Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Um Paradigma Baseado em Algoritmos Genéticos para o Aprendizado de Regras Fuzzy

Pablo Alberto Dalbem de Castro

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação

São Carlos
Maio de 2004

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C355pb

Castro, Pablo Alberto Dalbem de.
Um paradigma baseado em algoritmos genéticos para o aprendizado de regras Fuzzy / Pablo Alberto Dalbem de Castro. -- São Carlos : UFSCar, 2005.
84 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2004.

1. Inteligência artifical. 2. Sistemas Fuzzy. 3. Algoritmos genéticos. 4. Sistemas Fuzzy-genético. 5. Geração automática de regras Fuzzy. I. Título.

CDD: 006.3 (20ª)

Agradecimentos

À minha orientadora Profa. Dra. Heloisa de Arruda Camargo pelos ensinamentos, dedicação, competência e pela orientação segura.

Aos membros da banca examinadora, pela valiosa contribuição.

Aos professores e funcionários do Programa de Pós-Graduação em Ciência da Computação da UFSCar, pelo atendimento sempre solícito.

Aos meus familiares e amigos, pelo apoio e cooperação.

Especialmente à minha namorada, pelo incentivo e carinho nos momentos difíceis.

A todos que contribuíram de forma direta ou indireta para a realização deste trabalho.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro.

Resumo

A construção da base de conhecimento de sistemas *fuzzy* tem sido beneficiada intensamente por métodos automáticos que extraem o conhecimento necessário a partir de conjuntos de dados que representam exemplos do problema. A computação evolutiva, em particular os algoritmos genéticos, tem sido alvo de um grande número de pesquisas que tratam, usando abordagens variadas, a questão da geração automática da base de conhecimento de sistemas fuzzy como um processo de busca e otimização.

Este trabalho apresenta uma metodologia para o aprendizado de bases de regras *fuzzy* a partir de exemplos por meio de Algoritmos Genéticos usando a abordagem Pittsburgh. A metodologia é composta por duas etapas. A primeira é a geração genética da base de regras e a segunda é a otimização genética da base de regras previamente obtida, a fim de eliminar regras redundantes e desnecessárias. A primeira etapa utiliza um algoritmo genético auto-adaptativo, que altera dinamicamente os valores das taxas de cruzamento e mutação, a fim de garantir diversidade genética na população e evitar convergência prematura.

As funções de pertinência são previamente definidas pelo algoritmo de agrupamento *fuzzy* FC-Means e permanecem fixas durante todo o processo de aprendizado. O domínio da aplicação é a classificação de padrões multi-dimensionais, onde os atributos e, algumas vezes, as classes são *fuzzy*, portanto, representados por valores lingüísticos.

O desempenho da metodologia proposta é avaliado por simulações computacionais em alguns problemas de classificação do mundo real. Os testes focaram a acuidade das bases de regras geradas em diferentes situações. A alteração dinâmica dos parâmetros do algoritmo mostrou que melhores resultados podem ser obtidos e o uso da condição de *"don't care"* permitiu gerar um reduzido número de regras mais compreensíveis e compactas.

Palavras-Chave: Sistemas Fuzzy, Algoritmos Genéticos, Algoritmos Genéticos Auto-Adaptativos, Sistemas Fuzzy-Genéticos, Geração Automática de Regras Fuzzy.

Abstract

The construction of the knowledge base of fuzzy systems has been beneficited intensively from automatic methods that extract the necessary knowledge from data sets which represent examples of the problem. The evolutionary computation, especially genetic algorithms, has been the focus of a great number of researches that deal with the problem of automatic generation of knowledge bases as search and optimization processes using different approaches.

This work presents a methodology to learn fuzzy rule bases from examples by means of Genetic Algorithms using the Pittsburgh approach. The methodology is composed of 2 stages. The first one is the genetic learning of rule base and the other one is the genetic optimization of the rule base previously obtained in order to exclude redundant and unnecessary rules. The first stage uses a Self Adaptive Genetic Algorithm, that changes dynamically the crossover and mutation rates ensuring genetic diversity and avoiding the premature convergence.

The membership functions are defined previously by the fuzzy clustering algorithm FC-Means and remain fixed during all learning process. The application domain is multidimensional pattern classification, where the attributes and, sometimes, the class are fuzzy, so they are represented by linguistic values.

The proposed methodology performance is evaluated by computational simulations on some real-world pattern classification problems. The tests focused the accuracy of generated fuzzy rules in different situations. The dynamic change of algorithm parameters showed that better results can be obtained and the use of "don't care" conditions allowed to generate a small number of comprehensible and compact rules.

Keywords: Fuzzy Systems, Genetic Algorithms, Self-Adaptive Genetic Algorithms, Genetic Fuzzy Systems, Automatic Generation of Fuzzy Rules.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo

1 *Introdução*

1.1 Contexto

Os sistemas *fuzzy* englobam uma metodologia fundamental para representar e processar informação lingüística, com mecanismos para tratar a imprecisão e a incerteza inerentes ao conhecimento do mundo real, numa tentativa de viabilizar a modelagem de sistemas complexos. Por isso têm sido aplicados com sucesso em diversas áreas, como classificação, modelagem e controle, entre outras [Cordón et al., 2001a]. Os sistemas *fuzzy* utilizam, geralmente, regras Se-Então para representar o conhecimento do especialista. As regras são uma maneira conveniente e popular para expressar o conhecimento, proporcionando transparência e compreensibilidade ao sistema [Berg et al., 2002].

Uma das tarefas mais importantes e difíceis no desenvolvimento de um sistema *fuzzy* é gerar a sua base de conhecimento, a qual é composta pela base de dados (os conjuntos *fuzzy* e suas funções de pertinência) e pela base de regras *fuzzy* [Cordón & Herrera, 2001]. A técnica mais comum para realização desta tarefa, e que foi utilizada por muito tempo, consistia em extrair o conhecimento diretamente de um especialista no domínio do problema. Entretanto, este processo não é simples. A falta de experiência do engenheiro de conhecimento em tal domínio, a não familiarização com a teoria dos conjuntos *fuzzy* ou a dificuldade do especialista em expressar seu conhecimento podem gerar uma base de conhecimento incompleta ou com desempenho aquém do desejado. Para evitar este inconveniente, nos últimos anos têm sido utilizadas técnicas automáticas para gerá-la, que consistem em extrair automaticamente o conhecimento necessário

a partir de dados numéricos que representam amostras ou exemplos do problema. Dentre as técnicas mais bem sucedidas estão os algoritmos de agrupamento (clustering) [Liao et al.,], os métodos baseados em gradiente [Nomura et al., 1992], as Redes Neurais [Nauck & Cruse, 1997] e os Algoritmos Genéticos [Cordón et al., 2001b].

Os AGs estão sendo utilizados com sucesso sob diferentes enfoques no que diz respeito à geração da base de conhecimento de sistemas *fuzzy*. Em [Yuan & Zhuang, 1996], [González & Pérez, 1999] e [Ishibuchi et al., 1999b] é usado um AG para gerar somente a base de regras *fuzzy* para problemas de classificação de padrões. A definição genética de conjuntos *fuzzy* para problemas de controle pode ser encontrada em [Karr, 1991], [Cordón et al., 2001d] e [Cordón et al., 2001e]. A base de conhecimento completa (regras e conjuntos *fuzzy*) foi gerada usando AG em [Cordón & Herrera, 2001], [Magdalena & Velasco, 1997], [Magdalena & Velasco, 1996] e [Homaifar & McCormick, 1995]. O ajuste dos parâmetros das funções de pertinência dos conjuntos *fuzzy* é apresentado em [Herrera et al., 1995b] e [Gudwin et al., 1998]. A otimização da base de regras, a fim de eliminar regras redundantes e desnecessárias, pode ser encontrada em [Ishibuchi et al., 1997]. Há ainda trabalhos que geram um sistema *fuzzy* completo, incluindo a base de regras, a base de dados e operadores que definem o mecanismo de inferência. Por exemplo, em [Delgado, 2002] e [Delgado et al., 2001] é utilizado um processo evolutivo hierarquicamente estruturado, onde um algoritmo genético trabalha com diferentes populações representando diferentes componentes do sistema.

Um estudo comparativo entre dois diferentes enfoques pode ser encontrado em [Camargo et al., 2004]. Os autores investigaram e compararam o desempenho das bases de conhecimento obtidas em duas situações. Na primeira, um algoritmo genético gera a base de regras tendo os conjuntos *fuzzy* previamente definidos; na outra, o algoritmo genético gera os conjuntos *fuzzy* e as regras são obtidas por um método *"ad-hoc"*.

Quando o enfoque é na geração da base de regras por AG, pode-se distingüir 3 diferentes tipos de abordagens, de acordo com [Cordón et al., 2001a]: abordagem Pittsburgh, abordagem Michigan e abordagem Iterativa. A diferença principal entre elas é a forma de representação das possíveis soluções. Na abordagem Pittsburgh, cada cromossomo codifica uma base de regras

inteira, enquanto que na Michigan cada cromossomo representa apenas uma regra. A Iterativa é uma abordagem que também codifica apenas uma regra em cada cromossomo, porém ela é acompanhada de um processo iterativo para gerar e adicionar novas regras na base. As 3 abordagens serão detalhadas no Capítulo 4.

Qualquer uma das tarefas previamente citadas pode ser considerada como um processo de busca no espaço das possíveis soluções. Por isso a combinação *fuzzy* + AG tem grande aceitação na comunidade científica, uma vez que estes algoritmos são robustos e possuem a capacidade de percorrer de forma global espaços de busca irregulares e extensos, encontrando uma solução ótima ou quase ótima rapidamente [Yuan & Zhuang, 1996]. Um sistema *fuzzy* que possui algum componente da base de conhecimento gerado ou aperfeiçoado por algoritmos genéticos recebe o nome de ***Sistema Fuzzy-Genético*** [Cordón et al., 2001b].

Atualmente, entre os inúmeros trabalhos encontrados na literatura abordando geração da base de conhecimento de sistemas *fuzzy* usando AG, são reportados resultados equiparáveis ou melhores aos obtidos por outros métodos já consagrados [Hoffman et al., 2002], [Ishibuchi et al., 1999b].

1.2 Motivação

A partir da década de 80 houve um crescimento no número de pesquisas sobre aplicação de AG para gerar bases de regras *fuzzy* a partir de conjuntos de exemplos do problema e tendo os conjuntos *fuzzy*, bem como suas funções de pertinência, definidos previamente.

Segundo [Carse et al., 2003], entre os trabalhos encontrados na literatura sobre geração de regras *fuzzy* por AGs, a maioria tira conclusões usando apenas 1 ou dois conjuntos de dados, observam o comportamento das regras geradas frente a uma única forma de raciocínio e somente frente à uma específica partição *fuzzy*.

Além disso, alguns trabalhos estão preocupados somente com a acuidade da base de regras obtidas, deixando de lado sua interpretabilidade. Uma base de regras é tida como interpretável se ela pode ser facilmente compreendida por um operador humano. Para isso, ela deve conter

um número reduzido de regras, as quais devem possuir poucas variáveis na parte antecedente.

1.3 Objetivo

O objetivo deste trabalho é investigar a geração automática de regras *fuzzy* a partir de dados numéricos que representam exemplos ou amostras do problema. Para tanto, foi criada uma metodologia genética baseada na abordagem Pittsburgh composta por 2 etapas:

1. ***Geração da Base de Regras*** - Nesta etapa um AG é usado para gerar regras *fuzzy* capazes de representar o conhecimento existente no conjunto de exemplos.

2. ***Otimização da Base de Regras*** - Esta etapa visa excluir, usando também AG, as regras redundantes e desnecessárias que por ventura foram geradas na etapa anterior, resultando numa base de regras compacta.

A metodologia utilizada se baseia na existência de uma base de dados previamente definida que contenha os conjuntos *fuzzy* associados a cada variável do problema, bem como suas funções de pertinência que os definem semanticamente.

A classe de problemas abordada é a de classificação de padrões multi-dimensionais. É necessário que exista um conjunto de dados numéricos que representam amostras do problema na forma *atributos-classe*, a partir dos quais o AG consegue extrair as regras *fuzzy*.

Na etapa 1, geração da base de regras, é utilizado um algoritmo genético auto-adaptativo, que altera em tempo de execução os valores das taxas de mutação e de cruzamento, a fim de permitir uma melhor exploração do espaço de busca.

Para avaliar o desempenho da metodologia apresentada sob diversas condições, diferentes testes serão realizados utilizando alguns conjuntos de dados.

Espera-se que a metodologia proposta seja capaz de gerar uma base de regras compacta, que tenha alto poder de classificação e que seja facilmente compreendida por um ser humano (interpretável).

1.4 Organização deste Trabalho

Esta dissertação está dividida em capítulos, organizados da seguinte forma:

No Capítulo 2 são apresentados os conceitos básicos de algoritmos genéticos, sua origem, suas vantagens, funcionamento e componentes.

No Capítulo 3 são introduzidos os fundamentos dos conjuntos *fuzzy*, conceito e exemplo de variáveis lingüísticas, definição de sistemas *fuzzy* e, por último, são apresentados os sistemas *fuzzy* para classificação de padrões, bem como o formato das regras e duas formas de raciocínio utilizada por eles.

No Capítulo 4 é descrita a proposta de uma metodologia genética para geração de regras *fuzzy* a partir de conjuntos de exemplos.

No Capítulo 5 são reportados alguns experimentos que foram realizados para avaliar a metodologia. Algumas discussões sobre os resultados obtidos são feitas também neste capítulo.

Por fim, no Capítulo 6 são apresentadas as conclusões sobre o trabalho realizado e possíveis trabalhos futuros.

No Apêndice A está descrito o método de Wang e Mendel (WM) para geração de regras *fuzzy* a partir de exemplos, o qual foi comparado com a metodologia proposta.

No Apêndice B é apresentado o algoritmo de agrupamento (*clusterização*) FC-Means. Este algoritmo foi utilizado para gerar as funções de pertinência das variáveis lingüísticas.

Capítulo

2 *Algoritmos Genéticos*

2.1 Considerações Iniciais

Em 1859, Charles Darwin apresentou o conceito de *Seleção Natural*, princípio segundo o qual os indivíduos mais adaptados ao meio apresentam maior possibilidade de sobreviver e gerar descendentes. Este princípio é resultado da observação de que as mais diferentes formas de vida são suscetíveis a adaptação, que ocorre por meio de lentas transformações genéticas, conforme os indivíduos evoluem. O processo de evolução ocorre pormeio de ciclos de gerações fixas. Cada indivíduo nasce, cresce, normalmente gera um ou mais filhos e morre.

A *Computação Evolutiva* (CE) é uma área da ciência da computação que abrange modelos computacionais inspirados na Teoria da Evolução das Espécies, essencialmente no conceito de Seleção Natural, para solução de problemas nas mais diversas áreas do conhecimento. Atualmente, os modelos mais conhecidos e usados na CE são: Estratégias Evolutivas [Rechenberg, 1973] [Bäck & Schwefel, 1995] , Programação Evolutiva [Fogel et al., 1966] [Fogel, 1991] e Algoritmos Genéticos [Holland, 1975] [Goldberg, 1989].

Estes modelos mantêm uma população de indivíduos que representam possíveis soluções para o problema e cada solução é avaliada por uma função que determina quão boa ela é para o problema. Uma nova população será formada com os melhores indivíduos, dentre os quais alguns serão modificados a fim de gerar novas e, possivelmente, melhores soluções para o problema. Embora estes modelos compartilhem de uma base conceitual comum, eles apresentam certas particularidades quanto à representação das possíveis soluções e mecanismos

de evolução, conforme apresentado em [Hoffmeister & Bäck, 1991] e [Bäck et al., 1993].

Neste capítulo serão apresentados os conceitos básicos do modelo computacional mais conhecido e utilizado da Computação Evolutiva: os Algoritmos Genéticos. Na seção 2.2 são apresentados o conceito, a origem e vantagens dos Algoritmos Genéticos. A seção 2.3 mostra o funcionamento dos Algoritmos Genéticos e na seção 2.4 são expostas as operações genéticas, fundamentais para a simulação do processo evolutivo. Por fim, a seção 2.5 faz uma breve introdução aos algoritmos genéticos auto-adaptativos.

2.2 Algoritmos Genéticos

Algoritmos Genéticos (AG) são algoritmos de busca e otimização que se baseiam nos princípios da genética e da seleção natural para encontrar a solução de um determinado problema [Goldberg, 1989].

Os AGs foram criados na década de 60 por John Holland e outros pesquisadores da Universidade de Michigan. Com base nas pesquisas realizadas, Holland publica em 1975 o livro *Adaptation in Natural and Artificial Systems* [Holland, 1975], primeiro trabalho sobre o assunto. Desde então, estes algoritmos vêm sendo estudados e aplicados com sucesso na busca de soluções ótimas ou aproximadamente ótimas nas mais diversas áreas [Mitchell, 1996]: em problemas de otimização numérica e combinatória, na área de economia para criar estratégias do mercado econômico, em programação automática para desenvolver outras estruturas computacionais, em aprendizado de máquina para geração automática da base de conhecimento de sistemas *fuzzy* e outras. Os AGs diferem dos métodos de busca e otimização tradicionais, como Hill Climbing e Simulated Annealing, nos seguintes aspectos [Goldberg, 1989]:

- Os AG trabalham com a codificação dos dados e não com os próprios dados.
- AGs trabalham com uma população de soluções e não apenas com uma solução.
- Não requerem informações auxiliares. Alguns métodos utilizam derivadas para calcular os pontos mais promissores do espaço de busca. Os AGs usam apenas o valor da função objetivo (no paradigma genético, ela é chamada de função de aptidão).

- AGs usam regras de transição probabilísticas para guiar suas buscas e não regras determinísticas.

Essas quatro características tornam o AG um método de busca robusto, adequado a problemas em que o espaço de busca é irregular, extenso e desconhecido.

2.3 Funcionamento dos Algoritmos Genéticos

A aplicação de um Algoritmo Genético na resolução de um problema específico requer a definição apropriada dos seguintes elementos: codificação, população inicial, função de aptidão, operadores genéticos e critério de parada.

Os AGs iniciam o processo de busca pela melhor solução a partir de um conjunto de soluções potenciais. Essas possíveis soluções do problema são codificadas como uma seqüência finita de caracteres de um determinado alfabeto e recebem o nome de *cromossomos* ou *indivíduos*.

Em termos biológicos, as características de um indivíduo (cor dos olhos, altura, cor do cabelo, etc) são referenciados como **fenótipo** e a codificação genética dessas características são chamadas de **genótipo**. Com os algoritmos genéticos é semelhante. O fenótipo é a solução de um determinado problema e o genótipo é a codificação dessa solução.

Usualmente a codificação das soluções é feita usando números binários ou números inteiros. A codificação binária é historicamente importante, uma vez que foi utilizada nos primeiros trabalhos de John Holland. Este tipo de representação é mais fácil de utilizar e manipular. Entretanto, se o problema necessitar uma representação que implique em cromossomos longos, a representação mais adequada é a inteira, tendo em vista que ela pode ser mais facilmente compreendida pelo ser humano e requer menos esforço computacional.

Na Figura 2.1 é ilustrado um conjunto de cromossomos com codificação inteira. Cada posição no cromossomo recebe o nome de *gene* e o conjunto de cromossomos candidatos à solução é denominado *população*.

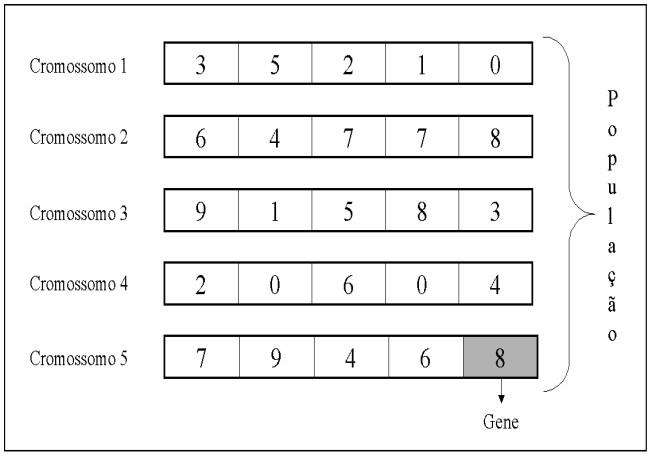

Figura 2.1. População de cromossomos

Qualquer AG básico possui a estrutura apresentada no Algoritmo 1, adaptada de [Michalewicz, 1996], em que P(t) denota a população na geração t.

