Aprendizado de Máquina por Exemplos Usando Árvores de Decisão.

Maria Inés Castiñeira ¹

Orientação:

Dra. Maria Carolina Monard

*Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos – USP,
como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação
e Matemática Computacional.*

USP - São Carlos
Setembro de 1990

¹Trabalho realizado com auxílio do CNPq

*A mis padres y mis amigos,
con cariño especial, a Ely.*

Agradecimentos

À Maria Carolina, pela dedicação e amizade brindada ao longo do desenvolvimento deste trabalho.

À Maria do Carmo Nicoletti da UFSCar, pelas valiosas e constantes sugestões.

Aos Drs. Eduardo Gutierrez Borges e Artur Lopes Gonçalves do Departamento de Pediatria da USP - Rib. Preto, pelos dados sobre Pediatria.

Ao pesquisador Luiz Rocha da Embrapa - Fazenda Canchim - São Carlos, pelos dados sobre as gramíneas.

Ao Daniel e Maria Cecilia pelo incentivo e felicidade do dia a dia.

Aos amigos Solange, Maritza, Júlia, Júnia e João pela amizade e constante participação neste trabalho.

Ao CNPq pelo apoio financeiro.

Aqueles que de um modo ou de outro contribuíram para a realização deste trabalho.

“...Scientific knowledge through demonstration² is impossible unless a man knows the primary immediate premises... We must get to know the primary premisses by induction; for the method by which even sense-perception implants the universal is inductive...”

- Aristotle, Posterior Analytics, Book II, Chapter 19 (circa 330 B.C.)

²Isto é, o que hoje é chamado de “dedução”.

Abstract

Machine Learning is an important research area of Artificial Intelligence, since the ability to learn is central to intelligent behavior. Making generalizations — induction — is the means by which humans learn most of their knowledge.

In this work we describe several approaches to Machine Learning and concentrate our attention on a family of learning systems called TDIDT — Top Down Induction Decision Trees. The task of these systems is to induce general descriptions of concepts, from examples of this concepts, using decision trees as a knowledge formalism. Although decision trees are a simple formalism, the learning methodologies used by the TDIDT family are less complex than the methodologies used by other systems that employ a more powerful language to express the results of the learning process. Nevertheless, decision trees are capable of capturing knowledge which is useful to solve difficult problems.

In general, TDIDT family's algorithms develops a decision tree from a set of examples in three main stages: construction of the tree to classify the examples, pruning such a tree to give statistical reliability and processing of the pruned tree to improve understandability. In this work the first two stages are considered.

Related to the first stage, we propose an efficient Prolog implementation for the construction of decision trees. The decision tree is grown by choosing, at each node, the attribute which divides "best" the set of examples considered. In this particular implementation the attribute is chosen by an entropy measure, although it is simple to redefine and implement in the system another kind of measure.

Related to the second stage we propose a pruning method which estimates the classification errors in the nodes of the decision tree previously constructed and then, considering this error, decides whether to prune certain subtrees. This method was applied to several domains and sets of data to measure the size of the pruned tree and its accuracy. Results show that the complexity of the pruned decision tree decreases while its accuracy increases; both measures are heavily dependent on the domain.

Resumo

O Aprendizado de Máquina é uma importante área de pesquisa em Inteligência Artificial pois a capacidade de aprender é essencial para um comportamento inteligente. Em particular, um dos objetivos da pesquisa em Aprendizado de Máquina é o de auxiliar o processo de aquisição de conhecimento facilitando a construção de Sistemas Baseados em Conhecimento.

Uma das formas de aprendizagem é por generalizações, isto é, através de processos indutivos. São várias as estratégias desenvolvidas para Aprendizado de Máquina por Indução. Uma delas está baseada na construção de árvores de decisão. Esta estratégia abrange uma determinada família de sistemas de aprendizado por indução: a família TDIDT — Top Down Induction Decision Trees.

Neste trabalho são apresentadas algumas estratégias de Aprendizado de Máquina, dando maior ênfase aos sistemas da família TDIDT, bem como detalhes da implementação realizada. Mostra-se que é possível realizar uma implementação geral dos algoritmos desta família.

Mostra-se também a importância dos diversos mecanismos de poda em árvores de decisão. Um método de poda específico é usado para podar árvores geradas em diversos domínios. Os resultados obtidos evidenciam que este método reduz a complexidade da árvore e produz ganhos significativos na classificação por ela realizada.

Conteúdo

1	Introdução	1
1.1	Considerações Iniciais	1
1.2	Organização deste Trabalho	2
2	Aprendizado de Máquina	4
2.1	Considerações Iniciais	4
2.2	Objetivos do Estudo de Aprendizado de Máquina	5
2.3	Classificação das Estratégias de Aprendizado de Máquina	6
2.3.1	Aprendizado por Implantação Direta do Conhecimento ou Aprendizado por Hábito	8
2.3.2	Aprendizado por Instrução	9
2.3.3	Aprendizado por Dedução	12
2.3.4	Aprendizado por Analogia	12
2.3.5	Aprendizado por Indução	15
2.4	Histórico da Pesquisa de Aprendizado de Máquina	18
2.4.1	Modelagem Neural e Sistemas Paramétricos	19
2.4.2	Aprendizado Geral de Conceitos por Representações Simbólicas	19
2.4.3	Sistemas de Aprendizado de Conhecimento-intensivo Explorando Várias Tarefas de Aprendizado	20
2.5	Considerações Finais	21
3	Aprendizado de Conceitos por Indução	22

3.1	Argumentos Dedutivos <i>versus</i> Argumentos Indutivos	22
3.1.1	Indução	24
3.1.2	Indução por Enumeração	26
3.1.3	Estatística Insuficiente	27
3.1.4	Desvios Estatísticos	27
3.2	Aprendizado por Indução	28
3.2.1	Paradigma Indutivo	30
3.2.2	Tipos de Descrições	32
3.2.3	Aprendizado de um Único Conceito <i>versus</i> Múltiplos Conceitos . . .	33
3.2.4	Linguagem Descritiva	34
3.2.5	Problema do Conhecimento de Fundo	36
3.2.6	Importância dos Descritores Iniciais	36
3.2.7	O Critério de Preferência	37
3.2.8	Regras de Generalização	38
3.3	Estratégias de Aprendizado por Indução	44
3.3.1	Estratégias de Controle: Métodos “Data-driven” <i>versus</i> Métodos “Model-driven”	44
3.3.2	Aprendizado por Exemplos como Busca	46
3.3.3	Aprendizado por Exemplos Incremental e Não Incremental	51
3.3.4	Indução Construtiva	51
3.4	Considerações Finais	52
4	Família TDIDT	53
4.1	Construindo Árvores de Decisão	53
4.2	ID3	58
4.2.1	Entropia	60
4.2.2	O Sistema ID3	62

4.3	Família TDIDT	64
4.4	Histórico de Sistemas de Aprendizado por Exemplos	69
4.5	Vantagens e Desvantagens	70
4.6	Considerações Finais	72
5	Implementação de um Sistema para Construção de Árvores Indutivas de Decisão	73
5.1	Considerações Gerais da Implementação	73
5.2	Especificação da Implementação	74
5.3	Considerações Finais	77
6	Poda em Árvores de Decisão	78
6.1	Resultados Experimentais	79
6.1.1	Conjunto de Dados	79
6.1.2	Método Experimental	81
6.1.3	Resultados	81
6.2	Considerações Finais	84
7	Conclusões	85
7.1	Considerações Gerais	85
7.2	Conclusões	86
7.3	Sugestões de Trabalhos Futuros	87
8	Bibliografia	88

Lista de Figuras

2.1	Armazenando Valores da Árvore de Jogos	9
2.2	Problemas de Geometria	13
2.3	Problema de Geometria com Ângulos	14
2.4	Um Frame de AM: o Conceito de Número Primo, [Rich 83].	17
3.1	Conceitos no Mundo dos Blocos.	29
3.2	Instâncias Positivas e Negativas de Células Doentes	47
3.3	Duas Possíveis Descrições de Células Doentes	47
3.4	Ordem Parcial no Espaço das Hipóteses	49
3.5	Gerando Hipóteses do Específico ao Geral	50
4.1	Árvore de Decisão Incompleta	54
4.2	Construindo a Árvore de Decisão	56
4.3	Construindo a Árvore de Decisão	56
4.4	Construindo a Árvore de Decisão	57
4.5	Árvore de Decisão	58
4.6	Árvore de Decisão Maior	59
4.7	Mecanismo de Janela do ID3	63
4.8	A Família TDIDT	66
4.9	Círculo Verde	70
5.1	Lista de Dados	75

6.1	Divisão dos Dados nos Conjuntos E , C_p e C_e	82
-----	---	----

Lista de Tabelas

4.1	Conjunto de Observações ou Exemplos	55
6.1	Árvore sem Poda	81
6.2	Árvore com Poda	83
6.3	Erro-Precisão da Árvore	84

Capítulo 1

Introdução

1.1 Considerações Iniciais

Um dos objetivos dos pesquisadores da área de Inteligência Artificial – IA – é tornar os computadores mais acessíveis aos seres humanos. É por este motivo que muitas pesquisas em IA exploram formas de permitir que computadores aprendam; esta área é denominada Aprendizado de Máquina – AM.

Uma das motivações para o estudo de AM é a viabilidade na automação da aquisição do conhecimento, isto é, permitir que o computador extraia conceitos e relações a partir de bases de dados ou a partir de sessões interativas com o usuário, para, posteriormente, utilizá-los em qualquer atividade baseada em conhecimento. Uma possível atividade é o desenvolvimento de Bases de Conhecimento para auxiliar a construção de Sistemas Especialistas – SE.

Eventualmente, o estudo de técnicas baseadas em AM poderia contribuir para um melhor entendimento do nosso próprio processo de aprendizado. Entretanto, ainda que possam ser feitas diversas comparações e relacionamentos entre as várias facetas do aprendizado humano e o aprendizado de máquina, este aspecto de AM não está entre os objetivos deste trabalho.

Existem diversas estratégias de AM. Uma delas é denominada Aprendizado por Indução. Nesta estratégia de aprendizado, o sistema de aprendizado incrementa seu conhecimento, ou então descobre novos padrões ou regras, através da inferência indutiva realizada sobre fatos fornecidos ou observados.

Entre os sistemas que realizam Aprendizado por Indução há uma família de sistemas — denominada TDIDT “Top Down Induction Decision Trees” — que constroem árvores de decisão a partir de exemplos. De uma maneira geral, pode-se dizer que estes sistemas constroem regras a partir dos dados fornecidos, ou geram diagnósticos a partir de exemplos observados. Estes diagnósticos ou regras são representados na forma de árvores de

decisão. Estes sistemas estão sendo utilizados, prioritariamente, para auxiliar a extração do conhecimento do especialista no processo de construção de Sistemas Baseados em Conhecimento.

O objetivo deste trabalho é mostrar diversas estratégias de AM dando ênfase aos sistemas de Aprendizado por Indução da família TDIDT. É mostrado, também, que é possível realizar uma implementação geral dos algoritmos desta família. As idéias gerais utilizadas nessa implementação — realizada na linguagem Prolog — são apresentadas. O sistema implementado foi aplicado sobre dados de diversos domínios, gerando as correspondentes árvores de decisão.

Discute-se, também, a importância dos mecanismos de poda como meio de reduzir a complexidade das árvores de decisão, bem como para aumentar a sua precisão. Um método de poda específico é proposto e implementado. Este método foi utilizado em diversos domínios e foram realizadas várias medições para verificar a complexidade e precisão das árvores resultantes. Os resultados obtidos também são discutidos, eles confirmam a eficiência do método proposto.

1.2 Organização deste Trabalho

A dissertação apresentada está organizada da seguinte forma:

O capítulo 2 abrange diversos tópicos de Aprendizado de Máquina. Primeiramente, devido ao amplo universo do tema e ao fato deste ser o primeiro trabalho de pesquisa em AM no ICMSC, foi realizado um levantamento geral, relativamente exaustivo, desta área, onde se procurou estabelecer o que é AM, quais os objetivos no estudo de AM, uma classificação das diferentes estratégias de aprendizado, bem como um breve histórico das pesquisas nesta área.

O capítulo 3 descreve os problemas relacionados com a estratégia de Aprendizado por Indução. Primeiramente é discutida a indução lógica através de um paralelo com a dedução lógica, isto é, quais os objetivos e características de cada uma delas; enquanto os argumentos dedutivos servem para explicitar e validar o conhecimento, os argumentos indutivos atendem à finalidade de ampliar o alcance do conhecimento já existente. É mostrado, também, que argumentos indutivos são de fundamental importância nos processos de descoberta e generalização científica; em particular, eles são os componentes básicos utilizados nos processos de Aprendizado por Indução.

No mesmo capítulo são estudadas as características e problemas do paradigma indutivo. Entre estas, são mencionadas: a linguagem de descrição do problema, a importância e tipo dos descritores, as regras de generalização que são utilizadas na construção dos conceitos, bem como os critérios de preferência e conhecimento de fundo que podem, eventualmente, servir como guia no processo indutivo. Resumidamente, pode-se dizer que este tópico trata da descrição do problema a ser resolvido.

O capítulo 3 finaliza apresentando algumas das soluções propostas, isto é, são descritas as metodologias das diferentes estratégias de Aprendizado por Indução.

No capítulo 4 é descrito o processo de construção de árvores de decisão com um exemplo detalhado. É enfatizada a necessidade de um critério ou função de avaliação para escolher o melhor atributo durante o processo de construção da árvore. A seguir é descrito o sistema ID3, que utiliza a entropia como função de avaliação. Este é um dos sistemas mais conhecidos da família TDIDT. Outros sistemas pertencentes à família TDIDT são apresentados, assim como diversas funções de avaliação. Neste capítulo também é apresentado um histórico das pesquisas nessa área, bem como as linhas de pesquisa mais recentes.

No capítulo 5 são discutidas as principais idéias utilizadas na implementação de um sistema geral para a construção de árvores indutivas de decisão. Esta implementação, na linguagem Prolog, foi realizada como resultado prático deste trabalho.

No capítulo 6 é discutida a importância da poda em árvores de decisão, como mecanismo através do qual se procura evitar a construção de subárvores não confiáveis. Também são descritos os dois tipos de poda existentes: pré-poda e pós-poda. Além disso, discute-se: o método de poda proposto, os conjuntos de dados utilizados para verificar a validade desse método, e a metodologia utilizada. Finalmente, são feitas considerações sobre os resultados obtidos.

No capítulo 7 são apresentadas as conclusões e algumas sugestões de trabalhos futuros.

Capítulo 2

Aprendizado de Máquina

Neste capítulo é introduzido o problema de Aprendizado de Máquina em geral. Nas duas primeiras seções é descrito o que é AM e qual o interesse em estudar este tópico. Logo a seguir é apresentada uma classificação das estratégias de AM, com uma descrição resumida de cada uma destas estratégias. A última seção apresenta um histórico das pesquisas de AM.

2.1 Considerações Iniciais

Uma das essências da nossa inteligência é a capacidade de aprender. O ser humano aprende com a experiência, aprende ouvindo o que lhe é dito e aprende ampliando, através de reordenações sucessivas, o alcance do conhecimento que já possui.

Num processo de aprendizagem é de vital importância a orientação externa desempenhada pelo ambiente ou professor, que estimulam e orientam esta tarefa. Charniak e Mc Dermott [Charniak 86] comentam que:

Para um organismo aprender, ele já deve saber muita coisa. O aprendizado começa com a organização do conhecimento, que vai crescendo e ficando melhor organizado. Sem uma forte indicação do que aprender, nada será aprendido.

Não são necessárias muitas considerações para se concluir que qualquer máquina inteligente (ainda que com uma inteligência limitada) deve apresentar alguma capacidade para aprender. Aprendizado de Máquina é a área de estudo da Inteligência Artificial na qual se pesquisa o desenvolvimento deste tipo de habilidade pela máquina.

Carbonell e Langley [Carbonell 87] definem Aprendizado de Máquina como a área que estuda métodos computacionais apropriados para a aquisição de novos conhecimentos,

novas habilidades e novas formas de organização do conhecimento já existente. Geralmente se associa aprendizado com aquisição de conhecimento. Porém, como ressalta Elaine Rich [Rich 83], ainda que a aquisição de novos conhecimentos seja um componente do aprendizado, nem todo programa que envolve aquisição de conhecimento é um programa de aprendizado. Isto não significa que o conhecimento específico sobre o domínio relevante a ser aprendido não seja importante. Um bom desempenho no processo de aprendizado dependerá — entre outras coisas — de se ter o conhecimento apropriado disponível.

2.2 Objetivos do Estudo de Aprendizado de Máquina

Há várias razões para se estudar Aprendizado de Máquina. Uma delas é para obter um melhor entendimento dos mecanismos de aprendizado humano.

Um outro objetivo da pesquisa de Aprendizado de Máquina é o de automatizar a aquisição do conhecimento. Atingir esta automação, total ou parcialmente, pode ser um passo importante para facilitar a construção de Sistemas Baseados em Conhecimento, entre os quais pode-se mencionar, especificamente, os Sistemas Especialistas [Jackson 86].

Um Sistema Especialista – SE – é um programa que se comporta como um especialista humano em algum domínio específico de conhecimento. Um dos componentes deste tipo de programa é a Base de Conhecimento, que contém a modelagem do conhecimento e, algumas vezes, heurísticas para manipular este conhecimento. Um outro componente de um Sistema Especialista é o Motor de Inferência, responsável pelo tipo de raciocínio do sistema. Sistemas Especialistas são caros de se construir porque o processo de extração do conhecimento do especialista, bem como sua representação na Base de Conhecimento numa forma adequada para a máquina de inferência, é lento e requer a atuação conjunta do especialista da área e do especialista em modelar o conhecimento — chamado de Engenheiro do Conhecimento. A experiência mostra que, mesmo quando há disponível um especialista no domínio — geralmente estas são pessoas muito requisitadas — a instalação e depuração de regras na Base de Conhecimento leva de uma a quatro regras por homem/dia [Quinlan 86a], [Shapiro 87].

A razão disto é que, geralmente, os especialistas humanos são peritos na sua área de atuação, mas tem dificuldade para explicar as razões de suas decisões profissionais; eles não são peritos em descrever seus próprios processos de raciocínio. Além disso, os especialistas raramente são cientistas em computação. Geralmente não sabem como adicionar regras de conhecimento a um determinado sistema de software, uma vez que desconhecem qual formato estas devem ter para que possam ser utilizadas pela Máquina de Inferência de um SE.

Assim, seria interessante ter uma interface direta entre o especialista e o sistema de conhecimento. Tendo uma máquina que faça diretamente o processo de transferência do conhecimento do especialista para a representação manipulada pela máquina, espera-se

reduzir o custo de desenvolvimento de novos Sistemas Especialistas.

2.3 Classificação das Estratégias de Aprendizado de Máquina

Diferentes atributos podem ser utilizados para categorizar as estratégias de Aprendizado de Máquina; é por este motivo que diversas classificações de AM podem ser achadas na literatura.

Bierman [Bierman 88] propõe uma divisão dos sistemas de aprendizado em dois grupos, e introduz outros parâmetros para categorizar esses sistemas:

- implantação direta do conhecimento ou “rote memorization”;
- aprendizado por inferência indutiva.

O primeiro grupo enquadra sistemas de aprendizado por memorização direta, ou seja, é aprendizado por simples armazenamento de dados. O sistema aprendiz não faz nenhum tipo de esforço ou raciocínio para elaborar ou adquirir novos conhecimentos. Um exemplo deste paradigma é o programa de damas de Samuel [Samuel 63], descrito mais detalhadamente em seções posteriores.

No segundo grupo — aprendizado por inferência indutiva — o sistema aprendiz deve relacionar de alguma forma os fatos observados, tendo um papel mais ativo no processo de aprendizado.

Tanimoto [Tanimoto 87] divide os mecanismos de aprendizado em paramétricos e estruturais:

- paramétricos são aqueles nos quais a melhora no seu desempenho está baseada apenas no ajuste ou comparação de parâmetros;
- estruturais são aqueles nos quais o sistema aprendiz deve trabalhar relacionando, ou descobrindo, os elementos constituintes da estrutura a ser apreendida.

Carbonell, Michalski e Mitchell [Carbonell 83] propõem os seguintes três tipos de classificações para AM:

- classificação baseada em estratégias fundamentais de aprendizado: as estratégias se organizam segundo a quantidade de inferência que o sistema deve realizar sobre a informação disponível;

- classificação baseada no tipo de representação do conhecimento utilizada pelo sistema aprendiz;
- classificação segundo o domínio da aplicação.

A primeira destas classificações, baseada nas estratégias fundamentais de aprendizado, é também adotada por Michalski [Michalski 87] e será discutida a seguir.

Suponha um aluno que, através da observação e resolução de uma determinada quantidade de exemplos e problemas, descobre que

“aqueles números que só tem dois divisores são divisíveis pela unidade e por eles mesmos”.

Isto é, ele descobre por si próprio o conceito de número primo. Dessa forma, a quantidade de inferência e o esforço realizado para aprender esse conceito é consideravelmente maior do que se um professor tivesse lhe ensinado. Analogamente, as estratégias de aprendizado abrangem um amplo espectro, que começa com as estratégias onde o sistema aprendiz desempenha um papel passivo, e termina com aquelas onde o aprendiz é autosuficiente para aprender.

Em geral, conforme aumenta a quantidade de inferência que o aprendiz pode fazer, diminui a responsabilidade do professor, ou ambiente externo, que guiam o processo de aprendizado.

A seguir, as diferentes estratégias de aprendizado são apresentadas segundo a ordem crescente de complexidade de inferência realizada pelo aprendiz, muito embora em todo ato prático de aquisição de conhecimento, mais do que uma estratégia é utilizada simultaneamente. As estratégias de aprendizado são:

- Implantação direta do conhecimento ou aprendizado por hábito;
- Aprendizado por instrução;
- Aprendizado por dedução;
- Aprendizado por analogia;
- Aprendizado por indução, que abrange:
 - Aprendizado através de exemplos e
 - Aprendizado por observação e descoberta.

Cada uma destas estratégias será brevemente analisada nas próximas seções. Entretanto, cabe ressaltar que há diversos tipos de aprendizado além dos aqui mencionados, como por

exemplo: aprendizado da linguagem, algoritmos de aprendizado genético [Carbonell 87], modelos conexionistas de aprendizado [Hinton 89] [Mc Clelland 87], aprendizado por falha [Charniak 86], aprendizado por experimentação e aprendizado por reconceitualização [Subramanian 86] entre outros.

2.3.1 Aprendizado por Implantação Direta do Conhecimento ou Aprendizado por Hábito

Este é um caso extremo de aprendizado no qual nenhuma inferência, ou qualquer outro tipo de transformação do conhecimento, é requerido por parte do aprendiz — o conhecimento fornecido pela fonte é diretamente aceito pelo sistema aprendiz. Esta estratégia inclui aprendizado por memorização direta e aprendizado por estar programado.

O programa de damas de Samuel [Samuel 63] — um dos programas mais antigos de jogos —, aprendeu a jogar damas suficientemente bem para vencer seu criador, utilizando as técnicas de aprendizado por hábito. Este programa utiliza o procedimento de busca minimax [Pearl 84] para explorar árvores de jogo de damas. A técnica de aprendizado por hábito é realizada armazenando cada um dos valores que são calculados pelo procedimento minimax para as diferentes posições do tabuleiro. Quando o procedimento minimax avalia uma jogada, ele cria uma árvore de jogo para essa determinada situação. A raiz da árvore representa a posição do tabuleiro onde está a peça cuja jogada será avaliada.

Para exemplificar, suponha que para uma determinada jogada o programa explora a árvore mostrada na figura 2.1(a), retornando o valor 10 para a posição “A” do tabuleiro. Este valor corresponde à pesquisa em apenas uns poucos níveis na árvore de jogo; por problemas de tempo, não é conveniente fazer uma busca exaustiva. A seguir o programa armazena este valor para a posição “A” do tabuleiro, isto é, ele “aprende” que o valor correspondente a esta posição é 10. Se em alguma jogada subsequente surgir uma árvore de jogo como a mostrada na figura 2.1(b), o procedimento minimax não precisa calcular novamente o valor do nó “A”; ele simplesmente utiliza a contagem previamente armazenada. Isto cria o efeito de ter pesquisado diversos lances adicionais na árvore de jogo.

