Matheus Garcia Barbosa de Figueiredo

Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incerteza com Ênfase na Teoria de Conjuntos Aproximados

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel

Orientador
Prof. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2003

Matheus Garcia Barbosa de Figueiredo

Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incerteza com Ênfase na Teoria de Conjuntos Aproximados

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel

Aprovada em *dezembro de 2003*

Mário Luiz Rodrigues Oliveira

Prof. Joaquim Quinteiro Uchôa
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Agradeço a Deus por estar vencendo mais uma etapa de minha vida,
aos meus familiares, pelo apoio e incentivo,
ao professor MSc. Joaquim Uchôa, pela orientação nesse projeto

À minha gatinha Nane

Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incertezas com Ênfase na Teoria de Conjuntos Aproximados

Desde sua introdução, a Teoria de Conjuntos Aproximados (TCA) tem atraído, principalmente, o interesse de pesquisadores interessados em aprendizado de máquina e aquisição de conhecimento para sistemas especialistas, o que têm propiciado o surgimento de diversas extensões aos seus conceitos originais. O presente trabalho tem por objetivo apresentar um novo algoritmo, denominado *ML-VPM*, baseado em uma dessas extensões, o modelo de conjuntos aproximados de precisão variável.

Implementation and Development of Knowledge Discovery and Uncertainty Handling Techniques with Emphasis in the Rough Sets Theory

Since its introduction, the Rough Sets Theory (RST) has mainly attracted the interest of researches interested in machine learning and knowledge aquisition for expert systems, which has allowed the sprouting of a lot of extensions to its original concepts. The current work aims to present a new algorithm, denominated *ML-VPM*, based on one of these extensions, the variable precision rough set model.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Diagnóstico médico, reparo de equipamentos, configuração de computadores, planejamento financeiro... desde o início de 1970, com o surgimento do Mycin (Edward H. Shortliffe) e Dendral (Edward Feigenbaum e Joshua Lederberg), os sistemas especialistas vêm ganhado grande atenção, e seu desenvolvimento tem se tornado uma das principais áreas de atuação da Inteligência Artificial.

Sistemas especialistas correspondem a uma subclasse dos Sistemas Baseados em Conhecimento (SBCs), mas na literatura ambos os termos são empregados indistintamente, assim como nesse projeto. Em linhas gerais, SBCs são definidos como programas de computador que resolvem problemas utilizando conhecimento representado explicitamente e que, não fosse essa representação, exigiriam um especialista humano no domínio do problema para a sua solução ([Uchôa (1998)]).

Segundo [Morales & Harris], o objetivo da pesquisa de sistemas especialistas era programar em um computador o conhecimento e a experiência de um especialista. Sistemas especialistas têm sido usados na medicina, em gerência de negócios, na busca de recursos naturais, e muito mais. De acordo com Randal Davis, professor do MIT, citado em [Morales & Harris],

> Sistemas especialistas podem ser especialistas — você os consulta para obter orientações. Ou podem ser colaboradores, ou ainda assistentes de um especialista. Dentro de todo um espectro há muitos sistemas úteis que podem ser contruídos.

Sob este ponto de vista, SBCs devem ser capazes de representar, manipular e comunicar informações. Segundo [Cawsey (1994)],

> Representação do conhecimento é crucial. Um dos resultados mais claros da pesquisa em Inteligência Artificial é que, até mesmo para solucionar problemas aparentemente simples, é necessário uma grande quantidade de conhecimento. Realmente, entender uma única sentença requer um conhecimento extensivo sobre a linguagem e o contexto. Realmente entender uma cena visual requer conhecimento sobre os tipos de objetos na cena. Solucionar problemas em um domínio particular geralmente requer conhecimento dos objetos no domínio e conhecimento sobre como raciocinar nesse domínio — em ambos os casos o conhecimento precisa ser representado.

Independente do domínio de aplicação dos sistemas especialistas, torna-se evidentemente necessário um mecanismo que possibilite a esses sistemas adquirir o conhecimento necessário para sua efetiva aplicação. Essa tarefa de transmitir o conhecimento do especialista humano (ou de outras fontes) para a base de conhecimentos do sistema especialista é chamada de aquisição de conhecimento, e é tida como uma das etapas mais difíceis de seu desenvolvimento.

De acordo com [Nicoletti (1994)], os SBCs de *primeira geração* tinham suas bases de conhecimento criadas a partir de um processo de articulação direta. Nesse caso, cabia ao engenheiro do conhecimento extrair o conhecimento de um especialista e convertê-lo em uma forma que pudesse ser "endendida" pelos computadores. Esse mecanismo de aquisição de conhecimento é ilustrado em [Cawsey (1994)]:

> Para extrair conhecimento do especialista, o engenheiro do conhecimento precisa primeiro se familiarizar com o domínio do problema, lendo textos introdutórios ou conversando com o especialista. Depois disso, inicia-se uma entrevista mais sistemática com o especialista. Tipicamente, é dada uma série de exemplos ao especialista, e ele então explica seu raciocínio para solucionar o problema. O engenheiro do conhecimento, então, abstrai regras gerais dessas explicações, e posteriormente as confirma com o especialista.

Ainda segundo [Nicoletti (1994)], os SBCs de *segunda geração* já buscavam utilizar, ainda que parcialmente, métodos que realizassem a aquisição de conhecimento de forma automática, através de algoritmos de aprendizado.

Entretanto, independente da estratégia de aprendizado adotada, os SBCs devem estar preparados para modelar e tratar informações consideradas imperfeitas. Muitas vezes, o que convenciona chamar de informações imperfeitas abrange in-

formações imprecisas, inconsistentes, parcialmente ignoradas e mesmo incompletas. Como apontou [Bonissone (1991)]:

> a presença da incerteza em SBCs pode se originar de várias fontes: da confiabilidade parcial que se tem na informação, da imprecisão inerente da linguagem de representação na qual a informação é expressa, da não completeza da informação e da agregação/sumarização da informação que provêm de múltiplas fontes.

Dessa forma, problemas relacionados com a incerteza podem ocorrer em todo SBC. Durante a construção das bases de conhecimento, deve-se ter sempre em mente que o conhecimento com que se trabalha frequentemente é impreciso, incompleto, inconsistente.

Lidar com incerteza e mudança é importante em IA, porque:

- O mundo muda, às vezes de forma imprevista, como resultado de ações externas (isto é, ações que não são realizadas pelas pessoas)
- As percepções sobre o mundo mudam (mesmo que o mundo não mude). Se novas evidências são obtidas sobre alguma coisa, toda uma rede de fatos relacionados pode mudar
- As percepções do mundo podem ser incertas. Podemos não estar certos se observamos as coisas corretamente, ou podemos obter conclusões plausíveis mas não certas a partir de nossas crenças também não certas

Lidar com incerteza não é muito difícil de ser tratado de forma *ad hoc*. Pode-se ter regras que apaguem as coisas da memória quando elas mudarem, e/ou ter regras e fatos com fatores numéricos indicando sua credibilidade. O que é difícil é lidar com a incerteza de maneira formal. Lógica de predicado de primeira ordem é inadequada, uma vez que é destinada a trabalhar com a informação que é completa, consistente e monotônica (isso significa que os fatos são apenas adicionados, e não apagados do conjunto de informações que sabemos que são verdade). Não há uma forma direta de usá-la para lidar com informações incompletas, imprecisas, inconsistentes e não-monotônicas. Dessa forma, torna-se necessário um mecanismo formal, com princípios bem definidos e de preferência simples, para lidar com o problema da incerteza e da mudança ([Cawsey (1994)]).

Em geral, lidar com incerteza é um grande tópico da IA. De acordo com [Uchôa (1998)],

> Existem vários modelos formais disponíveis para o tratamento de incerteza; apesar disso, muitas vezes o tratamento da incerteza em SBCs tem sido feito através de abordagens *ad hoc*, baseadas em representações e combinações de regras que não são fundamentadas em uma teoria formal e tampouco têm o respaldo de uma semântica bem definida.

A fim de permitir o tratamento de informações incertas, vários modelos têm sido propostos, dentre os quais se destacam:

- Teoria de Conjuntos Aproximados (TCA) [Pawlak (1982)], [Pawlak (1991)], [Uchôa & Nicoletti (1997)],[Uchôa (1998)]
- Teoria de Conjuntos *Fuzzy* (TCF) [Zadeh (1965)], [Klir & Yuan (1995)]
- Teoria de Dempster-Shaffer (TDS) [Shafer (1976)], [Uchôa et ali (1997)]
- Redes Neurais Artificiais (RNAs) [McCulloch & Pitts (1943)], [Fausett (1993)]

A Teoria de Conjuntos Aproximados foi apresentada em 1982, por Pawlak, como um novo formalismo matemático para tratar os problemas de incerteza, em especial um tipo delas: a *indiscernibilidade*. Posteriormente, foi verificado que seus conceitos também podiam ser aplicados no desenvolvimento de técnicas de descoberta de conhecimento. Desde então, tem havido um crescente interesse em seu formalismo, o que tem proporcionado o surgimento de algumas extensões, bem como evidenciado algumas de suas limitações.

Este projeto trabalha sobre uma das extensões da TCA, mais especificamente sobre o modelo de conjuntos aproximados de precisão variável, apresentado em [Ziarko (1990)]. O capítulo 2 apresenta o conceito de Sistemas Baseados em Conhecimento, com enfoque para a tarefa de aquisição de conhecimento, em especial através de aprendizado de máquina. No capítulo 3 são apresentados os principais conceitos da teoria de conjuntos aproximados, conforme propostos por [Pawlak (1982)], indispensáveis para a compreensão do modelo de precisão variável. Esse modelo, por sua vez, é apresentado no capítulo 4, e o algoritmo de aprendizado desenvolvido a partir dele vem no capítulo 5. Por fim, no capítulo 6 são apresentados e discutidos resultados de testes realizados com o algoritmo em bases de dados conhecidas.

O algoritmo desenvolvido foi incorporado ao *ILROS*, um protótipo que implementa os principais conceitos e medidas da TCA, e alguns algoritmos de aprendizado, desenvolvido em estudos anteriores a esse. Maiores informações sobre o *ILROS* podem ser obtidas em [Uchôa & Nicoletti (1998b)].

Capítulo 2

Sistemas Baseados em Conhecimento

A extração automática de conhecimento para Sistemas Baseados em Conhecimento (SBCs), comumente conhecidos como Sistema Especialistas, é a razão principal pela qual é dada grande ênfase à Teoria de Conjuntos Aproximados. As seções a seguir apresentarão alguns dos principais conceitos dos SBCs, como sua estrutura e mecanismos de aquisição de conhecimento. É dado destaque à estratégia de aprendizado indutivo de máquina, por ser aquela que mais tem sido pesquisada e a que mais tem contribuído para o desenvolvimento de SBCs. A questão do tratamento de incertezas também é abordada ao final do capítulo.

2.1 Sistemas Baseados em Conhecimento

SBCs são definidos formalmente como programas de computador que resolvem problemas utilizando conhecimento representado explicitamente e que, não fosse essa representação, exigiriam um especialista humano no domínio do problema para a sua solução. Conhecimento e processo de resolução de problemas são pontos críticos e essenciais durante o desenvolvimento de um SBC. A arquitetura básica de um SBC pode ser visualizada na Figura 2.1.

Um SBC possui, então, três módulos principais, a saber:

1. **Base de Conhecimentos (BC):** contém o conhecimento específico do domínio da aplicação. É composto de fatos sobre o domínio, regras que descrevem

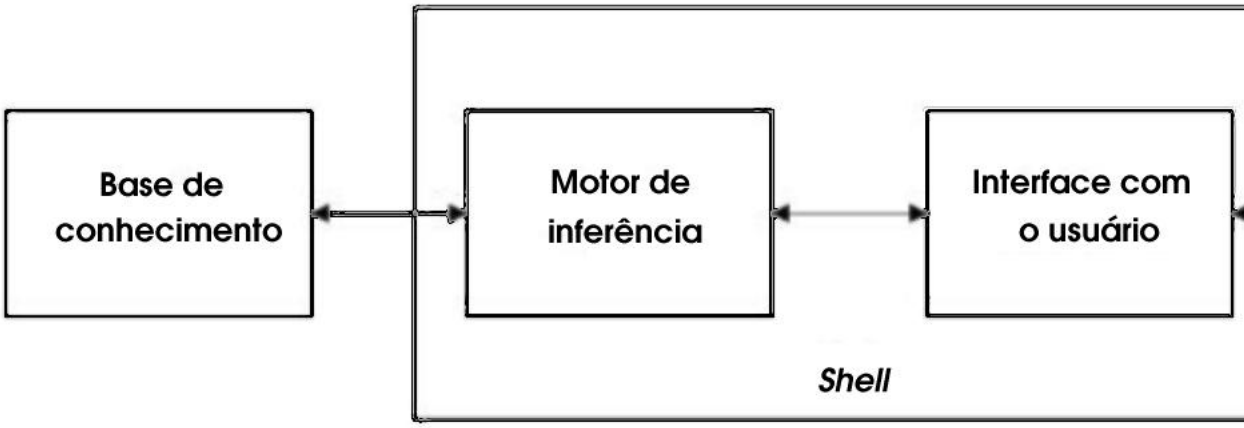


Figura 2.1: Arquitetura básica de um SBC

relações no domínio e métodos e heurísticas para resolução de problemas no domínio.