Algoritmo 1 Algoritmo Genético

Início
$t \Leftarrow 0$;
Iniciar P(t);
Avaliar P(t);
Enquanto condição de parada não for satisfeita **faça**
$t \Leftarrow t + 1$;
Selecionar P(t) de P($t-1$);
Aplicar Cruzamento em P(t);
Aplicar Mutação em P(t);
Avaliar P(t);
Fim enquanto
Fim

A forma de criar a *população inicial* consiste, na maioria das vezes, em gerar aleatoriamente seqüências de caracteres do alfabeto escolhido. Porém certos problemas exigem que os cromossomos da população inicial não tenham caracteres repetidos. Dessa forma outros métodos precisam ser empregados. O tamanho da população, isto é, o número de cromossomos a ser tratado pelo AG é um fator que afeta diretamente o desempenho e a eficiência do algoritmo. Com uma população pequena o algoritmo fica mais rápido, porém com pouca cobertura no espaço de soluções. Já uma grande população pode fornecer uma cobertura mais representativa

do domínio do problema, entretanto levará mais tempo para ser processada e exigirá maior esforço computacional.

Cada indivíduo da população possui um *valor de aptidão* calculado por uma função, indicando quão boa aquela solução é para o problema. Assim como na teoria da seleção natural de Darwin, os indivíduos mais aptos terão mais chance de sobreviver e de serem escolhidos para sofrer as operações genéticas, enquanto os menos aptos "morrem". A função de aptidão depende do problema a ser tratado pelo algoritmo e, embora não exista um processo sistemático para criá-la, se ela não for bem definida, bons resultados não serão encontrados.

A população de soluções evolui através de sucessivas iterações, chamadas *gerações*, em busca de uma solução ótima ou quase ótima. Essa evolução é proporcionada pelas operações genéticas de seleção, cruzamento e mutação, as quais serão vistas na seção 2.4.

Espera-se que, atingido o *critério de parada*, o AG tenha convergido para a melhor solução. Entre os principais critérios de parada cabe destacar: número máximo de gerações e número de gerações em que não ocorre aperfeiçoamento do melhor indivíduo.

Uma aplicação comum para AG é a otimização de funções, onde o objetivo é encontrar um valor para algum parâmetro que maximize ou minimize tal função. Como exemplo, suponha que deseja-se usar um algoritmo genético para encontrar o valor máximo da função

$$y = -x^2 + 8x + 15$$

Para simplificação, será considerado que o máximo valor que x pode assumir é um número inteiro entre 0 e 31. Sendo assim, pode-se codificar as possíveis soluções com números binários na faixa [0..31] com cromossomos de 5 bits. A função de aptidão, *f*, será a própria função a ser maximizada. Observando a Tabela 2.1, é possível ver quão aptos são os cromossomos.

É sabido que o valor máximo que a função pode retornar é para x=4. Sendo assim, os melhores cromossomos apresentados na Tabela 2.1 são 00010 e 00111. Conforme a população evolui, é provável que um cromossomo que represente a solução, 00100, seja encontrado.

Tabela 2.1. Exemplo da avaliação dos cromossomos pela função de aptidão

Cromossomo (genótipo)	Valor de x (fenótipo)	*f(x)*
00010	2	27
00111	7	22
10110	22	-293
01011	11	-18

2.4 Operações Genéticas

As operações genéticas são utilizadas para possibilitar a evolução da população gerando e alterando indivíduos, a fim de que uma melhor solução para o problema seja encontrada. De acordo com [Mitchell, 1996], as operações genéticas básicas são seleção, cruzamento e mutação. Cada uma delas será detalhada nas subseções seguintes.

2.4.1 Seleção

A **seleção** é o processo por meio do qual os cromossomos mais aptos são selecionados e copiados para a próxima população de acordo com seus respectivos valores de aptidão. É a versão artificial da seleção natural e tem início após o cálculo da aptidão de cada indivíduo. Entre os vários métodos para fazer esta seleção, os mais conhecidos serão apresentados a seguir. Uma comparação entre eles pode ser encontrada em [Bäck, 1994] e [Goldberg & Deb, 1991].

- *Roleta*: Neste método, apresentado em [Holland, 1975], a probabilidade de um cromossomo C_i ser selecionado dentre os N indivíduos da população é diretamente proporcional ao seu valor de aptidão $\mathrm{F}(C_i)$ e é dada por (2.1).

$$\mathrm{P}_{sel}(C_i) = \frac{F(C_i)}{\sum_{j=1}^{NPop} F(C_j)} \tag{2.1}$$

Este operador recebe este nome porque os cromossomos podem ser representados como segmentos consecutivos em uma roleta imaginária. Cada um deles ocupa um espaço na roleta proporcional ao seu valor de aptidão. Se o tamanho da população é *NPop*, a roleta será girada

NPop vezes. A cada vez que a roleta parar de girar, o cromossomo que o marcador estiver indicando será copiado para a próxima população. É claro que os cromossomos que possuem maior espaço na roleta terão mais chance de serem selecionados.

Na Figura 2.2 os cromossomos 10011, 10001, 11100 e 01010 estão dispostos na roleta de acordo com seu valor de aptidão e a seta representa o marcador. O cromossomo 10011 terá chance de ser selecionado mais vezes, tendo em vista que ele ocupa maior espaço na roleta.

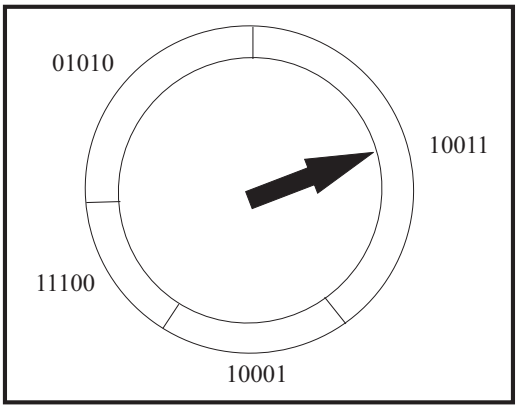

Figura 2.2. Seleção pelo método da Roleta

A seleção proporcional ao valor de aptidão pode ser problemática se os indivíduos da população apresentarem valores muito diferentes entre si. Se no início do AG, o melhor cromossomo ocupar 90% da roleta, os outros terão poucas chances de serem selecionados. Se o tamanho da população for pequeno, isso pode acarretar em perda de diversidade e pode levar à *convergência prematura*, pois a busca fica limitada a poucos pontos, causando uma perda no desempenho do AG.

- *Amostragem Universal Estocástica*: Este operador de seleção foi apresentado em [Baker, 1987] e é semelhante ao método da Roleta, porém ao invés de 1 marcador, existem *NPop* marcadores igualmente espaçados, sendo *NPop* o tamanho da população. Este tipo de seleção é mais rápido que a Roleta, pois em uma só "rodada" já são selecionados todos os indivíduos. Neste método, os cromossomos menos aptos têm mais chance de serem selecionados, garantindo diversidade na população. Na Figura 2.3, os cromossomos estão novamente dispostos na roleta, junto com os marcadores igualmente espaçados. O cromossomo 10011 será selecionado duas

vezes, os cromossomos 01010 e 11100 serão apenas uma vez e 10001 nenhuma.

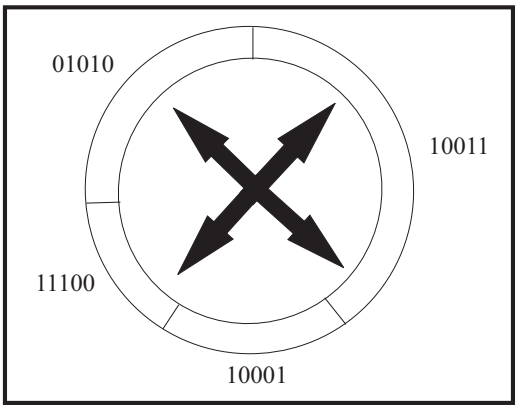

Figura 2.3. Seleção por Amostragem Universal Estocástica

- *Torneio*: Um número qualquer k, $2 \leq k \leq NPop$, de indivíduos é selecionado aleatoriamente e o que possuir melhor valor de aptidão entre eles será copiado para a próxima população. Este processo é repetido até que a nova população esteja completa.

É fato que um cromossomo que tenha melhor valor de aptidão que outro tem maior probabilidade de ser selecionado. No entanto, nada impede que seja escolhido o pior, perdendo-se assim o melhor. Para evitar que isso aconteça, pode-se adicionar a qualquer um dos métodos de seleção descritos uma técnica chamada ***Elitismo*** [De Jong, 1975], que consiste em substituir aleatoriamente n indivíduos da população atual pelos n melhores da população anterior.

2.4.2 Cruzamento

A operação de **cruzamento** consiste em escolher dois indivíduos, denominados cromossomos pais, para cruzar e gerar filhos que os substituirão na população. A idéia central do cruzamento é a combinação de características dos indivíduos mais aptos por meio de troca de informações entre eles. Uma taxa de cruzamento precisa ser especificada para definir com que probabilidade (P_c) os cromossomos serão cruzados. Se esta taxa for muito alta, mais rapidamente novos indivíduos serão introduzidos na população, porém a perda de indivíduos mais aptos pode

ocorrer. Se a taxa for muito pequena, o algoritmo pode se tornar lento. O ideal seria que este valor não permanecesse fixo durante todo o processo de busca e que fosse modificado dinamicamente no decorrer da evolução, possibilitando uma melhor exploração do espaço de busca. A seção 2.5 introduz os Algoritmos Genéticos Auto-Adaptativos, os quais possuem a capacidade de alterar dinamicamente os valores de alguns parâmetros utilizados no algoritmo, entre eles o da taxa de cruzamento.

Na literatura são encontrados vários operadores de cruzamento. A seguir, serão apresentados os mais conhecidos:

- *Cruzamento de um ponto ou padrão*: Um ponto de cruzamento é escolhido aleatoriamente dentro dos cromossomos pais dividindo cada um em dois segmentos. O segundo segmento de um cromossomo, isto é, aquele após o ponto de cruzamento será trocado com o segundo segmento do outro cromossomo, criando dois novos indivíduos. No exemplo mostrado na Figura 2.4, os cromossomos 11101 e 01110 são escolhidos para o cruzamento e serão intercambiados logo após o segundo gene. Os filhos gerados são 11110 e 01101.

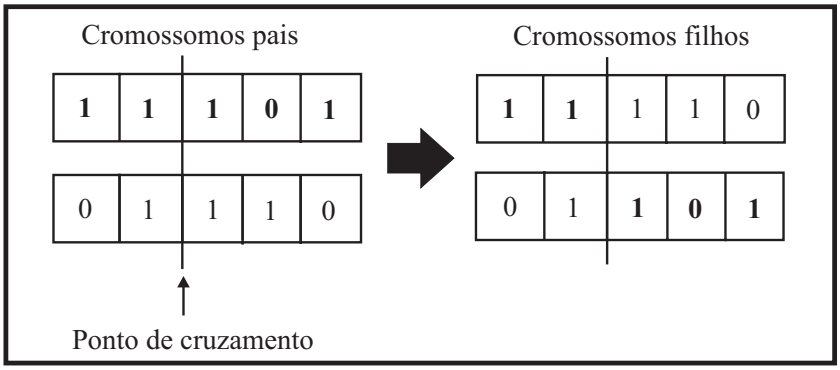

Figura 2.4. Cruzamento de um ponto

Uma variação deste operador é o cruzamento multiponto, que utiliza n pontos de cruzamento e divide os cromossomos em $n+1$ segmentos. Dessa forma, é permitido que mais de um segmento seja trocado entre os pais.

Esses operadores de cruzamento foram concebidos para serem aplicados em cromossomos com codificação binária, mas nada impede de serem usados também em outros tipos de codificações.

- *Cruzamento max-min aritmético*: Este operador de cruzamento, utilizado em [Herrera et al., 1995b], é aplicado a cromossomos com codificação real. Se $C_v = \{c_{v_1}, ..., c_{v_n}\}$ e $C_w = \{c_{w_1}, ..., c_{w_n}\}$ são dois cromossomos que serão cruzados, então os filhos gerados são os cromossomos C_1, C_2, C_3 e C_4, tal que seus genes são calculados por:

$$c_{1_i} = a * c_{w_i} + (1 - a) * c_{v_i}$$
$$c_{2_i} = a * c_{v_i} + (1 - a) * c_{w_i}$$
$$c_{3_i} = min\{c_{v_i}, c_{w_i}\}$$
$$c_{4_i} = max\{c_{v_i}, c_{w_i}\}$$

O parâmetro $a \in [0, 1]$ pode ser uma constante ou pode variar conforme a geração em que a população se encontra e i = 1,...,n. Desses quatro novos indivíduos gerados, somente os dois melhores farão parte da população. Este operador de cruzamento é mostrado na Figura 2.5, onde o valor do parâmetro a foi estipulado em 0.35.

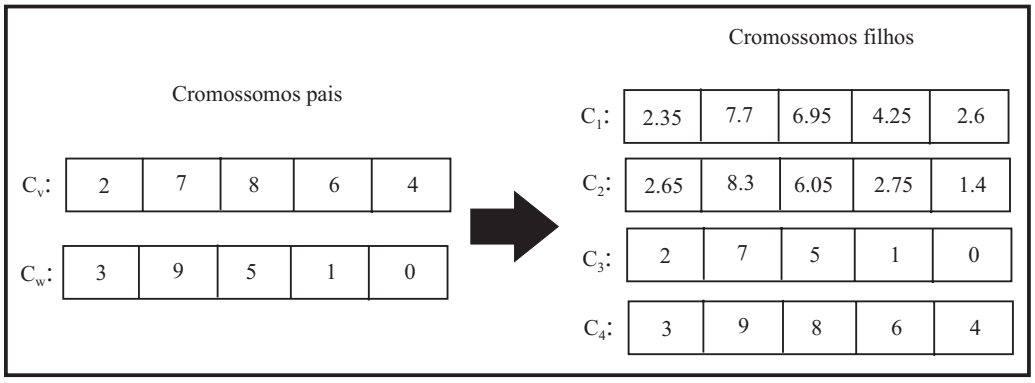

Figura 2.5. Cruzamento max-min aritmético

2.4.3 Mutação

A **mutação** é uma operação que tem por finalidade introduzir e manter a diversidade genética na população, escolhendo aleatoriamente um gene e trocando seu valor. Uma taxa de mutação é definida para indicar com que probabilidade (P_m) os genes serão alterados. Um número muito alto pode tornar o algoritmo desprovido de uma direção em sua pesquisa pelo espaço de busca. Já uma taxa de mutação muito baixa pode deixar o processo de busca lento.

Assim como a taxa de cruzamento, a taxa de mutação deveria ser alterada dinamicamente durante a execução do algoritmo, a fim de explorar de uma melhor forma o espaço de busca.

Dentre os operadores de mutação, os dois mais conhecidos e utilizados são:

- *Mutação padrão*: Consiste em alterar o valor do gene escolhido para participar da mutação. Isso significa que se estiver usando codificação binária, basta trocar 1 por 0 (zero) e vice-versa. Quando a codificação for real, substitui-se o valor do gene por um dos possíveis valores que ele pode assumir. Na Figura 2.6 é ilustrada esta operação. O último gene do cromossomo 11101 sofrerá mutação e o cromossomo gerado será 11100.

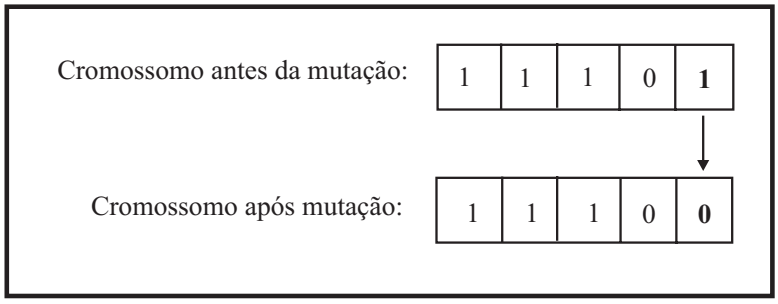

Figura 2.6. Mutação padrão

- *Mutação não uniforme*: Este operador de mutação, proposto e apresentado em [Michalewicz, 1996], é mais utilizado em cromossomos com codificação real. Se C=$(c_1, ..., c_k, ..., c_n)$ é um cromossomo e o elemento c_k foi selecionado para sofrer mutação, então o cromossomo resultante será C'=$(c_1, ..., c_{k'}, ...c_n)$, onde:

$$c_{k'} = \begin{cases} c_k + \Delta(t, LS(c_k) - c_k) & \text{se } a{=}0 \\ c_k - \Delta(t, c_k - LI(c_k)) & \text{se } a{=}1 \end{cases} \tag{2.2}$$

O parâmetro a é um dígito binário gerado aleatoriamente, LS e LI são o limite superior e limite inferior, respectivamente, na escala de possíveis valores que c_k pode assumir.

A função $\Delta(t, y)$ é uma função do tipo:

$$\Delta(t, y) = \text{y}(1\text{-}r^{(1-t/T)^b}) \tag{2.3}$$

em que t é a geração atual, T é o número máximo de gerações, r é um valor no intervalo [0,1] e b, um parâmetro que controla o grau de não uniformidade do operador.

Esta função retorna um valor qualquer no intervalo *[t,y]*. A probabilidade desse valor ser próximo a 0 (zero) aumenta conforme aumenta o número de gerações, t.

O operador de mutação não uniforme é mostrado na Figura 2.7, onde o terceiro gene do cromossomo será alterado. No exemplo, é assumido que a escala de valores que este gene pode assumir é [1, 10], o valor de a é 1, b vale 2, t é igual a 10, T vale 1000, r vale 0.5 e o valor retornado pela função $\Delta(t, y)$ é 3.1555.

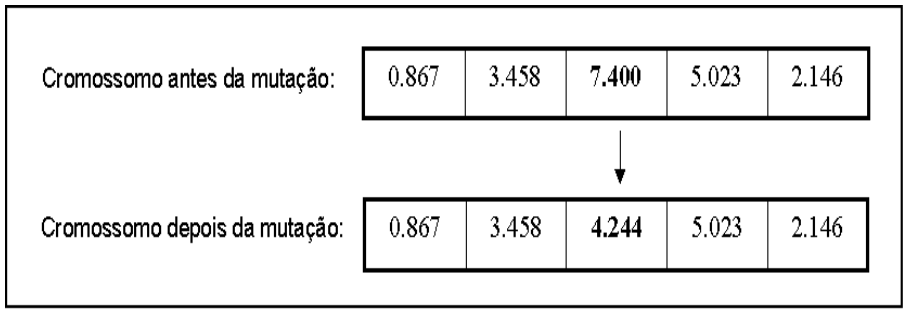

Figura 2.7. Mutação não uniforme

Pode-se ver que na mutação padrão o novo valor do gene independe do valor atual e qualquer valor no intervalo especificado é aceitável. Já a mutação não-uniforme, ao contrário, vai acumulando uma "história" ao longo do processo evolutivo, já que o novo valor depende sempre do valor atual e a perturbação adicionada é definida aqui como sendo inversamente proporcional ao número de gerações já transcorridas.

