CELSO YOSHIKAZU ISHIDA

PROGRAMAÇÃO GENÉTICA ORIENTADA A GRAMÁTICA E A MINERAÇÃO DE BASE DE DADOS RELACIONAIS

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre. Curso de Pós-Graduação em Informática, Setor de Ciências Exatas da Universidade Federal do Paraná.

Orientadora: Prof.ª Dr.ª Aurora T. R. Pozo

CURITIBA
2002

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática do aluno *Celso Yoshikazu Ishida*, avaliamos o trabalho intitulado, "Programação Genética Orientada a Gramática e a Mineração de Base de Dados Relacional", cuja defesa foi realizada no dia 09 de maio de 2002, às quatorze horas, no anfiteatro A do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 09 de maio de 2002.

Profª. Dra. Aurora Trinidad Ramirez Pozo
DINF/UFPR - Orientadora

Profª. Dra. Maria Carolina Monard
USP-SC Membro Externo

Profª. Dra. Silvia Regina Vergilio
DINF/UFPR

Baseado nesta dissertação, foi escrito e aceito um artigo para publicação no fórum *Congress on Evolutionary Computation* (CEC) do *2002 IEEE World Congress on Computational Intelligence* (WCCI).

GPSQL Miner: SQL-Grammar Genetic Programming in Data Mining

Celso Y. Ishida, Aurora T. R. Pozo
Federal University of Paraná - Computer Science Department
cishida@inf.ufpr.br, aurora@inf.ufpr.br

Abstract - The present work describes GPSQL Miner, a Grammar based Genetic Programming system for mining relational databases. This system uses Grammar Genetic Programming for classification's task and one of its main features is the representation of the classifiers. The system automatically creates an adequate grammar for each database using Database Management System (DBMS). The use of Grammar based Genetic Programming and DBMS enable the process automation. The tool was tested with some databases and the results were compared with other algorithms. These first experiments have shown promising results for the classification task.

Resumo

Na economia atual, a análise de informações que agregam valor aos negócios é essencial para a sobrevivência das empresas. Devido ao enorme volume de informações existentes, as ferramentas de mineração de dados (*Data Mining*) são a promessa para a descoberta das informações mais interessantes para os executivos das grandes corporações.

Este trabalho projeta e implementa uma nova ferramenta de *Data Mining* para a construção de classificadores a partir de bancos de dados relacionais, denominada GPSQL Miner. Esta ferramenta gera classificadores em formato SQL utilizando Programação Genética (GP) orientada à gramática (GGP – *Grammar Genetic Programming*) e acessa os registros através de um Sistema Gerenciador de Banco de Dados (SGBD).

O paradigma GP foi escolhido devido à grande capacidade apresentada no tratamento de dados inválidos ou imprecisos, e a facilidade de adaptá-lo a diferentes aplicações, seja pela configuração de parâmetros ou pela implementação ou modificação de operadores. Apesar destas facilidades, em GP, indivíduos inválidos podem ser gerados. Com a utilização da gramática, é possível evitar este problema fazendo com que apenas indivíduos válidos sejam gerados, além de possibilitar uma melhor definição do espaço de busca.

A localização das informações no banco de dados relacional tem um papel importante para a mineração de dados. A partir das informações contidas no SGBD, o sistema cria uma gramática adequada para cada problema a ser utilizada pela GGP para criação dos classificadores. A criação automática da gramática permite a automação do processo.

O trabalho propõe um novo formato para os classificadores que lembra um resumo do comando SQL: SELECT. Este comando foi escolhido pois a sua função de

selecionar os registros de acordo com uma restrição lembra como será feita a avaliação das regras dentro de um banco de dados relacional.

A eficiência da ferramenta é avaliada através da comparação com vários outros algoritmos em várias bases de teste. Os resultados obtidos mostram que a ferramenta é eficiente para a tarefa de classificação.

Índice

Índice de Figuras

Índices de Tabelas

Glossário de Termos Utilizados

Sigla	Definição
Accuracy	Precisão
AG	Algoritmo Genético
Best	Melhor Classificador
Best-Test	Melhor Classificador na base de teste
BNF	Backus-Naur Form
CFG	Context-free grammars
Crossover	Operador Genético de Cruzamento
Data Mining	Mineração de Dados
Data Warehouse	Armazém de Dados
Derivation Trees	Árvores de Derivação
ERCs	Constantes Aleatórias Efêmeras
ERP	Enterprise Resource Planning – Sistema de Gestão Empresarial
Fitness	Função de Avaliação
Full	Um Método de Inicialização da População inicial
GGP	Grammar Genetic Programming
GP	Genetic Programming
GPSQL Miner	Sistema para classificadores SQL que utiliza Grammar Genetic Programming
Grow	Um Método de Inicialização da População inicial
ID	Número Interno da Conexão com banco de dados Oracle
ILP	Inductive Logic Programming
Join	Relacionamento entre duas ou mais tabelas
LOGENPRO	The LOgic grammar based GENetic PROgramming system
Mutation	Operador Genético de Mutação
KDD	Knowledge Discovery Database
Predicting Attributes	Atributos de Predição
Run	Uma execução. Indica que o sistema achou um classificador com um conjunto fixo de parâmetros
SGBD	Sistema Gerenciador de Banco de Dados
Tupla	Registro de uma tabela

1. Introdução

No cenário da economia atual, as empresas têm uma grande necessidade de obter informações que realmente agregam valor às decisões gerenciais. Estas informações, não triviais, são importantes para o melhoramento da qualidade de seus produtos ou até mesmo para a abertura de novos negócios. Pode-se dizer que tal atividade é uma necessidade para a sobrevivência de qualquer corporação [Ishida 1999].

