UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Implementação de uma Abordagem de Extração e Otimização de Regras Fuzzy Utilizando Sistemas Imunológicos Artificiais

João Roberto Del Ducca Cunha

Itajubá, Dezembro de 2010

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

João Roberto Del Ducca Cunha

Implementação de uma Abordagem de Extração e Otimização de Regras Fuzzy Utilizando Sistemas Imunológicos Artificiais

Dissertação submetida ao programa de pós-Graduação em Engenharia Elétrica como parte dos requisitos para a obtenção do Título de Mestre em Ciências em Engenharia Elétrica

Orientador: Prof. Leonardo de Mello Honório
Co-orientador: Prof. Armando M. Leite da Silva

Dezembro de 2010
Itajubá – MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

C972i

Cunha, João Roberto Del Ducca
Implementação de uma abordagem para extração e otimiza_ção de Regras Fuzzy utilizando Sistemas Imunológicos Artificiais / João Roberto Del Ducca Cunha. -- Itajubá, (MG) : [s.n.], 2010.
84 p. : il.

Orientador: Prof. Dr. Leonardo de Mello Honório.
Coorientador: Prof. Dr. Armando Martins Leite da Silva.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Sistemas Imunológicos Artificiais. 2. Fuzzy. 3. Mineração de dados. I. Honório, Leonardo de Mello, orient. II. Silva, Ar_mando Martins da, coorient. III. Universidade Federal de Itajubá. IV. Título.

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

João Roberto Del Ducca Cunha

Implementação de uma Abordagem de Extração e Otimização de Regras Fuzzy Utilizando Sistemas Imunológicos Artificiais

Aprovado em 15 de Dezembro de 2010

COMISSÃO DE AVALIAÇÃO

Prof Dr. Leonardo de Mello Honório – Orientador - UNIFEI

Prof Dr. Armando Martins Leite da Silva – Co-Orientador – UNIFEI

Prof. Dr. André Luis Marques Marcato – UFJF

Prof Dr. Carlos Henrique Valério de Moraes – UNIFEI

Temos horror às situações cujo controle não está nas nossas mãos.

A verdade, porém, é esta: as situações que realmente nos fazem

crescer são precisamente aquelas que não comandamos.

(Jacques Philippe)

Dedicatória

À minha esposa Sandra Regina da Silva Del Ducca Cunha, exemplo de força e superação.

Aos meus filhos Ana Luíza, João Vítor e João Rafael, minhas maiores fontes de alegria.

Aos meus pais e toda minha família, os maiores incentivadores da realização desse trabalho.

Agradecimentos

À Deus, que é a fonte maior de inspiração e de fortaleza, sem as quais esse trabalho não seria possível.

Aos colegas e professores que fazem parte do CRTI e GESIS, pela amizade e colaboração.

À FAPEMIG pelo importante apoio dado em todo esse período de realização do trabalho.

À UNIFEI por ter me acolhido como aluno junto a essa tradicional instituição de ensino e ter me dado todas as oportunidades e condições para o desenvolvimento e conclusão desse projeto.

Ao meu Orientador, Leonardo de Mello Honório, ao meu co-orientador Armando M. Leite da Silva, meus agradecimentos pela orientação, paciência e compreensão na realização deste trabalho.

Aos meus pais, pelo incentivo e motivação no período de realização deste trabalho.

À minha esposa Sandra por sua presença fundamental em minha vida.

À todos aqueles que direta ou indiretamente colaboraram para que este projeto fosse concluído.

RESUMO

Esse trabalho de pesquisa apresenta um estudo analítico sobre a aplicabilidade e o potencial de uso de uma abordagem evolutiva denominada Sistemas Imunológicos Artificiais (SIA) no processo de Mineração de Dados (MD).

Descreve o algoritmo CAISFLO, que realiza a extração e a evolução de um sistema baseado em lógica *fuzzy,* composto pela base de regras e suas funções de pertinência, utilizando-se de um algoritmo de SIA de seleção clonal denominado GbCLONALG. Exemplos para a melhor compreensão desse algoritmo são apresentados.

Detalha uma aplicação computacional que implementa o algoritmo CAISFLO, tanto em tarefas de classificação, quanto em tarefas de regressão.

Ao final apresenta uma análise comparativa do algoritmo CAISFLO com outros algoritmos existentes na literatura. Em tarefas de classificação, o CAISFLO obteve resultados similares. Em tarefas de regressão, mostrou-se bem eficiente, dando uma precisão muito grande para o conjunto de dados testado.

ABSTRACT

This research presents an analytical study on the applicability and potential use of an evolutionary approach called Artificial Immune Systems (AIS) in the process of Data Mining (DM).

This research describes the CAISFLO algorithm, which performs the extraction and the evolution of a system based on fuzzy logic, which consists of basic rules and their membership functions, using an clonal selection algorithm called GbCLONALG. Examples for the understanding of the algorithm are presented.

Details of a computer application that implements the algorithm CAISFLO in both classification tasks, and regression tasks is shown.

At the end presents a comparative analysis of CAISFLO algorithm with other algorithms in the literature. In classification tasks, the CAISFLO obtained similar results. In regression tasks, proved very effective, giving a very high accuracy for the tested data.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SGBD	Sistema de Gerenciamento de Banco de Dados
SIA	Sistemas Imunológicos Artificiais
MD	Mineração de Dados
CAISFLO	Co-Evolutionary Imunological System
KDD	Knowledge Data Discovery
AE	Algoritmos Evolucionários
AG	Algoritmos Genéticos
PG	Programação Genética
LCS	Learning Classifier System
TCD	Teoria de Conjuntos Difusos
RMS	Root Mean Square
IA	Inteligência Artificial
VT	Vetor Tangente

SUMÁRIO

CAPÍTULO 1
INTRODUÇÃO

1.1 Motivação

A introdução dos sistemas computacionais e sua grande evolução ocorrida nos últimos anos fizeram com que grandes quantidades de dados inerentes a diferentes processos de negócio fossem armazenadas eletronicamente. Neles está disponível uma imensa quantidade de informações altamente úteis e estratégicas, que possuem enorme potencial podendo ser utilizadas em áreas de planejamento, gestão e tomadas de decisão. Porém os Sistemas de Gerenciamento de Banco de Dados (SGBDs) convencionais não possuem a capacidade de converter esse imenso bloco de dados disponível em informações de formato compreensível para que possa ser utilizado pelos tomadores de decisão. Por esse motivo, houve a crescente necessidade de serem desenvolvidas técnicas que realizassem a extração do conhecimento e de aprendizagem baseado nos dados disponíveis. Com isso surgiram os conceitos de processo de Extração de Conhecimento de Banco em Dados (também conhecido como processo KDD, do inglês knowledge discovery in databases) e de Mineração de Dados (conhecido do inglês *Data Mining*), utilizando-se de ferramentas principalmente baseadas em inteligência artificial para revelar informações estratégicas escondidas em grandes massas de dados e descrever características do passado, assim como predizer tendências para o futuro (Gimenes, 2000).

Muitas áreas atualmente têm alcançado benefícios do conceito de mineração de dados na qual se pode citar a medicina com o diagnóstico de doenças e o empresarial com estratégias de marketing gerencial. A área de engenharia elétrica também tem aproveitado os pontos positivos desse conceito e de técnicas inteligentes para auxiliar em processos de automação, como controle e operação de sistemas elétricos, planejamento de sistemas e diagnósticos de falhas.

O sistema elétrico, por se tratar de um setor altamente estratégico e vital, necessita que os sistemas inteligentes que dão suporte aos tomadores de decisão sejam desenvolvidos de forma que possuam grande precisão a fim de se minimizar erros que possam provocar prejuízos financeiros e sociais. Assim, a extração do conhecimento e o processo de aprendizagem de máquina que irão servir de base para tomadas de decisão devem ser bem realizados.

Visando sempre a maior precisão nos resultados do processo de extração de informações, cada vez mais autores têm se dedicado em estudar e desenvolver algoritmos e ferramentas utilizando conceitos já consagrados, como Redes Neurais, Algoritmos Genéticos (AG), entre outros.

Uma dessas técnicas que ganhou bastante destaque e tem uso bastante difundido na área de Inteligência Artificial (IA) e de mineração de dados é a Lógica *Fuzzy*, pois ela trabalha com variáveis lingüísticas assemelhando ao modo humano de raciocínio, que facilita na hora da tomada de decisão. Além disso, ela tem a capacidade de lidar de forma gradual com valores quantitativos e contínuos, que é o formato da maioria dos dados disponíveis nos bancos de dados, diferentemente dos métodos tradicionais.

Tem tido também bastante êxito as técnicas baseadas em Algoritmos Evolucionários, principalmente os AG que é uma poderosa ferramenta para realização de otimização.

Nos últimos anos, ferramentas baseados em sistemas híbridos têm alcançado grande sucesso ao aliar os pontos positivos de duas ou mais metodologias de IA. Muitos autores têm proposto algoritmos que aliam os conceitos de lógica *fuzzy* com AG.

Mais recentemente, foi proposto o conceito de Sistemas Imunológicos Artificiais (SIA), que simulam o sistema imunológico dos seres vivos, principalmente o humano, para a realização de otimizações e reconhecimento de padrões.

Em (Vermaas, 2009) é proposto o algoritmo *Co-Evolutionary Imunological System* (CAISFLO) que gera um sistema baseado em regras fuzzy utilizando-se uma técnica de co-evolução, utilizando o conceito de SIA para escolha da melhor combinação de regras e da otimização das funções de pertinência.

Este trabalho tem como objetivo o desenvolvimento e análise de uma ferramenta de mineração de dados baseada no algoritmo CAISFLO e sua validação utilizando bases de dados de domínio público, comparando-se os resultados obtidos com outras abordagens disponíveis.

1.2 Organização da Dissertação

Este trabalho é dividido nas seguintes etapas:

No segundo capítulo, é apresentada uma revisão teórica sobre o conceito de mineração de dados e a utilização abordagens como a lógica *fuzzy* e algoritmos evolucionários no auxílio à extração de conhecimento, mostrando o que há na literatura sobre o assunto. É feita também uma apresentação breve sobre a técnica de SIA. Este capítulo tem o objetivo principal de mostrar que a lógica *fuzzy* e os algoritmos evolucionários têm sido amplamente estudados e utilizados em problemas de MD e como seu uso tem tido grande importância para a melhoria da precisão nos resultados.

No terceiro capítulo é apresentado o algoritmo CAISFLO onde é feita uma descrição de seu funcionamento, o método de extração das regras *fuzzy* e das funções de pertinência, além do detalhamento da utilização do conceito de SIA para dar maior eficiência ao processo de MD.

No quarto capítulo é feita uma descrição da ferramenta que foi desenvolvida aplicando o algoritmo CAISFLO. Também são apresentadas as telas e os diagramas de classes dessa aplicação.

No quinto capítulo a ferramenta é testada em algumas bases de dados conhecidas na literatura, e os resultados são analisados e comparados com outras técnicas disponíveis.

Na conclusão serão analisados os resultados gerados pela ferramenta e feita uma discussão qualitativa final sobre o algoritmo CAISFLO.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Revisão sobre Mineração de Dados

Mineração de Dados é uma das etapas do KDD que por sua vez é todo o conjunto de processos de descoberta de conhecimentos úteis a partir de dados históricos armazenados (Fayyad, 1996a). O processo de KDD é composto por várias etapas, sendo as principais ilustradas na Figura 2.1.

O KDD possui varias definições na literatura, sendo uma das mais aceitas a apresentada em (Fayyad, 1996a) que define KDD como *o processo não-trivial de identificar, em dados, padrões válidos, novos, potencialmente úteis e ultimamente compreensíveis*.

Já o processo de MD é definido como sendo *a etapa no processo de KDD que consiste na aplicação de análise de dados e algoritmos de descoberta de conhecimento que, sob limitações aceitáveis de eficiência computacional, produz uma enumeração particular de padrões a partir dos dados* (Fayyad,1996b).

Durante a etapa de MD, informações relevantes são extraídas a partir de um histórico de dados e disponibilizadas para diversas finalidades, como ações automáticas e tomadas de decisões futuras.

Os conjuntos de dados utilizados no processo de KDD para descoberta de conhecimento podem estar armazenados nos mais diversos formatos e os tipos de dados definem quais padrões ou relacionamentos a serem minerados. As

funcionalidades ou tarefas definem o propósito do uso do sistema de MD, o tipo de padrões ou relacionamentos entre registros e variáveis que podem ser utilizados. Em MD, geralmente duas tarefas, ou funcionalidades, básicas podem ser distinguidas na busca dos padrões de dados:

- ***Descrição -*** é a busca por padrões nos dados com o objetivo de apresentá-los ao usuário.
- ***Predição -*** é a busca por padrões nos dados com o objetivo de fazer previsões.

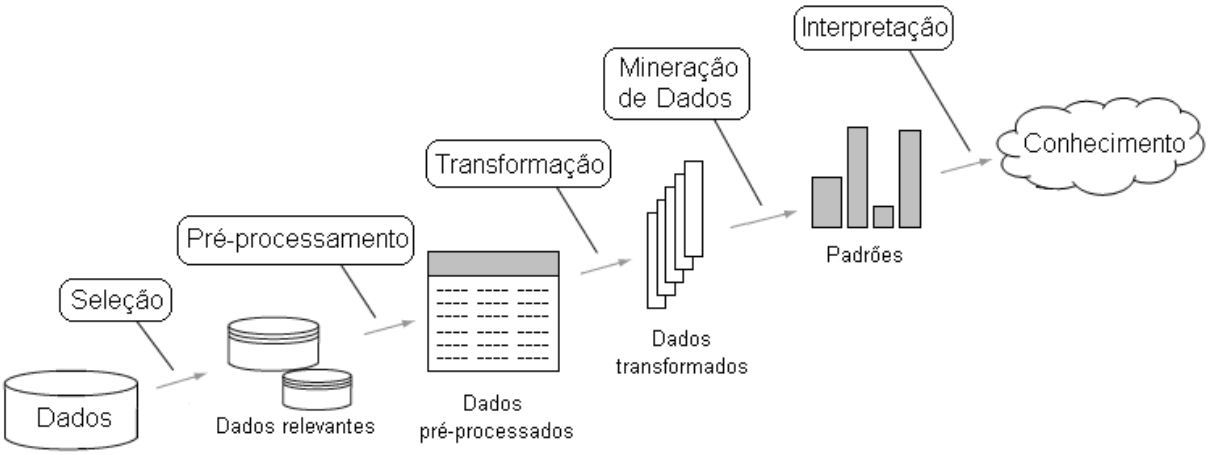


Figura 2.1 - Etapas do processo de KDD (Fayyad, 1996)

Para a realização dessas funcionalidades do MD, uma variedade de métodos são utilizados, alguns deles mais comuns citados em (Fayyad, 1996c) são:

- ***Classificação -*** é a função que associa cada dado individual em uma das diversas classes pré-definidas a partir de uma característica que o torna comum a um grupo de dados de uma dada classe;
- ***Regressão -*** é a função que mapeia cada dado individual por uma variável de previsão e descobre relações funcionais entre as variáveis;

- ***Agrupamento (Clusterização) -*** é a associação de cada dado individual a um ou mais cluster a partir de uma característica que o torna comum a um grupo de dados de um dado cluster. Os *cluster*s são definidos por meio do agrupamento de dados baseados em medidas de similaridade ou modelos probabilísticos;
- ***Sumarização -*** fornece uma descrição compacta para um subconjunto de dados;
- ***Modelo de dependência -*** descreve dependências significativas entre as variáveis. Podem ser estruturais que identificam quais variáveis são dependentes, e qualitativas que especificam os pesos das dependências utilizando uma escala numérica;
- ***Identificação de Associação -*** determina relações entre as variáveis.

Nesses métodos de MD, são utilizadas ferramentas e técnicas baseadas em algoritmos de aprendizado de máquina, reconhecimento de padrões e análises estatísticas. Essa interatividade entre as funcionalidades, técnicas e algoritmos utilizados em MD é ilustrada na Figura 2.2. Nas últimas décadas, ganharam destaque as técnicas baseadas em inteligência artificial, como a lógica *fuzzy*, redes neurais e algoritmos genéticos.

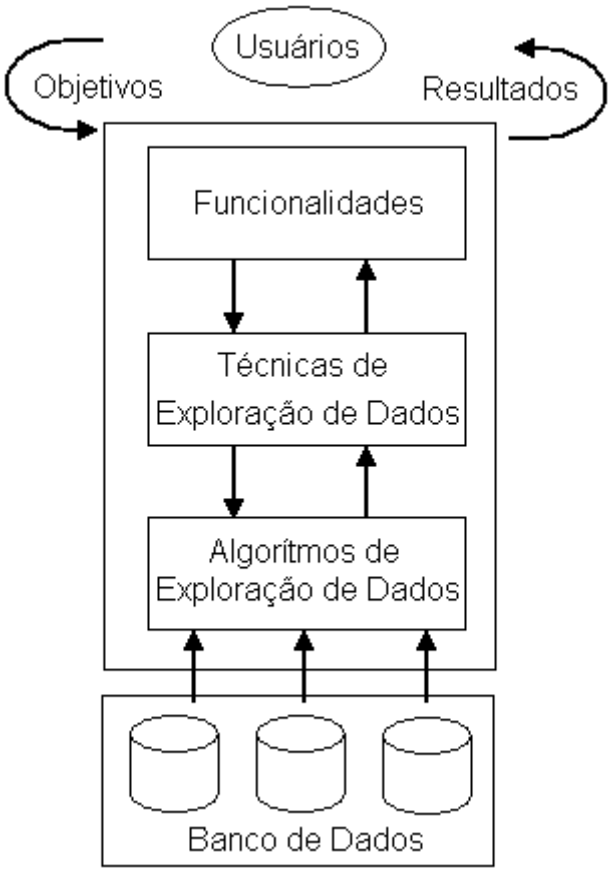


Figura 2.2 - Interatividade entre as funcionalidades e técnicas de MD

Esses algoritmos consistem em três componentes principais citados abaixo (Fayyad,1996b):

- ***Modelo de representação -*** é o formato utilizado para descrever os padrões encontrados na base de dados;
- ***Modelo de avaliação -*** mede a partir de funções, valores quantitativos que mostra o quanto um padrão descoberto na base de dados alcança os objetivos no processo de KDD;
- ***Método de busca -*** consiste de dois componentes: parâmetros de busca e modelos de busca. No parâmetro de busca, o algoritmo deve procurar por parâmetros que aperfeiçoam o modelo de avaliação.

As principais técnicas utilizadas nos algoritmos durante processo de MD são:

- ***Árvores de decisão -*** é um fluxograma em forma de árvore, onde os *nós* são os testes a serem aplicados a um atributo, os *ramos* são os resultados destes testes e as folhas representam a distribuição dos registros. É utilizada quando os dados a serem analisados possuem atributos diferentes e não existe uma forma lógica para a representação;
- ***Regras de associação -*** descreve a relação entre itens em uma base de dados. Classicamente, as regras de associação indicam que a presença de um item implica na presença de outro em uma transação;
- ***Redes Neurais -*** é um processador maciça e paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento e torná-lo disponível para o uso;
- ***Modelos Lineares -*** Redes Bayesianas - São diagramas que organizam o conhecimento numa dada área através de um mapeamento entre causas e efeitos.

Como será visto adiante, o foco principal deste trabalho será o método de classificação e regressão utilizando regras de classificação e associação.

2.2 MD Utilizando Lógica Fuzzy

A teoria de lógica *fuzzy*, suas técnicas e ferramentas têm sido amplamente aplicadas atualmente nas diversas etapas do KDD, como a seleção e preparação de dados, modelagem de dados no formato de conjuntos *fuzzy* e criação de sumários de dados *fuzzy*.

Como foi visto anteriormente, em muitas bases de dados onde ficam armazenados grandes conjuntos de informações, a mineração de dados pode ter uma grande importância ao obter informações interessantes e úteis. O setor comercial é um dos que mais pode se beneficiar com o uso dessas informações, pois a mineração de dados pode revelar produtos que são comercializados em conjunto, e assim, através do uso de estratégias de marketing, entre outras decisões, podem-se potencializar as vendas desses itens. Um algoritmo bastante eficaz e utilizado para a mineração de dados voltada para esse propósito é o APRIORI, proposto em (Agrawal,1994), que extrai regras de associação a partir de dados discretos.

Porém, em muitos tipos de aplicações da mineração de dados, quando os dados são contínuos e imprecisos, como os atributos quantitativos (temperatura, peso, altura, etc.), a classificação desses dados em conjuntos discretos podem provocar perdas de informações, ou gerar respostas imprecisas. Para esses casos é bastante eficiente a lógica *fuzzy* proposta em (Zadeh, 1965), a qual fornece uma transição gradual de um conjunto para o outro para uma determinada variável. Para isso, utiliza-se da pertinência, ou grau de confiança, de um determinado elemento quantitativo para o atributo. A pertinência é calculada utilizando-se as funções de

pertinência, também conhecidas como *membership functions*, e medirá o quanto significativo um valor é com relação a uma variável nebulosa (conjunto *fuzzy*).

