JUNIOR ASSIS BARRETO BERNARDES

ALGORITMO DE APRENDIZADO DE MÁQUINA E REPRESENTAÇÃO DE INCERTEZA EM SISTEMAS BASEADOS EM CONHECIMENTO SOB A ÓTICA DE FUNÇÕES DE PERTINÊNCIA APROXIMADA

Lavras

Minas Gerais - Brasil

2010

JUNIOR ASSIS BARRETO BERNARDES

ALGORITMO DE APRENDIZADO DE MÁQUINA E REPRESENTAÇÃO DE INCERTEZA EM SISTEMAS BASEADOS EM CONHECIMENTO SOB A ÓTICA DE FUNÇÕES DE PERTINÊNCIA APROXIMADA

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 a obtenção do título de Bacharel em Ciência da Computação

Orientador

Prof. Joaquim Quinteiro Uchôa

Lavras

Minas Gerais - Brasil

2010

JUNIOR ASSIS BARRETO BERNARDES

ALGORITMO DE APRENDIZADO DE MÁQUINA E REPRESENTAÇÃO DE INCERTEZA EM SISTEMAS BASEADOS EM CONHECIMENTO SOB A ÓTICA DE FUNÇÕES DE PERTINÊNCIA APROXIMADA

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 a obtenção do título de Bacharel em Ciência da Computação

Aprovada em *09 de Novembro de 2010*

Prof. Cláudio Fabiano Mota Toledo

Prof. Ahmed Ali Abdala Esmin

Prof. Joaquim Quinteiro Uchôa

Orientador

Lavras

Minas Gerais - Brasil

2010

Aos meus pais e ídolos, Jorge e Janete, e

à minha amada e eterna companheira Carolina, dedico.

AGRADECIMENTOS

Gostaria de agradecer a Deus pela força, paciência, sabedoria e saúde.
Gostaria de agradecer aos meus pais, Jorge e Janete, por tudo que eles fizeram por mim durante toda minha vida. Jamais chegaria até aqui se não fosse por vocês. **Muito Obrigado por tudo!**
Gostaria de agradecer a minha namorada e futura esposa Carolina pelo carinho, amor incondicional, cumplicidade, incentivo e compreensão nas ausências. Você foi fundamental nessa conquista assim como é fundamental na minha vida! **Muito Obrigado por tudo!**
Gostaria de agradecer aos meus sogros Júlio e Jane, pelo apoio e confiança de sempre.
Gostaria de agradecer a todos os meus familiares pelo apoio durante toda minha vida.
Gostaria de agradecer a todos os meus amigos, em especial a minha turma (2007/01) representada aqui por Leandro Matioli, Cláudio Vinícius e João Lucas, pela amizade e pelas ajudas fundamentais.
Gostaria de agradecer ao prof. Joaquim Uchôa pela oportunidade, pela ajuda e orientação durante todo o trabalho e pela compreensão nos momentos difíceis.

RESUMO

A Teoria de Conjuntos Aproximados tem sido utilizada em várias áreas de pesquisa, principalmente naquelas relacionadas com representação de conhecimento e aprendizado de máquina. Este trabalho investiga a possibilidade do uso de funções de pertinência aproximada (uma extensão das funções de pertinência clássicas no contexto de *conjuntos aproximados*) para auxiliar e modelar a construção de algoritmos de aprendizado de máquina e representação de incerteza.

Palavras-chave: Aprendizado de Máquina; Sistemas Baseados em Conhecimento; Teoria de Conjuntos Aproximados; Funções de Pertinência Aproximada.

ABSTRACT

The Rough Sets Theory has been used in various areas of research, especially those related to knowledge reasoning and machine learning. This work investigates the possibility of using rough membership functions (an extension of the classical membership functions in the context of *rough sets*) to help and model the construction of machine learning algorithms and representation of uncertainty.

Keywords: Machine Learning; Knowledge-Based Systems; Rough Set Theory; Rough Membership Functions.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Com o advento da *Internet* e diante de sua atual popularização, tem-se o imenso espalhamento de gigantescas quantidades de informações por todo o planeta e uma relativa facilidade de acesso a tais informações. Esse fato tem fomentado uma necessidade já existente por mecanismos de descoberta de conhecimentos a partir de um grande volume de dados. Um mecanismo largamente utilizado nesse sentido é o aprendizado de máquina que, de acordo com Wang, Ma e Zhou (2009), consiste em estudar a utilização de computadores para simular atividades humanas de aprendizagem e desenvolver métodos auto-incrementais de obtenção de novos conhecimentos e novas habilidades e identificação de conhecimento já existente.

Para o aprendizado de máquina se tornar viável nesses casos, as informações devem estar disponíveis de uma maneira sistemática, para que os algoritmos de aprendizado possam agir sobre elas. Desta forma, as informações são, normalmente, organizadas em Sistemas Baseados em Conhencimento (SBCs). Como visto em (UCHÔA, 1998),

> Sistemas baseados em conhecimento são formalmente 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.

A criação de SBCs é uma das áreas mais importantes da inteligência artificial. Os SBCs são, geralmente, representados pelos sistemas especialistas que são definidos, segundo Uchôa (1998), como sistemas capazes de resolver problemas bem específicos do mundo real, cuja resolução demandam uma considerável habilidade, conhecimento e heurísticas.

Os SBCs devem ser capazes de representar, manipular e comunicar informações. É fato que tais sistemas devem estar preparados para modelar e tratar incertezas. Muitas vezes, o que se convenciona chamar de incertezas abrange infor-

mações imprecisas, inconsistentes, e incompletas. Como apontado por Bonissone (1991) citado por Uchôa (1998),

> 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.

Nesse contexto, a Teoria de Conjuntos Aproximados (TCA) foi proposta por Pawlak (1982), como um novo modelo matemático para representação do conhecimento, tratamento de incerteza e classificação aproximada. Em (UCHÔA, 1998) pode ser verificado que a TCA pode ser utilizada com sucesso na implementação de métodos de representação de conhecimento incerto, bem como um formalismo matemático subsidiando aprendizado de máquina. Este formalismo é responsável por tornar tal modelo superior aos outros existentes, mas é subestimado na literatura.

A TCA é capaz de representar apenas um tipo de incerteza: a indiscernibilidade. Além disso, a TCA também não necessita e nem leva em conta qualquer tipo de informação a respeito dos dados. Assim, em alguns casos práticos, existe a possibilidade de os conjuntos aproximados serem excessivamente aproximados.

Desta maneira, o objetivo deste trabalho é, de maneira geral, estudar o formalismo para a representação de incerteza proporcionado pela TCA e uma possível aplicação do mesmo na construção de um novo algoritmo de aprendizado de máquina utilizando funções de pertinência aproximada (uma extensão dos conceitos básicos da TCA ao conceito de pertinência entre elementos e conjuntos).

Quanto aos objetivos específicos, tem-se:

- apresentar as propriedades matemáticas das funções de pertinência aproximada;

- verificar a existência de modelos alternativos da Teoria de Conjuntos Aproximados inspirados em funções de pertinência aproximada;
- avaliar algoritmos de aprendizado de máquina baseados em funções de pertinência aproximada, caso existam;
- avaliar o relacionamento da função de pertinência *Fuzzy* e funções de pertinência aproximada;
- desenvolver um algoritmo de aprendizado de máquina baseado em funções de pertinência aproximada (uma extensão dos conceitos básicos da TCA ao conceito de pertinência entre elementos e conjuntos) juntamente com um protótipo que utilize este algoritmo e outros conceitos básicos da TCA;
- testar o protótipo desenvolvido com bases de dados públicas usador por outros algoritmos inspirados na TCA, para efeito comparativo.

Nos capítulos a seguir serão apresentados uma revisão dos conceitos básicos de Sistemas Baseados em Conhecimento e Aprendizado de Máquina (Capítulo 2); uma revisão dos principais conceitos e medidads da Teoria dos Conjuntos Aproximados (Capítulo 3); uma breve revisão dos conceitos de funções de pertinência aproximada e conjuntos *Fuzzy* (Capítulo 4); um resumo dos principais algoritmos de aprendizado de máquina baseados na TCA (Capítulo 5); uma proposta de um novo algoritmo de aprendizado de máquina inspirado na TCA, que utiliza o conceito de funções de pertinência aproximada (Capítulo 6); os resultados obtidos neste trabalho (Capítulo 7) e uma conclusão (Capítulo 8).

2 Sistemas Baseados em Conhecimento

Sistemas Baseados em Conhecimento (SBCs) podem ser definidos como programas de computador que resolvem problemas utilizando uma linha de raciocínio em cima do conhecimento representado explicitamente e que, não fosse essa representação, exigiriam um especialista humano no domínio do problema para a sua solução. A aquisição e a representacação do conhecimento e a arquitetura do processo de resolução de problemas são pontos críticos e essenciais durante o desenvolvimento de um SBC. Segundo Hayes-Roth e Jacobstein (1994), esse processamento racional do conhecimento geralmente é utilizado por três motivos em especial:

- para aumentar a capacidade de empregar linhas de raciocínio por parte das aplicações;
- para incrementar a flexibilidade dos sistemas que o utilizam;
- para aumentar e melhorar as qualidades humandas dessas aplicações.

A arquitetura básica de um SBC pode ser visualizada na Figura 1.

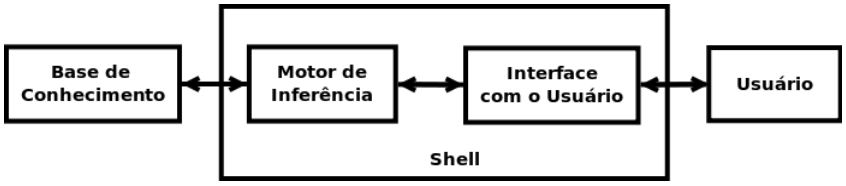


Figura 1: Arquitetura básica de um SBC (Extraída de (UCHÔA, 1998)).

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

1. **Base de Conhecimentos (BC):** contém o conhecimento específico do domínio da aplicação organizado em forma de objetos ou regras. Tal conehcimento pode ser entendido como um tipo de informação que pode aumentar a eficiência e a eficácia na resolução de um problema. Objetos são estruturas

de dados, normalmente organizadas em formas de árvores que representam a hierarquia e a relação entre as estruturas, usadas para representar conhecimento a respeito de coisas reais. Já a regras representam o conhecimento de forma declarativa, descrevendo como é o funcionamento, e/ou imperativa, dizendo ao SBC como se comportar. A BC, de uma forma geral, pode contar com três tipos de conhecimento: fatos que expressam proposições válidas, crenças que expressam proposições plausíveis e heurísticas ou regras que expressam métodos de aplicar julgamento em situações em que não existem algoritmos válidos.

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 o encadeamento para frente *(forward chaining)*, onde hipóteses conhecidas são utilizadas para derivar novos fatos e/ou conclusões, ou o encadeamento para trás *(backward chaining)*, em que se utiliza objetivos e/ou conclusões já conhecidos para validar as hipóteses. Quando o conhecimento do SBC está expresso em formas de árvore de objetos, o MI faz uso de algoritmos de busca e percurso em árvores no processamento das informações da BC.

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.

De acordo com Niwa (1990), os SBC podem ser categorizados, quanto à transferência de conhecimento, em Sistemas Especialistas *(experts systems)* e Sistemas de Compartilhamento de Conhecimento *(knowledge sharing systems)*. Niwa (1990) ainda define transferência de conhecimento como o *"fluxo de conhecimento a partir dos fornecedores (especialistas) até os usuários finais (iniciantes), passando pelo SBC"*.

Já nos Sistemas de Compartilhamento de Conhecimento, é utilizado o paradigma de compartilhamento *(sharing paradigm)*, em que os especialistas, além de

fornecerem conhecimentos específicos para a BC, utilizam o sistema para obter respostas e raciocínios, avaliam tais retornos e trocam experiências entre si a fim de melhorar a BC. A Figura 2 traduzida de (NIWA, 1990) ilustra o paradigma de compartilhamento:

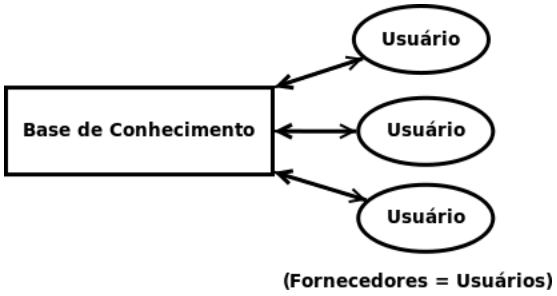


Figura 2: Paradigma de compartilhamento presente nos sistemas de compartilhamento de conhecimento (Adaptada de (NIWA, 1990)).

Os Sistemas Especialistas (SE) são desenvolvidos e utilizados sob um paradigma chamado paradigma de consulta *(consulting paradigm)*, onde os especialistas fornecem o conhecimento específico de domínio para a base de conhecimentos e os usuários finais apenas consultam o SE para obter respostas e explicações resultantes do processamento efetuado pelo motor de inferência. A Figura 3 traduzida de (NIWA, 1990) mostra o paradigma de consulta:

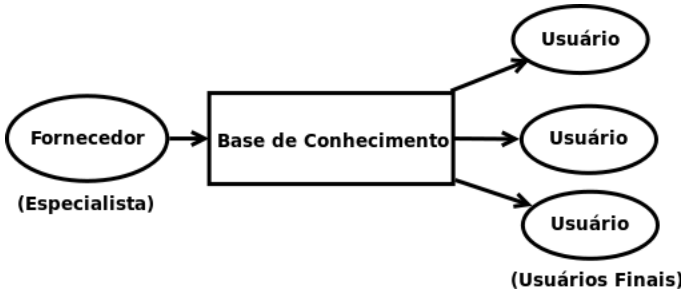


Figura 3: Paradigma de consulta presente nos sistemas especialistas (Adaptada de (NIWA, 1990)).

Para [Hayes-Roth, (1984)], *"especialistas são distinguidos pela qualidade e quantidade de conhecimento que eles possuem; eles sabem mais e o que eles sa-*

bem fazem deles mais eficientes e eficazes". Assim, um Sistema Especialista 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. Além disso, devido a sutileza da classificação dos SBCs apresentada anteriormente, tem-se utilizado, na literatura, os termos Sistemas Baseados em Conhecimento e Sistemas Especialistas quase sem distinção.

A grande maioria dos SBCs, de acordo com Uchôa (1998), 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. Desta maneira, como citado anteriormente, o MI utiliza das técnicas de encadeamento para raciocinar e gerar uma resposta consistente. 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?

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).

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, aumentar 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.

2.1 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 (1989b) agrupa os métodos e técnicas para AC em:

1. **Manuais -** que tipicamente consistem de entrevistas e análise de protocolos. O engenheiro de conhecimento é responsável por intermediar a relação entre os especialistas fornecedores de conhecimento e a base de conhecimentos do SBC. Ele elicita conhecimento do especialista, modela-o e codifica-o para ser inserido na BC. A Figura 4 extraída de (BOOSE, 1989a) mostra o contexto da aquisição manual de conhecimento.

2. **Baseadas em Computador:**

   (a) ***Baseadas em Aprendizado -*** as quais, via de regra, generalizam situações específicas em conceitos. São baseadas em técnicas de aprendi-

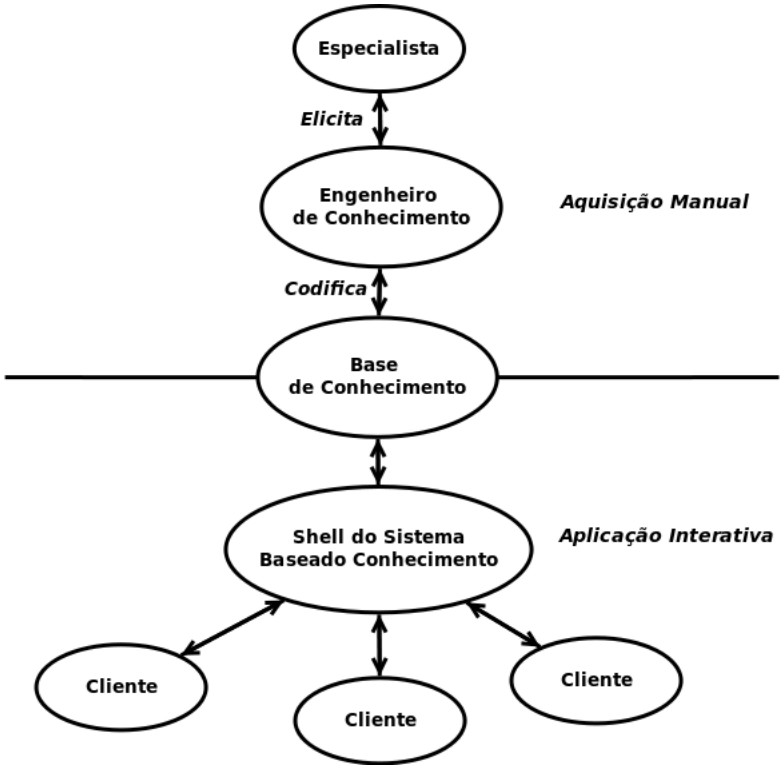


Figura 4: Aquisição manual de conhecimento (Adaptada de (BOOSE, 1989a)).

zado de máquina tradicionais. Nesse caso, a necessidade dos engenheiros de conhecimento para intermediar a relação entre os especialistas e o SBC é reduzida, já que o processo de extração e representação do conhecimento é automatizado. Porém, como apresentado em (BOOSE, 1989a) e ilustrado na Figura 5, os engenheiros têm outros papéis, como notificar os especialistas de como está o processo de aquisição do conhecimento, gerenciar a codificação do conhecimento, editar a base de conhecimentos parcialmente codificada juntamente com os especialistas, validar a aplicação da base de conhecimentos e treinar os usuários finais (clientes) para o uso efetivo da base de conhecimentos.

(b) ***Interativas -*** englobam, principalmente, ferramentas que entrevistam o especialista, fazem análise textual, extraem e analisam conhecimento

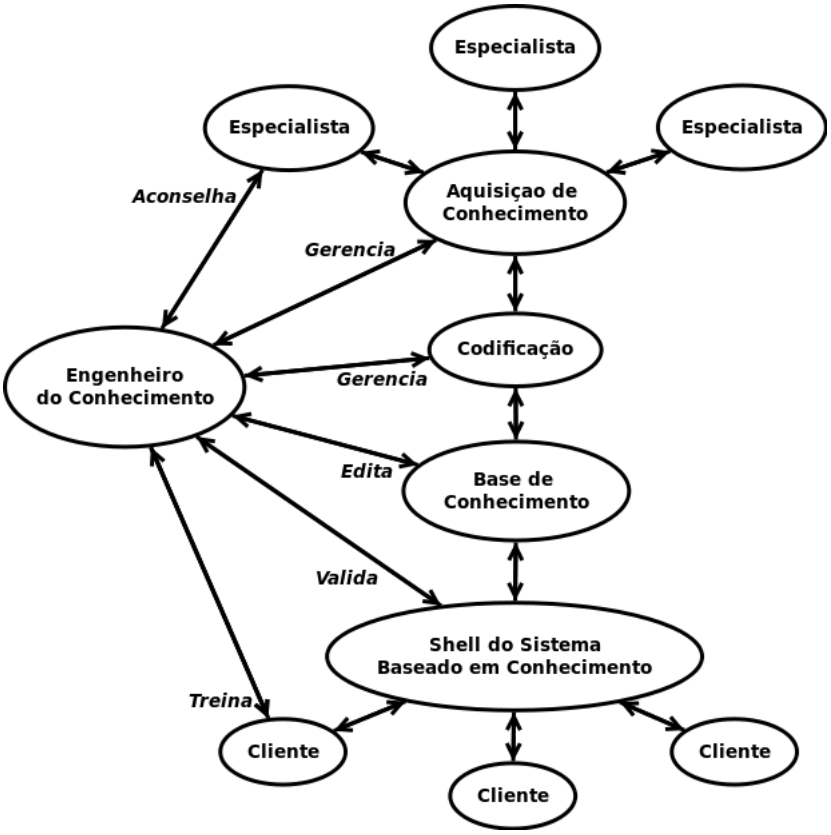


Figura 5: Aquisição automatizada de conhecimento (extraída de (BOOSE, 1989a)).

de múltiplas fontes separadamente e as combinam para uso. Técnicas manuais e automatizadas são combinadas para extrair o conhecimento dos especialistas (manual) e inserí-lo de acordo com uma representação adequada na base de conhecimentos do SBC (automatizada).