Este tipo de aprendizagem, muito simples, evidencia a necessidade de consideração de algumas características que se tornam cada vez mais importantes em sistemas de aprendizado mais complexos. Essas características incluem:

- **Organização da memória.** O armazenamento organizado da informação é necessário para permitir que seja mais rápido utilizar um valor armazenado do que recalculá-lo.
- **Generalização.** É necessária a utilização de algum tipo de generalização para diminuir o número de objetos que devem ser armazenados. O programa de damas de Samuel utiliza técnicas de generalização muito simples, como por exemplo: todas as posições do tabuleiro são armazenadas como se o lado das peças branco fosse jogar.

Figura 2.1: Armazenando Valores da Árvore de Jogos

Isto diminui pela metade o número de posições a serem armazenadas. Entretanto, à medida que a complexidade do processo de aprendizado aumenta, também aumenta a necessidade de generalização.

- **Direção promissora.** É importante que o programa saiba escolher, entre várias alternativas válidas de ação, aquela que seja mais promissora. Isto é realizado pelo programa de Samuel que favorece posições com a mesma contagem, mas que são obtidas com um número menor de movimentos.

2.3.2 Aprendizado por Instrução

Nesta estratégia de aprendizado, o aprendiz adquire o conhecimento de um instrutor ou outra fonte organizada, tal como um livro ou publicação. Estas informações, porém, não são copiadas diretamente na memória. O sistema aprendiz deve fazer uma transformação da informação transmitida pela fonte, selecionando os fatos mais relevantes e/ou transformando estes para o seu formato de representação interna.

Já nos primórdios da Inteligência Artificial surgiu o termo “advice taker” para descrever um programa que pudesse aprender de um professor [McCarthy 58]. Porém, os progressos realizados até hoje para poder implementar a idéia de McCarthy têm sido vagarosos. Um dos principais problemas são os canais de comunicação aprendiz/professor.

Idealmente, esta comunicação deveria ser feita em linguagem natural. Assim sendo, tem-se pela frente todos os problemas inerentes ao processamento de linguagem natural — ambigüidade, pronomes de referência, etc. [Caldeira 89]. Além disso, o sistema deveria ser capaz de traduzir corretamente aquela informação para sua própria estrutura de representação interna; usualmente o professor não conhece qual é o tipo de representação empregado pelo sistema aprendiz.

A partir desta última colocação surgiram duas abordagens diferentes. Na primeira delas, o sistema deve ser capaz de relacionar, ou traduzir, os ensinamentos feitos em linguagem natural (ou em uma linguagem próxima à natural) para suas próprias estruturas de representação interna. Exemplo disto é o programa Foo/Bar de Mostow [Mostow 83, 83a], que aprende um determinado jogo de cartas.

Na segunda abordagem é necessário que o professor entenda completamente como a informação é indagada e, mais superficialmente, qual é o formato no qual esta informação é armazenada pela máquina, para só então poder transmiti-la ao sistema na forma por ele requerida.

Como um exemplo desta segunda abordagem, será descrito superficialmente o sistema Teiresias de Davis [Davis 82], que é uma interface do sistema Mycin [Shortliffe 76] — Sistema Especialista que diagnostica infecções bacterianas.

Todas as indagações realizadas ao sistema Teiresias são resolvidas pelo Motor de Inferência do Mycin, através de um processo de encadeamento regressivo ou “backward chaining” — isto é, pesquisando a lista de hipóteses e procurando reunir evidências, de maneira a ser possível concluir pela validade de alguma(s) da(s) hipótese(s). O conhecimento prévio do especialista sobre o tipo de raciocínio utilizado pelo Mycin, o auxilia na hora de depurar as regras.

O especialista fornece problemas-amostra a Teiresias para validar e depurar as regras do sistema. Se para um determinado problema o especialista discordar da conclusão a que o sistema chegou, então Teiresias mostra o raciocínio que o levou àquela conclusão. Teiresias trabalha voltando na cadeia do raciocínio, até o especialista reconhecer o lugar do erro. Se a cadeia é errônea, é porque há alguma conclusão que não deveria ter sido feita, ou então porque está faltando alguma conclusão que deveria ter sido inferida.

Como exemplo do primeiro caso — há alguma conclusão indevida — suponha que o sistema possua a seguinte regra:

Se (X tem dor_de_cabeça) $\Rightarrow$ X pode_tomar aspirina.

Não é aconselhável, entretanto, que pessoas com úlcera tomem aspirinas. Neste caso, o programa deve especializar esta regra, isto é, modificar as suas premissas de maneira a tornar a regra mais restrita. Uma solução seria modificar a regra da seguinte forma:

*Se (X tem dor_de_cabeça) e (X não_tem úlcera)
 $\Rightarrow$ X pode_tomar aspirina.*

Para exemplificar o segundo caso — falta de alguma conclusão que deveria ter sido inferida — suponha agora que o sistema possua a seguinte regra:

*Se (X tem dor_de_cabeça) e (X não_tem úlcera) e
(o_medico aprovar) $\Rightarrow$ X pode_tomar aspirina.*

e a seguinte situação do mundo real:

José — uma pessoa muito sadia — está de férias em um vilarejo onde não há médico, sente uma terrível dor de cabeça e pergunta a esse sistema se pode tomar uma aspirina.

O sistema responderá que não, pois José não tem a aprovação do médico. Uma conclusão que deveria ter sido inferida não o foi, pois as regras das quais se segue a conclusão falharam por causa de uma premissa insatisfeita. Neste caso, o sistema deve generalizar adicionando alguma nova regra, ou relaxando algum dos antecedentes da regra já existente.

Teiresias não tenta achar as mudanças certas para especializar ou generalizar; ele pergunta ao especialista. A maior parte do tempo Teiresias utiliza menus ou perguntas que admitem respostas do tipo “sim”, “não”, “não sei”, etc. para dirigir o diálogo com o especialista. Ele só não adota este estilo quando deve assimilar uma nova regra, ou um novo conjunto como antecedente de uma regra já existente. Neste caso, deve aceitar entradas em linguagem natural. Segue-se um exemplo, extraído de uma interação com Teiresias, no qual o que o especialista digita está em itálico:

The new rule will be called *rule 383*

If : 1— *the patient's infection is primary-bacteremia*
and
2— *the site is one of the sterile sites*
and
3—
Then : *the category is enterobacteriaceae*

Teiresias traduz isto para sua representação interna e logo traduz novamente para o inglês, para verificar com o especialista se o significado é o mesmo. Se o especialista disser NO, então Teiresias tenta outras traduções alternativas, ou então convida o especialista a refazer a frase.

Um outro sistema que utiliza a estratégia de aprendizado por instrução é o sistema Nanoklaus [Hass 83], que constrói uma base de conhecimento hierárquica conversando com o usuário.

2.3.3 Aprendizado por Dedução

Neste tipo de aprendizado o aprendiz adquire os conceitos através de dedução sobre conhecimento já possuído. Em outras palavras, esta estratégia inclui qualquer processo no qual o conhecimento adquirido é o resultado de uma transformação que preserva a verdade sobre um conhecimento já armazenado — incluindo a execução de cálculos. Isto é, esta estratégia aplica argumentos dedutivos para fazer uma transformação da informação — mais detalhes sobre argumentos dedutivos encontram-se na seção 3.1.

Uma forma de dedução é o aprendizado por explicação, “explanation-based learning”, também chamado de aprendizado analítico [Carbonell 87], que transforma a definição abstrata de um conceito — não utilizável diretamente — em uma definição operacional, utilizando um conceito-exemplo como guia. Esta abordagem é descrita em detalhes no trabalho de Mitchell, Keller e Kedar-Cabelli [Mitchell 86].

A idéia geral utilizada nesta abordagem é construir explicações do porquê um objeto satisfaz uma definição funcional, e utilizar estas explicações para construir uma descrição estrutural. Na realidade, é uma forma de aprendizado por exemplos que faz uso de dedução ao invés de indução, utilizando apenas um exemplo positivo e teoria do domínio a ser aprendido. Esta teoria, em geral, envolve um conjunto de axiomas e regras de inferência do domínio para poder explicar como uma determinada instância satisfaz o conceito.

Devido à necessidade de possuir uma teoria do domínio que está sendo explorado, esta estratégia de aprendizado pode ser aplicada em menos casos do que o aprendizado por indução. Porém, ela possui algumas vantagens com relação aos métodos de aprendizado indutivos, pois consegue manusear dados com ruído (“noisy data”); além disso só é necessária uma instância positiva, sem precisar de nenhuma instância negativa do conceito. Assim, as pesquisas atuais tentam combinar aprendizado analítico com aprendizado por indução para poderem tratar aqueles casos onde o conhecimento do domínio é incompleto, intratável (“intractable”) ou inconsistente.

Atualmente, a abordagem de aprendizado por dedução (“explanation-based learning”) está sendo pesquisada intensivamente [Fayyad 89].

2.3.4 Aprendizado por Analogia

Os seres humanos, frequentemente, acham que uma determinada situação que estão vivendo é semelhante à alguma outra situação já experimentada. Se a situação anteriormente vivenciada sugeria um curso de ação apropriado, então parece certo tomar um rumo similar na situação presente.

Por exemplo, suponha que uma pessoa X esteja num jogo de xadrez e seu adversário realiza um determinado tipo de ataque. X lembra de uma situação anterior na qual ele utilizou sem sucesso esse mesmo tipo de ataque contra seu oponente. Existe um mapeamento

simples entre as duas situações:

agora as peças de X $\longrightarrow$ *antes as peças do oponente*

agora as peças do oponente $\longrightarrow$ *antes as peças de X*

Este tipo de mapeamento não precisa ser entre situações idênticas, porém, deve ser possível ignorar as diferenças. Algumas são fáceis de ignorar — como por exemplo a cor das peças — enquanto que outras devem ser levadas em consideração — como por exemplo a posição do rei e da rainha.

Uma dificuldade com situações análogas é que este tipo de mapeamento não é unicamente invertível — consequência do não determinismo da generalização. Para exemplificar, a seguir são considerados dois exemplos diferentes na resolução de problemas de geometria, encontrados em [Anderson 83] e [Anderson 79] respectivamente.

No problema A do primeiro exemplo — ilustrado na figura 2.2 — o estudante mostra que $\overline{RN} = \overline{OY}$ com o seguinte argumento:

$$\begin{aligned}\overline{RO} &= \overline{NY} \\ \overline{ON} &= \overline{ON} \\ \overline{RO} + \overline{ON} &= \overline{ON} + \overline{NY} \longrightarrow \overline{RN} = \overline{OY}\end{aligned}$$

Problema A:

Dado $\overline{RO} = \overline{NY}$

prove $\overline{RN} = \overline{OY}$

Problema B:

Dado $\overline{AB} > \overline{CD}$

prove $\overline{AC} > \overline{BD}$

Figura 2.2: Problemas de Geometria

Quando ao mesmo estudante é apresentado o problema B, ele lembra o exercício anterior e faz o seguinte mapeamento:

$$\begin{aligned}
 R &\longrightarrow A \\
 O &\longrightarrow B \\
 N &\longrightarrow C \\
 Y &\longrightarrow D \\
 = &\longrightarrow >
 \end{aligned}$$

tentando, por analogia, resolver o problema da seguinte forma:

$$\begin{aligned}
 \overline{AB} &> \overline{CD} \\
 \overline{BC} &> \overline{BC} \\
 \text{Então...}
 \end{aligned}$$

Este raciocínio é logo descartado por absurdo — proveniente da correspondência entre o sinal de “=” e o “>”. Para chegar na solução é necessário mapear somente “uma” ocorrência do “=” no “>”.

Assim, como é difícil estabelecer as correspondências válidas em um mapeamento por analogia, é mais difícil ainda estabelecer a sua inversão válida.

No segundo exemplo é apresentado ao estudante o mesmo problema A da figura 2.2, e logo a seguir o problema C ilustrado na figura 2.3. Neste caso, a resolução do problema por analogia sai diretamente fazendo a correspondência de retas com ângulos, bem como de igualdade com congruência:

Problema C:

Dado $\angle BAC = \angle DAE$

Mostre $\angle BAD = \angle CAE$

Figura 2.3: Problema de Geometria com Ângulos

$$\begin{aligned}
&\angle BAC = \angle DAE \\
&\angle CAD = \angle CAD \\
&\angle BAC + \angle CAD = \angle CAD + \angle DAE \rightarrow \angle BAD = \angle CAE
\end{aligned}$$

Destes experimentos é possível concluir que analogia é uma ferramenta útil na resolução de problemas; entretanto os problemas devem ser suficientemente semelhantes — como no último exemplo descrito acima — para que a analogia possa ser estabelecida diretamente. Caso contrário, o método por analogia serve apenas para “sugerir” soluções para o problema sendo tratado, para fornecer idéias iniciais sobre um conceito ou solução, que devem ser elaborados e depurados para serem convertidos numa solução final.

Os primeiros trabalhos nesta área são devidos a Winston [Winston 79a, 79b, 81, 83]. Já Carbonell sugere a abordagem da Analogia Transformacional [Carbonell 83a] — um método direto de transferência de informação de uma solução anterior para o novo problema — e o método da Analogia Derivacional [Carbonell 86] — que permite estabelecer analogias entre problemas mais distantes utilizando vários problemas anteriores para a resolução do novo problema.

2.3.5 Aprendizado por Indução

Um sistema inteligente deve ser capaz de classificar objetos, ou eventos, como sendo equivalentes para atingir determinadas metas. Por exemplo, para satisfazer sua sede, uma criança de dois anos sabe pedir “água”, ou apontar para a garrafa de refrigerante que está sobre a mesa. De maneira análoga, um sistema inteligente deve ser capaz de formar conceitos, isto é, classes de entidades agrupadas por algum princípio ou regra. Este processo é denominado aprendizado de conceitos.

Na estratégia de aprendizado por indução, o sistema aprendiz adquire os conceitos através da utilização de inferências indutivas realizadas sobre fatos fornecidos ou observados — inferências indutivas serão discutidas mais detalhadamente na seção 3.1. Dependendo dos fatos necessários para realizar o aprendizado serem fornecidos por uma fonte externa ou observados pelo sistema aprendiz, pode-se distinguir dois tipos diferentes de estratégias de aprendizado:

- i) aprendizado por exemplos
- ii) aprendizado por observação e descoberta.

i) Aprendizado por Exemplos

Nesta estratégia de aprendizado, o aprendiz induz a descrição do conceito através de processos de generalização e especialização realizados sobre exemplos e, opcionalmente,

contra-exemplos do conceito. Estes exemplos e contra-exemplos podem ser fornecidos por um professor ou pelo meio ambiente.

Por exemplo, suponha que se queira ensinar o conceito “ave”; ao sistema aprendiz são fornecidos vários exemplos de aves tais como papagaio, avestruz, pingüim, periquito, galinha, etc., com suas respectivas descrições, bem como vários contra-exemplos tais como borboleta, avião, morcego, etc. A partir destas instâncias, o sistema pode induzir, por exemplo, que “aves” são animais com penas, ovíparos e vertebrados. Isto é, a tarefa do aprendiz consiste em determinar uma descrição geral de um conceito analisando exemplos individuais deste conceito.

ii) Aprendizado por Observação e Descoberta

Descoberta é uma forma restrita de aprendizado em que uma entidade adquire conhecimento sem o auxílio de alguém que já o possua. Nesta estratégia, o aprendiz analisa e observa entidades e determina que alguns subconjuntos destas entidades podem ser agrupados em certas classes, isto é, conceitos. Esta estratégia também é denominada “aprendizado não supervisionado”, pois nela não há um professor que supervisione o processo de aprendizado.

Um dos sistemas mais conhecidos neste paradigma é o programa AM de Lenat [Lenat 76, 82]. AM é capaz de descobrir conceitos na área de teoria dos números e de geometria. Ele começa a operar com o conhecimento de alguns poucos conceitos matemáticos básicos, como por exemplo, o conceito de conjunto, de “bag” — conjunto que pode ter elementos repetidos —, etc. A partir deste conhecimento inicial, ele próprio gera exemplos destes conceitos, examina estes exemplos procurando neles características semelhantes, gera novos conceitos e propõe novas relações entre conceitos. Este sistema provou ser capaz de fazer descobertas não triviais no domínio matemático.

Uma vez dados os conceitos iniciais, AM opera sem nenhuma intervenção humana — ele somente pergunta os nomes que devem ser atribuídos à suas descobertas. Neste sentido, ele faz descobertas por si próprio, embora ele redescubra muitos conceitos já conhecidos pelos matemáticos, tais como o de números naturais, adição, multiplicação, quadrados perfeitos, números pares e ímpares, números primos, etc. Por outra parte, AM também consegue descobrir importantes teoremas matemáticos, como por exemplo o Teorema de Euclides de Fatoração Única — existe uma única forma de fatorar um número composto em números primos — ou a Conjectura de Goldbach — todo número par maior que 2 é a soma de dois números primos.

O conhecimento de AM é armazenado em *frames*³. O frame é um tipo de representação estruturada de conhecimento que é utilizado geralmente para descrever classes de objetos. Ele consiste de uma coleção de *slots*³ que descrevem características dos objetos. Os slots podem ser preenchidos por outros frames descrevendo outros objetos. Associado a cada

³Uma possível tradução para *frame* é quadro, e para *slot*, escaninho. Neste trabalho serão mantidos os nomes originais.

Nome: Números Primos

Definições: Origem: Número-de-divisores-de $f(x)=2$

Cálculo-de-Predicados: $\text{Primo}(x) \Leftrightarrow (\forall z)(z|x \Rightarrow (z = 1 \otimes z = x))$

Iterativo: (para $x > 1$) : para i de 2 a $\sqrt{x}$, iXx

Exemplos: 2, 3, 5, 7, 11, 13, 17

Fronteira: 2, 3

Fronteira-falhas: 0, 1

Falhas: 12

Generalizações: Número, Número com um número par de divisores

Especializações: Ímpares Primos, Pares Primos, Primos Somáveis singularmente

Conjecturas: Fatoração única de conjectura de Goldbach, Extrema de Número-de-divisores-de

Intus: Uma metáfora do efeito dos primos é que são blocos de construção de todos os números

Analogias: Números divisores maximais são os extremos opostos do Número-de-divisores-de

Fatorar um grupo não simples em grupos simples

Interesse: Conjecturas ligando primos a vezes, a divisores de, a operações relacionadas

Valor: 800

Figura 2.4: Um Frame de AM: o Conceito de Número Primo, [Rich 83].

slot poderá haver um conjunto de condições que deverá ser satisfeito por qualquer slot a ele associado. Cada slot também poderá ser preenchido com um valor por “default” ou com informação procedimental.

Em AM cada frame representa um conceito. Por exemplo, o frame mostrado na figura 2.4 representa o conceito de número primo. AM também conta com um corpo de 242 regras heurísticas da forma “If ... Then”. Estas regras representam indicações a respeito de atividades que, provavelmente, levarão à descobertas interessantes. Exemplos de algumas das regras utilizadas por AM são mostrados a seguir:

- Se alguns exemplos (mas não a maioria) de algum conceito X também forem exemplos de outro conceito Y, então crie um novo conceito representando a intersecção de X com Y.

- Se poucos exemplos de um conceito X forem encontrados, então acrescenta-se à agenda a tarefa de encontrar uma generalização de X.

A agenda mencionada na regra anterior serve para controlar o processo de descoberta. Quando as heurísticas sugerem uma tarefa, esta é colocada em uma agenda central, juntamente com a razão de sua sugestão — o slot *Interesse* na figura 2.4 — e a ênfase com a qual foi indicada — o slot *Valor* na figura 2.4.

AM opera em ciclos, e em cada ciclo escolhe a tarefa mais promissora da agenda — aquela que tem maior *valor* — e a executa. Ele é capaz de formar hipóteses com base num pequeno número de exemplos, gerar mais exemplos, e logo testar as hipóteses neste conjunto maior de exemplos para ver se elas ainda são válidas.

Uma das principais limitações do sistema AM é que, apesar de poder descobrir novos conceitos, ele não pode descobrir novas regras heurísticas; isto é, ele não pode aprender a melhorar seu desempenho modificando suas regras.

EURISKO [Lenat 83] é uma extensão de AM projetado para contornar estas dificuldades. EURISKO trata as heurísticas como conceitos que podem ser criados e modificados pelos mesmos tipos de processos, como o são os conceitos no domínio de atividade. EURISKO foi aplicado tanto para o domínio de teoria de números como para diversos outros domínios, tais como o projeto de frotas navais e o projeto de VLSI — Very Large Scale Integration. Nestas duas áreas o desempenho de EURISKO surpreendeu os especialistas da área, ao propor idéias inovadoras que tiveram resultados muito bons.

2.4 Histórico da Pesquisa de Aprendizado de Máquina

O interesse nas pesquisas de AM começou com a criação do primeiro computador, com diferentes graus de intensidade dependendo do período considerado. No período atual, devido em parte às grandes melhorias incorporadas ao hardware das máquinas, há um trabalho intenso de pesquisa nesta área [McDonald 89].

É possível distinguir três períodos principais na história de Aprendizado de Máquina, cada um deles centralizado em um paradigma diferente [Carbonell 83]. Eles são:

- modelagem neural e sistemas paramétricos;
- aprendizado geral de conceitos por representações simbólicas;
- sistemas de aprendizado de conhecimento intensivo explorando várias tarefas de aprendizado.

2.4.1 Modelagem Neural e Sistemas Paramétricos

Este paradigma abarcou o período de fim da década do 50 até inícios do 70. Sua principal característica foi o interesse em construir Sistemas de Aprendizado de propósito geral, com pouca ou nenhuma estrutura inicial ou conhecimento específico do domínio a ser aprendido. Estes sistemas foram genericamente referenciados como Redes Neurais ou Sistemas de Organização Própria, pois eles se auto modificavam para se adaptar ao meio ambiente. O aprendizado em tais sistemas consistia de mudanças incrementais na probabilidade de que elementos tipo neurônios — as unidades básicas do sistema — transmitissem um sinal.

Devido à natureza primitiva da tecnologia dos computadores daquela época, a maioria das pesquisas conduzidas sob este paradigma foram teóricas ou envolviam a construção de sistemas de hardware. Como exemplos deste tipo de sistemas podem ser citados PERCEPTONS [Rosenblatt 58] [Minsky 72], PANDEMONIUM [Selfridge 59] e ADELAINE [Widrow 62].

Em paralelo à pesquisa sobre Modelagem Neural, pesquisadores em teoria de controle desenvolveram sistemas de controle adaptativo capazes de ajustar automaticamente seus parâmetros, com o objetivo de manter sua performance estável na presença de diversos distúrbios. Um dos sistemas mais conhecidos utilizando tais técnicas é o programa de damas de Samuel [Samuel 63], descrito anteriormente.

Resultados práticos apontados pela Modelagem Neural foram de sucesso limitado. Altas expectativas articuladas em vários trabalhos iniciais não foram alcançadas, e a pesquisa sob este paradigma começou a declinar. Porém, avanços recentes na tecnologia de VLSI e de Projeto Auxiliado por Computador implicam, hoje em dia, na facilidade da construção em massa de máquinas paralelas. Isto tem contribuído muito para reavivar o interesse em modelos computacionais inspirados em redes neurais, incentivando nos últimos anos o resurgimento da pesquisa nesta área [Levine 89].

2.4.2 Aprendizado Geral de Conceitos por Representações Simbólicas

Um segundo paradigma começou a emergir no início da década de 60, baseado no trabalho de psicólogos e pesquisadores de Inteligência Artificial em modelos de Aprendizado Humano. Como o aprendizado é um processo muito complexo e dificultoso, muitos pesquisadores acharam melhor dar uma ênfase maior ao estudo em profundidade de problemas mais restritos de aprendizado; conseqüentemente, este período foi muito fértil na produção de trabalhos de Aprendizado de Conceitos por Exemplos.

Este paradigma utiliza representações de estrutura gráfica ou lógica, ao invés de métodos estatísticos ou numéricos. Sistemas aprendem descrições simbólicas representando conhecimento de mais alto nível, e fazem fortes suposições estruturais sobre os conceitos a

serem adquiridos. A ênfase é dada ao descobrimento de princípios gerais de aprendizado, independentemente do domínio de aplicação.

