diretamente afetada pelo tamanho da rede. Além disso, esses métodos podem ser utilizados para aplicações envolvendo tanto atributos de valores discretos como de valores reais.

Na próxima seção, os métodos propostos são descritos.

1.4 Métodos Propostos

Neste trabalho, como mencionado, são propostos dois métodos para extrair representações simbólicas de RNAs. No primeiro método são utilizados sistemas de AM simbólico para extrair o conhecimento de uma RNA. O segundo método estende o primeiro, possibilitando que diversos sistemas de AM sejam utilizados para explicar o conhecimento embutido em uma RNA, integrando o conhecimento desses sistemas por meio de Algoritmos Genéticos — AGs.

A seguir, os dois métodos propostos para extração de conhecimento compreensível de RNAs são brevemente descritos.

1.4.1 Método Baseado em Sistemas de AM Simbólico

O método baseado em sistemas de AM simbólico, como o próprio nome sugere, utiliza sistemas de AM simbólico para extrair representações simbólicas de RNAs. Como mencionado anteriormente, uma das principais características dos sistemas de AM simbólico é que eles representam as hipóteses induzidas por meio de estruturas simbólicas, que são geralmente mais compreensíveis para os seres humanos. Na Figura 1.2 na próxima página é mostrado o esquema geral desse método proposto.

Nesse método, uma RNA é treinada a partir de um conjunto de dados E. Após o treinamento, a RNA é utilizada como uma "caixa preta" para classificar o próprio conjunto de dados E, criando novos valores para o atributo classe. Os novos valores do atributo classe refletem o conhecimento aprendido pela RNA, ou seja, refletem a hipótese $\mathbf{h}$ induzida pela RNA. O conjunto de dados rotulado pela RNA é fornecido como entrada para o sistema de AM simbólico. O sistema de AM simbólico induz uma representação simbólica com base nos novos valores para o atributo classe criados pela RNA. Dessa forma, o classificador simbólico $\mathbf{h}'$ induzido pelo sistema AM simbólico visa aproximar a hipótese $\mathbf{h}$ induzida

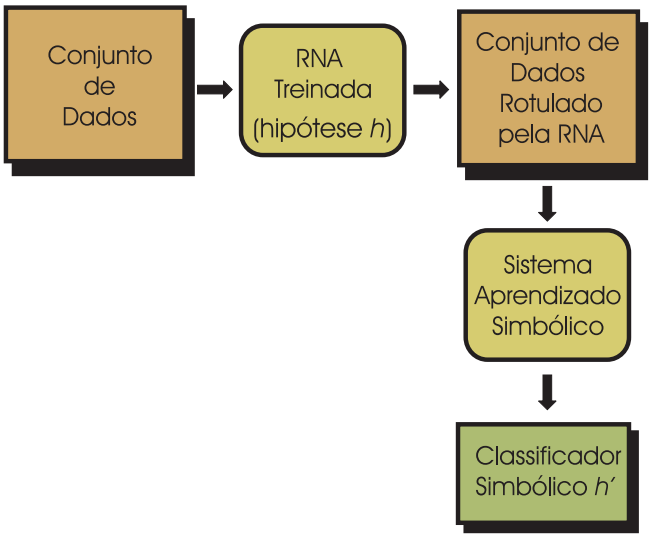

Figura 1.2: Esquema do método proposto baseado em sistemas de AM simbólico.

pela RNA. Esse classificador simbólico é então utilizado para explicar o comportamento da RNA.

Uma das qualidades desse método é que qualquer sistema de AM simbólico pode ser utilizado para induzir um classificador simbólico a partir do conjunto de dados rotulado pela RNA. Isso faz com que esse método seja simples e fácil de usar, pois pode-se utilizar sistemas de AM conhecidos e freqüentemente usados pela comunidade de AM.

1.4.2 Método Baseado em Sistemas de AM Simbólico e Algoritmos Genéticos

Esse método, além de sistemas de AM simbólico, utiliza também AGs para extrair representações simbólicas de RNAs. Como no método baseado em sistemas de AM simbólico, um conjunto de dados E utilizado no treinamento de uma RNA é rotulado por essa mesma RNA. Esse conjunto de dados é fornecido como entrada para vários sistemas de AM simbólico. Como resultado, são induzidos p classificadores simbólicos $\mathbf{h}'_1, \mathbf{h}'_2, \ldots \mathbf{h}'_p$, sendo que cada classificador $\mathbf{h}'_i$, $1 \leq i \leq p$, aproxima a hipótese $\mathbf{h}$ induzida pela RNA.

Cada sistema de AM simbólico representa o conceito induzido em uma determinada linguagem, dificultando assim a integração desses classificadores. É necessário então a unificação desses classificadores em uma linguagem comum. Para isso foi utilizado o trabalho de Prati, Baranauskas & Monard (2001b), no qual é proposto um formato específico de regras, o formato $\mathcal{PBM}$. Assim, cada classificador simbólico é convertido para o for-

mato padrão $\mathcal{PBM}$. Na Figura 1.3 é mostrado o esquema da primeira etapa do método proposto baseado em AG.

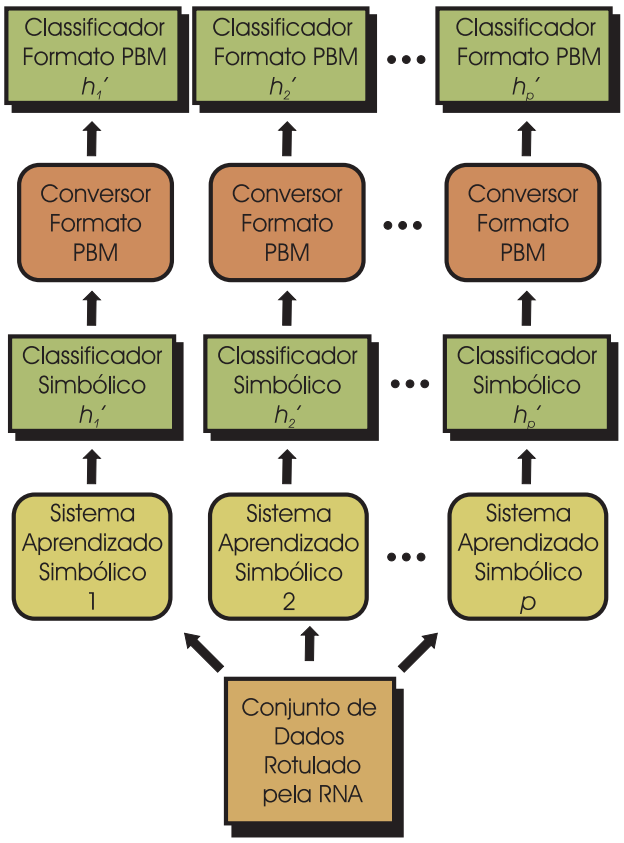

Figura 1.3: Esquema da primeira etapa do método proposto baseado em AG.

Com a conversão dos vários classificadores $\mathbf{h}'_1, \mathbf{h}'_2, \ldots \mathbf{h}'_p$ para uma sintaxe comum, é possível utilizá-los de forma integrada. Isso é necessário na segunda etapa do método proposto para extração de conhecimento simbólico baseado em AGs, mostrado na Figura 1.4 na página oposta. Nessa etapa, as regras que compõem os p classificadores são utilizadas para formar vários indivíduos da população inicial de um AG. Portanto, cada indivíduo é formado por um conjunto de regras, no qual cada regra foi aleatoriamente retirada de um dos p classificadores. Como resultado da execução do AG, tem-se um indivíduo vencedor, ou seja, um conjunto de regras, composto a partir dos classificadores induzidos por vários sistemas de AM simbólico. Esse conjunto de regras é então utilizado para explicar o comportamento da RNA.

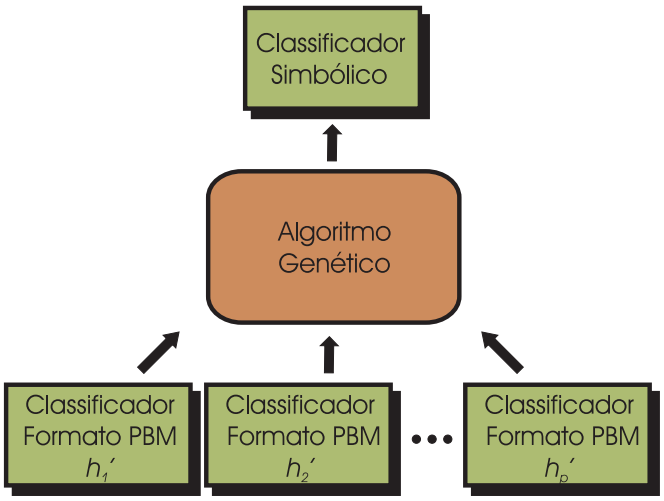

Figura 1.4: Esquema da segunda etapa do método proposto baseado em AG.

1.5 O Ambiente $NNRules$

Como resultado deste trabalho foi projetado e desenvolvido um ambiente computacional para extração de conhecimento de RNAs denominado $NNRules$. $NNRules$ é um ambiente para extração de conhecimento simbólico de RNAs, no qual estão implementados os dois métodos propostos para extração de conhecimento de RNAs. Além disso, o ambiente $NNRules$ dá suporte à realização de experimentos para avaliar e comparar os métodos propostos.

O ambiente $NNRules$ faz parte do projeto DISCOVER. O projeto DISCOVER é um projeto que envolve diversos pesquisadores do LABIC[4]. Um dos principais objetivos do projeto DISCOVER é diminuir o esforço de implementação dos procedimentos necessários para realizar um experimento. Muitas vezes, um procedimento semelhante é implementado diversas vezes, por membros diferentes, por falta de comunicação entre os membros ou por falta de documentação dos programas já implementados. Dessa forma, o projeto DISCOVER visa aumentar a comunicação e a padronização das implementações realizadas no grupo de pesquisa do LABIC.

Por fazer parte do projeto DISCOVER, o ambiente $NNRules$ utiliza alguns trabalhos que foram ou estão sendo realizados por pesquisadores do LABIC como, por exemplo, o ambiente DLE (Batista 2003), a sintaxe padrão de regras $\mathcal{PBM}$ (Prati, Baranauskas & Monard 2001b) e a extração de informações padronizadas para a avaliação de regras (Prati, Baranauskas & Monard 2001a).

[4]Laboratório de Inteligência Computacional.

No Capítulo 4 o ambiente *NNRules* é descrito detalhadamente.

1.6 Organização do Trabalho

Este trabalho está organizado em 6 capítulos. Neste capítulo foi descrito brevemente o objetivo do trabalho, alguns conceitos de AM e foram apresentados os métodos propostos para extrair descrições simbólicas de RNAs.

Capítulo 2 — Aprendizado Computacional

Nesse capítulo são descritos os sistemas de aprendizado utilizados neste trabalho: árvores de decisão, regras de decisão, RNAs e AGs.

Capítulo 3 — Extração de Conhecimento de RNAs

Nesse capítulo são descritos aspectos relacionados à tarefa de extração de conhecimento de RNAs, tais como: *bias* indutivo de algoritmos de aprendizado, representação dos modelos extraídos por esses algoritmos, compreensão de RNAs, processo de extração de regras de RNAs, importância de extrair conhecimento de RNAs, bem como alguns métodos propostos na literatura para extrair de conhecimento de RNAs.

Capítulo 4 — O Ambiente *NNRules*

Nesse capítulo é apresentado o projeto e a arquitetura do ambiente *NNRules*. Esse ambiente, além de implementar os métodos propostos para extração de conhecimento simbólico de RNAs, também dá suporte aos vários testes realizados para avaliar esse métodos.

Capítulo 5 — Avaliação Experimental dos Métodos Propostos

Nesse capítulo são descritos os experimentos realizados para avaliar os métodos propostos. Os resultados obtidos são comparados aos resultados obtidos pelo método de extração de conhecimento TREPAN. O método proposto baseado em sistemas de AM simbólico é semelhante ao método TREPAN.

Capítulo 6 — Conclusão

Por fim, nesse capítulo são apresentadas as principais conclusões deste trabalho, bem como os trabalhos futuros.

Apêndice A — Relatórios do Ambiente *NNRules*

Nesse apêndice são mostrados exemplos de relatórios gerados pelo Ambiente *NNRules*.

Apêndice B — Resultados do Teste de Hipótese

Nesse apêndice são mostrados os resultados dos testes de hipóteses referentes à comparação entre os métodos propostos e o método TREPAN.

Capítulo 2

Aprendizado Computacional

2.1 Considerações Iniciais

Os métodos propostos neste trabalho para extrair conhecimento compreensível de Redes Neurais Artificiais utilizam sistemas de aprendizado de máquina simbólico. Em AM existem vários sistemas de aprendizado que induzem classificadores. Neste capítulo são descritos sistemas de AM simbólico cujas representações simbólicas estão na forma de: árvores de decisão, descritos na Seção 2.2; e, regras de decisão, descritos na Seção 2.3.

Além dos sistemas de AM simbólico, RNAs são descritas na Seção 2.4. Um dos métodos propostos neste trabalho utiliza o método de otimização Algoritmos Genéticos para extrair conhecimento compreensível de RNAs e, assim, AGs são discutidos na Seção 2.5.

2.2 Árvores de Decisão

Uma árvore de decisão, como a ilustrada na Figura 2.1 na próxima página, é uma estrutura que consiste em um conjunto de nós internos, representados nessa figura por elipses, e um conjunto de nós folhas, representados por retângulos. O nó inicial é denominado de nó raiz. Cada nó interno representa um teste, e cada nó folha uma decisão. Dependendo do resultado obtido em um teste, deve-se descer por um dos ramos da árvore de decisão. Ao atingir um nó folha, é tomada uma decisão. A árvore de decisão mostrada na Figura 2.1 ilustra como poderia ser resolvido um problema muito simples cujo objetivo é diagnosticar se um paciente está ou não doente.

Existem diversos sistemas comerciais e acadêmicos que induzem árvores de decisão

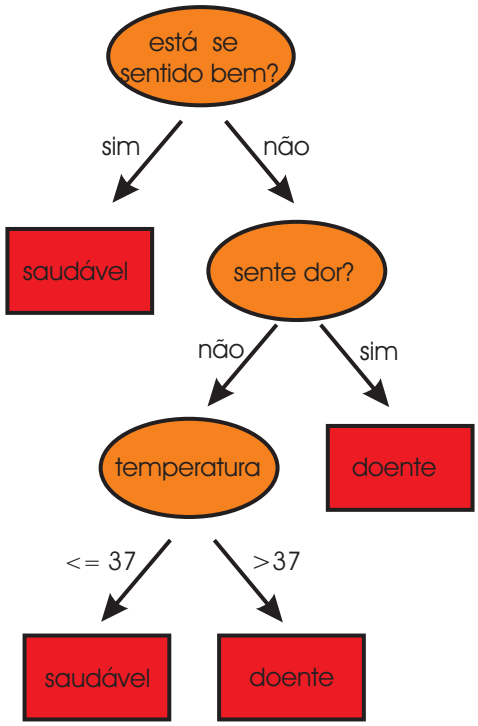

Figura 2.1: Exemplo de uma árvore de decisão.

utilizando exemplos. Um dos sistemas mais conhecidos e utilizados na comunidade de AM é o algoritmo C4.5 (Quinlan 1988).

O processo para construir a estrutura de uma árvore de decisão a partir de exemplos é conhecido como *indução de árvores de decisão*. Seja $E = \{E_1, E_2, \ldots E_N\}$ um conjunto de exemplos e $C = \{C_1, C_2, \ldots C_{Ncl}\}$ o conjunto de valores que o atributo classe pode assumir. Os passos a seguir descrevem o processo de construção de uma árvore de decisão (Baranauskas & Monard 2000).

1. Se todos os exemplos de E pertencem à classe C_j, então, um nó folha é criado e associado à classe C_j.

2. Se E não contém nenhum exemplo, então, um nó folha é criado, entretanto, a classe associada a este nó deve ser determinada a partir de outra informação e não de E. A classe associada a este nó folha pode ser, por exemplo, a classe mais freqüente de seu nó antecessor.

3. Se E contém exemplos pertencentes à várias classes, então, um teste é escolhido baseado em um atributo que possui um conjunto de valores mutuamente exclusivos, denotado por $V = \{V_1, V_2, \ldots V_r\}$. Em seguida, E é particionado nos subconjuntos

disjuntos P_1, P_2, ... P_r, no qual P_i contém todos os exemplos de E que possuem o valor V_i para o teste escolhido. Nessa situação são criados: um nó interno que é associado ao teste escolhido, e um ramo para cada um dos possíveis valores do atributo relacionado ao teste escolhido.

4. Os passos 1, 2 e 3 são recursivamente aplicados em cada um dos subconjuntos de exemplos P_1, P_2, ... P_r.

5. Geralmente, depois da construção da árvore de decisão, é realizado um processo denominado de *poda*, que será comentado posteriormente.

A seleção adequada do atributo associado ao teste de divisão de um nó interno tem influência direta no tamanho final da árvore de decisão. Algumas medidas da *Teoria da Informação* (Cover & Thomas 1991) podem ajudar na seleção dos atributos. Uma função bastante popular utilizada na avaliação de divisão de nós é conhecida como função de *entropia* (Mitchell 1997b, Capítulo 3).

É possível que uma árvore de decisão induzida seja muito específica aos exemplos a partir da qual foi gerada. Isso acontece pela adição de ramos na árvore de decisão para melhorar o desempenho em relação aos exemplos usados na indução da árvore. Entretanto, o conjunto de exemplos utilizado na indução da árvore é somente uma amostra de todos os exemplos possíveis. Assim, o desempenho da árvore de decisão para exemplos não utilizados na indução é freqüentemente ruim. Esse problema é denominado *overfitting*[1] e pode ser evitado nas árvores de decisão por meio de um processo denominado de poda. A poda de uma árvore de decisão consiste em reduzir o número de nós de uma árvore por meio da eliminação de ramos "fracos", os quais podem estar especializados a alguns dos exemplos usados durante o processo de indução.

Para ilustrar o processo de indução de uma árvore de decisão, a seguir é descrito um exemplo adaptado de (Quinlan 1988). Esse exemplo utiliza o conjunto de dados mostrado na Tabela 2.1 na página seguinte, o qual contém medições diárias de condições meteorológicas dos seguintes atributos:

- `outlook`: indica como o dia está e pode assumir os valores `sunny`, `overcast` ou `rain`;

[1] Vários termos neste trabalho são utilizados em inglês por serem amplamente difundidos na comunidade.

- `temperature`: indica a temperatura em graus Célsius e pode assumir um valor numérico;
- `humidity`: indica a porcentagem da umidade relativa do ar e pode assumir um valor numérico;
- `windy`: indica se há ou não vento e pode assumir os valores `yes` ou `no`.

Exemplo	Outlook	Temperature	Humidity	Windy	Voyage?
E_1	sunny	25	72	yes	go
E_2	sunny	28	91	yes	dont_go
E_3	sunny	22	70	no	go
E_4	sunny	23	95	no	dont_go
E_5	sunny	30	85	no	dont_go
E_6	overcast	23	90	yes	go
E_7	overcast	29	78	no	go
E_8	overcast	19	65	yes	dont_go
E_9	overcast	26	75	no	go
E_{10}	overcast	20	87	yes	go
E_{11}	rain	22	95	no	go
E_{12}	rain	19	70	yes	dont_go
E_{13}	rain	23	80	yes	dont_go
E_{14}	rain	25	81	no	go
E_{15}	rain	21	80	no	go

Tabela 2.1: Conjunto de dados `Voyage`.

Como descrito na Seção 1.2 na página 2, cada linha da Tabela 2.1 é um exemplo ou caso que representa um determinado dia. A classe de cada exemplo é determinada pela última coluna da tabela, denominada de atributo `classe`. Cada exemplo foi classificado como `go`, se as condições meteorológicas daquele dia eram favoráveis à uma viagem à fazenda; ou `dont_go`, caso contrário. O primeiro atributo, `Exemplo`, somente está relacionado para ajudar a identificar os exemplos.

Como o conjunto de dados possui exemplos que pertencem a mais de uma classe, é necessário escolher um teste baseado em um dos quatro atributos. A escolha adequada do atributo, como mencionado anteriormente, pode se basear em algumas medidas e, portanto, depende da implementação do algoritmo de indução utilizado. Suponha que o atributo `outlook` é selecionado. Como há três valores possíveis para esse atributo, então o conjunto de dados é particionado em três subconjuntos, como mostrado na Figura 2.2 na próxima página.

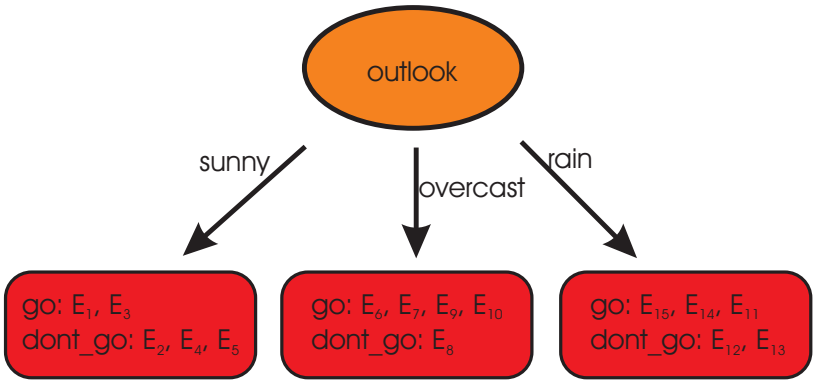

Figura 2.2: Construção da árvore de decisão (passo 1).

Pode ser observado que em cada subconjunto, resultante da aplicação do teste do nó raiz, existem exemplos pertencentes a classes diferentes. Portanto, é necessário escolher um novo teste, baseado em outros atributos, para cada subconjunto. Suponha que desta vez seja escolhido o atributo `humidity` para dividir os exemplos do subconjunto resultante do ramo `outlook = sunny`. O atributo `humidity` é quantitativo (numérico), esse tipo de atributo normalmente possui muitos valores distintos. Portanto, não é razoável dividir o nó nos diversos valores que um atributo quantitativo assume, da mesma forma que nos atributos qualitativos (discretos).

Para criar testes com atributos quantitativos, um algoritmo de árvore de decisão precisa adotar uma abordagem diferente daquela utilizada com atributos qualitativos. Normalmente, testes com atributos quantitativos são da forma:

<atributo> <operador> <valor de corte>

no qual `<atributo>` é o atributo envolvido no teste; `<operador>` é um operador relacional, freqüentemente pertencente ao conjunto de operadores $\{=, \neq, >, \geq, <, \leq\}$; e, `<valor de corte>` é um valor escolhido pelo indutor tal que o teste separe bem os exemplos de cada classe.

A escolha do valor de corte não é um processo trivial. Ela geralmente envolve uma busca na qual diversos valores são avaliados e o melhor deles é escolhido. Por simplicidade, suponha que o valor de corte 78 é escolhido para dividir os exemplos do subconjunto resultante do ramo `outlook = sunny`. Nesse caso, os exemplos são divididos em dois outros subconjuntos. Um subconjunto com exemplos que possuem valores menores ou iguais a 78 para o atributo `humidity`, e outro subconjunto com valores maiores que 78, como ilustrado na Figura 2.3 na página seguinte. Em cada subconjunto criado só existem

exemplos pertencentes à mesma classe. Nesse caso, não é mais necessário dividir os exemplos desses nós, e cada nó é rotulado com a classe a qual os exemplos pertencem.

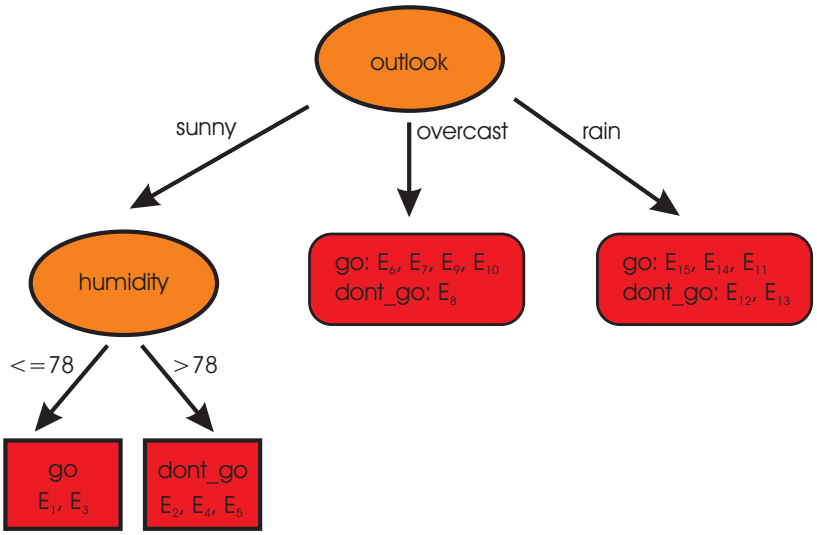

Figura 2.3: Construção da árvore de decisão (passo 2).

O mesmo processo é realizado para os demais ramos da árvore. Para o ramo `outlook = overcast` foi escolhido o atributo `humidity` e o valor de corte 70. E para o ramo `outlook = rain` foi escolhido o atributo `windy`, o qual separa os exemplos com os valores `yes` e `no`. Cada subconjunto criado possui somente exemplos de uma mesma classe. Dessa forma, os subconjuntos não são mais divididos e os nós são transformados em nós folhas, como mostrado na Figura 2.4.

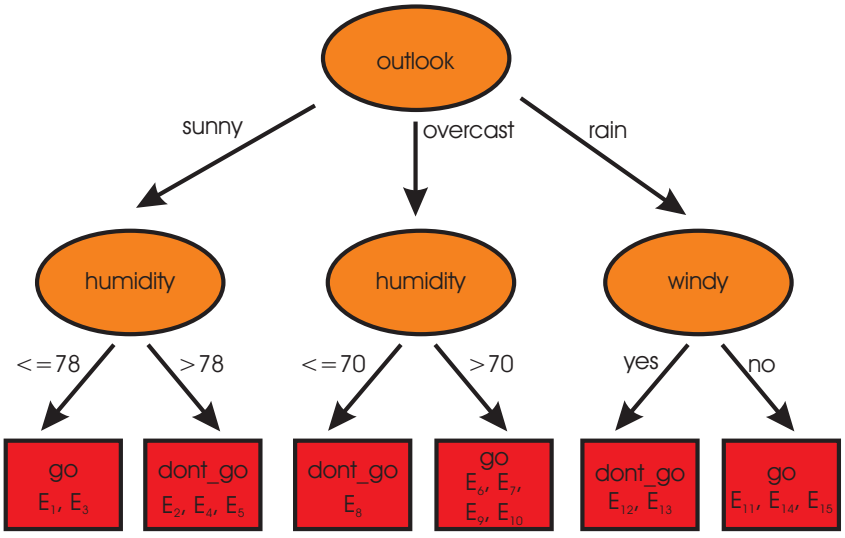

Figura 2.4: Construção da árvore de decisão (passo 3).

O nó folha resultante dos testes `outlook = overcast` e `humidity` <= `70` possui apenas o exemplo E_8 pertencente à classe `dont_go`. Isso pode indicar *overfitting*. Nesse caso, o indutor pode iniciar um processo de poda no qual cada um dos ramos da árvore de decisão é analisado. Por simplicidade, suponha que o indutor considerou o ramo resultante dos testes `outlook = overcast` e `humidity` <= `70` um ramo "fraco" e o podou, como mostrado na Figura 2.5. Dessa forma, é criado um nó folha com os exemplos: E_8, pertencente à classe `dont_go`; e E_6, E_7, E_9 e E_{10}, pertencentes à classe `go`. Esse nó folha é rotulado com a classe pertencente à maioria dos exemplos, ou seja, a classe `go`. Com isso, o processo de indução da árvore de decisão é encerrado.

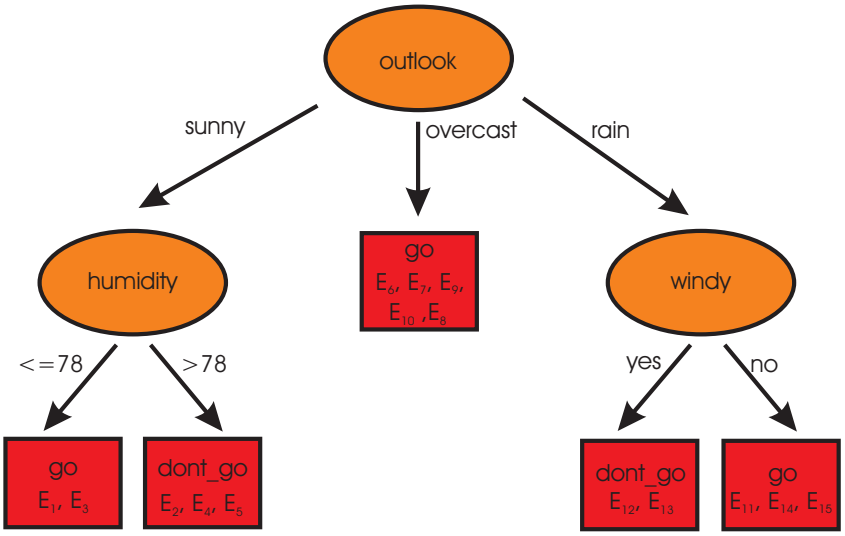

Figura 2.5: Árvore de decisão do conjunto de dados `Voyage`.

Uma árvore de decisão pode ser representada por meio de um conjunto de regras. Cada caminho que parte do nó raiz e alcança um nó folha corresponde a uma regra distinta. Estas regras são formadas por conjunções ($\wedge$) dos testes encontrados no percurso do nó raiz a um nó folha. As regras obtidas de uma árvore de decisão podem ser compostas por meio de disjunções ($\vee$), formando um conjunto de regras.

As regras que representam uma árvore de decisão são mutuamente exclusivas. Isto significa que um determinado exemplo somente pode ser coberto[2] por uma regra. A árvore de decisão da Figura 2.5 pode ser transformada no conjunto de regras a seguir.

[2]Diz-se que uma regra *cobre* um exemplo quando todas as condições da regra são satisfeitas pelo exemplo. Diz-se que uma regra *cobre positivamente* um exemplo quando todas as suas condições e sua conclusão são satisfeitas pelo exemplo. Diz-se que uma regra *cobre negativamente* um exemplo quando todas as suas condições são satisfeitas e sua conclusão não é satisfeita.

$$\textbf{if } (\texttt{outlook} = \texttt{sunny}) \wedge (\texttt{humidity} \leq 78) \textbf{ then } \texttt{classe} = \texttt{go} \vee$$
$$\textbf{if } (\texttt{outlook} = \texttt{sunny}) \wedge (\texttt{humidity} > 78) \textbf{ then } \texttt{classe} = \texttt{dont\_go} \vee$$
$$\textbf{if } (\texttt{outlook} = \texttt{overcast}) \textbf{ then } \texttt{classe} = \texttt{go} \vee$$
$$\textbf{if } (\texttt{outlook} = \texttt{rain}) \wedge (\texttt{windy} = \texttt{no}) \textbf{ then } \texttt{classe} = \texttt{go} \vee$$
$$\textbf{if } (\texttt{outlook} = \texttt{rain}) \wedge (\texttt{windy} = \texttt{yes}) \textbf{ then } \texttt{classe} = \texttt{dont\_go} \vee$$

Geometricamente, árvores de decisão dividem os exemplos por meio de hiperplanos paralelos aos eixos. Cada hiperplano corresponde a um teste diferente de uma árvore de decisão. Em duas dimensões esses hiperplanos correspondem a retas paralelas aos eixos x e y, como ilustrado na Figura 2.6. Nessa figura é mostrada uma interpretação geométrica para uma árvore de decisão de duas classes (+ e ∘) e dois atributos quantitativos (A_1 e A_2).

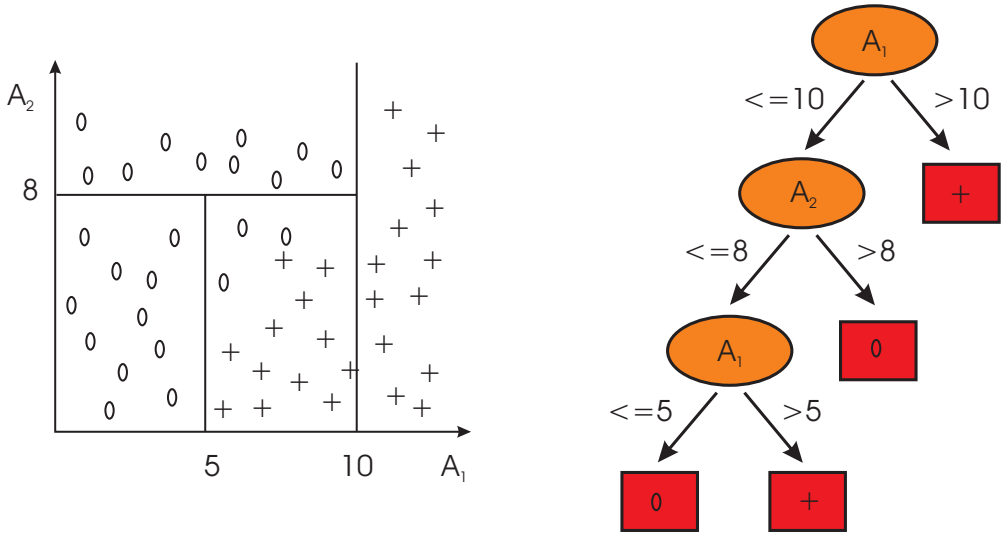

Figura 2.6: Interpretação geométrica para uma árvore de decisão.

Para classificar um novo exemplo utilizando uma árvore de decisão, deve-se percorrer a árvore do nó raiz até um nó folha verificando para o novo exemplo o resultado de cada teste encontrado. Quando um nó folha é alcançado, a classe do novo exemplo será a mesma desse nó folha.

Como mencionado anteriormente, as regras extraídas de uma árvore de decisão são mutuamente exclusivas e, portanto, um novo exemplo somente poderá ser classificado por uma única regra.

2.3 Regras de Decisão

Uma forma de induzir um conjunto de regras é induzir primeiramente uma árvore de decisão, e então transformar a árvore em um conjunto de regras equivalentes, como mencionado anteriormente. Nesse caso, tem-se uma regra para cada nó folha da árvore de decisão. Outra forma é induzir um conjunto de regras diretamente dos exemplos. Nesse processo, cada regra cobre um subconjunto de exemplos pertencentes a uma determinada classe, mas essas regras não são necessariamente disjuntas.

Existem vários indutores propostos pela comunidade de AM que utilizam conjuntos de regras para representar o conhecimento induzido. Um deles é o CN2 (Clark & Niblett 1989), que induz regras diretamente dos exemplos. Outro indutor bastante popular é o C4.5rules (Quinlan 1988), que induz um conjunto de regras a partir de uma árvore de decisão induzida pelo C4.5. O C4.5rules não transforma simplesmente a árvore de decisão em um conjunto de regras. Ele generaliza as regras apagando condições supérfluas, isto é, condições irrelevantes que não afetam a conclusão, sem comprometer seu desempenho.

2.3.1 Indução de Regras

O processo de indução de regras cria um conjunto de regras, no qual cada regra cobre um subconjunto de exemplos. De forma geral, os indutores criam conjuntos de regras ordenadas ou não ordenadas. Em um conjunto de regras ordenadas, a ordem das regras é relevante. Por exemplo, a segunda regra pode ser disparada se e somente se a primeira regra não for disparada. Em um conjunto de regras não ordenado não existe uma ordem implícita entre as regras. Várias regras podem disparar para um mesmo exemplo. Nesse caso, se as conclusões das regras disparadas forem diferentes, o indutor deve utilizar alguma estratégia para classificar o exemplo. A seguir é descrito o processo de indução de regras ordenadas e o processo de indução de regras não ordenadas.

2.3.1.1 Indução de Regras Ordenadas

O processo de indução de regras ordenadas consiste, à cada iteração, em buscar pelo melhor conjunto de condições (<condições>) que cubram um grande número de exemplos de uma classe C_i e poucos exemplos de outras classes C_j, com $j \neq i$. Ao encontrar essas condições, os exemplos cobertos pertencentes à classe C_i e, eventualmente, alguns poucos exemplos também cobertos por essas mesmas condições, mas pertencentes à classe C_j, são

removidos do conjunto de exemplos e a regra

$$\texttt{if } <\texttt{condições}> \texttt{ then } <\texttt{classe=}C_i>$$

é adicionada ao final de uma lista de regras. A busca continua até que nenhuma nova condição possa ser encontrada ou não existam mais exemplos a serem cobertos.

Para exemplificar o processo de indução de regras ordenadas, considere o mesmo conjunto de dados mostrado na Tabela 2.1 na página 20, utilizado anteriormente para exemplificar a construção de uma árvore de decisão.

Na Tabela 2.2 é mostrada uma lista de regras, na ordem em que foram induzidas, e também são mostrados, para cada regra, os exemplos que são positivamente cobertos (PC) e os exemplos que são negativamente cobertos (NC) pelas regras . Começando pela regra R_1, todos os exemplos que são cobertos por essa regra, ou seja, E_1, E_3, E_7, E_9, E_{14} e E_{15} (positivamente cobertos); e, E_8, E_{12} e E_{13} (negativamente cobertos), são removidos do conjunto de dados e um `else` é introduzido na lista de regras antes da próxima regra ser induzida. Em seguida, a regra R_2 é induzida e os exemplos cobertos por esta regra são removidos, um `else` é introduzido na lista de regras e o processo continua até que nenhuma regra possa ser induzida ou não haja mais exemplos. A regra R_4 é a regra *default*, que somente é disparada se nenhuma das regras anteriores (R_1, R_2, R_3) for disparada. Geralmente, a regra *default* é associada à classe majoritária[3].

	Regra	PC	NC
R_1	`if humidity` $<$ `83 then classe = go`	$E_1, E_3, E_7, E_9, E_{14}, E_{15}$	E_8, E_{12}, E_{13}
R_2	`else if temperature` $\geq$ `23 then classe = dont_go`	E_2, E_4, E_5	E_6
R_3	`else if outlook = rain then classe = go`	E_{11}	
R_4	`else default classe = go`	E_{10}	

Tabela 2.2: Conjunto de regras ordenadas induzidas para o conjunto de dados `Voyage`.