2.2 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. Ainda nesse sentido, Wang, Ma e Zhou (2009) definem Aprendizado de Máquina (AM) como o estudo da utilização de computadores para simular atividades humanas de aprendizagem e desenvolver métodos auto-incrementais de obtenção de novos conhecimentos e novas habilidades e identificação de conhecimento já existente.

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* é definido por alguns autores de maneira vaga porém com uma interseção entre tais definições: um conceito é uma classe de objetos. De maneira mais detalhada e formal, conceito pode ser entendido como 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.

Devido a uma relativa dificuldade de se definir de maneira exata termos como conceito e aprendizado diante da magnitude dos significados dos mesmos, encontra-se na literatura várias definições e explicações para aprendizado de máquina e a diferença entre eles é devido justamente à ênfase dada em alguns aspectos específicos destes termos.

De acordo com Xue e Zhu (2009), AM é o estudo de como o computador pode realizar e/ou simular o comportamento de aprendizagem do ser humano. O objetivo é obter novos conhecimentos e novas habilidades e organizar a estrutura do conhecimento, que pode ajudar num processo progressivo de aprendizagem. Seguindo esta linha, modelos de computação e de entendimento são criados baseados em pesquisas nas áreas humanas de psicologia e ciência cognigtiva. A Figura 6 traduzida de (XUE; ZHU, 2009) ilustra a estrutura básica de um sistema de aprendizado de máquina.

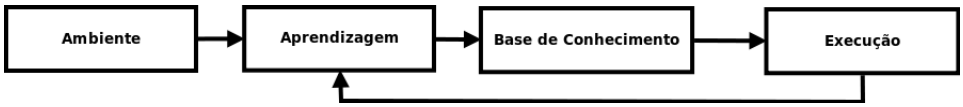


Figura 6: Estrutura básica de um sistema baseado em aprendizado de máquina (extraída de (XUE; ZHU, 2009)).

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, apresentada por Uchôa (1998), 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.

2.2.1 Aprendizado Indutivo de Máquina

Entre os vários modelos existentes para aprendizado apontados anteriormente, o aprendizado simbólico conhecido como aprendizado indutivo baseado em exemplos 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 7, traduzida de (SHAW; GENTRY, 1990), ilustra esse processo.

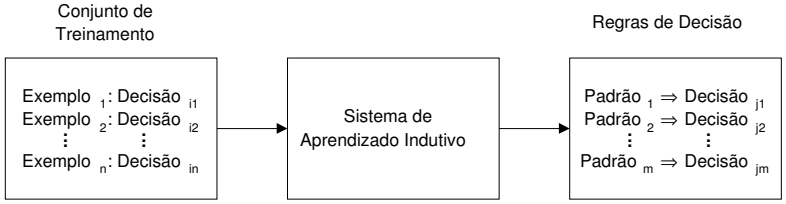


Figura 7: Esquema geral de aprendizado indutivo de regras (Adaptada de (SHAW; GENTRY, 1990))

.

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 consistência.

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. Entre as principais características de sistemas de aprendizado indutivo de máquina apresentadas por Uchôa (1998), encontram-se:

Incrementabilidade: num sistema incremental, a expressão do conceito vai sendo construída exemplo a exemplo e implica constante revisão do conceito (ou conceitos) até então formulado(s). Um novo exemplo pode, eventualmente, causar um rearranjo da expressão do conceito. A expressão do conceito vai se modificando à medida que os exemplos vão se tornando disponíveis. No caso não-incremental, o conjunto de treinamento deve estar totalmente disponível no início do processo de aprendizado, uma vez que o conceito é induzido considerando-se todos os exemplos de uma vez.

Teoria de Domínio: no caso em que o sistema não possui informação a respeito do problema de aprendizado sendo tratado, o sistema induz a expressão do conceito apenas a partir dos exemplos disponíveis. Entretanto, para que a solução de problemas difíceis de aprendizado sejam encontradas, é fundamental que um volume substancial de conhecimento sobre o problema esteja disponível, de maneira a subsidiar a indução do conceito. Esse conhecimento prévio existente é conhecido como Teoria de Domínio ou Conhecimento de Fundo.

Linguagem de Descrição: em inferência indutiva, os exemplos, a Teoria de Domínio e as hipóteses formuladas necessitam ser expressos em alguma linguagem e geralmente são utilizadas linguagens formais.

Uma vez desenvolvido o sistema de AM, é interessante avaliar o resultado deste sistema. De uma forma geral, entre os critérios mais usuais para se medir o sucesso de um sistema de AM, segundo Uchôa (1998) encontram-se:

Precisão de classificação: é geralmente definida como o percentual de exemplos corretamente classificados pela hipótese induzida.

Transparência da descrição induzida do conceito: geralmente é essencial que a descrição do conceito, gerada por um sistema de AM, seja compreensível pelo ser humano. O entendimento do conceito, por parte do ser humano, não apenas aumenta a credibilidade do sistema de AM, mas também permite que o conceito possa ser assimilado e utilizado pelo especialista humano. Em muitas situações a transparência da descrição é medida pelo número de descritores e operadores usados na descrição do conceito.

Complexidade computacional: está relacionada com os recursos computacionais necessários (tempo e espaço) para realizar o aprendizado.

Como já comentado, para a expressão de um modelo de aprendizado indutivo são necessárias linguagens que descrevam os exemplos de treinamento, assim como linguagens que descrevam os conceitos aprendidos. Vários formalismos lógicos têm sido usados em sistemas de aprendizado indutivo para a representação de exemplos e conceitos. Em geral, distinguem-se dois tipos de descrição: *descrição baseada em atributos* (ou proposicional) e *descrição relacional*. Em uma descrição baseada em atributos, exemplos são descritos como vetores de pares atributo-valor e uma classe associada. Em uma descrição relacional, exemplos do conceito a ser aprendido são fornecidos como fatos.

2.2.2 O Algoritmo ID3

O Algoritmo ID3 (*Iterative Dichotomiser* 3) foi proposto por Quinlan (1986) para solucionar problemas de indução, que possuem muitos atributos descrevendo uma quantidade alta de objetos, que necessitam de uma árvore de decisão relativamente boa sem efetuar muita computação para construí-la.

A ideia básica do algoritmo consiste nos seguintes passos:

1. Selecionar um subconjunto W (também chamado de janela), de tamanho V, a partir do conjunto de exemplos de treinamento E;
2. Aplicar o algoritmo para a construção da árvore de decisão T em W;
3. Verificar se existem algum exemplo em E que não pertença a W e que não seja bem classificado pela árvore T, isto é, não se adequa a nenhuma regra que T gera;
4. Se existirem tais exemplos, acrescentá-los em W e voltar ao passo 2; Se não existirem, retornar T.

O algoritmo usado para construir a árvore de decisão no ID3 possui uma heurística de seleção de atributos baseada no ganho de informação durante o processo de construção em si. Ou seja, o algoritmo visa ganhar o máximo possível em informação à medida em que vai adicionando nós na árvore. Além disso, possui a cacterística de construir árvores minimais. Segundo Quinlan (1986), esse método baseado na seleção de atributos possui dois pressupostos. Sendo C um conjunto contendo p objetos da classe P e n objetos da classe N:

- Um objeto qualquer de C vai ser determinado como pertencente à classe P com probabilidade $p/(p+n)$ e à classe N com probabilidade $n/(p+n)$;
- A árvore de decisão pode ser vista, de uma maneira particular, como uma fonte de mensagens P e N para objetos de entrada. A informação necessária para dar essas mensagens é dada por:

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Se um atributo A, com valores $\{A_1, A_2, ..., A_v\}$ é usado como raiz da árvore, ele vai particionar C em $\{C_1, C_2, ..., C_v\}$, onde C_i contém objetos com valores A_i para o atributo A. Suponha que C_i contenha p_i objetos da classe P e n_i objetos da classe N. Desta maneira, a informação necessária para a subárvore C_i é $I(p_i, n_i)$ e a informação necessária para a árvore tendo A como raiz é dada pela média ponderada:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

onde os pesos são as proporções de objetos de C que pertencem à C_i ($p_i + n_i$). O ganho de informação obtido com a escolha de A é, portanto: $gain(A) = I(p,n) - E(A)$. Note que o valor de $I(p,n)$ é constante para todos atributos em C. Assim, maximizar o ganho significa minimizar $E(A)$, que também é chamado de entropia por alguns autores.

O algoritmo de construção da árvore de decisão consiste em ir adicionando nós, representando os atributos, na árvore de acordo com o ganho de informação. No momento em que existir apenas objetos de uma classe, insere-se um nó folha com o nome daquela classe.

Para exemplificar a construção de uma árvore, segue a aplicação do algoritmo ID3 na Tabela 1, que foi adaptada de (QUINLAN, 1986):

Para o conjunto C original, temos os seguintes cálculos:

- $I(y,n) = -\frac{4}{4+8}\log_2\frac{4}{4+8} - \frac{8}{4+8}\log_2\frac{8}{4+8} = 0.918295834054$
- $E(outlook) = 0.784159127851$
- $E(windy) = 0.632802336953$
- $E(temperature) = 0.770426041486$
- $E(humidity) = 0.595437252311$

Tabela 1: Exemplos de treinamento para o algoritmo ID3 contendo dados do clima de algumas manhãs de sábado, onde as classes são y e n.

No.	outlook	windy	temperature	humidity	D
1	overcast	true	cool	high	Y
2	sunny	false	mild	high	N
3	sunny	false	hot	high	N
4	overcast	false	hot	normal	N
5	sunny	true	hot	low	Y
6	rainy	false	mild	high	N
7	rainy	false	hot	high	N
8	rainy	false	hot	normal	N
9	overcast	true	hot	low	Y
10	rainy	false	mild	normal	Y
11	rainy	true	hot	normal	N
12	rainy	false	hot	high	N

Como o valor de $E(humidity)$ foi o menor, o atributo *humidity* é o escolhido para ser a raíz da árvore. Como o atributo *humidity* possui os valores $\{high, normal, low\}$, ele divide C em $\{C_{high}, C_{normal}, C_{low}\}$, onde C_{high} contém objetos com valores *high* para *humidity*, C_{normal} contém objetos com valores *normal* e C_{low} contém objetos com valores *low* para *humidity*.

Percebemos que C_{low} contém apenas objetos pertencentes à classe y. Assim, cria-se um nó folha contendo o valor 'Y'. Como tanto em C_{high} quanto em C_{normal} existem objetos das duas classes, aplica-se recursivamente o algoritmo de construção em C_{high}, C_{normal}, eliminando o atributo *humidity* dos possíveis de ser escolhidos, já que foi utilizado anteriormente.

Para C_{high} temos os seguinte cálculos:

- $I(y,n) = -\frac{1}{1+5}\log_2\frac{1}{1+5} - \frac{5}{1+5}\log_2\frac{5}{1+5} = 0.650022421648$
- $E(outlook) = 0$
- $E(windy) = 0$
- $E(temperature) = 0$

Com isso, percebemos que todos os atributos aumentam o ganho de maneira igual. Assim, pode-se escolher qualquer um deles. Normalmente escolhe-se o primeiro, mas há algumas abordagens que determinam escolher aquele que tem menos valores, para simplificar a árvore de decisão. Nesse caso, escolheu-se o que tem menos valores: *windy*.

O atributo *windy* possui os valores $\{true, false\}$ e particiona C_{high} em $\{C_{high,true}, C_{high,false}\}$. O valor de $E(windy)$ sendo 0, significa que o atributo divide bem os objetos, agrupando-os sem que objetos de classes diferentes fiquem na mesma partição. Pode-se confirmar isso observando os objetos das partições: $C_{high,true}$ só possui objetos da classe *Y* e $C_{high,false}$ só possui objetos da classe *N*.

Para C_{normal} temos os seguinte cálculos:

- $I(y,n) = -\frac{1}{1+3}\log_2\frac{1}{1+3} - \frac{3}{1+3}\log_2\frac{3}{1+3} = 0.811278124459$
- $E(outlook) = 0.688721875541$
- $E(windy) = 0.688721875541$
- $E(temperature) = 0$

Pode-se notar que o atributo *temperature* deve ser escolhido pois proporciona um maior ganho de informação por possuir menor entropia que os outros.

O atributo *temperature*, em C_{normal} possui os valores $\{mild, hot\}$ e particiona C_{normal} em $\{C_{normal,mild}, C_{normal,hot}\}$. O valor de $E(temperature)$ sendo 0, significa que o atributo divide bem os objetos, agrupando-os sem que objetos de classes diferentes fiquem na mesma partição. Pode-se confirmar isso observando os objetos das partições: $C_{normal,mild}$ só possui objetos da classe *Y* e $C_{normal,sunny}$ só possui objetos da classe *N*. Nesinformação se momento verifica-se que não há mais computações a serem feitas, terminando o algoritmo. Depois dos passos acima, a árvore de decisão mostrada na Figura 8 é construída.

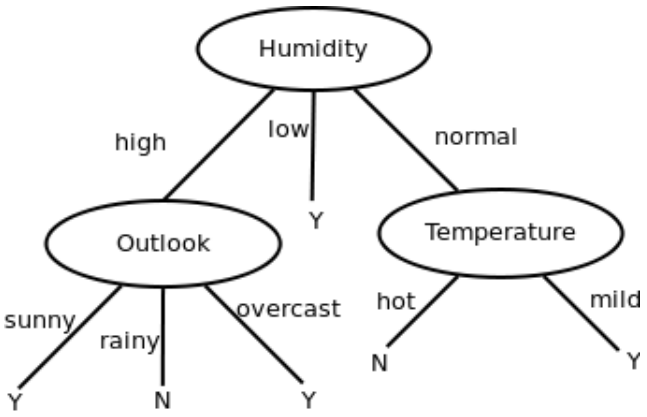


Figura 8: Árvore de decisão construída pelo algoritmo ID3

.

2.2.3 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.

Como apontado em Bonissone (1991) citado por Uchôa (1998),

> 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.

A incerteza num domínio de aplicação pode estar presente nos dados de entrada, na solução do problema ou em ambos. Por exemplo, no caso de diagnóstico médico os sinais e sintomas que o médico coleta e trabalha apresentam vários problemas de incerteza, tais como: a inexatidão dos relatos do paciente e a percepção da intensidade de cada sintoma, entre outros. Além disso, o próprio raciocínio clínico não pode ser realizado com certeza, isto é podem existir dois pacientes com

dois conjuntos similares de sinais e sintomas e seus diagnósticos serem diferentes. Torna-se então necessário formas de representação de incerteza na base de conhecimento e no raciocínio de sistemas especialistas aplicados a domínios com a presença de incerteza.

Entre as abordagens mais utilizadas para a representação e tratamento de incertezas, temos:

1. **Fatores de Certeza:** O sistema especialista associa a cada uma de suas informações um determinado grau de certeza. Este método surgiu com o sistema MYCIN (SHORTLIFFE; BUCHANAN, 1975) e foi muito utilizado no desenvolvimento dos primeiros sistemas especialistas.

2. **Teoria dos Conjuntos Difusos (*Fuzzy Set*):** representa a incerteza por imprecisão, isto é trabalha com conjuntos com limites imprecisos. Muito utilizada para representar termos lingüisticamente imprecisos, como "homem gordo", por exemplo. Na lógica clássica, com base na Teoria Clássica dos Conjuntos, um elemento pertence ou não ao conjunto; enquanto, na lógica *fuzzy* (ZADEH, 1965), abordada em 4.1, um elemento possui um grau de pertinência ao conjunto, que varia de 0 a 1, este grau é obtido por meio da função de pertinência que representa o conjunto no domínio de aplicação.

3. **Teoria de Evidência:** a teoria de Dempster-Schafer (SHAFER, 1976), como é conhecida, trata a representação de incertezas com medidas de crença, que são obtidas por meio de funções de crença. Estas funções tornam-se úteis quando agregadas pela regra de combinação de Dempster de modo a gerar uma única função de crença.

A Teoria de Conjuntos Aproximados,, formalismo explorado neste trabalho, foi proposta por Pawlak (1982) como sendo um modelo matemático para representação do conhecimento, tratamento de incerteza e classificação aproximada.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.

3 Elementos da Teoria de Conjuntos Aproximados e suas aplicações em Sistemas de Representação de Conhecimento

3.1 Introdução

Um conjunto aproximado é um modelo matemático usado para tratar um tipo de incerteza - *a indiscernibilidade*. De uma forma bem simples, conjuntos aproximados podem ser considerados conjuntos com fronteiras nebulosas, ou seja, conjuntos que não podem ser caracterizados precisamente utilizando-se o conjunto de atributos disponíveis. Uma das principais vantagens da Teoria de Conjuntos Aproximados (TCA) é a de não necessitar de qualquer informação adicional ou preliminar a respeito de dados, tais como: distribuição de probabilidade, atribuição de crenças, grau de pertinência ou possibilidade. Este capítulo foi baseado nas teorias propostas em (PAWLAK, 1991) e na notação utilizada em (UCHÔA, 1998) para representar tais conceitos, já que são econtradas várias notações na literatura sem padrão algum.

3.2 Espaços Aproximados

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 sobre U, denominada *relação de indiscernibilidade*. Dados $x, y \in U$, se xRy então x e y são *indiscerníveis* em A, ou seja, a classe de equivalência definida por x é a mesma que a definida por y, i.e., $[x]_R = [y]_R$.

As classes de equivalência induzidas por R em U são denominadas *conjuntos elementares*. Se E é um conjunto elementar, $des(E)$ denota a descrição dessa classe de equivalência. Essa descrição é função do conjunto de atributos que define

R. Note que, dados $x, y \in E$, onde E é um conjunto elementar em A, x e y são indiscerníveis, i.e., no espaço $A = (U, R)$ não se consegue distinguir x de y, pois $des(x) = des(y) = des(E)$.

3.3 Classificação Aproximada de um Conjunto

Dado um espaço aproximado $A = (U, R)$ e um conjunto $X \subseteq U$, com o objetivo de verificar o quão bem X é representado pelos conjuntos elementares de A, são definidas:

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

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

Nas notações utilizadas, quando o espaço aproximado for conhecido e não houver risco de confusão, a referência ao espaço será abolida. Assim, por exemplo, $A_{sup}(X)$ será usado em substituição a $A_{A-sup}(X)$.

3.4 Regiões do Espaço Aproximado

Dado um espaço aproximado $A = (U, R)$ e $X \subseteq U$, as aproximações inferior e superior permitem a classificação do espaço aproximado em regiões:

1. *região positiva* de X em A, formada por todas as classes de equivalência de U contidas inteiramente no conjunto X e dada por,

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

2. *região negativa* de X em A, formada pelos conjuntos elementares de A que não estão contidos na aproximação superior de X e dada por,

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

3. *região duvidosa* de X em A, formada pelos elementos que pertencem a aproximação superior mas não pertencem à aproximação inferior e dada por,

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

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 9. A aproximação inferior e a aproximação superior de X em $A = (U,R)$ são mostradas na Figura 10(a) e 10(b). A Figura 11, por sua vez, apresenta as regiões de X.*

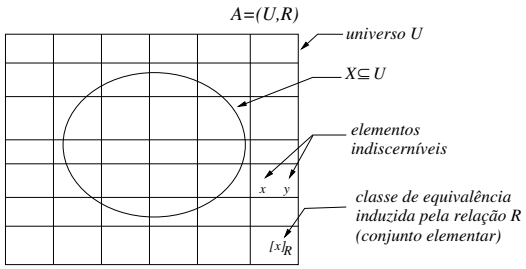
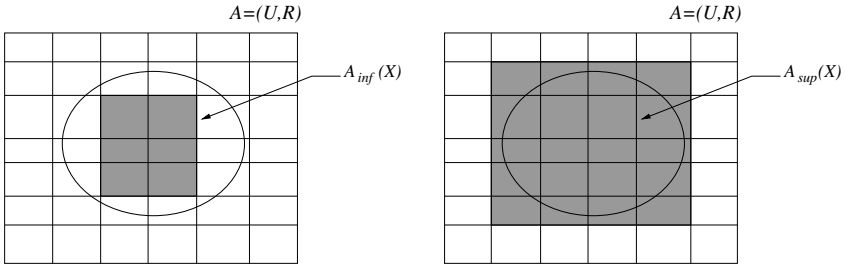


Figura 9: Conjunto X no espaço aproximado $A = (U,R)$ (Extraída de (UCHÔA, 1998)).



(a) Aproximação Inferior de X. (b) Aproximação Superior de X.

Figura 10: Aproximações de X em A (Extraída de (UCHÔA, 1998)).

