

Universidade Federal de Minas Gerais

Programa de Pós Graduação em Engenharia Elétrica

Classificação de Dados Híbridos Através de Algoritmos Evolucionários

Marconi de Arruda Pereira

Tese submetida à banca examinadora designada pelo colegiado do programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial à obtenção de título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. João Antônio de Vasconcelos (DEE/UFMG)

Co-Orientador: Prof. Dr. Clodoveu Augusto Davis Júnior (DCC/UFMG)

Linha de Pesquisa: Otimização

Belo Horizonte, 2012

Agradecimentos

Ao Deus Pai, que me guia e faz feliz, mesmo quando tudo parece não estar dando certo.

À Maria Fernanda e à Marília, por me proporcionarem grandes alegrias.

À minha família (Maria Fernanda, Marília, meus pais, minha irmã, Leandro, Dayze, Divino, Josinha e Rafael) por sempre me darem apoio e suporte quando preciso.

Aos amigos da Obra, em particular Ravic, Pe. Antônio, Pe. Luis, Pe. Marcos, Pe. Luciano e Pe. Zé Antônio, pelo apoio e conselhos nos momentos de dificuldades.

Aos professores João Vasconcelos e Clodoveu Davis Jr., por terem orientado de maneira tão efetiva e compreensiva.

Ao professor Eduardo Carrano pela grande ajuda com as avaliações estatísticas, que proporcionaram uma melhoria significativa ao trabalho.

Aos amigos do CEFET/MG, que sempre fizeram o possível para que eu pudesse me dedicar ao máximo a este trabalho de doutorado.

A todos vocês, de coração, muito obrigado.

Resumo

A mineração de dados tem se tornado uma aliada do tomador de decisão atual, tanto nas grandes quanto nas pequenas corporações: informações não triviais são identificadas de maneira a possibilitar correções e ajustes nas estratégias econômicas e administrativas. Aliando-se a este fato, vê-se um grande crescimento no uso de informações georreferenciadas, de maneira que a mineração de dados convencional já não é capaz de responder a todas às questões de uma corporação. A pesquisa bibliográfica realizada no contexto deste trabalho mostrou que ainda existem poucos métodos capazes de extrair conhecimento a partir de dados híbridos, principalmente quando se trata de dados convencionais (numéricos e textuais) e geográficos (pontos, linhas e polígonos). O objetivo principal deste trabalho consiste no desenvolvimento de novos algoritmos que sejam capazes de explorar todos os atributos, convencionais ou geográficos, de um banco de dados, visando extrair as informações relevantes. Os algoritmos evolucionários foram escolhidos como ponto de partida para a elaboração desses novos algoritmos, pelos seguintes motivos: (a) são flexíveis, uma vez que podem ser aplicados em diversos contextos; (b) são robustos, pois tendem a explorar satisfatoriamente o espaço de busca, encontrando soluções viáveis. Para se alcançar o objetivo proposto, foi elaborado um primeiro algoritmo de obtenção de regras de classificação (NGAE), baseado num Algoritmo Genético, que pode ser aplicado em bancos de dados compostos por dados numéricos. Posteriormente, foi elaborado outro algoritmo (DMGeo), baseado na Programação Genética, que visa obter regras de classificação de padrões que possuem atributos numéricos e atributos geográficos. Finalmente, o DMGeo foi estendido para uma versão multiobjetivo mais robusta e eficiente, chamada MDMGeo. Todos os algoritmos propostos foram comparados com outros algoritmos eficientes na tarefa de classificação de dados, utilizando bancos de dados benchmark além de bancos reais. Os experimentos realizados mostram que o resultado final obtido é um conjunto de ferramentas robusto e eficiente, em particular, quando aplicado sobre dados híbridos.

Abstract

Data mining has become an ally of the decision maker today, in both large and small corporations: nontrivial information is identified in order to allow corrections and adjustments in the economic and administrative actions. Moreover, we can see an increasing use of georeferenced information so that conventional data mining is not able to answer all the questions of a corporation. A survey of geographical data mining showed that there are few tools capable of extracting knowledge from georeferenced data, especially when it comes from a database storing conventional (numerical and textual) and geographical (points, lines and polygons) data. The main objective of this work is to develop new algorithms that are able to explore all the attributes, conventional and geographical, of a database in order to extract relevant information. The evolutionary algorithms were chosen as a starting point for the development of these new algorithms by the following reasons: (a) they are flexible, since they can be applied in different contexts, (b) they are robust, they tend to adequately explore the searching space, finding viable solutions. To achieve our objective the first algorithm described in the thesis obtains classification rules (NGAE) based on a genetic algorithm, which can be applied to databases storing numeric data. The second algorithm (DMGeo) is based on genetic programming, which aims to obtain classification rules for patterns that have numerical and spatial attributes. Finally, DMGeo has progressed to a multiobjective version, more robust and efficient, called MDMGeo. All proposed algorithms were compared with other efficient algorithms applied to classification problems, using benchmark datasets and real datasets. Experiments show that the final result is a set of robust and efficient tools, in particular, when applied to a database composed by hybrid attributes.

Contribuições

As contribuições obtidas por esse trabalho estão identificadas nos tópicos a seguir.

Publicações

Pereira, M.A.; Vasconcelos, J.A. “**A Niche Genetic Algorithm for Discovery Classification Rules in Real Databases**”. In: Congresso Brasileiro de Automática 2010 - CBA2010, Bonito, 2010.

De Arruda Pereira, M., Davis Jr., C.A., De Vasconcelos, J.A. - **A niched genetic programming algorithm for classification rules discovery in geographic databases**. Lecture Notes in Computer Science V. 6457, pp. 260-269, 2010.

Pereira, M.A. ; Vasconcelos, J.A. “**Algoritmo de Recozimento Simulado (Simulated Annealing Algorithm)**”. In: António Gaspar-Cunha, Ricardo H. C. Takahashi e Carlos Henggeler Antunes. (Org.). Manual de Computação Evolutiva e Metaheurística. Coimbra, Portugal: Editora da Universidade de Coimbra, 2012.

Algoritmos

- NGAE (*Niche Genetic Algorithm with Elitism*), detalhado no Capítulo 4.
- DMGeo (*Niche Genetic Programming Algorithm Data Mining in Geographic Databases*), detalhado no Capítulo 5.
- MDMGeo (*Multiobjective Adaptive Niche Genetic Programming Algorithm for Data Mining*), detalhado no Capítulo 6.

Outras Contribuições

Esse trabalho introduz uma nova linha de pesquisa, Mineração de Dados Híbridos, ainda não investigada em trabalhos desenvolvidos no âmbito do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais.

Nomenclatura

Acrônimos

AE	Algoritmo Evolucionário
DT	Árvore de Decisão
AG	Algoritmo Genético
DE-9IM	Modelo de 9 Interseções Dimensionalmente Estendido
FN	<i>False Negative</i>
FP	<i>False Positive</i>
KDD	<i>Knowledge Discovery in Databases</i>
MD	Mineração de Dados
PG	Programação Genética
RNA	Rede Neural Artificial
SIG	Sistema de Informações Geográficas
SVM	<i>Support Vector Machine</i>
SQL	<i>Structured Query Language</i>
TN	<i>True Negative</i>
TP	<i>True Positive</i>

Algoritmos

DMGeo	<i>Niched Genetic Programming Algorithm for Geographic Data Mining</i>
MDMGeo	<i>Multiobjective Adaptative Niched Genetic Programming Algorithm</i>
NGAE	<i>Niched Genetic Algorithm with Elitism</i>

Índice Geral

1	Introdução	1
1.1	Características dos Métodos de Classificação de Dados Híbridos	1
1.2	Objetivos do Trabalho	2
1.3	Organização do Documento	2
2	Conceitos e Revisão Bibliográfica	4
2.1	Introdução	4
2.2	Mineração de Dados	4
2.2.1	Histórico	4
2.2.2	Fundamentos e Definições.....	5
2.3	Representações Geográficas e Relacionamentos Topológicos.....	9
2.3.1	Definições	9
2.3.2	Relacionamentos Topológicos.....	12
2.4	Algoritmos Evolucionários.....	17
2.4.1	Introdução	17
2.4.2	Conceitos Básicos e Estrutura	19
2.4.3	Algoritmos Genéticos	21
2.4.4	Programação Genética	24
2.5	Aplicação de Algoritmos Evolucionários em Mineração de Dados.....	27
2.5.1	Representação dos Indivíduos	29
2.6	Trabalhos Relacionados.....	37
2.7	Mineração de Dados Espaciais	42
2.8	Conclusão	44
3	Algoritmos Não evolucionários aplicados à classificação de dados	46
3.1	Introdução	46
3.2	Redes Neurais com Função de Base Radial (RBF)	47
3.3	Máquinas de Vetores Suporte (Support Vector Machine - SVM).....	48
3.4	Árvore de Decisão	51
3.5	Conclusão	54
4	O Algoritmo Genético NGAE para classificação	55
4.1	Introdução	55
4.2	Algoritmo Genético NGAE	56
4.2.1	Codificação do Indivíduo	57

4.2.2	Avaliação da <i>Fitness</i>	58
4.2.3	Mutação	60
4.2.4	Cruzamento.....	61
4.2.5	Nicho e Elitismo	62
4.2.6	Melhorias Implementadas para Avaliação dos Indivíduos.....	62
4.3	Experimento e Resultados	63
4.4	Conclusão	67
5	Programação Genética em Mineração de Dados Híbridos	68
5.1	Introdução	68
5.2	Algoritmo DMGeo	68
5.2.1	Indivíduo.....	68
5.2.2	Avaliação da <i>Fitness</i>	71
5.2.3	Mutação	72
5.2.4	Cruzamento.....	73
5.2.5	Melhorias na Avaliação dos Indivíduos	73
5.3	Experimento e Resultados	74
5.3.1	Problemas Utilizados.....	74
5.3.2	Resultados e Análise.....	75
5.4	Conclusão	77
6	Algoritmo Multi-objetivo Baseado na Programação Genética e Nichos para Classificação de Dados Híbridos.....	79
6.1	Introdução.....	79
6.2	Modelagens do Problema e do Algoritmo.....	80
6.3	O Indivíduo.....	81
6.4	Avaliação das Regras.....	82
6.4.1	Vetor de Funções objetivo.....	82
6.5	População Arquivo - Elitismo Global	86
6.6	Nichos e Controle de Diversidade Genética.....	86
6.7	Mecanismo de Classificação	87
6.8	Resultados Experimentais	88
6.9	Conclusão	96
7	conclusões e trabalhos futuros.....	98
	Anexo.....	101
	Referências Bibliográficas.....	107

Índice de Figuras

Figura 2.1 – Etapas do KDD (HAN; KAMBER, 2001).....	6
Figura 2.2 – Matriz de 4-Interseções para relações entre duas regiões (DAVIS Jr; QUEIROZ, 2005 <i>apud</i> EGENHOFER et al., 1994).	13
Figura 2.3 – Matriz de 9-Interseções para Relações entre duas regiões. (DAVIS Jr; QUEIROZ, 2005 <i>apud</i> EGENHOFER; HERRING, 1991).....	14
Figura 2.4 - Matriz resultante do relacionamento entre dois objetos no DE-9IM. Fonte: (STROBL, 2010)	14
Figura 2.5 - Exemplo do relacionamento Equals no DE-9IM. Fonte: (STROBL, 2010).....	15
Figura 2.6 - Exemplo do relacionamento Disjoint no DE-9IM. Fonte: (STROBL, 2010).....	16
Figura 2.7 - Exemplo do relacionamento Intersects no DE-9IM. Fonte: (STROBL, 2010) ..	16
Figura 2.8 - Exemplo do relacionamento Touches no DE-9IM. Fonte: (STROBL, 2010)....	16
Figura 2.9 - Exemplo do relacionamento Crosses no DE-9IM. Fonte: (STROBL, 2010)	16
Figura 2.10 - Exemplo do relacionamento Overlaps no DE-9IM. Fonte: (STROBL, 2010) .	17
Figura 2.11 - Exemplo do relacionamento within no DE-9IM. Fonte: (STROBL, 2010)	17
Figura 2.12 - Exemplo do relacionamento Contais no DE-9IM. Fonte: (STROBL, 2010) ...	17
Figura 2.13 – Técnicas de Otimização Global (COELLO et al., 2007).	19
Figura 2.14 – Exemplo de Indivíduo na PG.	25
Figura 2.15 – Pontos obtidos pelo AG. O gráfico exibido à esquerda mostra o resultado do algoritmo <u>com</u> utilização de técnica de nicho. O gráfico da direita mostra o resultado <u>sem</u> a técnica (GOLDBERG, 1989, p. 193).	30
Figura 2.16 - Exemplo de cruzamento onde a propriedade de fechamento não é observada.	34
Figura 2.17 – Exemplo de Indivíduo da PG com os terminais do tipo <i>booleano</i> (FREITAS, 2002).....	35
Figura 2.18 – Exemplo de Indivíduo da PG Fortemente Tipada.....	36
Figura 2.19 – Representação da regra (SRINIVASA et al., 2007).....	39
Figura 2.20 – Cadeia de Caracteres binários de representação da regra (SRINIVASA et al., 2007).....	39
Figura 2.21 – Exemplo de indivíduo do Algoritmo Genético (SIKORA et al., 2007).....	40
Figura 3.1 - Conjunto de dados não linearmente separáveis. Fonte: (LORENA, CARVALHO, 2007).....	51

Figura 3.2 - Representação de um conjunto de regras de classificação na forma de Arvore de Decisão. Adaptado de (QUINLAN, 1986).	53
Figura 4.1 – Fluxo de dados do algoritmo NGAE.....	56
Figura 4.2 – Indivíduo do NGAE representando uma regra de classificação.	58
Figura 4.3 – Indivíduo do NGAE com cromossomos desabilitados.	58
Figura 4.4 – Indivíduo do NGAE após a operação de Mutação.....	60
Figura 5.1 – Representação de um indivíduo do DMGeo.	69

Índice de Tabelas

Tabela 2.1 - Relacionamentos topológicos e os respectivos significados no DE-9IM. Fonte: (STROBL, 2010)	15
Tabela 2.2 – Mapeamento da terminologia da genética natural para a terminologia utilizada em Algoritmos Genéticos (GOLDBERG, 1989, p. 22).....	18
Tabela 2.3 – Parâmetros e respectivos valores utilizados no algoritmo de manutenção da Diversidade Genética (VASCONCELOS et al., 2001).	23
Tabela 2.4 – Matriz de Confusão.	28
Tabela 2.5 – Exemplo de Matriz de Confusão gerada em um problema de classificação.	28
Tabela 2.6 – Definição dos tipos de dados de entrada e saída das funções.....	36
Tabela 3.1 - Dados do Problema do Clima.....	52
Tabela 4.1 – Elementos da Álgebra Relacional.....	59
Tabela 4.2 – Acurácia global de cada algoritmo.	64
Tabela 4.3 – Acurácia obtida em cada classe.	64
Tabela 4.4 - p -valor medido para a acurácia global dos algoritmos - Banco 1	66
Tabela 4.5 - Classificação dos algoritmos de acordo com o p -valor - Banco 1	66
Tabela 4.6 - p -valor medido para a acurácia global dos algoritmos - Banco 2	66
Tabela 4.7 - Classificação dos algoritmos de acordo com o p -valor - Banco 2	66
Tabela 4.8 - p -valor medido para a acurácia global dos algoritmos - Banco 3	66
Tabela 4.9 - Classificação dos algoritmos de acordo com o p -valor - Banco 3	66
Tabela 5.1 – Acurácia de cada algoritmo.	75
Tabela 5.2 – Acurácia obtida em cada classe	76
Tabela 5.3 - p -valor medido na comparação da acurácia global - <i>Vinho</i>	76
Tabela 5.4 - Classificação dos algoritmos - <i>Vinho</i>	76
Tabela 5.5 - p -valor medido na comparação da acurácia global - <i>Hepatite</i>	76
Tabela 5.6 - Classificação dos algoritmos - <i>Hepatite</i>	76
Tabela 5.7 - p -valor medido na comparação da acurácia global - <i>Infraestrutura</i>	77
Tabela 5.8 - Classificação dos algoritmos - <i>Infraestrutura</i>	77
Tabela 5.9 - p -valor medido na comparação da acurácia global - <i>Desenvolvimento urbano</i> . ..	77
Tabela 5.10 - Classificação dos algoritmos - <i>Desenvolvimento urbano</i>	77
Tabela 6.1 - Matriz de confusão obtida pelo primeiro classificador	83
Tabela 6.2 - Matriz de confusão obtida pelo segundo classificador.....	83

Tabela 6.3 - Avaliação dos classificadores baseada nos valores da sensibilidade e especificidade	83
Tabela 6.4 - Tamanho da população e número de gerações utilizados em cada banco.....	89
Tabela 6.5 – Configuração dos parâmetros dos algoritmos clássicos para cada um dos bancos	89
Tabela 6.6 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - hepatite	90
Tabela 6.7 - <i>p</i> -valor obtido na comparação utilizando acurácia global - hepatite.....	90
Tabela 6.8 - Classificação dos algoritmos de acordo com os dois critérios - hepatite	90
Tabela 6.9 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - vinho.....	91
Tabela 6.10 - <i>p</i> -valor obtido na comparação utilizando acurácia global - vinho	91
Tabela 6.11 - Classificação dos algoritmos de acordo com os dois critérios - vinho.....	91
Tabela 6.12 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - banco 1	92
Tabela 6.13 - <i>p</i> -valor obtido na comparação utilizando acurácia global - banco 1	92
Tabela 6.14 - Classificação dos algoritmos de acordo com os dois critérios - banco 1	92
Tabela 6.15 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - banco 3	92
Tabela 6.16 - <i>p</i> -valor obtido na comparação utilizando acurácia global - banco 3.....	93
Tabela 6.17 - Classificação dos algoritmos de acordo com os dois critérios - banco 3	93
Tabela 6.18 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - desenvolvimento urbano.....	93
Tabela 6.19 - <i>p</i> -valor obtido na comparação utilizando acurácia global - desenvolvimento urbano	93
Tabela 6.20 - Classificação dos algoritmos de acordo com os dois critérios - desenvolvimento urbano.....	94
Tabela 6.21 – Exemplos de regras geradas pelo MDMGeo no problema Desenvolvimento urbano	94
Tabela 6.22 – Exemplos de regras geradas pelo MDMGeo no problema Infraestrutura	94
Tabela 6.23 - <i>p</i> -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - infraestrutura	95
Tabela 6.24 - <i>p</i> -valor obtido na comparação utilizando acurácia global - infraestrutura.....	95
Tabela 6.25 - Classificação dos algoritmos de acordo com os dois critérios - infraestrutura	95

Tabela 6.26 - Resumo da performance dos algoritmos de acordo com a média do produto da sensibilidade x especificidade	95
Tabela 6.27 - Resumo da performance dos algoritmos de acordo com a média da acurácia global	96

Índice de Algoritmos

Algoritmo 2.1 – Estrutura básica dos algoritmos evolucionários.....	20
Algoritmo 2.2 – Estrutura básica dos Algoritmos Genéticos (GOLDBERG, 1989).....	21
Algoritmo 2.3 – Algoritmo de ajuste da diversidade genética (VASCONCELOS et al., 2001).	22
Algoritmo 2.4 – Pseudocódigo da operação de Mutação na PG Fortemente Tipada.	36
Algoritmo 2.5 – Pseudocódigo da operação de Cruzamento na PG Fortemente Tipada.	37
Algoritmo 4.1 – Pseudocódigo do NGAE.	56
Algoritmo 4.2 – Operador de Mutação do NGAE.....	61
Algoritmo 4.3 – Operador de Cruzamento do NGAE.	61
Algoritmo 5.1 – Pseudocódigo da geração de Indivíduos do DMGeo.....	70
Algoritmo 5.2 – Pseudocódigo do operador de Mutação do DMGeo.	72
Algoritmo 6.1 - Pseudocódigo do algoritmo de extração de regras	81
Algoritmo 6.2 - Pseudocódigo do Controle de Diversidade Genética	87
Algoritmo 7.1 - Geração de <i>join geográfico</i>	106

1 INTRODUÇÃO

O tema Classificação de Dados se firmou como um objeto de trabalho dos mais relevantes, tanto para a ciência básica quanto para a ciência aplicada, pois tem produzido formas diversas de se extrair conhecimento não trivial dos diferentes repositórios de dados, que se multiplicam a cada dia. Em particular, é cada vez maior o volume de repositórios compostos por dados híbridos, isto é, bancos de dados compostos por atributos convencionais (números e textos) e também por atributos não convencionais, tais como os georreferenciados, armazenados na forma de pontos, linhas e polígonos (maiores detalhes na seção 2.3.1.1).

Existem hoje muitos métodos que auxiliam a atividade de extração de regras (ou modelos) de classificação a partir de bancos de dados, tais como árvores de decisão, redes neurais artificiais e métodos evolucionários, como por exemplo, algoritmos genéticos e programação genética. Cabe destacar a simplicidade, robustez e eficiência que os métodos evolucionários apresentam. Em particular, este tipo de método vem sendo utilizado cada vez mais pela comunidade de classificação de dados (maiores detalhes na seção 2.5).

1.1 Características dos Métodos de Classificação de Dados Híbridos

Uma classificação de dados bem sucedida demanda algumas tarefas antes de executar o algoritmo de extração de informações:

- Definir exatamente que tipo de informações se quer extrair. Nesta etapa é fundamental o envolvimento das pessoas que entendem do domínio da aplicação da qual se quer extrair as informações;
- Preparar os dados, visando remover informações irrelevantes e que podem prejudicar a procura por informações mais importantes.

Levando-se em consideração estas tarefas, entende-se que um bom método de classificação de dados híbridos deve possuir as seguintes características:

- Tratar de maneira homogênea tanto os dados convencionais quanto os não convencionais, isto é, a ferramenta não deve demandar nenhuma

configuração ou pré-processamento para que se manipulem os dados não convencionais;

- Manipular os dados como um todo, procurando as correlações entre os atributos, independente do seu tipo.

Os métodos que se apresentam atualmente na literatura não são capazes, por exemplo, de identificar correlações entre dados geográficos e dados numéricos.

1.2 Objetivos do Trabalho

O objetivo principal deste trabalho de doutorado é conceber novos métodos evolucionários, mono e multiobjetivo, especialmente dedicados à classificação de dados híbridos, em particular dados espaciais. Para atender este objetivo geral, os objetivos específicos listados a seguir deverão ser alcançados:

- Conhecer e estudar as alternativas existentes para classificação de dados convencionais e dados geográficos;
- Conceber e desenvolver novos métodos evolucionários robustos e eficientes para a classificação de dados híbridos;
- Aplicar os métodos desenvolvidos em problemas reais e em *benchmarks* a fim de validá-los e compará-los com outros algoritmos da literatura.

1.3 Organização do Documento

Este documento está organizado em 7 capítulos. No capítulo 1 apresenta-se a introdução do texto, discutindo a relevância do tema e objetivos, geral e específicos. Em seguida, no capítulo 2 apresentam-se os principais conceitos envolvidos neste trabalho: mineração de dados, estruturas geográficas, relacionamentos topológicos e algoritmos evolucionários. Acrescenta-se a isso a revisão bibliográfica, a qual destaca diversos trabalhos que buscam realizar mineração de dados convencionais e espaciais, além de apresentar como os métodos evolucionários têm sido aplicados em mineração de dados, especificamente na tarefa de geração de regras de classificação.

No capítulo 3 apresentam-se os algoritmos não evolucionários de classificação de dados mais relevantes encontrados na literatura.

No capítulo 4 descreve-se em detalhes um algoritmo genético, chamado NGAE, proposto neste trabalho para a classificação de dados convencionais.

No capítulo 5, apresenta-se o segundo algoritmo proposto neste trabalho, chamado DMGeo, baseado na programação genética, destinado à classificação de dados geográficos e convencionais.

No capítulo 6, detalha-se o algoritmo MDMGeo, uma versão multiobjetivo e com diversas outras melhorias em relação ao DMGeo. Todos os capítulos referentes aos algoritmos propostos incluem uma comparação com outros algoritmos relevantes na literatura (e detalhados no capítulo 3), utilizando uma metodologia estatística consolidada, a fim de embasar corretamente a comparação.

As conclusões deste trabalho, bem como os direcionamentos para trabalhos futuros, encontram-se no capítulo 7.

2 CONCEITOS E REVISÃO BIBLIOGRÁFICA

2.1 Introdução

Os principais conceitos na área de classificação de dados, necessários à boa compreensão do texto, são apresentados neste capítulo. Inicia-se pelo conceito de mineração de dados e em seguida apresentam-se os conceitos de estruturas geográficas e relacionamentos topológicos, e algoritmos evolucionários. Além disso, os trabalhos mais relevantes e mais recentes nessas áreas são discutidos.

2.2 Mineração de Dados

Segundo Chen et al., (1996), a mineração de informações e conhecimento em grandes bancos de dados é considerada como um dos tópicos chave em pesquisas relacionadas a sistemas de bancos de dados e aprendizado de máquina¹. É também uma área importante para companhias industriais, pois pode ser usada para obtenção de informações que alavanquem as vendas dos seus produtos bem como para a descoberta de novas oportunidades de negócio. Esta seção apresenta um estudo sobre este assunto, identificando as técnicas que vêm sendo utilizadas atualmente, bem como as novas técnicas apresentadas nos principais periódicos e congressos relacionados ao assunto.

2.2.1 Histórico

Segundo Han e Kamber (2001), a partir dos anos 60, bancos de dados e tecnologias de armazenamento de informações começaram a ser utilizadas. Desde processamentos primitivos de arquivos até sistemas sofisticados começaram a ser utilizados. Pesquisas e desenvolvimentos na área de sistemas de banco de dados, originados na década de 70, permitiram o surgimento de sistemas de banco de dados relacionais, onde os dados eram gravados em tabelas e proviam-se métodos de modelagem de dados e técnicas de indexação e organização de dados. Foi também nesta época que surgiram as primeiras linguagens de consultas (o que trouxe flexibilidade de acesso aos dados), métodos de otimização de consulta e interfaces para usuários. Métodos eficientes para

¹ do inglês, *Machine Learning*.

*Processamento de Transação On-Line (OLTP*²), nos quais a consulta é vista como uma transação de somente-leitura, contribuíram muito para a evolução e ampla aceitação da tecnologia relacional como a maior ferramenta para um eficiente armazenamento, recuperação e gerenciamento de grandes volumes de dados.

Desde a década de 80, a tecnologia de banco de dados tem se caracterizado por uma adoção ampla do modelo relacional e uma crescente pesquisa e desenvolvimento em sistemas de bancos de dados, cada vez mais inovadores e robustos. Estes estudos originaram modelos de dados mais avançados, tais como o relacional estendido e o orientado a objetos. Surgiram também os modelos dedutivos, orientados às aplicações, incluindo os bancos de dados espaciais, temporais, multimídia, além dos bancos de dados para escritórios, que tiveram um amplo crescimento e grande aceitação do mercado.

Questões relacionadas à distribuição, diversificação e compartilhamento foram amplamente estudadas. Sistemas de bancos de dados heterogêneos e sistemas de informação baseados na internet também começaram a ser temas de pesquisa a partir desta época. Com a crescente evolução e queda nos preços do hardware, ferramentas automáticas de coleta de dados e tecnologias cada vez mais avançadas de banco de dados possibilitaram a existência de um grande conjunto de dados coletados e armazenados. Assim, obteve-se um grande volume de dados, mas com grande frequência, uma escassez de informações. A partir da década de 90 surgiram os métodos de mineração de dados e de *Data Warehouse*³ (armazém de dados) que permitiram extrair informações desses repositórios de dados.

2.2.2 Fundamentos e Definições

A mineração de dados pode ser entendida como “o processo de extração não trivial de conhecimento prévio implícito e demais informações potencialmente úteis, tais como regras de conhecimento, restrições, padrões a partir dos dados de um banco de dados” (PIATETSKY-SHAPIRO; FRAWLEY, 1991).

A mineração de dados faz parte de um processo maior chamado Descoberta de Conhecimento em Bancos de Dados, ou simplesmente KDD (*Knowledge Discovery in*

² *On-Line Transaction Processing*

³ Optou-se por utilizar o termo “*Data Warehouse*” sem tradução à partir deste ponto do texto por se tratar de um conceito conhecido pela comunidade de banco de dados. Entretanto, o termo foi traduzido na primeira vez em que foi citado, a fim de tornar este documento claro e didático para leitores que não são da área de banco de dados.

Databases). KDD é um processo iterativo, composto de uma sequência de fases:

- **Limpeza dos Dados:** Ruídos e dados irrelevantes são eliminados;
- **Integração de Dados:** Diversas fontes e bancos de dados distintos podem ser integrados a fim de se formar a massa de dados a ser trabalhada;
- **Seleção de Dados:** Os dados relevantes para a análise são destacados do banco de dados;
- **Transformação de Dados:** Os dados são transformados e consolidados em formas apropriadas para a mineração. Operações de agregação e composição dos dados são aplicadas nesta etapa;
- **Mineração de Dados:** Fase principal onde métodos inteligentes são aplicados a fim de se extrair padrões dos dados;
- **Avaliação de Padrões:** Os padrões dos dados, extraídos na etapa anterior, são avaliados a fim de se verificar quais representam conhecimentos relevantes para o domínio da aplicação;
- **Apresentação do Conhecimento:** Etapa final, onde o conhecimento extraído deve ser exibido para o usuário final.

A Figura 2.1 ilustra as etapas do KDD.

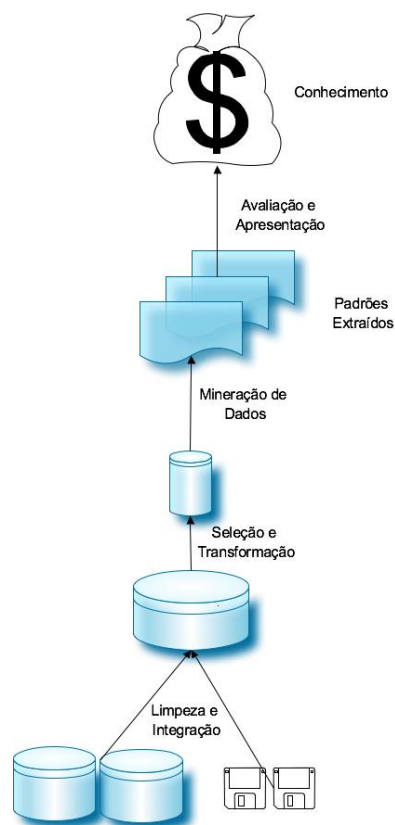


Figura 2.1 – Etapas do KDD (HAN; KAMBER, 2001).

Han e Kamber (2001) dizem que a mineração de dados pode ser classificada em duas categorias:

- **Descritiva:** caracteriza as propriedades gerais dos dados do banco;
- **Preditiva:** realiza inferências nos dados correntes a fim de realizar previsões.

Em alguns casos, os usuários podem não saber a priori quais são os dados que lhes trarão maiores ganhos de conhecimento. Nesses casos, deve-se trabalhar com diferentes estratégias de mineração de dados em paralelo, visando à extração de padrões que melhor atendam aos usuários.

Podem-se extrair diferentes tipos de padrões a partir de um banco de dados. A seguir será feita uma revisão baseada no trabalho de Han et al., (2001) sobre os diferentes tipos de padrões e regras que podem ser extraídos através da mineração de dados.

2.2.2.1 Descrição de Classe ou Conceito: Caracterização e Discriminação

Dados podem ser associados a classes ou conceitos. Por exemplo, em uma loja de produtos de informática, *computadores* e *impressoras* podem ser classes de itens de venda, e *grandes compradores* e *pequenos compradores* podem ser conceitos atribuídos a clientes.

As descrições de conceito/classe podem ser derivadas em:

- **Caracterização de dados:** *calculando a soma dos dados da classe em estudo (também chamada de classe alvo⁴) em termos gerais;*
- **Discriminação de dados:** *comparando a classe em estudo com uma ou um conjunto de outras classes (também chamadas de classes contrastantes⁵);*
- **Caracterização e Discriminação de dados:** neste caso as duas abordagens descritas anteriormente são empregadas.

⁴ do inglês *target class*

⁵ do inglês *contrasting classes*

2.2.2.2 Classificação e Predição

Classificação é o processo de encontrar um *modelo* (ou conjunto de funções) que descreve e distingue classes ou conceitos de dados. De posse deste modelo é possível identificar objetos cuja classe ainda não é conhecida

O modelo derivado dos dados é baseado na análise do conjunto de *dados de treinamento*, ou seja, o conjunto de dados cuja classificação já é conhecida previamente. Este modelo pode ser representando por diversas formas, tais como:

- Regras de Classificação SE-ENTÃO
- Árvores de Decisão (QUINLAN, 1986);
- Formulação Matemática;
- Redes Neurais Artificiais (LU et al., 1996)

Vale ressaltar que uma árvore de decisão pode ser facilmente transformada em regras de classificação. Diversos trabalhos, tais como: Braga et al., (2007), Gustafson; Vanneschi (2008), Lu et al., (1996), Quinlan (1986), Castanheira (2008), dentre outros, apontam as árvores de decisão e as redes neurais artificiais como as técnicas mais utilizadas para classificação e predição por serem relativamente simples de se programar e por já existir programas de domínio público disponíveis na internet. Em particular, destaca-se o pacote Weka (2012) que traz implementações de árvores de decisão além de outros algoritmos para serem utilizadas em aplicações de mineração de dados. Ainda sobre o estudo bibliográfico realizado, percebeu-se também que métodos evolucionários têm sido utilizados com sucesso em tarefas de classificação e predição, conforme será detalhado na seção 2.5.

Normalmente, a classificação é utilizada para se inferir a qual classe um objeto pertence. Entretanto, em algumas aplicações o usuário pode tentar prever alguns dados que tenham sido perdidos ou que não estejam disponíveis em alguns objetos que já estejam classificados previamente. A essa estratégia dá-se o nome de predição.

Classificação e predição devem ser precedidas por uma análise de relevância, cujo objetivo é tentar identificar atributos que não contribuem com o processo de classificação e predição. Esses atributos podem ser excluídos.

Os dados são avaliados de acordo com seu *ganho de informação*. Esse valor é calculado baseado no conceito de *entropia*, que consiste no grau de impureza dos dados. Formalmente, definem-se esses conceitos da seguinte forma:

Definição 1 – **Entropia**: A entropia de um conjunto de dados S é dada por $Ent(S) = -\sum_{i=1}^m p_i \log_2(p_i)$, onde m é o número de classes e p_i é a probabilidade de ocorrência da classe i no conjunto S .

Definição 2 – **Ganho de Informação**: O ganho de informação do atributo B é dado por $GI(B,A) = Ent(A) - Ent(B/A)$, onde $Ent(A)$ é a entropia de A e $Ent(B/A)$ é a entropia de B tendo sido definido o valor do atributo A .