Um trabalho de influência neste paradigma é o sistema de Aprendizado de Winston [Winston 75], que aprende conceitos do mundo dos blocos. Em paralelo, diferentes abordagens para aprender conceitos estruturais a partir de exemplos surgiram, incluindo uma família de programas de Aprendizado Indutivo baseada em lógica, a família AQVAL de Michalski [Michalski 73,76,78].

2.4.3 Sistemas de Aprendizado de Conhecimento-intensivo Explorando Várias Tarefas de Aprendizado

O terceiro paradigma representa o período mais recente da pesquisa em AM, iniciado em meados de 70. Pesquisadores ampliaram seu interesse além do Aprendizado de Conceitos a Partir de Exemplos, e começaram a investigar um espectro mais amplo de métodos de Aprendizado, tais como Aprendizado por Instrução [Davis 82], Aprendizado por Analogia [Winston 79a], e Aprendizado por Descoberta e Classificação [Mitchell 83]. Uma outra característica deste paradigma é a importância dada ao conhecimento específico do domínio, e a incorporação de habilidades para gerar e selecionar tarefas no sistema de aprendizado. Esta última característica pode ser traduzida como a incorporação no sistema de heurísticas para, controlar seu foco de atenção para gerar tarefas de Aprendizado, propor experimentos para colher dados de treinamento, bem como escolher, entre os possíveis conceitos a serem adquiridos, os mais promissores. Este paradigma está baseado na idéia de que, para obter novos conhecimentos, um sistema já deve possuir algum conhecimento inicial.

Vários trabalhos influenciaram fortemente as pesquisas deste período, entre eles merecem ser citados:

- O programa META-DENDRAL [Buchanan 78], que gera regras explicando dados de Espectrometria de Massa para uso no sistema DENDRAL [Lindsay 80]. DENDRAL é um Sistema Especialista que infere a estrutura de compostos químicos orgânicos utilizando dados do espectrograma de massa e da ressonância magnética nuclear (NMR).
- O sistema AM [Lenat 82] que gera seus próprios exemplos para descobrir conceitos matemáticos; o sistema Eurisko [Lenat 83] —uma continuação do sistema AM— que trabalha em diversas áreas e aprende, não só os conceitos específicos do domínio, bem como quais as heurísticas mais apropriadas para trabalhar nesse domínio.
- O trabalho de Michalski e Stepp [Michalski 80a] [Michalski 83b], que continuaram pesquisando sistemas da família AQ, os quais realizam aprendizado por observação e classificação taxonômica a partir de exemplos.

- O algoritmo ID3, idealizado por Quinlan [Quinlan 79,83], que acha regras nos exemplos dados construindo árvores de decisão. Este trabalho deu origem a uma série de sistemas que, incorporando diversas melhoras, foram comercializados e são utilizados hoje em dia na construção de Sistemas Especialistas [Mowforth 86].

2.5 Considerações Finais

Neste capítulo foi realizada uma revisão geral de AM, isto é, o que é AM, quais os objetivos de se estudar AM, classificações das estratégias de AM e um pequeno histórico das pesquisas nesta área.

Entre as motivações para o estudo de AM foi enfatizada a de automatizar a aquisição do conhecimento para facilitar a construção de Sistemas Baseados em Conhecimento. É de interesse do grupo de pesquisa em IA do ICMSC integrar os resultados obtidos neste trabalho, com os que já vem sendo desenvolvidos referentes à construção de Núcleos de Sistemas Especialistas [Monard 89], [Monard 90], [Rodrigues 90a], [Rodrigues 90b], [Rodrigues 90c].

Ainda não existe na literatura um consenso a respeito da classificação das estratégias de aprendizado. Devido a isso, neste capítulo foram discutidas brevemente algumas classificações existentes, adotando uma em particular para descrever cada um dos tipos de AM.

Capítulo 3

Aprendizado de Conceitos por Indução

Este capítulo está dividido em três seções. Na primeira delas é abordado o problema da indução lógica, suas características e diferenças com relação à dedução lógica. Na segunda seção é apresentado o paradigma do Aprendizado por Indução e, na última seção, são descritas as estratégias de Aprendizado Indutivo.

3.1 Argumentos Dedutivos *versus* Argumentos Indutivos

A lógica trata de argumentos e inferências. Um de seus propósitos básicos é apresentar métodos capazes de identificar os argumentos logicamente válidos. Um argumento é uma coleção de enunciados que estão relacionados uns com os outros [Salmon 81]. Um destes enunciados é a conclusão e os outros, um ou mais, constituem a evidência corroborativa e são chamados de premissas.

Considere o seguinte exemplo de argumento lógico, independentemente dos filósofos comerem ou não espaguete:

Todos os filósofos comem espaguete
Kant é um filósofo
Kant come espaguete

Neste caso, o argumento está formado por três enunciados. Os dois primeiros enunciados são as premissas e o terceiro é a conclusão. No que se segue, os argumentos serão apresentados de uma forma padronizada, escrevendo as premissas em primeiro lugar e a seguir a conclusão, identificando-a com o símbolo $\Rightarrow$.

Todos os filósofos comem espagete
Kant é um filósofo
 ⇒ *Kant come espagete*

Salmon divide os argumentos lógicos em duas grandes classes: os dedutivos e os indutivos. A seguir, são mostrados exemplos de cada um deles:

argumento dedutivo:

Todo mamífero tem coração
Todas as baleias são mamíferos
 ⇒ *Todas as baleias têm coração*

argumento indutivo:

Todas as baleias que foram observadas
tinham coração.
 ⇒ *Todas as baleias têm coração.*

Há certas características básicas, discutidas a seguir, que distinguem os argumentos dedutivos dos argumentos indutivos:

Argumentos Dedutivos	Argumentos Indutivos
Se todas as premissas são verdadeiras a conclusão é verdadeira	Se todas as premissas são verdadeiras, a conclusão é provavelmente verdadeira — mas não necessariamente verdadeira — com exceção dos argumentos matemáticos indutivos.
Toda a informação do conteúdo factual da conclusão já está, pelo menos implicitamente, nas premissas.	A conclusão contém informação que não está nem implicitamente nas premissas.

Argumentos dedutivos e argumentos indutivos atendem à finalidades diversas. O argumento dedutivo destina-se a deixar explícito o conteúdo das premissas, enquanto que o argumento indutivo destina-se a ampliar o alcance de nossos conhecimentos. A relação que há entre a generalização científica e a evidência observacional sustentadora é de tipo indutivo.

A validade de um argumento dedutivo depende, exclusivamente, da relação que se estabelece entre as premissas e a conclusão. Dizer que um argumento dedutivo é válido, significa

dizer que as premissas estão de tal modo relacionadas com a conclusão que a conclusão precisa ser verdadeira quando as premissas são verdadeiras. Validade é uma propriedade dos argumentos — que são uma coleção de enunciados — e não uma propriedade dos enunciados isoladamente considerados. Verdade, por outro lado, é uma propriedade de enunciados isolados, não de argumentos.

3.1.1 Indução

Os argumentos indutivos, ao contrário do que sucede com os dedutivos, levam à conclusões cujos conteúdos excedem o das premissas. É esse traço característico da indução que torna os argumentos indutivos indispensáveis para a fundamentação de uma significativa porção de nossos conhecimentos. Entretanto, é esse mesmo fato que levanta questões extremamente complicadas, dificultando a análise de resultados obtidos com auxílio de métodos indutivos.

Ao contrário do que sucede com um argumento dedutivo e válido, um argumento indutivo e correto pode, perfeitamente, admitir uma conclusão falsa, ainda que suas premissas sejam verdadeiras. Mesmo não podendo garantir que a conclusão de um argumento seja verdadeira quando as premissas são verdadeiras, pode-se afirmar que as premissas de um argumento indutivo correto sustentam ou atribuem certa verossimilhança à sua conclusão. Quando as premissas de um argumento indutivo são verdadeiras, o melhor que pode ser dito é que a sua conclusão é *provavelmente* verdadeira.

Como já mencionado acima, uma exceção disto é a indução matemática. Em um argumento matemático indutivo correto, partindo de premissas verdadeiras obtém-se, invariavelmente, conclusões verdadeiras.

Há certos enganos que podem tornar os argumentos indutivos completamente inúteis ou inúteis de um ponto de vista prático. Enganos desse gênero são denominados *falácias indutivas*. Quando um argumento indutivo é falaz, as premissas não sustentam a conclusão. Entre os argumentos indutivos corretos, porém, pode-se cogitar um grau de sustentação ou de apoio. As premissas de um argumento indutivo correto podem tornar a conclusão extremamente provável, moderadamente provável ou provável com certo grau de certeza.

Há uma segunda diferença entre os argumentos indutivos e os dedutivos. Dado um argumento dedutivo válido, é possível acrescentar novas premissas, colocando-as com as já existentes, sem afetar a validade do argumento. Em contraste, o grau de sustentação que as premissas de um argumento indutivo conferem à conclusão pode ser alterado por evidências adicionais, acrescentadas ao argumento sob a forma de premissas novas que figurem ao lado das premissas inicialmente consideradas. Como a conclusão de um argumento indutivo pode ser falsa mesmo quando as premissas forem verdadeiras, a evidência adicional, admitindo que relevante, pode nos capacitar a determinar, com mais precisão, se a conclusão é de fato verdadeira. A evidência adicional pode afetar o grau de sustentação da conclusão. A seguir, este caso é ilustrado com um exemplo.

Seja o seguinte argumento:

- a) *A grande maioria dos homens brasileiros de 35 anos viverá ainda por mais três anos.*
José é um brasileiro de 35 anos.
 $\Rightarrow$ *José viverá ainda por mais três anos.*

Suponha-se, porém, que José tenha contraído o vírus da AIDS. É possível, então, construir este outro argumento:

- b) *A grande maioria das pessoas afetadas pelo vírus da AIDS não viverá por mais três anos.*
José está afetado pelo vírus da AIDS.
 $\Rightarrow$ *José não viverá por mais três anos.*

Todas as premissas, tanto em a) como em b), poderiam ser verdadeiras; elas não são incompatíveis. Entretanto, as conclusões são contraditórias. Este tipo de situação só pode estar presente em argumentos indutivos. Não adianta acrescentar *provavelmente* nas conclusões, pois dizer que José provavelmente viverá mais três anos e provavelmente não viverá mais três anos continua sendo contraditório. Ou seja, não é possível simplesmente combinar as premissas de a) e b) para formar um novo argumento. Considerando agora as seguintes premissas:

- c) *A grande maioria dos homens brasileiros de 35 anos viverá ainda por mais três anos.*
A grande maioria das pessoas afetadas pelo vírus da AIDS não viverá por mais três anos.
José é um brasileiro de 35 anos afetado pelo vírus da AIDS.
 $\Rightarrow$??

pode-se observar que delas não se obtém nenhuma conclusão, nem mesmo indutivamente, acerca da sobrevivência de José. Dispõe-se, porém, de evidências adicionais. Sabe-se que os brasileiros de 35 anos não são a exceção à regra: a grande maioria dos brasileiros de 35 anos, se afetados pelo vírus da AIDS não vive por mais três anos. Com isto, pode-se construir o seguinte argumento:

- A grande maioria dos brasileiros de 35 anos, afetados pelo vírus da AIDS não viverá por mais três anos.*
José está afetado pelo vírus da AIDS.
 $\Rightarrow$ *José não viverá por mais três anos.*

Admitindo-se que as premissas acima encerram toda a evidência relevante disponível, pode-se aceitar a conclusão do argumento. Pode-se ver, agora, o alcance da observação

feita anteriormente, a qual salientava que a evidência adicional é relevante para os argumentos indutivos, e não o é para os argumentos dedutivos.

É possível afirmar que a conclusão de um argumento indutivo é aceitável se:

1. as premissas são verdadeiras
2. o argumento é formalmente correto
3. as premissas do argumento englobam toda a evidência relevante disponível. Esta condição é chamada de *condição da evidência total*.

Existem diversos tipos de argumentos indutivos, como por exemplo: indução por enumeração, silogismo estatístico, argumento de autoridade e argumento contra o homem, entre outros. A seguir, é descrito apenas o argumento de indução por enumeração, por ser este um dos argumentos indutivos usualmente mais utilizado nos processos de Aprendizado de Máquina.

3.1.2 Indução por Enumeração

O argumento indutivo mais simples é o da indução por enumeração. Em argumentos desse tipo, chega-se à uma conclusão acerca de todos os elementos de um conjunto partindo de premissas que se referem à características observadas em alguns elementos daquele conjunto. O seguinte argumento é usado para exemplificar este conceito:

Admita-se que num saco de grãos de soja, depois de misturar bem os grãos, retira-se do saco uma amostra formada com grãos de diferentes partes do volume considerado. Um exame desta amostra atesta que todos os grãos da amostra são do tipo A. Conclui-se que todos os grãos contidos no saco são do tipo A.

que pode ser descrito da seguinte forma:

Todos os grãos da amostra retirada do saco são do tipo A.
 $\Rightarrow$ *Todos os grãos no saco são do tipo A.*

Este argumento é uma generalização construída a partir de uma certa amostra. Não é essencial que a conclusão de uma indução enumerativa assuma a forma *todos os F são G*. Pode acontecer que a conclusão afirme que *uma dada percentagem dos F são G*. A forma geral deste argumento é:

Z % dos elementos observados em F são G.
 $\Rightarrow$ *Z % dos F são G.*

É evidente que a indução por enumeração pode, facilmente, levar a conclusões falsas, mesmo que as premissas sejam verdadeiras. O que se pode tentar fazer é construir os correspondentes argumentos indutivos de modo a minimizar a probabilidade de chegar a conclusões falsas quando as premissas são verdadeiras. Em relação aos argumentos indutivos por enumeração, há duas maneiras específicas de diminuir a possibilidade de erro, isto é, há duas falácias indutivas que devem ser evitadas. Elas são:

- a estatística insuficiente
- os desvios estatísticos.

3.1.3 Estatística Insuficiente

A falácia da estatística insuficiente consiste em efetuar uma generalização indutiva antes de contar com um número de dados suficientes para sustentar a generalização. É uma espécie de conclusão “apressada” — devido ao fato de contar com poucos elementos na amostra.

Considere-se que os resultados de uma pesquisa eleitoral, em um país de porte médio, estejam baseados na entrevista a 5 pessoas. Essa amostra não é suficientemente ampla para realizar uma generalização digna de crédito. Não é possível fazer uma projeção da intenção de voto de toda a população a partir desses dados. Um número maior de pessoas deveria ser entrevistado para serem obtidos resultados mais dignos de confiança.

O número suficiente de casos a examinar pode ser determinado, entre outros, pela experiência prévia adquirida no estudo do problema particular.

3.1.4 Desvios Estatísticos

É também importante realizar um processo de seleção das observações, a fim de evitar distorções nos resultados. Desejando-se generalizações indutivas merecedoras de confiança, elas devem se apoiar em amostras representativas. A falácia dos desvios estatísticos, também chamada de *estatística tendenciosa*, consiste, justamente, em basear a generalização em uma determinada amostra que não é representativa.

No caso da pesquisa eleitoral, suponha-se que sejam entrevistadas um número suficiente de pessoas, mas todas elas pertencentes a uma mesma classe social e residentes numa mesma cidade. Pode-se afirmar que os resultados desta pesquisa serão “tendenciosos”. O ideal seria entrevistar alguns eleitores da zona rural e alguns da zona urbana, alguns da classe alta, alguns da classe média, e outros da classe operária, procurando também variar as regiões do país onde se fariam as entrevistas. Dessa maneira, a variedade da amostra poderia refletir a situação geral da população.

3.2 Aprendizado por Indução

A habilidade das pessoas de fazer generalizações corretas a partir de poucos fatos espalhados, ou descobrir padrões em coleções de observações que parecem caóticas, é um tópico de pesquisa de grande interesse. Esta habilidade é atingida através de um processo denominado *aprendizado indutivo*, isto é, inferência indutiva realizada sobre fatos fornecidos por um professor ou pelo meio ambiente.

Como mencionado anteriormente, este tipo de aprendizado abrange duas estratégias: aprendizado por exemplos (também denominado aprendizado de conceitos por exemplos) e aprendizado por observação e descoberta.

A tarefa do aprendizado por exemplos é induzir descrições gerais dos conceitos a partir de instâncias específicas desses conceitos. No aprendizado por observação e descoberta, a preocupação está centrada em estabelecer novos conceitos ou teorias que caracterizam fatos dados. Esta última estratégia de aprendizado inclui tópicos tais como: formação de teoria automatizada, descoberta de relações entre os dados e construção automática de taxonomia.

Neste ponto é interessante fazer uma distinção. Geralmente, na literatura, o termo *aprendizado de conceitos* — como visto na seção 2.3.5 — é utilizado como significando apenas *aprendizado de conceitos por exemplos*. Na realidade, aprendizado de conceitos tem um escopo mais amplo, abrangendo diversas estratégias de aprendizado, uma das quais é aprendizado por exemplos. Um dos motivos da utilização incorreta desta nomenclatura é histórico. Acontece que aprendizado de conceitos por exemplos foi uma das primeiras estratégias a ser estudada, e é conhecido bastante sobre ela. Aprendizado por exemplos e aprendizado por observação e descoberta — aprendizado indutivo em geral — são formas fundamentais de aprendizado de conceitos.

A descrição de um conceito deve ser suficiente para distinguir este conceito de outros conceitos. Entidades individuais da classe representada pelo conceito são chamadas de *instâncias* ou *exemplos do conceito*, enquanto que aqueles elementos que não pertencem à classe são denominados de *contra-exemplos* ou *instâncias negativas*. Por exemplo, o pinguim é uma instância do conceito de “ave”, mas é um contra-exemplo de “pássaro”.

A utilização de contra-exemplos é particularmente útil para delimitar o escopo de um conceito, isto é, os contra-exemplos servem para evitar supergeneralização. O que diferencia “pássaro” de “ave” para o pinguim ser um contra-exemplo do primeiro conceito e uma instância positiva do segundo? Em especial, são importantes aqueles contra-exemplos que tem características em comum com as instâncias da classe. Winston [Winston 75] denomina este tipo de exemplos como “*near-miss*” ou “*quase-acerto*”, e ressalta a sua função de reduzir a busca de descrições de conceitos. A figura 3.1 mostra um exemplo e um quase-acerto do conceito de arco manipulado pelo programa de Winston, que opera no mundo dos blocos. Querendo aprender o que é um arco pode-se dizer que:

Arco

Quase arco

Figura 3.1: Conceitos no Mundo dos Blocos.

Um arco é um objeto composto por três elementos: um bloco horizontal apoiado sobre dois blocos verticais.

O primeiro exemplo da figura é um arco, o segundo exemplo tem semelhanças mas não é um arco. Entretanto, a descrição anterior é satisfeita pelos dois exemplos mostrados na figura. Este fato leva a revisar a descrição gerada.

Para excluir esta instância negativa, pode-se adicionar a condição dos blocos verticais não estarem apoiados um no outro:

Um arco é um objeto composto por três elementos: um bloco horizontal apoiado sobre dois blocos verticais entre os quais deve existir uma separação.

Dessa forma, o quase-acerto de arco mostrado na figura 3.1 serviu para ajudar a definir corretamente o conceito de arco.

A suposição de que um conceito é uma classe de equivalência implica que cada instância é igualmente representativa da classe, e que a descrição do conceito tem limites precisos. Isto é, uma entidade dada é um exemplo do conceito que está sendo aprendido, ou não é um exemplo deste conceito; ela não pode ser “mais ou menos” um exemplo do conceito. Tal idealização facilita muito a pesquisa nesta área.

Porém, ela ignora muitos aspectos importantes da noção humana de “conceito”. Os conceitos humanos, exceto para aqueles casos que ocorrem predominantemente nas ciências — conceitos como número primo, círculo, mamífero, etc. — são estruturas com limites flexíveis ou imprecisos. Exemplos distintos podem ser representativos de um mesmo conceito “em diferentes graus”, e esta representatividade pode variar dependendo do contexto. Qual é a definição do conceito ditador? Franco, Getulio Vargas, Fidel, Hitler e Perón — dependendo do interlocutor — representam este conceito em diferentes graus.

Certamente, para muitos cubanos Fidel é um contra-exemplo de ditador. É por este motivo que em AM considera-se que todos os exemplos são igualmente representativos da classe, e que os conceitos têm limites bem precisos. Porém, relaxar estas restrições no estudo de AM pode ser uma linha interessante de pesquisa a ser explorada [Michalski 87].

3.2.1 Paradigma Indutivo

O processo de aprendizado por indução envolve operações de generalização, transformação, correção e refinamento das representações de conhecimento. Com exceção de alguns poucos casos, o conhecimento adquirido, em princípio, não pode ser completamente validado.

Em contraste com a dedução, as premissas iniciais da indução são fatos específicos, e não necessariamente axiomas gerais. O objetivo da inferência indutiva é formular possíveis asserções gerais que expliquem os fatos dados e sejam capazes de prever novos fatos. Dos dois aspectos da inferência indutiva — geração de hipóteses plausíveis e sua validação — o primeiro é de maior interesse na pesquisa de aprendizado indutivo.

A inferência indutiva é, usualmente, caracterizada como o raciocínio que leva do específico ao geral, ou do particular ao universal. Porém, esta definição é muito vaga. Michalski [Michalski 83a] distingue cada um dos componentes do processo indutivo e suas propriedades da seguinte forma:

Dados:

- a) premissas P (que podem ser fatos, objetos, situações, processos, etc)
- b) uma tentativa de afirmação indutiva, isto é, uma possível hipótese que em alguns casos pode não existir
- c) conhecimento de fundo, constituído por algum conhecimento relevante do domínio do problema, bem como suposições e restrições impostas às premissas e tentativas de indução

Achar: uma afirmação indutiva H (hipótese) que tautológica ou fracamente implique as premissas P e satisfaça o conhecimento de fundo.

Uma hipótese H , no contexto do conhecimento de fundo, implica fracamente ou tautologicamente as premissas P , se utilizando este conhecimento de fundo e as regras de inferência, pode ser mostrado que as premissas são uma consequência lógica da hipótese:

$$\text{Hipótese} \ \& \ \text{Conhecimento_de_fundo} \Rightarrow \text{Premissas}$$

Seja por exemplo:

Premissas: À noite Mich é pardo. À noite Napoleão é pardo. À noite Popi é pardo.

Conhecimento de Fundo: Mich, Napoleão e Popi são gatos.

Hipótese: À noite todos os gatos são pardos.

Note que, como já foi mencionado anteriormente, a inferência indutiva não preserva verdade, ou seja, suas conclusões podem ser incorretas apesar das premissas serem corretas. A inferência dedutiva é uma operação que preserva a verdade, isto é:

$$\begin{array}{ll} \text{Se} & H \Rightarrow P \text{ é verdadeiro,} \\ \text{e} & H \text{ é verdadeiro} \\ \text{então} & P \text{ é verdadeiro.} \end{array}$$

Em contraste, derivar H de P (inferência indutiva) não preserva a verdade mas preserva falsidade, isto é, se algumas premissas falsificam P, então elas também falsificam H. Assim, se não for verdade que à noite Popi é pardo, então também não será verdade que à noite todos os gatos são pardos.

Uma hipótese H implica as premissas P tautologicamente se P é consequência lógica de H, isto é, se $H \Rightarrow P$ é verdade em todas as suas interpretações. A condição que H implica P fracamente significa que as premissas P não são certas, mas somente possíveis ou consequências parciais de H. Nesse caso, trabalha-se com hipóteses parciais ou probabilísticas, que acontecem para algumas mas não todas as premissas. No que segue, faz-se referência à hipóteses que implicam premissas tautologicamente.

Sendo P e H como definidos anteriormente (P conjunto de premissas, fatos ou declarações observadas, e H a hipótese ou afirmação indutiva) H pode ser visto como um conjunto de regras de reconhecimento de conceitos. Isto é, a descrição de um conjunto de condições, tal que, quando estas são satisfeitas por um objeto, então este objeto é considerado uma instância da classe K descrita por H.

Uma afirmação indutiva deve satisfazer às condições de *completeza* e *consistência*. Um conceito é consistente com respeito aos exemplos, se ele cobre alguns ou todos os exemplos positivos e nenhum dos exemplos negativos. A descrição de um conceito é completa com respeito aos exemplos se ela cobre todos os exemplos positivos.