As informações da maioria das grandes corporações estão armazenadas e organizadas em sistemas de ERP (*Enterprise Resource Planning*). Os sistemas de ERP podem organizar todas as tarefas operacionais da empresa, possibilitando ao usuário modelar todo o panorama de informações que possui e integrá-las segundo suas funções operacionais. Um sistema ERP é basicamente composto de módulos que atendem às necessidades de informação para apoio à tomada de decisão de setores. Todos os módulos são integrados entre si, formando uma base de dados única e não redundante.

Os sistemas de ERP conseguem organizar os dados operacionais, contudo, é necessário um trabalho adicional para o aproveitamento das informações para as decisões gerenciais. Em meio a um enorme volume de informações, se faz necessário e útil a produção de um resumo através de um *Data Warehouse* (armazém de dados) para a seleção dos dados mais importantes.

A construção do *Data Warehouse* forma a base para a aplicação de toda uma nova geração de ferramentas para análise e exploração de dados. Cabe ao *Data Warehouse* organizar, limpar e estruturar todos os dados de uma organização, podendo ser visto como excelente fonte de dados para a aplicação de algoritmos que revelam o conhecimento implícito sobre a organização, ou seja, descobrir regras sobre seus relacionamentos com clientes, análise de crédito, problemas de qualidade, gerência de materiais e muitas outras aplicações.

O processo de descoberta de conhecimento pode ser conhecido como *Knowledge Discovery Database* (KDD). KDD pode ser definido como um processo não trivial para identificação de padrões válidos, novos, potencialmente úteis e compreensíveis [Fayyad 1996]. As informações encontradas devem ser novas e benéficas aos usuários.

KDD é um processo interativo composto por 5 passos [Fayyad 1996] que poderão ser repetidos se o conhecimento descoberto não for satisfatório.

1. Seleção dos conjuntos de dados relevantes do banco de dados.

2. Pré-processamento dos dados para tratamento de ruídos e campos incompletos.

3. Transformação dos dados para redução do número de variáveis a serem consideradas.

4. Utilização de um algoritmo conveniente para mineração de dados (*Data Mining*) do modelo selecionado sobre os dados preparados.

5. O resultado do *Data Mining* é interpretado e avaliado.

Dentro do processo de KDD, o passo de *Data Mining* é um dos mais importantes. Existem vários algoritmos para realização deste passo: Classificadores Bayesianos [Weiss 1991] e Redes Neurais, Indução através de Árvores de Decisões e Regras [Mitchell 1997], Aprendizado de Regras de Associação [Agrawal 1993], Agrupamento (*Clustering*), Algoritmos Genéticos [Holland 1992], Programação Genéticas [Wong 2000] e diferentes tarefas a serem realizadas que serão detalhadas no capítulo 2. Dentro destas tarefas escolhemos como foco deste trabalho a tarefa de classificação.

Algoritmos de Aprendizado de Máquina (AM) têm sido úteis para classificação em *Data Mining.* Porém, algumas observações são relevantes: os algoritmos de AM tentam aprender utilizando pequenos volumes de dados; em contraste a esta perspectiva, *Data Mining* tem como objetivo trabalhar com grandes volumes de dados. Desta forma, é comum

realizar o processo de classificação em *Data Mining* considerando pelo menos duas etapas. Na primeira etapa, os classificadores são induzidos em diferentes conjuntos de dados utilizando AM. A segunda etapa combina estes classificadores numa abordagem integrada. Este trabalho enfoca a primeira etapa que é a indução de classificadores em pequenos volumes de dados.

Uma das maiores aplicações da tarefa de classificação em *Data Mining* são os sistemas de apoio à decisão, entre eles destacamos: aplicações reais como análise de risco e previsões. Em qualquer destas aplicações os dados manipulados são imprecisos, incompletos e ambíguos. Além disso, o conhecimento humano também é impreciso e incerto. Entretanto, as aplicações de mineração de dados envolvem tomadas de decisões que devem ser corretas e precisas, usando dados complexos, incompletos e, às vezes, incorretos [Ramos 1999].

Um paradigma que se destaca nestas condições é a Programação Genética (GP). A GP apresenta grande capacidade no tratamento de dados inválidos ou imprecisos e facilidade de adaptação a diferentes aplicações, seja pela configuração de parâmetros ou pela implementação ou modificação de operadores [Wong 2000]. Outra vantagem desta abordagem é a implementação implícita de procuras paralelas. Um sistema de GP procura pelas soluções mais promissoras em todo o espaço de busca em forma paralela [Wong 2000].

Apesar destas facilidades, GP pode gerar indivíduos inválidos prejudicando ou adiando a procura pelo melhor classificador. Com a utilização da gramática, a Programação Genética Orientada à gramática (GGP – *Grammar Genetic Programming*) gera apenas indivíduos válidos através da definição do espaço de busca, além de orientar todas as operações genéticas.

GP é uma área ainda pouco explorada no contexto de mineração de dados apresentando grandes desafios e muitas linhas de pesquisa. Dentre os poucos trabalhos existentes que envolvam GP e *Data Mining*, pode-se citar o LOGENPRO. LOGENPRO (The LOgic grammar based GENetic PROgramming system) é um sistema que combina

GP com *Inductive Logic Programming* (ILP) para a tarefa de *Data Mining*. As regras do LOGENPRO estão no formato 'if-then', e cada regra é representada como um indivíduo. Cada regra possui um número fixo de atributos tendo a necessidade da criação de operadores especiais. O sistema LOGENPRO utiliza gramática para orientação da criação dos indivíduos.