Assim, os métodos de classificação e predição procuram identificar os atributos que apresentam maior ganho de informação a fim de rotular os dados em classes ou inferir informações sobre o banco.

Cabe aqui destacar que o foco desta tese está na tarefa de classificação dos dados, dentro do processo maior de KDD.

2.3 Representações Geográficas e Relacionamentos Topológicos

Esta seção se destina a apresentar as principais definições sobre representações geográficas, que se fazem necessárias para se entender e resolver um problema de mineração de dados em Sistemas de Informações Geográficas (SIG). Estudos mais profundos sobre sistemas de informações geográficas e sobre bancos de dados geográficos podem ser encontrados respectivamente em (WORBOYS; DUCKHAM, 2004) e (CASANOVA et al., 2005).

2.3.1 Definições

Um banco de dados geográficos pode utilizar basicamente dois tipos de representações para dados localizáveis espacialmente:

- Vetorial;
- Matricial.

Essas alternativas de representação serão detalhadas nas subseções a seguir.

2.3.1.1 Representações Vetoriais

Representações vetoriais utilizam listas de coordenadas para codificar as fronteiras de uma entidade geográfica, isto é, caracterizam a localização e a forma geométrica de cada entidade utilizando coordenadas. Existem três formas básicas de estruturas

vetoriais: pontos, linhas e polígonos. Essas estruturas são definidas no plano cartesiano⁶ e serão detalhadas a seguir.

Um **Ponto** consiste em um par ordenado de coordenadas espaciais (x, y), que pode ser utilizado para identificar localizações ou ocorrências no espaço. Essa é a estrutura mais simples das representações vetoriais, pois ela não possui informações do tipo comprimento (ou perímetro) e área. Desta forma, o ponto é utilizado em entidades geográficas simples, ou quando as dimensões da entidade real (comprimento e área) não são relevantes, por exemplo, localização de veículos, ocorrências policiais e ocorrências de doenças. Uma **Linha** (ou *linha poligonal*) consiste num conjunto de pontos interconectados a fim de representar entidades unidimensionais. Essa estrutura possui uma importante dimensão: o comprimento. Linhas são utilizadas em entidades geográficas cujo comprimento é relevante, porém a área não é relevante. São exemplos de entidades que podem ser representadas por linhas: ferrovias, rodovias e rios. É importante destacar que, mesmo que as entidades citadas possuam alguma área associada quando observadas em detalhe, a escolha de linhas para sua representação indica que essa característica não é relevante para a aplicação em questão.

Um **Polígono** (ou área) consiste em uma região no plano, delimitada por uma ou mais linhas poligonais fechadas, ou seja, nas quais o último vértice coincide com o primeiro. Podem existir várias linhas fechadas como parte da representação do polígono, constituindo ilhas e buracos em relação a um polígono-base. A fronteira do polígono divide o plano em duas regiões: exterior e interior. O polígono possui dimensões importantes, tais como perímetro e área. São exemplos de entidades geográficas que podem ser representadas com polígonos: setores censitários, bacias hidrográficas, áreas de produção agrícola.

Um Sistema de Gerenciamento de Banco de Dados Espaciais (SGBDE) trata as estruturas vetoriais da seguinte forma (CASANOVA et al., 2005 *apud* DAVIS Jr., 1997):

- **Ponto:** consiste num par ordenado (x, y) de coordenadas espaciais;
- **Reta e Segmento de Reta:** Sejam p_1 e p_2 dois pontos distintos no plano. A combinação linear $\alpha.p_1 + (1 - \alpha).p_2$ é uma reta no plano, onde α é um

⁶ Depois que passam por um processo de projeção cartográfica. O tipo de projeção e seus parâmetros precisam ser conhecidos para que as coordenadas de cada entidade possam ser adequadamente interpretadas.

número real. Se $0 \leq \alpha \leq 1$, tem-se um único segmento de reta no plano, onde p_1 e p_2 são os pontos extremos;

- **Linha Poligonal:** Sejam v_0, v_1, \dots, v_{n-1} n pontos no plano. Sejam $S_0 = v_0v_1, S_1 = v_1v_2, \dots, S_{n-2} = v_{n-2}v_{n-1}$ uma sequência de $n-1$ segmentos conectando os pontos. Esses segmentos formam uma poligonal L se e somente se:

- a) A interseção de segmentos consecutivos é apenas o ponto extremo compartilhado por eles, de maneira que $S_i \cap S_{i+1} = v_{i+1}$;
- b) Segmentos não consecutivos não se interceptam, de maneira que $S_i \cap S_j = \emptyset, \forall i, j$ tais que $j \neq i+1$;
- c) $v_0 \neq v_{n-1}$ ou seja, a poligonal não é fechada.

- **Polígono:** consiste na região do plano limitada por uma linha poligonal fechada.

As condições (a) e (b) de linhas poligonais excluem a possibilidade de auto-interseção. O único ponto de interseção são os vértices dos segmentos consecutivos. Essa observação é válida tanto para linhas poligonais quanto para polígonos. A diferença entre essas duas definições é a condição (c), pois a linha poligonal é aberta e o polígono é uma região fechada.

Existem diversos algoritmos que podem ser aplicados nessas estruturas a fim de se extrair informações relevantes, por exemplo, área do polígono, coordenadas baricêntricas, interseção de retângulos, interseção de segmentos de reta, centroide de um polígono, simplificação de poligonais, dentre outras.

2.3.1.2 Representações Matriciais

Consistem em uma estrutura de grade regular, na qual se representa, célula a célula, o atributo do elemento em questão. O espaço é representado na forma de uma matriz $m \times n$ onde m é o número de linhas e n é o número de colunas, de maneira que existem $m \times n$ células. Cada célula, cuja localização é dada pelos números da linha e da coluna dentro da matriz, possui o valor do atributo estudado.

A representação matricial considera o espaço como uma superfície plana, onde cada célula representa uma porção do terreno. A relação entre a área coberta no terreno e o tamanho da célula define a resolução da matriz. Assim, quanto maior o número de células cobrindo o mesmo espaço geográfico, maior será a resolução, permitindo que se

tenha um maior detalhamento sobre o atributo estudado. Evidentemente, quanto maior for o número de células, maior também será o espaço necessário para o seu armazenamento.

Estruturas matriciais são muito utilizadas para representar fenômenos que variam continuamente no espaço estudado, como, por exemplo, elevação do terreno, clima e tipo do terreno. As imagens de satélite são um exemplo de uso de representações matriciais.

2.3.2 Relacionamentos Topológicos

Segundo Davis Jr e Queiroz (2005), caracterizar os relacionamentos topológicos faz com que se atribua um contexto semântico aos algoritmos geométricos, *i.e.*, um usuário de informações geográficas pode interpretar os relacionamentos existentes entre os objetos. Por exemplo, é relevante conhecer quais são as ruas que cruzam uma grande avenida, ou ainda quais são as cidades que fazem fronteira com uma grande cidade.

A fim de definir os relacionamentos topológicos, faz-se necessário primeiro, conhecer algumas definições da geometria vetorial no espaço \mathbb{R}^2 , a saber:

- Ponto: definido por um par de coordenadas no espaço \mathbb{R}^2 .
- Linha: conjunto de pontos conectados;
- Polígono: linha em que o ponto inicial é igual ao ponto final, delimitando uma área;
- Fronteira: conjunto dos pontos de contorno de um polígono, ou pontos inicial e final de uma linha, ou o conjunto vazio (ponto). Denotada por ∂L .
- Interior de L : consiste nos demais pontos de L que não fazem parte de ∂L . Denotado por L° .
- Região A : conjunto de pontos com um interior conectado (denotado por A°), um fronteira conectada (denotada por ∂A) e um único exterior conectado (denotado por A^-). Essa região A não possui buracos.

Egenhofer e Franzosa (1995) propõem a definição dos relacionamentos topológicos com base na **matriz de 4-interseções**, a qual considera oito relacionamentos topológicos binários, que representam a interseção entre a fronteira e o interior de duas geometrias. A Figura 2.2 apresenta o modelo de Matriz de 4-Interseções.

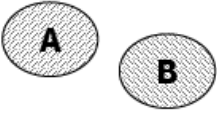
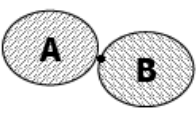
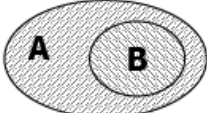
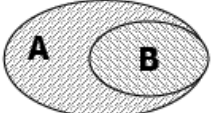

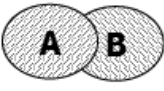


 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \emptyset & \emptyset \end{matrix} \\ A^\circ & \begin{matrix} \emptyset & \emptyset \end{matrix} \end{matrix}$ <p>disjoint</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \neg\emptyset & \emptyset \end{matrix} \\ A^\circ & \begin{matrix} \emptyset & \emptyset \end{matrix} \end{matrix}$ <p>meet</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \emptyset & \emptyset \end{matrix} \\ A^\circ & \begin{matrix} \neg\emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>contains</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \neg\emptyset & \emptyset \end{matrix} \\ A^\circ & \begin{matrix} \neg\emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>Covers</p>
 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \neg\emptyset & \emptyset \end{matrix} \\ A^\circ & \begin{matrix} \emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>equal</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \neg\emptyset & \neg\emptyset \end{matrix} \\ A^\circ & \begin{matrix} \neg\emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>overlap</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \emptyset & \neg\emptyset \end{matrix} \\ A^\circ & \begin{matrix} \emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>inside</p>	 $\begin{matrix} \partial A & \begin{matrix} \partial B & B^\circ \end{matrix} \\ \begin{matrix} \neg\emptyset & \neg\emptyset \end{matrix} \\ A^\circ & \begin{matrix} \emptyset & \neg\emptyset \end{matrix} \end{matrix}$ <p>Covered By</p>

Figura 2.2 – Matriz de 4-Interseções para relações entre duas regiões (DAVIS Jr; QUEIROZ, 2005 *apud* EGENHOFER et al., 1994).

O modelo mostra os resultados das interseções considerando os valores vazio (\emptyset) e não vazio ($\neg\emptyset$). Existe situações onde se faz necessário considerar as dimensões das interseções não vazias, por exemplo, um polígono A é considerado vizinho de B quando eles possuem pelo menos uma aresta em comum; se tudo o que tiverem em comum for um ponto, pode-se não considerá-los vizinhos.

A matriz de 4-interseções não é suficiente para definir completamente relacionamentos topológicos entre geometrias com estruturas mais complexas, tais como regiões com ilhas e separações, nem para incluir comparações envolvendo pontos e linhas, pois o exterior das geometrias não é considerado. Assim, Egenhofer e Herring (1991) apresentam um novo modelo, chamado de matriz de 9-interseções, no qual são considerados os resultados das interseções entre as fronteiras, interiores e exteriores de duas geometrias. A Figura 2.3 apresenta o modelo da matriz de 9-interseções.









 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ disjoint	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \neg \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ meet	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ contains	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \neg \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ covers
 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \neg \emptyset & \emptyset & \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \emptyset & \neg \emptyset & \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \emptyset & \emptyset & \neg \emptyset \end{bmatrix}$ equal	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ overlap	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \emptyset & \neg \emptyset & \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \emptyset & \neg \emptyset & \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ inside	 $\partial B \quad B^\circ \quad B^-$ $\partial A \begin{bmatrix} \neg \emptyset & \neg \emptyset & \emptyset \end{bmatrix}$ $A^\circ \begin{bmatrix} \emptyset & \neg \emptyset & \emptyset \end{bmatrix}$ $A^- \begin{bmatrix} \neg \emptyset & \neg \emptyset & \neg \emptyset \end{bmatrix}$ covered by

Figura 2.3 – Matriz de 9-Interseções para Relações entre duas regiões. (DAVIS Jr; QUEIROZ, 2005 *apud* EGENHOFER; HERRING, 1991).

A fim de considerar a dimensão das interseções entre os objetos na caracterização dos relacionamentos topológicos, foi criado o modelo de 9 interseções dimensionalmente estendido (*Dimensionally Extended Nine-Intersection Model* ou DE-9IM) (STROBL, 2010). Esse modelo considera o interior (I), exterior (E) e fronteira (B) dos dois objetos em questão e analisa a dimensão (0D, 1D, 2D) das interseções entre as partes desses objetos. Essa interseção pode ser descrita na forma de uma matriz 3x3 da seguinte forma (Figura 2.4):

$$DE-9IM(A, B) = \begin{bmatrix} \dim(I(A) \cap I(B)) & \dim(I(A) \cap B(B)) & \dim(I(A) \cap E(B)) \\ \dim(B(A) \cap I(B)) & \dim(B(A) \cap B(B)) & \dim(B(A) \cap E(B)) \\ \dim(E(A) \cap I(B)) & \dim(E(A) \cap B(B)) & \dim(E(A) \cap E(B)) \end{bmatrix}$$

Figura 2.4 - Matriz resultante do relacionamento entre dois objetos no DE-9IM. Fonte: (STROBL, 2010)

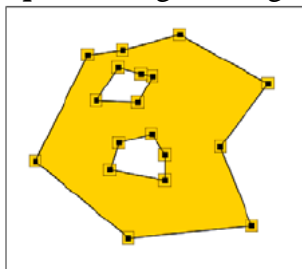
Esse modelo é a base da implementação dos relacionamentos espaciais em gerenciadores de bancos de dados como o PostGIS. Os relacionamentos espaciais básicos, descritos no modelo DE-9IM, são: “*Equals*”, “*Disjoint*”, “*Intersects*”, “*Touches*”, “*Crosses*”, “*Within*”, “*Contains*” e “*Overlaps*”. A Tabela 2.1 detalha cada relacionamento e o significado correspondente.

Tabela 2.1 - Relacionamentos topológicos e os respectivos significados no DE-9IM. Fonte: (STROBL, 2010)

Relacionamentos Topológico	Significado
Equals	As geometrias são topologicamente iguais
Disjoint	As geometrias não possuem nenhum ponto em comum
Intersects	As geometrias possuem pelo menos um ponto em comum (inverso de Disjoint)
Touches	As geometrias possuem pelo menos um ponto da fronteira em comum, mas não pontos interiores
Crosses	As geometrias compartilham alguns mas não todos os pontos interiores. Além disso, a dimensão da interseção é menor que pelo menos uma das geometrias
Overlaps	As geometrias compartilham alguns mas não todos os pontos em comum e a interseção possui a mesma dimensão das geometrias
Within	A geometria A está no interior da geometria B
Contains	A geometria B está no interior da geometria A (inverso de Within)

Esses relacionamentos foram utilizados na implementação do DMGeo, que é descrito no Capítulo 5. A seguir serão apresentados exemplos desses relacionamentos. À esquerda são mostrados os objetos geográficos e à direita a matriz resultante do relacionamento. Os valores possíveis nessa matriz são -1, 0, 1 e 2, onde o primeiro valor denota que não há interseção entre os objetos e os demais valores indicam a dimensão da interseção.

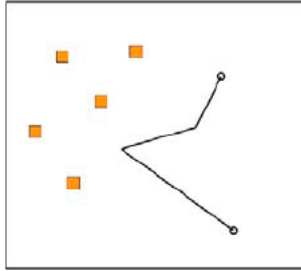
Equals: Polígono A igual ao polígono B.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	2	-1	-1
Boundary (A)	-1	1	-1
Exterior (A)	-1	-1	2

Figura 2.5 - Exemplo do relacionamento Equals no DE-9IM. Fonte: (STROBL, 2010)

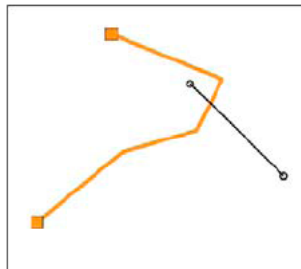
Disjoint: Linha A é disjunta (*disjoint*) do multiponto B.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	-1	-1	1
Boundary (A)	-1	-1	0
Exterior (A)	0	-1	2

Figura 2.6 - Exemplo do relacionamento Disjoint no DE-9IM. Fonte: (STROBL, 2010)

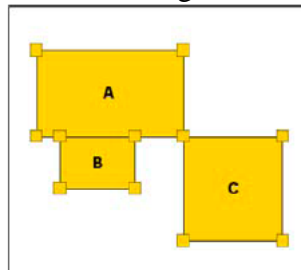
Intersects: Linha A intercepta (*intersects*) a linha B.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	0	-1	1
Boundary (A)	-1	-1	0
Exterior (A)	1	0	2

Figura 2.7 - Exemplo do relacionamento Intersects no DE-9IM. Fonte: (STROBL, 2010)

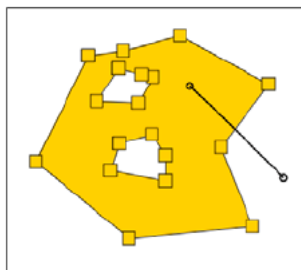
Touches: Polígono A toca (*touches*) os polígonos B e C.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	-1	-1	2
Boundary (A)	-1	1/0	1
Exterior (A)	2	1	2

Figura 2.8 - Exemplo do relacionamento Touches no DE-9IM. Fonte: (STROBL, 2010)

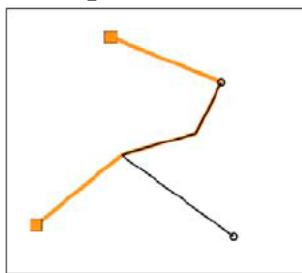
Crosses: Linha B cruza (*crosses*) o polígono A.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	1	0	2
Boundary (A)	0	-1	1
Exterior (A)	1	0	2

Figura 2.9 - Exemplo do relacionamento Crosses no DE-9IM. Fonte: (STROBL, 2010)

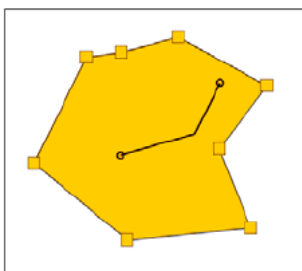
Overlaps: Linha A sobrepõe (*overlaps*) a linha B.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	1	-1/0	1
Boundary (A)	0/-1	-1	0
Exterior (A)	1	0	2

Figura 2.10 - Exemplo do relacionamento Overlaps no DE-9IM. Fonte: (STROBL, 2010)

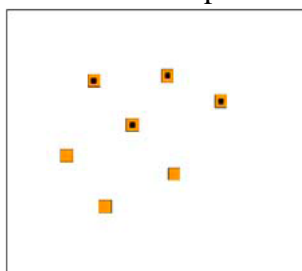
Within: Linha A está dentro (*within*) do polígono B.



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	1	-1	-1
Boundary (A)	0	-1	-1
Exterior (A)	2	1	2

Figura 2.11 - Exemplo do relacionamento Within no DE-9IM. Fonte: (STROBL, 2010)

Contains: Multipontos A (quadrados) contem (*contains*) os multipontos B (círculos).



	Interior (B)	Boundary (B)	Exterior (B)
Interior(A)	0	-1	0
Boundary (A)	-1	-1	-1
Exterior (A)	-1	-1	2

Figura 2.12 - Exemplo do relacionamento Contais no DE-9IM. Fonte: (STROBL, 2010)

2.4 Algoritmos Evolucionários

2.4.1 Introdução

Através da observação da natureza, em particular pela observação das técnicas de sobrevivência e evolução das espécies, muitos algoritmos de otimização foram inspirados e concebidos. Podem-se citar como exemplos os algoritmos baseados nos processos de colônias de formigas (“*Ant Colony Optimization*” de Dorigo e Stützle (2004)), do voo de pássaros (“*Particle swarm optimization*” de Kennedy e Eberhart (1995)) e da caça de predadores (“*A spatial predator-prey approach to multi-objective optimization: a preliminary study*” de Laumanns et al., (1998)). A terminologia

utilizada nos algoritmos de inspiração biológica é também importada da própria biologia e adaptada ao modelo computacional. Goldberg (1989) apresenta o mapeamento dos termos usados na genética natural para os termos utilizados na genética artificial, implementada nos algoritmos genéticos (HOLLAND, 1975) (GOLDBERG, 1989). A Tabela 2.2 resume este mapeamento.

Tabela 2.2 – Mapeamento da terminologia da genética natural para a terminologia utilizada em Algoritmos Genéticos (GOLDBERG, 1989, p. 22).

Genética Natural	Genética Artificial
Gene	Caractere
Alelo	Valor do Caractere
Cromossomo	Cadeia de Caracteres
Locus	Posição do Caractere na Cadeia (Caracteres)
Genótipo	Estrutura, Indivíduo
Fenótipo	Ponto Solução, Estrutura Decodificada

Coello et al., (2007) apresentam uma classificação das técnicas de otimização global (Figura 2.13) na qual os algoritmos se caracterizam em: enumerativos, determinísticos e estocásticos.

Os algoritmos estocásticos, como é o caso dos evolucionários, utilizam procedimentos probabilísticos para guiar a pesquisa para a(s) região(ões) onde se encontra(m) a(s) solução(ões) ótima(s). Em particular, os algoritmos evolucionários apresentam as seguintes características (GOLDBERG, 1989) (FREITAS, 2002) (SOARES, 2008):

- Eficiência em encontrar as regiões ótimas;
- Flexibilidade para se adaptar a diferentes problemas;
- Independência do uso de derivadas (das funções as quais se deseja otimizar);
- Possibilidade de se encontrar diversos pontos ótimos em uma única execução.

Algoritmos evolucionários têm sido cada vez mais aplicados para resolver problemas multiobjetivos. Dentre inúmeros trabalhos científicos dedicados a tratar deste assunto, dois têm sido citados com frequência: Deb (2001); e Coello Coello (2006). Algoritmos evolucionários também têm sido muito utilizados no contexto de mineração de dados, como por exemplo, em (LI et al., 2004), (SRINIVASA et al., 2007), (SIKORA; PIRAMUTHU, 2007), (DEHURI et al., 2008), (ALCALÁ-FDEZ et al., 2009), (AL-OBEIDAT et al., 2011), (HUANG et al., 2012) e (ÁLVAREZ; VÁZQUEZ, 2012). Contudo, poucos trabalhos relacionam mineração de dados, algoritmos

evolucionários e informações geográficas (WHIGHAM et al., 2000), (BOGORNÝ et al., 2006) e (PEREIRA et al., 2010).

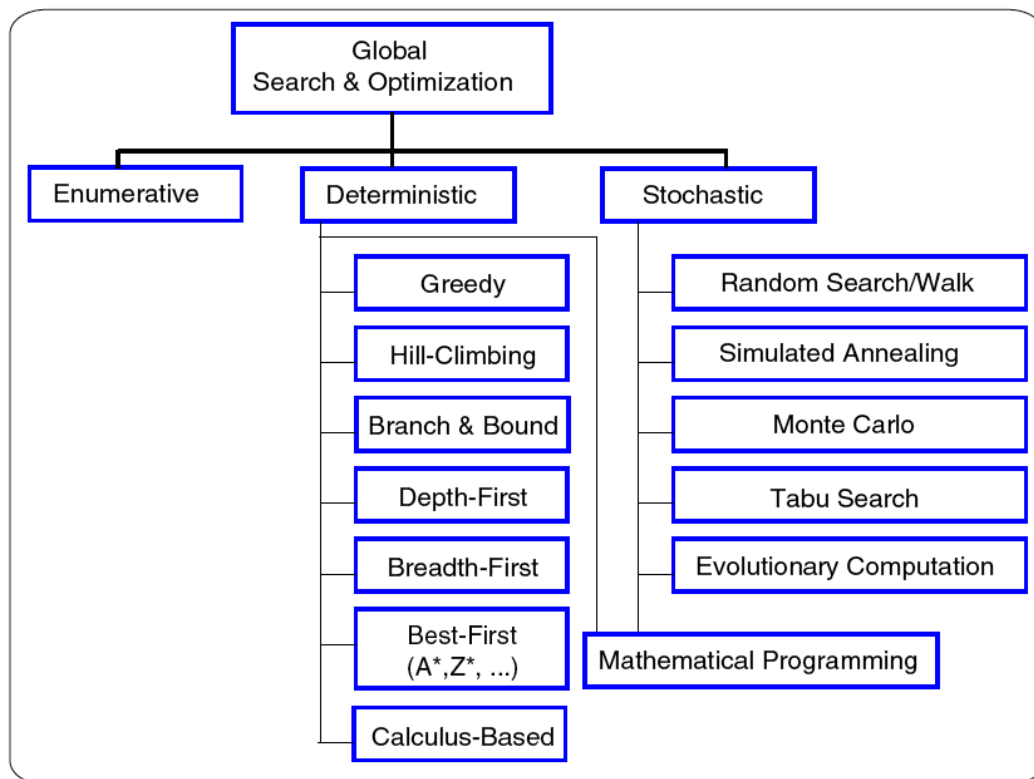


Figura 2.13 – Técnicas de Otimização Global (COELLO et al., 2007).

Os algoritmos evolucionários tentam reproduzir artificialmente algum processo natural. Isto porque se acredita que se determinado mecanismo está presente e é eficaz na natureza, então há uma chance de sucesso se o mesmo for aplicado em modelos artificiais (SOARES, 2008, p. 16). O custo computacional muitas vezes determina a viabilidade do método.

Goldberg (1989) destaca que grande parte dos algoritmos evolucionários é baseada no processo de evolução por seleção natural descrito por Darwin. Contudo, o termo “evolucionário” é usado de maneira mais ampla e é aplicado aos métodos que desenvolvem soluções na quais os melhores indivíduos têm maiores probabilidades de serem selecionados. Assim, os algoritmos evolucionários são similares no propósito de desenvolver soluções de forma evolutiva.

2.4.2 Conceitos Básicos e Estrutura

Os conceitos básicos dos algoritmos evolucionários são:

- **Indivíduo:** é um ponto no espaço de otimização, isto é, é o conjunto de valores das variáveis candidatas a otimizar a função objetivo. Representa uma solução candidata para o problema;
- **População** (ou soluções candidatas): conjunto de indivíduos;
- **Objetivo:** função que se deseja minimizar ou maximizar;
- **Fitness:** Medida do desempenho do indivíduo com relação ao modelo matemático do problema de otimização. Esta medida é utilizada pelo operador de seleção, onde pode ser atribuída uma probabilidade de sobrevivência e reprodução para cada um dos indivíduos.

A estrutura básica destes algoritmos é mostrada no Algoritmo 2.1.

Algoritmo 2.1 – Estrutura básica dos algoritmos evolucionários.

<p>Entrada (<i>parâmetros</i>)</p> <p>Saída (<i>soluções candidatas</i>)</p> <ol style="list-style-type: none"> 1. Gere população inicial; 2. enquanto o critério de parada não é satisfeito faça 3. Avalie cada indivíduo da população corrente; 4. Aplique o método de <i>seleção</i> ; 5. Aplique operador(es) de geração de novos indivíduos; 6. fim enquanto; 7. retorne a melhor <i>solução</i>.

Os parâmetros de entrada dos algoritmos evolucionários variam para todo tipo de algoritmo, mas sempre têm relação com os detalhes da formação da população e com operadores evolucionários. O passo 1 do Algoritmo 2.1 corresponde à criação da população inicial, na maioria das vezes, de forma aleatória. O laço compreendido entre os passos 2 e 6 corresponde efetivamente à evolução da população. Em cada iteração deste laço, primeiro se avalia os indivíduos da população corrente e aplicam-se os operadores genéticos de seleção, cruzamento e mutação para produzir uma nova geração. A seleção é responsável por implementar o conceito de evolução natural, ou seja, chances maiores de sobrevivência e reprodução são atribuídas àqueles indivíduos de melhor desempenho, *i.e.*, de melhor *fitness*. A aplicação de operadores de geração de novos pontos no espaço de otimização é responsável pela geração da nova população. Este processo se repete até que algum critério de convergência seja atingido. O passo 7 retorna a melhor solução dentre todas as soluções da última geração.

2.4.3 Algoritmos Genéticos

Holland (1975) apresentou a primeira versão do algoritmo que se tornaria o mais popular entre os algoritmos evolucionários: Algoritmo Genético. A sua versão mais popular é o Algoritmo Genético Simples, SGA (*Simple Genetic Algorithm*), no qual a população de indivíduos possui tamanho fixo e os indivíduos são codificados em uma cadeia (*string*) binária. As operações aplicadas à população são:

- Seleção;
- Cruzamento;
- Mutação.

Cada operação possui uma probabilidade própria e, no caso do SGA, elas são fixas. O Algoritmo 2.2 apresenta a estrutura básica dos algoritmos genéticos.

Algoritmo 2.2 – Estrutura básica dos Algoritmos Genéticos (GOLDBERG, 1989).

<p>Entrada (<i>parâmetros</i>)</p> <p>Saída (<i>população</i>)</p> <ol style="list-style-type: none">1. Inicie o contador de gerações <i>c</i> com 0.2. Gere a <i>população</i> inicial.3. enquanto o critério de parada não é atingido faça4. Avalie os indivíduos da <i>população</i>.5. Aplique o método de <i>seleção</i>.6. Aplique o operador de <i>cruzamento</i>.7. Aplique o operador de <i>mutação</i>.8. Incremente <i>c</i>.9. fim enquanto.10. retorne a <i>população</i> corrente.

Os algoritmos genéticos recebem como entrada parâmetros que estão relacionados à população e aos operadores evolucionários (seleção, cruzamento e mutação). A saída será a população final gerada ou a melhor solução encontrada. Cabe destacar que a população é um conjunto de soluções, de maneira que a solução com um melhor desempenho deve ser escolhida como a solução final para o problema. Os passos 1 e 2 formam o ponto de partida da evolução da população: o contador de gerações *c* é zerado; a população é configurada de acordo com os parâmetros iniciais. Os passos de 3 a 9 formam o laço onde, a cada iteração, a população é avaliada e em seguida sofre as operações que produzem uma nova geração de maneira a proporcionar sua evolução. O laço se encerra quando algum critério de convergência for alcançado. São exemplos de critérios de convergência:

- Número máximo de iterações (o contador *c* atinge um determinado limite)

- A população não apresenta evolução em seu desempenho durante um dado número de gerações consecutivas.

Uma característica importante da população de um AG, que pode influenciar na qualidade do resultado final, é a diversidade genética, a qual consiste na medida da representatividade do espaço de otimização. Assim, quanto maior a semelhança entre os indivíduos, menor a diversidade genética; quanto menor for a semelhança entre os indivíduos, maior a diversidade genética.

Segundo Vasconcelos et al., (2001), a manutenção de uma alta diversidade genética durante a execução do algoritmo diminui as chances de se convergir para um ótimo local e aumenta as chances de se encontrar um ótimo global. O trabalho de Vasconcelos et al., (2001) apresenta um algoritmo de controle da diversidade genética que é mostrado no Algoritmo 2.3.

Algoritmo 2.3 – Algoritmo de ajuste da diversidade genética (VASCONCELOS et al., 2001).

<pre> 1. se ($mdg > V_{max}$) 2. $p_m = k_m * p_m$; 3. $p_c = p_c / k_c$; 4. senão se ($mdg < V_{min}$) 5. $p_c = k_c * p_c$; 6. $p_m = p_m / k_m$; 7. fim se 8. se ($p_m > p_{m_max}$) $p_m = p_{m_max}$; 9. se ($p_m < p_{m_min}$) $p_m = p_{m_min}$; 10. se ($p_c > p_{c_max}$) $p_c = p_{c_max}$; 11. se ($p_c < p_{c_min}$) $p_c = p_{c_min}$; </pre>
--

Neste algoritmo, mdg é um índice de diversidade genética, definido como sendo a razão do valor médio de fitness, considerando a fitness de todos os indivíduos da população, pelo valor de fitness máximo em cada geração. V_{max} e V_{min} são respectivamente o limite superior e inferior para mdg , p_m é a probabilidade de mutação, p_c é a probabilidade de cruzamento, k_m é o fator de ajuste da mutação, k_c é o fator de ajuste do cruzamento, p_{c_min} , p_{c_max} , p_{m_min} e p_{m_max} são respectivamente os limites inferior e superior de probabilidades de cruzamento e mutação.

O valor de mdg é limitado pelo intervalo $0 \leq mdg \leq 1$. Assim, mdg igual a 1, significa que todos os indivíduos da população possuem o mesmo valor da função *fitness*, ou seja, a diversidade genética é mínima o que denota a convergência do algoritmo para um mesmo ponto do espaço de otimização. Para se evitar uma convergência prematura, as probabilidades de cruzamento e mutação são ajustadas de

acordo com o valor de mdg . Assim, quando se tem uma alta diversidade genética (mdg inferior ao valor de V_{min}), diminui-se a probabilidade de mutação e aumenta-se a de cruzamento; quando a diversidade genética é muito baixa (mdg excede o valor V_{max}), aumenta-se a probabilidade de mutação e diminui-se a de cruzamento (vide Algoritmo 2.3). Os valores dos parâmetros utilizados por Vasconcelos et al., (2001) são mostrados na Tabela 2.3.

Tabela 2.3 – Parâmetros e respectivos valores utilizados no algoritmo de manutenção da Diversidade Genética (VASCONCELOS et al., 2001).

Parâmetro	V_{min}	V_{max}	P_{c_min}	P_{c_max}	P_{m_min}	P_{m_max}	k_m	k_c
Valor	0.005	0.150	0.500	1.000	0.001	0.250	1.100	1.100

2.4.3.1 Nicho

A utilização de técnica de nicho consiste em dividir-se a população em subpopulações a fim de proporcionar que as subpopulações sejam capazes de encontrar outros ótimos locais.

Basicamente, a utilização de técnica de nicho visa evitar a obtenção de um único ponto ótimo. Existem problemas que podem apresentar mais de um ponto ótimo. Assim, pode-se desejar conhecer uma maior variedade de pontos que minimizam a função objetivo em um problema de minimização. A utilização de técnica de nicho é recomendada quando a função a ser minimizada é multimodal, com diversas regiões de mínimos locais e/ou globais.

A implementação de técnicas de nicho requer alguns ajustes nas operações genéticas de maneira que essas operações passem a ocorrer no nicho e não mais em toda população, ou seja, a seleção, cruzamento e mutação são operações aplicadas considerando a subpopulação do nicho.

2.4.3.2 Elitismo Simples e Global

Durante a execução do algoritmo, o melhor indivíduo gerado até então pode se perder (não ser selecionado para compor a próxima geração ou ser destruído pelas operações de cruzamento e/ou mutação). Existem duas técnicas que visam preservar o histórico das melhores soluções obtidas. A primeira delas é a técnica de elitismo simples, que consiste em se armazenar o melhor indivíduo de cada geração e propagá-lo para a geração seguinte. Dessa maneira, garante-se que a melhor solução não será perdida.

A técnica do elitismo global consiste em substituir os indivíduos pais pelos filhos somente se eles tiverem maior valor de *fitness*.