Como pode ser visto no exemplo da Figura 2.3, o atributo altura de uma pessoa pode ser divido em três conjuntos *fuzzy* (pequeno, médio e grande) e os graus de pertinência calculados através das funções de pertinência. Para um valor de altura igual a 1,80m são ativados os conjuntos *fuzzy* M, com pertinência igual a 0,7 e G com pertinência igual a 0,3. Esse processo de codificação de um valor numérico em valor lingüístico *fuzzy* é denominado *fuzzificação*.

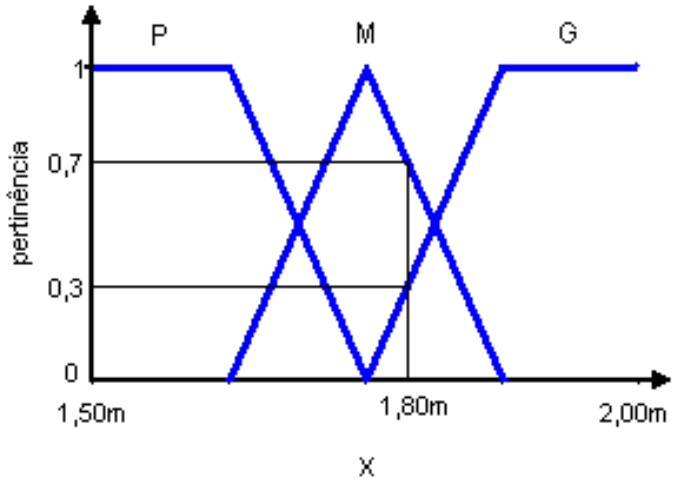


Figura 2.3 - Fuzzificação através das funções de pertinência

As principais contribuições da Teoria de Conjuntos Difusos (TCD) no processo de aprendizagem de máquina e MD são: gradualidade, interpretabilidade, robustez, representação da incerteza e incorporação do conhecimento *background* (Hüllermeier, 2005).

A seguir, são apresentadas algumas das principais aplicações de Lógica *fuzzy* em MD, de acordo com (Hüllermeier, 2005).

2.2.1 Analise de Agrupamentos Difusos

Nos métodos tradicionais de agrupamentos, também conhecido como clusterização, cada objeto é atribuído a um único grupo, também conhecido por *cluster*. Dessa forma, esses grupos são separados por fronteiras bem definidas. Porém na prática, em muitas das variáveis existentes na maioria dos ambientes modelados, as transições entre grupos não ocorrem de forma abrupta, mas sim de forma gradual. Por esse motivo, o uso da lógica *fuzzy* possui grande utilidade. No agrupamento por *fuzzy*, um objeto pode pertencer a diferentes grupos ao mesmo tempo, e a medida da adesão desse objeto a um determinado grupo é medido pelo grau de pertinência calculado através das funções de pertinência. Essas funções, atribuídas aos diferentes agrupamentos, correspondem como uma forma de partição da variável e possuem como valor máximo a unidade.

O principal algoritmo de agrupamento utilizando lógica *fuzzy* é o *Fuzzy C-Means* (FCM), apresentado em (Dunn, 1973) e aprimorado em (Bezdek, 1981), sendo usado principalmente em reconhecimento de padrões. Outros algoritmos de agrupamentoização usando lógica *fuzzy* podem ser vistos em (Höppner, 1999).

O agrupamento por lógica *fuzzy* tem se mostrado extremamente útil na prática sendo aplicada em diversas áreas, por exemplo, em aplicações recentes de bioinformática (Gasch, 2002).

2.2.2 Geração de Bases de Regras Fuzzy

O aprendizado de regras é a aplicação principal da TCD no processo de descoberta de conhecimento e MD. Isto é intuitivo justamente porque os sistemas *fuzzy* baseiam a etapa de inferência na utilização de regras armazenadas em uma base, que podem ser fornecidas por especialistas ou mesmo extraídas a partir de dados numéricos.

O processo de aprendizado de bases de regras *fuzzy* é útil em MD tanto para tarefas de classificação quanto de regressão, utilizando-se de diferentes tipos de modelos *fuzzy*.

Em funções de regressão, o sistema utiliza-se dos processos de fuzzificação e de defuzzificação, mapeando um valor numérico de entrada no formato *fuzzy*, que é processado pelo sistema, e posteriormente o valor de saída é convertido de volta a um valor numérico. Em modelos Takagi-Sugeno, o processo de defuzzificação não é necessário.

Nos métodos de classificação, os conseqüentes das regras são modelados por valores separados por categorias ou classes (ou seja, um conjunto *fuzzy singleton*). Para esse método, as regras são do tipo:

SE (tamanho $\in$ ALTO) e (peso $\in$ LEVE) ENTÃO (classe = A)

Existem na literatura várias propostas de técnicas de geração de sistemas *fuzzy*, na qual podem se destacar em (Wang, 1992), (Li, 2002), (Abe, 1995) e (Zhao, 2003).

Um aspecto importante a ser ressaltado no processo de aprendizagem das regras *fuzzy* é a utilização de métodos híbridos que combinam a TCD com outras metodologias, como algoritmos evolutivos e redes neurais. Algoritmos evolutivos são muitas vezes utilizados para otimizar uma base de regras *fuzzy* ou para pesquisar o espaço das bases de regra no potencial de uma forma mais (ou menos) de forma sistemática (Cordon, 2004). Muito interessante também são métodos *neuro-fuzzy* estudados em (Nauck,1997) em que a idéia é codificar um sistema *fuzzy* como uma rede neural e aplicar métodos padrão a fim treinar essa rede. Dessa forma, sistemas *neuro-fuzzy* combinam as vantagens de representação de sistemas nebulosos com a exibilidade e adaptabilidade das redes neurais (Hullemeier, 2005).

No método proposto por Wang e Mendel em (Wang, 1992), são extraídas as regras *fuzzy* do tipo *SE X=A ENTÃO Y=B*, onde X e Y são os atributos das variáveis, e A e B são os conjuntos *fuzzy.* Esses conjuntos são gerados dividindo-se o domínio do atributo em regiões. O grau de pertinência de um valor com relação a um conjunto fuzzy é calculado através das funções de pertinência, podendo estas ser dos tipos crescente, decrescente, triangular, trapezoidal entre outros. A primeira parte da regra representada por X=A é chamada de antecedente, enquanto a segunda parte representada por Y=B é chamada de consequente.

Nesse algoritmo, primeiramente o intervalo de domínio limitado pelos valores máximo e mínimo do conjunto de dados de cada variável (entradas e saída) é dividida em 2N+1 regiões *fuzzy*. Cada variável pode ser dividida em diferentes quantidades de regiões, porém o mais comum são N=1 (3 regiões) e N=2 (5 regiões).

Cada tupla[1] extraída do conjunto de dados contendo os valores de cada variável é então submetida às funções de pertinência, que determinam o grau de pertinência para cada valor com relação aos conjuntos *fuzzy*. Quando um dado ativa duas ou mais regiões diferentes, escolhe-se aquela que fornecer o maior grau de pertinência. Então, cada conjunto (ou linha) de dado gera apenas uma regra.

Para melhor ilustrar o processo descrito acima, seja o Exemplo 2.1, ilustrado pelas Figura 2.4, Figura 2.5 e Figura 2.6, com dois conjuntos de dados numéricos, compostos por 2 variáveis de entrada e uma de saída.

Exemplo 2.1 – Seja as seguintes tuplas formadas pelos valores de entrada e saída. O domínio das variáveis foi dividido nas regiões PP (muito pequeno), P (pequeno), M (médio), G (grande) e GG (muito grande):

- tupla 1 =($x_1^{(1)}$,$x_2^{(1)}$;$y^{(1)}$);
- tupla 2 = ($x_1^{(2)}$,$x_2^{(2)}$;$y^{(2)}$).

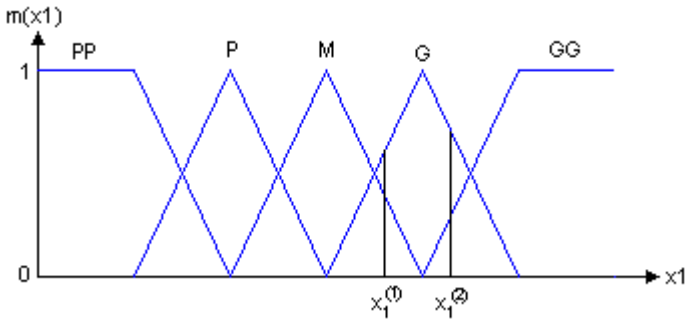


Figura 2.4 - Fuzzificação da variável x_1

[1] Cada linha de um banco de dados, formada por um conjunto de colunas, representa um registro.

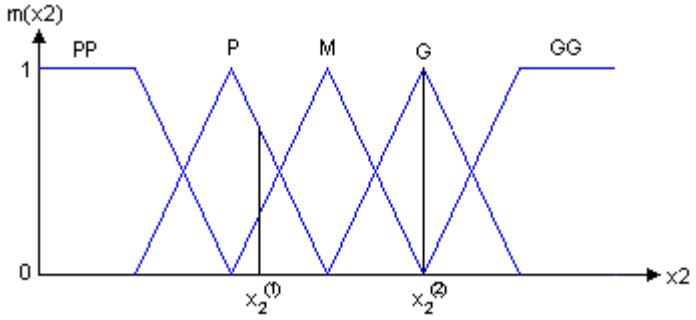


Figura 2.5 - Fuzzificação da variável x_2

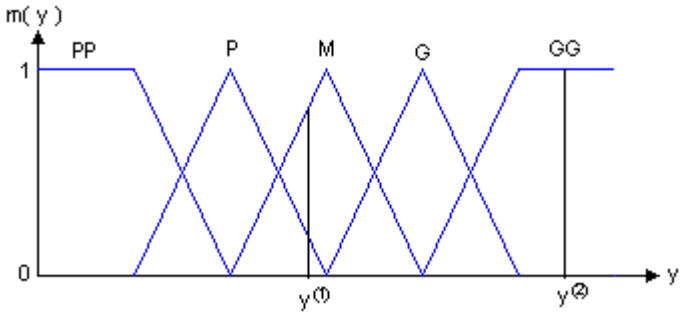


Figura 2.6 - Fuzzificação da variável y

Através das funções de pertinência, os valores numéricos são convertidos em variáveis lingüísticas (fuzzificação) e recebem pesos (graus de pertinência). Para o Exemplo 2.1, os graus de pertinência para cada elemento são mostrados na Tabela 2.1 e Tabela 2.2.

Tabela 2.1 - Graus de Pertinência para a tupla 1

	PP	P	M	G	GG
$X_1^{(1)}$	0	0	0,4	0,6	0
$X_2^{(1)}$	0	0,7	0,3	0	0
$Y^{(1)}$	0	0,2	0,8	0	0

Tabela 2.2 - Graus de Pertinência para a tupla 2

	PP	P	M	G	GG
$X_1^{(2)}$	0	0	0	0,7	0,3
$X_2^{(2)}$	0	0	0	1	0
$Y^{(2)}$	0	0	0	0	1

Assim, de acordo com o algoritmo, teremos para cada tupla as seguintes regras geradas:

$(x_1^{(1)},x_2^{(1)};y^{(1)}) \Rightarrow$ SE x_1 = G E x_2 = P ENTÃO y = M

$(x_1^{(2)},x_2^{(2)};y^{(2)}) \Rightarrow$ SE x_1 = G E x_2 = G ENTÃO y = GG

2.2.3 Árvores de Decisão Fuzzy

A árvore de decisão é um método de classificação de dados, representado por uma estrutura semelhante a uma árvore. Neste tipo de representação existe um sistema hierárquico de *nós* e *folhas* que são conectados por *ramos*. O *nó* representa uma decisão a ser tomada com um teste a ser aplicado sobre um atributo. Os *ramos* representam as decisões específicas tomadas a partir desse *nó*, representando um valor possível do atributo, levando a uma sub-árvore (nós secundários) ou *folhas*. As *folhas* representam valores das classes.

As árvores de decisão já induzidas, podem ser representadas como um conjunto de regras SE-ENTÃO, em que um caminho completo entre os *ramos* desde o *nó raiz* até uma *folha* é uma regra.

As árvores de decisão são induzidas utilizando geralmente os algoritmos ID3 (Quinlan, 1979) e C4.5 (Quinlan, 1993) e CART (Breiman, 1984). O princípio básico da maioria dos algoritmos de indução de uma árvore de decisão é a partição do conjunto de dados de exemplo de uma forma recursiva. Cada *nó* interno de uma árvore de decisão define uma partição dos exemplos atribuídos a esse *nó*. Isto é feito através da classificação de elementos de acordo com o valor de um atributo de entrada específica. Esse processo de partição dos exemplos termina quando o critério de parada escolhido é satisfeito e assim o nó se torna uma folha.

Esses algoritmos clássicos possuem a mesma desvantagem já mencionada nos métodos de agrupamentos quando em face de conjuntos de exemplos contínuos, tornando as árvores sensíveis a ruídos. Por isso, vários estudos foram realizados aliando a TCD como em (Adamo, 1980), (Lee, 1989), (Peng, 2001), (Weber, 1992), (Janikow, 1998) e (Olaru, 2003).

2.2.4 Análises de Associação Fuzzy

Os métodos baseados em dependências entre variáveis como regras de classificação e regressão e analise de associação, são bastante utilizados na etapa de MD. Apesar de esses métodos utilizarem da mesma expressão SE-ENTÃO, a diferença entre eles é que a análise de associação é mais descritiva, além de ser considerada de forma isolada, como um padrão em particular entre os dados. Já as

o modelo baseado em regras de classificação formam geralmente uma base de regras.

As regras de associação envolvem geralmente dois conjuntos de atributos binários, representado pela expressão SE *A* ENTÃO *B*, significando que se um objeto possui as características A, então é bem provável que possua também as características B. Essas regras de associação são avaliadas geralmente identificando a quantidade de objetos da base de dados que satisfaz a condição A e a conclusão B e a freqüência relativa entre os objetos que satisfazem B entre aqueles que satisfazem A. Uma regra é aceitável se satisfaz um critério de qualidade pré-estabelecido pelo especialista.

Como já mencionado, as regras de associação são bastante úteis na área comercial, encontrando associações entre itens de uma cesta de produtos que são adquiridas por um consumidor. Porem nem sempre os dados são discretos, podendo ser numéricos ou categóricos, como a idade de um funcionário e seu salário. Assim como visto nos itens anteriores, essa é a motivação principal para uso da TCD também em análise de associações.

Alguns estudos sobre modelos e aplicações de TCD aplicadas na mineração de regras de associação podem ser visto em (Chen, 2003) e (Delgado, 2003).

2.3 MD Utilizando Algoritmos Evolucionários

Algoritmos Evolucionários (AEs) são algoritmos de busca que usam os princípios baseados na evolução natural das populações e seleção das espécies

para desenvolver soluções de problemas (Goldberg, 1989). Utilizam-se da teoria das evoluções de Charles Darwin que apresenta o conceito de que indivíduos cujo genótipo que possui características mais favoráveis ao meio vão se tornando mais comuns às futuras gerações, ao mesmo tempo em que características menos favoráveis vão se tornando raras. Assim, as futuras gerações se tornam cada vez mais aptas e as espécies mais capazes prevalecem sobre as espécies menos capazes.

Os AEs são usados para a solução dos mais variados problemas, inclusive em problemas de MD, pela robustez e o método de busca adaptativa que localiza as soluções de forma global dentro do universo populacional, diferente da maioria dos métodos tradicionais e gulosos que realizam busca em soluções locais. Como resultado de sua pesquisa global, os AEs tendem a lidar melhor com as interações entre os atributos do que os métodos de mineração de dados gulosos (Freitas, 2002). Assim, os AEs possuem a capacidade de descobrir conhecimentos interessantes que podem ser esquecidos pelo método guloso.

Nos AEs, cada individuo em uma população representa um candidato à solução do problema, e evolui cada vez mais em direção à melhor solução. Os indivíduos são avaliados usando a função de aptidão (também conhecida como *fitness)*, que medirá a qualidade deste como solução do problema. Os melhores indivíduos, ou seja, aqueles que obtiverem o maior valor fornecido pela função de aptidão, são selecionados para se reproduzir e compor a próxima geração. Durante essa reprodução, eles sofrem operações baseados na genética, como o cruzamento, onde dois indivíduos trocam parte do material genético entre si, e a mutação, onde parte do material genético do indivíduo é trocado por um material genético aleatório. Esses novos indivíduos substituem os indivíduos pais, formando

a nova geração. Esse processo se repete de forma iterativa, até ser satisfeito um critério de parada que pode ser um número fixo de gerações ou um valor aceitável para a solução ser encontrada.

A Figura 2.7 ilustra o diagrama do processo descrito acima.

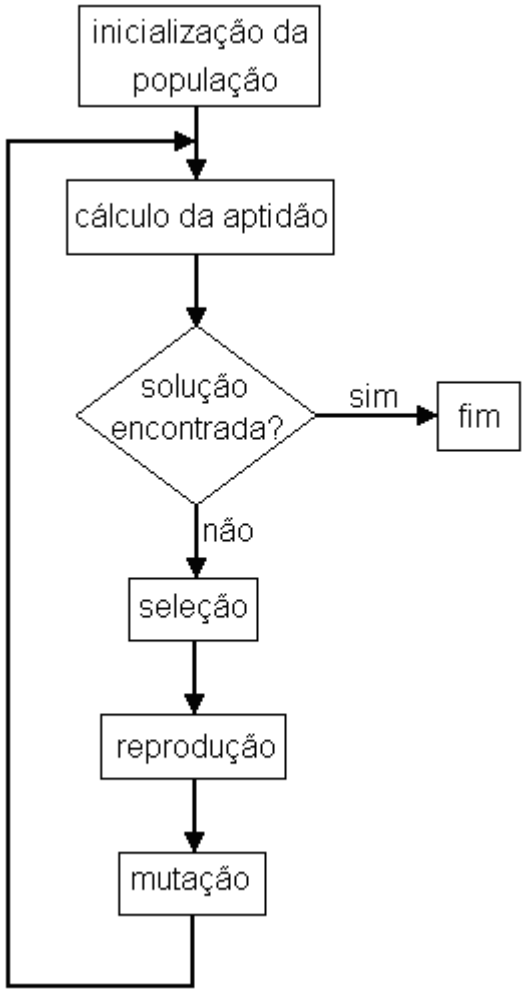


Figura 2.7 - Diagrama de um algoritmo evolucionário

Dentre os algoritmos evolucionários, o que mais possui destaque são os Algoritmos Genéticos, desenvolvidos pelo professor John Holland e sua equipe (Holland, 1975), e com certeza o que possui maior uso em MD, onde os candidatos à solução geralmente são os valores das variáveis. Outro tipo de AEs que merece destaque é a Programação Genética (PG), cujos candidatos à solução são tanto os valores das variáveis quanto as funções.

A maior diferença entre AG e PG é a forma como a solução é representada. (Freitas, 2002). Em AG, cada indivíduo representa uma solução para uma instância em particular do problema a ser solucionado. Já a PG, um candidato a solução representa uma solução genérica (um programa ou um algoritmo) do problema a ser solucionado.

A Figura 2.8 mostra o pseudo-código para AG e PG

```
Geração da população inicial de indivíduos (soluções candidatas)
Cálculo da aptidão de cada indivíduo
REPITA
        Seleção dos indivíduos baseado na aptidão
        Aplicação dos operadores genéticos aos indivíduos
          selecionados criando novos indivíduos
        Cálculo da aptidão de cada um dos novos indivíduos
        Atualiza a população (novos indivíduos substituem antigos
          indivíduos)
ATÉ (critério de parada)
```

Figura 2.8 - Pseudo-código para AG e PG

2.3.1 Algoritmos Genéticos

Devido a sua forma de trabalhar com uma gama de soluções a cada geração, os AGs são capazes de encontrar várias soluções não dominadas ao longo do processo de otimização. Essa propriedade aliada a sua adaptabilidade a diferentes tipos de problemas, tornam os AGs importantes ferramentas de otimização multiobjetivo.

Nos AGs, um cromossomo é um conjunto de dados que é uma possível solução do problema. Esses cromossomos então passam por evoluções que inclui a avaliação, seleção, e os operadores genéticos de cruzamento e mutação que podem ser vistos na Figura 2.9. Os indivíduos mais aptos serão obtidos após alguns ciclos de evolução.