1.1 Objetivos

Este trabalho implementa o sistema GPSQL Miner, uma nova ferramenta de *Data Mining* integrada a um banco de dados relacional, especificamente para a tarefa nobre de classificação. O sistema utiliza-se da GGP para criação dos classificadores em formato SQL com acesso ao banco de dados relacional Oracle®.

Uma característica explorada neste trabalho refere-se à integração com banco de dados relacional. A integração facilita a pesquisa das informações necessárias para a criação automática de uma gramática adequada a cada problema. A utilização da gramática, juntamente com a integração do banco de dados, permite a automação do processo. Com isso, o sistema é capaz de classificar necessitando apenas da informação de quais atributos do banco de dados serão utilizados. Entre outras vantagens da integração com o Sistema Gerenciador de Banco de Dados (SGBD) é permitir ao sistema a manipulação de grande volumes de dados, controle de segurança, entre outras vantagens [Duarte 2001]. Embora o sistema esteja projetado para manipular grandes bases de dados, este trabalho focaliza na primeira fase de *Data Mining* que é a utilização de AM em pequenas bases de dados.

Uma vez que, as informações estejam em um banco de dados relacional, propõe-se um novo formato para os classificadores que lembra um resumo do comando SQL: SELECT. Este comando foi escolhido pois a sua função de selecionar os registros de acordo com uma restrição lembra como será feita a avaliação das regras dentro de um banco de dados relacional. Um outro motivo é que o novo formato mostra uma definição

clara e simples para o conhecimento descoberto. O conhecimento descoberto é similar à regra '*if-then*'.

1.2 Organização do Trabalho

O conceito de *Data Mining* é introduzido no Capítulo 2 juntamente com a tarefa de classificação. No final do mesmo, um exemplo é descrito para melhor entendimento da tarefa de classificação.

No Capítulo 3, os principais conceitos da GP e os parâmetros utilizados são introduzidos. Logo após, define-se a GGP.

Um trabalho relacionado é citado no Capítulo 4. O trabalho descrito é o LOGENPRO (The LOgic grammar based GENetic PROgramming system) um sistema que combina GP com *Inductive Logic Programming* (ILP).

No Capítulo 5, um novo formato de classificador é definido. Logo após, o sistema GPSQL Miner é apresentado e seus módulos são descritos em detalhes. São apresentados os 3 módulos principais: SQL-BNF, GGP e Oracle-Evaluate. Logo após a sua definição apresenta-se um exemplo completo de funcionamento do sistema.

No capítulo 6, existe um exemplo para mostrar as funcionalidades do sistema. São apresentados exemplos para os arquivos de entrada, como SQL Parameter e BNF; exemplos de arquivos de saída e exemplos de como os resultados são resumidos e apresentados.

Os experimentos e os resultados comparados com outros algoritmos estão no Capítulo 7. Já a conclusão e os trabalhos futuros fazem parte do último capítulo.

2. Data Mining

O termo *Data Mining* ou mineração de dados é utilizado para denotar a descoberta de padrões úteis dos dados. Consiste na aplicação de análise de dados e algoritmos de descoberta para produzir padrões ou modelos.

Data Mining tem se tornado recentemente um tópico popular de pesquisa, sendo que, um dos motivos é o aumento da exploração de informações e no volume das bases de dados. Podem-se ter todas as informações dentro de um *Data Warehouse*, contudo, é necessário que as informações úteis sejam descobertas através de ferramentas, como por exemplo, ferramentas de *Data Mining*. Muitas técnicas de *Data Mining* aplicam métodos de busca para poderem encontrar estas informações novas, úteis e interessantes [Wong 2000].

Diferentes algoritmos de *Data Mining* têm como objetivo achar diferentes tipos de conhecimento. Chen [Chen 1996] agrupou as técnicas para descoberta de conhecimento em seis categorias:

1. Mineração de regras de associação. Procura associação entre regras na forma: "$A_1 \wedge ... \wedge A_m \Rightarrow B_1 \wedge ... \wedge B_n$", aonde A_i e B_j são atributos. A regra significa que se A_1 e .. e A_m aparecer nos registros, então B_1 e ... e B_n usualmente aparecerão.

2. Generalização de dados e sumarização. Destaca de modo resumido as características gerais de um grupo da classe objetivo e presente nos dados numa visão de alto nível.

3. Classificação. Formula um modelo de classificação baseado nos dados. O modelo pode ser usado para classificar um item despercebido dentro de uma das classes pré-definidas baseando-se no valor dos atributos.

4. *Data Clustering*. Identifica um conjunto finito de grupos ou categorias para descrever os dados. Os itens são agrupados dentro de grupos de tal forma que a semelhança entre os grupos é minimizada e a semelhança dentro dos elementos do grupo é maximizada. As características comuns dos grupos são analisadas e apresentadas.

5. *Pattern based similarity search* (padrão baseado na procura de similaridades). Procura por padrões em dados temporários ou específicos, tais como banco de dados financeiros ou de multimídia.

6. *Mining path traversal patterns*. Tenta capturar o acesso de padrões de informações em sistemas de fornecimento, como por exemplo, *World Wide Web*.

