UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RODRIGO BENTES KATO

CLASSIFICAÇÃO DE DADOS UTILIZANDO ALGORITMOS GENÉTICOS E LÓGICA DIFUSA

UFPA / ITEC / PPGEE
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM – PARÁ - BRASIL
2008

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RODRIGO BENTES KATO

CLASSIFICAÇÃO DE DADOS UTILIZANDO ALGORITMOS GENÉTICOS E LÓGICA DIFUSA

Dissertação apresentada para obtenção do grau de mestre em Engenharia Elétrica Ênfase em Computação Aplicada do Curso de Mestrado em Engenharia Elétrica da Universidade Federal do Pará – UFPa.

Orientador: Prof. Dr. Roberto Célio Limão de Oliveira.

UFPA / ITEC / PPGEE
CAMPUS UNIVERSITÁRIO DO GUAMÁ
BELÉM – PARÁ - BRASIL
2008

K19c Kato, Rodrigo Bentes

Classificação de dados utilizando algoritmos genéticos e lógica difusa / Rodrigo Bentes Kato; orientador, Roberto Célio Limão de Oliveira.-2008

Dissertação (Mestrado) – Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2008.

1. Exploração de dados (computação). 2. Algoritmos genéticos. 3. Sistemas difusos. I.Título.

CDD – 22. ed. 005.74

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RODRIGO BENTES KATO

CLASSIFICAÇÃO DE DADOS UTILIZANDO ALGORITMOS GENÉTICOS E LÓGICA FUZZY

DISSERTAÇÃO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ E JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA COM ÊNFASE EM COMPUTAÇÃO APLICADA.

DATA DA DEFESA:_____/_____/_____

CONCEITO:

BANCA EXAMINADORA:

_____
Prof. Dr. Roberto Célio Limão de Oliveira
(ORIENTADOR - UFPA)

_____
Francisco Edson Lopes da Rocha
(COORDENADOR DO PPGCC - UFPA)

_____
Adriana Rosa Garcez Castro
(PPGEE - UFPA)

VISTO: _____
Prof. Dr. Marcus Vinícius Alves Nunes
(COORDENADOR DO PPGEE - UFPA)

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que estiveram ao meu lado, que me auxiliam e ajudam no meu crescimento diário. Desta forma, agradeço à minha família que de forma direta ou indiretamente esteve presente em minha vida, incentivando-me em meus estudos e possibilitando que eu chegasse até aqui. Em especial a minha mãe "Marly", ao meu pai "Otávio", a minha irmã "Rayla", aos meus irmãos "Rogério" e "Ricardo", e sem esquecer é claro, de minhas vós "Eneide" e "Misako". E meus avôs "Moacyr" e "Arlindo" (*in memorian*).

Agradeço a minha namorada "Camila Alves" pelo incentivo, dedicação, compreensão e apoio, em mais uma etapa de minha vida.

Ao meu amigo de projeto Paulo Igor, que esteve ao meu lado no decorrer deste longo trabalho, onde juntos conseguimos tirar dúvidas e aprender bastante. Aos meus companheiros de mestrado e grandes amigos Ohashi, Marcelo, Renato, Denis, César, Rosivaldo, Noura, Johny, Vinícius, Douglas, Bianchi, Gustavo, Marçola e Neto.

Aos mestres da Universidade Federal do Pará e do Centro Universitário do Pará (CESUPA), pelos ensinamentos.

Agradeço toda a atenção a mim dispensada pelo meu orientador, Roberto Célio Limão de Oliveira, e minha co-orientadora, Aruanda Simões Gonçalves Meiguins, e também pelo incentivo e amizade. Espero que todos continuem a acreditar em nós alunos, mostrando que somos capazes.

“Ele (Deus) é o dono de tudo. Devo a Ele a oportunidade que tive de chegar aonde cheguei. Muitas pessoas têm essa capacidade, mas não têm a oportunidade. Ele a deu pra mim, não sei por quê. Só sei que não posso desperdiçá-la.”

(Ayrton Senna)

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE FÓRMULAS

LISTA DE SÍMBOLOS

EAs - Algoritmos Evolucionários.
AGs - Algoritmos Genéticos.
KDD - *Knowledge Discovery in Databases*.

RESUMO

Várias das técnicas tradicionais de Mineração de Dados têm sido aplicadas com êxito e outras esbarram em limitações, tanto no desempenho como na qualidade do conhecimento gerado. Pesquisas recentes têm demonstrado que as técnicas na área de IA, tais como Algoritmo Genético (AG) e Lógica Difusa (LD), podem ser utilizadas com sucesso. Nesta pesquisa o interesse é revisar algumas abordagens que utilizam AG em combinação com LD de forma híbrida para realizar busca em espaços grandes e complexos. Este trabalho apresenta o Algoritmo Genético (AG), utilizando Lógica Difusa, para a codificação, avaliação e reprodução dos cromossomos, buscando classificar dados através de regras extraídas de maneira automática com a evolução dos cromossomos. A Lógica Difusa é utilizada para deixar as regras mais claras e próximas da linguagem humana, utilizando representações lingüísticas para identificar dados contínuos.

Palavras Chaves: Algoritmo Genético, Mineração de Dados, Lógica Difusa.

ABSTRACT

Several of the traditional techniques of Data Mining have been applied successfully and others have some limitations. Both, in performance and the quality of knowledge generated. Recent research has shown that the techniques in the field of IA, such as GA and Fuzzy sets, can be used successfully. In this research we are interested in investigating the applicability of a hybrid combination of genetic algorithms and fuzzy sets to find rules in large and complex spaces. This paper presents a Genetic Algorithm (GA), using Fuzzy Logic, for coding, assessment and reproduction of chromosomes, looking for classifying data using extracted rules for the automatic way with the evolution of chromosomes. The Fuzzy Logic is used to make the rules clearer and closer to human language, using linguistic representations to identify continuous data.

Keywords: Genetic Algorithms, Data Mining, Logic Fuzzy.

1 INTRODUÇÃO

A mineração de dados consiste na extração do conhecimento em grandes bases de dados para a descoberta de regras interessantes e de padrões (HAM & KAMBER, 2001). O entendimento ou o aprendizado das informações implícitas encontradas nesses montantes de dados é de grande importância para o suporte a decisão ou aplicações técnicas.

Existem diversas tarefas de mineração de dados. Cada uma pode ser utilizada para um determinado tipo de problema. A classificação é provavelmente uma das tarefas mais estudadas. Ela consiste em analisar um conjunto de exemplos pré-classificados para desenvolver um modelo capaz de classificar uma população maior de registros (SEIXAS & GIMENES, 2003).

Os Algoritmos Evolucionários (AEs) atualmente representam uma área muito promissora de pesquisa em Inteligência Computacional devido à sua simplicidade e robustez. Os AEs têm como representante mais amplamente utilizado os Algoritmos Genéticos (AGs). Seu uso tem sido estendido a problemas de diversas áreas do conhecimento, geralmente problemas de otimização, que apresentam um espaço de busca muito grande para encontrar soluções apropriadas (RODRIGUES *et. al*., 2003).

O objetivo básico do Algoritmo Genético é formar um conjunto de estruturas, cromossomos, para representar uma determinada solução de um problema. Os AGs utilizam operações inspiradas na teoria de evolução das espécies e de seleção natural de Charles Darwin para modificar uma população de cromossomos gerando novas e melhores soluções a partir das anteriores (SANTOS *et. al.*, 2007).

A utilização de Algoritmo Genético com a Lógica Difusa tem como objetivo a criação de um algoritmo hibrido para facilitar a procura de dados em grandes espaços de busca e tornar o resultado mais próximo da linguagem humana.

O conceito formal de conjuntos difusos (*fuzzy*), fundamentado na Lógica Difusa (Lógica *Fuzzy*), foi introduzido por Lofti A. Zadeh em 1965 (COSENZA *et. al.*, 2006). No mundo real há muitas informações vagas, imprecisas, ambíguas ou de natureza probabilística e um dos motivos para se usar conjuntos difusos é que eles incorporam os termos lingüísticos, intrínsecos da linguagem natural, e o raciocínio aproximado, que os tornam flexíveis e

poderosos para tratar com incertezas a fim de montarem regras mais compreensíveis para o entendimento humano (SHAW e SIMÕES., 1999).

Os humanos quando pensam usam muito esse tipo de informação difusa. Humanos têm a capacidade de associar um grau a um determinado objeto sem compreender conscientemente como se chega a ele.

Por exemplo, quando se tem regras usando a idade de uma pessoa. Esse atributo possui limite muito brusco onde de uma hora para outra sua classificação muda. Hoje em dia uma pessoa é considerada idosa por volta de uns 65 anos, então aos 64 ela é jovem e quando atinge 65 se torna idosa de forma abrupta. Porém, determinar a idade dessa maneira causa algumas imprecisões. Por isso é usado as variáveis lingüísticas como idoso, adulto e jovem. Assim essas variáveis vão convergindo até que se torne 100% idoso à medida que a idade vai aumentando, com isso os valores se tornam mais flexíveis e precisos.

Apesar da existência de várias técnicas convencionais de mineração de dados para realizar a tarefa de classificação, as discussões acima indicam que a aplicação de Algoritmos Genéticos combinada com conjuntos difusos, é uma alternativa viável e promissora para essa importante tarefa (ROMÃO *et. al.*, 2004).

Uma das motivações desse trabalho é mostrar a interdisciplinaridade das áreas de mineração de dados, Algoritmos Evolucionários e Lógica Difusa, e assim apresentar como a Lógica Difusa combinada os Algoritmos Genéticos ajudam na formação das regras para a tarefa de mineração. Para isso é desenvolvida uma ferramenta que tem como objetivo a tarefa de extração de regras de classificação, onde se pretende extrair regras novas, compreensíveis e interessantes, de uma determinada base de dados.

O objetivo principal foi desenvolver um sistema que fosse capaz de realizar a extração de informações obscuras a partir de bases de dados complexas de forma automática, sem exigir a presença de um especialista técnico para configurá-los. A ferramenta recebe como entrada uma base de dados, que tem a indicação de quais atributos serão representados pela Lógica Difusa. O algoritmo monta as regras de classificação que posteriormente serão evoluídas usando o AG. Dessa maneira pretende-se extrair regras interessantes e implícitas da base de dados.

1.1 TRABALHOS RELACIONADOS

Existem vários trabalhos de mineração de dados alguns utilizando a técnica de Algoritmos Genéticos e outros a Lógica Difusa para mineração de dados. Abaixo são mostrados alguns exemplos.

O GA-MINER (GONÇALVES & ALVES, 2001) é uma ferramenta de mineração de dados, onde o conhecimento sobre um conjunto de dados pode ser adquirido de forma direcionada ou não direcionada. Sendo assim, o usuário tem a possibilidade de fazer perguntas específicas ou genéricas para a ferramenta, que utilizará como representação cláusulas que descrevem o subconjunto de dados.

O LS-1 (NOGUEIRA, 2004) é uma das ferramentas que seguem a linha de Pìttsburg (FREITAS, 2002), onde cada cromossomo é representado por um conjunto de regras. Este trabalho tem o seu funcionamento baseado em autômatos finitos, sendo que as regras de produção, que possuem a sua manipulação como característica básica, são evoluídas por um AG. A representação deste minerador é estruturada na semântica do domínio do problema.

TÚPAC et. al (1999) utilizam AG com Lógica Difusa para gerar um conjunto de regras de inferência para um controlador difuso. A ferramenta tem o objetivo de construir um modelo de controlador difuso capaz de apresentar o comportamento mais próximo do desejado.

A ferramenta KEEL (*Knowledge Extraction based on Evolutionary Learning*) tem como objetivo integrar os diversos modelos da extração do conhecimento dos dados usando Algoritmos Evolucionários. Dessa maneira, permite que se avaliem os modelos de aprendizagem evolucionária e assim, pode-se projetar novas metodologias ou melhorar as atuais (HERRERA *et. al.*, 2007).

A ferramenta KEEL modela e resolve problemas reais para a evolução do aprendizado genético. A qual permite utilizar e construir diferentes modelos para Mineração de Dados (HERRERA *et. al.*, 2004).

CAMPOS (2003) utiliza a extração utilizando computação evolutiva de regras associativas em um serviço de urgências psiquiátricas. O objetivo dessa pesquisa em problemas de urgências psiquiátricas é a extração de informações significativas e compreensíveis, representadas por regras de associação com o objetivo de melhorar a precisão nas predições obtidas.

As ferramentas GA-MINER e LS-1 têm objetivos semelhantes, que é a geração de regras usando um AG, porém ambas possuem representação e características diferenciadas. O GA-MINER, por exemplo, tem como característica o refinamento de hipóteses utilizando medidas estatísticas (NOGUEIRA, 2004). Já o LS-1 se utiliza de autômatos finitos.

A ferramenta apresentada nesse trabalho se diferencia das listadas acima, pois nela é usado um algoritmo híbrido onde é usado o AG e a Lógica Difusa para a extração da regras. Outro diferencial seria a implementação em linguagem orientada a objetos (Java).

1.2 ORGANIZAÇÃO DO TRABALHO

O trabalho está dividido em capítulos da seguinte forma. O Capítulo 2 apresenta uma breve introdução de mineração de dados, as etapas do processo de descoberta de conhecimento (KDD - *Knowledge Discovery in Databases*), além de descrever as principais tarefas e técnicas de mineração de dados.

No Capítulo 3, são apresentados os conceitos de Algoritmo Genético, seu mecanismo de funcionamento e seus principais componentes e operadores genéticos.

O Capítulo 4 apresenta uma introdução de Lógica Difusa e seus conceitos. Algumas das principais funções de pertinência também são apresentadas, além das operações, propriedades e modificadores em conjuntos difusos.

O Capítulo 5 apresenta algumas abordagens para a geração de regras difusas por um Algoritmo Genético, além dos tipos de codificação de indivíduos e determinação de conseqüente das regras.

O Capítulo 6 descreve a implementação da ferramenta através dos principais diagramas de modelagem baseados na UML. Ao final são apresentados os testes realizados com a ferramenta e os resultados obtidos.

O Capítulo 7 apresenta as considerações finais sobre o trabalho e os resultados obtidos com o protótipo, bem como os trabalhos futuros.

2 MINERAÇÃO DE DADOS

2.1 INTRODUÇÃO

Desde o século XX, com o surgimento da informática, uma grande quantidade de informação tem sido coletada e armazenada. A informação e o conhecimento obtidos podem ser utilizados para diversas aplicações, que vão do gerenciamento de negócios, controle de produção e análise de mercado ao projeto de engenharia e exploração científica (HAN & KAMBER, 2001). A capacidade de um indivíduo, ou empresa, de agir e de tomar decisões de forma ágil e precisa é freqüentemente associada ao conhecimento que possui sobre os dados em questão (MOTTA, 2006).

O processo de mineração de dados permite que se busquem, em grandes bases de dados, padrões que tenham valor para uma situação específica (NAVEGA, 2004). Por exemplo, em uma empresa que deseja aumentar a venda de telefones celulares, podem-se classificar as pessoas em duas categorias: as que já compraram pelo menos um telefone celular e pessoas que nunca compraram este tipo de aparelho.

No exemplo acima, uma ferramenta de mineração poderia utilizar os atributos da base de dados, chamados de preditivos, tais como classe social, estilo de vida, região demográfica, entre outros, para encontrar padrões e relações interessantes entre os dados disponíveis. Com este modelo, definir-se-ia um conjunto de treinamento contendo dados de compradores e não-compradores de celular. Em seguida se geraria a árvore de classificação, que é uma técnica utilizada na construção de modelos de análise e classificação de dados e possui esse nome devido a sua estrutura hierárquica que se assemelha com uma árvore invertida, e a partir daí a empresa poderia identificar novos clientes como potenciais compradores (ou não) de celular, formando uma base de clientes que receberiam, de tempos em tempos, mala-direta ou *mailing* sobre promoções de celulares.

Mineração de dados é a técnica que define o processo automatizado de captura e análise de conjuntos de dados visando à extração de informações ocultas potencialmente interessantes. Mineração de dados é um campo multidisciplinar que combina banco de dados, inteligência computacional e aprendizagem de máquina (HAN & KAMBER, 2001).

O propósito de minerar dados é facilitar o entendimento de grandes quantidades de dados descobrindo regras interessantes ou exceções (KLEMETTINEN *et. al.*, 2003). A principal idéia em mineração de dados consiste em extrair conhecimento de dados. Segundo (FAYYAD, 1996 apud NAVEGA, 2004), a definição de mineração de dados é:

> *"... o processo não-trivial de identificar, em dados, padrões válidos, novos, potencialmente úteis e ultimamente compreensíveis"*

Para se encontrar padrões, é necessário formatar os dados brutos de uma maneira a valorizar as características gerais. Isso se faz por não haver muito conhecimento a se extrair de eventos isolados. Como por exemplo, na compra de uma quantidade muito grande, em uma loja, de um determinado produto, em uma única data pode apenas significar que esse cliente em particular procurava uma grande quantidade desse produto naquele exato momento. Mas isso provavelmente não indica nenhuma tendência de mercado.

Depois da definição dos objetivos do processo de descoberta do conhecimento é então feita a determinação da tarefa de mineração mais adequada a ser usada. O objetivo do processo de descoberta do conhecimento é, portanto, a descoberta de padrões interessantes. A aplicação de um algoritmo de busca de padrões, mineração de dados, na base de dados é vista como o principal passo desse processo (FREITAS, 1999).

Podem-se considerar padrões como informações que se repetem, como por exemplo, na seqüência original: 123791230612453123125806123189. Aqui se percebe que as seqüências 123 e 12 se repetem mais do que os outros e então se pode fazer induções, que geram algumas representações genéricas: "123**" "124**" "125**" e "12***", onde * pode ser qualquer número.

2.2 ETAPAS DO PROCESSO DE DESCOBERTA DE DADOS

Na Figura 2.1 são mostradas as três grandes etapas do processo de descoberta do conhecimento em base de dados: pré-processamento, extração de padrões e pós-processamento. Porém foram incluídas mais uma fase nessas etapas, que é uma antes ao processo de mineração de dados que se refere à utilização do conhecimento obtido (REZENDE, 2003).

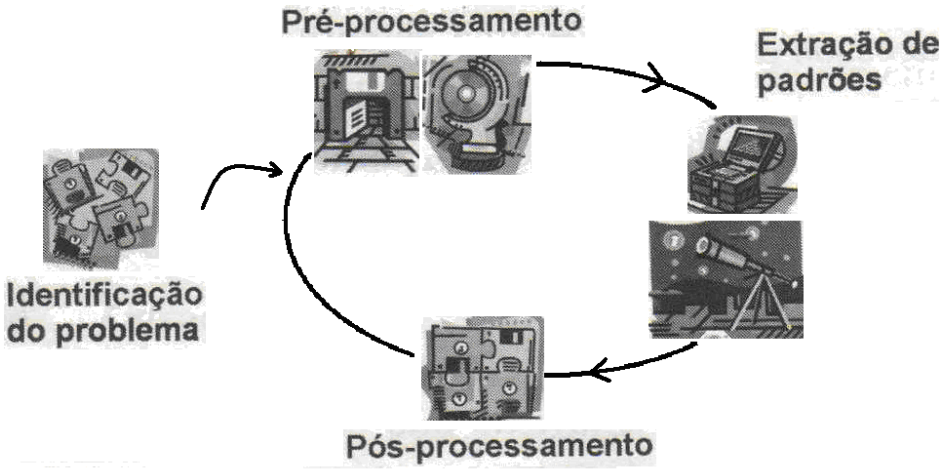

Figura 2.1: Etapas do Processo de Mineração de Dados.
Fonte: Solange, 2003 (Com adaptações)

2.2.1 Identificação do Problema

Nessa etapa são definidas as principais metas, objetivos e restrições. Com essas questões determinadas tem-se toda a ajuda necessária para as próximas etapas do processo de extração de conhecimento (REZENDE, 2003).

A preparação dos dados é essencial e tem fundamental importância para o sucesso da mineração de dados e varia de acordo com o algoritmo de mineração escolhido, costumando consumir mais de 50% do tempo e recursos destinados ao projeto. Os dados são selecionados, purificados e pré-processados (ARBEX *et al.*, 2004).

2.2.2 Pré-Processamento

O pré-processamento visa eliminar os dados que não interessam para a descoberta da informação, que podem ser campos omissos e dados anômalos (*outliers*) até a junção de variáveis ou linhas, combinação de campos e transformação de variáveis (DANTAS *et. al.*, *2003*). Nessa fase pode-se eliminar esse dado, chamado de ruído, ou estimá-lo, fazendo uma coleta das informações necessárias.

Os dados selecionados têm como principal objetivo selecionar um subconjunto de atributos relevantes para a tarefa alvo, dentre todos os atributos disponíveis. De maneira

simples, pode-se definir um atributo como relevante se ele é capaz de distinguir exemplos pertencentes a classes diferentes.

2.2.3 Extração de Padrões

De acordo com as informações é determinada qual das diversas tarefas serão usadas na mineração para localizar padrões nos dados (REZENDE, 2003). De acordo com a tarefa escolhida também se determina quais das técnicas são adequadas e o melhor ajuste dos parâmetros do algoritmo para a tarefa em questão. As tarefas e técnicas de mineração de dados são abordados em maior detalhe na seção 2.3.

Dependendo do algoritmo de mineração de dados em uso, os dados são formatados de maneiras diferentes. Com uma boa preparação dos dados evitam-se problemas em outras fases do processo.

2.2.4 Pós-Processamento

Nessa fase são feitas a verificação e a validação do conhecimento capturado. Segundo KAMBER (*apud* HAM, 2001) elas podem ser calculadas usando o chamado grau de suporte e o de confiança, que será explicado na tarefa de associação (seção 2.3.1.3)

O pós-processamento se dá pela utilização dos algoritmos ajustados em situações reais dos sistemas por meio do uso de um sistema inteligente ou um ser humano com apoio de alguns processos de tomada de decisão (REZENDE, 2003).