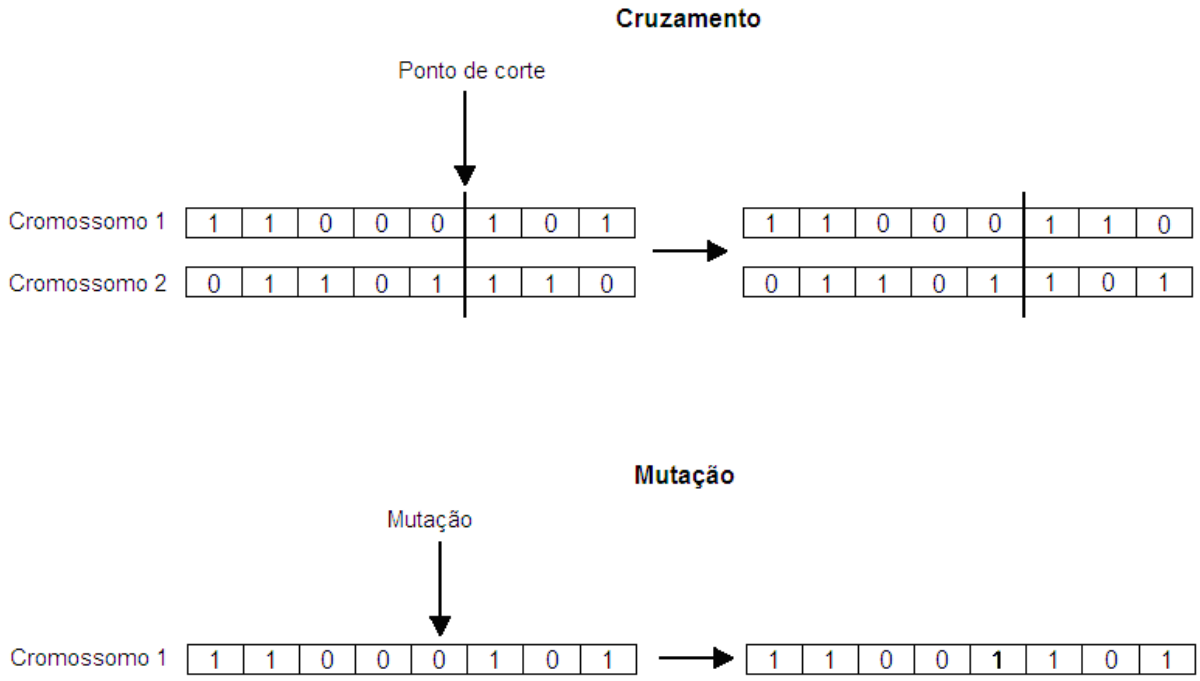


Figura 2.9 - Operadores genéticos em AG

Normalmente os AGs são vistos como otimizadores de funções, embora a quantidade de problemas para o qual os AGs se aplicam seja bastante abrangente.

2.3.1 AE para a Descoberta de Regras de Classificação

Os AEs podem ser usados com o objetivo de descobrir regras do tipo SE-ENTÃO, em que os antecedentes são compostos por uma ou mais condições da forma atributo-operador-valor.

Segundo (Freitas, 2002), podem ser distinguidos dois tipos de abordagens no processo de descoberta de regras utilizando AE, ambas baseadas no *Learning Classifier System* (LCS) desenvolvido pelo professor John Holland na década de 70:

- a primeira em que cada indivíduo é representado por um conjunto de regras (abordagem *Pittsburgh*), podendo o número de regras que compões cada indivíduo ser variável, mudando automaticamente durante as evoluções, ou fixo;
- ou a segunda em que cada indivíduo é representado apenas por uma regra (abordagem *Michingan-style*).

A abordagem *Pittsburgh* possui a vantagem de um indivíduo representar uma solução completa ao problema, visto que mapeia todo um conjunto de regras. Porém gera indivíduos longos, aumentando o custo computacional durante as operações de mutação e cruzamento, e aptidão.

Já a abordagem *Michingan*, os indivíduos são curtos, e reduzem o custo computacional durante as operações, porém, é difícil se avaliar a qualidade da solução de um conjunto de regras e suas interações.

Exemplos de uso de AEs para descoberta de regras pode ser visto em (De Jong, 1993), (Janikow, 1993) e (Pei, 1997), utilizando a abordagem *Pittsburgh* e em (Greene, 1993) e (Giordana, 1995) utilizando a abordagem *Michingan*.

2.3.2 AE para Agrupamentos

Apesar do grande número de aplicações de AG em diferentes tipos de problemas de otimização, existe muito pouca pesquisa sobre a utilização deste tipo de abordagem para o problema de agrupamentos. De fato, e tendo em conta a qualidade das soluções que esta tecnologia tem mostrado em diferentes tipos de campos e problemas, faz todo o sentido tentar usá-lo também em problemas de agrupamentos.

A flexibilidade associada aos AG é um aspecto importante a ter em mente. Com a mesma representação do genoma e apenas mudando a uma função de aptidão pode ter um algoritmo diferente. No caso de análise espacial, isto é particularmente importante, pois um pode tentar funções de aptidão diferentes em uma fase exploratória.

O AG minimiza o erro quadrado da dispersão do cluster:

$$E = \sum_{k=1}^{K} \sum_{x \in Ck} \left\| x - mk \right\|^2 \qquad (2.1)$$

Onde: *K* é o número de clusters, *mk* o centro do cluster Ck.

No genoma, cada gene representa um ponto de dados e define membros do cluster. Todos os operadores necessários à evolução podem ser implementados com este regime. Como apontado por (Demiriz,1999), o principal problema associado a este sistema de representação é que ele não é escalável, por outro lado, parece ser computacionalmente eficiente quando o número de dados não é muito grande.

2.4 Sistemas Imunológicos Artificiais

O sistema imunológico natural dos seres vivos desenvolvidos, principalmente o do ser humano, por sua capacidade de reconhecimento de indivíduos invasores, memória e aprendizado, e eliminação de agentes patogênicos, tem se tornado fonte de inspiração para cientistas e engenheiros desenvolverem ferramentas e soluções de problemas clássicos, como de otimização e de reconhecimento de padrões, o que culminou no desenvolvimento do novo conceito conhecido SIA. Nessa área, alguns trabalhos interessantes podem ser encontrados em (Ishida, 1996), (Hunt, 1996), (Dasgupta, 1999) e (de Castro, 2001).

O SIA usa as idéias colhidas da imunologia a fim de desenvolver sistemas capazes de executar uma grande quantidade de tarefas em várias áreas de pesquisa (de Castro, 2001).

Em (de Castro, 2001) é mostrado como a teoria da seleção clonal, aliada ao processo de maturação por afinidade e outros aspectos biológicos podem ser

usados no desenvolvimento de poderosas ferramentas capazes de resolver problemas complexos de engenharia.

Os SIAs são incluídos no conceito de algoritmos evolucionários, pois se baseia na evolução das populações, utilizando a diversidade, variação genética e seleção natural.

Dentre as propriedades dos sistemas imunológicos que tem despertado o interesse dos engenheiros e cientistas da computação, as que mais se destacam são de acordo com (De Castro, 2001):

- ***Unicidade -*** cada ser possui seu sistema imunológico próprio e único;
- ***Reconhecimento de padrões -*** agentes estranhos ao organismo são reconhecidos e eliminados pelas células do sistema imunológico;
- ***Detecção a anomalias -*** agentes causadores de anomalia ao organismo que nunca haviam entrado em contato com o sistema imunológico são detectados;
- ***Detecção imperfeita -*** não é necessário o reconhecimento completo do agente patogênico para que o sistema imunológico possa reagir a ele;
- ***Diversidade -*** as células e moléculas presentes no sistema imunológico são capazes de identificar praticamente uma infinidade de agentes patogênicos, sejam naturais ou artificiais;
- ***Aprendizagem -*** o sistema imunológico se aprimora e aumenta a eficiência no reconhecimento a um agente patogênico toda vez que este entra em contado;

- ***Memória -*** os componentes do sistema imunológico bem sucedidos no reconhecimento e combate às patologias são armazenados para uma resposta futura mais intensa e efetiva.

A Figura 2.10 ilustra o processo do principio clonal de um SIA.

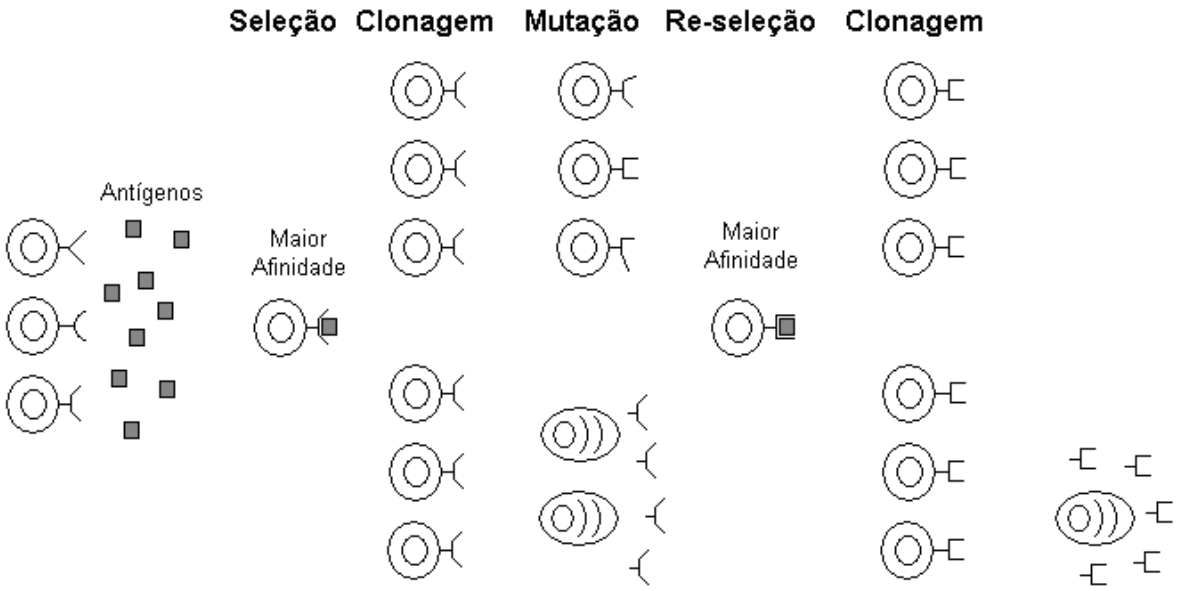


Figura 2.10 - O princípio clonal

Em (de Castro, 2001), é proposto um algoritmo denominado CLONALG que implementa o principio de seleção clonal de um SIA. No CLONALG é incluído o processo de maturação de afinidade da resposta imunológica. Ele é especificamente utilizado para resolver problemas de aprendizado de máquina, de otimização multimodal e de reconhecimento de padrões.

O processo do algoritmo CLONALG é mostrado na Figura 2.11 e cada uma das etapas do diagrama é descrita a seguir.

(1) É gerada uma população inicial de anticorpos, composta de anticorpos de memória mais o restante.

(2) Cada um dos indivíduos é avaliado utilizando uma função de avaliação que mede a afinidade de cada um deles em relação à função objetivo.

(3) Desses anticorpos, são selecionados os *n* melhor avaliados, que passarão por processo de clonagem e mutação.

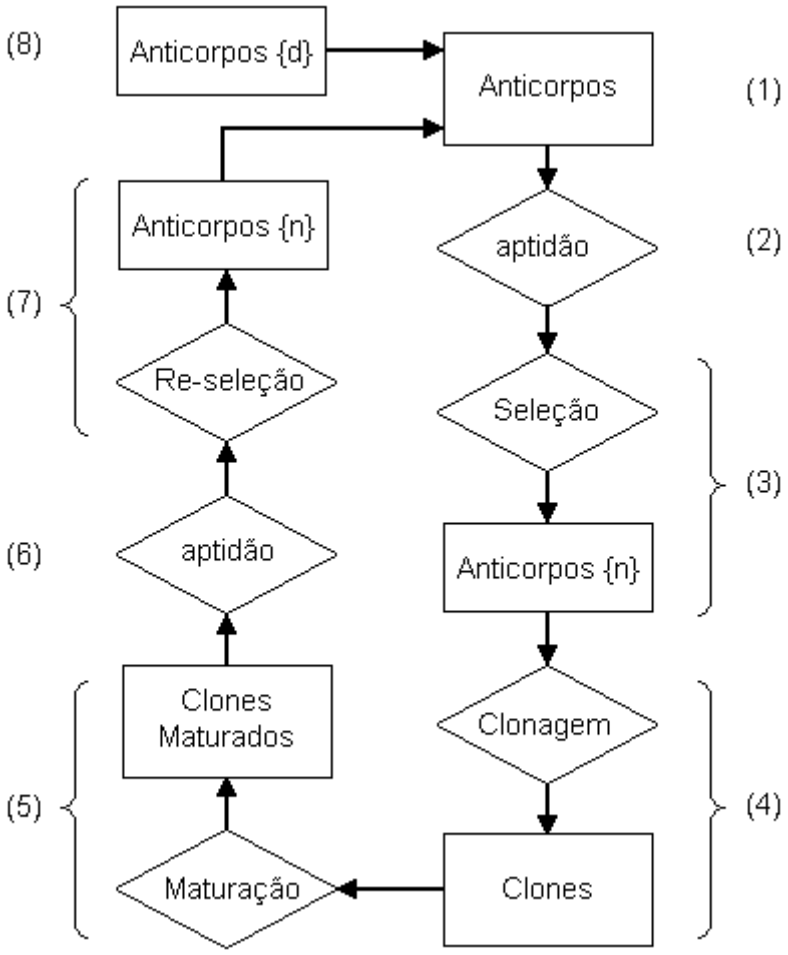


Figura 2.11 - Processo do algoritmo de seleção clonal (de Castro,2001)

(4) Cada anticorpo sofre processo de clonagem, onde cada indivíduo dá origem a outros denominados clones. A quantidade de replicação é diretamente proporcional à afinidade do anticorpo medido pela função de avaliação.

(5) Esses clones sofrem então mutação, cuja taxa é inversamente proporcional à afinidade.

(6) Esses clones maturados são também avaliados utilizando a função de avaliação

(7) Novamente são selecionados os melhores clones de cada anticorpo de memória. Se o melhor clone de cada anticorpo possui maior afinidade que o próprio anticorpo, esse clone substitui o anticorpo na população de memória.

(8) Uma nova população aleatória é gerada e adicionada a população de anticorpos num processo que simula a edição de receptor.

Em (Honório, 2007) uma variante do algoritmo CLONALG é implementado utilizando um gradiente para diminuição do espaço de busca usando o conceito de vetor tangente para extração de regras *fuzzy*. Esse algoritmo é denominado GbCLONALG e é utilizado nesse trabalho para seleção das melhores regras e da otimização das funções de pertinência.

Geração da população inicial de *m* anticorpos
Cálculo da aptidão de cada indivíduo
Seleção dos *n* melhores anticorpos baseado na aptidão (formação da população memória)
REPITA
    PARA CADA Anticorpo Ab_i na população dos *n* melhores anticorpos
        Clonagem dos anticorpos (quantidade diretamente proporcional à aptidão)
        Aplicação da mutação aos clones (fator de mutação inversamente proporcional à aptidão)
        Cálculo da aptidão de cada um dos clones mutados
        Seleção do melhor Clone mutado C_i^*
        SE aptidão do Clone C_i^* MAIOR aptidão do Anticorpo Ab_i
            Anticorpo Ab_i é substituído pelo Clone C_i^* na população
    FIM PARA CADA
    Geração de população de *m*-*n* anticorpos.
    Substituição da nova população pelos anticorpos fora da população memória
ATÉ (critério de Parada)

Figura 2.12 - Pseudocódigo do SIA

CAPÍTULO 3

ANÁLISE DO ALGORITMO CAISFLO E DA APLICAÇÃO DESENVOLVIDA

Neste capítulo será apresentada uma descrição do algoritmo CAISFLO que foi modelado na ferramenta desenvolvida durante este trabalho e que é baseada nos estudos apresentados em (Vermaas, 2009). O sistema desenvolvido possui a capacidade de gerar um sistema *fuzzy*, realizando a extração das regras *fuzzy* a partir de dados numéricos de treinamento, a escolha de uma ótima combinação dessas regras e posteriormente a otimização das funções de pertinência usando a técnica SIA.

O processo empregado no CAISFLO é apresentado a seguir em três etapas, que são:

1ª etapa: Aprendizado das regras usando os dados numéricos;

2ª etapa: Escolha do melhor repertório de regras utilizando SIA;

3ª etapa: Otimização dos pontos das funções de pertinência utilizando SIA.

A primeira etapa de aprendizado das regras é realizada utilizando um procedimento similar ao apresentado em (Wang, 1992). As etapas de escolha das regras e otimização das funções de pertinência utilizam o algoritmo GbCLONALG apresentado em (Honório, 2007).

O processo de SIA evoluirá os anticorpos de forma similar à abordagem *Pittsburgh*, sendo que cada anticorpo representará um conjunto de regras, candidato a ser a base de regras do sistema *fuzzy* desenvolvido.

O sistema desenvolvido é capaz de fazer previsão em sistemas de utilizando tarefas tanto de classificação quanto de regressão.

3.1 Gerando um Sistema Fuzzy a Partir de Dados Numéricos

A partir de um conjunto numérico de dados históricos obtido do ambiente a ser mapeado, o algoritmo estudado nesse trabalho é capaz de extrair o conhecimento ali disponível e representá-lo na forma de regras *fuzzy* do tipo 'SE-ENTÃO'. Essas regras são responsáveis por relacionar dados de entrada com dados de saída através de variáveis lingüísticas.

O algoritmo, portanto, gera ao final do seu processo um sistema *fuzzy* formado pela base de regras e as funções de pertinência.

No CAISFLO, os processos de extração das regras e de otimização das funções de pertinência são realizados de forma conjunta, pois eles são dependentes um do outro. Nesse processo de co-evolução, existem duas populações distintas, em que uma é utilizada pela etapa de seleção das melhores regras e outra utilizada pela etapa de otimização das funções.

Inicialmente, o intervalo de domínio das variáveis (atributos), tanto de entrada como de saída dos dados utilizados no processo de aprendizado é dividido em N

regiões *fuzzy*, modeladas por funções de pertinência que podem ser de quantidade, tipo e com posições diferentes para cada variável.

Como foi visto no capítulo anterior, há diversas técnicas capazes de gerar um sistema *fuzzy*, extraindo as regras e as funções de pertinência a partir de dados numéricos, utilizando-se diversas abordagens. Uma das mais conhecidas é a proposta em (Wang, 1992) que foi descrita no capítulo anterior, e que, como já mencionado, é utilizada na aplicação desenvolvida nesse trabalho. Porém aqui, as regras *fuzzy* são geradas utilizando-se uma variação dessa abordagem. A diferença para o algoritmo aqui estudado é que uma tupla de dados numéricos pode gerar mais de uma regra, pois não serão considerados apenas os conjuntos *fuzzy* que apresentarem o maior grau de pertinência entre aqueles ativados, mas sim todos os conjuntos *fuzzy* ativados.

A partir daí, utiliza-se do conceito de SIA para encontrar o melhor repertório de regras e otimizar as funções de pertinência. Para isso, foi utilizado o algoritmo GbCLONALG.

Resumindo todo o processo, inicialmente o algoritmo CAISFLO gera uma população de funções de pertinência fpPop = $\{fAb_1,...,fAb_n\}$, e a partir dela, e utilizando os dados numéricos, é gerada uma população de regras $rPop=\{rAb_1,...,rAb_n\}$. Logo após, a população rPop é evoluída, fornecendo uma nova população de regras, de onde são selecionas as melhores utilizando-se o *aptidão,* definido pela função de avaliação. Então a população fpPop é também evoluída, encontrando-se melhores coordenadas dos vértices das funções de pertinência, também utilizando-se o valor do aptidão. Esse processo se repete por um número fixo de vezes, que serão as gerações pré-definidas.

A Figura 3.1 ilustra o diagrama do processo descrito.