2. **Motor de Inferência (MI):** mecanismo responsável pelo processamento do conhecimento da BC, utilizando-se de alguma linha de raciocínio. Implementa as estratégias de inferência e controle do SBC. Quando o conhecimento do SBC está expresso como regras, as estratégias de controle empregadas pelo MI normalmente são encadeamento para trás *(backward chaining)* ou encadeamento para frente *(forward chaining)*. Quando o MI usa a estratégia de encadeamento para trás, uma lista de hipóteses é pesquisada procurando reunir evidências para viabilizar a conclusão da validade de alguma(s) dela(s). Esta estratégia corresponde à pergunta: É possível provar as hipóteses a partir dos dados disponíveis? Se a estratégia de controle for encadeamento para frente, o MI parte dos dados e, com base nas regras de conhecimento, deduz outras asserções procurando chegar à solução do problema. Essa estratégia corresponde à pergunta: O que é possível concluir a partir dos dados disponíveis?

3. **Interface com o Usuário (IU):** módulo responsável pela comunicação entre o usuário e o sistema. Deve fornecer, também, justificativas e explicações referentes às conclusões obtidas na BC, bem como do raciocínio utilizado.

Sistemas Especialistas (SEs) constituem uma importante subclasse dos SBCs. Um SE pode ser definido como um SBC que resolve problemas bem específicos do mundo real, problemas esses que requerem considerável habilidade, conhecimento e heurísticas para sua resolução. Na literatura, os termos Sistemas Baseados em Conhecimento e Sistemas Especialistas são, muitas vezes, utilizados quase sem distinção.

A grande maioria dos SBCs apresenta-se sob a forma de Sistemas Baseados em Regras (SBRs), que utilizam-se de regras para codificação do conhecimento. Uma regra é uma estrutura da forma $(a \Rightarrow b)$, e é lida `se` a `é verdade, então` b `também é verdade`, ou resumidamente `se` a`, então` b. O termo a é denominado antecedente da regra, e expressa as condições de execução da mesma. O termo b, por sua vez, é denominado conseqüente da regra e, caso satisfeitas as condições expressas no antecedente, torna-se um fato. Um fato é uma regra da forma $(\Rightarrow b)$, ou resumidamente (b), onde o antecedente é vazio, o que implica a verdade incondicional de b.

Observe que, em um SBR, o Motor de Inferência é um interpretador de regras. Os SBCs também podem apresentar-se sob a forma de Sistemas Baseados em Árvores de Decisão, que utilizam-se de uma árvore de decisão para codificação do conhecimento. Numa árvore de decisão, os nós não-folhas são considerados como os antecedentes da regra e sua análise permite que se percorra a árvore em busca de um nó folha que é considerado como o conseqüente da regra. Note que, em um SBC baseado em árvore de decisão, o Motor de Inferência é um algoritmo que percorre uma árvore em busca de um nó folha. Os nós são analisados, indicando o caminho que se deve seguir na árvore, sendo que os nós não-folha possuem um ou mais valores (atributos) e compõem o antecedente de uma regra (condições de uma regra), enquanto que os nós folhas que possuem também um ou mais valores irão compor o conseqüente da regra (conclusão de uma regra).

A capacidade de resolver problemas que demandam informação referente a um determinado domínio de conhecimento e explicar seu comportamento tornaram-se os aspectos principais de um SBC. Um SBC deve ser capaz de explicar seu comportamento e suas decisões ao usuário, respondendo o *porque* e *como* chegou a uma determinada solução. De uma maneira geral as perguntas *por que* referem-se a qual conhecimento respalda a conclusão; as perguntas *como*, por sua vez, referem-se aos passos de raciocínio seguidos para determinar a solução do problema. Esta característica é especialmente necessária quando o SBC lida com domínios incertos (diagnóstico médico, por exemplo); a explicação pode, de certa forma, aumen-

tar o grau de confiança que o usuário deposita no sistema, ou então, ajudá-lo a encontrar alguma falha no raciocínio do sistema.

Uma outra característica frequentemente necessária é a habilidade de lidar com incertezas e informações incompletas: a informação a respeito do problema a ser resolvido pode estar incompleta ou ser parcialmente confiável, bem como as relações no domínio do problema podem ser aproximadas. Espera-se, também, que um SBC seja flexível o suficiente para permitir, facilmente, a acomodação de novo conhecimento.

2.2 Aquisição de Conhecimento

Uma das principais atividades relacionadas ao desenvolvimento de Sistemas Baseados em Conhecimento consiste na transferência do conhecimento - informações e/ou formas de condução do raciocínio - do especialista humano (ou de qualquer outra fonte) à Base de Conhecimento do SBC. Este processo é conhecido como Aquisição de Conhecimento (AC) e é, reconhecidamente, o processo mais difícil durante o desenvolvimento de SBCs, exigindo um grande investimento em tempo e esforço.

Além da extração do conhecimento necessário, tal conhecimento deve ser traduzido para o esquema formal de representação usado pelo SBC e deve ser repetidamente refinado, até que o sistema atinja um grau de desempenho próximo ao de um especialista humano, quando da resolução do problema. Sobre esse processo, pode ser observado em [Nicoletti (1994)] que:

> ... o processo de extração do conhecimento do especialista envolve um intenso questionamento que, em certas situações, pode interferir na sua própria percepção de como elabora o raciocínio. Geralmente um especialista acha difícil detalhar descrições de seu conhecimento e de como o usa; por outro lado, muitas vezes é difícil para o engenheiro do conhecimento traduzir com exatidão aquele conhecimento, geralmente amplo e multifacetado, em uma linguagem de representação restrita. Devido a essas dificuldades, muitas vezes o conhecimento extraído pode ser inconsistente (como conseqüência de diferenças individuais entre especialistas), incompleto e impreciso.

Existe um vasto conjunto de métodos e ferramentas que facilitam a tarefa de AC. A disponibilidade dessa variedade reflete o fato da AC ser um processo multidimensional, podendo ocorrer em diferentes estágios do desenvolvimento de um

SBC, e podendo envolver vários tipos de conhecimento. Uma classificação proposta em [Boose (1989)] agrupa os métodos e técnicas para AC em:

1. **manuais -** que tipicamente consistem de entrevistas e análise de protocolos;
2. **baseadas em computador:**
   (a) ***interativas* -** que englobam, principalmente, ferramentas que entrevistam o especialista, que fazem análise textual, que extraem e analisam conhecimento de múltiplas fontes separadamente e as combinam para uso;
   (b) ***baseadas em aprendizado* -** as quais, via de regra, generalizam situações específicas em conceitos.

2.3 Aprendizado de Máquina

Como observado em [Michalski & Tecuci (1993)], aprendizado pode ser caracterizado como um processo multidimensional que, via de regra, ocorre através da aquisição de conhecimento declarativo, do desenvolvimento de habilidades motoras e cognitivas através de instrução e prática, da organização do conhecimento existente em representações mais efetivas, da descoberta de novos fatos e/ou teorias através da observação e experimentação ou, então, através da combinação e/ou composição dessas dimensões.

Um sistema inteligente deve ser capaz de formar conceitos, i.e., classes de entidades unidas por algum princípio. Tal princípio pode ser o fato das entidades terem um uso ou objetivo comum ou, simplesmente, terem características perceptuais similares. Para a utilização do conceito, o sistema deve também desenvolver métodos eficientes de reconhecimento de pertinência de uma dada entidade ao conceito. O estudo e a modelagem de processos pelos quais o sistema adquire, refina e diferencia conceitos são objetivos da área de pesquisa chamada de *Aprendizado de Conceito*, uma subárea de AM.

Em Aprendizado de Conceito, o termo *conceito* usualmente indica uma classe de equivalência de entidades, que pode ser descrita via um conjunto relativamente limitado de expressões e que deve ser suficiente para a diferenciação entre conceitos. Entidades individuais na classe são chamadas de *instâncias* do conceito. O fato do conceito ser definido como uma classe de equivalência torna cada instância de uma classe igualmente representativa do conceito em questão; tal definição estabelece, também, os limites da descrição daqueles conceitos - uma entidade

satisfaz ou não satisfaz o conceito. Mais detalhes sobre aprendizado de conceito podem ser encontrados em [Morales & Harris] e [Langley (1996)].

Observe que, dada a complexidade existente no processo de aprendizado, não há em relação a ele uma definição única e tampouco um entendimento suficiente. Podem ser encontradas na literatura definições que se diferenciam entre si pela ênfase dada a diferentes aspectos do processo de aprendizado.

Uma abordagem mais limitada de AM, geralmente adotada por pessoas que trabalham especificamente com SBCs, é a de que aprendizado é a aquisição de conhecimento explícito. Muitos SBCs representam conhecimento como um conjunto de regras que necessitam ser adquiridas, organizadas e estendidas. Isto enfatiza a importância de tornar explícito o conhecimento adquirido, de maneira que ele possa facilmente ser verificado, modificado e explicado.

Na literatura há diversas taxionomias de sistemas de AM. Algumas das várias razões que dão origem a esta diversidade de classificações são: diferentes estratégias de aprendizado adotadas, existência de diferentes representações do conhecimento, domínios de aplicação variados, etc. Uma possível taxionomia de métodos de Aprendizado de Máquina classifica os paradigmas em:

Aprendizado Simbólico: aquisição de conceitos expressos em símbolos, através de conjuntos de exemplos;

Aprendizado Baseado em Instância: métodos que simplesmente armazenam as instâncias de treinamento;

Aprendizado Baseado em Algoritmos Genéticos: que inclui ambos: algoritmos genéticos, que induzem hipóteses descritas usando cadeias de bits, e programação genética, que induzem hipóteses descritas como programas;

Aprendizado Conexionista: buscam modelar o processo de funcionamento dos neurônios e/ou áreas do cérebro humano;

Aprendizado Analítico: aprendizado baseado em explicações e certas formas de métodos de aprendizado analógicos e baseados em casos.

Note que, dentre todos esses modelos, apenas o Aprendizado Analítico é de natureza dedutiva; todos os demais são indutivos.

2.3.1 Aprendizado Indutivo de Máquina

Entre os vários modelos existentes para aprendizado apontados anteriormente, o aprendizado simbólico conhecido como aprendizado indutivo baseado em exemp-

los é o que mais tem sido pesquisado e o que mais tem contribuído efetivamente para a implementação de sistemas de aprendizado de máquina. A partir de um conjunto de exemplos, expressões para tarefas classificatórias podem ser aprendidas (induzidas) como, por exemplo, diagnóstico de doenças, previsão meteorológica, predição do comportamento de novos compostos químicos, predição de propriedades mecânicas de metais com base em algumas de suas propriedades químicas, etc. A Figura 2.2, extraída de [Shaw & Gentry (1990)], ilustra esse processo.

Figura 2.2: Esquema geral de aprendizado indutivo de regras

No aprendizado indutivo baseado em exemplos, também referenciado como aprendizado indutivo, o conjunto de exemplos, também denominado de *conjunto de treinamento*, é fornecido ao sistema por um instrutor ou pelo ambiente (base de dados, sensores, etc.). Esse conjunto de treinamento é geralmente composto de *exemplos positivos* (exemplos do conceito) e *exemplos negativos* (contra-exemplos do conceito). A indução do conceito corresponde a uma busca no espaço de hipóteses, de forma a encontrar aquelas que *melhor* classificam os exemplos. Nesse contexto, *melhor* pode ser definido em termos de precisão e compreensibilidade.

De uma maneira geral, um sistema que aprende a partir de exemplos recebe como dados informações na forma de situações específicas, cada uma delas devidamente classificadas (geralmente por especialista humano no domínio), caracterizando o que se convenciona chamar de *aprendizado supervisionado*, e produz, como resultado, uma(s) hipótese(s) que generaliza(m) aquelas situações inicialmente fornecidas. Maiores informações sobre o aprendizado indutivo de máquina podem ser obtidas em [Uchôa (1998)].

2.3.2 Tratamento de Incerteza em Aprendizado de Máquina

Um dos principais problemas em Aprendizado de Máquina é o de representação de incerteza. É fato que todo SBC possui problemas relativos à incerteza. Esta pode se manifestar de diversas formas: imprecisão, incompleteza, inconsistência, etc. Torna-se, portanto, desejável que a estratégia de AM implementada, bem como o próprio SBC, possua mecanismos de tratamento da incerteza. Esses mecanismos freqüentemente surgem sob a forma de medidas de incerteza.