2.4.4 Programação Genética

A programação genética (PG) possui grande semelhança com os algoritmos genéticos (AGs). A diferença principal está na representação dos indivíduos: enquanto nos AGs codificam-se os indivíduos em código binária, Gray, real, ou outro, a PG codifica os indivíduos na forma de uma árvore. Essa codificação apresentada na PG gera algumas alterações significativas nos operadores genéticos, além de possibilitar alguns grandes benefícios, tais como conter não somente valores de variáveis, mas também funções. As alterações e benefícios apresentados pela PG em relação ao AG serão detalhas a seguir.

2.4.4.1 Representação dos Indivíduos

Conforme já mencionado, o fato do indivíduo na PG ser representado na forma de árvore possibilita a esse indivíduo conter não somente valores de variáveis, mas também funções. Segundo Koza (1992), o conjunto das funções (nós internos da árvore (ZIVIANI; BOTELHO, 2006)) que compõem o indivíduo é chamado de conjunto de funções⁷. Montana (1995) considera que o termo que melhor se aplica a esse conjunto é conjunto não terminal⁸, uma vez que o conjunto de nós terminais (nós externos (ZIVIANI; BOTELHO, 2006)) pode conter funções que não recebem parâmetros. Esse texto utilizará o termo “conjunto de funções”, por ser o mais utilizado na literatura, devido à grande influência exercida pelo livro de Koza (1992).

Segundo Koza (1992), o indivíduo da PG, composto pelos conjuntos terminal e não terminal, deve satisfazer pelo menos duas propriedades:

- Suficiência⁹: indica que o poder de expressividade do indivíduo deve ser bom o suficiente para representar uma solução para o problema em questão.
- Fechamento¹⁰: indica que uma função deve poder receber como parâmetro de entrada o valor produzido como saída de outra função ou um valor que pode ser assumido por um dos terminais.

Na prática, obedecer à propriedade de fechamento pode ser uma tarefa muito difícil.

⁷ do inglês *Function Set*

⁸ do inglês *Nonterminal Set*

⁹ do inglês *Sufficiency*

¹⁰ do inglês *Closure*

Entretanto, não observar essa propriedade leva a situações que geram indivíduos inconsistentes. Certamente essa é uma das maiores dificuldades dos projetistas de algoritmos baseados em PG, principalmente na aplicação das operações de mutação e cruzamento. Segundo Freitas (2002), a garantia de fechamento dos nós terminais e nós funções é especialmente importante, pois não há sentido em uma operação do tipo *Idade* + *Salário* uma vez que esses dois atributos possuem domínios diferentes. Dessa forma, ainda segundo Freitas (2002), o domínio utilizado pelo banco de dados pode ser útil a fim de se validar o tipo dos parâmetros passados para cada função.

Além das duas características necessárias (suficiência e fechamento), Freitas (2002) aponta como característica desejável que o conjunto de nós que compõem o indivíduo tenha mais uma característica: parcimônia¹¹. Essa característica indica que o indivíduo deve conter somente as funções e os terminais estritamente necessários, *e.g.* (*NÃO X*) ao invés de (*NÃO (NÃO (NÃO X))*). Uma solução parcimoniosa é mais simples de ser avaliada e conseqüentemente mais simples de ser entendida pelo usuário final. A Figura 2.14 mostra um exemplo de indivíduo da PG. A expressão codificada pelo indivíduo é: $10 \times (7 / (3 + 5))$.

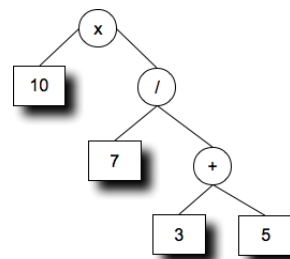


Figura 2.14 – Exemplo de Indivíduo na PG.

2.4.4.2 Cruzamento

A operação de cruzamento na PG, assim como no AG, consiste na troca de material genético entre os indivíduos pais. Entretanto, conforme destacado na seção 2.4.4.1, a fim de se garantir a propriedade de fechamento, é preciso verificar se o tipo das sub-árvores que serão inseridas são compatíveis com o tipo esperado. Considerando o exemplo da Figura 2.14, não se pode substituir o nó folha cujo valor é um número inteiro igual 5 por um nó do tipo lógico cujo valor é *true*. Em contrapartida, o mesmo nó pode ser substituído pela sub-árvore que representa a expressão $\text{seno}(\pi)$.

¹¹ do inglês *Parsimony*

Uma característica bastante significativa da operação de cruzamento é a sua capacidade de alterar o tamanho dos indivíduos durante a execução do algoritmo. Considerando novamente o exemplo da Figura 2.14, trocar o nó folha cujo valor é 5 por uma sub-árvore que representa a expressão $\text{seno}(\pi)$, fará com que a profundidade da árvore aumente. O sucessivo aumento da profundidade da árvore poderá gerar casos em que se tem uma expressão complexa, mas que pode ser simplificada (este é o típico caso onde não se obedece a propriedade parcimônia – seção 2.4.4.1). Para se evitar que a árvore que compõe o indivíduo cresça, porém sem apresentar um ganho significativo de desempenho na função *fitness*, algumas técnicas foram propostas e serão detalhadas na seção 2.6. Uma das ideias mais comuns é considerar a minimização do número de nós como um dos objetivos do algoritmo, ou seja, além de procurar otimizar a função principal o algoritmo deverá procurar pelo menor indivíduo (em termos de número de nós, ou até mesmo profundidade da árvore) que otimiza a função.

2.4.4.3 Mutação

O operador de mutação consiste na troca de um nó (terminal ou não) por outro nó (terminal ou uma sub-árvore) gerado. É notório que aqui, assim como no cruzamento, os indivíduos podem aumentar ou diminuir de tamanho. Freitas (2002) classifica a operação de mutação em quatro categorias principais, baseadas na capacidade de alteração de tamanho da árvore:

- Mutação de Ponto¹²: Consiste na troca de um nó externo por outro nó externo;
- Mutação de Colapso¹³: Um nó externo substitui um nó interno (sub-árvore). Nessa caso, a árvore diminui sua altura;
- Mutação de Expansão¹⁴: Um nó interno (sub-árvore) substitui um nó externo. Nesse caso, a árvore aumenta sua altura;
- Mutação de sub-árvore¹⁵: Um nó interno substitui outro nó interno. Nesse caso a árvore pode aumentar ou diminuir sua altura ou ainda não ter seu tamanho alterado.

¹² do inglês *Point Mutation*

¹³ do inglês *Collapse Mutation*

¹⁴ do inglês *Expansion Mutation*

¹⁵ do inglês *Subtree Mutation*

Uma implementação da PG pode utilizar somente uma categoria de mutação a fim de inserir uma tendência de comportamento do tamanho dos indivíduos (FREITAS, 2002). Por exemplo, utilizar somente a mutação de colapso fará com que a aplicação desse operador diminua o tamanho dos indivíduos.

2.5 Aplicação de Algoritmos Evolucionários em Mineração de Dados

Algoritmos evolucionários podem ser aplicados em diversas tarefas de mineração de dados: classificação, clusterização, limpeza ou redução de informações, dentre outras. Nesse trabalho, como já dito anteriormente, o foco é o desenvolvimento de algoritmos evolucionários para classificação de dados.

Formalmente, o problema de classificação pode ser entendido como:

Definição 3 – Problema de Classificação em Mineração de Dados: Sejam n o número de classes de um problema, c o conjunto de instâncias a serem classificadas e p a quantidade de instâncias do conjunto c . Encontre os n conjuntos (com um ou mais elementos) que contenham regra(s) a fim de classificar cada um das p instâncias em uma das n classes.

Conforme descrito na seção 2.4, os algoritmos evolucionários procuram o(s) melhor(es) indivíduos que maximizam uma determinada medida de *fitness*. Em aplicações de mineração de dados, cujo objetivo é realizar a classificação de padrões, os algoritmos evolucionários podem ser aplicados da seguinte forma:

- Os indivíduos da população representam uma regra de classificação ou até mesmo um conjunto de regras;
- A função *fitness* deverá mensurar a qualidade de cada indivíduo. Desta forma, o algoritmo evolucionário procurará por indivíduos que apresentem maior qualidade.

Podemos entender como “qualidade” de cada indivíduo a sua capacidade de classificar corretamente o maior número de padrões (ou registros) de um banco de dados.

Existem diversas formas de se medir a qualidade de uma regra de classificação. As mais comuns se baseiam em cálculos realizados a partir dos coeficientes da matriz de confusão (Tabela 2.4).

Tabela 2.4 – Matriz de Confusão.

Real \ Classif.	Positivo	Negativo
Positivo	Verdadeiro Positivo (TP)	Falso Negativo (FN)
Negativo	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Os coeficientes da matriz são calculados considerando cada uma das classes do problema. Considere um banco de dados contendo informações sobre transformadores elétricos, classificados em 3 classes:

- A) Normal;
- B) Falha Elétrica;
- C) Falha Térmica.

Considerando um banco de dados contendo 100 transformadores, onde 47 são Normais, 24 possuem falha elétrica e 29 possuem falha térmica, uma possível classificação é mostrada na seguinte matriz de confusão:

Tabela 2.5 – Exemplo de Matriz de Confusão gerada em um problema de classificação.

Real \ Classif.	Normal	Falha Elétrica	Falha Térmica
Normal	40	2	5
Falha Elétrica	6	16	2
Falha Térmica	4	2	23

No exemplo dado, o número de classificações corretas é mostrado na diagonal principal da matriz: 40 normais, 16 com falhas elétricas e 23 com falhas térmicas. Os demais valores constituem classificações incorretas. Por exemplo, 6 transformadores foram classificados como normais, mas na realidade, eles possuem falha elétrica.

Conforme já dito, os coeficientes da matriz de confusão permitem o cálculo de algumas métricas de qualidade dos classificadores para cada uma das classes. Dentre essas métricas, pode-se citar: acurácia; sensibilidade e especificidade, onde

$$\text{Acurácia (A)} = \frac{TP+TN}{TP+FN+FP+TN}$$

$$\text{Sensibilidade (A)} = \frac{TP}{TP+FN}$$

$$\text{Especificidade (A)} = \frac{TN}{TN+FP}$$

Considerando a matriz mostrada na Tabela 2.5, temos os seguintes valores referentes à classe ‘Normal’:

$$TP = 40,$$

$$FN = 2 + 5 = 7,$$

$$FP = 6 + 4 = 10,$$

$$TN = 16 + 2 + 2 + 23 = 43.$$

Assim, os valores de acurácia, sensibilidade e especificidade obtidos nessa classe são:

$$\text{Acurácia (Normal)} = \frac{40+43}{40+7+10+43} = 0,83,$$

$$\text{Sensibilidade (Normal)} = \frac{40}{40+7} = 0,85,$$

$$\text{Especificidade (Normal)} = \frac{43}{43+10} = 0,81.$$

De maneira similar, a acurácia, a sensibilidade e a especificidade podem ser calculadas para as demais classes: Falha Elétrica e Falha Térmica.

2.5.1 Representação dos Indivíduos

Existem na literatura diversas formas de se representar os indivíduos de um algoritmo evolucionário. Esta seção irá descrever as principais formas de representação de indivíduos utilizadas em algoritmos genéticos e na programação genética.

2.5.1.1 Algoritmos Genéticos

Basicamente, existem duas formas principais de representar um indivíduo em tarefas de classificação em mineração de dados (FREITAS, 2002), a saber:

- Abordagem Michigan: Cada indivíduo representa uma única regra de classificação;
- Abordagem Pittsburgh: Cada indivíduo representa um conjunto de regras de classificação.

Essas abordagens possuem vantagens e desvantagens que serão destacadas a seguir. A abordagem Michigan, onde cada indivíduo representa uma regra, foi a forma utilizada por Goldberg (1989) em aplicações de aprendizado de máquina (*machine learning*). Conforme já visto, algoritmos genéticos visam evoluir sua população de maneira a gerar um melhor indivíduo que maximize a função *fitness*. Como, em geral, as aplicações de classificação em mineração de dados procuram não por uma, mas por um conjunto de regras, a fim de classificar os padrões em diversas classes (geralmente mais de duas; vide Definição 3) tem-se aqui um primeiro impasse: como fazer com que os algoritmos genéticos gerem um conjunto de regras que classifiquem os padrões em cada uma das

classes, no problema de classificação?

Uma primeira forma seria executar o algoritmo n vezes, onde n é o número de classes. Assim, a cada execução, a melhor regra de classificação seria identificada e, ao final das n execuções, a classificação dos padrões poderia ser feita. A função *fitness* utilizada em cada uma das execuções deve ser adaptada a fim de identificar a melhor regra para a n -ésima classe (vide Definição 3). Evidentemente, essa solução teria como desvantagem o grande esforço computacional.

Goldberg (1989) propôs a utilização da técnica de nicho (seção 2.4.3), a fim de se obter as melhores regras de classificação para as n classes do problema a cada execução do algoritmo. A utilização de técnicas de nicho permite um maior compartilhamento¹⁶ dos indivíduos no espaço de busca. A Figura 2.15 mostra os resultados obtidos com e sem a utilização da técnica de nicho.

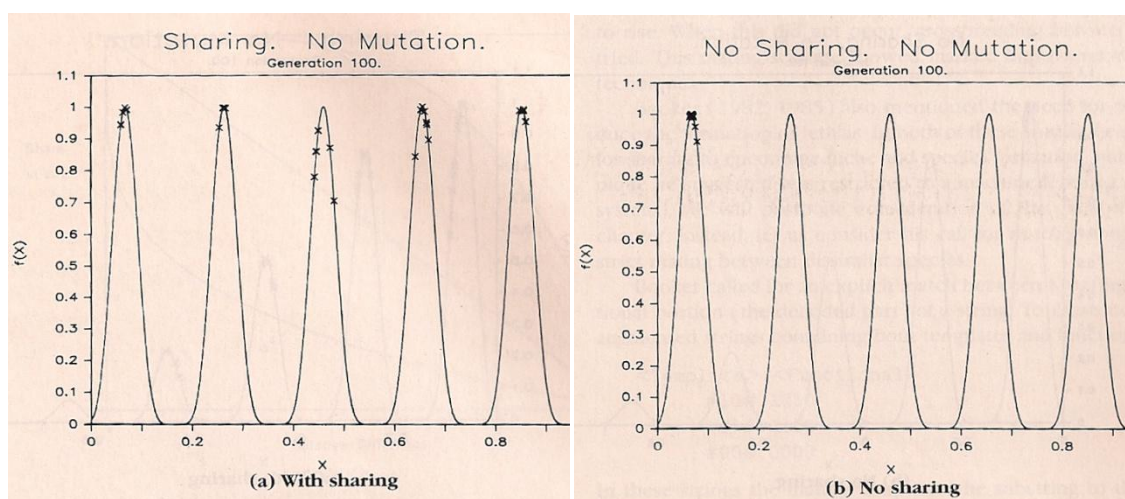


Figura 2.15 – Pontos obtidos pelo AG. O gráfico exibido à esquerda mostra o resultado do algoritmo com utilização de técnica de nicho. O gráfico da direita mostra o resultado sem a técnica (GOLDBERG, 1989, p. 193).

A abordagem Pittsburgh, na qual cada indivíduo representa um conjunto de regras, permite que sejam obtidas regras que identifiquem todos os padrões que pertençam às n classes do problema. Entretanto, as operações de cruzamento e mutação se tornam muito mais complexas.

Freitas (2002) destaca que a abordagem Pittsburgh procura pelo melhor conjunto de regras, entretanto, este pode não ser o conjunto das melhores regras, ou seja, como o AG avalia o conjunto de regras (indivíduo) como um todo, o desempenho de cada uma das regras não é levado em consideração. Desta forma, pode ser que o desempenho do

¹⁶ do inglês *sharing*

melhor conjunto de regras obtido possa ser melhorado, através de um rearranjo dos conjuntos (indivíduos) gerados pelo algoritmo.

Na prática, a maioria das aplicações estudadas utiliza a abordagem Michigan. Essa abordagem será considerada nas discussões feitas a partir deste ponto.

Geralmente, um AG utiliza uma codificação de baixo nível, binária, por exemplo (GOLDBERG, 1989). Entretanto, codificações de alto nível, *e.g.* “Idade < 20”, também podem ser utilizadas. A seguir serão avaliadas as vantagens e desvantagens de cada uma dessas codificações em aplicações de mineração de dados.

Atributos que dizem respeito a categorias podem ser facilmente codificados de maneira binária. Por exemplo, considere o atributo estado civil. Os valores possíveis são: Solteiro, Casado, Divorciado, Viúvo. Dois bits são suficientes para codificar esses valores. Contudo, o atributo contínuo apresenta maiores dificuldades de codificação, principalmente se considerarmos intervalos desses valores. Considere, por exemplo, o atributo Idade. Esse atributo pode ser dividido em intervalos do tipo: 0-19, 20-39, 40-69, 70-∞. Assim, cada um dos intervalos pode receber uma codificação binária, entretanto, seria muito mais simples utilizar uma codificação de alto nível do tipo: $0 \leq idade \leq 19$. Desta forma, a alteração dos limites da restrição fica muito mais simples, facilitando assim a implementação dos operadores genéticos. Uma outra grande vantagem da codificação em alto nível, principalmente quando aplicada em atributos contínuos, está na facilidade de entendimento por parte do usuário final. É muito mais simples e intuitivo entender “ $0 \leq idade \leq 19$ ” do que “ $idade = 00$ ”. Claramente, a segunda codificação não é intuitiva e inclusive induz a uma interpretação errada: o valor ‘00’ significa o primeiro intervalo (de 0 a 19) e não o valor absoluto 0 (zero).

Um grande problema muito estudado pela comunidade científica é descobrir quantos e/ou quais atributos do banco de dados utilizar. Um banco que possui um grande volume de atributos possui também um grande espaço de busca, dificultando a procura pelo melhor resultado. Muitas estratégias podem ser aplicadas a fim de reduzir o volume de atributos, tais como Análise de Componentes Principais (PCA¹⁷), Árvore de Decisão, dentre outras. O AG pode ter seus indivíduos codificados de maneira a variar seu tamanho. Um exemplo de implementação dessa variação de tamanho consiste em implementar um mecanismo onde os atributos podem ser habilitados ou desabilitados em cada um dos indivíduos (regras).

¹⁷ do inglês *Principal Component Analysis*

A primeira estratégia sugere que os atributos possuam uma posição fixa nos indivíduos. Os atributos podem ser “habilitados” ou “desabilitados” em cada indivíduo a fim de se alterar as regras geradas. Essa estratégia será detalhada a seguir.

Considere um problema contendo n atributos e seja $atrib_i$ o i -ésimo atributo, com $i = 1, 2, \dots, n$. A codificação dos indivíduos que permite informar se um determinado atributo será ou não considerado na regra deverá ter as seguintes informações para cada cromossomo:

- p_val_i : Primeiro valor. Geralmente é o limite inferior do i -ésimo atributo;
- p_oper_i : primeiro operador;
- $nome_atrib_i$: nome do i -ésimo atributo;
- s_oper_i : segundo operador
- s_val_i : Segundo valor. Caso o atributo em questão seja contínuo, esse valor será o limite superior;
- hab : Habilitação da regra.

Caso o atributo em questão seja discreto, o primeiro valor (p_val) e o primeiro operador (p_oper) podem ser dispensados. Um exemplo de regra, utilizando as informações conforme descrito anteriormente seria:

‘Estado Civil = Solteiro’ - 1 | ‘0 ≤ idade ≤ 19’ - 1 | ‘Sexo = Masculino’ - 0

Essa regra é formada por 3 condições. Na primeira condição, temos $atrib_1$ igual a *‘Estado Civil’*, s_oper_1 igual a ‘=’, s_val_1 igual a *‘Solteiro’* e hab igual a 1. Esse último valor indica que a restrição está habilitada. As duas condições seguem a mesma estrutura da primeira, cabendo destaque para os seguintes fatos: (a) a segunda regra diz respeito a um atributo de valores contínuos, por isto a regra está representando um intervalo; (b) a terceira regra está desabilitada, pois o valor do parâmetro hab é igual a 0. Assim, a decodificação do indivíduo em questão seria:

(‘Estado Civil = Solteiro’) E (‘0 ≤ idade ≤ 19’)

A segunda estratégia consiste em codificar os indivíduos utilizando um genótipo de tamanho variável. Exemplos de dois indivíduos de genótipos diferentes são:

‘Estado Civil = Solteiro’ | ‘0 ≤ idade ≤ 19’

‘30 ≤ idade ≤ 59’ | ‘Sexo = Masculino’ | ‘Estado Civil = Casado’

É notório o fato de que essa representação trás dificuldades para a implementação dos operadores genéticos, principalmente o cruzamento: é preciso identificar em que

posição do indivíduo os atributos se encontram a fim de se realizar a permuta de material genético.

O detalhamento feito a respeito da codificação dos indivíduos levou em consideração somente as restrições da regra, ou seja, a cláusula “SE”. Agora será detalhada a codificação da consequência da regra, ou seja, a cláusula “ENTÃO”. Considere o seguinte Indivíduo:

$$'Estado Civil = Solteiro' \mid '20 \leq idade \leq 49'$$

A transformação desse indivíduo em uma regra de classificação gerará a seguinte expressão:

$$SE (Estado Civil = Solteiro) E ('20 \leq idade \leq 49') ENTÃO C$$

onde C é uma das classes do problema em questão.

Existem diferentes estratégias para se definir qual é a classe C que irá compor a regra. A primeira estratégia consiste em inserir um campo no indivíduo a fim de codificar a classe. Esse valor seria obtido aleatoriamente e, com a execução do algoritmo, os operadores genéticos poderiam alterar esse valor. A classe seria tratada como uma variável qualquer do indivíduo. Freitas (2002) não recomenda a utilização dessa abordagem, uma vez que o desempenho da regra está intimamente ligado com a relação entre as restrições (cláusula ‘SE’) com a sua consequência (cláusula ‘ENTÃO’). Um indivíduo pode possuir um conjunto de restrições que identifica corretamente uma determinada classe do problema, entretanto a classe considerada nesse indivíduo pode não corresponder à classe correta, fazendo com que o desempenho do indivíduo seja baixo. Por exemplo, considere um indivíduo que seleciona 70 padrões pertencentes à classe c_1 e 5 pertencentes à classe c_2 . Esse indivíduo teria um desempenho ruim se dissesse que os padrões pertencem à classe c_2 , contudo seria uma boa regra para classificar os padrões da classe c_1 .

Uma segunda abordagem, que supriria a grande deficiência apontada na abordagem anterior, consiste em se definir a classe da regra de acordo com os padrões obtidos pela regra. A ideia básica consiste em escolher a melhor classe que identifica o conjunto de padrões selecionados. Considerando o exemplo anterior, como 70 padrões (93% do total) pertencem à classe c_1 e 5 (7%) pertencem à classe c_2 , a melhor classe a ser escolhida para o indivíduo em questão é a classe c_1 .

Uma terceira abordagem consiste em se dividir a população em subpopulações, onde cada uma dessas partes será associada a uma das classes do problema. Dessa

forma, as operações de cruzamento ocorrerão entre indivíduos de uma mesma subpopulação.

2.5.1.2 Programação Genética

Conforme visto na seção 2.4.4.1, uma propriedade a ser mantida na PG é a de fechamento, a qual diz que a saída de um nó interno (função) ou terminal pode ser usada como entrada de algum outro nó interno da árvore que representa o indivíduo.

Existem diversos tipos de dados com os quais uma aplicação de mineração de dados lida: valores discretos, tais como Estado Civil, Sexo, entre outros, além de valores contínuos, tais como idade, altura. É preciso que os nós que compõem o indivíduo da PG se relacionem de maneira a garantir a propriedade de fechamento o que não é uma tarefa trivial. A criação de uma regra do tipo “*Idade* > *Altura*” não faz sentido, apesar dos atributos “*Idade*” e “*Altura*” serem do mesmo tipo (valores numéricos positivos inteiros). Esses atributos possuem uma semântica totalmente distinta. Igualmente ruim seria gerar uma regra do tipo “*Sexo* > *Masculino*”. A Figura 2.16 apresenta um exemplo no qual indivíduos inconsistentes são gerados após um cruzamento. Nesse exemplo, o nó “Masculino” foi permutado com o nó “13”. Para se contornar esse tipo de problema e garantir a propriedade de fechamento, algumas soluções serão detalhadas a seguir.

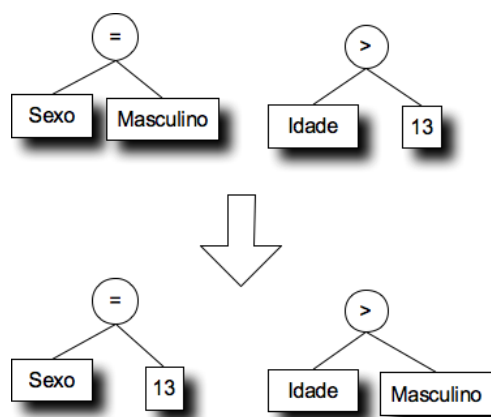


Figura 2.16 - Exemplo de cruzamento onde a propriedade de fechamento não é observada.

A primeira estratégia descrita em Freitas (2002), é a de tornar a saída de todos os terminais em dados do tipo *booleano*¹⁸, onde o valor de saída do terminal possui dois valores, que podem ser considerados como 0 ou 1, verdadeiro ou falso. A estrutura dos

¹⁸ do inglês *Booleanizing All Terminals*.

terminais é da forma $\langle \text{atrib}, \text{ope}, \text{val} \rangle$, onde o *atrib* é o atributo, *ope* é um operador relacional ($=$, $<$, $>$, \leq , \geq) e *val* é um valor pertencente ao domínio do atributo em questão. A Figura 2.17 mostra um exemplo de indivíduo da PG onde todos os terminais foram dispostos no tipo *booleano*.

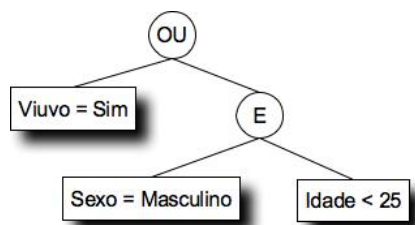


Figura 2.17 – Exemplo de Indivíduo da PG com os terminais do tipo *booleano* (FREITAS, 2002).

A regra gerada pelo indivíduo da Figura 2.17 é dada por:

$$SE (Viúvo = Sim) OU ((Sexo = Masculino) E (Idade < 25))$$

A cláusula ‘ENTÃO’ pode ser determinada por alguma das estratégias descritas na seção 2.5.1.1. Maiores detalhes sobre o método de transformar o tipo dos atributos e terminais em booleanos podem ser encontrados em (FREITAS, 2002, p.140-146).

Uma segunda estratégia utilizada a fim de se garantir a propriedade de fechamento consiste em utilizar na árvore da PG nós fortemente tipados¹⁹, isto é, cada nó da camada *k* só pode aceitar como nós da camada *k+1* que tenham o tipo de dado compatível com o que esse nó *k* espera.

O conceito “fortemente tipado”²⁰ foi utilizado primeiramente e ainda é muito utilizado atualmente para descrever uma característica de linguagens de programação. Dizer que uma linguagem de programação é fortemente tipada, significa que todos os possíveis valores a serem atribuídos a uma variável devem pertencer um determinado tipo. Por exemplo, uma variável cujo tipo de dado é ponto flutuante (*float*), só pode receber valores do tipo ponto flutuante. A tentativa de atribuir um valor de tipo diferente, por exemplo, o tipo caractere (*char*), gerará um erro.

De maneira análoga à descrita anteriormente, pode-se dizer que um algoritmo de PG é fortemente tipado quando os nós da árvore que forma cada um dos indivíduos, possuem tipos de dados compatíveis. Pode ser criada uma tabela contendo a definição dos tipos dos parâmetros de entrada bem como o retorno das funções, a fim de auxiliar a verificação dos tipos, durante a execução do algoritmo. A Tabela 2.6 mostra um

¹⁹ O nó é fortemente tipado se não se permitem violações dos tipos de dados que ele pode receber.

²⁰ do inglês *Strongly-Typed*

exemplo das definições dos parâmetros e retorno das funções.

Tabela 2.6 – Definição dos tipos de dados de entrada e saída das funções.

Função	Tipo de dado dos parâmetros	Tipo de dado retornado
+, -, *, /	(real, real)	real
<, >, ≤, ≥	(real, real)	booleano
AND, OR	(booleano, booleano)	booleano
=	(inteiro, inteiro)	booleano

A implementação de um algoritmo de PG fortemente tipado implica em modificações na estrutura convencional desse algoritmo. A função de criação dos indivíduos, bem como os operadores genéticos devem ser implementados de maneira a garantir que sejam criados e manipulados somente indivíduos válidos. A Figura 2.18 mostra um exemplo de indivíduo da PG fortemente tipada. Esse indivíduo foi criado levando em consideração a tipagem descrita na Tabela 2.6.

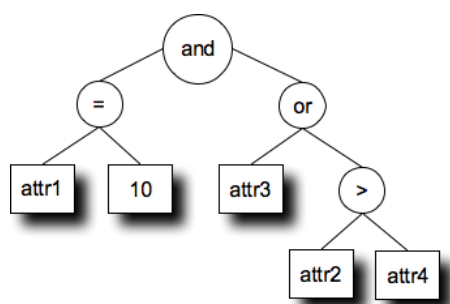


Figura 2.18 – Exemplo de Indivíduo da PG Fortemente Tipada.

Nesse exemplo, seguindo a tipagem da Tabela 2.6, identifica-se que *attr1* é do tipo inteiro, *attr3* é do tipo booleano e os atributos *attr2* e *attr4* são do tipo real.

A operação de mutação pode ser implementada conforme o pseudocódigo apresentado no Algoritmo 2.4.

Algoritmo 2.4 – Pseudocódigo da operação de Mutação na PG Fortemente Tipada.

1.	Selecione aleatoriamente um nó <i>n</i> ;
2.	Gere outro nó <i>n'</i> do mesmo tipo do nó <i>n</i> ;
3.	Substitua o nó <i>n</i> pelo nó <i>n'</i> ;

É importante destacar que, no pseudocódigo apresentado no Algoritmo 2.4, o nó *n* pode ser a raiz de uma sub-árvore. Nesse caso, o nó *n'* gerado poderá ser um nó simples ou outra sub-árvore. Contudo tanto o nó *n* quanto o *n'* devem ser de um mesmo tipo de dado.

A implementação da operação de cruzamento é mostrada de maneira genérica no Algoritmo 2.5.

Algoritmo 2.5 – Pseudocódigo da operação de Cruzamento na PG Fortemente Tipada.

entrada (Indivíduo $p1$, Indivíduo $p2$, inteiro max)
saída (Filho $f1$, Filho $f2$)
1. tentativas := 0; $n1$:= <i>vazio</i> ; $n2$:= <i>vazio</i> ;
2. $f1$:= COPIA($p1$); $f2$:= COPIA($p2$);
3. enquanto (tentativas < max) E ($n2$ = <i>vazio</i>) faça
4. Selecione aleatoriamente um nó $n1$ de $f1$;
5. Procure em $f2$ um nó n_temp do mesmo tipo do nó $n1$;
6. se encontrou um nó n_temp
7. $n2$:= n_temp ;
8. fim enquanto ;
9. se ($n2$ = <i>vazio</i>)
10. retorne $p1$ e $p2$;
11. Senão
12. Troque os nós $n1$ e $n2$ de $f1$ e $f2$;
13. retorne $f1$ e $f2$;
14. fim se ;

A operação de cruzamento, apresentada no Algoritmo 2.5, procura selecionar dois nós de um mesmo tipo tanto na primeira quando na segunda árvore²¹ recebidas como parâmetros de entrada. Caso encontre esses dois nós de um mesmo tipo, o algoritmo realiza a permuta desses nós (ou sub-árvores) entre as duas árvores em questão: o nó $n1$ sai da árvore $f1$ e vai para o lugar do nó $n2$ na árvore $f2$; o nó $n2$ sai da árvore $f2$ e vai para o lugar do nó $n1$ na árvore $f1$. Caso não sejam encontrados esses dois nós de um mesmo tipo durante um número específico de tentativas, (o limite de tentativas, max , é passado como parâmetro) o algoritmo retorna uma cópia inalterada dos dois indivíduos recebidos como parâmetros.

2.6 Trabalhos Relacionados

A literatura científica possui uma diversidade de estratégias que vêm sendo aplicadas em problemas de extração de conhecimento não trivial a partir de bancos de dados: abordagens não supervisionadas (DUDA et al., 2000), tais como k-vizinhos mais próximos (*k-Nearest Neighbors*, k-NN) e classificadores Bayesianos (KITTLER et al., 1998), além de algoritmos supervisionados, tais como Árvores de Decisão (AD) (seção 3.4), Redes Neurais Artificiais (RNA) (HAYKIN, 2008) e SVM (seção 3.3). Dentre os artigos estudados ao longo deste trabalho, os quais aplicam os algoritmos citados a fim

²¹ Na PG, cada indivíduo representa uma árvore.

de extrair regras de classificação, pode-se citar:

- k-NN: (WANG et al., 2006), (YANG et al., 2007), (JIANG et al., 2008) e (CHANG et al., 2011);
- Classificadores Bayesianos: (BATTIT; COLLA, 1994), (HU et al., 2007), (BRESSAN et al., 2009) e (ZHANG, 2011);
- AD: (EXARCHOS et al., 2007), (CHEN; HUNG et al., 2009), (SAMANTARAY, 2010) e (SARKAR et al., 2012);
- RNA: (LIAO, 2005), (ARAÚJO et al., 2005), (MANTAS et al., 2006) e (LEI; GHORBANI, 2012);
- SVM: (LIN, 2002), (CHEN; WANG, 2003), (MARTENS, 2007) e (FU; LEE, 2012).

Existem também muitos trabalhos focados na resolução de problemas de classificação, que propõem métodos baseados em algoritmos evolucionários (FREITAS, 2002), tais como AG (seção 2.4.3), PG (seção 2.4.4), Sistemas Imunológicos Artificiais (ALVES et al., 2004) (WANG et al., 2011), Algoritmos da Otimização da Colônia de Formigas (PARPINELLI et al., 2002) (KABIR et al., 2012) e Otimização de Enxame de Partículas (SOUSA, 2004) (MOHAMAD et al., 2011).

Algoritmos que sejam capazes de manipular diretamente bases de dados híbridas, sem um pré-processamento ou uma estrutura particular de representação destes dados, não foram encontrados na literatura. Uma base de dados híbrida é composta por atributos convencionais (e.g. numéricos, textuais, lógicos) e não convencionais (e.g. geográficos). Geralmente, os algoritmos que manipulam dados híbridos adotam algum tipo de estrutura particular para representar os atributos não convencionais. Por exemplo, (LEE et al., 2009) (WANG et al., 2009) (WHIGHAM, 2000), que manipulam dados geográficos, aplicam estruturas particulares para representar as entidades geográficas. Na maioria das vezes, estas estruturas alternativas de representação não são desejáveis, uma vez que o desempenho do algoritmo de classificação e a interpretação dos resultados se tornam altamente dependentes da estrutura adotada. Atualmente, existem muitos gerenciadores de bancos de dados que utilizam um modelo padrão para representar e armazenar os atributos (EGENHOFER, 1993) (ESTER et al., 2000). Esses gerenciadores de bancos de dados contêm extensões geográficas, dotadas de um conjunto de funções que são amplamente conhecidas, mas somente um pequeno número de algoritmos de classificação é capaz de explorar esse recurso. Além do mais, os

métodos existentes (BOGORNÝ et al., 2006) necessitam de um pré-processamento dos dados, fazendo com que não seja possível combinar a aplicação de funções convencionais e não convencionais em uma mesma regra de classificação. A seguir, alguns trabalhos relacionados serão melhor detalhados.