Nessa etapa verificam-se as informações adicionais que foram descobertas e a importância dos fatos gerados (ARBEX *et al.*, 2004). Dessa maneira, o conhecimento é incorporado à performance do sistema e reportado às partes interessadas. Porém nem todas as informações são importantes, somente uma pequena fração das informações pode ser interessante para os usuários.

Alguns exemplos de aplicação são em: supermercados, crédito em bancos (classificação), fraude em sistema de saúde, segurança (clonagem de celulares, acessos a páginas web para preços), anti-descoberta (militar, espionagem, proteção a testemunhas),

novas filiais (analisar amostras para ver sucessos e fracassos), mala-direta (perfil), entre outros.

2.3 TAREFAS DE MINERAÇÃO DE DADOS

O objetivo do processo de descoberta de conhecimento em bases de dados influencia na seleção do algoritmo de mineração de dados mais adequado. Em geral, as tarefas dividem-se em duas grandes categorias: descritivas e preditivas.

Existem diversas tarefas de mineração de dados, as quais podem ser usadas individualmente ou em conjunto, sendo que as principais são: modelagem de dependência, análise de agrupamento, regra de associação, classificação, regressão, detecção de desvios e padrões seqüenciais.

2.3.1 Modelagem Descritiva

Tem como objetivo apenas descrever dados de forma concisa. Ela pode ser usada para ajudar o entendimento do analista humano ou servir de passo preliminar para a mineração preditiva (ROBIN, 2004).

Nessa modelagem é utilizada a tarefa de agrupamento, modelagem de dependência, regra de associação, padrões seqüenciais e detecção de desvios.

2.3.1.1 Análise de Agrupamento ou *Clustering*

A proposta de análise de agrupamento é basicamente usada para resolver problemas de segmentação. A tarefa consiste em particionar os dados minerados em diversos grupos (*clusters*) de objetos, de tal modo que: (a) cada grupo possua objetos que são similares uns aos outros; e (b) os objetos de cada grupo são diferentes das amostras dos objetos pertencentes a outros grupos (FREITAS, 2002). Essa divisão em segmentos é realizada automaticamente por algoritmos que identificam padrões, características em comum.

Muitas vezes a análise de agrupamento é uma das primeiras etapas dentro de um processo de mineração de dados, já que identifica grupos de registros correlatos, que serão usados como ponto de partida para futuras explorações. O exemplo clássico é o de segmentação demográfica, que serve de início para a determinação das características de um grupo social, visando desde hábitos de compras até utilização de meios de transporte.

Devido não haver um conjunto de dados pré-definidos, faz-se necessário o uso de um método de aprendizado não-supervisionado onde existe maior autonomia do algoritmo.

2.3.1.2 Modelagem de Dependência

Em geral, modelagem de dependência consiste em descobrir dependências entre atributos (FREITAS, 2002). Modelos de dependências existem em dois níveis: estrutural e quantitativo.

O nível estrutural do modelo especifica quais variáveis são localmente dependentes de outras. Já o nível quantitativo do modelo especifica o grau das dependências através do uso de uma escala numérica. Normalmente, as dependências são modelos com probabilidades associadas, como mostram as redes de crenças (redes bayesianas) baseadas no raciocínio probabilístico.

2.3.1.3 Regras de Associação

Segundo AGRAWAL (2004) apud GONÇALVES (2001), as regras de associação são regras compostas por causa (antecedente) e efeito (conseqüente), e é um exemplo clássico de mineração de dados descritiva. Uma regra de associação na forma $X \rightarrow Y$ indica, de modo geral, que transações de um banco de dados que contêm um conjunto de itens de dados X tendem a conter também outro conjunto de itens de dados Y.

Na análise e planejamento de venda de produtos trabalha-se com um conjunto de itens dentro de uma simples transação. O objetivo é encontrar um grande número de tendências que podem ser usadas para compreender e explorar naturalmente padrões de compras.

Existem duas medidas associadas às regras de associação: o grau de confiança e o grau de suporte. O grau de suporte determina a probabilidade de que uma transação qualquer satisfaça tanto o antecedente quanto o conseqüente.

O grau de confiança indica a probabilidade de que uma transação satisfaça o antecedente, dado que ela satisfaz o conseqüente. Esses valores têm um mínimo que é estipulado pelo usuário nas ferramentas, chamado de mínimo de suporte e mínimo de confiança. A descoberta de regras de associação consiste em extrair dos dados minerados todas as regras com suporte e confiança maior ou igual ao limiar que o usuário especificou (FREITAS, 2002).

2.3.1.4 Padrões Seqüenciais

A seqüência é um tipo especial de associação (BRAUNER *et. al.*, 2003). Seqüências podem ser amplamente empregadas em sistemas médicos, sistemas meteorológicos, mala direta, supermercados, planejamento de inventários e planejamento de vendas de produtos, entre outros.

Há uma associação temporal nos fatos e, como nas regras de associação, existe um relacionamento de causa e efeito. A diferença é que nas regras seqüenciais os itens que se relacionam estão em transações diferenciadas, ao contrário das regras de associação onde os itens que se relacionam estão dentro da mesma transação.

Um exemplo para essa tarefa seria a aquisição do segundo volume de um livro, de forma que esta compra pode ter sido conseqüência direta do primeiro volume. Isto é, por alguma razão uma pessoa comprou o primeiro volume de um livro e, isto a levou a comprar o segundo volume.

2.3.1.5 Detecção de Desvios

A tarefa de detecção de desvios tem por objetivo descobrir um conjunto de itens de dado que não seguem padrões definidos de valores, ou seja, são exceções ou *outliers*. É necessária a criação de padrões de forma antecipada para descobrir este conjunto de desvios.

A relação entre a tarefa de agrupamento e a de detecção de desvios é que no agrupamento além do objetivo de encontrar grupos de dados com características semelhantes entre si, também tem o objetivo de descobrir grupo de dados diferentes entre os grupos, as exceções (SANTOS, 2006).

2.3.2 Modelagem Preditiva

A modelagem preditiva tem como objetivo prever dados. Existem dois tipos de predição: valores indisponíveis ou tendências pendentes e classes pré-classificadas para um determinado dado. Esta ultima está ligada à tarefa de classificação. Predição é, entretanto, mais referenciada à previsão de valores numéricos que estão faltando, ou acréscimo e decréscimo de tendências em dados temporais (POLITI, 2006).

Esta modelagem está dividida em duas tarefas de mineração: classificação e regressão.

2.3.2.1 Classificação

Na tarefa de classificação cada objeto de uma base de dados pertence a uma classe, que é indicada por um valor de um atributo. Este atributo é determinado utilizando valores numéricos pequenos, cada um correspondendo a uma classe. Em cada caso têm-se duas partes, um conjunto de valores previstos e um conjunto de classes previstas. O classificador é usado para prever os valores mais tarde (FREITAS, 2002).

Detecção de fraudes e aplicações de risco são exemplos de casos em que este tipo de análise é bastante apropriada. Em geral, algoritmos de classificação podem ser baseados em árvores de decisão, redes neurais ou Algoritmo Genético, e começam com um treinamento a partir de transações, por exemplo, pré-classificadas. O algoritmo classificador usa estes exemplos para determinar um conjunto de parâmetros, codificados em um modelo, que será mais tarde utilizado para a discriminação do restante dos dados.

Uma vez que o algoritmo classificador foi desenvolvido de forma eficiente, ele é usado de forma preditiva para classificar novos registros naquelas mesmas classes pré-definidas. Por exemplo, um classificador pode ser treinado a identificar empréstimos

arriscados, a partir das informações cadastrais de milhares de interessados, e usado como suporte à decisão no momento de conceder um empréstimo a alguém.

O uso de um conjunto de dados pré-determinados para a classificação caracteriza um método de aprendizado supervisionado, onde o algoritmo é controlado por parâmetros pré-classificados.

2.3.2.2 Regressão

Nessa tarefa é usada uma função de aprendizado que mapeia os dados com predição variável de valores discretos, sendo muito útil para modelos preditivos (ALVARES, 2004). Dado um conjunto de pontos, métodos de regressão calculam fórmulas capazes de fornecer pontos intermediários, anteriores ou posteriores.

Um exemplo de regressão é a construção de um modelo que calcule a probabilidade que a bolsa de valores estará em alta e assim prever quais os melhores dias ou as melhores ações para se investir.

2.4 TÉCNICAS DE MINERAÇÃO

Técnicas de mineração de dados são ferramentas ou algoritmos que perfaçam as tarefas previamente identificadas e solicitadas pelo minerador ou analista (BARRETO, 2006). Existem diversas técnicas de mineração de dados disponíveis na literatura, como por exemplo: Árvore de Decisão, Apriori, Redes Neurais e Algoritmos Genéticos.

2.4.1 Árvore de Decisão

A árvore de decisão é um classificador simbólico representado como estrutura de dados em árvore, onde cada nó interno indica o teste em um atributo, cada ramo representa um resultado do teste, e os nós terminais (folhas) representam classes ou distribuições de classe. O topo da árvore é a sua raiz (MOTTA, 2006).

O algoritmo de treinamento constrói a árvore de decisão recursivamente, de cima para baixo, identificando o atributo mais importante (atributo divisor ou de teste), isto é, aquele que faz a maior diferença para a classificação das amostras disponíveis (atributo que possui o maior ganho de informação) (MOTTA, 2006).

2.4.2 Apriori

O algoritmo Apriori é um dos mais conhecidos quando o assunto é mineração de regras de associação em grandes bancos de dados centralizados. Ele encontra todos os conjuntos de itens freqüentes, denominados *frequent itemsets*.

O algoritmo Apriori pode ser dividido em duas etapas. A primeira etapa realiza a contagem de ocorrências dos itens para determinar os itens freqüentes de tamanho unitário (l-item freqüente). Na segunda iteração, 2- itens freqüentes (conjuntos com dois itens) candidatos são gerados pela junção do 1- item freqüente e seus suportes são determinados pela pesquisa no banco de dados. Assim, o algoritmo, apresentado na Figura 2.2, prossegue iterativamente, até que o *k*- itens freqüentes encontrado seja um conjunto vazio (BORGELT & KRUSE, 2002).

```
Procedimento Apriori
  L1 = {1 – item freqüente};
  para (k = 2; Lk-1 ≠ ∅; k++) faça
    Ck = apriori_gen (Lk-1);
    para todas as transações t ∈ D faça
      Ct = subconjunto (Ck, t);
      para todos os candidatos c ∈ Ct faça
        c.contador++;
      fim
    fim
    Lk = { c ∈ Ck | c.contador >= min_sup};
  fim
  Resposta = UkLk;
fim
```

Figura 2.2: Algoritmo Apriori.
Fonte: BORGELT & RUSE, 2002.

2.4.3 Redes Neurais

As redes neurais podem ser usadas em mineração de dados para resolver problemas de classificação, aproximação e agrupamento.

Em problemas de classificação o objetivo é atribuir um padrão de entrada a uma classe entre um conjunto de classes já conhecidas. Em problemas de aproximação os dados são fornecidos dentro de um determinado limite, onde a função é definida e o modelo neural é ajustado para fornecer uma boa aproximação. Em problemas de análise de agrupamento tenta-se descobrir características estatisticamente relevantes em um determinado conjunto de dados e como dividi-los em classes. Neste problema, somente os dados de entrada são conhecidos, cabe a rede determinar as classes presentes nesse conjunto de dados (THOMÉ, 2005).

2.4.4 Algoritmos Genéticos

Algoritmos Genéticos são usados em mineração de dados para formar regras mais compreensíveis e interessantes. Os AGs são técnicas de busca inspiradas na teoria da evolução das espécies de Charles Darwin (MURAKAMI & BRINATI, 2004).

A implementação de um Algoritmo Genético começa com uma população aleatória de cromossomos. Essas estruturas são avaliadas através de uma *fitness* (função de avaliação). Essa avaliação é feita de forma que os maiores valores são associados aos cromossomos que representam uma melhor solução para o problema. A *fitness* da solução é definida com relação à população corrente (MIRANDA, 2003).

A principal vantagem de se extrair regras por Algoritmo Genético, quando comparadas com outros métodos (redes neurais artificiais, sistemas neuro-difusos, modelos estatísticos), é que as regras evoluídas são auto-explicativas, bem detalhadas. O modelo genético é capaz de encontrar regras com muita precisão e abrangentes.

2.5 CONSIDERAÇÕES FINAIS

Neste capitulo foi apresentado um breve conceito sobre mineração de dados com a descrição das principais técnicas e tarefas.

Os métodos mostrados são os mais amplamente utilizados, e servem para resolver grande parte dos problemas primários na mineração, porém na literatura existem diversas tarefas e técnicas que são mais complexas.

A ferramenta apresentada no trabalho utiliza o AG na tarefa de classificação, com o objetivo de encontrar as melhores regras da base selecionada e junto com a Lógica Difusa para tornar as regras encontradas mais compreensíveis para os seres humanos.

3 ALGORITMO GENÉTICO

3.1 INTRODUÇÃO

Um Algoritmo Genético (AG) simula a evolução e a seleção de indivíduos baseados no princípio do mais adaptado segundo a teoria proposta por Charles Darwin em 1859. O AG incorpora uma solução potencial para um problema específico numa estrutura semelhante à de um cromossomo e aplica operadores de seleção e operadores genéticos como mutação e cruzamento a essas estruturas. Essa evolução baseia-se na idéia de que um processo evolutivo ocorre quando quatro condições são satisfeitas (KOZA, 1992):

- Um indivíduo tem habilidade de reproduzir a si mesmo;
- Há uma população desses indivíduos;
- Há variedade entre os indivíduos dessa população; e
- Uma diferença na habilidade de sobrevivência no ambiente está associada com essa variedade.

Na natureza, indivíduos possuem diferentes estruturas de cromossomo que formam a população. Cromossomos são formados por seqüências de símbolos de um alfabeto que, na natureza, tem quatro elementos associados às moléculas: *adenina* (A), *citosina* (C), *guanina* (G) e *timina* (T). As diferentes estruturas nos cromossomos indicam as diferenças na taxa de sobrevivência e reprodução. Sendo assim, indivíduos que possuem facilidade em certas tarefas podem ser considerados mais aptos nesse ambiente, sobrevivendo e se reproduzindo em taxas maiores do que os menos aptos.

Ao longo de gerações são criados novos indivíduos numa população. Indivíduos menos aptos tendem a ser eliminados, e novos indivíduos são criados. Com o tempo, a população passa a ter indivíduos cujos cromossomos refletem maior habilidade para executar determinadas tarefas, sobreviver e se reproduzir. Essas são características de um processo de seleção de indivíduos ao longo de gerações.

Sendo assim, a estrutura e comportamento dos indivíduos numa população se alteram ao longo do tempo por causa de um processo de seleção natural. Devido a essa diversidade da população o algoritmo tende a evoluir, criando indivíduos mais aptos.

Normalmente os Algoritmos Genéticos são vistos como otimizadores de funções, porém existe uma grande quantidade de problemas em que se podem aplicar os algoritmos genéticos. Alguns exemplos são: a organização de horários escolares, a criação de regras de classificação, entre outros. Esses algoritmos codificam uma solução em potencial, para um problema específico, em uma estrutura de dados chamada de cromossomos. Depois se aplicam diversos operadores à procura de um cromossomo mais apto, ou seja, uma solução melhor para o problema. Porém a mudança do cromossomo é realizada de uma forma que preserve sua informação crítica, chamada de bloco construtor. A melhor solução encontrada na população no final das iterações é considerada como a solução final do problema dada pelo algoritmo.

3.2 CONCEITOS BÁSICOS

Segundo FOGEL (2000), nas últimas décadas surgiu pelo menos três alternativas de simulação da evolução, que mais tarde ficaram conhecidas como Algoritmos Genéticos. Este método foi primeiro apresentado por (FRAZEN, 1957a, b, 1960, 1962, 1967, 1968; BREMERMANN, 1962; BREMERMANN *et. al.*, 1966) e por John Holland e seus estudantes da Universidade de Michigan (BAGLEY, 1967; ROSENBERG, 1967; HOLLAND, 1969, 1973).

Resultado dos primeiros estudos de John Holland na década de 1960, e posteriormente aperfeiçoado por seus alunos, uma abstração da evolução biológica chamada de Algoritmo Genético foi apresentada por Holland em 1975 no seu livro "*Adaptation in Natural and Artificial Systems*". O elemento essencial de um AG é população baseada em variações e seleções randômicas de indivíduos (FOGEL, 2000).

Algoritmos Genéticos são inicializados primeiramente com uma população de cromossomos ou indivíduos, que em geral são produzidos aleatoriamente. Esses cromossomos são avaliados e recebem valores de *fitness* para aqueles que apresentem uma melhor solução para o problema alvo tenham maiores probabilidades de ser reproduzidos do que os cromossomos que representam soluções fracas. Depois são aplicados os operadores de seleção, cruzamento e mutação e assim é completado um ciclo do algoritmo.

Os princípios de seleção e reprodução são aplicados, a cada ciclo, a uma população de cromossomos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis (SALVADOR, 2003). A Figura 3.1 mostra como funcionam os Algoritmos Genéticos. O primeiro passo é a formação da população inicial, em seguida é calculada a função objetivo (*fitness*) dos indivíduos da população. Os passos seguintes são as aplicações dos operadores de seleção, cruzamento e mutação nos indivíduos para dar-se a evolução. Ao final desses passos vem o critério de parada do algoritmo, onde é feita a pergunta se já foi encontrada a solução. Caso a resposta seja sim, o AG chega ao fim. Se o resposta for não, o algoritmo voltará para o cálculo da *fitness* desses novos indivíduos evoluídos e aplicar neles os operadores de seleção, cruzamento e mutação, isso se repetirá até que seja encontrada a solução para o problema.

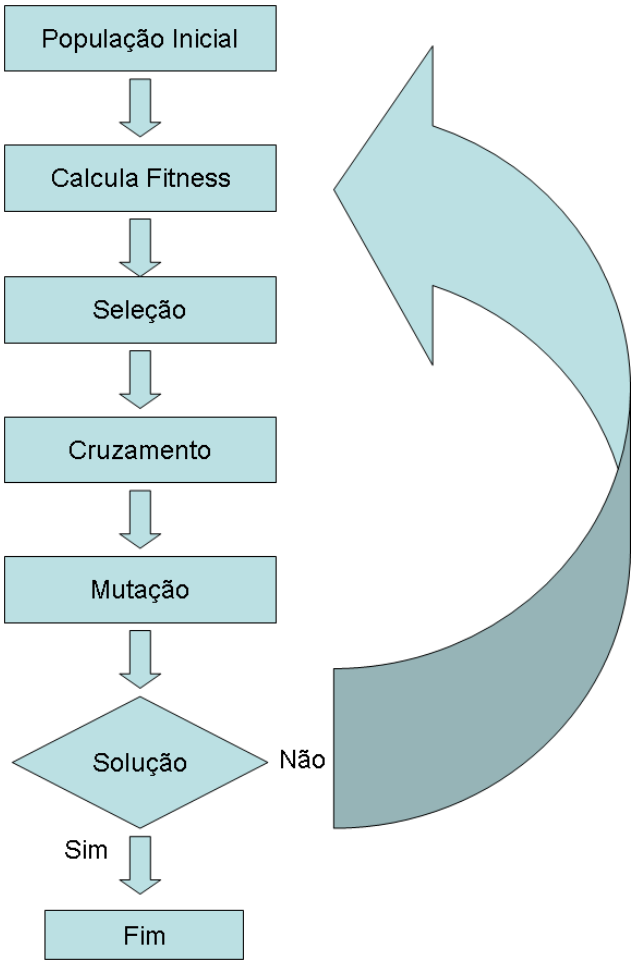

Figura 3.1: Fluxograma de um Algoritmo Genético.

Há várias implementações de AGs. Pode-se, por exemplo, utilizar diferentes operadores de seleção, diferentes tipos de operadores de cruzamento e mutação.

Segundo (FOGEL, 2000) Algoritmos Genéticos são na maioria das vezes representados pela forma canônica como pode ser visto a seguir:

1) Os candidatos à população são iniciados com algumas restrições. Cada candidato é geralmente codificado em um vetor *x*, chamado de cromossomo, com elementos chamados genes e suas posições, chamadas alelos.

2) É definida e escolhida uma *fitness*, que segundo HARTL & CLARK (1989) apud FOGEL (2000) é definida como a habilidade de um indivíduo sobreviver e reproduzir-se em um ambiente.

3) Cada cromossomo, x_i, $i = 1, 2, ..., P$, da população é codificado em uma forma apropriada para a evolução e então recebe um valor de *fitness*.

4) Cada cromossomo recebe uma probabilidade de cruzamento, p_i, $i = 1, 2, ..., P$, com valor proporcional a sua *fitness* em relação à população.

5) Os cromossomos selecionados, por meio do operador seleção, geram descendentes via operadores genéticos, como cruzamento ou de mutação.

6) O processo é terminado se a solução satisfatória for encontrada. Caso contrário, o algoritmo retorna para o passo 3, onde novos cromossomos são avaliados e novas gerações são criadas.

3.3 INICIALIZAÇÃO DOS ALGORITMOS GENÉTICOS

O primeiro passo em um AG é a codificação das variáveis de busca em um cromossomo (NETO, 2003). A maneira como as soluções são representadas é de fundamental importância para o sucesso dos Algoritmos Evolucionários. A forma de representação pode variar de muitas maneiras e basicamente depende do tipo de problema tratado. De modo geral, as formas de representação que têm recebido mais atenção recentemente são aquelas que consideram características específicas do problema.

Um elemento crucial nos estudos dos AGs é o conceito de esquema (GOLDBERG, 1989). Enquanto uma estrutura que representa um cromossomo é uma seqüência definida sobre um alfabeto *A*, um esquema é uma seqüência de mesmo comprimento, mas definida

sobre o alfabeto $A \in \{*\}$, onde o símbolo * significa *do-not-care* (indiferente). Segundo HOLLAND (1975) apud FOGEL (2000), sugere-se que cada cromossomo evoluído oferece informação parcial sobre uma *fitness* esperada de todos os possíveis esquemas que o cromossomo possui. Esta característica é chamada de paralelismo implícito que, através de um simples exemplo, representa o ganho de informação devido aos vários esquemas.