A Teoria de Conjuntos Aproximados, formalismo tratado neste trabalho, possui mecanismos para expressão de, entre outros, um tipo fundamental de incerteza: a indiscernibilidade. A indiscernibilidade surge quando não é possível distinguir objetos de um mesmo conjunto; representa a situação em que esses objetos parecem todos ser um único objeto.

Representação da indiscernibilidade, bem como técnicas para a classificação de objetos, usando a TCA, são as bases que sustentam o presente trabalho e serão abordadas em detalhes a seguir.

Capítulo 3

Elementos da Teoria de Conjuntos Aproximados

Uma característica marcante da TCA é seu forte formalismo matemático. Nesse capítulo são apresentados os principais conceitos dessa teoria, que serviram de base para o desenvolvimento de diversos algoritmos de aprendizado. Há vários outros conceitos não apresentados aqui, optou-se por apresentar apenas os essenciais para a compreensão do modelo de precisão variável, a ser apresentado no capítulo 4.

3.1 Espaço aproximado

Um *espaço aproximado* é um par ordenado $A = (U, R)$, onde:

- U é um conjunto não vazio, denominado *conjunto universo*;
- R é uma relação de equivalência definida em U, denominada *relação de indiscernibilidade*.

  A relação R particiona o universo em classes de equivalência, que são chamadas de *conjuntos elementares*. Dois elementos quaisquer que pertençam a um mesmo conjunto elementar são ditos *indiscerníveis* em R. Dado um elemento $x \in U$, $[x]_R$ representa o conjunto elementar que possui o elemento x com base na relação R.

3.2 Aproximações de um conjunto

Dado um espaço aproximado $A = (U, R)$, o interesse é definir um determinado conjunto $X \subseteq U$ por meio dos conjuntos elementares de A. Se é possível definir X precisamente pela união de conjuntos elementares, então ele é dito *definível*, caso contrário, X é dito aproximado, e pode ser representado por meio das aproximações superior e inferior:

- a *aproximação inferior* de X em A, $A_{A-inf}(X)$, corresponde a união de todos os conjuntos elementares que estão contidos em X:

$$A_{A-inf}(X) = \{x \in U \mid [x]_R \subseteq X\};$$

- a *aproximação superior* de X em A, $A_{A-sup}(X)$, corresponde a união de todos os conjuntos que possuem intersecção não vazia com X:

$$A_{A-sup}(X) = \{x \in U \mid [x]_R \cap X \neq \varnothing\}.$$

Quando não há risco de confusão, costuma-se omitir a referência ao espaço aproximado na representação das aproximações. Assim, será utilizado $A_{inf}(X)$ em substituição a $A_{A-inf}(X)$ e $A_{sup}(X)$ em substituição a $A_{A-sup}(X)$.

3.3 Regiões do espaço aproximado

As aproximações superior e inferior dividem o espaço aproximado em três regiões, a saber:

1. A *região positiva* de X em A é formada pela união dos conjuntos elementares de A que estão inteiramente contidos em X. Corresponde, então, à aproximação inferior. Se um elemento pertence à região positiva de X, pode-se afirmar com certeza que ele pertence a X.

$$pos_A(X) = A_{inf}(X)$$

2. A *região negativa* de X em A corresponde ao complemento da $A_{sup}(X)$ em relação a U, ou seja, é formada pelos conjuntos elementares de A que não estão contidos na aproximação superior de X. Se um elemento pertence à região negativa, pode-se afirmar com certeza que ele não pertence a X

$$neg_A(X) = U - A_{sup}(X)$$

3. A *região duvidosa* de X em A é formada pelos elementos que pertencem a aproximação superior mas não pertencem à aproximação inferior. Se um elemento pertence à região duvidosa, não é possível determinar se ele pertence ou não a X.

$$duv_A(X) = A_{sup}(X) - A_{inf}(X)$$

Exemplo 1 *Seja U um conjunto universo e R uma relação de equivalência em U, definindo o espaço aproximado $A = (U, R)$. Seja também X, como ilustra a Figura 3.1. A aproximação inferior e a aproximação superior de X em $A = (U, R)$ são mostradas na Figura 3.2(a) e 3.2(b). A Figura 3.3, por sua vez, apresenta as regiões de X.*

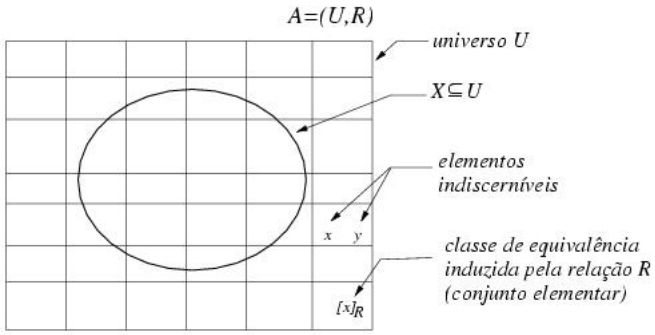
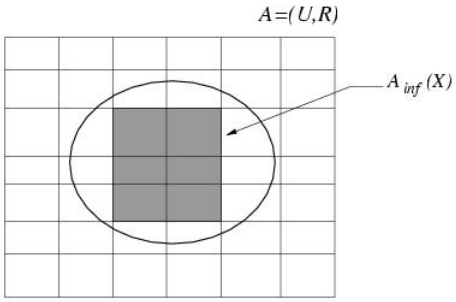


Figura 3.1: Conjunto X no espaço aproximado $A = (U, R)$.

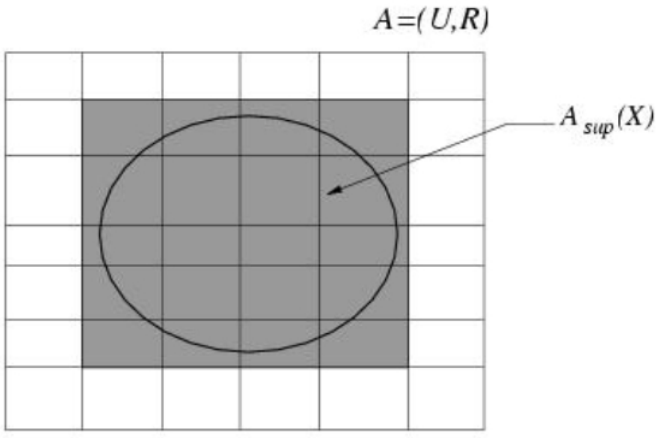
Seja $A = (U, R)$ um espaço aproximado e seja $X \subseteq U$. O conjunto X pode ou não ter suas fronteiras claramente definidas em função das descrições dos conjuntos elementares de A. Isso leva ao conceito de conjuntos aproximados: um *conjunto aproximado* em A é a família de todos os subconjuntos de U que possuem a mesma aproximação inferior e a mesma aproximação superior em A. Ou seja, possuem a mesma região positiva, negativa e duvidosa.

3.4 Acuracidade de um aproximação

Seja um espaço aproximado $A = (U, R)$. Com a finalidade de verificar o quão bem o conjunto $X \subseteq U$ pode ser representado por A, são definidas as seguintes medidas:



(a) Aproximação Inferior de X.



(b) Aproximação Superior de X.

Figura 3.2: Aproximações de X em A.

- *medida interna* de X em A, $\omega_{A-inf}(X) = |A_{A-inf}(X)|$;
- *medida externa* de X em A, $\omega_{A-sup}(X) = |A_{A-sup}(X)|$;
- *qualidade da aproximação inferior* de X em A, $\gamma_{A-inf}(X) = \frac{\omega_{A-inf}(X)}{|U|}$;
- *qualidade da aproximação superior* de X em A, $\gamma_{A-sup}(X) = \frac{\omega_{A-sup}(X)}{|U|}$.

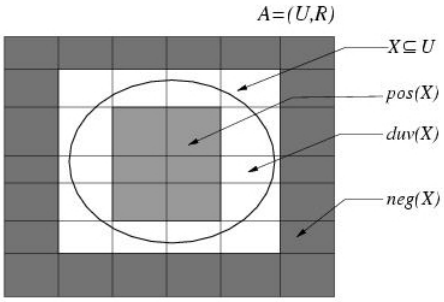


Figura 3.3: Regiões de X em A.

A *acuracidade* de X é definida pela relação entre a qualidade da aproximação inferior de X e a qualidade da aproximação superior de X, em símbolos:

$$\omega_A(X) = \frac{\omega A_{inf^A}(X)}{\omega A_{sup^A}(X)} = \frac{|A_{inf}(X)|}{|A_{sup}(X)|}.$$

Quando o espaço aproximado for conhecido, costuma-se representar essas medidas como ω_{inf}, ω_{sup}, γ_{inf}, γ_{sup} e ω em substituição a ω_{A-inf}, ω_{A-sup}, γ_{A-inf}, γ_{A-sup} e ω_A.

3.5 Índice discriminante de atributos

O índice discriminante é uma medida que informa o grau de certeza com que se pode afirmar que um elemento $x \in U$ pertence ou não a um conjunto $X \subset U$, com base nas regiões do espaço aproximado $A = (U, R)$.

Uma vez que sabemos que elementos das regiões positiva certamente pertencem a X e que elementos da região negativa certamente não pertencem a X, o índice discriminante vai depender da fração que a região duvidosa ocupa no universo U, sendo igual ao percentual de elementos que pertencem às regiões positiva e negativa com relação ao universo U. Em símbolos:

$$\alpha_R(X) = \frac{|U| - |duv_A(X)|}{|U|},$$

Se X é definível, a região duvidosa é vazia, e o índice discriminante é 1.

3.6 Função de Pertinência Aproximada

Em [Pawlak (1994)] é proposta uma função de pertinência aproximada, que permite calcular a pertinência de um elemento do universo U a um conjunto $X \subseteq U$. Dado um espaço aproximado $A = (U, R), X \subseteq U$ e $x \in U$, a pertinência de x a X no espaço A é dada por:

$$\mu_X^A(x) = \frac{|[x]_R \cap X|}{|[x]_R|},$$

onde $[X]_R$ denota a classe de equivalência induzida pela relação de equivalência R, que contém o elemento x (e todos os seus equivalentes, por R). Quando A é conhecido e não há risco de confusão, escreve-se μ_X em substituição a μ_X^A.

3.7 Comentários Finais

Este capitulo apresentou os conceito principais da TCA. Foi com base nesses conceitos que foram desenvolvido os algoritmos *RS1+*, apresentado em [Uchôa (1998)], e *ML-VPM*, que será apresentado no capítulo 5.

Outros conceitos da TCA podem ser verificados em [Pawlak (1991)] e [Uchôa (1998)].

Capítulo 4

O modelo probabilístico de precisão variável

Uma das principais limitações do modelo original da TCA é o fato desse desconsiderar as informações não-determinísticas (probabilísticas). O presente capítulo aborda um modelo extendido da TCA, apresentado em [Ziarko (1990)], proposto para tratar esse tipo de informação. Esse modelo foi denominado *modelo de conjuntos aproximados de precisão variável*, ou apenas *modelo de precisão variável*, e foi com base nele que foi possível desenvolver o algoritmo *ML-VPM*, que será apresentado no capítulo 5.

4.1 Motivação

Proposta em [Pawlak (1982)], a Teoria de Conjuntos Aproximados surgiu como um formalismo matemático para representação de conhecimento incerto, sendo posteriormente utilizada para desenvolvimento de algoritmos de aprendizado de máquina. Desde então tem havido um crescente interesse em seu formalismo, o que tem propiciado o surgimento de diversas aplicações nas mais variadas áreas.

No entanto, ao mesmo tempo em que se notou um crescente interesse na teoria, acabaram se tornando evidentes algumas de suas limitações, em especial a dificuldade de se tratar informações incertas, as informações não-determinísticas ou probabilísticas. Segundo [Ziarko (1990)], o modelo original da TCA desconsidera totalmente esse tipo de informação, considerando apenas as informações determinísticas, que geram regras com 100% de precisão (credibilidade).

Conforme apontado por esse autor,

> Frequentemente, a informação disponível permite-nos apenas uma classificação parcial. A Teoria de Conjuntos Aproximados pode ser usada para modelar esse tipo de classificação, mas a classificação deve estar totalmente correta ou precisa. A classificação dentro de um grau controlado de incerteza, ou com um pequeno erro na classificação, está fora do alcance dessa abordagem.

Ainda segundo o autor,

> Outra limitação do modelo original de conjuntos aproximados reside na suposição de que todo o universo de objetos em consideração é conhecido, e que todas as conclusões derivadas à partir desse modelo são aplicáveis apenas à esse conjunto de objetos. Na prática, no entanto, há uma necessidade evidente de se generalizar conclusões obtidas à patir de um pequeno conjunto de exemplos para uma população maior.