Srinivasa et al. (2007) descrevem um modelo de Algoritmo Genético Adaptativo SAMGA (*Self-Adaptative Migration Model Genetic Algorithm*), no qual as probabilidades de mutação, cruzamento de cada indivíduo, assim como o tamanho da população, são ajustados dinamicamente. Esse algoritmo é utilizado para extrair regras de classificação de padrões.

Cada regra pode ser representada por uma cadeia de caracteres binária. Considerando o seguinte exemplo:

SE (18 ≤ idade ≤ 21) E (50 ≤ peso ≤ 70) ENTÃO classe = normal

Se existirem somente duas classes, normal (0) e anormal (1), então esta regra pode ser representada no formato apresentado na Figura 2.19 e Figura 2.20.

Idade		peso		classe
18	21	50	70	0

Figura 2.19 – Representação da regra (SRINIVASA et al., 2007).

Idade		peso		classe
00010010	00010101	00110010	01000110	0

Figura 2.20 – Cadeia de Caracteres binários de representação da regra (SRINIVASA et al., 2007).

O SAMGA proposto utilizou a abordagem Michigan. A implementação foi comparada com os resultados de um algoritmo genético simples e os resultados mostraram que o SAMGA obteve maior acurácia em suas respostas.

Sikora et al. (2007) apresentam um *framework*²² para seleção de regras com a utilização de um algoritmo genético aplicado em mineração de dados. O algoritmo considera cada indivíduo como uma regra e a função *fitness* é calculada através da contagem do número de padrões que casam com a regra (indivíduo). Essa contagem considera tanto os exemplos positivos quanto os exemplos negativos. Especificamente, o cálculo da função *fitness* é feito da seguinte maneira:

$$F = \alpha + C.k(p - n)$$

²² Optou-se por manter o termo *framework* por ser amplamente utilizado pelas comunidades de engenharia e computação. Um *framework* é uma estrutura que suporta o desenvolvimento de outro projeto de software. Uma tradução para o termo seria “arcabouço”.

onde α é o número total de padrões que casam com a regra, C é uma constante, k é o número de atributos, p é o número de exemplos positivos cobertos pela regra e n é o número de exemplos negativos também cobertos pela regra.

Cada indivíduo possui, além dos dados referentes aos atributos, uma sequência de bits para dizer se o atributo será considerado ou não pela regra. Exemplo: o seguinte indivíduo (Figura 2.21):

$$\left(\begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \\ [0 \ 0.1] & [0.12 \ 0.24] & [0.23 \ 0.5] & [0.4 \ 0.7] & [0.2 \ 0.87] \\ 0 & 1 & 0 & 1 & 1 \end{array} \right)$$

Figura 2.21 – Exemplo de indivíduo do Algoritmo Genético (SIKORA et al., 2007).

Neste caso somente o X_2 , X_4 e X_5 serão considerados, pois possuem o bit de ativação igual a 1. Assim, a regra representada por este indivíduo é:

$$SE (0.12 \leq x_2 \leq 0.24) E (0.4 \leq x_4 \leq 0.7) E (0.2 \leq x_5 \leq 0.87) ENTÃO classe = positivo$$

O conjunto total de padrões é dividido em dois subconjuntos: treinamento (*training set*) e validação (*test set*). O algoritmo genético recebe o conjunto de treinamento para que sejam geradas as regras para classificação. As regras são aplicadas ao conjunto de validação para que seja avaliada a qualidade de cada uma delas.

Os resultados do *framework* proposto por Sikora e Piramuthu (2007) e de um algoritmo genético simples foram comparados, mostrando a superioridade do *framework* em termos de maior acurácia e menor tempo de execução.

Dehuri et al. (2008) apresentam um algoritmo genético multiobjetivo com elitismo para a geração e classificação (avaliação) de regras (indivíduos) extraídas do banco de dados. Neste trabalho, o cálculo da função objetivo leva em consideração a acurácia, a especificidade e a sensibilidade. Estes valores são utilizados para que as regras encontradas maximizem o número de positivos verdadeiros e minimizem o número de falsos positivos extraídos do banco de dados. Os resultados mostraram que as regras apresentam uma acurácia alta nos testes realizados.

Pappa et al. (2009) mostram um algoritmo multiobjetivo, baseado em gramáticas e na programação genética (MOGGPA) que é aplicado na classificação de dados. Os objetivos principais do método proposto são gerar um modelo de regras com grande acurácia e baixa complexidade (tamanho). O uso de uma gramática associada à programação genética traz como grande benefício a possibilidade de se implementar um mecanismo de gerar somente indivíduos consistentes (propriedade de fechamento, seção

2.4.4.1), uma vez que se procura manipular indivíduos válidos pela gramática (WHIGHAM, 1995), (McKAY et al., 2010). O algoritmo proposto foi aplicado em 20 bases de dados disponíveis no repositório *UCI Machine Learning* e comparado com outros algoritmos clássicos. Os resultados mostraram que o método proposto apresentou resultados superiores na maioria dos casos.

Shintemirov et al. (2009) apresentam um método no qual a programação genética é associada com algoritmos de classificação tais como redes neurais artificiais (RNA), Máquina de Vetor Suporte (SVM) e K-Vizinhos Próximos (KNN), a fim de aumentar a acurácia destas ferramentas. O método é aplicado no problema DGA - *Digital Gases Analysis* - (para maiores detalhes vide capítulo 4 e (MORAIS; ROLIM, 2006)). A programação genética é usada a fim de identificar razões entre as concentrações dos gases dissolvidos no óleo isolante de transformadores de potência. Estas razões são utilizadas como entradas para os algoritmos RNA, SVM e KNN. Os resultados dos experimentos mostram que o uso da programação genética, bem como a identificação das razões entre as concentrações dos gases, aumentaram a acurácia dos classificadores.

Zafra e Ventura (2010) propõem um algoritmo, chamado G3P-MI, baseado numa gramática associada à programação genética, o qual é aplicado em um problema de aprendizado de múltiplas instâncias (DIETTERICH et al., 1997). O algoritmo usa regras do tipo SE-ENTÃO codificadas na forma de árvores. Cada indivíduo pode classificar um conjunto de instâncias. O algoritmo foi aplicado em 10 bases de dados e o desempenho do algoritmo foi comparado, apresentando resultado superior, ao desempenho de outros algoritmos de diferentes naturezas.

Arunadevi e Rajamani (2011) apresentam um algoritmo híbrido, o qual combina um algoritmo genético com um algoritmo de otimização baseado em colônia de formigas, a fim de extrair regras de associação para segmentação de clusteres. Entre os objetivos estão a minimização do número de regras além de minimizar o tempo de geração das regras. Os experimentos mostraram que o algoritmo proposto é capaz de gerar um modelo de regras mais simples, porém com acurácia semelhante à apresentada pelas demais ferramentas clássicas.

Romero et al. (2012) propõem a aplicação de um algoritmo genético baseado numa gramática para extração de regras de associação em sistema de aprendizado à distância. O algoritmo foi aplicado numa base de dados real referente a uma disciplina de um curso de graduação. Os resultados obtidos foram utilizados a fim de implementar inovações na condução desse cursos. Foi medido o desempenho dos alunos antes e após

a aplicação dessas inovações. Os resultados mostram as inovações proporcionaram evoluções relevantes no aprendizado dos alunos.

Conforme pode ser visto, nenhum dos trabalhos apresentados propõe uma exploração de dados híbridos. Todas as aplicações estudadas atuam em bases nas quais a natureza dos dados é comum. Na próxima seção serão apresentados algoritmos que são capazes de manipular tipos de dados não convencionais: os dados espaciais.

2.7 Mineração de Dados Espaciais

A mineração de dados espaciais apresenta alguns desafios novos, se comparada com a sua aplicação em dados convencionais. Podemos citar as seguintes particularidades das informações geográficas:

- São dependentes de sua posição no espaço;
- Sofrem alterações com o passar do tempo;
- A vizinhança participa ativamente das alterações ocorridas com os dados;
- Possuem informações textuais e gráficas;
- Informações iguais muitas vezes são encontradas em escalas diferentes;
- Possuem diferentes formas de representação, algumas padronizadas (OPENGIS, 2009) outras não.

Segundo Silva et al. (2006), alguns pesquisadores da comunidade de Tecnologia da Informação têm se dedicado ao problema de integração das informações analíticas e geográficas (HAN et al., 1997) (SHEKHAR et al., 2000) (SHEKHAR; CHAWLA, 2003). Entretanto, a grande maioria dos estudos tem se voltado principalmente para operações do sistema de interface, deixando um pouco de lado a extração de conhecimentos propriamente dita.

Ester et al. (2000) apresentam um grafo de vizinhança e caminhos, além de um conjunto de primitivas de banco de dados para manipulação deste grafo. Esta estrutura de dados procura proporcionar melhorias de desempenho em algoritmos de mineração de dados que exploram informações de vizinhança, tais como:

- Clusterização Espacial;
- Caracterização Espacial;
- Classificação Espacial;
- Detecção de Tendências Espaciais.

O grafo proposto por Ester et al. implementa também um índice espacial, que visa aumentar a eficiência das operações de junção²³ de registros. Diversas comparações foram realizadas com o uso deste grafo, com e sem o uso dos índices espaciais. Os resultados mostraram que o uso dos índices diminuiu o tempo de processamento em até 50% em alguns casos.

Wu e Lu (2002) modelaram o problema de previsão de alocação de equipamentos para transmissão e distribuição de energia. O trabalho publicado por estes pesquisadores apresenta a aplicação de técnicas de mineração de dados para a descoberta de informações úteis para a escolha de regiões destinadas à produção de energia elétrica. As informações obtidas, associadas aos mapas de consumo, permitiram prever quais seriam as melhores áreas para a instalação de estações de produção bem como as linhas de distribuição de energia.

Bogorny et al. (2006) apresentam uma ferramenta que permite a integração entre o Weka, um *framework* que implementa diversos métodos clássicos de mineração de dados, e um sistema de informações geográficas. A ferramenta proposta foi implementada na forma de uma extensão do Weka e é usada da seguinte maneira:

- O usuário informa o banco de dados geográfico a ser utilizado. A seguir, a ferramenta permite ao usuário escolher um conjunto de entidades ou um conjunto de instâncias.
- Após essa escolha, a ferramenta processa as relações geográficas entre os elementos do conjunto montado pelo usuário. Os relacionamentos podem ser divididos em: (a) relacionamentos topológicos (vide seção 2.3.2): *Contains*, *Covers*, *CoveredBy*, *Crosses*, *Disjoint*, *Equal*, *Touches* e *Within*; e (b) relacionamento de distância, onde a distância é um parâmetro informado pelo usuário. É importante destacar que nessa etapa as informações geográficas são processadas de maneira a gerar informações convencionais, em formatos textuais ou numéricos;
- A ferramenta gera um arquivo no formato 'arff' (formato de entrada de dados utilizado pelo Weka) contendo as relações geográficas entre os itens selecionados pelo usuário. Essas informações não possuem mais as características geográficas, como localização e geometria;

²³ do inglês *join*

- O arquivo ‘arff’ é processado pelo Weka. Os algoritmos de mineração de dados, implementados no Weka, são aplicados nos dados armazenados no arquivo fornecido.

Em resumo, o trabalho apresentando por Bogorny et al. (2006) é baseado em um pré-processamento dos dados geográficos a fim de transformá-los em dados convencionais de maneira que métodos de mineração de dados comuns possam processá-los.

Zheng e Xie (2011) propõem um algoritmo de recomendação de viagens à partir da mineração de dados obtidos de diversos usuários de aparelhos de GPS (*Global Position System*). O algoritmo proposto gera dois tipos de recomendações: o primeiro se refere à rota mais procurada pelos usuários, ou seja, aquela a qual a maioria dos usuários faz. A segunda está relacionada às preferências do usuário. Os dados de cada usuário devem ser armazenados em um grafo hierárquico, o qual conterà todo o histórico de cada usuário. As rotas mais executadas são extraídas desse grafo. O trabalho propõe ainda uma forma de extrair uma rota personalizada a partir das preferências de cada usuário.

Phillips e Lee (2012) apresentam uma metodologia de análise de uma base de dados, na qual fatores socioeconômicos e sócio-demográficos são avaliados a fim de extrair regras que identificam situações com alto potencial para surgimento de crimes. Assim, os autores propõem uma representação do banco de dados na forma de um grafo o qual explora as interconexões entre os fatores previamente mencionados e identificam um padrão para o surgimento de um alto índice de criminalidade.

A maioria dos trabalhos estudados propõe novas estruturas para que se possa extrair regras de classificação ou associação. O trabalho que é capaz de manipular os dados geográficos em sua forma natural não é capaz de gerar regras híbridas, as quais podem relacionar tanto os dados convencionais quanto os dados geográficos a fim de extrair conhecimento não trivial dos mesmos.

2.8 Conclusão

Foram apresentados neste capítulo os principais conceitos e definições relacionados às áreas de otimização com algoritmos evolucionários, mineração de dados e mineração de dados em sistemas de informação geográfica. Foram também destacados diversos trabalhos relevantes, relacionados aos temas abordados.

Nota-se que, apesar de ainda existirem poucos trabalhos nesta área, surgem cada

vez mais novos métodos que visam explorar dados não convencionais, principalmente geográficos, em aplicações de mineração de dados, particularmente em classificação de dados. Conforme dito, é satisfatório o número de pesquisas que explora este assunto utilizando imagens. Há porém uma grande variedade de possibilidades a serem exploradas no que diz respeito aos dados vetoriais (seção 2.3.1.1), principalmente quando se leva em conta a possibilidade de explorar esses dados em sua essência, evitando reduzi-lo a estruturas convencionais, limitando seu poder de expressão.

3 ALGORITMOS NÃO EVOLUCIONÁRIOS APLICADOS À CLASSIFICAÇÃO DE DADOS

3.1 Introdução

Existem diversos algoritmos, além dos evolucionários, que podem ser aplicados com sucesso na atividade de classificação de dados. Alguns dos algoritmos mais relevantes, de acordo com a literatura estudada, serão descritos neste capítulo e serão utilizados ao longo de todo o trabalho a fim de comparar os resultados obtidos pelos novos algoritmos propostos.

Pode-se dividir os algoritmos de classificação que serão detalhados a seguir em dois grupos:

- Baseados na minimização do risco estrutural (SRM): Rede Neural *Radial Basis Function* (RBF) e *Support Vector Machine* (SVM);
- Baseados na minimização das informações (atributos) necessários para classificação: Árvore de Decisão, versão ID3 e superiores.

Todos os algoritmos utilizados neste trabalho utilizam o modelo de aprendizado supervisionado, no qual o conjunto de dados é dividido em dois conjuntos: treinamento e testes.

O conjunto de treinamento deve ser usado pelo algoritmo de classificação como forma de obtenção de regras eficientes que sejam capazes de classificar novos dados, ou seja, dados diferentes dos encontrados no treinamento. Desta forma, as regras obtidas são aplicadas no conjunto de testes a fim de verificar sua eficiência e capacidade de generalização. Cabe observar que a obtenção de regras com alto grau de acurácia na etapa de treinamento não é garantia de o mesmo sucesso nos dados de testes. Acontece que regras muito adaptadas a um determinado conjunto de dados tendem a ser ineficientes na classificação de dados novos. Assim, buscar um conjunto de regras que seja mais simples e genérico também deve ser um dos objetivos de um algoritmo de extração de regras de classificação, além, é claro, do objetivo de obter regras com grandes valores de acurácia, ou dizendo de uma maneira mais técnica, buscar por regras que maximizem os valores da diagonal principal da matriz de confusão (Tabela 2.4) e

minimizem os demais valores desta matriz.

A obtenção de um conjunto de treinamento é, portanto, uma etapa muito importante no processo de extração de regras de classificação: é fundamental procurar ao máximo por um conjunto de dados que seja significativo (BRAGA et al., 2007), de maneira tal que o algoritmo a ser aplicado consiga extrair regras eficientes na classificação de dados novos, isto é, dados diferentes dos encontrados no conjunto de treinamento.

A seguir serão detalhados alguns dos métodos mais relevantes encontrados na literatura estudada e que servirão de base nas comparações com os algoritmos propostos neste trabalho.

3.2 Redes Neurais com Função de Base Radial (RBF)

Uma rede neural com funções de base radial, ou simplesmente RBF (*Radial Basis Function*), é um algoritmo que visa a obtenção de um modelo que interpola um conjunto de pontos em um espaço com muitas dimensões (BROOMHEAD; LOWE, 1988) e pode ser utilizado em aproximação de funções, predição de séries temporais, classificação de dados, dentre outras aplicações (BRAGA et al., 2007).

Uma rede neural RBF é composta por três camadas (entrada, camada escondida - com funções não lineares, com muitas dimensões - e saída) e deve gerar uma função de aproximação da seguinte forma (VIEIRA, 2006):

$$f(w, x) = \sum_{i=1}^n w_i \Phi(\|x - c_i\|) \quad (3.1)$$

onde n é o número de neurônios, w_i é o peso (*weight*) do neurônio, $\Phi(\|x - c_i\|)$ é a função de base radial, geralmente não linear, e c_i é o centro da função. A função Φ é aplicada na norma²⁴ (geralmente Euclidiana²⁵) do vetor $x - c_i$. Um exemplo de função de base radial é a função Gaussiana (HAYKIN, 2008, p. 239):

$$\Phi_j(x) = \exp\left(-\frac{1}{2\sigma_j^2} \|x - x_j\|^2\right), \quad j = 1, 2, \dots, N \text{ para algum } \sigma > 0 \quad (3.2)$$

onde σ_j^2 é a variância em j . Cada uma das funções de base radial busca por uma combinação de aproximações locais de tal maneira que o resultado final obtido pela RBF seja uma boa aproximação global (Vieira, 2006, p. 39).

Redes neurais RBF vêm sendo utilizadas largamente em pesquisas cujo objetivo é

²⁴ $\|a\|$ representa a norma do vetor a .

²⁵ a norma euclidiana do vetor a é: $\|a\| = \sqrt{\sum_{i=1}^n |a_i|^2}$

obter uma boa classificação de dados. Assim, diversos aspectos dessas redes são explorados, por exemplo:

- Busca por melhores funções radiais (LIN; PENG, 2011);
- Otimização de diversos objetivos, tais como acurácia e complexidade dos classificadores (QASEM; SHAMSUDDIN, 2011).

Maiores detalhes sobre redes neurais RBF podem ser obtidos em (BRAGA et al., 2007) e (HAYKIN, 2008).

3.3 Máquinas de Vetores Suporte (*Support Vector Machine - SVM*)

Máquinas de Vetores Suporte têm sua origem nas técnicas baseadas em núcleos (*kernels*) e são consideradas hoje como uma das técnicas mais relevantes para classificação de dados, em virtude dos resultados cada vez mais promissores que têm apresentado (LORENA, CARVALHO, 2007).

A versão mais simples de uma SVM visa obter um hiperplano (*i.e.* fronteira) que separa os dados pertencentes a duas classes através da minimização do risco estrutural (CORTES; VAPNIK, 1995). Versões da SVM para classificação de mais de duas classes foram apresentadas e avaliadas (HSU; LIN, 2002) e ainda hoje melhorias são identificadas a fim de melhor explorar dados multi-classe de diversos domínios (MELGANI; BRUZZONE, 2004), (ÜBEYLI, 2008) e (FEKI et al., 2012).

A equação do hiperplano que separa o conjunto de dados é dada pela seguinte fórmula (HAYKIN, 2008, p. 269):

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (3.3)$$

onde \mathbf{x} é o vetor de entrada, \mathbf{w} é o vetor de pesos e $b \in \mathbb{R}$. Esta equação pode ser estendida a fim definir as duas regiões que contêm respectivamente cada uma das classes, da seguinte forma:

$$f(\mathbf{x}) = \begin{cases} \text{se } \mathbf{w}^T \mathbf{x} + b \geq 0 \text{ então } +1 \\ \text{se } \mathbf{w}^T \mathbf{x} + b < 0 \text{ então } -1 \end{cases} \quad (3.4)$$

onde $f(\mathbf{x})$ mapeia em uma das classes $\{+1, -1\}$.

O problema de identificação do hiperplano de separação das classes pode ser definido como um problema de otimização da seguinte maneira (VIEIRA, 2006) (LORENA; CARVALHO, 2007):

$$\begin{aligned} &\text{Minimizar } \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{Sujeito a: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \forall i = 1, 2, \dots, n \end{aligned} \quad (3.5)$$

onde n é a quantidade de padrões do conjunto de treinamento.

O problema de otimização apresentado em (3.5) pode ainda ser melhor trabalhado utilizando-se o método de Lagrange, em sua forma dual (LORENA, CARVALHO, 2007):

$$\begin{aligned} &\text{Maximizar } \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\text{Sujeito a: } \begin{cases} \alpha_i \geq 0, \forall i = 1, 2, \dots, n \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \end{aligned} \quad (3.6)$$

onde α é o vetor que representa os multiplicadores de Lagrange e α^* é a solução ótima para o problema dual.

A solução do problema primal é dada por \mathbf{w}^* e b^* . Conhecendo-se o valor de α^* , pode-se calcular o valor de \mathbf{w}^* a partir da seguinte equação (LORENA, CARVALHO, 2007):

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (3.7)$$

O cálculo de b^* pode ser realizado utilizando o valor de α^* e as condições de Karush-Kuhn-Tucker, da seguinte maneira (LORENA, CARVALHO, 2007):

$$\alpha_i^* (y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1) = 0 \quad \forall i = 1, 2, \dots, n \quad (3.8)$$

Denomina-se como Vetores Suporte aqueles vetores que se encontram exatamente nos hiperplanos de separação dos dados, assim, o valor de $\alpha_i^* > 0$. Pontos dos vetores suporte não participam do cálculo de \mathbf{w}^* . Eliminando as variáveis primais do problema de otimização, a equação que define o hiperplano de separação é dada por (VIEIRA, 2006):

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^n y_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x} + b) \right) \quad (3.9)$$

onde b é calculado utilizando um vetor suporte e a equação (3.8).

A teoria sobre SVM descrita até este ponto não é suficiente para se obter bons resultados em base de dados reais, uma vez que os dados encontrados em problemas práticos não são tão bem comportados, ou seja, é muito comum haver interseções entre

as classes e ainda podem ocorrer ruídos (*outliers*). Outro desafio em aplicações práticas está no fato dos dados não serem linearmente separáveis. Uma primeira abordagem que permite mitigar os efeitos dos problemas descritos é a inclusão de uma folga na superfície de separação dos dados (HAYKIN, 2008). Assim, alguns dados podem violar, com certa tolerância, as restrições descritas na formulação apresentada na equação (3.5) que pode ser reescrita da seguinte maneira (LORENA, CARVALHO, 2007):

$$\text{Minimizar } \frac{1}{2} \|\mathbf{w}\|^2 + C(\sum_{i=1}^n \xi_i) \quad (3.8)$$

$$\text{Sujeito a: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \dots, n$$

onde $\xi_i \in (0,1]$ é a variável de folga e C é um parâmetro que visa ponderar a minimização do erro no conjunto de treinamento. Um bom valor de C precisa ser identificado a partir de testes com dados reais, e varia de acordo com a natureza do problema (HAYKIN, 2008).

SVMs com variáveis de folga são também chamadas de SVM com margens suaves (LORENA, CARVALHO, 2007). De maneira análoga, o problema apresentado pode ser reescrito na forma dual e resolvido utilizando o método de Lagrange.

Conforme já mencionado, em alguns problemas reais os padrões a serem classificados podem não ser linearmente separáveis, de tal forma que nem mesmo uma SVM com margens suaves é capaz de obter um bom modelo de classificação. Uma alternativa para solucionar este problema é mapear os dados do seu espaço original para um espaço com mais dimensões e utilizar uma SVM linear. A Figura 3.1 apresenta um exemplo cujos padrões estão no seu espaço original, onde não é possível criar uma fronteira linear de separação das classes. Em (a) estão todos os padrões identificados por classe e em (b) é mostrada a fronteira não linear de separação dos mesmos. Pode-se utilizar funções não lineares nos vetores de entrada a fim de mapeá-los em um espaço intermediário com mais dimensões, de maneira tal que seja possível identificar um hiperplano linear ótimo que separe as classes (VIEIRA, 2006). A Figura 3.1(c) mostra o resultado desta transformação, além do hiperplano.

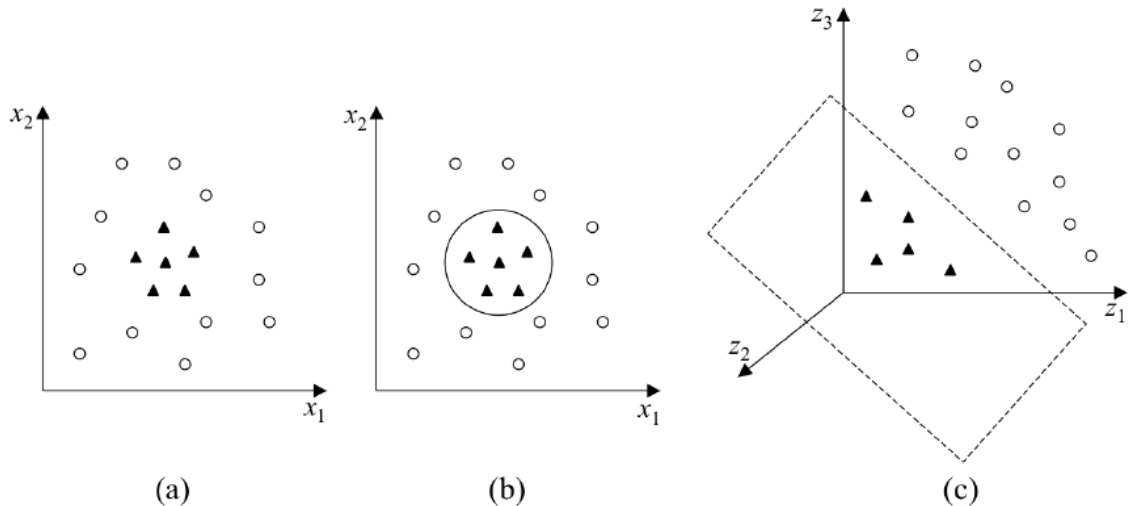


Figura 3.1 - Conjunto de dados não linearmente separáveis. Fonte: (LORENA, CARVALHO, 2007).

A transformação do vetor de entrada pode se dar a partir do núcleo (kernel) do produto interno deste vetor com um vetor suporte. Este núcleo pode ser polinomial, RBF ou ainda Perceptron de duas camadas (VEIRA, 2006) (HAYKIN, 2008).

3.4 *Árvore de Decisão*

Concebida no paradigma de divisão em conquista (CORMEN et al., 2002) as Árvore de Decisão (ADs) são estruturas utilizadas para codificar classificadores de dados. A seguir será detalhado um exemplo chamado "problema do clima", adaptado do problema original apresentado em (QUINLAN, 1986) e melhorado em (WITTEN; FRANK, 2005), onde a solução é obtida utilizando-se AD.

O "problema do clima" consiste em classificar um determinado dia como apto ou inapto para se praticar um esporte. Para tal, estão disponíveis os seguintes atributos com os respectivos valores possíveis:

- Panorama: Ensolarado, Nublado e Chuvoso;
- Temperatura: Alta, Média, Baixa;
- Umidade: Alta, Normal;
- Vento: Forte, Fraco.

A tabela a seguir apresenta os dados coletados referentes a um conjunto de dias, além da identificação da classe Apto ou Inapto para o esporte.

Tabela 3.1 - Dados do Problema do Clima

Panorama	Temperatura	Umidade	Vento	Esporte
Ensolarado	Alta	Alta	Fraco	Inapto
Ensolarado	Alta	Alta	Forte	Inapto
Nublado	Alta	Alta	Fraco	Apto
Chuvoso	Média	Alta	Fraco	Apto
Chuvoso	Baixa	Normal	Fraco	Apto
Chuvoso	Baixa	Normal	Forte	Inapto
Nublado	Baixa	Normal	Forte	Apto
Ensolarado	Média	Alta	Fraco	Inapto
Ensolarado	Baixa	Normal	Fraco	Apto
Chuvoso	Média	Normal	Fraco	Apto
Ensolarado	Média	Normal	Forte	Apto
Nublado	Média	Alta	Forte	Apto
Nublado	Alta	Normal	Fraco	Apto
Chuvoso	Média	Alta	Forte	Inapto

A dificuldade principal no problema do clima consiste em identificar uma regra ou um conjunto de regras de maneira a classificar corretamente um determinado dia. Um exemplo de conjunto de regras é:

SE Panorama = Ensolarado E Umidade = Alta	ENTÃO Esporte = Inapto
SE Panorama = Chuvoso E Vento = Forte	ENTÃO Esporte = Inapto
SE Panorama = Nublado	ENTÃO Esporte = Apto
SE Panorama = Ensolarado E Umidade = Normal	ENTÃO Esporte = Apto
SE nenhuma alternativa anterior	ENTÃO Esporte = Apto

O conjunto de regras deve ser aplicado de cima para baixo, da seguinte maneira: verifica-se se a amostra casa com a primeira regra; se sim, rotula-se o dado de acordo com esta; caso contrário, verifica-se se a amostra casa com a regra seguinte e assim por diante. Cabe destacar que uma regra não precisa conter necessariamente todos os atributos disponíveis do problema. Idealmente, procura-se pelo menor conjunto de regras que possuam somente aqueles atributos mais significativos. Desta forma, o problema poderá ter uma solução eficaz e de fácil interpretação por humanos. Entretanto, isto nem sempre é possível. Durante o desenvolvimento deste trabalho de

doutorado, foram utilizadas algumas bases de dados reais, cujo conjunto de atributos disponíveis era da ordem de dezenas. Nem todos os atributos eram significativos para a concepção do conjunto de regras, o que permite construir um classificador mais simples.

Uma forma de codificar o conjunto de regras descrito anteriormente é mostrada na Figura 3.2. Os nós internos são formados pelos atributos do problema; os nós externos são as classes; as ligações entre os nós se dão a partir de um dos possíveis valores do atributo pai.



Figura 3.2 - Representação de um conjunto de regras de classificação na forma de Árvore de Decisão. Adaptado de (QUINLAN, 1986).

Muitos algoritmos foram propostos a fim de gerarem ADs que sejam capazes de classificar de forma cada vez mais eficiente conjuntos de dados, a saber:

- ID3 (QUINLAN, 1986);
- C4.5 (QUINLAN, 1993);
- J4.8 (WITTEN; FRANK, 2005);
- C5.0, versão comercial, estendida da versão C4.5 (WITTEN; FRANK, 2005).

As ADs são geralmente construídas no modelo *top-down* por algoritmos gulosos (CORMEN et al., 2002) que identificam os atributos com maior ganho de informação (Definição 2). O atributo que proporcionar maior ganho será a raiz da árvore; os atributos subsequentes em termos de ganho serão adicionados como filhos. Este processo se encerra até que todos os dados sejam classificados corretamente ou até que todos os atributos sejam alocados na árvore.

Conforme já mencionado, árvores menores tendem a apresentar um desempenho melhor que árvores mais complexas (QUINLAN, 1986). Desta forma, faz-se necessário utilizar algum algoritmo de poda das árvores geradas. Este algoritmo pode adotar alguma das seguintes estratégias (WITTEN; FRANK, 2005, p. 192): pós-poda

(*postprunning*) ou pré-poda (*preprunning*).

Na pré-poda, procura-se decidir quando a construção de uma sub-árvore deve ser encerrada, durante o processo de construção da árvore como um todo. Assim, evita-se que uma sub-árvore seja montada para depois ser descartada ao final do processo.

A estratégia de pós-poda consiste em avaliar a árvore gerada e remover ou trocar alguns dos seus nós e sub-árvores a fim de simplificar o resultado final. Segundo Witten e Frank (2005) esta é a estratégia mais utilizada, apesar de ser este um tema de pesquisa ainda em aberto, pois não há ainda nenhuma teoria formal que prova a eficácia de uma estratégia sobre a outra.

O uso de AD em aplicações práticas é bastante intenso e amplo, sendo alvo de diversas pesquisas e aplicado de problemas na área de saúde (PAVLOPOULOS, 2004) até predição de enchentes e desastres naturais (SUN et al., 2011).

3.5 Conclusão

Neste capítulo foram apresentados alguns dos algoritmos mais relevantes encontrados na literatura estudada. Eles foram utilizados ao longo de todo o desenvolvimento deste trabalho de doutorado como referência no que diz respeito aos resultados obtidos por cada um deles.

As versões dos algoritmos utilizados são recentes, fornecidas pela ferramenta Weka, a qual permite a configuração e utilização de maneira simples e rápida para o usuário.

4 O ALGORITMO GENÉTICO NGAE PARA CLASSIFICAÇÃO

4.1 Introdução

Este capítulo apresenta um algoritmo genético especialmente proposto para a tarefa de classificação, utilizando banco de dados reais, nos quais é comum a existência de um desbalanceamento na quantidade de amostras de cada uma das classes existentes. O algoritmo proposto, chamado *Niched Genetic Algorithm with Elitism* (NGAE) foi aplicado em um problema real de análise de gases dissolvidos (DGA²⁶) no óleo isolante de transformadores elétricos, a fim de classificar o respectivo estado como: (a) Normal; (b) Falha Elétrica Incipiente; e (c) Falha Térmica Incipiente.

O problema em questão consiste em analisar o óleo isolante e identificar indícios de uma falha em potencial. Neste sentido, o DGA é uma das principais técnicas utilizadas na previsão de uma falha nesses equipamentos (MORAIS; ROLIM, 2006). O óleo, em contato com o calor produzido por um curto ou parte quente, reage com a produção de gases de forma diferenciada, isto é, os gases produzidos são dependentes da intensidade do calor. Assim, a análise desses gases permite identificar que tipo de falha tende a ocorrer ou se o equipamento se encontra em estado normal. Desta forma, procura-se identificar, num transformador em operação, qual é a combinação de gases que está sendo produzida, a fim de detectar uma falha incipiente.