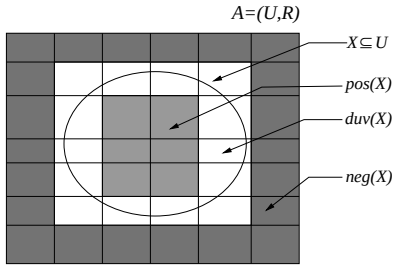


Figura 11: Regiões de X em A (Extraída de (UCHÔA, 1998)).

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*[1] 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.

[1] *rough set*, na língua inglesa

3.5 Igualdade Aproximada entre Conjuntos

Dado um espaço aproximado $A = (U,R)$, o conjunto $X \subseteq U$ pode ou não ter sua fronteira claramente definida em função das descrições dos conjuntos elementares de A; isso leva ao conceito de *conjuntos aproximados*.

Seja $A = (U,R)$ um espaço aproximado e seja X um conjunto clássico. O conjunto aproximado X' é uma aproximação de X, através dos dois subconjuntos de U/R, $A_{A-inf}(X)$ e $A_{A-sup}(X)$, que se "aproximam" o "máximo possível" (tanto quanto permite a partição) de X, "por dentro" e "por fora" respectivamente. O conjunto aproximado de X pode, portanto, ser definido pelo par $X' = < A_{A-inf}, A_{A-sup} >$.

As considerações anteriores levam ao conceito de igualdade aproximada: dois ou mais conjuntos são aproximadamente iguais se e somente se possuem a mesma região positiva, negativa e duvidosa, i.e., definem o mesmo conjunto aproximado. Formalmente, num espaço aproximado $A = (U,R)$, com $X \subseteq U$, o conceito de igualdade aproximada é abordado da seguinte forma:

1. X é aproximadamente inf-igual a Y, $X =_{inf} Y$, se e somente se $A_{A-inf}(X) = A_{A-inf}(y)$.
2. X é aproximadamente sup-igual a Y, $X =_{sup} Y$, se e somente se $A_{A-sup}(X) = A_{A-sup}(y)$.
3. X é aproximadamente igual a Y, $X \approx Y$, se e somente se $A_{A-inf}(X) = A_{A-inf}(y)$ e $A_{A-sup}(X) = A_{A-sup}(y)$, ou seja, se $X =_{inf} Y$ e $X =_{sup} Y$.

Exemplo 2 *Todos os conjuntos mostrados na Figura 12 são aproximadamente iguais. Nesse espaço, qualquer um representa o conjunto orignial X.*

3.6 Inclusão Aproximada de Conjuntos

Na seção anterior, o conceito de igualdade entre conjuntos foi estendido a conjuntos aproximados. É desejável, portanto, que outros conceitos relativos a

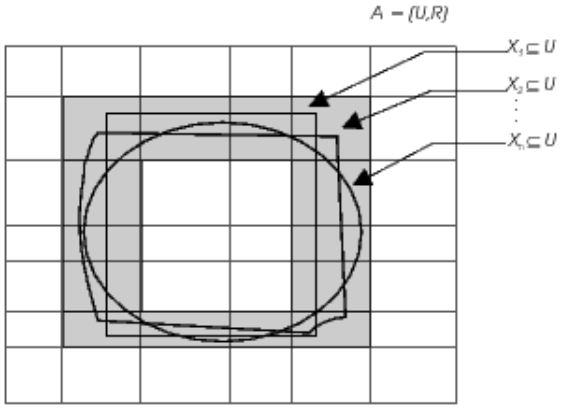


Figura 12: A família de conjuntos $\{X_1, X_2, ..., X_n\}$, todos tendo a mesma aproximação inferior e a mesma aproximação superior, define um conjunto X no espaço aproximado $A = (U, R)$ (Extraída de (UCHÔA, 1998)).

conjuntos sejam também estendidos para que possam ser utilizados em espaços aproximados, como é o caso da inclusão aproximada de conjuntos. Dados o espaço aproximado $A = (U, R)$ e $X, Y \subseteq U$, define-se que:

1. X é aproximadamente inf-incluído em Y, $X \subset_{inf} Y$, se e somente se $A_{A-inf}(X) \subseteq A_{A-inf}(y)$.
2. X é aproximadamente sup-incluído em Y, $X \subset_{sup} Y$, se e somente se $A_{A-inf}(X) \subseteq A_{A-inf}(y)$.
3. X é aproximadamente incluído em Y, $X \subset_{\sim} Y$, se e somente se $A_{A-inf}(X) \subseteq A_{A-inf}(y)$ e $A_{A-sup}(X) \subseteq A_{A-sup}(y)$, ou seja, se $X \subset_{inf} Y$ e $X \subset_{sup} Y$.

3.7 Medidas em um Espaço Aproximado

Dado um espaço aproximado $A = (U, R)$ e $X \subseteq U$, pode-se definir medidas que indiquem o quão bem $X \subseteq U$ pode ser representado pelo espaço aproximado $A = (U, R)$.

3.7.1 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:

- *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, $\gamma_{A-sup}(X) = \frac{\omega_{A-sup}(X)}{|U|}$.

A medida de *acuracidade* de X é definida pela relação entre a cardinalidade da aproximação inferior de X pela cardinalidade 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 A é conhecido e não há risco de confusão, escreve-se 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.

Exemplo 3 *Seja $A = (U,R)$ um espaço aproximado e seja $X \subseteq U$ conforme mostra a Figura 13. Então é possível determinar a acuracidade de X:*

$$\omega_A(X) = \frac{\omega A_{inf^A}(X)}{\omega A_{sup^A}(X)} = \frac{|A_{inf}(X)|}{|A_{sup}(X)|} = \frac{41}{117}.$$

3.7.2 Índice Discriminante

Seja o espaço aproximado $A = (U,R)$ e $X \subseteq U$. O número total de objetos em U que podem, com certeza, ser classificados em dois subconjuntos disjuntos, X e $U - X$, é igual ao número de objetos que não pertencem à região $duv(X)$. Esse

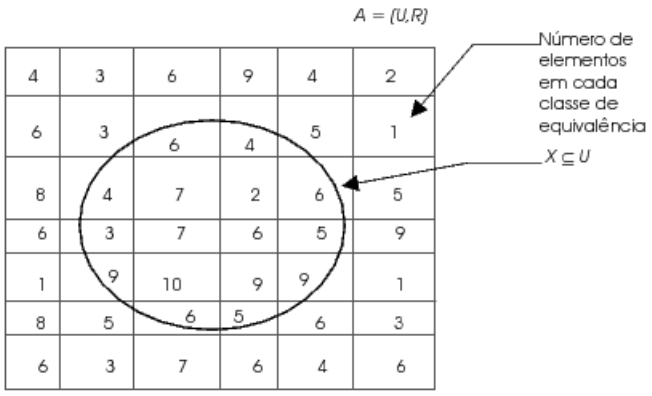


Figura 13: Espaço aproximado $A = (U,R)$ e conjunto X. Em cada partição está o número de elementos contidos nela (Extraída de (UCHÔA, 1998)).

número é, pois:

$$|U - duv(X)| = |U - (A_{sup}(X) - A_{inf}(X))| = |U| - |A_{sup}(X) - A_{inf}(X)|.$$

Dessa forma, a razão:

$$\alpha_R(X) = \frac{|U| - |A_{sup}(X) - A_{inf}(X)|}{|U|},$$

definida com *índice discriminante de A com relação a X*, fornece uma medida do grau de certeza na determinação da pertinência (ou não) de um elemento de U no conjunto X.

3.8 Representação de Conhecimento

3.8.1 Sistemas de Representação de Conhecimento (SRCs)

Os conceitos da TCA são utilizados principalmente no contexto de Sistemas de Representação de Conhecimento. Um *Sistema de Representação de Conhecimento* (SRC) é uma quádrupla $S = (U,Q,V,\rho)$, onde U é o universo finito de S.

Os elementos de U são chamados objetos, que são caracterizados por um conjunto de atributos Q e seus respectivos valores. O conjunto de valores de atributos é dado por $V = \bigcup_{q \in Q} V_q$, onde V_q é o conjunto de valores do atributo q. Por sua vez, $\rho : U \times Q \to V$ é uma *função de descrição* tal que $\rho(x,q) \in V_q$, para $x \in U$ e $q \in Q$.

Dado um SRC $S = (U, Q, V, \rho)$, é importante observar que cada subconjunto de atributos $P \subseteq Q$ define um único espaço aproximado $A = (U, \tilde{P})$ onde $\tilde{P}$ é a relação de indiscernibilidade (equivalência) induzida por P.

Exemplo 4 *Seja o SRC representado pela Tabela 2, onde* $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$, $Q = \{a, b, c, d\}$ *e* $V = \{1, 2, 3, 4, 5, 6\}$. *Neste sistema de representação de conhecimento tem-se:*

- $\rho(x_1, a) = 4$;
- $\rho(x_2, b) = 1$;
- $\rho(x_5, d) = 1$;
- $\rho(x_3, c) = 5$.

Tabela 2: SRC onde $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$ e $Q = \{a, b, c, d\}$

U	a	b	c	d
x_1	4	1	3	1
x_2	6	1	4	1
x_3	6	2	5	2
x_4	6	1	5	1
x_5	4	2	4	1
x_6	4	2	4	2
x_7	4	2	3	1
x_8	4	1	4	1
x_9	6	1	4	2
x_{10}	4	1	4	2

Exemplo 5 *Seja S o SRC do Exemplo 4 e* $P = \{d\}$. *Neste caso tem-se que os elementos* x_3, x_6, x_9 *e* x_{10} *são indiscerníveis com relação a P pois possuem o*

mesmo valor. Além disto, $A = (U, \tilde{P})$ é um espaço aproximado e seus conjuntos elementares são $E_1 = \{x_1, x_2, x_4, x_5, x_7, x_8\}$, $E_2 = \{x_3, x_6, x_9, x_{10}\}$.

O Exemplo 6 demonstra o cálculo dos principais conceitos e medidas da TCA.

Exemplo 6 *Seja S o SRC definido no Exemplo 4 e seja $P = \{a, b\}$, um subconjunto de atributos de Q. Nesse caso, os conjuntos elementares do espaco aproximado $A = (U, \tilde{P})$ são: $E_1 = \{x_1, x_8, x_{10}\}$, $E_2 = \{x_2, x_4, x_9\}$, $E_3 = \{x_3\}$ e $E_4 = \{x_5, x_6, x_7\}$. Seja $X = \{x_2, x_3\}$. Tem-se que:*

$$A_{A-inf}(X) = \{x_3\}$$
$$A_{A-sup}(X) = \{x_2, x_3, x_4, x_9\}$$

$$\alpha_P(X) = \frac{10-|4-1|}{10} = \frac{7}{10} = 0.7$$

$$\omega_{A-inf}(X) = 1$$
$$\omega_{A-sup}(X) = 4$$
$$\omega_A(X) = 0.25$$

$$\gamma_{A-inf}(X) = 0.1$$
$$\gamma_{A-sup}(X) = 0.4$$

3.8.2 Tabelas de Decisão

No contexto da TCA o interesse recai, principalmente, sobre tabelas de decisão, um tipo particular de SRC. Uma *tabela de decisão* é um SRC onde os atributos de Q são divididos em condições e decisões. Tem-se então $Q = C \cup D$, onde C é o conjunto das condições e D o conjunto das decisões. Como geralmente o conjunto D é unitário, uma tabela de decisão é descrita por $S = (U, C \cup \{\delta\}, V, \rho)$, onde U, V e ρ são tais como num SRC, C é o conjunto de condições e δ é o atributo de decisão. Por $Class_S(\delta)$ entende-se a classificação de S, i.e., a família de conjuntos elementares do espaço aproximado induzido por $\{\delta\}$.

Exemplo 7 *Seja S o SRC do Exemplo 4. Seja d o atributo de decisão em S, i.e.,* $\delta = d$. *S é uma tabela de decisão, onde* $C = \{a,b,c\}$. *Os conjuntos elementares do espaço aproximado induzido por C são* $\{x_1\}$, $\{x_2,x_9\}$, $\{x_3\}$, $\{x_4\}$, $\{x_5,x_6\}$, $\{x_7\}$ *e* $\{x_8,x_{10}\}$. *Por sua vez, a classificação de S é dada por* $Class_S(\delta) = \{\{x_1,x_2,x_4,x_5,x_7,x_8\},\{x_3,x_6,x_9,x_{10}\}\}$.

Dada uma tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho)$, é importante verificar o quão bem a família de conjuntos elementares induzidos pelas condições $P \subseteq C$ espelha a família de conjuntos elementares induzidos por $\{\delta\}$. Para isso, considerando o espaço aproximado induzido por P, são definidas:

- *região positiva* de δ induzida por P,

$$pos(P,\delta) = \bigcup_{X \in Class_S(\delta)} A_{P-inf}(X);$$

- *grau de dependência* de δ com relação a P,

$$\kappa(P,\delta) = \frac{|pos(P,\delta)|}{|U|};$$

- *fator de significância* de um atributo $a \in P$, com relação à dependência existente entre δ e P,

$$FS(a,P,\delta) = \frac{(\kappa(P,\delta) - (\kappa(P-\{a\},\delta))}{\kappa(P,\delta)},$$

  se $\kappa(P,\delta) > 0$.

Exemplo 8 *Seja S o SRC definido no Exemplo 4 e seja* $P = \{a,b\}$. *Os conjuntos elementares do espaço aproximado induzido por P são:*
$E_1 = \{x_1,x_8,x_{10}\}$, $E_2 = \{x_2,x_4,x_9\}$, $E_3 = \{x_3\}$ *e* $E_4 = \{x_5,x_6,x_7\}$.
Por sua vez, $Class_S(d) = \{\{x_1,x_2,x_4,x_5,x_7,x_8\},\{x_3,x_6,x_9,x_{10}\}\}$.
Com isso, é possível verificar os seguintes resultados:

$$pos(P,d) = \bigcup_{X \in Class_S(d)} A_{P-inf}(X) =$$

$$= A_{P-inf}(\{x_1,x_2,x_4,x_5,x_7,x_8\}) \cup A_{P-inf}(\{x_3,x_6,x_9,x_{10}\}) =$$

$$= \{x_1,x_4,x_7\} \cup \{x_3\} = \{x_1,x_3,x_4,x_7\}$$

$$\kappa(P,d) = \frac{|pos(P,d)|}{|U|} = \frac{4}{10} = 0.4.$$

Para o cálculo de:

$$FS(a,P,d) = \frac{(\kappa(P,d)-(\kappa(P-\{a\},d))}{\kappa(P,d)},$$

e de :

$$FS(b,P,d) = \frac{(\kappa(P,d)-(\kappa(P-\{b\},d))}{\kappa(P,d)},$$

bem como o de :

$$FS(c,P,d) = \frac{(\kappa(P,d)-(\kappa(P-\{c\},d))}{\kappa(P,d)},$$

é necessário calcular antes o valor de $\kappa(\{b,c\},d)$*,* $\kappa(\{a,c\},d)$ *e* $\kappa(\{a,b\},d)$*, respectivamente, para este exemplo, considerando-se que* $P_1 = \{b,c\}$*,* $P_2 = \{a,c\}$ *e* $P_3 = \{a,b\}$*, tem-se que as famílias dos conjuntos elementares dos espaços aproximados induzidos por* P_1*,* P_2 *e* P_3 *são dadas, respectivamente, por* $\{\{x_1\},\{x_2,x_8,x_9,x_{10}\},\{x_5,x_6\},\{x_7\},\{x_3\},\{x_4\}\}$*,* $\{\{x_1,x_7\},\{x_5,x_6,x_8,x_{10}\},\{x_3,x_4\},\{x_2,x_9\}\}$ *e* $\{\{x_1,x_8,x_{10}\},\{x_2,x_4,x_9\},\{x_3\},\{x_5,x_6,x_7\}\}$*. Donde:*

$$pos(P_1,d) = \{x_1,x_3,x_4,x_7\} \text{ e}$$
$$\kappa(P_1,d) = \tfrac{4}{10} = 0.4,$$

$$pos(P_2,d) = \{x_1,x_7\} \text{ e}$$
$$\kappa(P_2,d) = \tfrac{2}{10} = 0.2,$$

$$pos(P_3,d) = \{x_3\} \text{ e}$$
$$\kappa(P_3,d) = \tfrac{1}{10} = 0.1.$$

Portanto,

$$FS(a,P,d) = \frac{(\kappa(P,d)-(\kappa(\{b,c\},d))}{\kappa(P,d)} = \frac{\frac{4}{10}-\frac{4}{10}}{\frac{4}{10}} = 0,$$

$$FS(b,P,d) = \frac{(\kappa(P,d)-(\kappa(\{a,c\},d))}{\kappa(P,d)} == \frac{\frac{4}{10}-\frac{2}{10}}{\frac{4}{10}} = 0.5,$$

$$FS(c,P,d) = \frac{(\kappa(P,d)-(\kappa(\{a,b\},d))}{\kappa(P,d)} == \frac{\frac{4}{10}-\frac{1}{10}}{\frac{4}{10}} = \frac{3}{4} = 0.75.$$

Diz-se, ainda com respeito a uma tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho\}$ e $P \subseteq C$ que P é *independente* com relação à dependência existente entre δ e P se, para todo subconjunto próprio $R \subset P$, for verdade que $pos(P,\delta) \neq pos(R,\delta)$, i.e., $\kappa(P,\delta) \neq \kappa(R,\delta)$. Caso haja algum $R \subset P$ tal que $pos(P,\delta) = pos(R,\delta)$, i.e., $\kappa(P,\delta) = \kappa(R,\delta)$, então P é dito ser *dependente* com relação à dependência existente entre δ e P.

3.8.3 Redução do Conjunto Inicial de Atributos

Um problema crucial no contexto de Sistemas de Representação de Conhecimento é o de encontrar subconjuntos do conjunto original de atributos com o mesmo poder discriminatório deste. A obtenção destes subconjuntos pode auxiliar tanto na redução de custo computacional em tarefas que utilizem-se de SRCs, como até mesmo custo temporal ou financeiro (caso atributos de difícil obtenção ou alto custo possam ser eliminados). No contexto da TCA, a obtenção de redutos está intimamente ligada à anàlise de dependência entre atributos.

Um conjunto $R \subset P$ é dito ser um *reduto* de P com relação à dependência existente entre δ e P se for independente com relação à dependência existente entre δ e R, e $pos(P,\delta) = pos(R,\delta)$, i.e., $\kappa(P,\delta) = \kappa(R,\delta)$.

Exemplo 9 *Seja S o SRC do Exemplo 4, onde d é o atributo de decisão em S. Como já visto, S e uma tabela de decisão, onde a classe de condições é dada por* $C = \{a,b,c\}$ *e a decisão é dada por* $\{\delta\} = \{d\}$. *A família* $Class_S(d)$ *foi*

determinada no Exemplo 7. Tem-se os seguintes resultados em S:

$$\kappa(C,d) = \frac{|pos(C,d)|}{|U|} = \frac{|4|}{|10|} = 0.4$$

$$\kappa(\{a,b\},d) = \frac{|1|}{|10|} = 0.1$$

$$\kappa(\{a,c\},d) = \frac{|2|}{|10|} = 0.2$$

$$\kappa(\{b,c\},d) = \frac{|4|}{|10|} = 0.4$$

Neste caso C é dependente e $\{b,c\}$ *é o único subconjunto de C a possuir um reduto de C, pois* $\{b,c\}$ *é o único subconjunto com o mesmo grau de dependência de C. Tem-se ainda:*

$$\kappa(\{b\},d) = \frac{|0|}{|10|} = 0$$

$$\kappa(\{c\},d) = \frac{|2|}{|10|} = 0.2$$

Portanto, o conjunto $R = \{b,c\}$ *é o único reduto do conjunto de condições com relação a dependência entre* δ *e C.*

4 Teoria dos Conjuntos *Fuzzy* e Funções de Pertinência Aproximada

4.1 Conjuntos *Fuzzy*

Sejam U o conjunto universo e $A \subseteq U$. O conjunto A pode ser expresso de três formas distintas:

1. por seus elementos, por exemplo $A = \{1,2,3,4,5\}$
2. por uma propriedade, por exemplo $A = \{x | x \text{ é par}\}$
3. por uma função característica $\mu_A(x) : U \to \{0,1\}$, definida como:

$$\mu_A(x) = \begin{cases} 1, & \text{se } x \in A \\ 0, & \text{se } x \notin A \end{cases}$$

Essa função descreve o conjunto A atribui o valor 0 ou 1 a cada elemento do conjunto universo U, discriminando, com essa atribuição, os elementos que são de A daqueles que não são de A; é uma maneira alternativa de descrever A. O conjuntos $\{0,1\}$ é chamado conjunto de avaliação.