Tratando apenas informações determinísticas, as regras geradas ficam totalmente dependentes do conjunto de treinamento utilizado, e tendem a diminuir sempre que é adicionado um novo elemento ao conjunto. Isso porque "o conhecimento com que se trabalha raramente está completo ou é exato" [Uchôa (1998)]. A adição de novos exemplos aumentam a probabilidade de um ruído nos dados, que pode fazer com que elementos iguais pareçam diferentes, e assim regras determinísticas deixem de ser geradas.

Dessa forma, hipóteses derivadas a partir de um conjunto de exemplos não devem se basear apenas em classificações livres de erro. Classificações parcialmente corretas também devem ser levadas em consideração. Qualquer regra de classificação incorreta fornece uma informação importante se a maioria dos elementos aos quais essa regra se aplica puderem ser corretamente classificados.

A principal vantagem desse modelo é a habilidade de reconhecer a presença de dependências de dados em situações onde esses dados seriam considerados independentes no modelo original. Tais situações ocorrem quando as dependências de dados são não-funcionais, ou não-determinísticas. Esse tipo de informação não-determinística, parcialmente correta e imprecisa pode ser capturada e analisado a partir de técnicas que extendam os conceitos da TCA, as quais serão apresentadas a seguir.

4.2 Características

4.2.1 Relação de inclusão maioritária

A idéia básica por trás do modelo de precisão variável é a generalização do conceito de inclusão entre conjuntos. A definição padrão não representa qualquer tipo de inclusão parcial, mesmo que seja uma inclusão quase completa. A fim de definir a relação de inclusão maioritária, torna-se necessário definir a medida de **erro de classificação relativo** de um conjunto X em um conjunto Y. Na definição abaixo, $card(X)$ representa o número de elementos (cardinalidade) de X.

$$\begin{array}{rcll} e(X,Y) & = & 1 - (card(Y \cap X)/card(X)) & \text{se } card(X) > 0 \\ & = & 0 & \text{se } card(X) = 0 \end{array}$$

Ou seja, se os elementos de X forem classificados como elementos de Y, em $e(X,Y) * 100\%$ dos casos seriam cometidos erros de classificação. A medida $e(X,Y)$ é chamada de **erro de classificação relativo**, enquanto a medida $e(X,Y) * card(X)$ é o **erro de classificação absoluto**.

Dessa forma, podemos redefinir o conceito de inclusão de conjuntos para a chamada **inclusão maioritária**, onde

$$Y \supseteq X \qquad\qquad se\ e(X,Y) \leq \beta,\ onde\ 0 \leq \beta\ <\ 0.5$$

O valor de β é chamado **erro de classificação máximo**, e corresponde ao maior valor do erro a ser considerado ao classificarmos os elementos de um conjunto X como elementos de um conjunto Y. Seu valor não deve ser maior que $0{,}5$, a fim de permitir que a maioria dos elementos seja classificada corretamente. Conjuntos incluídos em um conjunto Y com um erro de classificação máximo igual a β são ditos β-incluídos em Y.

Assim, seja $X_1 = \{x_1, x_2, x_3, x_4\},\ X_2 = \{x_1, x_2, x_5\},\ X_3 = \{x_1, x_6, x_7\}$ e $Y = \{x_1, x_2, x_3, x_8\}$. De acordo com a relação de inclusão β-maioritária, podemos dizer que

$$Y \overset{\beta}{\supseteq} X_1 \qquad\qquad Y \overset{\beta}{\supseteq} X_2 \qquad\qquad Y \overset{\beta}{\not\supseteq} X_3$$

4.2.2 Aproximações

Assim como no modelo original, um espaço aproximado é definido como o par ordenado $A = (U, R)$, que consiste em um universo e uma relação de equivalência definida sobre esse universo, chamada de relação de indiscernibilidade. Essa relação particiona o universo em classes de equivalência, chamadas de conjuntos elementares.

Da mesma forma, o interesse em um espaço aproximado é classificar, por meio dos conjuntos elementares de A, um conjunto $X \subseteq U$. Se o conjunto X puder ser classificado por meio dos conjuntos elementares, é dito definível, caso contrário, é um conjunto aproximado, e pode ser representado por meio das aproximações superior e inferior.

Os conceitos de aproximação superior e inferior são extendidos pelo modelo de precisão variável para incluir um certo grau de erro, fazedo uso da relação de inclusão maioritária definida anteriormente. A inclusão desse fator corresponde a uma extensão natural a essas medidas, de modo que a maioria das propriedades que se aplicam a elas continuam válidas. Dado um espaço aproximado $A = (U, \widetilde{R})$ e um grau de erro de classificação máximo $0 \leq \beta < 0,5$., pode-se definir as aproximações da seguinte maneira:

- **Aproximação inferior probabilística**: é a união de todos os conjuntos elementares de A cujo erro de classificação em X é menor ou igual a β, ou seja, é formada pela união de todos os conjuntos elementares de A que estão β-incluídos em X.

$$\begin{aligned} A^{\beta}_{inf}(X) &= \{E \in class(R) \mid X \overset{\beta}{\supseteq} E\} \\ &= \{E \in class(R) \mid e(X,Y) \leq \beta\} \end{aligned}$$

- **Aproximação superior probabilística**: é a união de todos os conjuntos elementares de A cujo erro de classificação em X é maior que B, ou seja, é formada por todos os conjuntos elementares que estão β-incluídos no complemento de X.

$$A^{\beta}_{sup}(X) = \{E \in class(R) \mid e(X,Y) < 1 - \beta\}$$

A aproximação inferior de um conjunto X pode ser interpretada como a coleção de todos os elementos do universo que podem ser classificados em X com um erro

de classificação não maior que β. De forma similar, a região β-negativa de X é a coleção de elementos do universo que podem ser classificados no complemento de X com um erro de classificação não maior que β.

De forma similar ao modelo original, as aproximações particionam o universo em três regiões:

- **Região positiva**: corresponde a aproximação inferior probabilística, compreende os elementos que podemos classificar como pertencentes a X com um grau de erro não maior que β, ou seja, são os elementos com maior probabilidade de pertencerem a X.

- **Região negativa**: corresponde ao complemento da aproximação superior probabilística, compreende os elementos que podemos classificar como não-pertencentes a X (ou pertencentes ao complemento de X) com um erro de classificação não maior que β, ou seja, são os elementos com menor probabilidade de pertencerem a X.

- **Região duvidosa**: corresponde aos elementos que não podemos classificar como pertencentes a X ou ao seu complemento com um erro de classificação não maior que X. Tais elementos podem tanto estar em X como fora dele, não sendo possível classificá-los.

A partir da redefinição desses conceitos, foi possível desenvolver um novo algoritmo de aprendizado de máquina, que extrai informações desconsideradas por outros que se baseiam no modelo original. Esse novo algoritmo foi denominado *ML-VPM*, e será apresentado no capítulo 5.

Assim como as aproximações, outros conceitos da TCA também podem ser extendidos para conter esse grau de erro, como acuracidade, pertinência e índice discriminante. Optamos por não apresentá-los aqui, pois seguem a mesma idéia que foi empregada nas aproximações, e não são empregados diretamente no algoritmo. Maiores informações sobre esses conceitos podem ser obtidas em [Ziarko (1990)].

Capítulo 5

O algoritmo de aprendizado ML-VPM

Conforme já dito anteriormente, o fato de não tratar informações probabilísticas foi a motivação principal para o surgimento de extensões à TCA, como os modelos de precisão variável, apresentados no capítulo 4. O presente capítulo apresenta o algoritmo *ML-VPM*, proposto por nós e desenvolvido com base nesse modelo. É apresentado também um exemplo de execução passo-a-passo para um determinado SRC.

5.1 Características

Em geral, algoritmos de aprendizado indutivo representam conhecimento sob a forma de regras de decisão, que são estruturas que possuem um antecedente (conjunto de condições) e um conseqüente (decisão). Foi dessa forma que o algoritmo *ML-VPM* foi projetado, para extrair regras de decisão à partir de um conjunto de exemplos, sendo então mais um método de aprendizado indutivo.

Conforme apresentado em [Ziarko (1990)], são características desejáveis das regras extraídas a partir de um algoritmo de aprendizado:

- discriminem entre diferentes categorias de decisão, total ou parcialmente
- capturem os fatores essenciais que afetem o resultado da classificação e não levem em conta fatores irrelevantes

- sejam não-redundantes em termos de minimizar o número exigido de regras e suas condições
- sejam gerais, recebam forte suporte nos dados disponíveis (conjunto de treinamento)
- exibam uma baixa taxa de erro ao serem generalizadas

Em geral, são essas as metas a serem alcançadas pelos vários métodos de aprendizado desenvolvidos até então, cada um trabalhando de uma forma diferente. A característica principal desse algoritmo é o fato de extrair regras probabilísticas, ao contrário de outros que buscam apenas identificar padrões determinísticos nos dados. Dessa forma, ele acaba gerando um número maior de regras que os demais, mas permite que mais elementos possam ser classificados com as regras geradas.

5.2 Algoritmo *ML-VPM*

Esta seção tem por objetivo apresentar o algoritmo *ML-VPM*. Para facilitar seu entendimento, o algoritmo está descrito em alto-nível, deixando de lado detalhes de implementação para destacar apenas os passos a serem seguidos na extração de conhecimento.

Dado um universo U, um conjunto de atributos de condição C, um atributo de decisão δ e um grau de erro de classificação máximo β, o algoritmo poderia ser resumido nos seguintes passos:

1. Calcular $class(\delta) = \{X_1 \dots X_J\}$, a família de conjuntos elementares do espaço aproximado $A = (U, \delta)$
2. Fazer $j = 1$, $J =$número de conjuntos elementares de $class(\delta)$
3. Fazer $C' = C$, $U' = U$, $X' = X_j, Cant' = \emptyset$
4. Ordenar os atributos de C' de acordo com o índice discriminante
5. Fazer $i = 1$; $I =$número de elementos de C'
6. Fazer $Unovo = U'$, $Xnovo = X'$, $Cnovo = Cant \cup \{a_i\}$
7. Calcular $A = (Unovo, Cnovo)$

8. Se o número de conjuntos elementares de A for igual a um 1, ir para o passo 19

9. Calcular $AinfX$, a aproximação inferior de $Xnovo$ em $A = (Unovo, Cnovo)$. Se $AinfX = \emptyset$, para o passo 12

10. Para cada conjunto elementar em $AinfX$, gerar uma regra determinística

11. Fazer $Xnovo = Xnovo - AinfX$

12. Fazer $U = duv(Xnovo)$. Se $Unovo = \emptyset$, ir para o passo 19

13. Calcular $AprobInfX$, a aproximacao inferior probabilistica de $Xnovo$ em $A(Unovo, Cnovo)$. Se for vazia, ir para o passo 15

14. Para cada conjunto elementar em $AprobInfX$, gerar uma regra probabilística

15. Calcular $Cproximo$, o conjunto de atributos para a chamada recursiva, contendo todos os atributos de C' que aparecem depois do atributo a_i selecionado no passo 6. Se $Cproximo = \emptyset$, ir para o passo 19

16. Fazer $k = 1$; K =número de conjuntos elementares de $A = (Unovo, Cnovo)$

17. Voltar ao passo 4, com $Cant' = Cnovo$, $C' = Cproximo$, U' =conjunto elementar k, $X' = Xnovo \cap U'$

18. Fazer k=k+1. Se k <= K, ir para o passo 17, senão ir para o passo 19

19. Fazer $i = i + 1$. Se $i <= I$, ir para o passo 6. Senão, se a chamada atual veio de uma chamada recursiva, ir para o passo 18, caso contrário ir para o passo 20

20. Fazer $j = j + 1$. Se $j <= J$, ir para o passo 3, senão ir para o passo 21

21. Retornar todas as regras geradas nos passos 10 e 14 e o algoritmo termina

O exemplo 2 ilustra passo-a-passo a aplicação do algoritmo *ML-VPM* em um determinado SRC.

Exemplo 2 *Considere o SRC representado pela 5.1, onde* $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ *e* $Q = \{a, b, c, d\}$

U	**a**	**b**	**c**	**d**
x_1	4	1	3	1
x_2	6	2	5	2
x_3	6	1	5	1
x_4	4	2	4	1
x_5	4	1	4	1
x_6	6	1	4	2

Tabela 5.1: Exemplo de SRC

Considerando como atributos de condição o conjunto $C = \{a, b, c\}$, como atributo de decisão o atributo $\delta = d$, e como grau de erro de classificação máximo o valor $\beta = 0,499999$, o algoritmo *ML-VPM* realizaria os passos descritos a seguir:

1. Calcular $class(\delta)$

Conjuntos elementares	δ
$X_1 = \{x_1, x_3, x_4, x_5\}$	1
$X_2 = \{x_2, x_6\}$	2

2. $j = 1$, $J = 2$
3. $C' = \{a, b, c\}$, $U' = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $X' = X_j = \{x_1, x_3, x_4, x_5\}$, $Cant' = \emptyset$
4. Ordenar os atributos de condição de acordo com o indice discriminante

atributo	$Cant'\cup$ atributo	$\alpha_{C'_{A=(U',Cant'\cup atributo)}}(X')$
a	$\{a\}$	0.5
b	$\{b\}$	0
c	$\{c\}$	0,16

   $C' = \{a, c, b\}$
5. $i = 1, I = 3$
6. Fazer $Cnovo = Cant'\cup\{a\} = \{a\}$, $Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_1, x_3, x_4, x_5\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1, x_4, x_5\}, \{x_2, x_3, x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado.