O problema de classificação de dados de análise cromatográfica consiste então em encontrar as regras que permitem determinar se a operação dos transformadores está normal ou se apresenta potencial para falha incipiente. O problema foi modelado como um problema de otimização, no qual se deseja maximizar a seguinte função, para cada uma das classes:

$$f(X) = \text{Acurácia}(X) * \text{Sensibilidade}(X) * \text{Especificidade}(X) \quad (4.1)$$

onde a acurácia, sensibilidade e especificidade são calculadas de acordo com os coeficientes da matriz de confusão (vide Seção 2.5). X representa cada uma das classes do problema.

²⁶ do inglês *Dissolved Gas Analysis*

4.2 Algoritmo Genético NGAE

O NGAE divide o banco de dados em dois conjuntos: treinamento e testes. O conjunto de treinamento (70% dos registros, escolhidos aleatoriamente) é utilizado para avaliar os indivíduos, através da equação (4.1). As melhores regras obtidas devem ser validadas utilizando os registros do conjunto de testes (30% restantes dos registros). A Figura 4.1 mostra o fluxograma de funcionamento da ferramenta.

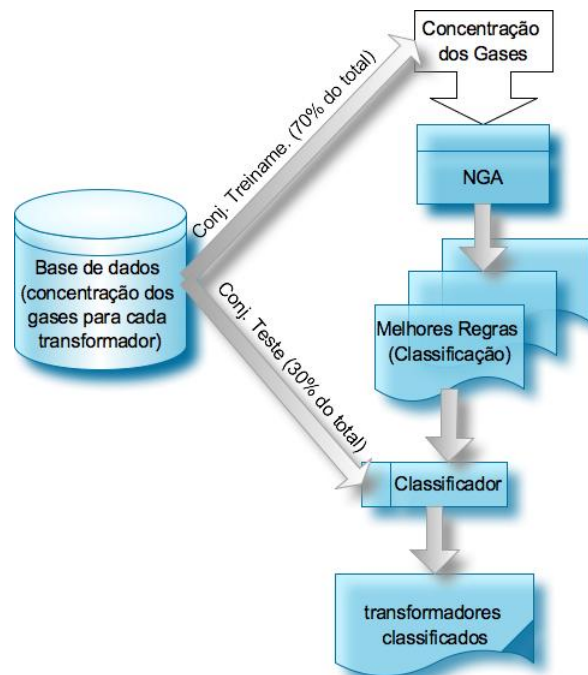


Figura 4.1 – Fluxo de dados do algoritmo NGAE.

O pseudocódigo do NGAE é mostrado no Algoritmo 4.1. Esse algoritmo recebe um conjunto de dados para treinamento e retorna as melhores regras encontradas para a classificação dos padrões em cada uma das classes do problema.

Algoritmo 4.1 – Pseudocódigo do NGAE.

entrada (Conj. Treinamento <i>ct</i> , Número Máximo de Gerações <i>max_ger</i> , Probabilidade de Cruzamento <i>pc</i> , Probabilidade de Mutação <i>pm</i>)
saída (melhores indivíduos para classificar as amostras de cada classe do problema)
1. Gere a população inicial;
2. enquanto <i>número_de_gerações</i> < <i>max_ger</i> faça
3. para cada indivíduo da população faça
4. Procure pelo <i>fitness</i> do indivíduo na memória cache;
5. se (a memória cache não possui a <i>fitness</i> do indivíduo)
6. Acesse o banco de dados (conjunto <i>ct</i>) e calcule a <i>fitness</i> ;
7. Armazena a <i>fitness</i> na cache;
8. fim se ;
9. fim para ;
10. se (<i>número_de_gerações</i> > 1)

11. Aplique elitismo;
12. **fim se;**
13. **para cada** nicho **faça**
14. Selecione os melhores indivíduos do nicho;
15. **fim para;**
16. Aplique os seguintes operadores genéticos: seleção, cruzamento e mutação;
17. Incremente o contador de gerações;
18. **fim enquanto;**
19. **retorne** os melhores indivíduos.

As regras de classificação geradas pelo algoritmo consistem em um predicado lógico que é aplicado na seleção em banco de dados, na forma de uma cláusula WHERE na linguagem SQL (ELMASRI; NAVATHE, 2005). O detalhamento da codificação do indivíduo bem como os operadores genéticos e as demais técnicas utilizadas serão detalhadas a seguir.

4.2.1 Codificação do Indivíduo

Conforme já dito anteriormente, o indivíduo consiste em um predicado definido através da SQL a ser aplicado em uma cláusula WHERE. As tabelas que armazenam o conjunto de dados de treinamento e o conjunto de testes são estruturadas da seguinte forma:

cromatografia(F1, F2, F3, F4, F5, Classe);

onde F1, F2, F3, F4 e F5 são respectivamente as concentrações dos gases (em ppm) H₂ (hidrogênio), CH₄ (metano), C₂H₂ (acetileno), C₂H₄ (etileno) e C₂H₆ (etano) e *Classe* identifica a classificação do transformador. As classes possíveis são: A = Normal; B = Falha Elétrica; C = Falha Térmica.

Os indivíduos manipulados pelo NGAE geram regras da seguinte forma:

SE F1 > x1 E F2 < x2 E F3 > x3 E F4 > x4 E F5 < x5 ENTÃO Classe = A

onde x1, x2, x3, x4 e x5 são valores reais pertencentes ao intervalo $[x_{min_i}, x_{max_i}]$. Os valores, x_{min_i} e x_{max_i} são respectivamente os valores mínimo e máximo encontrados no banco de dados para o atributo F_i , $i \in \{1, 2, 3, 4, 5\}$. Na população inicial, os valores de x1, x2, x3, x4 e x5 são gerados aleatoriamente, respeitando o intervalo descrito anteriormente. Os sinais de comparação “maior que” (“>”) e “menor que” (“<”) também são gerados aleatoriamente da seguinte maneira:

$r = \text{floor}(2 * \text{rand}());$ Se $r = 0$ use ‘>’, senão use ‘<’.

O indivíduo armazena ainda um atributo referente à cláusula ‘ENTÃO’, ou seja, referente à classe. O valor desse atributo é gerado aleatoriamente, com igual

probabilidade de escolha para cada um dos possíveis valores existentes no banco de dados. No problema de classificação apresentado, esses valores possíveis para o atributo são:

- Classe = 'A'
- Classe = 'B'
- Classe = 'C'

Dessa forma, a regra apresentada previamente como exemplo é codificada da seguinte maneira (Figura 4.2):

F1 > X1	F2 < X2	F3 > X3	F4 > X4	F5 < X5	Classe = 'A'
1	1	1	1	1	

Figura 4.2 – Indivíduo do NGAE representando uma regra de classificação.

Os indivíduos são compostos por seis cromossomos, onde os cinco primeiros podem ser desabilitados durante a execução do algoritmo. Na Figura 4.2, é mostrado um bit situado logo abaixo de cada um dos 5 primeiros cromossomos. Esses bits informam se o respectivo cromossomo se encontra habilitado (valor 1) ou não (valor 0). O algoritmo garante que pelo menos um desses cinco cromossomos estará habilitado. Os operadores genéticos podem alterar o valor dos bits que habilitam/desabilitam os cromossomos. Assim, as regras geradas pelos indivíduos podem sofrer alterações de tamanho (para maiores detalhes sobre essa técnica vide seção 2.4.2). Um exemplo de redução de tamanho da regra é dado na Figura 4.3.

F1 > X1	F2 < X2	F3 > X3	F4 > X4	F5 < X5	Classe = 'A'
1	0	1	0	1	

Figura 4.3 – Indivíduo do NGAE com cromossomos desabilitados.

A regra gerada a partir do indivíduo mostrado na Figura 4.3 é a seguinte:

$$SE F1 > x1 E F3 > x3 E F5 < x5 ENTÃO Classe = A$$

4.2.2 Avaliação da *Fitness*

O cálculo da *fitness* é realizado através de consultas SQL, realizadas no banco de dados. Este documento mostra as consultas utilizando a notação usual da álgebra relacional e código SQL-99. Os elementos da álgebra relacional que foram utilizados são mostrados na Tabela 4.1.

Tabela 4.1 – Elementos da Álgebra Relacional.

Elemento	Descrição
F<função>	Função de Agregação. Esse operador aplica uma função (<i>count</i> , <i>max</i> , <i>min</i> , etc.) nos atributos.
$\sigma_{\langle \text{restrição} \rangle}(R)$	Seleção. Aplica uma restrição lógica <restrição> na relação R.
–	Diferença entre conjuntos (representada pelo sinal menos ‘–’). Usada entre relações. Por exemplo, A – B resulta um conjunto que possui todos os elementos de A que não estão em B.

Para o cálculo da *fitness* de cada indivíduo, primeiro faz-se o cálculo dos coeficientes da matriz de confusão (seção 2.5) e em seguida avalia-se a equação (3.1).

Considerando a relação $R(F1, F2, F3, F4, F5, Classe)$, contendo a concentração de gases e a classe de cada um dos transformadores, o cálculo dos coeficientes da matriz de confusão é mostrado a seguir.

4.2.2.1 Cálculo dos Coeficientes da Matriz de Confusão

O valor referente a **verdadeiros positivos** é o número de tuplas cobertas pela regra representada no indivíduo em questão. Esse valor é obtido como resultado da seguinte consulta:

$$F_{\text{count}_{\text{all}}(\sigma_I(R))},$$

onde I é o conjunto de restrições (concentração de gases e a classe) codificadas no indivíduo e R é a relação (tabela) que contem as informações sobre a concentração dos gases e a respectiva classe de cada um dos transformadores.

Usando SQL-99, essa consulta é expressa da seguinte forma:

```
SELECT COUNT (*)
FROM R
WHERE I
```

O **falso positivo** é calculado contando-se o número total de tuplas (amostras) obtidas pela regra codificada pelo indivíduo e que não pertençam à classe predita por essa regra. Considerando o exemplo na Figura 4.3, o cálculo do valor de falsos positivos consiste na contagem de tuplas que satisfazem a restrição: “ $F1 > x1$ E $F3 > x3$ E $F5 < x5$ ” e não pertencem à classe A, isto é, “ $Classe \neq 'A'$ ”. A consulta que realiza essa contagem é:

$$F_{\text{count}_{\text{all}}(\sigma_{I \text{ without Class AND InotClass}}(R))}$$

Usando SQL-99, essa consulta pode ser expressa da seguinte forma:

```
SELECT COUNT (*)
FROM R
WHERE I withoutClass AND InotClass
```

O valor de **verdadeiros negativos** é calculado a partir da contagem do número de tuplas (amostras) que não pertençam à classe em questão, subtraído pelo número de falsos positivos.

A quantidade de **falsos negativos** é calculada contando-se a quantidade de tuplas que pertençam à classe em questão, subtraído do número de verdadeiros positivos.

4.2.2.2 Cálculo da Fitness

Após o cálculo dos coeficientes da matriz de confusão, a função *fitness* é calculada utilizando a fórmula (4.1):

$$F(I,X) = \text{Acurácia}(I,X) * \text{Sensibilidade}(I,X) * \text{Especificidade}(I,X),$$

onde $F(I, X)$ é a função *fitness* do indivíduo I que identifica padrões da classe X .

4.2.3 Mutação

O operador de mutação utilizado no NGAE é o *Bit by Bit* (VASCONCELOS et al., 2001) com igual probabilidade de ser modificado. A mutação pode modificar o valor numérico da restrição, o operador relacional, a classe ou simplesmente mudar a atividade (habilitar ou desabilitar) a restrição. Considerando o exemplo na Figura 4.3, um possível resultado da aplicação da operação de mutação no indivíduo é (Figura 4.4):

F1 < X1	F2 < X2	F3 > X3	F4 > X4	F5 < X5	Classe = 'C'
1	0	1	0	1	

Figura 4.4 – Indivíduo do NGAE após a operação de Mutação.

No exemplo mostrado na Figura 4.4, o operador de mutação modificou o primeiro cromossomo, no qual a restrição “F1 > X1” transformou-se em “F1 < X1” e a classe foi modificada de “Classe = ‘A’” para “Classe = ‘C’”.

Uma vez que o indivíduo foi selecionado para sofrer mutação, as probabilidades de mudança durante a mutação nesse indivíduo são:

- Para cada posição do cromossomo:
 - Existe 50% de chance de ocorrer uma modificação
 - Se a mutação for realizada, cada operador relacional, o valor numérico X_i e o índice de atividade possuem 33% de chance de modificação.
- A modificação na classe do indivíduo acontece com probabilidade de 25%.

É importante destacar que a operação de mutação visa aumentar a diversidade genética a fim de melhorar a busca no espaço das soluções possíveis (vide seção 2.4.2 para maiores detalhes). Assim, o operador de mutação é implementado da seguinte forma (Algoritmo 4.2):

Algoritmo 4.2 – Operador de Mutação do NGAE.

- | |
|---|
| <ol style="list-style-type: none">1. Seleciona-se um conjunto de indivíduos nos quais será aplicada a mutação;2. Gera-se uma cópia de cada um desses indivíduos;3. Aplica-se a mutação em cada uma das cópias geradas;4. Inserem-se esses indivíduos modificados na população. |
|---|

O objetivo de se gerar cópias, efetuar a mutação, e acrescentá-las na população é preservar os indivíduos originais e aumentar a diversidade genética.

4.2.4 Cruzamento

O operador de cruzamento utilizado pelo NGAE é baseado no cruzamento uniforme, apresentado por Vasconcelos et al. (2001), onde todos os cromossomos têm a mesma probabilidade de serem modificados. Um par de indivíduos sempre gera dois outros indivíduos. O algoritmo é implementado da seguinte maneira:

Algoritmo 4.3 – Operador de Cruzamento do NGAE.

- | |
|--|
| <ol style="list-style-type: none">1. Selecione dois indivíduos do mesmo nicho (classe) utilizando o método da roleta (GOLDBERG, 1989);2. Gere uma cópia exata dos dois indivíduos selecionados;3. Aplique o cruzamento da codificação real apresentado em (GOLDBERG, 1989) nos valores numéricos dos cromossomos;4. Para cada cromossomo, troque sua habilitação com probabilidade de 25%;5. Insira os indivíduos filhos na população; |
|--|

O Algoritmo 4.3 mostra que o operador de cruzamento, assim como a mutação, gera uma cópia de cada indivíduo, realiza o cruzamento e insere esses indivíduos novos na população.

O algoritmo cruza indivíduos do mesmo nicho, ou seja, que classificam amostras de uma mesma classe. Um fato que merece destaque é que o cruzamento modifica somente os valores numéricos dos cromossomos, além de permutar sua habilitação com probabilidade de 25%. Entretanto, o operador relacional do cromossomo não é modificado nesta operação.

4.2.5 Nicho e Elitismo

A fim de obter um conjunto de regras que permita a identificação de padrões pertencentes a cada uma das classes do problema, uma técnica de nicho foi utilizada (vide seção 2.4.3.1 para maiores detalhes). No problema em questão, DGA, onde existem três classes, três nichos foram utilizados para identificar as amostras de cada uma das classes. Mais especificamente, a população é dividida em três subpopulações, sendo designado um nicho para cada subpopulação. Cada nicho será responsável por desenvolver os indivíduos de uma determinada classe. Assim, conforme visto na seção 4.2.4, a operação de cruzamento é aplicada em indivíduos pertencentes ao mesmo nicho.

Para garantir que os melhores indivíduos de cada nicho sejam sempre selecionados para as próximas gerações, foi utilizada uma técnica de elitismo. Nos experimentos realizados com o NGAE, 10% do tamanho da população é selecionada para o elitismo a cada nova geração, onde aproximadamente 3% provêm de cada um dos nichos.

Cabe destacar que o algoritmo faz cópias dos indivíduos selecionados para o cruzamento e a mutação e depois os insere na população original. O operador de seleção é aplicado a fim de estabelecer o tamanho original da população. Esta forma de aplicar os operadores genéticos caracteriza um tipo de elitismo, pois todos os indivíduos em questão (pais e filhos) são avaliados conjuntamente, de maneira que os melhores são escolhidos para compor a próxima geração.

4.2.6 Melhorias Implementadas para Avaliação dos Indivíduos

Conforme descrito previamente, o cálculo da função *fitness* de cada indivíduo é realizado através de consultas SQL ao banco de dados. Como essas consultas demandam acesso a disco, que consome muito mais tempo do que acesso à memória principal do computador, um mecanismo simples de memória *cache* foi implementado. Essa memória *cache* consiste em uma tabela *hash* onde a chave é a regra de classificação (na forma de string) e o valor armazenado é o valor da *fitness*. O uso dessa memória *cache* reduziu o tempo do experimento em aproximadamente 80%. Esse valor significativo pode ser compreendido quando se leva em consideração o fato de que algoritmos genéticos utilizam o valor da função *fitness* em diversos momentos a cada geração, em operações de seleção.

Outra técnica utilizada a fim de aumentar a qualidade dos resultados e diminuir o

tempo para se obter boas regras é aplicar uma normalização na concentração dos gases, de maneira a considerar a proporção de cada um deles em cada transformador. Cada concentração é calculada da seguinte maneira:

$$nova_concentração_i = \frac{concentração_real_i}{\sum_{j=1}^n concentração_real_j} \quad (3.3)$$

onde i é o índice da concentração do gás, $n = 5$ e $i \in \{1, 2, 3, 4, 5\}$.

Por exemplo, considerando um transformador com a seguinte concentração de gases (em ppm):

$$H_2 = 10, CH_4 = 9, C_2H_2 = 14, C_2H_4 = 10, e C_2H_6 = 1$$

A proporção relativa de cada um desses gases é:

$$H_2 = 0.227, CH_4 = 0.205, C_2H_2 = 0.318, C_2H_4 = 0.227, e C_2H_6 = 0.023$$

A utilização da técnica de normalização das concentrações dos gases aumentou a acurácia das regras geradas em torno de 15%.

4.3 Experimento e Resultados

O algoritmo proposto foi testado utilizando três bancos de dados distintos, denominados Banco 1, Banco 2 e Banco 3, com 224, 51 e 149 instâncias respectivamente. Os resultados foram comparados com os resultados obtidos com a utilização de outros três métodos de classificação, a saber: *Support Vector Machine* (SVM), Rede Neural *Radial Basis Function* (RBF) e Árvore de Decisão (J48). Todos esses métodos estão implementados na ferramenta Weka. O experimento foi repetido 30 vezes para cada algoritmo, usando cada um dos bancos. Os resultados que serão apresentados consistem nas médias obtidas das execuções de cada algoritmo. Foi realizada uma etapa de configuração do algoritmo, a fim de encontrar a configuração paramétrica que proporcionasse bons resultados em um tempo de execução que não fosse muito superior ao dos demais algoritmos. Os parâmetros utilizados no NGAE, além dos já mencionados anteriormente, são:

- Número de indivíduos na população: 100;
- Número de gerações: 100;
- Taxa de cruzamento: 85%;
- Taxa de mutação: 2%.

A acurácia global observada para cada algoritmo foi utilizada como indicador do desempenho.

A Tabela 4.2 contém uma síntese dos experimentos realizados. Nela, estão indicados os valores da acurácia global obtido por cada algoritmo.

Tabela 4.2 – Acurácia global de cada algoritmo.

	J48	RBF	SVM	NGAE
Banco 1 (224)	0,6842	0,6578	0,7763	0,7170
Banco 2 (51)	0,7058	0,8235	0,5882	0,7120
Banco 3 (149)	0,7800	0,8400	0,8400	0,8197

Alguns dos bancos de dados utilizados são desbalanceados, isto é, o número de amostras pertencentes a cada classe não é similar. Por exemplo, o banco de dados identificado como Banco 3 possui 122 amostras da classe A, 10 da classe B e 17 da classe C. Essa situação faz com que os algoritmos encontrem melhores regras de classificação para a classe predominante (classe A) e piores regras para as demais classes (B e C). Em particular, como o critério de comparação é a acurácia, um alto valor de verdadeiro positivo nas regras da classe predominante ou um alto valor de verdadeiro negativo nas regras das classes não predominantes já é suficiente para garantir um bom valor de acurácia. Os algoritmos clássicos obtiveram bons valores de acurácia por classe na base 3. Como os modelos gerados por essas ferramentas privilegiaram a classe predominante, gerou-se muitos verdadeiros positivos na classe A e muitos verdadeiros negativos nas classes B e C. Esse fato é suficiente para garantir um alto valor de acurácia.

Tabela 4.3 – Acurácia obtida em cada classe.

		J48	RBF	SVM	NGAE
Banco 1	A (84)	0,7105	0,7105	0,8157	0,7866
	B (62)	0,9210	0,8421	0,9442	0,7382
	C (78)	0,7368	0,8815	0,8026	0,8715
	σ	0,1147	0,0895	0,0725	0,0675
Banco 2	A (15)	0,7058	0,8235	0,5882	0,7962
	B (22)	1,0000	1,0000	1,0000	0,8578
	C (14)	0,7058	0,8235	0,5882	0,7120
	σ	0,1699	0,1019	0,2378	0,0732
Banco 3	A (122)	0,8000	0,8600	0,8400	0,8388
	B (10)	0,9200	0,9000	0,9600	0,8840
	C (17)	0,8400	0,9200	0,9600	0,8747
	σ	0,0611	0,0306	0,0693	0,0239

Os resultados estratificados por classe são mostrados na Tabela 4.3. O desvio padrão (σ) foi calculado a fim de auxiliar na comparação, uma vez que os algoritmos apresentaram desempenhos similares em muitos casos. O algoritmo proposto, NGAE,

apresentou o menor desvio padrão em todos os bancos de dados. Esse fato indica que o algoritmo obtém um desempenho similar para todas as classes, deixando evidente a sua maior robustez, pois não privilegiou as classes predominantes.

A fim de obter uma comparação mais criteriosa do ponto de vista estatístico, utilizou-se uma abordagem de comparação proposta por Carrano et al. (2011). Nesta abordagem, um processo de reamostragem (*bootstrapping*) é aplicado nos resultados apresentados por cada um dos métodos, a fim de construir uma função empírica de distribuição de probabilidade (*probability distribution function* - PDF) para o valor médio da acurácia observada. Essas PDFs, que possuem uma distribuição normal devido ao teorema do limite central, são comparadas utilizando One-Way ANOVA e o teste de múltiplas comparações de Tukey (LINDMAN, 1974).

As hipóteses consideradas nos testes são:

$$\begin{cases} H_0: \mu_A = \mu_B = \dots = \mu_n \\ H_I: \exists i, j \in \{1, \dots, n\} \mid \mu_i \neq \mu_j \end{cases}$$

onde μ_i é a média do i -ésimo algoritmo e n é a quantidade de algoritmos. H_0 é a hipótese nula e H_I é a hipótese alternativa.

Estes testes geram uma ordenação dos métodos, a qual é validada utilizando testes de permutação. O resultado final do procedimento é uma ordenação estatisticamente embasada dos esquemas de entradas e o p -valor ²⁷(*p-value*) relacionado com cada uma das classificações (retornada pelo teste de permutação).

A notação adotada para representar os resultados alcançados após as comparações entre os algoritmos é apresentada na forma de uma tabela (Tabela 4.4, Tabela 4.6 e Tabela 4.8). Trata-se de uma matriz na qual são exibidos os p -valores obtidos após a comparação entre os algoritmos. Todos os algoritmos são comparados entre si. Uma vez que o valor de confiança adotado é de 95%, p -valores menores que 0,0500 sugerem que os métodos são estatisticamente diferentes. De posse dos p -valores entre os algoritmos, é possível classificar os algoritmos. As classificações dos algoritmos de acordo com desempenho obtido em cada banco são exibidas na Tabela 4.5, Tabela 4.7 e Tabela 4.9.

Os p -valores obtidos nas comparações realizadas no banco 1 sugerem diferenças entre os algoritmos. Por exemplo, o p -valor entre NGAE e J48 é 0,0038, o que indica 99,62% de confiança para afirmar que o primeiro algoritmo foi melhor que o segundo

²⁷ Assumindo que a hipótese nula é verdadeira, o p -valor pode ser interpretado como a probabilidade de obter um teste estatístico que é no mínimo tão extremo quanto aquele obtido atualmente. A hipótese nula pode ser rejeitada quando o p -valor é menor que o nível alfa de confiança.

no banco em questão.

Tabela 4.4 - p -valor medido para a acurácia global dos algoritmos - Banco 1

	J48	RBF	SVM	NGAE
J48	0,5000	0,0000	1,0000	0,9962
RBF	1,0000	0,5000	1,0000	1,0000
SVM	0,0000	0,0000	0,5000	0,0000
NGAE	0,0038	0,0000	1,0000	0,5000

Tabela 4.5 - Classificação dos algoritmos de acordo com o p -valor - Banco 1

	J48	RBF	SVM	NGAE
Ordem	3	4	1	2

A ordenação dos algoritmos de acordo com o p -valor obtido nos experimentos realizados no banco 1 está disposta na Tabela 4.5. SVM obteve o melhor resultado, seguido pelo NGAE, J48 e RBF.

Tabela 4.6 - p -valor medido para a acurácia global dos algoritmos - Banco 2

	J48	RBF	SVM	NGAE
J48	0,5000	1,0000	0,0000	1,0000
RBF	0,0000	0,5000	0,0000	0,0000
SVM	1,0000	1,0000	0,5000	1,0000
NGAE	0,0000	1,0000	0,0000	0,5000

Tabela 4.7 - Classificação dos algoritmos de acordo com o p -valor - Banco 2

	J48	RBF	SVM	NGAE
Ordem	3	1	4	2

Tabela 4.8 - p -valor medido para a acurácia global dos algoritmos - Banco 3

	J48	RBF	SVM	NGAE
J48	0,5000	1,0000	1,0000	1,0000
RBF	0,0000	0,5000	0,5734	0,0006
SVM	0,0000	0,4266	0,5000	0,0000
NGAE	0,0000	0,9994	1,0000	0,5000

Tabela 4.9 - Classificação dos algoritmos de acordo com o p -valor - Banco 3

	J48	RBF	SVM	NGAE
Ordem	3	1	1	2

Os p -valores obtidos sugerem diferença entre os algoritmos em todos os casos. Chama a atenção o fato de não haver um mesmo algoritmo vencedor em todos os bancos. O NGAE ficou todas as vezes em segundo lugar.

4.4 Conclusão

Este capítulo propôs um novo algoritmo evolucionário que pode ser aplicado em problemas de classificação. O algoritmo utiliza técnicas de nicho, elitismo, memória cache e modela os indivíduos como cláusulas WHERE em SQL. A fim de avaliar o desempenho do algoritmo proposto, foi utilizado um problema de classificação (DGA, voltado à previsão de falhas incipientes em transformadores de potência). Algoritmos clássicos na tarefa de classificação, tais como Rede Neural, SVM e Árvore de Decisão, foram utilizados, a fim de comparar seus resultados com os obtidos pelo algoritmo proposto. A comparação dos resultados mostra que o método NGAE proposto é competitivo e robusto, uma vez que apresentou resultados similares e com menor desvio padrão em todos os casos.

O algoritmo apresentado, assim como a maioria dos algoritmos genéticos, possui a desvantagem de proporcionar pouca flexibilidade na codificação das regras geradas. Por exemplo, cada atributo possui uma posição fixa no indivíduo, definida pelo programador. Além disto é difícil codificar um indivíduo utilizando-se muitos atributos, pois o tamanho do indivíduo cresce numa proporção linear em relação à quantidade de atributos do banco de dados. A utilização de uma árvore na codificação do indivíduo acaba com essa desvantagem, e foi por isso que a continuidade desse trabalho se deu utilizando-se a programação genética, conforme poderá ser visto a seguir.

5 PROGRAMAÇÃO GENÉTICA EM MINERAÇÃO DE DADOS HÍBRIDOS

5.1 Introdução

Este capítulo apresenta um algoritmo baseado na programação genética, especializado na tarefa de classificação de dados híbridos. O algoritmo, chamado “*Niched Genetic Programming Algorithm for Geographic Data Mining*”, DMGeo, foi especialmente desenvolvido a fim de poder lidar com dados convencionais e não convencionais numa mesma regra de classificação. Trata-se de um algoritmo inovador, uma vez que as soluções encontradas na literatura não são capazes de lidar com dados convencionais e não convencionais ao mesmo tempo. No algoritmo desenvolvido, o indivíduo foi modelado seguindo o mesmo princípio apresentado no capítulo anterior, onde esse indivíduo deve representar um predicado lógico, definido da mesma maneira que uma cláusula WHERE da SQL. O desempenho de cada indivíduo no problema é medido através da avaliação de uma função *fitness*, definida na seção 5.2.2. Utilizou-se novamente uma técnica de nicho e elitismo, análogas às apresentadas no capítulo anterior. O algoritmo DMGeo foi testado e comparado com outras soluções similares, mostrando ser uma alternativa promissora para tratar esta classe de problemas.

5.2 Algoritmo DMGeo

5.2.1 Indivíduo

A Figura 5.1 mostra um exemplo de um indivíduo do DMGeo, no qual operadores lógicos, nomes de atributos e restrições são combinadas de maneira a formar uma cláusula filtro. É importante notar que a árvore representa as restrições e condições para classificar a amostra, entretanto, o indivíduo armazena a classe predita (esperada) fora da árvore, ou seja, a classe predita não é um nó da árvore.

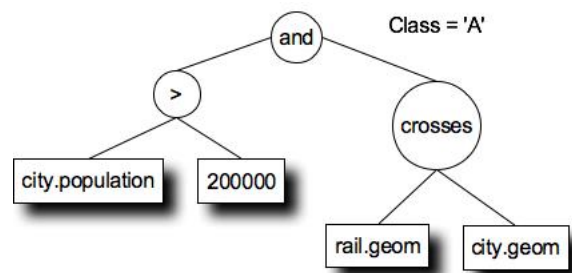


Figura 5.1 – Representação de um indivíduo do DMGeo.

No exemplo mostrado na Figura 5.1, o indivíduo utiliza a função topológica “*crosses*”, a qual determina se as fronteiras da cidade são interceptadas por uma ferrovia (*railway*). A fim de garantir a propriedade de fechamento (seção 2.4.4.1), o algoritmo garante que os tipos de dados das restrições são respeitados, isto é, valores numéricos são comparados somente com valores numéricos e assim por diante.

Os nós da árvore incluem as seguintes informações:

- **Tipo.** Os tipos implementados no DMGeo são booleano (lógico), numérico (real) e geográfico (ponto, linha ou polígono).
- **Corpo do Nó.** O corpo do nó pode ser uma restrição ou uma chamada de uma função geográfica.
- **Parâmetros.** Caso o corpo do nó seja uma chamada de função, esse nó deverá receber parâmetros, isto é, outros nós da árvore. Caso trate-se de um nó terminal, não haverá recebimento de parâmetros.

Os nós da árvore podem ser dos seguintes tipos:

- **Nós Funções:** são formados por funções que, nesta implementação, possuem exatamente dois parâmetros e formam o conjunto possível de nós internos. Podem ser divididos em dois grupos: convencionais e não convencionais (e.g., geográficos).
- **Nós Terminais:** O conjunto de nós terminais é formado por valores gerados aleatoriamente ou por atributos do banco de dados.

O conjunto de nós funções, conforme citado previamente, é composto por funções (ou operadores) convencionais, tais como =, <, >, >=, <=, AND e OR, e por funções não convencionais. Essas últimas funções são implementadas por um banco de dados com extensões geográficas. Neste trabalho foram utilizadas as implementações do PostGIS, as quais incluem as seguintes funções de relações espaciais (STROBL, 2010):

contains (contém), *covers* (cobre), *coveredBy* (coberto por), *crosses* (cruza), *disjoint* (disjunto), *equals* (igual), *touches* (toca), *within* (dentro), e *distance* (distância). O operador *distance* é o único que não gera um valor booleano como resultado. Esse operador retorna um valor numérico que pode ser comparado com outro valor numérico, através de um operador de comparação convencional.

Para criar uma população, o usuário precisa indicar uma tabela alvo, a qual contém os padrões a serem classificados, assim como seus atributos. O conjunto de treinamento deverá conter um atributo que indica a classe à qual a amostra pertence. A criação dos indivíduos do DMGeo é detalhada no pseudocódigo apresentado no Algoritmo 5.1.

Algoritmo 5.1 – Pseudocódigo da geração de Indivíduos do DMGeo.

<p>entrada: Tabela <i>tabela_alvo</i>, /* tabela com todos os atributos dos padrões a serem classificados */</p> <p>saída: Indivíduo <i>I</i>.</p> <ol style="list-style-type: none"> 1. Crie uma lista de possíveis Nós Terminais, <i>folhas_banco_dados</i>, com todos os atributos da <i>tabela_alvo</i>; 2. Crie uma lista de nós não Terminais, <i>nós_internos</i>, contendo todos os operadores (geográficos ou convencionais); 3. Crie um indivíduo vazio chamado <i>I</i>; 4. enquanto <i>cont_nós</i> <= <i>num_max_nós_internos</i> faça 5. Selecione aleatoriamente um nó de <i>nós_internos</i> e chame-o de <i>c</i>; 6. se (<i>I</i> está vazio) 7. Insira <i>c</i> como raiz da árvore; <i>cont_nós</i> := <i>cont_nós</i> + 1; 8. senão 9. Procure na árvore do indivíduo <i>I</i> um nó função que tenha um parâmetro do mesmo tipo de <i>c</i> e que ainda não tenha recebido nesse parâmetro nenhum outro valor; Chame este nó de <i>cn</i>; 10. se (<i>cn</i> <> <i>null</i>) 11. Adicione <i>c</i> como parâmetro (filho) do nó <i>cn</i>; <i>cont_nós</i> := <i>cont_nós</i> + 1; 12. fim se; 13. fim se; 14. fim enquanto; 15. para cada nó <i>i</i> na árvore do indivíduo <i>I</i> faça 16. <i>nt</i> := <i>retorna_nó_função(i)</i>; /* a função <i>retorna_nó_função</i>(índice <i>i</i>) retorna o <i>i</i>-ésimo nó função usando um algoritmo de caminhamento em árvores binárias */ 17. se (<i>nt</i> não possui a lista de filhos completa) 18. se (<i>rand()</i> > 0.5) 19. Selecione um nó, chamado <i>nc</i>, à partir do conjunto <i>folhas_banco_dados</i>; 20. senão 21. Gere um nó, <i>nc</i>, com um valor aleatório; 22. fim se; 23. Adicione <i>nc</i> como parâmetro (filho) de <i>nt</i>; 24. fim se; 25. fim para; 26. Avalie a função <i>fitness</i> de <i>I</i> utilizando todas as classes possíveis; Escolha como classe do indivíduo aquela que obtém o maior valor da <i>fitness</i>; 27. retorne <i>I</i>;
--

O algoritmo de geração dos indivíduos procura criar indivíduos com um determinado número (*num_max_nós_internos*) de nós funções (laço da linha 4 a 14). Após a geração desses nós, o algoritmo completa a árvore com nós terminais (linha 19)

compostos por atributos do banco de dados (*folhas_banco_dados*) ou ainda valores constantes gerados aleatoriamente (linha 21). Evidentemente, a fim de garantir a propriedade de fechamento, os tipos dos parâmetros devem ser respeitados. O algoritmo DMGeo é baseado na PG fortemente tipada (seção 2.5.1.2).