A interpretação geométrica para regras ordenadas é semelhante, mas não idêntica, à interpretação geométrica dada à árvores de decisão. Os exemplos são particionados por meio de hiperplanos paralelos aos eixos. Cada exemplo é coberto por apenas uma regra, pois o processo de classificação pára quando esse exemplo é coberto por uma regra. Entretanto, deve ser ressaltado que a interpretação dessas regras é diferente da interpretação de regras provenientes de uma árvore de decisão. Cada regra proveniente de uma

[3]Classe majoritária é a classe a qual a maioria dos exemplos do conjunto de dados pertence.

árvore de decisão pode ser interpretada independentemente, pois elas são disjuntas. No caso de regras ordenadas induzidas diretamente dos exemplos, a interpretação de cada regra independentemente das outras, somente é possível pela negação das condições das regras anteriores a ela. Por exemplo, para uma interpretação independente das regras da Tabela 2.2 na página anterior, elas deveriam ser escritas da seguinte forma:

R'_1 `if humidity` $<$ `83 then classe = go`

R'_2 `if not(humidity` $<$ `83)` $\wedge$ `temperature` $\geq$ `23 then classe = dont_go`

R'_3 `if not(humidity` $<$ `83)` $\wedge$ `not(temperature` $\geq$ `23)` $\wedge$ `outlook = rain then classe = go`

R'_4 `if not(humidity` $<$ `83)` $\wedge$ `not(temperature` $\geq$ `23)` $\wedge$ `not(outlook = rain) then classe = go`

Para classificar um novo exemplo, utilizando um conjunto de regras ordenadas, começa-se pela primeira regra até que seja encontrada uma regra que cubra o exemplo. A classe do novo exemplo será a mesma classe da regra que cobre o exemplo. Note que o exemplo deve ser classificado exclusivamente por esta regra, mesmo que as regras seguintes também cubram o exemplo. Portanto, a ordem de aplicação das regras é muito importante. Caso nenhuma regra seja satisfeita, o exemplo é classificado como sendo da classe *default*.

2.3.1.2 Indução de Regras Não Ordenadas

No processo de indução de regras não ordenadas, são induzidas regras que predizem cada uma das classes separadamente. A cada iteração, uma classe é eleita e todas as regras que predizem essa classe são induzidas. Esse processo é repetido para cada uma das classes do conjunto de dados.

O processo de indução de regras para uma classe inicia com o conjunto de dados completo. Diferente do processo de indução de regras ordenadas, somente os exemplos cobertos positivamente por cada regra construída são removidos do conjunto de dados. Os exemplos cobertos positivamente devem ser removidos do conjunto de dados para que uma mesma regra não seja induzida mais de uma vez. Os exemplos cobertos negativamente permanecem no conjunto de dados, pois cada regra induzida deve ser confrontada com todos os exemplos ainda não cobertos positivamente. O processo de indução de exemplos para uma classe termina quando não for possível encontrar uma regra que cubra exemplos dessa classe, ou quando os exemplos dessa classe terminam. Nesse caso, o processo de

indução de regras para outra classe é iniciado utilizando todos os exemplos do conjunto de dados novamente.

Para ilustrar o processo de indução de regras não ordenadas, o mesmo conjunto de dados `voyage`, mostrado na Tabela 2.1 na página 20, é utilizado. Na Tabela 2.3 é mostrado um conjunto de regras na ordem em que foram induzidas e também os exemplos positivamente cobertos (PC) e os exemplos negativamente cobertos (NC), para cada uma das regras. Inicialmente, foram induzidas regras para a classe `go`. Para a primeira regra induzida, R_1, todos os exemplos cobertos positivamente por essa regra, E_6, E_7, E_9 e E_{10}, são removidos do conjunto de dados. Os exemplos cobertos negativamente, nesse caso, E_8, permanecem no conjunto de dados. Em seguida, a próxima regra é induzida e o processo continua até que nenhuma nova regra possa ser induzida ou terminem os exemplos cobertos positivamente. Quando uma dessas condições é satisfeita, o algoritmo passa a induzir regras para outra classe, nesse caso, a classe `dont_go`. O processo de indução para a classe `dont_go` inicia com todos os exemplos novamente. As regras que predizem a classe `dont_go` são as regras R_3, R_4 e R_5. A R_6 é a regra *default* e somente é disparada se nenhuma das regras anteriores, R_1, R_2, R_3, R_4 ou R_5 for disparada.

	Regra	PC	NC
R_1	`if outlook = overcast then classe = go`	E_6, E_7, E_9, E_{10}	E_8
R_2	`if (outlook = rain)` $\wedge$ `(windy = no) then classe = go`	E_{11}, E_{14}, E_{15}	
R_3	`if (outlook = sunny)` $\wedge$ `(humidity` $>$ `77) then classe = dont_go`	E_2, E_4, E_5	
R_4	`if (outlook = rain)` $\wedge$ `(windy = yes) then classe = dont_go`	E_{12}, E_{13}	
R_5	`if (outlook = overcast)` $\wedge$ `(temperature` $<$ `20) then classe = dont_go`	E_8	
R_6	`default classe = go`		

Tabela 2.3: Conjunto não ordenado de regras induzidas para o conjunto de dados `Voyage`.

Após o processo de indução, o conjunto de regras pode ser utilizado para predizer a classe de novos exemplos. Como as regras são não ordenadas, uma forma de classificar um exemplo é verificar quais regras cobrem o novo exemplo e utilizar essas regras para decidir a classe desse exemplo. Um problema que pode ocorrer com freqüência é que regras disparadas podem predizer classes diferentes, e alguma política de desempate deve ser utilizada para decidir qual classe deve ser predita.

Uma forma de criar uma política de desempate é adicionar a cada regra algum índice que forneça um indicativo do poder de predição da regra. Uma forma de fazê-lo

é verificar quantos exemplos do conjunto de treinamento a regra cobre positivamente e negativamente. Por exemplo, pode-se adicionar a cada regra uma lista $[n_{C_1}, n_{C_2} \ldots, n_{C_{Ncl}}]$. Sendo n_{C_i} o número de exemplos do conjunto de treinamento cobertos pela regra para a classe C_i. O conjunto de regras induzido anteriormente — Tabela 2.3, adicionado dessas listas é mostrado na Tabela 2.4.

	Regra	$[n_{\texttt{go}}, n_{\texttt{dont\_go}}]$
R_1	`if outlook = overcast then classe = go`	$[4, 1]$
R_2	`if (outlook = rain)` $\wedge$ `(windy = no) then classe = go`	$[3, 0]$
R_3	`if (outlook = sunny)` $\wedge$ `(humidity` $>$ `77) then classe = dont_go`	$[0, 3]$
R_4	`if (outlook = rain)` $\wedge$ `(windy = yes) then classe = dont_go`	$[0, 2]$
R_5	`if (outlook = overcast)` $\wedge$ `(temperature` $<$ `20) then classe = dont_go`	$[0, 1]$
R_6	`default classe = go`	$[9, 6]$

Tabela 2.4: Conjunto não ordenado de regras induzidas para o conjunto de dados `Voyage` com listas contendo informações sobre o poder de predição das regras.

Suponha que a classificação de um novo exemplo dispare as regras $R1$ e $R5$. A regra $R1$ prediz a classe `go` e a regra $R5$ prediz a classe `dont_go`. Considerando que a regra $R1$ cobre 4 exemplos da classe `go` e 1 exemplo da classe `dont_go`, e a regra $R5$ cobre 0 exemplo da classe `go` e 1 da classe `dont_go`, pode-se calcular quantos exemplos são cobertos no total pelas duas regras para as duas classes, e predizer a classe mais freqüente. Nesse caso, a cobertura total das duas regras é [4, 2], portanto, a classe atribuída pelo classificador a esse exemplo é a classe `go`. A abordagem utilizada para decidir qual classe vai ser predita para um novo exemplo dado um conjunto de regras não ordenadas pode variar para cada sistema de aprendizado simbólico. Entretanto, essas abordagens são freqüentemente similares à abordagem ilustrada.

Geometricamente, regras não ordenadas podem definir regiões sobrepostas no espaço de descrição do problema, já que um exemplo pode ser coberto por mais de uma regra. Na Figura 2.7 na próxima página é mostrada uma interpretação geométrica para um conjunto de quatro regras não ordenadas. Nesse exemplo, A_1 e A_2 são os atributos, `+` e $\circ$, as classes.

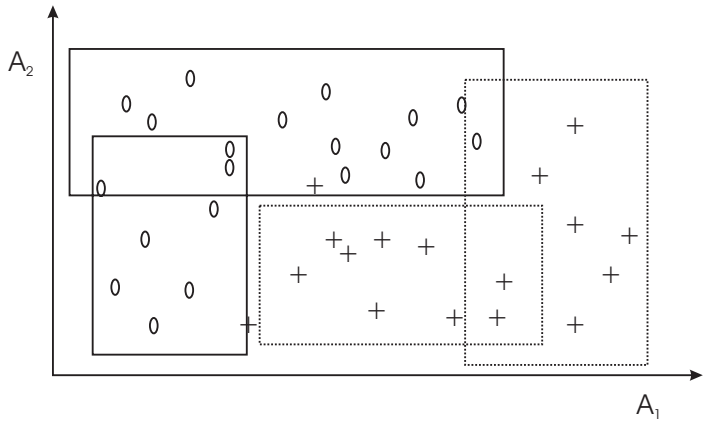

Figura 2.7: Interpretação geométrica para um conjunto de regras não ordenadas.

2.4 Redes Neurais Artificiais

Redes Neurais Artificiais — RNAs — são modelos matemáticos que se assemelham às estruturas neuronais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização (Braga, Carvalho & Ludemir 2000; Haykin 1994; Braga, Carvalho & Ludermir 2003). As RNAs são compostas por unidades de processamento, também denominadas de neurônios, que podem estar distribuídas em uma ou várias camadas, como é mostrado na Figura 2.8 na página oposta. O estado de uma unidade, em um determinado instante, é representado por sua ativação, um valor numérico geralmente no intervalo $[0, 1]$ ou $[-1, 1]$. A primeira camada da rede, camada de entrada, possui unidades cujas ativações representam valores para os atributos do domínio do problema ao qual rede está sendo aplicada. Atributos quantitativos são representados por uma unidade. Atributos qualitativos com n possíveis valores, geralmente são representados por n unidades de entrada em uma codificação conhecida como *one-of-n*[4] (Masters 1993, Capítulo 16). As unidades da camada de saída representam as decisões feitas pela rede. Entre as unidades de entrada e as unidades de saída podem existir outras camadas, denominadas de camadas intermediárias. As unidades entre as camadas da rede estão conectadas por meio de pesos, no qual cada peso corresponde a um valor numérico.

Existe uma grande variedade de arquiteturas e métodos de treinamento de RNAs, tanto para problemas de aprendizado supervisionado como para não supervisionado. Neste

[4]Na codificação *one-of-n* o atributo é representado por n unidades. Cada unidade correspondente a um valor do atributo. Por exemplo, suponha que o atributo `fruta` possa assumir os valores: `maça`, `banana` e `laranja`. Nesse caso, o atributo `fruta` é representado por 3 unidades. Quando um exemplo é apresentado à rede, somente uma unidade é ativa, por exemplo, assume o valor 1. As outras unidades ficam inativas, por exemplo, assumem o valor 0.

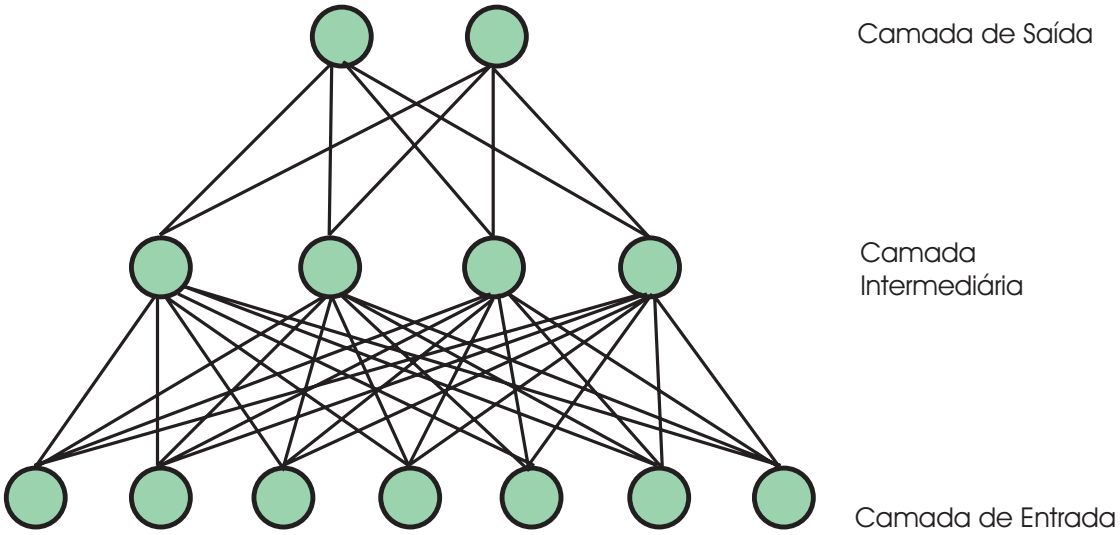

Figura 2.8: Exemplo de uma RNA.

trabalho são utilizadas RNAs *feed-forward* aplicadas à problemas de classificação. Redes *feed-forward* são redes que não possuem unidades de uma camada conectadas à unidades da mesma camada ou de camadas anteriores.

Geralmente, a solução de problemas utilizando uma RNA inicia-se pela fase de aprendizado ou treinamento, no qual exemplos representativos das classes do problema são apresentados à rede. A RNA extrai automaticamente, por meio de um algoritmo de treinamento, as características necessárias para representar implicitamente os exemplos. Essas características são utilizadas posteriormente pela rede, durante a sua fase de uso, para classificar outros exemplos em função de seu grau de similaridade com as representações armazenadas na rede.

Durante o treinamento, os pesos associados às conexões são calculados e modificados até a rede atingir uma configuração que produza as saídas desejadas com uma margem de erro mínima, para os exemplos apresentados à rede durante o treinamento. Existem vários métodos de treinamento de RNAs, também denominados de regras de aprendizado de RNAs.

Para descrever o treinamento de uma RNA será utilizado um modelo bem simples de uma RNA denominada *perceptron* (Rosenblatt 1962). Esse modelo possui apenas uma camada de entrada e uma camada de saída, como ilustrado na Figura 2.9 na próxima página. x_1 e x_2 representam a entrada da rede, ou seja, valores dos atributos de um exemplo; w_1 e w_2 são os pesos, valores reais; θ é uma constante conhecida como *threshold*; e O é a saída da rede.

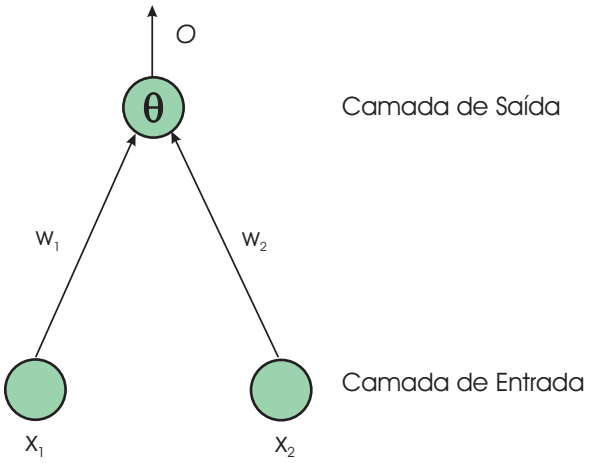

Figura 2.9: Exemplo de uma rede *perceptron*.

No início do treinamento, os pesos são inicializados com números gerados randomicamente, geralmente no intervalo $[-1, 1]$. Em seguida, os exemplos são apresentados à rede até que um critério de parada seja satisfeito. A cada exemplo apresentado, a saída O é calculada utilizando a Equação 2.1:

$$O = \begin{cases} 1 & \text{Se } \sum_i w_i x_i - \theta > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{2.1}$$

Note que se o somatório dos pesos w_i multiplicados pelas entradas x_i exceder o *threshold* (θ), a saída será mapeada para 1, caso contrário, a saída será mapeada para 0. A função que dá a ativação da unidade de processamento é denominada de função de ativação. No caso da rede *perceptron* a função de ativação é a função *threshold*.

Em seguida, a saída da rede O é comparada à saída desejada y. Se essas saídas forem iguais, os pesos da rede não são modificados e o próximo exemplo é apresentado à rede; caso contrário, os pesos devem ser modificados utilizando a Equação 2.2:

$$w_i = w_i + \Delta w_i \tag{2.2}$$

onde

$$\Delta w_i = (y - O) x_i$$

O objetivo do treinamento de uma rede *perceptron* é encontrar uma combinação

linear dos pesos e *threshold* que consiga separar os exemplos de uma classe C_1, dos exemplos de uma classe C_2 (Weiss & Kulikowski 1991). Quando o problema possui apenas 2 atributos, o treinamento se resume a encontrar uma reta que separe os exemplos, como mostrado na Figura 2.10. Nesse caso, o *threshold* indica em que ponto a reta intercepta o eixo y. Deve ser observado que a rede *perceptron* somente consegue resolver problemas linearmente separáveis.

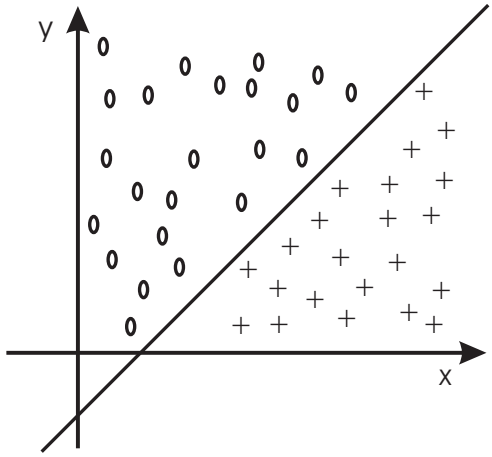

Figura 2.10: Interpretação geométrica para a rede *perceptron*.

Em (Widrow & Hoff 1960) é sugerida uma regra de aprendizado semelhante à regra de aprendizado da rede *perceptron*, conhecida como *Regra Delta*. A principal diferença entre elas é que no aprendizado da rede *perceptron* a saída O é mapeada para os valores 0 ou 1 por meio da função *threshold*. Já no aprendizado utilizando a Regra Delta, a função de ativação comumente utilizada é a função linear. A Regra Delta foi utilizada para o treinamento da rede ADALINE.

Durante o ajuste dos pesos, a Regra Delta tenta minimizar a distância entre a saída da rede, ou seja, a classe predita pela rede, O, e a classe verdadeira do exemplo, y. Quando essa distância é reduzida para um valor desejável durante o treinamento, o mesmo é convergido.

A principal limitação das redes *perceptron* e ADALINE é que elas não conseguem solucionar problemas que não são linearmente separáveis. Problemas mais complexos necessitam de redes com mais camadas, como a rede ilustrada na Figura 2.8 na página 31, denominada *perceptron* multi-camadas — MLP[5] . Esse tipo de rede pode ser treinada com uma regra de aprendizado denominada *Regra Delta Generalizada*, mais conhecida como

[5] *Multi-Layer Perceptron.*

algoritmo *backpropagation* (Rumelhart, Hilton & Williams 1986). O algoritmo *backpropagation* é um dos mais populares e utilizados algoritmos de treinamento de RNAs.

Assim como na rede *perceptron* de uma camada, na rede MLP um exemplo é apresentado à rede, que calcula sua saída, compara-a à saída desejada e então altera os pesos da rede para conseguir uma saída mais próxima à desejada na próxima iteração. Ao apresentar um exemplo a uma rede não treinada, esta produzirá qualquer saída. Para que a rede aprenda os exemplos durante o treinamento é necessário que a saída da rede aproxime-se cada vez mais da saída desejada, isto é, que o valor da distância entre a saída da rede e a saída desejada seja reduzido continuamente. Isto é conseguido ajustando-se os pesos entre as unidades de processamento. A Regra Delta Generalizada, que é uma versão generalizada da Regra Delta para redes com camadas intermediárias, faz isso calculando a distância entre a saída da rede e a saída desejada das unidades da camada de saída para uma entrada particular, e então esse valor é propagado para trás, de uma camada para sua camada anterior. Cada unidade de processamento da rede tem seus pesos ajustados, reduzindo-se assim a distância entre a saída da rede e a saída desejada, até um determinado limite, quando diz-se então que a rede aprendeu.

A função de ativação geralmente utilizada pelas redes MLPs é a função logística dada pela Equação 2.3:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

O intervalo de ativação da função logística é $[0, 1]$. Como pode ser visto na Figura 2.11, a função logística é uma aproximação contínua da função *threshold*.

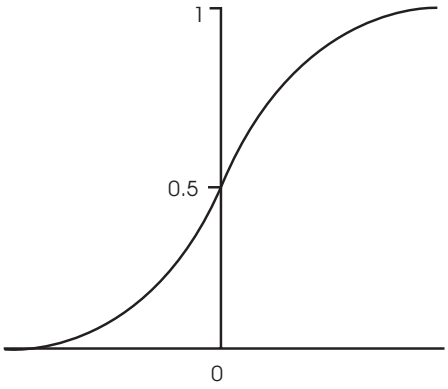

Figura 2.11: Função logística.

Uma rede com várias camadas é capaz de resolver problemas não linearmente se-

paráveis, como o ilustrado na Figura 2.12. Nesse exemplo, a unidade de processamento 1 encontrou a reta pontilhada, a unidade de processamento 2 encontrou a reta sólida e a unidade de processamento 3 pode combinar essas duas retas e classificar os exemplos corretamente.

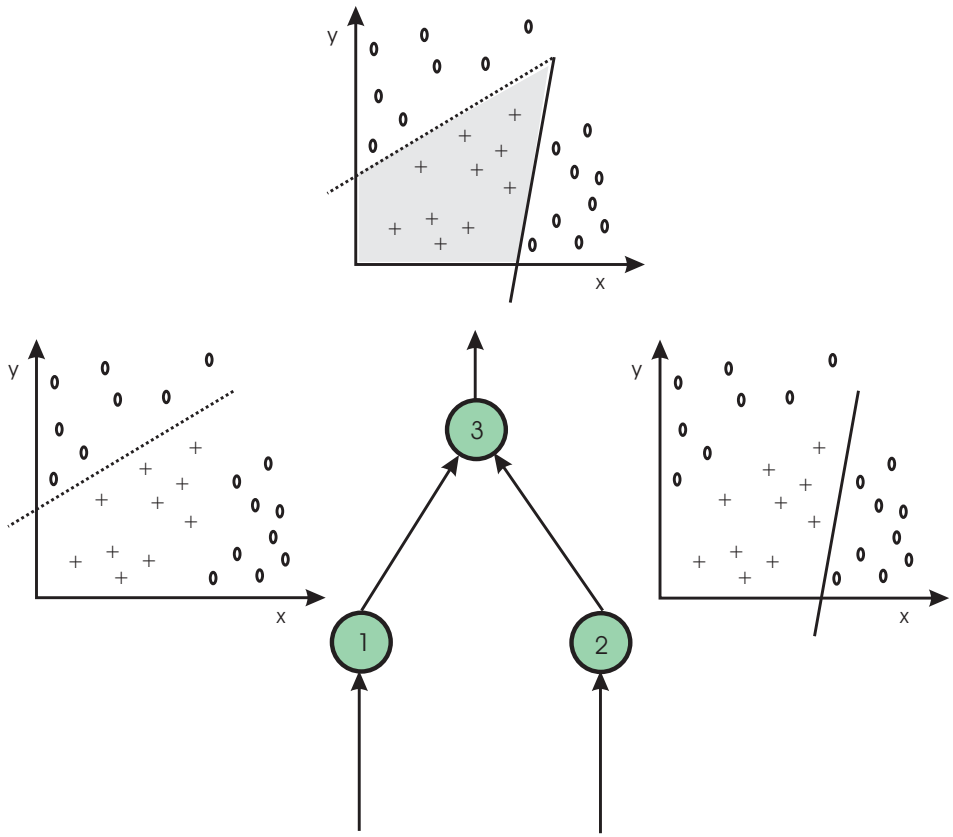

Figura 2.12: Interpretação geométrica para a rede MLP.

Para classificar um novo exemplo utilizando uma RNA treinada basta calcular a saída O da rede. No caso da rede *perceptron*, a saída O é calculada utilizando a Equação 2.1. Nesta fase, determinada de fase de uso da RNA, os pesos não são ajustados.

Uma das características mais importantes de RNAs é que as mesmas são aproximações universais de funções multivariáveis contínuas (Cybenko 1989). Isso significa que qualquer problema de aproximação de funções pode ser resolvido por meio de RNAs, independentemente do número de variáveis envolvidas (Braga, Carvalho & Ludemir 2000).

2.5 Algoritmos Genéticos

Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução dos seres vivos. Foram introduzidos por Holland (1975) e popularizados por um dos seus alunos, Goldberg (1989). Os AGs possuem uma analogia direta com a teoria do naturalista e fisiologista inglês Darwin (1859), segundo a qual quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.

Um AG é um procedimento iterativo que mantém uma população de indivíduos. Cada um dos indivíduos é um candidato à solução de um problema específico. A cada iteração, denominada de geração, os indivíduos da população atual são avaliados quanto à sua aptidão para a solução do problema. Com base nessas avaliações aplicam-se alguns operadores genéticos aos indivíduos, formando-se uma nova população, que substituirá a população atual. Isto é feito de modo que, quanto maior a aptidão de um indivíduo atual, maior a sua influência na formação dos indivíduos da nova população. Assim, com o passar do tempo, a seleção tende a fazer com que a população seja formada por indivíduos cada vez melhores (soluções cada vez mais próximas da solução ótima para o problema). Como critério de parada do algoritmo genético, geralmente define-se um limite para o número de gerações. Dentre os indivíduos da última geração, aqueles mais aptos representam a melhor solução encontrada pelo algoritmo. Pode-se também especificar que o algoritmo encerrará quando for gerado algum indivíduo que satisfaça alguma condição mínima de aptidão (Freitas & Kirner 1992). A Figura 2.13 mostra uma geração de um AG. Os operadores genéticos *crossover* e mutação serão descritos posteriormente.

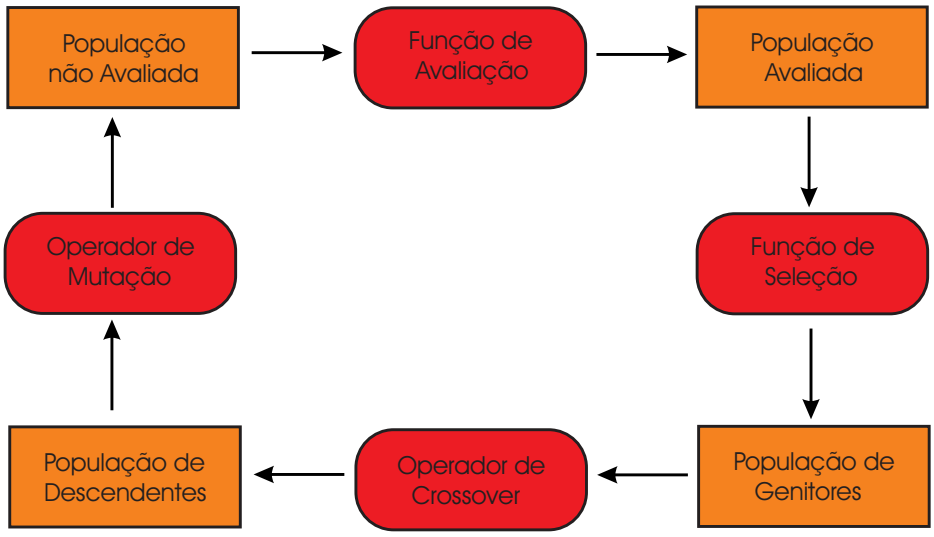

Figura 2.13: Geração de um AG.

Uma importante propriedade dos AGs é que eles mantém uma população de soluções potenciais. Com isso, a cada iteração, a busca é realizada paralelamente, a partir de vários pontos do espaço de busca, o que difere dos outros métodos de busca que processam, a cada iteração, um único ponto do espaço de busca.

2.5.1 Componentes de um AG

Um algoritmo genético clássico possui alguns componentes, entre eles:

Representação do Indivíduo

Os indivíduos do AG, também denominados de cromossomos, representam soluções potenciais de um problema. A representação mais tradicional é a binária, na qual um indivíduo é representado por meio de uma cadeia de 0's e $1's$ de tamanho fixo. Para problemas nos quais a representação binária não é a mais natural e apropriada, existem outros tipos de representações que podem ser utilizadas (Michalewicz 1999).

Geração da População Inicial

O AG precisa de um mecanismo para gerar uma população inicial de soluções potenciais para a resolução de um problema. Na maioria dos AGs a população inicial é criada de forma aleatória.

Função de Aptidão

A função de aptidão[6] determina a aptidão de cada indivíduo, ou seja, quão boa a solução por ele representada é para a resolução de um problema. Ela é responsável por determinar se o indivíduo continuará fazendo parte da população ou não. A escolha da função de avaliação está diretamente relacionada ao domínio do problema.

Método de Seleção

Uma vez que os AGs baseiam-se no princípio da seleção natural, eles devem ser capazes de identificar os indivíduos mais aptos, para que permaneçam na população durante o processo de evolução (procedimento iterativo), e os mais fracos, para que sejam excluídos do processo. Existem vários métodos que podem ser utilizados para selecionar indivíduos para formar uma nova população (Michalewicz 1999; Lacerda & de Carvalho 1999). Um critério de seleção bastante comum em AG é a seleção proporcional, que consiste em selecionar os indivíduos com probabilidade proporcional à sua aptidão. Conceitualmente, a seleção proporcional é equivalente a dividir

[6] *Fitness.*

uma roleta em n_i partes, sendo n_i o número de indivíduos da população. Cada uma das partes da roleta é proporcional à aptidão do indivíduo associado àquela parte. A roleta é girada n_i vezes, e a cada uma delas o indivíduo indicado pelo ponteiro é selecionado e inserido na nova população.

Com o objetivo de preservar e usar na próxima geração as melhores soluções encontradas na geração atual, pode-se utilizar uma estratégia denominada *elitismo*. Essa estratégia consiste em conservar os m_i melhores indivíduos da população atual, copiando-os para a próxima geração sem nenhuma alteração. O restante dos indivíduos, $n_i - m_i$, são normalmente gerados por meio do método de seleção.

Operadores Genéticos

Os operadores genéticos quando aplicados à população geram novos indivíduos e são os principais mecanismos utilizados pelos AGs para explorar regiões desconhecidas do espaço de busca. *Crossover* e mutação são os operadores genéticos mais utilizados.

crossover

Esse operador genético permite a troca de informação entre soluções diferentes. Ele combina características de dois indivíduos pais, para formar dois indivíduos filhos por meio da troca de segmentos correspondentes dos indivíduos pais. Por exemplo, considere os seguintes indivíduos pais representados por uma cadeia de $O's$ e $1's$ de comprimento 8:

$$\begin{array}{ll} pai_1 & \textit{0 0 1 0 0 0 0 0} \\ pai_2 & 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \end{array}$$

Aplicando o operador *crossover* depois da terceira posição,

$$\begin{array}{ll} pai_1 & \textit{0 0 1} \mid \textit{0 0 0 0 0} \\ pai_2 & 1\ 0\ 0 \mid 1\ 1\ 1\ 1\ 1 \end{array}$$

são gerados os seguintes indivíduos filhos:

$$\begin{array}{ll} filho_1 & \textit{0 0 1}\ 1\ 1\ 1\ 1\ 1 \\ filho_2 & 1\ 0\ 0\ \textit{0 0 0 0 0} \end{array}$$

mutação

Esse operador genético introduz variabilidade dentro da população e consiste em arbitrariamente alterar o valor de um ou mais genes[7] de um indivíduo. Por

[7] Os indivíduos (também conhecidos por cromossomos) da população de um AG são formados de unidades adjacentes chamadas de *genes*. Os genes codificam um elemento particular de uma solução.

exemplo, considere o indivíduo abaixo representado por uma cadeia de $O's$ e $1's$ de comprimento 8:

1 0 0 0 0 0 0 1

Aplicando o operador de mutação na quarta posição é gerado o seguinte indivíduo:

1 0 0 *1* 0 0 0 1

Valores de Parâmetros

Um AG utiliza vários parâmetros, entre eles: o tamanho da população, o número de genes dos indivíduos, a probabilidade de aplicação dos operadores genéticos, o número de gerações, etc. Os valores para esses parâmetros são definidos geralmente por tentativa e erro.

2.5.2 Exemplo

Para ilustrar o funcionamento de um AG, considere um exemplo bem simples apresentado em (Mitchell 1997a), no qual os indivíduos são representados utilizando um vetor binário de comprimento 8 e a função de aptidão é igual ao número de $1's$ do indivíduo. Os outros parâmetros são: tamanho da população $n_i = 4$, probabilidade de *crossover* $p_c = 0.5$ e probabilidade de mutação $p_m = 0.1$. Todos os valores dos parâmetros do AG, assim como a função de aptidão, foram escolhidos para simplificar o exemplo.

Inicialmente os indivíduos da população são criados. Todos os 8 elementos do vetor binário representando cada gene do indivíduo são inicializados aleatoriamente. Os quatro indivíduos abaixo formam a população inicial:

$$A = (0\ 0\ 0\ 0\ 0\ 1\ 1\ 0)$$
$$B = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 0)$$
$$C = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$$
$$D = (0\ 0\ 1\ 1\ 0\ 1\ 0\ 0)$$

O próximo passo é calcular a função de aptidão para cada indivíduo. Lembrando que a função de aptidão, para esse exemplo, é o número de $1's$ presentes em cada indivíduo. O cálculo da função de aptidão para os quatro indivíduos é:

$$A = (0\ 0\ 0\ 0\ 0\ 1\ 1\ 0) = 2$$
$$B = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 0) = 6$$
$$C = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0) = 1$$
$$D = (0\ 0\ 1\ 1\ 0\ 1\ 0\ 0) = 3$$

Deve-se agora selecionar alguns indivíduos para formar um nova população. Considere que, utilizando a seleção proporcional, foram selecionados para formar a nova população os seguintes indivíduos:

$$A' = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 0) = 6\ (\text{B})$$
$$B' = (0\ 0\ 1\ 1\ 0\ 1\ 0\ 0) = 3\ (\text{D})$$
$$C' = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 0) = 6\ (\text{B})$$
$$D' = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0) = 1\ (\text{C})$$

Note que o indivíduo A não foi selecionado, enquanto que o indivíduo B foi selecionado duas vezes. Agora pode-se aplicar o operador genético *crossover* na nova população. Com probabilidade de *crossover* $p_c = 0.5$, espera-se que (na média) 50% dos indivíduos, ou seja, dois indivíduos, sofram *crossover* em cada geração. Considere que o operador de *crossover* seja aplicado nos indivíduos A' e B' depois da primeira posição, formando os indivíduos $A'' = 10110100$ e $B'' = 01101110$. A população atual agora é formada pelos seguintes indivíduos:

$$A'' = (1\ 0\ 1\ 1\ 0\ 1\ 0\ 0)$$
$$B'' = (0\ 1\ 1\ 0\ 1\ 1\ 1\ 0)$$
$$C' = (1\ 1\ 1\ 0\ 1\ 1\ 1\ 0)$$
$$D' = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$$

O próximo operador genético a ser aplicado é o operador de mutação, com probabilidade $p_m = 0.1$. Espera-se que 10% (na média) dos elementos do vetor (genes), que representam cada indivíduo, sofram mutação. Neste caso, há $4 \times 8 = 32$ genes na população, portanto, são esperadas 3.2 mutações (em média) em cada geração. Considere que o indivíduo A'' seja mutado no sexto gene, o indivíduo C' no segundo gene e o indivíduo D' no primeiro gene. A população agora é constituída dos seguintes indivíduos:

$$A = (1\ 0\ 1\ 1\ 0\ 0\ 0\ 0)$$
$$B = (0\ 1\ 1\ 0\ 1\ 1\ 1\ 0)$$
$$C = (1\ 0\ 1\ 0\ 1\ 1\ 1\ 0)$$
$$D = (1\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$$

Com isto, uma geração é completada. Note que o melhor indivíduo foi perdido, mas a aptidão média da população aumentou de 12/4 para 15/4.

2.6 Considerações Finais

Neste capítulo foram descritos alguns sistemas de AM simbólico, RNAs e também AGs, pois estes são utilizados neste trabalho.

Como mencionado anteriormente, em diversas aplicações não é só a precisão dos classificadores induzidos pelos sistemas de AM que deve ser considerada. Nessas aplicações, outro fator muito importante é o grau de compreensibilidade proporcionado ao ser humano. Em relação à compreensibilidade, os sistemas de AM podem ser classificados em sistemas "caixa preta" e sistemas orientados a conhecimento (Michalski 1983), como descrito a seguir.

Sistemas "caixa-preta"

Os sistemas desse tipo desenvolvem sua própria representação do conceito, isto é, sua representação interna pode não ser facilmente interpretada por humanos e não fornece nenhum esclarecimento e explicação do processo de classificação de novos exemplos.

Sistemas Orientados a Conhecimento

As hipóteses induzidas por sistemas desse tipo são representadas por estruturas simbólicas que são compreensíveis por humanos.

Árvores de decisão e regras de decisão são exemplos de sistemas orientados a conhecimento, já que representam as hipóteses induzidas por meio de estruturas simbólicas, as quais são legíveis e interpretáveis por humanos. RNAs são exemplos de sistemas "caixa-preta", pois a representação da hipótese por elas induzida é difícil de ser compreendida por humanos.

No próximo capítulo assuntos referentes à compreensibilidade das hipóteses induzidas por RNAs são analisadas, bem como são comentados alguns trabalhos propostos para tornar essas hipóteses compreensíveis.

Capítulo 3

Extração de Conhecimento de RNAs

3.1 Considerações Iniciais

Diversos pesquisadores têm investigado como converter hipóteses induzidas por RNAs em representações de mais fácil compreensão para os seres humanos (Andrews, Diederich & Tickle 1995). Essa área de pesquisa é geralmente denominada de *extração de conhecimento de* RNAs. Um segundo nome bastante utilizado para denominar essa área de pesquisa é *extração de regras de* RNAs. O termo *regras* é utilizado porque *regras de decisão* são freqüentemente usadas como linguagem para a representação do conhecimento extraído das RNAs. Entretanto, esse nome não é tão abrangente quanto o primeiro, o qual engloba todos os métodos de extração de conhecimento, independentemente da representação utilizada.