2.5 Algoritmos Genéticos Auto-Adaptativos

O uso de algoritmos genéticos requer a escolha de valores para um conjunto de parâmetros, como tamanho da população, taxa de cruzamento e taxa de mutação. Estes valores exercem grande influência na capacidade do algoritmo de explorar eficientemente o espaço de busca. O uso de parâmetros com valores inadequados pode causar vários efeitos indesejáveis, dentre os quais se destaca um problema conhecido como *convergência prematura*, que é uma estagnação prematura da busca causada pela perda de diversidade populacional.

Um grande número de experimentos para descobrir valores ótimos para tais parâmetros são reportados em [Grefenstette, 1986]. Entretanto, apesar da exaustiva quantidade de testes, os valores ideais encontrados para estes parâmetros permanecem fixos durante todo o processo de busca e estão dentro de uma intervalo relativamente grande que, dependendo do valor escolhido, podem levar a uma significativa diferença no desempenho do algoritmo. Além disso, alguns valores podem ser eficientes para um problema específico, mas não para outro.

Uma vez que algoritmos genéticos implementam a idéia de evolução, é mais que natural esperar algum comportamento *auto-adaptativo* desses algoritmos. O ajuste em tempo de execução dos valores de tais parâmetros pode ser uma maneira de melhorar a forma com que o algoritmo explora o espaço de busca. São estabelecidos valores iniciais para estes parâmetros e durante o curso de evolução estes valores são ajustados com base no atual estado da população, em termos de diversidade genética. Algumas formas de calcular a diversidade genética de uma população de cromossomos são apresentadas na subseção 2.5.1.

Percebe-se a necessidade da variação dinâmica de parâmetros analisando o comportamento interno de um algoritmo genético. Nas primeiras gerações é possível que apareçam indivíduos com valores de aptidão muito superiores aos outros e, por apresentarem tal superioridade, têm mais chances de gerar filhos levando a população a se aglutinar em torno deles. Isto causaria uma convergência prematura provocada pela perda de diversidade genética na população. Logo, é recomendável que a manutenção da diversidade seja preservada e, para isso, o algoritmo poderia, por exemplo, aumentar a taxa de mutação, operação genética responsável por inserir novos indivíduos na população. Dessa forma, no decorrer das gerações, o espaço de busca vai ficando cada vez mais diversificado e os melhores pontos deste espaço vão sendo registrados nos cromossomos. Uma vez que o espaço de busca esteja bem variado, os fatores que mantinham a diversidade genética devem cair, evitando que o processo de busca fique aleatório e sem direção. Portanto, a taxa de mutação deve ser reduzida e a de cruzamento aumentada, para explorar os melhores pontos encontrados.

Os AGs que possuem a capacidade de ajustar os valores de seus parâmetros dinamicamente são chamados de *Algoritmos Genéticos Auto-Adaptativos*. Nos trabalhos encontrados na

literatura sobre ajuste dinâmico de valores de parâmetros, o motivo para tal ajuste é quase sempre o mesmo: evitar a convergência prematura, melhorando a forma com que o algoritmo explora o espaço de busca. Em [Baker, 1985] foi utilizado um algoritmo genético que altera dinamicamente o tamanho da população. Já em [Fogarty, 1989] foi usado um algoritmo que varia a taxa de mutação e em [Booker, 1987] para alterar a taxa de cruzamento. Por fim, a alteração dinâmica das taxas de cruzamento e mutação pode ser encontrada em [Sirinivas & Patnaik, 1994].

Além de encontrar trabalhos que usam AG para alterar dinamicamente os valores de alguns parâmetros, é possível encontrar também trabalhos em que o AG escolhe em tempo de execução os operadores genéticos que serão usados [Herrera et al., 1996] e trabalhos que usam AG que modificam a codificação das soluções, alterando de binária para real e vice versa [Schraudolph & Belew, 1992].

2.5.1 Formas de Calcular a Diversidade Genética

Na seção 2.5 foi citado que os valores dos parâmetros eram alterados de acordo com o estado atual da população, em termos de diversidade genética. Mas como calcular a diversidade genética da população? De acordo com [Herrera & Lozano, 1996], existem dois tipos de medidas de diversidade genética que descrevem o estado da população: **Medidas de Diversidade Genotípicas** (MDG) e **Medidas de Diversidade Fenotípicas** (MDF).

As MDG se concentram na variação do material genético existente nos cromossomos da população para medir a diversidade genética. Em [Whitley et al., 1991] e [Louis & Rawlins, 1993] foram utilizadas medidas de distância de Hamming entre todos os cromossomos da população, enquanto que a distância euclideana foi usada em [Herrera & Lozano, 1996] e medida de entropia em [Grefenstette, 1987].

Já as MDF atentam para os valores de aptidão dos cromossomos. Muitos dos trabalhos na literatura obtêm o índice de diversidade genética calculando a razão entre o melhor e o pior valor de aptidão encontrados na população, como por exemplo em [Sirinivas & Patnaik, 1994] ou a razão entre o valor de aptidão médio e o melhor valor, como em [Lee & Takagi, 1993].

2.6 Considerações Finais

Este capítulo teve por finalidade apresentar os conceitos e fundamentos básicos dos Algoritmos Genéticos, suficientes para entender seu funcionamento no contexto deste trabalho. O capítulo seguinte faz uma breve introdução à teoria dos conjuntos *fuzzy* e aos sistemas *fuzzy*, em particular os sistemas de classificação *fuzzy*.

Capítulo

3 *Sistemas Fuzzy*

3.1 Considerações Iniciais

Os conjuntos *fuzzy* e a lógica *fuzzy*, introduzidos por Lotfi A. Zadeh em 1965 [Zadeh, 1965], provêm a base para geração de técnicas apropriadas para um adequado processamento de informações incertas, inerentes ao mundo real.

Por exemplo, dada a temperatura de um ambiente, é preciso saber por um motivo qualquer se essa temperatura é quente ou não. Supondo que esse conceito será expresso em intervalos binários, é suposto que a temperatura é considerada quente se estiver acima de 40ºC. Entretanto, se a temperatura for 39ºC, por diferença de 1ºC, bruscamente a temperatura deixa de ser quente? A teoria *fuzzy* atenta para este inconveniente expressando transições graduais de significado por meio de *graus de verdade* que assumem valores no intervalo [0,1]. Dessa forma, no exemplo anterior, a sentença "40ºC é quente" possui um grau de verdade maior que o da sentença "39ºC é quente". Quando a sentença possui grau de verdade 1, significa que ela é completamente verdadeira e quando possui grau 0 significa que ela é completamente falsa.

Estudos recentes apontam para uma grande variedade de aplicações da teoria *fuzzy*. A mais importante se refere aos sistemas *fuzzy* baseados em regras [Casillas et al., 2000]. A utilização de sistemas *fuzzy* representa um marco na tecnologia da informação, tendo em vista que eles possuem habilidade para expressar e tratar a subjetividade presente no raciocínio humano [Pedrycz & Gomide, 1998]. Estes tipos de sistemas estão sendo aplicados com sucesso a problemas de controle, classificação e tomada de decisão nas mais diversas áreas: economia,

engenharia, medicina, meteorologia e manufatura, entre outras.

A finalidade deste capítulo é apresentar alguns conceitos fundamentais para o entendimento e utilização dos sistemas *fuzzy* no contexto deste trabalho. Na seção 3.2 são apresentados os conceitos básicos dos conjuntos *fuzzy* como funções de pertinência e algumas operações básicas que podem ser realizadas com estes conjuntos. A seção 3.4 define o que são variáveis lingüísticas. Na seção 3.5 são conceituados os sistemas *fuzzy* e na seção 3.6 os sistemas de classificação *fuzzy*.

3.2 Conjuntos Fuzzy

A noção de conjunto é básica não só na matemática mas também na vida diária. Ela ocorre freqüentemente quando tenta-se organizar, sumarizar e generalizar o conhecimento sobre objetos. Seja X um conjunto de objetos, denominado *universo de discurso*, e x um elemento de X. Um conjunto A, $A \subseteq X$, é definido como uma coleção de objetos $x \in X$.

Na teoria clássica dos conjuntos, chamados de conjuntos *crisp*, um elemento pertence ou não a um dado conjunto. Isto pode ser expresso pela função apresentada em (3.1), chamada *função característica*, que atribui valor 1 ao elemento se ele for membro do conjunto ou 0 se ele não for.

$$\mathrm{F}_A(x) = \begin{cases} 1 & \text{se x} \in \text{A} \\ 0 & \text{se x} \notin \text{A} \end{cases} \tag{3.1}$$

Entretanto, o mundo real apresenta situações em que os conjuntos clássicos não são flexíveis o suficiente para uma descrição apurada de significado, tendo em vista que eles forçam de forma abrupta a transição de pertinência absoluta para a não pertinência absoluta.

Já os conjuntos *fuzzy*, considerados uma generalização dos conjuntos clássicos, possibilitam a transição gradual de significado, permitindo que cada elemento pertença parcialmente a um determinado conjunto. A função característica de um conjunto A, chamada de *função de pertinência* na teoria dos conjuntos *fuzzy*, associa cada elemento do universo de discurso a um valor no intervalo [0,1], o qual indica o grau de pertinência do elemento ao conjunto A.

Sendo x um elemento genérico de um conjunto X, um conjunto *fuzzy* A, $A \subseteq X$, é

representado como um conjunto de pares ordenados (μ_A(x)/x), como expresso em (3.2):

$$\text{A} = \{(\mu_A(\text{x})/\text{x}) \mid x \in X\} \tag{3.2}$$

A função de pertinência μ_A é uma função do tipo $\mu_A(x) : X \rightarrow [0, 1]$ que atribui para cada elemento x ∈ X um grau de pertinência ao conjunto A no intervalo [0,1] [Klir & Yuan, 1995].

Como exemplo, considere X = {10, 20, 25, 30, 35, 40, 45, 50, 60} uma coleção de idades e A o conjunto *fuzzy* das "pessoas jovens", dado por A = {(1/10), (1/20), (0.9/25), (0.8/30), (0.6/35), (0.5/40), (0.4/45), (0/50), (0/60)}. Então, o conjunto *fuzzy* A pode ser definido como uma coleção de idades com valores de pertinência variando entre 0 (exclusão completa) e 1 (pertinência completa). Elementos com grau de pertinência 0 não precisam ser listados.

3.2.1 Tipos de Funções de Pertinência

Toda a teoria dos conjuntos *fuzzy* está baseada no conceito de pertinência. Em princípio, qualquer função da forma $\mu_A(x) : X \rightarrow [0, 1]$ descreve uma função de pertinência associada a um conjunto *fuzzy* A que depende não somente do conceito a ser representado, mas também do contexto em que é usada. As formas de funções mais conhecidas, bem como seus gráficos, são apresentados a seguir:

- Funções Triangulares: especificadas por três parâmetros **a**, **m** e **b**, sendo $\mathbf{a} \leq \mathbf{m} \leq \mathbf{b}$.

$$\mu_A(x) = \begin{cases} 0 & \text{se } x \leq a \\ \frac{x-a}{m-a} & \text{se } x \in [a, m] \\ \frac{b-x}{b-m} & \text{se } x \in [m, b] \\ 0 & \text{se } x \geq b \end{cases} \tag{3.3}$$

Na Figura 3.1 é mostrado o gráfico desta função.

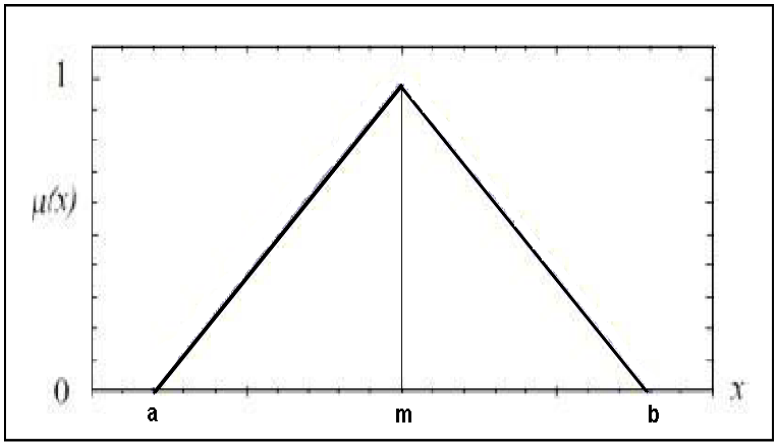

Figura 3.1. Função de pertinência triangular

- Funções Trapezoidais: especificadas por quatro parâmetros **a**, **m**, **n** e **b**, sendo $\mathbf{a} \leq \mathbf{m}, \mathbf{n} \leq \mathbf{b}$ e $\mathbf{m} < \mathbf{n}$.

$$\mu_A(x) = \begin{cases} 0 & \text{se } x < a \\ \frac{x-a}{m-a} & \text{se } x \in [a, m] \\ 1 & \text{se } x \in [m, n] \\ \frac{b-x}{b-n} & \text{se } x \in [n, b] \\ 0 & \text{se } x > b \end{cases} \tag{3.4}$$

Na Figura 3.2 está o gráfico da função trapezoidal.

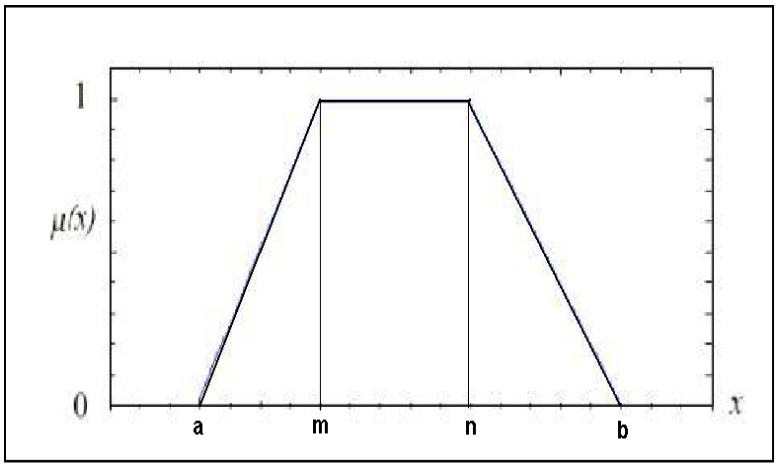

Figura 3.2. Função de pertinência trapezoidal

- Funções Gaussianas: especificadas por dois parâmetros **m** e **k**, com **k** > 0.

$$\mu_A(x) = e^{-k(x-m)^2} \tag{3.5}$$

Na Figura 3.3 está o gráfico da função gaussiana.

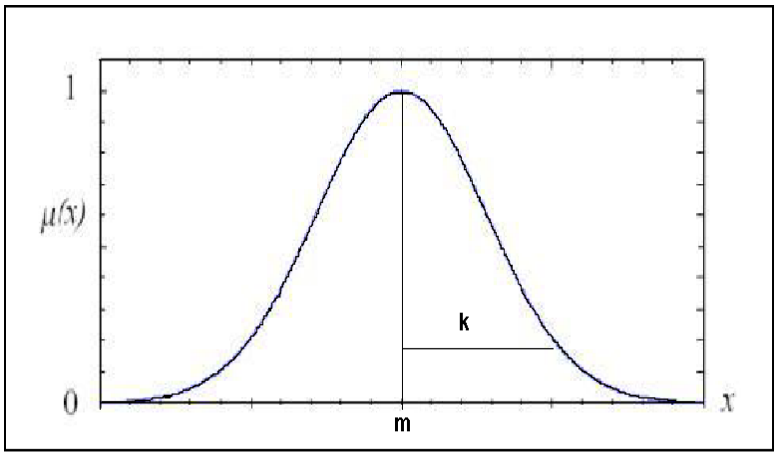

Figura 3.3. Função de pertinência gaussiana

3.3 Operações em Conjuntos *Fuzzy*

As três operações básicas em conjuntos clássicos - complemento, intersecção e união - podem ser generalizadas para os conjuntos *fuzzy*. Para cada uma das três operações, existe uma família de operadores *fuzzy* que as implementa [Klir & Yuan, 1995].

Seja A um conjunto *fuzzy* definido num universo X. Por definição, $\mu_A(x)$ é interpretado como o grau de pertinência de x ao conjunto A. O **complemento** de um conjunto *fuzzy* A é denotado por $\mu_{\overline{A}}(x)$. Logo, $\mu_{\overline{A}}(x)$ indica o grau com que x **não** pertence ao conjunto A. A operação de complemento pode ser definida por uma função $c : [0,1] \rightarrow [0,1]$ que satisfaz as seguintes propriedades:

- Monotonicidade: Para todo a,b ∈ [0,1], se a≤b, então c(a) ≥ c(b)
- Condição limite: $c(0)$=1 e $c(1)$=0
- Involução: $c(c(\mathrm{x}))$ = x para x ∈ [0,1]

A função c concerne uma classe de funções que realiza a operação complemento de um conjunto *fuzzy*. A função mais usada para a operação complemento de um determinado conjunto *fuzzy* A é mostrada em (3.6)

$$\mu_{\overline{A}}(x) = 1 \text{ - } \mu_A(x) \tag{3.6}$$

A operação de **intersecção** de dois conjuntos *fuzzy* A e B é especificada por uma função do tipo $i : [0,1] \times [0,1] \rightarrow [0,1]$ que satisfaz as seguintes propriedades:

- Comutatividade: i(x,y) = i(y,x)
- Associatividade: i(x,i(y,z)) = i(i(x,y),z)
- Monotonicidade: Se x≤y e w≤z, então i(x,w)≤i(y,z)
- Condição limite: i(0,x) = 0 e i(1,x) = x

A função i define uma classe de funções que realizam a operação de intersecção. Essa classe de funções é conhecida como *T-norma* [Klir & Yuan, 1995]. Os seguintes exemplos são as T-normas mais utilizadas para realizar a intersecção de dois conjuntos *fuzzy* A e B: A *intersecção padrão* (3.7), *produto algébrico* (3.8) e *diferença limitada* (3.9).

$$\mu_A(x) \cap \mu_B(x) = min[\mu_A(x), \mu_B(x)] \tag{3.7}$$

$$\mu_A(x) \cap \mu_B(x) = \mu_A(x).\mu_B(x) \tag{3.8}$$

$$\mu_A(x) \cap \mu_B(x) = max[0, \mu_A(x) + \mu_B(x) - 1] \tag{3.9}$$

A operação de **união** de dois conjuntos *fuzzy* A e B é especificada por uma função da forma $u : [0,1] \times [0,1] \rightarrow [0,1]$ que satisfaz as seguintes propriedades:

- Comutatividade: u(x,y) = u(y,x)
- Associatividade: u(x,u(y,z)) = u(u(x,y),z)
- Monotonicidade: Se x≤y e w≤z, então u(x,w)≤u(y,z)

- Condição limite: $u(0,x) = x$ e $u(1,x) = 1$

A função u compreende uma família de funções que realizam a operação de união. Essa classe de funções é conhecida como *S-norma* ou *T-conorma* [Klir & Yuan, 1995]. Os seguintes exemplos são as S-normas mais utilizadas para realizar a união de dois conjuntos *fuzzy* A e B: A *união padrão* (3.10), *soma algébrica* (3.11) e *soma limitada* (3.12).

$$\mu_A(x) \cup \mu_B(x) = max[\mu_A(x), \mu_B(x)] \tag{3.10}$$

$$\mu_A(x) \cup \mu_B(x) = \mu_A(x) + \mu_B(x) - \mu_A(x).\mu_B(x) \tag{3.11}$$

$$\mu_A(x) \cup \mu_B(x) = min[1, \mu_A(x) + \mu_B(x)] \tag{3.12}$$

Outros tipos de operadores para complemento, intersecção e união de conjuntos *fuzzy* podem ser encontrados em [Klir & Yuan, 1995], [Zimmermann, 1991] e [Pedrycz & Gomide, 1998].

3.4 Variáveis Lingüísticas

Grande parte da experiência e do conhecimento dos seres humanos está armazenada na forma lingüística, mais geral e imprecisa, dificultando seu aproveitamento por computadores, os quais exigem informações precisas. Com o conceito de variáveis lingüísticas, problemas naturalmente imprecisos e complexos passam a ser manipuláveis por computadores. A tradução da informação lingüística (imprecisa) para informação numérica (precisa) utilizada por computadores pode ser realizada usando os conjuntos *fuzzy*.