Para um dado conjunto de premissas, pode ser gerado um grande número de hipóteses que implique nessas premissas. O conjunto de todas aquelas possíveis hipóteses, isto é, aquelas hipóteses que são completas e consistentes com relação aos exemplos dados, são chamadas de *hipóteses candidatas*. Conhecimento de fundo e um critério de preferência são necessários para fornecer restrições que possibilitem reduzir o grande número de possíveis hipóteses candidatas.

Supondo agora que no exemplo anterior se tivesse o seguinte conhecimento de fundo:

1. Mich, Napoleão e Popi são gatos.
2. Os gatos são mamíferos.
3. Mamíferos são animais.

então, a partir destes dados, podem ser criadas as seguintes hipóteses candidatas:

1. À noite todos os gatos são pardos.
2. À noite todos os mamíferos são pardos.
3. À noite todos os animais são pardos.

Se o critério de preferência estabelece adotar a hipótese que seja mais específica então, neste caso, será adotada a hipótese número 1.

3.2.2 Tipos de Descrições

Três tipos principais de descrição de conceitos podem ser encontrados nos algoritmos de Aprendizado por Exemplos [Diettrich 83]:

- descrições características;
- descrições discriminantes e
- descrições taxonômicas .

Certamente as pessoas não utilizam apenas estas formas restritas para a formação de conceitos, nem todos os conceitos humanos são possíveis de serem expressos por esses três tipos de descrições. Essas descrições foram escolhidas como abstrações e simplificações úteis das estruturas humanas de conceito, que podem servir como guia para a pesquisa de AM.

Uma descrição característica acontece quando o conceito é descrito em termos dos atributos que distinguem exemplos que são instâncias da classe (exemplos positivos) de exemplos que não o são (contra-exemplos). Não é requerido nenhum conhecimento de qualquer outro conceito que exista no universo: somente a pertinência com respeito ao conceito dado. A descrição característica (generalização conjuntiva) é uma expressão que satisfaz a condição de completeza. Geralmente é a conjunção de algumas propriedades simples, comuns a todos os objetos da classe. Este tipo de descrição tenta discriminar a classe dada de qualquer outra possível classe. Por exemplo, uma cadeira pode ser descrita como um objeto que serve para se sentar e tem um encosto.

Em uma descrição discriminante, um número finito de possíveis conceitos são descritos em termos dos atributos, que são suficientes para classificar exemplos como membros de um dos conceitos. A descrição discriminante satisfaz as condições de completeza e consistência. Ela especifica uma ou várias formas alternativas para distinguir a classe dada, de um número fixo de outras classes. As mais interessantes são as *descrições discriminantes mínimas*, que são as mais curtas — isto é, obtidas com o menor número de descritores —, e são expressões distinguindo todos os objetos da classe dada dos objetos das outras classes. Tais descritores tentam especificar a informação mínima suficiente para identificar a classe dada entre um número fixo de outras classes.

Por exemplo, no final de uma partida de xadrez pode-se classificar, em teoria, todos os possíveis movimentos em três classes: vitória para as peças brancas, vitória para as peças pretas ou empate. Pode-se construir uma árvore de decisão para classificar os possíveis movimentos nestas três classes. Esta árvore de decisão será uma descrição discriminante. Sistemas como o ID3 — Iterative Dichotomizer Tree — que utilizam árvores de decisão para descrever conceitos, constroem descrições discriminantes. Este sistema será visto em detalhe no capítulo 4.

Quando o conceito representa uma hierarquia ou uma taxonomia de todos os exemplos conhecidos, a descrição do conceito é dita taxonômica. Tal descrição agrupa objetos similares e mostra as possíveis relações entre membros do grupo. Por exemplo, a descrição da família dos “animais carnívoros” é uma descrição taxonômica; ela inclui os felinos, caninos, etc.

3.2.3 Aprendizado de um Único Conceito *versus* Múltiplos Conceitos

É interessante distinguir o aprendizado de um único conceito, do aprendizado de uma coleção de conceitos. No aprendizado de um único conceito tem-se dois casos:

- quando as observações são exemplos do conceito (instâncias positivas)
- quando as observações são exemplos e contra-exemplos (instâncias positivas e negativas).

No primeiro caso — somente exemplos positivos — pela falta de contra-exemplos, a condição de consistência não é aplicável. No segundo caso, quando também há contra-exemplos do conceito a ser aprendido, o processo de aprendizado é consideravelmente simplificado. Esses contra-exemplos podem ser considerados como representando outras classes, e pode ser provada a condição de consistência. Os contra-exemplos mais úteis são aqueles que diferem pouco dos exemplos positivos (“near-miss”). Como já mencionado, tais exemplos trazem maiores restrições ao processo de generalização do que os exemplos gerados aleatoriamente.

No aprendizado de conceitos múltiplos também é possível distinguir dois casos:

- quando as descrições das diferentes classes devem ser mutuamente disjuntas, isto é, cada evento não pode satisfazer mais do que uma descrição;
- quando as descrições dos conceitos têm sobreposição, isto é, cada evento pode satisfazer mais do que uma descrição.

Em algumas situações é desejável o segundo caso. Por exemplo, se um paciente tem duas doenças, seus sintomas devem satisfazer descrições de ambas as doenças. Neste caso a condição de consistência não é aplicável.

3.2.4 Linguagem Descritiva

Uma das dificuldades nas inferências indutivas é a falta de limite para o nível de detalhe no qual a realidade observada pode ser descrita. Assim, é necessário especificar cuidadosamente os objetivos do problema a ser resolvido. Isto inclui definir uma linguagem e o escopo no qual as afirmações serão expressas, bem como as formas de inferências que serão utilizadas.

Para expressar as premissas, hipóteses e conhecimento de fundo deve-se utilizar alguma linguagem. Geralmente em AM é utilizada uma linguagem formal tal como lógica proposicional, cálculo de predicados, ou outros formalismos de tipo lógico, ou ainda um sistema de representação de conhecimento como redes semânticas, árvores de decisão, frames, “scripts” ou estruturas conceituais.

A escolha da linguagem adequada de representação é uma decisão importante no projeto de um sistema de aprendizado. Considere querer aprender, através dos exemplos mostrados na seguinte tabela, o conceito de “número mágico” [Charniak 86]:

No. mágico	No. não mágico
7	3
60	9
70	364
546	626
627	708

O número 628 é um número mágico? Certamente esta é uma pergunta difícil de responder nas condições dadas. Porém, se os números são expressos na base 9 tudo fica mais claro:

No. mágico		No. não mágico	
base decimal	base 9	base decimal	base 9
7	7	3	3
60	66	9	10
70	77	364	444
546	666	626	765
627	766	708	866

Como 628 na base 9 é 767, então poder-se-ia concluir que ele é um número mágico, pois os números mágicos parecem ser aqueles cujos dígitos são 6, 7, ou uma combinação de ambos, quando o número é expresso na base 9. Isto é, é preciso enfatizar que representações apropriadas podem ser um fator importante para facilitar a inferência indutiva. Em outras palavras, a notação utilizada para representar os conceitos deve ser tal que situações similares tenham descrições semelhantes.

Um sistema interessante de AM, que ilustra a importância da escolha da linguagem descritiva utilizada, é o sistema LEX [Mitchell 83]. LEX é um sistema que aprende como fazer integração simbólica. Isto é, LEX aprende as regras de como achar uma fórmula matemática que, quando derivada, representa a fórmula original. Sem entrar nos detalhes de como funciona o sistema LEX, quando ele verifica que uma determinada propriedade ou operador se aplica ao número 3 e também ao número 5, então, LEX generaliza que esta propriedade se aplica a Constante. Porém, em algumas situações, a generalização apropriada pode ser Constante-Ímpar, mas a linguagem utilizada por LEX não inclui este termo e LEX não é capaz de construir novos termos.

Desta forma, se durante o processo de aprendizado o vocabulário da linguagem utilizada pelo processo aprendiz é estático, então, os conceitos capazes de serem descritos por esta linguagem também são limitados. Por isto, deve-se verificar “a priori” que todos os conceitos que o sistema aprendiz pode achar são plausíveis de serem expressos na notação adotada.

Neste contexto, é instrutivo distinguir entre o *espaço das instâncias* e o *espaço das descrições* ou *espaço das versões*. O espaço das instâncias consiste de “todos” os possíveis exemplos e contra-exemplos do conceito a ser aprendido. O espaço das descrições é o conjunto de todas as descrições das instâncias que são possíveis de serem formuladas empregando a linguagem de descrição especificada.

Constante-Ímpar é um elemento do espaço das instâncias do domínio explorado por LEX. Porém, Constante-Ímpar não é um elemento constitutivo do espaço das descrições formuladas por LEX. Assim, o espaço das versões representa todas as possíveis versões “que são plausíveis de serem expressadas na notação escolhida”.

Em determinadas aplicações é importante que os resultados da indução sejam descrições simbólicas de entidades dadas, semântica e estruturalmente similares àquelas que um especialista humano produziria observando as mesmas entidades. Neste caso, os componentes dessas descrições devem ser compreensíveis, interpretáveis diretamente em linguagem na-

tural, e também devem relatar conceitos qualitativos e quantitativos em uma forma integrada. Estes requisitos são muito importantes, por exemplo, no desenvolvimento de bases de conhecimento para Sistemas Especialistas. Nestas aplicações, é interessante que o especialista humano possa verificar as afirmações indutivas com facilidade e confiabilidade, bem como traduzí-las para seu próprio domínio de conhecimento.

3.2.5 Problema do Conhecimento de Fundo

Um elemento essencial na tarefa de produzir um sistema indutivo correto, é explicar a maneira pela qual o conhecimento já adquirido do mundo existente afeta a aquisição indutiva do novo conhecimento. Uma indução enumerativa é ratificada pela descoberta de mais instâncias confirmatórias enquanto não ocorram instâncias que a contradigam.

Em particular, é de interesse investigar porque a uma generalização é dado um grande crédito enquanto outra é considerada muito suspeita, apesar de ambas terem o mesmo número de instâncias positivas e nenhuma instância negativa. Por exemplo, considere o caso do viajante na Itália que pela primeira vez encontra um italiano. Escutando o nativo falar italiano, imediatamente conclui que todos os italianos falam italiano, mas descobrindo que seu nome é Giuseppe ele não conclui que todos os italianos são chamados de Giuseppe [Russell 86]. Certamente, a diferença reside no conhecimento prévio do viajante sobre países, linguagens e nomes de pessoas.

A idéia geral que se procura expor e formalizar aqui, é que uma generalização indutiva pode ser confirmada ou não pela observação de suas instâncias ou contra-exemplos, bem como pela observação de outras regularidades de maior nível no domínio. Naturalmente, estas regularidades estarão baseadas em outras instâncias e em outras regularidades. A idéia geral é usar nossa experiência do mundo exterior para sustentar a aceitação de uma dada regra.

No caso do viajante na Itália, a generalização de que todos os italianos falam italiano é sustentada pela regularidade mais geral de que em um dado país a maioria da população fala a mesma língua; por outro lado, não é assumido que todos os italianos são chamados de Giuseppe devido à regularidade mais geral de que a maioria dos grupos sociais utilizam nomes diversos para diferentes indivíduos. Ou seja, está se usando, quando possível, a dedução em ajuda à indução, como uma maneira de permitir ao conhecimento de fundo influenciar nosso processo indutivo.

3.2.6 Importância dos Descritores Iniciais

Suponha que se queira ensinar ao computador o que é uma sequência no jogo de pôquer. Neste caso, seria apropriado poder descrever os exemplos de uma sequência em função das relações existentes entre 5 diferentes cartas, onde cada uma destas cartas é caracterizada pelo número e pelo naipe. Os seguintes são exemplos de sequências:

(6, ♣), (7, ♣), (8, ♣), (9, ♣), (valeta, ♣)

(9, ♥), (valeta, ♥), (dama, ♥), (rei, ♥), (ás, ♥)

Assim, pode-se dizer que nesta tarefa de aprendizado, número e naipe são descritores relevantes da entidade carta. Acrescentar o nome do fabricante do baralho ou a cor do baralho, não será de muita utilidade. Neste caso, nome do fabricante e cor não são descritores relevantes.

Esse exemplo serve para ilustrar um problema fundamental subjacente em toda tarefa de aprendizado indutivo: que informação deve ser fornecida à máquina e que informação se espera que a máquina produza ou aprenda? O componente principal de entrada para um sistema de aprendizado é um conjunto de afirmações observacionais. Os descritores utilizados nestas asserções são características observadas e medidas disponíveis dos objetos em consideração. A seleção destes descritores é relevante na tarefa de aprendizado quando o professor especifica o problema. A determinação desses descritores é um dos principais componentes do problema de aprendizado. Se eles captam as propriedades essenciais dos objetos, é simples arranjar estes descritores em uma expressão que constitui uma afirmação indutiva.

Dado um problema específico, geralmente é possível especificar um conjunto de valores que cada descritor poderá adotar na caracterização de cada objeto da população sob consideração. Tal conjunto é denominado o *domínio do descritor*. Este domínio é utilizado para restringir a amplitude na qual o descritor pode ser generalizado. Por exemplo, para uma carta o descritor número pode assumir os valores 2, 3, ... 8, 9, valeta, dama, rei e ás. Porém, no jogo de pôquer, dependendo do número de jogadores, este domínio pode ser reduzido, por exemplo, a 8, 9, ... rei e ás.

3.2.7 O Critério de Preferência

O número de afirmações indutivas, consistentes com os dados observados, pode ser muito grande. O problema, então, consiste em escolher as afirmações indutivas mais apropriadas. Em tal escolha, devem ser considerados aspectos particulares do problema, entrando em jogo os *critérios de preferência* na seleção das hipóteses. Geralmente, as hipóteses são escolhidas com base no critério da simplicidade e facilidade de refutação.

Algumas medidas específicas para avaliar a qualidade das induções podem ser:

- simplicidade geral para a compreensão humana. Por exemplo, o número de descritores e número de operadores utilizados numa afirmação indutiva;

- grau de incerteza de cada descritor;
- custo de medir os valores dos descritores. Supondo, por exemplo, que uma determinada doença possa ser caracterizada diretamente pelo resultado de uma única análise muito cara, ou pelo conjunto de um número maior de sintomas de fácil verificação. Neste caso, apesar da primeira caracterização da doença ser mais simples, pode ser adotada a segunda pelo seu menor custo;
- a quantidade de informação necessária para codificar a afirmação indutiva.

A importância dada a cada medida depende do propósito da construção das afirmações indutivas.

3.2.8 Regras de Generalização

Uma regra de generalização é a transformação de uma dada descrição em uma descrição mais geral, que tautologicamente implique a descrição inicial. Por exemplo, uma generalização de:

todos os napolitanos comem pizza

é

todos os italianos comem pizza.

Uma regra de especificação faz a transformação oposta: dada uma descrição, ela gera uma consequência lógica desta — se é verdade que todos os italianos comem pizza então é válido afirmar que todos os napolitanos comem pizza.

Uma regra de reformulação transforma uma descrição em outra logicamente equivalente, por exemplo:

$$\text{se } x + 8 - 10 > y - 2 \text{ então } x > y.$$

As regras de especialização e reformulação são as regras de inferência convencionalmente preservadoras de verdade, utilizadas na lógica dedutiva. Por outro lado, as regras de generalização preservam falsidade; isto significa que se um evento falsifica alguma descrição, então, também falsifica a sua generalização. Se a afirmação, todos os sicilianos comem pizza for falsa, então, este fato também falsifica a sua generalização: todos os italianos comem pizza. Porém, a falsificação da generalização não implica na falsificação da sua

consequência. Pode acontecer de nem todos os italianos comerem pizza, mas de todos os sicilianos gostarem de comer pizza.

Define-se que um conceito G é *mais geral* do que um conceito H se todas as ocorrências positivas de H também são ocorrências positivas de G. Isto significa que G possui todas as ocorrências positivas de H e, possivelmente, algumas outras. Neste caso H é mais específico do que G. A idéia de mais geral e mais específico é similar a idéia de maior-igual e menor-igual. Devido a isto, os cientistas dizem que há uma ordem parcial no universo dos conceitos [Arariboia 89].

Uma observação importante sobre as regras de generalização e especialização é que elas fornecem um conjunto de métodos para revisar conceitos. Entretanto, saber em que direção revisar um conceito (isto é, saber se ele deve ser mais específico ou mais geral) não diz muito a respeito de “como” modificá-lo. A quantidade de possíveis regras ou operadores de generalização a serem aplicados é grande e, portanto, o espaço de busca de possíveis generalizações é muito amplo.

Michalski [Michalski 83b] discute de forma sistemática e completa os diversos tipos de regras de generalização. A seguir são apresentadas algumas dessas regras de generalização que são relevantes para o nosso trabalho.

A regra da condição abandonada

Esta regra estabelece que a descrição de um conceito pode ser generalizada removendo uma expressão ligada conjuntivamente. Esta é uma das regras mais empregadas para a generalização de informação. Para exemplificar, suponha que se tenha a seguinte descrição parcial do conceito navio [Arariboia 89]:

Conceito1: $\text{move-se}(\text{navegando}) \wedge \text{propulsão}(\text{motor}) \wedge$
 $\text{transporta}(\text{passageiros}) \wedge \text{transporta}(\text{carga})$
 $\wedge \text{faz-viagens}(\text{longas}).$

que diz que um navio se move navegando e tem propulsão a motor e transporta passageiros e ... Admitindo que neste ponto seja observado um navio que só realiza viagens curtas, então, o Conceito1 não se aplica a este navio. É preciso generalizar este conceito; uma possibilidade é abandonar a exigência de que as viagens sejam longas. Com isto, obtém-se o Conceito2:

Conceito2: $\text{move-se}(\text{navegando}) \wedge \text{propulsão}(\text{motor}) \wedge$
 $\text{transporta}(\text{passageiros}) \wedge \text{transporta}(\text{carga}).$

Este conceito, mais geral que o anterior pois engloba maior número de instâncias, foi obtido abandonando a condição da duração das viagens.

A regra da adição alternativa

A descrição de um conceito pode ser generalizada adicionando uma alternativa (mediante a utilização da disjunção lógica, $\vee$). No caso do exemplo anterior, para o Conceito1 também ser aplicado a navios que fazem viagens curtas, em vez de abandonar a exigência da duração das viagens, pode-se adicionar a alternativa de que as viagens possam ser longas ou curtas, obtendo-se o seguinte conceito:

Conceito3: $\text{move-se}(\text{navegando}) \wedge \text{propulsão}(\text{motor}) \wedge$
 $\text{transporta}(\text{passageiros}) \wedge \text{transporta}(\text{carga}) \wedge$
 $\text{faz-viagens}(\text{longas}) \vee \text{faz-viagens}(\text{curtas}).$

A regra de extensão da referência.

Esta regra é aplicada quando se substitui o valor, ou conjunto de valores, $V1$ de um certo descritor D por um valor $V2$, tal que $V1$ está contido ou pertence a $V2$. Aqui, $V1$ e $V2$ podem ser interpretados como um conjunto de valores que o descritor D pode assumir e $\text{DOM}(D)$ denota o domínio do descritor D . Esta regra estabelece que a descrição de um conceito pode ser generalizada ampliando a referência de um descritor.

Suponha que se queira descrever o conceito “jovem”. Uma criança pode falar que é jovem aquela pessoa com menos de 18 anos:

$\text{jovem}(X) \text{ se } \text{pessoa}(X) \text{ e } \text{idade}(X) \leq 18 \text{ anos}$

ou seja, o descritor idade pode assumir qualquer valor menor ou igual a 18 anos. Já para um adolescente, jovem pode ser aquele que não tem mais de 30 anos:

$\text{jovem}(X) \text{ se } \text{pessoa}(X) \text{ e } \text{idade}(X) \leq 30 \text{ anos}$

isto é, o descritor idade poderá assumir qualquer valor menor ou igual a 30. Os possíveis valores de idade da primeira descrição $[0 \dots 18]$, foram substituídos por um intervalo maior $[0 \dots 30]$, fazendo com que esta segunda descrição seja mais geral, pois um maior número de pessoas poderão ser catalogadas como jovens.

Se os possíveis valores do descritor D são estendidos para todo seu domínio, $D = \text{DOM}(D)$, então, esta condição sempre será verdadeira e, portanto, pode ser removida. Por exemplo, suponha-se agora a seguinte descrição de jovem:

“jovem é toda pessoa, de qualquer idade, que está sempre disposta a enfrentar novos desafios”

Neste caso, o descritor idade pode assumir qualquer valor pertencente a seu domínio, isto é, a idade passou a ser um fator irrelevante para caracterizar uma pessoa jovem; por este motivo ela pode ser removida da descrição:

“jovem é toda pessoa que está sempre disposta a enfrentar novos desafios”

Assim sendo, esta regra se transforma na regra de generalização da condição abandonada explicada acima.

A regra de transformar constantes em variáveis

Variáveis são mais gerais do que constantes. Suponha o conjunto de descrições de situações mostrado a seguir:

1. (X acima MESA) $\wedge$ (Y acima MESA)
2. (X acima MESA) $\wedge$ (Y acima Z)
3. (X acima Z) $\wedge$ (Y acima Z)
4. (X acima Z) $\wedge$ (Y acima W)

A descrição 1) é mais específica do que a 2). Por sua vez, a descrição 2) é mais específica do que a 3) e a 4) é a mais geral de todas elas. Isto é, a fórmula mais geral se aplica a um maior número de situações. A fórmula 4) é verdade se alguma coisa X está acima de algo e alguma coisa Y está acima de algo. Já na fórmula 3) X e Y devem estar acima do mesmo objeto, e na fórmula 1) ambos devem estar acima de uma mesa.

Este processo de generalizar uma fórmula substituindo constantes por variáveis é chamado de “*variabilização*”. Este processo é importante pois muitos programas de aprendizado começam variabilizando suas entradas. Por exemplo, o programa de Winston — seção 3.2 — começa assumindo que a definição do conceito a ser aprendido é igual à descrição do primeiro exemplo observado, apenas trocando as constantes por variáveis.

A regra de transformar uma conjunção numa disjunção

Esta regra de generalização troca uma conjunção lógica por uma disjunção lógica. Voltando ao exemplo da descrição do navio, o Conceito2 descreve um navio como:

Conceito2: move-se(navegando) $\wedge$ propulsão(motor) $\wedge$
transporta(passageiros) $\wedge$ transporta(carga).

Admitindo que, neste ponto, observa-se um navio que só transporta carga, então, diante desta ocorrência positiva o Conceito2 não serve mais. Torna-se necessário generalizá-lo de modo que ele inclua este último exemplo observado. Utilizando a regra acima, o Conceito2 se transforma em:

Conceito4: $\text{move-se}(\text{navegando}) \wedge \text{propulsão}(\text{motor}) \wedge$
 $\text{transporta}(\text{passageiros}) \vee \text{transporta}(\text{carga}).$

que diz que um navio se move navegando, tem propulsão à motor e transporta passageiros ou transporta carga. É evidente que o Conceito4 é mais geral do que o Conceito2. Se um navio satisfaz a condição de transportar passageiros ou carga ao mesmo tempo, ele satisfaz à condição de transportar uma coisa ou a outra.

Regra de generalização por hierarquia

Predicados e classes, geralmente, estão agrupados em famílias com membros mais ou menos gerais. Por exemplo:

Subindo ou descendo na árvore obtém-se termos com diferentes graus de generalidades. Este tipo de generalização pode ser denominado “hierarquia é-um (is-a)” [Charniak 86] ou “climbing generalization tree rule” [Michalski 83a].

Regra da universalização

Esta regra troca uma lista de fórmulas específicas por combinações quantificadas. Supondo as seguintes descrições:

- a) O objeto_1 acima da mesa é uma esfera
O objeto_2 acima da mesa é uma esfera
- b) O objeto_1 acima da mesa é uma esfera
O objeto_2 acima da mesa é uma esfera
O objeto_3 acima da mesa é uma esfera
O objeto_4 acima da mesa é uma esfera

estas duas descrições podem ser generalizadas como:

todo objeto que está acima da mesa é uma esfera

Em geral, esta regra pode ser esquematizada da seguinte forma:

Se $p(x_1) \Rightarrow q(x_1)$
e $p(x_2) \Rightarrow q(x_2)$
e $p(x_3) \Rightarrow q(x_3)$

...
Então $\forall(x) p(x) \Rightarrow q(x)$

3.3 Estratégias de Aprendizado por Indução

Na seção anterior foram apresentadas as características do problema de Aprendizado por Indução. Nesta seção serão estudadas algumas das soluções propostas. Serão, pois, introduzidas as metodologias e características das diferentes estratégias de Aprendizado por Indução.