Regras de decisão constituem uma linguagem amplamente utilizada para representar conhecimento. Essa linguagem foi escolhida para representar o conhecimento induzido por diversos sistemas de AM simbólico (Boswell 1990a; Boswell 1990b; Quinlan 1988; Cameron-Jones & Quinlan 1993).

Neste capítulo é realizado um levantamento bibliográfico sobre os métodos que utilizam regras de decisão e árvores de decisão para representar o conhecimento extraído de uma RNA, uma vez que regras de decisão são provavelmente a linguagem de representação mais utilizadas por esses métodos. Como descrito no Capítulo 2, árvores de decisão podem ser traduzidas facilmente em um conjunto de regras de decisão. Entretanto, quando métodos de extração de conhecimento são utilizados, o conhecimento obtido não precisa necessariamente estar na forma de regras de decisão. Outras linguagens que são utilizadas

para expressar o conhecimento extraído de uma RNA são valores estatísticos e protótipos de agrupamentos.

Este capítulo está organizado nas seguintes seções: na Seção 3.2 são descritas algumas restrições impostas pelos algoritmos de aprendizado às hipóteses por eles induzidas; na Seção 3.3 é discorrido sobre linguagens utilizadas pelos algoritmos de aprendizado para representar hipóteses; na Seção 3.4 é discutida a seguinte questão: por que utilizar RNAs para solucionar problemas para os quais é fundamental a compreensibilidade da hipótese induzida?; na Seção 3.5 são analisadas as razões da dificuldade em se compreender as hipóteses induzidas por RNAs; na Seção 3.6 é descrita a importância em se entender as hipóteses induzidas por RNAs; na Seção 3.7 é descrito um processo muito simples de extração de regras de uma RNA; na Seção 3.8 são apresentados os critérios propostos na literatura para a avaliação e classificação de métodos de extração de conhecimento de RNAs; na Seção 3.9 são descritos brevemente alguns métodos de extração de conhecimento de RNAs; na Seção 3.10, o método TREPAN é descrito detalhadamente por ser utilizado neste trabalho; e, por fim, na Seção 3.11 são feitas algumas considerações finais sobre os assuntos tratados neste capítulo.

3.2 *Bias* Indutivo

Como já mencionado, o principal objetivo de um sistema indutivo é aprender uma função **h** que aproxima a função conceito f, a qual é desconhecida — Seção 1.2 na página 2. Para a maioria dos problemas reais, os conjuntos de dados possuem tamanho reduzido, e não existem exemplos suficientes para caracterizar completamente a função f. Dessa forma, dado um conjunto de exemplos, existem diferentes hipóteses que podem ser induzidas a partir dos exemplos desse conjunto. Por exemplo, suponha que exemplos de um determinado conceito são representados como pontos no plano, como mostrado na Figura 3.1 em (a). Uma possível hipótese, que aproxima a verdadeira função f, poderia ser obtida conectando cada ponto com o próximo por meio de linhas retas, como é mostrado em (b). Outra possível aproximação para a função conceito poderia ser a utilização de uma função polinomial suave, como pode ser visto em (c). Uma terceira possibilidade seria a hipótese induzida mostrada em (d), que ignora um dos exemplos.

Qualquer preferência que um algoritmo de AM dá a uma hipótese sobre outra, além da simples consistência com os exemplos observados durante o aprendizado, é denominada *bias de aprendizado* ou *bias indutivo* (Russel & Norvig 2003). Como quase sempre existe

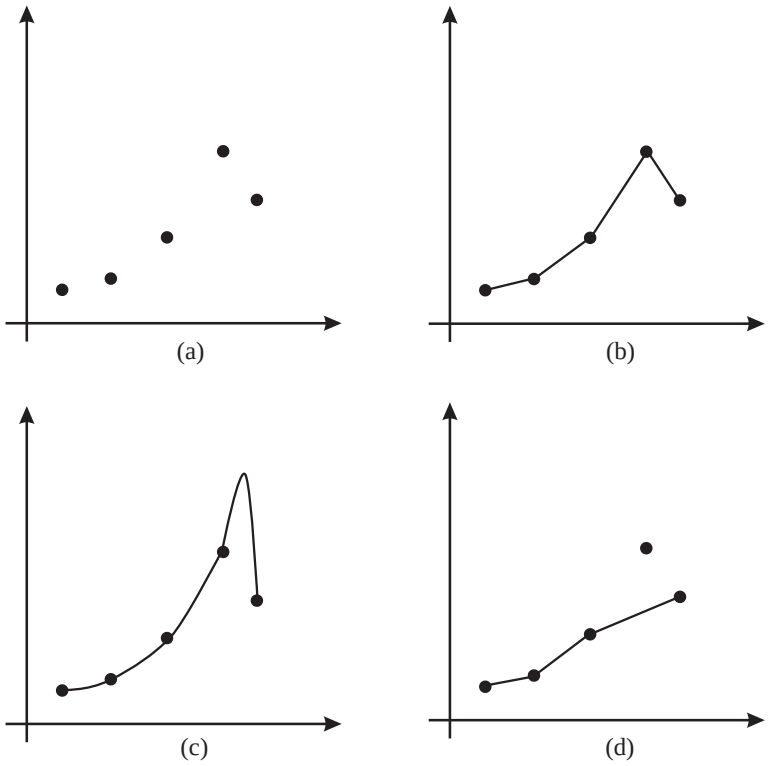

Figura 3.1: Possíveis hipóteses (b), (c) e (d) induzidas a partir de um conjunto de exemplos representado em (a) (Russel & Norvig 1995).

um número grande de hipóteses consistentes, todos os indutores possuem alguma forma de *bias*.

O *bias* indutivo de um algoritmo de aprendizado pode ser dividido em: *bias* absoluto e relativo, descritos a seguir.

Bias Absoluto

Bias absoluto ou de restrição do espaço de hipóteses[1] refere-se às restrições que um algoritmo de aprendizado impõe sobre a hipótese que esse algoritmo pode aprender. Normalmente, essas restrições são definidas pela linguagem de representação da hipótese adotada pelo algoritmo de aprendizado. Por exemplo, uma rede *perceptron* limita a busca ao espaço de funções lineares, enquanto que árvores de decisão traçam hiperplanos paralelos aos eixos.

Bias Relativo

Bias relativo ou *bias* de preferência[2] refere-se à preferência por algumas das várias

[1] *Restricted hypothesis space bias.*
[2] *Preference bias.*

hipóteses que um sistema de aprendizado pode induzir a partir de um conjunto de exemplos. Por exemplo, muitos algoritmos de aprendizado inicialmente consideram uma hipótese simples para um determinado conjunto de dados. Caso a hipótese não seja satisfatória, esses algoritmos exploram progressivamente hipóteses mais complexas até encontrarem uma hipótese aceitável.

Sabe-se que não existe um algoritmo de aprendizado que seja sempre superior aos demais (Wolpert 1995). Isto é, não existe um algoritmo de aprendizado que possua um desempenho superior para todos os domínios de problemas. Portanto, muitas vezes, uma das alternativas consideradas para melhor solucionar um problema de aprendizado consiste em selecionar o algoritmo com o *bias* indutivo mais adequado para o problema em questão. Geralmente isso é realizado estimando-se a precisão dos classificadores utilizando um método de reamostragem, como por exemplo o *cross-validation* (Batista 1997; Weiss & Kulikowski 1991).

3.3 Representação de Hipóteses

Para solucionar problemas com o uso de computadores é importante definir como traduzi-los em termos computacionais. Em AM isso significa como representar exemplos, hipóteses e conhecimento do domínio[3]. As linguagens utilizadas para representar exemplos, hipóteses e conhecimento do domínio são denominadas de *linguagens de representação* ou *linguagens de descrição*, mais especificamente:

- linguagem de representação de exemplos;
- linguagem de representação de hipóteses;
- linguagem de representação de conhecimento de domínio.

Algoritmos de aprendizado diferem consideravelmente em como eles representam as hipóteses induzidas. Por exemplo, existem algoritmos de aprendizado que representam suas hipóteses como árvores de decisão (Breiman, Friedman, Olshen & Stone 1984; Quinlan 1988), listas de decisão (Rivest 1987; Clark & Niblett 1989), valores numéricos como

[3]Conhecimento de domínio (*background knowledge*) ou conhecimento de fundo, é o conhecimento prévio a respeito do domínio. Por exemplo, inclui informações a respeito dos valores válidos dos atributos, um critério de preferência para a escolha entre possíveis atributos ou mesmo hipóteses, restrições de relacionamentos entre atributos, novos atributos possivelmente derivados dos atributos originais, alguma hipótese inicial, entre outras informações.

RNAs (Rumelhart, Hilton & Williams 1986), regras de decisão (Michalski 1983; Quinlan 1988), cadeias de Markov (Rabiner 1989), redes Bayesianas (Herkerman 1995), e até mesmo utilizando o próprio conjunto de exemplos, como é feito pelos métodos baseados em exemplos[4] (Aha, Kibler & Albert 1991; Stanfill & Waltz 1986). Um algoritmo de aprendizado é limitado às hipóteses que pode induzir pela capacidade representacional de sua linguagem de representação (Mitchell 1980).

As várias representações para expressar as hipóteses aprendidas diferem em quão compreensíveis elas podem ser para os humanos. Linguagens de representação de hipóteses que têm uma sintaxe lógica (como por exemplo, árvores de decisão e regras de decisão), são denominadas representações simbólicas (Michalski 1986). Uma vantagem das representações simbólicas é que elas geralmente são fáceis de serem entendidas pelos seres humanos. Por exemplo, em uma árvore de decisão é fácil verificar quais atributos fazem parte da hipótese e também os relacionamentos importantes existentes entre os atributos da hipótese.

Ao contrário de sistemas de AM simbólico, que utilizam representação simbólica para expressar suas hipóteses, outros sistemas de aprendizado, como por exemplo RNAs, cadeias de Markov e redes Bayesianas, representam suas hipóteses utilizando estruturas de grafos com parâmetros de valores numéricos. Algumas dessas representações também podem ser compreensíveis. Por exemplo, em redes Bayesianas, os nós dos grafos geralmente correspondem a atributos específicos do domínio do problema, e as arestas correspondem às dependências conhecidas entre os atributos. Por outro lado, em RNAs, muitas unidades e conexões não têm tais correspondências com o vocabulário do domínio do problema, dificultando assim a compreensão das hipóteses aprendidas por RNAs. Não é fácil verificar em uma RNA, por exemplo, quais atributos possuem contribuições importantes para a saída da rede. Além disso, muitas vezes é difícil verificar como os atributos interagem em uma determinada hipótese.

3.4 Por Que Utilizar Redes Neurais?

Se a compreensibilidade é um aspecto importante em um determinado domínio de problema, por que então utilizar RNAs para solucionar esse problema? Por que não, ao invés de uma RNA, utilizar outros sistemas de aprendizado que produzem hipóteses que são mais compreensíveis para os seres humanos? A resposta é que para muitos desses proble-

[4] *instance-based methods.*

mas, RNAs podem possuir um *bias* indutivo mais apropriado que os demais sistemas de aprendizado.

Em alguns casos como, por exemplo, em problemas de predição temporal e seqüencial, RNAs podem utilizar um *bias* de restrição de espaço de hipótese mais apropriado que outros algoritmos de aprendizado. Redes recorrentes (Jordan 1986; Pineda 1987), as quais geralmente são utilizadas para resolver esse tipo de problema, possuem um mecanismo que faz com que a informação de um passo prévio seja mantida no passo seguinte. Isso significa que redes recorrentes podem utilizar suas unidades intermediárias para aprender atributos derivados que sejam relevantes para o problema que está sendo considerado. Além disso, essas redes podem utilizar o estado desses atributos derivados do passo atual para ajudar na resposta da rede para classificar o próximo exemplo no próximo passo.

Em outros casos, RNAs são escolhidas como sistemas de aprendizado porque induzem hipóteses que generalizam melhor que outros algoritmos. Alguns estudos empíricos têm mostrado que em alguns domínios de aplicação, o desempenho obtido por RNAs é superior ao desempenho de alguns dos algoritmos de AM simbólico mais conhecidos (Atlas, Cole, Connor, El-Sharkawi, II, Muthusamy & Bernard 1989; Shavlik, Mooney & Towell 1991; Fisher & McKusick 1989; Weiss & Kapouleas 1989).

3.5 Redes Neurais e Compreensibilidade

A hipótese aprendida por uma RNA é definida por três características principais:

- topologia da rede;
- funções de ativação utilizadas nas unidades intermediárias e de saída, e;
- valores numéricos associados às conexões da rede, ou seja, os pesos, bem como o valor do *threshold* de cada unidade.

As hipóteses aprendidas por RNAs são de difícil compreensão por diversas razões. Uma dessas razões é a grande quantidade de pesos que uma rede pode possuir. Esses pesos codificam o relacionamento entre os atributos de entrada e o(s) valor(es) de saída. Embora os pesos em uma rede bem simples e pequena como, por exemplo, em uma rede *perceptron* permitam uma compreensão mais fácil do comportamento dessa rede, para uma rede típica, o grande número de pesos pode tornar o entendimento dessa rede uma tarefa muito complicada, senão impossível.

Em redes com várias camadas, os pesos representam relacionamentos não lineares entre os valores dos atributos de entrada e os valores de saída. Por essa razão não é possível determinar o efeito de um determinado atributo sobre um valor de saída, apenas isolando esse atributo de entrada. Esse efeito também deve ser medido pelos valores de outros atributos. O relacionamento não linear entre os valores dos atributos de entrada e o(s) valor(es) da saída da rede é representado pelas unidades intermediárias que combinam as entradas de vários atributos, permitindo à RNA "reconhecer" e aprender as dependências entre os atributos.

Entender o que representam as unidades intermediárias não é fácil porque estas unidades muitas vezes aprendem "representações distribuídas" (Hilton 1986). Unidades intermediárias podem ser vistas como representações de alto nível ou como "atributos derivados". Na representação distribuída, esses atributos derivados podem não fazer sentido ou não corresponder ao vocabulário do domínio do problema que está sendo considerado. Ao invés disso, atributos que são significantes em um contexto de um domínio de problema são muitas vezes codificados por padrões de ativação por meio de muitas unidades intermediárias. Similarmente, cada unidade intermediária pode representar vários atributos derivados.

3.6 Importância da Compreensibilidade

Até recentemente, a consideração mais importante em um sistema de aprendizado indutivo supervisionado era induzir um modelo que possuísse alto desempenho, ou seja, que fosse capaz de predizer corretamente as classes dos exemplos que não foram vistos durante a indução.

Entretanto, a compreensibilidade dos modelos aprendidos também é uma importante consideração. Compreensibilidade refere-se a como os algoritmos de aprendizado representam seus modelos, e a como esses modelos podem ser inspecionados e entendidos pelos seres humanos. De acordo com Andrews, Diederich & Tickle (1995), existem várias razões que justificam a utilização de um método para extração de conhecimento de RNAs, tais como:

Exploração de Dados e Dedução de Novas Teorias

Sistemas de aprendizado desempenham um papel importante no processo de descobertas científicas. Um sistema pode descobrir novos relacionamentos e características importantes nos dados de entrada, permitindo a formulação de novas teorias. A extração de conhecimento de RNAs pode levar à descoberta de dependências e relacionamentos entre os dados;

Facilitar a Aceitação pelo Usuário

Uma das características mais elogiadas pelos usuários de sistemas de aprendizado simbólico é a explicação das decisões tomadas por esses sistemas. Tem sido verificado que a capacidade de gerar explicações, mesmo limitadas (em termos de utilidade e coerência), é um aspecto crucial para a aceitação desses sistemas pelos seus usuários. No caso de RNAs, por exemplo, a falta de regras explícitas que fundamentem as decisões apresentadas pela rede dificulta sua aceitação pelos usuários. Essa deficiência constitui uma barreira clara à expansão de seu uso;

Melhoria da Generalização das Soluções da Rede

Por expressar o conhecimento adquirido pela rede durante o seu treinamento, o conhecimento simbólico pode ser utilizado para descobrir as circunstâncias na qual a rede pode cometer erros de generalização. O usuário também pode utilizar esse conhecimento extraído para identificar regiões no espaço de entradas que não estão suficientemente representadas no conjunto de treinamento. Dados de treinamento adicionais podem ser então incluídos para melhorar o desempenho da rede;

Integração com Sistemas de AM Simbólico

A representação do conhecimento induzido por uma RNA, como um conjunto de regras, pode ser utilizado para comunicação das redes com sistemas de aprendizado simbólico. As regras criam uma espécie de linguagem comum entre as duas técnicas, facilitando a integração das mesmas;

Redefinição da Rede

O conhecimento extraído da rede pode ainda ser utilizado para verificar a adequação da arquitetura escolhida para a aplicação na qual a rede está sendo utilizada.

3.7 Extraindo Regras de RNAs

Uma abordagem para entender a hipótese representada por uma RNA treinada é tentar extrair algum conhecimento dessa rede e representá-lo em uma linguagem que possa ser compreensível. Regras de decisão constituem uma linguagem bastante compreensível, que pode ser utilizada para representar o conhecimento extraído de uma RNA treinada.

Extração de regras é um caso especial de extração de conhecimento e é a forma mais utilizada para representar o conhecimento extraído de uma RNA. Este trabalho propõe o uso dessa abordagem e essa é uma das razões pelas quais o processo de extração de regras de uma RNA treinada é detalhado neste capítulo.

Várias definições têm sido propostas para extração de regras de RNAs. Duas definições que consideramos apropriadas são:

> *Em essência, o processo de extrair explicações (ou regras) de uma* RNA *treinada é interpretar em uma forma compreensível o efeito coletivo da topologia da rede; das funções de transferência utilizadas nas unidades intermediárias e de saída; e, dos valores numéricos associados com as conexões da rede (os pesos), bem como do valor do* threshold *de cada unidade* (Andrews, Diederich & Tickle 1995).

> *Dada uma* RNA *treinada e os exemplos utilizados para treiná-la, extrair explicações consiste em produzir uma descrição da hipótese da rede que seja compreensível, e ainda que seja fiel ao seu comportamento quanto à predição* (Craven 1996).

Na Figura 3.2 é mostrado um exemplo de uma RNA muito simples, um *perceptron*, com apenas um neurônio de saída. Os valores de entrada e a saída são valores booleanos. Na Tabela 3.1 são mostradas três regras extraídas a partir dessa rede.

Y	$\leftarrow$	A_1	$\wedge$	A_2	$\wedge$	A_3
Y	$\leftarrow$	A_1	$\wedge$	A_2	$\wedge$	$\neg A_5$
Y	$\leftarrow$	A_1	$\wedge$	A_3	$\wedge$	$\neg A_5$

Tabela 3.1: Regras extraídas.

Redes como a da Figura 3.2, com atributos discretos de entrada e saída, podem ser descritas por um conjunto de regras. As regras especificam condições sobre os atributos de

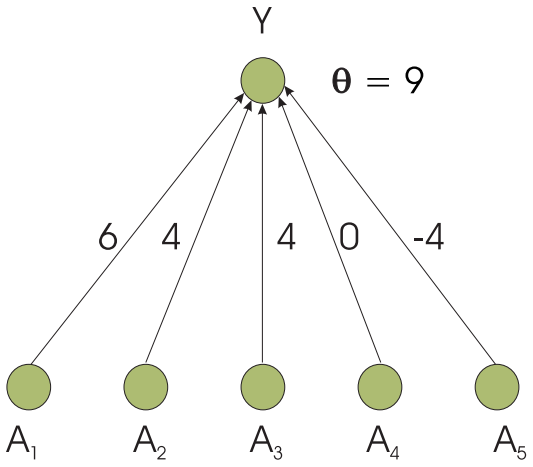

Figura 3.2: Rede *perceptron* com pesos.

entrada que, quando satisfeitas, garantem um determinado estado de saída. No exemplo é assumido que o valor `falso` para um atributo de entrada booleano é representado por uma ativação igual a 0, e o valor `verdadeiro` é representado por uma ativação igual a 1. Também é assumido que o valor da unidade de saída Y utiliza a função apresentada na Equação 3.1 para computar sua ativação.

$$O = \begin{cases} 1 & \text{Se } \sum_i w_i x_i - \theta > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{3.1}$$

Na Equação 3.1, O é a ativação da unidade de saída, x_i é a ativação da i-ésima unidade de entrada, w_i é o peso da conexão entre a i-ésima unidade de entrada e a unidade de saída, e θ é o *threshold* da unidade de saída. Nesse exemplo, se o valor do atributo A_i for `verdadeiro`, então $x_i = 1$.

Na Tabela 3.1 são mostradas três regras que descrevem em que condições a unidade de saída tem valor de ativação igual a 1. Considere a regra a seguir:

$$O \leftarrow A_1 \wedge A_2 \wedge \neg A_5$$

Essa regra especifica que quando A_1 = `verdadeiro`, A_2 = `verdadeiro` e A_5 = `falso`, a ativação da unidade de saída, representada por O, terá uma ativação igual a 1, ou seja, a rede prediz que Y = `verdadeiro`. Para verificar que esta é uma regra válida, basta substituir na Equação 3.2 os valores de ativação dos atributos de entrada e os pesos

correspondentes a esses atributos.

$$x_1 w_1 + x_2 w_2 + x_5 w_5 - \theta > 0 \tag{3.2}$$

$$1 \times 6 + 1 \times 4 + 0 \times (-4) - 9 > 0$$

A soma dos pesos multiplicados pelos valores de ativação dos atributos de entrada faz com que o *threshold* da unidade de saída seja excedido. Mas, nesse caso, qual o efeito dos outros atributos, aqueles atributos que não fazem parte da regra, sobre a ativação da unidade de saída? Supondo que o valor de ativação da unidade de saída seja definido pela Equação 3.3:

$$0 \leq x_3 w_3 + x_4 w_4 \leq 4 \tag{3.3}$$

nesse caso, pode-se notar que independentemente dos valores que os atributos A_3 e A_4 assumam, a ativação da unidade de saída sempre será 1. Portanto, a regra é válida e descreve o comportamento da rede para aqueles exemplos cujos valores "casam" com os antecedentes da regra. Para verificar se a regra é geral, considere que se qualquer literal dos antecedentes da regra for retirado, a regra não mais descreve o comportamento da rede para aqueles exemplos. Por exemplo, se o literal $\neg A_5$ for retirado da regra, os exemplos que eram anteriormente cobertos pela regra não serão mais cobertos. Isso pode ser comprovado pela Equação 3.4:

$$-3 \leq \sum x_i wi - \theta \leq 5 \tag{3.4}$$

Substituindo todos os valores possíveis para os atributos de entrada que não fazem parte da regra (A_3, A_4 e A_5) e seus respectivos pesos na Equação 3.5:

$$\sum_i w_i x_i - \theta = y \tag{3.5}$$

tem-se:

$$
\begin{array}{rcrcrcrcrcrcr}
1\times 6 & + & 1\times 4 & + & 0\times 4 & + & 1\times 0 & + & 1\times(-4) & - & 9 & = & -3 \\
1\times 6 & + & 1\times 4 & + & 0\times 4 & + & 1\times 0 & + & 0\times(-4) & - & 9 & = & 1 \\
1\times 6 & + & 1\times 4 & + & 0\times 4 & + & 0\times 0 & + & 1\times(-4) & - & 9 & = & -3 \\
1\times 6 & + & 1\times 4 & + & 0\times 4 & + & 0\times 0 & + & 0\times(-4) & - & 9 & = & 1 \\
1\times 6 & + & 1\times 4 & + & 1\times 4 & + & 1\times 0 & + & 1\times(-4) & - & 9 & = & 1 \\
1\times 6 & + & 1\times 4 & + & 1\times 4 & + & 1\times 0 & + & 0\times(-4) & - & 9 & = & 5 \\
1\times 6 & + & 1\times 4 & + & 1\times 4 & + & 0\times 0 & + & 1\times(-4) & - & 9 & = & 1 \\
1\times 6 & + & 1\times 4 & + & 1\times 4 & + & 0\times 0 & + & 0\times(-4) & - & 9 & = & 5
\end{array}
$$

Por meio dessas substituições pode-se verificar melhor que sem o literal $\neg A_5$ na regra, a rede não prediz $Y =$ `verdadeiro` para os exemplos cobertos anteriormente.

Extrair regras de uma RNA muito simples, como a da Figura 3.2 na página 52, geralmente não é um processo complicado. Mas como é o processo de extrair regras de redes que possuem uma função de transferência contínua, várias camadas, e várias unidades de saída? Sempre que uma rede é utilizada para problemas de classificação, existe um procedimento de decisão implícito que é utilizado pela rede para decidir a qual classe pertence um determinado exemplo. No exemplo da Figura 3.2, o procedimento de decisão consiste simplesmente em predizer $Y =$ `verdadeiro` quando a ativação da unidade de saída é 1, e predizer $Y =$ `falso` quando é 0. Se a função de ativação logística for utilizada ao invés da função *threshold* na unidade de saída, então o procedimento de decisão deve predizer $Y =$ `verdadeiro` quando a ativação exceder um valor especificado, por exemplo 0.5. Se for utilizado uma unidade de saída para cada classe, isso para problemas que envolvem várias classes, então o procedimento de decisão deve predizer a classe associada com a unidade de saída que tem a maior ativação. Em geral, uma regra extraída descreve um conjunto de condições sob a rede, que com seu procedimento de decisão, prediz uma determinada classe.

3.8 Critérios de Classificação e Avaliação

Devido às diferentes estratégias que têm sido propostas e desenvolvidas para extração de conhecimento de RNAs, Andrews, Diederich & Tickle (1995) sugeriram uma taxonomia para classificar e avaliar os métodos de extração de conhecimento de RNAs. Essa taxonomia é mais adequada aos métodos de extração de regras de RNAs, mas também pode ser adaptada e aplicada para métodos que não representam o conhecimento extraído na forma de regras de decisão. Essa taxonomia classifica e avalia os métodos de acordo com os seguintes itens:

1. *Poder de expressão* (ou, alternativamente, *formato*) *das regras extraídas.* Os autores sugerem três agrupamentos de formatos de regras:

   - regras simbólicas (lógica proposicional e lógica de primeira ordem), e;
   - regras baseadas em lógica *fuzzy* e conjuntos *fuzzy* (Zadeh 1965).

2. *Qualidade das regras extraídas.* Esse critério analisa as regras quanto a:

   - precisão: porcentagem de exemplos não vistos anteriormente que foram corretamente classificados;
   - fidelidade: quantos dos exemplos não vistos anteriormente, classificados pela rede neural, têm a mesma classificação ao serem submetidos ao conjunto de regras extraído da rede;
   - compreensibilidade: analisar o conjunto de regras em relação à quantidade de regras e ao número de antecedentes das regras.

3. *Granularidade* dos métodos de extração de regras. Esse critério é uma extensão do esquema de classificação utilizado por Towell & Shavlik (1993). O processo de extração de regras de uma RNA pode ocorrer de duas maneiras: utilizando as funções aprendidas pelas unidades da rede; ou utilizando a classificação realizada pela rede para os dados de entrada. Em decorrência disso, as abordagens descritas a seguir foram propostas para classificar os métodos de extração de regras.

   - global: os métodos que seguem essa abordagem, também denominada didática ou funcional, vêem a rede treinada como uma caixa preta. A extração de regras é uma tarefa de aprendizado cujo objetivo é aprender a função computada pela rede. Portanto, os métodos que seguem essa abordagem procuram extrair regras que mapeiam a entrada diretamente na saída, sem se preocupar com os passos intermediários. As regras são extraídas da função implementada pela rede.
   - local: essa abordagem, também denominada decomposicional ou estrutural, extrai as regras a partir das unidades da rede treinada. Assim, a estrutura da rede é a principal fonte para a geração das regras.
   - eclética: essa abordagem, também denominada combinacional, é uma combinação das abordagens anteriores e inclui métodos que utilizam conhecimento sobre a arquitetura interna e/ou vetores de pesos para complementar um algoritmo simbólico que procura simular o funcionamento da rede.

- composicional: nessa abordagem, o processo de extração de regras é realizado por meio de agrupamentos das unidades, ao invés de unidades individuais. Essa abordagem foi sugerida em (Tickle, Andrews, Golea & Diederich 1998) como extensão da taxonomia.

4. *Complexidade algorítmica* do método de extração de regras. Esse critério refere-se à demanda computacional associada à utilização do algoritmo.

5. *Portabilidade* do método de extração de regras em relação às várias arquiteturas de RNAs. Nesse caso, um determinado método pode ser classificado como:

   - geral: quando o algoritmo pode ser aplicado a um grande número de modelos de redes diferentes;
   - específico: para o algoritmo que deve ser aplicado a apenas um ou poucos modelos de redes.

Além dos critérios descritos anteriormente, Craven (1996) sugeriu ainda outro critério: *Escalabilidade*. Esse critério avalia os métodos de extração de regras quanto à sua aplicabilidade em redes cujos espaços de entrada, número de unidades e pesos, são muito grandes.

Craven & Shavlik (1999) argumentam que muitos pesquisadores não dão a devida atenção aos critérios de *Portabilidade* e *Escalabilidade*. Eles também argumentam que o impacto e sucesso dos métodos de extração de regras dependerá de quão bem esses dois critérios serão tratados em um futuro próximo.

Adicionalmente, uma outra questão importante que deve ser considerada pelos pesquisadores é a disponibilidade do software. Disponibilidade do software refere-se à disponibilidade do método desenvolvido para usuários potenciais. Para tornar um método disponível é necessário que seu código seja portável para que usuários possam fazer *download*, instalar e executar o método em suas próprias máquinas. Os programas C4.5 (Quinlan 1988) e CN2 (Clark & Niblett 1989) são bons exemplos disso. Esses sistemas têm sido muito difundidos na comunidade de AM, em parte porque o código é robusto e portável, estando disponível para aqueles que o querem utilizar. Uma vantagem em se disponibilizar o código portável é que usuários podem facilmente investigar variantes do algoritmo e desenvolver novas funcionalidades.

3.9 Trabalhos Relacionados

Alguns métodos de extração de conhecimento de RNAs foram propostos e desenvolvidos com o objetivo de tornar a hipótese induzida por uma RNA mais compreensível para os seres humanos. Esses métodos, como descrito na Seção 3.8 na página 54, são classificados em globais, locais, ecléticos e composicionais.

A seguir são apresentados de maneira suscinta alguns métodos globais, locais, ecléticos e composicionais de extração de conhecimento de RNAs.

3.9.1 Métodos Globais

Os métodos classificados como globais utilizam a rede como uma *caixa preta*, ou seja, eles mapeiam as entradas diretamente na saída da rede, sem se preocupar com a estrutura interna da rede.

Muitos métodos de extração de regras de RNAs utilizam busca para extrair regras. A busca consiste em procurar por regras candidatas, explorando o espaço de regras, e testar na rede cada uma dessas regras individualmente para verificar sua validade.

A Figura 3.3 mostra o espaço de busca de regras para um problema com 3 atributos booleanos A_1, A_2 e A_3. Cada nó do grafo corresponde a antecedentes de uma possível regra, e as arestas entre os nós indicam relacionamentos de especialização (de cima para baixo). O nó do topo do grafo representa a regra mais geral, e os nós que estão no último nível do grafo, nós folhas, representam as regras mais específicas.

Um dos problemas relacionados à extração de regras baseada em busca é o tamanho do espaço de regras, o qual é geralmente muito grande. Para um problema que possui n atributos binários, há 3^n regras possíveis. Isso porque cada atributo pode estar ausente no antecedente da regra, ou pode ocorrer como um literal negativo ou positivo no antecedente. Várias heurísticas têm sido propostas para limitar o tamanho do espaço de busca no processo de extração de regras.

Um dos primeiros métodos de extração de regras foi desenvolvido por Saito & Nakano (1988). Esse método utiliza busca em largura[5] para extrair regras em domínios de problemas binários. Para solucionar o problema do tamanho do espaço de busca, esse método utiliza algumas heurísticas. Uma das heurísticas consiste em limitar o número de conjunções no antecedente das regras extraídas. Para verificar se uma regra candidata é

[5] *Breadth-first.*

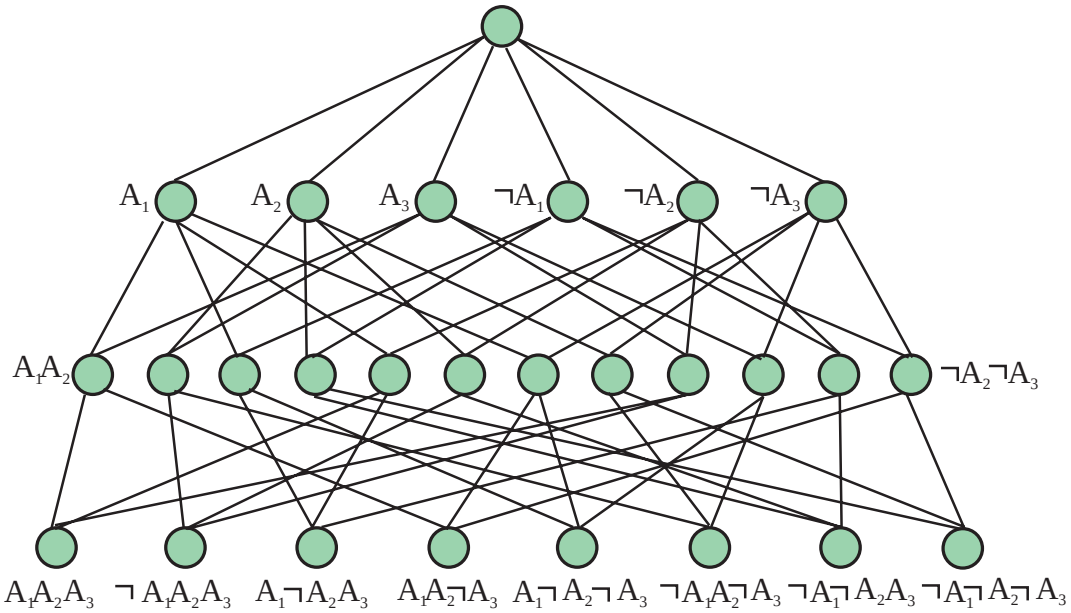

Figura 3.3: Espaço de busca de regra.

uma regra válida, esse método testa na RNA a regra candidata. Devido a essas restrições, algumas vezes esse método aceita regras que são muito gerais.

Gallant (1993) desenvolveu um método para extrair regras que, como o método de Saito e Nakano, limita a profundidade da busca por regras. A principal diferença entre os métodos é o procedimento utilizado para testar as regras candidatas na RNA. O procedimento de testar regras candidatas no método proposto por Gallant garante que somente regras válidas serão aceitas.

Thrun (1995) desenvolveu um método denominado *Validity Interval Analysis* (VI-Analysis), que é uma versão mais poderosa e generalizada do método desenvolvido por Gallant. Esse método testa as regras propagando intervalos de ativação denominados *intervalos de validade* na RNA. Um intervalo de validade de uma unidade da rede especifica o intervalo máximo para seu valor de ativação. Inicialmente, o usuário deve associar intervalos arbitrários para todas as unidades. Posteriormente, esses intervalos são refinados pelo método VI-Analysis, que exclui valores de ativação que são provavelmente inconsistentes com os pesos e *thresholds* da RNA.

O método EN, desenvolvido por Pau & Götzche (1992), fornece recursos para responder perguntas feitas a uma RNA treinada do tipo: **Por que?**: que relaciona as saídas da RNA com suas entradas; e, **Como?**: que relaciona as entradas da RNA com suas saídas. Em (Batista, Milaré & Monard 1997; Milaré, Batista & Monard 1997; Monard, Milaré &

Batista 1998) são propostas algumas variações para o método EN.

O método de extração de regras desenvolvido por Tchoumatchenko & Ganascia (1994) extrai regras de voto majoritário de uma RNA treinada. Uma regra de voto majoritário é uma lista de literais que podem ser considerados como evidência a favor ou contra uma determinada classe. Para utilizar esse método, a RNA deve ser treinada com um algoritmo especial.

O método ANN-DT (Artificial Neural-Network Decision Tree), proposto por Schmitz, Aldrich & Gouws (1999), extrai uma árvore de decisão de uma RNA. Esse método utiliza a RNA para rotular exemplos artificiais criados pela interpolação dos exemplos usados para treinar a rede.

O método proposto por Krishnan, Sivakumar & Bhattacharya (1999) extrai árvores de decisão de RNAs. Um algoritmo genético é utilizado para gerar exemplos artificiais, os quais são classificados pela RNA treinada. Os exemplos criados são filtrados por um mecanismo de seleção que retém exemplos que satisfazem a distribuição do conjunto de exemplos utilizado no treinamento da RNA e descarta os outros exemplos. Finalmente, um sistema de AM simbólico, como o ID3 ou o C4.5, é utilizado para induzir árvores de decisão a partir dos exemplos gerados.

Bologna (2002) propôs um método para extrair árvores de decisão de um tipo de RNA denominada *Discretized Interpretable Multi Layer Perceptrons* — DIMLPs. As DIMLPs criam bordas de decisão as quais são hiperplanos paralelos aos eixos. Com essa restrição, a árvore de decisão extraída da RNA possui uma taxa de fidelidade de 100% no conjunto de treinamento. Entretanto, essa restrição é uma limitação imposta ao *bias* de representação de hipóteses, se comparado com outras arquiteturas de RNAs.

O método TREPAN, proposto por Craven (1996), extrai árvores de decisão de RNAs treinadas. Esse método é descrito mais detalhadamente na Seção 3.10 na página 62 por ser utilizado neste trabalho.

3.9.2 Métodos Locais

Nos métodos locais de extração de regras de RNAs, a rede é decomposta em um conjunto de redes de apenas uma camada com uma única unidade. Um conjunto de regras é extraído para descrever cada unidade intermediária e de saída em relação às unidades que estão conectadas à ela. As regras extraídas para cada unidade individual são então combinadas em um conjunto de regras que descreve a rede como um todo.

Os métodos locais dependem do tipo de função de transferência utilizada nas unidades da rede. Existem alguns métodos de extração de regras para redes que utilizam a função de transferência sigmóide em suas unidades intermediárias e de saída. Esses métodos assumem que as unidades intermediárias e de saída podem ser aproximadas por funções *thresholds*, e cada unidade pode ser descrita por uma variável binária que indica se a unidade está ativada (ativação ≈ 1) ou desativada (ativação ≈ 0). Feita essa suposição, pode-se extrair um conjunto de regras.