No próximo passo, para iniciar um processo evolutivo, deve-se gerar uma população inicial. A população inicial pode ser gerada tanto aleatoriamente, para se ter uma "biodiversidade" da população para assim poder ter uma boa abrangência no espaço de busca, quanto através de outro método específico para tentar gerar indivíduos com alguma qualidade. O número de indivíduos dessa população, que em muitos algoritmos se mantém fixo durante todo o processo evolutivo, é considerado um parâmetro desses algoritmos e sua determinação é feita de modo empírico.

O passo seguinte é determinar a *fitness*, o primeiro passo para a seleção. A formulação de uma função que associe as estruturas da população a valores numéricos que são usados para avaliar, indivíduo a indivíduo. Sua aptidão é feita de acordo com as características de cada problema abordado. Com isso indivíduos mais aptos devem ter maior chance de ser selecionados. A formulação dessa função reflete os objetivos de otimização do problema.

3.4 OPERADORES DE SELEÇÃO

O processo de seleção pode ser feito de várias maneiras, mas normalmente as estruturas são selecionadas para a formação de uma nova população de modo proporcional à sua adaptação. Estruturas mais aptas são selecionadas com mais freqüência do que as menos adaptadas para assim favorecer a reprodução de indivíduos com melhor *fitness* (NETO, 2003).

A técnica denominada seleção por roleta foi formulada por John Holland em 1975. A probabilidade de um indivíduo ser selecionado é proporcional a sua *fitness*. Depois dos cálculos de *fitness*, os indivíduos são posicionados na roleta, de acordo com sua probabilidade, então a mesma é girada e escolhido um indivíduo. Essa é uma das técnicas mais usadas em AGs. A Figura 3.2 mostra um exemplo dessa técnica.

N	Cromossomo	Fitness	%
1	000110010	6	50
2	010100101	3	25
3	101111010	1,5	15
4	101001010	1,5	15

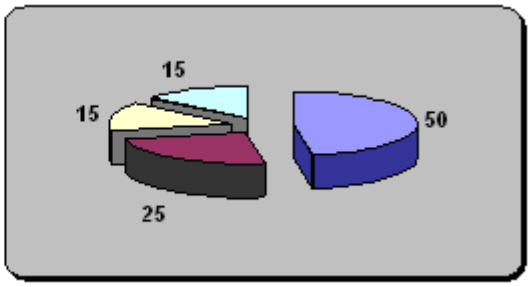

Figura 3.2: Seleção por Roleta.

Na figura observam-se quatro cromossomos e suas *fitness* correspondentes. A porcentagem que está apresentada na coluna ao lado serve para indicar o quanto de espaço que esse cromossomo vai ocupar na roleta, como ilustrado no gráfico ao lado. E assim, pode-se verificar que o cromossomo de número 1 tem maior probabilidade de ser escolhido entre os outros três.

Com o passar do tempo, novos mecanismos foram propostos. Na chamada seleção por torneio, apresentada por David Goldberg em 1991, duas ou mais estruturas são escolhidas aleatoriamente na população e, de acordo com sua probabilidade, o indivíduo mais adaptado será selecionado para reprodução.

Os melhores cromossomos de uma população podem ser perdidos através das gerações, devido o algoritmo não guardá-los, e assim acaba cruzando com outros (FOGEL, 2000). Segundo GREFENSTETTE (1986) apud FOGEL (2000) isto pode ser resolvido com o emprego da heurística chamada elitismo, em que sempre se mantém o melhor cromossomo da população.

Com isso o resultado da busca geralmente se torna mais rápido e bastante efetivo. Entretanto usando o elitismo pode-se chegar a uma convergência prematura que tem sido uma grande preocupação dos Algoritmos Genéticos (FOGEL, 2000). Esse problema ocorre quando a população de cromossomos alcança uma configuração tal que o cruzamento já não produz descendentes que tenham resultados melhores que seus pais (FOGEL, 2000).

3.5 OPERADORES DE CRUZAMENTO

A essência do operador de cruzamento é a troca do material genético entre dois indivíduos, que são chamados de pais (FREITAS, 2002). A reprodução dos indivíduos selecionados pode ser feita com uma simples cópia (clonagem) inserida na nova população sem nenhuma modificação, ou com associações a outras estruturas (cruzamento). Existem diversos maneiras de se fazer o cruzamento. Nessas associações, dois indivíduos selecionados devem produzir dois novos indivíduos.

Na primeira, dados dois cromossomos de comprimento l que são chamados de pais, sorteia-se um número p (ponto de cruzamento) tal que $0 < p < l$ para produzirem dois filhos. O primeiro filho receberá os genes do primeiro pai de zero até p e todos os genes do segundo pai de p até l, e o segundo filho receberá os genes de segundo pai de zero até p e do primeiro pai de p até l (LUCAS *et al.*, 2003). Na Figura 3.3, pode-se ver um exemplo do cruzamento de um ponto (1PX).

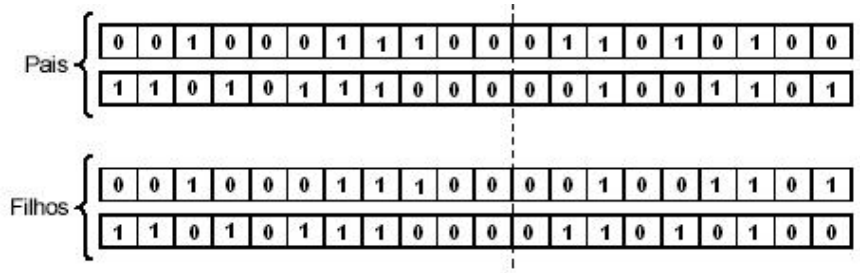

Figura 3.3: Cruzamento de um Ponto (1PX).

Semelhante ao cruzamento de um ponto, também é comumente utilizado o cruzamento de dois pontos (2PX), onde são escolhidos dessa vez dois pontos de corte. A Figura 3.4 mostra um exemplo desse cruzamento.

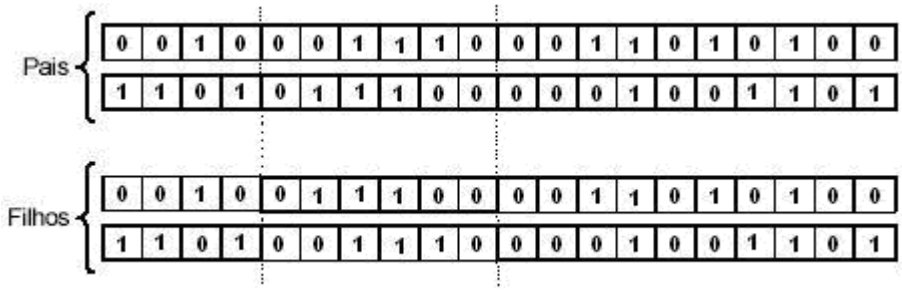

Figura 3.4: Cruzamento de dois Pontos (2PX).

O cruzamento multi-ponto (MX) é uma generalização dos operadores apresentados acima. Neste são sorteados um número fixo *n* de pontos de corte (LUCAS *et. al.*, 2003).

No cruzamento segmentado (SX), também é sorteado um número fixo *n*, mas ao contrário do multi-ponto é sorteado um número de pontos de corte toda vez que é executado (LUCAS *et. al.*, 2003).

No cruzamento uniforme (UX), cada posição do gene dos cromossomos, dos pais, são trocados numa posição fixa e assim formam seus descendentes (FREITAS, 2002). A Figura 3.5 ilustra o procedimento. Nesse exemplo foram escolhidos, aleatoriamente, os genes 2, 4 e 5 para o cruzamento. Ao lado é verificada a mudança, em suas respectivas posições, da carga genética dos indivíduos, onde o indivíduo X recebe os genes do Y e Y recebe o de X.

X1	X2	X3	X4	X5	X6
Y1	Y2	Y3	Y4	Y5	Y6

(a) Antes do cruzamento.

X1	Y2	X3	Y4	Y5	X6
Y1	X2	Y3	X4	X5	Y6

(b) Depois do cruzamento.

Figura 3.5: Cruzamento Uniforme (FREITAS, 2002).

3.6 OPERADOR DE MUTAÇÃO

O operador de mutação é um procedimento dedicado à manutenção da diversidade da população (ARROYO & ARMENTANO, 2001).

A mutação consiste em selecionar um indivíduo e mudar certas características do mesmo, dessa maneira produzindo um novo indivíduo para a população. Os procedimentos de mutação podem variar bastante e dependem das formas de representação escolhidas. A mutação pode ser aplicada *bit* a *bit* no cromossomo ou uma simples cópia dos descendentes.

A escolha de qual individuo vai sofrer ou não a mutação é feita de acordo com probabilidades que são parâmetros do AG. O papel da mutação é garantir a heterogeneidade da população ao longo das gerações.

A mutação em AG é um operador que age em um indivíduo e é normalmente aplicada com uma probabilidade relativamente baixa (FREITAS, 2002).

3.7 CONSIDERAÇÕES FINAIS

Neste Capítulo foram apresentados os princípios básicos de Algoritmos Evolucionários, tais como, os operadores de seleção, operadores de cruzamento e mutação.

As diferentes etapas do AG podem ser aplicadas de diferentes maneiras que dependendo do tipo de seu problema. Por exemplo, há várias técnicas de cruzamento, seleção e de mutação a serem usados. O Quadro 3.1 apresenta um resumo das principais etapas do AG abordadas nesse capítulo.

Quadro 3.1: Resumo das etapas do AG

Cruzamento	Seleção	Mutação
Um ponto de corte (1PX).	Roleta – indivíduo selecionado com chance proporcional à sua aptidão.	Bit a bit – cada elemento do indivíduo é trocado.
Dois pontos de corte (2PX).	Torneio – indivíduo mais apto dentre os sorteados é selecionado.	Clonagem – cópia do indivíduo.
Troca da posição fixa dos genes (UX).		
n pontos de corte toda a vez que é executado (SX).		
n de pontos de corte (MX).		

Na ferramenta desse projeto foi usado o operador de cruzamento com um ponto de corte (1PX), o método de seleção escolhido foi a roleta, o operador de mutação foi o bit a bit e ainda foi adotado o emprego do elitismo para serem guardadas as melhores regras.

A questão da codificação utilizada para o AG desenvolvido é a codificação de alto nível onde será explicado no capítulo 6, onde é apresentada a implementação e as discussões sobre a ferramenta.

4 LÓGICA DIFUSA

4.1 INTRODUÇÃO

O Controle de processos industriais foi a área pioneira beneficiada pela tecnologia decorrente da Lógica Difusa, sendo as primeiras experiências datadas de 1975 quando foi demonstrado no Queen College, em Londres, que um controlador difuso muito simples conseguiu controlar eficientemente uma máquina a vapor. Na mesma época, a primeira aplicação industrial significativa foi desenvolvida pela indústria de cimento F.L.Smidth Corp. da Dinamarca (GOMIDE & ROCHA, 1992.).

Hoje em dia, uma grande variedade de aplicações comerciais e industriais está disponível, destacando-se neste cenário o Japão e mais recentemente, os EUA e a Alemanha, que estão bem avançados na área de pesquisa e desenvolvimento da Lógica Difusa.

Dentre os exemplos típicos, incluem-se produtos de consumo tais como ar condicionado, onde os aparelhos possuem controle de vários sensores diferentes para a temperatura e a umidade, e assim operam de forma a conservar energia; câmeras de vídeo, onde os controladores difusos são aplicados ao foco automático e ao controle da íris da câmera; máquinas de lavar roupa, onde sensores adequados (temperatura da água, concentração de detergente, peso de roupas, nível de água, tipo de tecido, tipo de sujeira, grau de sujeira) controlam os ciclos da máquina: bater, enxaguar e centrifugar.

O sistema difuso foi criado para aproximar o raciocínio humano ao da lógica executada pelos computadores. Tradicionalmente, temos em um conjunto convencional limites bruscos, onde de uma hora para outra sua classificação muda. Por exemplo, o conjunto da idade de uma pessoa, quando a mesma chega aos 65 anos ela se torna idosa, fazendo com que a transição das classes (*idosos*) para as não-classes (*não-idosos*) se torne abrupta e repentina (FREITAS, 2002).

Nas próximas seções serão apresentados alguns conceitos sobre Lógica Difusa, funções de pertinência e conjuntos difusos.

4.2 CONCEITOS

O sistema difuso utiliza variáveis lingüísticas ao invés de variáveis numéricas, tornando assim mais fácil o entendimento humano, pois variáveis lingüísticas utilizam apenas palavras, como: *alto*, *baixo*, que são representadas por conjuntos difusos. O sistema difuso é composto pelo que chamamos de regras difusas, *SE-ENTÃO*. As regras *SE-ENTÃO* têm algumas palavras representadas por funções de pertinência.

Entretanto, para transformar os sinais de entrada e saída não difusos (variáveis reais), em variáveis lingüísticas (difusas) são necessários elementos adicionais entre a entrada e saída do processo. Estes elementos são: a base de regras construídas por um especialista, a máquina de inferência, que contém as funções de pertinência, o *fuzzificador* e o *defuzzificador*, que estão posicionados na entrada e saída do sistema, respectivamente, como pode ser visto na Figura 4.1.

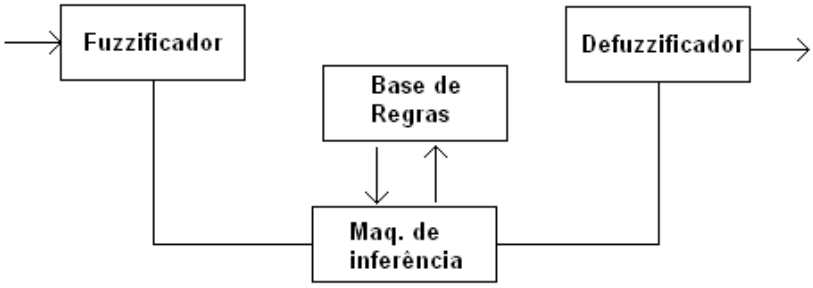

Figura 4.1: Representação de um Sistema Difuso.

Para a construção de um sistema difuso primeiro deve-se construir um conjunto de regras lógicas, chamado de algoritmo difuso. Essa construção é realizada por um especialista para determinar as funções de pertinência. Desse modo, a forma das regras empregadas depende do processo a ser investigado. Em particular, no contexto de mineração de dados, Lógica Difusa é um jeito natural de representar atributos discretos (valores reais) (FREITAS, 2002).

Um exemplo do uso de regras difusas para classificação de dados pode ser a regra abaixo, representada de forma discreta e de forma difusa.

Regra discreta: *Se Salário > 5000 E CargaH < 50 então Profissão = ?*

Regras difusas: *Se Salário é alto E CargaH é Baixa então Profissão = ?*,

onde *Alto* e *Baixo* são representados pelas funções de pertinência das Figuras 4.2 e 4.3.

Nas Figuras 4.2 e 4.3, têm-se as variáveis Salário e Carga Horária que serão *fuzzificadas* e receberão os valores Baixo, Médio ou Alto, dependendo de seus valores discretos.

As variáveis serão 100% Baixo caso seu valor esteja entre 0 e o primeiro valor mostrado no eixo do x. Se estiverem entre primeiro e o segundo, receberão certa possibilidade para Baixo e outra para Médio, sendo que a soma das duas tem que ser igual a 100%. Caso elas tenham valores entre o segundo e o terceiro ponto no eixo do x serão *fuzzificadas* como 100% Médio. Para valores entre o terceiro e o quarto ponto, elas receberão certa possibilidade para Médio e outra para Alto, sendo que a soma das duas tem que ser igual a 100%. E serão 100% Alto elas terão que ter valores maior ou igual ao quarto ponto.

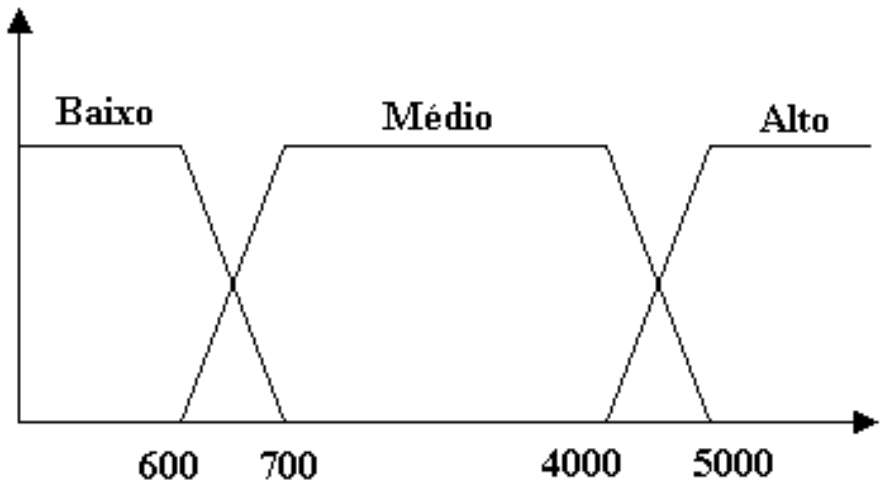

Figura 4.2: Função de Pertinência do Salário.

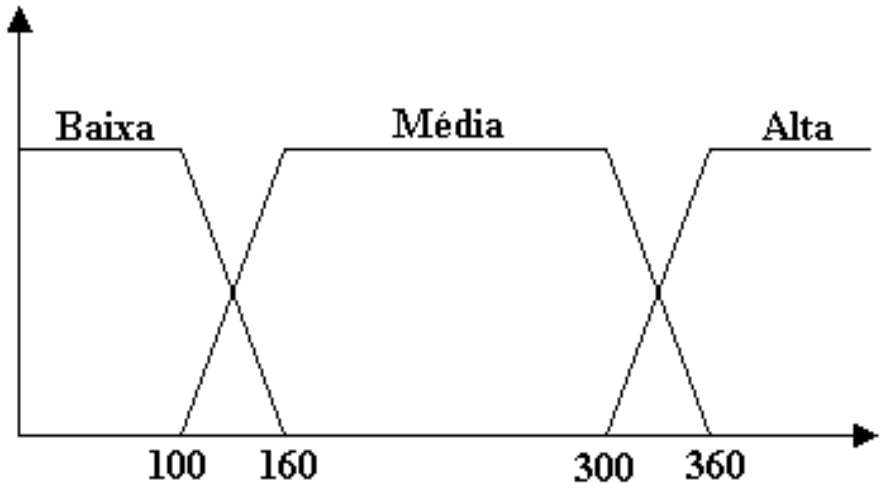

Figura 4.3: Função de Pertinência da Carga Horária.

O *fuzzificador* é definido como um mapeamento de pontos de valores numéricos para um conjunto difuso (WANG, 1997). Um conjunto difuso *A* definido no universo de discurso *U* é caracterizado por uma função de pertinência μ_A, a qual mapeia os elementos de *U* para o intervalo [0, 1], como mostrado na Fórmula 4.1.

$$\mu_A: U \Rightarrow [0, 1], \qquad (4.1).$$

Desta forma, a função de pertinência associa a cada elemento *x* pertencente a *U* um número real $\mu_A(x)$ no intervalo [0, 1], que representa o grau de possibilidade de que o elemento *x* venha a pertencer ao conjunto *A*, isto é, o quanto é possível para o elemento *x* pertencer ao conjunto *A*.

Defuzzificador faz o inverso do *fuzzificador*, onde o valor da variável difusa será transformado num valor discreto. O objetivo é obter-se um único valor numérico que melhor represente o valor difuso de saída (SHAW & SIMÕES, 1999).

4.2 FUNÇÕES DE PERTINÊNCIA

Funções de pertinência são funções que associam a cada valor x um numero, de um conjunto difuso, no intervalo fechado [0,1] que caracteriza o grau de pertinência de x em um conjunto difuso. As principais funções usadas são:

4.2.1 Função *Singleton*

Função *Singleton* tem valor 1 para o ponto *a* e 0 para todos os outros pontos em *U*. A Figura 4.4 mostra o gráfico correspondente. A definição da função *Singleton* é apresentada na Fórmula 4.2 (ANDRADE, 2006).

$$\mu_A(x) = \begin{Bmatrix} 1, & se \quad x = a \\ 0, & caso\, contrário \end{Bmatrix} \qquad (4.2).$$

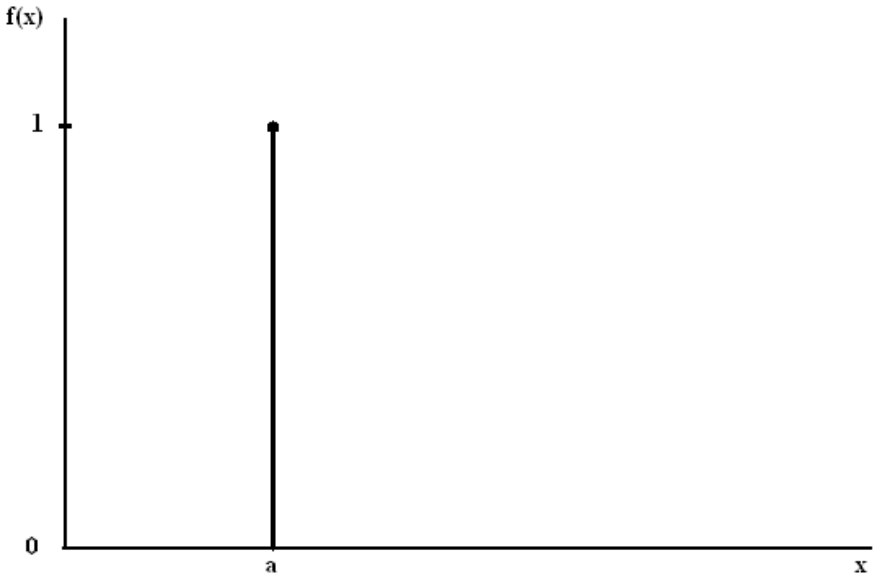

Figura 4.4: Função *Singleton*.