Essa função pode ser generalizada de maneira que os valores atribuídos aos elementos do conjunto universo U pertençam a um intervalo específico de valores e indiquem o grau de pertinência daqueles elementos ao conjunto em questão. Valores maiores indicam um maior grau de pertinência ao conjunto. Essa função é chamada função de pertinência e o conjunto definido por ela de conjunto *fuzzy* (KLIR; YUAN, 1995). O intervalo de valor mais comumente usado na literatura é o [0,1]. Neste caso, cada função de pertinência associa elementos de um dado conjunto universo U a valores no intervalo [0,1]. Dado então um conjunto universo U, a função de pertinência que caracteriza um conjunto *fuzzy* A é notada por:

$$\mu_X(A) : U \to [0,1]$$

Exemplo 10 *A Figura 14 ilustra conjuntos fuzzy representando os conceitos de pequeno, médio e grande para um dado objeto. No caso foram utilizadas as funções* $\mu_p(x) = e^{-2.5(x-7.5)^2}$, $\mu_m(x) = e^{-3.5(x-11.5)^2}$ *e* $\mu_g(x) = e^{-2.5(x-15.5)^2}$, *onde* μ_p, μ_p *e* μ_p *representam, respectivamente, as funções de pertinência aos conjuntos pequeno, médio e grande.*

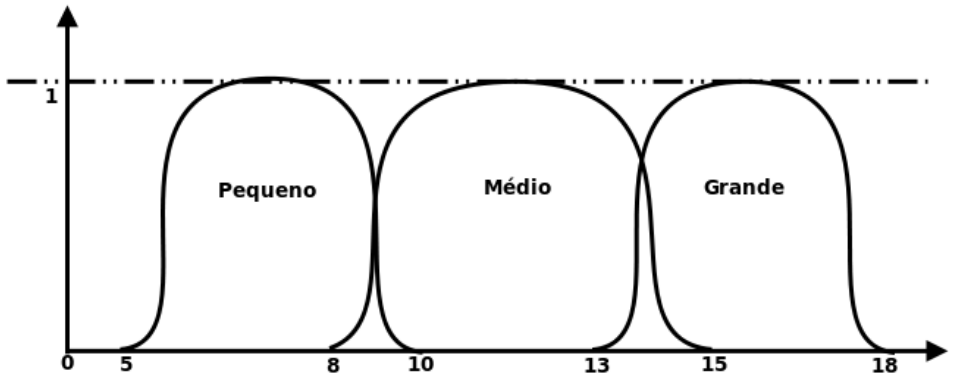


Figura 14: Exemplos de conjuntos *fuzzy*.

Dados X e Y conjuntos *fuzzy* em U, podem ser definidas as seguintes operações:

1. complemento:
$$\mu_{\bar{X}}^{A}(x) = 1 - \mu_X(x)$$

2. união:
$$\mu_{X\cup Y}^{A}(x) = max[\mu_X(x), \mu_Y(x)]$$

3. interseção:
$$\mu_{X\cap Y}^{A}(x) = min[\mu_X(x), \mu_Y(x)]$$

O termo *fuzzy* pode ser traduzido para a língua portuguesa como “nebuloso”. Desta maneira, os conjuntos *fuzzy* modelam conceitos “nebulosos”, vagos, associando, através de uma função, elementos de um conjunto (conceito) a valores

pertencentes ao intervalo $[0,1]$, valores estes que representam o grau com que o elemento pertence ao conceito em questão.

4.2 Funções de Pertinência Associadas a Conjuntos Aproximados

Considere um conceito cuja fronteira não esteja precisamente definida, isto é, um conceito vago. Devem existir instâncias, em torno de sua fronteira, que não podem ser classificadas, com base na informação existente, nem como instâncias do conceito e nem como instâncias do seu complemento com relação ao universo, ou seja, determinados elementos do universo não podem ser classificados com certeza, como elementos do conjunto.

Como na TCA a única maneira de "ver" os elementos de um conjunto é através da informação existente sobre esses elementos, é perfeitamente plausível que a informação disponível sobre eles não seja suficiente para discriminá-los. Assim sendo, sob a ótica da informação existente, elementos podem ser considerados idênticos. A incerteza na TCA está relacionada à questão da pertinência (ou não) de elementos a conceitos, como função das informações disponíveis sobre eles.

Para a abordagem da incerteza sob a perspectiva de conjuntos aproximados, é fundamental que se defina uma função de pertinência relacionada ao conceito de conjuntos aproximados nos moldes da função de pertinência *fuzzy*. Essa função é chamada de função de pertinência aproximada[2]. A pesquisa dessa função de pertinência aproximada justifica-se, entre outros, pela suposição de que essa função poderia ser utilizada na construção de conjuntos *fuzzy*, através da atribuição de grau de pertinência aos seus elementos. Desta forma, dada uma função de pertinência aproximada, é desejável que ela possua algumas características semelhantes àquelas de funções de pertinência *fuzzy*.

[2] *rough membership function*, na língua inglesa

4.2.1 Função de Pertinência Aproximada: Primeira Proposta

A primeira proposta de uma função de pertinência aproximada encontra-se descrita em Pawlak (1985). Dados um espaço aproximado $A = (U, R)$ e $X \subseteq U$ um conjunto aproximado em A, X pode ser expresso através de uma função de pertinência definida em U, assumindo valores em $\{0, 0.5, 1\}$, da seguinte maneira:

$$\mu_X^A(x) = \begin{cases} 1, & \text{se e somente se } x \in pos(X) \\ 0.5, & \text{se e somente se } x \in duv(X) \\ 0, & \text{se e somente se } x \in neg(X) \end{cases}$$

ou, equivalentemente,

$$\mu_X^A(x) = \begin{cases} 1, & \text{se e somente se } x \in A_{inf}(X) \\ 0.5, & \text{se e somente se } x \in (A_{sup}(X) - A_{inf}(X)) \\ 0, & \text{se e somente se } x \in (-A_{sup}(X)) \end{cases}$$

Esta proposta de "tradução" dos principais conceitos associados a um conjunto aproximado, i.e., sua aproximação inferior, sua aproximação superior e sua fronteira, para uma função de pertinência com valores no conjunto $\{0, 0.5, 1\}$ não reflete, como deveria, a situação de elementos pertencentes à fronteira do conjunto. Dependendo da informação existente, da granularidade da partição induzida por R em U, bem como da própria expressão de X, em função das classes de equivalência induzidas por R, o grau de pertinência de um elemento de X à fronteira, quando esta não for vazia, pode variar entre quase 0 e quase 1. *Defini-lo como* 0.5 *é assumir o risco de estar, deliberadamente, introduzindo mais incerteza* (NICOLETTI; UCHÔA, 1997).

Além desta função de pertinência aproximada não refletir precisamente os conceitos da TCA, pode ser facilmente verificado que ela não satisfaz as operações de união e intersecção para funções de pertinência *fuzzy*, que são definidas como:

$$\mu_{X \cup Y}^A(x) = max[\mu_X^A(x), \mu_Y^A(x)]$$

$$\mu^A_{X\cap Y}(x) = min[\mu^A_X(x), \mu^A_Y(x)]$$

Satisfaz, entretanto, a operação de complemento que é definida por:

$$\mu^A_{\bar{X}}(x) = 1 - \mu^A_X(x)$$

As figuras 15 e 16 mostram regiões para as quais as operações de união e intersecção, respectivamente, não se verificam.

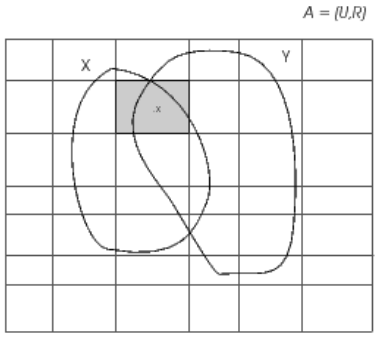


Figura 15: Para qualquer ponto x da área sombreada, $\mu^A_{X\cup Y}(x) \neq max[\mu_X(x), \mu_Y(x)]$, uma vez que $\mu^A_{X\cup Y}(x) = 1$ e $max[\mu^A_X(x), \mu^A_Y(x)] = 0.5$ (Extraída de (UCHÔA, 1998))

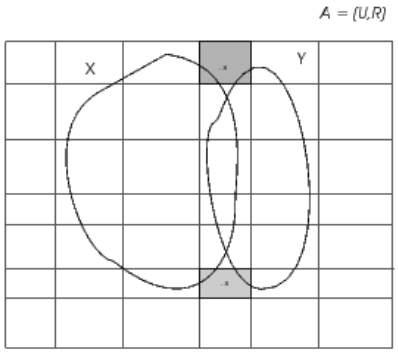


Figura 16: Para qualquer ponto x da área sombreada, $\mu^A_{X\cap Y}(x) \neq min[\mu^A_X(x), \mu^A_Y(x)]$, uma vez que $\mu^A_{X\cap Y}(x) = 0$ e $min[\mu^A_X(x), \mu^A_Y(x)] = 0.5$ (Extraída de (UCHÔA, 1998))

Se, entretanto, tais operações forem redefinidas, tendo em mente que as situações mostradas nas figuras 15 e 16 devam ser tratadas à parte, a função de pertinência aproximada pode ser estendida à união e à intersecção de conjuntos. Em Wygralak (1989) a redefinição das operações de união e intersecção é proposta como:

$$\mu^A_{X\cup Y}(x) = \begin{cases} min\left[1, \mu^A_X(x) + \mu^A_Y(x)\right], \\ \qquad \text{se } \mu^A_X(x) = \mu^A_Y(x) = 0.5 \text{ e } [x]_R \subseteq X \cup Y \\ max\left[\mu^A_X(x), \mu^A_Y(x)\right], \text{caso contrário} \end{cases}$$

$$\mu^A_{X\cap Y}(x) = \begin{cases} max\left[0, \mu^A_X(x) + \mu^A_Y(x) - 1\right], \\ \qquad \text{se } \mu^A_X(x) = \mu^A_Y(x) = 0.5 \text{ e } [x]_R \cap (X \cap Y) = \varnothing \\ min\left[\mu^A_X(x), \mu^A_Y(x)\right], \text{caso contrário} \end{cases}$$

Note que esta proposta mantém as definições das operações de união e intersecção anteriormente apresentadas e acrescenta a elas um "tratamento" extra para as situações de exceção exibidas nas figuras 15 e 16. Note também que essa redefinição continua não exprimindo com exatidão a situação de elementos pertencentes à fronteira do conjunto. É óbvio que essa definição foi proposta com o intuito único de "acomodar" as exceções mostradas na 15 e na 16 e "fazer valer" as propriedades.

4.2.2 Função de Pertinência Aproximada: Segunda Proposta

Em Pawlak (1994a) é proposta uma nova função de pertinência aproximada, que traduz mais fielmente a pertinência de um elemento do universo U a qualquer das regiões definidas por 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:

$$\mu^A_X(x) = \frac{|[x]_R \cap X|}{|[x]_R|}$$

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 $\mu_X^A(x)$. Observe que $[x]_R$ possui ao menos um elemento, qualquer $x \in U$. É importante observar que a função de pertinência, por sua vez, é função do conhecimento existente a respeito dos elementos do universo, ou seja, da partição U/R. Essa definição permite apenas que se fale em pertinência aproximada de um elemento x do universo, ao conjunto X, dado o conhecimento U/R. Ela serve para medir a que extensão os elementos da classe de equivalência $[x]_R$ estão em X, com relação a U/R, num espaço aproximado $A = (U,R)$.

É importante notar que as duas funções de pertinência aproximada propostas são dependentes da relação de equivalência R. Dado o espaço aproximado $A = (U,R)$, $X \subseteq U$ e $x \in U$, para ambas as definições, conforme mostrado em (UCHÔA, 1998), são válidas as seguintes propriedades:

1. $\mu_X(x) = 0 \Leftrightarrow x \in neg(X)$
   Demonstração: $x \in neg(X) \Leftrightarrow x \in U - A_{sup}(X) \Leftrightarrow [x]_R \cap X = \varnothing \Leftrightarrow \mu_X(x) = 0$.

2. $\mu_X(x) = 1 \Leftrightarrow x \in pos(X)$
   Demonstração: $x \in pos(X) \Leftrightarrow x \in A_{inf}(X) \Leftrightarrow [x]_R \subseteq X \Leftrightarrow \mu_X(x) = 1$.

3. $0 < \mu_X(x) < 1 \Leftrightarrow x \in duv(X)$
   Demonstração: $x \in duv(X) \Leftrightarrow [x]_R \cap X \neq \varnothing$ e $[x]_R \cap (U - X) \neq \varnothing \Leftrightarrow \mu_X(x) > 0$ e $\mu_X(x) < 1 \Leftrightarrow 0 < \mu_X(x) < 1$.

4. $0 \leq \mu_X(x) \leq 1$
   Demonstração: Dado $x \in U$, x obrigatoriamente pertencerá a uma, e somente uma, das regiões de X a saber: $neg(X)$, $duv(X)$ e $pos(X)$. Assim, pelas propriedades anteriores, $\mu_X(x) = 0$ ou $0 < \mu_X(x) < 1$ ou $\mu_X(x) = 1 \Leftrightarrow$

$0 \leq \mu_X(x) \leq 1$.

5. $\mu_{-X}(x) = 1 - \mu_X(x)$
   Demonstração: $\mu_{-X}(x) = \frac{|[x]_R \cap (U - X)|}{|[x]_R|} = \frac{|[x]_R \cap (U)|}{|[x]_R|} - \frac{|[x]_R \cap (X)|}{|[x]_R|} =$
   $1 - \frac{|[x]_R \cap (X)|}{|[x]_R|} = 1 - \mu_X(x)$.

6. se xRy, então $\mu_X(x) = \mu_X(y)$
   Demonstração: segue diretamente do fato que, se xRy, então $[x]_R = [y]_R$ e, portanto, $\mu_X(x) = \frac{|[x]_R \cap (X)|}{|[x]_R|} = \frac{|[y]_R \cap (X)|}{|[y]_R|} = \mu_X(y)$.

7. se $R = \{(x,x)|x \in U\}$, então μ_X é a função característica (função de pertinência crisp)
   Demonstração: se $R = \{(x,x)|x \in U\}$, então $|[x]_R| = 1$, para qualquer $x \in U$. Mais ainda, $|[x]_R \cap (X)| = 1$ se $x \in X$ e $|[x]_R \cap (X)| = 0$ se $x \notin X$.

8. $\mu_{X \cup Y}(x) \geq max[\mu_X(x), \mu_Y(x)]$
   Demonstração: $\mu_{X \cup Y}(x) = \frac{|[x]_R \cap (X \cup Y)|}{|[x]_R|} \geq \frac{|[x]_R \cap (X)|}{|[x]_R|} = \mu_X(x)$. De forma semelhante, obtém-se $\mu_{X \cup Y}(x) \geq \mu_Y(x)$. Donde $\mu_{X \cup Y}(x) \geq max[\mu_X(x), \mu_Y(x)]$.

9. $\mu_{X \cap Y}(x) \leq min[\mu_X(x), \mu_Y(x)]$
   Demonstração: $\mu_{X \cap Y}(x) = \frac{|[x]_R \cap (X \cap Y)|}{|[x]_R|} \leq \frac{|[x]_R \cap (X)|}{|[x]_R|} = \mu_X(x)$. De forma semelhante, obtém-se $\mu_{X \cap Y}(x) \leq \mu_Y(x)$. Donde $\mu_{X \cap Y}(x) \leq min[\mu_X(x), \mu_Y(x)]$.

Exemplo 11 *Seja um espaço aproximado $A = (U,R)$ e $X \subseteq U$, conforme mostra a Figura 17. O grau de pertinência dos elementos x e y, dado por*

$$\mu_X(x) = \frac{5}{8} = 0.635 \quad e \quad \mu_X(y) = \frac{4}{10} = 0.4$$

evidencia valores mais representativos para elementos da fronteira do que 0.5. A observação de tais valores permite dizer que x "pertence mais" a X que y.

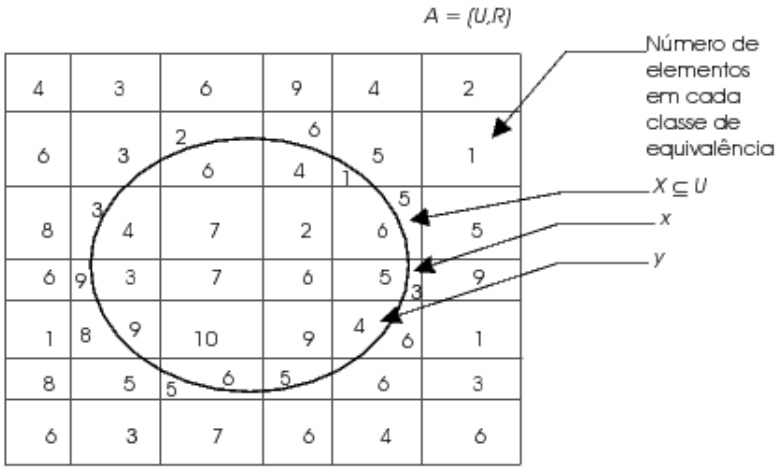


Figura 17: Espaço aproximado $A = (U, R)$ com os respectivos números de elementos de cada classe de equivalência (Extraída de (UCHÔA, 1998)).

4.2.3 Reescrita dos Conceitos Básicos da TCA Usando a Função de Pertinência Aproximada

Conforme observado em Pawlak (1994b), a função de pertinência definida anteriormente pode ser utilizada para redefinir os conceitos de aproximações e regiões de um conjunto. Com efeito, dado um espaço aproximado $A = (U, R)$ e um conjunto $X \subseteq U$, então:

$$A_{inf}(X) = \{x \in U | \mu_X(x) = 1\}$$

$$A_{sup}(X) = \{x \in U | \mu_X(x) > 0\}$$

$$pos(X) = \{x \in U | \mu_X(x) = 1\}$$

$$neg(X) = \{x \in U | \mu_X(x) = 0\}$$

$$duv(X) = \{x \in U | 0 < \mu_X(x) < 1\}$$

4.3 Relação entre Conjuntos *Fuzzy* e Conjuntos Aproximados

Em (UCHÔA, 1998) é apresentada uma discussão a respeito da "concorrência" entre os conjuntos *fuzzy* e os conjuntos aproximados, no que diz respeito à modelagem de incerteza. Em (PAWLAK, 1984) e (PAWLAK, 1985), o autor insiste em mostrar que o conceito de conjuntos aproximados é mais abrangente que o conceito de conjuntos *fuzzy*, afirmando que os conjuntos *fuzzy* constituem um caso particular dos conjuntos aproximados. Porém, para isso ser verdade, teria que acontecer o seguinte:

$$A_{inf}(X) \cup A_{inf}(Y) = A_{inf}(X \cup Y) \; e \; A_{sup}(X) \cap A_{sup}(Y) = A_{sup}(X \cap Y)$$

o que, conforme mostrado por Uchôa (1998), não é sempre verdade.

Já Wygralak (1989) afirma que tal abragência dos conjuntos aproximados em relação aos conjuntos *fuzzy* só é verificado quando o conjunto de avaliação da função de pertinência *fuzzy* é $\{0, 0.5, 1\}$. Porém a conclusão mais contundente é aquela apresentada em (KLIR; YUAN, 1995), citado por Uchôa (1998), que afirma que *"conjuntos fuzzy e conjuntos aproximados modelam tipos diferentes de incerteza. Desde que ambos tipos são relevantes em algumas aplicações, é útil combinar os dois conceitos."*

Os conjuntos *fuzzy*, conforme dito anteriormente, lidam com conceitos vagos, atribuindo valores pertencentes ao intervalo $[0, 1]$ para representar a pertinência de elementos a esses conceitos que, geralmente, dependem do contexto de aplicação.

Já os conjuntos aproximados trabalham com um outro tipo de incerteza: a indiscernibilidade. Neste caso, acontece que somente as informações disponíveis sobre os objetos não são suficientes para distinguir um do outro com relação a um ou mais conceitos.

Com isso, já que um conceito não sobrepõe o outro, tais abordagens devem ser utilizadas em cooperação quando esses dois tipos de incertezas existir, conforme

proposto por Sarkar e Yegnanarayana (1998). Nesse trabalho, considerou-se um problema de classificação onde havia indiscernibilidade nos padrões de entrada e vagueza com relação as saídas do problema. Assim, foi proposto um modelo híbrido, em que conceitos da TCA eram utilizados para modelar a indiscernibilidade existente nas entradas e conceitos de conjuntos *fuzzy* eram usados para lidar com a vagueza das saídas.