Como a abordagem local assume que as unidades intermediárias e de saída da rede podem ser aproximadas por uma função *threshold*, ela é somente aplicável se as unidades intermediárias utilizam ou a função de transferência logística ou a função de transferência tangente-hiperbólica, mostradas na Figura 3.4.

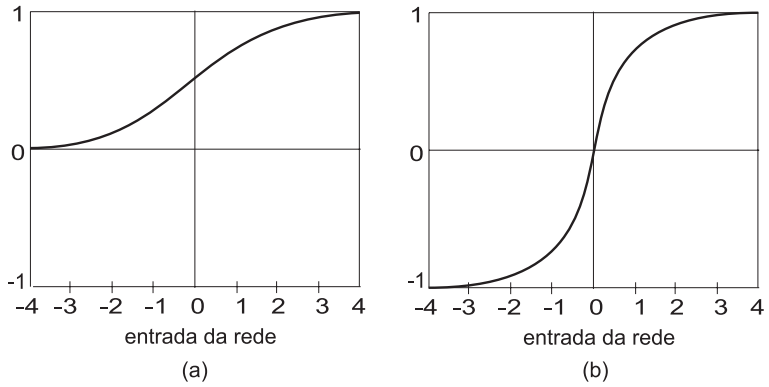

Figura 3.4: Funções de transferência sigmóides. (a) Função logística. (b) Função tangente-hiperbólica.

O método KT (Fu 1991; Fu 1994) foi desenvolvido para ser aplicado em RNAs que utilizam função de ativação sigmóide. Esse método consiste em buscar por combinações de pesos em cada unidade intermediária e de saída que, quando satisfeitas, garantem que uma determinada unidade é ativada, indiferente do estado de outras entradas dessa unidade. Apesar da busca por combinações de pesos ser limitada por algumas heurísticas, ela ainda é exponencial em relação ao número de pesos.

O método M-of-N, desenvolvido por Towell & Shavlik (1993), extrai regras de RNAs baseadas em conhecimento, denominadas *Knowledge-Based Neural Networks — KBANN* (Towell, Shavlik & Craven 1991). Nesse tipo de RNA, os pesos iniciais são especificados por um conjunto de regras proposicionais.

Hayashi (1991) desenvolveu um método de extração de regras *fuzzy* que é semelhante ao método *KT* desenvolvido em (Fu 1991; Fu 1994). A principal diferença é que os literais

das regras podem representar condições *fuzzy* (Zadeh 1965). Uma condição *fuzzy* possui diferentes graus de satisfação.

Blasig (1993) desenvolveu um método que utiliza uma função especial de treinamento, na qual as unidades intermediárias e de saída tendem a ficar em um estado que seja favorável à extração de regras. Essa função faz com que os pesos sejam atraídos em direção aos valores discretos do conjunto $\{-6, 0, 6\}$, e os *thresholds* em direção a valores ímpares múltiplos de 3. Quando os parâmetros da rede (pesos e *threshold*) estão nesse estado, depois do treinamento, cada unidade é diretamente traduzida em uma regra.

O método desenvolvido por Setiono & Liu (1995) modifica o processo de treinamento da rede para simplificar o processo de extração de regras. O método tenta minimizar o número de pesos da rede podando a rede iterativamente e utilizando uma função de treinamento que faz com que os pesos decaiam em direção ao valor 0. Em seguida, as unidades intermediárias são discretizadas e a rede é treinada novamente para compensar essa discretização. Os estados discretos para cada unidade intermediária são determinados agrupando os valores de ativação que ocorrem para cada unidade quando a rede é utilizada para classificar os exemplos de treinamento. Finalmente, um método local é utilizado para extrair regras das unidades intermediárias e de saída. Esse método extrai conjuntos concisos de regras de redes que têm um número pequeno de unidades intermediárias.

Tan (1994) desenvolveu um método de extração de regras *fuzzy* para redes de arquitetura denominada Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds & Rosen 1992). O processo de extração de regras nesse contexto consiste em traduzir diretamente partes da arquitetura da rede em regras.

Tsukimoto (2000) desenvolveu um método cuja idéia básica é aproximar as unidades da rede por funções *booleanas*. Esse método somente pode ser aplicado para RNAs que possuam função de ativação sigmoidal na camada de saída.

3.9.3 Métodos Ecléticos

Métodos ecléticos utilizam a combinação das duas estratégias descritas anteriormente para extrair conhecimento de uma RNA. Eles se caracterizam por extrair conhecimento de uma RNA utilizando as unidades de processamento individuais, estratégia local, e também utilizam a rede como caixa preta, estratégia global.

O método proposto por Tickle, Orlowski & Diederich (1994), denominado DEDEC (*Decision Detection by Rule Extration from Neural Networks*), é utilizado para a extração

de regras de redes *perceptrons* multi-camadas treinadas com o algoritmo *backpropagation*. Para extrair um conjunto de regras, o método DEDEC utiliza tanto a RNA treinada quanto informações extraídas de seus pesos.

3.9.4 Métodos Composicionais

Os métodos composicionais extraem conhecimento de uma RNA utilizando conjuntos de *clusters* de unidades de processamento ao invés de unidades de processamento individuais da rede.

O método desenvolvido por Schellharmmer, Diederich, Towsey & Brugman (1997) extrai regras gramaticais de RNAs recorrentes, especificamente de um tipo particular de redes recorrentes, denominada rede *Elman* (Elman 1990), muito utilizada para resolução de problemas de processamento de linguagem natural. Esse tipo de rede possui conexões recorrentes, das unidades da camada intermediária para uma camada denominada camada de contexto ou camada de estado, que faz parte da entrada da rede. As ativações das unidades intermediárias na iteração t são copiadas para as unidades de contexto e apresentadas com as unidades de entrada na iteração $t + 1$. As unidades intermediárias têm como objetivo mapear as entradas externas e também o estado anterior interno para produzir a saída desejada. O conhecimento da rede é representado nas ativações das unidades intermediárias e é explicitado por meio de análise de *cluster* ou de outros métodos estatísticos. A partir da análise de *clusters* são extraídos autômatos de estado finito e esses autômatos representam a gramática aprendida pela rede.

Hruschka (2001) desenvolveu um método para extrair regras de RNAs baseado em AGs de agrupamento. O método consiste em dois passos: aplicação de um algoritmo de agrupamento para encontrar *clusters* nos valores de ativação das unidades intermediárias para cada classe, e; geração de um conjunto de regras que descrevem os valores discretizados das unidades intermediárias em relação às entradas.

3.10 O Método TREPAN

O método TREPAN (TREes PArroting Networks) (Craven 1996) extrai uma árvore de decisão de uma RNA treinada. Devido à sua característica de generalidade, esse método pode ser aplicado para uma grande variedade de RNAs supervisionadas, sem a necessidade da RNA ser de uma arquitetura específica ou do algoritmo com que ela foi treinada ser de

um determinado tipo.

O objetivo do método TREPAN é extrair uma hipótese compreensível de uma RNA treinada. Essa hipótese é representada por meio de uma árvore de decisão que tenta representar os conceitos aprendidos pela rede. TREPAN possui algumas semelhanças, e também algumas diferenças, com os algoritmos convencionais de indução de árvore de decisão como, por exemplo, o algoritmo C4.5 (Quinlan 1988).

O método TREPAN constrói uma árvore de decisão particionado recursivamente um conjunto de exemplos. Ele mantém uma fila de nós que são expandidos em subárvores à medida que esses nós vão sendo removidos da fila. Junto à cada nó dessa fila são mantidos os seguintes três conjuntos:

Conjunto de Exemplos

Os exemplos desse conjunto são os exemplos utilizados no treinamento da RNA. Para cada nó da fila de nós há um conjunto dos exemplos que alcançam o nó, ou seja, há um conjunto de exemplos que estão disponíveis para cada nó;

Conjunto de Exemplos Artificiais

O método TREPAN pode criar exemplos artificiais. Os exemplos artificiais são utilizados, juntamente com os exemplos de treinamento, para selecionar o teste de divisão do nó, se esse for um nó interno da árvore, ou para determinar a classe de um nó, se esse for um nó folha;

Conjunto de Restrições

O conjunto de restrições descreve as condições que os exemplos devem satisfazer para alcançar um nó.

Para expandir um nó da árvore, o método TREPAN seleciona um teste de divisão para aquele nó e cria um nó filho para cada valor do teste. Cada nó filho se torna ou um nó folha da árvore de decisão ou será colocado dentro da fila de nós para futuramente ser expandido. A seguir são descritas as principais características do algoritmo TREPAN.

Exemplos e Oráculo

Durante o processo de indução da árvore de decisão, o algoritmo TREPAN utiliza a RNA como uma caixa preta para classificar os exemplos que são usados na indução da árvore. Nesse processo, a RNA é denominada de *oráculo*.

Os exemplos, a partir dos quais a árvore de decisão é induzida, são os exemplos que foram utilizados no treinamento da RNA. Como mencionado, o método TREPAN

pode criar também, se necessário, exemplos artificiais. Esses exemplos artificiais, depois de serem classificados pela RNA, são utilizados na indução da árvore de decisão. Quando um exemplo é fornecido à RNA para ser classificado, é realizada uma consulta à RNA. Portanto, as consultas são utilizadas para dois propósitos: classificar os exemplos de treinamento com que a RNA foi treinada e classificar exemplos artificiais criados pelo método TREPAN.

As classes retornadas pela RNA não são necessariamente as classes originais a qual os exemplos de treinamento ou artificiais pertencem. A classificação desses exemplos não é dada mais pela função conceito f, mas sim pela hipótese $\mathbf{h}$ aprendida pela RNA.

Para classificar um nó folha da árvore de decisão ou para escolher um teste de divisão para um nó interno, o método TREPAN utiliza um determinado número mínimo de exemplos. À medida que a profundidade da árvore aumenta, os exemplos são particionados entre os nós. O número de exemplos associado à cada nó tende a diminuir e, para minimizar esse problema, o método pode criar exemplos novos. Como ilustração, suponha que o método TREPAN utilize no mínimo $min\_exemplos$ para classificar um nó folha ou escolher um teste de divisão em que estão disponíveis $\hat{M}$ exemplos em um determinado nó, sendo $\hat{M} < min\_exemplos$. O método TREPAN cria $min\_exemplos - \hat{M}$ exemplos antes de tomar qualquer decisão sobre aquele nó.

Para criar um novo exemplo, é utilizado o conjunto de restrições associado ao nó corrente da árvore de decisão. Essas restrições devem ser satisfeitas pelo novo exemplo. Por exemplo, suponha que o método TREPAN tenha que criar um novo exemplo que alcance o nó hachurado da árvore de decisão da Figura 3.5 na página oposta. Esse exemplo deve satisfazer as seguintes restrições:

$$A_1 = \texttt{verdadeiro} \text{ e } (A_2 = \texttt{verdadeiro} \text{ ou } A_3 = \texttt{verdadeiro}).$$

A primeira restrição refere-se ao teste do nó raiz da árvore de decisão e a segunda restrição refere-se ao teste *m-de-n* do nó filho esquerdo do nó raiz. Esse tipo de teste será descrito posteriormente.

Para converter um conjunto de restrições em exemplos artificiais, o método TREPAN cria os exemplos considerando a atual distribuição do conjunto de dados. TREPAN utiliza distribuições empíricas[6] para gerar valores para os atributos qualitativos e

[6] *Empirical distributions.*

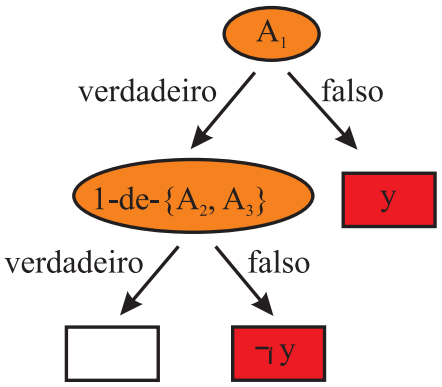

Figura 3.5: Árvore de decisão sendo expandida.

estimativa da densidade de núcleo[7] (Silverman 1986) para gerar valores para os atributos quantitativos.

O método TREPAN compara as distribuições de cada atributo separadamente durante o processo de modelamento dos atributos qualitativos e quantitativos na geração de um novo exemplo. Isso faz com que as dependências entre os atributos não sejam consideradas, o que representa uma limitação do método.

Expansão da Árvore de Decisão

O método TREPAN utiliza a expansão primeiro o melhor[8] para construir a árvore de decisão, ao contrário da maioria dos algoritmos de indução de árvore de decisão, que utiliza a expansão em profundidade[9]. TREPAN tenta maximizar a fidelidade entre a árvore e a RNA toda vez que adiciona um nó na árvore de decisão. O melhor nó é o que possui o maior potencial para aumentar a fidelidade entre a árvore de decisão e a RNA. A função utilizada para avaliar um nó z é definida pela Equação 3.6:

$$f(z) = reach(z) * (1 - fidelity(z)) \tag{3.6}$$

na qual $reach(z)$ é a fração estimada dos exemplos do conjunto de dados e dos exemplos artificiais que estão disponíveis em z; e $fidelity(z)$ é a fidelidade estimada da árvore em relação à rede para esses exemplos. O valor de $reach(z)$ é calculado como o produto das freqüência dos exemplos dos ramos entre o nó raiz da árvore e o nó z. O valor de $fidelity(z)$, que é a fração de exemplos que a árvore e a rede

[7] *Kernel density estimates.*
[8] *Best-first.*
[9] *Depth-first.*

concordam em suas predições, é calculado utilizando os exemplos do conjunto de exemplos e também os exemplos artificiais que estão disponíveis no nó z.

Testes de Divisão

Escolher um teste de divisão para um nó interno de uma árvore de decisão significa escolher como o espaço de exemplos será particionado nesse nó. Diferentemente de alguns algoritmos de indução de árvore de decisão, que utilizam nos nós internos da árvore testes de divisão com apenas um atributo, o método TREPAN utiliza expressões *m-de-n* em seus testes. A expressão *m-de-n* foi introduzida como critério de divisão nos nós internos de uma árvore de decisão por Murphy & Pazzani (1991). Uma expressão *m-de-n* é uma expressão *booleana* que possui um *threshold* inteiro, m, e um conjunto de n literais *booleanos*. Uma expressão *m-de-n* é satisfeita quando ao menos m dos n literais são satisfeitos. Por exemplo, suponha três atributos A_1, A_2 e A_3. A expressão $2 - de - \{A_1, \neg A_2, A_3\}$ é logicamente equivalente a $(A_1 \wedge \neg A_2) \vee (A_1 \wedge A_3) \vee (\neg A_2 \wedge A_3)$.

Para construir um teste *m-de-n*, o método TREPAN utiliza uma busca heurística. Inicialmente é selecionado o melhor teste no nó corrente. Essa seleção é realizada utilizando o critério *ganho de informação*[10] (Quinlan 1988). Para atributos qualitativos, um teste é utilizado para separar os exemplos em relação aos valores do atributo. Para atributos quantitativos, os testes utilizam valores de corte, por exemplo, $A_1 < 0.75$, $A_1 \geq 0.75$.

Critério de Parada

O método TREPAN utiliza dois critérios de parada: local e global. Um critério de parada local baseia-se somente nas características do nó corrente e dos exemplos do conjunto de dados que estão disponíveis nesse nó, para decidir se o nó será expandido em uma sub-árvore ou se tornará um nó folha. Um critério de parada global considera o estado de toda a árvore de decisão, não somente o estado do nó que está sendo expandido.

Um dado nó torna-se um nó folha da árvore de decisão se, com alta probabilidade, ele cobre somente exemplos de uma classe. Para determinar a "pureza" de um nó, o algoritmo TREPAN estima a proporção de exemplos, $prop(C_i)$, cobertos pelo nó que são da classe mais comum, C_i, do nó.

[10] *Information gain.*

No critério de parada global, o algoritmo TREPAN utiliza um parâmetro que limita o número máximo de nós internos que a árvore pode ter. O usuário pode atribuir o valor que desejar a esse parâmetro.

Em adição ao parâmetro que limita o tamanho da árvore, o método TREPAN também pode utilizar um conjunto de validação para decidir sobre a árvore que será retornada. Como a árvore é gerada utilizando a expansão primeiro o melhor, TREPAN pode gerar várias árvores a partir da árvore antecessora. Essas árvores são diferentes somente nas sub-árvores que correspondem ao nó expandido no último passo. O conjunto de validação é então utilizado para medir a fidelidade de cada árvore. A árvore que possuir maior fidelidade com a rede é retornada.

Poda

Depois que o método TREPAN termina de expandir a árvore, ou seja, depois que o critério de parada é satisfeito, é realizada uma poda simples, antes de retornar a árvore final. Essa poda consiste em detectar sub-árvores que predizem a mesma classe em todos os seus nós folhas e utilizar somente um nó folha no lugar dessa sub-árvore. A árvore é simplificada o quanto possível, mas as modificações realizadas não interferem no comportamento preditivo da árvore, já que essas modificações somente desconsideram nós internos redundantes da árvore.

3.11 Considerações Finais

Neste capítulo foram discutidas a dificuldade e a importância da compreensão da hipótese induzida por RNAs. Para que as hipóteses induzidas por RNAs possam ser mais facilmente compreensíveis para os seres humanos, foram propostos diversos métodos com o objetivo de extrair conhecimento compreensivel de RNAs. Alguns desses métodos foram descritos brevemente neste capítulo, bem como critérios para avaliá-los e classificá-los.

Muitos dos métodos apresentados possuem algumas limitações, entre elas pode-se citar a falta de portabilidade e/ou escalabilidade. Como mencionado, portabilidade refere-se à variedade de tipos de RNAs às quais um método pode ser aplicado. Alguns métodos de extração de conhecimento de RNAs são limitados porque requerem a utilização de um procedimento especial de treinamento da rede. Alguns métodos são limitados porque impõem restrições sobre a arquitetura da rede ou porque requerem que as unidades intermediárias utilizem a função de ativação sigmóide. Outra limitação bastante comum é que diversos métodos são específicos para problemas que possuem somente atributos

qualitativos.

Escalabilidade refere-se à aplicabilidade dos métodos de extração de conhecimento em redes cujos espaços de entrada, número de unidades e parâmetros livres são muito grandes. O tamanho do espaço de regras cresce exponencialmente com o número de atributos de entrada. Alguns métodos exploram eficientemente grandes espaços de regras, outros métodos não possuem essa capacidade. Nos métodos locais, o tamanho dos conjuntos de regras extraídas de cada unidade tende a aumentar com o tamanho da rede, fazendo com que o conjunto final de regras seja tão ou mais complexo do que a própria rede (Craven 1996).

No próximo capítulo é descrito o projeto e a implementação do ambiente proposto e desenvolvido neste trabalho. Esse ambiente implementa dois métodos para extração de conhecimento simbólico de RNAs. Os métodos propostos podem ser aplicados para qualquer tipo de RNA supervisionada, sem nenhuma restrição.

Capítulo 4

O Ambiente *NNRules*

4.1 Considerações Iniciais

Neste capítulo é apresentada uma descrição do ambiente computacional *NNRules* para extração de conhecimento simbólico de RNAs. O ambiente *NNRules* utiliza sistemas de AM simbólico e Algoritmos Genéticos para extrair conhecimento simbólico na forma de regras de uma RNA treinada.

O objetivo principal do ambiente *NNRules* é obter conhecimento simbólico com alta fidelidade com a RNA treinada, isto é, obter conhecimento que classifique um conjunto de exemplos da mesma forma que a RNA treinada utilizada no processo de extração de conhecimento.

Este capítulo está organizado em cinco seções. Na Seção 4.2 são descritos os principais objetivos do desenvolvimento do ambiente *NNRules*. O ambiente *NNRules* é constituído de três módulos principais. O módulo RuleBase gerencia uma base de regras que integra regras induzidas por diversos sistemas de AM simbólico. O módulo RuleSet cria conjuntos de regras a partir das regras armazenadas por uma instância do módulo RuleBase. O módulo GA implementa um AG que faz uso dos módulos RuleSet e RuleBase para encontrar um conjunto de regras que apresente alta fidelidade com a RNA treinada da qual as regras foram extraídas, conforme o método de extração de conhecimento baseado em AG apresentado no Capítulo 1. Na Seção 4.3 são apresentados a arquitetura e o projeto de cada um dos módulos do ambiente *NNRules*. Na Seção 4.4 é descrito o funcionamento do ambiente *NNRules*. Finalmente, na Seção 4.5 são feitas algumas considerações finais sobre o ambiente proposto.

4.2 Objetivos

O principal objetivo do desenvolvimento do ambiente $NNRules$ é criar um ambiente que permita avaliar experimentalmente o uso de sistemas de AM simbólico e AGs para a extração de conhecimento simbólico de RNAs.

O projeto do ambiente $NNRules$ foi criado tendo-se em vista o desenvolvimento de um ambiente flexível que pudesse ser facilmente modificado para avaliar e testar novas idéias. Por esse motivo, decidiu-se dividir o ambiente em módulos, os quais podem ser compostos da forma mais adequada para a realização de um determinado experimento.

Os módulos, os quais são implementados por meio de classes, podem ser vistos como bibliotecas. Foi implementado, também, um conjunto de *scripts*, os quais utilizam os métodos implementados pelos módulos para automatizar uma determinada tarefa. Os *scripts* utilizados nos experimentos realizam tarefas que vão desde o pré-processamento dos dados até a execução dos métodos de extração de conhecimento de RNAs.

O ambiente $NNRules$ faz parte do projeto DISCOVER. O projeto DISCOVER é um projeto que envolve diversos pesquisadores do nosso laboratório — LABIC. Um dos principais objetivos do projeto DISCOVER é diminuir o esforço necessário para implementar novas idéias, bem como para realizar análises experimentais relacionadas à aquisição automática de conhecimento. Muitas vezes, um procedimento semelhante é implementado diversas vezes, por membros diferentes da equipe, por falta de comunicação entre os membros ou por falta de documentação dos procedimentos já implementados. Dessa forma, o projeto DISCOVER visa aumentar a comunicação e a padronização das implementações realizadas em nosso grupo de pesquisa.

O ambiente $NNRules$ foi implementado em Perl[1]. A linguagem Perl também é a linguagem utilizada na implementação do ambiente DISCOVER. Essa linguagem possui como uma das suas principais qualidades a possibilidade de implementar rapidamente protótipos para avaliação. Neste trabalho foram utilizadas diversas bibliotecas que fazem parte do ambiente DISCOVER tais como a biblioteca DOL (Batista & Monard 2003) para pré-processamento de dados, o ambiente SNIFFER (Batista 2003) para gerenciamento de experimentos e as bibliotecas para conversão de classificadores simbólicos para o formato padrão $\mathcal{P}BM$ (Prati, Baranauskas & Monard 2001b) do DISCOVER.

[1] *Practical Extraction and Report Language.*

4.3 Os Módulos do Ambiente $NNRules$

Como mencionado anteriormente, o ambiente computacional $NNRules$ é um ambiente para extração de conhecimento simbólico de RNAs. Esse ambiente utiliza sistemas de AM simbólico e AGs para extrair conhecimento simbólico na forma de regras de RNAs treinadas.

O ambiente $NNRules$ está dividido em três módulos principais:

RuleBase
: O módulo RuleBase cria uma base de regras e permite que essa base de regras seja manipulada. Como descrito mais adiante, as regras que compõem uma base podem ser extraídas de classificadores induzidos por diversos sistemas de AM simbólico. O módulo RuleBase permite carregar regras em uma estrutura comum e realizar uma série de manipulações sobre essas regras, tais como extrair informações sobre a complexidade sintática das regras e verificar se uma determinada regra cobre um exemplo.

RuleSet
: O módulo RuleSet cria um classificador a partir das regras armazenadas em uma instância do módulo RuleBase. Um conjunto de regras armazenado em uma instância do módulo RuleSet pode ser qualquer subconjunto de regras armazenadas em uma instância do módulo RuleBase. O módulo RuleSet não armazena as regras diretamente, mas, ao invés disso, armazena referências às regras armazenadas e gerenciadas em uma instância do módulo RuleBase. O módulo RuleSet implementa diversas manipulações sobre um conjunto de regras como, por exemplo, adicionar e remover regras do conjunto de regras gerenciado pelo módulo, gravar um conjunto de regras em um arquivo na sintaxe padrão $\mathcal{PBM}$ e classificar um exemplo utilizando o conjunto de regras gerenciado pelo módulo.

GA
: O módulo GA utiliza os módulos RuleBase e RuleSet para implementar e simular um Algoritmo Genético. O objetivo do módulo GA é encontrar um conjunto de regras que possua alta fidelidade de classificação com a RNA que rotulou os dados. Para isso, o módulo GA utiliza regras induzidas por diferentes sistemas de AM simbólico. Cada indivíduo manipulado pelo AG é uma instância do módulo RuleSet. O módulo GA utiliza os operadores genéticos para modificar os indivíduos, ou seja, os conjuntos de regras. Cada conjunto de regras é uma instância do módulo RuleSet.

A seguir, são explicados com mais detalhes cada um dos módulos que constituem o ambiente $NNRules$. É realizada também uma descrição do projeto e da arquitetura desses módulos.

4.3.1 O Módulo RuleBase

O módulo RuleBase armazena e gerencia uma base de regras. As regras armazenadas em uma mesma instância do módulo RuleBase devem pertencer a um domínio comum, isto é, devem fazer referência a um conjunto comum de atributos.

As regras armazenadas em uma instância do módulo RuleBase podem ser resultado da indução realizada por diferentes sistemas de AM simbólico. Infelizmente, os sistemas de AM simbólico utilizam arquivos texto com sintaxes proprietárias para armazenar os classificadores induzidos. Para solucionar esse problema, o módulo RuleBase utiliza uma classe chamada `RuleStream` para converter classificadores simbólicos em diferentes sintaxes para uma sintaxe de regras padrão chamada $\mathcal{PBM}$ (Prati, Baranauskas & Monard 2001b), e também calcular informações padrão para a avaliação de regras (Prati, Baranauskas & Monard 2001a) para cada uma das regras do conjunto de regras que constitui o classificador.

Na Figura 4.1 na próxima página é mostrado o conteúdo de um arquivo com regras na sintaxe $\mathcal{PBM}$ e as informações padrão calculadas para cada uma dessas regras.

Na sintaxe $\mathcal{PBM}$ cada regra é escrita no seguinte formato:

$$\text{R<número> } \mathbf{IF} \text{ <condições> } \mathbf{THEN\ CLASS} = C_i \ [f_{bh},\ f_{b\bar{h}},\ f_{\bar{b}h},\ f_{\bar{b}\bar{h}},\ n]$$

no qual <número> é um valor referencial para cada regra, <condições> é uma conjunção de condições que envolvem os atributos do conjunto de dados, e C_i é a predição realizada pela regra. Os números entre colchetes são as informações padronizadas, também denominadas de *matriz de contingência*, calculadas para cada regra.

A matriz de contingência é uma generalização da matriz de confusão. Uma matriz de confusão é aplicada ao classificador como um todo, ou seja, o classificador é tratado como uma caixa preta. Já uma matriz de contingência é calculada para cada regra do tipo $B \leftarrow H$, exigindo dessa forma, um classificador simbólico.

```
Standard Rules Conversor v1.2.1          Copyright (c)Ronaldo C. Prati
Inducer: cn2                             Input File: voyage.cn2.output
Date: Tue Jan  7 02:52:54 2003

Rules Evaluated as UNORDERED
Names File: voyage.names   Data File: voyage.data

R0001   IF humity < 83.00
           AND windy = no
        THEN CLASS = go [0.3333, 0, 0.2667, 0.4, 15]

R0002   IF outlook = overcast
           AND temperature > 19.50
        THEN CLASS = go  [0.2667, 0, 0.3333, 0.4, 15]

R0003   IF outlook = sunny
           AND humity < 76.00
        THEN CLASS = go [0.1333, 0, 0.4667, 0.4, 15]

R0004   IF outlook = rain
           AND humity > 87.50
        THEN CLASS = go [0.0667, 0, 0.5333, 0.4, 15]

R0005   IF outlook = sunny
           AND humity > 83.00
        THEN CLASS = dont_go [0.2, 0, 0.2, 0.6, 15]

R0006   IF temperature < 24.00
          AND humity < 80.50
          AND windy = yes
        THEN CLASS = dont_go [0.2, 0, 0.2, 0.6, 15]

R0007   DEFAULT CLASS = go
```

Figura 4.1: Arquivo de regras no formato $\mathcal{P}BM$ com informações padrão.

Utilizando a letra b para denominar o valor do antecedente, ou corpo[2] da regra, e a letra h para denominar o valor do conseqüente, ou cabeça[3] da regra, os valores da matriz de contingência, ilustrada na Tabela 4.1, representam:

	H	$\overline{H}$	
B	f_{bh}	$f_{b\bar{h}}$	f_b
$\overline{B}$	$f_{\bar{b}h}$	$f_{\bar{b}\bar{h}}$	$f_{\bar{b}}$
	f_h	$f_{\bar{h}}$	1

Tabela 4.1: Matriz de contingência.

- f_{bh} é a porcentagem de exemplos cobertos pelo corpo da regra e corretamente classificados por ela;
- $f_{b\bar{h}}$ é a porcentagem de exemplos cobertos pelo corpo da regra, mas incorretamente classificados por ela;
- $f_{\bar{b}h}$ é a porcentagem de exemplos não cobertos pelo corpo da regra, mas a classe prevista pela cabeça da regra é a mesma classe y_i do exemplo;
- $f_{\bar{b}\bar{h}}$ é a porcentagem de exemplos não cobertos pelo corpo da regra, e a classe prevista pela cabeça da regra não é a mesma classe y_i do exemplo;
- n é o número de exemplos utilizados para gerar a matriz de contingência da regra.

Lavrač, Flach & Zupan (1999) mostram que é possível definir a maioria das medidas propostas na literatura para a avaliação de regras, a partir da matriz de contingência dessas regras.

Estão implementados no sistema DISCOVER diversos conversores de regras, os quais podem converter classificadores simbólicos induzidos por vários sistemas de AM simbólico, em classificadores na sintaxe $\mathcal{PBM}$. Na Tabela 4.2 são mostrados os sistemas de aprendizado que atualmente possuem conversores para a sintaxe $\mathcal{PBM}$. A segunda coluna da tabela indica a linguagem de representação do conceito utilizada pelo indutor.

A classe `RuleBase` é a principal classe do módulo RuleBase. Essa classe cria uma base de regras a partir das regras dos classificadores simbólicos convertidos para a sintaxe $\mathcal{PBM}$. A implementação da classe `RuleBase` é simplificada pelo uso da classe `RuleStream`.

[2] *Body.*
[3] *Head.*

Indutor	Linguagem de Representação
C4.5	árvore de decisão
C4.5rules	regras de decisão
C5.0	árvore de decisão
CN2	regras de decisão
ID3	árvore de decisão
RIPPER	árvore de decisão
OC1	árvore de decisão oblíqua

Tabela 4.2: Sistemas de aprendizado cujos classificadores podem ser convertidos para a sintaxe $\mathcal{P}BM$.

A classe `RuleStream` transforma um ou mais arquivos texto com as descrições dos classificadores simbólicos em arquivos na sintaxe $\mathcal{P}BM$. A classe `RuleStream` cria também um *stream* de regras, ou seja, fornece um método que, quando chamado, retorna a próxima regra. As regras de todos os classificadores convertidos são fornecidas uma a uma, até que se esgotem os arquivos. Ou seja, a classe `RuleBase` somente precisa chamar um método da classe `RuleStream` para receber a próxima regra convertida para o formato $\mathcal{P}BM$.

A base de regras gerenciada pela classe `RuleBase` armazena, entre outras informações, as regras, suas respectivas matrizes de contingência e informações sobre os sistemas de aprendizado que induziram cada uma das regras.

Na Figura 4.2 na próxima página é apresentado um diagrama da arquitetura do módulo RuleBase. Nesse diagrama, p classificadores são carregados na base de regras. Esses p classificadores podem ter sido gerados de diversas formas. Nos experimentos descritos no Capítulo 5, os p classificadores são gerados por p sistemas de aprendizado diferentes sobre um mesmo conjunto de dados. Entretanto, formas alternativas poderiam utilizar um ou mais sistemas de aprendizado, com variações dos parâmetros desses sistemas de aprendizado, sobre o mesmo conjunto de dados. Ou, ainda, com um ou mais sistemas de aprendizado sobre diversas amostras de um mesmo conjunto de dados.

O módulo RuleBase provê diversas funcionalidades para a manipulação de regras. Entre as principais estão prover informações sobre a matriz de contingência das regras e sobre o sistema de aprendizado que induziu as regras; informar quais são as regras *default* e remover essas regras da base; prover informações sobre a complexidade sintática das regras; e, verificar se uma regra cobre um determinado exemplo, entre outros.

As regras armazenadas na base são referenciadas por meio de um índice, isto é, cada regra possui um índice único que a identifica. Os demais módulos do ambiente $NNRules$ que utilizam o módulo RuleBase armazenam somente os índices das regras armazenadas, e utilizam esses índices para realizar operações sobre as regras.

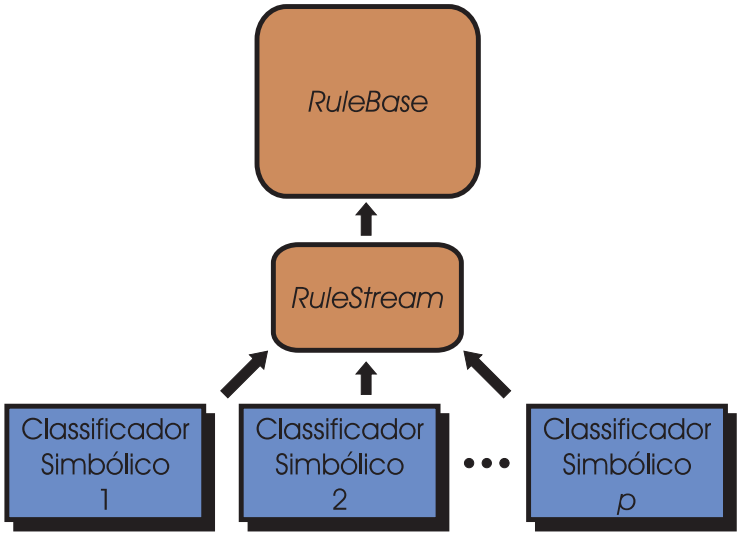

Figura 4.2: Arquitetura do módulo RuleBase.

Na Figura 4.3 é apresentado o projeto do módulo RuleBase por meio de um diagrama de classes em UML[4] (Booch, I & Rumbaugh 1998). Esse módulo é implementado pelas classes `RuleBase` e `RuleStream`. A classe `RuleStream` faz uso das classes `StdRuleLib` (Prati, Baranauskas & Monard 2001b) e `StdInfoUnordered` (Prati, Baranauskas & Monard 2001a) do projeto DISCOVER. A classe `StdRuleLib` converte um classificador simbólico em um classificador simbólico na sintaxe $\mathcal{PBM}$. A classe `StdInfoUnordered` calcula e adiciona a cada regra do classificador na sintaxe $\mathcal{PBM}$ sua matriz de contingência.

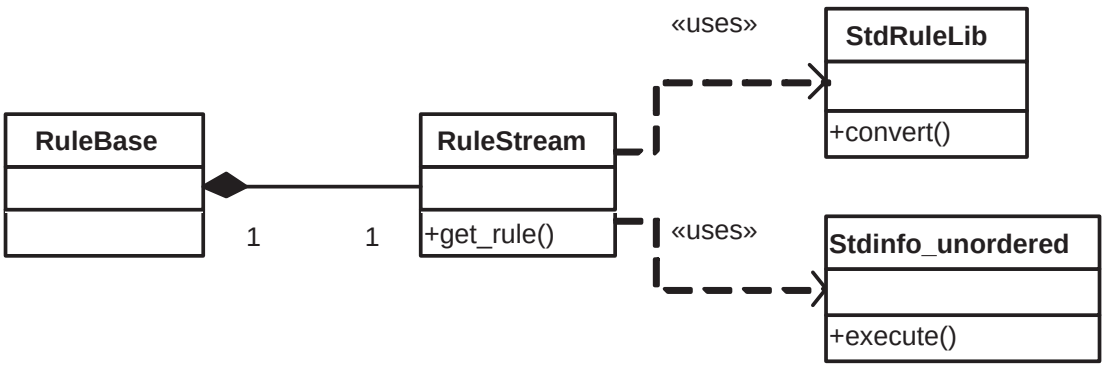

Figura 4.3: Projeto do módulo RuleBase.

[4] *The Unified Modeling Language.*

4.3.2 O Módulo RuleSet

O módulo RuleSet permite implementar classificadores com subconjuntos de regras armazenadas em uma instância do módulo RuleBase. Uma instância do módulo RuleSet não armazena as regras diretamente, mas somente uma lista com os índices das regras que pertencem ao classificador. Como mencionado anteriormente, essas regras estão armazenadas em uma instância do módulo RuleBase.

Pode-se criar diversas instâncias do módulo RuleSet para uma mesma base de regras armazenada em uma instância do módulo RuleBase. Cada classificador pode ser composto por diferentes regras, bem como por diferentes quantidades de regras.

Uma das principais características do módulo RuleSet é que ele permite classificar um exemplo segundo um conjunto de regras nele armazenado, utilizando diversas abordagens. Existem diversas abordagens para verificar como um conjunto de regras cobre um novo exemplo, como descrito na Seção 2.3 na página 25. Na abordagem *ordenada*, assume-se que as regras possuem uma ordem e o novo exemplo é classificado com a classe da primeira regra cujo corpo cobre o exemplo. Na abordagem *não ordenada* todas as regras que disparam são consideradas e o exemplo é classificado com a classe predominante dessas regras.

Nos experimentos descritos no Capítulo 5 foram avaliadas algumas abordagens para classificar um novo exemplo. Uma vez que não existe uma ordem implícita nas regras, ou seja, as regras não estão ordenadas por qualquer índice de qualidade, as abordagens analisadas neste trabalho são todas não ordenadas. Entre as abordagens que apresentaram melhores resultados estão as denominadas *SingleRule* e *MultipleRules*.