4.2.2 Função Triangular

Este tipo de função serve para indicar apenas um único ponto de máximo no conjunto. A definição da função Triangular é mostrada na Fórmula 4.3 (BORGES & GUIMARÃES, 2006).

$$\mu_A(x,a,b,c) = \begin{Bmatrix} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{Bmatrix} \qquad (4.3).$$

onde 0 é o menor valor possível para a função, 1 o maior valor, *a* o início da ascendente da função, *b* o valor de máximo, e *c* o final da descendente. Para uma melhor compreensão, apresenta-se o gráfico na Figura 4.5.

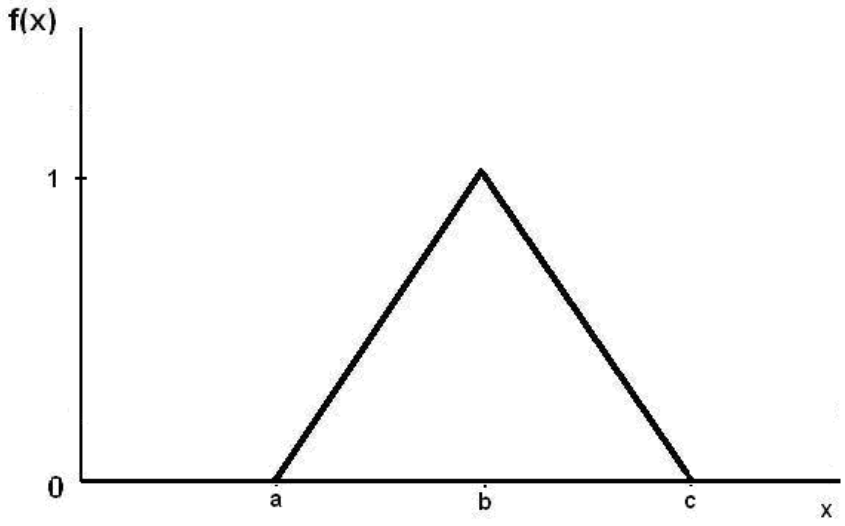

Figura 4.5: Função Triangular.

4.2.3 Função Trapezoidal

Nesta função é possível representar todo um intervalo de pontos de máximo, conforme a Fórmula 4.4 (BORGES & GUIMARÃES, 2006).

$$\mu_A(x,a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases} \qquad (4.4).$$

onde 0 é o menor valor possível para a função, 1 o maior valor, *a* o início do trapézio, *b* o início do intervalo de máximo, *c* o fim do intervalo de máximo, e *d* o final do trapézio. Veja o exemplo ilustrado na Figura 4.6.

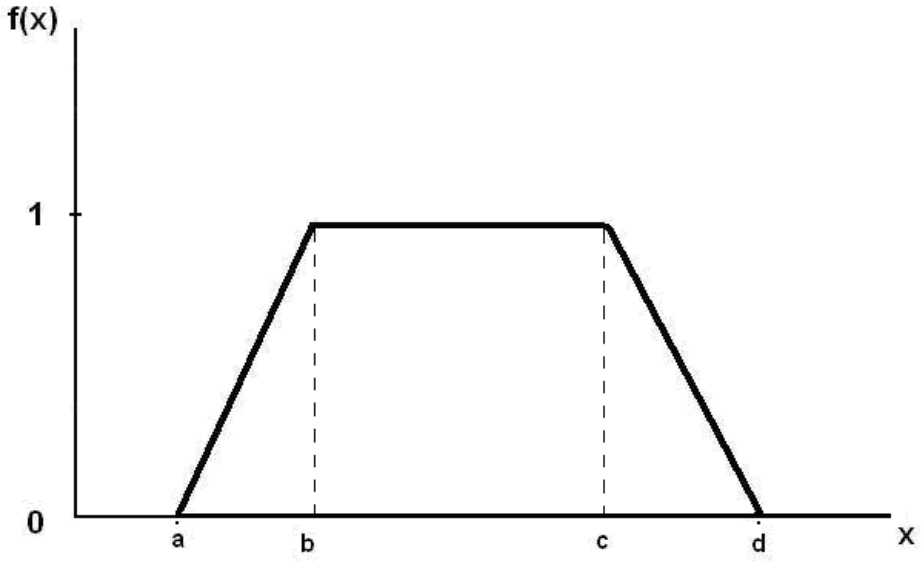

Figura 4.6: Função Trapezoidal.

4.2.4 Função Gaussiana

A função Gaussiana é apresentada na Fórmula 4.5 (OLIVEIRA FILHO, 2006).

$$f(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad (4.5)$$

onde x_0 a posição do máximo (centro da Gaussiana) σ a largura. Uma Gaussiana de largura 1 e centro em $x = 0$, apresentaria um gráfico semelhante ao apresentado na Figura 4.7.

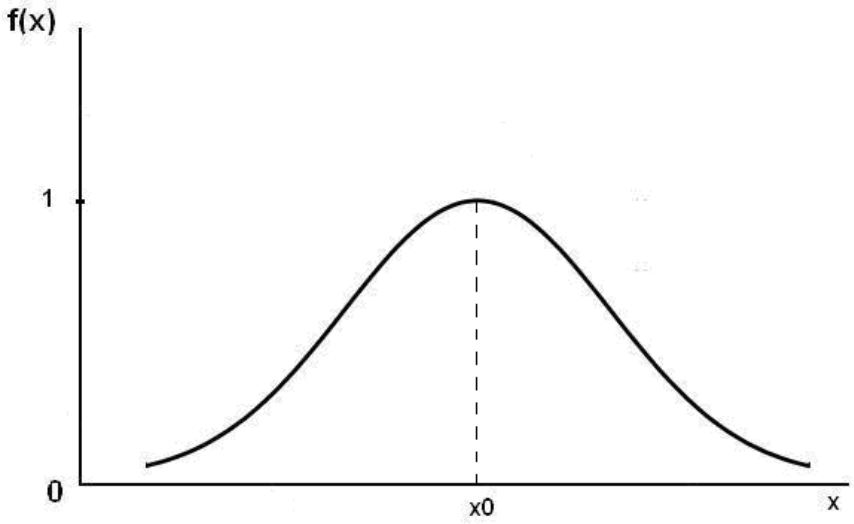

Figura 4.7: Função Gaussiana.

4.3 OPERAÇÕES EM CONJUNTOS DIFUSOS

A Lógica Difusa também utiliza alguns conceitos da teoria dos conjuntos, na qual o sistema difuso está baseado. Tendo *A* e *B* como conjuntos difusos definidos no mesmo universo *U*, como mostra a Figura 4.8, podem-se aplicar as cinco operações de conjuntos: igualdade, está contido, complemento, interseção e união.

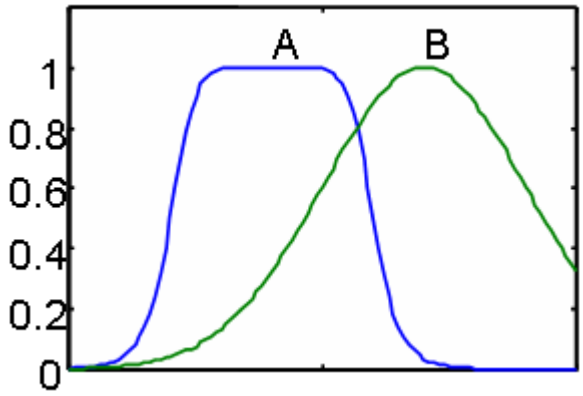

Figura 4.8: Conjuntos A e B.

- A igualdade de *A* e *B* (*A* = *B*) ocorre se e somente se, a função de pertinência, μ_A (x) = μ_B (x) para todo x $\in$ *U*, como pode ser visto na Figura 4.9;

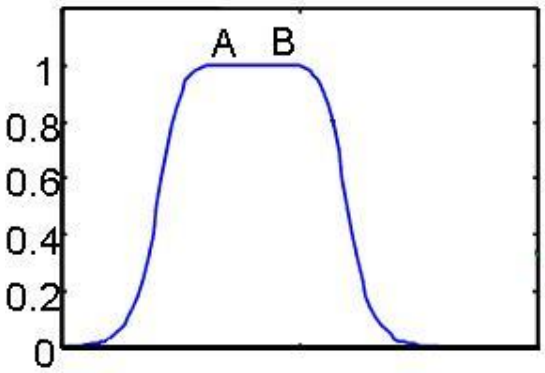

Figura 4.9: Igualdade.

- *A* está contido em *B* (*A* $\subset$ *B*) ocorre se e somente se, a função de pertinência, μ_A (x) $\leq$ μ_B (x) para todo *x* $\in$ *U*, como apresentado na Figura 4.10;

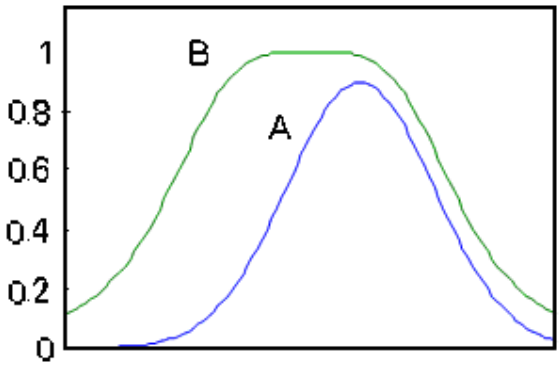

Figura 4.10: Está contido.

- O complemento de *A* é o conjunto difuso $\bar{A}$ em um universo *U*, apresentado na Figura 4.11. Definido pela função de pertinência apresentada na Fórmula 4.6;

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \qquad (4.6).$$

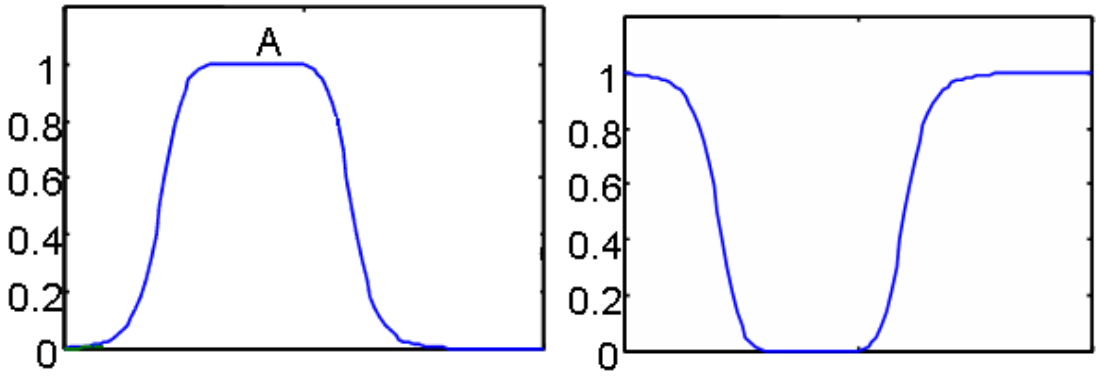

Figura 4.11: Complemento.

- A união de *A* e *B* (*A* U *B*) em um universo *U*, é definido pela função de pertinência da Fórmula 4.7. Um exemplo é apresentado na Figura 4.12; e

$$\mu_{A \cup B}(x) = \max\left[\mu_A(x), \mu_B(x)\right] \qquad (4.7).$$

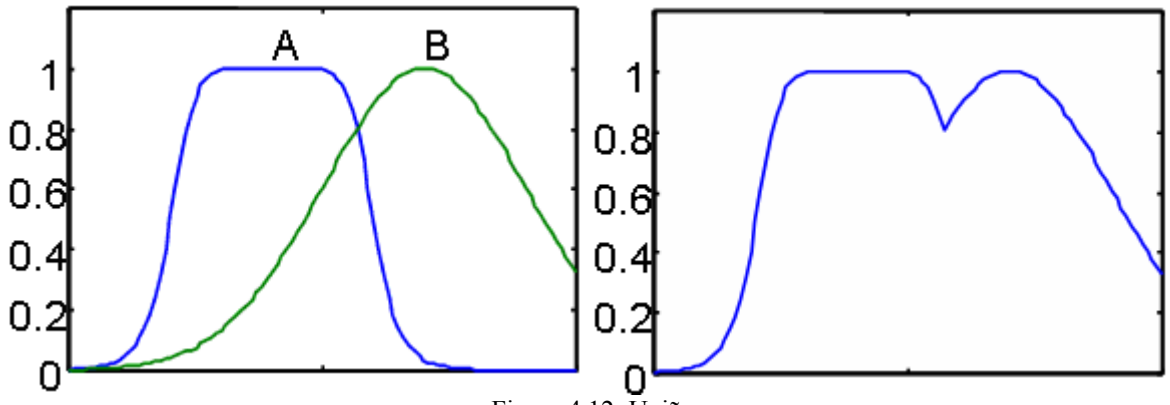

Figura 4.12: União.

- A interseção de *A* e *B* (*A* ∩ *B*) em um universo *U* é definido pela Fórmula 4.8 da função de pertinência. Um exemplo é apresentado na Figura 4.13.

$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)] \qquad (4.8).$$

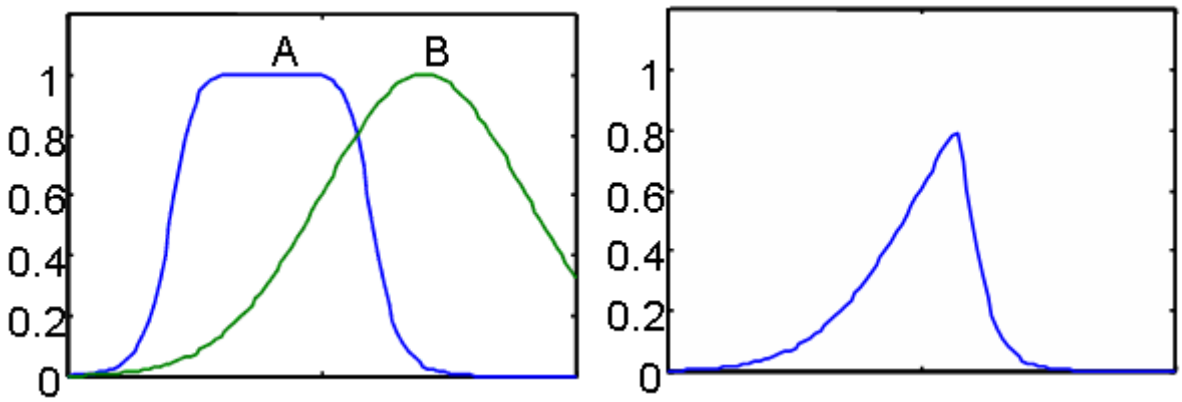

Figura 4.13: Interseção.

4.4 PROPRIEDADES DOS CONJUNTOS DIFUSOS

Dado um universo de discurso *U*, e três conjuntos difusos $A \subset U$, $B \subset U$, $C \subset U$, então as seguintes propriedades são válidas para os conjuntos difusos:

- Propriedade comutativa: Quaisquer que sejam os conjuntos *A* e *B*, tem-se a Fórmula 4.9;

$$A \bigcap B = B \bigcap A$$
$$A \bigcup B = B \bigcup A \quad (4.9).$$

- Propriedade associativa: Quaisquer que sejam os conjuntos *A*, *B* e *C*, tem-se a Fórmula 4.10;

$$(A \bigcap B) \bigcap C = A \bigcap (B \bigcap C)$$
$$(A \bigcup B) \bigcup C = A \bigcup (B \bigcup C) \quad (4.10).$$

- Reflexiva: Qualquer que seja o conjunto *A*, tem-se a Fórmula 4.11;

$$A \bigcap A = A$$
$$A \bigcup A = A \quad (4.11).$$

- Propriedade distributiva: Quaisquer que sejam os conjuntos *A, B* e *C*, tem-se a Fórmula 4.12;

$$A \bigcap (B \bigcup C) = (A \bigcup B) \bigcap (A \bigcup C)$$
$$A \bigcup (B \bigcap C) = (A \bigcap B) \bigcup (A \bigcap C) \quad (4.12).$$

- Conjunto difuso e conjunto nulo: O conjunto vazio é o elemento neutro para a união dos conjuntos, para todos os conjuntos. A interseção do conjunto vazio com qualquer conjunto *A*, fornece o próprio conjunto vazio, como é apresentado na Fórmula 4.13;

$$A \bigcup \phi = A$$
$$A \bigcap \phi = \phi \quad (4.13).$$

- Conjunto difuso e conjunto universal: O conjunto universal *U* é o elemento neutro para a interseção de conjuntos. A união do conjunto *A* com o conjunto universal *U*, fornece o próprio conjunto *U*, como é apresentado na Fórmula 4.14;

$$A \bigcap U = A$$
$$A \bigcup U = U \quad (4.14).$$

- Involução: A negação de um conjunto *A* negado resulta no mesmo conjunto *A*, como é visto na Fórmula 4.15; e

$$\overline{(\overline{A})} = A \qquad (4.15).$$

- Teorema de Morgan: A união do complemento de dois conjuntos *A* e *B* é a interseção dos complementos desses conjuntos e o complemento da interseção de sois conjuntos *A* e *B* é a união do complemento desses conjuntos, como é mostrado na Fórmula 4.16.

$$\begin{aligned} \overline{(A \cap B)} &= \overline{A} \cup \overline{B} \\ \overline{(A \cup B)} &= \overline{A} \cap \overline{B} \end{aligned} \qquad (4.16).$$

Todas essas propriedades são aplicadas a qualquer conjunto difuso, com exceção ao:

- Conjunto difuso e seu complemento: A interseção de um conjunto *A* com seu complemento resulta no conjunto vazio e a união de um conjunto *A* com seu complemento resulta no conjunto universo *U*, como é apresentado na Fórmula 4.17.

$$\begin{aligned} A \cap \overline{A} &\neq \phi \\ A \cup \overline{A} &\neq U \end{aligned} \qquad (4.17).$$

4.5 MODIFICADORES

O sistema difuso também utiliza modificadores que servem para alterar os valores das variáveis lingüísticas. Alguns exemplos de modificadores são: *muito*, *pouco*, *não muito*, *mais ou menos*. Por exemplo, os modificadores para a variável, *carga_horária*, poderiam ser *não muito alta*, *muito baixa* ou *mais ou menos alta*. Outros modificadores são: *complemento*, *negação*, e os conectores *e* e *ou*.

O modificador *muito* da variável, *alto*, é caracterizado matematicamente como um conjunto difuso em *U* com a função de pertinência elevada ao quadrado (WANG, 1997), mostrado na Fórmula 4.18.

$$\mu_{\text{muito A}}(x) = [\mu_A(x)]^2, \qquad (4.18).$$

O modificador *mais ou menos* de *A* é um conjunto difuso em *U* com a seguinte função de pertinência apresentada na Fórmula 4.19.

$$\mu_{\text{mais ou menos } A}(x) = [\mu_A(x)]^{1/2} \qquad (4.19).$$

O modificador *pouco* da variável, *alto*, é caracterizado matematicamente como um conjunto difuso em *U* com a função de pertinência elevada a 1,3 (CANUTO, 2006), como é visto na Fórmula 4.20.

$$\mu_{\text{pouco } A}(x) = [\mu_A(x)]^{1,3} \qquad (4.20).$$

Os conectivos *e* e *ou* são equivalentes às operações de união e intersecção de conjuntos, respectivamente, podendo dar origem a conjuntos complexos definidos, porém representados lingüisticamente de maneira simples.

4.6 CONSIDERAÇÕES FINAIS

Nesse capítulo foram apresentados alguns conceitos básicos de Lógica Difusa, operações e propriedades dos conjuntos difusos, funções de pertinências e modificadores.

Foram desenvolvidas, no sistema difuso desse projeto, apenas as funções de pertinência triangular e a trapezoidal. Suas entradas vão ser os valores dos atributos selecionados pela ferramenta na base de dados, esses valores são todos reais. As saídas recebidas são Alto, Médio ou Baixo. Essa parte será explicada com mais detalhes no capítulo 6, onde serão apresentada a implementação e as discussões sobre a ferramenta.

5 GERANDO REGRAS DIFUSAS POR MEIO DE ALGORITMOS GENÉTICOS

5.1 INTRODUÇÃO

Neste capítulo será explicado como as regras difusas são geradas usando algoritmo genético. Na seção 5.2 é apresentada abordagens para o uso de AG na descoberta de regras difusas. Nas três subseções 5.2.1, 5.2.2 e 5.2.3 são explicadas cada uma das abordagens. Na 5.3 é mostrada como são usadas as regras difusas para a mineração de dados. As seções 5.4 e 5.5 explicam como é montada a matriz de confusão com regras normais e usando regras difusas, respectivamente. As seções 5.6 e 5.7 mostram como é calculada a função de avaliação (*fitness*) e como são classificadas as instancias com regras difusas e a 5.8 o resumo do capítulo.

5.2 ABORDAGENS PARA O USO DE AG NA DESCOBERTA DE REGRAS DIFUSAS

Segundo (FREITAS, 2002) há três maneiras de se usar Algoritmos Evolucionários para a descoberta de regras difusas, como mostradas na Figura 5.1

- Algoritmo Genético usando regras difusas;
- Algoritmo Genético usando função de pertinência; e
- Algoritmo Genético usando regras difusas e função de pertinência.