5.2.2 Avaliação da *Fitness*

O cálculo da função *fitness* é totalmente baseado em consultas SQL, submetidas ao banco de dados.

O cálculo da função *fitness* se refere a somente um indivíduo (regra de classificação) por vez e pode ser dividida em duas partes:

- Determinação dos coeficientes da matriz de confusão;
- Cálculo da performance através da função *fitness*.

O DMGeo assume que a relação $R (Attr1, Attr2, ..., AttrN, Classe)$ contém todos os atributos (geográficos ou não) que podem ser usados na classificação.

O cálculo dos coeficientes da matriz de confusão e o cálculo de *fitness* são feitos de forma similar à descrita na seção 4.2.2.

5.2.2.1 Cálculo dos Coeficientes da Matriz de Confusão

O número de **verdadeiros positivos** para cada indivíduo é o número de tuplas selecionadas utilizando o predicado nas quais as respectivas classes coincidem com a classe predita pelo indivíduo. O valor de **verdadeiros negativos** é calculado contando-se o número de tuplas que não são cobertas pelo predicado do indivíduo e que não pertençam à classe do indivíduo.

O número de **falsos positivos** é identificado contando-se as tuplas obtidas pelo indivíduo (regra) que não pertençam à classe predita. Considerando o exemplo na Figura 5.1, o cálculo de falsos positivos consiste na contagem do número de tuplas que satisfazem o predicado “(city.population > 200.000) AND Crosses (rail.geom, city.geom)” e possuem “Class \diamond ‘A’”. Analogamente, o número de **falsos negativos** é calculado contando-se o número de tuplas que não são obtidas pelo indivíduo, mas que pertencem à classe esperada pelo indivíduo.

5.2.2.2 Cálculo da Função *Fitness*

O cálculo da função *fitness* utiliza os valores de acurácia, sensibilidade e especificidade

medidos para cada indivíduo. Para maiores detalhes, vide seção 2.5.

A fórmula utilizada nesse trabalho é:

$$f(I, X) = \text{Acurácia}(X) * \text{Sensibilidade}(X) * \text{Especificidade}(X) \quad (5.1)$$

onde $f(I, X)$ é o valor de *fitness* e I é o indivíduo que identifica padrões pertencentes à classe X .

5.2.3 Mutação

Diferentemente do que acontece nos algoritmos genéticos, o operador de mutação na programação genética não é simples de ser implementado. Primeiramente, é necessário ter certeza de que a árvore do indivíduo se mantém válida após a mutação, isto é, o operador de mutação não pode substituir um nó (ou sub-árvore) por um nó de tipo de dado diferente.

Conforme descrito na seção 2.4.4.3, existem quatro categorias de mutação: ponto, colapso, expansão e sub-árvore. O DMGeo utiliza as quatro categorias. É importante destacar que, independente da categoria da mutação, este operador trabalha da seguinte forma:

1. Seleciona-se aleatoriamente um nó (terminal ou função);
2. Gera-se outro nó (simples ou uma sub-árvore) do mesmo tipo do nó selecionado do passo 1;
3. Substitui-se o nó selecionado pelo novo nó gerado.

O pseudocódigo do operador de mutação é mostrado no Algoritmo 5.2.

Algoritmo 5.2 – Pseudocódigo do operador de Mutação do DMGeo.

<p>entrada: População <i>pop</i>, Probabilidade <i>PM</i></p> <ol style="list-style-type: none"> 1. Selecione aleatoriamente $pm * \text{tamanho_da}(pop)$ indivíduos e insira-os no conjunto <i>indivíduos_mutação</i>; 2. para cada indivíduo I em <i>indivíduos_mutação</i> faça 3. Selecione aleatoriamente um nó n do indivíduo I; 4. Gere outro nó n' do mesmo tipo do nó n; 5. Substitua o nó n pelo nó n'; 6. se (n' é um nó Interno) 7. Complete o nó n' com nós terminais gerados aleatoriamente; 8. fim se; 9. fim para; 10. Insira o conjunto <i>indivíduos_mutação</i> na população <i>pop</i>;

O Algoritmo 5.2 mostra que a mutação gera novos indivíduos (“mutantes”) baseados em elementos selecionados dentro da população. Esses novos indivíduos são

inseridos na própria população original e a remoção do excesso fica por conta do operador de seleção. Vale destacar que as linhas 6 a 8 se destinam a preencher com nós terminais a sub-árvore que irá substituir o nó n . Esses nós terminais são gerados aleatoriamente, porém devem respeitar a propriedade de Fechamento, isto é, os nós gerados são do mesmo tipo de dado esperado pelo nó não terminal n' .

5.2.4 Cruzamento

O operador de cruzamento utilizado no algoritmo DMGeo é baseado na implementação clássica desse operador na programação genética. A estrutura básica do operador é mostrada a seguir:

1. Selecione dois indivíduos, de um mesmo nicho, utilizando a técnica da roleta (GOLDBERG, 1989);
2. Gere uma cópia de cada um desses indivíduos;
3. Selecione aleatoriamente dois nós, de um mesmo tipo, provenientes dos dois indivíduos copiados;
4. Troque os dois nós selecionados entre os indivíduos.

Para maiores detalhes da implementação desse operador, vide Algoritmo 2.5 na seção 2.4.4.2.

5.2.5 Melhorias na Avaliação dos Indivíduos

Assim como o NGAE, o DMGeo implementou uma memória cache para armazenar o valor da *fitness* dos indivíduos (vide seção 4.2.6).

Outra técnica que visa reduzir o tempo de processamento é a simplificação da geometria de alguns objetos geográficos. O clássico algoritmo de simplificação de linhas, apresentado por Douglas e Peucker (1973), foi utilizado. A quantificação do impacto produzido por essas simplificações vai além do escopo desse trabalho e, portanto, não foi analisada. Entretanto, pretende-se realizar essa quantificação como um trabalho futuro.

5.3 Experimento e Resultados

5.3.1 Problemas Utilizados

O desempenho do algoritmo DMGeo foi avaliado a partir de sua aplicação em quatro problemas de classificação armazenados em bancos de dados disponíveis na internet: dois bancos disponíveis no repositório da *UCI Machine Learning* (2010), denominados ‘*Hepatitis*’ e ‘*Wine*’, os quais são compostos por dados numéricos, e dois outros bancos de dados gerados a partir de dados disponíveis no repositório Geominas (2010), chamados ‘*Infraestrutura*’ e ‘*Desenvolvimento urbano*’, compostos por dados numéricos e dados geográficos.

O banco de dados ‘*Hepatitis*’ consiste em determinar a classe dos pacientes (vivo ou morto) infectados com hepatite, considerando 19 atributos tais como idade (número), sexo (masculino, feminino), fadiga (sim, não) entre outros. Esses atributos podem ser utilizados para criar regras de classificação. Os dados de hepatite estão disponíveis no repositório de bases *UCI Machine Learning*. O conjunto de dados é composto por 155 padrões distintos, onde 32 são classificados como "por morrer" (*die*) e 123 como "por viver" (*live*).

O banco de dados ‘*Wine*’ contém dados de três grupos diferentes de vinho, de acordo com sua origem. O banco é composto por 178 instâncias, das quais 59 são de origem A, 71 origem B e 48 origem C. Os dados armazenados por esse banco são constituídos por 13 atributos numéricos, tais como acidez e intensidade da cor. Os valores desses atributos foram obtidos a partir de uma análise química.

O banco ‘*Desenvolvimento urbano*’ contém um conjunto de cidades classificadas em três classes, de acordo com o seu respectivo nível de desenvolvimento: alto, médio e baixo. Existem 852 cidades, das quais 264 pertencem ao nível ‘alto’ (classe A), 296 pertencem ao nível ‘médio’ (classe B) e 292 pertencem ao nível ‘baixo’ (classe C). O problema armazenado nesse banco é constituído por 22 atributos numéricos, tais como quantidade de escolas (públicas e privadas), quantidade de indústrias consumidoras de energia elétrica, índice GINI, dentre outros. O banco ainda contém atributos geográficos, tais como limites municipais (armazenados como polígonos), ferrovias e rodovias (armazenadas como linhas poligonais).

O banco ‘*infraestrutura*’ contém dados de um conjunto de 685 regiões classificadas em três classes: 'alta' (classe A, com 72 amostras), 'baixa' (classe B, com 489 amostras)

e 'média' (classe C, com 124 amostras). Nesse problema, há informações geográficas, tais como os limites das regiões, aeroportos e rodovias, além de atributos convencionais, tais como número de instalações elétricas, população, quantidade de indústrias, dentre outros.

Os bancos de dados provenientes do repositório da UCI foram utilizados a fim de avaliar o desempenho do DMGeo utilizando somente atributos numéricos. A maior contribuição do algoritmo proposto é lidar com bancos de dados compostas tanto por atributos numéricos quanto por atributos geográficos. O repositório Geominas possui dados com essa propriedade.

A fim de realizar uma comparação, três algoritmos foram utilizados para resolver os problemas apresentados. Esses algoritmos são: árvore de decisão (J48), rede neural *Radial Basis Function* (RBF) e *Support Vector Machine* (SVM). Os resultados obtidos com essas três técnicas foram comparados com os resultados obtidos pelo DMGeo.

Todos os algoritmos utilizaram validação cruzada com 5 partições (*folds*). O experimento foi repetido 6 vezes a fim de gerar 30 resultados para cada método, usando cada um dos bancos. Após uma etapa de configuração paramétrica, optou-se por utilizar no DMGeo um tamanho da população = 200, número de gerações = 200, probabilidade de cruzamento = 90% e probabilidade de mutação = 2%.

5.3.2 Resultados e Análise

A síntese dos 30 resultados obtidos são apresentados à seguir.

Tabela 5.1 – Acurácia de cada algoritmo.

	J48	RBF	SVM	DMGeo
<i>Vinho</i>	0,9090	0,9788	0,9789	0,9408
<i>Hepatite</i>	0,7723	0,8199	0,8491	0,7707
<i>Infraestrutura</i>	0,6198	0,5795	0,5703	0,7272
<i>Desenv. urbano</i>	0,6204	0,5786	0,5708	0,8193

A Tabela 5.1 mostra a acurácia global obtida em cada algoritmo. A Tabela 5.2 mostra o resultado obtido por cada um dos algoritmos em cada uma das classes dos problemas, além do desvio padrão (σ) dos valores por classe. É importante enfatizar que um baixo valor do desvio padrão significa que o método foi capaz de obter uma classificação homogênea entre as classes, ou seja, o método não privilegia uma classe em relação às outras.

Tabela 5.2 – Acurácia obtida em cada classe

		J48	RBF	SVM	DMGeo
Vinho	A	0,9720	0,9194	0,9785	0,9408
	B	0,8710	0,9589	0,9797	0,8995
	C	0,8885	0,9799	0,9597	0,9484
σ		0,0540	0,0120	0,0112	0,0248
Hepatite	A	0,7705	0,8208	0,8496	0,6511
	B	0,7711	0,8212	0,8487	0,7772
σ		0,0009	0,0007	0,0004	0,0710
Infraestrutura	A	0,7802	0,7787	0,7819	0,7515
	B	0,5112	0,4689	0,4707	0,6505
	C	0,5713	0,4693	0,4696	0,5799
σ		0,1374	0,1787	0,1800	0,0862
Desenv. urbano	A	0,7804	0,7812	0,7799	0,7500
	B	0,5186	0,4712	0,4698	0,7106
	C	0,5714	0,4707	0,4707	0,6598
σ		0,1385	0,1791	0,1787	0,0452

A comparação estatística aplicada no capítulo anterior (seção 4.3) foi aplicada também nesses experimentos. Os resultados são apresentados a seguir.

Tabela 5.3 - *p*-valor medido na comparação da acurácia global - *Vinho*

	J48	RBF	SVM	DMGeo
J48	0,5000	1,0000	1,0000	1,0000
RBF	0,0000	0,5000	0,5147	0,0000
SVM	0,0000	0,4853	0,5000	0,0000
DMGeo	0,0000	1,0000	1,0000	0,5000

Tabela 5.4 - Classificação dos algoritmos - *Vinho*

	J48	RBF	SVM	DMGeo
Ordem	3	1	1	2

Tabela 5.5 - *p*-valor medido na comparação da acurácia global - *Hepatite*

	J48	RBF	SVM	DMGeo
J48	0,5000	1,0000	1,0000	0,1285
RBF	0,0000	0,5000	1,0000	0,0000
SVM	0,0000	0,0000	0,5000	0,0000
DMGeo	0,8715	1,0000	1,0000	0,5000

Tabela 5.6 - Classificação dos algoritmos - *Hepatite*

	J48	RBF	SVM	DMGeo
Ordem	3	2	1	3

Chama a atenção o empate na terceira posição entre J48 e DMGeo no banco de dados sobre hepatite. Como o *p*-valor é 0,1285 (maior que o valor de confiança de 0,0500) não se pode considerar um algoritmo melhor que outro.

Tabela 5.7 - *p*-valor medido na comparação da acurácia global - *Infraestrutura*

	J48	RBF	SVM	DMGeo
J48	0,5000	0,0000	0,0000	1,0000
RBF	1,0000	0,5000	0,0000	1,0000
SVM	1,0000	1,0000	0,5000	1,0000
DMGeo	0,0000	0,0000	0,0000	0,5000

Tabela 5.8 - Classificação dos algoritmos - *Infraestrutura*

	J48	RBF	SVM	DMGeo
Ordem	2	3	4	1

Tabela 5.9 - *p*-valor medido na comparação da acurácia global - *Desenvolvimento urbano*

	J48	RBF	SVM	DMGeo
J48	0,5000	0,0000	0,0000	1,0000
RBF	1,0000	0,5000	0,0000	1,0000
SVM	1,0000	1,0000	0,5000	1,0000
DMGeo	0,0000	0,0000	0,0000	0,5000

Tabela 5.10 - Classificação dos algoritmos - *Desenvolvimento urbano*

	J48	RBF	SVM	DMGeo
Ordem	2	3	4	1

Os algoritmos RBF e SVM obtiveram um desempenho melhor nos bancos compostos por dados convencionais. O DMGeo obteve bons resultados nos bancos de dados compostos por dados híbridos. As regras geradas neste caso possuem tanto operadores e valores convencionais, quanto operadores e valores geográficos. Esses resultados mostram, portanto, que o DMGeo pode tirar proveito desses atributos geográficos a fim de obter melhores resultados. Outro ponto relevante a ser destacado é o fato do desbalanceamento na distribuição dos dados entre as classes não ter exercido influência significativa nos resultados obtidos no DMGeo.

5.4 Conclusão

Esse capítulo mostrou um novo algoritmo evolucionário proposto para problemas de classificação com dados geográficos. O algoritmo utiliza técnica de nicho, elitismo e memória *cache* a fim de melhorar seu desempenho. Os indivíduos são modelados de maneira a codificar uma cláusula WHERE da linguagem SQL em um gerenciador de bancos de dados espaciais. A fim de avaliar o desempenho do algoritmo proposto, um conjunto de problemas de classificação foi utilizado. Algoritmos clássicos de classificação, tais como Árvore de Decisão, Rede Neural e SVM foram utilizados a fim de se obter resultados para comparação. Os resultados mostram que o algoritmo

proposto é competitivo e robusto, uma vez que obteve os melhores resultados nos casos em que há dados convencionais e geográficos. Nota-se também que o algoritmo é capaz de obter bons resultados para todas as classes do problema, mesmo quando os dados são desbalanceados.

Uma desvantagem do algoritmo proposto nesse capítulo é a geração de regras grandes, o que dificulta a sua interpretação e ainda pode resultar em *overfitting* na classificação dos dados (WITTEN; FRANK, 2005) (TSAI, 2006) (VIEIRA et al., 2006). Essa desvantagem motivou a construção de uma versão multiobjetivo, mais eficiente e robusta, a qual será apresentada no próximo capítulo.

6 ALGORITMO MULTI-OBJETIVO BASEADO NA PROGRAMAÇÃO GENÉTICA E NICHOS PARA CLASSIFICAÇÃO DE DADOS HÍBRIDOS

6.1 Introdução

Este capítulo introduz um algoritmo multiobjetivo baseado na programação genética, resultado da evolução do algoritmo apresentado no capítulo anterior. As principais contribuições desta nova versão são:

- A) Inclusão de um procedimento de controle da complexidade das regras;
- B) Inclusão de um histórico da população a fim de promover elitismo global;
- C) Implementação de um mecanismo de controle da diversidade genética, o qual aumenta a capacidade de obter boas regras de classificação para todas as classes presentes no problema de classificação;
- D) Proposta de três abordagens de uso das regras geradas para a classificação dos dados.

O algoritmo proposto foi aplicado na maximização da efetividade dos classificadores gerados e minimização do tamanho de cada uma das regras. Uma regra efetiva é aquela que identifica corretamente um conjunto de amostras de uma classe. Nos trabalhos anteriores, observou-se que a acurácia não é uma boa medida de efetividade, uma vez que uma regra com um baixo valor de verdadeiro positivo, porém com um alto valor de verdadeiro negativo (muito comum em bancos desbalanceados), possui um alto valor de acurácia.

Observou-se, em versões preliminares do algoritmo, a eliminação de regras com grande efetividade, mas que possuíam um tamanho grande se comparadas à média do tamanho das demais. Isso impactava diretamente o desempenho do método. Uma vez que a efetividade é um objetivo mais importante que o tamanho das regras, propôs-se um mecanismo para atenuar este comportamento indesejável. Esse mecanismo controla a probabilidade de aceitar regras maiores durante a execução do algoritmo, de maneira que esta probabilidade é mantida alta durante as primeiras iterações e vai decrescendo ao longo do processo evolucionário. Ao final da execução, os dois objetivos estão

igualmente balanceados.

O algoritmo proposto foi concebido para resolver uma grande variedade de problemas reais. Além disto, as características do algoritmo sugerem que ele se comporta bem mesmo manipulando bancos de dados desbalanceados. Este tipo de situação complica a resolução de problemas com estas características (MURPHEY et al., 2004), especialmente quando existem mais de duas classes (MENAHEM et al., 2009).

É importante enfatizar que todos os problemas utilizados neste trabalho foram modelados de tal maneira a possibilitar a aplicação de outros algoritmos clássicos a fim de estabelecer uma comparação justa entre os desempenhos mensurados.

6.2 Modelagens do Problema e do Algoritmo

Assim como nos algoritmos propostos anteriormente neste trabalho (Capítulos 4 e 5), o indivíduo é representado por um predicado booleano, definido da mesma maneira que uma cláusula WHERE da instrução de seleção (SELECT) da linguagem SQL (ELMASRI; NAVATHE, 2003). O desempenho do indivíduo no problema é avaliado utilizando um vetor de funções de *fitness*, o qual determina a chance de seleção deste indivíduo.

O problema de classificação é modelado como um problema de otimização biobjetivo, no qual os objetivos são (1) maximizar a efetividade das regras extraídas; e (2) minimizar a complexidade (tamanho) das regras. Assim, o principal objetivo da modelagem é gerar boas regras de classificação e com baixa complexidade (tamanho pequeno), uma vez que regras menores são mais simples de serem entendidas e tendem a evitar *overfitting* (TSAI, 2006) (VIEIRA et al., 2006) (WITTEN; FRANK, 2005). O algoritmo proposto foi modelado para, durante sua execução, gerar somente regras válidas sintaticamente. Uma vez que o primeiro objetivo é mais importante que o segundo, este trabalho propõe um mecanismo externo, que controla a chance de aceitar soluções maiores (no operador de seleção) durante a execução do algoritmo. Nesse mecanismo, a chance de aceitação de soluções maiores é mantida alta durante as primeiras iterações, porém decresce gradualmente ao longo do processo evolucionário. Ao final da execução, os dois objetivos estarão igualmente balanceados.

O algoritmo é dividido em duas fases (1) extração das regras e (2) classificação dos dados. Formalmente, o problema de otimização a ser resolvido é:

$$\text{Maximizar} \left[\frac{\text{Acurácia}(I) * \text{Sensibilidade}(I) * \text{Especificidade}(I)}{1/\text{quantidade_de_nós}(I)} \right]$$

A primeira fase (Algoritmo 6.1) gera regras utilizando o conjunto de treinamento e os conjuntos de funções e de terminais. As melhores regras são utilizadas na segunda fase, na qual três diferentes estratégias de classificação podem ser utilizadas. Os detalhes sobre estas estratégias de classificação são apresentados na Seção 6.7.

Algoritmo 6.1 - Pseudocódigo do algoritmo de extração de regras

Entrada: **Conjunto de Funções** *conjunto_de_funções*, **Conjunto de Terminais** *conjunto_de_terminais*, **Tamanho da População (inteiro)** *n*, **Número de Gerações (inteiro)** *número_max_de_gerações*, **Dados** *dados_de_treinamento*, **Classes** *classes*.

Saída: **Subpopulações**

```

1. Gere n indivíduos usando conjunto_de_funções e conjunto_de_terminais;
2. Identifique a quantidade de classes diferentes em classes. Chame esta quantidade de dc;
3. Crie dc subpopulações;
4. para cada indivíduo i faça                                /* este laço procura pela melhor classe para rotular o indivíduo */
5.     melhor_fitness := 0; classe_do_indivíduo := nulo;
6.     para cada classe cl em classes faça
7.         Usando o conjunto dados_de_treinamento, calcule a fitness f do indivíduo i considerando este indivíduo pertencendo a
           classe cl;
8.         se f > melhor_fitness faça
9.             melhor_fitness := f;
10.            classe_do_indivíduo := cl;
11.         fim se;
12.     fim para;
13. Ponha o indivíduo i na subpopulação correspondente a classe classe_do_indivíduo;
14. fim para;
15. enquanto não atingir número_max_de_gerações faça
16.     Calcule o vetor de funções fitness de cada indivíduo utilizando dados_de_treinamento;
17.     Defina as probabilidades de cruzamento e mutações iguais para todas as subpopulações;
18.     para cada subpopulação faça
19.         Aplique o operação Seleção;
20.         Aplique o operação Cruzamento;
21.         Aplique o operação Mutação;
22.     fim para;
23.     Aplique o Controle da Diversidade Genética;                                /* veja os detalhes na Seção 6.6 */
24.     Armazene os novos 10% melhores indivíduos na População Arquivo;          /* Elitismo. Veja os detalhes na Seção 6.5 */
25.     Se População Arquivo está cheia faça
26.         Remova os indivíduos excedentes que possuem os piores valores de fitness;
27.     fim se;
28. fim enquanto;
29. retorne as subpopulações;

```

Os detalhes do algoritmo são discutidos ao longo das próximas seções.

6.3 O Indivíduo

O indivíduo foi modelado para representar uma regra de classificação (PEREIRA et al. 2010), a qual deve ser utilizada como um filtro para a seleção de padrões no banco de dados. A estrutura do indivíduo é a mesma apresentada na Seção 5.2.1. Duas partes compõem o indivíduo: (a) a árvore que codifica o filtro dos atributos; e (b) o rótulo da classe que indica a qual classe as amostras selecionadas pela regra pertencerão. O algoritmo escolhe a melhor classe para o indivíduo testando todas as possibilidades. A

classe que proporcionar o melhor valor de *fitness* é escolhida (essa escolha é detalhada nos passos 4 a 12 no Algoritmo 6.1).

6.4 Avaliação das Regras

O desempenho de uma regra (indivíduo) é determinado pela avaliação de um vetor de funções *fitness*. A primeira função é baseada em consultas SQL realizadas no banco de dados do problema. A segunda função está relacionada à complexidade da regra. Neste caso, o número de nós da árvore do indivíduo é utilizado como indicador da complexidade da regra.

O cálculo do valor da primeira função *fitness* é dividido em duas etapas: (1) cálculo da matriz de confusão e (2) cálculo da efetividade da regra. Os coeficientes da matriz de confusão identificados na etapa (1) são utilizados na etapa (2). Para maiores detalhes sobre matriz de confusão vide Seções 4.2.2.1 e 5.2.2.1.

6.4.1 Vetor de Funções objetivo

O primeiro valor está relacionado à efetividade da regra de classificação (indivíduo), que neste caso é medida pelo produto entre a *sensibilidade* e a *especificidade* da regra.

A composição deste valor como resultado do produto da sensibilidade pela especificidade é importante para evitar o *overfitting*, assim como reduzir (ou até mesmo eliminar) a tendência de privilegiar as classes predominantes em bancos de dados desbalanceados. Se a regra apresenta um desempenho muito baixo com relação a um dos valores da sensibilidade ou da especificidade, o valor da função *fitness* também será baixo, independente do outro indicador. Desta forma, a função *fitness* irá penalizar, por exemplo, as regras que não são capazes de identificar os padrões que pertençam à classe em questão (gera um valor baixo de verdadeiros positivos), porém excluem todos aqueles que não pertençam a esta classe (gera um valor alto de verdadeiros negativos). O uso da acurácia, calculado como $(TP+TN)/(TP+TN+FP+FN)$, usado por muitos algoritmos de classificação, pode resultar num valor alto se o número de TN é alto, mesmo quando o valor de TP é próximo de zero.

Considere o seguinte exemplo. Tem-se um banco de dados com 100 pacientes, onde 80 são normais e 20 possuem câncer. Assuma que dois classificadores foram aplicados a essa base. O primeiro classificador obteve a matriz de confusão indicada na Tabela 6.1.

Tabela 6.1 - Matriz de confusão obtida pelo primeiro classificador

Real \ Classif.	Normal	com cancer
Normal	60	20
com cancer	5	15

onde TP = 60, FN=20, FP = 5 e TN = 15.

A acurácia é $(60+15)/(60+20+5+15)=0.75$. Nesse caso, 75% (60/80) dos pacientes normais 75% (15/20) dos pacientes com câncer foram classificados corretamente.

Agora, assumamos que o segundo classificador foi aplicado ao problema e obteve a matriz de confusão indicada na Tabela 6.2.

Tabela 6.2 - Matriz de confusão obtida pelo segundo classificador

Real \ Classif.	Normal	com câncer
Normal	80	0
com câncer	20	0

onde TP = 80, FN=0, FP = 20 e TN = 0.

Esse classificador rotulou todos os pacientes como normais. A acurácia nesse caso é $(80+0)/(80+0+0+20)=0.80$. Assim, esse classificador obteve uma acurácia maior que a obtida pelo primeiro classificador. Além disso, todos os pacientes que são efetivamente normais foram identificados corretamente, porém nenhum paciente com câncer foi identificado.

Na Tabela 6.3 são apresentados os valores de sensibilidade e especificidade, além do produto entre eles, medido para cada um dos classificadores.

Tabela 6.3 - Avaliação dos classificadores baseada nos valores da sensibilidade e especificidade

Classificador	Sensibilidade	Especificidade	Sensibilidade * Especificidade
#1	$\frac{60}{60+20} = 0.75$	$\frac{15}{15+5} = 0.75$	$0.75 * 0.75 = 0.56$
#2	$\frac{80}{80+0} = 1.00$	$\frac{0}{0+20} = 0.00$	$1.00 * 0.00 = 0.00$

Outra característica importante do algoritmo proposto é o fato da população ser dividida em subpopulações. As subpopulações representam nichos, sendo que cada um destes nichos é manipulado de tal maneira a proporcionar ao algoritmo a obtenção de

boas regras de classificação para cada classe independentemente.

Considere-se, por exemplo, um banco de dados que armazena padrões pertencentes a duas classes, sendo que 90% deles pertencem à classe A e 10% pertencem à classe B. Algoritmos regulares tendem a considerar padrões da classe B como ruídos, gerando regras que privilegiam a classe A. Portanto, uma regra que classifica todos os padrões como pertencentes à classe A apresentará um alto valor de acurácia, se o cálculo for realizado da forma tradicional, conforme descrita anteriormente. Neste cenário, o algoritmo proposto terá dois nichos, um para desenvolver indivíduos que classificam padrões da classe A e outro para gerar regras que classificam padrões da classe B. As classes terão chances similares de gerar boas regras de classificação. Maiores detalhes sobre a técnica de nicho utilizada e as respectivas subpopulações são fornecidos na seção 6.6.

A segunda função *fitness* é calculada como o inverso do número de nós de um indivíduo, conforme mostrado em (6.1):

$$F_2(I) = 1/\text{número_de_nós} \quad (6.1)$$

Observou-se que algoritmos de classificação que não consideram a complexidade das regras geradas têm, normalmente, maior chance de incorrer em *overfitting* (TSAI, 2006) (VIEIRA et al., 2007). Desta forma, a segunda função *fitness* pode ser interpretada como um mecanismo para se evitar *overfitting*. Além do mais, regras pequenas possuem a vantagem de serem mais simples de se entender do que regras grandes.

A comparação das soluções é realizada baseada no princípio de dominância, uma vez que a comparação escalar tradicional não pode ser aplicada para comparar vetores. Dados dois vetores X e Y, considere as seguintes relações:

$$\begin{aligned} X < Y &\Leftrightarrow \{X_i \leq Y_i \mid \forall i \in \{1, \dots, n\}\} \text{ e } \{\exists i \in \{1, \dots, n\} \mid X_i < Y_i\} \\ X = Y &\Leftrightarrow X_i = Y_i, \forall i \in \{1, \dots, n\} \end{aligned}$$

e simetricamente:

$$X \neq Y \Leftrightarrow \exists i \in \{1, \dots, n\} \mid X_i \neq Y_i$$

Baseado nessas relações, o princípio de dominância define que uma solução A é melhor que uma solução B se, e somente se, a relação $A < B$ se aplica. Nesse caso, diz-se que A domina B ou, analogamente, B é dominado por A. Além disso, se a relação X

$\neq Y$ se aplica, porém nem $A \prec B$ nem $B \prec A$ são válidas, então não é possível haver uma distinção entre A e B, de tal maneira que o resultado da comparação entre as duas soluções é um empate. No caso específico deste trabalho, os empates são resolvidos baseados no valor observado na primeira função *fitness*: se as soluções A e B são diferentes, mas elas não dominam uma a outra, então a solução com o melhor valor de F_1 é escolhida. Esta abordagem é justificada pela maior importância da primeira função *fitness* nos problemas analisados.

Durante a etapa de configuração e em experimentos preliminares, notou-se que a função F_2 gera uma grande pressão no algoritmo pela seleção de indivíduos menores. Neste caso, o algoritmo poderia não ser capaz de explorar amplamente o espaço de busca. A fim de conter este efeito, foi implementado um critério que permite ao algoritmo aceitar indivíduos com um número maior de nós nas iterações iniciais.

Quando o algoritmo está procurando por um conjunto de pontos não-dominados, um indivíduo com um valor maior no segundo objetivo pode ser aceito utilizando a seguinte probabilidade:

$$P(I_1, I_2) = \begin{cases} 1 & \text{se } F_2(I_2) \leq F_2(I_1) \\ \exp\left(\frac{F_2(I_1) - F_2(I_2)}{T}\right) & \text{se } F_2(I_2) > F_2(I_1) \text{ com } T > 0 \end{cases} \quad (6.2)$$

onde $P(I_1, I_2)$ é a probabilidade de aceitação do indivíduo I_2 cujo valor na posição 2 no vetor *fitness* é pior que o valor apresentado pelo indivíduo I_1 e T é um parâmetro do algoritmo e $0 < F_2(I_i) \leq 1$. O parâmetro T é inicialmente configurado e, ao longo das gerações, esse valor é decrementado continuamente até que chegue próximo a zero. Nos experimentos realizados, o melhor valor de T foi 10 e a taxa de decréscimo foi de 15% por geração. Este procedimento é similar ao empregado no algoritmo *Simulated Annealing* (KIRKPATRICK et al., 1983) (VASCONCELOS et al., 1996)(PEREIRA; VASCONCELOS, 2012) para aceitação de novas soluções.

Esta estratégia possibilita a aceitação de indivíduos maiores, com bons valores de efetividade durante o início da evolução, apesar do valor eventualmente desfavorável do segundo objetivo. Isto possibilita uma maior diversidade genética durante as operações de cruzamento e mutação.

Desta forma, o operador de seleção é implementado da seguinte forma (baseado no modelo torneio):

1. Selecione dois indivíduos aleatoriamente;

2. Compare os indivíduos utilizando o conceito de dominância, considerando a probabilidade de aceite de pior valor de F_2 (equação 6.2) e resolvendo os empates de acordo com o valor de F_1 (conforme descrito anteriormente);
3. Retorne o indivíduo vencedor no passo anterior.

6.5 População Arquivo - Elitismo Global

A fim de garantir que os melhores indivíduos de todos os nichos serão preservados durante toda a execução do algoritmo, uma estratégia de elitismo global foi implementada: ao final de cada geração, os 10% melhores indivíduos de cada nicho são selecionados e armazenados, numa população arquivo, também chamada de população *offline*. Estes indivíduos são reintroduzidos na população principal toda vez que a média do primeiro valor da *fitness* (F_1) de uma subpopulação específica decresce. É importante destacar que o algoritmo não seleciona simplesmente os melhores indivíduos da população como um todo, mas sim os melhores de cada subpopulação. Esta estratégia permite uma exploração adequada para todas as classes. Quando a população arquivo atinge seu limite de armazenamento, os indivíduos com piores valores de *fitness* são descartados.

Nos experimentos realizados, observou-se que a população arquivo aumentou o desempenho da classificação para todas as classes do problema (PEREIRA; VASCONCELOS, 2010). Esta característica é particularmente útil em bancos de dados desbalanceados.

6.6 Nichos e Controle de Diversidade Genética

A técnica de nicho implementada no algoritmo proposto consiste na inserção de um rótulo nos indivíduos a fim de agrupar aqueles que pertencem a um mesmo nicho (GOLDBERG, 1989).

A fim de garantir que todos os nichos irão evoluir igualmente, foi proposto um mecanismo de controle da diversidade genética (VASCONCELOS et al., 2001) por nicho. A diversidade genética é calculada em cada subpopulação online como a razão entre a média do primeiro valor da função *fitness* dos indivíduos pertencentes à subpopulação, pelo maior valor da primeira função *fitness* considerando toda a população. O pseudocódigo do algoritmo que implementa o controle da diversidade

genética é apresentado no Algoritmo 6.2.