SingleRule

A abordagem *SingleRule* verifica quais regras são disparadas por um dado exemplo. Dentre as regras disparadas, é selecionada aquela com maior poder de predição. O poder de predição de uma regra pode ser estimado verificando-se sua matriz de contingência. Uma estratégia utilizada é o cálculo do *índice de precisão menos erro*, Ipe, o qual é calculado subtraindo do percentual de exemplos cobertos corretamente pela regra o percentual de exemplos cobertos incorretamente, ou seja:

$$Ipe = f_{bh} - f_{b\bar{h}} \tag{4.1}$$

Uma outra estratégia utilizada nos experimentos, mas que não apresentou resultados

tão bons quanto o Ipe, foi o *índice de cobertura*, Ic, o qual é calculado somando o percentual de exemplos cobertos corretamente pela regra com o percentual de exemplos cobertos incorretamente, ou seja:

$$Ic = f_{bh} + f_{b\bar{h}} \tag{4.2}$$

O algoritmo para classificar um exemplo segundo a abordagem *SingleRule* é descrito pelo Algoritmo 4.1.

Algoritmo 4.1 Algoritmo que classifica um exemplo segundo a abordagem *SingleRule*.

Require: $R = \{R_1, R_2, \ldots R_{NR}\}$: um classificador simbólico com NR regras;
E_q: um exemplo de consulta a ser classificado;
$I(R_i)$: um índice de avaliação de regras aplicado sobre uma regra R_i.

1: Seja $\hat{R} \leftarrow \{\hat{R}_1, \hat{R}_2, \ldots \hat{R}_{\hat{NR}}\}$, o subconjunto de regras de R que cobrem E_q
2: $\hat{R}_S \leftarrow \arg\max_{\hat{R}_i \in \hat{R}} I(R_i)$
3: **return** classificação dada por $\hat{R}_S$

MultipleRules

A abordagem *MultipleRules* utiliza todas as regras que cobrem um dado exemplo para definir a classificação desse exemplo. Para determinar a classe do exemplo, é utilizado um índice que fornece uma estimativa do poder de predição de cada regra, de forma similar à abordagem *SingleRule*. Para uma determinada classe C_j são totalizados os valores do índice de todas as regras disparadas que predizem essa classe. A classe com maior valor total do índice é a classificação do novo exemplo. No Algoritmo 4.2 na próxima página é apresentado o algoritmo para classificar um exemplo segundo a abordagem *MultipleRules*.

Na Figura 4.4 na página oposta é apresentada a arquitetura do módulo RuleSet. Uma aplicação pode possuir diversas instâncias do módulo RuleSet. Cada instância gerencia um conjunto de regras diferente, sendo que todas as regras gerenciadas estão armazenadas em uma instância do módulo RuleBase.

O projeto do módulo RuleSet é apresentado na Figura 4.5 na página 80 utilizando um diagrama de classes em UML. O módulo RuleSet é implementado por meio de uma classe abstrata com o mesmo nome do módulo. Uma classe abstrata tem como principal objetivo definir uma interface comum para as suas sub-classes. Uma classe abstrata delega a implementação de alguns ou todos os seus métodos para as suas sub-classes. De uma

Algoritmo 4.2 Algoritmo que classifica um exemplo segundo a abordagem *MultipleRules*.

Require: $R = \{R_1, R_2, \ldots R_{NR}\}$: um classificador simbólico com NR regras;
E_q: um exemplo de consulta a ser classificado;
$I(R_i)$: um índice de avaliação de regras aplicado sobre uma regra R_i;
$C = \{C_1, C_2, \ldots C_{Ncl}\}$: o conjunto de classes definidas para o domínio.

1: Seja $\hat{R} = \{\hat{R}_1, \hat{R}_2, \ldots \hat{R}_{\hat{N}R}\}$, o subconjunto de regras de R que cobrem E_q
2: **for all** $C_j \in C$ **do**
3: $I_{C_j} \leftarrow 0$ {Índice total das regras que predizem C_j}
4: **for all** $\hat{R}_i \in \hat{R}$ **do**
5: **if** $\hat{R}_i$ prediz a classe C_j **then**
6: $I_{C_j} \leftarrow I_{C_j} + I(\hat{R}_i)$
7: **end if**
8: **end for**
9: **end for**
10: **return** $\arg\max_{C_j \in C} I_{C_j}$

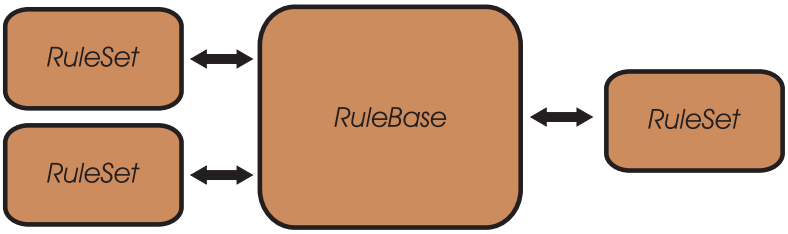

Figura 4.4: Arquitetura do módulo RuleSet.

forma geral, a classe abstrata deve implementar tudo que há de comum entre as suas sub-classes, deixando em aberto somente as partes que variam. As partes da implementação que variam devem ser implementadas nas sub-classes, conforme a necessidade de cada uma delas.

A classe abstrata `RuleSet` implementa as funcionalidades comuns a todas as sub-classes. Cada uma das suas sub-classes implementa uma abordagem diferente para classificar um exemplo dado um conjunto de regras. Entre as funcionalidades providas pela classe abstrata `RuleSet` estão métodos para adicionar e remover regras no conjunto de regras, para adicionar e remover uma regra *default*, para salvar um conjunto de regras na sintaxe $\mathcal{PBM}$, entre outras.

Foram implementadas cinco sub-classes que implementam diferentes abordagens para classificar um exemplo dado um conjunto de regras. As classes `RuleSetC45`, `RuleSetCN2` e `RuleSetC45Rules` implementam as abordagens de classificação de exemplos utilizadas nos sistemas de AM simbólico C4.5, CN2 e C4.5rules, respectivamente. Essas classes per-

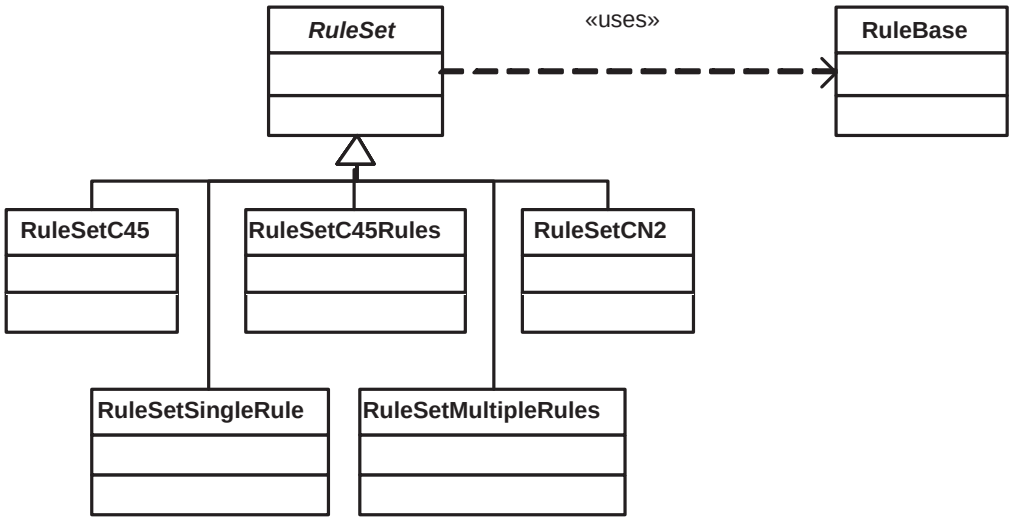

Figura 4.5: Projeto do módulo RuleSet.

mitem, dado um conjunto de regras induzido por um desses classificadores e carregado em um módulo RuleBase, imitar a abordagem de classificação desses sistemas e, dessa forma, verificar como cada um desses sistemas classifica um determinado exemplo. Essas sub-classes podem ser utilizadas, por exemplo, para calcular as taxas de infidelidade entre esses classificadores simbólicos e uma RNA.

As classes `RuleSetSingleRule` e `RuleSetMultipleRules` implementam as abordagens *SingleRule* e *MultipleRules* descritas anteriormente. Deve ser ressaltado que é muito simples incluir no método RuleSet tanto abordagens utilizadas por outros sistemas de AM, além do C4.5, C4.5rules e CN2, quanto novas abordagens para classificar exemplos. Para isso, é somente necessário implementar as novas sub-classes correspondentes. Nos experimentos realizados foram utilizados o índice de precisão menos erro, Ipe, bem como o índice de cobertura Ic, definidos respectivamente pelas Equações 4.1 e 4.2 na página 78, como índices para estimar o poder de predição das regras.

Os resultados experimentais obtidos utilizando o índice Ic foram sempre inferiores aos resultados obtidos utilizando o índice Ipe (Milaré & Carvalho 2003). Assim, somente os resultados obtidos utilizando o índice Ipe são discutidos em detalhes no Capítulo 5.

4.3.3 O Módulo GA

O módulo GA implementa um Algoritmo Genético para realizar uma busca por um conjunto de regras que possua grande fidelidade com uma RNA. Como descrito previamente,

os AGs são métodos de busca inspirados no processo evolutivo. Por meio de operações como *crossover* e *mutação*, os AGs buscam por uma solução que seja ótima para o problema analisado.

O ambiente $NNRules$ utiliza um AG na busca por um conjunto de regras que classifique um conjunto de exemplos da mesma forma que uma RNA treinada classifica esses exemplos. Dessa forma, o conjunto de regras pode ser utilizado para explicar essa RNA. Sendo assim, o AG tem como principal objetivo encontrar um conjunto de regras que maximize a taxa de fidelidade entre o conjunto de regras e a RNA que será explicada pelo conjunto de regras.

No AG implementado no ambiente $NNRules$, cada indivíduo é um conjunto de regras gerenciado por uma instância do módulo RuleSet. As operações de *crossover* e mutação foram implementadas no módulo AG por meio dos métodos de adição e remoção de regras provida pela interface do módulo RuleSet.

Como explicado na Seção 1.4, um dos métodos propostos neste trabalho para extração de conhecimento de RNAs consiste em treinar uma RNA sobre um conjunto de dados. Posteriormente, esse mesmo conjunto de dados é rotulado pela RNA treinada e, com isso, um novo valor para o atributo classe é gerado. Os valores do atributo classe original são removidos e o conjunto de dados, mas com os novos valores do atributo classe, é fornecido para diversos sistemas de AM simbólico.

Os classificadores simbólicos induzidos pelos sistemas de AM são convertidos para regras na sintaxe $\mathcal{PBM}$ e carregados em uma instância do módulo RuleBase. Uma vez que as regras estão armazenadas na base de regras, o AG inicia a busca por um conjunto de regras com alta fidelidade com a RNA que gerou os novos valores do atributo classe.

O AG possui diversos parâmetros que devem ser ajustados para que o algoritmo forneça bons resultados. Um desses parâmetros é o número de indivíduos, n_i, que compõem a população. Inicialmente, o AG cria n_i indivíduos com regras selecionadas aleatoriamente, mas sem repetição, utilizando as regras armazenadas na base de regras. Por aleatório sem repetição deve-se entender que os indivíduos são gerados aleatoriamente, mas que um mesmo indivíduo não possui regras repetidas. Entretanto, uma mesma regra pode ocorrer em diferentes indivíduos. Um outro parâmetro importante é o tamanho de cada indivíduo, t_i, o qual, nesta aplicação, é o número de regras que cada indivíduo possui. Cada instância do módulo RuleSet pode armazenar um conjunto de regras de tamanho diferente. Atualmente, cada indivíduo gerado aleatoriamente para a primeira geração possui um mesmo tamanho t_i. Entretanto, são utilizados *crossovers assimétricos*, e o número

de regras de cada indivíduo pode aumentar ou diminuir a cada geração.

Em um *crossover* assimétrico são escolhidas duas posições, uma para cada indivíduo pai. Os indivíduos pais são divididos em dois segmentos, conforme as posições escolhidas. Por fim, são criados os indivíduos filhos por meio da composição dos segmentos dos indivíduos pai. Por exemplo, imagine os seguintes indivíduos pai compostos por seqüências de números inteiros, os quais correspondem a índices das regras armazenadas no módulo RuleBase:

$$\begin{array}{ll} pai_1 & \textit{07 02 11 00 04 08 03 01} \\ pai_2 & 05\ 23\ 10\ 06\ 12\ 09\ 20\ 15 \end{array}$$

O operador de *crossover* assimétrico escolhe duas posições, por exemplo, suponha que para o indivíduo pai_1 a posição depois do terceiro gene é escolhida, e para o indivíduo pai_2 é escolhida a posição depois do quinto gene.

$$\begin{array}{ll} pai_1 & \textit{07 02 11} \mid \textit{00 04 08 03 01} \\ pai_2 & 05\ 23\ 10\ 06\ 12 \mid 09\ 20\ 15 \end{array}$$

Então, como resultado, são gerados os seguintes indivíduos filhos:

$$\begin{array}{ll} filho_1 & \textit{07 02 11}\ 09\ 20\ 15 \\ filho_2 & 05\ 23\ 10\ 06\ 12\ \textit{00 04 08 03 01} \end{array}$$

Por meio da seleção realizada a cada geração, o AG pode tender a aumentar ou diminuir o número de regras em cada indivíduo. Dessa forma, um número apropriado de regras para os indivíduos pode ser encontrado para o domínio analisado.

Outros parâmetros relevantes dos AGs que devem ser ajustados para cada aplicação são: a probabilidade de *crossover*, p_c, a qual indica o percentual de indivíduos que são utilizados em *crossovers* a cada geração; a probabilidade de mutação, p_m, a qual indica o percentual de genes dos indivíduos que são mutados a cada geração; e o critério de parada, o qual é um critério que deve ser satisfeito para que o AG pare a execução e forneça um indivíduo como resposta.

Um aspecto importante para o bom funcionamento do AG é a definição da função de aptidão adequada. A função de aptidão avalia a aptidão dos indivíduos e fornece um índice para cada um deles. Os indivíduos são selecionados segundo esse índice e os mais aptos têm maior probabilidade de serem selecionados para a próxima geração.

Como mencionado anteriormente, nos experimentos apresentados no Capítulo 5, a *taxa de fidelidade* é utilizada como função de aptidão. A taxa de fidelidade é o percentual de exemplos de um conjunto de dados classificados de forma idêntica pelo conjunto de regras, isto é, um indivíduo, e a RNA que se deseja explicar.

A arquitetura do módulo GA é ilustrada na Figura 4.6. O módulo GA manipula n_i indivíduos, os quais são implementados por meio de n_i instâncias do módulo RuleSet. Cada instância do módulo RuleSet faz referência à base de regras gerenciada por uma instância do módulo RuleBase. Por meio de manipulações das regras contidas em cada indivíduo RuleSet, o módulo GA implementa as operações necessárias, como *crossover* e mutação.

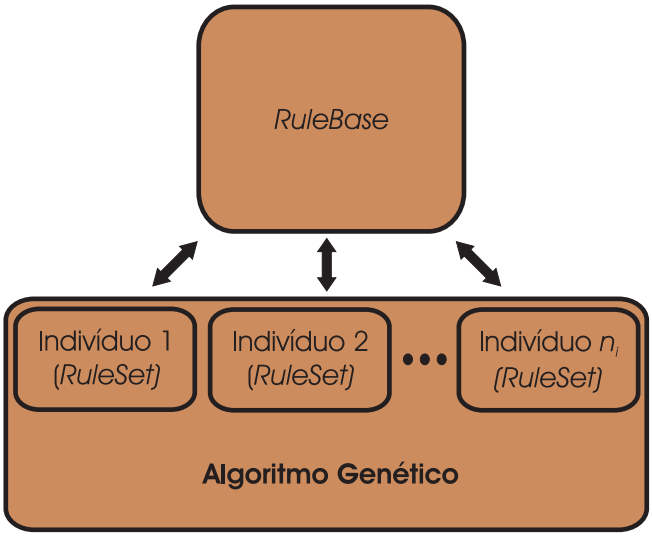

Figura 4.6: Arquitetura do módulo GA.

O projeto do módulo GA é apresentado por meio de um diagrama de classes em UML na Figura 4.7. O módulo GA é implementado por meio da classe `GA`, a qual faz uso de uma composição de objetos da classe `RuleSet`. Por sua vez, a classe `RuleSet` faz referência à classe RuleBase, a qual gerencia uma base de regras.

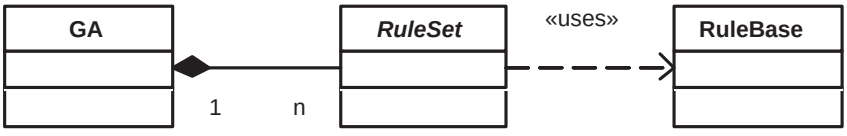

Figura 4.7: Projeto do módulo GA.

4.4 O Funcionamento do Ambiente $NNRules$

Nesta seção são fornecidos alguns detalhes sobre a implementação e o funcionamento do ambiente $NNRules$. Os módulos do ambiente $NNRules$ podem ser compostos tanto para implementar o método de extração de conhecimento de RNAs baseado puramente em sistemas de AM simbólico, quanto o método de extração de conhecimento de RNAs baseado em sistemas de AM simbólico e AGs.

Em ambos os casos, foi implementado um conjunto de *scripts* na linguagem Perl que implementam um experimento, compondo os módulos necessários e realizando chamadas aos métodos desses módulos para a realização das operações que compõem o experimento. Os experimentos iniciam com um conjunto de dados convertido para a sintaxe DSX. A sintaxe DSX[5] (Batista & Monard 2003) é a sintaxe padrão do sistema DISCOVER e permite declarar conjuntos de dados com diversos tipos de atributos, entre outros recursos.

Nos experimentos, os conjuntos de dados são separados em conjuntos de treinamento, teste e validação. Os conjuntos de treinamento são utilizados na indução dos classificadores. Os conjuntos de validação são utilizados para estimar os parâmetros dos modelos como, por exemplo, identificar uma arquitetura de RNA que forneça um desempenho adequado para o problema. Os conjuntos de teste são utilizados exclusivamente para medir o desempenho dos métodos de extração de conhecimento simbólico na forma de *taxa de infidelidade* e *compreensibilidade sintática*, descritas na Seção 5.3.2 na página 100.

Os conjuntos de treinamento, teste e validação são pré-processados para serem fornecidos à RNA. As tarefas de pré-processamento são realizadas com o auxílio da biblioteca DOL (Batista & Monard 2003). Entre as tarefas de pré-processamento necessárias estão a normalização dos atributos quantitativos e a conversão dos atributos qualitativos para atributos quantitativos por meio da codificação *one-of-n*. Após o pré-processamento, os conjuntos de treinamento, validação e teste são salvos na sintaxe do aplicativo SNNS[6] (Zell 1995).

O aplicativo SNNS é utilizado para construir e treinar a RNA utilizada no experimento. Diversas arquiteturas de RNAs são testadas e aquela que apresenta o melhor desempenho no conjunto de validação é selecionada. A RNA é, então, utilizada para rotular os exemplos dos conjuntos de treinamento, validação e teste. Os valores do atributo classe original são removidos dos dados.

[5] *Discover Dataset Sintax.*
[6] *Stuttgart Neural Network Simulator.*

Os conjuntos de treinamento, validação e teste são também salvos na sintaxe usada pelo método de extração de conhecimento de RNAs TREPAN, descrito no Seção 3.10 na página 62. O método TREPAN é utilizado pois esse método possui diversas similaridades com o método de extração de conhecimento baseado em sistemas de AM simbólico proposto neste trabalho. Dessa forma, o método TREPAN é utilizado como "*benchmark*" para verificar se os métodos propostos são de fato eficientes.

Os dados rotulados pela RNA são utilizados tanto para a extração de conhecimento por meio do método que utiliza sistemas de AM simbólico, quanto pelo método que utiliza sistemas de AM simbólico e AGs.

No caso do método de extração de conhecimento baseado em sistemas de AM simbólico, os conjuntos de treinamento e validação são concatenados e fornecidos diretamente para o sistema de AM simbólico e um classificador é induzido. O classificador simbólico é, então, fornecido ao módulo RuleBase. O módulo RuleBase converte o classificador simbólico para a sintaxe $\mathcal{PBM}$, e as regras no formato $\mathcal{PBM}$ são carregadas na base de regras.

É criada uma instância do módulo RuleSet relacionada com o sistema de AM simbólico que induziu o classificador. Nos experimentos realizados foram escolhidos os sistemas C4.5, C4.5rules e CN2 para serem utilizados como sistema de AM simbólico no método proposto baseado em AM simbólico e também no método baseado em sistemas de AM e AGs. As classes do módulo RuleSet, correspondentes ao C4.5, C4.5rules e CN2, são utilizadas para classificar os exemplos do conjunto de teste. Por meio da classificação do conjunto de teste é medida a taxa de infidelidade entre o classificador simbólico correspondente e a RNA, bem como a complexidade sintática do classificador simbólico.

O ambiente $NNRules$ gera relatórios para o usuário com informações sobre a taxa de infidelidade no conjunto de teste bem como a complexidade sintática dos classificadores e também do método TREPAN. Exemplos de relatórios gerados pelo ambiente $NNRules$ para os classificadores simbólicos e para o método TREPAN são mostrados nas Seções A.1 na página 123 e A.2 na página 125, respectivamente, do Apêndice A.

No caso do método de extração de conhecimento baseado em sistemas de AM simbólico e AGs, o conjunto de treinamento é fornecido a diversos indutores, os quais induzem diferentes classificadores. Como mencionado anteriormente, outras abordagens podem ser utilizadas, como a geração de diferentes classificadores por meio de diferentes amostras do conjunto de treinamento, ou por meio da variação de parâmetros utilizados pelos indutores no processo de indução.

Os diferentes classificadores simbólicos são, então, fornecidos ao módulo RuleBase. O módulo RuleBase converte os classificadores simbólicos para a sintaxe $\mathcal{PBM}$. Em seguida, as regras no formato $\mathcal{PBM}$ dos classificadores são carregadas na base de regras.

Uma vez que as regras foram carregadas na base de regras, o módulo GA é iniciado. Como já mencionado, o módulo GA cria n_i indivíduos os quais são compostos por conjuntos aleatórios mas sem repetição de regras armazenadas na base de regras. Cada indivíduo é armazenado e gerenciado por uma instância do módulo RuleSet. O usuário pode decidir qual classe do módulo RuleSet ele irá utilizar para gerenciar os indivíduos. Atualmente, o ambiente $NNRules$ provê duas classes do módulo RuleSet que podem ser utilizadas em conjunto com o módulo GA: a classe `RuleSetSingleRule` e a classe `RuleSetMultipleRules`.

O AG inicia a busca por um indivíduo que possua alta fidelidade com a RNA que rotulou os dados. A cada geração, os indivíduos são avaliados segundo a taxa de fidelidade[7] entre eles e a RNA. A taxa de fidelidade é calculada sobre os dados do conjunto de treinamento. O indivíduo com maior taxa de fidelidade a cada geração é sempre selecionado para a próxima geração. É calculada também a taxa de fidelidade média de todos os indivíduos da população em uma dada geração. A geração com melhor taxa de fidelidade média é salva para ser recuperada futuramente.

Atualmente, é utilizado como critério de parada um valor máximo, n_g, para o número de gerações. Após n_g gerações, o AG recupera a geração que obteve melhor taxa de fidelidade média em relação ao conjunto de treinamento. Dessa geração é escolhido um indivíduo, isto é, um conjunto de regras que será a resposta do AG para a explicar a RNA. A escolha do indivíduo da geração de indivíduos que obteve melhor taxa de fidelidade média é feita utilizando o conjunto de validação.

Uma outra abordagem poderia ter sido utilizada, como por exemplo, guardar o indivíduo com melhor taxa de fidelidade em relação ao conjunto de validação e ao final das n_g gerações esse indivíduo seria o escolhido para explicar a RNA.

Para escolher um indivíduo da geração que obteve melhor taxa de fidelidade média é aplicada a seguinte abordagem:

1. Cada indivíduo é avaliado utilizando o conjunto de validação. O indivíduo que

[7] O módulo GA utiliza a taxa de fidelidade entre a RNA e os indivíduos como função de aptidão. Essa decisão faz com que o problema a ser solucionado pelo AG seja um problema de maximização. Contudo, nos relatórios gerados pelo ambiente $NNRules$ são apresentadas a taxa de infidelidade e a complexidade sintática dos indivíduos.

obtiver a maior taxa de fidelidade para o conjunto de validação é escolhido como resposta para o AG.

2. Caso existam dois ou mais indivíduos com a mesma taxa de fidelidade para o conjunto de validação, então esses indivíduos são avaliados no conjunto de treinamento, e aquele que obtiver a maior taxa de fidelidade é retornado pelo AG.

3. Se ainda existirem dois ou mais indivíduos com as mesmas taxas de fidelidade para os conjuntos de validação e treinamento, então é aplicado o princípio *Occam's razor* (Mitchell 1997b, Capítulo 3). Segundo esse princípio, se duas ou mais hipóteses apresentam o mesmo desempenho, então é dado preferência à hipótese mais simples. Dessa forma, é avaliada a complexidade sintática de cada indivíduo e aquele que apresentar a menor complexidade sintática é retornado.

4. Se existirem dois ou mais indivíduos com as mesmas taxas de fidelidade e complexidade sintática, então um desses indivíduos é sorteado e retornado pelo AG.

Alternativamente, existe a possibilidade do módulo GA realizar um *pós-processamento* no conjunto de regras. O pós-processamento remove do conjunto de regras algumas regras que não são disparadas. Dessa forma, o pós-processamento é dependente da estratégia utilizada para classificar um exemplo. Algumas estratégias tendem a utilizar poucas regras para classificar um exemplo. Por exemplo, a estratégia *SingleRule* implementada pela classe `RuleSetSingleRule`, utiliza apenas uma regra para classificar um dado exemplo. Por outro lado, a estratégia *MultipleRules* utiliza todas as regras que cobrem um exemplo para classificar esse exemplo. Sendo assim, dado um conjunto de exemplos e um conjunto de regras, cada estratégia pode utilizar diferentes regras para classificar os exemplos do conjunto de exemplos e, até mesmo, não utilizar algumas das regras.

O objetivo do passo de pós-processamento é tentar identificar quais regras estão sendo utilizadas no processo de classificação e quais regras não estão. Deve-se notar que a remoção de regras pode levar a uma perda na precisão do conjunto de regras, mas também a uma redução na complexidade sintática do classificador. A possível perda na precisão do conjunto de regras é decorrente do fato que o pós-processamento é realizado utilizando um conjunto de exemplos de tamanho reduzido. As regras removidas poderiam ser úteis na classificação de novos exemplos que não foram utilizados no passo de pós-processamento. Mas, o contrário também pode acontecer, uma regra que classificaria um exemplo incorretamente pode ser removida no pós-processamento e, nesse caso, a precisão do conjunto de regras pode aumentar.

O passo de pós-processamento é realizado utilizando todos os exemplos disponíveis, ou seja, os exemplos dos conjuntos de treinamento e validação. O passo de pós-processamento inicia quando o AG termina a execução das n_g gerações. A geração com melhor desempenho médio é escolhida e todos os indivíduos são pós-processados, isto é, são verificadas quais regras disparam para os conjuntos de treinamento e validação. Para cada conjunto de regras, as regras que não são disparadas são removidas. Após o pós-processamento, é escolhido um indivíduo conforme explicado previamente, e esse indivíduo é retornado pelo AG.

O ambiente $NNRules$ gera relatórios para o usuário com informações sobre os indivíduos da geração com maior taxa de fidelidade média antes do pós-processamento — Seção A.4 na página 128 do Apêndice A, e também depois do pós-processamento — Seção A.5 na página 131 do Apêndice A. As regras, no formato da sintaxe $\mathcal{P}BM$, do indivíduo escolhido após o pós-processamento são gravadas em um arquivo, como mostrado no exemplo da Seção A.6 na página 134 do Apêndice A.

4.5 Considerações Finais

O objetivo deste capítulo foi oferecer uma visão geral sobre o ambiente computacional $NNRules$ para extração de conhecimento de RNAs. Esse ambiente é constituído de três módulos principais: o módulo para armazenar e gerenciar uma base de regras RuleBase, o módulo para criar e manipular um conjunto de regras RuleSet e o módulo GA, o qual implementa um AG para encontrar um conjunto de regras que possua alta fidelidade com uma RNA treinada.

Além dos módulos implementados ao ambiente computacional $NNRules$, foram também desenvolvidos vários *scripts* que executam tarefas específicas. Os *scripts* utilizam os módulos dos ambiente $NNRules$ e também a biblioteca DOL do projeto DISCOVER.

Uma das qualidades do ambiente $NNRules$ é a sua flexibilidade. Da forma como foi projetado e desenvolvido, é possível fazer uma série de alterações e inclusões de código sem desprender muito tempo e esforço. Por exemplo, na estrutura de dados utilizada para armazenar as regras, é possível, além de utilizar as informações armazenadas até então (a própria regra, o indutor que a induziu e sua matriz de contingência), incluir outras informações facilmente. Também, pode-se implementar outras abordagens para classificar um exemplo dado um conjunto de regras. Para isso, basta implementar uma sub-classe da classe abstrata RuleSet.

O ambiente *NNRules* foi utilizado em diversos experimentos que visam identificar se os métodos propostos neste trabalho podem ser utilizados para extrair conhecimento de RNAs. Uma descrição detalhada dos experimentos, bem como os resultados obtidos, são apresentados no próximo capítulo.

Capítulo 5

Avaliação Experimental dos Métodos Propostos

5.1 Considerações Iniciais

Conforme descrito na Seção 1.4 na página 10, neste trabalho são propostos dois métodos para extração de conhecimento simbólico de RNAs, sendo que um dos métodos é baseado em sistemas de AM simbólico e o outro baseado em sistemas de AM simbólico e Algoritmos Genéticos.

O objetivo deste capítulo é descrever a metodologia utilizada para avaliar esses métodos, bem como os experimentos realizados e os resultados obtidos. Este capítulo está organizado da seguinte forma: na Seção 5.2 são descritos os conjuntos de dados utilizados na realização dos experimentos; na Seção 5.3 são apresentados os experimentos realizados e os resultados obtidos nesses experimentos; na Seção 5.4 são apresentadas as considerações finais deste capítulo.

5.2 Conjuntos de Dados Utilizados

Nos experimentos realizados foram utilizados seis conjuntos de dados envolvendo problemas de classificação em diversos domínios de aplicação. Todos os conjuntos de dados foram retirados do repositório de dados da Universidade da Califórnia em Irvine — UCI (Blake, Keogh & Merz 1998). Os conjuntos de dados, além de serem de diversos domínios, variam em: número de exemplos; número de atributos; alguns possuem somente atributos

discretos, somente atributos contínuos ou ambos; e atributos com valores faltantes. Os seis conjuntos de dados e o nome pelo qual serão referenciados neste trabalho são:

- *Wisconsin Breast Cancer Database* — `breast`;
- *Credit Approval* — `crx`;
- *Heart Disease Databases* — `heart`[1];
- *Pima Indians Diabetes Database* — `pima`;
- *Sonar, Mines vs. Rocks* — `sonar`, e;
- *1984 United States Congressional Voting Records Database* — `votes`[2].

A seguir é apresentada uma breve descrição de cada um dos seis conjuntos de dados, bem como um resumo de suas características.

`breast` Esse conjunto de dados foi obtido dos hospitais da Universidade of Wisconsin, Madison, pelo Dr. William H. Wolberg. O problema consiste em predizer se uma amostra de tecido retirado da mama de uma paciente é um câncer maligno ou benigno. A cada amostra foi atribuído um vetor 9-dimensional. Cada componente encontra-se no intervalo de 1 a 10, sendo que 1 significa estado normal e 10 estado anormal. O grau de quão maligno o tecido é foi determinado por meio da retirada de uma amostra de tecido da mama da paciente e realização de uma biópsia. Um diagnóstico benigno é confirmado por biópsia ou por exames periódicos, dependendo da escolha do paciente.

`crx` Esse conjunto de dados está relacionado com aplicações para obtenção de um cartão de crédito. Todos os nomes dos atributos e valores foram alterados para símbolos sem significado para proteger a confidencialidade dos dados.

`heart` Os dados deste conjunto de dados foram coletados na Cleveland Clinic Foundation e um dos principais responsáveis foi Robert Detrano. O problema consiste em predizer se o paciente sofre de alguma doença do coração ou não. O conjunto de dados possui 76 atributos, mas somente um subconjunto de 14 atributos é utilizado nos experimentos.

[1] Esse conjunto de dados foi utilizado em (Craven 1996) para avaliar o algoritmo Trepan.
[2] Esse conjunto de dados foi utilizado em (Craven 1996) para avaliar o algoritmo Trepan.

`pima` Esse conjunto de dados foi doado por V. Sigillito do Laboratório de Física Aplicada, Universidade Johns Hopkins. É um subconjunto de uma base de dados maior mantida pelo Instituto Nacional de Diabetes e Doenças Digestivas e Renais nos Estados Unidos. Todos os pacientes são mulheres com idade mínima de 21 anos, descendentes indígenas Pima e que vivem próximas à Phoenix, Arizona, EUA. O problema consiste em predizer se uma paciente apresentará um resultado positivo para diabetes, de acordo com os critérios estabelecidos pela Organização Mundial de Saúde, mediante exames laboratoriais.

`sonar` Esse conjunto de dados foi utilizado por Gorman e Sejnowski no estudo de classificação de sinais de sonar utilizando uma RNA (Gorman & Sejnowski 1988). O problema consiste em determinar entre sinais de sonar que representam um cilindro de metal daqueles que representam uma rocha ligeiramente cilíndrica. O conjunto de dados possui 111 exemplos obtidos por varredura de sonar de um cilindro de metal em vários ângulos e sob diversas condições, e 97 exemplos obtidos por varredura de rochas sob as mesmas condições. Cada um dos exemplos possui 60 volores reais no intervalo entre 0 e 1. Cada valor representa a energia em uma banda de freqüência particular integrada sobre um determinado período de tempo. A classe associada com cada exemplo contém a letra `R` se o objeto é uma rocha e `M` se ele é uma mina (cilindro de metal).

`votes` Esse conjunto de dados foi doado por Jeff C. Schlimmer e representa votações realizadas por membros da câmara de deputados dos Estados Unidos em 1984. O valor de cada atributo indica se um determinado representante votou sim, não ou não votou. O problema consiste em predizer se um representante é membro do partido Democrata ou se é membro do partido Republicano.

Na Tabela 5.1 na página seguinte é apresentado um resumo de algumas características dos conjuntos de dados descritos anteriormente. Essas características referem-se a:

- `#Exemplos` – número total de exemplos presentes em cada conjunto de dados;
- `#Atributos` – número total de atributos e número de atributos contínuos e discretos;
- `Classes e %Classes` – valores das classes e suas distribuições;
- `Erro Majoritário` – erro cometido quando novos exemplos são classificados como sendo da classe majoritária;

- `Valores Desconhecidos` - se o conjunto de dados apresenta ou não valores desconhecidos.

Conjunto de Dados	#Exemplos	#Atributos (cont.,disc)	Classes	%Classes	Erro Majoritário	Valores Desconhecidos
`breast`	699	9(9,0)	`benign`	65,52%	34,48%	sim
			`malignant`	34,48%		
`crx`	690	15(6,9)	`-`	55,50%	44,50%	sim
			`+`	44,50%		
`heart`	303	13(6,7)	`absence`	54,13%	45,87%	sim
			`presence`	45,87%		
`pima`	768	8(8,0)	`0`	65,02%	34,98%	não
			`1`	34,98%		
`sonar`	208	60(60,0)	`M`	53,37%	46,63%	não
			`R`	46,63%		
`votes`	435	16(0,16)	`republican`	54,80%	45,20%	não
			`democrat`	45,20%		

Tabela 5.1: Características dos conjuntos de dados.

Na Tabela 5.1 pode-se notar que os conjuntos de dados `breast`, `crx` e `heart` possuem exemplos em que para um ou mais atributos há valores desconhecidos. Como o número de exemplos com valores desconhecidos é pequeno, decidiu-se remover esses exemplos dos conjuntos de dados. Assim, foram descartados 2.3% de exemplos do conjunto de dados `breast`; 5.4% de exemplos do conjunto de dados `crx`; e 2.0% de exemplos do conjunto de dados `heart`.

5.3 Experimentos Realizados

Para avaliar os métodos propostos para extração de conhecimento simbólico de RNAs, alguns experimentos foram realizados utilizando os conjuntos de dados `breast`, `crx`, `heart`, `pima`, `sonar` e `votes`, descritos previamente. Os métodos propostos utilizam sistemas de AM simbólico, assim, foram escolhidos os sistemas de AM simbólico C4.5, C4.5rules (Quinlan 1988) e CN2 (Clark & Niblett 1989) para serem usados nos experimentos. Deve ser ressaltado que esses três sistemas são muito utilizados pela comunidade de AM para realizar experimentos e comparações com outras propostas, devido à sua robutez.

Os experimentos realizados foram divididos em três etapas para facilitar a compreensão e a descrição. Na Tabela 5.2 na próxima página são mostrados os principais passos realizados em cada uma dessas etapas. A seguir, as etapas são descritas em detalhes na

ordem em foram executadas. Como já mencionado, o ambiente computacional $NNRules$, desenvolvido neste trabalho, provê suporte à realização de todas as tarefas envolvidas na realização dos experimentos.

Etapa 1
Particionamento dos conjuntos de dados
Treinamento das RNAs
Execução do C4.5, C4.5rules e CN2
Medição da taxa de erro
Etapa 2 - Método Baseado em Sistemas de AM Simbólico
Execução do TREPAN
Geração de conjuntos de treinamento com novos valores para o atributo classe
Execução do C4.5, C4.5rules e CN2
Medição da taxa de infidelidade
Medição da compreensibilidade
Etapa 3 - Método Baseado em Sistemas de AM Simbólico e AGs
Execução do AG
Medição da taxa de infidelidade
Medição da compreensibilidade

Tabela 5.2: Passos executados em cada uma das etapas do experimento.

5.3.1 Etapa 1

O objetivo da Etapa 1 é treinar as RNAs das quais o conhecimento será extraído nas próximas etapas. Para isso, os conjuntos de dados são particionados segundo um método de *resampling.*, os sistemas de AM simbólico C4.5, C4.5rules e CN2 são executados e a taxa de erro da RNA e dos sistemas de AM simbólico são medidas e comparadas. Nessa etapa, os sistemas de AM simbólico são usados como *benchmarks* para verificar se a topologia e os parâmetros escolhidos para treinamento das RNAs foram adequados para os conjuntos de dados utilizados. A seguir, os passos da Etapa 1 dos experimentos são detalhadamente descritos, bem como os resultados obtidos.