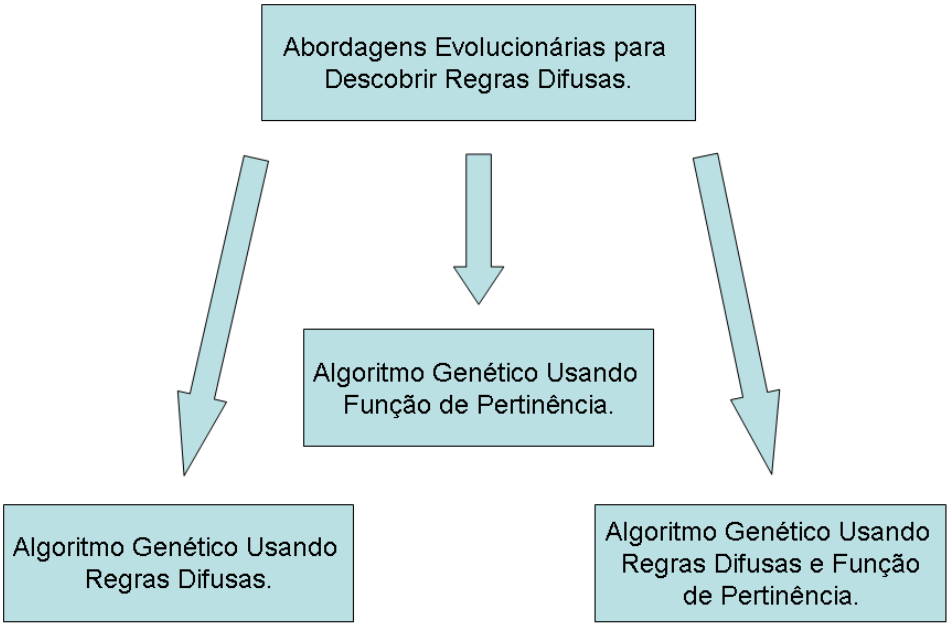

Figura 5.1: Usando Algoritmo Genético para Descobrir Regras Difusas.

As abordagens citadas serão discutidas a seguir. Este trabalho se baseia na abordagem usando Regras Difusas.

5.2.1 Algoritmos Genéticos Usando Regras Difusas

Essa classificação é usada para encontrar os melhores conjuntos de regras difusas, porém os valores atribuídos às funções de pertinência são pré-definidos. Então, o AG encontra valores dos atributos que são relevantes para predizer valores precisos de outros atributos. O uso do AG para descobrir regras difusas é similar ao de regras discretas (FREITAS, 2002).

Na verdade, do ponto de vista da codificação da regra, usando regra difusa tal como *Salário = baixo*, onde *baixo* é um valor lingüístico associado a um conjunto difuso, não é diferente do uso de regra discreta, tal como *Salário = 0...700*, onde o atributo *Salário* era previamente normalizado em intervalos discretos tal como 0...700. Em ambos os casos têm-se regras do tipo $Attr_i = Val_{ij}$ onde Val_{ij} é o valor de *j* pertencente ao domínio do atributo *i*.

5.2.2 Algoritmos Genéticos Usando Função de Pertinência

Em outra abordagem, o AG é usado apenas na função de pertinência. Este algoritmo é útil, por exemplo, quando as regras discretas já foram descobertas por um método convencional, e somente se quer *fuzzificar* as regras discretas encontradas (FREITAS, 2002).

O trabalho do AG é *fuzzificar* os nós folhas da árvore de decisão. Por exemplo, considere a árvore de decisão discreta mostrada na Figura 5.2:

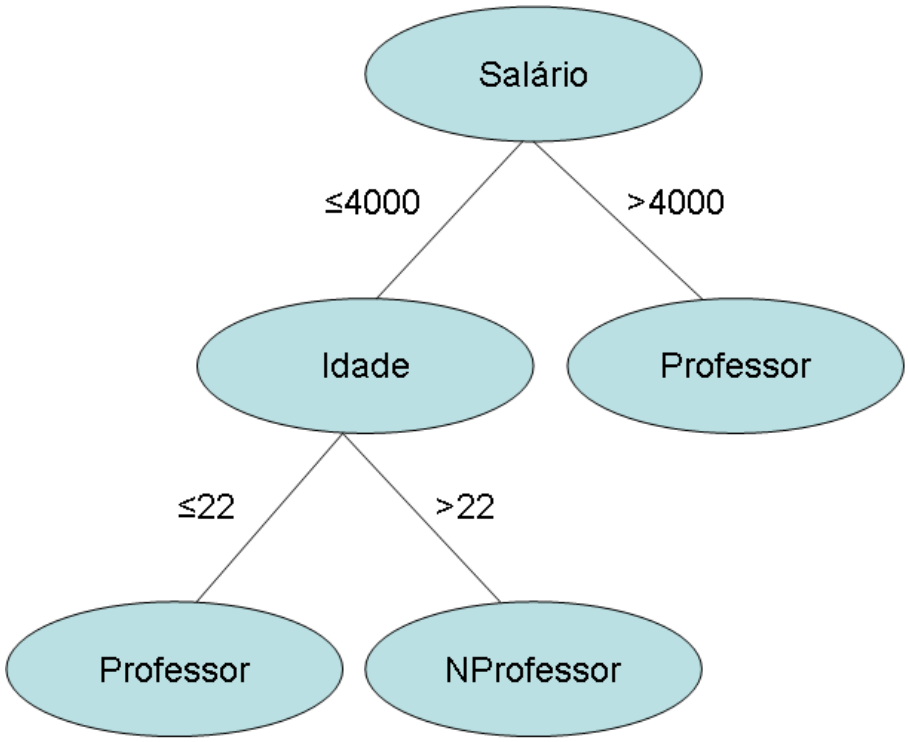

Figura 5.2: Exemplo da Árvore de Decisão.

A árvore de decisão tem dois nós (internos) que irão ser *fuzzificados*. Neste exemplo o AG tem que definir duas variáveis difusas. A primeira variável a ser *fuzzificada* é *Salário* ≤ *4000* e *Salário* > *4000* e depois a outra que irá ser *fuzzificada* é *Idade* ≤ *22* e *Idade* > *22*.

5.2.3 Algoritmos Genéticos Usando Lógica Difusa e Função de Pertinência

Aqui é usado AG tanto para formar as regras difusas quanto apenas para transformar os atributos difusos. Por exemplo:

Se (Atr_1 é Val_{1j}) e... e (Atr_m é Val_{mj}) Então ...

onde *m* é o total de atributos que se tem no banco, e o valor de *j* indica o domínio de cada atributo do banco (FREITAS, 2002).

Nessa abordagem, cada Val_{ij}, onde $i = 1... m$, será *fuzzificado*. Cada Val_{ij} é usado na função de pertinência e então seu valor *fuzzificado* é calculado. Por exemplo, para a regra *Se Salário = 100*, o valor 100 é avaliado na função de pertinência da Figura 5.3, obtendo-se o resultado 100% *baixo*. A regra formada é, portanto "*Se Salário é baixo*".

5.3 REGRAS DIFUSAS PARA MINERAÇÃO DE DADOS

A técnica de modelagem difusa tem sido aplicada em sistemas de mineração de dados para dois principais propósitos (WANG, 1997). O primeiro seria, com o uso de regras difusas, tornar o resultado mais compreensível para os usuários. E o segundo, aplicações de mineração de dados usando regras difusas são menos sensíveis a dados ruidosos ou incompletos. O uso de regras difusas ao invés de discretas maximiza tanto a robustez quanto o fator de entendimento para o usuário.

Por exemplo, uma aplicação de classificação poderia usar a regra 2 abaixo em vez da regra 1. A regra 1 é uma regra discreta onde o valor dela é nomeado para *Salário* e *Carga Horária*. A regra 2 é claramente uma regra maior, onde o valor de *Renda* é determinado pelas funções de pertinência da Figura 5.3 e da Figura 5.4.

1) Discreto: *Se Salário ≤ 600 E CH > 360 Então CLASSE = C*

2) Difusa: *Se Salário é baixo E CH é alta Então CLASSE= C*

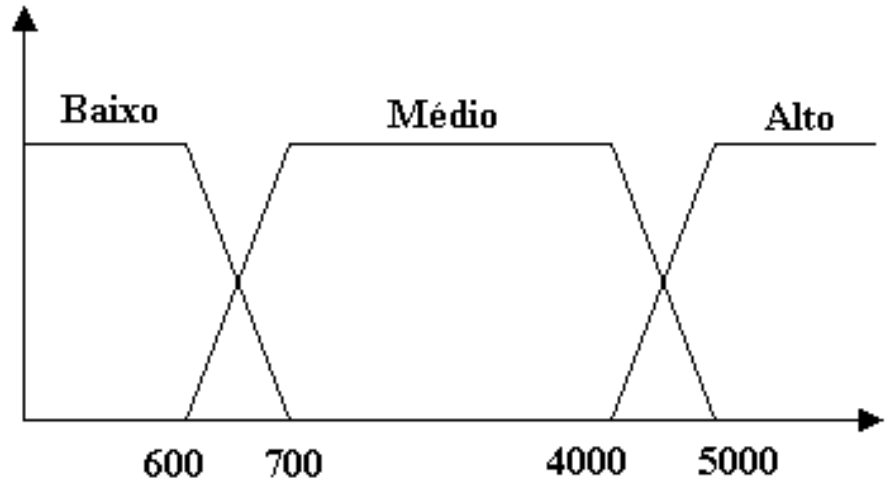

Figura 5.3: Função de Pertinência do *Salário*.

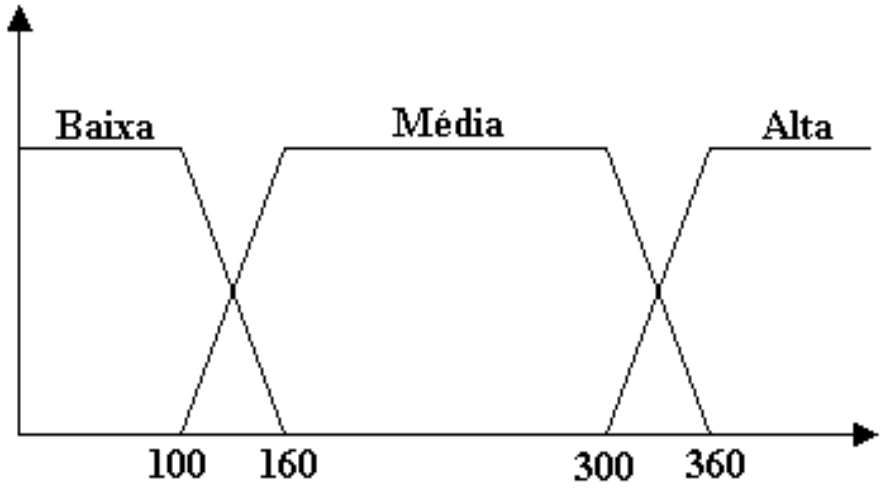

Figura 5.4: Função de Pertinência da *Carga Horária*.

Quando são geradas regras difusas, os atributos no antecedente da regra são normalmente *fuzzificados*, mas o conseqüente da regra é mantido preferencialmente com valor discreto. Considerando que as regras de predição são na maioria das vezes para se tomar decisões, os usuários preferirão uma conclusão com valores discretos a valores difusos.

Por exemplo, um sistema poderia ser usado para classificar se um cliente ficará inadimplente. Neste caso, uma regra que indica que o cliente tem um pouco de chance de ficar inadimplente não é muito interessante. Portanto, regras difusas devem manter os valores do conseqüente discretos.

5.4 MATRIZ DE CONFUSÃO

Para determinar a *fitness* de uma regra, é necessário montar a matriz confusão como na Figura 5.5. Considerando a regra *Se A então Classe = C*, as células da matriz de confusão indicam o número de instâncias na base de dados que atendem ou não a condição *A*, e que apresentam ou não o valor *C* para a classe prevista.

	C	Não-C
Satisfaz A	VP	FP
Não Satisfaz A	FN	VN

Figura 5.5: Matriz de Confusão.

A matriz de confusão é formada pelos quatro componentes abaixo.

- *VP* (Verdadeiro Positivo): indica a soma do número de instâncias que satisfazem à condição *A* e pertencem à classe *C*;

- *VN* (Verdadeiro Negativo): indica a soma do número de instâncias que não satisfazem à condição *A* e não pertencem à classe *C*;

- *FP* (Falso Positivo): indica a soma do número de instâncias que satisfazem à condição *A* e não pertencem à classe *C*; e

- *FN* (Falso Negativo): indica a soma do número de instâncias que não satisfazem à condição *A* e pertencem à classe *C*.

5.5 MATRIZ DE CONFUSÃO PARA REGRAS DIFUSAS

A matriz de confusão para regras difusas é similar à de regras discretas. Para regras difusas os valores de *TP*, *FN*, *FP* e *VN* estão em um intervalo de [0... 1], que corresponde ao

valor da função de pertinência do antecedente da regra. Neste caso eles têm o seguinte significado:

- *VP* (Verdadeiro Positivo): indica a soma dos valores da função de pertinência que satisfazem à condição *A* e pertencem à classe *C*;

- *VN* (Verdadeiro Negativo): indica a soma dos valores da função de pertinência que não satisfazem à condição *A* e não pertencem à classe *C*;

- *FP* (Falso Positivo): indica a soma dos valores da função de pertinência de instâncias que satisfazem à condição *A* e não pertencem à classe *C*; e

- *FN* (Falso Negativo): indica a soma dos valores da função de pertinência que não satisfazem à condição *A* e pertencem à classe *C*.

5.6 FUNÇÃO DE AVALIAÇÃO (*FITNESS*)

A função utilizada para a *fitness* neste algoritmo corresponde à seguinte fórmula:

$$\frac{(VP+VN)}{(VP+VN+FN+FP)}, \qquad (5.1).$$

as funções que usam três ou quatro das variáveis *VP*, *VN*, *FP* e *FN* tem a vantagem de usar mais informações (FREITAS, 2002).

O valor da *fitness* é importante no processo de seleção dos indivíduos para compor uma nova população. Sempre que uma nova população é gerada, o processo seletivo busca um melhor conjunto de indivíduos, copiando alguns diretamente para a próxima população e combinando pares de indivíduos para produzir um par de indivíduos novos, em um processo chamado *crossover*. Os indivíduos mais aptos, ou com maior valor de *fitness*, têm maior probabilidade de serem selecionados.

5.7 CLASSIFICANDO INSTÂNCIAS COM REGRAS DIFUSAS

Para classificar novos exemplos de dados usando regras difusas, frequentemente precisa-se combinar as predições de várias regras difusas, ou selecionar uma delas, para assim decidir qual classe será escolhida para um exemplo de dados (FREITAS, 2002).

5.7.1 Determinação do Conseqüente

O conseqüente é um atributo fixo, para o qual o modelo de classificação está sendo construído. Segundo (FREITAS, 2002) o processo que determina o conseqüente da regra é dividido em três passos. Por exemplo, supondo que o conjunto de dados tenha cinco linhas, três da classe C_1 e duas da classe C_2. O primeiro passo é mostrado na Figura 5.6(a), onde a primeira coluna da tabela indica a classe e a segunda indica o grau de pertinência.

No segundo passo, para cada classe, é feita a soma do grau de pertinência de todos os dados treinados da classe. O resultado deste passo pode ser visto na Figura 5.6(b). No terceiro passo, a classe escolhida para o conseqüente da regra é a classe que tiver o maior valor das somas do grau de pertinência feito no cálculo do passo dois. O resultado pode ser visto na Figura 5.6(c).

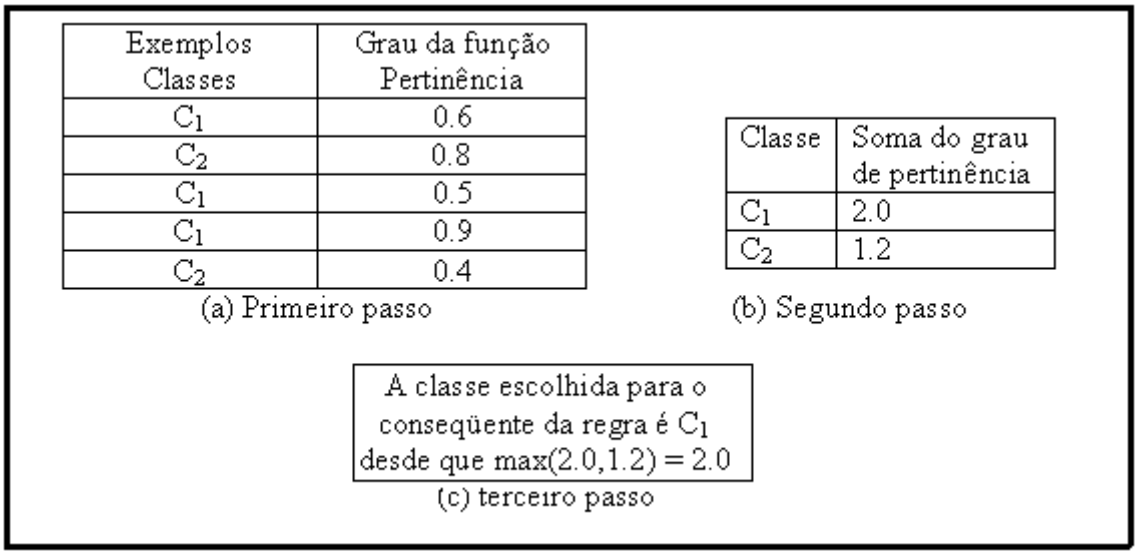

Figura 5.6: Determinação do Conseqüente.

Para determinar o conseqüente da regra, que será o valor previsto pela regra para a classe selecionada, seleciona-se o valor com maior grau de pertinência, maximizando assim a qualidade da regra produzida. Por exemplo, no caso da classificação de profissões entre ser *professor* (C_1) e *não ser professor* (C_2). Se a condição do antecedente da regra seleciona pessoas com *Salário* $\leq$ 4000 e com *Idade* > 22, o algoritmo irá fazer o cálculo da pertinência, de *professor* ou *não professor*, que satisfaçam este antecedente para especificar o conseqüente da regra. Se a soma do grau de pertinência de quem tem *Idade* > 22 e *Salário* $\leq$ 4000 com *Classe = professor* for maior do que os com *Classe = não professor* na base de dados, então a regra correspondente será *Se Salário* $\leq$ *4000 e com Idade > 22 ENTÃO Professor = Verdadeiro*.

5.8 CONSIDERAÇÕES FINAIS

Nesse capítulo foram mostradas três diferentes do uso de Algoritmos Evolucionários para a descoberta de regras difusas. Também algumas abordagens sobre como pode ser feita a codificação de indivíduos no AG.

No capítulo seguinte serão mostradas as implementações e as discussões sobre a ferramenta proposta por este trabalho usando um Algoritmo Evolucionário para a descoberta de regras difusas

6 IMPLEMENTAÇÃO E DISCUSSÃO

6.1 INTRODUÇÃO

Neste capítulo será apresentado como foi feita a implementação utilizando a linguagem de programação Java. Foi desenvolvido um total de 12 classes e todas as classes são explicadas nas secções seguintes.

O programa desenvolvido utiliza a abordagem de Michigan, pois é a melhor escolha para esse tipo de problema (FREITAS, 2002.). A regra usada é de tamanho variável, porque a regra de tamanho fixo ocasiona indivíduos com *fitness* muito baixa, para esse problema. Já a codificação implementada, nesse projeto, é a de alto nível para tornar as regras próximas dos seres humanos.

Serão apresentados também alguns diagramas de UML utilizados na modelagem do projeto.

6.2 DIAGRAMA DE CLASSES.

O diagrama de classes mostra a estrutura estática das classes do sistema. O diagrama em pacotes do programa pode ser visualizado na Figura 6.1. É apresentado apenas um diagrama resumido, pois se torna mais fácil o entendimento e é essencial para o entendimento do programa. O apêndice A contém o diagrama completo.

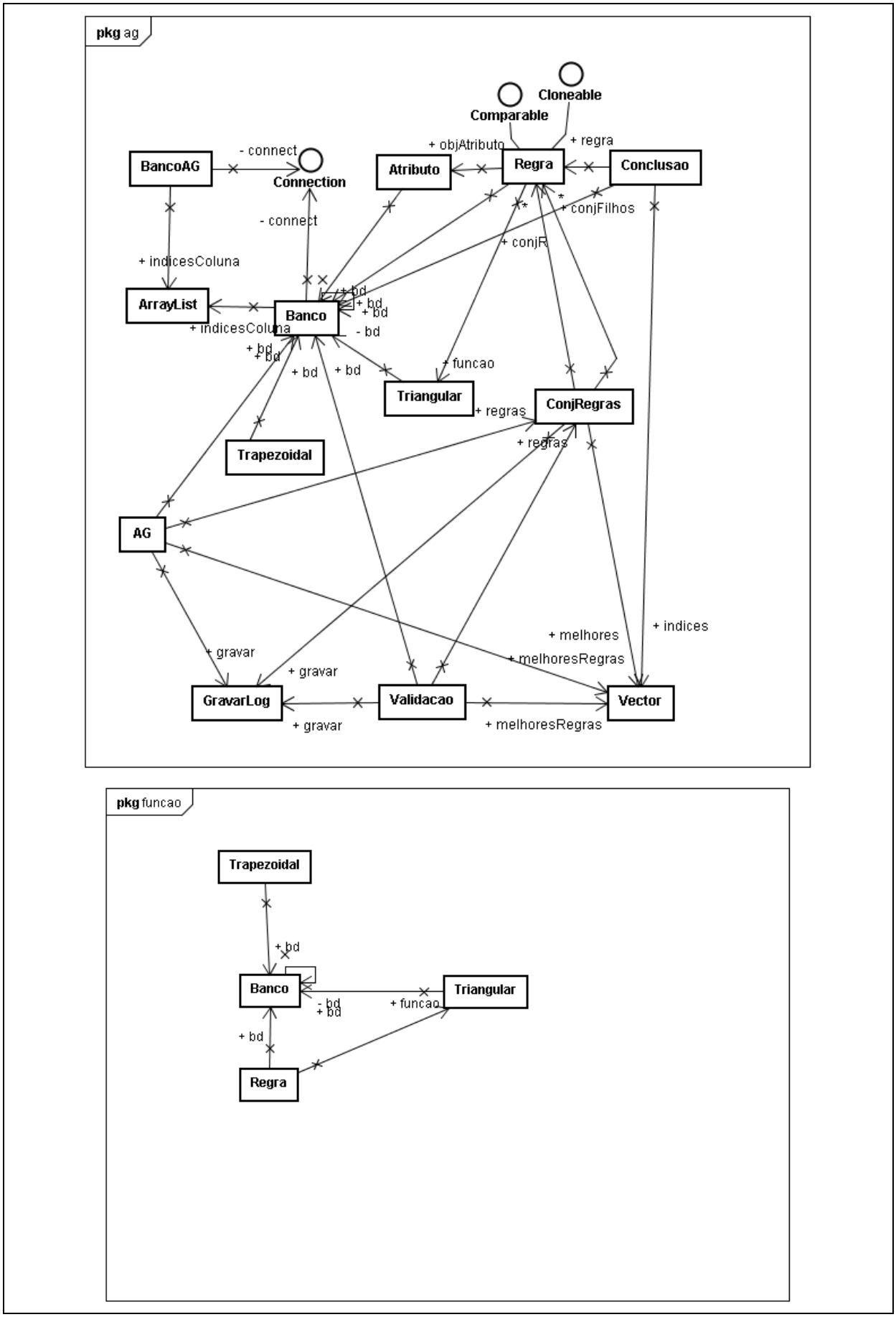

Figura 6.1: Diagrama de Classes em Pacotes.