Algoritmo 6.2 - Pseudocódigo do Controle de Diversidade Genética

```
1.  $gdg := 0$ ;
2. para cada subpopulação online  $i$  faça
3.   Calcule a diversidade de  $i$ . Chame este valor da diversidade de  $dg_i$ ;
4.   se  $dg_i > gdg$  faça
5.      $gdg := dg_i$ ;
6.   fim se;
7. fim para;
8. para cada subpopulação  $i$  faça
9.   se  $dg_i < 0.4 * gdg$  faça
10.    Aumente o número de indivíduos da subpopulação  $i$  em 10%; Selecione estes indivíduos da subpopulação offline,
        proveniente do mesmo nicho;
11.    Aumente as probabilidades de mutação e cruzamento da subpopulação  $i$  em 10%;
12.    Aplique o operador Seleção em  $i$ ;
13.    Aplique o operador Cruzamento em  $i$ ;
14.    Aplique o operador Mutação em  $i$ ;
15.   fim se;
16. fim para;
```

Neste algoritmo, o valor da diversidade, chamado dg_i , de cada subpopulação i , é calculado. O maior valor apurado em cada geração é chamado gdg ; cada valor dg_i é comparado com o gdg . Se $dg_i < 0.4 * gdg$ então indivíduos da população *offline*, pertencentes ao mesmo nicho, são selecionados e inseridos na subpopulação *online*. Além disto, as probabilidades de mutação e cruzamento da subpopulação i são incrementadas em 10% na geração corrente. Esta avaliação é feita em cada geração.

Os valores constantes utilizados nesse algoritmo (40% e 10% respectivamente) foram definidos empiricamente. O controle de diversidade genética é executado a cada geração. As probabilidades de mutação e cruzamento são restabelecidas para os valores padrão ao final de cada geração, a fim de evitar um crescimento descontrolado destes parâmetros.

O controle da diversidade genética é executado após a aplicação dos operadores de mutação e cruzamento (linha 23 do Algoritmo 6.1).

6.7 Mecanismo de Classificação

O algoritmo proposto utiliza o número de gerações como critério de parada. Uma vez encerrado o processo de produção de novas gerações, existem amplas possibilidades de estratégias de uso da população obtida, a fim de classificar o conjunto de testes. (lembrando que a população é obtida utilizando o conjunto de treinamento). Três abordagens foram adotadas a fim de avaliar o desempenho das regras obtidas, a saber:

1. **Contagem de regras que casam:** a amostra é rotulada com a classe indicada pelo maior número de regras geradas. Por exemplo, se um registro é selecionado por três regras da classe A, duas regras da classe B e uma regra da classe C, então a amostra é classificada como A.
2. **Contagem ponderada de regras que casam:** esta abordagem é similar à anterior, porém, neste caso, o valor da *fitness* na primeira posição é somado a fim de ponderar o valor final e orientar a rotulação da amostra. Por exemplo, se um registro é selecionado por três regras da classe A (com valores de *fitness* 0.4, 0.2 e 0.2) e 1 regra da classe B (com *fitness* 0.9), então a amostra é classificada como B.
3. **Casamento com melhor regra:** A melhor regra, baseada no primeiro valor da *fitness*, é selecionada em cada nicho. Esta regra será responsável por selecionar todas as amostras da classe correspondente ao nicho. Nessa abordagem, um banco de dados com três classes terá três classificadores, um para cada classe.

Os resultados obtidos com o algoritmo proposto foram comparados aos resultados obtidos por três outros (J48, RBF e SVM). Os resultados e a análise são apresentados na próxima seção.

6.8 Resultados Experimentais

Os experimentos foram realizados reutilizando alguns dos bancos de dados adotados nos experimentos apresentados nos capítulos 5 e 6: Hepatite, vinho, DGA (bancos 1 e 3), desenvolvimento urbano e infraestrutura.

Durante os experimentos descritos nesta seção, o algoritmo proposto foi configurado com os seguintes parâmetros:

- Taxa de mutação: 2%.
- Taxa de cruzamento: 80%.
- Elitismo por nicho: 10% dos melhores indivíduos.

O tamanho da população e o número de gerações utilizados em cada um dos bancos são mostrados na Tabela 6.4.

Tabela 6.4 - Tamanho da população e número de gerações utilizados em cada banco

	Hepatite	Vinho	Banco 1	Banco 3	Desen. urbano	Infraestrutura
Tam. da população	500	500	500	500	300	300
Núm. de gerações	200	200	200	200	100	100

Os valores desses parâmetros foram definidos a partir de uma etapa de configuração. Alguns parâmetros foram experimentados ao longo da implementação do algoritmo a fim de identificar a melhor configuração que proporcione resultados significantes sem demandar um esforço computacional alto. Por exemplo, aumentando o número de indivíduos e gerações é possível obter melhores valores de *fitness*, porém o tempo de execução também aumenta.

A validação cruzada 10-*fold* (10 partições) (COHEN, 1995) foi utilizada em todos os experimentos a fim de treinar e testar as regras geradas. Não foi permitida a existência de padrões repetidos nas partições (*folds*), isto é, os conjuntos de dados são avaliados a fim de eliminar padrões duplicados. Os padrões de cada partição são selecionados a priori randomicamente.

O experimento completo (utilizando as 10 partições) foi executado 5 vezes, a fim de gerar 50 resultados para cada algoritmo. As populações obtidas foram então utilizadas de acordo com as três estratégias descritas na seção 0. Os resultados obtidos serão identificados como algoritmos 1, 2 e 3, e serão comparados com os resultados obtidos pela Árvore de Decisão (versão J48), *Radial Basis Function* (RBF) e *Support Vector Machine* (SVM). Foram utilizadas as implementações da WEKA 3.6.6 destes três algoritmos. Ainda como medida para proporcionar uma comparação justa, os algoritmos J48, RBF e SVM também foram executados 5 vezes em cada um dos bancos de dados, utilizando diferentes valores nos respectivos parâmetros. A melhor configuração encontrada para cada banco foi utilizada nos experimentos. A configuração que proporcionou os melhores resultados para cada um dos problemas é mostrada na Tabela 6.5.

Tabela 6.5 – Configuração dos parâmetros dos algoritmos clássicos para cada um dos bancos

	Parâmetros	Hepatite	Vinho	Banco 1	Banco 3	D. urbano	Infraestrutura
J48	Poda	sim	sim	sim	sim	sim	sim
	Fator de confiança	0.025	0.25	0.025	0.025	0.025	0.025
RBF	Clusters	3	3	2	4	3	3
	Desvio padrão mínimo por cluster	0.1	0.1	0.01	0.01	0.01	0.01
SVM	Complexidade C	1.0	2.0	3.0	4.0	4	4
	Função kernel	Polykernel	Norm. Polykernel	Norm. Polykernel	Polykernel	Polykernel	Polykernel
	Expoente da função	2.0	2.0	3.0	1.0	3.0	3.0

O desempenho obtido em cada uma das 50 execuções é apresentado a seguir. A comparação entre os algoritmos foi conduzida levando-se em consideração dois critérios distintos: (1) o produto da sensibilidade pela especificidade; e (2) acurácia. A utilização do primeiro critério é importante, pois, caso exista algum desequilíbrio (predomínio de verdadeiro positivo ou de verdadeiro negativo) na classificação das amostras, a regra será penalizada. A utilização do segundo critério se justifica devido ao fato da grande maioria dos trabalhos científicos estudados utilizar esse critério. Dessa forma, os resultados obtidos aqui podem ser comparados com demais estudos publicados.

Na Tabela 6.6 estão dispostos os p -valores obtidos com a comparação dos resultados obtidos no banco hepatite, utilizando como parâmetro a média dos produtos da sensibilidade pela especificidade obtidos em cada uma das classes do problema. A Tabela 6.7 apresenta os p -valores obtidos na comparação que utilizou a acurácia global dos algoritmos. Na Tabela 6.8 são apresentadas as classificações dos algoritmos de acordo com cada um dos dois critérios de comparação.

Tabela 6.6 - p -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - hepatite

	1	2	3	J48	RBF	SVM
1	0,5000	0,5793	1,0000	0,0010	0,9431	0,0081
2	0,4207	0,5000	1,0000	0,0000	0,9010	0,0000
3	0,0000	0,0000	0,5000	0,0000	0,0013	0,0000
J48	0,9990	1,0000	1,0000	0,5000	1,0000	0,9986
RBF	0,0569	0,0990	0,9987	0,0000	0,5000	0,0000
SVM	0,9919	1,0000	1,0000	0,0014	1,0000	0,5000

Tabela 6.7 - p -valor obtido na comparação utilizando acurácia global - hepatite

	1	2	3	J48	RBF	SVM
1	0,5000	0,5231	1,0000	0,5110	0,9994	1,0000
2	0,4769	0,5000	1,0000	0,4719	1,0000	1,0000
3	0,0000	0,0000	0,5000	0,0000	0,4945	1,0000
J48	0,4890	0,5281	1,0000	0,5000	1,0000	1,0000
RBF	0,0006	0,0000	0,5055	0,0000	0,5000	1,0000
SVM	0,0000	0,0000	0,0000	0,0000	0,0000	0,5000

Tabela 6.8 - Classificação dos algoritmos de acordo com os dois critérios - hepatite

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	2	2	1	4	2	3
acurácia	3	3	2	3	2	1

Conforme observado na Tabela 6.8, de acordo com o primeiro critério, o algoritmo 3 obteve o melhor desempenho, seguido por um empate entre 1, 2 e RBF. Esse empate

se deve ao fato do p -valor obtido na comparação entre os algoritmos não ter superado o valor de confiança estabelecido.

A comparação de acordo com o segundo critério apresenta um cenário diferente do primeiro: o melhor algoritmo é o SVM, seguido pelo algoritmo 3 e RBF empatados no segundo lugar. No terceiro lugar acontece um empate entre o algoritmo 1, 2 e J48.

Os resultados das comparações dos resultados obtidos no banco vinho são mostrados na Tabela 6.9, Tabela 6.10 e Tabela 6.11.

Tabela 6.9 - p -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - vinho

	1	2	3	J48	RBF	SVM
1	0,5000	0,3757	0,0786	0,0000	0,0001	0,0072
2	0,6243	0,5000	0,1354	0,0000	0,0000	0,0206
3	0,9214	0,8646	0,5000	0,0000	0,0507	0,2967
J48	1,0000	1,0000	1,0000	0,5000	1,0000	1,0000
RBF	0,9999	1,0000	0,9493	0,0000	0,5000	0,9988
SVM	0,9928	0,9794	0,7033	0,0000	0,0012	0,5000

Tabela 6.10 - p -valor obtido na comparação utilizando acurácia global - vinho

	1	2	3	J48	RBF	SVM
1	0,5000	0,3757	0,0786	0,0000	0,3832	0,4715
2	0,6243	0,5000	0,1354	0,0000	0,5591	0,6369
3	0,9214	0,8646	0,5000	0,0000	0,9558	0,9684
J48	1,0000	1,0000	1,0000	0,5000	1,0000	1,0000
RBF	0,6168	0,4409	0,0442	0,0000	0,5000	0,7650
SVM	0,5285	0,3631	0,0316	0,0000	0,2350	0,5000

Tabela 6.11 - Classificação dos algoritmos de acordo com os dois critérios - vinho

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	1	1	1:2:3	4	3	2
acurácia	1:2	1:2	1:2	3	1	1

O algoritmo 3 ocupa uma posição intermediária, no primeiro critério de comparação. O algoritmo não é pior que 1 e 2, nem é melhor que o SVM. Contudo, 1 e 2 são melhores que SVM.

Houve um empate entre quase todos os algoritmos de acordo com o segundo critério. A exceção é J48 que ficou isolado no segundo lugar.

Os resultados que serão apresentados a seguir (Tabela 6.12 a Tabela 6.17) referem-se a dois dos bancos do problema DGA, utilizados no capítulo 4. Somente os bancos 1 e 3 foram utilizados, uma vez que o banco 2 possui poucos registros. Como os experimentos nesse caso utilizam validação cruzada com 10 partições, cada partição foi composta por poucas amostras, gerando resultados com um desvio padrão muito

elevado. Dessa forma, o banco 2 foi descartado do experimento.

Tabela 6.12 - p -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - banco 1

	1	2	3	J48	RBF	SVM
1	0,5000	0,5934	0,6156	0,0000	0,0000	0,0000
2	0,4066	0,5000	0,5089	0,0000	0,0000	0,0000
3	0,3844	0,4911	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,0034	0,0000
RBF	1,0000	1,0000	1,0000	0,9966	0,5000	0,0000
SVM	1,0000	1,0000	1,0000	1,0000	1,0000	0,5000

Tabela 6.13 - p -valor obtido na comparação utilizando acurácia global - banco 1

	1	2	3	J48	RBF	SVM
1	0,5000	0,5032	1,0000	1,0000	1,0000	1,0000
2	0,4968	0,5000	1,0000	1,0000	1,0000	1,0000
3	0,0000	0,0000	0,5000	0,0000	0,0000	0,0000
J48	0,0000	0,0000	1,0000	0,5000	0,9859	1,0000
RBF	0,0000	0,0000	1,0000	0,0141	0,5000	0,6864
SVM	0,0000	0,0000	1,0000	0,0000	0,3136	0,5000

Tabela 6.14 - Classificação dos algoritmos de acordo com os dois critérios - banco 1

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	1	1	1	2	3	4
acurácia	4	4	1	3	2	2

Os resultados obtidos de acordo com ambos os critérios apontam o algoritmo 3 como o melhor no banco 1. Os demais algoritmos apresentaram alteração de posição. Os algoritmos 1 e 2, por exemplo, saíram da primeira posição de acordo com o primeiro critério e foram para a quarta posição de acordo com o segundo critério.

Tabela 6.15 - p -valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - banco 3

	1	2	3	J48	RBF	SVM
1	0,5000	0,8021	0,9308	0,0000	0,0022	0,4445
2	0,1979	0,5000	0,7276	0,0000	0,0000	0,0892
3	0,0692	0,2724	0,5000	0,0000	0,0000	0,0066
J48	1,0000	1,0000	1,0000	0,5000	0,9770	1,0000
RBF	0,9978	1,0000	1,0000	0,0230	0,5000	1,0000
SVM	0,5555	0,9108	0,9934	0,0000	0,0000	0,5000

Tabela 6.16 - *p*-valor obtido na comparação utilizando acurácia global - banco 3

	1	2	3	J48	RBF	SVM
1	0,5000	0,7223	1,0000	0,0000	0,0098	0,7898
2	0,2777	0,5000	1,0000	0,0000	0,0000	0,5426
3	0,0000	0,0000	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,9630	1,0000
RBF	0,9902	1,0000	1,0000	0,0370	0,5000	1,0000
SVM	0,2102	0,4574	1,0000	0,0000	0,0000	0,5000

Tabela 6.17 - Classificação dos algoritmos de acordo com os dois critérios - banco 3

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	1:2	1:2	1	4	3	2
acurácia	2:3	2	1	4	3	2

A classificação obtida com os resultados no banco 3 (Tabela 6.17) apontam o desempenho do algoritmo 3 como melhor que os demais, em ambos os critérios. Os algoritmos 2 e 3 vem a seguir, sendo seguidos por SVM, RBF e J48.

Os bancos cujos resultados foram apresentados até esse ponto não possuem dados híbridos. Os resultados que serão apresentados a seguir se referem aos bancos que misturam atributos convencionais com atributos geográficos e é nesse cenário que os algoritmos propostos apresentam a contribuição mais relevante. Os resultados referentes ao banco desenvolvimento urbano estão dispostos da Tabela 6.18 a Tabela 6.20. Os resultados do banco infraestrutura estão descritos da Tabela 6.23 a Tabela 6.25.

Tabela 6.18 - *p*-valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - desenvolvimento urbano

	1	2	3	J48	RBF	SVM
1	0,5000	0,4009	0,0000	0,0000	0,0000	0,0000
2	0,5991	0,5000	0,0000	0,0000	0,0000	0,0000
3	1,0000	1,0000	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,0000	0,0000
RBF	1,0000	1,0000	1,0000	1,0000	0,5000	0,0000
SVM	1,0000	1,0000	1,0000	1,0000	1,0000	0,5000

Tabela 6.19 - *p*-valor obtido na comparação utilizando acurácia global - desenvolvimento urbano

	1	2	3	J48	RBF	SVM
1	0,5000	0,6866	1,0000	0,0000	0,0000	0,0000
2	0,3134	0,5000	1,0000	0,0000	0,0000	0,0000
3	0,0000	0,0000	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,0000	0,7625
RBF	1,0000	1,0000	1,0000	1,0000	0,5000	1,0000
SVM	1,0000	1,0000	1,0000	0,2375	0,0000	0,5000

Tabela 6.20 - Classificação dos algoritmos de acordo com os dois critérios - desenvolvimento urbano

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	1	1	2	3	4	5
Acurácia	2	2	1	3	4	3

Os algoritmos propostos ficaram nas primeiras posições tanto no banco desenvolvimento urbano (Tabela 6.20), quando no banco infraestrutura (Tabela 6.25). Os algoritmos conseguiram tirar proveito dos dados geográficos de tal forma a obterem um desempenho superior aos apresentados pelos métodos clássicos. A seguir são apresentadas algumas regras geradas nos problemas com dados geográficos. A Tabela 6.21 apresenta as regras geradas no banco desenvolvimento urbano. As regras geradas para o banco infraestrutura são apresentadas na Tabela 6.22.

Tabela 6.21 – Exemplos de regras geradas pelo MDMGeo no problema Desenvolvimento urbano

Classe	Regra	Acurácia	Sensibilidade x Especificidade
A	número consumidores industriais de energia ≥ 16758	0,8286	0,6688
B	rodovia federal não cruza município E ferrovia cruza município	0,7371	0,6580
C	ferrovia não cruza município E ($1521 \leq$ número consumidores industriais de energia ≤ 36479)	0,7605	0,5372

Tabela 6.22 – Exemplos de regras geradas pelo MDMGeo no problema Infraestrutura

Classe	Regra	Acurácia	Sensibilidade x Especificidade
A	(consumo residencial de energia $>$ quantidade de consumidores rurais) E (tamanho da população $<$ consumo industrial de energia) E (ferrovia não cruza município) E (rodovia federal cruza município) E (consumo industrial ≥ 16481)	0,9678	0,9523
B	(quantidade de consumidores rurais ≤ 172261) E (rodovia federal não cruza município)	0,7751	0,6644
C	(consumo industrial de energia \geq tamanho da população) E (consumo industrial de energia ≤ 75064) E (rodovia federal cruza município) E (consumo industrial de energia \leq consumo diverso ²⁸ de energia)	0,8982	0,6973

²⁸ Consumo diverso de energia consiste no consumo realizado por clientes que não são classificados como indústria, comércio, residência ou rural.

Tabela 6.23 - *p*-valor obtido na comparação utilizando a média do produto da sensibilidade x especificidade - infraestrutura

	1	2	3	J48	RBF	SVM
1	0,5000	0,5018	0,8778	0,0000	0,0000	0,0000
2	0,4982	0,5000	0,8787	0,0000	0,0000	0,0000
3	0,1222	0,1213	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,0000	0,0000
RBF	1,0000	1,0000	1,0000	1,0000	0,5000	0,0000
SVM	1,0000	1,0000	1,0000	1,0000	1,0000	0,5000

Tabela 6.24 - *p*-valor obtido na comparação utilizando acurácia global - infraestrutura

	1	2	3	J48	RBF	SVM
1	0,5000	0,1297	1,0000	0,0000	0,0000	0,0000
2	0,8703	0,5000	1,0000	0,0000	0,0000	0,0000
3	0,0000	0,0000	0,5000	0,0000	0,0000	0,0000
J48	1,0000	1,0000	1,0000	0,5000	0,2055	0,0305
RBF	1,0000	1,0000	1,0000	0,7945	0,5000	0,0969
SVM	1,0000	1,0000	1,0000	0,9695	0,9031	0,5000

Tabela 6.25 - Classificação dos algoritmos de acordo com os dois critérios - infraestrutura

Critério	1	2	3	J48	RBF	SVM
sensibilidade x especificidade	1	1	1	2	3	4
Acurácia	2	2	1	3	3:4	4

Os resultados observados para os seis algoritmos e os seis bancos de dados estão resumidos em duas tabelas a seguir. A Tabela 6.26 resume as informações referentes ao produto sensibilidade x especificidade. A Tabela 6.27 contém as informações referentes à acurácia. Nas tabelas são exibidas as médias e os desvios padrão medidos em cada um dos bancos. Os melhores valores estão destacados com a fonte em negrito, a fim de facilitar a leitura.

Tabela 6.26 - Resumo da performance dos algoritmos de acordo com a média do produto da sensibilidade x especificidade

		Banco de Dados					
Algoritmo		Hepatite	Vinho	Banco 1	Banco 3	Des. urbano	Infraestrutura
1	Média	0,5053	0,9634	0,5725	0,5409	0,6484	0,6498
	Desv. Padrão	0,2746	0,0422	0,1139	0,1325	0,0916	0,0772
2	Média	0,5165	0,9609	0,5911	0,5485	0,6399	0,6500
	Desv. Padrão	0,2788	0,0418	0,1015	0,1406	0,0762	0,0772
3	Média	0,6459	0,9511	0,6036	0,5485	0,5807	0,6704
	Desv. Padrão	0,1708	0,0417	0,0830	0,1406	0,0577	0,0943
J48	Média	0,3721	0,8698	0,5138	0,2172	0,0267	0,0278
	Desv. Padrão	0,0797	0,0225	0,0329	0,0755	0,0329	0,0603
RBF	Média	0,5676	0,9409	0,5264	0,1755	0,3424	0,1628
	Desv. Padrão	0,0532	0,0125	0,0297	0,0747	0,0111	0,0451
SVM	Média	0,4114	0,9478	0,5719	0,0000	0,3424	0,0350
	Desv. Padrão	0,0484	0,0099	0,0316	0,0000	0,0111	0,0080

Tabela 6.27 - Resumo da performance dos algoritmos de acordo com a média da acurácia global

		<i>Banco de Dados</i>					
<i>Algoritmo</i>		Hepatite	Vinho	Banco 1	Banco 3	Des. urbano	Infraestrutura
1	Média	0,7759	0,9552	0,6676	0,6439	0,6979	0,8317
	Desv. Padrão	0,1110	0,0502	0,1029	0,0916	0,0773	0,0344
2	Média	0,7775	0,9530	0,6786	0,6441	0,7528	0,8237
	Desv. Padrão	0,1124	0,0494	0,0860	0,0928	0,0642	0,0738
3	Média	0,8288	0,9736	0,7854	0,8250	0,7706	0,8728
	Desv. Padrão	0,0713	0,0208	0,0582	0,0479	0,0278	0,0267
J48	Média	0,7764	0,9098	0,6225	0,7517	0,5831	0,7215
	Desv. Padrão	0,0242	0,0160	0,0277	0,0428	0,0343	0,0354
RBF	Média	0,8287	0,9613	0,6321	0,7718	0,5546	0,7163
	Desv. Padrão	0,0180	0,0085	0,0249	0,0489	0,0295	0,0258
SVM	Média	0,8588	0,9625	0,6799	0,7761	0,5871	0,7100
	Desv. Padrão	0,0125	0,0072	0,0232	0,0368	0,0198	0,0239

Os algoritmos propostos obtiveram melhor desempenho em todos os bancos, considerando o produto sensibilidade x especificidade com critério de comparação. Esse critério é relevante uma vez que demonstra o real desempenho do algoritmo, mesmo em bancos desbalanceados, como é o caso de hepatite, banco 3 e infraestrutura. Foram nesses bancos que os algoritmos clássicos obtiveram pior desempenho, com relação ao referido critério.

Outro aspecto relevante é o fato de que as regras geradas pelo algoritmo proposto neste trabalho possuem cerca de 30% do tamanho das regras obtidas pela versão mono-objetivo deste algoritmo, apresentada na seção 5.2 e em Pereira et al. (2010). Um benefício importante é que, com regras mais simples, diminui-se também o *overfitting* na classificação dos dados (VIEIRA, 2006)(WITTEN; FRANK, 2005)(QUINLAN, 1993).

6.9 Conclusão

Nesta seção, um novo algoritmo que extrai regras de classificação e utiliza estas regras em três abordagens distintas é proposto. O algoritmo utiliza técnica de nicho, probabilidades de mutação e cruzamento distintas por nicho, além de uma população arquivo para prover indivíduos que auxiliam na obtenção de boas regras de classificação, que identifiquem corretamente indivíduos de todas as classes existentes no problema. Dois objetivos, efetividade e complexidade, foram considerados a fim de obter boas regras e evitar *overfitting* de classificação. Outra vantagem desta abordagem é a obtenção de regras menores, mais fáceis de serem interpretadas.

Uma vez que este é um algoritmo adaptativo, é possível utilizar um conjunto de funções com operadores simples, tais como $>$, $<$, $=$, AND, além de utilizar funções de um domínio específico, tais como operadores geográficos. Além do mais, a alteração de elementos do conjunto de funções não demanda do usuário a implementação de alterações no uso do algoritmo.

O algoritmo foi utilizado para classificar dados em seis problemas distintos: hepatite, vinho, DGA (2 bancos), desenvolvimento urbano e infraestrutura. Os experimentos foram realizados utilizando também três outros algoritmos relevantes: Árvore de Decisão (J48), rede *Radial Basis Function* e *Support Vector Machines*. Estes métodos são amplamente aplicados em problemas de classificação de dados e foram adotados para validar os esquemas propostos.

Os resultados obtidos pelo método proposto superaram os demais resultados em quase todas as instâncias testadas, de acordo os critérios adotados. Um último ponto relevante a ser destacado é que o algoritmo proposto obteve um bom desempenho mesmo nos bancos de dados desbalanceados. Este fato sugere que o método proposto é consideravelmente robusto em relação a este aspecto.

7 CONCLUSÕES E TRABALHOS FUTUROS

Esta tese teve como foco o estudo e a proposta de algoritmos que sejam capazes de lidar com bancos de dados compostos por dados híbridos, em particular dados geográficos. O levantamento bibliográfico mostrou que o problema é relevante, uma vez que diversos trabalhos dedicam-se ao assunto, porém o foco tem se voltado para a utilização de imagens e para o pré-processamento dos dados geográficos, de modo a transformá-los em dados convencionais e, então, fazer uso de algoritmos convencionais. Esse tipo de alternativa provoca perdas no poder de expressão dos dados e, portanto, empobrece as análises. Ferramentas relevantes apresentadas na literatura foram descritas e detalhadas para servir de base de comparação para os algoritmos propostos neste trabalho.

O primeiro algoritmo proposto neste trabalho foi o NGAE (PEREIRA; VASCONCELOS, 2010). Trata-se de um método baseado no Algoritmo Genético, no qual foram acrescentadas técnicas a fim de melhorar a qualidade dos resultados apresentados, principalmente em se tratando de um problema de classificação com mais de duas classes. É preciso destacar que o foco deste algoritmo ainda não está na manipulação de dados híbridos. Contudo, esse foi um importante estudo para esta tese, pois o método gerado serviu de base para o desenvolvimento dos demais algoritmos elaborados nesse trabalho.

Posteriormente, foi proposto o algoritmo DMGeo (PEREIRA et al., 2010), que se baseia na Programação Genética e é capaz de lidar com dados convencionais e não convencionais (e.g., geográficos). Este algoritmo foi aplicado com sucesso tanto em bancos convencionais quanto em bancos compostos por dados híbridos. Os resultados mostraram que a ferramenta foi capaz de aproveitar a heterogeneidade do banco, apresentando um melhor desempenho quando os dados geográficos e convencionais associados influenciam a classificação.

Finalmente, este trabalho se encerra com a apresentação do MDMGeo, que é uma versão multiobjetivo e com outras melhorias no DMGeo. O algoritmo foi comparado a outros métodos relevantes da literatura, apresentando um resultado superior nos casos estudados. Deve-se levar em consideração que os objetivos do trabalho foram alcançados, uma vez que o método é capaz de lidar com dados convencionais (e.g. numéricos, textuais, lógicos) e não convencionais (e.g. geográficos) de maneira

homogênea, gerando regras de tamanho variado, que são capazes de correlacionar todos os tipos de dados presentes no banco, não necessitando de pré-processamentos, e com desempenho comparável ou superior ao dos principais algoritmos existentes na literatura.

Outro aspecto relevante dos algoritmos propostos é sua capacidade de lidar com bancos de dados desbalanceados. A maioria dos problemas reais possui desbalanceamento, o que faz das ferramentas propostas não só relevantes do ponto de vista científico, quanto do ponto de vista prático.

Trabalhos Futuros

Sugere-se que este trabalho seja continuado explorando a arquitetura da ferramenta implementada. Atualmente, a ferramenta pode ser vista como um *framework*, brevemente descrito no anexo dessa tese, no qual pode-se variar:

- a definição das funções *fitness*;
- os conjuntos de nós terminais e funções;
- o método utilizado e a respectiva representação do indivíduo (solução viável).

Quaisquer um destes pontos pode ser explorado para buscar melhores resultados. O método pode ser trocado, por exemplo, de Programação Genética para Colônia de Formigas; a modelagem das funções pode passar a contemplar outros objetivos relevantes para o problema; novos nós podem ser definidos a fim de explorar ainda mais a natureza dos dados de cada problema.

Por fim, os algoritmos apresentados nessa tese utilizam a quantidade de gerações como critério de parada. Sugere-se estudar formas mais eficazes de parada, as quais considerem a evolução da população de uma geração para próxima. Assim, o algoritmo poderá se encerrar assim que a população atingir seu nível máximo de evolução, evitando processamento desnecessário (nos casos onde o número máximo de gerações estabelecido for muito grande) ou ainda uma convergência prematura (quando o número máximo de gerações foi muito pequeno).

Foram utilizados poucos bancos de dados reais e poucos dados geográficos. Assim, sugere-se que se aplique os algoritmos em mais bases de dados, com mais dados geográficos, explorando não só as funções topológicas, mas também todas as funções disponíveis, inclusive as espaço-temporais.

Os algoritmos propostos podem ser estendidos para que sejam aplicados a outras categorias de dados, que não se restrinjam a dados geográficos.

ANEXO

Framework de Classificação de padrões

Introdução

Os algoritmos apresentados nos capítulos anteriores foram desenvolvidos de maneira a compor um *framework* de classificação de padrões. Segundo Markiewicz e Lucena (2000), um *framework* é um componente de *software* que se destina a resolver uma categoria de problemas semelhantes. Esse componente possui partes extensíveis ou flexíveis chamadas *hot spots*, os quais podem ser alterados, dependendo do problema que o *framework* deve solucionar. As partes imutáveis, que constituem o núcleo do *framework*, são chamadas *frozen spots*.

Um *framework* se distingue de uma biblioteca de funções, pois é ele quem dita o fluxo de controle da aplicação, i.e., apresenta uma metodologia ou um processo para solução de um problema ou execução de uma tarefa. Bibliotecas, por outro lado, fornecem somente um conjunto de funções que devem ser combinadas pelo desenvolvedor. Em suma, enquanto um *framework* detém o controle de um processo de solução de um problema bem como sua execução, uma biblioteca fornece apenas soluções para partes do problema.

No caso do presente trabalho, a ideia é constituir um *framework* cujos componentes são módulos da solução de problemas de classificação de padrões sobre dados híbridos usando algoritmos evolucionários. Como existe grande flexibilidade de configuração e parametrização desses algoritmos, o *framework* ajudará a determinar combinações coerentes e alternativas de configuração adequadas a cada situação de uso.

O desenvolvimento de um *framework* deve passar por três etapas: análise de domínio, design e instanciação. As etapas da concepção do *framework* de classificação de padrões gerado neste trabalho de doutorado serão detalhadas a seguir.

Domínio

Conforme já detalhado, o *framework* concebido visa resolver problemas de classificação de padrões, cujos bancos de dados são compostos por dados híbridos. A classificação de padrões é modelada como um problema de otimização, mais especificamente como uma

maximização multiobjetivo, cujas soluções viáveis consistem em regras de classificação. Formalmente, o modelo é dado por:

$$\text{Maximizar } F(r) = \begin{bmatrix} f_1(r) \\ f_2(r) \\ \vdots \\ f_n(r) \end{bmatrix} \quad (7.1)$$

Sujeito a: r válida para o problema em questão

onde r é uma solução (regra de classificação ou um conjunto de regras) válida para o problema e $f_i(r)$ é o valor da i -ésima função objetivo aplicada à(s) regra(s) de classificação r .

São exemplos de funções $f_i(r)$: Acurácia de r , Sensibilidade de r , $1/(\text{quantidade de termos de } r)$, etc.

Design

O *framework* proposto possui os seguintes *hot spots*:

- Funções objetivo - $f_i(r)$: o usuário pode definir o conjunto de funções objetivo a serem maximizadas, utilizando ao máximo seu conhecimento acerca do problema, a fim de auxiliar com que o *framework* extraia o melhor conjunto de regras;
- Composição do conjunto de funções: as funções presentes nas regras podem pertencer a qualquer domínio (e.g., funções trigonométricas, aritméticas, geográficas), desde que respeitem as seguintes restrições:
 - As funções respeitem a propriedade de fechamento (seção 2.4.4.1);
 - Sejam suportadas pelo Sistema de Gerenciamento de Banco de Dados que está subjacente ao *framework*.
- Estratégia de classificação. Atualmente estão presentes três possibilidades, à escolha do usuário (seção 6.7):
 - Casamento com a melhor regra;
 - Contagem de regras que casam;
 - Contagem ponderada de regras que casam.
- Parâmetros de configuração:
 - Número de gerações da Programação Genética;
 - Tamanho da População;

- Taxa de Cruzamento;
- Taxa de Mutação;
- Tamanho da População Arquivo;
- Elitismo por nicho.

Os seguintes *frozen spots* são implementados pelo *framework*:

- Cálculo da matriz de confusão para cada regra r gerada;
- Acoplamento a um Sistema de Gerenciamento de Banco de Dados (SGBD): as regras são formatadas como cláusulas WHERE da linguagem SQL e executadas pelo SGBD para que a matriz de confusão seja calculada.

Instanciação

O *framework* proposto pode ser instanciado de maneiras diferentes, de acordo com o problema a ser resolvido, bem como levando em conta a experiência do usuário que irá configurar os *hot spots*, de forma a obter os melhores resultados.

No Capítulo 6 foram apresentados diferentes problemas onde o *framework* foi aplicado. Naquele contexto, diversas instâncias de algoritmos de classificação baseados no *framework* foram utilizados. Primeiro, diversas funções objetivo foram testadas; propostas de funções objetivo retiradas da literatura foram utilizadas para identificar as melhores opções. Posteriormente, variou-se os parâmetros da Programação Genética. Finalmente, os nós do conjunto de funções foram alterados para obter o menor conjunto possível, que não comprometesse os resultados. Cabe aqui destacar que um menor conjunto de nós função implica em um menor espaço de busca, o que pode ajudar o algoritmo a convergir para um bom resultado em um menor número de iterações (gerações) do que se utilizasse um conjunto com muitas funções. Contudo, se o conjunto de nós for reduzido demais, algumas características importantes dos dados podem não ser exploradas, comprometendo o resultado final.

Estudo de Caso: Dados Geográficos

Conforme já destacado, o *framework* proposto foi aplicado em diferentes bancos de dados. Contudo, os problemas que trouxeram maior desafio do ponto de vista da sua composição foram aqueles que envolvem dados geográficos.