5 Algoritmos de Aprendizado de Máquina Utilizando a Teoria dos Conjuntos Aproximados

Como visto anteriormente, a incerteza está sempre presente nos bancos de informações que os sistemas especialistas possuem para construir seus aprendizados. Desta maneira, os algoritmos responsáveis por essa construção devem estar aptos em tratar tal incerteza, para que o sistema especialista consiga fornecer respostas consistentes mesmo em casos reais, em que as informações não são perfeitas.

Neste capítulo, serão apresentados alguns algoritmos de aprendizado de máquina que utilizam elementos da TCA para construir suas regras de aprendizado que mapeam as informações de sua base de conhecimentos. Além disso, serão apresentados os resultados que alguns destes algoritmos conseguiram em testes utilizando um banco de dados comum, a fim de comparar os desempenhos.

5.1 O Algoritmo RS1

O RS1 é um algoritmo de aprendizado indutivo de máquina baseado no conceito de índice discriminante de atributos, que foi apresentado na seção 3.7.2. Esse algoritmo foi proposto por Wong e Ziarko (1986) e seus passos estão apresentandos a seguir, considerando a tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho)$:

1. Calcular $Class_S(\delta) = \{X_1, ..., X_n\}$, a família de conjuntos elementares do espaço aproximado induzido por $\{\delta\}$.
2. Fazer $j = 1$
3. Fazer $U' = U, C' = C, B = \varnothing, X = X_j$ e $S' = \{U', C \cup \{\delta\}, V, \rho\}$.
4. Calcular o conjunto de índices discriminantes $\{\alpha_{B'}(X) | B' = B \cup \{c\}, \forall c \in C'\}$ em S'.
5. Selecionar o conjunto de atributos $B' = B \cup \{c\}$ com maior valor $\alpha_{B'}(X)$.
6. Fazer $B = B'$.

7. Se $A_{B-inf}(X) = \varnothing$, ir para o passo 10.

8. Identificar os conjuntos elementares $E = \{E_1, ..., E_r\}$ do espaço aproximado induzido por B que estão contidos em $A_{B-inf}(X)$.

9. Para cada elemento $E_k \in E$, gerar uma regra de decisão determinística (consistente). Considerando que B possui m atributos, então cada regra tem a forma $(a_i = v_{a_i})\&...\&(a_m = v_{a_m}) \Rightarrow (b = v_b)$, onde $a_1, ..., a_m \in B$. Cada elemento $(a_k = v_{a_k})$ do antecedente é construído da seguinte forma: a_k recebe o nome do k-ésimo atributo de B e v_{a_k} recebe o valor atribuido a E_k por esse atributo.

10. Fazer $U' = U' - ((U' - A_{B-sup}(X)) \cup A_{B-inf}(X))$, $X = X - A_{B-inf}(X)$ e $S' = \{U', C \cup \{\delta\}, V, \rho\}$.

11. Se $U' = \varnothing$ ir para o passo 16.

12. Fazer $C' = C' - B$.

13. Se $C' \neq \varnothing$, voltar ao passo 4.

14. Identificar todos conjuntos elementares $E = \{E_1, ..., E_r\}$ do espaço aproximado induzido por B.

15. Para cada elemento $E_k \in E$, gerar uma regra de decisão não-determinística (inconsistente) de forma semelhante à descrita no passo 9.

16. Fazer $j = j + 1$.

17. Se $j \leq n$ voltar ao passo 3.

18. Devolver todas as regras encontradas nos passos 9 e 15.

Tabela 3: SRC onde $U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ e $Q = \{a, b, c, d, e\}$

U	a	b	c	d	e
x_1	1	0	0	1	1
x_2	1	0	0	0	1
x_3	0	0	0	0	0
x_4	1	1	0	1	0
x_5	1	1	0	2	2
x_6	2	2	0	2	2
x_7	2	2	2	2	2

Exemplo 12 *(Extraído de (UCHÔA, 1998)) Dada a tabela 3, aplicando os passos do algoritmo RS1 vistos anteriormente, obtemos as seguintes regras:*

- $(b = 0)\&(a = 1) \Rightarrow (e = 1)$
- $(a = 0) \Rightarrow (e = 0)$
- $(a = 1)\&(b = 1)\&(d = 1) \Rightarrow (e = 0)$
- $(d = 2) \Rightarrow (e = 2)$

Conforme comentado por Uchôa (1998) esse algoritmo é, sob determinadas condições, apenas um caso específico do algoritmo bem mais conhecido e explorado ID3 (QUINLAN, 1986).

5.2 O algoritmo RS1+

Em Uchôa (1998) são identificadas alguns problemas e propostas as devidas melhorias com relação ao algoritmo anterior:

- O algoritmo original não consegue perceber quando a adição de atributos no B' não melhora o índice discriminante do B' anterior e segue adicionando os outros atributos de C. A melhoria proposta foi armazenar o valor do melhor (maior) valor do índice discriminante para a utilização durante toda a execução.
- No passo 5 do algoritmo apresentado anteriormente, quando existe dois conjuntos de atributos com o mesmo valor dos índices discriminantes, e maiores que todos os outros, há uma indecisão em qual conjunto escolher. Dependendo dessa escolha, há a possibilidade de se gerar regras maiores (com antecedentes maiores e desnecessários) aumentando, assim, o tempo de resposta do sistema. A melhoria proposta foi a eliminação de atributos supérfluos no momento seguinte da geração das regras. Tal eliminação consiste em retirar um atributo (e seu respectivo valor) da regra e verificar se o número de regras inconsistens aumenta.

- Quando tem-se a existência de regras chamadas inconsistentes (ou não-determinísticas), ou seja, regras que possuem o mesmo antecendente e consequentes diferentes, há a necessidade em decidir qual regra será considerada, uma vez que as duas fazendo parte do conhecimento do sistema gera uma contradição e inconsistência no raciocínio do mesmo. O RS1 não possui mecanismos para evitar essa questão. Ele apenas gera as regras inconsistentes e as armazena, não realizando um tratamento posterior. A solução proposta foi agregar um valor à cada regra. Tal valor representa a credibilidade da regra. Assim, quando existir regras inconsistentes, permanece apenas a que tiver maior credibilidade. Obviamente, as regras consistentes têm credibilidade igual a 1 (100%). Essa credibilidade é calculada utilizando o conceito de funções de pertinência aproximadas, visto na seção 4.2.

O algoritmo RS1+, proposto por Uchôa (1998), é justamente a implementação dessas melhorias no algoritmo RS1.

5.2.1 Adição de custos nos atributos

Existe ainda um questionamento em cima do algoritmo RS1+ (e consequentemente em cima do RS1 também) com relação ao tratamento individual dado a cada atributo e, em alguns casos, seus respectivos valores. A questão é que esse algoritmo não leva em conta os custos relativos a cada atributo o que, na maioria das vezes, não reflete a realidade visto que, em um diagnóstico médico, por exemplo, um exame de ressonância magnética é bem mais impactante (discriminatório) que uma simples medição de temperatura. Porém o primeiro é bem mais caro e esses dois lados devem ser levados em conta no raciocínio. Assim, atributos custosos devem, quando for possível e razoável, ser substituídos por um ou mais atributos menos custosos mas com uma importância proporcional.

Com base nesse questionamento, foi proposta em (ANDRADE NETO, 2002) uma melhoria no RS1+ para tratar tais custos. Tal melhoria consiste na agregação de um valor numérico à cada atributo para representar o custo desse atributo. Ob-

viamente, quanto maior o custo do atributo, menos interessante é a sua utilização no raciocínio do sistema.

Para implementar essa modificação, inicialmente os custos, que nesse caso também são considerados como dados de entrada para o algoritmo, são normalizados da seguinte maneira:

Dado T um vetor de custos, a normalização dos seus valores é feita da seguinte forma:

$$T[i] = \frac{k}{T[i]},$$

onde k é o valor do menor custo.

Agora considerando o SRC $S = (U, C \cup \{\delta\}, V, \rho)$, um conjunto de custos T e uma função de custos σ. U, V e ρ são tais como num SRC, C é o conjunto de condições, δ é o atributo de decisão, T é um conjunto de custos normalizados e $\sigma : C \to T$ é uma *função de custo*, tal que $\sigma(x) \in T$, se $x \in C$. Seja também o espaço aproximado $A = (U, R)$ e $X \subseteq U$, o novo valor do índice discriminante será dado por:

$$\alpha_{R \cup \{c\}}(X) = \frac{|U| - |A_{sup}(X) - A_{inf}(X)|}{|U|} \cdot \sigma(c)$$

Com isso, para que um atributo a cujo custo é q vezes maior que o custo do atributo b seja garantidamente escolhido, a deve possuir um índice discriminante mais que q vezes maior que o de b. Caso os custos sejam todos iguais, com a normalização os elementos de T serão todos 1 e, portanto, não vão influenciar na multiplicação da fórmula de α.

Com relação às escolhas dos atributos, inicialmente é escolhido o atributo com o maior índice discriminante, cuja fórmula está apresentada logo acima. Caso este índice seja maior que o anterior, então prossegue-se normalmente. Caso não seja maior que o anterior (uma exigência do algoritmo original), passa-se a executar o RS1+ normalmente, sem levar em conta os custos. Desta maneira, evita-se que regras inconsistentes sejam geradas em função da eliminação de atributos custosos e, por outro lado, faz com que esses últimos sejam usados apenas em casos críticos.

5.2.2 Adição de custos nos valores dos atributos

Ainda em (ANDRADE NETO, 2002), propõe-se uma agregação de custos nos valores dos atributos. Para isso, é sugerida a reformulação da base de dados com a criação de novos atributos, antes que a mesma seja utilizada para a construção das regras.

De acordo com este autor, para cada atributo a_i com n valores possíveis e custos associados a esses valores, deve-se eliminar esse atributo e gerar n novos atributos a_{i_j}, $1 \leq j \leq n$, de forma que cada novo atributo seja booleano, demostrando presença ou ausência do valor associado, ou seja,

$$\rho(x_k, a_i) = j \Rightarrow \rho(x_k, a_{i_j}) = \mathbf{s}$$

$$\rho(x_k, a_i) \neq j \Rightarrow \rho(x_k, a_{i_j}) = \mathbf{n}$$

O custo do novo atributo a_{i_j} será então o custo do valor j associado a a_i.

Exemplo 13 *(Extraído de (ANDRADE NETO, 2002)) Dado o SRC da Tabela 4. Os atributos a, b e c têm pesos associados aos seus valores, descritos na Tabela 5. O novo SRC a ser utilizado como entrada do RS1+ encontra-se na Tabela 6, e os pesos dos seus atributos são os pesos dos custos dos valores na Tabela 5.*

Tabela 4: SRC onde $U = \{x_1, x_2, x_3, x_4, x_5\}$ e $Q = \{a, b, c\}$

U	a	b	c
x_1	1	3	5
x_2	2	4	4
x_3	1	2	5
x_4	1	3	5
x_5	2	2	4

Tabela 5: Pesos dos valores associados aos atributos do SRC da Tabela 4

Atributo	a	a	b	b	b	c	c
Valor	1	2	2	3	4	4	5
Peso	1	3	2	3	4	3	3

Tabela 6: SRC da Tabela 4 com custos agregados aos valores associados aos atributos.

U	a_1	a_2	b_2	b_3	b_4	c
x_1	s	n	n	s	n	5
x_2	n	s	n	n	s	4
x_3	s	n	s	n	n	5
x_4	s	n	n	s	n	5
x_5	n	s	s	n	n	4

5.3 O algoritmo *lmurf*

O algoritmo *lmurf* (*Learning Machine using Rough Functions*) foi proposto e desenvolvido por Domingues (2002) para mostrar a possibilidade da criação de algoritmos de aprendizado de máquina subsidiados por funções aproximadas. Tal algoritmo produz uma árvore de decisão e as regras podem ser obtidas através do percurso dessa árvore até os nós folhas.

5.3.1 Funções Aproximadas

Sejam $A = (X,S)$ e $B = (Y,P)$ dois espaços aproximados quaiquer e seja uma função $f : X \to Y$. A representação de f em A é a função $f_{(A,B)} : X \to 2^Y$, tal que:

$$f_{(A,B)}(x) = \{y \in Y | z \in [x]_S, f(z) = y\},$$

ou seja, a (A,B)-aproximação de f em x é o conjunto formado por todos os elementos de y que são imagem dos elementos de X pertencentes à mesma classe de equivalência de x. Conforme apresentado em (UCHÔA, 1998), a *aproximação inferior* e a *aproximação superior* de f em x, que são, respectivamente,as funções $f_{inf}(x) : X \to Y/P$ e $f_{sup}(x) : X \to Y/P$, definidas por

$$f_{inf}(x) = A_{inf}(f_{(A,B)}(x)), \forall x \in X,$$

$$f_{sup}(x) = A_{sup}(f_{(A,B)}(x)), \forall x \in X.$$

Diz-se que uma função f é *exata* em $x \in X$ se e somente se $f_{inf}(x) = f_{sup}(x)$. Do contrário, f é *inexata*. O conjunto $f_{sup}(x) - f_{inf}(x)$ é o erro da aproximação de f em x, podendo ser medido por $\upsilon_f(x) = (f_{sup}(x) - f_{inf}(x))/f_{sup}(x)$. Como o interesse maior é aproximar todos os pontos de X, define-se a *aproximação inferior* e a *aproximação superior* de $f_{(A,B)}$ em X, em símbolos $A_{inf}(f) : X \to 2^Y$ e $A_{sup}(f) : X \to 2^Y$, respectivamente, da seguinte forma:

$$A_{inf}(f) = \{(x, [y]_S) \in X \times Y/P | y \in f_{inf}(x)\},$$

$$A_{sup}(f) = \{(x, [y]_S) \in X \times Y/P | y \in f_{sup}(x)\}.$$

5.3.2 Funcionamento do algoritmo

O algoritmo lmurf é descrito em (DOMINGUES, 2002) da seguinte maneira:

> O algoritmo possui como entrada uma tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho)$ e uma variável $pert = 0$. Com esta tabela, o algoritmo calcula $F_{inf}(X_r)$, $\forall X_r \in Class_s(\delta)$, $1 \leq r \leq |Class_s(\delta)|$, com maior número de elementos e gera proposição(ões) consistente(s) para os nós da árvore de decisão. Se $F_{inf}(X_r) = \varnothing$, o algoritmo irá calcular $F_{sup}(X_r)$, $\forall X_r \in Class_s(\delta)$, $1 \leq r \leq |Class_s(\delta)|$) e escolher $F_{sup}(X_r) \neq U$ com maior número de elementos para gerar proposição(ões) inconsistentes para os nós da árvore de decisão. Caso $F_{inf}(X_r) = \varnothing$ e $F_{sup}(X_r) = U$, o algoritmo calcula $\mu_{X_r}^{B_i}(x)$ para todo $X_r \in Class_s(\delta)$, $1 \leq r \leq |Class_s(\delta)|$, e todo x pertencente a cada conjunto elementar de cada um dos espaços aproximados $B_i = (U, P_i)$, tal que P_i é a relação de equivalência induzida pelo atributo $c_i \in C$, $1 \leq i \leq |C|$. Se $|\mu_{X_r}^{B_i}(x)| \leq pert$ ou $|B_i| = 1$, uma proposição de conclusão é ge-

rada e inserida em um novo nó da árvore de decisão, senão o algoritmo divide a tabela de decisão S em duas novas tabelas $S_1 = (U_1, C \cup \{\delta\}, V, \rho)$ e $S_2 = (U_2, C \cup \{\delta\}, V, \rho)$, através do maior valor obtido com o cálculo da função de pertinência aproximada $\mu_{X_r}^{B_i}(x)$. Uma proposição baseada na função de pertinência aproximada é gerada e inserida em um novo nó da árvore decisão. O algoritmo é então reiniciado recursivamente com S_1 e depois com S_2.

Dada uma tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho\}$, pode-se verificar, detalhadamente, todos os passos do algoritmo *lmurf*:

1. Inicializar variáveis (fazer `pert=0`, `restart=false`, `nó_pai=NULO`, `nó_atual=nó_pai`)

2. Se U = $\varnothing$ ou `restart=true`, retornar a árvore de decisão (cujo nó raiz aponta para `nó_pai`) e terminar o algoritmo

3. Fazer `restart=true` e calcular $S = Class_s(\delta) = \{X_1, \ldots, X_n\}$, a família de conjuntos elementares do espaço aproximado $A = (U, \{\delta\})$. Se a cardinalidade de S for igual a 1 ($|S| = 1$), ir para o passo 15

4. Calcular $F_{inf}(X_r) = A_{inf}(F_{(A,B_i)}(X_r))$, para todo $X_r \in S$, $1 \leq r \leq |S|$, e todo $B_i = (U, P_i)$, tal que P_i é a relação de equivalência induzida pelo atributo $c_i \in C$, $1 \leq i \leq |C|$. $F_{(A,B_i)}(X_r)$ é a função de classificação aproximada do conjunto elementar X_r, baseada na função identidade f e no espaço aproximado $B_i = (U, P_i)$. Se $F_{inf}(X_r) = \varnothing$, para todo $1 \leq r \leq |S|$, ir para o passo 7

5. Fazer `restart=false`. Selecionar a $F_{inf}(X_r)$, $1 \leq r \leq |S|$, com maior número de elementos. Identificar os conjuntos elementares $E = \{E_1, \ldots, E_j\}$, $1 \leq j \leq |F_{inf}(X_r)|$, contidos em $F_{inf}(X_r)$. Para cada $E_k \in E$, $1 \leq k \leq j$, gerar uma proposição de condição para um nó não-folha da árvore de decisão e uma proposição de decisão para um nó folha da árvore. Cada nó da árvore tem a forma $(a = v_{ae_1}) ou \ldots ou (a = v_{ae_j})$, onde a é o atributo que gera o espaço aproximado B_i utilizado em $F_{inf}(X_r)$. Por sua vez, $e_k \in U$, $1 \leq k \leq j$, é um elemento qualquer do conjunto elementar E_k. Cada proposição de condição $(a = v_{ae_k})$ é constituída da seguinte forma: a recebe o nome do atributo que gera o espaço aproximado B_i

utilizado em $F_{inf}(X_r)$ e v_{ae_k}, $1 \leq k \leq j$, recebe o valor atribuído a cada elemento e_k por esse atributo. As proposições de decisão são constituídas de forma semelhantes, com a exceção de que a recebe o nome do atributo de decisão δ. Fazer `nó_não-folha=proposição(ões) de condição` e `nó_folha=proposição(ões) de decisão`. Executar a função `Inserir_nó(nó_não-folha)` e `Inserir_nó(nó_folha)`.

As conclusões tiradas a partir da árvore de decisão e que envolvam estes nó, são classificadas como conclusões consistentes.

6. Fazer $U = U - F_{inf}(X_r)$. Se para o novo conjunto U, o número de conjuntos elementares do espaço aproximado B_i utilizado em $F_{inf}(X_r)$ for igual a 1, remover o atributo $c_i \in C$, $1 \leq i \leq |C|$, que induziu o espaço aproximado B_i. Ir para o passo 2

7. Calcular $F_{sup}(X_r) = A_{sup}(F_{(A,B_i)}(X_r))$, para todo $X_r \in S$, $1 \leq r \leq |S|$, e todo $B_i = (U, P_i)$, tal que P_i é a relação de equivalência induzida pelo atributo $c_i \in C$, $1 \leq i \leq |C|$. $F_{(A,B_i)}(X_r)$ é a função de classificação aproximada do conjunto elementar X_r, baseada na função identidade f e no espaço aproximado $B_i = (U, P_i)$. Se $F_{sup}(X_r) = U$, para todo $1 \leq r \leq |S|$, ir para o passo 10

8. Fazer `restart=false`. Selecionar a $F_{sup}(X_r)$, $1 \leq r \leq |S|$, com maior número de elementos. Identificar os conjuntos elementares $E = \{E_1, \ldots, E_j\}$, $1 \leq j \leq |F_{sup}(X_r)|$, contidos em $F_{sup}(X_r)$. Para cada $E_k \in E$, $1 \leq k \leq j$, gerar uma proposição de condição para um nó não-folha da árvore de decisão e uma proposição de decisão para um nó folha da árvore. Cada nó da árvore tem a forma $(a = v_{ae_1}) ou \ldots ou (a = v_{ae_j})$, onde a é o atributo que gera o espaço aproximado B_i utilizado em $F_{sup}(X_r)$. Por sua vez, $e_k \in U$, $1 \leq k \leq j$, é um elemento qualquer do conjunto elementar E_k. Cada proposição de condição $(a = v_{ae_k})$ é constituída da seguinte forma: a recebe o nome do atributo que gera o espaço aproximado B_i utilizado em $F_{sup}(X_r)$ e v_{ae_k}, $1 \leq k \leq j$, recebe o valor atribuído a cada elemento e_k por esse atributo. As proposições de decisão são constituídas de forma semelhantes, com a exceção de que a recebe o nome do atributo de decisão δ. Fazer `nó_não-folha=proposição(ões) de condição, nó_ folha=proposição(ões) de decisão`. Inserir o nó não-folha e o nó folha.