As quatro características desejáveis de um sistema de *Data Mining* são [Freitas 1997]:

- Descoberta de conhecimento compreensível, tipicamente expresso pelo alto nível das regras;
- Integração com banco de dados [Han 1996], [Imielinski 1996];
- Alto grau de autonomia necessária para descobrir conhecimento previamente desconhecido pelo usuário [Zytkow 1993];
- Eficiência para o processo de descobrimento, necessário para lidar com grandes banco de dados

Um outro detalhe importante, é o fato da ferramenta ser capaz de lidar com dados inválidos ou imprecisos. Dados inválidos dificultam a busca de conhecimento, em alguns casos chegam a impossibilitar a busca de uma solução.

O sistema GPSQL Miner apresenta uma nova ferramenta para classificação em um banco de dados relacional. Do ponto de vista comercial, ter uma ferramenta para

classificação ligada ao SGBD é uma vantagem pois geralmente as informações estão contidas num banco de dados relacional. Outra vantagem é diminuir a redundância de dados. De um outro ponto de vista, a estreita integração entre o algoritmo de *Data Mining* e o SGBD tem diversas vantagens. Em [Freitas 1997b] maiores detalhes podem ser encontrados.

- A primeira vantagem é o aproveitamento da escalabilidade. Se um sistema implementa algum algoritmo de *Data Mining* ligado diretamente ao banco de dados, essa união possibilita ao sistema o uso de grandes banco de dados.

- Re-utilização dos dados e minimização de dados redundantes. Manter os dados no SGBD durante todo o processo de *Data Mining* possibilita ao sistema utilizar novamente os dados e minimiza a redundância dos dados. A minimização da redundância é crucial em aplicações de *Data Mining* em larga escala [Brown 1995].

- Aproveitamento do controle de segurança e privacidade dos dados. Um sistema integrado pode aproveitar estes benefícios oferecidos pelo SGBD. Esta vantagem é importante pois as aplicações de *Data Mining*, por sua natureza, necessitam de segurança e privacidade dos dados [O'Leare 1995]

- Exploração automática de paralelismo em servidores SQL paralelos (*Parallel SQL Servers*). Uma vez que, as requisições ao banco de dados são feitas com comandos SQL, os servidores implementam automaticamente a paralelização de várias requisições sendo transparente para o algoritmo de *Data Mining*.

2.1 Classificação

A classificação é um dos objetivos do sistema GPSQL Miner. Como foi visto anteriormente, a tarefa de classificação formula um modelo baseado nos dados. O modelo pode ser usado para classificar um novo item em uma das classes pré-definidas baseando-se no valor dos atributos [Freitas 1997b]. A este modelo chamamos de classificador.

Algoritmos de Aprendizado de Máquina (AM) têm sido úteis para classificação em *Data Mining*. Porém algumas observações são relevantes: os algoritmos de AM tentam aprender utilizando pequenos volumes de dados; em contraste a esta perspectiva, *Data Mining* tem como objetivo trabalhar com grandes volumes de dados. Desta forma é comum realizar o processo de classificação em *Data Mining* considerando pelo menos duas etapas. Na primeira etapa, os classificadores são induzidos em diferentes conjuntos de dados utilizando AM. A segunda etapa combina estes classificadores numa abordagem integrada. Este trabalho enfoca a primeira etapa que é a indução de classificadores em pequenos volumes de dados.

Num sistema classificador, o objetivo é a descoberta de algum tipo de relacionamento entre os atributos utilizados para a predição e o atributo objetivo, de forma que o conhecimento descoberto pode ser usado para predizer a classe (um valor do atributo objetivo) para uma tupla nova ou não conhecida [Freitas 1998].

O classificador pode ser expresso como um conjunto de regras. As regras são usualmente utilizadas para expressar conhecimento e são facilmente compreendidas pelos humanos. Sua utilização é comum em sistemas especialistas de apoio à decisão [Wong 2000].

Como exemplo da tarefa de classificação, pode-se citar o problema de predizer quais os consumidores que comprariam o livro "A Guide to French Restaurants in England" [Freitas 1998]. Suponha que uma editora internacional tenha publicado o livro em inglês, francês e alemão; e que a empresa contenha em seu banco de dados, os dados de seu consumidores, informações como sexo, idade, país de origem e se o consumidor

comprou ou não o livro em questão. Cada linha de uma tabela, pode ser chamada de tupla ou registro da tabela, por exemplo, na Tabela 2.1 pode-se ver 10 tuplas, ou 10 registros, ou 10 registros de clientes diferentes.

O objetivo do problema é obter informações sobre o atributo 'Compra' que pode ser chamado de atributo objetivo, e os seus valores chamados de classes. As demais informações (Gênero, País, Idade) necessárias para predizer sobre possíveis consumidores são chamados de atributos de predição. A classificação consiste em determinar quais valores dos atributos de predição estão associados a cada um dos valores do atributo objetivo.

Este conhecimento descoberto, o classificador, pode ser usado para predizer se qualquer novo comprador adquirirá ou não o livro. As informações deste comprador não estão no banco de dados da companhia, consequentemente, o valor do atributo objetivo é desconhecido.

Como os valores possíveis para atributo objetivo são chamados de classe, neste exemplo existem 2 classes: 'Sim' e 'Não'.