O diagrama mostrado acima é representado por dois pacotes. O primeiro é o pacote chamado de AG onde é apresentada a classe Atributo, dentro dela cada atributo é montado e validado. Na classe Regra é onde ocorre a montagem dos atributos para formar uma regra. A classe Conclusão é a parte onde o conseqüente da regra será escolhido. Na classe ConjRegra vão ser montadas várias regras. Feita as regras ela então é validada e gravada em um arquivo de log.

O pacote chamado Função onde é apresentada a classe que contém os métodos para cálculos das funções de pertinência. Nesse pacote é utilizada a Lógica Difusa pela ferramenta. Nele são desenvolvidas duas funções de pertinência, representadas pelas classes Triangular e Trapezoidal na Figura 6.1.

6.3 DIAGRAMA DE CASO DE USO

Nesse diagrama verifica-se o comportamento dinâmico dos objetos de um sistema. A figura 6.2 apenas a simula a criação de um conjunto de regras e a busca pelas melhores regras, pois foram consideradas as essenciais no algoritmo.

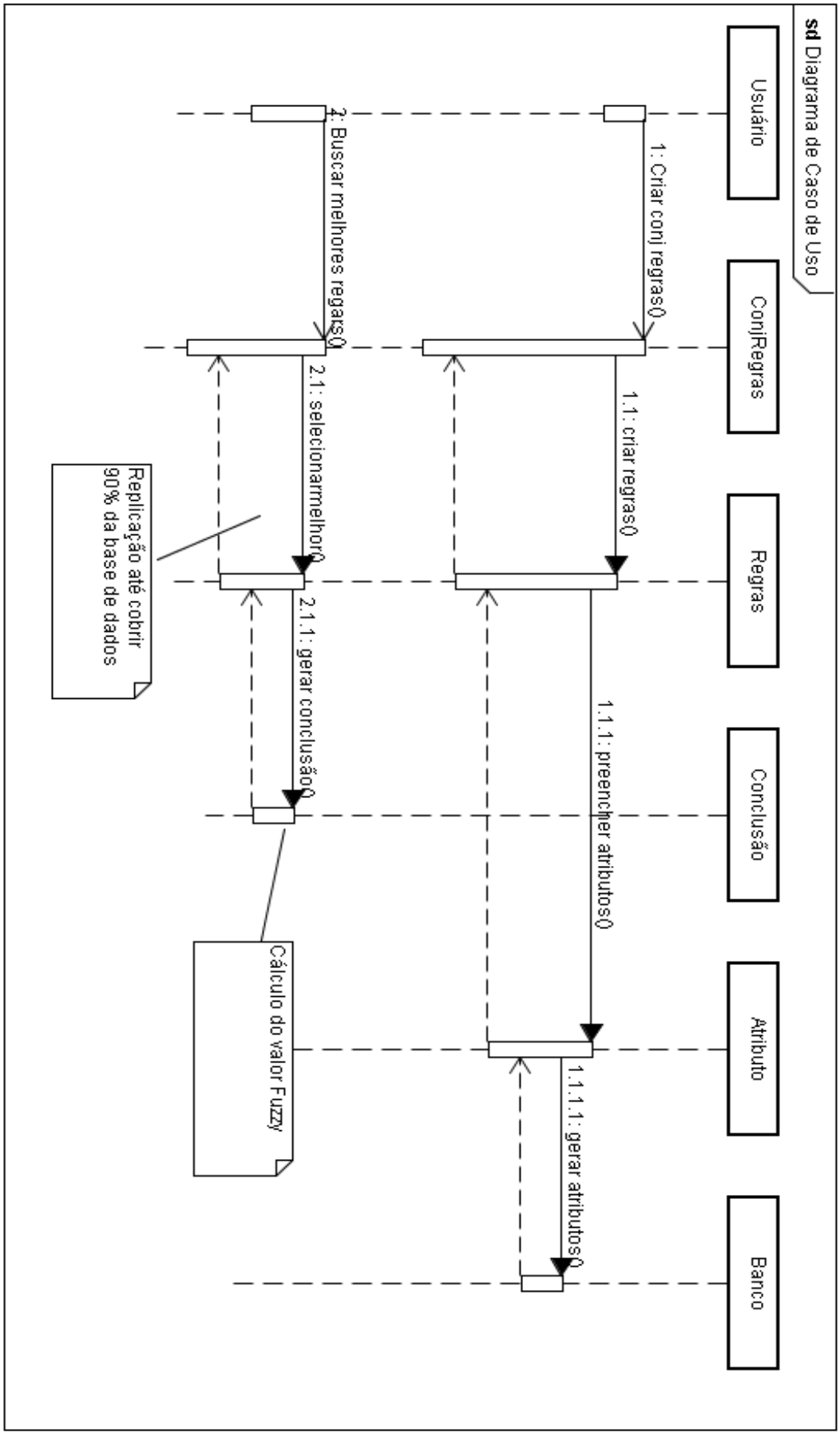

Figura 6.2: Diagrama de Caso de Uso.

5.4 CODIFICAÇÃO DO INDIVÍDUO

Cada cromossomo, nesse trabalho, é uma regra na forma *"SE antecedente ENTÃO conseqüente"*. O antecedente da regra é formado por condições ligadas pelo operador lógico *E*. Cada condição corresponde a um atributo da base de dados, que é armazenada em um vetor de 4 campos, como detalhado a seguir. O conseqüente foi explicado na secção 5.7.1.

A primeira posição da condição identifica um atributo da base de dados. Os nomes de todos os atributos são coletados automaticamente por chamadas JDBC (*Java Database Connectivity,* 2006), permitindo assim que o algoritmo produza regras com nomes de atributos válidos. A segunda posição indica o operador da regra difusa da condição, onde operador difuso *É* ou qualquer operador relacional pode ser usado. A terceira posição está relacionada com o valor previsto para o atributo na condição, sendo considerados também automaticamente todos os valores válidos que depois serão *fuzzificados* e jogados na regra. A quarta posição indica se esta condição está ativa. Como em cada regra são montadas condições para todos os atributos da base de dados, é através desta informação que se determinam quais atributos serão efetivamente considerados na regra.

A Figura 6.3 mostra um exemplo de condição, que seleciona itens da base de dados onde o atributo *Salário* tem valor *baixo*. A condição está ativa.

Nome	Operador	Valor	Ativa?
Salário	É	Baixo	V

Figura 6.3: Representação do Atributo da Regra.

As regras são formadas automaticamente, selecionando-se aleatoriamente valores válidos para cada atributo e determinando quais deles estão ativos ou inativos. O processo de evolução das regras, semelhante ao processo evolutivo descrito por Darwin, onde os indivíduos sofrem um processo de seleção natural, é realizado em várias épocas ou iterações do algoritmo, até que se atinja o número máximo de iterações especificado como parâmetro da ferramenta.

6.4.1 Pittsburgh ou Michigan

Existem, em geral, duas abordagens para a representação das regras em um Algoritmo Genético para Mineração de Dados. Pela abordagem de Michigan, cada cromossomo representa uma regra, ao passo que pela abordagem de Pittsburgh cada cromossomo representa um conjunto de regras (FREITAS, 2002).

Naturalmente, a escolha de uma das abordagens depende do objetivo da mineração de dados. Em alguns casos, o resultado deve ser um pequeno conjunto de regras interessantes ou surpreendentes. As regras devem ser avaliadas individualmente e, para tanto, a abordagem de Michigan é a melhor escolha.

5.6.2 Fixa ou Variável

Presume-se que o algoritmo de descoberta de regra seja autônomo o bastante para decidir que número de condições é satisfatório para cada regra. Têm-se duas abordagens básicas para esse tipo de problema. O genótipo com tamanho fixo que é satisfatório usando regras de tamanho variável e o genótipo com tamanho variável que é equivalente a regra de tamanho variável (FREITAS, 2002).

A abordagem usada neste projeto foi a variável, com um tamanho máximo para a regra, porém o tamanho da mesma varia de um indivíduo para o outro. Isso porque regras grandes são muito específicas, gerando assim baixo valor de aptidão.

6.4.3 Binária ou Alto Nível

Em geral na codificação binária, cada atributo previsto tem certo número de bits, que depende dos tipos de dados contidos no atributo (FREITAS, 2002).

Um exemplo de codificação binária seria o atributo *EstadoCivil* com três valores: *solteiro*, *casado* e *divorciado*. Pode-se usar neste caso a seguinte codificação binária: 100 para *solteiro*, 010 para *casado* e 001 para *divorciado*.

No caso da codificação de alto nível é usada a linguagem natural. Por exemplo, no caso acima onde o atributo é *EstadoCivil* os valores poderiam ser: *casado*, *solteiro* e *divorciado*.

A codificação binária tende a ser incômoda quando se tem uma grande quantidade de valores discretos.

A codificação usada nesse projeto foi a de alto nível, pois se quer tornar as regras mais compreensíveis para os usuários.

6.5 IMPLEMENTAÇÃO

A ferramenta é implementada através de doze classes dividida em dois pacotes (*ag* e *função*). O pacote *função* contém as funções de pertinência que vão representar os atributos difusos, já o pacote *ag* tem a finalidade de montar um conjunto de regras (*população*), onde cada regra (*individuo*) é formada por um conjunto de atributos que são representados por um vetor de quatro campos. O primeiro campo contém o nome da coluna da base extraída, o segundo recebe o símbolo do operador que pode ser igual, maior, menor, maior ou igual, menor ou igual e caso seja um operador difuso é representado pelo *É*, o terceiro campo está relacionado com o valor do atributo capturado na base e por último, o quarto campo indica se o atributo é válido ou não, como é mostrado na Figura 6.4.

Nome	Operador	Valor	Ativa?
Qntd_Crédito	*É*	*baixo*	*V*

Figura 6.4: Representação do Atributo.

A captura dos atributos é feita de maneira aleatória. Primeiro o vetor vai ser preenchido com todos os nomes das colunas da base utilizada, em seguida verifica se a coluna é difusa ou não (indicado pelo usuário) caso o operador seja difuso, então o símbolo do operador irá receber o operador *É*, caso contrário vai ser escolhido aleatoriamente entre os cinco operadores ($>$, $\geq$, $<$, $\leq$ ou $=$). No caso de atributos não numéricos o operador escolhido é

o igual. Em seguida, preenche-se o valor do atributo que também aleatoriamente, onde vai ser encolhido um valor de um até o tamanho da tabela. Por ultimo, também de forma aleatória, a variável ativa irá receber *V* ou *F*, que indica se o atributo é valido ou não. Para não montar uma regra muito especifica e assim torná-la uma regra ruim (baixa *fitness*), foi colocado uma condição de que 70% dos atributos recebessem *F*, porém os atributos que irão receber essa validação são determinados aleatoriamente.

Para atributos difusos é usado o pacote *função* que determinará o seu valor, estando impelmentados as funções *triangular* e a *trapezoidal*. Essas funções são montadas utilizando os valores da coluna da base, onde o usuário determinou que fosse difusa. Os atributos são determinados da seguinte maneira: na função *triangular*, primeiro é determinado qual é o valor mínimo e o valor máximo da coluna que corresponde onde cada função de pertinência atinge 100% *baixo* e 100% *alto* respectivamente (*pmax* e *pmin*). Depois de encontrado o mínimo e o máximo, é calculada a média entre eles (*pmed*) que é quando se encontra a outra variável da função de pertinência, onde recebe 100% *médio*, em seguida calcula-se a média do *pmed* como o *pmin* e a média do *pmed* com o *pmax* para calcular o final da função de pertinência *médio* como podem ver na Figura 6.5.

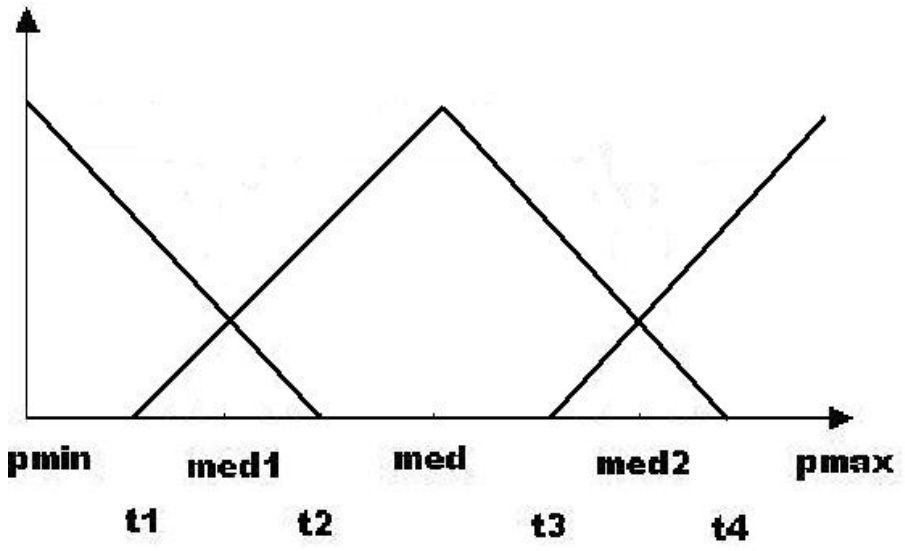

Figura 6.5: Função Triangular.

Depois das funções montadas, qualquer valor da coluna difusa plotado no eixo do x que retorna um valor no eixo do y, esse valor vai ser a porcentagem de quanto ele é *baixo*, *médio* ou *alto*.

Com um conjunto de atributos é formada uma regra. No exemplo abaixo, Tabela 6.1, é mostrado o conjunto:

Quadro 6.1: Atributos.

Status	=	*A14*	*V*
Duração_Mes	É	*Alto*	*F*
Histórico	=	*A30*	*F*
Finalidade	=	*A49*	*F*
Qntd_Credito	É	*Alto*	*V*
Economia	=	*A61*	*F*
Emprego_Desde	=	*A74*	*F*
Taxa_Prestaçao	É	*Alto*	*F*
Status_Pessoais	=	*A93*	*F*
Outros_Devedores	=	*A103*	*F*
Residente_Desde	É	*Alto*	*F*
Propriedade	=	*A23*	*F*
Idade	É	*Médio*	*F*
Outros_Planos	=	*A143*	*V*
Moradia	=	*A153*	*F*
Numero_Credito	É	*Baixo*	*F*
Ocupação	=	*A174*	*F*
Qntd_Manutençao	É	*Baixo*	*F*
Telefone	=	*A191*	*F*
Trabalhador_Estrangeiro	=	*A201*	*V*

A regra é formada apenas com os atributos que estão ativos, dessa maneira a regra formada com os atributos acima seria:

SE Status = A14 E Qntd_Credito É Alto E Outros_Planos = A143 E Trabalhador_Estrangeiro = A201 ENTÃO Classe = ?

Figura 6.6: Representação da Regra do Algoritmo.

Depois de montada a regra, determina-se qual a classe que vai ser utilizada. A classe da regra é capturada da seguinte maneira: é feito uma busca usando *count* para cada uma das classes, a maior é a escolhida. Por meio do exemplo mostrado anteriormente irão ser feitos dois *counts* usando a regra, um para cada *classe 1* e outro para a *classe 2*, como mostrado na Figura 6.7.

SELECT COUNT(*) FROM credito WHERE Status = A14 E Qntd_Credito É Alto E Outros_Planos = A143 E Trabalhador_Estrangeiro = A201 E Classe = 1

SELECT COUNT(*) FROM credito WHERE Status = A14 E Qntd_Credito É Alto E Outros_Planos = A143 E Trabalhador_Estrangeiro = A201 E Classe = 2

Figura 6.7: *Query* para Encontrar a Classe.

Depois de formada uma regra, o algoritmo monta outras regras até completar uma geração. As variáveis *número de regras* de cada geração e o *números de gerações* são determinados pelo usuário. Nesse momento entra a fase onde os operadores do AG são aplicados. Onde é implementada a seleção, cruzamento e mutação dos indivíduos (regras), como explicado no Capítulo 3. O operador de seleção e cruzamento escolhidos para serem usados no algoritmo é a seleção por roleta e cruzamento de dois pontos, respectivamente.

Na base existe uma coluna *cobertura*, que é iniciada com zero. É iniciada assim para indicar que todas as linhas possam ser capturadas para formar o atributo. No final de cada geração do AG, os melhores indivíduos (regras) são comparados com a base e as linhas da base que são cobertos pelas regras começam a receber o valor 1 para indicar que elas não podem ser usadas na próxima geração, como no exemplo da Figura 6.8 e na Tabela 6.2:

MELHOR 0 : Status = A11 Então 1

MELHOR 1 : Status = A14 Então 1

Figura 6.8: As Melhores Regras.

Tabela 6.1: Base Crédito.

Cobertura	Índice	Classse	Status	Dur_Mes	Historico	Finalidade	Qntd_Cred	Eco	Emp_Des
1	1	1	A11	6	A34	A43	1169	A65	A75
0	2	2	A12	48	A32	A43	5951	A61	A73
1	3	1	A14	12	A34	A46	2096	A61	A74
1	4	1	A11	42	A32	A42	7882	A61	A74
1	5	2	A11	24	A33	A40	4870	A61	A73
1	6	1	A14	36	A32	A46	9055	A65	A73
1	7	1	A14	24	A32	A42	2835	A63	A75
0	8	1	A12	36	A32	A41	6948	A61	A73
1	9	1	A14	12	A32	A43	3059	A64	A74
0	10	2	A12	30	A34	A40	5234	A61	A71

Após feita a cobertura, o algoritmo passa para a próxima geração, lembrando que as linhas que possuírem cobertura igual a 1 não podem ser usadas para formar atributos das próximas regras.

O algoritmo tem o critério de parada determinada pelo usuário, ele determina que a base seja coberta até certo limite, quando chegar nesse limite o algoritmo termina e retornas as regras encontradas. No programa, contém uma variável que conta o número de linhas cobertas, que verifica quantas vezes o atributo Cobertura é igual a 1. Se essa variável atingir 90% da base coberta o algoritmo atinge um dos critérios de parada.

O teste das regras do algoritmo utiliza a validação cruzada para verificar se as regras encontradas são regras verdadeiras. A validação cruzada é feita da seguinte maneira, dividi-se a base em dez partes, sendo uma para teste e as outras nove para treino. Isso é feito para todas as dez partes como no exemplo da Figura 6.9.

1	***Célula 1 para teste e as outras nove para treino.***
2	***Célula 2 para teste e as outras nove para treino.***
3	***Célula 3 para teste e as outras nove para treino.***
4	***Célula 4 para teste e as outras nove para treino.***
5	***Célula 5 para teste e as outras nove para treino.***
6	***Célula 6 para teste e as outras nove para treino.***
7	***Célula 7 para teste e as outras nove para treino.***
8	***Célula 8 para teste e as outras nove para treino.***
9	***Célula 9 para teste e as outras nove para treino.***
10	***Célula 10 para teste e as outras nove para treino.***

Figura 6.9: Validação.

6.6 - RESULTADOS

A ferramenta desenvolvida foi testada em bases de dados de domínio público (NEWMAN *et. al.*, 2004). As duas bases selecionadas foram: *WINE RECOGNITION DATA* e *CRÉDITO*.

A primeira base possui 4170 exemplos com 9 atributos, sendo 8 numéricos e 1 nominal. A segunda base possui apenas 170 exemplos com 13 atributos, sendo os 13 numéricos.

Os resultados obtidos são comparados com os algoritmos *J48, Decision Stump, Naive Bayes, Bagging, REPTree* e *NBTree* da ferramenta *WEKA* (The University of Waikato, 2006), através do cálculo do erro médio absoluto, como pode ser visto na Fórmula 6.1. Os algoritmos foram escolhidos de maneira aleatória e a comparação é feita através da média dos erros, baseado em alguns trabalhos (FERNANDES & MENEZES, 2005) onde eles utilizam esse método para comparação.

$$erro = \frac{\left\| p(1) \right| + ... + \left| p(n) \right\|}{n}, \qquad (6.1)$$

sendo p a probabilidade das linhas não cobertas e n o total das linhas do banco

Foram feitos testes usando várias variáveis. Para a base *WINE RECOGNITION DATA* na Figura 6.10, a função de pertinência usada é a *triangular*, o *número de regras* e de *gerações* a cada treino é de 100 e 60, respectivamente (Apêndice B). Já na Figura 6.11 só foi trocado a função de pertinência, agora a usada é a *trapezoidal* (Apêndice C).