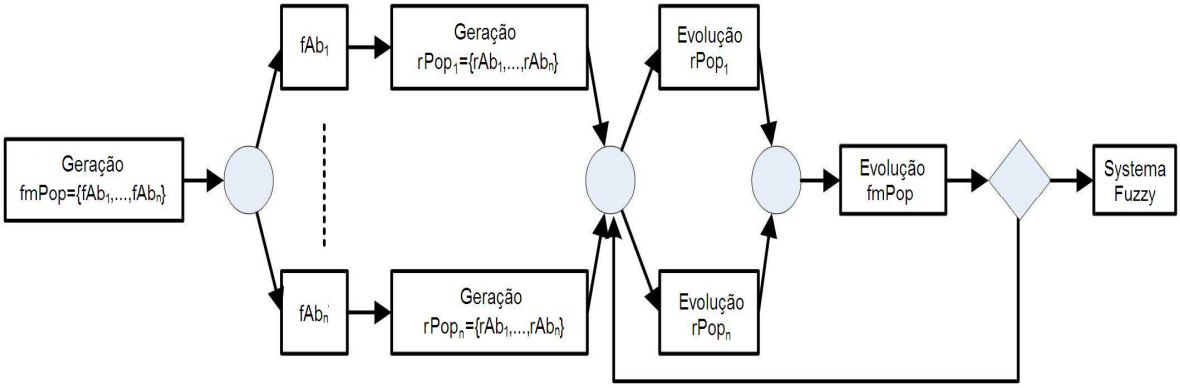


Figura 3.1 - Diagrama do processo evolutivo do sistema

3.2 Divisão das Variáveis em Conjuntos Fuzzy

Essa é a parte preliminar do algoritmo CAISFLO. Nessa etapa são gerados os conjuntos *fuzzy* e a também criadas as funções de pertinência iniciais.

Para ilustrar esta etapa, seja uma variável X na qual se deseja dividir em conjuntos *fuzzy*, cujo domínio é representado pelo intervalo $[x_{min},x_{max}]$. Essa variável é dividida em N conjuntos. Cada um recebe uma denominação lingüística, podendo ser alto ou baixo, pequeno ou grande, etc.

Na aplicação desenvolvida durante esse trabalho, a quantidade de conjuntos foi definida como sendo N=3 nomeados por P (pequeno), M (médio) e G (grande), e modelados pelas funções de pertinência do tipo trapezoidal. Essas funções de pertinência, como já mencionado anteriormente, são responsáveis por associar a cada valor da variável X um grau de pertinência $\mu(X)$.

Para a determinação das coordenadas dos vértices de cada conjunto, o intervalo de domínio da variável é dividido por N+1, encontrando-se com isso um valor incremental conforme a Equação 3.1.

$$X_{inc} = (x_{max}-x_{min})/(N+1) \tag{3.1}$$

Assim, uma variável que for dividida em N=3 conjuntos será representada conforme a Figura 3.2.

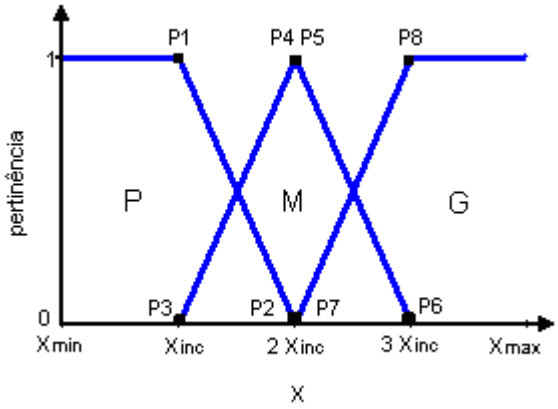


Figura 3.2 – Conjuntos Fuzzy e funções de pertinência

Tabela 3.1 - Coordenadas dos pontos das funções de pertinência

Pontos	Coordenada x	Coordenada y
P1	x_{inc}	1
P2	$2x_{inc}$	0
P3	x_{inc}	0
P4	$2x_{inc}$	1
P5	$2x_{inc}$	1
P6	$3x_{inc}$	0
P7	$2x_{inc}$	0
P8	$3x_{inc}$	1

Ressalta-se aqui que a região M tem função de pertinência do tipo trapezoidal, portanto possui 4 vértices (P3, P4, P5 e P6), sendo que P4 e P5 são coincidentes e com coordenadas ($2x_{inc}$,1). Na fase de otimização esses pontos serão ajustados e poderão deixar de ser coincidentes.

3.3 Geração das Regras

Como visto no Capítulo 2, a geração das regras *fuzzy* utilizando a abordagem proposta por (Wang,1992) gera regras a partir de dados numéricos utilizando o critério de maior grau de pertinência, descartando-se as regiões ativados com graus de pertinência baixo.

Porém, como se vê em (Zadeh, 1965), o processo de determinação do valor de saída a partir de um sistema *fuzzy* (defuzzificação), depende do grau de pertinência calculado a partir das funções dos conjuntos que foram ativados, mesmo aqueles que possuem baixos graus de pertinência. A geração de regras *fuzzy* apenas utilizando conjuntos que fornecem graus de pertinência altos pode-se ocasionar perda de precisão. Por este motivo, no algoritmo estudado, todos os conjuntos ativados pelos dados numéricos, sejam de graus de pertinência altos e baixos, serão combinados para formar as regras. Assim, diferente da abordagem proposta em (Wang,1992), cada tupla pode gerar mais de uma regra.

Neste processo, podem-se gerar então regras conflitantes, ou seja, regras que possuem mesmo antecedentes com conseqüentes diferentes. Como será visto

mais adiante, durante o processo evolutivo mostrado na Figura 3.1 este problema será eliminado.

Para a melhor compreensão dessa etapa, suponha o par de tuplas formados pelos elementos numéricos apresentados no Exemplo 2.1.

Tabela 3.2 - Graus de Pertinência para a tupla 1

	PP	P	M	G	GG
$X_1^{(1)}$	0	0	0,4	0,6	0
$X_2^{(1)}$	0	0,7	0,3	0	0
$Y^{(1)}$	0	0,2	0,8	0	0

Tabela 3.3 - Graus de Pertinência para a tupla 2

	PP	P	M	G	GG
$X_1^{(2)}$	0	0	0	0,7	0,3
$X_2^{(2)}$	0	0	0	1	0
$Y^{(2)}$	0	0	0	0	1

De acordo com a Tabela 3.2 e a Tabela 3.3, os dados geram as seguintes regras *fuzzy*:

Tupla 1:

R_1: SE x_1 = M E x_2 = P ENTÃO Y = P
R_2: SE x_1 = M E x_2 = P ENTÃO Y = M
R_3: SE x_1 = M E x_2 = M ENTÃO Y = P
R_4: SE x_1 = M E x_2 = M ENTÃO Y = M
R_5: SE x_1 = G E x_2 = P ENTÃO Y = P
R_6: SE x_1 = G E x_2 = P ENTÃO Y = M
R_7: SE x_1 = G E x_2 = M ENTÃO Y = P

R_8: SE x_1 = G E x_2 = M ENTÃO Y = M

Tupla 2:

R_9 : SE x_1 = G E x_2 = G ENTÃO Y = GG

R_{10}: SE x_1 = GG E x_2 = G ENTÃO Y = GG

Tabela 3.4 - População de regras geradas a partir do exemplo dado

	PP	P	M	G	GG
SE x_1 = M E x_2 = P	0	1	1	0	0
SE x_1 = M E x_2 = M	0	1	1	0	0
SE x_1 = G E x_2 = P	0	1	1	0	0
SE x_1 = G E x_2 = M	0	1	1	0	0
SE x_1 = G E x_2 = G	0	0	0	0	1
SE x_1 = GG E x_2 = G	0	0	0	0	1

3.4 O Uso do SIA no Algoritmo CAISFLO

Conforme foi visto no Capitulo 2, o SIA têm grande utilidade em problemas de otimização e aprendizagem, pois essa abordagem objetiva a realização da busca da solução utilizando-se heurísticas tanto em escala local, utilizando-se das mutações, quanto em escala global utilizando a edição de receptor.

A Figura 3.3 ilustra o diagrama do SIA que é utilizado no algoritmo aqui proposto.

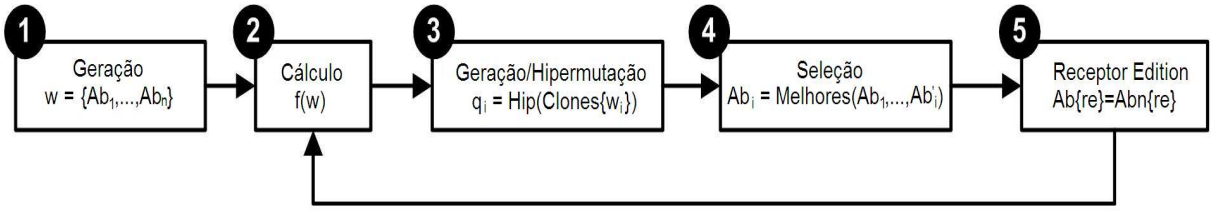


Figura 3.3 - Esquema do SIA

Como já comentado, o algoritmo CLONALG, utiliza-se da seleção clonal e o processo de maturação de afinidade imunológica para o processo de otimização, aprendizado e reconhecimento de padrões. O algoritmo GbCLONALG proposto em (Honório, 2007) é uma variante do CLONALG, que utiliza a técnica do vetor tangente (VT) que usa informações coletadas durante o processo de mutação, objetivando a redução do esforço computacional e conseqüentemente menor tempo de processamento, pois promove uma redução no número de clones. Esse vetor mede o grau de sensibilidade de uma função frente a pequenos estímulos em suas variáveis. Funciona como uma heurística que fornece um direcionamento para as mutações, afim de que essas não sejam aleatórias.

A expressão que representa o VT do algoritmo GbCLONALG é apresentado a seguir pela equação 3.2:

$$VT(f(x_1,...,x_n)) = \begin{bmatrix} \frac{f(x_1+\Delta x_1,...,x_n)-f(x_1,...,x_n)}{|\Delta x_1|} \\ \vdots \\ \frac{f(x_1,...,x_n+\Delta x_n)-f(x_1,...,x_n)}{|\Delta x_n|} \end{bmatrix} \quad (3.2)$$

onde:

n	é o número de variáveis de controle, ou variáveis de entrada;
f()	é a função objetivo a ser otimizada;
$x_1, ..., x_n$	são as variáveis de controle, ou variáveis de entrada;
$\Delta x_1, ..., \Delta x_n$	é um incremento de valor aleatório aplicado a cada variável x_k.

3.4.1 SIA para a Escolha do Melhor Repertório de Regras

Com a etapa de geração de regras a partir dos dados numéricos conforme mostrado anteriormente, um grande universo populacional composto por todas as regras é criado e o objetivo a partir daí é escolher o melhor subconjunto de regras que fornecerá os melhore resultados.

No início, é gerada *n* subconjuntos de combinações aleatórias de regras em quantidade também aleatória tomadas do universo populacional, formando-se a primeira população de *n* Anticorpos. É nessa etapa que o conflito de regras como foi visto anteriormente é eliminado, pois durante a combinação, não é permitido que regras de mesmo antecedente e de diferente conseqüente componham um mesmo Anticorpo. Para cada um desses subconjuntos (Anticorpos rAb) é calculada a aptidão.

Para dados que representam processos de regressão, a aptidão é calculada utilizando a Equação 3.3.

$$\text{aptidão}(Ab) = [Ab(inD) - outD]^2 \qquad (3.3)$$

onde:

Ab	é o anticorpo do qual está se calculando a aptidão
inD	é uma tupla formada por elementos de entrada;
outD	é o valor de saída para a tupla inD;
Ab(InD)	é a saída fornecida pelo sistema devido ao dado de entrada inD.

Para dados que representam processos de classificação, a aptidão é calculada utilizando-se a Equação 3.4 definida em (Lopes, 1997).

$$\text{aptidão(r)} = \frac{VP}{VP+FN} \times \frac{VN}{VN+FP} \tag{3.4}$$

onde:

VP é número de verdadeiros positivos, ou seja, número de exemplos que satisfazem as cláusulas da regra e possuem a mesma classe fornecida pela regra;

FN é número de falsos negativos, ou seja, número de exemplos que não satisfazem as cláusulas da regra, mas possuem a mesma classe fornecida pela regra;

VN é número de verdadeiros negativos, ou seja, número de exemplos que não satisfazem as cláusulas da regra e não possuem a mesma classe fornecida pela regra;

FP é número de falsos positivos, ou seja, número de exemplos que satisfazem as cláusulas da regra e não possuem a mesma classe fornecida pela regra.

A Equação 3.4 fornece a aptidão para cada regra. Como aqui cada anticorpo é um conjunto de regras, é utilizada então a média das aptidões de cada regra.

São selecionados então os *m* melhores anticorpos avaliados, e a partir deles são gerados clones que posteriormente sofrerão a evolução. A quantidade de clones gerados por cada anticorpo depende da avaliação feita pela função aptidão(r). Quanto maior a avaliação, menor o numero de clones gerados.

A Figura 3.4 ilustra o diagrama do processo descrito.

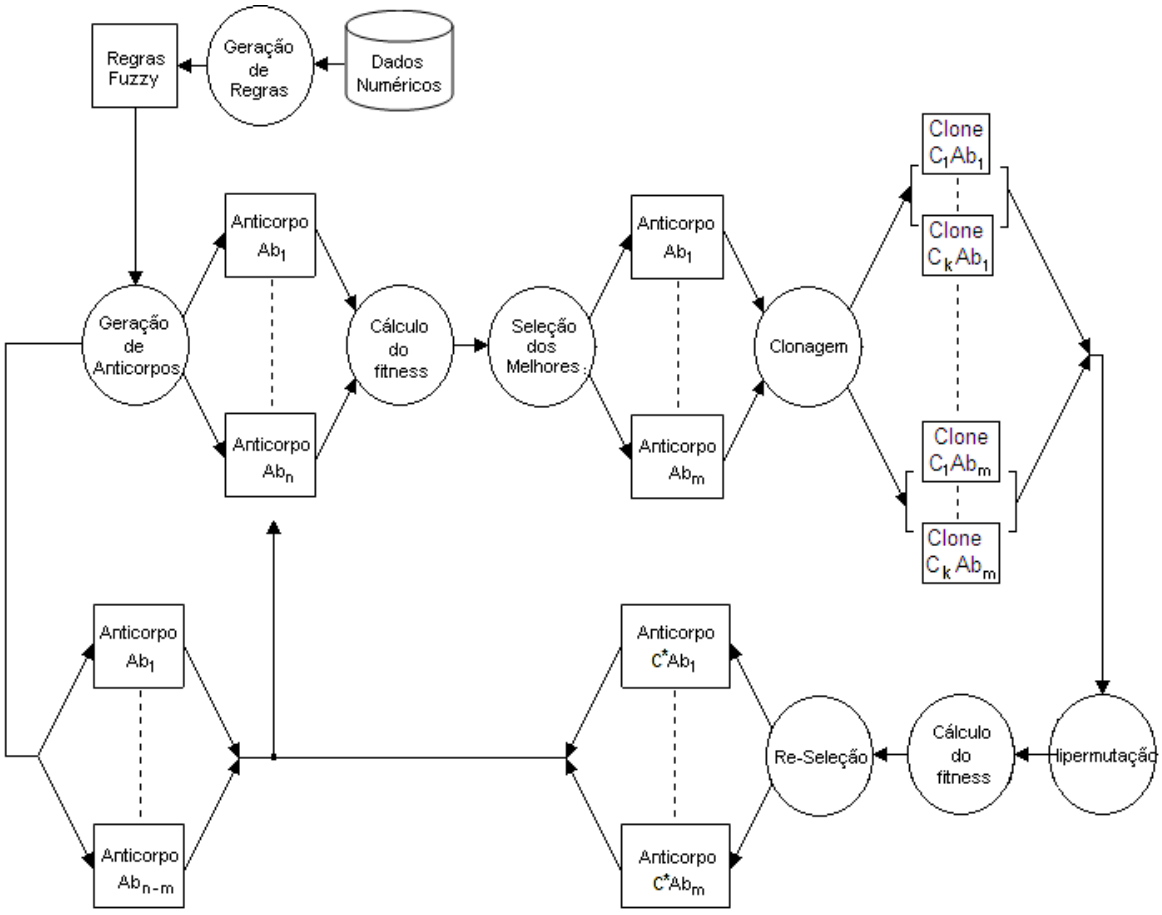


Figura 3.4 - Diagrama do processo de extração de regras do Algoritmo CAISFLO

Essa evolução pode ser feita de três formas diferentes para anticorpos com quantidade *n* de regras:

- ***Adição de regras*** **-** são adicionadas ao anticorpo regras não conflitantes e não repetidas tomadas da população de regras;
- ***Remoção de regras*** **-** são removidas do anticorpo regras nele existentes. A quantidade de remoção é no mínimo 1 e no máximo *n-1* regras;

- ***Substituição de regras -*** são substituídas regras do anticorpo em quantidade mínima de 1 e máxima de *n-1*, tomadas da população de regras.

Para cada clone é calculada a aptidão, utilizando-se também a Equação 3.3 e a Equação 3.4, e então escolhidos os melhores que irão compor a nova geração de anticorpos. A quantidade de anticorpos selecionados será *n/2*.

A esses melhores anticorpos são adicionados outros gerados do universo populacional inicial, numa etapa conhecida como editor de recepção.

Esse processo se repete por *m* gerações, sendo o valor de *m* escolhido previamente.

3.4.2 SIA para Otimização das Funções de Pertinência

Nesse processo, cada anticorpo *fpAb* irá representar uma variável contínua com seu conjunto de funções de pertinência.. Assim, para um sistema que possuir 5 variáveis contínuas, o sistema gerará 5 anticorpos, que serão compostos pelos valores das abscissas das coordenadas dos vértices de cada conjunto *fuzzy*. A Figura 3.1 ilustra uma variável com seus respectivos conjuntos *fuzzy* e a Tabela 3.1 as coordenadas dos pontos. A partir desse exemplo, o anticorpo será:

$$fpAb = \{x_{inc}, 2x_{inc}, 3x_{inc}\} \quad (3.5)$$

A partir daí, conforme mostrado na Figura 3.2, é calculada a aptidão para cada anticorpo, utilizando-se também a Equação 3.3 ou a Equação 3.4, que depois é

clonado numa quantidade pré-determinada e diretamente proporcional ao valor da aptidão. Cada clone então sofre uma mutação, que consiste em realizar uma pequena variação dos valores dos pontos dos vértices das funções de pertinência. Essa variação Δx possui um valor aleatório e inversamente proporcional à aptidão do clone.

A Figura 3.5 ilustra um exemplo de função de pertinência de uma variável otimizada pelo algoritmo CAISFLO.

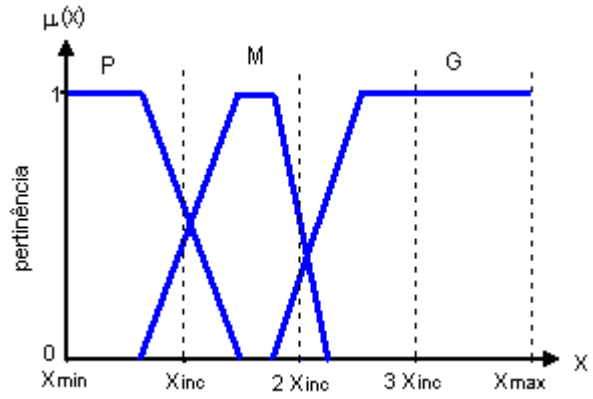


Figura 3.5 - Função de pertinência otimizada pelo CAISFLO

3.4.3 Obtenção dos Resultados

3.4.3.1 Teste em Tarefas de Regressão

Para dados que representam processos de regressão, os valores de saída fornecidos pelo sistema devem ser numéricos. Porém, como os dados de entrada passam por processo de fuzzificação, os dados de saída são também no formato *fuzzy*, ou seja, representados por conjuntos *fuzzy*. Por esse motivo é necessária, antes de fornecer o resultado, a realização de um processo inverso à fuzzificação, a defuzificação. Na literatura existem diversas formas de defuzzificação, sendo a mais

utilizada a que utiliza o centróide, que é também o formato usado no sistema desenvolvido.

Para calcular a acurácia dos resultados obtidos, é calculada a raiz do erro quadrático médio (RMS) de acordo com a Equação 3.6 e a Equação 3.7.

$$\text{Acurácia}_{\%} = (1\text{-RMS}) \times 100 \tag{3.6}$$

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{1}^{N}(y_{real} - y_{sistema})^2} \tag{3.7}$$

onde:

N número de tuplas nos dados de teste;

y_{real} valor de saída na tupla do dado de teste;

$y_{sistema}$ valor de saída fornecido pelo sistema.

3.4.3.2 Teste em Tarefas de Classificação

Para dados que representam processos de classificação, o valor da acurácia é obtido calculando-se a razão entre os exemplos classificados corretamente e o total de dados de teste. De acordo com a Equação 3.8.

$$\text{Acurácia}_{\%} = \frac{C}{N} \text{x}100 \tag{3.8}$$

onde:

C numero de tuplas nos dados de teste que tem o valor de saída corretamente classificada pela base de regras;

N número total de tuplas no conjunto de dados de teste.