Nesta seção, a instância que trata dados geográficos, bem como suas principais

contribuições, serão detalhadas.

Conjunto de Funções

O conjunto de funções do *framework* é composto por funções convencionais, tais como $>$, $<$, $=$, $>=$, $<=$ e \neq . Conforme já dito anteriormente, a ferramenta é acoplada a um SGBD, que executa as funções compostas a partir dos indivíduos através do seu processador de consultas SQL. Assim, o *framework* pode utilizar quaisquer tipos de função, desde que esses tipos estejam sintaticamente corretos do ponto de vista do SQL implementado pelo SGBD e, eventualmente, suas extensões, como a geográfica.

A instância do *framework* que lida com dados geográficos utiliza o SGBD PostGIS, que é uma extensão do PostgreSQL que o complementa com diversas funções geográficas. O algoritmo de busca implementado pelo *framework*, baseado na Programação Genética, gera somente regras de classificação (indivíduos da PG) válidas, de tal maneira que o SGBD as executa, registra os resultados e permite ao *framework* calcular a matriz de confusão gerada por cada uma delas.

Por se basear em consultas SQL, o conjunto de funções também pode conter operadores definidos pelo próprio usuário. Neste trabalho foi criado um operador chamado *join geográfico*, que será detalhado a seguir.

A junção (*join*) das tabelas é uma tarefa corriqueira quando se trata do uso de um SGBD (ELMASRI; NAVATHE, 2005). Esta junção é definida em dois passos: 1) é calculado o produto cartesiano entre as tabelas, onde todas as tuplas da primeira tabela são combinadas a todas as tuplas da segunda tabela; 2) o resultado é filtrado, geralmente utilizando a cláusula SQL WHERE, fazendo com que as combinações inválidas sejam removidas. No caso mais comum, a cláusula WHERE contém uma *condição de junção*, um critério para considerar válida a combinação entre as tuplas das tabelas originais. O filtro inserido na cláusula WHERE é definido pelo usuário e pode ser uma expressão simples, e.g., *tabelaA.atributo = tabelaB.atributo* (onde só serão retornados os registros do produto cartesiano os nos quais um atributo contido na tabelaA é igual a um atributo contido na tabelaB).

Para se entender o funcionamento da função *join geográfico*, é preciso primeiro definir dois conceitos criados:

- **Tabela alvo:** Consiste na tabela do banco de dados que armazena a geometria da entidade que se deseja classificar. Considerando o exemplo da

seção 5.2, onde o objetivo é classificar o desenvolvimento das cidades, a tabela alvo é aquela que contém a geometria das cidades em questão.

- **Tabela adjacente:** é a tabela que contém a entidade que se deseja relacionar à entidade contida na tabela alvo com base em características geográficas, para extrair regras de classificação. Esse relacionamento será expresso através de uma cláusula de junção geográfica, em que uma função geográfica (e.g., CONTAINS, INSIDE, TOUCHES) é usada para comparar as geometrias das tabelas alvo e adjacentes. Cada problema pode possuir uma ou mais tabelas adjacentes, sendo necessário compor uma cláusula de junção para cada uma delas. Ainda considerando o exemplo na seção 5.2, as tabelas adjacentes contêm informações (geometria e atributos) relacionados a rodovias, aeroportos e serviços públicos. Neste caso específico trata-se de um conjunto de três tabelas adjacentes, e portanto três cláusulas de junção: cidades-rodovias, cidades-aeroportos e cidades-serviços públicos.

A função *join geográfico* utilizada nesta instância é definida como sendo uma conjunção (i.e., AND) de duas outras funções: (a) uma cláusula de junção baseada em uma função geográfica; e (b) um filtro nos atributos convencionais de uma das tabelas adjacentes. Ambas as funções visam gerar uma combinação válida entre a tabela alvo e uma das tabelas adjacentes. A fim de entender melhor o seu funcionamento, considere o exemplo a seguir.

Exemplo de aplicação de *Join Geográfico*

Considere o problema de classificação do desenvolvimento urbano de um conjunto de cidades. A geometria das cidades, bem como o nome da cidade e o código de cada uma das cidades estão armazenados na tabela chamada 'mg_mun96', cujos atributos têm os nomes 'the_geom', 'nom_muni' e 'cod_muni' respectivamente. Considere as seguintes tabelas adjacentes:

- mg_rodov: contém a geometria (the_geom) e o nome (nom_rodov) das rodovias do problema.
- mg_aero: contém, por exemplo, a geometria (the_geom), comprimento (pista_comp), largura (pista_larg) e o tipo do piso da pista (pista_piso) de cada aeroporto, além de outros atributos.

A geração de um *join geográfico* se dá de acordo com o Algoritmo 7.1.

Algoritmo 7.1 - Geração de *join geográfico*

1. Seja *fgp* o conjunto de funções puramente geográficas do problema;
2. Escolha, aleatoriamente, uma função geográfica *f* à partir do conjunto *fgp*;
3. Insira parâmetros válidos na função, de tal forma que um deles deve ser a geometria da tabela alvo e um outro parâmetro deve ser a geometria de alguma das tabelas adjacentes (*ta*);
4. Escolha, aleatoriamente, um dos atributos não geográficos da tabela adjacente, *ta*, cuja geometria foi escolhida no passo anterior. Chame este atributo de *at*;
5. Utilizando um operador relacional (=, >, <, etc), *fr*, gere um filtro válido para o atributo *at*;
6. Concatene, utilizando o operador AND, a função *f* e a função *fr*;

Os passos 1 a 3 visam obter uma chamada de função geográfica com os parâmetros preenchidos. Um deles deverá ser a geometria da tabela alvo; um outro deverá ser a geometria de uma das tabelas adjacentes. Um exemplo desta função, considerando o exemplo em questão pode ser: `contains(mg_mun96.the_geom, mg_aero.the_geom)`, que retorna verdadeiro caso a geometria contida na tabela `mg_mun96` contenha a geometria armazenada em `mg_aero`. Contudo, somente esta função apresenta pouca significado, uma vez que diversas cidades podem conter aeroportos. Assim, os passos 4 a 6 acrescentam a possibilidade de que seja criado um filtro adicional, restringindo os valores da tabela adjacente utilizando uma função relacional *fr*. Desta forma, um exemplo de função relacional seria: `mg_aero.pista_piso = 'asfalto'`. Concluindo, o operador *join geográfico* gerado será `contains(mg_mun96.the_geom, mg_aero.the_geom) AND mg_aero.pista_piso = 'asfalto'`, ou seja, a consulta retornará cidades que têm um aeroporto com pista asfaltada em seu território.

Conclusão

Nesse capítulo foi mostrado o *framework* desenvolvido, bem como um estudo de caso. A ferramenta desenvolvida se mostrou eficiente, principalmente quando aplicada em dados geográficos. Contudo, é possível aplicá-la em dados de outras naturezas, desde que se especifique o conjunto de operadores a serem aplicados nesses dados.

Referências Bibliográficas

- [1] Alcalá-Fdez, J.; Sánchez, L.; García, S.; del Jesus, M.J.; Ventura, S.; Garrell, J.M.; Otero, J.; Romero, C.; Bacardit, J.; Rivas, V.M.; Fernández, J.C.; Herrera, F.; **"KEEL: A software tool to assess evolutionary algorithms for data mining problems"**, Soft Computing, Vol. 13, N. 3, pp. 307-318, 2009.
- [2] Al-Obeidat, F; Belacel, N; Carretero, J.A.; Mahanti, P; **"An evolutionary framework using particle swarm optimization for classification method PROAFTN"**, Applied Soft Computing, Vol. 11, N. 8, pp. 4971-4980, 2011
- [3] Álvarez, V.P; Vázquez, J.M; **"An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an a priori discretization"**, Expert Systems with Applications, V. 39, N 1 , pp. 585-593, 2012.
- [4] Alves, R.T; Delgado, M.R.; Lopes, H.S.; Freitas, A.A. **"An artificial immune system for fuzzy-rule induction in data mining"**. Parallel Problem Solving from Nature - PPSN VIII. Lecture Notes in Computer Science. Vol. 3242, pp. 1011-1020, 2004.
- [5] Araújo, M.B., Pearson, R.G., Thuiller, W., Erhard, M. **"Validation of species-climate impact models under climate change"**. Global Change Biology, Vol. 11, No. 9, pp. 1504-1513, 2005.
- [6] Arunadevi, J.; Rajamani, V. **"A Two step optimized spatial Association rule Mining Algorithm by hybrid evolutionary algorithm and cluster segmentation"**. Global Journal of Computer Science and Technology, Vol. 11, N. 12, 2011.
- [7] Battiti, R., Colla, A.M. **"Democracy in neural nets: Voting schemes for classification"**. Neural Networks, Vol. 7 No.4, pp. 691-707, 1994.
- [8] Bogorny, V.; Palma, A. T.; Engel, P.M.; Alvares, L.O.; **"Weka-GDPM – Integrating Classical Data Mining Toolkit to Geographic Information Systems"** SBBW Workshop on Data Mining Algorithms and Applications (WAAMD'06), p.9-16, Florianópolis, Brasil, Out. 16-20, 2006.
- [9] Braga, A. P.; Carvalho, A. P. L. F.; Ludermir, T. B.; **"Redes Neurais Artificiais: Teoria e Aplicações"**, 2 edição. LTC, Rio de Janeiro, 2007.
- [10] Bressan, G.M., Oliveira, V.A., Hruschka Jr., E.R., Nicoletti, M.C. **"Using Bayesian networks with rule extraction to infer the risk of weed infestation in a corn-crop"**. Engineering Applications of Artificial Intelligence, Vol. 22 No.4-5, pp. 579-592, 2009.
- [11] Broomhead, D. S.; Lowe, D. **"Multivariable functional interpolation and adaptive networks"**. Complex Systems, Vol. 2, pp. 321-355, 1988.
- [12] Câmara, G. **"Representação Computacional de Dados Geográficos"**. In: Casanova, M. A.; Câmara, G.; Davis Jr, C.; Vinhas, L.; Queiroz, G. R.; (Eds.) **"Bancos de Dados Geográficos"**, MundoGEO, Curitiba, 2005
- [13] Carrano, E. G.; Wanner, E. F.; Takahashi, R. H. C. **"A Multicriteria Statistical Based Comparison Methodology for Evaluating Evolutionary Algorithms"**. In: IEEE Transactions on Evolutionary Computation, Vol. 15, N. 6, pp. 848 - 870, 2011.
- [14] Casanova, M. A.; Câmara, G.; Davis Jr, C.; Vinhas, L.; Queiroz, G. R.; (Eds.) **"Bancos de Dados Geográficos"**, MundoGEO, Curitiba, 2005.
- [15] Castanheira, L. G. **"Aplicação de Técnicas de Mineração de Dados em Problemas de Classificação de Padrões"**. Dissertação de Mestrado. PPGEE/UFGM. Belo Horizonte, Setembro de 2008.
- [16] Chang, R.; Pei, Z.; Zhang, C. **"A modified editing k-nearest neighbor rule"**. Journal of Computers, Vol. 6, N. 7, pp. 1493-1500, 2011.
- [17] Chen, M. S.; Han, J.; Yu, P. S. **"Data Mining: An Overview from a Database Perspective"**, IEEE Transaction on Knowledge and Data Engineering, Vol. 8, No. 6, Dez. 1996.

- [18] Chen, Y.L., Hung, L.T.H. "Using decision trees to summarize associative classification rules". Expert Systems with Applications, Vol. 36 No.2, P. 1, pp. 2338-2351, 2009.
- [19] Chen, Y., Wang, J.Z. "Support Vector Learning for Fuzzy Rule-Based Classification Systems". IEEE Transactions on Fuzzy Systems, Vol. 11, No. 6, pp. 716-728, 2003.
- [20] Clementini, E.; DI Felice, P.; Van Oosterom, P.; "A Small Set of Formal Topological Relationships Suitable for End-User Interaction", In: Abel, D.; Ooi, B. C.; (eds), "SSD'93: Lecture Notes in Computer Science", Vol. 692, New York, NY, Springer-Verlag, p.277-295, 1993.
- [21] Coello Coello, C.A.; "Evolutionary multi-objective optimization: a historical view of the field". Computational Intelligence Magazine, IEEE , V.1, N.1, pp. 28- 36, Feb. 2006
- [22] Coello Coello, C. A.; Lamont, G. B.; Van Veldhuizen, D. A.; "Evolutionary Algorithms for Solving Multi-Objective Problems", 2 edition, Springer, 2007.
- [23] Cohen, P.R. "Empirical Methods for Artificial Intelligence". The MIT Press, (1995).
- [24] Côrtes, S. C.; Porcaro, R. M.; Lifschitz, S. "Mineração de Dados – Funcionalidades, Técnicas e Abordagens". PUC-Rio, Maio de 2002
- [25] Cortes, C.; Vapnik, V.; "Support-vector networks". Machine Learning, Vol. 20, N. 3, pp. 273-297, 1995.
- [26] Cormen, T.H; Leiserson, C.E.; Rivest, R.L.; Stein, C. "Algoritmos - Teoria e Prática". Campus, 2002.
- [27] Davis Jr., C. A.; "Uso de vetores em GIS". Fator GIS, Curitiba (PR), v. 21, n. 4, p. 22-23, 1997.
- [28] Davis Jr., C. A.; Queiroz, G.R.; "Algoritmos geométricos e relacionamentos topológicos" In: Casanova, M. A.; Câmara, G.; Davis Jr, C.; Vinhas, L.; Queiroz, G. R.; (Eds.) "Bancos de Dados Geográficos", MundoGEO, Curitiba, 2005.
- [29] Deb, K.; "Multi-Objective Optimization using Evolutionary Algorithms". John Wiley & Sons, 2001.
- [30] Dehuri, S.; Patnaik, S.; Ghosh, A.; Mall, R. "Application of elitist multi-objective genetic algorithm for classification rule generation" Applied Soft Computing 8, p. 477–487, 2008.
- [31] Dietterich, T.G.; Lathrop, R.H.; Lozano-Perez, T.; "Solving the multiple instance problem with axis-parallel rectangles", Artificial Intelligence, Vol. 89, N. (1–2), pp. 31–71, 1997.
- [32] Dorigo, M.; Stützle, T. "Ant Colony Optimization". MIT Press, Cambridge, MA, 2004.
- [33] Douglas, D. H.; Peucker, T. K. "Algorithms for the reduction of the number of points required to represent a line or its caricature". The Canadian Cartographer, Vol. 10, N. 2, p. 112-122, 1973.
- [34] Duda, R.O.; Hart, P.E.; Stork, D.G. "Pattern Classification". 2ed. Wiley-Interscience, 2000.
- [35] Egenhofer, M. A. "Model for Detailed Binary Topological Relationships". Geomatica, v. 47, p. 261-273, 1993.
- [36] Egenhofer, M.; DI Felice, P.; Clementini, E.; "Topological Relations Between Regions with Hole". International Journal of Geographic Information Systems, Vol. 8, No. 2, p.129-144, 1994.
- [37] Egenhofer, M.; Franzosa, R. "On the Equivalence of Topological Relations". International Journal of Geographical Information Systems, Vol. 9, No. 2, p.133-152, 1995.
- [38] Egenhofer, M; Herring, J.; "Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographical Databases". Orono, ME: Department of Surveying Engineering, University of Maine, 1991.
- [39] Elmasri, R.E; Navathe, S.; "Sistemas de Banco de Dados". 4 ed. Addison-Wesley, 2005.
- [40] Ester, M.; Kriegel, A. F. H. P.; Sander, J. "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support" Data Mining and Knowledge Discovery, Vol. 4, N. 3-4, pp.193–216, 2000.

- [41] Exarchos, T.P., Tsipouras, M.G., Exarchos, C.P., Papaloukas, C., Fotiadis, D.I., Michalis, L.K. "**A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhythmic beat classification based on a set of rules obtained by a decision tree**". Artificial Intelligence in Medicine, Vol. 40, No. 3, pp. 187-200, 2007.
- [42] Feki, A.; Ishak, A.B.; Feki, S.; "**Feature selection using Bayesian and multiclass Support Vector Machines approaches: Application to bank risk prediction**". Expert Systems with Applications, Vol. 39, N. 3, pp. 3087-3099, 2012.
- [43] Fidalgo, R. N.; Times, V. C.; Silva, J.; e Souza, F. F. "**Geodwframe: A framework for guiding the design of geographical dimensional schemas**". In Data Warehousing and Knowledge Discovery (DaWaK), pp. 26–37, 2004.
- [44] Fonseca, F.; Egenhofer, M.; Davis, C.; Câmara, G.; "**Semantic Granularity in Ontology-Driven Geographic Information Systems**", *Annals of Mathematics and Artificial Intelligence*, Hingham, MA, USA, Vol. 36, Issue 1-2, p. 121 – 151, Set. 2002.
- [45] Freitas, A. A. "**Data Mining and Knowledge Discovery with Evolutionary Algorithms**", Springer, 2002. Natural Computing Series
- [46] Fu, J.; Lee, S. "**A multi-class SVM classification system based on learning methods from indistinguishable chinese official documents**". Expert Systems with Applications, Vol. 39, N. 3, pp. 3127-3134, 2012.
- [47] GeoMINAS - Programa Integrado de Uso da Tecnologia de Geoprocessamento pelos Órgãos do Estado de Minas Gerais. <http://www.geominas.mg.gov.br/>. Acessado in Jan. 2010
- [48] Goldberg, D. E. "**Genetic Algorithms in Search, Optimization and Machine Learning**", Addison Wesley, 1989
- [49] Goldberg, D. E.; Richardson, J. "**Genetic Algorithm with sharing for multimodal function optimization**". Proceedings of the Second International Conference on Genetic Algorithm, 1987
- [50] Gustafson, S.; Vanneschi, L. "**Crossover-Based Tree Distance in Genetic Programming**", IEEE Transactions on Evolutionary Computation, 2008
- [51] Han, J.; Kamber, M. "**Data Mining: Concepts and Techniques**" Morgan Kaufmann, USA, 2001.
- [52] Han, J.; Koperski, K.; e Stefanovic, N. "**GeoMiner: A System Prototype for Spatial data Mining**". In ACM SIGMOD'97, 1997.
- [53] Haykin, S. "**Neural Networks and Machine Learning**". 3 ed., Prentice Hall, 2008.
- [54] Holland, J. H. "**Adaptation in Natural and Artificial Systems**". The University of Michigan Press, 1975.
- [55] Hongguang, M.; Chongzhao, H.; Guohua, W.; Jianfeng, X.; Xiaofei, Z.; "**Data-mining based fault detection**", Journal of Electronics (China), Volume 22, Number 6, p. 605-611, Nov. 2005.
- [56] Hsu, C.-W.; Lin, C.-J.; "**A comparison of methods for multiclass support vector machines**". IEEE Transactions on Neural Networks, Vol. 13, N. 2, pp. 415-425, 2002.
- [57] Hu, W., Wu, O., Chen, Z., Fu, Z., Maybank, S. "**Recognition of pornographic Web pages by classifying texts and images**". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No.6, pp. 1019-1034, 2007.
- [58] Huang, B.; Kechadi, M.T.; Buckley, B.; "**Customer churn prediction in telecommunications**", Expert Systems with Applications, Vol 39, N. 1, pp. 1414-1425, 2012.
- [59] Jain, A. K.; Murty, M. N.; Flynn, P.J.; "**Data clustering: A review**", ACM Computing Surveys 31 (3), p. 316-323, 1999.
- [60] Jiang, X., Nariai, N., Steffen, M., Kasif, S., Kolaczyk, E.D. "**Integration of relational and hierarchical network information for protein function prediction BMC**". Bioinformatics, 9, no. 350, (2008).
- [61] Kabir, M.M.; Shahjahan, M.; Murase, K. "**A new hybrid ant colony optimization algorithm for feature selection**". Expert Systems with Applications, Vol. 39, N. 3, pp. 3747-3763, 2012.

- [62] Kennedy, J.; Eberhart, R. C.; "**Particle swarm optimization**". In Proc. in IEEE International Conference on Neural Networks, p. 1942–1948, 1995.
- [63] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. "**Optimization by simulated annealing**". Science, Vol. 220, No. 4598, pp. 671-680, 1983.
- [64] Kittler, J; Hatef, M; Duin, R.P.W.; Matas, J. "**On Combining Classifiers**". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3, 1998.
- [65] Köppen, M.; "**No-free-lunch theorems and the diversity of algorithms**". Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, 1, pp. 235-241, 2004.
- [66] Koza, J.R. "**Genetic Programming: on the programming of computers by means of natural selection**". MIT Press, 1992.
- [67] Lam, A.Y.S.; Li, V.O.K.; "**Generalization of the no-free-lunch theorem**". Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, art. no. 5346796, pp. 4322-4328, 2009.
- [68] Laumanns, M.; Rudolph, G.; e Schwefel, H. P.; "**A spatial predator-prey approach to multi-objective optimization: a preliminary study**". Proceedings of the Fifth Conference on Parallel Problem Solving from Nature (PPSN V), p. 241-249, 1998, Springer.
- [69] Lee, A.J.T.; Chen, Y.-A.; Ip, W.-C.; "**Mining frequent trajectory patterns in spatial-temporal databases**", Information Sciences, Vol. 179, Iss. 13, (2009), pp.2218-223.
- [70] Lei, J.Z.; Ghorbani, A.A. "**Improved competitive learning neural networks for network intrusion and fraud detection**". Neurocomputing, Vol. 75, pp. 135-145, 2012.
- [71] Li, M; Chen, X; Li, X; Ma, B; Vitanyi, P.M.B.; , "**The similarity metric**" Information Theory, IEEE Transactions on , V.50, N.12, pp. 3250- 3264, Dec. 2004
- [72] Liao, S.H. "**Expert system methodologies and applications-a decade review from 1995 to 2004**". Expert Systems with Applications, Vol. 28, No. 1, pp. 93-103, 2005.
- [73] Lin, Y. "**Support vector machines and the Bayes rule in classification**". Data Mining and Knowledge Discovery, Vol. 6, No. 3, pp. 259-275, 2002.
- [74] Lin, C.J.; Peng, C.C. "**A self-adaptive quantum radial basis function network for classification applications**". International Journal of Innovative Computing, Information and Control. Vol. 7, N. 12, pp. 6621-6634, 2011
- [75] Lindman, H. R. "**Analysis of Variance in Complex Experimental Designs**". San Francisco, USA: W. H. Freeman & Co, 1974.
- [76] Lorena, A.C.; Carvalho, A.C.P.L.F. "**Uma Introdução às Support Vector Machines**". Revista de Informática Teórica e Aplicada - RITA, Vol. 14, N. 2, pp. 43-67, 2007.
- [77] Lu, H.; Setiono, R.; Liu, H. "**Effective data mining using neural networks**". IEEE Transactions on Knowledge and Data Engineering 8 (6), pp. 957-961, 1996.
- [78] Mantas, C.J., Puche, J.M., Mantas, J.M. "**Extraction of similarity based fuzzy rules from artificial neural networks**". International Journal of Approximate Reasoning, Vol. 43, No. 2, pp. 202-221, 2006.
- [79] Markiewicz, M.; Lucena, C.J.P. "**Understanding object-oriented framework engineering**". Tech Reports. 11 p., 2000. Disponível em: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/00_38_markiewicz.pdf
- [80] Martens, D., Baesens, B., Van Gestel, T., Vanthienen, J. "**Comprehensible credit scoring models using rule extraction from support vector machines**". European Journal of Operational Research, Vol. 183, No. 3, pp. 1466-1476, 2007.
- [81] McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.; "**Grammar-based Genetic programming: A survey**". Genetic Programming and Evolvable Machines, Vol. 11, N. (3-4), pp. 365-396, 2010.

- [82] Melgani, F., Bruzzone, L.; "**Classification of hyperspectral remote sensing images with support vector machines**". IEEE Transactions on Geoscience and Remote Sensing, Vol. 42, N. 8, pp. 1778-1790, 2004
- [83] Menahem, E.; Rokach, L.; Elovici, Y.; "**Troika – An improved stacking schema for classification tasks**". Information Sciences, Vol. 179, Iss. 24, (2009), pp. 4097-4122.
- [84] Ministério do Meio Ambiente, <http://www.mma.gov.br/sitio/>, acessado em Fevereiro de 2009
- [85] Mohamad, M.S., Omatu, S., Deris, S., Yoshioka, M. "**A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data**". IEEE Transactions on Information Technology in Biomedicine, Vol. 15, N. 6, art. no. 6017123, pp. 813-822, 2011.
- [86] Montana, D.J. "**Strongly Typed Genetic Programming**". Evolutionary Computation, Vol. 3, No. 2, pp. 199-230, 1995.
- [87] Morais, D.R.; Rolim, J.G.; "**A hybrid tool for detection of incipient faults in transformers based on the dissolved gas analysis of insulating oil**". IEEE Transactions on Power Delivery, Vol. 21, No. 2, pp. 673- 680, 2006.
- [88] Montgomery, D.C; Runger, G.C; "**Estatística Aplicada e Probabilidade para Engenheiros**". 4 ed. LTC, 2009.
- [89] Murphey, Y.L.; Guo, H.; Feldkamp, L.A. "**Neural Learning from Unbalanced Data**". Applied Intelligence, No. 21, pp. 117–128, 2004.
- [90] OpenGIS. www.ogc.org acessado em Janeiro de 2009
- [91] Oxford American Dictionary. Versão digital 2.0.2, 2007
- [92] Pappa, G. L.; Freitas, A. A.; "**Evolving rule induction algorithms with multi-objective grammar-based genetic programming**". Knowledge and Information Systems, Vol. 19, No. 3, pp. 283-309, Jun. 2009. <http://dx.doi.org/10.1007/s10115-008-0171-1>
- [93] Parpinelli, R.S.; Lopes, H.S. and Freitas, A.A. "**Data Mining with an Ant Colony Optimization Algorithm**". IEEE Trans. on Evolutionary Computation, special issue on Ant Colony algorithms, 6(4), pp. 321-332, 2002.
- [94] Pavlopoulos, S.A.; Stasis, A.C.H.; Loukis, E.N.; "**A decision tree - Based method for the differential diagnosis of Aortic Stenosis from Mitral Regurgitation using heart sounds**". BioMedical Engineering Online, Vol. 3, art. no. 21, 15p., 2004.
- [95] Pereira, M. A.; Vasconcelos, J.A. "**A Niche Genetic Algorithm for Classification Rules Discovery in Real Databases**". In XVIII Congresso Brasileiro de Automática - CBA, pp. 824-829. Bonito, MS. Brazil, 2010.
- [96] Pereira, M.A; Davis Junior, C.A; Vasconcelos, J.A. "**A Niche Genetic Programming Algorithm for Classification Rules Discovery in Geographic Databases**". In: SEAL 2010, Lecture Notes in Computer Science - LNCS, V. 6457, pp. 260-269, 2010.
- [97] Phillips, P.; Lee, I.; "**Mining co-distribution patterns for large crime datasets**", Expert Systems with Applications, Vol. 39, I. 14, pp. 11556-11563, 2012.
- [98] Piatetsky-Shapiro, G.; Frawley, W. J.; "**Knowledge Discovery in Databases.**" AAAI/MIT Press, 1991.
- [99] Qasem, S.N; Shamsuddin, S.M.; "**Memetic Elitist Pareto Differential Evolution algorithm based Radial Basis Function Networks for classification problems**". Applied Soft Computing, Vol. 11, N. 8, pp. 5565-5581, 2011.
- [100] Quinlan, J. R. "**Induction of decision trees**", Machine Learning 1 (1), pp. 81-106, Mar 1986.
- [101] Quinlan, J.R., "**C4.5: Programs for Machine Learning**". Morgan Kaufmann, 1993.
- [102] Romero, C; Ventura, S.; "**Educational data mining: A survey from 1995 to 2005**", *Expert Systems with Applications*, Vol. 33, Issue 1, P. 135-146, 3 Mai. 2006, <<http://dx.doi.org/10.1016/j.eswa.2006.04.005>>. Acessado em 17 Fev. 2009.

- [103] Romero, C.; Zafra, A.; Luna, J.M.; Ventura, S.; "Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data". Expert Systems, 2012.
- [104] Samantaray, S.R. "Decision tree-initialised fuzzy rule-based approach for power quality events classification". IET Generation, Transmission and Distribution, Vol. 4, No. 4, pp. 538-551, 2010.
- [105] Sarkar, B.K.; Sana, S.S.; Chaudhuri, K. "A genetic algorithm-based rule extraction system". Applied Soft Computing Journal, Vol. 12, N. 1, pp. 238-254, 2012.
- [106] Shekhar, S.; Chawla, S. "Spatial Databases: A Tour", Pearson Education, Upper Saddle River, New Jersey, 2003.
- [107] Shekhar, S.; Lu, C. T.; Tan, X.; e Chawla, S. "Map cube: A visualization tool for spatial data warehouse", 2000
- [108] Shintemirov, A.; Tang, W.; Wu, Q. H. "Power Transformer Fault Classification Based on Dissolved Gas Analysis by Implementing Bootstrap and Genetic Programming". IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews. Vol. 39, N. 1, pp. 69-79, 2009.
- [109] Sikora, R.; Piramuthu, S. "Framework for efficient feature selection in genetic algorithm based data mining". European Journal of Operational Research 180, p. 723–737, 2007.
- [110] Silva, J.; Times, V. C.; Salgado, A. C. "An Open Source and Web Based Framework for Geographic and Multidimensional Processing" Proceedings of the 2006 ACM symposium on Applied computing, p63 – 67, 2006.
- [111] Soares, G. L.; "Algoritmos Determinísticos e Evolucionários Intervalares para Otimização Robusta Multi-Objetivo", Tese de Doutorado, PPGEE/UFGM, Belo Horizonte, Outubro de 2008.
- [112] Sousa, T.; Silva A.; Neves, A. "Particle Swarm based Data Mining Algorithms for classification tasks". Parallel Computing, Vol. 30. No.5-6, pp. 767-783, 2004.
- [113] Srinivasa, K. G.; Venugopal, K. R.; Patnaik, L. M. "A self-adaptive migration model genetic algorithm for data mining applications" Information Sciences 177, p. 4295–4313, 2007.
- [114] Strobl, C. "Dimensionally Extended Nine-Intersection Model (DE-9IM)" Disponível em: http://gis.hsr.ch/wiki/images/3/3d/9dem_springer.pdf . Acessado em Jul. de 2010.
- [115] Sun, D.; Yu, Y.; Goldberg, M.D.; "Deriving water fraction and flood maps from MODIS images using a decision tree approach". IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 4, N. 4, art. no. 5766784, pp.814-825, 2011.
- [116] Tan, K. C.; Yu, Q.; Lee, T. H.; "A Distributed Evolutionary Classifier for Knowledge Discovery in Data Mining" IEEE Transactions on Systems, Man, And Cybernetics – Part C: Applications and Reviews, Vol. 35, No. 2, May, 2005.
- [117] Übeyli, E.D.; "Multiclass support vector machines for diagnosis of erythemato-squamous diseases". Expert Systems with Applications, Vol. 35, N. 4, pp. 1733-1740, 2008.
- [118] UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. <http://archive.ics.uci.edu/ml>. Acessado em jun. 2010
- [119] Vasconcelos, J. A.; Ramírez, J. A.; Takahashi, R. H. C.; e Saldanha, R. R.; "Improvements in Genetic Algorithms", IEEE Transactions on Magnetics, 3414-3417, Vol. 37, No. 5, Set, 2001.
- [120] Vieira, D.A.G.; "Rede Perceptron Com Camadas Paralelas (plp - Parallel Layer Perceptron)", Tese de Doutorado, PPGEE/UFGM, Belo Horizonte, Dezembro de 2006
- [121] Vieira, D.A.G., Vasconcelos, J.A. Caminhas, W.M. "Controlling the parallel layer perceptron complexity using a multiobjective learning algorithm". Neural Computing & Applications, Springer London, Vol. 16, No.4, pp. 317-325, 2007.
- [122] Yang, S., Jian, H., Ding, Z., Hongyuan, Z., Giles, C.L. "IKNN: Informative K-nearest neighbor pattern classification". Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4702, LNAI, pp. 248-264, 2007.

- [123] Wang, X.-L.; Cheng, J.-H.; Yin, Z.-J.; Guo, M.-J. "**A new approach of obtaining reservoir operation rules: Artificial immune recognition system**". Expert Systems with Applications, Vol. 38, N. 9, pp. 11701-11707, 2011.
- [124] Wang, J., Neskovic, P., Cooper, L.N. "**Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence**" Pattern Recognition, Vol. 39, No. 3, pp. 417-423, 2006.
- [125] Wang, L.; Zhou, L.; Lu, J.; Yip, J.; "**An order-clique-based approach for mining maximal co-locations**", Information Sciences, Vol. 179, Iss. 19, (2009), pp.3370-3382.
- [126] Whigham, P.A.; "**Inductive bias and genetic programming**", IEE Conference Publication, N. 414, pp. 461-466, 1995.
- [127] Whigham, P. A. "**Induction of a marsupial density model using genetic programming and spatial relationships**", Ecological Modelling, Vol. 131, pp. 299-317, 2000.
- [128] Witten, I.H.; Frank, E. "**Data Mining - Practical Machine Learning Tools and Techniques**". 2 ed., Margan Kaufmann, 2005.
- [129] Weka Software <http://www.cs.waikato.ac.nz/ml/weka/> acessado em janeiro de 2009.
- [130] Wolpert, D.H.; Macready, W.G.; "**No free lunch theorems for optimization**". IEEE Transactions on Evolutionary Computation, Vol. 1, N. 1, pp. 67-82, 1997.
- [131] Worboys, M.F.; Duckham, M. "**GIS – A Computing Perspective**", 2 ed., Boca Raton: CRC Press, 2004.
- [132] Wu, H. C.; Lu, C. N.; "**A Data Mining Approach for Spatial Modeling in Small Area Load Forecast**", *IEEE Transactions on Power Systems*, Vol. 17. No. 2, p. 516-521, Mai, 2002. <<http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1007927>>. Acessado em 17 Fev. 2009
- [133] Zafra, A.; Ventura, S. "**G3P-MI: A genetic programming algorithm for multiple instance learning**". Information. Sciences. Vol. 180, N. 23 pp. 4496-4513, 2010.
- [134] Zheng, Y.; Xie, X.; "**Learning travel recommendations from user-generated GPS traces**". ACM Transactions on Intelligent Systems and Technology, Vol. 2, I. 1, 2011.
- [135] Zhang, H.; Liu, G.; Chow, T.W.S.; Liu, W. "**Textual and visual content-based anti-phishing: A Bayesian approach**". IEEE Transactions on Neural Networks, Vol. 22, N. 10, art. no. 5975221, pp. 1532-1546, 2011.
- [136] Ziviani, N.; Botelho, F. C. (Consultoria em Java e C++) "**Projeto de Algoritmos – com implementações em Java e C++**". Editora Cengage Learning, 2006.