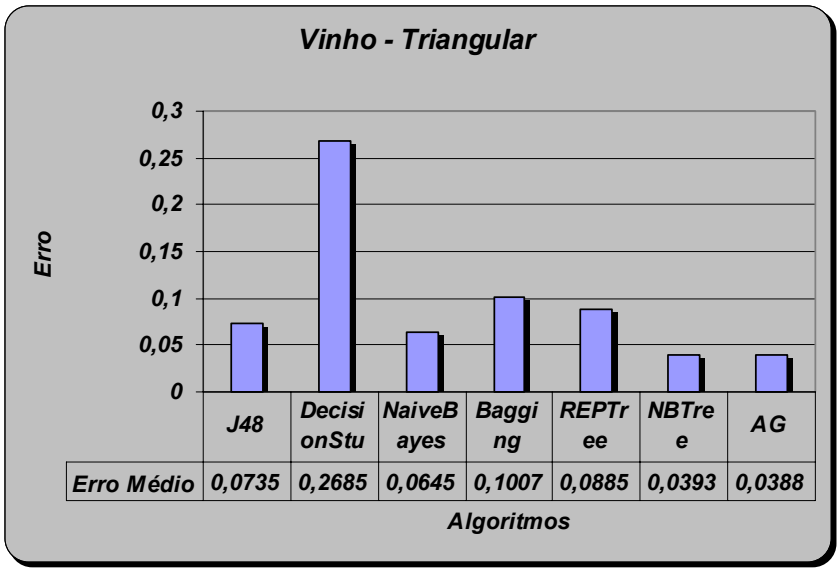

Figura 6.10: Resultados Usando a Função Triangular na Base WINE.

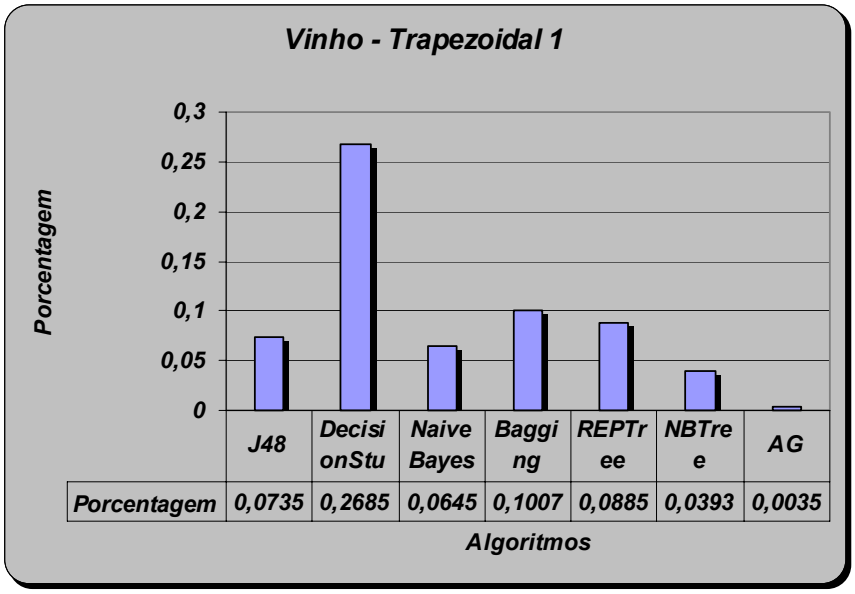

Figura 6.11: Resultado Usando a Função Trapezoidal na Base WINE.

A Figura 6.10 mostra que o algoritmo do AG ganhou todos os outros e o único que ficou bem próximo foi o NBTree, porém trocando a função de pertinência pela função *trapezoidal*, como pode ser visto na Figura 6.11, o AG se tornou praticamente imbatível.

Para a base *CRÉDITO* na Figura 6.12 a função de pertinência usada é a *triangular*, o *número de regras* e de *gerações* por treino é de 100 e 60, respectivamente (Apêndice D). Já na Figura 6.13 só foi trocado a função de pertinência, agora a usada é a *trapezoidal* (Apêndice E).

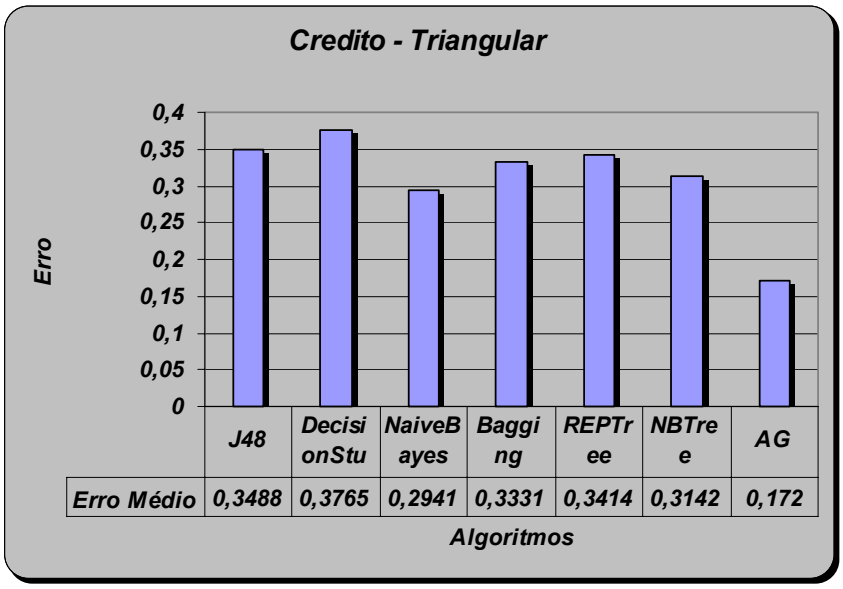

Figura 6.12: Resultado Usando a Função Triangular na Base CRÉDITO.

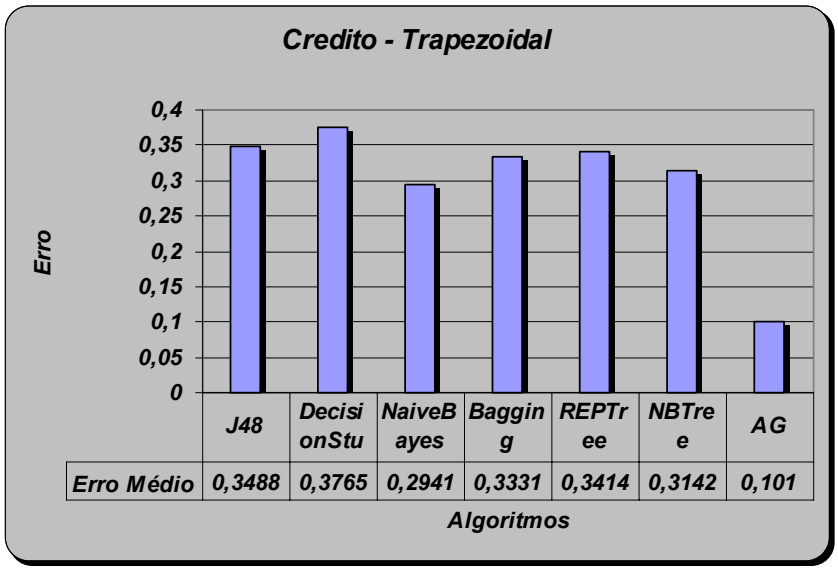

Figura 6.13: Resultado Usando a Função Trapezoidal na Base CRÉDITO.

As Figuras 6.12 e 6.13 mostram que o algoritmo AG ganha de todos os outros algoritmos testados e que nenhum deles conseguiram se equiparar ao algoritmo desenvolvido.

Na Figura 6.14 e na Figura 6.15 são apresentadas as árvores da base *WINE RECOGNITION DATA* com os algoritmos *J48* e *REPTree*, respectivamente.

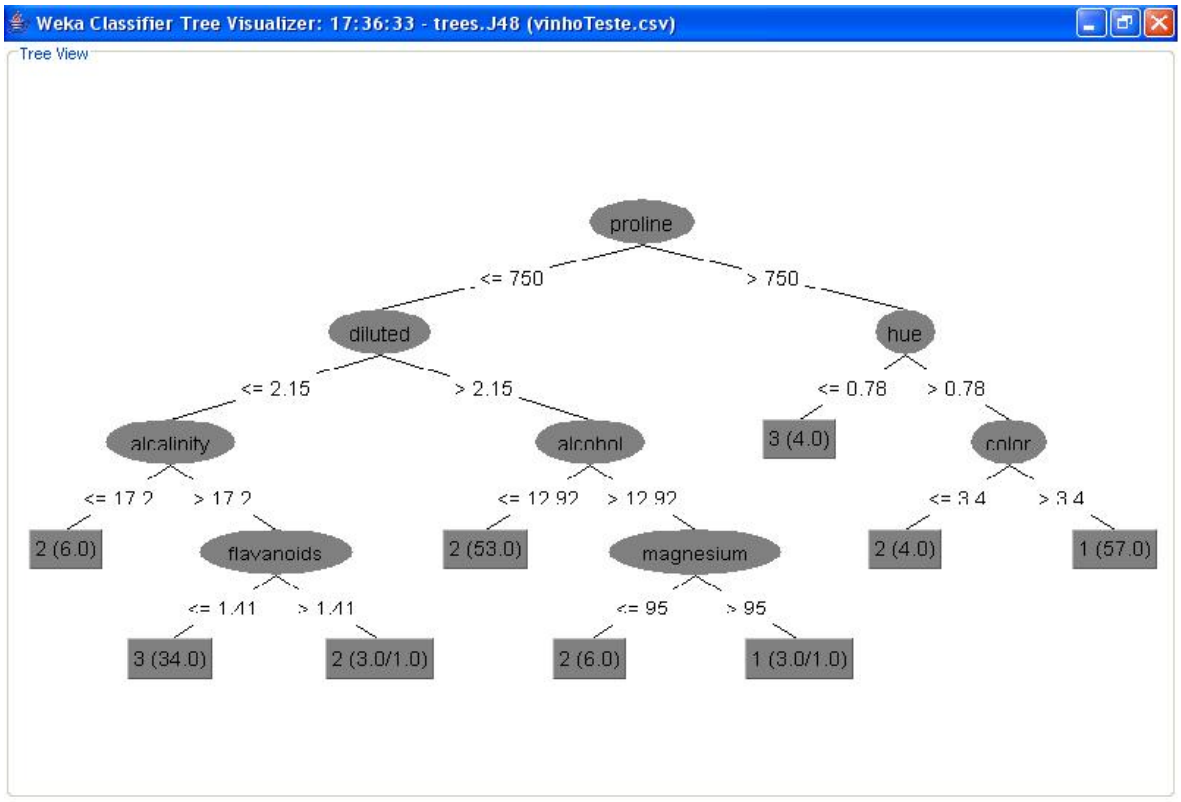

Figura 6.14: Árvore *J48 WINE*.

No caso do algoritmo J48 o tamanho da árvore é de 17 nós. Devido a ela não ser muito grande não se tem muitos problemas para a compreensão das regras formadas por ela, porém quando a árvore se tornar demasiadamente grande o usuário encontra dificuldade para interpretar as regras formadas pela árvore. As regras feitas foram:

SE proline ≤ 750 E diluted > 2.15 E alcohol > 12.92 E magnesium > 95: 1 (3.0/1.0).

SE proline ≤ 750 E diluted ≤ 2.15 E alcalinity > 17.2 E flavanoids ≤ 1.41: 3 (34.0).

SE proline > 750 E hue > 0.78 E color ≤ 3.4: 2 (4.0).

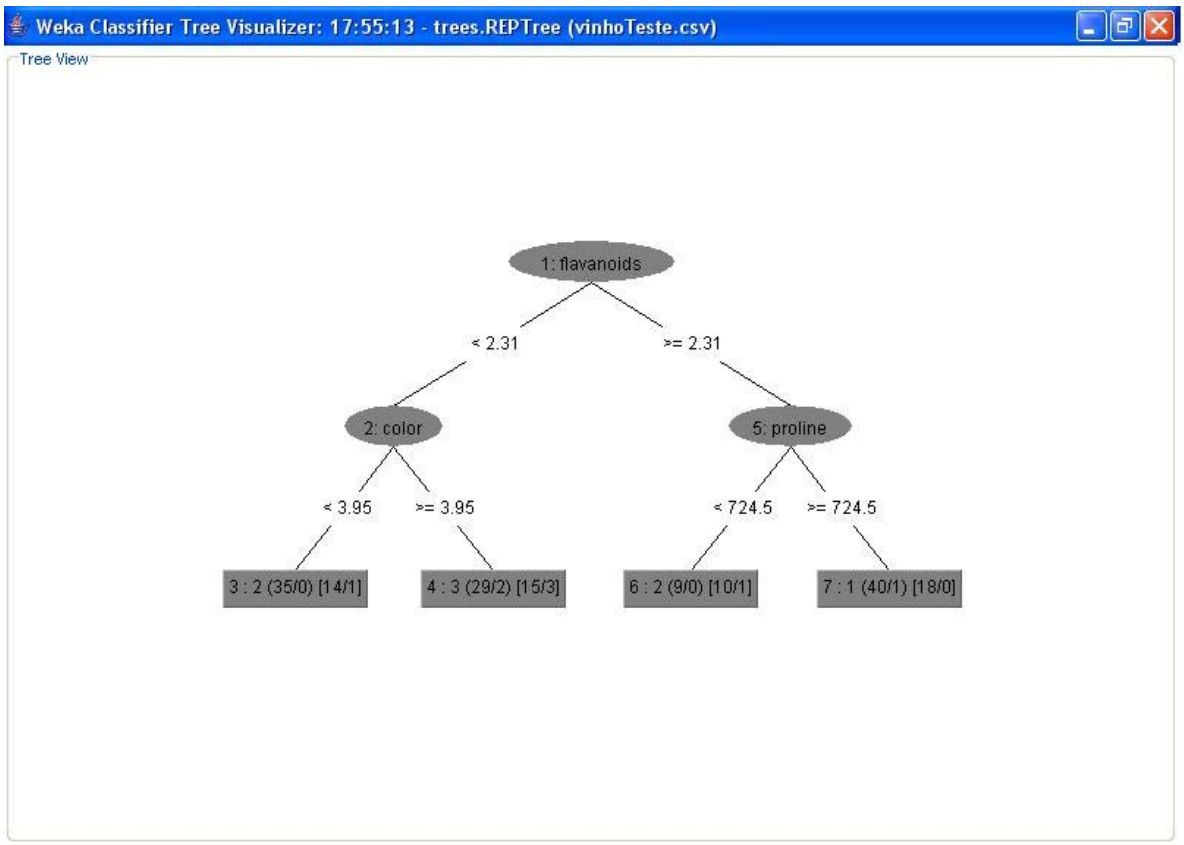

Figura 6.15: Árvore *REPTree WINE*.

No caso do algoritmo REPTREE o tamanho da árvore é de 7 nós. Nesse exemplo se torna ainda mais fácil o entendimento das regras. As regras feitas foram:

SE flavanoids < 2.31 E color < 3.95 : 2 (35/0) [14/1].

SE flavanoids < 2.31 E color ≥ 3.95 : 3 (29/2) [15/3].

SE flavanoids ≥2.31 E proline < 724.5 : 2 (9/0) [10/1].

SE flavanoids ≥ 2.31 E proline ≥ 724.5 : 1 (40/1) [18/0]

Na Figura 6.16 e na Figura 6.17 são apresentadas as árvores da base *CRÉDITO* com os algoritmos *J48* e *REPTree*, respectivamente.

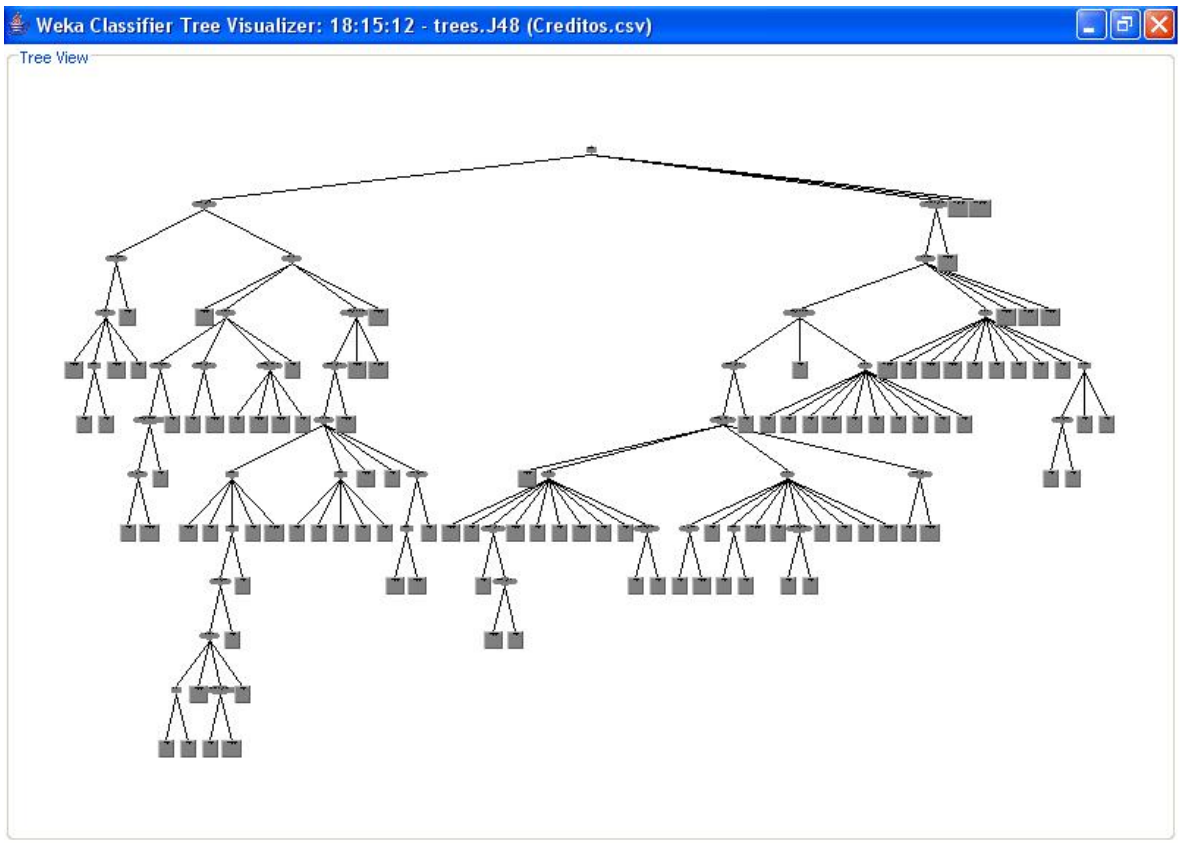

Figura 6.16: Árvore *J48 CRÉDITO*.

No caso do algoritmo J48 o tamanho da árvore é de 146 nós. Aqui já se pode notar alguma dificuldade de entender as regras devido à árvore ser de porte maior. Além da dificuldade de identificar a regra na mesma, por a árvore ser muito ramificada não foi possível colocar todas as regras montadas pelo algoritmo, sendo assim foram selecionadas aleatoriamente apenas algumas regras:

SE status = A11 E duracao_mes > 11 E ocupacao = A173 E outros_devedores = A101 E duracao_mes ≤ 30 E economias = A61 historico = A32 E telefone = A191 E numero_cred ≤ 1 propriedade = A123 E quantidade_credito ≤ 1386: 2 (3.0).

SE status = A12 E quantidade_credito ≤ 9857 E economias = A61 E outros_devedores = A101 E duracao_mes ≤ 42 status_pessoai = A92 finalidade = A42 E duracao_mes > 10 E duracao_mes ≤ 21: 1 (6.0/1.0).

SE status = A13: 1 (63.0/14.0).

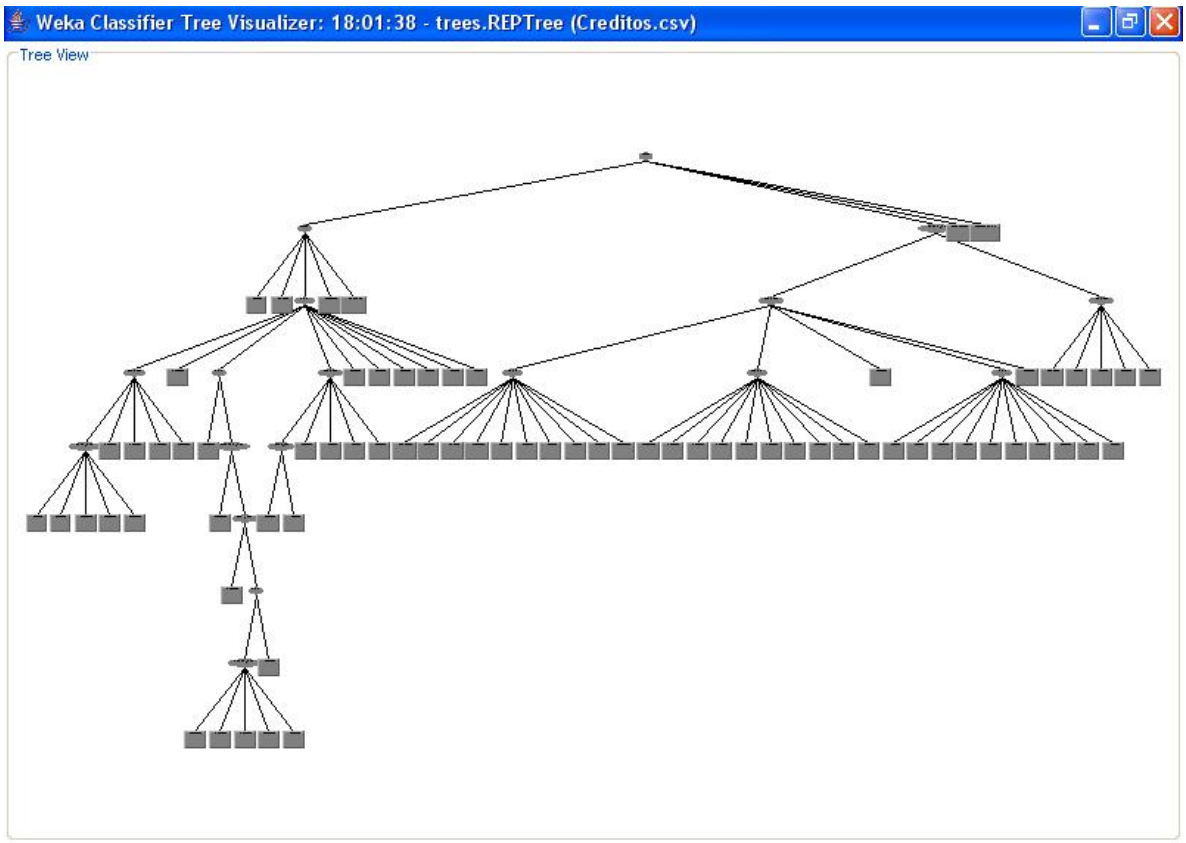

Figura 6.17: Árvore *REPTree CRÉDITO*.