Durante o processo de classificação de certa tupla do conjunto de dados de teste, uma ou mais regras podem ser ativadas, ou seja, a parte antecedente da regra coincidir com a parte antecedente da tupla. Nesse processo podem ocorrer três possibilidades:

- Todas as regras ativadas pela tupla possuem o mesmo conseqüente. Assim a saída predita será este conseqüente em comum a todas as regras.
- As regras ativadas possuem diferentes conseqüentes. Assim a saída predita será o conseqüente daquela regra que possuir maior afinidade com relação à tupla. A afinidade é calculada pela Equação 3.9.

$$Afin = \min[\mu(x_1), \mu(x_2),..., \mu(x_n)] \quad (3.9)$$

Onde $\mu(x_1)$, $\mu(x_2)$,... $\mu(x_n)$ são os graus de pertinência de um valor com relação as variável calculados pelas funções de pertinência.

- Nenhuma regra é ativada pela tupla. Assim a saída predita será assinalada pela classe padrão, que é aquela que for mais freqüente no conjunto de dados de treinamento.

CAPÍTULO 4

APLICAÇÃO DO ALGORITMO CAISFLO

Nesse capítulo é descrita a aplicação que foi desenvolvida para a realização do teste do algoritmo CAISFLO proposto nessa dissertação. Para isso serão apresentadas imagens das telas geradas durante a execução do programa além do diagrama de classes do sistema.

A aplicação foi desenvolvida utilizando a linguagem C# através do ambiente de desenvolvimento Visual Studio da Microsoft e a abordagem orientada a objetos.

Os dados utilizados para treinamento e teste do sistema fuzzy é armazenado em um arquivo texto já devidamente pré-processado.

4.1 Telas Geradas pelo Sistema

A Figura 4.1 mostra a tela inicial da aplicação, de onde é gerado o sistema *fuzzy* inicial, antes de passar pelo processo de otimização. Nessa tela são também geradas regras pelos métodos C4.5 e Wang e Mendel. As regras geradas são mostradas para o operador da aplicação.

A Figura 4.2 ilustra a tela da aplicação que é a responsável pelo processo de otimização de todo o sistema *fuzzy* utilizando-se o SIA. Como se pode ver pela figura, as duas opções, sendo escolha das melhores regras e otimização das funções de pertinência, são disponíveis para o operador da aplicação, através dos

botões. Aqui também se encontra a possibilidade de modificar a quantidade gerações, anticorpos e clones. Quando os dados representam tarefas de regressão, um gráfico com os pontos de saída é plotado para mostrar ao operador da aplicação a comparação entre as diversas abordagens implementadas. Um outro gráfico com é apresentado com as funções de pertinência otimizadas.

A Figura 4.3 mostra a tela onde é apresentado ao operador as regras que passaram pelo processo de otimização utilizando SIA e também faz-se o teste final para a validação do sistema *fuzzy* gerado.

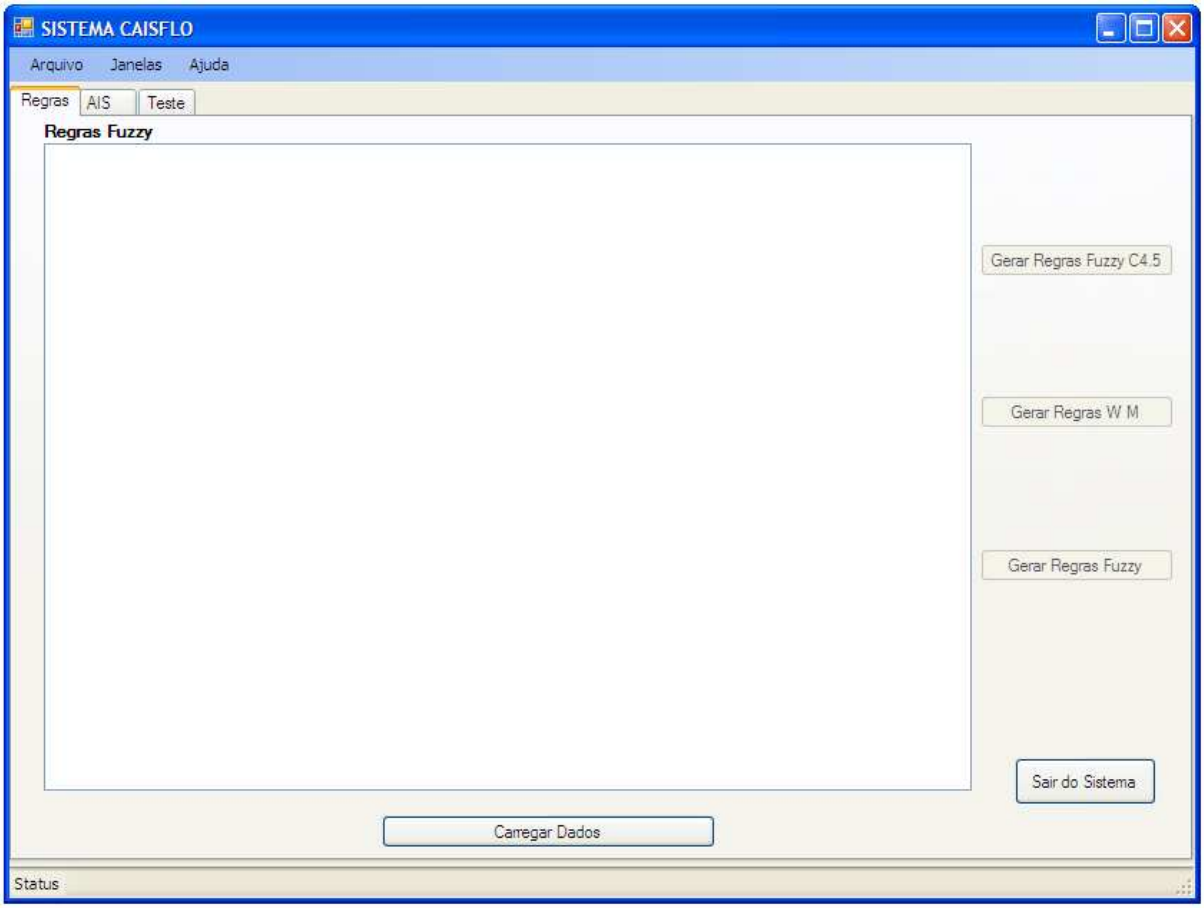
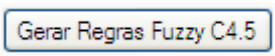
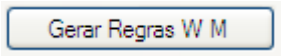


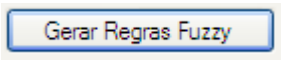
Figura 4.1 - Tela gerada pela aplicação – Aba Regras



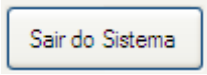
Gera regras baseado em árvores de decisão criadas pelo algoritmo C4.5.



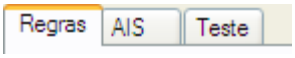
Gera regras baseado em no algoritmo proposto por Wang e Mendel.



Gera todas as combinações de regras fuzzy que serão utilizados no processo SIA.



Botão que encerra a aplicação.



Botões de abas.

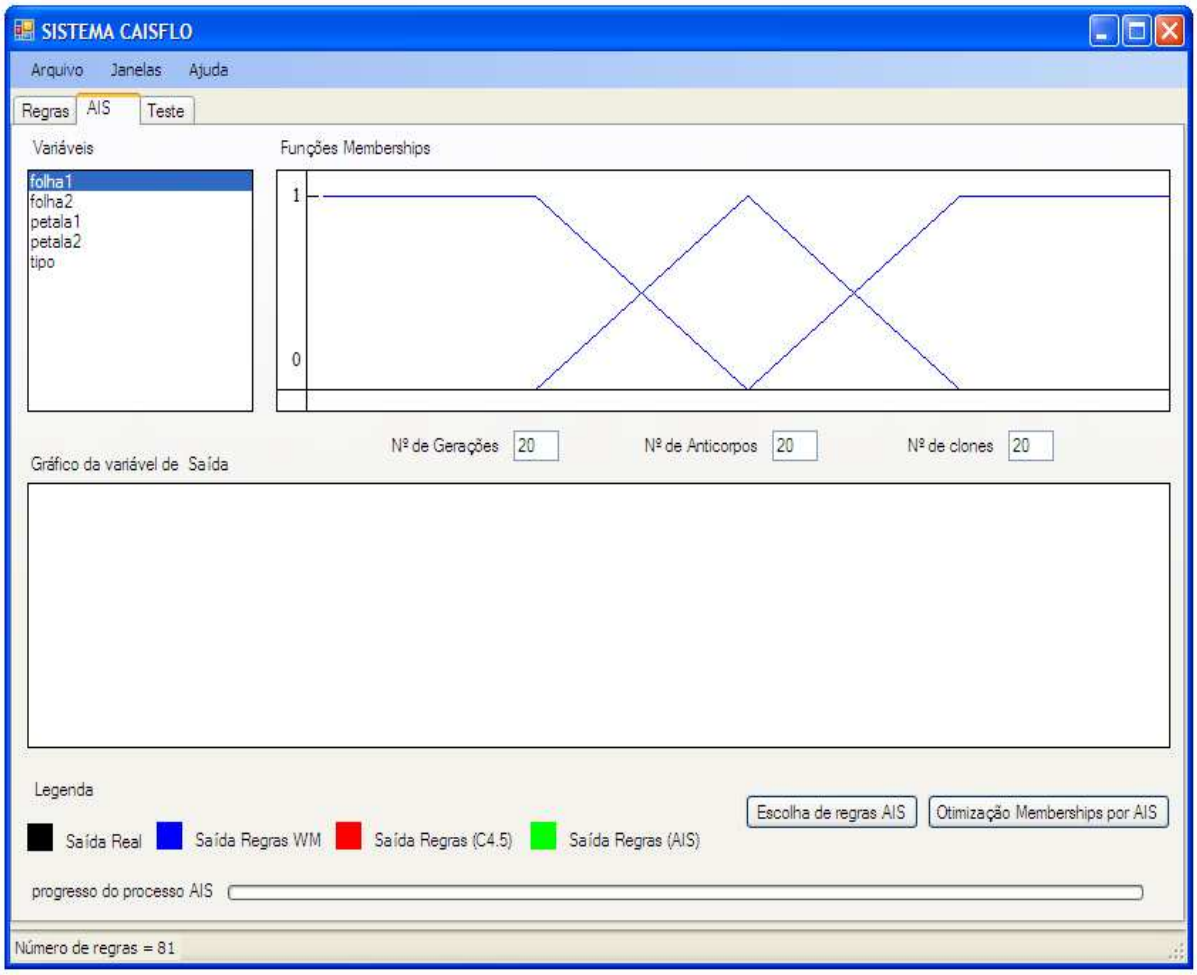
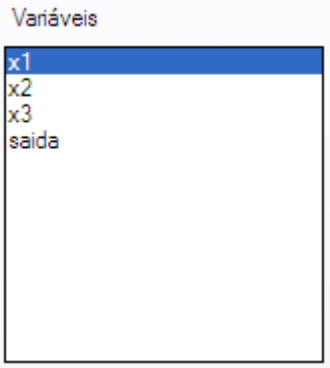
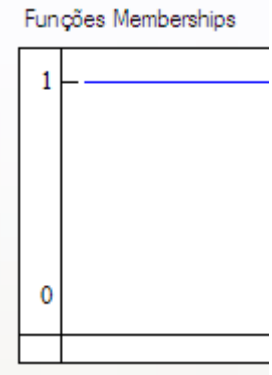


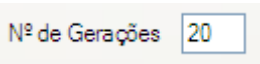
Figura 4.2 - Tela gerada pela aplicação – Aba SIA



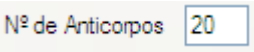
ListBox onde são mostradas as variáveis carregadas a partir dos arquivos de dados.



Panel onde são traçados os gráficos das funções memberships geradas a partir dos arquivos de dados. O grafico mostrado é referente à função membershiop da variável selecionada na ListBox *Variáveis*

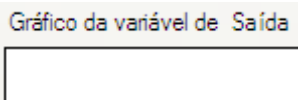


TextBox onde é digitada a quantidade de gerações a ser utilizada no processo de SIA



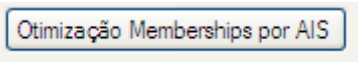
TextBox onde é digitada a quantidade de anticorpos a ser utilizada no processo de SIA

TextBox onde é digitada a quantidade de clones a ser utilizada no processo de SIA

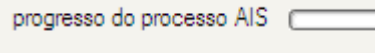


Panel onde é traçado o gráfico dos valores da variável de Saída.

Botão que aciona o processo de escolha das melhores regras fuzzy utilizando-se o método SIA



Botão que aciona o processo de otimização das funções memberships utilizando-se o método SIA



Barra de progresso para vizualização pelo usuário do progresso do processo que está sendo executado

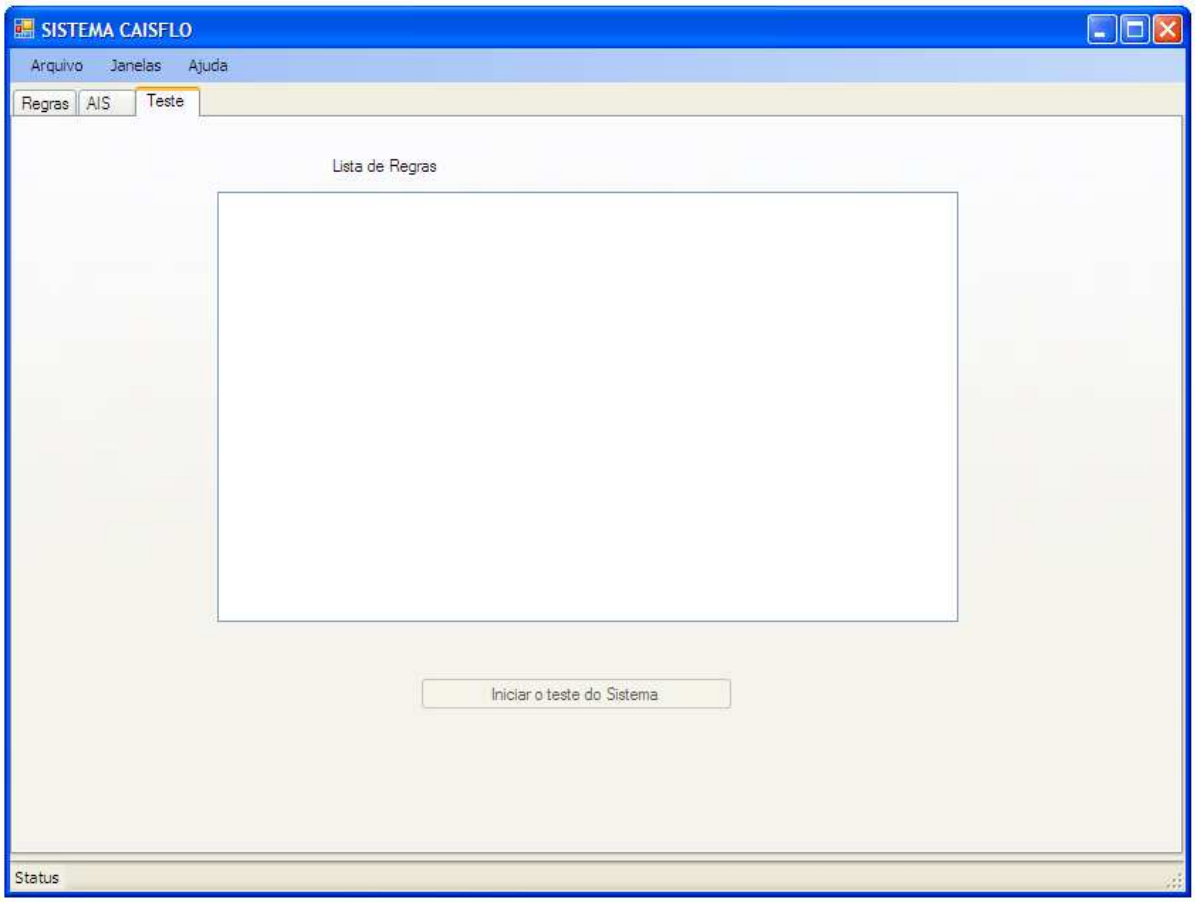
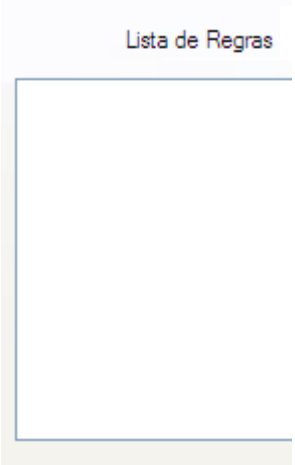
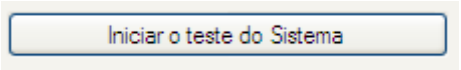


Figura 4.3 - Tela gerada pela aplicação – Aba Teste



Listbox onde são exibidas as regras geradas pelo sistema.



Botão que realiza o teste do sistema fuzzy gerado

A Figura 4.4, mostra uma tela da aplicação em operação. Nessa figura em específico o conjunto de dados é carregado.

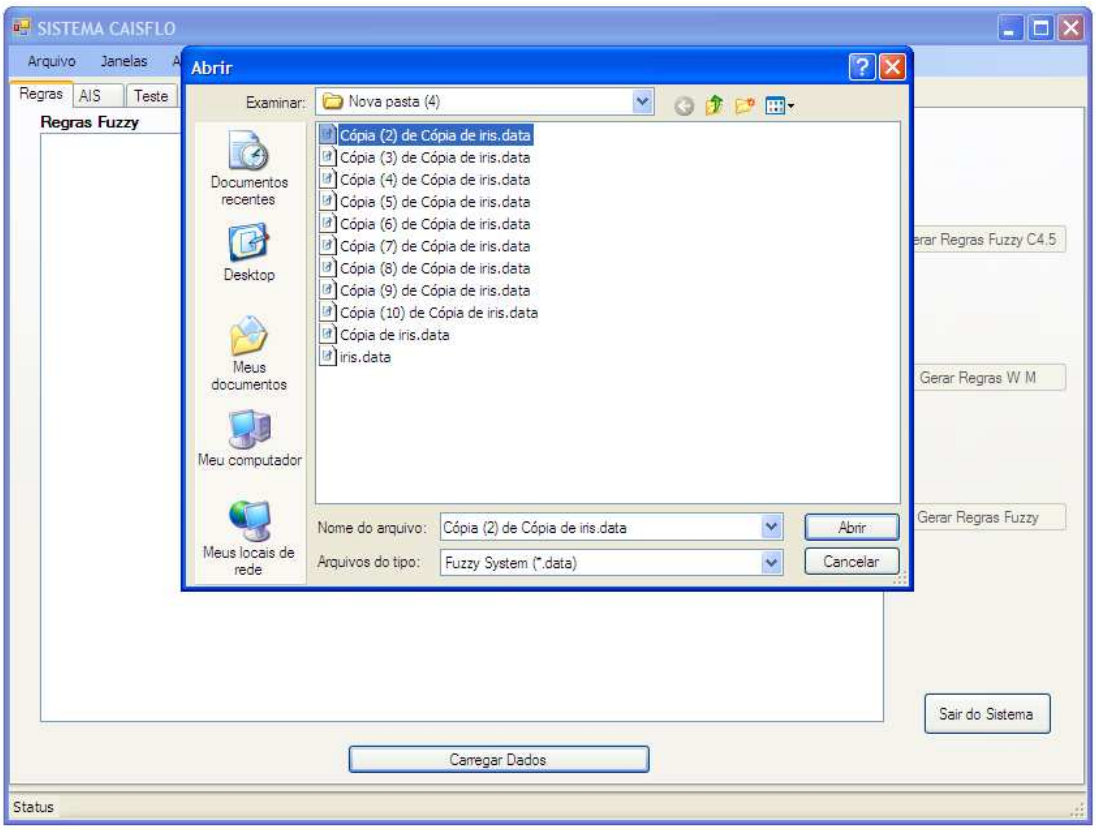


Figura 4.4 - Carregamento dos dados de treinamento

A Figura 4.5 mostra as regras geradas pela aplicação após o carregamento dos dados de entrada. Três abordagens podem ser acionadas para gerar as regras, o método C4.5, o método Wang e Mendel, e o método Wang e Mendel modificado para ser utilizado com o SIA.