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a\}$	$\{x_1, x_4, x_5\}$

10. Gerar a regra determinística $\boxed{\text{a=4} \implies \text{d=1}}$

11. Fazer $Xnovo = Xnovo - AinfX = \{x_3\}$

12. Fazer $Unovo = duv(Xnovo) = \{x_2, x_3, x_6\}$.

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a\}$	$\emptyset$

   Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \{b, c\}$

16. Fazer $k = 1$, $K = 1$

17. Ir para o passo 4 com $Cant' = \{a\}, C' = \{c, b\}, U' = \{x_2, x_3, x_6\}, X' = \{x_3\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante

atributo	$Cant'\cup$ atributo	$\alpha_{C'_{A=(U', Cant'\cup atributo)}}(X')$
b	$\{a, b\}$	0.33
c	$\{a, c\}$	0.33

$C' = \{b, c\}$

5. $i = 1, I = 2$
6. Fazer $Cnovo = Cant' \cup \{b\} = \{a, b\}, Unovo = \{x_2, x_3, x_6\}, Xnovo = \{x_3\}$
7. Calcular $A = (Unovo, Cnovo)$
8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3, x_6\}$	$\{\{x_2\}, \{x_3, x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, b\}$	$\emptyset$

   Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(Xnovo) = \{x_3, x_6\}$
13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a, b\}$	$\emptyset$

   Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. Fazer $Cproximo = \{c\}$
16. Fazer $k = 1, K = 1$
17. Ir para o passo 4 com $Cant' = \{a, b\}, C' = \{c\}, U' = \{x_3, x_6\}, X' = \{x_3\}$
4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{c\}$

5. $i = 1, I = 1$
6. Fazer $Cnovo = Cant' \cup \{c\} = \{a, b, c\}$
7. Calcular $A = (Unovo, Cnovo)$
8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior. Como houve melhora na classificação, o atributo é

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_3, x_6\}$	$\{\{x_3\}, \{x_6\}\}$

   utilizado.
9. Calcular a aproximação inferior exata de X em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, b, c\}$	$\{x_3\}$

10. Gerar a regra determinística | a=6 & b=1 & c=5 $\implies$ d=1 |
11. Fazer $Xnovo = X - AinfX = \emptyset$
12. Fazer $Unovo = duv(X) = \emptyset$. Ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 2$. Ir para o passo 18

18. $k = 2$. Ir para o passo 19

19. $i = 2$. Ir para o passo 6

6. Fazer $Cnovo = Cant' \cup \{c\} = \{a, c\}$, $Unovo = \{x_2, x_3, x_6\}$, $Xnovo = \{x_3\}$
7. Calcular $A = (Unovo, Cnovo)$
8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3, x_6\}$	$\{\{x_2, x_3\}, \{x_6\}\}$

Como houve melhora na classificação, o atributo é utilizado.

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, c\}$	$\emptyset$

Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(X) = \{x_2, x_3\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a, c\}$	$\emptyset$

Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 3$. Ir para o passo. Ir para o passo 18

18. $k = 2$. Ir para o passo 19

19. $i = 2$. Ir para o passo 6

6. Fazer $Cnovo = Cant' \cup \{c\} = \{c\}$, $Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_1, x_3, x_4, x_5\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjuntos elementares de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1\}, \{x_2, x_3\}, \{x_4, x_5, x_6\}\}$

Como houve melhora na classificação, o atributo é utilizado.

9. Calcular a aproximação inferior exata de X em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c\}$	$\{x_1\}$

10. Gerar a regra determinística $\boxed{\text{c=3} \implies \text{d=1}}$

11. Fazer $Xnovo = Xnovo - AinfX = \{x_3, x_4, x_5\}$

12. Fazer $Unovo = duv(Xnovo) = \{x_2, x_3, x_4, x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Bnovo)$

$Cnovo$	$AprobInfX$
$\{c\}$	$\{x_4, x_5, x_6\}$

14. Gerar a regra probabilística $\boxed{\text{c=4} \implies \text{d=1}}$, com 67% de credibilidade

15. $Cproximo = \{b\}$.

16. Fazer $k = 1, K = 2$

17. Ir para o passo 4 com $Cant' = \{c\}, C' = \{b\}, U' = \{x_2, x_3\}, X' = \{x_3\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{b\}$

5. Fazer $i = 1; I = 1$

6. Fazer $Cnovo = Cant' \cup \{b\} = \{c, b\}$, $Unovo = \{x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_3, x_4, x_5\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3\}$	$\{\{x_2\}, \{x_3\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c, b\}$	$\{x_3\}$

10. Gerar a regra determinística $\boxed{\text{c=5 \& b=1} \implies \text{d=1}}$

11. Fazer $Xnovo = Xnovo - AinfX = \emptyset$

12. Fazer $Unovo = duv(Xnovo) = \emptyset$. Ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19.

19. $i = 2$. Ir para o passo 18

18. $k = 2$. Ir para o passo 17

17. Ir para o passo 4 com $Cant' = \{c\}$, $C' = \{b\}$, $U' = \{x_4, x_5, x_6\}$, $X' = \{x_4, x_5\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{b\}$

5. $i = 1, I = 1$

6. Fazer $Cnovo = Cant \cup \{b\} = \{c, b\}$, $Unovo = \{x_4, x_5, x_6\}$, $Xnovo = \{x_4, x_5\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjuntos elementares de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_4, x_5, x_6\}$	$\{\{x_4\}, \{x_5, x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c, b\}$	$\{x_4\}$

10. Gerar a regra determinística $\boxed{\text{c=4 \& b=2} \implies \text{d=1}}$

11. Fazer $Xnovo = Xnovo - AinfX = \{x_5\}$

12. Fazer $Unovo = duv(X) = \{x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{c, b\}$	$\emptyset$

Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 2$. Ir para o passo 18

18. $k = 3$. Ir para o passo 19

19. $i = 3$. Ir para o passo 6

6. Fazer $Cnovo = Cant' \cup \{b\} = \{b\}$, $Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_1, x_3, x_4, x_5\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1, x_3, x_5, x_6\}, \{x_2, x_4\}\}$

Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{b\}$	$\emptyset$

Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(X) = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{b\}$	$\{x_1, x_3, x_5, x_6\}$

14. Gerar a regra probabilística $\boxed{\text{b=1} \implies \text{d=1}}$, com 75% de credibilidade

15. $Cproximo = \emptyset$. Ir para o passo 19.

19. $i = 2$. Ir para o passo 20

20. $j = 2$. Ir para o passo 3

3. $C' = \{a, b, c\}$, $U' = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $X' = X_j = \{x_2, x_6\}$, $Cant' = \emptyset$

4. Ordenar os atributos de condição de acordo com o índice discriminante.

atributo	$Cant'\cup$ atributo	$\alpha_{C'_{A=(U',Cant'\cup atributo)}}(X')$
a	$\{a\}$	0.5
b	$\{b\}$	0
c	$\{c\}$	0,16

   $C' = \{a, c, b\}$

5. $i = 1, I = 3$

6. $Cnovo = Cant'\cup\{a\} = \{a\}, Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}, Xnovo = \{x_2, x_6\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1, x_4, x_5\}, \{x_2, x_3, x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a\}$	$\emptyset$

   Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(X) = \{x_2, x_3, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a\}$	$\{x_2, x_3, x_6\}$

14. Gerar a regra probabilística $\boxed{\text{a=6} \implies \text{d=2}}$, com 67 % de credibilidade

15. $Cproximo = \{c, b\}$

16. $k = 1, K = 1$

17. Ir para o passo 4 com $Cant' = \{a\}$, $C' = \{c, b\}$, $U' = \{x_2, x_3, x_6\}$, $X' = \{x_2, x_6\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante.

atributo	$Cant'\cup$ atributo	$\alpha_{C'_{A=(U',Cant'\cup atributo)}}(X')$
b	$\{a, b\}$	0,33
c	$\{a, c\}$	0,33

$C' = \{b, c\}$

5. $i = 1, I = 2$

6. Fazer $Cnovo = Cant \cup \{b\} = \{a, b\}$, $Unovo = \{x_2, x_3, x_6\}$, $Xnovo = \{x_2, x_6\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3, x_6\}$	$\{\{x_2\}, \{x_3, x_6\}\}$

Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, b\}$	$\{x_2\}$

10. Gerar a regra determinística | a=6 & b=2 $\implies$ d=2 |
11. Fazer $Xnovo = Xnovo - AinfX = \{x_6\}$
12. Fazer $Unovo = duv(X) = \{x_3, x_6\}$
13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a, b\}$	$\emptyset$

Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \{c\}$
16. $k = 1, K = 1$
17. Ir para o passo 4 com $Cant' = \{a, b\}$, $C' = \{c\}$, $U' = \{x_3, x_6\}$, $X' = \{x_6\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{c\}$
5. $i = 1, I = 1$
6. Fazer $Cnovo = Cant \cup \{c\} = \{a, b, c\}$, $Unovo = \{x_3, x_6\}$, $Xnovo = \{x_6\}$
7. Fazer $A = (Unovo, Cnovo)$
8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_3, x_6\}$	$\{\{x_3\}, \{x_6\}\}$

Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, b, c\}$	$\{x_6\}$

10. Gerar a regra determinística $\boxed{\text{a=6 \& b=1 \& c=4} \implies \text{d=2}}$
11. Fazer $Xnovo = Xnovo - AinfX = \emptyset$
12. Fazer $Unovo = duv(X) = \emptyset$. Ir para o passo 15
15. $Cproximo = \emptyset$. Ir para o passo 19
19. $i = 2$. Ir para o passo 18
18. $k = 2$. Ir para o passo 19
19. $i = 2$. Ir para o passo 6
6. Fazer $Cnovo = Cant \cup \{c\} = \{a, c\}$, $Unovo = \{x_2, x_3, x_6\}$, $Xnovo = \{x_2, x_6\}$
7. Fazer $A = (Unovo, Cnovo)$
8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3, x_6\}$	$\{\{x_2, x_3\}, \{x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{a, c\}$	$\{x_6\}$

10. Como a aproximação inferior é não-vazia, é gerada a regra determinística $\boxed{\text{a=6 \& c=4} \implies \text{d=2}}$
11. Fazer $Xnovo = X - AinfX = \{x_2\}$
12. Fazer $Unovo = duv(X) = \{x_2, x_3\}$
13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{a, c\}$	$\emptyset$

Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 3$. Ir para o passo 18

18. $k = 2$. Ir para o passo 19

19. $i = 2$. Ir para o passo 6

6. Fazer $Cnovo = Cant \cup \{c\} = \{c\}$, $Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_2, x_6\}$

7. Fazer $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1\}, \{x_2, x_3\}, \{x_4, x_5, x_6\}\}$

Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c\}$	$\emptyset$

10. Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(Xnovo) = \{x_2, x_3, x_4, x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{c\}$	$\emptyset$

Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \{b\}$

16. $k = 1, K = 2$

17. Ir para o passo 4 com $Cant' = \{a, c\}$, $C' = \{b\}$, $U' = \{x_2, x_3\}$, $X' = \{x_2\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{b\}$

5. $i = 1, I = 1$

6. Fazer $Cnovo$= $Cant \cup \{b\}$ = $\{c, b\}$

7. Calcular $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_2, x_3\}$	$\{\{x_2\}, \{x_3\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de X em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c, b\}$	$\{x_2\}$

10. Gerar a regra determinística $\boxed{\text{c=5 \& b=2} \implies \text{d=2}}$

11. Fazer $Xnovo = Xnovo - AinfX = \emptyset$

12. Fazer $Unovo = duv(X) = \emptyset$. Como $Unovo = \emptyset$, ir para o passo 19

19. $i = 2$. Ir para o passo 18

18. $k = 2$. Ir para o passo 17

17. Ir para o passo 4 com $Cant' = \{c\}, C' = \{b\}, U' = \{x_4, x_5, x_6\}, X' = \{x_6\}$

4. Ordenar os atributos de condição de acordo com o índice discriminante. $C' = \{b\}$

5. $i = 1, I = 1$

6. Fazer $Cnovo = Cant \cup \{b\}$ = $\{c, b\}$, $Unovo = \{x_4, x_5, x_6\}$, $Xnovo = \{x_6\}$

7. Fazer $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_4, x_5, x_6\}$	$\{\{x_4\}, \{x_5, x_6\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{c, b\}$	$\emptyset$

   Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(X) = \{x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{c, b\}$	$\emptyset$

   Como a aproximação inferior probabilística é vazia, ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 2$. Ir para o passo 18

18. $k = 3$. Ir para o passo 19

19. $i = 3$. Ir para o passo 6

6. Fazer $Cnovo = Cant \cup \{b\} = \{b\}$, $Unovo = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $Xnovo = \{x_2, x_6\}$

7. Fazer $A = (Unovo, Cnovo)$

8. Verificar se o novo atributo melhorou a classificação obtida com o conjunto de atributos anterior

Conjunto elementar de $Cant$	Conjuntos elementares de $Cnovo$
$\{x_1, x_2, x_3, x_4, x_5, x_6\}$	$\{\{x_1, x_3, x_5, x_6\}, \{x_2, x_4\}\}$

   Como houve melhora na classificação, o atributo é utilizado

9. Calcular a aproximação inferior exata de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AinfX$
$\{b\}$	$\emptyset$

   Como a aproximação inferior é vazia, ir para o passo 12

12. Fazer $Unovo = duv(X) = \{x_1, x_2, x_3, x_4, x_5, x_6\}$

13. Calcular a aproximação inferior probabilística de $Xnovo$ em $A = (Unovo, Cnovo)$

$Cnovo$	$AprobInfX$
$\{b\}$	$\emptyset$

   Como a aproximação inferior é vazia, ir para o passo 15

15. $Cproximo = \emptyset$. Ir para o passo 19

19. $i = 2$. Ir para o passo 20

20. $j = 3$. Ir para o passo 21

21. Fim do algoritmo

Como resultado, foram geradas as seguintes regras determinísticas:

$$\begin{aligned} a = 4 &\implies d = 1 \\ a = 6 \;\&\; b = 2 &\implies d = 2 \\ a = 6 \;\&\; c = 4 &\implies d = 2 \\ c = 4 \;\&\; b = 2 &\implies d = 1 \\ c = 3 &\implies d = 1 \\ c = 5 \;\&\; b = 1 &\implies d = 1 \\ c = 5 \;\&\; b = 2 &\implies d = 2 \end{aligned}$$

E probabilísticas:

$$\begin{aligned} a = 6 &\implies d = 2 \; (67\% \text{ de credibilidade}) \\ b = 1 &\implies d = 1 \; (75\% \text{ de credibilidade}) \\ c = 4 &\implies d = 1 \; (67\% \text{ de credibilidade}) \end{aligned}$$

5.3 Conclusão

Conforme pode ser observado, o algoritmo gerou sete regras determinísticas e três regras probabilísticas a partir de de um SRC contendo sete exemplos. O algoritmo RS1+ [Uchôa (1998)] iria gerar apenas quatro regras determinísticas para esse mesmo SRC, que seriam suficientes para classificar esse conjunto de exemplos. Dessa forma, deixaria de considerar as informações probabilísticas, que poderiam ser importantes caso apenas alguns atributos sejam conhecidos, ou mesmo outras regras determinísticas que possam servir para classificar precisamente os elementos em uma determinada categoria. Por outro lado, teria um número menor de regras, o que tornaria mais rápida a busca por uma solução por um sistema especialista que utilizasse esse conjunto de regras.

Capítulo 6

Resultados e discussão

A fim de avaliar o desempenho do algoritmo, foram realizados testes com algumas das várias bases de dados disponíveis em [Merz & Murphy (1998)]. Visando permitir uma comparação de seus resultados com aqueles obtidos com outros algoritmos implementados no *ILROS*, foram escolhidas para testes duas das bases de dados utilizadas por eles, cujos resultados podem ser vistos em [Uchôa (1998)] e [Domingues & Uchoa (2002)].

- **car:** essa base consiste em 1728 exemplos de carros, classificado de acordo com sete atributos, todos eles com valores conhecidos. Foram realizados cinco divisões nessa base, da seguinte maneira: os exemplos foram divididos de forma aleatória em dois conjuntos, um com 1296 elementos (75% da base) e outro com 432 (25% da base). O primeiro conjunto foi utilizado para induzir as regras (conjunto de treinamento), e o segundo conjunto foi utilizado para testar as regras geradas (conjunto de teste).

  Para cada divisão foram realizados seis testes, que variam de acordo com o valor do grau de erro máximo (0, 0,25 ou 0,499999) e de acordo com o número de regras geradas (se o algoritmo deve gerar todas as regras possíveis a partir do conjunto de exemplos, desde que tenham um grau de erro menor que o especificado, ou se ele deve gerar apenas as regras suficientes para classificar o conjunto de treinamento).

- **monks problem:** os conjuntos de dados conhecidos como Monk's Problems foram propostos como base da primeira comparação de algoritmos de aprendizado de máquina. Existem, pois, três Monk's Problems diferentes,

aqui referenciados como monks-problem 1, monks-problem 2 e monks-problem 3. O domínio de todos eles, é o mesmo, e cada um deles possui 432 instâncias. Os Monk's Problems descrevem um domínio artificial de robótica, no qual robôs são descritos por seis atributos discretos e pertencem a uma entre duas classes disponíveis. O que deu origem a três diferentes versões de um mesmo problema foi o processo de geração dos dados. Para cada um deles, uma determinda expressão lógica envolvendo atributos de condição deveria ser satisfeita pela instância.

Assim como na base de dados **car**, foram realizados seis testes, que variam com o valor de β e também se o algoritmo deve gerar mais ou menos regras

Em cada teste foram avaliados o número de regras geradas pelo algoritmo (diferenciando as regras determinísticas das regras probabilísticas), o percentual de exemplos classificados correta e incorretamente e o percentual de regras determinísticas e probabilísticas que apresentaram 100% de suporte ao serem aplicadas no conjunto de teste. O resultado dos testes para cada uma das bases será mostrado a seguir.

1^a divisão

Número de regras geradas	570
Número de regras determinísticas	570
Número de regras probabilísticas	0
Percentual de classificações corretas	74,30%
Percentual de classificações incorretas	25,70%
Percentual de regras determinísticas com 100% de suporte	39,41
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.1: $Base\ car, 1^a\ divisão,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	943
Número de regras determinísticas	570
Número de regras probabilísticas	373
Percentual de classificações corretas	71,52%
Percentual de classificações incorretas	28,58%
Percentual de regras determinísticas com 100% de suporte	39,41%
Percentual de regras probabilísticas com 100% de suporte	16,77%

Tabela 6.2: $Base\ car, 1^a\ divisão,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	2357
Número de regras determinísticas	570
Número de regras probabilísticas	1787
Percentual de classificações corretas	59,11%
Percentual de classificações incorretas	40,89%
Percentual de regras determinísticas com 100% de suporte	39,41%
Percentual de regras probabilísticas com 100% de suporte	14,82%

Tabela 6.3: $Base\ car, 1^a\ divisão,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	230
Número de regras determinísticas	230
Número de regras probabilísticas	0
Percentual de classificações corretas	95,74%
Percentual de classificações incorretas	4,26%
Percentual de regras determinísticas com 100% de suporte	87,78%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.4: $Base\ car, 1^a\ divisão,\ \beta = 0,\ menos\ regras$

Número de regras geradas	252
Número de regras determinísticas	230
Número de regras probabilísticas	22
Percentual de classificações corretas	94,84%
Percentual de classificações incorretas	5,16%
Percentual de regras determinísticas com 100% de suporte	87,78%
Percentual de regras probabilísticas com 100% de suporte	75,00%

Tabela 6.5: $Base\ car, 1^a\ divisão,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	304
Número de regras determinísticas	230
Número de regras probabilísticas	74
Percentual de classificações corretas	69,61%
Percentual de classificações incorretas	30,39%
Percentual de regras determinísticas com 100% de suporte	87,78%
Percentual de regras probabilísticas com 100% de suporte	32,20%

Tabela 6.6: $Base\ car, 1^a\ divisão,\ \beta = 0,499999,\ menos\ regras$

2ª divisão

Número de regras geradas	596
Número de regras determinísticas	596
Número de regras probabilísticas	0
Percentual de classificações corretas	74,30%
Percentual de classificações incorretas	25,70%
Percentual de regras determinísticas com 100% de suporte	37,15
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.7: $Base\ car, 2^a\ divisão,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	939
Número de regras determinísticas	596
Número de regras probabilísticas	343
Percentual de classificações corretas	72,86%
Percentual de classificações incorretas	27,14%
Percentual de regras determinísticas com 100% de suporte	37,15%
Percentual de regras probabilísticas com 100% de suporte	16,43%

Tabela 6.8: $Base\ car, 2^a\ divisão,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	2351
Número de regras determinísticas	596
Número de regras probabilísticas	1755
Percentual de classificações corretas	62,72%
Percentual de classificações incorretas	37,28%
Percentual de regras determinísticas com 100% de suporte	37,15%
Percentual de regras probabilísticas com 100% de suporte	13,91%

Tabela 6.9: $Base\ car, 2^a\ divisão,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	235
Número de regras determinísticas	235
Número de regras probabilísticas	0
Percentual de classificações corretas	96,34%
Percentual de classificações incorretas	3,66%
Percentual de regras determinísticas com 100% de suporte	84,09%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.10: $Base\ car, 2^a\ divisão,\ \beta = 0,\ menos\ regras$

Número de regras geradas	258
Número de regras determinísticas	235
Número de regras probabilísticas	23
Percentual de classificações corretas	93,66%
Percentual de classificações incorretas	6,34%
Percentual de regras determinísticas com 100% de suporte	84,09%
Percentual de regras probabilísticas com 100% de suporte	64,29%

Tabela 6.11: $Base\ car, 2^a\ divisão,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	316
Número de regras determinísticas	235
Número de regras probabilísticas	81
Percentual de classificações corretas	71,05%
Percentual de classificações incorretas	28.95%
Percentual de regras determinísticas com 100% de suporte	84,09%
Percentual de regras probabilísticas com 100% de suporte	25,00%

Tabela 6.12: $Base\ car, 2^a\ divisão,\ \beta = 0,499999,\ menos\ regras$

3ª divisão

Número de regras geradas	566
Número de regras determinísticas	566
Número de regras probabilísticas	0
Percentual de classificações corretas	73,80%
Percentual de classificações incorretas	26,20%
Percentual de regras determinísticas com 100% de suporte	36,58
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.13: $Base\ car, 3^a\ divisão,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	908
Número de regras determinísticas	566
Número de regras probabilísticas	342
Percentual de classificações corretas	74,11%
Percentual de classificações incorretas	25,89%
Percentual de regras determinísticas com 100% de suporte	36,58%
Percentual de regras probabilísticas com 100% de suporte	17,83%

Tabela 6.14: $Base\ car, 3^a\ divisão,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	2336
Número de regras determinísticas	566
Número de regras probabilísticas	1770
Percentual de classificações corretas	62,96%
Percentual de classificações incorretas	37,06%
Percentual de regras determinísticas com 100% de suporte	36,58%
Percentual de regras probabilísticas com 100% de suporte	15,08%

Tabela 6.15: $Base\ car, 3^a\ divisão,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	217
Número de regras determinísticas	217
Número de regras probabilísticas	0
Percentual de classificações corretas	94,62%
Percentual de classificações incorretas	5,38%
Percentual de regras determinísticas com 100% de suporte	78,41 %
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.16: $Base\ car, 3^a\ divisão,\ \beta = 0,\ menos\ regras$

Número de regras geradas	235
Número de regras determinísticas	217
Número de regras probabilísticas	18
Percentual de classificações corretas	92,67%
Percentual de classificações incorretas	7,33%
Percentual de regras determinísticas com 100% de suporte	78,41%
Percentual de regras probabilísticas com 100% de suporte	58,33%

Tabela 6.17: $Base\ car, 3^a\ divisão,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	291
Número de regras determinísticas	217
Número de regras probabilísticas	74
Percentual de classificações corretas	73,26%
Percentual de classificações incorretas	26,74%
Percentual de regras determinísticas com 100% de suporte	78,41%
Percentual de regras probabilísticas com 100% de suporte	28,57%

Tabela 6.18: $Base\ car, 3^a\ divisão,\ \beta = 0,499999,\ menos\ regras$

4ª divisão

Número de regras geradas	543
Número de regras determinísticas	543
Número de regras probabilísticas	0
Percentual de classificações corretas	73,67%
Percentual de classificações incorretas	26,33%
Percentual de regras determinísticas com 100% de suporte	39,13%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.19: $Base\ car, 4^a\ divisão,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	911
Número de regras determinísticas	543
Número de regras probabilísticas	378
Percentual de classificações corretas	70,19%
Percentual de classificações incorretas	29,81%
Percentual de regras determinísticas com 100% de suporte	39,13
Percentual de regras probabilísticas com 100% de suporte	14,11