5.3.1.1 Descrição

Os passos da Etapa 1 foram realizados da seguinte forma:

1. Os seis conjuntos de dados foram convertidos para a sintaxe DSX. Como descrito previamente, a sintaxe DSX é a sintaxe para conjunto de dados utilizada no ambi-

ente DISCOVER. Em seguida, cada um dos seis conjuntos de dados foi embaralhado e dividido utilizando o método de *resampling k-fold stratified cross-validation* (Weiss & Kulikowski 1991), com $k = 10$. No particionamento do conjunto de dados pelo método de *resampling k-fold stratified cross-validation*, os exemplos são aleatoriamente divididos em k partições mutuamente exclusivas (*folds*), sendo que em cada uma das partições a distribuição das classes é preservada. Isso significa que se no conjunto de dados original há, por exemplo, 60% de exemplos da classe A e 40% de exemplos da classe B, então em cada partição haverá essa mesma proporção de classes. Aproximadamente, o tamanho de cada partição é igual a N/k, sendo N o número total de exemplos. Os exemplos nas (k - 1) partições são utilizados para treinamento e a hipótese induzida é testada na partição remanescente. Esse processo é repetido k vezes, cada vez considerando uma partição diferente para teste.

Como resultado, foram obtidos 10 conjuntos de treinamento e 10 conjuntos de teste para cada um dos seis conjuntos de dados. Em seguida, os conjuntos de treinamento foram particionados em dois subconjuntos: conjunto de treinamento, contendo 90% dos exemplos; e conjunto de validação, contendo 10% dos exemplos.

Em todas as etapas do experimento foram utilizados esses mesmos conjuntos de treinamento, validação e teste, inclusive a ordem com que os exemplos aparecem nesses conjuntos foi a mesma.

2. Como os sistemas de AM simbólico C4.5, C4.5rules e CN2 são usados nessa etapa como *benchmarks*, os conjuntos de treinamento e validação foram concatenados e convertidos para a sintaxe utilizada por esses sistemas. O conjunto de teste também foi convertido para a sintaxe do C4.5, C4.5rules e CN2.

   Em seguida, cada conjunto de treinamento (conjunto de treinamento e validação concatenados) foi fornecido aos sistemas de AM simbólico. C4.5, C4.5rules e CN2 foram executados com os valores *default* de seus parâmetros. Ao final desse passo, 30 classificadores (10 classificadores C4.5, 10 classificadores C4.5rules e 10 classificadores CN2) foram induzidos para cada um dos seis conjuntos de dados, perfazendo um total de 180 classificadores. A taxa de erro, ou seja, a média dos erros estimados em cada um dos 10 classificadores de cada sistema de AM simbólico para cada conjunto de dados, foi medida utilizando o conjunto de teste associado ao conjunto de treinamento.

3. Os conjuntos de treinamento, validação e teste foram devidamente normalizados e codificados para serem utilizados no treinamento das RNAs, bem como convertidos

para a sintaxe utilizada pelo simulador SNNS (Zell 1995).

Para cada conjunto de treinamento e validação foi treinada uma RNA utilizando o simulador SNNS. O algoritmo de treinamento utilizado para treinar as redes, para todos os seis conjuntos de dados, foi o algoritmo *Backpropagation with Momentum* (Rumelhart, Hilton & Williams 1986). O conjunto de validação foi utilizado para decidir quando parar o treinamento das RNAs. As arquiteturas das RNAs foram escolhidas depois de testar várias topologias e vários valores de parâmetros. A escolha se deu utilizando o desempenho das redes para os conjuntos de validação. Na Tabela 5.3 é mostrada a arquitetura e os valores dos parâmetros escolhidos para cada um dos conjuntos de dados.

Conjunto de Dados	Arquitetura	#Máximo de ciclos	Pesos	Parâmetros (η, μ, c, d_{max})
`breast`	9-3-1	1000	0.5, -0.5	0.1, 0.7, 0.25, 0.2
`crx`	43-7-1	350	0.5, -0.5	0.2, 0.6, 0.25, 0.1
`heart`	22-1-1	100	0.5, -0.5	0.2, 0.7, 0.25, 0.2
`pima`	8-2-1	100	0.5, -0.5	0.2, 0.6, 0.25, 0.2
`sonar`	60-12-1	200	0.5, -0.5	0.1, 0.7, 0.25, 0.1
`votes`	48-1	50	0.5, -0.5	0.2, 0.6, 0.1, 0.2

Tabela 5.3: Arquitetura e parâmetros das RNAs treinadas.

Os parâmetros da RNA mostrados na Tabela 5.3 significam:

- η: parâmetro de aprendizado, especifica o tamanho do passo do gradiente descendente;
- μ: termo *momentum*, especifica a quantidade de mudanças de pesos anteriores que é adicionada à mudança atual;
- c: valor de eliminação de superfícies planas, permite à rede ultrapassar superfícies planas, e;
- d_{max}: diferença entre a saída da rede e a saída desejada que é tolerada.

Ao final desse passo, 10 RNAs foram treinadas para cada conjunto de dados, perfazendo um total de 60 redes. A taxa de erro foi medida utilizando o conjunto de teste correspondente ao conjunto de treinamento e validação.

Na Figura 5.1 na página seguinte é mostrada uma visão geral da Etapa 1 dos experimentos realizados, cujos passos foram previamente descritos.

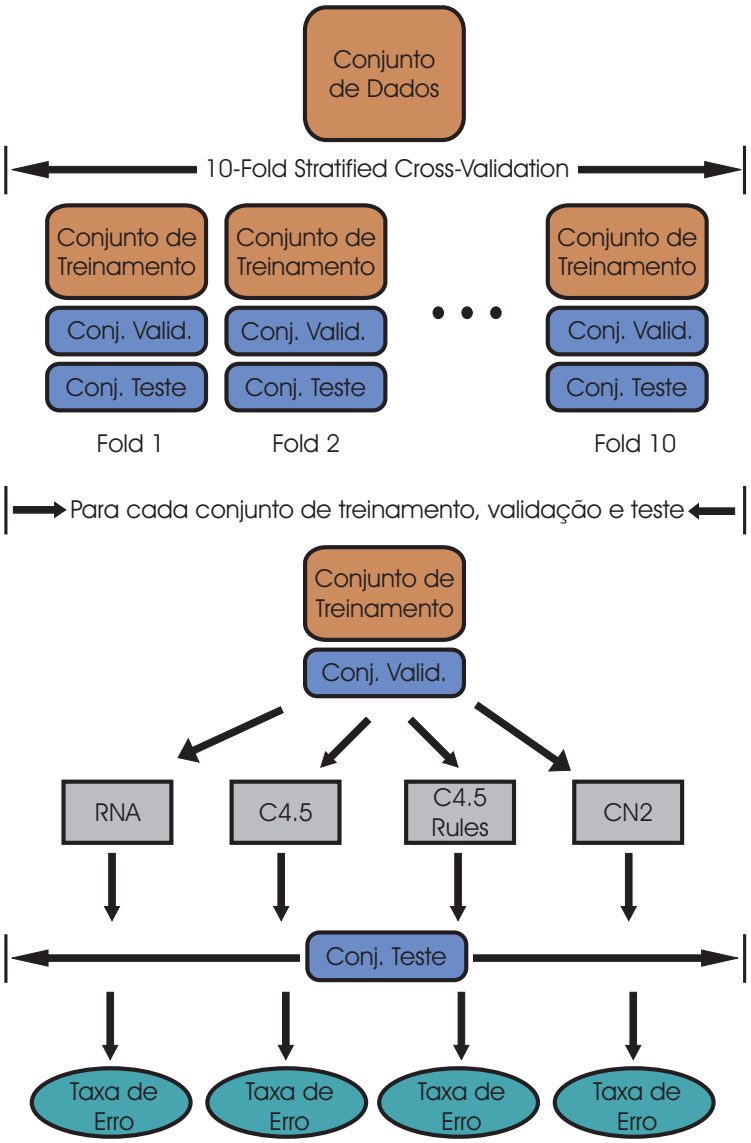

Figura 5.1: Visão geral da Etapa 1 dos experimentos realizados.

5.3.1.2 Resultados Obtidos

Nesta seção são apresentados os resultados da Etapa 1 dos experimentos, cujos passos foram descritos previamente. As taxas de erro (média e desvio padrão) obtidas pelo C4.5, C4.5rules, CN2 e RNA para os seis conjuntos de dados são mostradas na Tabela 5.4 na próxima página. Os valores em negrito indicam a menor taxa de erro para cada conjunto de dados. Pode ser observado que a taxa de erro obtida pela RNA só não foi menor que a do C4.5rules para o conjunto de dados `votes`, mesmo assim o comportamento da taxa de erro da RNA foi semelhante à do C4.5rules.

	RNA	C4.5	C4.5rules	CN2
breast	**2.98 ± 0.46**	3.51 ± 0.54	3.81 ± 0.77	4.97 ± 0.87
crx	**13.47 ± 1.01**	15.77 ± 1.61	14.39 ± 1.23	17.15 ± 0.85
heart	**17.78 ± 2.12**	23.33 ± 1.57	18.15 ± 1.40	21.11 ± 1.47
pima	**22.92 ± 1.15**	26.04 ± 1.45	26.57 ± 1.69	27.21 ± 1.05
sonar	**20.14 ± 3.21**	25.95 ± 2.76	25.48 ± 2.84	28.79 ± 3.86
votes	3.69 ± 0.86	5.06 ± 1.18	**3.67 ± 0.92**	5.98 ± 1.05

Tabela 5.4: Taxa de erro (média e desvio padrão).

Para verificar se a diferença dos resultados obtidos entre a RNA e os algoritmos de aprendizado simbólico é significante ou não, foi utilizado o teste de hipótese *k-fold cross-validated paired t* (Dietterich 1997). Segundo esse teste, dois algoritmos têm uma diferença significativa na taxa de erro, com 95% de confiabilidade, se o resultado do teste, em valor absoluto, for maior ou igual a 2.262. Na Tabela 5.5 é mostrado o resultado do teste de hipótese *10-fold cross-validated paired t*. Os valores em negrito indicam resultados significativos, com 95% de confiabilidade. Valores positivos significam que a RNA supera o sistema de AM simbólico. Valores negativos significam que o sistema de AM simbólico supera a RNA.

Pode ser observado que a RNA foi superior ao C4.5 com 95% de confiabilidade para os conjuntos de dados heart e pima. Quanto ao C4.5rules, a RNA foi superior com 95% de confiabilidade para o conjunto de dados pima. Em relação ao CN2, a RNA só não foi superior com 95% de confiabilidade para os conjuntos de dados heart e sonar.

	C4.5					
	breast	crx	heart	pima	sonar	votes
RNA	1.251	2.083	**2.422**	**2.276**	1.423	1.489
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
RNA	1.357	1.323	0.216	**2.978**	1.355	-0.014
	CN2					
	breast	crx	heart	pima	sonar	votes
RNA	**3.307**	**4.430**	1.273	**4.962**	2.152	**2.384**

Tabela 5.5: Resultados do teste de hipótese referentes à taxa de erro.

Na Figura 5.2 na próxima página são mostrados graficamente os resultados do teste de hipótese *10-fold cross-validated paired t*. Barras acima de zero significam que a RNA supera o sistema de AM simbólico (C4.5, C4.5rules ou CN2). Barras abaixo de zero signi-

ficam que o sistema de AM simbólico (C4.5, C4.5rules ou CN2) supera a RNA. Qualquer barra que ultrapasse a linha pontilhada do gráfico indica um resultado com 99% de confiabilidade (3.250 e −3.250). Qualquer barra que ultrapasse a linha contínua significa um resultado com 95% de confiabilidade (2.262 e −2.262). Pode-se observar que a RNA obteve 3 resultados com 99% de confiabilidade, sob o CN2 para os conjuntos de dados `breast`, `crx` e `pima`.

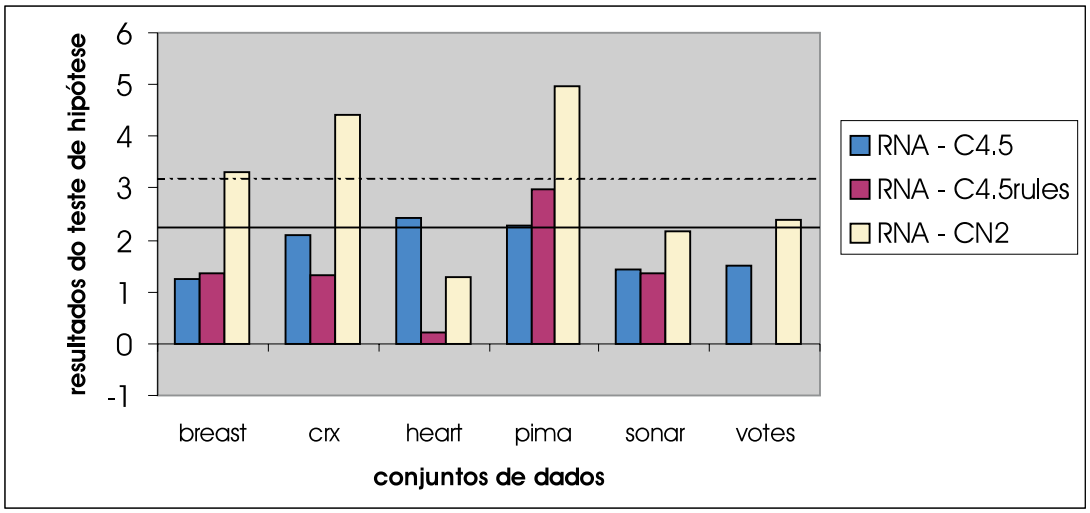

Figura 5.2: Resultados do teste de hipótese mostrados graficamente.

5.3.2 Etapa 2 - Método Baseado em Sistema de AM Simbólico

O objetivo da Etapa 2 é avaliar o método proposto baseado em sistemas de AM simbólico. Nessa etapa, as RNAs treinadas na Etapa 1 são utilizadas para rotular os conjuntos de dados, o método TREPAN é executado e os sistemas de AM simbólico C4.5, C4.5rules e CN2 são executados, todos os quatro com os conjuntos de treinamento com os novos rótulos encontrados pela RNA. A seguir, os passos da Etapa 2 dos experimentos são detalhadamente descritos, bem como os resultados obtidos.

5.3.2.1 Descrição

Os passos da Etapa 2 foram realizados da seguinte forma:

1. Inicialmente, os conjuntos de treinamento e teste, utilizados no treinamento e avaliação das RNAs na Etapa 1, foram convertidos para a sintaxe usada pelo método

TREPAN. TREPAN foi executado, sendo fornecidos os pesos e *thresholds* de cada RNA treinada na Etapa 1. O método TREPAN foi executado com diversos valores para seus parâmetros.

Na Tabela 5.6 são mostrados os diferentes valores para os parâmetros do método TREPAN utilizados nesse experimento. O parâmetro *MinSample* especifica o número mínimo de exemplos que o TREPAN utiliza para decidir sobre o teste de divisão de cada nó interno da árvore de decisão e também para rotular um nó folha. O valor *default* é 1000. Quando se atribui o valor 0 para este parâmetro, nenhum exemplo artificial é gerado pelo TREPAN. O parâmetro *SplitTest* define se os testes dos nós internos são expressões *m-de-n* (`mofn`) ou se são testes *booleanos* simples (`simple`), como os utilizados pelo algoritmo C4.5. A opção mofn1000 é a opção *default* utilizada pelo método TREPAN, ou seja, é utilizada expressão *m-de-n* e 1000 exemplos no mínimo devem estar disponíveis no nó corrente da árvore de decisão para que o nó seja expandido ou se torne um nó folha.

	MinSample	SplitTest
simple0	0	`simple`
simple1000	1000	`simple`
mofn0	0	`mofn`
mofn1000	1000	`mofn`

Tabela 5.6: Valores dos parâmetros do método TREPAN utilizados nos experimentos.

Ao final desse passo, 40 classificadores (10 classificadores simple0, 10 classificadores simple1000, 10 classificadores mofn0 e 10 classificadores mofn1000) foram induzidos para cada um dos seis conjuntos de dados, perfazendo um total de 240 classificadores.

A taxa de infidelidade entre a RNA e o TREPAN foi medida utilizando os conjuntos de teste. Como mencionado anteriormente, a taxa de fidelidade é definida como a porcentagem dos exemplos do conjunto de teste cuja classificação dada por um classificador qualquer concorda com a classificação dada pela RNA. A taxa de infidelidade é a porcentagem dos exemplos do conjunto de teste cuja classificação dada por um classificador qualquer não concorda com a classificação dada pela RNA. Nos experimentos realizados foi medida a taxa de infidelidade média, ou seja, a média das taxas de infidelidades estimadas em cada uma das 10 partições.

A compreensibilidade do conhecimento extraído pelo TREPAN, no caso árvores de decisão, também foi medida. Para isso foram utilizadas as seguintes medidas de

complexidade sintática: número de regras induzidas e número médio de condições por regra induzida.

Na Seção A.2 na página 125 — Apêndice A, é mostrado um exemplo de relatório gerado pelo ambiente *NNRules* com informações sobre a taxa de infidelidade e complexibilidade sintática medidas para o método TREPAN.

2. Os conjuntos de treinamento, validação e teste utilizados para treinar as RNAs na Etapa 1 foram rotulados pelas respectivas RNAs. Ao final desse passo, foram gerados 10 conjuntos de treinamento, validação e teste com novos rótulos para cada um dos seis conjuntos de dados.

3. Os conjuntos de treinamento e teste foram convertidos para a sintaxe dos sistemas de AM simbólico C4.5, C4.5rules e CN2. Em seguida, os sistemas de AM simbólico foram executados, com os valores *default* para seus parâmetros, sobre os conjuntos de treinamento. Ao final desse passo, 30 classificadores (10 classificadores C4.5, 10 classificadores C4.5rules e 10 classificadores CN2) foram induzidos para cada um dos seis conjuntos de dados, perfazendo um total de 180 classificadores. A taxa de infidelidade entre a RNA e os sistemas de AM simbólico foi estimada utilizando os conjuntos de teste associados aos conjuntos de treinamento. A complexidade sintática dos classificadores induzidos também foi medida.

   Na Seção A.1 na página 123 — Apêndice A, é mostrado um exemplo de relatório gerado pelo ambiente *NNRules* com informações sobre a taxa de infidelidade e complexibilidade sintática medidas para os sistemas de AM simbólico.

Na Figura 5.3 na próxima página é mostrada uma visão geral da Etapa 2 dos experimentos realizados, cujos passos foram descritos previamente.

5.3.2.2 Resultados Obtidos

Nesta seção são apresentados os resultados obtidos na Etapa 2 dos experimentos. A taxa de infidelidade (média e desvio padrão) para os seis conjuntos de dados obtidas pelos algoritmos de aprendizado simbólico C4.5, C4.5rules, CN2 e pelo TREPAN, este último executado com diferentes valores para seus parâmetros, são mostradas na Tabela 5.7 na página oposta. Os valores em negrito indicam a menor taxa de infidelidade para cada conjunto de dados.

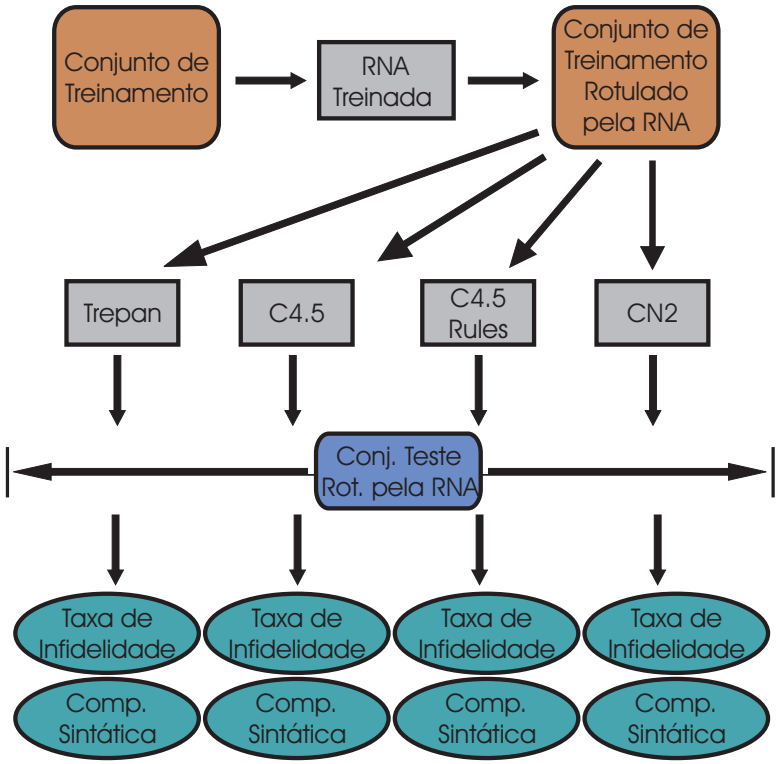

Figura 5.3: Visão geral da Etapa 2 dos experimentos realizados.

	breast	crx	heart	pima	sonar	votes
C4.5	2.93 ± 0.53	**2.91 ± 0.81**	14.07 ± 1.54	8.85 ± 1.07	22.98 ± 3.21	3.21 ± 0.97
C4.5rules	2.48 ± 0.53	3.06 ± 0.72	10.00 ± 1.11	8.46 ± 1.01	23.43 ± 3.25	2.98 ± 1.07
CN2	**1.90 ± 0.62**	5.82 ± 0.75	13.70 ± 1.99	9.12 ± 0.94	22.57 ± 3.35	2.75 ± 0.95
simple0	3.37 ± 0.44	3.36 ± 0.71	14.07 ± 1.81	10.81 ± 0.65	24.90 ± 3.06	3.67 ± 1.03
simple1000	4.39 ± 0.95	5.37 ± 1.85	12.22 ± 2.53	8.86 ± 1.15	**22.14 ± 2.91**	**2.30 ± 0.96**
mofn0	2.78 ± 0.63	3.83 ± 0.84	12.22 ± 1.75	10.42 ± 1.31	25.57 ± 3.15	2.99 ± 0.97
mofn1000	3.07 ± 0.80	4.75 ± 1.49	**7.04 ± 1.40**	**7.95 ± 1.01**	27.36 ± 2.73	2.76 ± 0.75

Tabela 5.7: Taxa de infidelidade (média e desvio padrão) — Etapa 2.

Pode ser observado na Tabela 5.7 que os sistemas de AM simbólico obtiveram taxa de infidelidade menor para dois dos conjuntos de dados: `breast` com o CN2 e `crx` com o C4.5. Já o TREPAN obteve taxa de infidelidade menor para o restante dos conjuntos de dados: `heart` e `pima` com o mofn1000; e, `sonar` e `votes` com o simple1000. Entretanto, para três desses conjuntos de dados (`pima`, `sonar` e `votes`), há no mínimo um sistema de AM simbólico que obteve taxa de infidelidade semelhante à menor taxa de infidelidade conseguida pelo TREPAN. Por exemplo, simple1000 obteve a melhor taxa de infidelidade, 22.14 ± 2.91, para o conjunto de dados `sonar`. O sistema de AM simbólico CN2 obteve para esse conjunto de dados a taxa de infidelidade de 22.57 ± 3.35, que é muito semelhante à taxa obtida pelo simple1000.

Os resultados referentes à complexidade sintática, número de regras induzidas e número médio de condições por regra, são mostrados nas Tabelas 5.8 e 5.9, respectivamente. Os melhores resultados estão em negrito.

Para medir o número de regras induzidas, as árvores de decisão do TREPAN e do C4.5 foram transformadas em conjuntos de regras. As regras *default* induzidas pelo C4.5rules e pelo CN2 não foram consideradas.

Observando a Tabela 5.8 nota-se que o TREPAN executado com a opção mofn1000 obteve, para os conjuntos de dados `breast`, `heart` e `sonar`, os menores valores quanto ao número de regras induzidas. O algoritmo simbólico C4.5rules obteve os menores valores para os outros conjuntos de dados, `crx`, `pima` e `votes`.

	`breast`	`crx`	`heart`	`pima`	`sonar`	`votes`
C4.5	8.90 ± 0.91	14.10 ± 3.44	17.40 ± 1.33	25.00 ± 0.92	13.50 ± 0.34	9.40 ± 1.07
C4.5rules	8.00 ± 0.54	**8.70 ± 1.58**	11.40 ± 0.82	**18.30 ± 1.16**	8.10 ± 0.38	**5.80 ± 0.44**
CN2	12.10 ± 0.55	13.50 ± 1.42	16.00 ± 0.75	24.40 ± 0.97	25.20 ± 0.53	13.30 ± 0.67
simple0	9.40 ± 0.48	31.50 ± 4.81	18.10 ± 0.99	26.10 ± 1.48	12.40 ± 0.60	13.60 ± 0.90
simple1000	10.60 ± 0.73	31.00 ± 8.03	20.00 ± 1.92	29.30 ± 2.01	9.50 ± 1.10	11.00 ± 1.61
mofn0	7.40 ± 0.78	11.10 ± 1.76	9.20 ± 0.83	23.30 ± 1.38	6.90 ± 0.53	7.10 ± 0.53
mofn1000	**2.40 ± 0.31**	9.90 ± 1.94	**5.30 ± 0.83**	26.00 ± 1.55	**4.10 ± 0.67**	6.50 ± 0.72

Tabela 5.8: Número de regras induzidas (média e desvio padrão) — Etapa 2.

Para medir o número médio de condições de uma regra quando a árvore de decisão utiliza expressões *m-de-n* nos testes dos nós internos, tais como mofn0 e mofn1000, optou-se por contar as condições da expressão como n condições. Assim, por exemplo, a expressão $2-de-\{x_1, \neg x_2, x_3\}$ possui 3 condições. Note que as expressões $1-de-3$, $2-de-3$ e $3-de-3$ possuem o mesmo número de condições, ou seja, 3, entretanto, elas não possuem o mesmo nível de compreensibilidade. É mais simples compreender a expressão $3-de-\{x_1, \neg x_2, x_3\}$, que é logicamente equivalente à:

$$x_1 \wedge \neg x_2 \wedge x_3 \tag{5.1}$$

do que compreender a expressão $2-de-\{x_1, \neg x_2, x_3\}$, que é logicamente equivalente à:

$$(x_1 \wedge \neg x_2) \vee (x_1 \wedge x_3) \vee (\neg x_2 \wedge x_3) \tag{5.2}$$

Uma outra alternativa para contar as condições de expressões *m-de-n* seria utilizar

a Equação 5.3:

$$\binom{n}{m} m = \frac{n!}{m!(n-m)!} m \tag{5.3}$$

onde m e n são, respectivamente, o *threshold* inteiro e o número de literais *booleanos* da expressão *m-de-n*. Por exemplo, considerando a expressão $2-de-\{x_1, \neg x_2, x_3\}$, na qual $m = 2$ e $n = 3$, e substituindo os valores de m e n na Equação 5.3 tem-se:

$$\binom{3}{2} 2 = \frac{3!}{2!(3-2)!} 2 = 6$$

Portanto a expressão $2-de-\{x_1, \neg x_2, x_3\}$, que é logicamente equivalente à Expressão 5.2 na página anterior, possuiria 6 condições. Entretanto, foi decidido não utilizar a Equação 5.3 para determinar o número de condições de uma expressão *m-de-n*, pois, em alguns casos esse valor seria muito alto. Por exemplo, considere um classificador com apenas a regra:

$$if\ 6-de-\{x_1, \neg x_2, \neg x_3, x_4, \neg x_5, x_6, \neg x_7, x_8\}\ then\ class = 1$$

Ao substituir m por 6 e n por 8 na Equação 5.3, tem-se 56 condições. Lembrando que o classificador possui apenas uma regra, o número médio de condição por regra desse classificador seria 56, o que é um número excessivamente grande, penalizando assim o uso de expressões *m-de-n* do TREPAN.

Pode ser observado na Tabela 5.9 que, mesmo com a simplificação adotada, a qual favorece o TREPAN, os melhores resultados, ou seja, os menores valores, para todos os conjuntos de dados, foram obtidos pelos sistemas de AM C4.5rules e CN2. O TREPAN não obteve nenhum melhor resultado.

	breast	crx	heart	pima	sonar	votes
C4.5	3.63 ± 0.17	2.62 ± 0.46	3.25 ± 0.06	5.66 ± 0.11	4.27 ± 0.11	2.68 ± 0.20
C4.5rules	2.57 ± 0.05	**2.17 ± 0.10**	**2.41 ± 0.04**	**2.92 ± 0.05**	2.84 ± 0.13	**2.52 ± 0.11**
CN2	**2.56 ± 0.07**	2.84 ± 0.08	2.92 ± 0.10	3.02 ± 0.05	**2.03 ± 0.02**	2.64 ± 0.05
simple0	3.54 ± 0.11	2.78 ± 0.10	3.05 ± 0.11	5.58 ± 0.13	3.93 ± 0.13	2.83 ± 0.10
simple1000	4.00 ± 0.19	2.61 ± 0.27	3.27 ± 0.12	5.94 ± 0.17	3.81 ± 0.24	2.61 ± 0.18
mofn0	6.74 ± 0.54	7.62 ± 0.75	9.47 ± 0.59	10.23 ± 0.49	9.35 ± 0.49	5.46 ± 0.41
mofn1000	7.49 ± 0.48	10.10 ± 0.91	10.17 ± 0.41	10.73 ± 0.32	23.41 ± 1.51	9.04 ± 0.90

Tabela 5.9: Número médio de condição por regra induzida (média e desvio padrão) — Etapa 2.

Para verificar se a diferença dos resultados obtidos entre os sistemas de AM simbólico e o TREPAN são significativos ou não, foi utilizado o teste de hipótese *10-fold cross-*

validated paired t. Como mencionado anteriormente, segundo esse teste, dois algoritmos possuem diferença significativa, com 95% de confiabilidade, se o resultado do teste, em valor absoluto, for maior ou igual a 2.262.

Os valores do teste de hipótese *10-fold cross-validated paired t* são calculados pelo ambiente *NNRules* e colocados em um relatório. Na Seção A.7 na página 135 — Apêndice A, é mostrado um exemplo de relatório com os valores do teste de hipótese.

As tabelas nas quais são mostrados os resultados do teste de hipótese *10-fold cross-validated paired t* referentes à taxa de infidelidade, número de regras induzidas e número médio de condição por regra induzida, são apresentadas no Apêndice B na página 137. Em cada tabela, os sistemas de AM simbólico são comparados ao método TREPAN. Os valores em negrito indicam resultados significativos, com 95% de confiabilidade. Valores positivos significam que um sistema de AM simbólico supera o TREPAN. Valores negativos significam que o TREPAN supera um sistema de AM simbólico.

Na Tabela 5.10 são mostrados os resultados do teste *10-fold cross-validated paired t* tanto para taxa de infidelidade, número de regras induzidas e número médio de condição por regra induzida. Nessa tabela, o símbolo $\triangle$ indica que um sistema de AM simbólico (C4.5, C4.5rules ou CN2) supera o algoritmo TREPAN, enquanto o símbolo $\blacktriangle$ indica que o algoritmo de aprendizado simbólico supera o TREPAN com 95% de confiabilidade. Por outro lado, os símbolos $\triangledown$ e $\blacktriangledown$ indicam que o TREPAN supera os sistemas de AM simbólico sem e com 95% de confiabilidade, respectivamente. A ausência de símbolo significa que não há diferença entre os sistemas de AM simbólico e o TREPAN.

Na Tabela 5.11 na próxima página são mostrados a quantidade de cada símbolo $\triangle$, $\blacktriangle$, $\triangledown$ e $\blacktriangledown$ presentes na Tabela 5.10.

Em relação à taxa de infidelidade, pode ser observado que o método TREPAN consegue 4 resultados significativos, com 95% de confiabilidade, e os sistemas de AM simbólico 2 resultados significativos. Um fato importante é que quando o método TREPAN é executado com seus parâmetros *default*, opção mofn1000, ele consegue apenas 2 resultados significativos. Com a opção mofn1000, o método TREPAN gera exemplos artificiais e utiliza expressões *m-de-n*. Deve ser observado que, segundo os autores do TREPAN, o uso de exemplos artificiais é uma das características positivas do método. Assim, a princípio, era de se esperar que o TREPAN, com a opção mofn1000, obtivesse resultados melhores comparados aos obtidos pelos sistemas de AM simbólico, já que os sistemas de AM simbólico não utilizam nenhum artifício na indução dos classificadores. Entretanto, para os conjuntos de dados utilizados nos experimentos, com exceção do conjunto `heart`, não há

	Taxa de Infidelidade											
	simple0			simple1000			mofn0			mofn1000		
	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2
breast	△	△	△	△	△	▲	▽	△	△	△	△	△
crx	△	△	▼	△	△	▽	△	△	▼	△	△	▽
heart		△	△	▽	△	▽	△	△	▽	▼	▽	▼
pima	△	▲	△	△	△	▽	△	△	△	▽	▽	▽
sonar	△	△	△	▽	▽	▽	△	△	△	△	△	△
votes	△	△	△	▽	▽	▽	▽	△	△	▽	▽	△
	Número de Regras Induzidas											
	simple0			simple1000			mofn0			mofn1000		
	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2
breast	△	△	▼	▲	▲	▽	▽	▽	▼	▼	▼	▼
crx	▲	▲	▲	▲	▲	▲	▽	△	▼	▽	△	▼
heart	△	▲	△	△	▲	△	▼	▽	▼	▼	▼	▼
pima	△	▲	△	△	▲	▲	▽	▲	▽	△	▲	△
sonar	▼	▲	▼	▼	△	▼	▼	▽	▼	▼	▼	▼
votes	▲	▲	△	△	▲	▽	▽	△	▼	▽	△	▼
	Número Médio de Condição por Regra Induzida											
	simple0			simple1000			mofn0			mofn1000		
	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2	C4.5	C4.5rules	CN2
breast	▽	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
crx	△	▲	▽	▽	△	▽	▲	▲	▲	▲	▲	▲
heart	▽	▲	△	△	▲	▲	▲	▲	▲	▲	▲	▲
pima	▽	▲	▲	△	▲	▲	▲	▲	▲	▲	▲	▲
sonar	▼	▲	▲	▽	▲	▲	▲	▲	▲	▲	▲	▲
votes	△	▲	△	▽	△	▽	▲	▲	▲	▲	▲	▲

Tabela 5.10: Resultados do teste de hipótese *10-Fold Cross-Validated Paired t*.

Taxa Infidelidade				Número de Regras				Condições por Regra			
△	▲	▽	▼	△	▲	▽	▼	△	▲	▽	▼
45	2	20	4	18	19	12	23	8	54	9	1

Tabela 5.11: Número de cada símbolo da Tabela 5.10.

resultados com diferenças significativas entre os sistemas de AM simbólico e o método TREPAN.

Quanto ao número de regras induzidas, nota-se que o método TREPAN obteve resultados significativos com as opções mofn0 e mofn1000. Isso se deve à utilização de expressões *m-de-n* na árvore de decisão, em que o número de regras tende a ser menor do que quando se utilizam testes simples. Contudo, os sistemas de AM simbólico conseguiram 2 resultados significativos comparados às opções mofn0 e mofn1000, para o conjunto de dados pima. Quando comparados às opções simple0 e simple1000, os sistemas de AM simbólico obtiveram 17 resultados significativos contra 5 do TREPAN.

Em relação ao número médio de condição por regra induzida, os sistemas de AM simbólico obtiveram resultados significativos para todos os conjuntos de dados quando comparados às opções mofn0 e mofn1000. Esse resultado era esperado, pois quando se

utiliza expressões *m-de-n*, o número de regras tende a ser menor, mas o número de condição por regra tende a ser maior do que quando se utiliza testes simples. Quando comparados às opções simple0 e simple1000, os sistemas de AM simbólico obtiveram 18 resultados significativos contra 1 do TREPAN.

Finalmente, considerando os resultados obtidos globais apresentados na Tabela 5.11 na página precedente, pode-se concluir que o comportamento dos sistemas de AM simbólico é semelhante ao do TREPAN. Na realidade, há uma tendência dos sistemas de AM simbólico superarem o TREPAN. É bom lembrar que os algoritmos simbólicos foram executados utilizando somente os valores *default* para seus parâmetros. Entretanto, espera-se que um ajuste adequado desses parâmetros melhore a taxa de infidelidade dos mesmos.

5.3.3 Etapa 3 - Método Baseado em Sistemas de AM Simbólico e AGs

O objetivo da Etapa 3 é avaliar o método proposto baseado em sistemas de AM simbólico e AGs. Nessa etapa, os classificadores induzidos na Etapa 2 pelos sistemas de AM simbólico C4.5, CN2 e C4.5rules com os conjuntos de treinamento rotulados pelas RNAs são utilizados para formar os indivíduos do AG. O indivíduo, ou seja, o conjunto de regras vencedor, é utilizado para explicar a RNA.

A seguir os passos da Etapa 3 são detalhadamente descritos, bem como os resultados obtidos.

5.3.3.1 Descrição - Etapa 3

Os passos da Etapa 3 foram realizados da seguinte forma:

1. Inicialmente, os classificadores induzidos pelos sistemas de AM simbólico C4.5, C4.5rules e CN2 foram convertidos para o formato padrão de regras $\mathcal{P}BM$ e a matriz de contingência para cada regra foi calculada.

2. Para cada um dos seis conjuntos de dados, o AG foi executado diversas vezes variando os valores de alguns de seus parâmetros, bem como variando a abordagem utilizada para classificar um exemplo dado um conjunto de regras. A função de aptidão utilizada maximiza a taxa de fidelidade para cada um dos indivíduos, cada indivíduo sendo formado por um conjunto de regras. Após a execução do AG, foi realizado

um pós-processamento no indivíduo vencedor. O objetivo do pós-processamento é remover as regras do indivíduo vencedor que não são disparadas. As abordagens utilizadas para classificar um exemplo dado um conjunto de regras foram *SingleRule* e *MultipleRules*, descritas na Seção 4.3.2 na página 77. Os valores para os parâmetros:

- n_g — número de gerações;
- n_i — número de indivíduos;
- t_i — tamanho inicial do indivíduo, ou seja, o número de regras de cada indivíduo;
- p_c — probabilidade de *crossover*;
- p_m — probabilidade de mutação.

utilizados com as abordagens *SingleRule* e *MultipleRules* para todos os conjuntos de dados são mostrados na Tabela 5.12.

parâmetros	valores
n_g	20
n_i	20
t_i	10
p_c	0.25
p_m	0.01

Tabela 5.12: Valores dos parâmetros do AG.