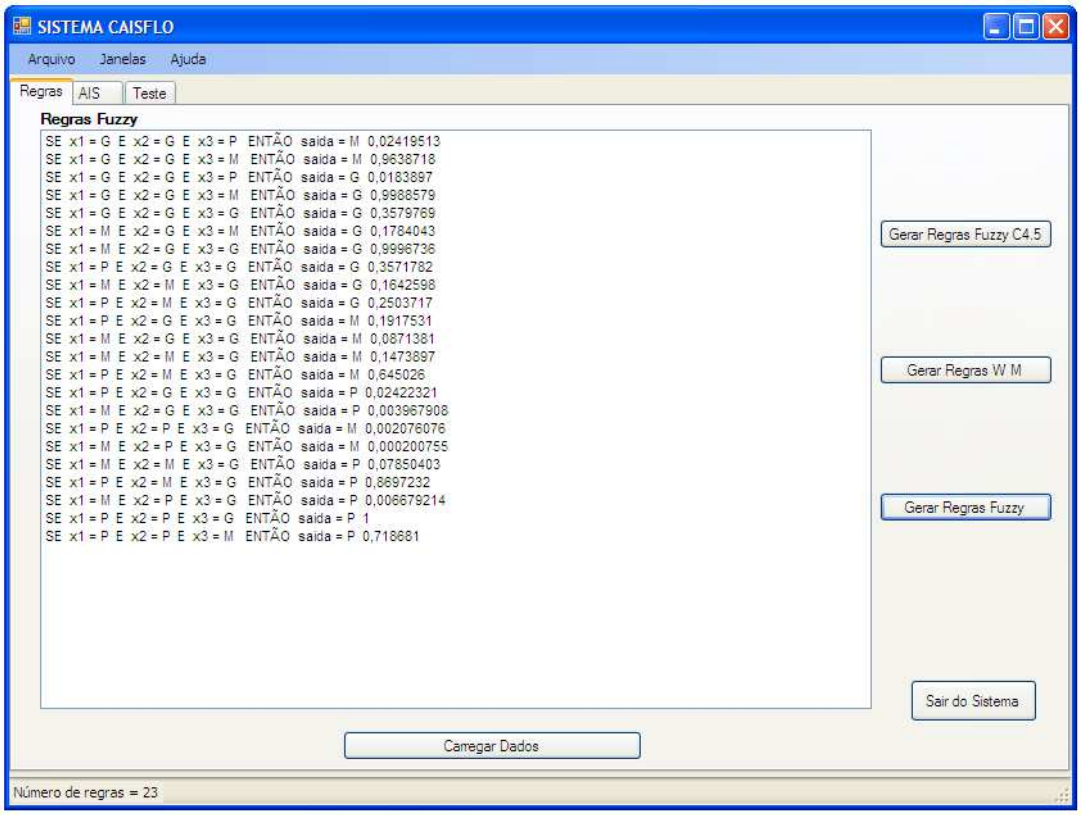


Figura 4.5 - Tela principal exibindo as regras geradas.

A Figura 4.6 ilustra os gráficos das saídas e das funções de pertinência. Em detalhe percebe-se a comparação dentre os gráficos gerados por um conjunto de valores de saída estimados pelo sistema utilizando-se as abordagens já mencionadas nessa dissertação. Notam-se também as funções de pertinência já com os pontos alterados pelo método SIA. Já a figura 4.7 ilustra a tela onde as regras geradas pela aplicação são mostradas para o operador. A figura 4.8 mostra a tela de carregamento do conjunto de dados de teste do sistema *fuzzy* gerado, e a

Figura 4.9 mostra a tela onde é feio o cálculo da acurácia do desse sistema com relação aos dados de teste carregados.

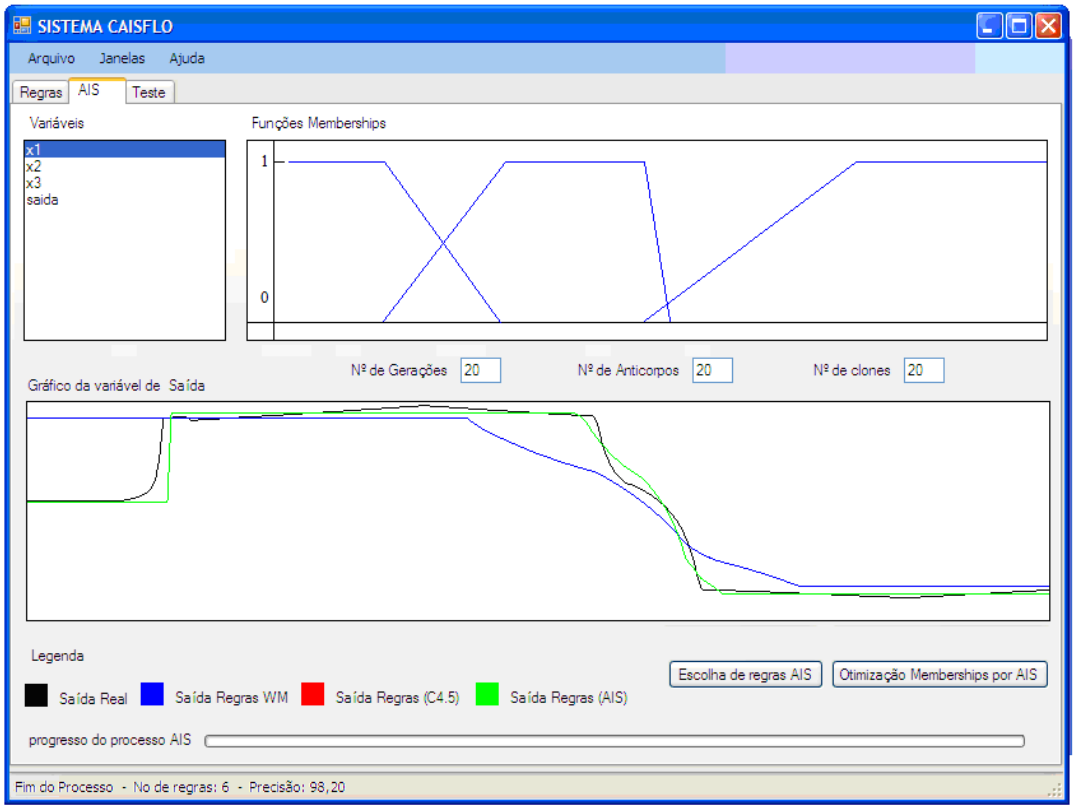


Figura 4.6 - Tela de exibição dos resultados obtidos – Processo de regressão.

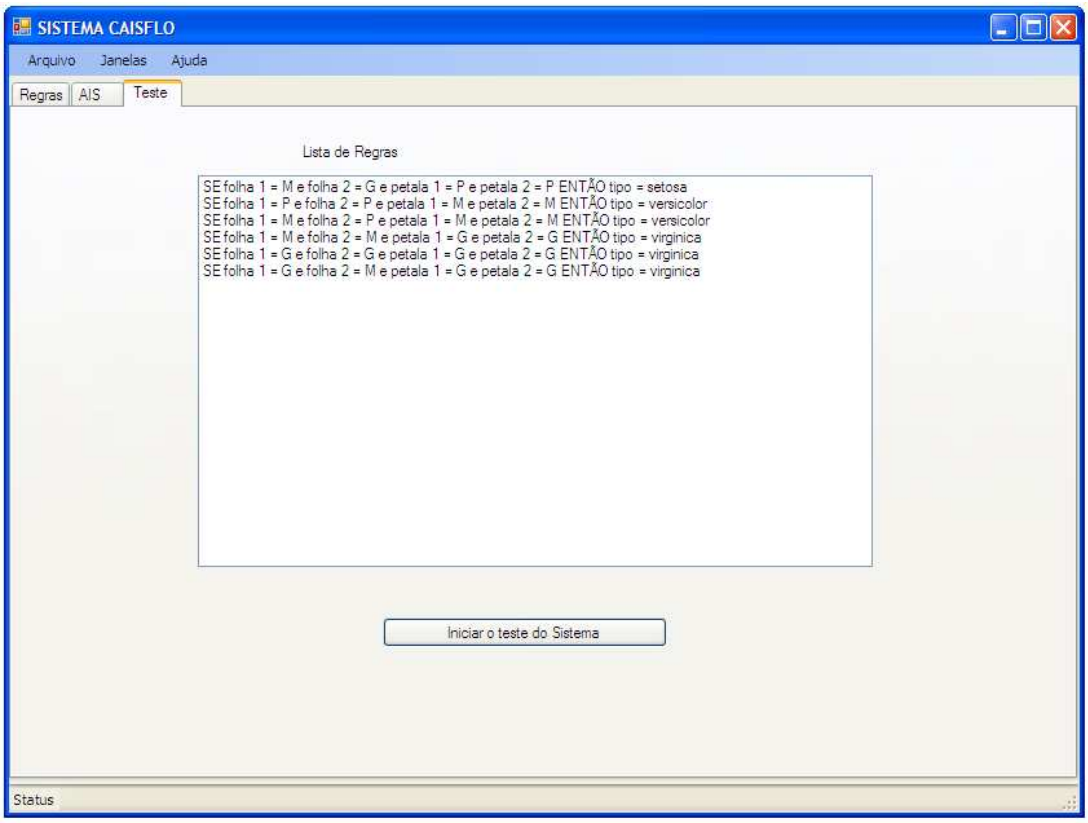


Figura 4.7 - Tela mostrando as regras geradas pelo sistema

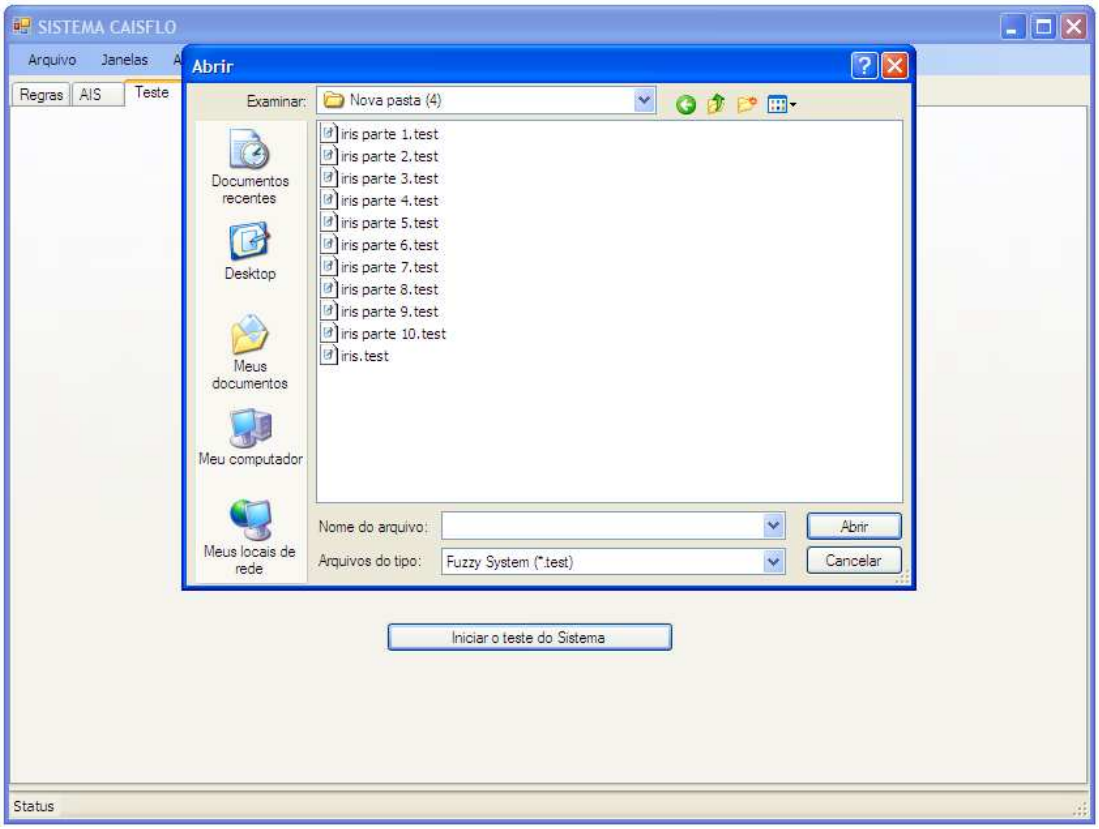


Figura 4.8 - Tela de carregamento dos dados de teste

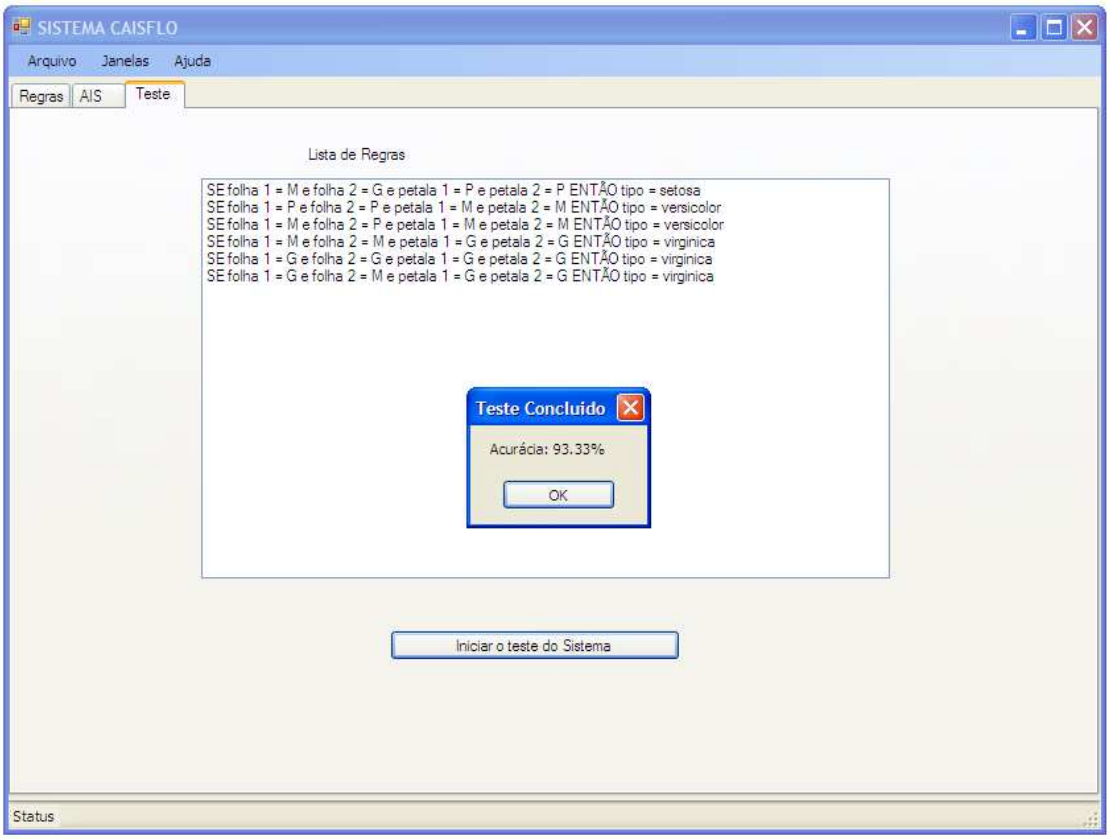


Figura 4.9 - Tela com o resultado da acurácia

4.2 Diagrama de Classes

A Figura 4.10 ilustra o diagrama de classes da ferramenta desenvolvida neste trabalho. A diante, cada uma das classes será detalhada.

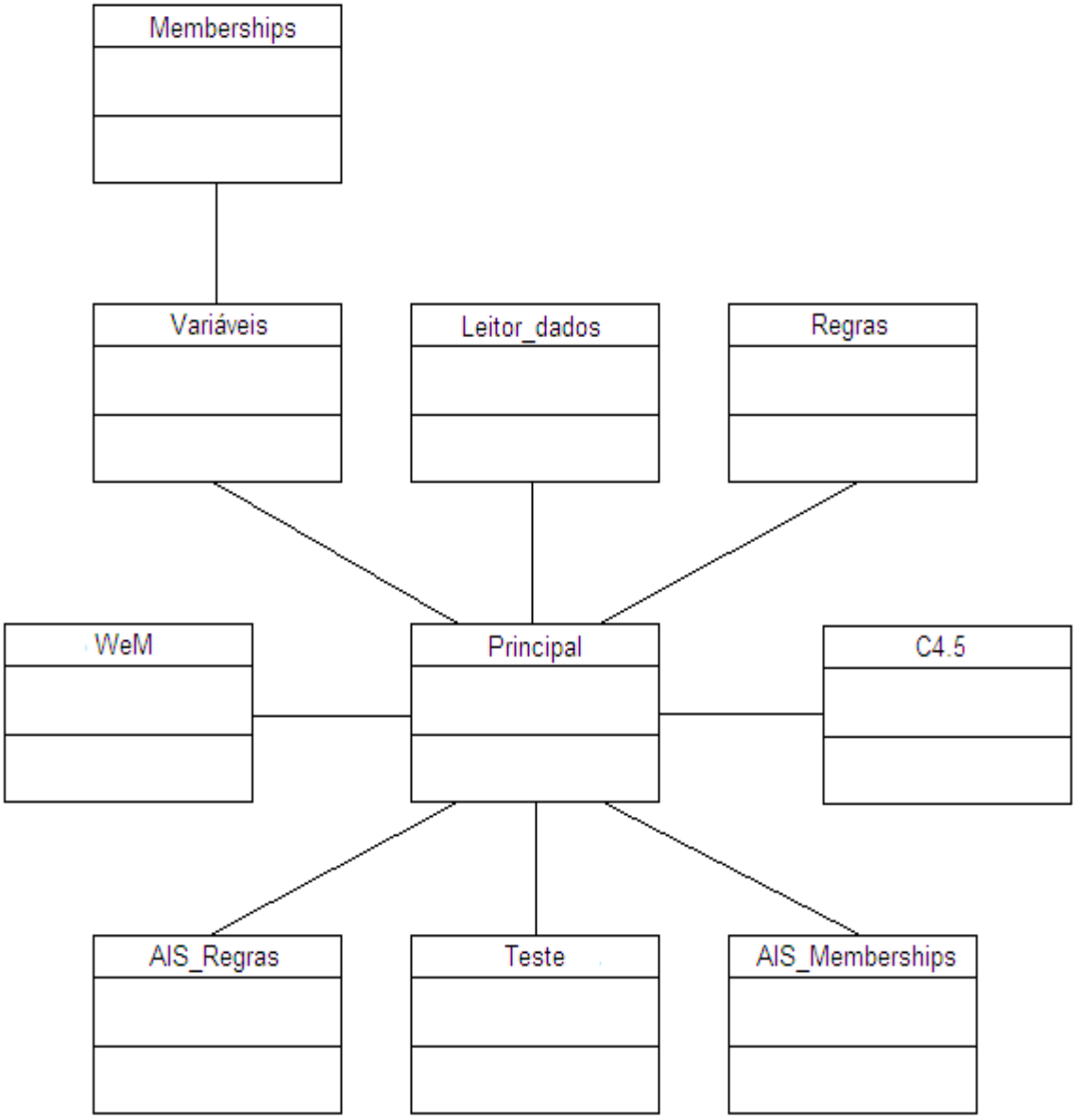


Figura 4.10 - Diagrama de classes do sistema desenvolvido

Sistema
Dados: DataTable Lista_Regras: List<Regras> Lista_Variaveis: List<Variaveis> Acurácia: Float
Gera_Regras() Ler_Dados() Desenha_Saidas() Gera_Variaveis(Dados) Otimiza_Regras() Otimiza_Memberships() Regras_C45() Regras_WM() Realiza_Teste()

Figura 4.11 - Classe Sistema

Classe Sistema: classe principal do sistema

Dados: tTabela onde ficam armazenados os dados de treinamento

Lista_Regras: lista onde ficam armazenadas as regras geradas

Lista_Variaveis: lista onde ficam armazenadas as variáveis geradas

Acurácia: float que armazena o valor da acurácia do sistema gerado

Ler_Dados: método que carrega a classe responsável por ler os dados e armazenálos em Dados e Dados_Teste.

Gera_Variaveis: método responsável por criar a Lista_Variáveis, carregando a classe *Variaveis* que é responsável por gerar as variáveis a partir dos Dados de treinamento lidos.

Gera_Regras: método responsável por criar a *Lista_Regras*, carregando a classe *Regras* que é responsável por gerar as regras a partir da *Lista_Variáveis* e *Dados*.

***Regras_C45*:** método responsável por criar a *Lista_Regras* utilizando a técnica C4.5, carregando a classe *Regras* que é responsável por gerar as regras a partir da *Lista_Variáveis* e *Dados*.

***Regras_WM*:** método responsável por criar a *Lista_Regras* utilizando a técnica de Wang e Mendel, carregando a classe *Regras* que é responsável por gerar as regras a partir da *Lista_Variáveis* e *Dados*.

Otimiza_Regras(): método responsável pela ativação do processo de escolha das melhores regras utilizando a técnica SIA, carregando a classe *SIA_Regras*.