Tabela 6.20: $Base\ car, 4^a\ divisão,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	2294
Número de regras determinísticas	543
Número de regras probabilísticas	1751
Percentual de classificações corretas	59,48%
Percentual de classificações incorretas	40,52%
Percentual de regras determinísticas com 100% de suporte	39,13%
Percentual de regras probabilísticas com 100% de suporte	14,32%

Tabela 6.21: $Base\ car, 4^a\ divisão,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	211
Número de regras determinísticas	211
Número de regras probabilísticas	0
Percentual de classificações corretas	93,22%
Percentual de classificações incorretas	6,78%
Percentual de regras determinísticas com 100% de suporte	77,78 %
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.22: $Base\ car, 4^a\ divisão,\ \beta = 0,\ menos\ regras$

Número de regras geradas	226
Número de regras determinísticas	211
Número de regras probabilísticas	15
Percentual de classificações corretas	91,68%
Percentual de classificações incorretas	8,32%
Percentual de regras determinísticas com 100% de suporte	77,78 %
Percentual de regras probabilísticas com 100% de suporte	63,64 %

Tabela 6.23: $Base\ car, 4^a\ divisão,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	279
Número de regras determinísticas	211
Número de regras probabilísticas	68
Percentual de classificações corretas	73,01
Percentual de classificações incorretas	26,99
Percentual de regras determinísticas com 100% de suporte	77,78%
Percentual de regras probabilísticas com 100% de suporte	30,51%

Tabela 6.24: $Base\ car, 4^a\ divisão,\ \beta = 0,499999,\ menos\ regras$

5ª divisão

Número de regras geradas	580
Número de regras determinísticas	580
Número de regras probabilísticas	0
Percentual de classificações corretas	74,22%
Percentual de classificações incorretas	25,78%
Percentual de regras determinísticas com 100% de suporte	38,40%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.25: $Base\ car, 5^a\ divisão,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	946
Número de regras determinísticas	580
Número de regras probabilísticas	366
Percentual de classificações corretas	73,20%
Percentual de classificações incorretas	26,80%
Percentual de regras determinísticas com 100% de suporte	38,40%
Percentual de regras probabilísticas com 100% de suporte	17,97%

Tabela 6.26: $Base\ car, 5^a\ divisão,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	2360
Número de regras determinísticas	580
Número de regras probabilísticas	1780
Percentual de classificações corretas	63,07%
Percentual de classificações incorretas	36,93%
Percentual de regras determinísticas com 100% de suporte	38,40%
Percentual de regras probabilísticas com 100% de suporte	16,99%

Tabela 6.27: $Base\ car, 5^a\ divisão,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	241
Número de regras determinísticas	241
Número de regras probabilísticas	0
Percentual de classificações corretas	94,69%
Percentual de classificações incorretas	5,31%
Percentual de regras determinísticas com 100% de suporte	80,00 %
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.28: $Base\ car, 5^a\ divisão,\ \beta = 0,\ menos\ regras$

Número de regras geradas	262
Número de regras determinísticas	241
Número de regras probabilísticas	21
Percentual de classificações corretas	93,79%
Percentual de classificações incorretas	6,21%
Percentual de regras determinísticas com 100% de suporte	80,00%
Percentual de regras probabilísticas com 100% de suporte	76,92%

Tabela 6.29: $Base\ car, 5^a\ divisão,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	321
Número de regras determinísticas	241
Número de regras probabilísticas	80
Percentual de classificações corretas	71,39%
Percentual de classificações incorretas	28,61%
Percentual de regras determinísticas com 100% de suporte	80,00%
Percentual de regras probabilísticas com 100% de suporte	32,31%

Tabela 6.30: $Base\ car, 5^a\ divisão,\ \beta = 0,499999,\ menos\ regras$

Base monks 1

Número de regras geradas	237
Número de regras determinísticas	237
Número de regras probabilísticas	0
Percentual de classificações corretas	67,83%
Percentual de classificações incorretas	32,17%
Percentual de regras determinísticas com 100% de suporte	12,66
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.31: $Base\ monks - problem1,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	391
Número de regras determinísticas	237
Número de regras probabilísticas	154
Percentual de classificações corretas	64,73%
Percentual de classificações incorretas	35,27%
Percentual de regras determinísticas com 100% de suporte	12,66%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.32: $Base\ monks - problem1,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	734
Número de regras determinísticas	237
Número de regras probabilísticas	497
Percentual de classificações corretas	56,99 %
Percentual de classificações incorretas	43,01%
Percentual de regras determinísticas com 100% de suporte	12,66%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.33: $Base\ monks - problem1,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	27
Número de regras determinísticas	27
Número de regras probabilísticas	0
Percentual de classificações corretas	100%
Percentual de classificações incorretas	0%
Percentual de regras determinísticas com 100% de suporte	100%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.34: $Base\ monks - problem1,\ \beta = 0,\ menos\ regras$

Número de regras geradas	29
Número de regras determinísticas	27
Número de regras probabilísticas	2
Percentual de classificações corretas	95,12%
Percentual de classificações incorretas	4,88%
Percentual de regras determinísticas com 100% de suporte	100%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.35: $Base\ monks - problem1,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	38
Número de regras determinísticas	27
Número de regras probabilísticas	11
Percentual de classificações corretas	77,91%
Percentual de classificações incorretas	22,09%
Percentual de regras determinísticas com 100% de suporte	100%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.36: $Base\ monks - problem1,\ \beta = 0,499999,\ menos\ regras$

Base monks 2

Número de regras geradas	551
Número de regras determinísticas	551
Número de regras probabilísticas	0
Percentual de classificações corretas	74,17%
Percentual de classificações incorretas	25,83%
Percentual de regras determinísticas com 100% de suporte	32,12
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.37: $Base\ monks - problem2,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	761
Número de regras determinísticas	551
Número de regras probabilísticas	210
Percentual de classificações corretas	72,86%
Percentual de classificações incorretas	27,14%
Percentual de regras determinísticas com 100% de suporte	32,11%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.38: $Base\ monks - problem2,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	1323
Número de regras determinísticas	551
Número de regras probabilísticas	772
Percentual de classificações corretas	64,72%
Percentual de classificações incorretas	35,28%
Percentual de regras determinísticas com 100% de suporte	32,12%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.39: $Base\ monks - problem2,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	108
Número de regras determinísticas	108
Número de regras probabilísticas	0
Percentual de classificações corretas	70,34%
Percentual de classificações incorretas	29,66%
Percentual de regras determinísticas com 100% de suporte	41,67%
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.40: $Base\ monks - problem2,\ \beta = 0,\ menos\ regras$

Número de regras geradas	115
Número de regras determinísticas	108
Número de regras probabilísticas	7
Percentual de classificações corretas	71,43%
Percentual de classificações incorretas	28,57%
Percentual de regras determinísticas com 100% de suporte	41,67%
Percentual de regras probabilísticas com 100% de suporte	28,57%

Tabela 6.41: $Base\ monks - problem2,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	144
Número de regras determinísticas	108
Número de regras probabilísticas	36
Percentual de classificações corretas	65,97%
Percentual de classificações incorretas	34.03%
Percentual de regras determinísticas com 100% de suporte	41,67%
Percentual de regras probabilísticas com 100% de suporte	0%

Tabela 6.42: $Base\ monks - problem2,\ \beta = 0,499999,\ menos\ regras$

Base monks 3

Número de regras geradas	278
Número de regras determinísticas	278
Número de regras probabilísticas	0
Percentual de classificações corretas	80,47%
Percentual de classificações incorretas	19,53%
Percentual de regras determinísticas com 100% de suporte	39,21
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.43: $Base\ monks - problem3,\ \beta = 0,\ todas\ as\ regras$

Número de regras geradas	426
Número de regras determinísticas	278
Número de regras probabilísticas	148
Percentual de classificações corretas	79,08%
Percentual de classificações incorretas	20,92%
Percentual de regras determinísticas com 100% de suporte	39,21%
Percentual de regras probabilísticas com 100% de suporte	20,27%

Tabela 6.44: $Base\ monks - problem3,\ \beta = 0,25,\ todas\ as\ regras$

Número de regras geradas	709
Número de regras determinísticas	278
Número de regras probabilísticas	431
Percentual de classificações corretas	67,48%
Percentual de classificações incorretas	32,52%
Percentual de regras determinísticas com 100% de suporte	39,21%
Percentual de regras probabilísticas com 100% de suporte	11,60%

Tabela 6.45: $Base\ monks - problem3,\ \beta = 0,499999,\ todas\ as\ regras$

Número de regras geradas	39
Número de regras determinísticas	39
Número de regras probabilísticas	0
Percentual de classificações corretas	93,40%
Percentual de classificações incorretas	6,60%
Percentual de regras determinísticas com 100% de suporte	82,05 %
Percentual de regras probabilísticas com 100% de suporte	não se aplica

Tabela 6.46: $Base\ monks - problem3,\ \beta = 0,\ menos\ regras$

Número de regras geradas	44
Número de regras determinísticas	39
Número de regras probabilísticas	5
Percentual de classificações corretas	89,19 %
Percentual de classificações incorretas	10,81%
Percentual de regras determinísticas com 100% de suporte	82,05%
Percentual de regras probabilísticas com 100% de suporte	20%

Tabela 6.47: $Base\ monks - problem3,\ \beta = 0,25,\ menos\ regras$

Número de regras geradas	52
Número de regras determinísticas	39
Número de regras probabilísticas	13
Percentual de classificações corretas	75%
Percentual de classificações incorretas	25%
Percentual de regras determinísticas com 100% de suporte	82,05%
Percentual de regras probabilísticas com 100% de suporte	23,08%

Tabela 6.48: $Base\ monks - problem3,\ \beta = 0,499999,\ menos\ regras$

Pode-se observar, pelos testes realizados, que o número de regras probabilísticas aumenta à medida que se aumenta o grau de erro máximo. Isso é devido ao fato de que podemos classificar mais elementos às custas de um erro maior. Com isso, o percentual de classificações corretas diminui à medida que se aumenta esse valor. Obviamente, a variação do grau de erro não influi sobre o número de regras determinísticas geradas.

Nos testes onde foram geradas mais regras (aqueles onde o algoritmo buscou extrair todas as regras possíveis do conjunto de treinamento), o percentual de classificações corretas foi bem menor que o observado quando foram geradas apenas as regras suficientes para classificar o conjunto de treinamento (menos regras).

A opção de gerar mais regras, além de consumir um tempo de processamento maior, gera um número de regras muito grande, o que acaba inviabilizando sua utilização em diversas aplicações. Essa opção poderia ser interessante caso fossem utilizados pesos de atributos, onde, por exemplo, alguns atributos teriam um custo maior e deveriam ser evitados caso houvesse outra opção (outra regra), com o mesmo poder de classificação, da qual ele não faça parte. Nesse caso, poderia ser feita uma filtragem nas regras geradas a fim de eliminar aquelas que puderem ser substituídas por outras contendo atributos de menor custo. No entanto, seria mais interessante investir numa outra abordagem, modificando o algoritmo ou gerando outro desenvolvido especificamente para trabalhar nessas com essas condições.

Por fim, deve-se levar em conta que, ao se gerar mais regras do que o suficiente para classificar o conjunto de treinamento, pode-se obter várias regras que se apliquem a um mesmo exemplo, sejam elas probabilísticas ou determinísticas. Surge então a questão de como utilizá-las. Se houver ao menos uma regra determinística que se aplique a um mesmo exemplo, ela pode ser utilizada para classificá-lo, uma vez que todas as suas regras determinísticas devem levar a um mesmo resultado. Caso haja apenas regras probabilísticas, cabe à aplicação decidir qual delas deve ser utilizada. Pode-se utilizar aquela que possuir a maior credibilidade, ou então a que possuir o menor número de atributos, por exemplo. Mas a escolha da ação a ser tomada nesse caso deve ser definida por aqueles que forem desenvolver um sistema especialista que utilize essas regras.

A principal diferença entre esse algoritmo e o algoritmo RS1+, apresentado em [Uchôa (1998)], é o fato de esse último tentar gerar regras probabilísticas sempre que é adicionado um novo atributo ao conjunto de atributos em consideração, mesmo que futuramente sejam geradas regras determinísticas que se apliquem aos exemplos já classificados por regras probabilísticas. O algoritmo RS1+, por sua vez, prossegue adicionando atributos sempre que eles implicam em uma melhora na classificação obtida com o conjunto de atributos anterior, mas sempre buscando gerar apenas regras determinísticas. Regras probabilísticas somente são geradas depois que todos os atributos foram utilizados e ainda assim não foi possível classificar elementos de uma classe numa mesma categoria. Nesse caso, são geradas regras para cada uma das categorias envolvidas, porém com grau de credibilidade

proporcional ao número de elementos do conjunto classificados em cada uma delas. Essa é outra diferença entre os algoritmos, pois o algoritmo *ML-VPM* gera apenas regras probabilísticas que possuam mais de 50% de credibilidade, o que faria com que fosse gerada, nesse caso, no máximo uma regra probabilística.