3.3.1 Estratégias de Controle: Métodos “Data-driven” *versus* Métodos “Model-driven”

Uma importante característica dos métodos de aprendizado é a forma na qual as descrições são geradas e/ou procuradas no espaço das descrições, em relação aos exemplos ou fatos do espaço das instâncias. Pode-se distinguir três tipos de métodos [Michalski 87]:

- “data-driven” (ou bottom-up)
- “model-driven” (ou top-down)
- mistos

O método “data-driven” — ou bottom-up — começa selecionando um ou mais exemplos, depois formula uma hipótese que explique estes exemplos e logo generaliza, ocasionalmente especializa, esta hipótese para poder explicar novos exemplos. Em outras palavras, este método procede do específico ao geral.

Um bom exemplo deste método é o programa de Winston [Winston 75]. Este programa aprende descrições estruturais de conceitos do mundo dos blocos (por exemplo o conceito arco, casa, etc) através de exemplos e contra-exemplos fornecidos por um professor. Num passo inicial é assumido que a descrição do primeiro exemplo observado é a descrição do conceito. Se o próximo exemplo é positivo e não satisfaz a descrição corrente do conceito, então, esta descrição é generalizada para incluir este exemplo. Se o exemplo seguinte é negativo mas satisfaz a descrição do conceito, então, esta descrição é especializada para excluir este contra-exemplo. Este processo é repetido com cada novo exemplo até convergir numa descrição que classifique corretamente todas as instâncias positivas e negativas. A generalização e especialização do conceito é efetuada aplicando as regras acima mencionadas (tornar constantes em variáveis, abandonar condições, etc.).

O método “model-driven” — ou top-down — começa com uma hipótese muito geral e depois a especializa, ocasionalmente a generaliza, para que ela caracterize novos exemplos. Este método procede do geral ao específico.

Um exemplo deste modelo é o programa Meta-Dendral [Buchanan 78], que gera regras explicando dados de Espectrometria de Massa. Este sistema de aprendizado teve um

desempenho muito bom em um domínio do mundo real, manipulando dados com ruído. Outros exemplos de sistemas que utilizam métodos model-driven são o programa CLS [Hunt 66] e sua versão modificada ID3 [Quinlan 83].

O método misto combina elementos dos dois métodos anteriores, tentando herdar as suas vantagens e evitar suas desvantagens. Ele utiliza os exemplos para mudar de uma hipótese a outra, mais geral ou específica, e testar as hipóteses, que são logo modificadas para explicar novos exemplos.

AQ11 [Michalski 78] é um exemplo deste tipo de abordagem. Este programa de aprendizado de propósito geral formula regras descrevendo várias classes de exemplos. A entrada do programa consiste de vetores de valores de atributos descrevendo as classes de exemplos. Suponha que se queira aprender os conceitos de “pessoa branca” e “pessoa de cor”. Para descrever estas duas classes de pessoas pode-se utilizar os atributos cor de pele, cabelo e cor de olhos. Assim, o vetor:

(claro, castanhos, verdes, brânca)

descreve alguém de cútis clara, cabelos castanhos e olhos verdes como sendo uma pessoa branca. Além dos vetores de atributos, a entrada também consiste de conhecimento de fundo do domínio da aplicação e um critério de preferência das hipóteses. A saída pode ser vista como regras do tipo:

condição $\Rightarrow$ classe

onde “*condição*” pode ser uma conjunção ou a disjunção de conjunções, tais que elas descrevem todas as entidades pertencentes a uma determinada classe. Uma versão simplificada do algoritmo AQ é mostrada a seguir:

1. selecione aleatoriamente um exemplo positivo (denominado semente);
2. compare a semente com o primeiro exemplo negativo, gere hipóteses que expliquem a semente mas excluam o exemplo negativo;
3. especialize as hipóteses para excluir todos os exemplos negativos;
4. selecione a melhor hipótese segundo o critério de preferência. Se esta hipótese se aplica a todos os exemplos positivos, então esta hipótese é a solução; senão ache todos os exemplos positivos que não são explicados por esta hipótese;
5. Repita os passos 1 até 4 para o conjunto restante e continue até todos os exemplos positivos serem considerados. A disjunção das hipóteses selecionadas ao final de cada ciclo é uma descrição completa e consistente de todos os exemplos positivos, e maximiza o critério de preferência.

Assim, o programa constrói uma descrição disjuntiva quando não consegue construir uma descrição conjuntiva.

Em geral, métodos data-driven são mais eficientes, enquanto que métodos model-driven são mais tolerantes a erros nos dados [Diettrich 83].

Carbonell [Carbonell 87] descreve estes três métodos, numa forma um pouco mais restrita, como realizando Aprendizado por Exemplos Como Busca, descrito a seguir.

3.3.2 Aprendizado por Exemplos como Busca

A construção de afirmações indutivas, a partir de observações, pode ser conceitualmente caracterizada como uma busca heurística num espaço de estados onde:

- *estados* são descrições simbólicas: o estado inicial é o conjunto de observações;
- *operadores* são as seguintes regras de inferência: generalização, especialização e reformulação — explicadas anteriormente;
- *estado final* ou *objetivo* é uma afirmação indutiva que: implica as observações, satisfaz o problema do conhecimento de fundo e maximiza o critério de preferência dado.

Muitos sistemas de Aprendizado por Exemplos realizam uma busca através do espaço dos possíveis conceitos. Para exemplificar, seja a amostra de células malignas e células benignas [Carbonell 87] mostradas na figura 3.2.

A figura 3.2 mostra 5 exemplos de células, 3 das quais (P1, P2, P3) são identificadas com uma determinada doença X, enquanto que as outras duas (N1, N2) representam células sadias. Pode ser observado que cada célula tem dois corpos, e cada corpo tem os seguintes três atributos:

- número de núcleos: 1 ou 2
- cor: claro ou escuro
- número de caudas: 1 ou 2

A figura 3.3 apresenta, no mesmo formato gráfico das instâncias, duas possíveis descrições de uma célula doente. Observe que somente alguns dos atributos estão presentes em cada descrição; isto significa que os outros atributos são irrelevantes.

Assim, a hipótese a) estabelece que, para uma célula predizer a doença X, ela deve ter um corpo com dois núcleos e o outro corpo deve ser claro com uma única cauda. A descrição b) estabelece que uma célula doente deve ter um corpo com dois núcleos, enquanto que

Figura 3.2: Instâncias Positivas e Negativas de Células Doentes

Figura 3.3: Duas Possíveis Descrições de Células Doentes

o outro corpo deve ser claro. A descrição b) cobre mais instâncias que a descrição a). Pode-se, então, dizer que b) é mais geral que a).

A estrutura geral do espaço das possíveis hipóteses é mostrada na figura 3.4. No topo da figura estão as hipóteses mais específicas — a descrição mais específica é a conjunção lógica das três instâncias positivas —, na parte de baixo da figura são representadas as hipóteses mais gerais. Neste desenho pode-se constatar como a ordem dos conceitos é parcial. Por exemplo, as hipóteses a) ou b) são mais específicas que d), mas a) não é mais específica nem mais geral do que b) e vice-versa.

Pode-se considerar dois tipos de operadores para se movimentar neste espaço do problema: um para fazer uma hipótese existente mais geral e outro para fazê-la mais específica. Há também dois esquemas básicos para realizar a busca no espaço das descrições dos conceitos. No primeiro inicia-se com a hipótese mais geral e, à medida que novas instâncias são encontradas, são produzidas descrições mais específicas. No outro, inicia-se com a hipótese mais específica relaxando suas condições para fazê-la mais geral, a medida que novos dados são observados.

Para exemplificar o primeiro método, do específico ao geral ou “data-driven”, será assumido que são apresentadas ao aprendiz as instâncias da fig.3.2 numa forma incremental. Neste esquema, a hipótese inicial é inicializada com a primeira instância positiva observada. Assim, observando a instância P1 da fig. 3.2, o sistema criará a hipótese inicial H1 mostrada na fig. 3.5. Esta descrição é muito específica e explica somente o exemplo observado. Quando é apresentado o segundo exemplo positivo (P2), o sistema verifica que esta primeira hipótese não se aplica à P2 e, portanto, deve ser generalizada. O aprendiz remove H1 e a substitui por uma hipótese mais geral que explica P1 e P2. Neste caso há duas possíveis hipóteses: H2 (que ignora o número de núcleos) e H3 (que somente considera o número de núcleos).

As instâncias positivas (como no caso P2) levam o sistema a formular hipóteses mais gerais, enquanto que instâncias negativas lhe permitem eliminar competidores. Por exemplo, o fato da hipótese H3 poder ser aplicada ao contra-exemplo N1 sugere que esta hipótese é geral demais. Como a hipótese H2 não se aplica a N1, então, ela é escolhida para continuar expandindo o conceito, enquanto que H3 é descartada. Deve ser observado que esta estratégia se baseia no suposto que todas as instâncias estão corretamente rotuladas.

Depois de processar as três primeiras instâncias, o sistema aprendiz acha P3. Como a hipótese H3 falha para descrever esta instância positiva, o sistema gera descrições mais gerais: H4 e H5. A hipótese H4 descreve como célula doente aquela que tem um corpo com uma cauda e o outro corpo com duas, enquanto que para H5 uma célula é doente se tem um corpo claro e o outro escuro. Quando a segunda instância negativa (N2) é observada, o sistema verifica que H4 classifica incorretamente esta célula. H4 é descartada e o sistema mantém ativa a hipótese H5. O sistema aprendiz continua trabalhando desta forma, produzindo descrições mais gerais quando as instâncias positivas o requerirem e eliminando hipóteses quando as instâncias negativas forem incorretamente rotuladas.

Figura 3.4: Ordem Parcial no Espaço das Hipóteses

Figura 3.5: Gerando Hipóteses do Específico ao Geral

O segundo método — do geral ao específico ou “model-driven” — parte com a hipótese mais geral possível. A estrutura geral é similar ao método anterior, novas instâncias levam a criar novas hipóteses ou a eliminar descrições existentes. Agora, entretanto, as instâncias negativas servem para fazer uma hipótese mais específica, enquanto que as instâncias positivas servem para eliminar hipóteses. Ou seja, os papéis desempenhados pelos exemplos positivos e pelos exemplos negativos são invertidos com relação ao método anterior.

Enquanto que o método do específico ao geral cria as descrições mais específicas possíveis que cobrem os dados, o método do geral ao específico gera as descrições mais gerais possíveis. Assim, pode-se utilizar um ou outro método segundo sejam desejadas regras pessimistas ou otimistas. Porém, se for fornecido um número suficiente de exemplos, ambos sistemas convergem, eventualmente, à mesma descrição do conceito.

Estas duas abordagens — do específico ao geral e do geral ao específico — podem ser combinadas, de diferentes formas, em um único método para buscar o espaço das descrições. A combinação traz como resultados algumas vantagens que não estão presentes em cada método isolado.

Um exemplo é o método do espaço das versões [Mitchell 78] utilizado no sistema LEX [Mitchell 83]. Este método armazena dois conjuntos de hipóteses: o conjunto de hipóteses mais específicas que explicam os dados e o conjunto das descrições mais gerais que se aplicam aos dados. À medida que novos exemplos são observados, estes dois conjuntos

vão sendo modificados até convergir numa única hipótese.

3.3.3 Aprendizado por Exemplos Incremental e Não Incremental

Muitas pesquisas de aprendizado por exemplos têm utilizado variações das abordagens acima descritas. Entretanto, há um tipo de sistema que trabalha com um enfoque diferente.

Nos métodos mencionados anteriormente, os exemplos observados ficam disponíveis ou são considerados um a um pelo sistema, não todos simultaneamente. Isto é, o sistema considera o primeiro exemplo, positivo ou negativo, e segundo este exemplo constrói uma determinada hipótese do conceito a ser aprendido. A seguir considera um segundo exemplo e modifica ou não a hipótese, baseando-se em como esta classifica a segunda instância. À medida que mais exemplos são apresentados, o sistema continua modificando o conceito. Por este motivo, estes métodos são denominados incrementais. Por exemplo, o algoritmo de Winston assume que os exemplos são apresentados ao sistema numa ordem “inteligente”, de forma a otimizar a razão de convergência para o conceito — utilização de exemplos “near miss”.

A outra abordagem, denominada não incremental, requer que todos os exemplos, positivos e negativos, estejam disponíveis antes do sistema aprendiz começar a trabalhar. Um exemplo desta abordagem são os sistemas da família TDIDT [Quinlan 86a]; descritos no capítulo 4, e a maioria dos sistemas baseados no algoritmo AQ [Michalski 73, 78, 80b].

3.3.4 Indução Construtiva

Há sistemas de aprendizado que são capazes de ampliar e modificar a linguagem utilizada para descrever exemplos e conceitos. Estes sistemas possuem uma série de atributos iniciais para descrever os exemplos, além de várias regras de inferência, heurísticas ou procedimentos que são utilizados para construir novos atributos. Por exemplo, um sistema que trabalha com corpos geométricos pode, a partir dos atributos de medidas lineares — altura, comprimento, raio, etc. — construir o atributo volume correspondente a cada corpo. Esses sistemas, capazes de modificar os atributos utilizados na linguagem descritiva, são chamados de *Métodos Construtivos*. Por outro lado, aqueles métodos que têm os seus atributos definidos “a priori” e não são capazes de incluir nenhum novo atributo são chamados de *Métodos Seletivos*.

Os seres humanos, muitas vezes, devem descobrir quais são as melhores representações, antes do sistema de aprendizado começar a trabalhar. O desempenho de muitos sistemas de Inteligência Artificial é criticamente dependente da escolha da linguagem de representação. É por estas razões que a indução construtiva é uma característica importante que deve estar presente em determinados sistemas de aprendizado.

3.4 Considerações Finais

Neste capítulo foram abordados três tópicos referentes a aprendizado de conceitos por indução.

Primeiramente foi discutida a indução lógica através de um paralelo com a dedução lógica, isto é, foram evidenciados quais os objetivos e características de cada uma delas. Enquanto os argumentos dedutivos servem para explicitar e validar o conhecimento, os argumentos indutivos atendem à finalidade de ampliar o alcance do conhecimento já existente. Argumentos indutivos são de fundamental importância nos processos de descoberta e generalização científica. Em particular, eles são os componentes básicos utilizados nos processos de Aprendizado por Indução.

A seguir foram estudadas as características e problemas do paradigma indutivo. Neste tópico foram abordados, entre outros, a linguagem de descrição do problema, a importância e tipo dos descritores, as regras de generalização que são utilizadas na construção dos conceitos, bem como os critérios de preferência e conhecimento de fundo que podem eventualmente servir como guia no processo indutivo. Resumidamente, pode-se dizer que este tópico trata da descrição tanto do problema a ser resolvido, quanto de suas principais características.

Na última seção foram apresentadas algumas das soluções propostas, isto é, foram descritas as metodologias das diferentes estratégias de Aprendizado por Indução.

Capítulo 4

Família TDIDT

Neste capítulo é apresentada a família de sistemas TDIDT — Top Down Induction Decision Trees — que realizam aprendizado por exemplos construindo árvores de decisão. Na primeira seção é descrito o processo de construção de árvores de decisão a partir de exemplos. A segunda seção descreve o sistema ID3. Na terceira seção são apresentados os componentes da família TDIDT, bem como uma relação das propriedades que caracterizam este tipo de algoritmos. A seguir é detalhado um histórico das pesquisas referentes a esta família de Sistemas e, finalmente, são enumeradas as vantagens e desvantagens da utilização deste tipo de sistemas.

4.1 Construindo Árvores de Decisão

Os sistemas da família TDIDT — que serão detalhados em seções posteriores — se caracterizam por construir árvores de decisão para tarefas de classificação. Estas árvores são construídas a partir da raiz, descendo até as folhas.

A seguir é descrito o processo de construção de uma árvore de decisão, a partir de exemplos, de tal forma que a árvore classifique corretamente esses exemplos ou observações, e sirva também para gerar regras referentes a essas observações.

Para ilustrar esse processo, será considerado um programa de auditório que realiza o “namoro na TV”. O telespectador quer descobrir um processo para prever se um candidato vai ou não arrumar namorada. Ele pode consultar o apresentador do programa, o qual tem muita experiência no assunto. Neste caso, poder-se-ia desenvolver um diálogo como o seguinte:

apresentador — O candidato é bonito ou feio?
 telespectador — Ele é feio.
 apresentador — Ele é inteligente?
 telespectador — Não.
 apresentador — O candidato tem dinheiro (pouco, mais ou menos ou muito)?
 telespectador — Ele tem muito dinheiro.
 apresentador — Pelo que você me fala o candidato provavelmente arrumará namorada.

que evidencia o uso de uma certa estrutura de decisão por parte do apresentador. Tal estrutura poderia ser uma árvore de decisão. A figura 4.1 mostra como este diálogo pode ser representado na forma de árvore de decisão.

Figura 4.1: Árvore de Decisão Incompleta

Esta árvore de decisão parcial representa apenas o caminho resultante do diálogo acima descrito. Uma árvore completa poderia ser construída representando todas as possíveis perguntas com todas as possíveis respostas.

A partir da árvore de decisão, pode ser formulado um conjunto equivalente de regras. Por exemplo, percorrendo a árvore da figura 4.1 da raiz até o nó final (folha) pode-se deduzir a seguinte regra:

Se o candidato é feio
e não é inteligente
e tem muito dinheiro
então ele arrumará namorada.

Neste exemplo inteligência, beleza e situação financeira são chamados de atributos ou indicadores. Agora o telespectador quer achar mais regras para predizer se um determinado candidato será capaz ou não de arrumar namorada. Para isto, ele confecciona uma tabela com uma lista de alguns candidatos que já passaram pelo programa, alguns fatos sobre eles (suas características ou atributos) e seu desempenho (se arrumaram ou não namorada). Uma amostra destes dados é ilustrada na Tabela 4.1.

inteligência	beleza	sit.financieira	arrumou namorada
sim	bonito	rico	+
não	feio	pobre	-
sim	feio	pobre	+
sim	feio	meio	+
não	bonito	pobre	-
não	bonito	meio	+
não	bonito	rico	+
não	feio	rico	+

Tabela 4.1: Conjunto de Observações ou Exemplos

Cada linha desta tabela é um exemplo ou observação, os rótulos das colunas (inteligência, beleza, etc.) são os atributos, sendo que o último atributo (arrumou namorada) é chamado atributo de classe. É este atributo que divide os exemplos nas diferentes classes — neste caso uma classe positiva e outra negativa. Os candidatos que arrumaram namorada formam a classe positiva (+) e os que não arrumaram namorada formam a classe negativa (-).

O objetivo é determinar uma relação entre as classes e os valores dos outros atributos.

Por exemplo, a primeira observação da tabela diz que um candidato muito inteligente, bonito e rico arrumou namorada. Porém, esta tabela não diz nada a respeito do desempenho de futuros candidatos. Para isto constroi-se uma árvore de decisão completa representando estes dados.

Primeiro deve ser selecionado um atributo para ser a raiz da árvore e então dividir as observações segundo os valores deste atributo; seja ele o atributo inteligência. Deve-se, então, dividir as observações em dois grupos: um com os candidatos inteligentes e outro com os não inteligentes; coloca-se cada um destes grupos no extremo correspondente dos ramos da árvore, como mostrado na figura 4.2. Assim, todas as observações nas quais o candidato é inteligente são colocadas no ramo rotulado com inteligência:sim. Nestas

Figura 4.2: Construindo a Árvore de Decisão

observações foi retirado o atributo inteligência, pois sabe-se que todos estes candidatos são inteligentes. De maneira análoga, todas as observações de candidatos pouco inteligentes vão para o outro ramo.

Todas as observações do ramo inteligência:sim pertencem à mesma classe, ou seja, todas indicam que o candidato arrumou namorada. Quando isto acontece, as observações devem ser substituídas pela classe comum a todas elas. No presente caso, a árvore adquire a forma da figura 4.3.

Figura 4.3: Construindo a Árvore de Decisão

No ramo dos pouco inteligentes ainda existem observações de duas classes diferentes. Quando isto acontece, escolhe-se outro atributo e repete-se a divisão em ramos. Será considerado o atributo situação financeira para realizar a próxima ramificação. Desta vez deve-se separar os dados que estão no ramo dos “não inteligentes” em três sub-ramos, segundo a situação financeira:

- tem muito dinheiro (rico)

- situação financeira média (+ ou -)
- tem pouco dinheiro (pobre).

A figura 4.4 mostra o resultado desta ramificação.

Figura 4.4: Construindo a Árvore de Decisão

Agora, cada sub-ramo tem observações que representam classes homogêneas. Pode-se então substituir as observações por suas classes. O resultado é mostrado na figura 4.5.

Como cada folha da árvore contém uma única classe, o processo de ramificação ou construção da árvore está completo. Assim, se seus ramos forem percorridos, primeiro em profundidade até chegar nas folhas, serão achadas as seguintes regras:

1. Se o candidato é inteligente
então arruma namorada.
2. Se ele não é inteligente
e é rico
então arruma namorada.
3. Se ele não é inteligente
e sua situação financeira é média
então arruma namorada.
4. Se ele não é inteligente
e é pobre
então não arruma namorada.

Deve ser observado que é produzida uma regra para cada nó terminal da árvore.

Figura 4.5: Árvore de Decisão

A figura 4.5 mostra uma característica interessante do processo de construção da árvore de decisão. Apesar do conjunto de observações originais conter três atributos (beleza, inteligência e situação financeira), o atributo *beleza* não foi necessário para classificar as observações do conjunto. Neste caso, pode-se dizer que o atributo *beleza* é um atributo *irrelevante*.

Certamente, a árvore da figura 4.5 não é a única árvore que pode ser construída a partir desse conjunto de exemplos. Dependendo do atributo escolhido em cada passo, podem ser geradas diferentes árvores representando os mesmos dados. Por exemplo, se tivesse sido escolhido o atributo *beleza* para começar a dividir o conjunto amostra, teria sido gerada uma árvore como a da figura 4.6.

Esta árvore é mais complexa e ramificada do que a da figura 4.5, embora represente os mesmos dados. Esta árvore também gera regras menos eficientes, pois as regras possuem mais condições e são mais numerosas — neste caso tem-se 7 regras contra 4 da árvore anterior.

No caso de problemas reais, com grande número de atributos e maior número de observações, é importante poder gerar regras concisas e no menor número possível.

Do que foi dito, conclui-se que é necessário um método para descobrir qual é o atributo relevante que deve ser utilizado para ramificar cada nó da árvore de decisão. Existem diversos métodos. Um deles, baseado na teoria da informação, calcula a entropia de cada atributo. Este método é o utilizado pelo sistema ID3, descrito logo a seguir.

4.2 ID3

O sistema ID3 [Quinlan 79] — um dos mais conhecidos membros da família TDIDT, que será descrita na seção 4.3 — escolhe o melhor atributo para ramificar cada nó da árvore,

Figura 4.6: Árvore de Decisão Maior

baseando-se na entropia da informação.

4.2.1 Entropia

Entropia é uma grandeza que mede desordem, tanto de objetos físicos quanto de informações. Quanto maior a entropia maior a desordem.

Arariboia explica o conceito de entropia de uma forma muito clara [Arariboia 89]:

Para entender o que é ordem, imagine que haja várias gavetas contendo objetos embalados em caixas. Se cada uma das gavetas contém apenas objetos de um só tipo, não precisamos abrir as caixas para saber que objetos estão dentro delas. Em outras palavras, se a caixa foi retirada de uma gaveta de lápis, não precisamos abrir a referida caixa para saber que contém lápis. Não precisamos, em suma, de nenhuma informação para descobrir o que está dentro da caixa. E porque não foi necessária nenhuma informação? Porque os objetos estavam adequadamente organizados dentro das gavetas. Se, porém, cada gaveta contivesse objetos dos mais variados tipos, teríamos que abrir a caixa para saber qual objeto foi retirado de cada gaveta. Precisariamos de informação para classificar o objeto. Um grupo de objetos está tanto mais organizado quanto menos informação externa for preciso para classificá-lo. A propósito, a entropia, no caso em que cada gaveta contém apenas um tipo de objeto, é nula, pois tudo está em ordem.