Otimiza_Memberships(): método responsável pela ativação do processo de otimização das funções memberships a técnica SIA, carregando a classe *SIA_Memberships*.

Realiza_Teste(): método responsável pelo ativação doprocesso de teste do sistema, carregando a classe teste.

Leitor_Dados
Narquivo: string Linha_texto: string
Ler_Dados() Armazea_Dados()

Figura 4.12 - Classe Leitor_Dados

Classe Leitor_Dados: classe responsável pelo processo de leitura e armazenamento dos Dados etreinamento e de teste.

Nome_arquivo: string que armazena o nome do arquivo.

Linha_Texto: string que armazena uma linha(tupla) do arquivo de dados por vez.

Ler_Dados(): método responsável por ler cada tupla no arquivo de dados e armazená-lo em *Linha_texto*.

Armazena_Dados(): método responsável por armazenar os valores de *Linha_texto* na tabela *Dados*.

Variaveis
Formato: String Tipo: String Min: Float Max: Float Lista_Memberships: List<Memberships>
Gera_Variavel() Adiciona_Memberships()

Figura 4.13 - Classe Variáveis

Classe Variáveis: classe responsável pela geração das variáveis do sistema.

Formato: string que armazena o formato da variavel, podendo ser numérico ou classificativo

Tipo: string que armazena o tipo da variável podendo ser entrada ou saída

Min: float que armazena o valor mínimo da variável obtido a partir dos dados de treinamento.

Max: float que armazena o valor máximo da variável obtido a partir dos dados de treinamento.

Lista_Memberships: Lista que armazena as funções memberships da variável

Gera_Variavel: método que gera cada variável.

Adiciona_Memberships: método responsável por gerar cada uma das funções memberships da variável e armazená-las na *Lista_Memberships*.

Memberships
Nome: String Pertinência: Float Pontos: PointF[]
Calcula_Centroide() Atualiza_Pontos() Calcula_Pertinência()

Figura 4.14 - Classe Memberships

Classe Memberships: classe responsável por gerar as funções memberships de cada variável

Nome: string onde é armazenado o nome da função membership.

Pertinência: float onde é armazenado o valor do grau de pertinência da função membership com relação a um valor dos dados.

Pontos: pointF[] onde ficam armazenados os valores das coordenadas dos pontos de cada função membership.

Calcula_Centroide(): método que faz o cálculo do centroide para posterior realização da defuzificação. Só é ativado quando a variável é do tipo numérica.

Atualiza_Pontos(): método responsável pela atualização dos valores dos pontos durante o processo de otimização das funções memberships.

Calcula_Pertinência(): método responsável por calcular a o valor da pertinência de um dado numérico com relação a um certo membership e armazená-lo em *Pertinencia*.

Regra
Antescedentes: String[] Consequente: String Fitness: Float ID: Int
Gera_Regra() Calcula_Fuzzy()

Figura 4.15 - Classe Regra

Regra: classe responsável pela geração das regras fuzzy.

Antescedentes: string[] onde são armazenados os valores fuzzy da parte antescedente da regra gerada.

Consequente: string onde é armazanado o valor fuzzy da parte consequente da regra gerada.

Fitness: float onde fica armazenado o valor do fitness da regra em relação aos dados.

ID: int onde fica armazenado o valor de identificação da regra.

Gera_Regra(): método responsável por gerar a regra.

Calcula_Fuzzy(): método responsável pela conversão de um dado numérico em fuzzy.

WeM
Maxpert: Float
Gera_Regra()

Figura 4.16 - Classe WeM

Classe WeM: classe responsável por gerar sistema fuzzy pela técnica de Wang e Mendel.

Maxpert: float onde fica armazenado o valor da maior pertinência entre as memberships das variáveis

Gera_Regra: método responsável por gerar as regras fuzzy pela técnica de Wang e Mendel.

C45
Ganho: Float Entropia: Float Arvore:
Gera_Arvore() Calcula_Ganho() Calcula_Entropia()

Figura 4.17 - Classe C45

Classe C45: classe responsavel pela geração de regras utilizando-se a técnica C4.5

Entropia: float onde é armazenado o valor da entropia de um conjunto de dados

Ganho: float onde é armazenado o valor do ganho de um dado atributo.

Arvore: onde é armazenado a árvore de decisão C4.5.

Gera_Arvore: método responsável por gerar a árvore de decisão C4.5

Calcula_Ganho: método responsável pelo cálculo do Ganho.

Cálculo_Entropia: método responsável pelo cálculo da entropia.

SIA_Regras
Fitness: Float Lista_Anticorpos: List<Regra> Lista_Clones: List<Regra>
Calcula_Fitness() Clone() Gera_Anticorpos() Gera_Clones() Seleciona_Melhores() Mutação()

Figura 4.18 - Classe SIA_Regras

SIA_Regras: classe responsável pelo processo de escolha das melhores regras utilizando-se a técnica SIA

Fitness: float onde é armazenado o valor do Fitness de um dado anticorpo ou clone em relação aos dados.

Lista_Anticorpos: lista onde são armazenados os anticorpos gerados.

Lista_Clones: lista onde são armazenados os clones gerados

Calcula_Fitness: método responsável pela cálculo do fitness.

Gera_Anticorpos: método responsável por gerar os anticorpos.

Gera_Clones: método responsável por gerar os clones.

Seleciona_Melhores: método responsável por selecionar os melhores anticorpos ou clones.

Mutação: método responsável por realizar a mutação dos clones.

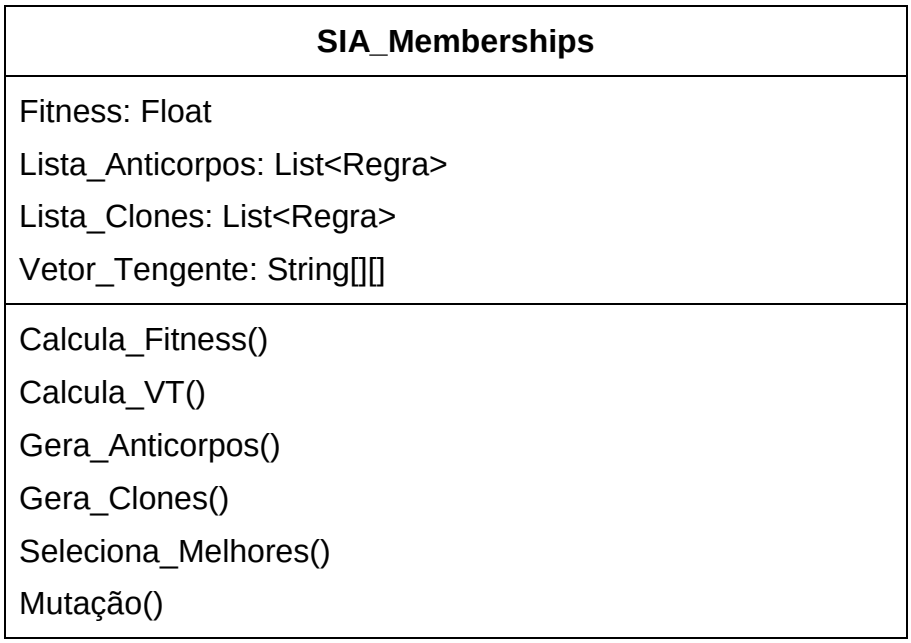


Figura 4.19 - Classe SIA_Memberships

SIA_Memberships: classe responsável pelo processo de otimização das funções memberships utilizando-se a técnica SIA.

Fitness: float onde é armazenado o valor do Fitness de um dado anticorpo ou clone em relação aos dados.

Lista_Anticorpos: lista onde são armazenados os anticorpos gerados.

Lista_Clones: lista onde são armazenados os clones gerados.

Vetor_Tangente: matriz onde é armazenado o vetor tangente.

Calcula_Fitness: método responsável pela cálculo do fitness.

Calcula_VT: método responsável pelo cálculo do vetor tangente.

Gera_Anticorpos: método responsável por gerar os anticorpos.

Gera_Clones: método responsável por gerar os clones.

Seleciona_Melhores: método responsável por selecionar os melhores anticorpos ou clones.

Mutação: método responsável por realizar a mutação dos clones.

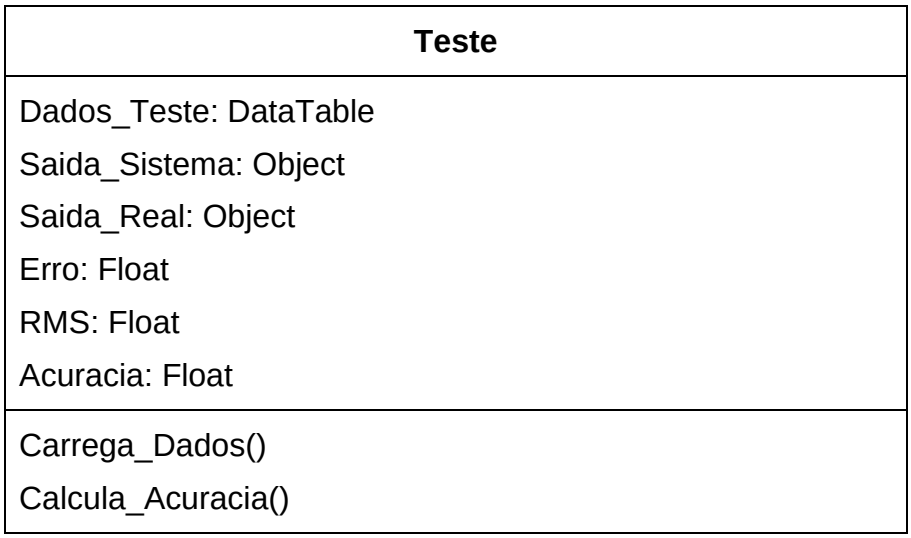


Figura 4.20 - Classe Teste

Classe Teste: classe responsável pelo processo de teste do sistema fuzzy gerado.

Dados_Teste: tabela onde ficam armazenados os dados de teste.

Saída_Sistema: float onde fica armazenada o valor de saída fornecida pelo sistema para um determinado dado de entrada.

Saída_Real: float onde fica armazenado o valor de saída fornecido pelo próprio dado de entrada.

Erro: float onde fica armazenado o valor do erro.

RMS: float onde fica armazenado o valor da média da raiz do erro quadrático.

Acuracia: float onde fica armazenado o valor da acurácia calculada.

Carrega_Dados: método responsável por calcular os dados de teste, chamando a classe *Leitor_Dados.*

Calcula_Acurácia: método responsável pelo cálculo da acurácia do sistema fuzzy gerado.

CAPÍTULO 5

RESULTADOS

A fim de se obter os resultados para realização de uma análise comparativa do sistema desenvolvido com outros algoritmos, foram utilizados alguns conjuntos de dados obtidos da base de dados da UCI (Universidade da Califórnia e Irvine) de domínio público disponíveis através do endereço na internet http://www.ics.uci.edu/~mlearn/MLRepository.html para o teste do processo para problema de classificação, e o conjunto de dados utilizado em (Vermaas,2009) para o teste do processo para tarefa de regressão.

Para a realização dos testes, a metodologia utilizada na obtenção dos resultados foi a validação cruzada 10-fold (Hand, 1997), onde os dados são divididos em 10 partes de igual tamanho. São realizados 10 testes utilizando 90% dos dados para treinamento e 10% para o teste. A cada teste, utilizam-se diferentes conjuntos para treinamento e teste.

Por inferência, verificou-se que os melhores resultados foram obtidos quando se aplicou 30 gerações, 30 anticorpos e 30 clones. Quantidades pequenas de gerações, anticorpos e clones resultaram em processo mais rápido, porém em resultados insatisfatórios. Quantidades grandes tornaram o processo demorado, além de não provocar melhorias sensíveis nos resultados.

5.1 Tarefas de Classificação:

A comparação dos resultados e avaliação do desempenho do algoritmo CAISFLO, foi realizada utilizando-se a outros algoritmos que também se baseiam no principio evolucionário como o CEFR-MINER (Mendes, 2001), ESIA (Liu, 2000), BGP (Rouwhorst, 2000) e IFRAIS (Alves, 2004).

Os conjuntos de dados utilizados para as comparações foram Iris, Crx, Câncer, pois estes conjuntos foram utilizados nos testes dos algoritmos a qual estamos comparando.

- **CRX**

É um conjunto de dados que fornece informações de movimentações financeiras com cartão de crédito. É um conjunto interessante de ser usado em classificadores, pois possui atributos dos diferentes formatos. Possui pequena quantidade de valores ausentes, totalizando 5% do total.

Possui um 15 atributos de entrada, sendo 6 contínuos e 9 categóricos, e um atributo de saída do tipo classificativo que pode ser positivo ou negativo.

Possui 690 tuplas dedados. Por possuir muitas tuplas e variáveis acaba gerando grande quantidade de regras e possui tempo de processamento longo.

A Figura 5.1 ilustra as variáveis da base de dados CRX.

```
 A1: b, a.
 A2: contínuo.
 A3: contínuo.
 A4: u, y, l, t.
 A5: g, p, gg.
 A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
 A7: v, h, bb, j, n, z, dd, ff, o.
 A8: contínuo.
 A9: t, f.
A10: t, f.
A11: contínuo.
A12: t, f.
A13: g, p, s.
A14: contínuo.
A15: contínuo.
A16: +,-   (classe)
```

Figura 5.1 - Variáveis da base de dados CRX

- **Iris**

Uma das bases de dados mais utilizadas na literatura. Fornece informações sobre a planta Iris, com três espécies diferentes: Setosa, Versicolor e Virginica. Essa base de dados possui quatro atributos de valor contínuo e um atributo de saída classificativo, conforme pode ser visto na Figura 5.2. Possui 150 tuplas de dados, sendo 50 para cada espécie da planta. Não possui valores ausentes.

Por ser uma base de dados de poucos valores, gera uma pequena quantidade de regras, que podem ser vistas na Figura 5.3.

```
1. largura da sépala em cm
2. comprimento da sépala em cm
3. largura da pétala em cm
4. comprimento da pétala em cm
5. classe:
   -- Iris Setosa
   -- Iris Versicolour
   -- Iris Virginica
```

Figura 5.2 - Variáveis da base de dados Iris

```
SE X1 = P  E X2 = M  E X3 = G  E X4 = G  ENTÃO  Y = VIRGINICA
SE X1 = P  E X2 = P  E X3 = M  E X4 = M  ENTÃO  Y = VERSICOLOR
SE X1 = G  E X2 = M  E X3 = G  E X4 = G  ENTÃO  Y = VERSICOLOR
SE X1 = G  E X2 = P  E X3 = G  E X4 = G  ENTÃO  Y = VIRGINICA
SE X1 = M  E X2 = P  E X3 = M  E X4 = M  ENTÃO  Y = VERSICOLOR
SE X1 = P  E X2 = M  E X3 = P  E X4 = P  ENTÃO  Y = SETOSA
SE X1 = M  E X2 = M  E X3 = G  E X4 = M  ENTÃO  Y = VERSICOLOR
SE X1 = P  E X2 = P  E X3 = G  E X4 = G  ENTÃO  Y = VIRGINICA
SE X1 = P  E X2 = P  E X3 = G  E X4 = M  ENTÃO  Y = VERSICOLOR
SE X1 = M  E X2 = G  E X3 = P  E X4 = P  ENTÃO  Y = SETOSA
SE X1 = P  E X2 = P  E X3 = P  E X4 = P  ENTÃO  Y = SETOSA
SE X1 = G  E X2 = P  E X3 = G  E X4 = M  ENTÃO  Y = VERSICOLOR
SE X1 = G  E X2 = P  E X3 = G  E X4 = G  ENTÃO  Y = VIRGINICA
```

Figura 5.3 - Regras geradas pela base de dados Iris

- **Câncer**

É um conjunto de dados obtidos a partir de experimentos realizados pelo Hospital da Universidade de Wisconsin, nos Estados Unidos com pacientes portadores de câncer de mama. Possui 10 atributos de entrada com valores de 1 a 10, informando certas características das células da glândula mamária, e um atributo de saída do tipo classificatório, dizendo se o tumor é benigno ou maligno. As variáveis são mostradas na Figura 5.4.

Possui 699 tuplas de dados no total, o que leva a gerar grade quantidade de regras.

```
1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Class: (2 for benign, 4 for malignant)
```

Figura 5.4– Variáveis da base de dados Câncer

A Tabela 5.1 mostra os resultados da acurácia do CAISFLO como classificador e a comparação com outras abordagens de algoritmos de classificação presente na literatura.

Tabela 5.1 - Precisão dos resultados para o processo de classificação

	CAISFLO	**CEFR-MINER**	**IFRAIS**	**ESIA**	**BGP**
Crx	86,7	84,7	86,29	77,39	N/A
Iris	95,33	*95,3*	*N/A*	95,33	94,1
Cancer	95,5	N/A	95,75	N/A	N/A

Como pode ser visto na Tabela 5.1, o algoritmo CAISFLO obteve resultados semelhantes a outras abordagens da literatura.

5.2 Tarefa de Regressão

Para comparação dos resultados, foram utilizados os algoritmos (Wang, 1992) e C4.5 que foram implementados juntamente no sistema desenvolvido nesse trabalho.

Os dados aqui utilizados, retirados do trabalho proposto por (Vermaas,2009), foram colhidos a partir de um estacionamento de um veículo. Para isso, utilizou-se um programa de simulação de estacionamento em 3D que reproduz a dinâmica do veículo. Foram então obtidas 4 informações conforme visto na Figura 5.5:

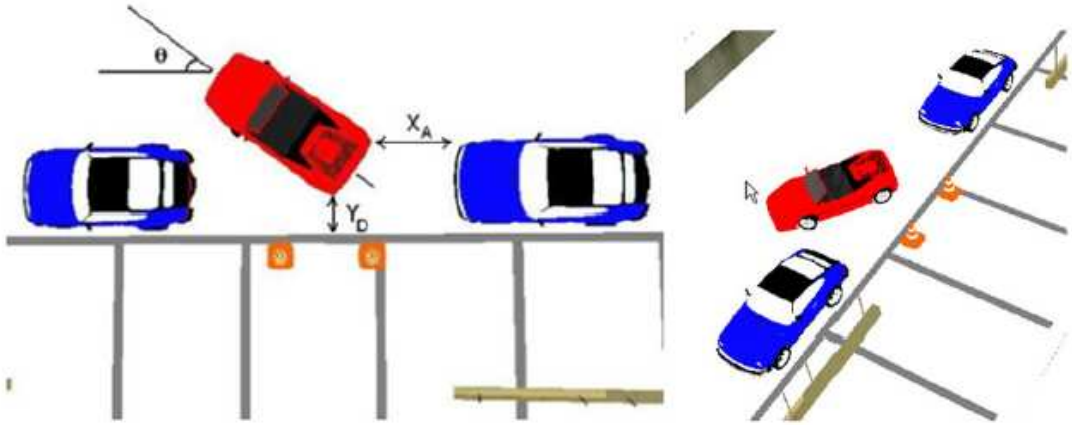


Figura 5.5 - Variáveis colhidas pelo simulador de estacionamento

- X_A – Distância em metros da traseira do veículo a um obstáculo localizado atrás do mesmo;
- Y_D – Distância em metros da lateral esquerda traseira do veículo até a guia;
- θ - Ângulo em graus que o eixo longitudinal do veículo faz com relação à guia;
- Saída – Ângulo em graus do volante operado pelo usuário.

Foram obtidas 256 tuplas, com valores numéricos contínuos para cada uma das 4 variáveis.

A Figura 5.6, a Figura 5.7 e a Figura 5.8 ilustram os gráficos dos resultados utilizando os algoritmos de (Wang, 1992), C4.5 e CAISFLO respectivamente.