TABELA 2.1: ENTRADA DE DADOS PARA TAREFA DE CLASSIFICAÇÃO [FREITAS 1998]

Cod	Gênero	País	Idade	Compra (objetivo)
1	Masculino	França	25	Sim
2	Masculino	Inglaterra	21	Sim
3	Feminino	França	23	Sim
4	Feminino	Inglaterra	34	Sim
5	Feminino	França	30	Não
6	Masculino	Alemanha	21	Não
7	Masculino	Alemanha	20	Não
8	Feminino	Alemanha	18	Não
9	Feminino	França	34	Não
10	Masculino	França	55	Não

As regras são freqüentemente representadas na forma de "*if-then*". Estas regras são interpretadas como: "***if*** (se) os atributos de predição (*predicting attributes*) da tupla satisfazem as condições do antecedente da regra, ***then*** (então) a tupla tem a classe indicada no conseqüente da regra." A partir dos dados da Tabela 2.1 o algoritmo de classificação poderia extrair as quatro regras mostradas na Figura 2.1. Os atributos de predição podem

ser chamados de parte antecedente da regra, ou seja, a parte entre o string 'if' e 'then'; o atributo objetivo e a classe são parte conseqüente da regra, depois do 'then'.

```
If ( País = "Alemanha") then (Compra = "Não")
If ( País = "Inglaterra") then (Compra = "Sim")
If ( País = "França" and Idade <= 25) then (Compra = "Sim")
If ( País = "França" and Idade > 25) then (Compra = "Não")
```

FIGURA 2.1: REGRAS DESCOBERTAS A PARTIR DA TABELA 2.1

Um atributo descritor pode ser descrito de diversas maneiras, portanto, existem diferentes formatos para os descritores. Um descritor pode determinar um valor para um atributo nominal, um conjunto de valores para um atributo contínuo, pode ser usado para comparar valores de atributos ou pode ser comparado com outro atributo de mesmo tipo. Em se tratando de banco de dados, um atributo é uma coluna de uma tabela.

A primeira regra da Figura 2.1: "If (País = "Alemanha") then (Compra = "Não")" pode ser interpretada como: "Se o 'País' do cliente for 'Alemanha' então o comprador 'Não' irá comprar". Em outras palavras, os compradores da Alemanha não são consumidores em potencial do livro em questão.

Existem dois critérios que são mais freqüentemente usados para avaliar a qualidade das regras de classificação encontradas. O primeiro critério é a capacidade de compreensão do conhecimento descoberto que é um assunto mais subjetivo [Freitas 1998]. O outro critério é a precisão de predição das regras descobertas que é medida pela taxa de erro na classificação.

Recentemente, a compreensibilidade das estruturas das árvores tem recebido alguma atenção [Lim 1999]. A compreensibilidade tipicamente diminui com o aumento do tamanho e da complexidade da árvore. Se duas árvores empregam o mesmo tipo de testes e têm a mesma taxa de precisão, a árvore que tiver menos folhas usualmente é preferida. Com relação à compreensibilidade, no trabalho presente será considerado apenas o tamanho

(número de condições) na escolha do melhor classificador, não sendo considerado nenhum método de simplificação da árvore para melhorar a sua compreensibilidade.

Uma maneira simples de medir a taxa de erro de classificação (Taxa de Erro) é dividir os dados disponíveis em dois subconjuntos exclusivos de registros, chamado de conjunto de treinamento e conjunto de teste [Freitas 1998].

O conjunto de treinamento é utilizado para descobrir a relação entre os atributos de predição e o atributo objetivo. Em outras palavras, o conjunto de treinamento equivale ao conhecimento anterior necessário para a criação do melhor classificador. O classificador é escolhido de acordo com alguns critérios, dentre os quais podemos citar a precisão. A precisão é a porcentagem de acerto de um classificador em uma determinada base. Para medir a precisão aparente basta dividir a quantidade de registros classificados corretamente pelo número total de registros da base.

Já o conjunto de teste é utilizado como se fosse um 'novo' conhecimento. Este conhecimento é utilizado apenas para verificar a porcentagem da taxa de erro de classificação das regras encontradas. Para calcular a taxa de erro basta dividir a quantidade de registros com erro de classificação pelo número total de registros da base de teste. Neste trabalho, a taxa de erro será utilizada apenas para avaliação do classificador na base de teste. Já a precisão também pode ser calculada na base de teste e seu significado é o oposto ou o complemento da taxa de erro, ou seja, a precisão pode ser igual a 100% menos a taxa de erro.

A necessidade de medir a taxa de erro na base de testes separadamente da base de treinamento pode ser mostrada por um simples argumento [Freitas 1998]. Suponha que não haja a separação dos conjuntos, e o sistema classificador descubra regras que possam classificar com 0% de erro para todos os registros conhecidos; é mais provável que este conjunto de regras ache uma alta taxa de erro na classificação de novos registros. Em outras palavras, este conjunto de regras não envolve generalização no todo, é um caso de especialização das regras em um conjunto específico de dados chamado de *overfitting*.

A medida da taxa de erro em um conjunto de dados de teste é um dos métodos mais simples e menos demorado para avaliar a precisão das regras descobertas [Freitas 1998]. Portanto, este método é o mais usado na caso de bases de dados enormes. Mais informações pode ser encontradas em [Weiss 1991].

2.2 Discussão do Capítulo

Neste capítulo foram vistos os conceitos gerais de *Data Mining*. Começando pela sua definição, seguida por uma breve descrição dos principais algoritmos existentes. Também foram citadas características desejáveis para um sistema de *Data Mining* e a definição da proposta de trabalho que é a classificação integrada ao SGBD.

A Seção 2.1 possui a definição da tarefa de classificação que será abordada pelo trabalho em questão. Foi citado um exemplo para melhor entendimento do que é classificação. Tem-se também o formato mais comum para representação de um classificador. E após, foi descrito um critério para avaliação da qualidade das regras encontradas.

No próximo capítulo são definidos os conceitos referentes à GGP.