Variáveis lingüísticas são variáveis cujos valores são palavras ou sentenças em linguagem natural em vez de números [Zimmermann, 1991]. Elas são definidas sobre uma variável numérica que possui um determinado intervalo, o qual é granularizado em termos lingüísticos definidos por conjuntos *fuzzy*. O processo de granularização de uma variável numérica em conjuntos *fuzzy* define a **partição *fuzzy*** da variável em questão.

O conjunto de termos lingüísticos que a variável lingüística pode assumir é referenciado como **domínio** da variável e é denotado por $D_{variável}$=$\{valor_1, valor_2, ...valor_n\}$.

Na Figura 3.4 é ilustrada a variável Temperatura de um ambiente que varia de 0ºC a 50ºC.

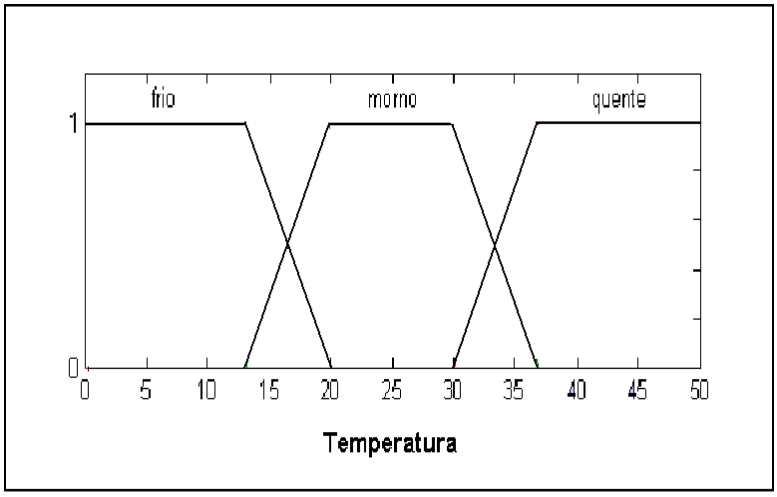

Figura 3.4. Variável lingüística Temperatura

São usados 3 termos lingüísticos para caracterizar a variável Temperatura da Figura 3.4: *frio*, *morno* e *quente*, os quais são especificados por conjuntos *fuzzy* definidos por suas respectivas funções de pertinência. No exemplo são utilizadas funções de pertinência trapezoidais. Portanto, o domínio da variável Temperatura é $D_{Temperatura}$= $\{frio, morno, quente\}$.

3.5 Sistemas Fuzzy

Quando se deseja modelar um sistema complexo, onde há necessidade de manipular conhecimento incerto e impreciso, aplica-se a teoria *fuzzy* a esses sistemas no lugar da lógica clássica, os quais passam a ser chamados de **sistemas *fuzzy***.

Um sistema *fuzzy* é qualquer sistema que incorpora a lógica *fuzzy* no processo de raciocínio e cujas variáveis do problema (ou pelo menos uma delas) podem assumir valores lingüísticos definidos por conjuntos *fuzzy* [Zadeh, 1973] [Klir & Yuan, 1995].

Este trabalho enfoca os sistemas *fuzzy* que utilizam regras *SE...ENTÃO* para representar o relacionamento entre as variáveis do problema. As regras são uma maneira conveniente e popular para expressar o conhecimento, propiciando transparência e compreensibilidade ao sistema [Berg et al., 2002]. As regras *fuzzy* possuem o seguinte formato:

SE antecedente, ENTÃO conseqüente

A parte antecedente é sempre uma proposição *fuzzy* - ou uma conjunção/disjunção destas - do tipo *"X é A"*, onde X é uma variável lingüística de entrada, A é um termo lingüístico que representa um conjunto *fuzzy* definido no universo U, o qual é caracterizado por uma função de pertinência.

Dependendo do conseqüente, pode-se distingüir 2 principais tipos de sistemas *fuzzy*: os do tipo *Mamdani* [Mamdani, 1977] e os do tipo *Takagi-Sugeno* [Takagi & Sugeno, 1985].

Os sistemas *fuzzy* do tipo Mamdani possuem proposições *fuzzy* no antecedente e no conseqüente da regra, ou seja, a parte conseqüente também é uma proposição *fuzzy*, ou uma conjunção/disjunção destas, do tipo *"Y é B"*, onde Y é uma variável lingüística de saída, B é um termo lingüístico que representa um conjunto *fuzzy* definido no universo V, o qual é caracterizado por uma função de pertinência. Um exemplo deste tipo de regra é:

SE Temperatura é quente E Pressão é alta, ENTÃO Potência é fraca

Na regra acima, *Temperatura*, *Pressão* e *Potência* são variáveis e *quente*, *alta* e *fraca* são termos lingüísticos que as variáveis podem assumir, respectivamente. A variável de saída do sistema é uma variável lingüística que possui um valor lingüístico.

Os sistemas do tipo Takagi-Sugeno possuem proposições *fuzzy* no antecedente e o conseqüente da regra é uma função aplicada sobre os valores de entrada, como é mostrado no exemplo:

SE Temperatura é quente E Pressão é alta, ENTÃO Potência é f(Temperatura,Pressão)

Da mesma forma, *Temperatura*, *Pressão* e *Potência* são variáveis e *quente* e *alta* são termos lingüísticos. A saída do sistema é um valor numérico calculado por *f(Temperatura,Pressão)*.

Tendo os valores de entrada e um conjunto de regras *fuzzy* definido, é necessário um mecanismo de inferência apropriado que os combine a fim de gerar conclusões. Algumas formas de raciocínio *fuzzy* podem ser encontradas em [Klir & Yuan, 1995] e [Pedrycz & Gomide, 1998].

Atualmente os sistemas *fuzzy* são aplicados em problemas de controle, modelagem, diagnóstico e classificação, entre outros [Cordón et al., 2001a].

O enfoque deste trabalho será dado aos sistemas de classificação *fuzzy*, os quais possuem o formato da regra e a forma de raciocínio específicos. Na seção seguinte será apresentado este tipo de sistema.

3.6 Sistema de Classificação Fuzzy

Classificação é uma importante tarefa encontrada nas áreas de reconhecimento de padrões, tomada de decisão, mineração de dados e modelagem [Berg et al., 2002]. Dado um conjunto de objetos E=$\{e_1, e_2, ..., e_L\}$, também chamados de *padrões*, o objetivo da classificação é atribuir uma classe $Classe_j$ de um conjunto de classes C=$\{Classe_1, Classe_2, ..., Classe_M\}$ a um objeto $e_p = \{a_{p_1}, a_{p_2}, ..., a_{p_n}\}$, o qual é descrito por n atributos.

Muitos métodos têm sido utilizados para a tarefa de classificação de padrões, tais como métodos estatísticos [Duda & Hart, 1973], Redes Neurais [Bishop, 1995] e Sistemas *Fuzzy* [Kecman, 2001].

Um sistema de classificação *fuzzy* possui 2 componentes principais: a Base de Conhecimento e o Mecanismo de Inferência, conforme ilustrado na Figura 3.5 [Cordón et al., 1999].

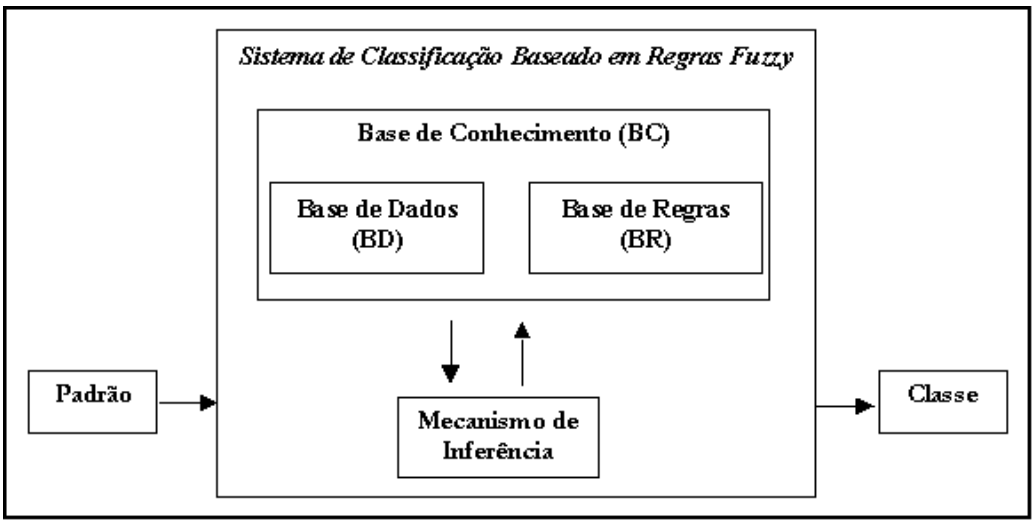

Figura 3.5. Componentes de um Sistema de Classsificação *Fuzzy*

A ***Base de Conhecimento (BC)*** é composta pela *Base de Dados (BD)*, a qual contém as definições dos conjuntos *fuzzy* relacionados aos termos lingüísticos usados nas regras *fuzzy* e pela *Base de Regras (BR)*, que armazena o conjunto de regras *fuzzy*. Uma típica regra *fuzzy* de classificação é:

$$R_k\text{: } \overbrace{\text{SE } X_1 \text{ é } A_{i_1} \text{ E } \ldots \text{ E } X_n \text{ é } A_{i_n}}^{Antecedente}, \overbrace{\text{ENTÃO } Classe_j}^{Conseqüente} \quad ,$$

sendo R_k o identificador da regra, $X_1, ..., X_n$ os atributos do padrão considerados no problema (representados aqui por variáveis lingüísticas), A_{i_1},...,A_{i_n} os valores lingüísticos usados para representar os valores de tais atributos e $Classe_j$ a classe, *fuzzy* ou não, a que pertence o padrão.

Um exemplo de uma regra *fuzzy* aplicada ao problema de classificação de plantas Íris [Blake & Merz, 1998] é como segue:

R_1: SE comprimento_sépala é Grande E largura_sépala é Pequena E comprimento_pétala é Médio E largura_pétala é Médio, ENTÃO Classe = Versicolor.

O ***Mecanismo de Inferência*** aplica o conjunto de regras *fuzzy* no padrão a ser classificado, determinando a classe a que ele pertence. A maioria dos sistemas *fuzzy* de classificação, veja por exemplo [Abe & Thawonmas, 1997], [Chi et al., 1996] e [González & Pérez, 1999], utiliza o *Método de Raciocínio Fuzzy Clássico* que classifica um padrão usando a regra que possuir maior grau de compatibilidade com esse padrão.

Considere o padrão a ser classificado $e_p = \{a_{p_1}, a_{p_2}, ..., a_{p_n}\}$ e um conjunto de regras R=$\{R_1, R_2, ..., R_S\}$. O Raciocínio *fuzzy* Clássico se dá pelas seguintes etapas:

- Calcular o grau de compatibilidade, *Compat(R_k,e_p)*, entre o padrão e_p e cada regra R_k, k=1...S, usando uma T-norma sobre o grau de pertinência dos valores dos atributos do padrão, a_{p_j}, aos correspondentes conjuntos *fuzzy* que aparecem no antecedente da regra, A_{i_j}, j=1...n.

$$Compat(R_k,e_p) = \text{T } (\mu_{A_{i_1}}(a_{p_1}), \mu_{A_{i_2}}(a_{p_2}), ..., \mu_{A_{i_n}}(a_{p_n})) \tag{3.13}$$

- Encontrar a regra que possui maior grau de compatibilidade com o padrão,

$$\text{Max } \{Compat(R_k,e_p)\} \quad , k=1...S \tag{3.14}$$

- Ao padrão e_p será atribuída a classe $Classe_j$, tal que $Classe_j$ é a classe da regra R_k que possui o maior grau de compatibilidade com o padrão.

Graficamente, esta forma de raciocínio *fuzzy* pode ser vista na Figura 3.6.

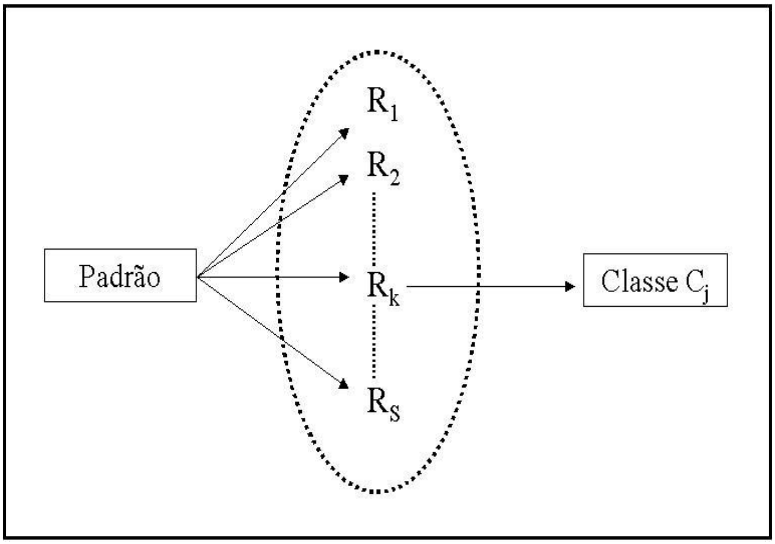

Figura 3.6. Raciocínio *fuzzy* Clássico

Esta forma de raciocínio usa somente uma regra para classificar o padrão e desperdiça as informações providas por outras regras, embora elas possuam menor grau de compatibilidade com o padrão.

Sendo assim, um outro método de raciocínio que combina as informações providas por todas as regras para classificar o padrão pode ser utilizada. Ele recebe o nome de *Método de Raciocínio Fuzzy Geral* e se dá pelos seguintes passos, adaptados de [Ishibuchi et al., 1999a]:

- Calcular o grau de compatibilidade, $Compat(R_k,e_p)$, entre o padrão e_p e cada regra R_k, k=1...S, usando uma T-norma sobre o grau de pertinência dos valores dos atributos do padrão, a_{p_j}, aos correspondentes conjuntos *fuzzy* que aparecem no antecedente da regra, A_{i_j}, j=1...n.

$$Compat(R_k,e_p) = \mathrm{T}\ (\mu_{A_{i_1}}(a_{p_1}),...,\mu_{A_{i_n}}(a_{p_n})) \tag{3.15}$$

- Calcular para cada Classe C, C=1...M, o $Class_C$ como segue:

$$\mathrm{Class}_C = \Sigma\,\{Compat(R_k,e_p)\ \mid C \text{ é a classe da regra } R_k\} \tag{3.16}$$

- O padrão e_p será classificado na classe C, sendo C a classe que possuir maior valor em (3.16).

Graficamente, o Método de Raciocínio *Fuzzy* Geral pode ser visto na Figura 3.7.

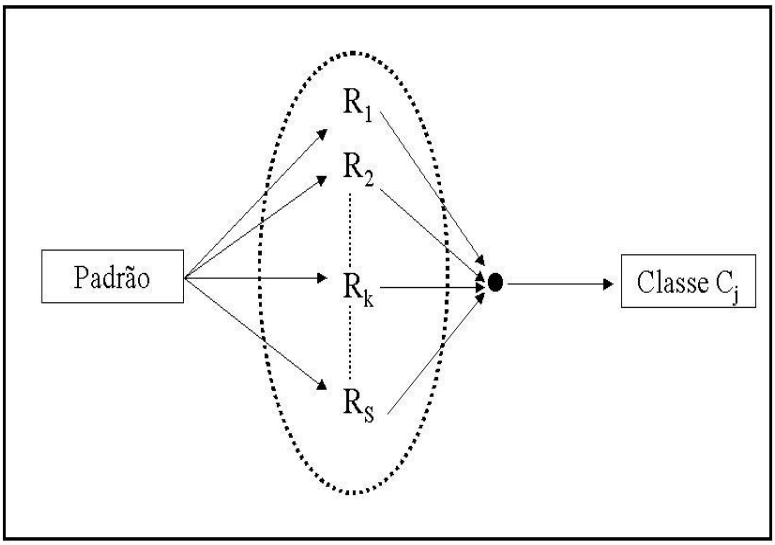

Figura 3.7. Raciocínio *fuzzy* Geral

3.7 Resolução de Conflitos

Ao utilizar o Método de Raciocínio Clássico é possível que mais que uma regra seja aplicável ao mesmo padrão (por exemplo, duas regras possuem o mesmo valor máximo em (3.13)), porém atribuindo-lhe classes diferentes. Sendo assim, é necessário estabelecer normas para determinar qual regra é mais adequada e será disparada para classificar o referido padrão.

No Raciocínio *Fuzzy* Geral pode acontecer de mais que uma classe possuir o mesmo máximo valor em (3.16). Da mesma forma, é preciso determinar qual será a classe a ser atribuída ao padrão.

Alguns dos mecanismos para resolver estes problemas de conflito são:

- Rejeitar a classificação.
- Para o Raciocínio Clássico: Escolher a regra que primeiro aparece na base de regras.
- Para o Raciocínio Geral: Escolher a classe C_i, desde que i seja o menor índice.

Particularmente, neste trabalho foram adotas as duas últimas técnicas para resolução de conflito.

3.8 Considerações Finais

Neste capítulo foram apresentados os principais conceitos de conjuntos *fuzzy* e sistemas *fuzzy*, em especial os sistemas de classificação *fuzzy*. O entendimento dessas teorias é de grande importância para a compreensão deste trabalho. No capítulo seguinte será apresentada a utilização de algoritmos genéticos para a construção da base de regras de sistemas de classificação *fuzzy*.

Capítulo

4

Metodologia Genética para Geração de Regras Fuzzy

4.1 Considerações Iniciais

Conforme apresentado no Capítulo 1, uma das tarefas mais importantes e difíceis no desenvolvimento de um sistema *fuzzy* é gerar a sua base de conhecimento [Cordón & Herrera, 2001].

A problemática existente para extrair o conhecimento necessário diretamente de um especialista deu lugar a técnicas automáticas que partem de amostras ou conjunto de exemplos do problema para compor a base de conhecimento.

Muitas vezes a construção automática da base de conhecimento de sistemas *fuzzy* pode ser considerada como um processo de busca no espaço das possíveis soluções [Cordón et al., 2001c]. Desta forma, as características dos algoritmos genéticos os tornam apropriados para realização de tal tarefa. Um sistema *fuzzy* que possui algum componente da base de conhecimento gerado ou aperfeiçoado por algoritmos genéticos recebe o nome de **Sistema *Fuzzy*-Genético** [Cordón et al., 2001b].

Este capítulo tem por objetivo apresentar a metodologia utilizada neste trabalho para geração de regras de classificação *fuzzy* usando algoritmos genéticos. A metodologia utilizada é composta por duas etapas: 1) geração da base de regras e 2) otimização da base de regras, a fim de eliminar regras desnecessárias. Primeiramente serão apresentadas as 3 abordagens clássicas para geração de regras usando AG e como elas lidam com o problema de competição/cooperação entre os indivíduos da população. Logo em seguida será apresentada a metodologia proposta,

detalhando cada etapa bem como os componentes dos AGs usados.

4.2 Geração Genética da Base de Regras e o Problema de Competição X Cooperação

A partir dos anos 80 houve um aumento no interesse em aplicar AG para a geração automática da base de regras de sistemas *fuzzy* a partir de conjuntos de exemplos do problema. Embora algoritmos genéticos não sejam algoritmos de aprendizado, suas características os tornam perfeitamente aptos para a realização de tal tarefa. Atualmente, pode-se distingüir 3 diferentes abordagens genéticas de aprendizado com relação à representação das regras, conforme discutido em [Cordón et al., 2001a]: Abordagem Pittsburgh, Abordagem Michigan e Abordagem Iterativa.