Foram também utilizados os valores $n_g = 40$, $n_i = 15$ e $p_c = 0.4$ com a abordagem *SingleRule*.

3. Por fim, a taxa de infidelidade e a complexidade sintática foram medidas. Na Seção A.3 na página 127 — Apêndice A, é mostrado um exemplo de relatório gerado pelo ambiente *NNRules* com informações sobre a taxa de infidelidade e complexidade sintática medidas para o AG.

Na Figura 5.4 na página seguinte é mostrada uma visão geral da Etapa 3.

5.3.3.2 Resultados Obtidos

Nesta seção são apresentados os resultados da Etapa 3 dos experimentos realizados, cujos passos foram descritos previamente.

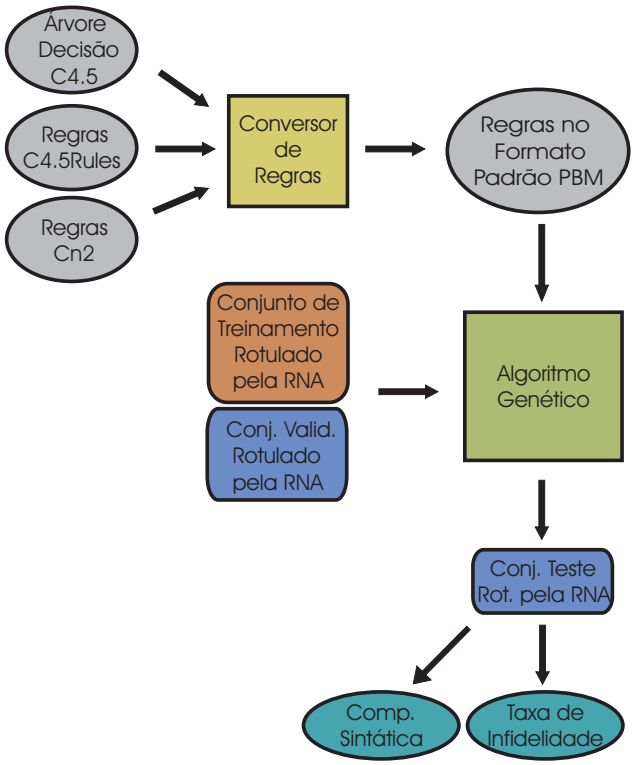

Figura 5.4: Visão geral da Etapa 3 do experimento realizado.

A taxa de infidelidade (média e desvio padrão) para os seis conjuntos de dados obtidas pelos sistemas de AM simbólico, TREPAN e AG são mostradas na Tabela 5.13 na próxima página. O número de regras induzidas é mostrado na Tabela 5.14 e o número médio de condição por regra induzida na Tabela 5.15 na página 112.

Os resultados da taxa de infidelidade, número de regras e número médio de condição por regra induzida, para os sistemas de AM simbólico e o método TREPAN, são os mesmos apresentados na Etapa 2, mas eles são apresentados novamente nesta seção para melhor visualização e comparação dos resultados. Nas tabelas, `SingleRule` significa o AG executado com a abordagem *SingleRule* e `SingleRulePP` refere-se à mesma abordagem, mas após o pós-processamento; `MultipleRules` significa a abordagem *MultipleRules* e `MultipleRulesPP` refere-se à mesma abordagem após o pós-processamento. A nona e a décima quarta linhas da tabela indicam os valores dos parâmetros n_g, n_i, t_i, p_c e p_m utilizados.

Para facilitar a descrição dos resultados, quando no texto houver uma referência, por exemplo, ao método baseado em sistemas de AM simbólico e AGs, sendo o AG executado com a abordagem *SingleRule* e com os valores dos parâmetros $n_g = 20$, $n_i = 20$, $t_i = 10$,

$p_c = 0.25$ e $p_m = 0.01$, esse método será referenciado como método `SingleRule`, enquanto que com os valores dos parâmetros $n_g = 40$, $n_i = 15$, $t_i = 10$, $p_c = 0.4$ e $p_m = 0.01$, será referenciando como método `*SingleRule`.

Nas tabelas, os menores valores, ou seja, os melhores resultados, aparecem em negrito. O símbolo ↑ indica um resultado significativo com 95% de confiabilidade, segundo o teste de hipótese *10-fold cross-validated paired t*, quando o método com melhor resultado é comparado aos demais métodos.

	breast	crx	heart	pima	sonar	votes
C4.5	↑ 2.93 ± 0.53	**2.91 ± 0.81**	↑ 14.07 ± 1.54	8.85 ± 1.07	22.98 ± 3.21	3.21 ± 0.97
C4.5rules	2.48 ± 0.53	3.06 ± 0.72	10.00 ± 1.11	8.46 ± 1.01	23.43 ± 3.25	2.98 ± 1.07
CN2	1.90 ± 0.62	↑ 5.82 ± 0.75	↑ 13.70 ± 1.99	9.12 ± 0.94	22.57 ± 3.35	2.75 ± 0.95
simple0	↑ 3.37 ± 0.44	3.36 ± 0.71	↑ 14.07 ± 1.81	↑ 10.81 ± 0.65	24.90 ± 3.06	3.67 ± 1.03
simple1000	↑ 4.39 ± 0.95	5.37 ± 1.85	12.22 ± 2.53	8.86 ± 1.15	22.14 ± 2.91	**2.30 ± 0.96**
mofn0	2.78 ± 0.63	3.83 ± 0.84	12.22 ± 1.75	10.42 ± 1.31	25.57 ± 3.15	2.99 ± 0.97
mofn1000	3.07 ± 0.80	4.75 ± 1.49	**7.04 ± 1.40**	**7.95 ± 1.01**	↑ 27.36 ± 2.73	2.76 ± 0.75
	AG: $n_g = 20$, $n_i = 20$, $t_i = 10$, $p_c = 0.25$, $p_m = 0.01$					
`SingleRule`	**1.61 ± 0.55**	3.52 ± 0.65	12.22 ± 1.57	8.85 ± 1.00	22.05 ± 3.24	2.76 ± 0.66
`SingleRulePP`	2.20 ± 0.70	4.44 ± 0.66	12.59 ± 1.58	10.42 ± 1.06	↑ 24.93 ± 3.36	3.91 ± 1.07
`MultipleRules`	2.04 ± 0.58	3.22 ± 0.67	13.33 ± 1.48	8.20 ± 0.97	↑ 25.00 ± 2.11	2.99 ± 0.83
`MultipleRulesPP`	2.04 ± 0.58	3.22 ± 0.67	13.33 ± 1.48	8.20 ± 0.97	↑ 25.00 ± 2.11	2.99 ± 0.83
	AG: $n_g = 40$, $n_i = 15$, $t_i = 10$, $p_c = 0.4$, $p_m = 0.01$					
`*SingleRule`	1.75 ± 0.64	3.68 ± 0.66	8.89 ± 0.99	8.60 ± 1.10	23.05 ± 2.74	2.98 ± 1.32
`*SingleRulePP`	2.63 ± 0.71	4.90 ± 0.60	11.48 ± 2.24	8.34 ± 1.21	**21.07 ± 2.24**	4.15 ± 1.37

Tabela 5.13: Taxa de infidelidade (média e desvio padrão) — Etapa 3.

	breast	crx	heart	pima	sonar	votes
C4.5	↑ 8.90 ± 0.91	↑ 14.10 ± 3.44	↑ 17.40 ± 1.33	↑ 25.00 ± 0.92	↑ 13.50 ± 0.34	↑ 9.40 ± 1.07
C4.5rules	↑ 8.00 ± 0.54	↑ 8.70 ± 1.58	↑ 11.40 ± 0.82	↑ 18.30 ± 1.16	↑ 8.10 ± 0.38	5.80 ± 0.44
CN2	↑ 12.10 ± 0.55	↑ 13.50 ± 1.42	↑ 16.00 ± 0.75	↑ 24.40 ± 0.97	↑ 25.20 ± 0.53	↑ 13.30 ± 0.67
simple0	↑ 9.40 ± 0.48	↑ 31.50 ± 4.81	↑ 18.10 ± 0.99	↑ 26.10 ± 1.48	↑ 12.40 ± 0.60	↑ 13.60 ± 0.90
simple1000	↑ 10.60 ± 0.73	↑ 31.00 ± 8.03	↑ 20.00 ± 1.92	↑ 29.30 ± 2.01	↑ 9.50 ± 1.10	↑ 11.00 ± 1.61
mofn0	↑ 7.40 ± 0.78	↑ 11.10 ± 1.76	↑ 9.20 ± 0.83	↑ 23.30 ± 1.38	↑ 6.90 ± 0.53	↑ 7.10 ± 0.53
mofn1000	**2.40 ± 0.31**	↑ 9.90 ± 1.94	**5.30 ± 0.83**	↑ 26.00 ± 1.55	**4.10 ± 0.67**	6.50 ± 0.72
	AG: $n_g = 20$, $n_i = 20$, $t_i = 10$, $p_c = 0.25$, $p_m = 0.01$					
`SingleRule`	↑ 14.60 ± 1.12	↑ 16.20 ± 3.15	↑ 21.50 ± 2.26	↑ 18.40 ± 1.97	↑ 20.80 ± 3.86	↑ 15.00 ± 1.62
`SingleRulePP`	↑ 7.00 ± 0.47	**4.50 ± 0.96**	7.90 ± 0.74	**9.60 ± 1.37**	↑ 7.60 ± 1.24	**5.30 ± 0.67**
`MultipleRules`	↑ 14.50 ± 1.92	↑ 19.80 ± 4.29	↑ 18.40 ± 3.06	↑ 30.20 ± 3.39	↑ 22.60 ± 2.91	↑ 14.00 ± 1.29
`MultipleRulesPP`	↑ 14.50 ± 1.92	↑ 18.80 ± 4.08	↑ 18.10 ± 2.98	↑ 30.20 ± 3.39	↑ 21.60 ± 2.71	↑ 12.90 ± 1.22
	AG: $n_g = 40$, $n_i = 15$, $t_i = 10$, $p_c = 0.4$, $p_m = 0.01$					
`*SingleRule`	↑ 17.60 ± 1.33	↑ 20.30 ± 5.08	↑ 24.30 ± 2.03	↑ 42.00 ± 3.71	↑ 31.60 ± 2.09	↑ 14.40 ± 0.95
`*SingleRulePP`	↑ 7.60 ± 0.56	↑ 5.70 ± 1.07	↑ 10.80 ± 0.87	↑ 19.60 ± 1.65	↑ 11.80 ± 1.45	6.10 ± 0.64

Tabela 5.14: Número de regras induzidas (média e desvio padrão) — Etapa 3.

Na Tabela 5.16 na página 113 os métodos são mostrados em ordem crescente pelo valor da taxa de infidelidade. Na Tabela 5.17 na página 114 eles são mostrados em ordem crescente pelo valor do número médio de regras induzidas, e na Tabela 5.18 na página 114 em ordem crescente pelo número médio de condição por regra induzida. Os números que

	breast	crx	heart	pima	sonar	votes
C4.5	↑ 3.63 ± 0.17	2.62 ± 0.46	↑ 3.25 ± 0.06	↑ 5.66 ± 0.11	↑ 4.27 ± 0.11	2.68 ± 0.20
C4.5rules	2.57 ± 0.05	**2.17 ± 0.10**	**2.41 ± 0.04**	**2.92 ± 0.05**	↑ 2.84 ± 0.13	2.52 ± 0.11
CN2	2.56 ± 0.07	↑ 2.84 ± 0.08	↑ 2.92 ± 0.10	3.02 ± 0.05	**2.03 ± 0.02**	2.64 ± 0.05
simple0	↑ 3.54 ± 0.11	↑ 2.78 ± 0.10	↑ 3.05 ± 0.11	↑ 5.58 ± 0.13	↑ 3.93 ± 0.13	2.83 ± 0.10
simple1000	↑ 4.00 ± 0.19	2.61 ± 0.27	↑ 3.27 ± 0.12	↑ 5.94 ± 0.17	↑ 3.81 ± 0.24	2.61 ± 0.18
mofn0	↑ 6.74 ± 0.54	↑ 7.62 ± 0.75	↑ 9.47 ± 0.59	↑ 10.23 ± 0.49	↑ 9.35 ± 0.49	↑ 5.46 ± 0.41
mofn1000	↑ 7.49 ± 0.48	↑ 10.10 ± 0.91	↑ 10.17 ± 0.41	↑ 10.73 ± 0.32	↑ 23.41 ± 1.51	↑ 9.04 ± 0.90
	AG: $n_g = 20$, $n_i = 20$, $t_i = 10$, $p_c = 0.25$, $p_m = 0.01$					
SingleRule	2.78 ± 0.11	↑ 2.97 ± 0.14	2.83 ± 0.08	3.73 ± 0.16	2.87 ± 0.09	2.55 ± 0.11
SingleRulePP	2.54 ± 0.08	2.25 ± 0.27	2.75 ± 0.12	2.96 ± 0.09	2.94 ± 0.14	**2.28 ± 0.16**
MultipleRules	↑ 2.79 ± 0.10	↑ 3.00 ± 0.16	2.89 ± 0.11	↑ 3.79 ± 0.08	2.91 ± 0.07	2.54 ± 0.11
MultipleRulesPP	↑ 2.79 ± 0.10	↑ 2.96 ± 0.16	2.88 ± 0.11	↑ 3.79 ± 0.08	2.95 ± 0.08	2.47 ± 0.09
	AG: $n_g = 40$, $n_i = 15$, $t_i = 10$, $p_c = 0.4$, $p_m = 0.01$					
∗SingleRule	2.65 ± 0.13	↑ 2.86 ± 0.15	2.78 ± 0.10	↑ 3.83 ± 0.07	2.94 ± 0.11	2.59 ± 0.12
∗SingleRulePP	**2.46 ± 0.08**	2.41 ± 0.20	2.75 ± 0.08	3.14 ± 0.09	2.91 ± 0.05	2.31 ± 0.08

Tabela 5.15: Número médio de condição por regra induzida (média e desvio padrão) — Etapa 3.

aparecem à direita do nome do método significam quantos resultados significativos, com 95% de confiabilidade, o método obteve quando comparado aos demais métodos.

Pode ser observado que, para o conjunto de dados `breast`, apesar do método `SingleRule` ter obtido melhor resultado para a taxa de infidelidade com 3 resultados significativos quando comparado aos demais métodos, ele não obteve bons resultados para o número de regras induzidas e para o número médio de condição por regra induzida. O método `SingleRulePP`, apesar de ter obtido 1 resultado significativo em relação à taxa de infidelidade, obteve 8 resultados significativos quanto ao número de regras induzidas e 7 em relação ao número médio de condição por regra induzida.

Para o conjunto de dados `crx`, o método C4.5rules obteve bons resultados para o número de regras induzidas e para o o número médio de condição por regra induzida. Para a taxa de infidelidade, esse método conseguiu apenas um resultado significativo, mas os outros métodos também não obtiveram muitos resultados significativos.

Para o conjunto de dados `heart`, os métodos ∗`SingleRule` e mofn1000 obtiveram os melhores resultados para a taxa de infidelidade. Entretanto, para o número de regras induzidas, o método ∗`SingleRule` não obteve um bom resultado, e o método mofn1000 não obteve bom resultado para o número médio de condição por regra induzida.

Pode ser observado que, para o conjunto de dados `pima`, os métodos ∗`SingleRulePP` e C4.5rules obtiveram bons resultados em relação à complexidade sintática. Quanto à taxa de infidelidade, o número máximo de resultados significativos conseguidos pelos métodos foi 1, não se destacando nenhum método.

Para o conjunto de dados sonar, o método ∗SingleRulePP obteve muito bons resultados para a taxa de infidelidade e o número de regras induzidas.

Para o conjunto de dados votes, nenhum método apresentou resultados significativos para a taxa de infidelidade. Entretanto, os métodos SingleRulePP, ∗SingleRulePP e C4.5rules apresentaram bons resultados para a complexidade sintática.

Finalmente, considerando de uma forma global os resultados obtidos, pode-se concluir que a idéia de utilizar Algorimos Genéticos parece promissora e merece ser melhor explorada. Um dos aspectos a ser investigado refere-se ao bom comportamento da abordagem *SingleRule* comparada à abordagem *MultipleRules*. Ou seja, aparentemente, o AG contrói melhores classificadores considerando isoladamente a melhor regra para classificar um novo exemplo, em vez de considerar todas as regras que cobrem o exemplo e decidir a classificação do mesmo em função da qualidade global dessas regras.

Outro aspecto a ser investigado refere-se à população inicial de indivíduos. Neste trabalho, optou-se por gerar esses indivíduos escolhendo de forma aleatória as regras induzidas por todos os classificadores. Em outras palavras, os indivíduos da população inicial não são, necessariamente, "bons" classificadores. Dessa forma, há uma maior possibilidade de atingir um máximo local. Deve ser ressaltado que neste trabalho consideramos essa escolha aleatória de regras para construir os indivíduos iniciais, com o objetivo de verificar o potencial da idéia proposta, mas sem favorecer o uso de AGs. Pretende-se, futuramente, pesquisar o comportamento de AGs inicializando cada indivíduo da população como "bons" classificadores para observar melhor o comportamento dos AGs.

	breast	crx	heart	pima	sonar	votes
1	SingleRule[3]	C4.5[1]	mofn1000[3]	mofn1000[1]	∗SingleRulePP[4]	simple1000
2	∗SingleRule[3]	C4.5rules[1]	∗SingleRule[6]	MultipleRules[1]	SingleRule[1]	CN2
3	CN2[1]	MultipleRules[2]	C4.5rules[1]	MultipleRulesPP[1]	simple1000	SingleRule
4	MultipleRules[2]	MultipleRulesPP[2]	∗SingleRulePP	∗SingleRulePP	CN2	mofn1000
5	MultipleRulesPP[2]	simple0[1]	SingleRule[1]	C4.5rules[1]	C4.5	C4.5rules
6	SingleRulePP[1]	SingleRule	mofn0	∗SingleRule	∗SingleRule	∗SingleRule
7	C4.5rules	∗SingleRule	simple1000	SingleRule	C4.5rules	MultipleRulesPP
8	∗SingleRulePP[1]	mofn0[1]	SingleRulePP	C4.5	simple0	MultipleRules
9	mofn0	SingleRulePP	MultipleRules	simple1000	SingleRulePP	mofn0
10	C4.5	mofn1000	MultipleRulesPP	CN2	MultipleRulesPP	C4.5
11	mofn1000	∗SingleRulePP	CN2	SingleRulePP	MultipleRules	CN2
12	simple0	simple1000	C4.5	mofn0	mofn0	SingleRulePP
13	simple1000	CN2	simple0	simple0	mofn1000	∗SingleRulePP

Tabela 5.16: Métodos ordenados pela taxa de infidelidade.

	breast	crx	heart	pima	sonar	votes
1	mofn1000[12]	SingleRulePP[11]	mofn1000[11]	SingleRulePP[12]	mofn1000[12]	SingleRulePP[9]
2	SingleRulePP[8]	∗SingleRulePP[10]	SingleRulePP[10]	C4.5rules[7]	mofn0[8]	C4.5rules[8]
3	mofn0[5]	C4.5rules[3]	mofn0[8]	SingleRule[6]	SingleRulePP[7]	∗SingleRulePP[8]
4	∗SingleRulePP[7]	mofn1000[3]	∗SingleRulePP[7]	∗SingleRulePP[8]	C4.5rules[7]	mofn1000[6]
5	C4.5rules[4]	mofn0[3]	C4.5rules[6]	mofn0[2]	simple1000[6]	mofn0[6]
6	C4.5[4]	CN2[2]	CN2	CN2[2]	∗SingleRulePP[4]	C4.5[4]
7	simple0[3]	C4.5[2]	C4.5	C4.5	simple0[5]	simple1000
8	simple1000[1]	SingleRule[1]	simple0	mofn1000[1]	C4.5[3]	MultipleRulesPP[2]
9	CN2	MultipleRulesPP[1]	MultipleRulesPP	simple0[1]	SingleRule[1]	CN2
10	MultipleRules	MultipleRules[1]	MultipleRules	simple1000	MultipleRulesPP[2]	simple0
11	MultipleRulesPP	∗SingleRule[1]	simple1000	MultipleRules	MultipleRules	MultipleRules
12	SingleRule	simple1000	SingleRule	MultipleRulesPP	CN2	∗SingleRule
13	∗SingleRule	simple0	∗SingleRule	∗SingleRule	∗SingleRule	SingleRule

Tabela 5.17: Métodos ordenados pelo número de regras induzidas.

	breast	crx	heart	pima	sonar	votes
1	∗SingleRulePP[7]	C4.5rules[8]	C4.5rules[6]	C4.5rules[8]	CN2[6]	SingleRulePP[2]
2	SingleRulePP[7]	SingleRulePP[2]	∗SingleRulePP[2]	SingleRulePP[5]	C4.5rules[5]	∗SingleRulePP[2]
3	CN2[4]	∗SingleRulePP[2]	SingleRulePP[2]	CN2[8]	SingleRule[1]	MultipleRulesPP[3]
4	C4.5rules[5]	simple1000[2]	∗SingleRule[2]	∗SingleRulePP[9]	MultipleRules[2]	C4.5rules[3]
5	∗SingleRule[7]	C4.5[2]	SingleRule[2]	SingleRule[5]	∗SingleRulePP[1]	MultipleRules[2]
6	SingleRule[5]	simple0[2]	MultipleRulesPP[2]	MultipleRules[5]	∗SingleRule[1]	SingleRule[3]
7	MultipleRules[5]	CN2[2]	MultipleRules[2]	MultipleRulesPP[5]	SingleRulePP[1]	∗SingleRule[2]
8	MultipleRulesPP[5]	∗SingleRule[2]	CN2[4]	∗SingleRule[5]	MultipleRulesPP[1]	simple1000[2]
9	simple0[3]	MultipleRulesPP[2]	simple0[2]	simple0[3]	simple1000[2]	simple0[2]
10	C4.5[3]	SingleRule[2]	C4.5[2]	C4.5[2]	simple0[3]	CN2[2]
11	simple1000[2]	MultipleRules[2]	simple1000[2]	simple1000[2]	C4.5[2]	C4.5[2]
12	mofn0	mofn0[1]	mofn0	mofn0	mofn0[1]	mofn0[1]
13	mofn1000	mofn1000	mofn1000	mofn1000	mofn1000	mofn1000

Tabela 5.18: Métodos ordenados pelo número médio de condição por regra induzida.

5.4 Considerações Finais

Neste capítulo foram descritos os experimentos realizados, utilizando seis conjuntos de dados, para avaliar os dois métodos propostos neste trabalho para extração de conhecimento de RNAs.

O método baseado em sistemas de AM simbólico foi testado utilizando os sistemas C4.5, C4.5rules e CN2. Os resultados obtidos pelo método foram semelhantes aos resultados obtidos pelo método TREPAN, no entanto, há uma tendência do método proposto superar o TREPAN em relação à taxa de infidelidade.

Quanto ao método baseado em sistemas de AM simbólico e AG, os resultados obtidos mostram que utilizar AGs para combinar o conhecimento de vários classificadores para extrair conhecimento de RNAs parece promissor e merece ser melhor explorado.

Capítulo 6

Conclusão

6.1 Considerações Iniciais

Neste capítulo são apresentadas as conclusões deste trabalho. Na Seção 6.2 são apresentadas as principais contribuições desta tese; na Seção 6.3 são discutidas algumas limitações dos métodos propostos para extração de conhecimento simbólico de RNAs; na Seção 6.4 são propostas algumas sugestões para trabalhos futuros. Finalmente, na Seção 6.5 são apresentadas as considerações finais do trabalho.

6.2 Principais Contribuições

RNAs são um dos métodos mais utilizados em aprendizado indutivo. Esses sistemas têm sido utilizados em uma ampla variedade de domínios demonstrando, freqüentemente, uma precisão de classificação superior a de outros métodos propostos pela comunidade de AM. RNAs são métodos muito poderosos, uma vez que o seu *bias* indutivo permite aprender qualquer mapeamento entre as unidades de entrada e as unidades de saída.

Por outro lado, as RNAs possuem uma limitação importante: as hipóteses induzidas por esses métodos são geralmente incompreensíveis para o usuário. Com o objetivo de solucionar essa limitação, diversos pesquisadores têm desenvolvido métodos para extrair conhecimento simbólico de RNAs.

Nesta tese são propostos dois métodos para extração de conhecimento simbólico de RNAs na forma de regras de decisão: um método baseado em sistemas de AM simbólico e um segundo método baseado em sistemas de AM simbólico e AGs. Também, é pro-

posto, projetado e implementado um ambiente computacional, chamado *NNRules*, que implementa os dois métodos de extração de conhecimento propostos.

Nas próximas seções são discutidas as principais contribuições dos métodos propostos nesta tese, bem como do ambiente computacional *NNRules*.

6.2.1 Extração de Conhecimento de RNAs com Sistemas de AM Simbólico

Nesse método, a extração de conhecimento de uma RNA é vista como um processo de indução. A RNA é utilizada para rotular um conjunto de dados e os sistemas de AM simbólico são utilizados para induzir uma hipótese **h'** que aproxima a hipótese **h** induzida pela RNA.

O principal mérito desse método é a sua simplicidade. Nos experimentos realizados, foram utilizados sistemas de AM simbólico amplamente difundidos na comunidade de AM como o CN2, o C4.5 e o C4.5rules. Apesar do método proposto ser bastante simples, os resultados obtidos são bastante competitivos se comparados com os resultados obtidos pelo método TREPAN.

Abordagens que utilizam métodos indutivos para a extração de conhecimento de RNAs são bastante promissoras pois podem ser aplicadas para explicar uma ampla gama de classificadores não simbólicos. Assim como as RNAs, outros sistemas de aprendizado carecem de um mecanismo de explicação para o usuário. Pode-se citar como exemplo os classificadores baseados em *Support Vector Machines* — SVM (Vapnik 1995; Vapnik 1998). Um outro exemplo, são os *ensembles* (Shapire 1990; Wolpert 1992). *Ensembles* consistem em conjuntos de classificadores individuais cujas predições são combinadas para determinar a classificação de um novo exemplo. As predições podem ser combinadas de diferentes formas como, por exemplo, por meio de uma votação realizada entre classificadores. Freqüentemente, um *ensemble* é mais preciso do que qualquer uma das hipóteses individuais contidas nele, entretanto, mesmo se os classificadores individuais forem simbólicos, a compreensibilidade do classificador final fica muito prejudicada, uma vez que a classificação final depende da predição individual realizada por cada classificador.

É também possível estender o método proposto para extração de conhecimento de RNAs para problemas de regressão e de aprendizado não supervisionado. No caso de problemas de regressão, é necessário utilizar algoritmos de AM simbólico que induzem árvores e regras de regressão. Nesse caso, a forma de avaliar a fidelidade da explicação simbólica

à RNA também precisa ser modificada. Uma possibilidade é verificar as diferenças entre os valores fornecidos pela rede e pelo regressor simbólico por meio da distância média absoluta ou do erro médio quadrático.

No caso de explicar uma RNA não supervisionada, o procedimento é similar ao utilizado nos experimentos deste trabalho. A RNA não supervisionada, após a fase de treinamento, pode classificar os exemplos segundos os *clusters* gerados. Por fim, o método proposto pode gerar um classificador simbólico que explica os *clusters*.

6.2.2 Extração de Conhecimento de RNAs com Sistemas de AM Simbólico e AGs

Nesse método, a utilização de AGs permite integrar o conhecimento extraído por diversos sistemas de AM simbólico em um único conjunto de regras que possua alta fidelidade com a RNA a ser explicada. Essa tarefa não é trivial, pois para que o conjunto de regras possua alta fidelidade com a RNA, é necessário que as regras do conjunto de regras se complementem.

A aplicação de AGs para integrar conhecimento pode também ser utilizada para outros propósitos, além da extração de conhecimento de RNAs. Por exemplo, AGs podem ser utilizados para gerar um classificador simbólico a partir de diversos classificadores também simbólicos. Como proposto por Fayyad, Piatetsky-Shapiro & Smyth (1996), uma abordagem para realizar descoberta de conhecimento em grandes bases de dados é, dada uma grande base de dados, gerar diversas amostras de tamanho reduzido. As amostras são analisadas por um sistema de AM simbólico e o conhecimento obtido em cada uma das amostras é integrado em um único classificador. Os AGs podem ser utilizados para esse fim, ou seja, para integrar o conhecimento de diversos classificadores simbólicos criados a partir de diferentes amostras em um classificador final também simbólico.

6.2.3 O Ambiente Computacional *NNRules*

O ambiente *NNRules* possui um projeto modular. Assim, é possível realizar modificações no projeto original e avaliar novas idéias. Por exemplo, é possível integrar facilmente novos sistemas de AM simbólico. Também, é possível implementar novas estratégias para classificar um exemplo, por meio da criação de uma nova classe no módulo RuleSet.

A implementação do módulo GA permite que o usuário experimente diversos pa-

râmetros, como o número e tamanho dos indivíduos, taxa de *crossover* e mutação, bem como novos critérios de parada, novas funções de aptidão e diferentes tipos de operadores de *crossover* e mutação.

Na realidade, o projeto do ambiente *NNRules* permite que esse ambiente seja utilizado para outros propósitos, além da extração de conhecimento de RNAs. Por exemplo, os módulos do ambiente *NNRules* podem ser utilizados para gerar um classificador simbólico a partir de diversos classificadores também simbólicos, como descrito anteriormente.

6.3 Limitações

Nesta seção são discutidas as limitações dos métodos de extração de conhecimento de RNAs. Inicialmente, é realizada uma discussão a respeito das limitações causadas pela diferença entre os *bias* indutivos das RNAs e das linguagens de representação de conhecimento simbólico. Logo em seguida são discutidas as limitações dos métodos propostos neste trabalho.

6.3.1 Limitações Causadas por Diferenças entre os Bias Indutivos

Os métodos de extração de conhecimento de RNAs possuem uma limitação intrínseca: a diferença dos *bias* indutivos. As RNAs possuem um *bias* indutivo flexível que permite aprender qualquer mapeamento entre as unidades de entrada e de saída. Por outro lado, o conhecimento simbólico, em especial, as regras de decisão, dividem o espaço de exemplos com hiperplanos paralelos aos eixos.

Como conseqüência dessa diferença de *bias* indutivo dos dois métodos, nem todo conceito aprendido por uma RNA pode ser descrito por um conjunto de regras. Em outras palavras, para uma dada RNA treinada, pode não ser possível encontrar um conjunto de regras que seja estritamente fiel à RNA.

Um exemplo simples dessa situação é quando a RNA separa duas classes por uma reta não paralela aos eixos como, por exemplo, a reta $y = x$. Nesse caso, um conjunto de regras pode aproximar essa reta por uma borda de decisão na forma de "escada". Apesar de que, para um determinado conjunto de dados ser possível obter 0% de taxa de infidelidade, não é possível garantir que, para todos os possíveis exemplos, as duas representações irão sempre concordar nas suas classificações.

Pode-se argumentar que é possível gerar uma borda de decisão na forma de "escada" bastante próxima da reta diagonal, de forma que as duas representações sejam virtualmente as mesmas, o número de regras para gerar tal borda de decisão seria tão grande que a complexidade do conhecimento gerado não seria muito diferente de tentar compreender os próprios pesos da RNA.

Nesse sentido, o conhecimento extraído de RNAs na forma de regras é normalmente uma aproximação do real conhecimento embutido nos pesos da rede. Apesar desse fato ser uma limitação, o uso de métodos de extração de regras não é invalidado por ele. Uma das principais preocupações dos usuários de RNAs é saber se a rede generalizou os dados de forma correta. A extração de regras de RNAs pode ajudar a validar a RNA, uma vez que o conhecimento simbólico pode ser inspecionado e regras inválidas podem ser identificadas. A partir das regras inválidas é possível questionar a RNA com novos exemplos que possuem as características fornecidas por tais regras e verificar se a rede classifica incorretamente esses exemplos.

Duas possíveis abordagens para extrair conhecimento simbólico totalmente fiel à uma RNA são: aumentar o poder de representação da linguagem de representação simbólica; ou restringir o poder de representação da RNA.

Diversos trabalhos visam aumentar o poder de representação das linguagens simbólicas. Por exemplo, pode-se citar a indução de árvores de decisão oblíquas. Nas árvores de decisão oblíquas, os testes dos nós internos são expressões lineares que envolvem os atributos. Uma outra possibilidade é a *indução construtiva*, na qual novos atributos são criados por meio da combinação dos atributos existentes. Um problema dessas soluções é que o conhecimento pode se tornar muito complexo. No caso da extração de conhecimento de RNAs, as expressões que compõem os atributos podem ser extraídas dos nós da RNA. Mas, normalmente, essas expressões são muito complexas e envolvem diversos atributos que nem sempre estão diretamente relacionados. Dessa forma, é difícil fornecer uma semântica para essas expressões.

A restrição do *bias* indutivo da RNA pode ser feito de diversas formas. Uma forma é alterar o treinamento da RNA para que a rede somente utilize retas paralelas aos eixos para separar as classes. Dessa forma, o poder de representação da RNA se torna similar ao poder de representação de uma arvore de decisão ou de um conjunto de regras de decisão. Entretanto, a RNA perde a característica que é o seu principal atrativo: a capacidade de separar classes com bordas de decisões complexas.

6.3.2 Limitações do Método Baseado em Sistemas de AM Simbólico

Tratar o problema de extração de conhecimento de RNAs como um problema de indução possui algumas limitações. Uma delas é o fato de que a indução depende de dados para ser realizada. Em alguns casos, conjuntos de exemplos de tamanho restrito podem não ser suficientes para caracterizar o conceito aprendido pela RNA, e a explicação apresentada pelo algoritmo simbólico pode não ser fiel à RNA. Aparentemente, o método TREPAN visa solucionar essa limitação por meio da criação de exemplos artificiais. Entretanto, essa abordagem deve ser aplicada com cuidado, pois a geração de exemplos artificiais pode criar exemplos improváveis, ou até mesmo impossíveis, por causa das restrições nas relações entre os atributos. Segundo experimentos realizados neste trabalho, a criação de exemplos artificiais no método TREPAN não necessariamente acarreta na criação de um classificador mais fiel à RNA.

6.3.3 Limitações do Método Baseado em Sistemas de AM Simbólico e AGs

Compor regras utilizando AGs é uma idéia promissora, mas ainda existe muito a ser explorado. Existem várias idéias que podem melhorar o desempenho desse método. Entre elas estão formas alternativas de gerar os indivíduos da primeira geração, e a forma em que o conjunto de regras é avaliado. Essas idéias são discutidas na próxima seção.

6.4 Trabalhos Futuros

Algumas idéias para trabalhos futuros são apresentadas a seguir.

Diversas idéias podem ser testadas nos métodos propostos baseados em sistemas de AM simbólico e AGs, entre elas:

- No método de extração de conhecimento de RNAs que utiliza sistemas de AM, pode-se utilizar outros sistemas de AM simbólico.
- Atualmente, os indivíduos da população inicial do AG são criados aleatoriamente. Pode-se tentar melhorar o tempo de convergência do AG se os indivíduos inicias

forem bons classificadores. Por exemplo, pode-se utilizar os classificadores induzidos pelos sistemas de AM simbólico como os indivíduos da população inicial.

- Com o objetivo de aumentar a diversidade das regras, outros sistemas de AM simbólico podem ser utilizados. Uma outra abordagem é modificar os parâmetros desses sistemas ou induzir classificadores sobre diversas amostras dos dados, com quantidade e diversidade maior de regras.

- Um aspecto importante de um conjunto de regras é que esse conjunto cubra uma parte expressiva do espaço de exemplos. Em outras palavras, é de pouco valor ter um grupo de regras altamente fiéis a uma RNA, se todas essas regras cobrem os mesmos exemplos. Se um conjunto de regras possui muitas regras redundantes, então é provável que grande parte dos exemplos sejam classificados pela regra *default*, a qual, nos nossos experimentos, classifica o exemplo como pertencente à classe majoritária. Uma forma de criar indivíduos com regras com coberturas complementares é introduzir essa informação na função de aptidão do AG.

6.5 Considerações Finais

Neste capítulo foram apresentadas as principais contribuições deste trabalho, bem como suas limitações e algumas sugestões para trabalhos futuros.

Como resultados da pesquisa em extração de conhecimento de RNAs realizada, foram publicados diversos trabalhos. Entre eles, um artigo publicado em periódico internacional (Milaré, Carvalho & Monard 2002), dois artigos publicados em congressos internacionais (Milaré, de Carvalho & Monard 2002a; Milaré, de Carvalho & Monard 2001), um artigo publicado em congresso latino-americano (Milaré & de Carvalho 2001a) e dois artigos publicados em congressos nacionais (Milaré, de Carvalho & Monard 2002b; Milaré & de Carvalho 2001b).