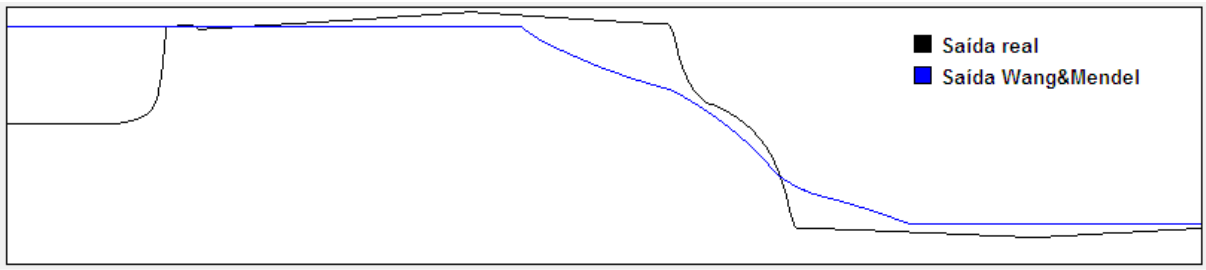


Figura 5.6 - Saídas geradas pelo método (Wang,1992)

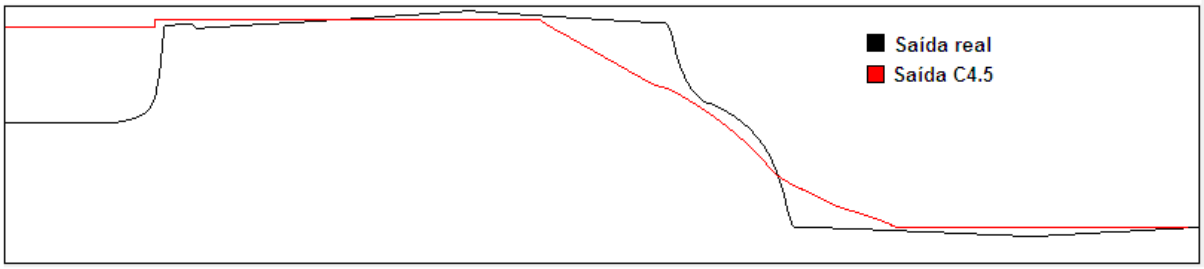


Figura 5.7 - Saídas geradas pelo método C4.5

Como se percebe, os dados de saída dos algoritmos de (Wang, 1992) e C4.5 apresentaram diferenças sensíveis com relação aos dados reais. Portanto o sistema gerado por esse algoritmo não pode ser utilizado para programar um controlador para o veículo.

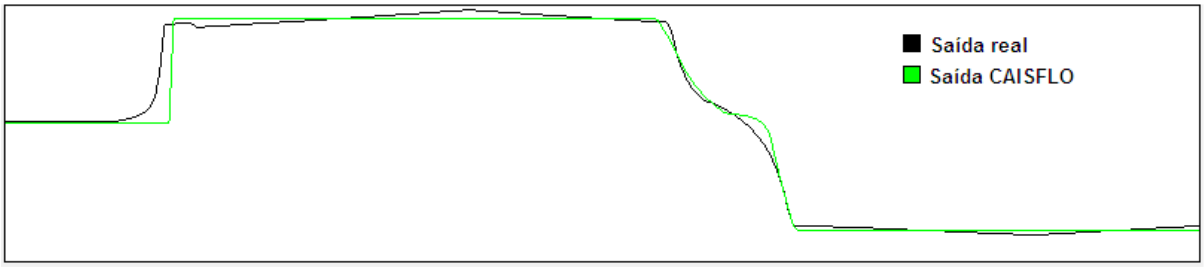


Figura 5.8 - Saídas geradas pelo CAISFLO

Já os dados de saída obtidos apartir do algoritmo CAISFLO mostram um resultado bem mais preciso, quase coincidindo totalmente com os dados reais. O sistema fuzzy pelo algoritmo gerado é mais apto para programar um controlador.

A Figura 5.9 mostra As funções de pertinência geradas pela aplicação.

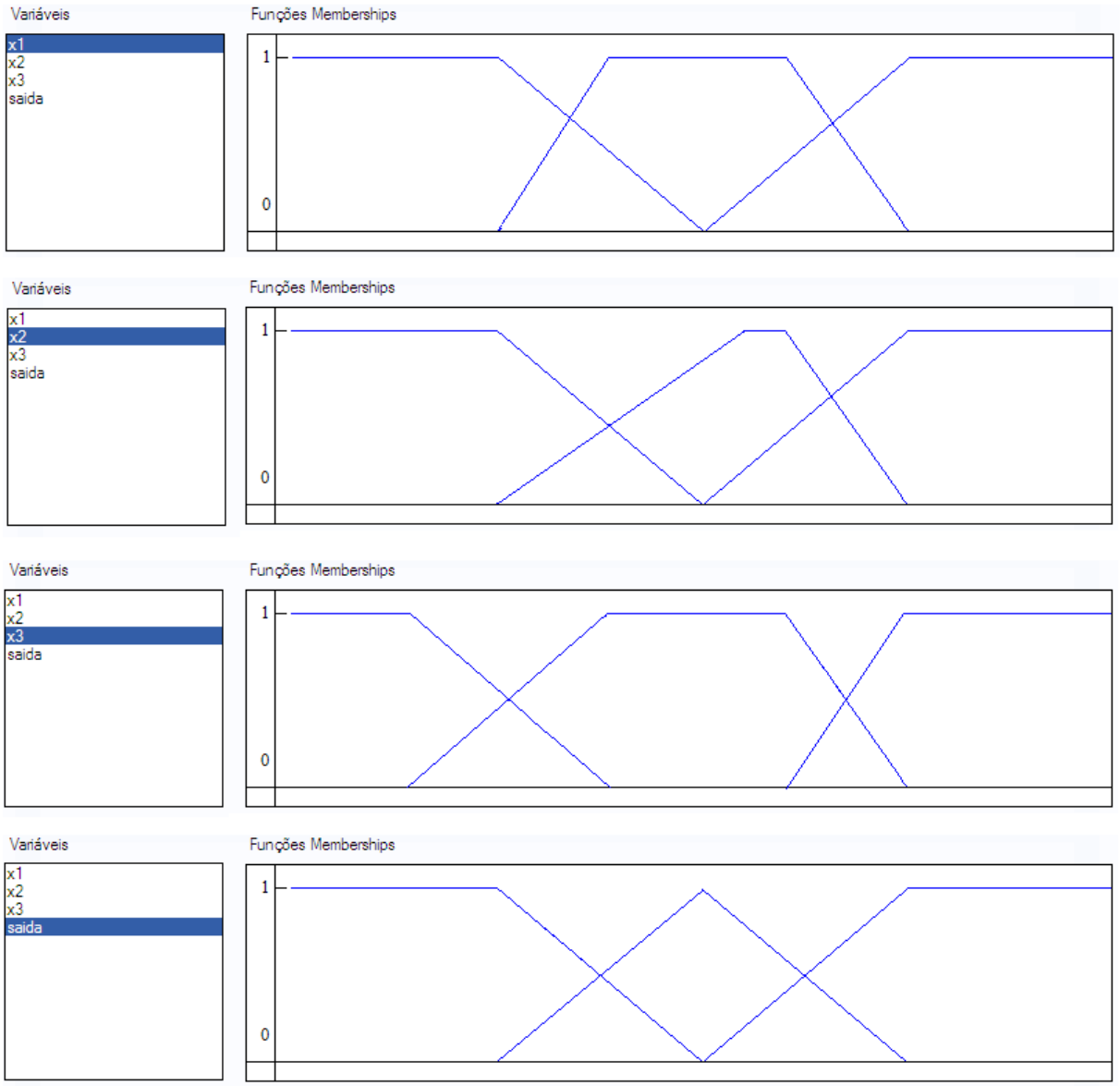


Figura 5.9 - Funções de pertinência geradas pela aplicação.

A Figura 5.10 apresenta as regras geradas pela aplicação.

```
SE X1 = P  E X2 = P  E X3 = G  ENTÃO  Y = P
SE X1 = P  E X2 = G  E X3 = G  ENTÃO  Y = G
SE X1 = M  E X2 = P  E X3 = G  ENTÃO  Y = M
SE X1 = M  E X2 = M  E X3 = G  ENTÃO  Y = P
SE X1 = M  E X2 = G  E X3 = G  ENTÃO  Y = G
SE X1 = G  E X2 = G  E X3 = P  ENTÃO  Y = M
SE X1 = G  E X2 = G  E X3 = G  ENTÃO  Y = G
```

Figura 5.10 - Regras Geradas pela aplicação

A tabela 5.2 apresenta a precisão dos resultados para os três algoritmos.

Tabela 5.2 - Precisão dos resultados para o processo de regressão

Algoritmo	**Wang**	**C4.5**	**CAISFLO**
Precisão em %	88,77	89,80	98,2

Como se pode perceber pelas Figuras 5.6, 5.7 e 5.8, e pela Tabela 5.2, o algoritmo CAISFLO gera saídas contínuas com maior precisão do que os outros dois algoritmos utilizados como base de comparação.

CAPÍTULO 6

CONCLUSÃO

Nesta dissertação, foi implementada e testada uma nova abordagem para a geração e otimização de um sistema *fuzzy* composto pela base regras e pelas funções de pertinência aplicando o conceito de sistemas imunológicos artificiais. Essa nova abordagem é baseada nos estudos apresentados no trabalho em (Vermaas, 2009), em que foi apresentado um novo algoritmo denominado CAISFLO. Neste trabalho, o algorítmo foi utilizado tanto para tarefas de regressão como de classificação.

No algoritmo CAISFLO, as tarefas de escolha do melhor conjunto de regras e de otimização das funções memberships são realidadas utilizando-se a abordagem dos Sistema Imunológicos Artificiais, que simula o sistema imunológico humano sendo utilizado em problemas de otimização e reconhecimento de padrões, de forma co-evolutiva, em que duas populações distintas foram utilizadas para cada uma das tarefas.

Durante o trabalho, foi desenvolvida uma aplicação computacional implementando o algoritmo CAISFLO, aplicando adaptações para realização das duas taefas descitas anteriormente, utilizando-se a linguagem C# e o ambiente de desenvolvimento Visual Studio da Microsoft.

O sistema proposto foi testado e validado utlizando-se bases de dados de domínio público para o método de classificação e um conjunto de dados de controle de estacionamento de um automóvel para o método de regressão. Para a obtenção do resultado da acurácia, a metodologia escolhida foi a validação cruzada 10-fold.

Para a tarefa de classificação, os resultados obtidos indicam que o sistema obteve resultados similares a alguns algorítimos presentes na literatura que utilizam o principio evolucionário na extração de regras.

Para o método de regressão o sistema mostrou-se bem eficiente, dando uma precisão muito grande para o conjunto de dados testado.

Ressalta-se porém que o algoritmo proposto possui certas desvantagens, as quais cita-se:

- ***Quantidade de regras -*** Para conjuntos de dados que possui grande quantidade de atributos e instancias, é gerado uma grande quantidade de regras, que possuem tamanho proporcional ao numero de atributos.
- ***Tempo de processamento -*** Para grandes quantidades de dados, o tempo de processamento se tornou muito grande, o que tornou o processo de aprendizagem lento.

Ressalta-se ainda que o algoritmo possui validade pois apresentou resultados satisfatórios comparados a outras abordages presentes na literatura.

Com o objetivo de melhorar os resultados e minimizar os pontos negativos aqui já mencionados, propõe-se testar novas metodologias de extração de regras, como a indução de árvores, podas das regras, utilização de outros tipos de funções de pertinência a fim de diminuir a quantidade de regras e melhorar os resultados obtidos.

REFERÊNCIAS BIBLIOGRÁFICAS

Abe, S.; Lan, M. Fuzzy rules extraction directly from Numerical Data for Function Approximation. *IEEE Transactions on Systems, Man, and Cybernetics 25(1),* 119–129, 1995.

Abido, M. A. A Niched Pareto genetic algorithm for multiobjective environmental/ economic dispatch. *Journal of Electrical Power and Energy Systems 25(2),* 97–105, 2003.

Adamo, J. M. Fuzzy decision trees. *Fuzzy Sets and Systems, 4(3),* 207–219, 1980.

Agrawal, R; Srikant, R. Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases.* San Francisco: Morgan Kaufmann, 1994.

Alves, R. T.; Delgado, M. R.; Lopes, H. S.; Freitas, A. A. An Artificial Immune System for Fuzzy-Rule Induction in Data Mining, Proc. Parallel Problem Solving from Nature *(PPSN-2004), LNCS 3242*, pp. 1011-1020, 2004.

Bäck, T. Evolutionary Algorithms in Theory and Practice, Oxford University Press, New York, 1996.

Baldwin, J. F. Fuzzy logic and fuzzy reasoning - *in Fuzzy Reasoning and Its Applications, E.H. Mamdani and B.R. Gaines, (eds.),* London: Academic Press, 1981.

Bezdek, J. C. Pattern Recognition with Fuzzy Objective Function Algorithms, 1981.

Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. Classification and Regression Trees. *Wadsworth International Group*, Belmont, CA, 1984.

Carse, B.; Fogarty, T. C.; Munro, A. Evolving fuzzy rule based controllers using genetic algorithms - *Fuzzy Sets Syst., vol. 80*, pp. 273–293, 1996.

Chen, G.; Wei, Q.; Kerre, E.; Wets, G. Overview of fuzzy associations mining. *In Proc. ISIS - 2003, 4th International Symposium on Advanced Intelligent Systems.* Jeju, Korea, 2003.

Cordón, O.; del Jesus, M. J.; Herrera, F. Genetic learning of fuzzy rule-based classification systems co-operating with fuzzy reasoning methods, *Int. J. Intell. Syst.,* vol. 13, pp. 1025–1053, 1998.

Cordón, O., Gomide, F., Herrera, F., Hoffmann, F. e Magdalena, L. Ten years of genetic fuzzy systems: current framework and new trends*. Fuzzy Sets and Systems,* pp. 141, 2004.

Dasgupta, D. Artificial Immune Systems and Their Applications, Ed. Springer-Verlag, 1999.

De Castro, L. N.; Von Zuben, F. J. Learning and Optimization Using the Clonal Selection Principle, *IEEE Transaction on Evolutionary Computation, Special Issue sobre Sistemas Imunológicos Artificiais*, 2001.

De Jong, K. A.; Spears, W. M.; Gordon, D. F. Using genetic algorithms for concept learning. *Machine Learning 13*, pp. 161-188, 1993.

Delgado, M.; Marin, N.; Sanchez, D.; Vila, M. A. Fuzzy association rules: general model and applications. *IEEE Transactions on Fuzzy Systems, 11(2),* pp. 214 - 225, 2003.

Dubois, H. P. D. What are fuzzy rules and how to use them. *Fuzzy Sets and Systems, 84,* pp.169–185, 1996.

Dunn, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics 3*, pp. 32-57, 1973.

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, Advances in Knowledge Discovery and Data Mining. AAAI Press. Menlo Park, Calif, 1996.

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. From Data Mining to Knowledge Discovery: An Overview. *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Menlo Park, Calif., 1996, pp. 1–34.

Fayyad, U. M.; Piatetski-Shapiro, G.; Smyth, P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM,* pp. 27-34, 1996.

Fonseca, C. M.; Fleming, P. J. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation, Vol. 3(1),* pp. 1-16, 1995

Freitas, A. A. Data Mining and Knowledge Discovery with Evolutionary Algorithms. Berlin, Germany: Springer-Verlag, 2002.

Gasch, A.P; Eisen, M. B. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. Genome Biology, 2002.

Gimenes, E. Data Mining - Data Warehouse. A Importância da Mineração de Dados em Tomadas de Decisão. 2000. Monografia (Graduação em Processamento de Dados) - Faculdade de Tecnologia de Taquaritinga, Taquaritinga, 2000.

Giordana, A.; Neri, F. Search-intensive concept induction. *Evolutionary Computation 3(4),* pp. 375-416, Winter, 1995.

Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning, Addison- Wesley, 1989.

Greene, D. P. e Smith, S. F. Competition-based induction of decision models from examples. *Machine Learning 13*, pp. 229-257. 1993.

Hand, D. J. Construction and Assessment of Classification Rules. John Wiley&Sons, 1997.

Holland, J. H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, Ann Arbor, MI., 1975.

Holland, J. H.; Holyoak, K. J.; Nisbett R. E.; Thagard, P. R. Induction: Processes of Inference, Learning and Discovery, MIT Press, 1986.

Honório, L.M.; da Silva, A. M. L.; Barbosa, D. A. A Gradient-Based Artificial Immune System Applied to Optimal Power Flow Problems. *In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628*, pp. 1–12. Springer, Heidelberg, 2007.

Hoppner, F.; Klawonn, F.; Kruse, R.; Runkler, T. Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition, WileyBlackwell, 1999.

Hüllermeier, E. Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets Syst., vol. 156*, pp. 387–407, 2005.

Hunt , J. E.; Cooke, D. E. Learning Using an Artificial Immune System. *Journal of Network and Computer Applications, vol. 19*, pp. 189-212, 1996.

Ishida, Y. The Immune System as a Self-Identification Process: A Survey and a Proposal, in Proc. of the IMBS'96, 1996.

Janikow, C. Z. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning, 13*, pp.189-228, 1993.

Janikow, C. Z. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, 28(1),* pp. 1-14, 1998.

Kuok, C.; Ada, F; Man, W.. Mining Fuzzy Association Rules. Department of Computer Science and Engineering, 1998. Disponível em: http://www.acm.org.

Lee, K. C.; Park, S. J. A knowledge-based fuzzy decision tree classifier for time series modeling. *Fuzzy Sets and Systems, 33(1),* pp.1–18, 1989.

Li, Y.; Ha, M.; Wang, X. Principle and Design of Fuzzy Controller Based on Fuzzy Learning from Examples. *In: Proc. of the 1st Int. Conf. on Machine Learning and Cybernetics, vol. (3),* pp. 1441–1446, 2002.

Liu, J.J.; Kwok, J.T. An Extended Genetic Rule Induction Algorithm, *in Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000),* pp. 458-463, 2000.

Lopes, H. S.; Coutinho, M. S.; Lima, W. C. An Evolutionary Approach to Simulate Cognitive Feedback Learning in Medical Domain. *In: Sanchez, E., Shibata, T., Zadeh, L.A. (eds.), Genetic Algorithms and Fuzzy Logic Systems. World Scientific*, Singapore (1997), pp.193-207, 1997.

Mendes, R. R. F.; Voznika, F. B.; Freitas, A. A.; Nievola, J. C. Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution*, in Lecture Notes in Artificial Intelligence, vol. 2168*, pp. 314-325, Springer-Verlag, 2001.

Nauck, D.; Klawonn, F.; e Kruse, R. Foundations of Neuro-Fuzzy Systems. Wiley and Sons, Chichester, UK, 1997.

Olaru, C.; Wehenkel, L. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems, 138(2),* 2003.

Pei, M.; Goodman, E. D.; Punch, W. F. Pattern discovery from data using genetic algorithms. Proc. 1st Pacific-Asia Conf. Knowledge Discovery & Data Mining, 1997.

Pei, Z. A Formalism to Extract Fuzzy If-Then Rules from Numerical Data Using Genetic Algorithms. *In: Int. Symposiumon EvolvingFuzzy Systems*, pp. 143–147, 2006.

Peng, Y. Soft discretization to enhance the continuous decision tree induction. *In: European Conference on Machine Learning,* Albert Ludwigs University – Alemanha, 2001.

Peter, F.; Abonyi, J. Association Rule and Decision Tree Based Methods for Fuzzy Rule Base Generation. *World Academy of Science, Engineering and Technology 13*, 2006.

Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos, CA, 1993.

Quinlan, J. R. Discovering rules by induction from large collections of examples. *In D. Michie, editor, Expert Systems in the Micro Electronic Age.* Edinburgh University Press, 1979.

Rouwhorst, S.E.; Engelbrecht, A.P. Searching the Forest: using Decision Tree as Building Blocks for Evolutionary Search in Classification. *in Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000),* pp. 633-638, 2000.

Smith, R. E. Learning classifier systems. *In: T. Back, D.B. Fogel and T. Michalewicz (Eds.) Evolutionary Computation 1: Basic Algorithms and Operators*, 114-123. Institute of Physics Publishing., 2000.

Vermaas, L. L. G.; Honório, L. M.; Freire, M.; Barbosa, D. Learning Fuzzy Systems by a Co-Evolutionary Artificial-Immune-Based Algorithm. *WILF 2009*, pp. 312-319, 2009.

Wang, L.; Mendel, J. M. Generating Fuzzy Rules by Learning from Examples. *IEEE Transactions on Systems, Man, and Cybernetics 22(6),* 1414–1427, 1992.

Wang, C. H.; Hong. T. P.; Tseng, S. S., Integrating fuzzy knowledge by genetic algorithms. *IEEE Trans. Evol. Comput., vol. 2*, pp. 138–149, 1998.

Weber, R. Fuzzy-ID3: a class of methods for automatic knowledge acquisition. *In IIZUKA-92, Proc. of the 2nd Intl. Conf. on Fuzzy Logic, vol. 1,* pp. 265-268, 1992.

Yen, John; Langari, Reza; Zadeh, Lotfi A. Industrial applications of fuzzy logic and intelligent systems. New York: IEEE Press, 1994.

Zadeh, L.A. Fuzzy sets, *Info. & Ctl., Vol. 8*, pp. 338-353, 1965.

Zhao, Y.; Collins, E.G.; Dunlap, D. Design of genetic fuzzy parallel parking control systems*. In: Proc. American Control Conference, vol. 5,* pp. 4107–4112, 2003.