3. Programação Genética orientada à Gramática (GGP)

Na vida, os seres que têm a melhor capacidade de se adaptarem às mudanças que ocorrem no meio ambiente são os que sobrevivem, segundo a teoria da Seleção Natural de Darwin [Darwin 1859]. A cada geração, esses indivíduos transmitem suas características genéticas para seus descendentes. Após muitas gerações, tem-se uma população de indivíduos selecionados naturalmente.

Baseado nestes conceitos, a Programação Genética (GP – *Genetic Programming*) foi criada com o objetivo de produzir software automaticamente. Foi introduzida por John Koza [Koza 1992], que se baseou na teoria de Darwin com a idéia dos Algoritmos Genéticos (AG) apresentada por John Holland [Holland 1992] [Mitchell 1996].

No lugar de uma população de seres vivos, em GP temos uma população de programas de computador, representados sob a forma de genes. Através da recombinação desses genes, o objetivo do algoritmo de GP é chegar ao programa de computador que melhor resolve um determinado problema. Para isso, começa-se com uma população inicial aleatória e, geração após geração, aplica-se os operadores genéticos para simular o processo evolutivo.

Assim, a GP também pode ser vista como uma técnica de busca, aplicada sobre o universo de todos os programas de computador que podem ser gerados em uma dada linguagem. O resultado desta busca é o programa que melhor resolve o problema proposto.

Como uma forma de definir melhor o espaço de busca é utilizada uma gramática. A gramática consiste na definição de como as árvores de derivação deverão ser construídas e como as operações para produção de novos indivíduos devem ser feitas. A gramática pode definir a linguagem em que os programas podem ser escritos, com isso, é possível a geração automática de programas.

A junção da GP com a gramática recebe o nome de Programação Genética orientada a Gramática (GGP). Whigham [Whigham 1995a] [Whigham 1995b] introduziu uma forma muito genérica de GGP. Ele utilizou *Context-Free Grammars* (CFG) como a estrutura da evolução para superar as necessidades de fechamento para GP.

Um CFG pode ser considerado uma tupla (S, Σ, N, P) aonde [Gustafson 1986]:

S – é o símbolo inicial (considerado não terminal);

N – conjunto de símbolos não terminais;

Σ - conjunto de símbolos terminais;

P – conjunto de regras de produção.

As produções estão na forma $x \rightarrow y$ aonde $x \in N$ e $y \in \{N \cup \Sigma\}^*$. Quando existe um número de produções que pode ser aplicado para um particular $x \in N$, a escolha é delimitada pelo símbolo disjuntivo '|'.

As produções especificam como os símbolos não terminais devem ser reescritos dentro de suas derivações até a expressão conter somente símbolos terminais. Por exemplo, um CFG para gerar uma expressão aritmética simples de uma variável pode ser:

```
N = {<exp>,<var>,<op>}
Σ = {x,+,-,*,/}
P = {S → <exp>
     <exp> → <var> | <exp> <op> <exp>
     <op>  →  + | - | * | /
     <var> →  x
    }
```

No exemplo acima, os não terminais são: <exp>, <var> e <op>. Já como terminais temos o string 'x' e os operadores aritméticos: +,-,* e /.

3.1 Árvores de Derivação

Uma árvore é a estrutura mais comum usada para representar programas em GP [Koza 1992]. Outras estruturas possíveis incluem grafos [Teller 1995], genomas lineares [Banzhaf 1998] e árvore de derivação baseada em gramática [Wong 2000] [Whigham 1996].

Árvores de derivação têm duas vantagens importantes sobre outras estruturas. Primeira, permite ao algoritmo ser livre de contexto, no sentido que representa uma abstração da linguagem de programação e o formato do programa. Segundo, é limitada ao efeito destrutivo dos operadores genéticos sobre os programas, não permitindo a criação de código sintaticamente incorreto.

Nesta aproximação, as árvores de derivação são construídas baseadas nas produções da *Context-Free Grammar* (CFG). As produções podem ser escritas em *Backus-Naur form* (BNF) como mostra a Figura 3.1.

```
<code>      ::= <exp>
<exp>       ::= <exp> <op>  <exp>
<exp>       ::= <var>
<op>        ::= + | - | * | /
<var>       ::= x
```

FIGURA 3.1: EXEMPLO BNF

Cada nodo da árvore de derivação pode conter um terminal (Ex.: variável "x" – 4ª. linha da Figura 3.1) ou um não terminal (Ex.: <exp>). Os não terminais constituem os nodos folhas das árvores, enquanto os outros nodos são compostos por terminais. Diferentemente de terminais, que tem um valor independente, os não terminais dependem da evolução dos seus componentes.

Para gerar um programa válido, a partir de uma gramática, primeiramente é selecionada a produção inicial. Por exemplo, se for considerada a gramática na Figura 3.1, todo programa é criado a partir do não terminal <code> da 1ª. linha. Um nodo filho é criado

com o não terminal ou terminal, à direita do não terminal inicial, no exemplo o primeiro nodo filho é <exp>. Se o nodo for um não terminal, uma produção do nodo deve ser escolhida aleatoriamente para escolha do nodo filho, no exemplo o <exp> tem duas produções que podem ser utilizadas para criação dos nodos filhos (linha dois e três da Figura 3.1). Esta operação de substituição deve continuar para cada um dos nodos filhos que sejam não terminais.

A Figura 3.2 mostra um exemplo de uma árvore de derivação criada para a expressão "x*x+x" a partir da gramática da Figura 3.1.