   As conclusões tiradas a partir da árvore de decisão e que envolvam estes nó, são classificadas como conclusões inconsistentes.

9. Fazer $U = U - F_{sup}(X_r)$. Se para o novo conjunto U, o número de conjuntos elementares do espaço aproximado B_i utilizado em $F_{sup}(X_r)$ for igual a 1, remover o atributo $c_i \in C$, $1 \leq i \leq |C|$, que induziu o espaço aproximado B_i. Ir para o passo 2

10. Calcular a pertinência aproximada $\mu_{X_r}^{B_i}(x) = \frac{|[x]_R \cap X_r|}{|[x]_R|}$, para todo $X_r \in S$, $1 \leq r \leq |S|$, e todo x pertencente a cada conjunto elementar de cada um dos espaços aproximados $B_i = (U, P_i)$, tal que P_i é a relação de equivalência induzida pelo atributo $c_i \in C$, $1 \leq i \leq |C|$. Selecionar a pertinência aproximada com o maior valor. Se a pertinência selecionada for menor ou igual a variavél `pert` ou se o número de conjuntos elementares do espaço aproximado B_i da pertinência aproximada de maior valor for igual a 1, ir para o passo 15

11. Fazer `pert` = $\mu_{X_r}^{B_i}(x)$. Para os elementos e_k, $1 \leq k \leq j$, contidos no conjunto elementar do espaço aproximado B_i da pertinência aproximada de maior valor, gerar uma proposição (condição) para um nó não-folha da árvore de decisão. A proposição de condição $(a = v_{ae_k})$ é constituída da seguinte forma: a recebe o nome do atributo que gerou o espaço aproximado B_i utilizado em $\mu_{X_r}^{B_i}(x)$ e v_{ae_k}, $1 \leq k \leq j$, recebe o valor atribuído a cada elemento e_k (do conjunto elementar do espaço aproximado B_i que gerou o maior valor para $\mu_{X_r}^{B_i}(x)$) por esse atributo. Fazer `nó_não-folha=proposição(ões) de condição`. Executar a função `Inserir_nó (nó_não-folha)`.

12. Dividir a tabela de decisão $S = (U, C \cup \{\delta\}, V, \rho)$ em duas novas tabelas: $S_1 = (U_1, C \cup \{\delta\}, V, \rho)$ e $S_2 = (U_2, C \cup \{\delta\}, V, \rho)$, onde U_1 é formado pelos elementos do conjunto elementar E_k, $1 \leq k \leq |B_i|$, do espaço aproximado $B_i = (U, P_i)$, tal que P_i é a relação de equivalência induzida pelo atributo $c_i \in C$, $1 \leq i \leq |C|$, que gerou a $\mu_{X_r}^{B_i}(x)$ de maior valor, e U_2 o conjunto formado pelos elementos que não pertence ao conjunto elementar E_k

13. Remover o atributo $c_i \in C$, $1 \leq i \leq |C|$, que induziu o espaço aproximado B_i da pertinência aproximada de maior valor, do conjunto de atributos C da tabela de decisão $S_1 = (U_1, C \cup \{\delta\}, V, \rho)$. Fazer `restart=false`. Realizar uma chamada recursiva com S_1, indo para o passo 2

14. Se com a tabela de decisão $S_2 = (U_2, C \cup \{\delta\}, V, \rho)$ o número de conjuntos elementares do espaço aproximado B_i que gerou o maior valor para $\mu_{X_r}^{B_i}(x)$ for igual a 1, remover o atributo c_i, $1 \leq i \leq |C|$, que induziu o espaço aproximado B_i, da tabela de decisão S_2. Fazer `restart=false`. Realizar uma chamada recursiva com S_2, indo para o passo 2

15. Para cada elemento $e_k \in U$, gerar uma proposição (decisão) para um nó folha da árvore de decisão. Cada nó folha da árvore tem a forma $(\delta = v_{ae_1}) ou \ldots ou (\delta = v_{ae_j})$,

onde δ é o atributo de decisão e $e_k \in U$, $1 \leq k \leq j$, é um elemento qualquer do conjunto U. Cada proposição de decisão $(\delta = v_{ae_k})$ é constituída da seguinte forma: δ recebe o nome do atributo de decisão e v_{ae_k}, $1 \leq k \leq j$, recebe o valor atribuído a cada elemento e_k por esse atributo. Fazer `nó_folha=proposição(ões) de decisão`. Executar a função `Inserir_nó (nó_folha)`. Fazer $U = \varnothing$ e ir para o passo 2

Exemplo 14 *(Extraído de (DOMINGUES, 2002)) Considere o SRC representado pela Tabela 7, onde* $U = \{x_1, x_2, x_3 \ldots, x_6\}$, *e* $Q = \{a, b, c, d\}$.

Tabela 7: SRC 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

Aplicando o algoritmo lmurf *na tabela de decisão* $S = (U, C \cup \{\delta\}, V, \rho)$ *onde* $C = \{a, b, c\}$ *e* $\delta = d$, *tem-se os seguintes resultados:*

- Árvore de Decisão

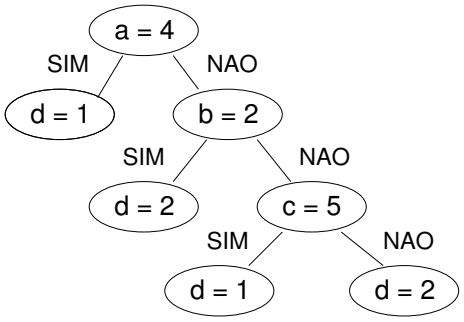


Figura 18: Árvore de decisão gerada pelo algoritmo *lmurf* aplicado no SRC da Tabela 7.

- Regras
  $(a = 4) \Rightarrow (d = 1)$
  $(a \neq 4)\ \&\ (b = 2) \Rightarrow (d = 2)$
  $(a \neq 4)\ \&\ (b \neq 2)\ \&\ (c = 5) \Rightarrow (d = 1)$
  $(a \neq 4)\ \&\ (b \neq 2)\ \&\ (c \neq 5) \Rightarrow (d = 2)$

5.4 O algoritmo ML-VPM

A Teoria dos Conjuntos Aproximados, como já dito, é um formalismo matemático que subsidia o tratamento de incerteza no desenvolvimento de algoritmos de aprendizado de máquina. Porém a teoria original, proposta em (PAWLAK, 1982), possui algumas limitações. Uma delas é o fato de que a TCA não tem mecanismos para lidar com informações incertas e probabilísticas, já que foi proposta pra modelar a indiscernibilidade. Com isso, tal teoria gera apenas regras determinísticas, com 100% de credibilidade.

Porém, isso nem sempre é adequado. Uma das características desejáveis em um sistema de aprendizado é justamente a generalização do conhecimento a partir de um banco de dados relativamente restrito. Ou seja, as regras geradas não devem mapear apenas o conhecimento presente estritamente na base de informações, mas também casos mais gerais. Para isso, devem ser geradas regras que não serão determinísticas para o conjunto de treinamento, ou seja, aprensetarão um erros de classificação, mas que serão capazes de generalizar e classificar bem os dados desconhecidos.

Diante dessa limitação, foi proposto o modelo probabilístico de precisão variável por Ziarko (1993) e, a partir dele, foi proposto o algoritmo ML-VPM, por Figueiredo (2003).

5.4.1 O modelo probabilístico de precisão variável

Segundo Ziarko (1993), *"o coração deste modelo extendido de conjuntos aproximados é a generalização da noção clássica da relação de inclusão entre conjuntos"*. Para essa generalização, o autor propõe uma medida para o erro de classificação, chamado de erro de classificação relativo, de um conjunto X em um conjunto Y, da seguinte maneira:

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

onde $card(X)$ representa a cardinalidade (número de elementos) de X.

Com esta definição, pode-se reformular a relação de inclusão entre conjuntos da seguinte maneira:

$$X \subseteq Y \text{ se, e somente se, } e(X,Y) = 0.$$

e, a partir desta reformulação, pode-se generalizão o conceito de inclusão entre conjuntos:

$$X \subseteq^{\beta} Y \text{ se, e somente se, } e(X,Y) < \beta, \text{ onde } 0 \leq \beta \leq 0,5.$$

esta última diz que X está β-íncluído em Y, ou seja, que os elementos de X podem ser classificados como elementos de Y com um erro de classificação β. Conforme o autor, o valor de β não deve ser maior que 0.5 para garantir que a maioria dos elementos de um conjunto possam ser classificados corretamente.

Uma vez propostas essas novas definições, é possível definir as chamadas aproximações e regiões probabilísticas, utilizando os conceitos apresentados acima:

- Aproximação inferior probabilística (região positiva): é a união dos conjuntos elementares que estão β-íncluídos em X.

$$A^{\beta}_{inf}(X) = \bigcup\{E \in class(R) :\ E \subseteq^{\beta} X\}, \text{ ou equivalentemente}$$
$$= \bigcup\{E \in class(R) :\ e(E,X) \leq \beta\}$$

- Aproximação superior probabilística:

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

- Região negativa probabilística: é justamente o complemento da aproximação superior probabilística:

$$neg^{\beta}(X) = \bigcup\{E \in class(R) :\ e(E,X) \geq 1-\beta\}$$

- Região duvidosa probabilística: são elementos que não podem ser classificados como pertencentes a X ou ao seu complemento com um erro de classificação não maior que β. É a diferença entre a aproximação positiva probabilística e aproximação negativa probabilística:

$$duv^{\beta}(X) = \bigcup\{E \in class(R) :\ \beta < e(E,X) < 1-\beta\}$$

5.4.2 Pseudo-código e funcionamento do algoritmo

Esta seção aprensenta o pseudo-código do algoritmo ML-VPM proposto e apresentado por Figueiredo (2003). O autor preferiu desenvolver tal código em alto nível, deixando independente de qualquer implementação. Além disso, dividiu a execução em dois algoritmos mostrados a seguir: um principal, chamado ML-VPM, e um auxiliar, chamado ML-VPM-Aux.

Algorithm 1 Algoritmo principal ML-VPM

procedure ML-VPM(C, δ, β)
    $A = (U, \bar{\delta})$
    for all $CE \in A$ **do**
        ML-VPM-Aux(C, δ, β, CE, $\varnothing$, U)
    end for
end procedure

Algorithm 2 Algoritmo auxiliar ML-VPM-Aux

procedure ML-VPM-AUX(C, δ, β, X, $C_{anterior}$, E)
    Ordenar C de acordo com o índice discriminante α.
    for all $a_i \in C$ **do**
        $C_{novo} = C_{anterior} \cup \{a_i\}$
        $A = (E, \overline{C_{novo}})$
        $n = |A|$ ▷ número de átomos de A
        if $n > 1$ **then** ▷ Tenta gerar regras determinísticas
            $A\_inf = A_{A-inf}(X)$
            if $A\_inf \neq \varnothing$ **then**
                GERA_ REGRAS_ DETERMINÍSTICAS(A_inf, C, δ)
            end if
            $E_{novo} = duv_A(X)$
            $X_{novo} = X$ - A_inf
            if $X_{novo} \neq 0$ **then** ▷ Tenta gerar regras probabilísticas
                $B = (E_{novo}, \overline{C_{novo}})$
                $A\_inf\_prob = A^{\beta}_{B-inf}(X_{novo})$
                if $A\_inf\_prob \neq \varnothing$ **then**
                    GERA_ REGRAS_ PROBABILÍSTICAS(A_inf_prob, C, δ)
                end if
                $C_{proximo} = \bigcup_{j=i+1}^{|C|} \{a_j\}\ (a_j \in C)$
                if $C_{proximo} \neq \varnothing$ **then**
                    for all $X_i \in B$ **do**
                        ML-VPM-Aux($C_{proximo}$, δ, β, X_{novo}, C, X_i)
                    end for
                end if
            end if
        end if
    end for
end procedure

Exemplo 15 *(Extraído de (FIGUEIREDO, 2003)) Considere o SRC representado pela tabela 7, onde* $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ *e* $Q = \{a, b, c, d\}$*. Considerando o conjuntos de atributos de condição* $C = \{a, b, c\}$ *e* $\delta = d$*, o algoritmo ML-VPM geraria as seguintes regras:*

- *Determinísticas:*

$$\begin{aligned}(a = 4) &\Rightarrow d = 1\\ (a = 6)\&(b = 2) &\Rightarrow d = 2\\ (a = 6)\&(c = 4) &\Rightarrow d = 2\\ (c = 4)\&(b = 2) &\Rightarrow d = 1\\ (c = 3) &\Rightarrow d = 1\\ (c = 5)\&(b = 1) &\Rightarrow d = 1\\ (c = 5)\&(b = 2) &\Rightarrow d = 2\end{aligned}$$

- *Probabilísticas:*

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

5.5 O algoritmo FID3

O algoritmo FID3 foi proposto por Ding, Zheng e Zang (2009) como uma forma de solucionar alguns problemas que, segundo eles, são inerentes ao ID3. Para os autores, o ganho de informação como medida para selecionar atributos tem uma polarização (bias) interno que favorece os atributos que possuem o maior número de valores possíveis. A escolha dos atributos por ganho de informação portanto, não pode ser sempre o melhor.

Um outro problema verificado é a instabilidade da construção da árvore de decisão a partir do ganho de informação. Os autores afirmam que a árvore de

decisão irá gerar regras de classificação diferentes, uma vez que os conjuntos de teste são modificados, mesmo que numa escala pequena.

Diante desses problemas, eles propõem uma medida baseada na dependência entre o atributo em questão e o atributo que representa as classes (atributo de decisão, ver Seção 3.8.2), dada por:

$$Gain_{fix}(A) = \sqrt[2]{\kappa(A,\delta) * \frac{Gain}{m}}$$

onde C é uma coleção de objetos, $\kappa(A,\delta) = |pos(A,\delta)|/|C|$ é o grau de dependência do atributo que representa a classe dos objetos δ do em relação a A, *Gain* é o ganho de informação clássico do ID3 (Seção 2.2.2) e m é o número de valores possíveis para o atributo $A \in C$.

O algoritmo FID3, portanto, consiste no seguinte:

- Entrada: Uma tabela de decisão $S = (U, C \cup \delta, V, \rho)$
- Saída: Uma árvore de decisão T

1. Criar um nó inicial para a árvore e vericar se todos os exemplos estão numa mesma classe.
2. Para cada atributo A de C, calcula-se o valor do grau de dependência κ de δ com relação a A. Se o valor de κ for 0, desconsidera-se o atributo. Ao final teremos o conjunto $C' = \{x | x \in C \ \& \ \kappa(x,\delta) \neq 0\}$.
3. Para cada atributo $C_i \in C'$ calcular o valor de $Gain_{fix}$. Escolher aquele atributo com o maior valor encontrado como nó raiz.
4. Separar os exemplos de acordo com os valores possíveis para C_i.
5. Se em uma partição todos os elementos forem da mesma classe, cria-se um nó folha com o valor da classe.

6. Caso contrário continua construindo a árvore de decisão de maneira recursiva em cada partição. Os atributos que fazem parte de um nó não podem ser usados nos descendentes desse nó.

7. Durante as etapas de particionamento e criação dos nós podem ocorrer duas situações adversas:

    - Se não existir mais exemplos de treinamento para ser classificados, pode-se criar o nó folha com o valor da classe na maioria dos exemplos;
    - Se chegar em um nó e todos os atributos já tiverem sido utilizados no caminho da raíz até esse nó, pode-se construir um nó folha com a valor da classe mais comum.

6 Algoritmo de Aprendizado Indutivo de Máquina Inspirado em Funções de Pertinência Aproximada

6.1 Introdução

Como visto anteriormente, as funções de pertinência aproximada são uma extensão das funções de pertinência clássica e medem o quanto um determinado elemento pertence a um determinado conjunto, num determinado espaço aproximado.

O objetivo deste capítulo é apresentar uma proposta de algoritmo que utiliza o conceito funções de pertinência aproximada para induzir conhecimento.

6.2 O Algoritmo *ID3-RMF* (ID3 *using Rough Membership Function*)

O ID3-RMF foi desenvolvido para demonstrar a possibilidade de desenvolvimento de algoritmos de aprendizado indutivo de máquina subsidiados por funções de pertinência aproximada. A ideia por trás do algoritmo consiste em uma reformulação do algoritmo clássico ID3 (QUINLAN, 1986) (Seção 2.2.2) utilizando o conceito de pertinência associado a conjuntos aproximados.

As funções de pertinência aproximada, como apresentado na seção 4.2, dão um grau de certeza da existência de um elemento em um conjunto qualquer, dentro de um determinado espaço aproximado. A partir daqui, considera-se somente a segunda proposta para as funções de pertinência aproximada (Seção 4.2.2). Desta maneira, o objetivo do algoritmo é, de maneira geral, substituir o cálculo da entropia e ganho de informação no ID3 por um cálculo baseado em uma média dos valores das pertinências aproximadas dos exemplos contidos nos conjuntos elementares induzidos por um atributo de condição nos conjuntos elementares induzidos pelo atributo de decisão, no espaço induzido pelo atributo de condição.

Investigando um pouco mais as funções de pertinência aproximada, percebe-se que elas retornam valores entre 0 e 1: Se o número de elementos na interseção da classe de equivalência do elemento com um determinado conjunto for alto, a

divisão desse número pela cardinalidade da classe resultará em um valor próximo de 1; se o número de elementos na interseção for baixo, o resultado será próximo de 0.

Porém, para o algoritmo aqui proposto, o interessante é selecionar atributos que induzem conjuntos elementares mais definíveis no espaço induzido pela decisão, ou seja, conjuntos elementares cujos elementos possuem um grau alto de certeza de pertinência ou de não-pertinência nos átomos de decisão. Com isso, são desejáveis valores de pertinência aproximada próximos de 1 (pertinência) ou próximos de 0 (não-pertinência). Para obter esses valores de interesse, foi proposto um reajuste da seguinte forma: seja $\mu_X(x)$ a pertinência aproximada de x em X, o novo valor a ser considerado pelo algoritmo será $|1-2\mu_X(x)|$.

Portanto, a escolha do atributo que irá fazer parte de um nó não-folha da árvore de decisão gerada pelo algoritmo ID3-RMF será feita com base nos valores das médias aritméticas das pertinências aproximadas de cada elemento nos átomos induzidos pelo atributo de condição nos átomos induzidos pelo atributo de decisão, no espaço aproximado induzido pelo atributo de condição. Contudo, o cálculo do valor da pertinência aproximada se mostra muito custoso, já que analisa interseções e, por definição, todos elementos de uma determinada classe terão o mesmo valor de pertinência aproximada num determinado conjunto. Desta maneira, calcula-se o valor da pertinência aproximada de apenas um elemento da classe e multiplica esse valor pelo número de elementos da classe.

Seja conjunto universo U de exemplos, C o conjunto de atributos de condição e d o atributo de decisão. O funcionamento do algoritmo pode ser descrito da seguinte maneira:

1. Verifica se todos os exemplos do universo pertencem à uma mesma classe;
2. Se todos elementos pertencerem à uma mesma classe, cria um nó folha com o valor daquela classe e retorna-o;
3. Caso ainda existam elementos pertencentes a classes diferentes, verifica se ainda existem atributos em C (a serem considerados naquele ramo da árvore, visto que,

como no algoritmo ID3, um atributo que faz parte de um nó não pode ser escolhido em qualquer descendente desse nó);

4. Se não há mais atributos a serem considerados em C, cria um nó folha com os valores das classes dos exemplos do universo e retorna-o;

5. Caso existam atributos em C ainda não utilizados, cria o espaço aproximado induzido por d (será usada a notação U/d para designá-lo).