Na **Abordagem Pittsburgh** cada cromossomo representa uma base de regras completa. Ao terminar a execução do AG, é retornado o melhor cromossomo da população, o qual representa a melhor base de regras encontrada. Em [Castro et al., 2003] e [Corcoran & Sen, 1994] é utilizada esta abordagem de aprendizado para gerar um conjunto de regras de classificação *fuzzy*, enquanto em [Carse et al., 1996] ela é utilizada em problemas de controle *fuzzy* e em [Castro et al., 2004a], para tomada de decisões *fuzzy*.

Já na **Abordagem Michigan** cada cromossomo representa apenas uma regra e a base de regras será formada por todos os indivíduos da população. Como solução, é retornada, dentre todas as gerações, a população que contiver melhor valor de aptidão, obtido em função dos valores de aptidão dos indivíduos que fazem parte dela. Essa abordagem é utilizada em [Parodi & Bonelli, 1993] e [Ishibuchi et al., 1999b] para geração de regras de classificação *fuzzy*.

A **Abordagem Iterativa**, assim como Michigan, considera cada cromossomo como uma regra, porém somente o melhor cromossomo é retornado como solução e o restante é descartado. A regra representada por este cromossomo é inserida na base de regras final e o algoritmo é executado novamente para gerar uma nova regra. Este processo é repetido até que o conjunto de regras obtido seja suficiente para representar o conjunto de exemplos. Dentre os trabalhos

relacionados a esta abordagem estão [Cordón & Herrera, 1996] e [Herrera et al., 1995a], aplicados a problema de controle *fuzzy* e [González & Pérez, 1999], a problema de classificação *fuzzy*.

No AG, indivíduos competem entre si de tal forma que os mais "fortes" (aptos) permanecem na população e os mais "fracos" (menos aptos) tendem a desaparecer. Sendo assim, nas abordagens Michigan e Iterativa as regras ***competem*** umas com as outras, pois cada indivíduo da população representa um regra individual. Entretanto, regras de uma mesma base de conhecimento devem ***cooperar*** entre si, de forma que seu uso combinado apresente melhorias no desempenho final do sistema. Uma regra fraca, no sentido de representar poucos exemplos, pode ser útil por complementar o reconhecimento de casos que não são cobertos por uma regra mais forte e, portanto, deve fazer parte da base de regras.

Desta forma, ao usar as abordagens Michigan e Iterativa é necessário estabelecer uma função de aptidão que seja hábil para medir quão boa é cada regra individualmente e também a qualidade de sua cooperação com as demais.

A abordagem Pittsburgh, embora possa tornar os cromossomos muito extensos, aumentando a complexidade do processo de busca e, conseqüentemente, impondo uma carga maior no processo computacional, possui uma vantagem sobre as outras que compensa e justifica o seu uso: o fato do conjunto completo de regras ser codificado em cada cromossomo, o que permite usar na função de aptidão medidas de desempenho do conjunto inteiro de regras e não somente de regras isoladas. Dessa forma, o problema de competição versus cooperação não existe nesta abordagem, tendo em vista que cada indivíduo representa todo o conjunto de regras, e assim a competição ocorre entre esses conjuntos.

4.3 Metodologia Utilizada

Nesta seção é apresentada a metodologia utilizada neste trabalho para criação da base de regras *fuzzy* para problemas de classificação de padrões a partir de conjuntos de exemplos do problema.

A metodologia a ser apresentada se baseia na existência de uma base de dados previamente definida que contenha os conjuntos *fuzzy* associados a cada variável do problema, bem como suas funções de pertinência que os definem semanticamente.

A metodologia proposta é baseada na abordagem Pittsburgh, sendo composta por 2 etapas, a saber:

1. ***Geração da Base de Regras*** - Nesta etapa um AG é responsável por gerar regras *fuzzy* capazes de representar o conhecimento existente no conjunto de exemplos.

2. ***Otimização da Base de Regras*** - Esta etapa visa excluir, usando também AG, as regras redundantes e desnecessárias que por ventura foram geradas na etapa anterior, resultando numa base de regras compacta.

Embora ambas as etapas utilizam algoritmos genéticos, elas são independentes e cada algoritmo possui sua própria codificação e função de aptidão.

Com estas duas etapas o algoritmo deve ser capaz de gerar uma base de regras compacta com alta habilidade para classificação correta e com regras facilmente compreensíveis para um operador humano.

A escolha da abordagem Pittsburgh foi motivada pela idéia de explorar a vantagem que ela oferece sobre as demais abordagens: possibilidade de avaliar toda a base de regras na função de aptidão, eliminando o problema de *competição/cooperação*.

Ambas as atividades - geração e otimização da base de regras - poderiam ser feitas numa etapa só. Entretanto, a complexidade da busca gerada por tal junção certamente iria comprometer o desempenho da base de regras obtida. Outro motivo para dividir as atividades é a possibilidade de aplicá-las separadamente. Por exemplo, o algoritmo genético para otimização pode ser aplicado como pós-processamento para outros métodos que geram base de regras. Desta forma pode-se analisar seu comportamento quando as regras são criadas por outros métodos de aprendizado.

4.3.1 Conjunto de exemplos

A metodologia proposta requer a disponibilidade de um conjunto finito de exemplos do problema, por meio do qual ela possa extrair as regras *fuzzy*. Este conjunto de exemplos é denotado por $E=\{e_1, e_2, ..., e_L\}$, em que cada exemplo e_i, i=1...L, está no formato atributos-classe.

É desejável que as regras geradas possuam uma boa capacidade de generalização, a fim de que elas possam classificar corretamente novos padrões e não somente os que fazem parte do conjunto de exemplos. Para avaliar tal habilidade, cada conjunto de exemplos é divido aleatoriamente em 2 conjuntos. Um recebe o nome de *conjunto de treinamento* e o outro de *conjunto de teste*.

Neste trabalho, cada conjunto de exemplos foi particionado aleatoriamente da seguinte forma: 70% para treinamento e 30% para teste. Este particionamento foi feito 5 vezes, gerando-se assim, 5 pares treinamento-teste. Os conjuntos de treinamento servirão para a metodologia gerar as regras *fuzzy*, as quais serão aplicadas nos conjuntos de teste. Desta forma, é possível observar o comportamento das bases de regras geradas quanto a capacidade de classificação de novos padrões.

A metodologia para geração de regras de classificação *fuzzy* a partir de exemplos é extremamente dependente da qualidade destes. Se o conjunto de exemplos está incompleto ou contém erros, a base de regras gerada não será boa.

4.3.2 Funções de pertinência

Na metodologia proposta, as funções de pertinência associadas aos conjuntos *fuzzy* foram definidas pelo algoritmo de agrupamento *fuzzy* FC-Means (FCM) [Baraldi & Blonda, 1999a] [Baraldi & Blonda, 1999b]. Agrupamento é uma técnica para reunir em grupos dados ou objetos que possuem algumas semelhanças. O algoritmo clássico de agrupamento gera partições tal que cada objeto pertence a um único grupo. Já o agrupamento *fuzzy* permite que um objeto pertença a vários grupos simultaneamente, com

diferentes graus de pertinência, definidos pelo próprio algoritmo de agrupamento.

O processo para geração de conjuntos *fuzzy* utilizando FCM foi aplicado no domínio de cada atributo individualmente da seguinte forma: suponha um conjunto de L exemplos para treinamento E=$\{e_1, e_2, ..., e_L\}$, com cada exemplo $e_p = \{a_{p_1}, a_{p_2}, ..., a_{p_n}\}$ sendo composto por n atributos. Define-se para cada atributo o número de conjuntos *fuzzy* a ser gerado e aplica-se então o FCM sobre cada domínio de atributo separadamente, gerando os conjuntos *fuzzy* (grupos), os quais são definidos pela função de pertinência usada pelo algoritmo de agrupamento.

4.3.3 Etapa 1 - Geração genética da base de regras

Nesta subseção será descrito o processo de geração da base de regras *fuzzy*. A partir de um conjunto de dados que representa amostras ou exemplos do problema, o AG procura por uma base de regras mais apropriada para classificar corretamente estes exemplos. As funções de pertinência são definidas previamente pelo algoritmo FCM e permanecem fixas durante todo o processo de geração. Este esquema pode ser visto na Figura 4.1.

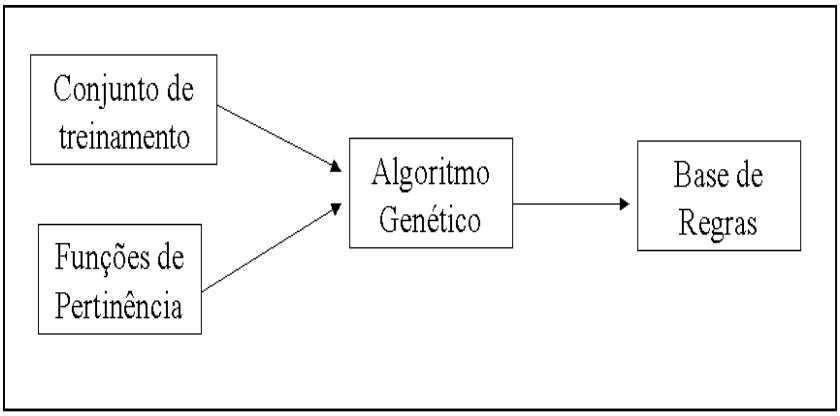

Figura 4.1. Processo genético de geração de regras

O AG desta etapa é responsável por evoluir uma população de bases de regras a fim de encontrar uma base de regras apropriada, que consiga classificar o maior número possível de exemplos. O algoritmo obedece ao seguinte esquema:

1. Gerar uma população inicial com NPop bases de regras.

2. Classificar todos os exemplos do conjunto de treinamento usando as bases de regras da população corrente e calcular o valor de aptidão para cada uma.

3. Aplicar as operações genéticas nas bases de regras.

4. Se atingir o critério de parada, retornar a base de regras com maior valor de aptidão. Caso contrário, voltar ao passo 2.

A seguir serão expostos em detalhes os principais componentes do AG utilizado nesta etapa, como codificação, função de aptidão, operadores genéticos, forma de gerar a população inicial e condição de término do algoritmo.

- **Codificação das Regras**

Primeiramente é preciso estabelecer a forma de codificar as possíveis soluções para que o AG possa manipulá-las. Foi definido que os cromossomos possuem tamanho fixo e é necessário estabelecer previamente a quantidade de regras que eles irão codificar. As regras são codificadas com números inteiros, os quais representam os índices dos conjuntos *fuzzy* que aparecem na sua parte antecedente e conseqüente. Se o conseqüente não for *fuzzy*, cada classe receberá um número que a identificará. O número 0 (zero) será usado para representar a condição *"don't care"* no antecedente das regras, ou seja, não importa o valor da variável.

Em se tratando da abordagem Pittsburgh, cada indivíduo da população representa uma base de regras completa. Como exemplo, considere um problema de classificação em que os padrões possuem 4 variáveis, sendo 3 atributos X_1, X_2 e X_3, mais 1 classe $Classe_j$, j=1..3. Os atributos são *fuzzy* e estão associados aos domínios $D_1 = \{A_{1_1}, A_{2_1}, A_{3_1}\}$, $D_2 = \{A_{1_2}, A_{2_2}, A_{3_2}\}$ e $D_3 = \{A_{1_3}, A_{2_3}, A_{3_3}\}$, respectivamente.

Observe o cromossomo C_i da Figura 4.2. Ele está codificando k regras e cada uma é representada por 4 genes, em que os 3 primeiros indicam o índice dos conjuntos *fuzzy* dos 3 atributos X_1, X_2 e X_3 e o quarto gene representa o índice da classe na regra correspondente.

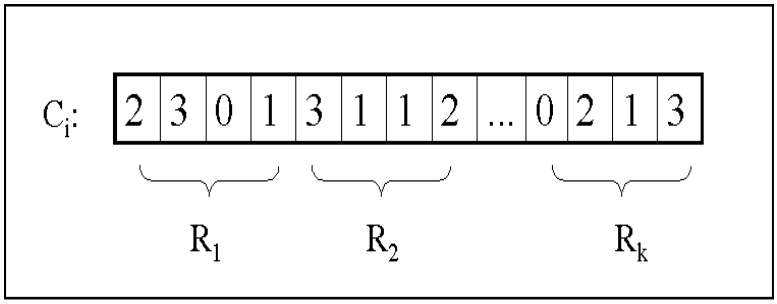

Figura 4.2. Codificação de um cromossomo na etapa de geração

A base de regras codificada pelo cromossomo da Figura 4.2 é como segue:

R_1: SE X_1 é A_{2_1} E X_2 é A_{3_2} E X_3 é *"don't care"*, ENTÃO $Classe_1$
R_2: SE X_1 é A_{3_1} E X_2 é A_{1_2} E X_3 é A_{1_3}, ENTÃO $Classe_2$
$\vdots$
R_k: SE X_1 é *"don't care"* E X_2 é A_{2_2} E X_3 é A_{1_3}, ENTÃO $Classe_3$

O uso do *"don't care"* torna as regras mais simples e mais compreensíveis, pelo fato delas terem menos termos na parte antecedente. Por exemplo, a regra R_1 pode ser reescrita como:

SE X_1 é A_{2_1} E X_2 é A_{3_2}, ENTÃO $Classe_1$

Além disso, sua utilização propicia uma maior capacidade de generalização da base de regras. O efeito do *"don't care"* será comprovado durante os testes e discutido no capítulo seguinte.

A forma de codificar as possíveis soluções do problema é um dos pontos mais importantes no desenvolvimento do AG. Foi escolhida a codificação inteira, pois em se tratando da abordagem Pittsburgh, cada cromossomo ficaria muito extenso se fosse utilizada a codificação binária, exigindo maior esforço computacional.

- **Formação da População Inicial**

Conforme observado em [Nakaoka et al., 1994], o desempenho da população inicial é, geralmente, muito baixo. Isso porque muitos exemplos não têm compatibilidade com as regras iniciais geradas aleatoriamente. Com base nessa observação, uma população inicial com um bom desempenho pode ser facilmente gerada de forma heurística a partir dos seguintes passos iterativos:

Passo 1: Selecionar aleatoriamente t exemplos do conjunto de exemplos. Para cada exemplo selecionado, compor 1 regra *fuzzy* que tenha maior compatibilidade com o exemplo, gerando-se então t regras *fuzzy*. Isto é feito atribuindo para cada atributo do exemplo o conjunto *fuzzy* em que ele tem maior valor de pertinência.

Por exemplo, suponha o exemplo $e_p = \{a_{p1}, a_{p2}, Classe_1\}$, onde a_{p1} e a_{p2} são os valores de seus atributos e $Classe_1$ a sua classe. Seja X_1 e X_2 as variáveis associadas aos seus respectivos domínios $D_1 = \{A_{1_1}, A_{2_1}\}$ e $D_2 = \{A_{1_2}, A_{1_2}\}$, conforme ilustrado na Figura 4.3.

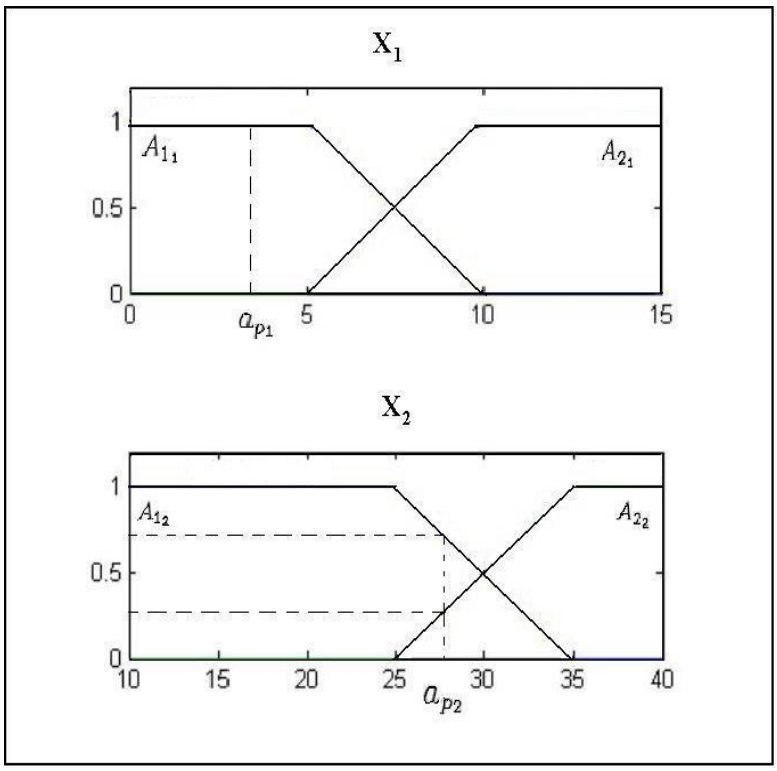

Figura 4.3. Partição *fuzzy* das variáveis X_1 e X_2

A regra mais compatível com o exemplo e_p é

Se X_1 é A_{1_1} e X_2 é A_{1_2}, Então $Classe_1$

Passo 2: Com uma probabilidade pré-especificada, substituir os conjuntos *fuzzy* do antecedente da regra por *"don't care"*.

Passo 3: Concatenar estas t regras formando parte de um cromossomo C_i, i=1..*NPop*, sendo *NPop* o tamanho da população. O restante do cromossomo será formado pela geração

aleatória de números inteiros que podem assumir valor de 0 a q_i, sendo q_i o número de conjuntos *fuzzy* utilizados para representar o atributo a_i.

Passo 4: Terminar a iteração se a população toda estiver formada. Caso contrário, voltar ao Passo 1.

A formação da população inicial que combina regras geradas a partir dos conjuntos de exemplos com regras geradas aleatoriamente fornece um melhor ponto de partida, pois combina regras específicas com regras que podem conter informações que o conjunto de exemplos não cobre.

- **Função de Aptidão**

A função de aptidão, responsável por avaliar cada cromossomo C_i, i=1..$NPop$ é definida com base no desempenho do conjunto de regras nele codificado, medido pelo número de exemplos classificados corretamente (NEC), usando algum método de raciocínio apresentado na subseção 3.6. No caso do conseqüente da regra possuir uma classe *fuzzy*, a classificação será considerada correta se esta classe for aquela à qual o padrão possuir maior grau de pertinência.

O melhor cromossomo será aquele que maximizar a função em (4.1):

$$\text{valor\_aptidão}(C_i) = \text{NEC}(C_i) \tag{4.1}$$

- **Operadores Genéticos**

A seleção é a primeira operação a ser empregada e utiliza o método da Amostragem Universal Estocástica para escolher os indivíduos. A operação de seleção é combinada com a técnica elitista, em que uma porcentagem dos melhores indivíduos da população sempre sobrevive para compor a próxima.

A segunda operação a ser realizada é o cruzamento de um ponto. Os pares de cromossomos a serem cruzados e o ponto de cruzamento são escolhidos aleatoriamente.

Por fim, a mutação padrão é aplicada para alterar um gene c_{ij} para um novo valor escolhido aleatoriamente dentro dos possíveis valores $\{0, 1, ..., G\}$, sendo G o número máximo que o gene

pode assumir. Em outras palavras, G é o número máximo de conjuntos *fuzzy* definido para a variável representada pelo gene c_{ij}.

- **Critério de Parada**

Como critério de parada do algoritmo foi adotado o número máximo de gerações. Retorna-se então como solução o cromossomo da população da última geração que possuir melhor valor de aptidão.

4.3.4 Etapa 2 - Otimização genética da base de regras

Tão logo o processo de geração termina, a base de regras contém algumas regras redundantes e desnecessárias. Regras redundantes e desnecessárias são aquelas regras que são semelhantes a outras ou aquelas que não são usadas para classificar nenhum exemplo. Assim, um processo para eliminá-las é requerido.

O objetivo desta etapa é encontrar, a partir das regras geradas na etapa anterior, um número reduzido de regras que possuam um bom nível de cooperação entre si. Pela Figura 4.4 pode-se ter uma idéia geral de funcionamento deste processo.