É mais ou menos intuitivo que devemos escolher o atributo que produza a ramificação de menor entropia (e, portanto, de maior organização). De preferência, gostaríamos de ter entropia nula. Isto significa que, em um dado ramo, todas as classes são iguais e, portanto, não precisamos de mais nenhuma informação para descobrir uma determinada classe, basta saber em que ramo ela está. Neste caso os ramos são como gavetas, contendo, cada uma, apenas um tipo de objeto.

Para calcular a entropia de um determinado atributo é necessário calcular a entropia de uma ramificação. A fim de exemplificar, é considerado o caso do atributo inteligência. Este atributo produz dois ramos, como mostrado na figura 4.2. O ramo dos candidatos inteligentes contém observações de uma classe só. Portanto, ele está ordenado e possui entropia zero; a desorganização dele é nula. O ramo da pouca inteligência contém observações de duas classes diferentes; em outras palavras, ele ainda está desorganizado. Segue-se o cálculo da entropia de um determinado ramo ainda desorganizado.

Se uma observação pôde ser classificada em n classes diferentes $c_1, \dots, c_n$, e a probabilidade de um objeto pertencer a classe c_i é $p(i)$, então, a entropia de classificação do ramo é dada por:

$$E(A = v_j) = - \sum_{i=1}^n p(i) \log_2 p(i) \quad (4.1)$$

onde $A = v_j$ significa que o atributo A tem o valor v_j . Isto é, $E(A = v_j)$ é a entropia do ramo correspondente ao valor v_j do atributo A . No caso do atributo inteligência tem-se que a entropia correspondente ao ramo de inteligência=não é:

$$\begin{aligned} E(\text{inteligência} = \text{nao}) &= -2/5 \log_2(2/5) - 3/5 \log_2(3/5) \\ &= 0.9709505 \end{aligned}$$

pois 2/5 das observações no ramo correspondente a pouca inteligência pertencem a classe negativa e 3/5 das observações pertencem a classe positiva. Analogamente, pode-se verificar que a entropia do ramo inteligência=sim é zero:

$$\begin{aligned} E(\text{inteligência} = \text{sim}) &= -3/3 \log_2(3/3) \\ &= -1 \log_2 1 \\ &= 0 \end{aligned}$$

Para calcular a entropia total de um atributo deve-se relacionar a entropia de cada um dos ramos correspondentes a este atributo. Seja:

M o número de observações totais,

$E_1, E_2, E_3, \dots, E_k$ as entropias de cada ramo pertencente ao atributo escolhido, e

$N_1, N_2, N_3, \dots, N_k$ o número de elementos de cada ramo.

Neste caso, a entropia de toda a ramificação, ou seja, a entropia do atributo A é dada por:

$$E(A) = \sum_{i=1}^k E_i * N_i / M \quad (4.2)$$

Por exemplo, no caso do atributo inteligência tem-se dois ramos, um contém três das oito observações totais e o outro possui cinco destas observações:

$$\begin{aligned} E(\text{inteligência}) &= 3/8 E(\text{inteligência} = \text{sim}) + 5/8 E(\text{inteligência} = \text{nao}) \\ &= 0 * 3/8 + 5/8 * 0.9709505 \\ &= 0.6068441 \end{aligned}$$

Realizando o mesmo processo com os outros dois atributos obtém-se:

$$E(\textit{beleza}) = 0.8112781$$
$$E(\textit{situac_financeira}) = 0.3443609$$

Portanto, o atributo *situac_financeira* é o melhor atributo, enquanto que *beleza* é o pior atributo para começar a construir a árvore de decisão — árvore da figura 4.6. Assim, para gerar árvores de decisão mais simples, em cada nó da árvore deve-se escolher o atributo de menor entropia para continuar ramificando a árvore.

4.2.2 O Sistema ID3

O sistema ID3 constrói uma árvore de decisão da forma acima explicada, utilizando o conceito de entropia para escolher, a cada passo, o melhor atributo.

Como ID3 foi idealizado para manusear grandes quantidades de dados, ele trabalha de forma iterativa. Inicialmente é escolhido aleatoriamente um subconjunto dos exemplos amostra, denominado janela (“window”), figura 4.7, e ID3 constrói uma árvore de decisão para estes dados na forma acima explicada. Esta árvore classifica corretamente os exemplos da janela. O resto dos exemplos são testados para ver se eles também são corretamente classificados por esta árvore. Se a árvore classifica todos os exemplos na forma certa, então o processo finaliza; senão, aqueles exemplos classificados incorretamente pela árvore são adicionados à janela e ID3 constrói novamente outra árvore de decisão. Este processo se repete até a árvore classificar corretamente todos os dados.

Geralmente não são necessárias muitas iterações para chegar na árvore final. Quinlan [Quinlan 86a] achou árvores de decisão corretas depois de poucas iterações para conjuntos de exemplos com mais de trinta mil objetos, descritos em termos de mais de cinquenta atributos. Embora este sistema não garanta a construção da árvore de decisão ótima, ele constrói boas árvores de decisão. Uma forma de achar a melhor árvore seria construir todas as possíveis árvores de decisão que classifiquem corretamente os dados e, entre elas, escolher a mais simples. Porém, o custo computacional desta abordagem é muito alto, em particular para grande quantidade de dados.

Os requisitos computacionais de ID3 para cada iteração são proporcionais ao produto do tamanho do conjunto amostra, do número de atributos e do número de nós não terminais da árvore. Quinlan observou que não há crescimento exponencial de tempo ou espaço quando as dimensões da tarefa indutiva são incrementadas. Por este motivo, esta técnica pode ser utilizada para tarefas de grandes proporções.

Mecanismo de Janela

Figura 4.7: Mecanismo de Janela do ID3

A seguir é esquematizado o algoritmo utilizado pelo sistema ID3, que opera sobre um conjunto de exemplos amostra, C , da seguinte forma:

1. Selecione aleatoriamente um subconjunto J — janela — do conjunto amostra, C , que contém todas as instâncias da amostra.
2. Construa uma árvore de decisão da seguinte forma:
 - (a) Se todas as instâncias em J pertencem a uma única classe, crie um nó desta classe e finalize.
Senão, selecione (utilizando algum critério heurístico) um atributo A , com valores $a_1, a_2, \dots, a_n$, e crie o seguinte nó da árvore:

- (b) Particione os exemplos do conjunto janela J nos subconjuntos $c_1, c_2, \dots, c_n$ segundo os valores de A .
 - (c) Aplique o algoritmo recursivamente para cada um dos subconjuntos c_i .
3. Selecione do conjunto total C todas aquelas observações que não são corretamente classificadas pela árvore de decisão acima criada.
4. Forme uma nova janela combinando os elementos que já estavam nela com aquelas observações achadas no passo anterior.
5. Repita os passos 2 a 4 até que a árvore classifique corretamente todos os exemplos do conjunto amostra C .

4.3 Família TDIDT

A seguir é descrita uma família de sistemas de aprendizado por exemplos, da qual ID3 é parte integrante, que têm fortes características em comum. Estes algoritmos são denominados TDIDT [Quinlan 86a]

Os sistemas desta família não são limitados a uma área de aplicação específica, tal como química, xadrez ou medicina, isto é, são sistemas de propósito geral. O objetivo de todos eles é classificar objetos. Eles produzem regras ou descrições de um determinado número de classes de objetos. Quando novos objetos são observados, estas regras devem prever

a que classe estes objetos pertencem. Por exemplo, as seguintes tarefas são tarefas de classificação:

- O diagnóstico de um problema de saúde a partir dos sintomas do paciente. Neste caso as classes são as diversas doenças ou as possíveis terapias;
- decidir, a partir de observações climáticas e condições dos solos, se uma determinada região geográfica é apta para um dado plantio ou não. Aqui as classes são *apta*, *não apta*;
- decidir, segundo diversas condições particulares e do mercado financeiro, qual o melhor investimento a realizar. Neste caso as classes poderiam ser: ouro, dólar, over, poupança, etc, e as regras geradas seriam do tipo:

Se (condições)
então aplique em (Classe_ Ki)

Os membros da família TDIDT são amplamente caracterizados por representarem seu conhecimento na forma de árvores de decisão. Esta forma de representação, relativamente simples, não tem o poder expressivo de redes semânticas ou outras representações de primeira ordem. Como consequência da simplicidade da representação do conhecimento utilizada, estes algoritmos são significativamente menos complexos que aqueles utilizados em outros sistemas que fazem uso de uma linguagem mais poderosa para expressar seus conhecimentos. Entretanto, é possível gerar conhecimento na forma de árvores de decisão capazes de resolver problemas difíceis.

A estratégia de aprendizado utilizada por estes sistemas é não incremental e seletiva — seções 3.3.3 e 3.3.4 — ou seja, os exemplos ou instâncias são apresentados ao sistema em forma simultânea. Além disso, o sistema constrói uma árvore de decisão em forma top-down, isto é, a estratégia de controle é “model driven” — seção 3.3.1.

Resumindo, estes sistemas desenvolvem árvores de decisão para tarefas de classificação. Estas árvores são construídas começando da raiz e descendo até as folhas (top-down). O nome desta família de sistemas enfatiza estas características: Top Down Induction Decision Trees (TDIDT).

Os objetos-exemplo, a partir dos quais são geradas as regras de classificação, são conhecidos somente através dos valores de um conjunto de atributos. Por sua vez, as árvores de decisão são construídas usando os mesmos atributos.

Os exemplos podem ter duas origens diferentes. Eles podem vir de uma base de dados existente, por exemplo, de uma que contenha um histórico de observações clínicas de pacientes. Objetos identificados em exemplos deste tipo dão um quadro estatístico confiável; entretanto, podem ser redundantes ou omissos com relação a casos raros que não tenham sido encontrados durante o período de registro das observações. Por outro lado, os exemplos podem ser um conjunto tutorial cuidadosamente escolhido e preparado

Figura 4.8: A Família TDIDT

por um especialista, onde cada exemplo diz respeito a uma regra de classificação completa e correta. Neste caso, o especialista deve tomar o cuidado de evitar casos redundantes e se lembrar de incluir exemplos de casos raros. Os sistemas aqui descritos trabalham satisfatoriamente com exemplos provenientes de qualquer uma destas duas origens.

A figura 4.8 mostra os sistemas que fazem parte da família TDIDT, junto com o ano de sua divulgação. Na figura são mostrados somente os sucessores diretos do CLS — Concept Learning System [Hunt 66]. Outros sistemas — como por exemplo o CN2 [Clark 87] — que foram influenciados pela família TDIDT [Gams 87], não são mostrados no desenho.

O algoritmo geral destes sistemas começa com uma árvore de decisão vazia, que é refinada gradualmente até que classifique corretamente todos os exemplos da amostra. O procedimento geral é o seguinte:

Dados um conjunto de exemplos de aprendizado E ;
 uma condição de parada $t(E)$; e
 uma função de avaliação $aval(E, atributo)$

Se todas as instâncias em E satisfazem a condição de término $t(E)$

então retorne o valor da classe

caso contrário

1. para cada atributo a_i determine o valor da função $aval(E, a_i)$. Seja a_j o atributo que possui o melhor valor ¹ de $aval(E, a_i)$.
2. se a_j possui valores $a_{j1}, a_{j2}, \dots, a_{jk}$ crie o seguinte nó da árvore:

¹Atributo mais relevante para realizar a ramificação, segundo a função de avaliação.

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

O critério de parada t pode ser definido tanto para construir a árvore de decisão, que classifica exatamente todos os elementos do conjunto de aprendizado em domínios completos — ID3 —, quanto para decidir pela não expansão da árvore quando a evidência for insuficiente nos exemplos fornecidos, este mecanismo é denominado pré-poda da árvore de decisão — C4, Assistant-86. A função de avaliação *aval* pode ser definida de várias formas diferentes.

As diversas condições de parada e funções de avaliação dão origem a vários algoritmos desta família. Por exemplo, o sistema ID3 utiliza a “entropia” como função de avaliação dos atributos, e seu critério de parada é que todos os exemplos no nó pertençam à mesma classe. Embora esta abordagem dê bons resultados neste sistema, nada garante que esses sejam os melhores resultados. Outro critério ou função de avaliação poderia ser utilizado.

Entre as diversas funções de avaliação podem ser citadas [Mingers 89a]:

- *Entropia* ou *Quinlan Information Measure (IM)* [Quinlan 79,83],
- *Medida da Razão do Ganho* ou *Gain-Ratio Measure (GR)*: medida proposta por Quinlan [Quinlan 86a] como uma correção do *IM*,
- *Estatística Qui-Quadrado* (χ^2): função baseada na estatística do Qui-Quadrado, proposta por Hart [Hart 85a] e Mingers [Mingers 87a],
- *A estatística G* (G): medida proposta por Mingers [Mingers 87a]
- *Índice de Diversificação GINI* ($GINI$) medida proposta por Breiman et al. [Breiman 84], similar a *IM*.

O primeiro destes sistemas, o CLS (Concept Learning System) [Hunt 66], constrói uma árvore de decisão que tenta minimizar o custo de classificar um objeto. Este custo tem componentes de dois tipos: o custo de determinar o valor de um dado atributo do objeto e o custo de classificar erroneamente o objeto. Em cada estágio, CLS explora o espaço de possíveis árvores de decisão de uma determinada altura, escolhe uma estratégia para

minimizar o custo neste espaço limitado e, a seguir, desce um nível na árvore. Dependendo da altura escolhida para explorar, CLS requer uma grande quantidade de cálculo.

ID3 [Quinlan 79, 83] foi desenvolvido a partir do sistema CLS, tentando melhorar algumas de suas características. Este é um dos sistemas mais conhecidos da família TDIDT. ID3 constrói a árvore de decisão numa forma iterativa, abandonando a abordagem de cálculo de custo e utilizando em seu lugar uma função que estima uma determinada medida de informação denominada entropia.

O sistema ACLS (Analog Concept Learning System) [Paterson 83] é uma generalização de ID3. Tanto em ID3 como em CLS, todos os atributos utilizados para descrever objetos podem assumir um número finito de valores pertencentes a um conjunto predeterminado. Por exemplo, o atributo cor pode assumir os valores preto, branco, ... verde e nenhum outro que não tenha sido estabelecido previamente. Já o sistema ACLS permite que os atributos utilizados também possam assumir qualquer valor numérico inteiro. Esta característica tem permitido que ACLS seja aplicado em tarefas difíceis, tais como reconhecimento de imagens.

Os quatro sistemas na figura 4.8 que estão abaixo do ACLS são produtos comerciais, produzidos por Intelligent Terminals, descendentes diretos deste último sistema. Embora esses sistemas não apresentem inovações nas estratégias utilizadas, eles incorporaram interfaces amigáveis e alguns outros utilitários para facilitar a tarefa de construir regras a partir de exemplos. Expert-Ease, por exemplo, depois de gerar as regras relativas às observações correspondentes, cria automaticamente um sistema de consulta — um Sistema Especialista que pode diagnosticar, baseando-se nas regras que aprendeu [Ford 87]. Todos estes sistemas foram bem sucedidos comercialmente [Mowfort 86].

Assistant [Kononenko 84] também é um descendente de ID3. Este sistema permite atributos com valores discretos e contínuos — atributos contínuos são representados por intervalos de reais —, bem como permite que as classes formem uma hierarquia; as classes não precisam ser necessariamente disjuntas, uma classe pode ser uma subdivisão de outra. Assistant não forma a árvore de decisão iterativamente como ID3; ele possui um algoritmo para escolher um “bom” conjunto amostra entre os exemplos disponíveis.

Assistant-86 [Cestnik 87] é uma implementação de Assistant para microcomputador tipo IBM-PC, rodando com o sistema operacional DOS. Todas as características do sistema Assistant estão também presentes em Assistant-86, porém, com algumas ligeiras modificações. Desta forma, Assistant-86 manuseia atributos com valores contínuos, dados incompletos e com ruído e domínios incompletos, isto é, domínios onde os atributos não são suficientes para a tarefa de classificação.

C4 [Quinlan 86c] utiliza basicamente o mesmo procedimento de ID3, mas tem incorporadas melhorias que tratam com atributos com valores discretos ou contínuos, valores desconhecidos de atributos e dados com ruído. Para tratar dados com ruído, C4 utiliza um critério de parada [Breiman 84], [Kononenko 84], [Quinlan 86b], e realiza também uma poda na árvore. O sistema gera várias árvores podadas, escolhendo no final a árvore mais promissora.

Os sistemas ID4 [Schlimmer 86], ID5 [Utgoff 88] e ID5R [Utgoff 89] são frutos de pesquisas recentes que procuram resolver o problema de tornar o sistema ID3 incremental.

4.4 Histórico de Sistemas de Aprendizado por Exemplos

O sistema CLS (Concept Learning System), idealizado por Hunt, Marin e Stone [Hunt 66], foi o primeiro a gerar automaticamente regras através de exemplos. As regras eram produzidas na forma de árvores de decisão. Michie e Chambers [Michie 68, 69] desenvolveram indutivamente uma base de regras para um sistema de tempo real — BOXES — que aprendia a balançar uma vara sobre um carrinho em movimento. Este sistema aprendia por si só, mediante tentativa e erro, ou observando as decisões tomadas por uma pessoa, especialmente treinada, enquanto ela realizava a mesma tarefa.

Estes tipos de sistemas foram introduzidos em aplicações na área de visão por computador. Tanto Winston [Winston 75] quanto Barrow e Popplestone [Barrow 71] introduziram grafos relacionais (redes semânticas) para descrever imagens. Seus programas modificavam as descrições visuais através de imagens dos exemplos.

Em diversos trabalhos, Michalski demonstrou as vantagens de utilizar métodos indutivos, em vez dos métodos tradicionais, para construir a Base de Conhecimento de um Sistema Especialista [Michalski 76, 80b]. As regras geradas por estes sistemas indutivos eram legíveis ao usuário, porém não na medida ideal.

Quinlan [Quinlan 79] modificou o sistema CLS tornando-o mais eficiente, e conseguindo manipular grandes quantidades de exemplos. Isto foi realizado adicionando incrementalmente uma seleção de contra-exemplos; desta forma, em cada interação, as regras eram reinduzidas e melhoradas. Posteriormente, Quinlan continuou trabalhando no seu sistema ID3 — Iterative Dichotomizer Tree —, produzindo descrições diretamente executáveis no computador para um certo domínio no jogo de xadrez [Quinlan 83]. Estas descrições eram cinco vezes mais eficientes, em termos de tempo de execução, do que as melhores descrições produzidas por humanos para esse mesmo problema de xadrez. Nesse mesmo artigo, Quinlan demonstrou que o erro introduzido nas descrições induzidas era pequeno e apenas função da amostra-exemplo.

Se bem os métodos acima descritos provaram ser ferramentas úteis para a construção e aquisição de regras, muitas vezes as regras por eles geradas não eram muito legíveis ou claras para o usuário. Esta dificuldade é acentuada em problemas de domínio muito amplo. Shapiro e Nibbellet [Shapiro 82, 83] foram os primeiros a desenvolver técnicas de indução estruturada para gerar regras que fossem mais inteligíveis aos seres humanos. Atualmente esta é uma das linhas de pesquisa nesta área [Arbab 85],[Shapiro 87].

Outras linhas de pesquisa tentam resolver os problemas abaixo, para tornar este tipo de sistemas mais eficiente:

- tornar o sistema ID3 incremental [Schlimmer 86] [Utgoff 88] [Utgoff 89],
- permitir que o especialista tenha um papel mais ativo no processo indutivo — *indução interativa* — [Buntine 88] [Kodratoff 87] [Michie 87],
- resolver os problemas referentes à incerteza nas regras geradas [Carvalho Gomes 88] [Quinlan 87b],
- resolver os problemas produzidos por ruído nos dados — em geral, através da utilização de métodos de poda na árvore — [Clark 87] [Mingers 89b] [Niblett 87] [Quinlan 87a],
- introduzir melhoras nos métodos de seleção de atributos [Hart 85b] [Mingers 89a] [Quinlan 88]

4.5 Vantagens e Desvantagens

Entre as vantagens de se utilizar árvores indutivas de decisão, podem ser citadas:

1. Árvores de decisão são inerentemente disjuntivas, pois cada ramo que sai de um nó de decisão corresponde a diferentes casos disjuntivos. Por exemplo, a árvore da figura 4.9 é equivalente à seguinte expressão em cálculo de predicados

$$\neg forma(x, triang) \vee \neg forma(x, quadrado) \vee forma(x, circulo) \\ \wedge [cor(x, verde) \vee \neg cor(x, azul)].$$

Figura 4.9: Círculo Verde

Consequentemente, árvores de decisão podem ser utilizadas para representar conceitos disjuntivos como “círculo verde ou triângulo vermelho”. Por este motivo, os

sistemas da família TDIDT podem adquirir e representar facilmente conceitos disjuntos; outros sistemas indutivos apresentam problemas para lidar com este tipo de conceito.

2. Os sistemas que utilizam árvores de decisão podem ser facilmente adaptados para manusear dados com ruído.
3. Os algoritmos para construir árvores de decisão executam pouca busca. Eles se baseiam numa função de avaliação para selecionar, em cada nó, o melhor atributo. Como já foi mencionado anteriormente, a simplicidade do formato de representação destes sistemas é contrabalanceada por sua eficiência.
4. Os sistemas que utilizam árvores de decisão conseguem manusear com facilidade um conjunto amostra com muitos elementos, descritos por um grande número de atributos. As árvores de decisão geradas são razoavelmente boas, e não requerem muita computação.

Entre as desvantagens na utilização dessa técnica, podem ser mencionadas as seguintes:

1. Existem diversas árvores que representam um mesmo conceito. Isto dificulta a verificação de árvores de decisão equivalentes.
2. Um outro problema é que, geralmente, é difícil para as pessoas entenderem um conceito representado como uma árvore de decisão grande.
3. Estes sistemas também estão limitados a uma representação de atributos e valores, enquanto que outros sistemas podem tratar descrições relacionais ou estruturais.
4. A natureza não incremental destes sistemas os faz ineficientes quando são incorporadas novas instâncias, uma vez que eles devem recalcular a árvore de decisão desde o começo.

Eventualmente, no sistema ID3, para o processo convergir pode ser necessário que a janela contenha todos os exemplos considerados. Na prática, entretanto, este problema não foi observado [Quinlan 86a].

Na utilização destes sistemas é importante destacar que a qualidade dos resultados obtidos, isto é, a qualidade das regras, depende da qualidade dos exemplos de entrada. Estes sistemas não podem descobrir conhecimento que não está presente ou expresso de uma forma adequada nos dados de entrada. Como consequência, um conjunto de exemplos pobre gerará resultados também pobres [Hart 85a].

Por exemplo, para descrever os objetos da amostra é melhor pecar por excesso de atributos do que por falta destes. Se houver atributos redundantes, o sistema consegue identificá-los e estes não são utilizados. Porém, se atributos relevantes estiverem ausentes, o sistema pode ter problemas para gerar uma árvore de decisão correta, ou então pode gerar árvores

ineficientes. Entretanto, quando utilizados apropriadamente, estes sistemas têm obtido muito sucesso em diversas áreas de aplicação, desde que vários deles estão disponíveis comercialmente [Mowforth-86].

4.6 Considerações Finais

Neste capítulo foi descrito o processo de construção de árvores de decisão através de um exemplo detalhado. A seguir foi apresentado o sistema ID3 que utiliza como função de avaliação a entropia. Outros sistemas pertencentes à família TDIDT também foram introduzidos.

É importante ressaltar que os vários algoritmos dessa família se originam, entre outros, das diversas funções de avaliação, e dos critérios de parada utilizados na construção da árvore de decisão.

Como já foi mencionado, embora estes sistemas trabalhem com um esquema de representação relativamente simples — conjuntos de pares (*atributo, valor*) e árvores de decisão — eles têm sido utilizados satisfatoriamente como ferramenta para auxiliar a construção de Bases de Conhecimento.

Capítulo 5

Implementação de um Sistema para Construção de Árvores Indutivas de Decisão

Como resultado prático deste trabalho foi implementado um sistema para construir árvores indutivas de decisão, na linguagem de programação Prolog. A seguir, são discutidas as principais idéias utilizadas na implementação. Uma documentação completa da implementação pode ser encontrada em [Castiñeira 90a].