No caso do algoritmo REPTree o tamanho da árvore é de 92 nós. Aqui acontece o mesmo problema da Figura 6.7. Devido à árvore ser de porte maior fica difícil o entendimento e a identificação das regras na mesma. Por esse motivo não foi possível apresentar todas as regras montadas pelo algoritmo, sendo assim foram selecionadas aleatoriamente apenas algumas regras:

SE status = A11 E historico = A32 E finalidade = A42 E idade >= 20.5 E quantidade_credito ≥ 2309 E taxa_presta >= 1.5 E idade < 43.5 E emprego_desde = A73 : 1 (4/1) [4/1].

SE status = A12 E quantidade_credito < 7834 E economias = A61 E finalidade = A43 : 1 (26/6) [22/5].

SE status = A13 : 1 (44/12) [19/2].

Por último, são mostradas algumas das regras encontradas pelo AG, para se fazer comparações com a da ferramenta desenvolvida. As primeiras são as regras da base *WINE*.

SE FLAVANOIDS É Medio1 E ASH É Baixo1 Então 1.

SE DILUTED É Baixo1 E NONFLAVANOIDS É Baixo1 E ALCALINITY É Baixo1 Então 2.

SE ALCALINITY É Baixo1 Então 3.

Agora serão apresentadas as regras da base *CRÉDITO*.

SE Numero_Creditos É Baixo1 Então 1.

SE Trabalhador_Estrangeiro = A202 E Quantidade_Manutencao É Baixo1 Então 1.

SE Numero_Creditos É Baixo1 Então 2.

6.7 DISCUSSÃO

Como foi mostrado nas figuras dos resultados, pode-se ver que o algoritmo implementado correspondeu às expectativas e não perdeu para nenhum dos outros algoritmos.

Um dos motivos que o AG não perdeu para os algoritmos J48 e Reptree é porque ao se trabalhar com árvore de decisão lida-se com o chamado *overfitting*, que seria a não capacidade de generalização das regras. Elas descrevem muito bem os exemplos trabalhados nos treinos, por isso qualquer exemplo ligeiramente diferente pode originar um erro de classificação.

O algoritmo de classificação J48 é a versão do Weka do algoritmo C4.5. O C4.5 e o Reptree foram escolhidos por serem algoritmos bem conhecidos e por gerar regras de fácil compreensão, porém como se pode perceber na Tabela 6.3 as regras da ferramenta desenvolvida são mais compreensíveis para o usuário do que as da ferramenta *WEKA*. Isso ocorreu devido a Lógica Difusa que ajuda também na interpretação da regra para os seres humanos.

Tabela 6.2 – Regras dos Algoritmos.

Regra 1 Crédito J4.8	*SE status = A11 E duracao_mes > 11 E ocupacao = A173 E outros_devedores = A101 E duracao_mes ≤ 30 E economias = A61 historico = A32 E telefone = A191 E numero_cred ≤ 1 propriedade = A123 E quantidade_credito ≤ 1386: 2.*
Regra 2 Crédito J48	*SE status = A12 E quantidade_credito ≤ 9857 E economias = A61 E outros_devedores = A101 E duracao_mes ≤ 42 status_pessoai = A92 finalidade = A42 E duracao_mes > 10 E duracao_mes ≤ 21: 1.*
Regra 3 Crédito J48	*SE status = A13: 1.*
Regra 1 Crédito Reptree	*SE status = A11 E historico = A32 E finalidade = A42 E idade >= 20.5 E quantidade_credito ≥ 2309 E taxa_presta >= 1.5 E idade < 43.5 E emprego_desde = A73 : 1.*
Regra 2 Crédito Reptree	*SE status = A12 E quantidade_credito < 7834 E economias = A61 E finalidade = A43 : 1.*
Regra 3 Crédito Reptree	*SE status = A13 : 1.*
Regra 1 Crédito AG-Difuso	*SE Numero_Creditos É Baixo1 Então 1.*
Regra 2 Crédito AG-Difuso	*SE Trabalhador_Estrangeiro = A202 E Quantidade_Manutencao É Baixo1 Então 1.*
Regra 3 Crédito AG-Difuso	*SE Numero_Creditos É Baixo1 Então 2.*

7 CONCLUSÃO E TRABALHOS FUTUROS

7.1 CONCLUSÃO

Um dos objetivos desse projeto é combinar o uso de Lógica Difusa, Algoritmo Genético e Mineração de Dados para descobrir regras interessantes e surpreendentes com maior compreensibilidade para os seres humanos. A definição de interessantes e surpreendentes segundo (FREITAS, 2002) é a geração de regras novas, porém não óbvias, por exemplo, montar uma regra onde SE mulher ENTÃO cabelo é longo. Esta regra não pode ser considerada surpreendente devido à maioria das mulheres terem cabelos longos.

Como se notou nos resultados desse trabalho, o AG é excelente para realizar descoberta de conhecimento independe do domínio da aplicação. Esse algoritmo apresenta grande relevância e, por conseguinte a necessidade de cada vez mais difundir seu uso.

Um dos objetivos principais dessa ferramenta foi atingido. O AG foi comparado com os algoritmos *J48, Decision Stump, Naive Bayes, Bagging, REPTree* e *NBTree* e como visto teve resultado superior em todas as bases não perdendo para nenhum dos algoritmos selecionados. Sendo assim se pôde mostrar a sua eficiência do uso do algoritmo híbrido, usando AG e Lógica Difusa, em comparação aos outros algoritmos.

Outro objetivo desse trabalho é montar regras mais compreensíveis com o uso de Lógica Difusa, como se pode notar na Tabela 6.3, também foi atingido. As Figuras 6.16 e 6.17 mostram regras muito complexas e de difícil entendimento para o ser humano, já as regras feitas pelo AG se nota maior compreensibilidade nelas.

Os algoritmos apresentados nas Figuras 6.14 e 6.15 não formaram regras muito grandes devido à base WINE ter apenas 170 linhas. Porém como foi visto nas outras árvores feitas, Figura 6.16 e 6.17, devido às árvores serem muito ramificada o tamanho da regra se torna demasiadamente grande e de difícil entendimento para os seres humanos.

7.2 TRABALHOS FUTUROS

- Os resultados obtidos são muito promissores e como trabalho futuro pretende-se melhorar a velocidade de processamento do algoritmo, cuja execução se mostrou mais demorada que as implementações do *WEKA*.

- Criar um componente que agregue outras tarefas de mineração de dados (associação, agregação, modelagem da dependência, regressão, detecção de desvios e padrão seqüencial) para tornar o programa mais completo.

- Criar um componente que agregue outras técnicas da inteligência computacional, como: Redes Neurais, Redes Bayesianas.

- Fazer uma interface para facilitar a operação do usuário.

Referências Bibliográficas

- AGRAWAL, Rakesh; Imielinski, Tomasz; Swami, Arun. Mining Association Rules between Sets of Items in Large Databases [on line] 1993. Disponível em http://rakesh.agrawal-family.com/papers/sigmod93assoc.pdf. Acesso em 15 jul. 2004.

- ALVARES, Reinaldo Viana. Mineração de Dados: Introdução e Aplicações. **Revista SQL Magazine**. Rio de Janeiro, v. 10, n. 10, p. 16-25, jun. 2004.

- ANDRADE, Marco Túlio Carvalho de. Computação – "Fuzzy" [on line] 2002. Disponível em http://www.pcs.usp.br/~pcs5711/transp/5711-cap4b-conjuntos-nebulosos-e-sua-representacao-2002ciclo1.pdf. Acesso em: 21. jun 2006.

- ARBEX, Eduardo Compasso; MIRANDA, Dhalila; REIS, Diego Belon dos. Associação Educacional Dom Bosco [on line]. Disponível em: http://www.inf.aedb.br/datamining/paginas/conceituacao.htm. Acesso em: 22 jun. 2004.

- ARROYO, José Elias C.; ARMENTANO, Vinícius A. Um Algoritmo Genético para Problemas de Otimização Combinatória Multiobjetivo. XXIII SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 6 a 9 de nov. Campos do Jordão – SP. 2001

- BAGLEY, J. D. The Behavior of Adaptive System Which Employ Genetic and Correlation Algorithms, Tese de Doutorado, Univ. Michigan, Ann Arbor. 1967

- BARRETO, Alexandre Serra. Considerações prévias à utilização empírica do Data-Mining [on line] 2005. Disponível em: http://www.sqlmagazine.com.br/Colunistas/AlexandreBarreto/02_DataMining.asp. Acesso em: 16 jun. 2006.

- BORGELT, Christian; KRUSE, Rudolf. Induction of Association Rules: Apriori Implementation. 14TH CONFERENCE ON COMPUTATIONAL STATISTICS (COMPSTAT 2002). Berlin – Alemanha, 2002.

- BORGES, Paulo Sérgio da Silva; GUIMARÃES, Márcio Ghisi. Um sistema de apoio à Dosimetria da Pena utilizando Fuzzy Logic [on line]. XX Congresso Nacional da Sociedade

Brasileira de Computação 2000. Disponível em http://www.niee.ufrgs.br/SBC2000/eventos/semish/semi009.pdf. Acesso em 21 jun. 2006.

- BRAUNER, Daniela Francisco; DANDOLINI, Gertrudes A.; SOUZA, João Arthur de. O Processo de Descoberta de Conhecimento em Banco de Dados: Um Estudo de Caso Sobre os Dados da Ufpel [on line] 2003. Disponível em http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2003/mono_daniela_brauner.pdf. Acesso em 28 set. 2003.

- BREMERMANN, H. J. Optimization Through Evolution and Recombination. In Self-Organizing System, edited by M. C. Yovis, G. T. Jacobi, and G. D. Goldstine, Washington D.C: Spartan Books, pp. 93-106. 1962.

- BREMERMANN, H. J.; ROGSON, M.; SALAFF, S. Global **Properties of Evolution Processes**. In Natural Automata and Useful Simulation, edited by H. H. Pattee, E. A. Edlsack, L. Fein, and A. B. Callahan, Washington D.C: Spartan Books, pp. 03-41. 1966.

- CAMPOS, Carlos Eduardo Aguilera. (2003). O desafio da integralidade segundo as perspectivas da vigilância da saúde e da saúde da família. Ciência e Saúde Coletiva, 8(2):569- 584, 2003.

- CANUTO, Anne Magály de Paula. Sistemas Baseados em Conhecimento e Sistemas Especialistas [on line] 1998. Disponível em: http://www.dimap.ufrn.br/~marcilio/IA/IA2004.1/SBC_completa.ppt. Acesso em: 21 de jun. de 2006.

- COSENZA, Harvey José Santos; VILLELA, Lamounier Erthal; MORE, Jesús Domech; CARVALHO, João Batista Brito de; COSENZA, Carlos Alberto Santos Ribeiro. Aplicação de um Modelo de Hierarquização como Instrumento para a Tomada de Decisão – caso de uma multinacional. XXVI ENEGEP – Fortaleza-CE 9 -11 de outubro de 2006.

- DANTAS. Benedito Tourinho; ROCHA, Armando; MARTINS, Daniel Borges Marques; COUTO, Daniel Lucena; ROCHA, Rogério. Conceito Básicos sobre Data Mining. **Revista Centefico**, Salvador, v. II, ano III, jul./dez. 2003.

- FERNANDES, Manuel do Couto; MENEZES, Paulo Márcio Leal de. Comparação entre Métodos para a Obtenção de Observações em Superfície Real no Maciço da geração de Mde para Tijuca- RJ. **RBC - Revista Brasileira de Cartografia** N° 52/02, 2005.

- FILHO, Kepler de Souza Oliveira. Fundamentos de Radiodiagnóstico por Imagem [on line] 1999. Instituto de Física da UFRGS. Disponível em http://www.if.ufrgs.br/ast/med/imagens/node58.htm. Acesso em: 21 de jun. 2006.

- FOGEL, David B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.** 2nd ed. New York. IEEE Press, 2000.

- FRAZEN, A. S. **Simulation of Genetic System by Automatic Digital Computers. I. Introduction**. Australian J. of Biol. Sci., v 10, pp 484-491. 1957a.

- FRAZEN, A. S. **Simulation of Genetic System by Automatic Digital Computers. II. Effects of Linkage on Rates of Advance under Selection**. Australian J. of Biol. Sci., v. 10, pp 442-499. 1957b.

- FRAZEN, A. S. **Simulation of Genetic System by Automatic Digital Computers. VI. Epistasis**. Australian J. of Biol. Sci., v. 13, pp 150-162. 1960.

- FRAZEN, A. S. **Simulation of Genetic System**. J. of Theoretical Biology, v. 2, pp. 329-346. 1962.

- FRAZEN, A. S. **Comments on Mathematical Challenges to the Neo-Darwinian Concept of Evolution**. In Mathematical Challenges to the Neo-Darwinian Interpretation of Evolution, edited by P. S. Moorhead and M. M. Kaplan, Philadelphia: Wistar Institute Press, p 107. 1967.

- FRAZEN, A. S. **The Evolution of Purposive Behavior**. In Purposive System, edited by H. von Foester, J. D. White, L. J. Peterson, and J. K. Russell, Washington, D.C: Spartan Books, pp 15-23. 1968.

- FREITAS, Alex. **A Genetic Algorithm for Generalized Rule Induction**. In: ROY, R. et al. Advances in Soft Computing - Engineering Design and Manufacturing. 3rd ON-LINE WORLD CONFERENCE ON SOFT COMPUTING. Springer-Verlag. 340-353. 1999.

- FREITAS, Alex. **Data Mining and Knowledge Discovery with Evolutionary Algorithms**. Berlin: Springer Verlag, 2002.

- GOLDBERG, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning, Adison-Wesley, 1989.

- GOMIDE, Fernando Antonio Campos; ROCHA, Armando de Freitas. NeuroFuzzy Controllers. Científico Internacional, IFAC - LCA'92, Vol. S/N, Viena, ITALIA, 1992

- GONÇALVES, Aruanda Simões; ALVES, João Bosco da Mota. Mineração de Dados Para Modelagem de Dependência Usando Algoritmos Genéticos 2001.

- HAN, J.; KAMBER, M. Data Mining: Concepts and Techniques 1st ed., San Francisco California, USA, Morgan Kaufmann. Publishers, 2001

- HERRERA, F.; FÉRNANDEZ, J. C.; RIVAS, V. M.; ALCALÁ-FDEZ, J;SÁNCHEZ, L.; GARCÍA, S.; JESUS, M. J. del; VENTURA, S.; GARRELL, J. M.; OTERO, J.; ROMERO, C.; BACARDIT, J.; KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems [on line] 2007. Disponível em http://sci2s.ugr.es/keel/description.php. Acesso em 18 de jul. 2007.

- HOLLAND, J. H. A new kind of turnpike theorem. Bulletin of the American Mathematical Society, 75(6) 1311–1317 pg. 1969.

- HOLLAND, J. H. Genetic algorithms and the optimal allocation of trials. SIAM Journal on Computing, 2(2), 88–105 pg. 1973.

- Java SE - Java Database Connectivity (JDBC) [on line] 2006. Disponível em http://java.sun.com/javase/technologies/database.jsp. Acesso em 30 ago. 2006.

- KLEMETTINEN, Mika; MANNILA, Heikki; RONKAINEN, Pirjo; TOIVONEN, Hannu; VERKAMO, Inkeri. Finding Interesting Rules from Large Sets of Discovered Association Rules [on line] 2002. Disponível em: http://www.acm.org/sigmod/dblp/db/indices/a-tree/a/Agrawal:Rakesh.html. Acesso em: 22 out. 2003.

- KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection**. Cambridge: MIT Press. 819p. 1992.

- LUCAS, Diogo Correa; PALAZZO, Luiz Antônio Moro; FERRUGEM, Anderson Priebe. Algoritmos Genéticos: um estudo de seus conceitos fundamentais e aplicação no problema de grade horária [on line] 2000. Disponível em: http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2000/Mono-Diogo.pdf. Acesso em: 28 set. 2003.

- MIRANDA, Marcio Nunes de. Algoritmos Genéticos: Fundamentos e Aplicações [on line]. Disponível em: http://www.gta.ufrj.br/~marcio/genetic.html. Acesso em: 28 set. 2003.

- MOTTA, Custódio Gouvêa Lopes da. Introdução a Técnicas de Data Mining – DM [on line] 2004. Disponível em http://www.sqlmagazine.com.br/Colunistas/AlexandreBarreto/02_DataMining.asp. Acesso em: 16 jun. 2006.

- MURAKAMI, Airton Minoru; BRINATI, Hernani Luiz. Projeto de Embarcações de Recreio utilizando o Algoritmo Genético [on line]. Produção em Iniciação Científica da EPUSP – 2002 [on line]. Disponível em: http://www.poli.usp.br/PesquisaPoli/Publicacoescpq/ProducaoIC2002/pdfs/pnvair02.pdf. Acesso em: 22 jun. 2004.

- NAVEGA, Sergio. Princípios Essenciais do Data Mining [on line] 2002. http://www.intelliwise.com/reports/i2002.pdf. Acesso em 22 de jun. 2004.

- NETO, Pedro Soares da Silva. Projeto e Análise de Algoritmos [on line] 2001. Disponível em: http://ipanema.ime.eb.br/~sneto/paa/Cap1.pdf . Acesso em: 19 set. 2003.

- NEWMAN, D. J.; Hettich, S.; Blake, C. L.; Merz, C. J. (1998). "UCI Repository of machine learning databases" [on line]. Disponível em: http://www.ics.uci.edu/~mlearn/MLRepository.html. Acesso em: 22 jun. 2004

- NOGUEIRA, Rilson Batista. Os Algoritmos Genéticos na Mineração de Dados [on line] 2003. CienteFico Ano III, v. II. Salvador-Ba. Julho - Dezembro de 2003. Disponível em http://www.cientefico.frb.br/Textos%20CienteFico%202003.2/INFO/Banco%20de%20Dad

os/Os%20Algoritmos%20Gen%E9ticos%20na%20Minera%E7%E3o%20de%20Dados.pdf. Acesso em 19 set. 2004.

- POLITI, Jacques. Implementação de uma Metodologia para Mineração de Dados Aplicada ao Estudo de Núcleos Convectivos. Dissertação de Mestrado, INPE. São Jose dos Campos-SP 2006.

- REZENDE, Solange Oliveira. **Sistemas Inteligentes – Fundamentos e Aplicações**. São Paulo: Manole, 2003.

- ROBIN, Jacques. Tipologia do Conhecimento de Saída da Mineração de Dados [on line]. Disponível em www.cin.ufpe.br/~compint/aulas-IAS/kdd-011/MiningOutput.ppt. Acesso em 22 jun. 2004.

- RODRIGUES, Ernesto Luis Malta; POZO, Aurora T. R. EVOLUÇÃO DE FUNÇÕES EM PROGRAMAÇÃO GENÉTICA ORIENTADA A GRAMÁTICAS [on line] 2002. Disponível em http://www.fesppr.br/~ernesto/Dissertacao.pdf. Acesso em 28 set. 2003.

- ROMÃO, Wesley; FREITAS, Alex A.; PACHECO, Roberto C. S. Uma Revisão de Abordagens Genético-Difusas para Descoberta de Conhecimento em Banco de Dados [on line] 2000. Disponível em http://www.periodicos.uem.br/ojs/index.php/ActaSciTechnol/article/view/3074/2219. Acesso em 27 jul. 2004.

- ROSENBERG, R. **Simulation of Genetic Population with Biochemical Properties.** Tese de Doutorado, Univ. Michigan, Ann Arbor. 1967.

- SALVADOR, Otávio; Introdução a Algoritmos Genéticos [on line] 2003. Disponível em: http://www.freedom.ind.br/otavio/trab/introducao_algoritmos_geneticos.pdf . Acesso em: 28 set. 2003.

- SANTOS, Rafael. Data Mining: Conceitos e Algoritmos [on line] 2006. Disponível em: http://www.lac.inpe.br/~rafael.santos/Docs/ci.dm.pdf. Acesso em 14 set. 2006.

- SANTOS, Joelma Carla. Seleção de Atributos usando algoritmos genéticos para classificação de regiões. XIII Simpósio Brasileiro de Sensoriamento Remoto. Florianópolis-SC, 21-26 de abril 2007.

- SEIXAS, José Alberto; GIMENES, Alberto. Data Mining – Mineração de Dados [on line] 2000. Disponível em: http://br.geocities.com/dugimenes/. Acesso em 19 set. 2003.

- SHAW, Ian S.; SIMÕES, Marcelo Godoy. **Controle e Modelagem Fuzzy**. São Paulo: Edgard Blücher LTDA, 1999.

- The University of Waikato [on line]. Disponível em http://www.cs.waikato.ac.nz/~ml/. Capturado em 30 ago. 2006.

- THOMÉ, Antonio Carlos Gay. Redes Neurais – Uma Ferramenta para KDD e Data Mining [on line] 2002. Disponível em: http://equipe.nce.ufrj.br/thome/grad/nn/mat_didatico/apostila_kdd_mbi.pdf. Acesso em 7 de Nov. 2005.

- TÚPAC, Yván J.; PACHECO, Marco Aurélio; VELLASCO, Marley; TANSCHEIT, Ricardo. Geração do Conjunto de Regras de Inferência para um Controlador Nebuloso usando Algoritmos Genéticos. 4o. SBAI – SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, São Paulo, SP, 08-10 de Setembro de 1999.

- WANG, Li-Xin. **A Course in Fuzzy System and Control**. New Jersey: Prentice Hall Series, 1997.