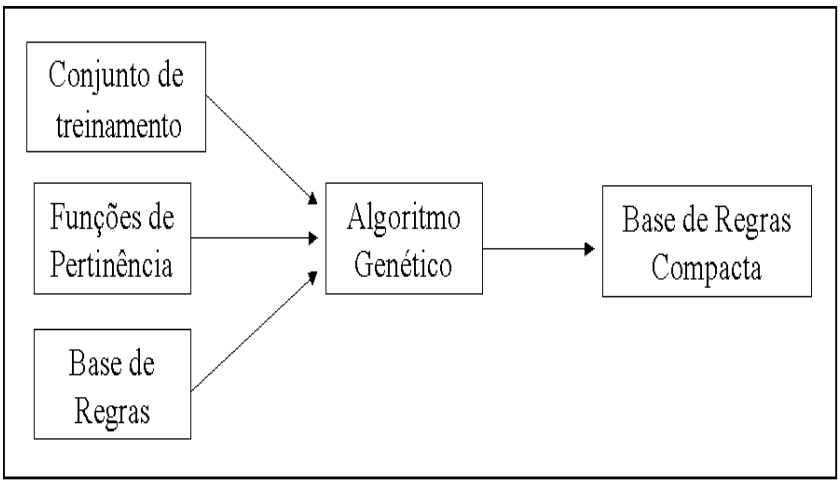

Figura 4.4. Processo genético de simplificação da base de regras

O algoritmo genético desta etapa é responsável por manter sub-conjuntos de regras (combinações de regras a partir das regras obtidas na etapa anterior) e encontrar a melhor

combinação que classifique o maior número de exemplos com a menor quantidade de regras. Tal algoritmo obedece ao seguinte esquema:

1. Gerar uma população inicial com NPop sub-conjuntos de regras a partir das regras previamente obtidas.

2. Classificar todos os exemplos de treinamento usando os sub-conjuntos de regras representados em cada indivíduo da população e calcular o valor de aptidão para cada um.

3. Aplicar as operações genéticas.

4. Se atingir o critério de parada, retorna o sub-conjunto de regras com maior valor de aptidão. Caso contrário, volta ao passo 2.

Em seguida são apresentados os componentes do AG utilizado nesta etapa, como codificação, a forma de gerar a população inicial e função de aptidão. Os operadores genéticos utilizados são os mesmos do algoritmo da etapa anterior, exceto que agora são aplicados a cromossomos com codificação binária. O critério de parada também é o mesmo da etapa anterior.

- **Codificação das Regras**

Nesta etapa cada sub-conjunto (combinação) de regras será representado por cromossomos C_i=(c_{i1},c_{i2},...,c_{im}) que são codificados com seqüências de dígitos binários e possuem tamanho fixo m, sendo m o número de regras geradas pela etapa anterior. Cada gene c_{ij} (um dígito binário) é associado a uma regra. Se c_{ij}=1, então a regra correspondente a este gene está ativada e faz parte do sub-conjunto representado pelo cromossomo C_i, caso contrário a regra está desativada e não faz parte do sub-conjunto em questão.

Suponha que na etapa anterior foi gerada uma base de regras com 10 regras, R_1, R_2,...,R_{10}, e que esta precisa ser otimizada. O cromossomo da Figura 4.5 representa um possível sub-conjunto composto somente pelas regras R_1, R_4, R_5, R_6 e R_{10}. Logo, estas regras estão ativadas e os exemplos de treinamento serão classificados somente por elas.

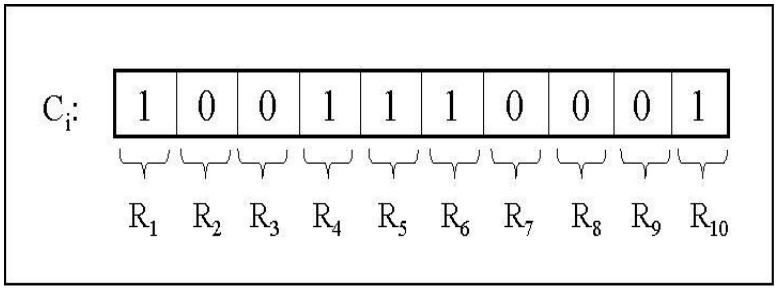

Figura 4.5. Codificação de um cromossomo na etapa de otimização

- **População Inicial**

A população inicial é gerada introduzindo um cromossomo que representa todas as regras previamente obtidas, ou seja, todos os genes do cromossomo recebem valor 1. Os cromossomos restantes são gerados aleatoriamente.

- **Função de aptidão**

A função de aptidão utilizada nesta etapa avalia cada cromossomo (sub-conjunto de regras candidatas) com base em dois critérios: número de exemplos corretamente classificados por ele e o número de regras contido nesse sub-conjunto. Serão privilegiados os cromossomos que apresentarem o maior número de exemplos classificados corretamente com o menor número de regras. Desta forma, a função de aptidão a ser maximizada é definida como:

$$\text{valor\_aptidão}(C_i) = \text{NEC}(C_i) * (\text{S - QTDR}(C_i)) \tag{4.2}$$

em que NEC(C_i)representa o número de exemplos classificados corretamente pelo sub-conjunto de regras representado no cromossomo i, S é o total de regras obtidas na etapa anterior e QTDR(C_i) indica a quantidade de regras existentes no sub-conjunto representado pelo mesmo cromossomo i.

4.4 Algoritmo Genético Auto-Adaptativo

Realizando alguns experimentos, os quais são reportados no capítulo seguinte, foi observado que o algoritmo da etapa de geração convergia rapidamente, deixando de explorar alguns

pontos no espaço de busca. Isto acontecia porque cada cromossomo codificava uma quantidade relativamente baixa de regras. No entanto, se este número fosse aumentado consideravelmente, o espaço de busca se tornaria maior e mais complexo.

Pensando nisso foi desenvolvida uma versão adaptativa do algoritmo, que altera dinamicamente os valores da taxa de mutação e da taxa de cruzamento. A mutação e o cruzamento são operadores que desempenham um papel importante para minimizar o problema da convergência prematura, introduzindo novos indivíduos na população e explorando a troca de material genético entre estes indivíduos, respectivamente. O ajuste dos valores dos parâmetros é feito com base na diversidade genética da população, a qual é obtida por (4.3).

$$\text{dg}=aptidão\_média/aptidão\_máxima \tag{4.3}$$

A aptidão_média representa a média dos valores de aptidão de todos os cromossomos da população e aptidão_máxima, o maior valor de aptidão encontrado na população daquela geração.

O coeficiente *dg* é calculado a cada 5 gerações e seu valor é um número no intervalo [0,1]. Quando *dg* é próximo a 1, indica que a diversidade é pequena. Para evitar convergência prematura, as taxas de cruzamento e mutação precisam ser modificadas a fim de introduzir novas características genéticas e reduzir a perda de material genético. Assim, a taxa de mutação precisa ser aumentada e a de cruzamento reduzida. Por outro lado, se *dg* é próximo a 0 (zero), significa que uma grande diversidade genética foi introduzida pelo operador de mutação. Para evitar uma busca aleatória, a taxa de mutação precisa ser reduzida e a de cruzamento ser aumentada. Este procedimento pode ser melhor compreendido pelo Algoritmo 2, adaptado de [Castro & Camargo, 2004c].

Os parâmetros VMax e VMin indicam os limites do intervalo de valores que *dg* pode assumir e $v \geq 1$ é uma constante. Msup e Minf são os limites superior e inferior, respectivamente, do intervalo que a taxa de mutação pode assumir e Csup e Cinf são os limites superior e inferior, respectivamente, do intervalo que a taxa de cruzamento pode assumir.

Se o número máximo de gerações é dado por *Maxgen*, então o procedimento descrito pelo

Algoritmo 2 Alteração Dinâmica de Valores de Parâmetros

Início
Se dg $\geq$ *Vmax*
{
taxa de mutação = min (taxa de mutação**v*, Msup);
taxa de cruzamento = max (taxa de cruzamento/*v*, Cinf);
}
Senão Se dg $\leq$ *Vmin*
{
taxa de mutação = max (taxa de mutação/*v*, Minf);
taxa de cruzamento = min (taxa de cruzamento**v*, Csup);
}
Fim

Algoritmo 2 é executado somente nas primeiras *Maxgen*/2 gerações. Isto porque neste instante a diversidade genética já deve estar alta e, então, o interessante será explorar o conteúdo genético da população. Assim, a taxa de mutação cai automaticamente para Minf e a taxa de cruzamento fica sendo Csup.

4.5 Considerações Finais

Este capítulo apresentou uma metodologia que usa algoritmos genéticos para gerar e otimizar bases de regras *fuzzy* a partir de um conjunto de exemplos. No capítulo seguinte esta metodologia será avaliada por diferentes tipos de testes em alguns conjuntos de dados.

Capítulo

5 *Resultados Experimentais*

5.1 Considerações Iniciais

Com a finalidade de testar o método previamente apresentado de geração automática de regras para problemas de classificação de padrões, foram realizados diferentes experimentos utilizando alguns conjuntos de dados obtidos no UCI Repository of Machine Learning Databases [Blake & Merz, 1998].

Primeiramente foi realizado um teste usando algoritmos genéticos em que os valores dos parâmetros permaneciam fixos durante todo o processo de execução. Em seguida, foi utilizado um algoritmo genético auto-adaptativo que altera dinamicamente os valores da taxa de cruzamento e mutação, a fim de garantir diversidade genética na população e evitar convergência prematura. Os resultados obtidos por este algoritmo são comparados com os resultados obtidos no algoritmo anterior.

Outro teste realizado foi quanto à granularidade da partição *fuzzy* dos domínios das variáveis. Foi observado se os resultados variam muito conforme o número de conjuntos *fuzzy* utilizados para representar as variáveis lingüísticas do problema.

Posteriormente foi analisada a importância do uso do *"don't care"* nos antecedentes das regras, em função da acuidade e interpretabilidade da base de regras obtida.

O desempenho das bases de regras geradas foi avaliado também sob duas diferentes formas de raciocínio. A primeira usa somente uma regra para classificar um padrão e a segunda forma combina informações de todas as regras contidas na base para a classificação.

Por fim, a metodologia genética para geração de regras foi comparada com um método *"ad-hoc"* bastante conhecido e utilizado, o método Wang-Mendel.

A metodologia foi implementada usando a linguagem C++. O programa resultante realiza a leitura de arquivos textos contendo a descrição dos conjuntos de dados utilizados.

Na seção 5.2 são apresentados os conjuntos de dados utilizados e a forma de gerar as funções de pertinência para os conjuntos *fuzzy*. Na seção 5.3 são reportados os diferentes testes realizados.

5.2 Descrição dos Conjuntos de Dados

Os conjuntos de dados utilizados durante os testes são apresentados na Tabela 5.1.

Tabela 5.1. Conjuntos utilizados nos experimentos

Conjunto de dados	Quantidade de exemplos	Número de variáveis	Número de classes	Tipo da classe
Iris	150	5	3	Não *fuzzy*
Vinho	178	14	3	Não *fuzzy*
Glass	214	10	6	Não *fuzzy*
Auto_mpg	392	8	3	*Fuzzy*
Boston Housing	506	13	3	*Fuzzy*

O conjunto de dados Iris contém dados sobre plantas e o objetivo é classificá-las em 3 diferentes tipos. O conjunto Vinho contém análises químicas de vinhos, os quais devem ser classificados em 3 tipos. Com o conjunto Glass o objetivo é identificar 6 diferentes tipos de vidros. O conjunto de dados Auto_mpg se refere ao consumo de combustível por automóveis na forma de milhas/galões. O conjunto Boston Housing concerne dados sobre o preço de casas nos subúrbios de Boston.

O domínio de cada um dos atributos dos conjuntos de dados estão representados por valores numéricos contínuos e precisam ser granularizados em um número de conjuntos fuzzy que representem os valores lingüísticos que cada atributo pode assumir, a fim de permitir a generalização do conhecimento.

As variáveis de saída (de classificação) dos conjuntos de dados Iris, Vinho e Glass possuem valores nominais. Para este trabalho esses valores nominais são transformados em valores numéricos e representados como 1, 2, 3, etc, por exemplo. Já as dos conjuntos Boston Housing e Auto_mpg possuem valores numéricos contínuos e precisam ser granularizados da mesma forma que os atributos. Neste trabalho, todas as variáveis contínuas foram granularizadas, gerando-se 3 conjuntos *fuzzy* para cada uma. Dessa forma foi definido que as classes para a representar a variável consumo no conjunto Auto_mpg são *baixo*, *médio* e *alto*. Seguindo o mesmo raciocínio, as classes para representar a variável preço das casas no conjunto Boston Housing são *baixo*, *médio* e *alto*.

5.3 Testes e Resultados

Conforme já havia sido citado na subseção 4.3.1, cada conjunto de dados foi particionado em 2 da seguinte forma: 70% para treinamento e 30% para teste. Este particionamento foi feito 5 vezes, gerando-se assim, para cada domínio de conhecimento, 5 pares treinamento-teste. O algoritmo genético é aplicado no conjunto de treinamento para gerar as regras, as quais são aplicadas no conjunto de teste para observar o comportamento do método quanto a classificação de novos padrões.

Os parâmetros utilizados nos AGs para realização destes experimentos podem ser vistos na Tabela 5.2.

Tabela 5.2. Parâmetros dos Algoritmos Genéticos

Parâmetro	Valor
Máximo número de gerações	1000
Tamanho da população	150
Probabilidade de mutação	0.01
Probabilidade de cruzamento	0.7

Os experimentos foram conduzidos utilizando-se o Método de Raciocínio Clássico e o produto algébrico como T-norma. Posteriormente foram realizados alguns testes utilizando

o Método de Raciocínio Geral. Foi estipulado o elitismo de 1, isto é, apenas o melhor indivíduo da população atual é copiado para a população seguinte.

Em todos os experimentos,os parâmetros que foram utilizados para medir a eficiência das bases de regras obtidas são acuidade e quantidade de regras geradas.

Na presente metodologia é necessário estabelecer previamente a quantidade de regras a serem codificadas em cada cromossomo. Foi observado em [Castro & Camargo, 2004b] que quando este número era muito pequeno, havia pouca diversidade genética e o algoritmo convergia rapidamente, deixando de explorar alguns pontos do espaço de busca. Como solução foi definido que cada cromossomo irá codificar um elevado número de regras a fim de garantir uma maior diversidade genética na população. Assim, os cromossomos codificam em torno de 150 a 160 regras, dependendo do conjunto de dados. Isso não significa que todas farão parte da base de regras final, pois algumas delas, desnecessárias, serão excluídas na etapa de otimização.

Os resultados finais aqui apresentados são obtidos pela média dos resultados de 5 execuções e podem ser vistos na Tabela 5.3.

Tabela 5.3. Resultados da classificação

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	95,2 %	0.549363	13
Boston Housing	89,8 %	0.748331	17
Iris	97,2 %	0.399889	10
Vinho	94,7 %	0.500247	12
Glass	80,3 %	0.423720	14

Os resultados demonstram claramente a habilidade do algoritmo para encontrar um conjunto reduzido de regras com habilidade de classificação considerada satisfatória. Pode-se ver também que, como era de se esperar, o desempenho do algoritmo tende a decrescer em problemas onde o número de variáveis é relativamente alto, tendo em vista o elevado número de regras que cada cromossomo codifica e a utilização da abordagem Pittsburgh, o que ocasiona um aumento no processo de busca tornando-o mais complexo.

Quando o mesmo teste é repetido para os conjuntos Boston Housing e Vinho variando o número de variáveis na parte antecedente, o desempenho melhora, conforme apresentado nas

Tabelas 5.4 e 5.5.

Tabela 5.4. Resultados para o conjunto de dados Vinho

Número de variáveis	Taxa de Classificação	Desvio Padrão	Número de Regras
11	96,3 %	0.473258	10
12	95,6 %	0.475102	11
13	95,1 %	0.497830	11
14	94,7 %	0.500247	12

Tabela 5.5. Resultados para o conjunto de dados Boston Housing

Número de variáveis	Taxa de Classificação	Desvio Padrão	Número de Regras
10	92,3 %	0.651022	14
11	91,6 %	0.709346	16
12	90,7 %	0.682715	17
13	89,8 %	0.748331	17

As variáveis aqui utilizadas foram escolhidas aleatoriamente sendo que, quando os conjuntos Vinho e Boston Housing possuem 13 e 12 variáveis, respectivamente, eles correspondem aos conjuntos de dados originais com todas as variáveis.

5.3.1 Resultados usando o Algoritmo Genético Auto-Adaptativo

Embora bons resultados foram obtidos, o fato dos cromossomos codificarem um alto número de regras para oferecer diversidade genética na população não se mostrou uma boa solução. O espaço de busca se torna maior, exigindo maior esforço computacional e tempo de processamento.

Numa tentativa de evitar este inconveniente, foi utilizado o algoritmo genético auto-adaptativo apresentado na seção 4.4 para garantir diversidade genética, em vez de definir um alto número de regras em cada cromossomo. Os valores dos parâmetros para o algoritmo genético auto-adaptativo foram definidos empiricamente e são apresentados na Tabela 5.6.

Os valores iniciais para a taxa de cruzamento e mutação são 0.7 e 0.015, respectivamente e cada cromossomo codifica desta vez apenas 60 regras.

Tabela 5.6. Parâmetros para o Algoritmo Genético Auto-Adaptativo

Parâmetros	Valores
Vmax	0.8
Vmin	0.3
Csup	0.9
Cinf	0.1
Msup	0.25
Minf	0.01
v	1.3

Os resultados da classificação usando a versão auto-adaptativa do algoritmo podem ser vistos na Tabela 5.7.

Tabela 5.7. Resultados usando Algoritmo Genético Auto-Adaptativo

Conjunto de dados	Taxa de Classificação	Desvio padrão	Número de regras
Auto_mpg	96.4 %	0.412284	11
Boston Housing	91.2 %	0.507402	13
Iris	99.8 %	0.381437	11
Vinho	98.2 %	0.479045	12
Glass	84.1 %	0.422670	13

Comparando estes resultados com os resultados previamente apresentados na Tabela 5.3, é possível ver que a mudança dinâmica de valores de parâmetros melhorou o desempenho do algoritmo, aumentando a habilidade de classificação das bases de regras, diminuindo o espaço de busca e, por conseguinte, sua complexidade. Por isso nos testes seguintes será utilizado somente o algoritmo genético auto-adaptativo.

5.3.2 Evolução das regras

A obtenção de bons resultados deve-se em parte ao método heurístico de gerar a população inicial, apresentado na subseção 4.1, o qual garante que o desempenho das regras seja bom desde as primeiras gerações. Na Figura 5.1 é possível observar como o algoritmo evolui os conjuntos de regras. Cada ponto do gráfico representa a taxa média de classificação sobre as 5

execuções, para o conjunto de treinamento da base de dados Vinho, em cada geração. Pode-se ver que a taxa média de classificação foi rapidamente melhorada desde as primeiras gerações.