5.1 Considerações Gerais da Implementação

É possível realizar uma implementação geral dos algoritmos da família TDIDT redefinindo, para algoritmos particulares, o critério de parada t e a função de avaliação *aval*. Esta foi a idéia adotada neste sistema, implementado na linguagem de programação Prolog [Arity 88]. Várias das idéias e técnicas avançadas usadas na implementação foram extraídas de [Turing 87].

As estruturas de dados utilizadas são listas, árvores e árvores binárias. Com o objetivo de realizar o cálculo da função de avaliação *aval* eficientemente, é construída uma árvore binária duplamente indexada. De forma geral, a árvore de decisão construída tem a seguinte estrutura:

$$\textit{Melhor Atributo} - [v_1 : [E_{11}, E_{12}, \dots], v_2 : [E_{21}, E_{22}, \dots], \dots, v_n : [E_{n1}, E_{n2}, \dots]]$$

onde $v_1, v_2, \dots, v_n$ são os valores que *Melhor Atributo* pode assumir e $[E_{i1}, E_{i2}, \dots]$ é a lista de objetos tal que o valor de *Melhor Atributo* é v_i . As subárvores E_{ij} possuem estrutura semelhante.

O sistema também possui opções para construir a árvore com informações adicionais, tais como o número total de exemplos classificados em cada um dos nós da árvore, o número de exemplos desclassificados, etc.

A implementação realizada é abrangente e leva em conta várias situações que podem acontecer durante o processo de construção da árvore, bem como durante o processo de avaliação e/ou poda.

Um problema, referente à eficiência computacional, aparece quando o número de exemplos do conjunto de aprendizado E é grande. Para este caso, foi implementado o mecanismo de *janela*. Este mecanismo consiste em selecionar aleatoriamente um subconjunto S de exemplos do conjunto E . A árvore de decisão é gerada com os exemplos contidos no subconjunto S , e é verificado se os exemplos não considerados são também classificados por esta árvore. Se houver exemplos não classificados, eles são adicionados à janela e o processo de construção da árvore é repetido até não existir mais exemplos não classificados pela árvore de decisão.

Durante o processo de construção da árvore, não é possível classificar exatamente exemplos com igual valor de atributos que pertencem a classes diferentes. Neste caso, após a manipulação de todos os exemplos do conjunto de aprendizado, a estratégia adotada nesta implementação é a de rotular essa folha com o valor da classe mais provável.

Outro problema tratado é o seguinte: algumas implementações precisam conhecer — durante o processo de construção da árvore — todos os valores de um atributo e, se um determinado valor de um atributo considerado pela árvore de decisão não estiver no conjunto de aprendizado, são criadas folhas adicionais rotuladas *null*. Esta abordagem, que requer conhecimento não somente dos atributos, mas de todos os possíveis valores que cada atributo pode assumir, exige o pré-processamento de todo o conjunto de exemplos. Além disso, não há garantia que exemplos do conjunto de teste — considerado posteriormente — possuam os mesmos valores de atributos que o conjunto de aprendizado.

Para resolver este problema, as folhas *null* são criadas dinamicamente — após a construção da árvore — se necessário. Isto acontece em uma situação em que for processado um exemplo que não consegue ser classificado pela árvore por possuir um valor de atributo que não está presente na árvore. Tal fato, significa que não é possível uma classificação exata para esse exemplo; contudo, há várias formas de contornar este problema. Uma delas, a adotada neste trabalho, é a de computar as probabilidades condicionais de cada classe no ramo que vai da raiz à folha *null* da árvore, e rotular a folha *null* com a classe mais provável.

5.2 Especificação da Implementação

No nível mais alto do programa tem-se:

```
id3_Turing(T):-
    atributos(A),
    exemplos(E),
    forme_arvore(E,A,T).
```

onde os dois primeiros programas: `atributos/1` e `exemplos/1` fazem o pré-processamento dos dados de entrada.

O programa `atributos/1` retorna uma lista contendo todos os atributos dos objetos considerados. Para entender melhor o funcionamento do programa será considerado o conjunto de dados da tabela 4.1. Com estes dados, o predicado `atributos/1` retornará a seguinte lista:

```
[inteligencia, beleza, dinheiro]
```

O programa `exemplos/1` constrói uma lista contendo todos os exemplos ou objetos a serem considerados. Cada um destes objetos é, por sua vez, uma lista de atributos e valores. No caso do exemplo considerado será retornada a lista de dados mostrada na figura 5.1.

```
[[inteligencia(s), beleza(b), dinheiro(r), classe(cp)],
 [inteligencia(n), beleza(f), dinheiro(p), classe(cn)],
 [inteligencia(s), beleza(f), dinheiro(p), classe(cp)],
 [inteligencia(s), beleza(f), dinheiro(m), classe(cp)],
 [inteligencia(n), beleza(b), dinheiro(p), classe(cn)],
 [inteligencia(n), beleza(b), dinheiro(m), classe(cp)],
 [inteligencia(n), beleza(b), dinheiro(r), classe(cp)],
 [inteligencia(n), beleza(f), dinheiro(r), classe(cp)]]
```

Figura 5.1: Lista de Dados

O terceiro predicado do programa — `forme_arvore/3` — é o responsável pela construção da árvore de decisão.

A idéia geral do programa `forme_arvore/3` é a seguinte: dada a lista de exemplos e atributos, obter o melhor atributo segundo um determinado critério, particionar os dados segundo os valores deste melhor atributo, retirar este melhor atributo da lista de possíveis atributos e chamar recursivamente a si próprio para continuar ramificando cada uma dessas partições.

A entrada explícita para o programa consiste da lista de exemplos — “janela” — e a lista que representa o conjunto de atributos. A entropia é a função de avaliação utilizada para selecionar o melhor atributo na construção da árvore de decisão. Desejando utilizar uma

outra função de avaliação, basta substituir o programa que realiza o cálculo da entropia por aquele que utilize a função desejada.

Para realizar o cálculo da entropia mais eficientemente, é construída uma árvore binária duplamente indexada. Isto é feito por razões de eficiência, uma vez que o cálculo da entropia consome um tempo razoável, e além disso tal procedimento deve ser efetuado em cada nó da árvore de decisão, para cada um dos atributos. A árvore binária é indexada segundo os valores do atributo e da classe.

Para inserir cada exemplo na árvore, é necessário saber os valores do atributo e da classe. Cada um destes valores é obtido, respectivamente, pelos programas `valor_at/3` e `class/2`. A implementação destes dois programas depende da representação de dados adotada para descrever os exemplos.

Depois do melhor atributo ser selecionado, a árvore duplamente indexada correspondente a este atributo é também utilizada para ramificar os exemplos, segundo os valores deste atributo. Isto é, ela também auxilia a construção da árvore indutiva.

O critério de parada é a condição de todos os exemplos no nó pertencerem a mesma classe, ou então não existir mais atributos para continuar ramificando.

A saída do programa é a árvore de decisão. Para o caso do exemplo considerado o programa `forme_arvore/3` fornece a seguinte solução:

`dinheiro - [r:cp, p:inteligencia - [n:cn, s:cp], m:cp]`

ou, graficamente:

Além do sistema para construção da árvore, também foi implementado um método específico de poda, que será discutido no próximo capítulo.

5.3 Considerações Finais

Neste capítulo foram descritas as idéias utilizadas na implementação de um sistema para construir árvores de decisão indutivas.

A implementação realizada é abrangente e permite, fazendo algumas modificações, utilizar diversas funções de avaliação e critérios de parada. Ela também confirma que a utilização de estruturas de dados apropriadas pode influenciar diretamente a eficiência do sistema. Em particular, a estrutura de árvore duplamente indexada, utilizada para calcular a entropia, faz com que este sistema fique muito mais rápido quando comparado com outras implementações.

A implementação realizada utiliza técnicas avançadas de programação Prolog que, infelizmente, não são publicadas. Uma maior divulgação destes tipos de técnicas com certeza contribuiriam para uma melhora no nível dos programas Prolog. Esse problema já foi evidenciado e comentado por Pereira [Pereira 87].

Como já foi mencionado, a documentação completa da implementação pode ser encontrada em [Castiñeira 90a]. Com esta publicação, espera-se contribuir para a difusão de técnicas avançadas de programação, com o fim de auxiliar futuros projetistas e programadores Prolog.

Capítulo 6

Poda em Árvores de Decisão

A poda em uma árvore de decisão é um mecanismo através do qual se procura evitar a construção de subárvores não confiáveis. Este mecanismo serve também para reduzir o tamanho de árvores de decisão e melhorar a precisão da classificação.

Há dois tipos de poda em árvore de decisão: *pré-poda*, realizada durante o processo de construção da árvore e *pós-poda*, realizada após a árvore de decisão ter sido construída.

Pré-poda é usada para parar a ramificação de um nó da árvore, quando a classificação correspondente deixa de ser confiável. A pós-poda é utilizada após a construção da árvore, quando o erro de classificação de novos exemplos é calculado e, em função desse erro, é decidido podar algumas subárvores.

Em geral, a pós-poda na árvore de decisão produz resultados mais satisfatórios que a pré-poda, uma vez que, com a pós-poda, tanto a exatidão quanto a complexidade das árvores de decisão tendem a melhorar [Hart 85a]. Portanto, é importante analisar a variação da exatidão e complexidade de árvores de decisão — geradas com exemplos de diversos domínios — após a aplicação do mecanismo de pós-poda.

Diversos métodos de poda têm sido propostos [Mingers 89b]; neste trabalho foi utilizado uma variação do *método de redução do erro* [Quinlan 87a].

Esse método utiliza um outro conjunto de dados, denominado *conjunto teste*, para realizar a poda. A árvore de decisão obtida é aquela que produz o menor número de exemplos não classificados para esse conjunto. O método é descrito a seguir.

Todos os exemplos do conjunto teste são verificados pela árvore de decisão e, em cada nó da árvore, o número de cada uma das classes é contado. Para cada nó não terminal i é contado o número de exemplos não classificados d_i , bem como o número de exemplos, dp_i , que seriam não classificados se esse nó fosse convertido numa folha como consequência da poda. Seja $g_i = d_i - dp_i$. O valor g_i , se positivo, é uma medida do ganho em precisão obtido podando a subárvore i . Isto é, $g_i > 0$ significa que um número maior de exemplos será bem classificado pela árvore de decisão, se a subárvore correspondente ao nó i for

podada ou, em outras palavras, que a precisão da árvore será incrementada podando esse nó.

Este processo é repetido em cada nó, a partir da raiz, descendo até as folhas, realizando a poda da correspondente subárvore se o ganho for positivo ou nulo. A diferença do método proposto com o método da redução do erro é que, este último, realiza a poda começando dos nós terminais, e subindo em direção à raiz.

A poda, quando o ganho é nulo, não incrementa a precisão da árvore, mas favorece árvores de menor tamanho. O processo finaliza quando não é mais possível diminuir o valor do erro. Dessa forma, obtém-se a menor árvore podada, que melhor classifica o *conjunto teste*.

6.1 Resultados Experimentais

A seguir, são apresentados os resultados obtidos aplicando o método de poda acima descrito em diversos domínios. As medidas foram realizadas sobre 7 conjuntos de dados; 4 deles são resultados de observações reais e 3 foram construídos artificialmente. Segue-se uma descrição dos dados utilizados.

6.1.1 Conjunto de Dados

Classificação de três tipos de Iris (Iris). São 150 exemplos de três diferentes tipos de planta com 50 exemplos de cada tipo. Os quatro atributos são valores contínuos: largura e comprimento da pétala e largura e comprimento da sépala. Estes dados foram obtidos de [Kendall 76], onde são utilizados em testes de análise discriminante.

Reconhecimento de dígitos em um “display” digital (LDE). Este domínio foi sugerido por Breiman et al [Breimann 84]. Cada dígito de um “display” digital é constituído por 7 linhas, cada uma das quais pode estar ligada ou desligada. As diferentes combinações das linhas ligadas geram os dez dígitos. Estes dígitos constituem as classes e há sete atributos binários, um por cada linha, que são utilizados para reconhecer cada uma destas classes. Este domínio, porém, não é exato. Assume-se que cada uma das linhas tem 10% de probabilidade de estar com defeito (isto é, estar ligada quando deveria estar desligada ou vice-versa). Este erro introduzido afeta o valor dos atributos mas não o valor da classe. Isto significa que o valor da classe é o do dígito que deveria aparecer no “display”, mesmo que não seja este o dígito mostrado. Este domínio contém muito ruído, pois a probabilidade da classe estar certa é $0.9^7 = 0.48$. Trezentos exemplos foram gerados aleatoriamente.

Diagnóstico de crianças neonatas (Pediatria). Estes dados provém de observações de duzentas e quarenta crianças recém-nascidas. Para cada uma delas foi diagnosticado o estado nutricional: PIG ou AIG (Pequena ou Adequada para a Idade Gestacional). Além

disso, foram realizadas diversas medições como peso, comprimento, perímetro craniano, dobras cutâneas, etc., e considerados os resultados de diversos exames laboratoriais tais como índice de colesterol, hemoglobulina, proteínas, etc. A classe, com dois valores, é o estado nutricional da criança. Os atributos, dezessete no total, são as observações acima mencionadas, junto com o sexo e idade gestacional da criança.

Classificação probabilística de disjunções (Prob-Disj). Este domínio artificial é utilizado por Quinlan [Quinlan 87a]. São consideradas duas classes — positiva e negativa — e nove atributos binários $a_0, a_1, \dots, a_8$. O critério utilizado para gerar os exemplos é o seguinte:

se $(a_0 \wedge a_1 \wedge a_2) \vee (a_3 \wedge a_4 \wedge a_5) \vee (a_6 \wedge a_7 \wedge a_8)$
 então a classe é positiva com probabilidade 0.9
 e negativa com probabilidade 0.1
 senão a classe é negativa com probabilidade 0.9
 e positiva com probabilidade 0.1

Como a classe de cada exemplo é determinada probabilisticamente, nenhum procedimento de classificação pode atingir mais do que 90% de precisão nesta tarefa. Trezentos exemplos foram gerados segundo o critério acima.

Classificação de disjunções (Disjunções). Este domínio foi criado com a idéia de testar a classificação em uma área onde a maioria dos casos pertence a uma única classe e a ocorrência de casos pertencentes às outras classes é pequena. A descrição do domínio é similar à descrição do domínio anterior. São consideradas duas classes — positiva e negativa — e nove atributos binários $a_0, a_1, \dots, a_8$. O critério utilizado para gerar os casos é o seguinte:

se $(a_0 \wedge a_1 \wedge a_2) \vee (a_3 \wedge a_4 \wedge a_5) \vee (a_6 \wedge a_7 \wedge a_8)$
 então a classe é positiva
 senão a classe é negativa

A classificação neste caso é exata, porém, 90% dos exemplos pertencem à classe positiva e só 10% à classe negativa. Trezentos exemplos foram gerados.

Classificação de duas gramíneas segundo a sua produtividade (Capim-1). Estes dados, com 150 exemplos, referem-se ao estudo de produtividade de duas diferentes gramíneas tratadas com 5 diferentes fosfatos. Por sua vez, cada fosfato foi utilizado em 5 diferentes dosagens. O experimento foi instalado em parcelas subdivididas com 3 repetições. As gramíneas são classificadas segundo a sua produtividade (matéria seca por hectare) sendo as demais características tratadas como atributos (tipo de capim, fertilizante, dose e número da repetição).

Classificação de duas gramíneas segundo a sua produtividade sem considerar a repetição (Capim-2). Aqui são utilizados os mesmos dados acima mencionados, mas sem o atributo *repetição*. Assim, exemplos com igual valor de atributos podem ter classificações diferentes, ou seja, não é possível uma classificação exata dos dados.

6.1.2 Método Experimental

Para cada um dos domínios acima descritos, e após embaralhar os dados, foi criado um conjunto de aprendizado – E – contendo 60% dos exemplos; os 40% dos exemplos restantes foram divididos em dois conjuntos – C_p e C_e – com igual número de exemplos. Assim, foi possível considerar que a proporção de exemplos que pertencem a cada uma das classes está dividida uniformemente nos três conjuntos. Foram criados 10 grupos diferentes de conjuntos (E, C_p, C_e) para cada domínio.

Os conjuntos de aprendizado foram usados para criar, em cada domínio, 10 árvores de decisão utilizando a entropia como função de avaliação. Em cada caso, foi usado o conjunto correspondente C_p para podar a árvore e o outro conjunto C_e para verificar o erro de classificação da árvore antes e após a poda, figura 6.1.

6.1.3 Resultados

Usualmente, a complexidade de árvores de decisão é medida pelo número de nós e folhas. Cada folha corresponde a uma classificação. A altura da árvore indica o número máximo de atributos considerados na classificação dos exemplos. O Comprimento Médio de Caminho – CMC – da árvore dá uma idéia mais geral da complexidade da árvore em função desses parâmetros.

As Tabelas 6.1 e 6.2 mostram, respectivamente, o valor médio do número de folhas, de nós e do CMC das árvores construídas em cada domínio, antes e após aplicar o mecanismo de poda. A poda da árvore foi sempre realizada utilizando o conjunto C_p de exemplos. Em todos os casos, os números entre parênteses indicam o desvio padrão.

Domínio	Altura	Nº Folhas	Nº de Nós	CMC
Iris	2.10 (0.30)	19.00 (1.94)	22.40 (2.28)	1.39 (0.05)
Pediatria	4.60 (0.66)	31.10 (5.94)	43.00 (7.98)	2.73 (0.16)
Capim-1	4.00 (0.00)	47.30 (5.60)	64.30 (7.10)	3.18 (0.07)
Disjunções	5.30 (0.45)	16.70 (2.32)	32.40 (4.65)	3.50 (0.14)
Capim-2	3.00 (0.00)	37.60 (3.47)	48.10 (4.23)	2.64 (0.03)
Prob-Disj.	9.00 (0.00)	52.60 (7.68)	109.50 (17.45)	5.50 (0.22)
LDE	7.00 (0.00)	56.00 (1.61)	130.00 (4.47)	5.36 (0.04)

Tabela 6.1: Árvore sem Poda

Deve ser observado que os quatro primeiros domínios — Iris, Pediatria, Capim-1 e Disjunções — contém dados determinísticos. Isto é, se 100% dos exemplos destes domínios fossem usados para criar a árvore de decisão, todos esses exemplos seriam corretamente classificados. Entretanto, deve ser ressaltado que a correta classificação de um conjunto de

Figura 6.1: Divisão dos Dados nos Conjuntos E , C_p e C_e

exemplos, não necessariamente fornece informação útil. O domínio do Capim-1 evidencia isso.

Domínio	Altura	Nº Folhas	Nº de Nós	CMC	% Redução
Iris	1.40 (0.49)	14.80 (4.19)	16.60 (5.35)	1.07 (0.19)	23
Pediatria	1.80 (0.98)	6.00 (2.10)	7.80 (2.99)	1.08 (0.48)	60
Capim-1	2.40 (0.92)	6.90 (3.91)	9.50 (5.01)	1.66 (0.66)	48
Disjunções	3.70 (1.90)	7.20 (3.63)	13.40 (7.26)	2.19 (1.13)	37
Capim-2	1.40 (0.92)	3.90 (3.30)	5.30 (4.20)	0.97 (0.65)	63
†	2.33 (0.94)	9.55 (6.65)	12.44 (8.32)	1.66 (0.71)	37
‡	2.55 (0.83)	13.00 (8.64)	16.77 (10.78)	1.89 (0.68)	28
Prob-Disj.	5.30 (0.90)	10.00 (3.19)	19.00 (6.39)	3.04 (0.43)	45
†	6.60 (1.28)	18.30 (6.02)	35.60 (12.03)	3.73 (0.60)	32
‡	7.50 (0.92)	28.10 (4.90)	55.20 (9.81)	4.41 (0.25)	20
LDE	6.50 (0.67)	17.30 (2.97)	33.60 (5.94)	3.70 (0.23)	31
†	6.80 (0.40)	23.10 (5.19)	45.20 (10.37)	4.02 (0.27)	25
‡	6.70 (0.46)	27.10 (4.23)	53.20 (8.46)	4.22 (0.22)	21

Tabela 6.2: Árvore com Poda

Os três últimos domínios — Capim-2, Prob-Disj e LDE —, contém dados que possuem um certo grau de incerteza. Para cada um destes três domínios, além da poda da árvore usando o conjunto C_p , foram realizadas duas experiências adicionais — linhas † e ‡. Os resultados nas linhas rotuladas † referem-se à poda da árvore usando C_p e 25% dos exemplos de E e, os das linhas rotuladas ‡, C_p e 50% de E .

A coluna Redução da Tabela 6.2 mostra o valor médio da redução da complexidade da árvore podada em função do CMC. Pode ser observado que a redução é significativa e, nos domínios com incerteza degrada-se, como era de se esperar, quando exemplos do conjunto E são também usados para podar a árvore.

Ainda que a árvore podada seja bem menos complexa que a árvore original, é fundamental verificar a sua precisão.

Um método, frequentemente utilizado para medir a precisão de uma árvore de decisão, é o de determinar o erro (ou precisão) na classificação de novos exemplos. A tabela 6.3 mostra os resultados obtidos. O erro foi sempre determinado usando o conjunto C_e de exemplos. As colunas Erro e Precisão mostram, respectivamente, o erro e a precisão equivalente da árvore antes e após a poda.

Pode ser observado que, na maioria dos casos, a precisão da árvore podada é semelhante a da original. No caso de Capim-1 e Capim-2 ela aumenta consideravelmente, mas, como foi observado anteriormente, este domínio não é representativo. Isto é evidenciado pelos resultados obtidos quando 50% dos exemplos do conjunto E , utilizados para a construção

da árvore original, são também utilizados para determinar a precisão da árvore podada — linha ‡ de Capim-2. A precisão da árvore podada diminui, o que mostra a não representatividade do conjunto de exemplos *E*.

Domínio	Erro		% Precisão	
	Sem Poda	Com Poda	Sem Poda	Com Poda
Iris	0.09 (0.05)	0.07 (0.03)	96-86	96-90
Pediatria	0.15 (0.05)	0.13 (0.06)	90-80	93-81
Capim-1	0.40 (0.08)	0.19 (0.06)	68-52	87-75
Disjunções	0.07 (0.04)	0.08 (0.03)	97-89	95-89
Capim-2	0.39 (0.07)	0.24 (0.09)	68-54	85-67
†		0.21 (0.06)		85-73
‡		0.24 (0.05)		81-71
Prob-Disj.	0.25 (0.05)	0.31 (0.08)	80-70	77-61
†		0.20 (0.03)		83-77
‡		0.18 (0.03)		85-79
LDE	0.32 (0.06)	0.35 (0.05)	74-62	70-60
†		0.29 (0.04)		75-67
‡		0.27 (0.05)		78-68

Tabela 6.3: Erro-Precisão da Árvore

Pretende-se, no futuro, pesquisar esta última característica, pois acredita-se que ela possa dar uma idéia objetiva da representatividade dos dados utilizados na tarefa de classificação.

6.2 Considerações Finais

Neste capítulo foi analisado um método de poda específico, que é uma variação do método de redução do erro, bem como sua aplicação em diversos domínios. Os resultados obtidos foram discutidos quantitativamente em função da complexidade e precisão das árvores e foram considerados animadores.

Deve ser observado que o método de poda proposto é mais drástico que o método de redução do erro. Pretende-se, no futuro, analisar outros métodos de poda, verificando seus comportamentos em domínios reais e artificiais, bem como realizar a transformação das árvores de decisão em regras de produção, incluindo fatores de certeza, a fim de verificar qualitativamente as regras de conhecimento geradas.

Os resultados expostos neste capítulo foram submetidos ao VII Simposio Brasileiro de Inteligência Artificial [Castiñeira 90b].

Capítulo 7

Conclusões

7.1 Considerações Gerais

Neste trabalho foram descritas algumas abordagens de Aprendizado de Máquina, dando ênfase a uma determinada família de sistemas que realizam Aprendizado por Indução utilizando árvores de decisão — a família TDIDT.

Podem ser encontrados pontos comuns na finalidade da construção de árvores de decisão indutivas e de alguns métodos estatísticos, como por exemplo análise discriminatória e regressão estatística. Mingers [Mingers 87b] analisa e compara a analogia existente entre árvores indutivas e regressão estatística, aplicando os dois métodos num mesmo exemplo.