6. Para cada atributo $c \in C$, cria o espaço aproximado induzido por c (será usada a notação U/c para designá-lo) e fazer $media_c = 0$; $\forall D_i \in U/d$, para cada $E_i \in U/c$, calcula $media_c = media_c + |1 - 2 * \mu_{Di}^{U/c}(E_i[0])| * |E_i|/(|U| * |U/d|)$. Onde $|E_i|$, $|U|$ e $|U/d|$ representam a cardinalidade (número de elementos) de um conjunto e $|1 - 2 * \mu_{Di}(E_i[0])|$ representa o módulo (valor absoluto) de um número.

7. Escolher o atributo A que tiver o maior valor entre as médias de pertinências aproximadas e criar um nó não-folha para armazenar esse atributo;

8. Para cada valor possível a_i de A, faz $U_{A_{a_i}}$ (elementos do universo U que possuem valor a_i para o atributo A) e voltar ao passo 1 considerando o universo U como sendo $U_{A_{a_i}}$ e os atributos de condição C como sendo $C - \{A\}$.

O algoritmo desenvolvido, assim como o ID3, produz uma árvore de decisão cujas regras são obtidas percorrendo a árvore de sua raiz até um de seus nós folhas. Ao percorrer a árvore, os nós são analisados indicando o caminho que se deve seguir, sendo que os nós não-folha compõem o antecedente de uma regra (condições de uma regra) e os nós folhas compõem o conseqüente da regra (conclusão de uma regra).

Para exemplificar a construção de uma árvore, segue a aplicação do algoritmo ID3-RMF na tabela 1, que foi adaptada de (QUINLAN, 1986), sendo U = $\{x1..x12\}$, C = $\{outlook, windy, temperature, humidity\}$ e d = D:

Início da execução do algoritmo

Tabela 8: Exemplos de treinamento para o algoritmo ID3-RMF contendo dados do clima de algumas manhãs de sábado, onde as classes são Y e N.

No.	outlook	windy	temperature	humidity	D
$x1$	overcast	true	cool	high	Y
$x2$	sunny	false	mild	high	N
$x3$	sunny	false	hot	high	N
$x4$	overcast	false	hot	normal	N
$x5$	sunny	true	hot	low	Y
$x6$	rainy	false	mild	high	N
$x7$	rainy	false	hot	high	N
$x8$	rainy	false	hot	normal	N
$x9$	overcast	true	hot	low	Y
$x10$	rainy	false	mild	normal	Y
$x11$	rainy	true	hot	normal	N
$x12$	rainy	false	hot	high	N

1. Em $U = \{x1..x12\}$ verifica-se que os elementos pertencem a classes diferentes;

3. Ainda existem atributos em $C = \{outlook, windy, temperature, humidity\}$;

5. $U/d = [[x1,x5,x9,x10],[x2,x3,x4,x6,x7,x8,x11,x12]]$

6. Para o conjunto C, temos os seguintes cálculos:

- $U/outlook = [[x1,x4,x9,],[x2,x3,x5],[x6,x7,x8,x10,x11,x12]]$ e $media_{outlook} = 0.5$
- $U/windy = [[x1,x5,x9,x11],[x2,x3,x4,x6,x7,x8,x10,x12]]$ e $media_{windy} = 0.667$
- $U/temperature = [[x1],[x2,x6,x10],[x3,x4,x5,x7,x8,x9,x11,x12]]$ e $media_{temperature} = 0.5$
- $U/humidity = [[x1,x2,x3,x6,x7,x12],[x4,x8,x10,x11],[x5,x9]]$ e $media_{humidity} = 0.667$

7. Como o valor de $media_{windy}$ e $media_{humidity}$ foram iguais, escolhe-se um deles. Nesse caso foi escolhido *humidity*.

8. Como o atributo *humidity* possui os valores $\{high, normal, low\}$, ele divide U em $U_{low} = [x5, x9]$, $U_{high} = [x1, x2, x3, x6, x7, x12]$ e $U_{normal} = [x4, x8, x10, x11]$. $C' = \{$*outlook*, *windy*, *temperature*$\}$.

Início da chamada recursiva # 1

1. Percebemos que U_{low} contém apenas objetos pertencentes à classe Y.

2.Assim, cria-se um nó folha contendo o valor 'Y'.

Fim da chamada recursiva # 1 e início da chamada recursiva # 2

1. Em $U_{high} = [x1, x2, x3, x6, x7, x12]$ existem objetos de classes diferentes;

3. Ainda existem atributos em $C = \{$*outlook*, *windy*, *temperature*$\}$;

5. $U_{high}/d = [[x1], [x2, x3, x6, x7, x12]]$

6. Para o conjunto C, temos os seguinte cálculos:

- $U/outlook = [[x1], [x2, x3], [x6, x7, x12]]$ e $media_{outlook} = 1$
- $U/windy = [[x1], [x2, x3, x6, x7, x12]]$ e $media_{windy} = 1$
- $U/temperature = [[x1], [x2, x6], [x3, x7, x12]]$ e $media_{temperature} = 1$

7.Com isso, percebemos que todos os atributos têm a média das pertinências aproximadas igual a 1. Assim, pode-se escolher qualquer um deles. Nesse caso, escolheu-se *outlook*.

8. O atributo *outlook* possui os valores $\{overcast, sunny, rainy\}$ e particiona U_{high} em $U_{high,overcast} = [x1]$, $U_{high,sunny} = [x2, x3]$ e $U_{high,rainy} = [x6, x7, x12]$. O valor de $media_{outlook}$ sendo 1, significa que o atributo induz conjuntos elementares totalmente definíveis nos conjuntos de decisão, ou seja, divide bem os objetos, agrupando-os sem que objetos de classes diferentes fiquem na mesma partição. $C' = \{windy, temperature\}$.

Início da chamada recursiva # 2-1

1. Em $U_{high,overcast} = [x1]$, todos os elementos pertencem à classe 'Y'.

2. Retorna um nó folha com o valor de 'Y'

Fim da chamada recursiva # 2-1 e início da chamada recursiva # 2-2

1. Em $U_{high,sunny} = [x2, x3]$, todos os elementos pertencem à classe 'N'.

2. Retorna um nó folha com o valor de 'N'

Fim da chamada recursiva # 2-2 e início da chamada recursiva # 2-3

1. Em $U_{high,rainy} = [x6, x7, x12]$, todos os elementos pertencem a classe 'N'.

2. Retorna um nó folha com o valor de 'N'

Fim da chamada recursiva # 2-3

Fim da chamada recursiva # 2 e início da chamada recursiva # 3

1. Em $U_{normal} = [x4, x8, x10, x11]$, existem elementos de classes diferentes

3. Ainda existem atributos em $C' = \{outlook, windy, temperature\}$

5. $U_{normal}/d = [[x4, x8, x11], [x10]]$

6.Para o conjunto C, temos os seguinte cálculos:

- $U/outlook$ = $[[x4], [x8, x10, x11]]$ e $media_{outlook} = 0.5$
- $U/windy$ = $[[x4, x8, x10], [x11]]$ e $media_{windy} = 0.5$
- $U/temperature$ = $[[x4, x8, x11], [x10]]$ e $media_{temperature} = 1$

7. Pode-se notar que o atributo *temperature* deve ser escolhido pois proporciona uma maior média de pertinências aproximadas.

8. O atributo *temperature*, em U_{normal} possui os valores $\{mild, hot\}$ e particiona U_{normal} em $U_{normal,mild} = [x10]$ e $U_{normal,hot} = [x4, x8, x11]$. O valor de $media_{temperature}$ sendo 1, significa que o atributo induz conjuntos elementares totalmente definíveis nos conjuntos de decisão, ou seja, divide bem os objetos, agrupando-os sem que objetos de classes diferentes fiquem na mesma partição. $C' = \{windy, temperature\}$.

Início da chamada recursiva # 3-1

1. Em $U_{normal,mild} = [x10]$, todos os elementos pertencem à classe 'Y'.

2. Retorna um nó folha com o valor de 'Y'

Fim da chamada recursiva # 3-1 e início da chamada recursiva # 3-2

1. Em $U_{normal,hot} = [x4, x8, x11]$, todos os elementos pertencem à classe 'N'.

2. Retorna um nó folha com o valor de 'N'

Fim da chamada recursiva # 3-2

Fim da chamada recursiva # 3 e da execução do algoritmo

Já não tem mais computações a fazer, termina-se o algoritmo. Depois dos passos acima, a árvore de decisão mostrada na Figura 19 é construída.

6.3 Metodologia de Testes e Detalhes de Implementação

Para verificar a qualidade do algoritmo proposto e a sua real contribuição, foram feitos testes utilizando algumas bases de dados contendo um número relevante de exemplos e atributos, que são capazes de mensurar bem a capacidade de aprendizado do algoritmo. Foram utilizadas duas metodologias de testes: uma com seleção aleatória dos dados de treinamento e classificação/teste e outra baseada em particionamento dos dados coletados.

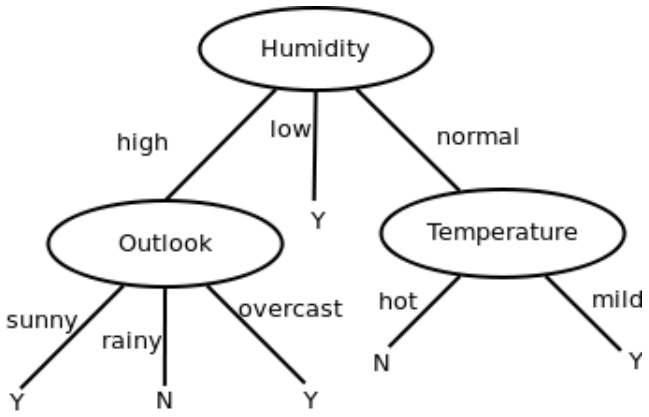


Figura 19: Árvore de decisão construída pelo algoritmo ID3-RMF

.

Na primeira metodologia, cada teste em si consistiu de uma bateria de dez testes, realizados com diferentes conjuntos de treinamento e classificação, extraídos a partir do conjunto original de dados. Nenhuma heurística foi utilizada para separar os dados para treinamento e teste, portanto não havia qualquer garantia de um elemento da base de teste possuir um representante na base de treinamento. Isso foi feito para verificar o comportamento do algoritmo em situações similares às reais.

A segunda metodologia adotada, denominada na literatura de validação cruzada, consiste em particionar os dados em k partições (ou pastas). Durante cada teste, uma pasta é selecionada para teste/classificação e as demais pastas foram utilizadas para treinamento do algoritmo. Com isso, tem-se que um dado utilizado para treinamento não é usado para teste/classificação, garantindo que os resultados do algoritmos não foram influenciados por uma classificação "viciada". No caso específico deste trabalho, foi utilizado um particionamento do conjunto original de dados em 10 pastas e também foi efetuada uma bateria de 10 testes, sendo que cada pasta era utilizada uma única vez para teste e nove vezes para treinamento. Para particionar a base de dados inicial foi utilizado o conceito de aritmética modular, fazendo com que o dado n ficasse na pasta $n \ mod \ 10$.

Com relação à implementação dos algoritmos, foi utilizada a linguagem de programação interpretada *Python*[3] e o interpretador *Python Interpreter* v2.6 (nativo do *Ubuntu Linux 10.04*[4]).

O computador utilizado para teste foi um *Dell Vostro 1510* com um processador *Intel Core 2 Duo T5670 1.8GHz Cache 2Mb*, 2Gb de Memória RAM *DDR2*, Disco *SATA* de 160Gb, *chipset nVidia GeForce 8400GS* 256Mb. Para os testes com a base *KDD99*, cujos resultados serão posteriormente apresentados, foram utilizados 8 computadores com a seguinte configuração: Processador *AMD Athlon 64 1800 Mhz*, 512MB de Memória RAM DDR2, Disco *SATA* de 80GB, *chipset nVidia*.

Para efetuar os testes, foi feita uma atualização no protótipo ILROS (*Inductive Learning using ROugh Sets*) apresentado em (UCHÔA, 1998). ILROS é um *software* que implementa os principais conceitos e medidas da TCA, além do algoritmo de aprendizado indutivo de máquina RS1+ descrito anteriormente. Tal protótipo foi implementado, inicialmente, utilizando a linguagem de programação C++ mas acabou ficando desatualizado e obsoleto.

Foi feito, neste trabalho, um *upgrade* nesse protótipo, que foi reimplementado na linguagem *Python*, para ficar compatível com a implementação dos algoritmos. O *design* da interface gráfica foi feito utilizando a biblioteca *pyGTK*[5], que é uma implementação da biblioteca GTK[6] para *Python*. Essa biblioteca foi escolhida por ser a base das interfaces gráficas dos sistemas operacionais utilizados pelos autores. Para auxiliar no desenho dos *layouts* das janelas, foi utilizado o *software Glade*[7].

Além da reimplementação do protótipo, acrescentou-se três algoritmos: o ID3, que ainda não estava presente e serve de base para este trabalho, o ID3-RMF aqui proposto e o ML-VPM, descrito na Seção 5.4. Foi implementada, também, a opção de escolher os atributos para indução do espaço aproximado, que não exis-

[3]Python: `http://www.python.org`
[4]Ubuntu Linux: `http://www.ubuntu.com`
[5]pyGTK: `http://www.pygtk.org`
[6]GTK: `http://www.gtk.org`
[7]Glade: `http://glade.gnome.org`

tia no ILROS original, mas já tinha sido proposto por Uchôa (1998). Por fim, foram acrescentadas algumas informações a respeito das regras geradas na tela de resultados para facilitar os testes neste trabalho.

7 Resultados e Discussão

7.1 Testes realizados com o algoritmo ID3-RMF

Para verificar a qualidade do algoritmo proposto nesse trabalho e sua real relevância perante aqueles consagrados na literatura, foram efetuados testes utilizando três bases, sendo duas delas reais. A seguir, estão expostos uma descrição sucinta dos arquivos de dados utilizados, os resultados obtidos e uma discussão comparativa dos mesmos.

7.1.1 *Cars*

A base *Cars*[8] consiste de um conjunto de dados descrevendo a aceitação de vários modelos de carros. A descrição de cada instância é feita através de 6 atributos discretos (buying, maint, doors, persons, lug-boot, safety, acceptability), 4 com valores nominais e 2 com valores numéricos. O conjunto original é formado por 1728 elementos, sendo que não há nenhum elemento com valor ausente.

Tabela 9: Resultados dos testes feitos na base *Cars* utilizando aleatoriamente 90% da base para treinamento e 10% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	272	270	290
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	6	6	6
Média de elementos no antecedente da regra	5.47	5.49	5.52
Grau de Suporte	87.20%	87.80%	87.20%
Regras não utilizadas na avaliação	223	229	246
Elementos não classificados	19	12	17
Tempo gasto para treinamento	0.091s	13.411s	0.802s
Tempo gasto para teste	0.176s	0.156s	0.172s

[8]Disponível em `http://archive.ics.uci.edu/ml/datasets/Car+Evaluation`

Tabela 10: Resultados dos testes feitos na base *Cars* utilizando aleatoriamente 75% da base para treinamento e 25% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	232	239	242
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	6	6	6
Média de elementos no antecedente da regra	5.37	5.41	5.40
Grau de Suporte	89.27%	89.01%	88.22%
Regras não utilizadas na avaliação	150	158	159
Elementos não classificados	25	25	23
Tempo gasto para treinamento	0.091s	9.741s	0.609s
Tempo gasto para teste	0.325s	0.275s	0.339s

Tabela 11: Resultados dos testes feitos na base *Cars* utilizando a metodologia de validação cruzada.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	267.4	272.2	283.8
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	6	6	6
Média de elementos no antecedente da regra	5.46	5.50	5.51
Grau de Suporte	88.89%	88.43%	87.09%
Regras não utilizadas na avaliação	217.9	224.2	235.7
Elementos não classificados	15.3	16.6	18.1
Tempo gasto para treinamento	0.096s	13.399s	0.805s
Tempo gasto para teste	0.168s	0.156s	0.197s

7.1.2 *Monkeys*

Os conjuntos de dados conhecidos como *Monk's Problems*[9] foram propostos como base da primeira comparação de algoritmos de aprendizado de máquina. Os *Monk's Problems* são três e 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 às 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. O domínio de todos eles, descrito a seguir, é o mesmo, e cada um deles possui 432 instâncias. Neste trabalho foram efetuados testes apenas com duas versões do problema, aqui referenciadas como *Monks1* e *Monks3*.

- *Monks1*

Tabela 12: Resultados dos testes feitos na base *Monks1* utilizando aleatoriamente 90% da base para treinamento e 10% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	72	28	46
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	5	3	5
Média de elementos no antecedente da regra	4.32	2.93	3.93
Grau de Suporte	92.30%	100%	100%
Regras não utilizadas na avaliação	46	8	21
Elementos não classificados	2	0	0
Tempo gasto para treinamento	0.02s	0.914s	0.06s
Tempo gasto para teste	0.01s	0.002s	0.008s

[9]Disponível em `http://archive.ics.uci.edu/ml/datasets/MONK's+Problems`

Tabela 13: Resultados dos testes feitos na base *Monks1* utilizando aleatoriamente 75% da base para treinamento e 25% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	84	37	37
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	6	3	4
Média de elementos no antecedente da regra	4.40	2.49	3.43
Grau de Suporte	95.88%	100%	100%
Regras não utilizadas na avaliação	39	0	0
Elementos não classificados	0	0	0
Tempo gasto para treinamento	0.016s	0.692s	0.047s
Tempo gasto para teste	0.02s	0.008s	0.007s

Tabela 14: Resultados dos testes feitos na base *Monks1* utilizando a metodologia de validação cruzada.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	90.4	28	44.2
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	6	3	4
Média de elementos no antecedente da regra	4.76	2.93	3.64
Grau de Suporte	94.46%	100%	100%
Regras não utilizadas na avaliação	60	0	15
Elementos não classificados	2.4	0	0
Tempo gasto para treinamento	0.019s	0.901s	0.06s
Tempo gasto para teste	0.013s	0.003s	0.007s

- *Monks3*

Tabela 15: Resultados dos testes feitos na base *Monks3* utilizando aleatoriamente 90% da base para treinamento e 10% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	12	12	14
Mínimo de elementos no antecedente da regra	1	1	2
Máximo de elementos no antecedente da regra	3	3	3
Média de elementos no antecedente da regra	2.17	2.17	2.21
Grau de Suporte	100%	100%	100%
Regras não utilizadas na avaliação	1	1	1
Elementos não classificados	0	0	0
Tempo gasto para treinamento	0.01s	0.616s	0.062s
Tempo gasto para teste	0.001s	0.02s	0.003s

Tabela 16: Resultados dos testes feitos na base *Monks3* utilizando aleatoriamente 75% da base para treinamento e 25% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	12	14	12
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	3	3	3
Média de elementos no antecedente da regra	2.17	1.79	2.17
Grau de Suporte	100%	100%	100%
Regras não utilizadas na avaliação	0	0	0
Elementos não classificados	0	0	0
Tempo gasto para treinamento	0.009s	0.471s	0.05s
Tempo gasto para teste	0.001s	0.003s	0.007s

Tabela 17: Resultados dos testes feitos na base *Monks3* utilizando a metodologia de validação cruzada.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	12	12	24.7
Mínimo de elementos no antecedente da regra	1	1	2
Máximo de elementos no antecedente da regra	3	3	4
Média de elementos no antecedente da regra	2.17	2.11	3.20
Grau de Suporte	100%	100%	99.77%
Regras não utilizadas na avaliação	0.6	0.6	8
Elementos não classificados	0	0	0
Tempo gasto para treinamento	0.01s	0.612s	0.063s
Tempo gasto para teste	0.002s	0.001s	0.003s

7.1.3 *Mushroom*

A base de dados *Mushroom*[10] consiste de um conjunto de dados com descrições hipotéticas de 23 espécies de cogumelos das famílias Agaricus e Lepiota. A descrição de cada instância utiliza-se de 22 atributos, todos com valores nominais; cogumelos são classificados em uma das duas classe: comestível ou venenoso. O conjunto original de dados possui 8124 instâncias, sendo que muitas delas possuem valores ausentes. Como o algoritmo não trata esse tipo de dado, escolheu-se um subconjunto do conjunto original contendo 5936 elementos, todos sem valores ausentes.