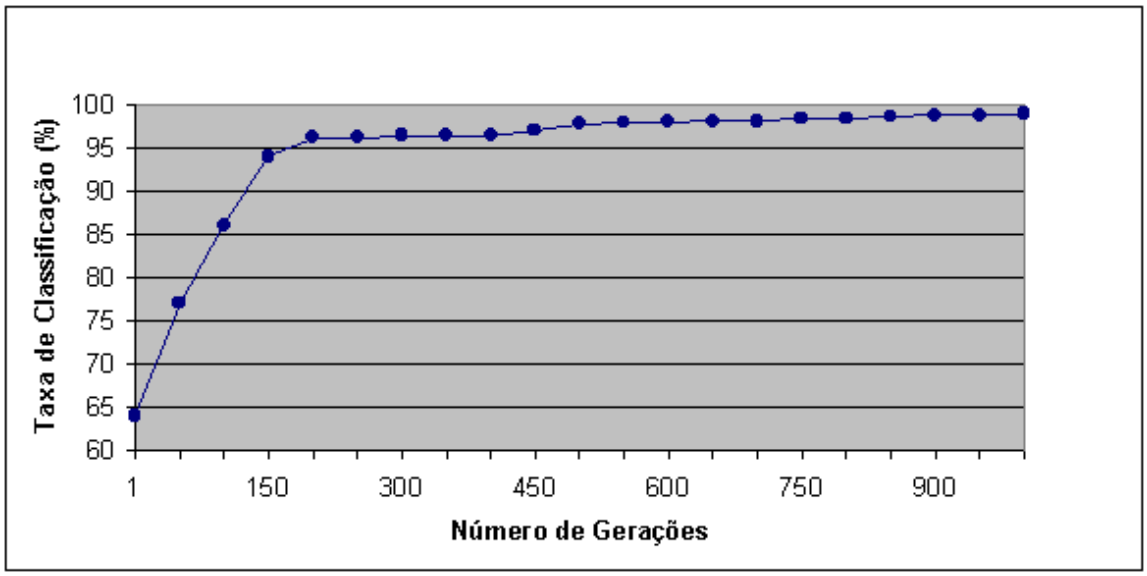

Figura 5.1. Evolução das regras

5.3.3 Resultados usando diferentes partições *fuzzy*

Nos experimentos descritos até agora, foram utilizados somente 3 conjuntos *fuzzy* para representar cada variável. Segundo [Cordón et al., 2001e], a partição *fuzzy* exerce uma influência significativa no desempenho da base regras gerada. Com base nisso, foi examinado também o desempenho do algoritmo mediante diferentes partições *fuzzy* para as variáveis. No lugar de 3 conjuntos *fuzzy*, foram feitos testes usando 4 e depois 5 conjuntos *fuzzy* para representar cada variável. Os resultados podem ser vistos nas Tabelas 5.8 e 5.9.

Tabela 5.8. Resultado da classificação usando 4 conjuntos *fuzzy*

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	96.6 %	0.549786	11
Boston Housing	90.8 %	0.644105	14
Iris	99.4 %	0.400267	11
Vinho	98.2 %	0.479741	13
Glass	84.4 %	0.431993	14

Tabela 5.9. Resultado da classificação usando 5 conjuntos *fuzzy*

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	96.4 %	0.520698	12
Boston Housing	91.7 %	0.718924	13
Iris	99.6 %	0.398240	12
Vinho	98.4 %	0.480024	13
Glass	82.8 %	0.440356	13

Conforme aumenta a quantidade de conjuntos *fuzzy* por variável, aumenta também a quantidade de possíveis combinações para formar as regras. Logo, era esperado que o desempenho do algoritmo decaísse, tendo em vista que o tamanho do espaço de busca cresce e se torna mais complexo. Entretanto, pode ser visto que para estes 3 casos o desempenho do algoritmo não é muito sensível quanto à escolha da partição *fuzzy*, conforme comparado na Figura 5.2.

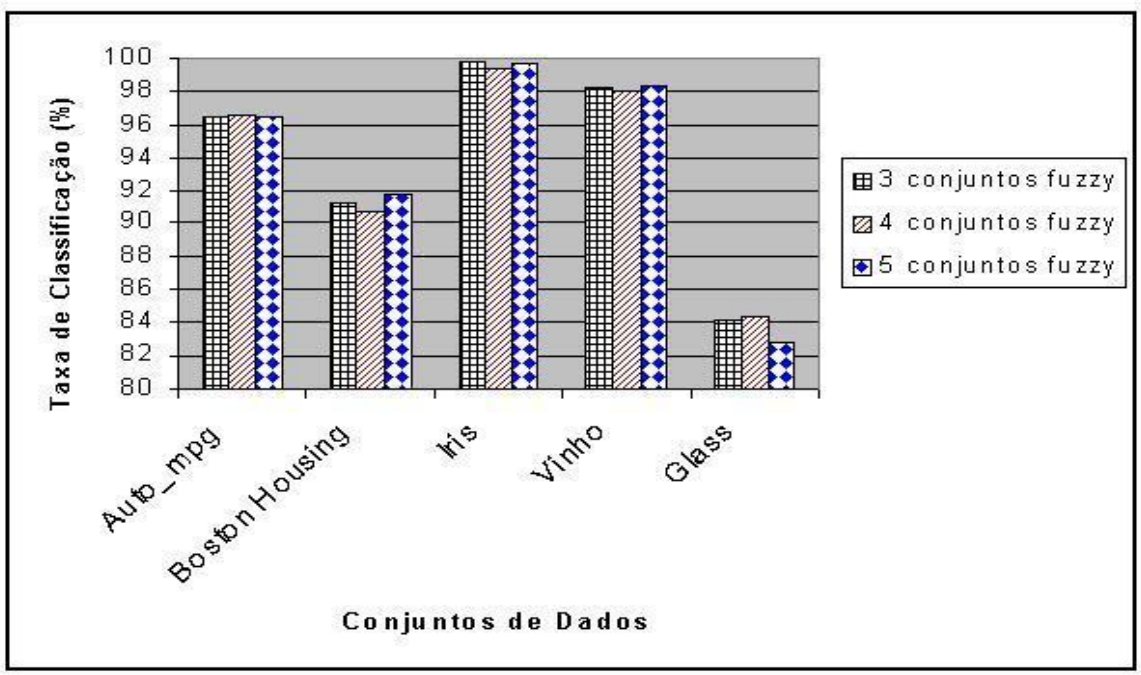

Figura 5.2. Resultados com diferentes partições *fuzzy*

5.3.4 Resultados usando funções de pertinência triangulares

As funções de pertinência estão sendo geradas pelo algoritmo de agrupamento *fuzzy* FC-Means. Seria interessante também observar o comportamento do algoritmo ao utilizar outra forma para gerá-las. Foi decidido então distribuir uniformemente funções de pertinência triangulares no domínio de cada variável. Os resultados podem ser vistos na Tabela 5.10.

Tabela 5.10. Resultado usando funções de pertinência triangulares

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	95.6 %	0.523760	13
Boston Housing	89.1 %	0.571282	12
Iris	96.9 %	0.442699	11
Vinho	95.2 %	0.406544	13
Glass	81.8 %	0.479022	12

Com base nestes resultados, para esta metodologia e estes conjuntos de dados, distribuir funções de pertinência triangulares no domínio de cada variável não é uma boa forma de gerar as partições *fuzzy*. Para tentar conseguir melhores resultados, seria recomendável uma terceira etapa, a de ajuste dos parâmetros das funções. Como a metodologia apresentada aqui não abrange tal etapa, continuará sendo utilizado o algoritmo FC-Means para criar as partições *fuzzy*.

5.3.5 Resultados usando o raciocínio *fuzzy* Geral

Embora o Raciocínio Clássico seja um método muito simples, os resultados obtidos até agora com sua utilização são bastante satisfatórios. No intuito de observar o comportamento do algoritmo mediante outra forma de raciocínio, foi examinado também o desempenho das bases de regras geradas frente ao Método de Raciocínio Geral. Os resultados são reportados na Tabela 5.11.

Tabela 5.11. Resultados da classificação com o Método de Raciocínio Geral

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	96.2 %	0.487324	12
Boston Housing	92.4 %	0.647980	11
Iris	99.4 %	0.401965	09
Vinho	97.8 %	0.420341	10
Glass	84.7 %	0.435069	11

Como pode-se ver na Tabela 5.11 e conforme já havia sido apresentado em [Castro & Camargo, 2004a] e [Castro & Camargo, 2004d], os resultados obtidos usando o Raciocínio Geral também são muito bons. Mas quando comparados aos apresentados na Tabela 5.7, onde a forma de inferência utilizada foi o Raciocínio Clássico, somente os conjuntos de dados Boston Housing e Glass obtiveram resultados superiores.

5.3.6 Importância do *"don't care"*

O uso do *"don't care"*, indiscutivelmente, melhora a capacidade de generalização da base de regras, a fim de que ela possa classificar corretamente novos padrões. Para mostrar a importância do seu uso, o algoritmo foi aplicado aos conjuntos de dados sem usar o *"don't care"* e os resultados podem ser vistos na Tabela 5.12.

Tabela 5.12. Resultados sem usar o *''don't care''*

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	89.9 %	0.480168	22
Boston Housing	85.4 %	0.589482	24
Iris	93.2 %	0.423943	19
Vinho	91.8 %	0.514077	26
Glass	79.6 %	0.498326	22

Além da taxa de classificação ter diminuído, o número de regras aumentou. Na Figura 5.3 é apresentado um comparativo entre as classificações usando *"don't care"* e sem o uso do *"don't care"*.

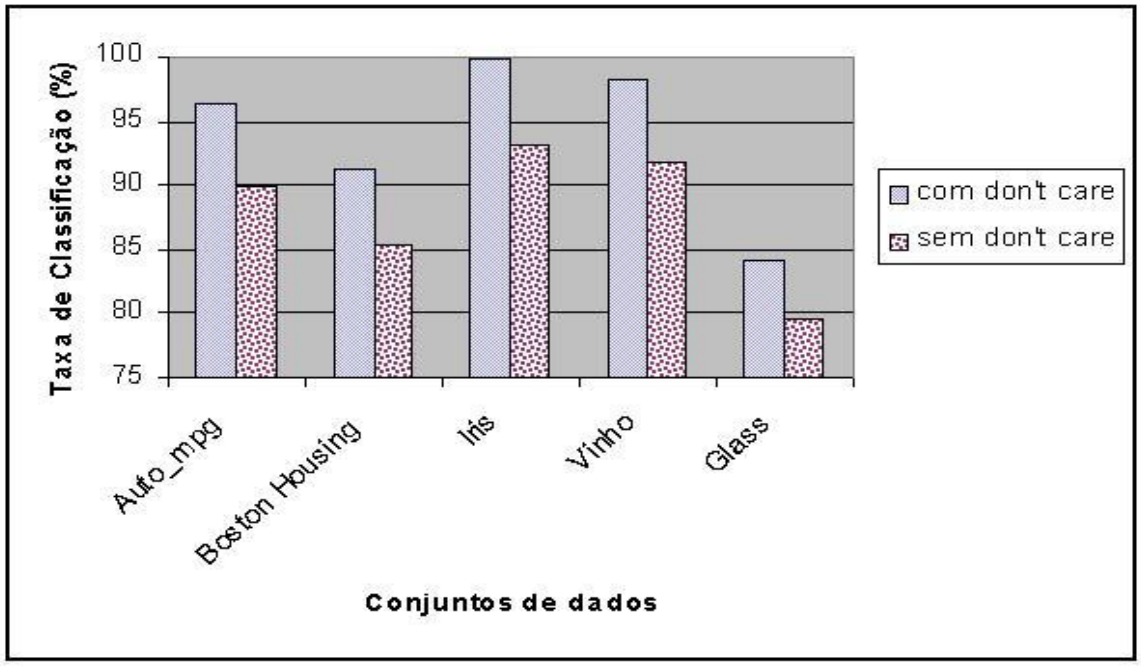

Figura 5.3. Resultados com e sem o uso do *''don't care''*

O uso do *"don't care"* também tem efeito na compreensibilidade das regras geradas, uma vez que estas tem poucos atributos na parte antecedente. Regras curtas podem ser mais facilmente compreendidas por humanos que regras longas, com muitos atributos.

5.3.7 Comparação entre AG e outro método de geração de regras

Por fim, a metodologia apresentada foi comparada com outro método para geração de regras *fuzzy* a partir de exemplos, o método de Wang e Mendel (WM) [Wang & Mendel, 1992]. Este método é descrito no Apêndice A. Os resultados para os conjuntos de dados Auto_mpg e Boston Housing são mostrados nas Tabelas 5.13 e 5.14, respectivamente.

Tabela 5.13. Comparativo entre AG e WM para Auto_mpg

Método	Taxa de Classificação	Desvio Padrão	Número de Regras
AG	96.4 %	0.412284	11
WM	92.5 %	0.326419	147

Tabela 5.14. Comparativo entre AG e WM para Boston Housing

Método	Taxa de Classificação	Desvio Padrão	Número de Regras
AG	91.2 %	0.507402	13
WM	93.4 %	0.392207	186

Para o conjunto de dados Auto_mpg, a geração de regras por AG continua apresentando melhor desempenho. Já para o conjunto Boston Housing, com o método WM obteve-se melhores resultados. Uma desvantagem do WM é o elevado número de regras que ele gera. Enquanto que o AG gerou 11 regras para o conjunto Auto_mpg e 13 para o Boston Housing, o WM gerou 147 e 186, respectivamente.

Foi justificado no Capítulo 4 que um dos motivos para realizar separadamente as tarefas de geração e otimização da base de regras é a possibilidade de aplicá-las de forma independentes. Assim, o próximo teste realizado foi gerar as regras *fuzzy* pelo método WM e aplicar o algoritmo genético para otimização da base de regras obtida como fase de pós-processamento. Os resultados podem ser conferidos na Tabela 5.15.

Tabela 5.15. Otimização via AG de uma base de regras gerada por WM

Conjunto de Dados	Taxa de Classificação	Desvio Padrão	Número de Regras
Auto_mpg	89.2 %	0.513608	63
Boston Housing	90.6 %	0.499624	78

Embora a taxa de classificação tenha caído um pouco, o número de regras também diminuiu de forma significativa. É possível afirmar que o algoritmo genético para otimização pode ser usado com qualquer outro método de geração de regras e apresentar bons resultados. É provável que se as regras geradas pelo método WM utilizassem a condição *"don't care"*, a taxa de classificação seria maior e a quantidade de regras geradas bem menor.

5.4 Considerações Finais

Neste capítulo foram descritos os diferentes testes realizados com a metodologia proposta em alguns conjuntos de dados para classificação de padrões. No capítulo seguinte é apresentado um parecer sobre os resultados obtidos, além das conclusões a respeito do trabalho.

Capítulo

6 *Conclusões*

A construção da base de regras de sistemas *fuzzy* pode ser considerada em muitos casos como um processo de busca no espaço das possíveis soluções. Por isso as características dos algoritmos genéticos os tornam apropriados para tal tarefa, uma vez que estes algoritmos são robustos e possuem a capacidade de percorrer de forma global espaços de busca irregulares e extensos, encontrando uma solução ótima ou quase ótima rapidamente.

O principal objetivo deste trabalho foi investigar a geração automática de regras *fuzzy* por algoritmo genéticos, em especial, utilizando a abordagem Pittsburgh.

Foi desenvolvida e apresentada uma metodologia que utiliza algoritmos genéticos para geração automática de regras *fuzzy* a partir de um conjunto de exemplos para problemas de classificação de padrões. A metodologia é composta de 2 etapas. A primeira é a geração genética de regras que utiliza a abordagem Pittsburgh e, portanto, toda a base de regras é codificada em cada cromossomo da população. A segunda etapa é a otimização genética da base de regras obtida, que visa excluir regras redundantes e desnecessárias.

Apesar da complexidade de busca aumentar em função da extensão dos cromossomos, uma vantagem justifica e torna viável a utilização da abordagem Pittsburgh: a possibilidade de avaliar o desempenho de todo o conjunto de regras na função de aptidão, evitando o problema de competição/cooperação entre as regras.

Para verificar o desempenho da metodologia apresentada, foram realizados alguns experimentos em 5 problemas de classificação de padrões. Bases de regras compactas foram geradas com alto poder de classificação e com a característica de possuírem regras curtas e

facilmente compreensíveis, fato este proporcionado pela presença da condição de *"don't care"*.

Ficou evidente que a versão adaptativa do algoritmo genético alcançou melhores resultados que o algoritmo tradicional, em relação ao desempenho das bases de regras geradas e complexidade do espaço de busca. Os resultados mostraram que a mudança dinâmica dos valores das taxas de cruzamento e mutação aumentou o desempenho do algoritmo, garantindo diversidade genética e evitando a convergência prematura.

O método heurístico para gerar a população inicial do algoritmo da primeira etapa contribuiu para que o desempenho do algoritmo fosse bom desde as primeiras gerações.

Os resultados mostraram também que, para os casos estudados, independente da partição *fuzzy* escolhida para representar as variáveis, o algoritmo consegue gerar bases de regras acuradas e compactas. O mesmo foi observado quanto ao método de raciocínio empregado. Independente da forma de raciocínio utilizada, o algoritmo se comporta muito bem e não apresenta grandes variações nos resultados.

Ao comparar as bases de regras obtidas por algoritmos genéticos com regras obtidas pelo método de WM, as duas técnicas apresentaram desempenho muito próximos, em termos de acuidade. Entretanto, em termos de interpretabilidade das bases de regras geradas, a superioridade da metodologia proposta é indiscutível. Enquanto ela gerou 13 regras para um determinado caso estudado, WM gerou 186 regras.

Os experimentos relativos à aplicação da etapa de otimização do conjunto de regras obtido por outro método permite concluir que o algoritmo de otimização pode ser usado com qualquer outro método de geração de regras e apresentar bons resultados.

Os bons resultados alcançados nos casos estudados mostraram que a metodologia é robusta e pode ser aplicável a problemas de classificação do mundo real.

6.1 Trabalhos Futuros

Embora a eficiência da abordagem proposta tenha sido comprovada através de inúmeros testes e simulações, algumas etapas ainda requerem experimentos e estudos mais avançados e

deverão ser tratadas em trabalhos posteriores.

Um aspecto que permanece em aberto e merece ser estudado futuramente é em relação à dimensionalidade do conjunto de dados. O fato da abordagem de aprendizado utilizada ser Pittsburgh, faz com que o desempenho do algoritmo seja reduzido frente a problemas com elevado número de variáveis. Para os conjuntos de dados utilizados nos experimentos neste trabalho e com a ajuda de alguns mecanismos (versão adaptativa do AG, método heurístico para geração de regras e uso do *"don't care"*), os resultados obtidos foram satisfatórios. Entretanto, até que ponto a metodologia se torna viável? Será que ela consegue obter bons resultados para conjuntos de dados com dimensionalidade maior que os apresentados neste trabalho?

Um próximo passo é aplicar a metodologia em problemas de classificação com elevado número de variáveis e observar seu comportamento. Se o desempenho for considerado baixo, pretende-se incluir uma etapa inicial de pré-processamento do conjunto de dados, com a finalidade de identificar e eliminar as variáveis irrelevantes para a classificação, reduzindo a dimensionalidade do problema. Estudos preliminares sobre o assunto podem ser encontrados em [Castro et al., 2004b].

Outros pontos importantes a serem considerados em trabalhos futuros são:

- Investigar melhor o ajuste automático de valores de parâmetros para o algoritmo genético, incluindo a possibilidade de acoplar técnicas de Inteligência Computacional para tal tarefa.

- Dar maior ênfase à interpretabilidade das regras, incluindo esta condição na própria função de aptidão.

- Os algoritmos genéticos utilizados na metodologia são bastante simples, do ponto de vista dos operadores genéticos empregados. Não foi necessário desenvolver nenhum operador especial, pois os operadores básicos encontrados na literatura se mostraram adequados. Entretanto, seria interessante testar os algoritmos com outros tipos de operadores.

- Comparar os resultados obtidos com outras metodologias genéticas para geração de regras *fuzzy* encontradas na literatura e, até mesmo, com o paradigma neural.

Apêndice

A *Método de Wang e Mendel*

O método de Wang e Mendel (WM) para geração de regras *fuzzy* é bastante conhecido e parte de um conjunto de exemplos de treinamento para extrair o conhecimento necessário. Assumindo que a partição *fuzzy* das variáveis é conhecida, a geração das regras é realizada pelos seguinte passos, adaptada de [Cordón et al., 2001e]:

- Para cada exemplo E_h, h=1..p, construir uma regra *fuzzy* que tenha maior compatibilidade com o exemplo. Isto é feito atribuindo para cada atributo do exemplo o conjunto *fuzzy* em que ele tem maior grau de pertinência. Nesse momento p regras serão geradas.