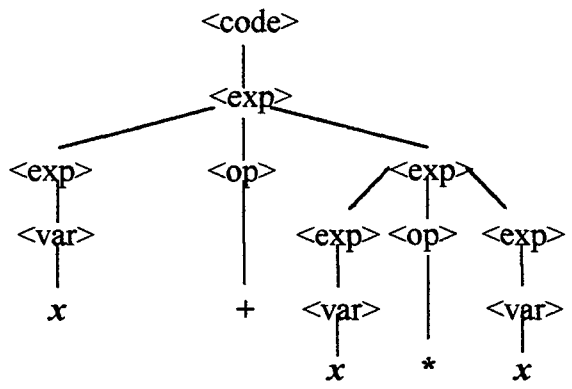

FIGURA 3.2: EXEMPLO ÁRVORE DE DERIVAÇÃO

3.2 Visão Geral do Algoritmo de GGP

O primeiro passo no processo evolutivo é gerar aleatoriamente uma população inicial de acordo com a gramática. Então, o algoritmo entra em um laço que é executado até uma solução ser encontrada. Este laço consiste de quatro ações principais:

- Avaliar cada programa através de uma função heurística especial (função de *fitness*, apresentada posteriormente neste capítulo), que expressa quão próximo cada programa está da solução ideal.

- Criar uma nova população selecionando os indivíduos de acordo com o valor da *fitness* e aplicando os operadores genéticos: reprodução, cruzamento e mutação [Koza 1992], sempre de acordo com a gramática especificada.
- A antiga população é destruída e a nova passa a fazer parte da geração seguinte.

Na natureza o processo evolutivo não tem fim. Porém, quando se trata de computadores, tempo e recursos são limitados. Portanto, é necessário estabelecer um critério de término que interromperá o processo. Um sistema de GGP pode, por exemplo, criar tantas gerações quantas forem necessárias até que uma solução satisfatória (indivíduo com melhor *fitness* seja encontrada, ou até que o número máximo de gerações tenha sido atingido).

3.3 Função de Aptidão (Fitness)

Na natureza, os seres vivos são selecionados naturalmente com base no seu grau de adaptabilidade ao meio ambiente. Em GP isto é expresso pela função de *fitness*. Os programas que melhor resolverem o problema receberão melhores valores de *fitness* e, consequentemente, terão maior chance de serem selecionados.

A escolha da função de *fitness*, assim como a escolha do método de avaliação utilizado por esta função, depende do problema [Banzhaf 1998]. Boas escolhas são essenciais para se obterem bons resultados, já que a função de *fitness* é a força-guia que orienta o algoritmo de GP [Gritz 1999].

3.4 Métodos de Seleção

Dada uma população em que a cada indivíduo foi atribuído um valor de *fitness*, existem vários métodos para selecionar os programas sobre os quais são aplicados os operadores genéticos. Os métodos de selecionamento mais comuns são:

- Método da Roleta – Neste método, cada indivíduo é representado na roleta por uma fatia proporcional ao seu *fitness*. Quanto maior o valor da *fitness* de um determinado programa, maior será a possibilidade de que ele seja selecionado. Para representar qual fatia da roleta será selecionada, é gerado um número aleatório entre zero e a soma da *fitness* de todos os indivíduos [Banzhaf 1998].
- Método do Torneio – 'N' indivíduos da população são escolhidos aleatoriamente para formar uma sub-população temporária. Deste grupo, o programa selecionado será aquele que possuir o maior valor de *fitness*. Este método é o mais utilizado pois oferece a vantagem de não exigir que a comparação seja feita entre todos os indivíduos da população [Banzhaf 1998].

3.5 Principais Operadores Genéticos

Uma vez que os indivíduos tenham sido selecionados, deve-se aplicar um dos operadores genéticos. Os três principais são: Reprodução, Cruzamento e Mutação.

Na Reprodução, um indivíduo é replicado para a próxima geração sem sofrer nenhuma mudança em sua estrutura. Equivale à reprodução assexuada dos seres vivos.

No Cruzamento, chamado de *crossover*, dois programas são recombinados para gerar dois programas filhos potencialmente diferentes. Um ponto aleatório de cruzamento é escolhido em cada indivíduo pai e os trechos de programa abaixo deste ponto são trocados. É equivalente à reprodução sexuada.

Em GGP, os operadores devem produzir descendentes válidos de acordo com CFG. Em geral, para *crossover* utiliza-se *point-typing* definido por Koza [Koza 1994] para preservar a forma da árvore. Um par de programas é selecionado a partir da população corrente baseado na estratégia de seleção. Um ponto de *crossover* com o não terminal 'A' é então determinado no primeiro classificador. Se o segundo programa não tem o nodo com 'A', o *crossover* é rejeitado. Caso contrário, um nodo com 'A' é aleatoriamente selecionado na segunda árvore e as sub-árvores são trocadas.

Na operação de mutação, somente uma árvore é selecionada e um ponto de mutação é determinado. Uma nova sub-árvore é criada usando o mesmo processo para a geração da população inicial. Esta nova sub-árvore é trocada pela sub-árvore do ponto de mutação. Teoricamente, esta operação ajuda a manter a diversidade na população, que é importante para evitar a convergência precoce do algoritmo para uma única solução. Entretanto, existem evidências de que a mutação pode ser ignorada [Koza 1992].

Em ambos os operadores, *crossover* e mutação, a profundidade máxima é utilizada para indicar a mais profunda árvore de derivação que pode ser gerada.

Outros operadores podem ser definidos de acordo com as necessidades de cada situação, como por exemplo, a inversão (permutação), a edição e o encapsulamento [Koza 1992].