[10]Disponível em `http://archive.ics.uci.edu/ml/datasets/Mushroom`

Tabela 18: Resultados dos testes feitos na base *Mushroom* utilizando aleatoriamente 90% da base para treinamento e 10% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	15	15	15
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	3	3	3
Média de elementos no antecedente da regra	2	1.54	2
Grau de Suporte	100%	100%	100%
Regras não utilizadas na avaliação	2	2	2
Elementos não classificados	0	0	0
Tempo gasto para treinamento	1.154s	378.14s	26.399s
Tempo gasto para teste	0.031s	0.038s	0.037s

Tabela 19: Resultados dos testes feitos na base *Mushroom* utilizando aleatoriamente 75% da base para treinamento e 25% para teste.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	15	15	12
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	3	3	3
Média de elementos no antecedente da regra	2	1.54	2
Grau de Suporte	100%	100%	100%
Regras não utilizadas na avaliação	0	0	0
Elementos não classificados	0	0	0
Tempo gasto para treinamento	1.059s	306.94s	19.707s
Tempo gasto para teste	0.076s	0.075s	0.074s

Tabela 20: Resultados dos testes feitos na base *Mushroom* utilizando a metodologia validação cruzada.

	ID3	**RS1**	**ID3-RMF**
Número de regras geradas	15	15	12.9
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	3	3	3
Média de elementos no antecedente da regra	2	1.54	1.83
Grau de Suporte	100%	100%	100%
Regras não utilizadas na avaliação	0.8	0.8	0.3
Elementos não classificados	0	0	0
Tempo gasto para treinamento	1.177s	398.45s	25.847s
Tempo gasto para teste	0.038s	0.037s	0.035s

7.1.4 *KDD99*

A base de dados *KDD99*[11] é uma base de dados bastante utilizada para testes de algoritmos de aprendizado de máquina e ferramentas de detecção de intrusos. Ela inclui vários exemplos de ataques computacionais, encobertos em meio a tráfego normal de rede, incluindo uma grande variedade de intrusões simuladas em um ambiente de rede militar. Cada conexão é detalhada em termos de 41 características, representadas por atributos discretos e contínuos, e é classificada de acordo com o tipo de ataque ocorrido, caso exista, ou como uma conexão normal. Existem 22 tipos de ataques disponíveis para classificação.

A base *KDD99* possui mais de 5 milhões de registros. Para viabilizar o trabalho, utilizou-se um subconjunto contendo 494021 registros (10% da base original). Além disso, dividiu-se esse subconjunto em 80 partições e aplicou-se uma variação do teste do tipo validação cruzada, conforme ilustrado na Tabela 21. Em cada teste, foram utilizadas nove partições (55577 registros) para treinamento e uma partição (6176 registros) para teste, o que resulta em $1/8$ dos registros utilizados.

[11] Disponível em `http://kdd.ics.uci.edu`

Tabela 21: Pastas usadas para treinamento e teste com o KDD99.

Treinamento	Teste
9, 17, 25, 33, 41, 49, 57, 65, 73	1
10, 18, 26, 34, 42, 50, 58, 66, 74	2
11, 19, 27, 35, 43, 51, 60, 67, 75	3
12, 20, 28, 36, 44, 52, 61, 68, 76	4
14, 21, 29, 37, 45, 53, 62, 69, 77	5
15, 22, 30, 38, 46, 54, 63, 70, 78	6
16, 23, 31, 39, 47, 55, 64, 71, 79	7
17, 24, 32, 40, 48, 56, 65, 72, 80	8

Tabela 22: Resultados dos testes feitos na base *KDD99* utilizando a metodologia validação cruzada, conforme a organização apresentada na Tabela 21.

	ID3	RS1	ID3-RMF
Número de regras geradas	1773.3	5338.6	1842.6
Mínimo de elementos no antecedente da regra	1	1	1
Máximo de elementos no antecedente da regra	4	5	4
Média de elementos no antecedente da regra	1.47	2.03	1.48
Grau de Suporte	98.40%	94.32%	98.23%
Regras não utilizadas na avaliação	1303	4582	1396
Elementos não classificados	91.6	342.1	102.5
Tempo gasto para treinamento	967.16*s*	119157.98*s*	7665.34*s*
Tempo gasto para teste	39.44*s*	120.60*s*	40.75*s*

7.1.5 Discussão dos valores obtidos nos testes

É possível verificar que na base de dados *Monks1*, com classificação binária, o percentual de regras não utilizadas pelo algoritmo RS1 foi bem menor em todos os casos. Isso é justificado por outro resultado: "Média de elementos no antecedente das regras". Observa-se que o RS1 gera regras menores, mais simples e, consequentemente, mais gerais que os outros algoritmos. Isso aumenta a capacidade de generalização de conceitos, ou seja, o RS1 se mostrou mais capaz de aprender os conceitos nos dados treinamento e aplicá-los bem nos dados de testes. Já

nas outras bases com classificação binária, as diferenças são menores, mas o RS1 continua se sobressaindo em relação aos outros dois.

Mas tudo isso tem um preço: o tempo de execução do algoritmo RS1 é bem superior aos outros dois e a diferença se amplia com o aumento no número de exemplos na base de dados. Isso se dá pela constante necessidade de se induzir um espaço aproximado e calcular aproximações nesse espaço, que são tarefas custosas pois dependem do agrupamento dos elementos em conjuntos elementares e isso leva a uma análise dos valores dos atributos de interese em todos os exemplos da base.

Como visto anteriormente, o RS1 baseia-se nas aproximações inferiores dos conjuntos elementares induzidos pelo atributo de decisão, levando em consideração os espaços aproximados induzidos por um conjunto de atributos de condição. Como o atributo de decisão normalmente possui poucos valores possíveis (no casos de teste, são binários), então os conjuntos elementares induzidos por ele terão muitos elementos, ao passo que os atributos de condições são normalmente multi-valorados, induzindo mais conjuntos elementares com menos elementos em cada um deles. Assim, existirão muitas aproximações inferiores, resultando numa quantidade alta de situações a serem analisadas tanto no momento de gerar a regra quanto no momento de verificar a repetição das mesmas.

Já os outros algoritmos utilizam uma abordagem baseada em escolhas de atributos para construir a árvore de decisão e induzir o conhecimento. Como pode ser visto nas descrições das bases de dados, o número de atributos que caracterizam os exemplos é muito menor que o número de exemplos. Com isso, esses algoritmos vão, naturalmente, encontrar um número menor de situações a serem analisadas, visto que um atributo utilizado em um nó não pode se repetir nos seus descendentes, levando um tempo menor para a indução do conhecimento.

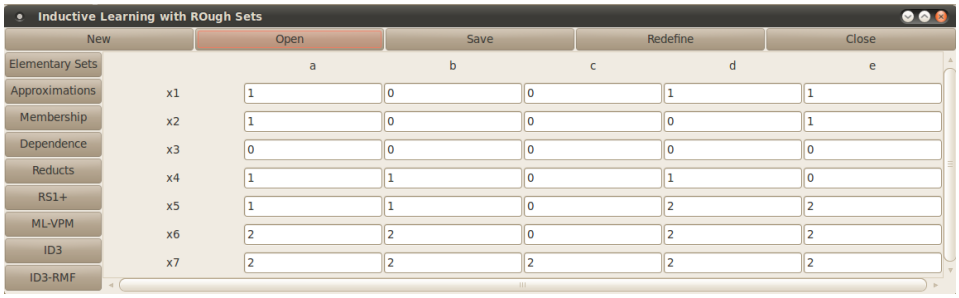
Entre aqueles que escolhem atributos para compor a árvore de decisão, o ID3-RMF se mostra um pouco mais lento que o ID3. Ambos possuem a mesma estrutura de execução, mas o cálculo de entropia do ID3 apenas “conta” elementos em um conjunto de exemplos, ao passo que o ID3-RMF utiliza o conceito de funções de pertinência aproximada, que faz uso de uma análise de interseções nos

cálculos. Porém, como também pode ser observado nos valores obtidos nos testes com a base *Monks1*, o algoritmo proposto neste trabalho superou o algoritmo ID3 clássico nos testes realizados, obtendo um grau de suporte maior, gerando regras menores e em uma quantidade inferior. Já na base *Car*, também com classificação binária, o ID3 clássico se comporta um pouco melhor, se equiparando ao RS1.

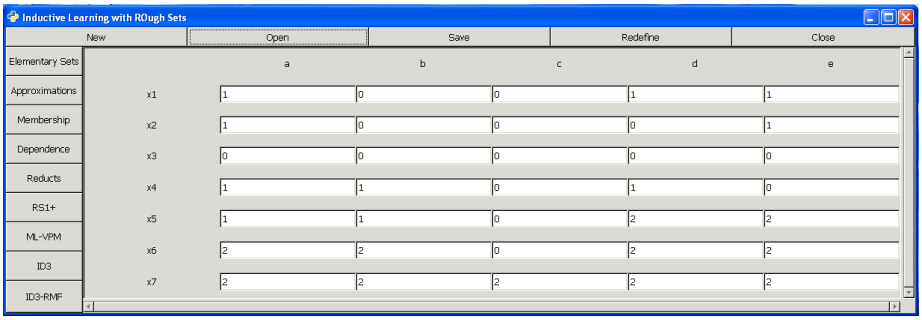
A base de dados *KDD99* é um caso especial: possui classificação não binária. Nos testes efetuados nela, o RS1 foi pior que os outros dois, gerando uma quantidade maior de regras supérfluas com um grau de suporte menor. Nesta base, o ID3-RMF, aqui proposto, se comportou de maneira equivalente ao ID3 clássico, com um alto grau de suporte e regras parecidas tanto em tamanho quanto em quantidade, mas com um tempo de treinamento superior.

7.2 Atualização do protótipo ILROS

Abaixo estão imagens de algumas telas do protótipo ILROS após a atualização utilizando a linguagem de programação *Python* e a biblioteca *pyGTK*.



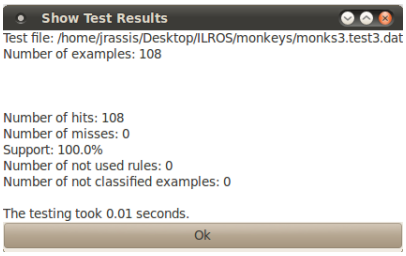
(a) Ambiente Ubuntu Linux 10.04



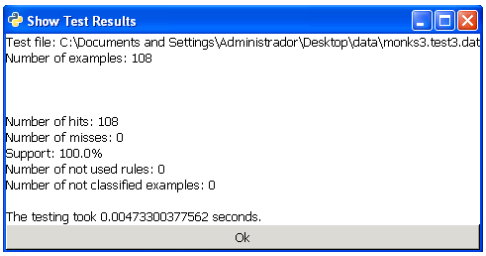
(b) Ambiente Microsoft Windows XP

Figura 20: Nova tela principal do ILROS

.



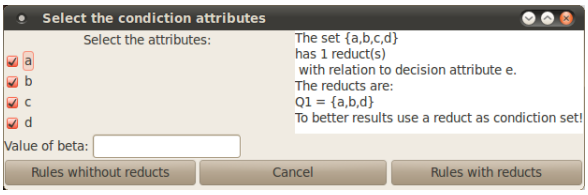
(a) Ambiente Ubuntu Linux 10.04



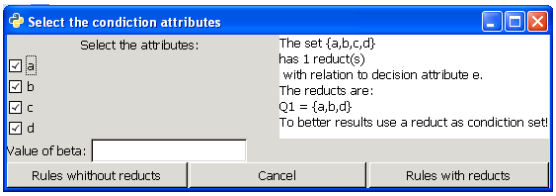
(b) Ambiente Microsoft Windows XP

Figura 21: Tela de resultados de um teste utilizando o ILROS

.



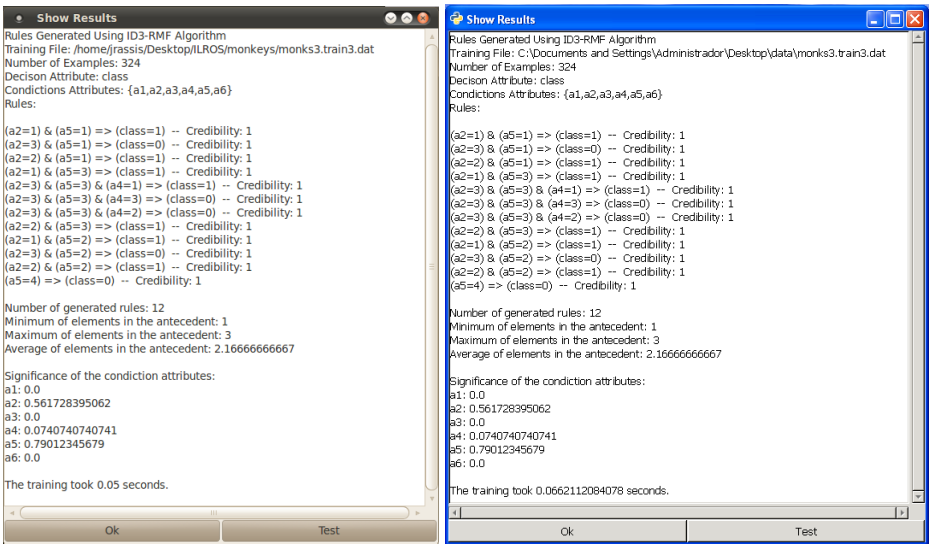
(a) Ambiente Ubuntu Linux 10.04



(b) Ambiente Microsoft Windows XP

Figura 22: Tela de escolha dos atributos de condição e valor do β para o algoritmo ML-VPM no ILROS

.



(a) Ambiente Ubuntu Linux 10.04 (b) Ambiente Microsoft Windows XP

Figura 23: Tela de resultados do treinamento utilizando o algoritmo ID3-RMF no ILROS

.

8 Conclusão

Neste trabalho foi proposto um novo algoritmo de aprendizado indutivo de máquina denominado ID3-RMF e inspirado na Teoria dos Conjuntos Aproximados. O ID3-RMF utiliza o conceito de função de pertinência aproximada para escolher aquele atributo que induz um espaço aproximado onde os conjuntos elementares induzidos pelo atributo de decisão são mais definíveis.

O algoritmo proposto se mostrou, nos testes efetuados, bastante estável em diferentes bases, com diferentes números de dados e atributos e diferentes tipos de classificação (binária e não binária), e com uma capacidade de aprendizado e generalização equivalentes a algoritmos já consagrados na literatura, como o ID3 clássico e o RS1. Com relação ao tempo de execução, foi um pouco mais lento que o ID3 clássico, devido ao gargalo das interseções entre conjuntos, porém muito mais rápido e igualmente eficiente ao o RS1.

Essa perda no tempo de execução para o ID3 pode ser compensada pelo fato de o ID3-RMF ser inspirado na Teoria de Conjuntos Aproximados, que é um formalismo matemático proposto não só para representação de incerteza mas também largamente utilizado para indução de conhecimento. Além disso, como visto no Capítulo 4, a função de pertinência aproximada é o elo de ligação entre a TCA e a Teoria de Conjuntos *Fuzzy*, que é outro formalismo consagrado e largamente utilizado para indução de conhecimento e representação de incerteza, possibilitando uma extensão do algoritmo ID3-RMF com conceitos da TCF de uma maneira relativamente simples e imediata.

Pode-se observar, portanto, que todos os três algoritmos tiveram graus de suporte muito parecidos nos três tipos de teste em cada base de dados, demonstrando, assim, a qualidade do algoritmo proposto.

A atualização do protótipo ILROS, utilizando para isso a linguagem *Python*, possibilitou a inserção de um software diferenciado e poderoso, que acabara ficando obsoleto, no contexto conteporâneo, uma vez que a linguagem *Python* é multiplataforma e flexível, possibilitando o desenvolvimento de interfaces *Web*, o que permitiria uma melhor distribuição e aplicação do software.

Por fim, este trabalho possibilitou a retomada das pesquisas em TCA na Universidade Federal de Lavras, que estavam estacionadas há alguns anos. Tal teoria, apesar de antiga (foi proposta no início da década de 80), ainda se mostra atualmente como um formalismo bastante promissor e eficiente não só para representação de incerteza mas também para indução de conhecimento, o que ficou claro neste trabalho e pode ser comprovado em Ding, Zheng e Zang (2009).

Referências

ANDRADE NETO, P. R. Descoberta de Conhecimento, Utilizando a Teoria de Conjuntos Aproximados, em SRCs com Informação Agregada. p. 45, 2002.

BONISSONE, P. Plausible reasoning. *Encyclopedia of Artificial Intelligence.*, p. 854–863, 1991.

BOOSE, J. H. Knowledge Acquisition for Knowledge-Based Systems: Notes on the State-of-the-Art. *Machine Learning*, v. 4, p. 377–394, 1989.

BOOSE, J. H. A knowledge acquisition: techniques and tools. *Knowledge Aquisition*, p. 3–37, 1989.

DING, B.; ZHENG, Y.; ZANG, S. A New Decision Tree Algorithm Based on Rough Set Theory. *Asia-Pacific Conference on Information Processing*, 2009.

DOMINGUES, M. A. Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incerteza com Ênfase na Teoria de Conjuntos Aproximados. p. 94, 2002.

FIGUEIREDO, M. G. B. Implementação e Desenvolvimento de Técnicas de Descoberta de Conhecimento e Tratamento de Incerteza com Ênfase na Teoria de Conjuntos Aproximados. p. 63, 2003.

HAYES-ROTH, F.; JACOBSTEIN, N. The State of Knowledge-Based System. *Communications of the ACM*, v. 37, n. 3, p. 27–39, mar. 1994.

KLIR, J. G.; YUAN, B. *Fuzzy sets and fuzzy logic: theory and applications.* [S.l.: s.n.], 1995.

MICHALSKI, R. S.; TECUCI, G. Multistrategy learning. *Tutorial T15, IJCAI-1993*, 1993.

NICOLETTI, M. C. *Ampliando os limites do aprendizado indutivo de máquina através das abordagens construtiva e relacional.* Tese (Doutorado) — Universidade de São Paulo, 1994.

NICOLETTI, M. C.; UCHÔA, J. Q. *O uso de funcoes de pertinencia na caracterização dos principais conceitos da teoria de conjuntos aproximados.* [S.l.], 1997.

NIWA, K. Toward Successful Implementation of Knowledge-Based Systems: Expert Systems vs. Knowledge Sharing Systems. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, v. 37, n. 4, 1990.

PAWLAK, Z. Rough sets. *International Journal of Computer and Information Sciences.*, p. 341–356, 1982.

PAWLAK, Z. Rough classification. *Internacional Journal of Man-Machine Studies.*, p. 469–483, 1984.

PAWLAK, Z. Rough sets and fuzzy sets. *Fuzzy Sets and Systems.*, p. 99–102, 1985.

PAWLAK, Z. *Rough sets: theoretical aspects of reasoning about data*. [S.l.: s.n.], 1991.

PAWLAK, Z. Rough sets, fuzzy sets and knowledge discovery. In: ____. [S.l.: s.n.], 1994. cap. Hard and soft sets., p. 130–135.

PAWLAK, Z. *Rough sets, rough relations and rough functions.* [S.l.], maio 1994.

QUINLAN, J. R. Induction of Decision Trees. *Machine Learning*, v. 1, p. 81–106, 1986.

SARKAR, M.; YEGNANARAYANA, B. Rough-Fuzzy Membership Ffnctions. *The IEEE International Conference on Fuzzy Systems Proceedings*, 1998.

SHAFER, G. *A mathemathical theory of evidence.* [S.l.: s.n.], 1976.

SHAW, M. J.; GENTRY, J. A. Inductive learning for risk classification. *IEEE Expert*, p. 47–53, fev. 1990.

SHORTLIFFE, E. H.; BUCHANAN, B. G. A Model of inexact reasoning in Medicine. *Math. Biosci.*, v. 23, p. 351–379, 1975.

UCHÔA, J. Q. *Representação e indução de conhecimento usando teoria de conjuntos aproximados*. Dissertação (Mestrado) — UFScar, 1998.

WANG, H.; MA, C.; ZHOU, L. A Brief Review of Machine Learning and its Application. *Information Engineering and Computer Science.*, 2009.

WONG, S. K. M.; ZIARKO, W. Comparison of rough-set and statistical methods in inductive learning. *Internacional Journal of Man-Machine Studies*, v. 24, p. 53–72, 1986.

WYGRALAK, M. Rough Sets and Fuzzy Sets - Some Remarks on Interrelations. *Fuzzy Sets and Systems.*, p. 241–243, 1989.

XUE, M.; ZHU, C. A Study and Application on Machine Learning of Artificial Intellligence. *International Joint Conference on Artificial Intelligence.*, 2009.

ZADEH, L. A. Fuzzt sets. *Information and control*, v. 8, p. 338–353, 1965.

ZIARKO, W. Variable Precision Rough Set Model. *Journal of Computer and System Sciences*, v. 46, p. 39–59, 1993.