Apêndice A

Relatórios do Ambiente *NNRules*

A.1 Exemplo de Relatório: Método Baseado em Sistemas de AM Simbólico

```
Summary Symbolic Learning Algorithms            Wed Jan 15 04:56:41 2003
Created by Claudia Regina Milaré                                    v1.0
------------------------------------------------------------------------
Input Files:  heart/fidelity/c4.5/it0..9/heart.stdrules
              heart/fidelity/parameter/Detailed.txt
Inducer: c4.5

Fold    #Conditions(Sum)    #Rules    C/R(Mean)    Infidelity
 0        88                  24       3.67         11.11
 1        54                  17       3.18         14.81
 2        62                  19       3.26         11.11
 3        42                  14       3.00         11.11
 4        47                  15       3.13         18.52
 5        40                  12       3.33         18.52
 6        61                  19       3.21         18.52
 7        76                  23       3.30          3.70
 8        59                  19       3.11         18.52
 9        40                  12       3.33         14.81
Mean      56.90               17.40    3.25         14.07
SE        5.03                1.33     0.06         1.54

------------------------------------------------------------------------
Input Files:  heart/fidelity/c4.5rules/it0..9/heart.stdrules
```

```
                heart/fidelity/parameter/Detailed.txt
Inducer: c4.5rules

Fold    #Conditions(Sum)    #Rules    C/R(Mean)    Infidelity
 0        32                  12       2.67          7.41
 1        22                   9       2.44          7.41
 2        24                  10       2.40          7.41
 3        20                   8       2.50         11.11
 4        33                  14       2.36         11.11
 5        25                  11       2.27         14.81
 6        27                  11       2.45         11.11
 7        36                  16       2.25          3.70
 8        32                  14       2.29         11.11
 9        22                   9       2.44         14.81
Mean      27.30               11.40    2.41         10.00
SE        1.76                0.82     0.04         1.11

-------------------------------------------------------------------------
Input Files:  heart/fidelity/cn2/it0..9/heart.stdrules
              heart/fidelity/parameter/Detailed.txt
Inducer: cn2

Fold    #Conditions(Sum)    #Rules    C/R(Mean)    Infidelity
 0        59                  18       3.28         14.81
 1        47                  19       2.47          7.41
 2        43                  13       3.31         14.81
 3        53                  19       2.79         22.22
 4        42                  15       2.80         25.93
 5        46                  16       2.88         14.81
 6        37                  13       2.85         11.11
 7        45                  15       3.00         11.11
 8        45                  18       2.50          7.41
 9        46                  14       3.29          7.41
Mean      46.30               16.00    2.92         13.70
SE        1.90                0.75     0.10         1.99
```

A.2 Exemplo de Relatório: Método Trepan

```
Summary Trepan                                  Wed Jan 15 03:39:52 2003
Created by Claudia Regina Milaré                                    v1.0
------------------------------------------------------------------------
Input Files:  heart/trepan/simple_0/it0..9/heart.stdrules
              heart/trepan/simple_0/it0..9/trepan_out.txt
Trepan parameter: simple_0

Fold  #Conditions(Sum)  #Rules  C/R(Mean) Infidelity
 0       63               21      3.00     (5) 18.52
 1       60               19      3.16     (3) 11.11
 2       84               23      3.65     (7) 25.93
 3       39               14      2.79     (3) 11.11
 4       46               16      2.88     (4) 14.81
 5       45               16      2.81     (3) 11.11
 6       54               17      3.18     (5) 18.52
 7       57               20      2.85     (2)  7.41
 8       75               21      3.57     (4) 14.81
 9       37               14      2.64     (2)  7.41
Mean     56.00            18.10   3.05     14.07
SE        4.81             0.99   0.11      1.81

------------------------------------------------------------------------
Input Files:  heart/trepan/simple_1000/it0..9/heart.stdrules
              heart/trepan/simple_1000/it0..9/trepan_out.txt
Trepan parameter: simple_1000

Fold  #Conditions(Sum)  #Rules  C/R(Mean) Infidelity
 0       91               27      3.37     (6) 22.22
 1       28               10      2.80     (1)  3.70
 2       97               25      3.88     (2)  7.41
 3       74               19      3.89     (5) 18.52
 4       60               21      2.86     (1)  3.70
 5       80               25      3.20     (3) 11.11
 6       48               15      3.20     (7) 25.93
 7       69               23      3.00     (4) 14.81
 8       80               24      3.33     (3) 11.11
 9       35               11      3.18     (1)  3.70
Mean     66.20            20.00   3.27     12.22
SE        7.31             1.92   0.12      2.53

```

```
41  ------------------------------------------------------------------------
42  Input Files:  heart/trepan/m_of_n_0/it0..9/heart.stdrules
43                heart/trepan/m_of_n_0/it0..9/trepan_out.txt
44  Trepan parameter: m_of_n_0
45
46  Fold  #Conditions(Sum)  #Rules  C/R(Mean) Infidelity
47   0       94               7     13.43     (2)  7.41
48   1       83               9      9.22     (2)  7.41
49   2      152              15     10.13     (3) 11.11
50   3       85              11      7.73     (4) 14.81
51   4       75               8      9.38     (3) 11.11
52   5       91              11      8.27     (7) 25.93
53   6       65               7      9.29     (3) 11.11
54   7      105              10     10.50     (2)  7.41
55   8       72               7     10.29     (4) 14.81
56   9       45               7      6.43     (3) 11.11
57  Mean     86.70            9.20   9.47     12.22
58  SE        8.98            0.83   0.59      1.75
59
60  ------------------------------------------------------------------------
61  Input Files:  heart/trepan/m_of_n_1000/it0..9/heart.stdrules
62                heart/trepan/m_of_n_1000/it0..9/trepan_out.txt
63  Trepan parameter: m_of_n_1000
64
65  Fold  #Conditions(Sum)  #Rules  C/R(Mean) Infidelity
66   0       66               6     11.00     (2)  7.41
67   1       68               7      9.71     (2)  7.41
68   2       62               6     10.33     (0)  0.00
69   3       49               5      9.80     (1)  3.70
70   4       20               2     10.00     (3) 11.11
71   5       24               2     12.00     (1)  3.70
72   6       16               2      8.00     (4) 14.81
73   7       68               6     11.33     (2)  7.41
74   8      112              10     11.20     (1)  3.70
75   9       58               7      8.29     (3) 11.11
76  Mean     54.30            5.30  10.17      7.04
77  SE        9.13            0.83   0.41      1.40
```

A.3 Exemplo de Relatório: Método Baseado em Sistemas de AM Simbólico e AGs

```
Summary Genetic Algoritm                          Sun Mar  2 14:32:10 2003
Created by Claudia Regina Milaré                                      v1.0

------------------------------------------------------------------------
Input File:  heart/fidelity/ga3/it0..9/before_post

Fold    #Condition(Sum)    #Rules    C/R(Mean)    Infidelity
 0        93                 31        3.00       (1)  3.70
 1        90                 34        2.65       (2)  7.41
 2        67                 23        2.91       (2)  7.41
 3        70                 28        2.50       (3) 11.11
 4        64                 25        2.56       (8) 29.63
 5        73                 25        2.92       (2)  7.41
 6        47                 16        2.94       (4) 14.81
 7        68                 24        2.83       (2)  7.41
 8        72                 29        2.48       (4) 14.81
 9        86                 28        3.07       (3) 11.11
Mean      73.00              26.30     2.79       11.48
SE         4.33               1.56     0.07        2.31

------------------------------------------------------------------------
Input File:  heart/fidelity/ga3/it0..9/after_post

Fold    #Condition(Sum)    #Rules    C/R(Mean)    Infidelity
 0        22                  7        3.14       (3) 11.11
 1        28                 12        2.33       (2)  7.41
 2        29                 11        2.64       (3) 11.11
 3        25                  9        2.78       (4) 14.81
 4        26                  9        2.89       (8) 29.63
 5        31                 11        2.82       (2)  7.41
 6        14                  5        2.80       (4) 14.81
 7        53                 20        2.65       (2)  7.41
 8        29                 12        2.42       (3) 11.11
 9        35                 12        2.92       (3) 11.11
Mean      29.20              10.80     2.74       12.59
SE         3.19               1.26     0.08        2.08
```

A.4 Exemplo de Relatório: Indivíduos Antes do Pós-Processamento

```
Genetic Algorithm Report                          Thu Jan  9 20:43:06 2003 Created by
Claudia Regina Milaré                                      v1.0

Training set:   breast/fidelity/ga_gus2/it0/breast.data
Validation set: breast/fidelity/ga_gus2/it0/breast.validation
Test set:       breast/fidelity/ga_gus2/it0/breast.test

Best GA cicle:  13
------------------------------------------------------------------------
Individual:                         0
Data set evaluation:                551
Validation set evaluation:          61
Number of rules:                    26
Mean number of conditions per rule: 2.61538461538462

Individual:                         1
Data set evaluation:                548
Validation set evaluation:          61
Number of rules:                    16
Mean number of conditions per rule: 2.5625

Individual:                         2
Data set evaluation:                552
Validation set evaluation:          61
Number of rules:                    13
Mean number of conditions per rule: 3.07692307692308

Individual:                         3
Data set evaluation:                551
Validation set evaluation:          61
Number of rules:                    10
Mean number of conditions per rule: 3

Individual:                         4
Data set evaluation:                549
Validation set evaluation:          61
Number of rules:                    13
Mean number of conditions per rule: 3
```

```
39
40  Individual:                          5
41  Data set evaluation:                 552
42  Validation set evaluation:           61
43  Number of rules:                     13
44  Mean number of conditions per rule: 3.07692307692308
45
46  Individual:                          6
47  Data set evaluation:                 548
48  Validation set evaluation:           61
49  Number of rules:                     16
50  Mean number of conditions per rule: 2.4375
51
52  Individual:                          7
53  Data set evaluation:                 552
54  Validation set evaluation:           61
55  Number of rules:                     13
56  Mean number of conditions per rule: 3
57
58  Individual:                          8
59  Data set evaluation:                 552
60  Validation set evaluation:           61
61  Number of rules:                     13
62  Mean number of conditions per rule: 3.07692307692308
63
64  Individual:                          9
65  Data set evaluation:                 552
66  Validation set evaluation:           61
67  Number of rules:                     13
68  Mean number of conditions per rule: 3.07692307692308
69
70  BEST INDIVIDUAL ANALISYS
71  ------------------------------------------------------------------------
72
73  Best Individual:                2
74
75  Data set evaluation:                 552
76  Validation set evaluation:           61
77  Number of rules:                     13
78  Mean number of conditions per rule: 3.07692307692308
79
80  Rule    Inducer     Conditions
81  ----    -------     ----------
```

```
R0000   c4.5        4
R0001   c4.5rules   2
R0002   cn2         2
R0003   cn2         4
R0004   c4.5        3
R0005   cn2         2
R0006   cn2         2
R0007   c4.5        5
R0008   c4.5rules   3
R0009   c4.5        5
R0010   cn2         2
R0011   c4.5        3
R0012   c4.5rules   3

------------------------------------------------------------------------
CONFUSION MATRICES FOR THE BEST INDIVIDUAL
------------------------------------------------------------------------

Training set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign      349        0
 malignan        1      203
------------------------------------------------------------------------

Validation set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign       42        0
 malignan        0       19
------------------------------------------------------------------------

Test set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign       43        1
 malignan        2       23
```

A.5 Exemplo de Relatório: Indivíduos Depois do Pós-Processamento

```
1
2  Genetic Algorithm Report                          Thu Jan  9 20:45:50 2003
3  Created by Claudia Regina Milaré                                      v1.0
4
5  Training set:   breast/fidelity/ga_gus2/it0/breast.data
6  Validation set: breast/fidelity/ga_gus2/it0/breast.validation
7  Test set:       breast/fidelity/ga_gus2/it0/breast.test
8
9  Best GA cicle:  13
10 ------------------------------------------------------------------------
11
12 Individual:                          0
13 Data set evaluation:                 551
14 Validation set evaluation:           60
15 Number of rules:                     8
16 Mean number of conditions per rule: 2.5
17
18 Individual:                          1
19 Data set evaluation:                 548
20 Validation set evaluation:           61
21 Number of rules:                     8
22 Mean number of conditions per rule: 2.125
23
24 Individual:                          2
25 Data set evaluation:                 552
26 Validation set evaluation:           61
27 Number of rules:                     7
28 Mean number of conditions per rule: 2.85714285714286
29
30 Individual:                          3
31 Data set evaluation:                 551
32 Validation set evaluation:           59
33 Number of rules:                     6
34 Mean number of conditions per rule: 2.5
35
36 Individual:                          4
37 Data set evaluation:                 549
38 Validation set evaluation:           61
```

```
Number of rules:                    5
Mean number of conditions per rule: 2.4

Individual:                         5
Data set evaluation:                552
Validation set evaluation:          61
Number of rules:                    7
Mean number of conditions per rule: 2.85714285714286

Individual:                         6
Data set evaluation:                548
Validation set evaluation:          61
Number of rules:                    8
Mean number of conditions per rule: 2

Individual:                         7
Data set evaluation:                552
Validation set evaluation:          59
Number of rules:                    7
Mean number of conditions per rule: 2.71428571428571

Individual:                         8
Data set evaluation:                552
Validation set evaluation:          61
Number of rules:                    7
Mean number of conditions per rule: 2.85714285714286

Individual:                         9
Data set evaluation:                552
Validation set evaluation:          61
Number of rules:                    7
Mean number of conditions per rule: 2.85714285714286

BEST INDIVIDUAL ANALISYS
------------------------------------------------------------------------

Best Individual:                2

Data set evaluation:                552
Validation set evaluation:          61
Number of rules:                    7
Mean number of conditions per rule: 2.85714285714286

```

```
Rule    Inducer     Conditions
----    -------     ----------
R0000   c4.5        4
R0001   cn2         2
R0002   cn2         4
R0003   cn2         2
R0004   cn2         2
R0005   c4.5rules   3
R0006   c4.5rules   3

---------------------------------------------------------------------------
CONFUSION MATRICES FOR THE BEST INDIVIDUAL
---------------------------------------------------------------------------

Training set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign      349        0
 malignan        4      200

-------------------------------------------------------------------------

Validation set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign       42        0
 malignan        0       19

-------------------------------------------------------------------------

Test set confusion matrix

           PREDICTED
ACTUAL      benign malignan
   benign       43        1
 malignan        2       23
```

A.6 Regras do Indivíduo Selecionado

```
Standard Rules Conversor v1.1.5     Copyright (c)Ronaldo C. Prati
Inducer: Genetic Algorithm          Input File: No file
Date: Thu Jan  9 20:45:58 2003

R0001  IF UniformityofCellShape > 2
           AND UniformityofCellSize > 1
           AND BareNuclei <= 3
           AND ClumpThickness > 5
       THEN CLASS = malignant [0.0434, 0, 0.3255, 0.6311, 553]

R0002  IF ClumpThickness > 6.50
           AND BlandChromatin > 2.00
       THEN CLASS = malignant [0.2152, 0, 0.1537, 0.6311, 553]

R0003  IF UniformityofCellSize < 4.50
           AND BareNuclei < 5.50
           AND BlandChromatin < 5.50
           AND NormalNucleoli < 1.50
       THEN CLASS = benign [0.5624, 0, 0.0687, 0.3689, 553]

R0004  IF UniformityofCellShape > 1.50
           AND BareNuclei > 5.50
       THEN CLASS = malignant [0.2622, 0, 0.1067, 0.6311, 553]

R0005  IF UniformityofCellSize > 1
           AND UniformityofCellShape > 2
           AND MarginalAdhesion > 2
       THEN CLASS = malignant [0.2857, 0.0018, 0.0831, 0.6292, 553]

R0006  IF ClumpThickness <= 5
           AND MarginalAdhesion <= 2
           AND BareNuclei <= 3
       THEN CLASS = benign [0.5479, 0.0018, 0.0831, 0.3670, 553]

R0007  DEFAULT CLASS = benign

```

A.7 Exemplo de Relatório: Teste de Hipótese

```
Created by Claudia Regina Milaré                Thu Jan 16 00:50:12 2003
================================================================================
                        HYPOTHESIS TEST REPORT
================================================================================
*  Diference is statistically significant (95% confidence level)

Positive results mean "top" algorithm has a lower error rate
Negative results mean "right side" algorithm has a lower error rate

Root: /home/claudia/SNNS/heart
Performing 10-fold paired t-test

--------------------------------------------------------------------------------
                             Infidelity Rate
--------------------------------------------------------------------------------
Comparing: c4.5

  1.048         => with: ga/before_post
  3.973*        => with: c4.5rules
  3.941*        => with: mofn1000
  1.102         => with: mofn0
  0.583         => with: ga0/after_post
 -0.000         => with: simple0
  0.582         => with: simple1000
  0.148         => with: cn2

Comparing: ga0/before_post

 -1.048         => with: c4.5
  1.203         => with: c4.5rules
  2.408*        => with: mofn1000
 -0.001         => with: mofn0
 -0.160         => with: ga0/after_post
 -0.832         => with: simple0
 -0.000         => with: simple1000
 -1.000         => with: cn2

Comparing: c4.5rules

 -3.973*        => with: c4.5
```

```
  -1.203         => with: ga0/before_post
   1.809         => with: mofn1000
  -1.765         => with: mofn0
  -1.105         => with: ga0/after_post
  -1.767         => with: simple0
  -0.758         => with: simple1000
  -1.678         => with: cn2
```

Apêndice B

Resultados do Teste de Hipótese

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple0	0.759	0.447	0.000	1.834	0.697	0.693
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple0	1.970	0.362	1.767	**2.431**	0.552	0.831
	CN2					
	breast	crx	heart	pima	sonar	votes
simple0	2.121	**-2.523**	0.152	1.901	0.728	1.189

Tabela B.1: Taxa de infidelidade: `simple0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple1000	1.788	1.328	-0.582	0.008	-0.253	-1.169
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple1000	2.048	1.355	0.758	0.310	-0.348	-0.809
	CN2					
	breast	crx	heart	pima	sonar	votes
simple1000	**2.419**	-0.340	-0.473	-0.191	-0.124	-0.987

Tabela B.2: Taxa de infidelidade: `simple1000` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn0	-0.181	1.207	1.765	1.001	0.577	-0.269
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn0	0.562	1.116	0.216	1.354	0.498	0.022
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn0	1.106	**-3.888**	-0.612	1.059	0.725	0.444

Tabela B.3: Taxa de infidelidade: `mofn0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn1000	0.183	1.365	**-3.941**	-0.643	1.133	-0.540
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn1000	0.885	1.321	-1.809	-0.389	0.990	-0.293
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn1000	1.233	-1.069	**-2.648**	-0.971	1.079	0.016

Tabela B.4: Taxa de infidelidade: `mofn1000` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple0	0.745	**3.530**	0.843	0.965	**-2.283**	**3.042**
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple0	2.201	**4.491**	**6.623**	**4.474**	**6.442**	**7.134**
	CN2					
	breast	crx	heart	pima	sonar	votes
simple0	**-4.832**	**4.762**	1.678	1.291	**-16.905**	0.279

Tabela B.5: Número de regras induzidas: `simple0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple1000	**2.613**	**2.470**	1.428	2.092	**-3.078**	0.937
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple1000	**3.702**	**2.751**	**5.198**	**5.192**	1.288	**3.621**
	CN2					
	breast	crx	heart	pima	sonar	votes
simple1000	-2.087	**2.546**	1.957	**3.216**	**-13.160**	-1.197

Tabela B.6: Número de regras induzidas: `simple1000` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn0	-1.893	-1.077	**-5.047**	-1.353	**-14.597**	-1.898
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn0	-0.678	1.238	-1.695	**4.564**	-1.809	1.677
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn0	**-6.429**	**-2.571**	**-5.623**	-0.879	**-25.569**	**-17.270**

Tabela B.7: Número de regras induzidas: `mofn0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn1000	**-9.459**	-1.457	**-8.994**	0.503	**-11.289**	-2.196
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn1000	**-8.358**	0.542	**-5.307**	**4.083**	**-4.819**	0.817
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn1000	**-19.575**	**-3.443**	**-12.301**	0.923	**-20.995**	**-7.231**

Tabela B.8: Número de regras induzidas: `mofn1000` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple0	-0.644	0.357	-1.555	-1.495	**-2.433**	0.864
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple0	**10.268**	**5.469**	**5.517**	**27.233**	**6.617**	**2.476**
	CN2					
	breast	crx	heart	pima	sonar	votes
simple0	**11.255**	-0.521	0.871	**23.854**	**14.275**	2.064

Tabela B.9: Número médio de condição por regra induzida: `simple0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
simple1000	**2.332**	-0.044	0.140	1.550	-1.584	-0.317
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
simple1000	**8.049**	2.197	**7.656**	**18.365**	**3.596**	0.478
	CN2					
	breast	crx	heart	pima	sonar	votes
simple1000	**8.152**	-0.749	**2.914**	**17.540**	**7.126**	-0.196

Tabela B.10: Número médio de condição por regra induzida: `simple1000` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn0	**6.491**	**6.966**	**11.048**	**10.463**	**10.737**	**5.513**
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn0	**7.992**	**7.775**	**12.135**	**15.577**	**14.374**	**6.673**
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn0	**7.877**	**6.640**	**11.144**	**15.669**	**14.709**	**7.167**

Tabela B.11: Número médio de condição por regra induzida: `mofn0` x sistemas de AM simbólico.

	C4.5					
	breast	crx	heart	pima	sonar	votes
mofn1000	**7.539**	**10.659**	**17.284**	**15.782**	**12.360**	**6.778**
	C4.5rules					
	breast	crx	heart	pima	sonar	votes
mofn1000	**10.304**	**9.357**	**18.233**	**23.342**	**12.904**	**7.199**
	CN2					
	breast	crx	heart	pima	sonar	votes
mofn1000	**10.690**	**8.200**	**16.893**	**22.933**	**14.184**	**7.242**

Tabela B.12: Número médio de condição por regra induzida: `mofn1000` x sistemas de AM simbólico.

Referências

Aha, D. W., Kibler, D. & Albert, M. (1991). Instance-based Learning Algorithms. *Machine Learning 6*, 37–66. 7, 47

Andrews, R., Diederich, J. & Tickle, A. B. (1995). A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems 8*(6), 373–389. 43, 49, 51, 54

Atlas, L., Cole, R., Connor, J., El-Sharkawi, M., II, R. M., Muthusamy, Y. & Bernard, E. (1989). Performance Comparisons between Backpropagation Networks and Classification Trees on three Real-World Applications. In *Proceedings of the 3rd IEEE Conference*, San Mateo, CA, pp. 622–628. Morgan Kaufmann. 48

Baranauskas, J. A. & Monard, M. C. (2000). Reviewing Some Machine Learning Concepts and Methods. Technical Report 102, ICMC-USP. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_102.ps.zip. 18

Batista, G. E. A. P. A. (1997). Um Ambiente de Avaliação de Algoritmos de Aprendizado de Máquina Utilizando Exemplos. Dissertação de Mestrado, ICMC-USP. 46

Batista, G. E. A. P. A. (2003). Pré-Processamento de Dados em Aprendizado de Máquina Supervisionado. Tese de Doutorado, ICMC-USP. 13, 70

Batista, G. E. A. P. A., Milaré, C. R. & Monard, M. C. (1997). Descrição da Implementação Prolog de uma Ferramenta para Extração de Conhecimento de Redes Neurais. Technical Report 54, ICMC-USP. 59

Batista, G. E. A. P. A. & Monard, M. C. (2003). Descrição da Arquitetura e do Projeto do Ambiente Computacional DISCOVER LEARNING ENVIRONMENT — DLE. Technical Report 187, ICMC-USP. 70, 84

Blake, C., Keogh, E. & Merz, C. J. (1998). UCI Repository of Machine Learning Datasets. 91

Blasig, R. (1993). GDS: Gradient Descent Generation of Symbolic Classification Rules. In *Proceedings of the Neural Information Processing Systems*, Volume 6, San Francisco, CA, pp. 1093–1100. Morgan Kaufmann. 61

Bologna, G. (2002). Rule Extraction from Bagged Neural Networks. In *Soft Computing Systems: Design, Management and Applications*, pp. 42–53. 59

Booch, G., I, J. & Rumbaugh, J. (1998). *The Unified Modeling Language User Guide*. Ddison-Wesley. 76

Boswell, T. (1990a). *Manual for CN2 Version 4.1*. The Turing Institute. 43

Boswell, T. (1990b). *Manual for NewId Version 4.1*. The Turing Institute. 43

Braga, A. P., Carvalho, A. C. P. L. F. & Ludemir, T. B. (2000). *Redes Neurais Artificiais: Teoria e Aplicações*. Rio de Janeiro: Livros Técnicos e Científicos. 30, 35

Braga, A. P., Carvalho, A. C. P. L. F. & Ludermir, T. B. (2003). *Redes Neurais Artificiais* (1 ed.), Chapter 6, pp. 141–168. Volume 1 of Rezende (2003). ISBN 85-204-1683-7. 30

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks. 6, 7, 46

Cameron-Jones, R. M. & Quinlan, J. R. (1993). Efficient Top-Down Induction of Logic Programs. Technical Report NSW 2006, Basser Department of Computer Science, University of Sydney, Australia. 43

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. & Rosen, D. B. (1992). A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks 3*, 698–713. 61

Ciaccia, P., Patella, M. & Zezula, P. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Internation Conference on Very Large Data Bases*, pp. 426–435. 7

Clark, P. & Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning 3*, 261–284. 25, 46, 56, 94

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory* (1 ed.). John Wiley & Sons. 19

Craven, M. W. (1996). *Extracting Comprehensible Models from Trained Neural Networks*. Ph. D. thesis, University of Wisconsin - Madison. 51, 56, 59, 62, 68, 92

Craven, M. W. & Shavlik, J. W. (1999). Rule Extraction: Where Do We Go from Here? Unpublished. 56

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoid Function. *Mathematics of Control, Signals and Systems 2*, 303–314. 35

Darwin, C. (1859). *A Origem das Espécies e a Seleção Natural*. Hemus. 36

Devogelaere, D., Rijckaert, M., Leon, O. G. & Lemus, G. C. (2002). Application of Feedforward Neural Networks for Soft Sensors in the Sugar Industry. In *Proceedings of the VII Brazilian Symposium on Neural Networks (SBRN'02)*, Pernambuco - Brasil, pp. 2–6. 1

Dietterich, T. G. (1997). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation 10*(7), 1895–1924. 99

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science 14*, 179–211. 62

Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM 39*(11), 27–34. 117

Fisher, D. & McKusick, K. (1989). An Empirical Comparison of ID3 and Back-Propagation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, pp. 788–793. Morgan Kaufmann. 48

Freitas, A. A. & Kirner, C. (1992). Introdução a Algoritmos Genéticos. In *Anais do I Workshop sobre Redes Neurais*, UFSCar, São Carlos, pp. 71–88. 36

Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review 13*(1), 3–54. 6

Fu, L. (1991). Rule Learning by Searching on Adapted Nets. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 590–595. Mit Press. 60

Fu, L. (1994). Rule Generation from Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics 24*(8), 1114–1124. 60

Gallant, S. I. (1993). *Neural Network Learning and Expert Systems*. MIT Press. 58

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. 36

Gorman, R. P. & Sejnowski, T. J. (1988). Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks* (1), 75–89. 93

Hall, C. (1992). Neural Net Technology: Ready for Prime Time? *IEEE Expert 7*(6), 2–4. 1

Hayashi, Y. (1991). A Neural Expert System with Automated Extraction of Fuzzy If-Then Rules. In *Proceedings of the Neural Information Processing Systems*, Volume 3, San Mateo, CA, pp. 578–584. Morgan Kaufmann. 60

Haykin, S. (1994). *Neural Networks - A Comprehensive Foundation*. Prentice-Hall. 30

Herkerman, D. (1995). A Tutorial on Learning Bayesian Networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, WA. 47

Hilton, G. (1986). Learning Distributed Representations of Concepts. In *Proceedings of the Eighth Annuak Conference of the Cognitive Science Society*, Amherst, MA. ERLBAUM, pp. 1–12. 49

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press. 36

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *National Academy of Sciences of the U.S.A. 79*, 2554–2558. 9

Hruschka, E. R. (2001). Algoritmos Genéticos de Agrupamento para Extração de Regras de Redes Neurais. Tese de Doutorado, COPPE, Rio de Janeiro. 62

Jordan, M. (1986). Serial Order: A Parallel Distributed Processing Approach. Technical Report 8604, University of California, Institute for Cognitive Science, San Diego. 48

Jr., C. T., Traina, A., Seeger, B. & Faloutsos, C. (2000). Slim-trees: High Performance Metric Trees Minimizing Overlap Between Nodes. In *Conference on Extending Database Technology – EDBT'2000*, pp. 51–65. 7

Krishnan, R., Sivakumar, G. & Bhattacharya, P. (1999). Extracting Decision Trees from Trained Neural Networks. *Pattern Recognition 32*, 1999–2009. 59

Lacerda, E. & de Carvalho, A. C. P. L. F. (1999). *Introdução aos Algoritmos Genéticos*, Volume 7 of *Sistemas Inteligentes - Aplicações a Recursos Hídricos e Ciências Ambientais*, pp. 99–150. Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil. 37

Lavrač, N., Flach, P. & Zupan, R. (1999). Rule Evaluation Measures: A Unifying View. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP-99)*, Volume 1634, pp. 74–185. Springer-Verlag. 74

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1990). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation 1*, 541–551. 1

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Morgan Kaufmann. 30

McCulloch, W. S. & Pitts, W. (1943). A Logical Calculus of the Ideas Immanet in Nervous Activity. *Bulletin of Mathematical Biophysics 5*, 115–133. 8

Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs* (3 ed.). Springer. 37

Michalski, R. (1983). A Theory and Methodology of Inductive Learning. *Artificial Intelligence 20*, 111–161. 41, 47

Michalski, R. S. (1986). Understanding the Nature of Learning: Issues and Research Directions. In *Machine Learning: An Artificial Intelligence Approach (Volume II*, Los Altos, CA. 47

Michalski, R. S., Mozetic, I., Hong, J. & Lavrac, N. (1986). The Multi-purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains. In *Fifth Annual National Conference on Artificial Intelligence*, pp. 1041–1045. 6

Milaré, C. R., Batista, G. & Monard, M. C. (1997). Uma Ferramenta para Extração de Conhecimento de Redes Neurais. In *XXV Seminário Integrado de Software e Hardware - SEMISH*, pp. 59–70. 59

Milaré, C. R. & Carvalho, A. C. P. L. F. (2003). Comparação Experimental de Métodos de Extração de Conhecimento de Redes Neurais Baseados em Aprendizado Indutivo. Technical report, ICMC-USP. (no prelo). 80

Milaré, C. R., Carvalho, A. C. P. L. F. & Monard, M. C. (2002). An Approach to Explain Neural Networks Using Symbolic Algorithms. *International Journal of Computational Intelligence and Applications – IJCIA 2*(4), 365–376. 121

Milaré, C. R. & de Carvalho, A. C. P. L. F. (2001a). Extracting Knowledge from Artificial Neural Networks using Symbolic Learning Algorithms. In *Proceedings of Argentine Simposium on Artificial Inteligence - ASAI'2001*, Volume 30, pp. 48–56. 121

Milaré, C. R. & de Carvalho, A. C. P. L. F. (2001b). Um Estudo Comparativo de Extração de Conhecimento Simbólico de Redes Neurais. In *III Encontro Nacional de Inteligência Artificial - III ENIA*. 121

Milaré, C. R., de Carvalho, A. C. P. L. F. & Monard, M. C. (2001). Extracting Rules from Neural Networks using Symbolic Algorithms: Preliminary Results. In *Proceedings of Fourth International Conference on Computational Intelligence and Multimedia Applications - ICCIMA 2001*, pp. 384–388. 121

Milaré, C. R., de Carvalho, A. C. P. L. F. & Monard, M. C. (2002a). Extracting Knowledge from Artificial Neural Networks: an Empirical Comparison of Trepan and Symbolic Learning Algorithms. In *Proceedings of Second Mexican International Conference on Artificial Intelligence - MICAI 2002*, pp. 272–281. 121

Milaré, C. R., de Carvalho, A. C. P. L. F. & Monard, M. C. (2002b). Uma Abordagem para Explicação de Redes Neurais Artificiais. In *I Workshop de Teses e Dissertações em Inteligência Artificial - WTDIA*, Porto de Galinhas, Brasil. 121

Minsky, M. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge: MIT Press. 9

Mitchell, M. (1997a). *An Introduction to Genetic Algorithms* (3 ed.). Cambridge: MIT Press. 39

Mitchell, T. M. (1980). The Need for Biases in Learning Generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University, New Brunswick, NJ. 47

Mitchell, T. M. (1997b). *Machine Learning*. McGraw-Hill. 3, 19, 87

Monard, M. C. & Baranauskas, J. A. (2003). *Conceitos sobre Aprendizado de Máquina* (1 ed.), Chapter 4, pp. 89–114. Volume 1 of Rezende (2003). ISBN 85-204-1683-7. 3

Monard, M. C., Milaré, C. R. & Batista, G. E. A. P. A. (1998). A Tool to Explore Explanation Facilities in Neural Network. In *Proceedings of ACNN'98*, pp. 128–132. 59

Moreno, J., Sebastian, G., Fernandez, M. & Caballero, A. (1998). A Neural Architecture for the Identification of Number Sequences. In *Vth Brazilian Symposium on Neural Networks*, Belo Horizonte, MG, pp. 247–252. 1

Morgan, J. & Messenger, R. (1973). THAID: A Sequential Search Program for the Analysis of Nominal Scale Dependent Variables. Technical report, Institute for Social Research, University of Michigan. 6

Murphy, P. M. & Pazzani, M. J. (1991). ID2-of-3: Constructive Induction of M-of-N Concepts for Discriminators in Decision Trees. In *Proceedings of the Eighth International Machine Learning Workshop*, pp. 183–187. Morgan Kaufmann. 66

Pau, L. F. & Götzche, T. (1992). Explanation Facility for Neural Networks. *Journal of Intelligent and Robotic Systems 5*, 193–206. 58

Pineda, F. J. (1987). Generalization of Back-Propagation to Recurrent Neural Networks. *Physical Review Letters 59*, 2229–2223. 48

Prati, R. C., Baranauskas, J. A. & Monard, M. C. (2001a). Extração de Informações Padronizadas para a Avaliação de Regras Induzidas por Algoritmos de Aprendizado de Máquina Simbólico. Technical Report 145, ICMC-USP. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip. 13, 72, 76

Prati, R. C., Baranauskas, J. A. & Monard, M. C. (2001b). Uma Proposta de Unificação da Linguagem de Representação de Conceitos de de Aprendizado de Máquina Simbólico. Technical Report 137, ICMC-USP. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip. 11, 13, 70, 72, 76

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning 1*, 81–106. Reprinted in Shavlik and Dieterich (eds.) Readings in Machine Learning. 6

Quinlan, J. R. (1987a). Generating Production Rules from Decision Trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Italy, pp. 304–307. 6

Quinlan, J. R. (1987b). Simplifying Decision Trees. *International Journal of Man-Machine Studies 27*, 221–234. 6

Quinlan, J. R. (1988). *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, CA. 18, 19, 25, 43, 46, 47, 56, 63, 66, 94

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77*(2), 257–286. 47

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações* (1 ed.), Volume 1 of *1*. Barueri, SP, Brasil: Editora Manole. ISBN 85-204-1683-7. 144, 147

Rivest, R. (1987). Learning Decision Lists. *Machine Learning 2*, 229–246. 46

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psycological Review 65*, 386–40. 8

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Books. 31

Rumelhart, D., Hilton, G. & Williams, R. (1986). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1. Cambridge, MA: MIT Press. 9, 34, 47, 97

Russel, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach* (1 ed.). Prentice Hall. 45

Russel, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2 ed.). Prentice Hall. 44

Saito, K. & Nakano, R. (1988). Medical Diagnostic Expert System Based on PDP Model. In *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, pp. 255–262. IEEE Press. 57

Schellharmmer, I., Diederich, J., Towsey, M. & Brugman, C. (1997). Knowledge Extraction and Recurrent Neural Networks: an Analysis of an Elman Network Trained on a Natural Language Learning Task. Technical Report 97-IS1, Queensland University of Technology, Australia. 62

Schmitz, G. P. J., Aldrich, C. & Gouws, F. S. (1999). ANN-DT: An Algorithm for Extraction of Decision Trees from Artificial Neural Networks. *IEEE Transactions on Neural Networks 10*(6), 1392–1401. 59

Setiono, R. & Liu, H. (1995). Understanding Neural Networks via Rule Extraction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, pp. 480–485. Morgan Kaufmann. 61

Shapire, R. E. (1990). The Strength of Weak Learnability. *Machine Learning 5*(2), 197–227. 116

Shavlik, J. W., Mooney, R. & Towell, G. (1991). Symbolic and Neural Net Learning Algorithms: an Empirical Comparison. *Machine Learning 6*, 111–143. 48

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. New York, NY: Chapman and Hall. 65

Stanfill, C. & Waltz, D. (1986). Instance-based Learning Algorithms. *Communications of the ACM 12*, 1213–1228. 8, 47

Tan, A. H. (1994). Rule Learning and Extration with Self-Organizing Neural Network. In *Proceedings of the 1993 Connectionist Models Summer School*, Hillsdale, NJ, pp. 192–199. Lawrence Erlbaum Associates. 61

Tchoumatchenko, I. & Ganascia, J. G. (1994). A Bayesian Framework to Integrate Symbolic and Neural Learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, pp. 302–308. Morgan Kaufmann. 59

Thrun, S. (1995). Extracting Rules from Artificial Neural Networks with Distributed Representations. In *Proceedings of the Neural Information Processing Systems*, Volume 7, Cambridge, MA, pp. 505–512. MIT Press. 58

Thrun, S. B. (1994). Extractiong Provably Correct Rules from Artificial Neural Networks. Technical Report IAI-TR-93-5, Institut für Informatik III, Universität Bonn. 1

Tickle, A. B., Andrews, R., Golea, M. & Diederich, J. (1998). The Truth will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. *IEEE Transactions on Neural Networks 9*(6), 1057–1068. 56

Tickle, A. B., Orlowski, M. & Diederich, J. (1994). DEDEC: Decision Detection by Rule Extraction from Neural Network. Technical report, Queensland University of Technology, Neurocomputing Research Center. 61

Towell, G. & Shavlik, J. W. (1993). The Extraction of Refined Rules from Knowledge-Based Neural Networks. *Machine Learning 131*, 71–101. 55, 60

Towell, G., Shavlik, J. W. & Craven, M. W. (1991). Interpretation of Artificial Neural Networks: Mapping Knowledge-Based Neural Networks into Rules. Technical report, Computer Sciences Department, University of Wisconsin, Madison, WI. 60

Tsukimoto, H. (2000). Extracting Rules from Trained Neural Networks. *IEEE Transactions on Neural Networks 11*(2), 377–389. 61

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag. 116

Vapnik, V. (1998). *Statistical Learning Theory*. New York: John Wiley and Sons, Inc. 116

Weiss, S. M. & Kapouleas, I. (1989). An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, pp. 688–693. Morgan Kaufmann. 48

Weiss, S. M. & Kulikowski, C. A. (1991). *Computer Systems that Learn. Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufmann. 3, 33, 46, 96

Widrow, B. & Hoff, M. E. (1960). Adaptive Switching Circuits. In *Western Eletronic Show and Convertion*, Institute of Radio Engineers, pp. 96–104. 33

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks 5*, 241–259. 116

Wolpert, D. H. (1995). The Relationship between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework. In *The Mathematics of Generalization*, pp. 117–214. 46

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control 8*, 338–353. 55, 61

Zell, A. (1995). Stuttgart Neural Network Simulator. http://www.ra-informatik.uni-tuebingen.de/SNNS/. 84, 97