Se a construção de árvores de decisão for considerada como uma técnica estatística, pode-se dizer que ela não está fundamentada em um modelo matemático: ela é não paramétrica [Niblett 87]. Essa característica a torna apropriada para problemas nos quais outras técnicas estatísticas falham. Isto é, árvores de decisão podem manusear dados com um grande número de atributos e não homogêneos — dados para os quais, em diferentes partes do espaço do problema, seus atributos estão relacionados de diferentes formas. O fato de se ter muitos atributos faz com que os modelos paramétricos não se ajustem satisfatoriamente. A não homogeneidade nos dados produz problemas mais sérios.

As principais limitações desta abordagem devem ser mencionadas. Primeiramente, como já foi discutido, o método utilizado é limitado pela quantidade e qualidade dos dados utilizados. Como em qualquer outra técnica estatística, estes métodos só conseguem encontrar regras e relações que estão presentes nos dados. Um conjunto pobre de dados gera resultados também pobres.

Em segundo lugar, a forma de representação do conhecimento é muito limitada. Não podem ser construídas árvores que testam vários atributos num mesmo nó, ou que procuram relações entre os descritores. Estas extensões poderiam ser realizadas, porém, o espaço de busca para tais estruturas poderia ser muito amplo e dificultoso.

Além disso, árvores de decisão são mais difíceis de entender do que conjuntos de unidades individuais como regras ou frames; embora já existam métodos para converter árvores de decisão em conjuntos de regras de produção [Corlett 83] [Quinlan 87a].

Segundo Clark [Clark 90], há dois tipos de limitações referentes a alguns dos algoritmos da família TDIDT.

O primeiro tipo de limitação é devido a deficiências no mecanismo utilizado. Essas deficiências podem ser superadas refinando-se o algoritmo, como de fato é feito pelos mais recentes algoritmos dessa família.

O segundo tipo de limitação é devido às restrições impostas, não por um algoritmo em particular, mas pelo paradigma de indução de regras a partir de exemplos. Essas restrições são causadas pela limitação inerente de aprender regras utilizando uma estrutura de representação simples. Porém, ir além deste paradigma para tentar construir sistemas de aprendizado mais poderosos e flexíveis, pode não ser uma tarefa fácil. Permitir uma maior flexibilidade introduzindo uma linguagem de representação mais poderosa no sistema de aprendizado, cria um problema de busca, o qual pode ficar computacionalmente impossível de resolver. Métodos de aprendizado que obtenham resposta para esse compromisso, são um dos principais objetivos das pesquisas recentes.

7.2 Conclusões

O uso de árvores de decisão em Aprendizado de Máquina deve considerar três aspectos: a construção da árvore de decisão, a poda dessa árvore e, fundamentalmente, a transformação da árvore em um conjunto de regras de produção.

Neste trabalho foram considerados os dois primeiros aspectos. Primeiramente, foi mostrado um algoritmo geral para a construção da árvore de decisão — o da família TDIDT — e foram apresentados alguns detalhes da implementação proposta. Também foi proposto e analisado um método de poda específico, utilizado em diversos domínios. Os resultados obtidos foram discutidos em função da complexidade e precisão das árvores. Fica evidenciado que a poda reduz a complexidade da árvore, e produz ganhos significativos na sua precisão.

Em geral, a precisão da árvore, após a poda, não é sensível a função de avaliação utilizada na construção da árvore. Ela depende, quase exclusivamente, da quantidade de ruído nos dados, bem como da estratégia de poda utilizada, mas não das funções de avaliação utilizadas. Foi observado que o mecanismo de poda pode incrementar a precisão da árvore em até 25%.

Cabe destacar que, apesar da simplicidade da representação de conhecimento utilizada, estes sistemas geram árvores de decisão capazes de resolver problemas difíceis. Eles representam uma técnica que já está sendo utilizada, satisfatoriamente, para auxiliar o processo de aquisição de conhecimento.

7.3 Sugestões de Trabalhos Futuros

São diversas as linhas de pesquisa que dão continuidade e complementam este trabalho. Algumas delas são:

- Implementação e análise das árvores de decisão utilizando outras funções de avaliação.
- Análise de outros métodos de poda, verificando o comportamento desses métodos em domínios reais e artificiais, bem como sua interação com diferentes funções de avaliação.
- Transformação das árvores de decisão em regras de produção, incluindo fatores de certeza, a fim de verificar qualitativamente as regras de conhecimento geradas.
- Integração deste sistema — como uma ferramenta para auxiliar a aquisição de Conhecimento — com o ambiente proposto em [Rodrigues 90a] para Auxiliar a Construção de Núcleos de Sistemas Especialistas.
- Refinamento e validação, pelo especialista, das regras produzidas utilizando este tipo de sistemas, como forma de confirmar a qualidade das regras geradas.

Bibliografia

- [Anderson 79] Anderson, J.R. e Kline P.J., A Learning System and its Psychological Implications. *Proceedings of the Sixth International Joint Conference on A.I.*, Tokio, Japão, 1979, 16-21.
- [Anderson 83] Anderson, J.R., Acquisition of Proof Skills in Geometry. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 191-220.
- [Arariboia 89] G. Arariboia, *Inteligência Artificial: um Curso Prático*. Editora LTC, Rio de Janeiro, 1989.
- [Arbab 85] Arbab, B. e Michie, D., Generating Expert Rules from Examples in Prolog. *Technical Report TIRM-85-010*, Turing Institute, Glasgow, 1985.
- [Arity 88] Arity Corporation. *The Arity/Prolog Programming Language*. 1988.
- [Barrow 71] Barrow, H.G. e Popplestone, R.J., Relational Descriptions in Picture Processing. *Machine Intelligence 6*, B.Meltzer e D.Michie (eds.), Edinburgh University Press, 1971, 377-396.
- [Biermann 88] Biermann, A.W., Fundamental Mechanisms in Machine Learning and Inductive Inference: Part 2, *Lecture Notes in Artificial Intelligence*, 345, R.T. Nossur (ed.), Springer Verlag, Germany, 1988, 125-145.
- [Breimann 84] Breimann, L., Friedman, J.H., Olshen, R.A. e Stone C.J., *Classification and Regression Trees*. Belmont: Wadsworth, 1984.
- [Buchanan 78] Buchanan, B.G. e Feigenbaum, E.A., Dendral and Meta-Dendral: Theirs Applications Dimension. *Heuristic Programming Project Memo 78-1*, Computer Science Department, Report STAN-CS-78-649. Stanford University, Stanford, CA, 1978.
- [Buntine 88] Buntine, W. e Sterling D., Interactive Induction. *Technical Report TIRM-88-030*, Turing Institute, Glasgow, April, 1988.
- [Caldeira 89] Caldeira, S.M.A., *Tratamento de Ambigüidade de Escopo de Quantificadores em Processamento de Linguagem Natural*. Tese de Mestrado, ICMSC-USP, Novembro, 1989.

- [Carbonell 83] Carbonell, J.G., Michalski, R.S. e Mitchell, An Overview of Machine Learning. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 3-24.
- [Carbonell 83a] Carbonell, J.G., Learning by Analogy: Formulating and Generalizing Plans from Past Experience. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 137-162.
- [Carbonell 86] Carbonell, J.G., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. *Machine Learning, An Artificial Intelligence Approach II*, Michalski, Carbonell e Mitchell (ed.), Kaufmann, California, 1986, 371-392.
- [Carbonell 87] Carbonell, J.G. e Langley P., Machine Learning. *Encyclopedia of Artificial Intelligence*, S.C. Shapiro (ed.), John Wiley & Sons, U.S.A., 1987, 464-488.
- [Carvalho Gomes 88] Carvalho Gomes, F.A., Mongiovi, G. e Menezes Silva, H., Um Método de Extração de Conhecimento a Partir de Exemplos. *Anais do V SBIA, Motta Pires (ed.), U. Federal de Rio Grande do Norte*, Natal, RN, Novembro de 1988, 322-331.
- [Castiñeira 90a] Castiñeira, M.I., Monard, M.C., Aprendizado de Máquina: Um Algoritmo Geral para a Construção de Árvores de Decisão. *Relatório Técnico do ICMSC - USP*. Em preparação.
- [Castiñeira 90b] Castiñeira, M.I., Monard, M.C., Análise de um Método de Poda para Árvores Indutivas de Decisão. *Anais do VII Simposio Brasileiro de Inteligência Artificial*. Trabalho aceito.
- [Cestnik 87] Cestnik B., Kononenko I. e Bratko I., Assistant 86: A Knowledge-Elicitation Tool for Sophisticated Users. *Progress in Machine Learning*, I. Bratko & N. Lavrac (ed.), Sigma Press, 1987, 31-45.
- [Charniak 86] Charniak, E. e Mc Dermott, D., *Introduction to Artificial Intelligence*. Addison Wesley, USA, 1986.
- [Clark 87] Clark, P. e Nibblet, T., Induction in Noisy Domains. *Progress in Machine Learning*, I. Bratko & N. Lavrac (ed.), Sigma Press, 1987, 11-30.
- [Clark 90] Clark, P., Machine Learning: Techniques and Recent Developments. *Technical Report TIRM-90-41, Turing Institute*, 1990.
- [Corlett 83] Corlett, R. Explaining induced decision trees. *Proc. of the Third Technical Conference of the BCS Expert Systems Group*. London, British Computer Society, 1983.

- [Davis 82] Davis, R., Teiresias: Applications of Meta-Level Knowledge. *Knowledge-Based Systems in Artificial Intelligence*, R. Davis e D.B. Lenat (ed.), McGraw-Hill, New York, 1982, 227-490.
- [Diettrich 83] Diettrich, T.G. e Michalski, R.S., A Comparative Review of Selected Methods for Learning From Examples. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 41-82.
- [Fayyad 89] Fayyad, U.M., Laird, J.E. e Irani, K.B., Conference Report: The Fifth International Conference on Machine Learning. *AI Magazine*, Summer 1989, 79-84.
- [Ford 87] Ford, Nigel, *How Machines Think*, J Wiley & Sons, Great Britain, 1987.
- [Gams 87] Gams, M. e Lavrac, N., Review of Five Empirical Learning Systems within a Proposed Schemata. *Progress in Machine Learning*, I. Bratko & N. Lavrac (ed.), Sigma Press, 1987, 46-66.
- [Hart 85a] Hart, A., Machine Induction: Practical Issues and Advice, *Proceedings of the First International ES Conference*, London, 1985, 71-78.
- [Hart 85b] Hart, A., Experience in the Use of an Inductive System in Knowledge Engineering. In *Research and Development in ES*. Bramer, M.A. (ed.) Cambridge University Press, 1985, 117-125.
- [Hass 83] Hass, N. e Hendrix, G.G., Learning by Being Told: Acquiring Knowledge for Information Management. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 405-427.
- [Hinton 89] Hinton G. E., Connectionist Learning Procedures. *Artificial Intelligence* 40, 1989, 185-234.
- [Hunt 66] Hunt E.B., Marin, J. e Stone, P.J., *Experiments in Induction*. Academic Press, New York, 1966.
- [Jackson 86] Jackson, P., *Introduction to Expert Systems*. Addison-Wesley, Great Britain, 1986.
- [Kendall 76] Kendall, Sir M.; Stuart, A. *The Advanced Theory of Statistics*. Vol.3, Charles Griffin & Company Ltd., 1976.
- [Kodratoff 87] Kodratoff, Y e Manago, M., Generalization and Noise. *International Journal of Man-Machine Studies*, 27. 1987, 181-204.
- [Kononenko 84] Kononenko, I., Bratko, I. e Roskar, E., Experiments in Automatic Learning of Medical Diagnostic Rules, *Technical Report: Josef Stefan Institute*, Ljubljana Yugoslavia, 1984.

- [Lenat 76] Lenat, D.B., *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. Ph.D Dissertation, Yale University, October 1976.
- [Lenat 82] Lenat, D.B., *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. *Knowledge-based Systems in AI*, R. Davis & D.B. Lenat (ed.), McGraw-Hill, New York, 1982, 3-228.
- [Lenat 83] Lenat, D.B., EURISKO: A Program That Learn New Heuristics and Domain Concepts. *The Nature of Heuristics III: Program Design and Results*. *Artificial Intelligence* 21, Mar., 1983, 61-98.
- [Levine 89] Levine, D.S., The Third Wave in Neural Networks. *AI Expert*, December 1989, 26-30.
- [Lindsay 80] Lindsay, R.K., Buchanan, B.G., Feigenbaum E.A. e Lederberg J. *Applications of A.I. for Organic Chemistry: The Dendral Project*. Mc. Graw-Hill, New York, 1980.
- [McCarthy 58] McCarthy, J., Programs with Common Sense. *Proceedings of the Symposium on the Mechanization of Thought Processes*, National Physical Laboratory, Teddington, England, 1958.
- [McClelland 87] McClelland, J., Demons. *Encyclopedia of Artificial Intelligence*, S.C. Shapiro (ed.), John Wiley & Sons, U.S.A., 1987, 232-233.
- [McDonald 89] McDonald, C., Machine Learning: A Survey of Current Techniques. *Artificial Intelligence Review*, 3, 1989, 243-280.
- [Michalski 73] Michalski, R.S., AQVAL/1 - Computer Implementation of a Variable Valued Logic System VL1 and Examples of its Application to Pattern Recognition. *Proceedings of the First International Joint Conference on Pattern Recognition*, Washington DC, 1973, 3-17.
- [Michalski 76] Michalski, R.S., Chilausky R. e Jacobsen, B., An Application of Variable-valued Logic to Inductive Learning of Plant Disease Diagnostic Rules. *Proceedings of the 6th Annual International Symposium on Multivariable Logic*, 1976.
- [Michalski 78] Michalski, R.S. e Larson J.B., Selection of Most Representative Training Examples and Incremental Generation of VL1 Hypothesis: The Underlying Methodology and the Description of Programs ESEL and AQ11. *Rep. 867, Dep. Computer Sci., Univ. Illinois, Urbana*, 1978.
- [Michalski 80a] Michalski, R.S., Pattern Recognition as Rule Guided Inductive Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, 4, July 1980, 349-361.

- [Michalski 80b] Michalski, R.S. e Chilausky, R.L., Learning by Being Told and Learning from Examples: an Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean disease Diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 1980, 125-161.
- [Michalski 83a] Michalski, R.S., A Theory and Metodology of Inductive Learning. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 83-133.
- [Michalski 83b] Michalski, R.S. e Stepp, R.E., Learning from Observation: Conceptual Clustering. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 331-364.
- [Michalski 87] Michalski, R.S., Concept Learning. *Encyclopedia of Artificial Intelligence*, S.C. Shapiro (ed.), John Wiley & Sons, U.S.A., 1987, 185-194.
- [Michie 68] Michie, D. e Chambers, R.A., BOXES as a Model for Pattern Formation. *Towards a theoretical biology 1*, Edinburgh: Edinburgh University Press, 1968, 206-215.
- [Michie 69] Michie, D. e Chambers, R.A., Man-Machine Cooperation on a Learning Task. *Computer Graphics Techniques and Applications*, R. Parslow, R. Prowse e R. Elliot Green (eds.), London: Plenum, 1969, 179-186.
- [Michie 87] Michie, D. Statistical classifiers compared with decision tree classifiers as applied to credit scoring. *MI-12: Machine Intelligence 12, Machine Analysis and Synthesis of Knowledge, Tallinn, Estonia, USSR, 1987*. Hayes, J., Michie, D. & Tyugu E. (ed.) Oxford: Oxford University Press.
- [Mingers 87a] Mingers, J., Expert Systems: Rule Induction with Statistical Data. *J. Operational Research Soc.*, Vol. 38, No. 1. 1987, 39-47.
- [Mingers 87b] Mingers, J., Rule Induction with Statistical Data — A Comparison with Multiple Regression. *J. Operational Research Soc.*, Vol. 38, No. 4. 1987, 347-351.
- [Mingers 89a] Mingers, J., An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning* 3, 1989, 319-342.
- [Mingers 89b] Mingers, J., An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning* 4, 1989, 226-243.
- [Minsky 72] Minsky M.S. e Papert, S., *Perceptrons*. MIT Press, Cambridge, Mass., 1972.
- [Mitchell 78] Mitchell, M.T., *Version Spaces: an Approach to Concept Learning*. Ph.D. Dissertation, Computer Science Department Report STAN-CS-78-711, Stanford University, Stanford, CA, 1978.

- [Mitchell 83] Mitchell, M.T., Utgoff, P.E., e Banerji, R., Learning by Experimentation: Acquiring and Refining Problem-solving Heuristics. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 163-190.
- [Mitchell 86] Mitchell, M.T., Keller, R.M., e Kedar-Cabelli, S.T., Explanation-Based Generalization: A Unifying View. *Machine Learning 1(1)*, 1986, 47-80.
- [Monard 89] Monard, M.C. e Rodrigues, S.R., Implementação Lógica de um Motor de Inferência com Raciocínio "Backward Chaining" para a Construção de Sistemas Especialistas. *Notas do ICMSC-USP, 51*, Outubro 1989.
- [Monard 90] Monard, M.C., Rodrigues, S.R. e Castiñeira, M.I., Um Ambiente para Auxiliar A Construção de Sistemas Especialistas. *Anais do 2 Congresso Iberoamericano de Inteligência Artificial*. Michoacán, México, Julho, 1990.
- [Mostow 83] Mostow, D.J., Machine Transformation of Advice into a Heuristic Search Procedure. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 367-403.
- [Mostow 83a] Mostow, D.J., A Problem Solver for Making Advice Operational. *Proc. AAAI 3*, 1983, 279-283.
- [Mowforth 86] Mowforth, P., Some Applications with Inductive Expert Systems Shells. *Technical Report, Turing Institute TIOP-86-002*, Glasgow, July, 1986.
- [Niblett 87] Niblett, T., Constructing Decision Trees in Noisy Domains. In *EWSL-87. Proc. of the Second European Working Session on Learning, May 1987, Bled, Yugoslavia*. Wilmslow: Sigma Press, 67-78.
- [Patterson 83] Patterson, A. e Niblett, T. *ACLS User Manual*. Glasgow: Intelligent Terminals Ltd, 1983.
- [Pearl 84] Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison Wesley, Canada, 1984.
- [Pereira 87] Pereira, F., The Indiscipline of Prolog Programming. *AI Expert*, June 1987.
- [Quinlan 79] Quinlan, J.R., Discovering Rules by Induction from Large Collections of Examples. *Expert Systems in the Micro Electronic Age*, D.Michie (ed.) Edinburgh University Press, Edinburgh, 1979, 168-201.
- [Quinlan 83] Quinlan, J.R., Learning Efficient Classification Procedures and their Application to Chess end Games. *Machine Learning, An Artificial Intelligence Approach*, Michalski, Carbonell e Mitchell (ed.), Tioga Publishing Company, California, 1983, 463-482.

- [Quinlan 86a] Quinlan, J.R., Induction of decision trees. *Machine Learning* 1, 1986, 81-106.
- [Quinlan 86b] Quinlan, J.R., The effect of noise on Concept Learning. *Machine Learning, An Artificial Intelligence Approach, II*, Michalski, Carbonell e Mitchell (ed.), Kaufmann, California, 1986, 149-165.
- [Quinlan 86c] Quinlan, J.R., Compton, P.J., Horn, K.A. e Lazarus, L., Inductive Knowledge Acquisition: A Case Study, *Proceedings of the 2nd Australian Conference on Applications of E.S.*, Sydney, 1986, 183-203.
- [Quinlan 87a] Quinlan, J.R., Simplifying decision trees. *Int. J. Man-Machine Studies* 27, 1987, pg 221-234.
- [Quinlan 87b] Quinlan, J.R., Generating Production Rules from Decision Trees. *Proc. 10 Int. Joint Conf. on Artificial Intelligence, August 1987, Milan, Italy*. Los Altos: Morgan Kaufmann, 1987, 304-307.
- [Quinlan 88] Quinlan, J.R., Decision Trees and Multi-Valued Attributes. *Machine Intelligence II*. Hayes, J.E., Michie, D. e Richard, J. (ed.), Oxford: Oxford University Press, 1988, 305-318.
- [Rich 83] Rich, E., *Inteligência Artificial*. Mc Graw-Hill, São Paulo, 1983.
- [Rodrigues 90a] Rodrigues, S.R., *Um Ambiente para Auxiliar a Construção de Núcleos de Sistemas Especialistas*. Dissertação de Mestrado, ICMSC - USP, São Carlos, 1990.
- [Rodrigues 90b] Rodrigues, S.R., Monard, M.C., Implementação Lógica de um Módulo Coletor de Dados para a Construção de Sistemas Especialistas. *Notas do ICMSC - USP, No. 63*, 1990.
- [Rodrigues 90c] Rodrigues, S.R., Monard, M.C., *Implementação Lógica de um Módulo de Explicação para a Construção de Sistemas Especialistas*. Em preparação.
- [Rosenblatt 58] Rosenblatt, F., The Perceptrons, a Probabilistic Model for Information Organization and Storage in the Brain. *Psychological Review* 65, 1958, 368-408.
- [Samuel 63] Samuel, A.L., Some Studies in Machine Learning Using the Game of Checkers. *Computers and Thought*, E.A. Feigenbaum & J. Feldman (eds.), Mc Graw-Hill, New York, 1963, 71-105.
- [Salmon 81] Salmon, W.S., *Lógica*. Zahar editores, Rio de Janeiro, 1981.
- [Selfridge 59] Selfridge, O.G., Pandemonium: a Paradigm for learning. *Proc. Symposium on Mechanization of Thought Processes*, Blake & Uttley (ed.), H.M. Stationery Office, London, 1959, 511-529.

- [Shapiro 82] Shapiro, A.D. e Nibblert, T., Automatic Induction of Classification Rules for a Chess Endgame. *Advances of Computer Chess 3*, M.R.B. Clarke (ed.), Oxford: Pergamon, 1982, 73-92.
- [Shapiro 83] Shapiro, A.D., *The Role of Structured Induction in Expert Systems*. Ph.D Thesis, Edinburgh: Edinburg University, 1983.
- [Shapiro 87] Shapiro, A.D., *Structured Induction in Expert Systems*. Turing Institute Press - Adison Wesley, Great Britain, 1987.
- [Schlimmer 86] Schlimmer, J.C. e Fisher, D., A Case Study of Incremental Concept Induction. *Proceedings of the Fifth Nat. Conf. on AI*. Philadelphia, PA: Morgan Kaufman, 1986, 496-501.
- [Shortliffe 76] Shortliffe, E.H., *Computer Based Medical Consultations: Mycin*. American Elsevier, New York, 1976.
- [Subramanian 86] Subramanian, S.D., CS229b: A Survey of A.I. Classnotes for Winter 84-85. *Report No STAN-CS-86-1104*, Department of Computer Science, Stanford University, Stanford, CA., 1986.
- [Tanimoto 87] Tanimoto, S., *The Elements of Artificial Intelligence*. Computer Science Press, U.S.A., 1987.
- [Turing 87] *Advance Prolog*. Turing Institute, 1987.
- [Utgoff 88] Utgoff, P.E., ID5: an Incremental ID3. *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI: Morgan Kaufmann, 1988, 107-120.
- [Utgoff 89] Utgoff, P.E., Incremental Induction of Decision Trees. *Machine Learning*, 4, 1989, 161-186.
- [Widrow 62] Widrow, B., *Generalization and Information Storage in Networks of Adeline 'Neurons'*. Yovitz, Jacobi e Goldstein (ed.), Spartans Books, Washington, DC, 1962, 435-461.
- [Winston 75] Winston, P.H., Learning Structural Descriptions from Examples. *The Psychology of Computer Vision*, Winston, P.H., (ed.), Mc Graw-Hill, New York, 1975, 157-209.
- [Winston 79a] Winston, P.H., Learning and Reasoning by Analogy. *Communications of the ACM* 23(12), 1979, 689-703.
- [Winston 79b] Winston, P.H., Learning by Creating and Justifying Transfer Frames. *AI: an MIT Perspective*, Winston & Brown (ed.), MIT Press, Cambridge, Mass, 1979.
- [Winston 81] Winston, P.H., Learning new Principles from Precedents and Exercises. *Artificial Intelligence* 19, 1981, 321-350.

- [Winston 83] Winston, P.H., Learning Physical Descriptionns from Functional Definitions, Examples, and Precedents. *Proc. AAAI* 3, 1983, 433-439.