Capítulo 7

Conclusão

O presente trabalho teve por objetivo apresentar um novo algoritmo de aprendizado, denominado algoritmo *ML-VPM*, que se baseia no modelo de precisão variável da TCA, apresentado em [Ziarko (1990)]. Seguindo esse modelo, esse algoritmo extrai um número maior de regras que aqueles baseados no modelo original da TCA, pois considera informações que eram desconsideradas por eles, as informações probabilísticas, indispensáveis para uma classificação não-determinística.

A idéia para utilizar esse tipo de informação parte do pressuposto de que essas informações probabilísticas, mesmo que parcialmente incorretas, podem ser importantes, desde que estejam dentro de um grau controlado de erro. Isso permite extrair uma tendência nos dados, que pode ter várias aplicações, inclusive em processos de tomada de decisão. Sua utilização tem ainda como reforço o fato de se estar trabalhando com informações incertas, pois um pequeno ruído em um exemplo poderia fazer com que uma regra determinística deixasse de ser gerada.

O algoritmo desenvolvido permite ao usuário definir o grau de erro máximo a ser considerado na geração de regras, que pode variar desde 0 (classificação sem erro, gera apenas regras determinísticas) até 0,499999 (maior valor no programa onde é possível obter regras nas quais a maioria dos elementos é classificada corretamente). É possível ainda definir se devem ser geradas todas as regras possíveis a partir do conjunto de treinamento, desde que possuam um grau de erro que não seja maior que o valor máximo, ou se devem ser geradas apenas as regras suficientes para classificar o conjunto de treinamento. Nesse último caso, haveria no máximo uma regra gerada que se aplicaria a cada exemplo, enquanto no caso anterior seria possível ter várias regras, determinísticas ou probabilísticas, que se aplicariam

a um mesmo exemplo. Os tipos de opções a serem utilizadas irão depender da aplicação.

O algoritmo *ML-VPM* foi incorporado ao sistema *ILROS*, um protótipo desenvolvido em [Uchôa (1998)] e que vem sendo aprimorado no decorrer desse projeto. Atualmente, esse protótipo conta com quatro algoritmos de aprendizado: o algoritmo RS1+ ([Uchôa (1998)]), o algoritmo ID3 ([Quinlan (1986)]), o algoritmo *Lmurf* ([Domingues & Uchoa (2002)]) e o algoritmo *ML-VPM*. Dentre algumas das melhorias que ainda podem ser incorporadas ao *ILROS*, destacam-se:

- **Tratamento de valores desconhecidos**
  Durante os testes dos algoritmos implementados no *ILROS*, foi observado que várias das bases de dados utilizadas apresentavam exemplos incompletos, ou seja, exemplos cujos valores de alguns atributos estavam ausentes. Como o *ILROS* não trata esse tipo de informação, foram utilizados apenas os elementos que não continham valores desconhecidos, sendo desconsideradas quaisquer informações encontradas nos exemplos incompletos.

  Essa limitação de não trabalhar com informações incompletas acaba prejudicando a geração de regras, devido à subutilização do conjunto de treinamento. Ao incorporar essa funcionalidade a um dos algoritmos de aprendizado do *ILROS*, este se tornará aplicável a um número muito maior de situações.

- **Tratamento de valores continuos**
  Podemos classificar os atributos que descrevem elementos em um conjunto de treinamento como qualitativos (não-ordenáveis) ou quantitativos (ordenáveis). Exemplos de atributos qualitativos seriam a presença de manchas na pele de um doente, a marca do motor de um veículo ou a cor preferida de uma pessoa. Exemplos de atributos quantitativos seriam a temperatura em uma fornalha, a quantidade de fertilizantes aplicada em uma cultura, ou mesmo a faixa etária dos clientes que compram uma determinada mercadoria.

  Esses últimos tipos de atributos se caracterizam por serem atributos contínuos, ou seja, são atributos que podem ser especificados por meio de uma faixa de valores, um intervalo.

  Os algoritmos implementados até então no *ILROS* não explorarm a funcionalidade desse tipo de atributo, utilizando-o como os demais. A proposta

é, então, incorporar a um de seus algoritmos esse recurso, gerando regras que trabalhem com intervalos de valores, em vez de trabalhar com valores únicos, o que reduz o número de regras geradas.

Num exemplo bem simples, que descreve o estado físico da água em função de sua temperatura, os algoritmos atuais poderiam gerar regras do tipo:

Se temperatura = 1^oC $\Longrightarrow$ estado = líquido
Se temperatura = 2^oC $\Longrightarrow$ estado = líquido
Se temperatura = 3^oC $\Longrightarrow$ estado = líquido
...
Se temperatura = 99^oC $\Longrightarrow$ estado = líquido

Com a incorporação do tratamento de valores contínuos, espera-se obter:

Se $1 \leq$ temperatura ≤ 99 $\Longrightarrow$ estado = líquido

O que tornaria uma busca por informações na base de conhecimentos muito mais eficiente.

- **Aprendizado pseudo-incremental**
  Em sua maior parte, os algoritmos de aprendizado indutivo disponíveis na literatura, assim como os implementados no protótipo *ILROS*, realizam o chamado aprendizado não-incremental, ou seja, extraem o conhecimento usando todo o conjunto de treinamento de uma vez.

  Contrasta com esse tipo de aprendizado o chamado aprendizado incremental, ou aprendizado construtivo, onde o conhecimento é obtido após a adição de cada novo exemplo, sem ter que reconstruir toda a base de dados. Um algoritmo de aprendizado incremental baseado na TCA é apresentado em [Tsumoto & Tanaka], que foi denominado PRIMEROSE-INC (Métodos de Indução de Regras Probabilísticas baseado em Conjuntos Aproximados para Métodos de Aprendizado Incremental). O resultado de testes mostrou que as regras geradas por esse algoritmo são as mesmas geradas por métodos não-incrementais, entretanto o custo computacional é bem menor.

  Nesse projeto, é proposto o desenvolvimento de um algoritmo que realize um aprendizado pseudo-incremental, um ”meio-termo” entre essas duas técnicas. A idéia desse tipo de aprendizado consiste em dividir o conjunto de treinamento em blocos, e usar esses blocos para induzir as regras de

forma gradativa, executando o algoritmo a medida que um novo bloco é adicionado. O sistema deve ser capaz de adicionar novas regras ou modificar as já existentes, de modo a suportar os novos exemplos adicionados.

O *ILROS* foi desenvolvido em linguagem C++, seguindo a metodologia de orientação a objetos, e é distribuído sob a licença GPL ([GNU (1999)]). Sua interface gráfica foi desenvolvida em *wxWindows* (`http://www.wxwindows.org`), o que o torna disponível para um grande número de plataformas. Pretende-se desenvolver uma página na internet a fim de divulgar os resultados desse projeto, bem como disponibilizar a nova versão do *ILROS* para *download*.

Referências Bibliográficas

[Bonissone (1991)] Bonissone, P. Plausible reasoning. In: Shapiro, S. C.; Eckroth, D. & Valassi, G. A. (Eds.). *Encyclopedia of Artificial Intelligence*. New York, John Wiley & Sons, 1991. p.854-863.

[Boose (1989)] Boose, J. H. *A knowledge acquisition: techniques and tools. Knowledge Aquisition* (1): 3-37, 1989.

[Cawsey (1994)] Cawsey, A. *Databases and Artificial Intelligence 3*. Disponível em http://www.cee.hw.ac.uk/ alison/ai3notes/all.html. Citado em 19 de Agosto de 1994

[Domingues & Uchoa (2002)] Domingues, M. A. & Uchôa, J. Q. *Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incertezas com Ênfase na Teoria de Conjuntos Aproximados*. [Monografia de graduação]. UFLA, DCC, 2002.

[Fausett (1993)] Fausett, L. V. *Fundamentals of neural networks: Architectures, Algorithms And Applications*. New York, Prentice Hall, 1993.

[GNU (1999)] Free Software Foundation. *GNU's Not Unix! the GNU project and the Free Software Foundation (FSF)*. url: `http://www.gnu.org`.

[Klir & Yuan (1995)] Klir, J. G. & Yuan, B. *Fuzzy sets and fuzzy logic: theory and applications*. New Jersey, Prentice Hall, 1995.

[Langley (1996)] Langley, P. *Elements of machine learning*. San Francisco, Morgan Kaufmann, 1996.

[McCulloch & Pitts (1943)] McCullock, W. & Pitts, W. *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematica Biophysics, (5):115-137.

[Merz & Murphy (1998)] Merz, C. J. & Murphy, P. M. UCI *Repository of Machine Learning databases*. Irvine, University of California, Department of Information and Computer Science, 1998. [http://www.ics.uci.edu/ mlearn/MLRepository.html].

[Michalski & Tecuci (1993)] Michalski, R. S. & Tecuci, G. *Multistrategy learning*. Tutorial T15, IJCAI-1993, 1993.

[Morales & Harris] Morales, E. & Harris, C. *The History of Artificial Intelligence* Disponível em http://web.mit.edu/STS001/www/Team7/home.html. Citado em 29 de novembro de 2002.

[Nicoletti & Uchôa (1997)] Nicoletti, M. C. & Uchôa, J. Q. *O uso de funções de pertinência na caracterização dos principais conceitos da teoria de conjuntos aproximados*. Relatório Técnico do Departamento de Computação 005/97. São Carlos, DC-UFSCar, 1997. 26p.

[Nicoletti (1994)] Nicoletti, M. C. *Ampliando os Limites do Aprendizado Indutivo de Máquina através de Abordagens Construtiva e Relacional*. Tese (Doutorado). Universidade Federal de São Carlos, Sâo Carlos/SP.

[Pawlak (1982)] Pawlak, Z. *Rough Sets*. International Journal of Computer and Information Sciences, 11(5):341-356, 1982.

[Pawlak, Wong & Ziarko (1988)] Pawlak, Z., Wong, S. K. M. & Ziarko, W. *Rough sets: probabilistic versus deterministic approach*. Int J. Man-Machine Studies. 1988, p. 81-95.

[Pawlak (1991)] Pawlak, Z. *Rough sets: theoretical aspects of reasoning about data*. London, Kluwer, 1991.

[Pawlak (1994)] Pawlak, Z. Hard and soft sets. In: Ziarko, W. P. (Ed). *Rough sets, fuzzy sets and knowledge discovery*. London, Springer-Verlag, 1994. p.130-135.

[Quinlan (1986)] Quinlan, J. R. *Induction of decision trees. Machine learning*, (1):81-106, 1986.

[Shafer (1976)] Shafer, G. *A mathemathical theory of evidence*. Princeton, Princeton University Press, 1976.

[Shaw & Gentry (1990)] Shaw, M. J. & Gentry, J. A. *Inductive learning for risk classification*. IEEE Expert, (February):47-53, 1990.

[Tsumoto & Tanaka] Tsumoto, S. & Tanaka, H. *Incremental Learning of Probabilistic Rules from Clinical Databases based on Rough Set Theory*. Disponível em http://www.amia.org/pubs/symposia/D004399.PDF. Citado em 29 de novembro de 2002.

[Uchôa & Nicoletti (1997)] Uchôa, J. Q. & Nicoletti, M. C. *Elementos teoria de conjuntos aproximados*. Relatório Técnico do Departamento de Computação 001/97. São Carlos, DC-UFSCar, 1997. 25p.

[Uchôa et ali (1997)] Uchôa, J. Q.; Panotim, S. M. & Nicoletti, M. C . *Elementos da Teoria de Dempster-Shafer*. Relatório Técnico do Departamento de Computação 007/97. São Carlos, DC-UFSCar, 1997, 34p.

[Uchôa & Nicoletti (1998b)] Uchôa, J. Q. & Nicoletti, M. C. *ILROS: um sistema de aprendizado indutivo de máquina baseado em conjuntos aproximados*. Relatório Técnico do Departamento de Computação. São Carlos, DC-UFSCar, 1998. 38p.

[Uchôa (1998)] Uchôa, J. Q. *Representação e indução de conhecimento usando conjuntos aproximados*. [Dissertação de Mestrado]. São Carlos, PPG-CC UFSCar, 1988. 256 p.

[Yao, Wong & Lin] Yao, Y. Y.; Wong, S. K. M. & Lin, T. Y. *A review of rough set models* Boston, Kluwer Academic, 1997.

[Zadeh (1965)] Zadeh, L. Fuzzy sets. *Information and Control*, (8):338-353, 1965.

[Ziarko (1990)] Ziarko, W. *Variable precision Rough Set Model*. Journal of Computer and System Sciences, 1 de Junho de 1990. p.39-59.