- Dar um grau de importância para cada regra gerada. Seja R_1: Se x_1 é A_1 e ... e X_n é A_n, Então Y é B a regra gerada pelo exemplo $E_1 = (a_1,...,a_n, y)$. Logo, o grau de importância dessa regra será dado por:

$$\mathrm{Gr}(R_k) = \mathrm{T}\ (\mu_{A_1}(a_1),\ \mu_{A_2}(a_2),\ ...,\ \mu_{A_n}(a_n),\ \mu_B(\mathrm{y})) \tag{A.1}$$

- Compor a base de regras final adicionando as regras geradas. Se, dentre as p regras, duas ou mais possuírem antecedentes iguais, aquela que tiver maior grau de importância será escolhida para formar a base de regras final.

Apêndice

B

Algoritmo de Agrupamento Fuzzy FC-Means

O algoritmo de agrupameanto (*clusterização*) *fuzzy* FC-Means, desenvolvido e apresentado em [Bezdek, 1981], permite que um objeto pertença simultaneamente a mais que um grupo (*cluster*) com diferentes graus de pertinência.

A partir de um conjunto de N objetos x_1,x_2,...,x_N, determinar o número de grupos, denotado por **c**. Definir o valor para o parâmetro m, que é o parâmetro de fuzificação. Geralmente, m=2. Se m=1, então tem-se o método de agrupamento *crisp*. Estabelecer o valor de ϵ, um parâmetro que será usado para comparar as matrizes de pertinência $U[\mathrm{u}_{ij}]^t$ e $U[\mathrm{u}_{ij}]^{t-1}$, sendo t a iteração atual.

O foco do algoritmo é a matriz de pertinência $U[u_{ij}]$ consistindo de c linhas e N colunas. Desta forma u_{ij} representa a pertinência do objeto j no grupo i.

O algoritmo é iterativo e obedece os seguintes passos:

Passo 1: Inicializar a matriz de pertinência $U[\mathrm{u}_{ij}]^0$, respeitando a restrição:

$$\sum_{i=1}^{c} u_{ij} = 1 \tag{B.1}$$

Passo 2: Calcular os centros dos grupos usando (B.2):

$$\mathrm{c}_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m} \tag{B.2}$$

Passo 3: Atualizar a matriz de pertinência pela equação (B.3):

$$\mathrm{u}_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| x_i - c_j \right\|^2}{\|x_i - c_k\|^2} \right)^{2/m-1}} \tag{B.3}$$

em que $\|*\|$ é a norma euclidiana.

Passo 4: Se $\left\| U^{(t)} - U^{(t-1)} \right\| < \epsilon$, então parar a iteração. Caso contrário, voltar ao Passo 2.

Referências Bibliográficas

[Abe & Thawonmas, 1997] Abe, S.; Thawonmas, R. "A fuzzy classifier with ellipsoidal regions". *IEEE Transactions on Fuzzy Systems* **5**(3), pp. 358–368, 1997.

[Baker, 1985] Baker, J. E. "Adaptive selection methods for genetic algorithms". In *Proceedings of the 1st IEEE International Conference on Genetic Algorithms*, pp. 101–111, 1985.

[Baker, 1987] Baker, J. E. "Reducing bias and inefficiency in the selection algorithm". In *Proc. of the Second International Conference on Genetic Algorithm and Their Applications*, pp. 14–21, Hillsdale - New Jersey, 1987.

[Baraldi & Blonda, 1999a] Baraldi, A.; Blonda, P. "A survey of fuzzy clustering algorithms for pattern recognition - Part I". *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics* **29**, pp. 778–785, 1999.

[Baraldi & Blonda, 1999b] Baraldi, A.; Blonda, P. "A survey of fuzzy clustering algorithms for pattern recognition - Part II". *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics* **29**, pp. 786–801, 1999.

[Bäck, 1994] Bäck, T. "Seletive pressure in evolutionary algorithms: A characterization of selection mechanisms". In *Proceedings of the 1st IEEE International Conference on Evolutionay Computation*, volume 1, pp. 57–62, 1994.

[Bäck et al., 1993] Bäck, T.; Rudolph, G.; Schwefel, H. P. "Evolutionary programming and evolution strategies: Similarities and differences". In *Proceedings of the Second Annual Conference on Evolutionary Programming*, pp. 11–22, 1993.

[Bäck & Schwefel, 1995] Bäck, T.; Schwefel, H. P. "Evolution strategies I: Variants and their computational implementation". In Periaux, J.; Winter, G.; Galán, M.; Cuesta, P. (eds), *Genetic Algorithms in Engineering and Computer Science*. John Wiley and Sons, 1995.

[Berg et al., 2002] Berg, J.; Kaymak, U.; Bergh, W. "Fuzzy classification using probability-based rule weighting". In *Proceedings of the 11th IEEE International Conference on Fuzzy Systems*, Hawaii - USA, 2002.

[Bezdek, 1981] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

[Bishop, 1995] Bishop, C. M. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[Blake & Merz, 1998] Blake, C. L.; Merz, C. J. UCI repository of machine learning databases [http://www.ics.uci.edu/m̃learn/mlrepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[Booker, 1987] Booker, L. "Improving search in genetic algorithms". In Davis, L. (ed), *Genetic Algorithms and Simulated Annealing*, pp. 61–73. Morgan Kaufmann Publishers, 1987.

[Camargo et al., 2004] Camargo, H. A.; Pires, M. G.; Castro, P. A. D. "Genetic design of fuzzy knowledge bases - a study of different approaches". In *23rd IEEE International Conference of NAFIPS*, volume 2, pp. 954–959, Alberta, Canadá, 2004.

[Carse et al., 1996] Carse, B.; Fogarty, T.; Munro, A. "Evolving fuzzy rule based controllers using genetic algorithms". *Fuzzy Sets and Systems* **80**(3), pp. 273–293, 1996.

[Carse et al., 2003] Carse, B.; Pipe, A.; Renners, I.; Grauel, A.; Gómez-Skarmeta, A. F.; Jiménez, F.; Sánchez, G.; Cordón, O.; Herrera, F.; Gomide, F.; Walter, I.; González, A.; Pérez, R. "Current issues and future directions in evolutionary fuzzy systems research". In *3rd European Conference for Fuzzy Logic and Technology (EUSFLAT)*, 2003.

[Casillas et al., 2000] Casillas, J.; Cordón, O.; Herrera, F. "Improving the Wang and Mendel's fuzzy rule learning method by inducing cooperation among rules". In *Proceedings of the 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference*, pp. 1682–1688, Madrid - Spain, 2000.

[Castro & Camargo, 2004a] Castro, P. A. D.; Camargo, H. A. "A study of the reasoning methods impact on genetic learning and optimization of fuzzy rules". In *XVII Simpósio Brasileiro de Inteligência Artificial (SBIA)*, São Luis - Maranhão, 2004.

[Castro & Camargo, 2004b] Castro, P. A. D.; Camargo, H. A. "Geração automática de regras fuzzy usando algoritmos genéticos". Relatório Técnico DC 001/04, Universidade Federal de São Carlos, 2004.

[Castro & Camargo, 2004c] Castro, P. A. D.; Camargo, H. A. "Learning and optimization of fuzzy rule base by means of a self-adaptive genetic algorithm". In *Proceedings of IEEE International Conference on Fuzzy Systems*, volume 2, pp. 1037–1042, Budapeste - Hungria, 2004.

[Castro & Camargo, 2004d] Castro, P. A. D.; Camargo, H. A. Um paradigma baseado em algoritmos genéticos para o aprendizado de regras fuzzy. In *II Workshop de Teses e Dissertações em Inteligência Artificial(WTDIA)*, São Luis - Maranhão, 2004.

[Castro et al., 2003] Castro, P. A. D.; Pires, M. G.; Camargo, H. A. "Aprendizado e seleção de regras nebulosas usando algoritmos genéticos". In *VI Simpósio Brasileiro de Automação Inteligente (SBAI)*, pp. 970–975, 2003.

[Castro et al., 2004a] Castro, P. A. D.; Pires, M. G.; Camargo, H. A.; Morandin Jr., O.; Kato, E. R. R. "Genetic learning of fuzzy rules applied to sequencing problem of fms". In *IEEE Trans. on Systems, Man, and Cybernetics*, The Hague - Holanda, 2004.

[Castro et al., 2004b] Castro, P. A. D.; Santoro, D. M.; Camargo, H. A.; Nicoletti, M. C. "Improving a pittsburgh learnt fuzzy rule base using feature subset selection". In *4th International Conference on Hybrid Intelligent Systems*, Kitakyushu - Japan, 2004.

[Chi et al., 1996] Chi, Z.; Yan, H.; T.Pham. *Fuzzy Algorithms with Applications to Image Processing and Patern Recognition*. World Scientific, 1996.

[Corcoran & Sen, 1994] Corcoran, A. L.; Sen, S. "Using real-valued genetic algorithms to evolve rule sets for classification". In *Proceedings of the 1st IEEE International Conference on Evolutionay Computation*, pp. 120–124, 1994.

[Cordón et al., 1999] Cordón, O.; del Jesus, M. J.; Herrera, F. "A proposal on reasoning methods in fuzzy rule-based classification systems". *International Journal of Approximate Reasoning* **20**, pp. 21–45, 1999.

[Cordón & Herrera, 1996] Cordón, O.; Herrera, F. "A hybrid genetic algorithm-evolution strategy process for learning fuzzy logic controller knowledge bases". In Herrera, F.; Verdegay, J. (eds), *Genetic Algorithms and Soft Computing*. Physica-Verlag, 1996.

[Cordón & Herrera, 2001] Cordón, O.; Herrera, F. "Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems". *Fuzzy Sets and Systems* **118**, pp. 235–255, 2001.

[Cordón et al., 2001a] Cordón, O.; Herrera, F.; Gomide, F.; Hoffmann, F.; Magdalena, L. "Ten years of genetic-fuzzy systems: a current framework and new trends". In *Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pp. 1241–1246, Vancouver - Canada, 2001.

[Cordón et al., 2001b] Cordón, O.; Herrera, F.; Hoffman, F.; Magdalena, L. *Genetic Fuzzy Systems - Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, 2001.

[Cordón et al., 2001c] Cordón, O.; Herrera, F.; Hoffmann, F.; Magdalena, L. "Recent advances in genetic fuzzy systems". *Journal of Information Sciences* **136**, pp. 1–5, 2001.

[Cordón et al., 2001d] Cordón, O.; Herrera, F.; Magdalena, L.; Villar, P. "A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base". *Journal of Information Sciences* **136**, pp. 85–107, 2001.

[Cordón et al., 2001e] Cordón, O.; Herrera, F.; Villar, P. "Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base". *IEEE Transactions on Fuzzy Systems* **9**, pp. 667–674, 2001.

[De Jong, 1975] De Jong, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*. PhD. Thesis, University of Michigan, 1975.

[Delgado et al., 2001] Delgado, M.; Von Zuben, F.; Gomide, F. "Hierarchical genetic fuzzy systems.". *Information Sciences* **136**(1-4), pp. 29–52, 2001.

[Delgado, 2002] Delgado, M. R. *Automatic Design of Fuzzy Systems: A Co-evolutionary Approach*. Tese de Doutorado, Universidade Estadual de Campinas, 2002.

[Duda & Hart, 1973] Duda, R. O.; Hart, P. E. *Pattern classification and Scene Analysis*. John Wiley, 1973.

[Fogarty, 1989] Fogarty, T. C. "Varying the probability of mutation in the genetic algorithm". In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 104–109, 1989.

[Fogel, 1991] Fogel, D. B. *System Identification Through Simulated Evolution*. Ginn Press, 1991.

[Fogel et al., 1966] Fogel, L. J.; Owens, A. J.; Walsh, M. J. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[Goldberg, 1989] Goldberg, D. E. *Genetic Algorithms in Search, Optimizationm, and Machine Learning*. Addison-Wesley, 1989.

[Goldberg & Deb, 1991] Goldberg, D. E.; Deb, K. A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G. (ed), *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[González & Pérez, 1999] González, A.; Pérez, R. "SLAVE: A genetic learning system based on iterative approach". *IEEE Transactions on Fuzzy Systems* **7**(2), pp. 176–191, 1999.

[Grefenstette, 1986] Grefenstette, J. J. "Optimization of control parameters for genetic algorithms". *IEEE Trans. on Systems, Man, and Cybernetics* **16**(1), pp. 122–128, 1986.

[Grefenstette, 1987] Grefenstette, J. J. "Incorporating problem specific knowledge into genetic algorithms". In Davis, L. (ed), *Genetic Algorithms and Simulated Annealing*, pp. 42–60. Morgan Kaufmann Publishers, 1987.

[Gudwin et al., 1998] Gudwin, R.; Gomide, F.; Pedrycz, W. "Context adaptation in fuzzy processing and genetic algorithms". *International Journal of Intelligent Systems* **13**, pp. 929–948, 1998.

[Herrera & Lozano, 1996] Herrera, F.; Lozano, M. "Adaptation of genetic algorithm parameters based on fuzzy logic controllers". In Herrera, F.; Verdegay, J. L. (eds), *Genetic Algorithms and Soft Computing*, pp. 95–125. Physica-Verlag, 1996.

[Herrera et al., 1995a] Herrera, F.; Lozano, M.; Verdegay, J. L. "Generating rules from examples using genetic algorithms". In Bouchon, B.; Yager, R.; Zadeh, L. (eds), *Fuzzy Logic and Soft Computing*. Word Scientific, 1995.

[Herrera et al., 1995b] Herrera, F.; Lozano, M.; Verdegay, J. L. "Tuning fuzzy logic controllers by genetic algorithms". *International Journal of Approximate Reasoning* **12**, pp. 299–315, 1995.

[Herrera et al., 1996] Herrera, F.; Lozano, M.; Verdegay, J. L. "Dynamic and heuristic fuzzy connectives based crossover operators for controlling the diversity and convergence of real coded genetic algorithms". *International Journal of Intelligent Systems* **11**, pp. 1013–1041, 1996.

[Hoffman et al., 2002] Hoffman, F.; Baesens, B.; Martens, J.; Put, F.; Vanthienen, J. "Comparing a genetic fuzzy and a neurofuzzy classifier for credit scoring". *International Journal of Intelligent Systems* **17**(11), pp. 1067–1083, 2002.

[Hoffmeister & Bäck, 1991] Hoffmeister, F.; Bäck, T. "Genetic algorithms and evolution strategies". In *1st International Conference on Parallel Problem Solving from Nature*, volume 496, pp. 455–470, 1991.

[Holland, 1975] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[Homaifar & McCormick, 1995] Homaifar, A.; McCormick, E. "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms". *IEEE Transactions on Fuzzy Systems* **3**, pp. 129–139, 1995.

[Ishibuchi et al., 1997] Ishibuchi, H.; Murata, T.; Turksen, I. B. "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems". *Fuzzy Sets and Systems* **89**, pp. 134–150, 1997.

[Ishibuchi et al., 1999a] Ishibuchi, H.; Nakashima, T.; Morisawa, T. "Voting fuzzy rule-based systems for pattern classification problems". *Fuzzy Sets and Systems* **103**, pp. 223–238, 1999.

[Ishibuchi et al., 1999b] Ishibuchi, H.; Nakashima, T.; Murata, T. "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems". *IEEE Transactions on Fuzzy Systems* **29**, pp. 601–618, 1999.

[Karr, 1991] Karr, C. L. "Genetic algorithm for fuzzy controllers". *AI Expert*, pp. 26–33, 1991.

[Kecman, 2001] Kecman, V. *Learning and Soft Computing*. MIT Press, 2001.

[Klir & Yuan, 1995] Klir, G.; Yuan, B. *Fuzzy sets and Fuzzy Logic - Theory and Applications*. Prentice-Hall, 1995.

[Lee & Takagi, 1993] Lee, M. A.; Takagi, H. "Dynamic control of genetic algorithms using fuzzy logic techniques". In *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 76–83, 1993.

[Liao et al.,] Liao, T. W.; Celmins, A. K.; Hammell, R. J. "A fuzzy C-Means variant for the generation of fuzzy term sets". *Fuzzy Sets and Systems* **135**(2).

[Louis & Rawlins, 1993] Louis, S. J.; Rawlins, G. J. E. "Syntactic analysis of convergence in genetic algorithms". In Whitley, L. D. (ed), *Foundations of Genetic Algorithms 2*, pp. 141–151. Morgan Kaufmann Publishers, 1993.

[Magdalena & Velasco, 1996] Magdalena, L.; Velasco, J. R. "Fuzzy rules-based controllers that learn by evolving its knowledge base". In Herrera, F.; Verdegay, J. L. (eds), *Fuzzy Logic and Soft Computing*. Physica-Verlag, 1996.

[Magdalena & Velasco, 1997] Magdalena, L.; Velasco, J. R. "Evolutionary based learning of fuzzy controllers". In Pedrycz, W. (ed), *Fuzzy Evolutionary Computation*. Kluwer Academic Publisher, 1997.

[Mamdani, 1977] Mamdani, E. H. "Application of fuzzy logic to approximate reasoning using linguist systems". *Fuzzy Sets and Systems* **26**, pp. 1182–1191, 1977.

[Michalewicz, 1996] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[Mitchell, 1996] Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[Nakaoka et al., 1994] Nakaoka, K.; Furuhashi, T.; Uchikawa, Y. "A study on apportionment of credits of fuzzy classifier system for knowledge acquisition of large scale systems.". In

Proceedings of the 3rd IEEE International Conference on Fuzzy Systems, pp. 1797–1800, Orlando - USA, 1994.

[Nauck & Cruse, 1997] Nauck, D.; Cruse, R. "A neuro-fuzzy method to learn fuzzy classification rules from data". *Fuzzy Sets and Systems* **89**, pp. 277–288, 1997.

[Nomura et al., 1992] Nomura, H.; Hayashi, L.; Wakami, N. "A learning method of fuzzy inference rules by descent method". In *Proceedings of the 1st IEEE International Conference on Fuzzy Systems*, pp. 203–210, San Diego - USA, 1992.

[Parodi & Bonelli, 1993] Parodi, A.; Bonelli, P. "A new approach to fuzzy classifier systems". In *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 223–230, San Mateo - USA, 1993.

[Pedrycz & Gomide, 1998] Pedrycz, W.; Gomide, F. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, 1998.

[Rechenberg, 1973] Rechenberg, I. *"Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.

[Schraudolph & Belew, 1992] Schraudolph, N. N.; Belew, R. K. "Dynamic parameter encoding for genetic algorithms". *Machine Learning* **9**, pp. 9–21, 1992.

[Sirinivas & Patnaik, 1994] Sirinivas, M.; Patnaik, L. M. "Adaptive probabilities of crossover and mutation in genetic algorithms". *IEEE Trans. on Systems, Man, and Cybernetics* **24**(4), pp. 656–667, 1994.

[Takagi & Sugeno, 1985] Takagi, T.; Sugeno, M. "Fuzzy identification of systems and its application to modeling and control". *IEEE Trans. on Systems, Man, and Cybernetics* **15**(1), pp. 116–132, 1985.

[Wang & Mendel, 1992] Wang, L. X.; Mendel, J. M. "Generating fuzzy rules by learning from examples". *IEEE Trans. on Systems, Man, and Cybernetics* **25**(2), pp. 353–361, 1992.

[Whitley et al., 1991] Whitley, D.; Mathias, K.; Fitzhorn, P. "Delta coding: an iterative search strategy for genetic algorithms". In *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 77–84, 1991.

[Yuan & Zhuang, 1996] Yuan, Y.; Zhuang, H. "A genetic algorithm for generating fuzzy classification rules". *Fuzzy Sets and Systems* **84**(4), pp. 1–19, 1996.

[Zadeh, 1965] Zadeh, L. A. "Fuzzy sets". *Information and Control* **8**(3), pp. 338–353, 1965.

[Zadeh, 1973] Zadeh, L. A. "Outline of a new approach to the analysis of complex system and decision process". *IEEE Trans. on Systems, Man, and Cybernetics* **3**, pp. 28–44, 1973.

[Zimmermann, 1991] Zimmermann, H. J. *Fuzzy Set Theory and its Applications.* Kluwer Academic Publishers, 1991.