3.6 Critério de Término

O critério de término é responsável por interromper o laço de repetição do processo evolutivo que, idealmente, não teria fim. Um dos critérios de término é o número máximo de gerações evitando que o algoritmo continue infinitamente sem encontrar nenhuma solução.

É comum estabelecer casos para avaliação de *fitness* (casos de treinamento) através da determinação de valores de entrada e respectivas saídas esperadas. Por isso, é importante estabelecer uma margem de erro (*threshold*) para fazer a comparação entre o valor de saída esperado e o que foi obtido pelo programa. Como pode ser muito custoso ou mesmo impossível chegar a uma solução "perfeita" em tempo finito, esse critério permite que o sistema aceite resultados que estejam dentro da margem de erro.

Após o término da execução, o sistema completou um *run*, ou seja, uma rodada para um conjunto de parâmetros.

3.7 Principais Parâmetros

Os parâmetros controlam o funcionamento do algoritmo, permitindo um ajuste fino por parte do usuário. Os principais parâmetros utilizados por um algoritmo de GGP são:

- Número máximo de gerações – Se for atingido, a evolução é interrompida para evitar que a repetição repita infinitamente.
- Tamanho da população – Número de programas em cada população.
- Tamanho do Torneio – Número de programas selecionados aleatoriamente para constituir a sub-população que é utilizada pelo método de seleção de Torneio (descrito anteriormente).
- Método de inicialização – Especifica como a primeira geração de programas é criada. As opções mais freqüentemente utilizadas são o método *full*, onde todos os programas terão o tamanho máximo possível, e o método *grow*, que permite uma população inicial mais diversa com programas de vários tamanhos.
- Profundidade mínima da árvore – É o menor tamanho de programa permitido. Este parâmetro é usado pelo método de inicialização.
- Profundidade máxima da árvore – Também é utilizado durante a criação da população inicial.

- Profundidade máxima da árvore após o cruzamento – Com a aplicação o operador de cruzamento, os programas tendem a ficar cada vez maiores, o que consumiria mais recursos do sistema do que o previsto. Este parâmetro fixa um limite. Caso o tamanho do(s) programa(s)-filho(s) seja(m) maior(es) do que este valor, o cruzamento não é efetivado e o(s) programa(s)-pai(s) é(são) reproduzido(s).
- Taxa de cruzamento – Probabilidade (valor entre 0 e 1) de se escolher o operador de cruzamento.
- Taxa de mutação – Probabilidade (valor entre 0 e 1) de se escolher o operador de mutação.
- Taxa de reprodução – Probabilidade (valor entre 0 e 1) de se escolher o operador de reprodução.
- Limiar ou Margem de erro - Valor de precisão utilizada quando são comparados os valores numéricos dos casos de teste (*fitness cases*).

3.8 Discussão do Capítulo

Neste capítulo foram apresentados os principais conceitos da Programação Genética orientada à Gramática. Primeiramente, foram explicados os conceitos em que se baseia a GP, seus objetivos e uma breve apresentação da GGP. Depois foram vistos os conceitos relativos à gramática.

Na Seção 3.1, foi apresentado como os indivíduos são armazenados pela GGP, seguido de um exemplo de como uma árvore de derivação pode ser criada a partir da gramática.

Na Seção seguinte, foram indicados os passos principais que um algoritmo de GGP deve executar. Na outras seções foram apresentados alguns conceitos principais como: *fitness*, Métodos de Seleção, Principais Operadores Genéticos.

Foram indicados alguns critérios para o sistema parar a busca do melhor classificador. E, finalmente, foram apresentados os principais parâmetros necessários para a GGP. O Capítulo 5 apresenta como todos estes conceitos são utilizados no sistema GPSQL Miner.

4. LOGENPRO

Data Mining e GGP são áreas de pesquisa relativamente novas. Apenas poucos trabalhos foram desenvolvidos com as duas áreas e, até o presente momento, não foi encontrado nenhum sistema que utilize *Data Mining* e GGP com acesso direto a um banco de dados relacional. Dentre os trabalhos existente LOGENPRO [Wong 2000] é o sistema que mais se assemelha à nossa proposta.

LOGENPRO (The LOgic grammar based GENetic PROgramming system) é um sistema de *Data Mining* desenvolvido para combinar o poder de procura paralela implícita da GP e o poder de representação da lógica de primeira ordem de ILP (*Inductive Logic Programming*) [Wong 2000].

A estrutura do sistema é baseado no formalismo das gramáticas lógicas [Wong 2000]. O formalismo é poderoso o suficiente para representar informações sensíveis ao contexto e conhecimento dependente de domínio. Este conhecimento pode acelerar a velocidade de aprendizagem e/ou melhorar a qualidade do conhecimento induzido pelo sistema LOGENPRO.

Quanto à tarefa de classificação, as principais características do LOGENPRO são [Wong 2000]:

1) O sistema usa uma gramática ajustada para especificar a estrutura da regra permitindo uma grande flexibilidade no formato da mesma. Cada regra é um indivíduo representado por árvores de derivação e tem a seguinte forma: "**if** antecedente **then** conseqüente". De modo genérico, o antecedente é um conjunto de atributos descritores e o conseqüente é o atributo objetivo com a classe. O formato das regras em cada problema pode ser diferente. Desta forma, para cada problema uma gramática específica é escrita

pelo usuário de tal forma que o formato das regras seja o melhor para cada domínio. A próxima seção define com mais detalhes a gramática do LOGENPRO.

2) Operador genético '*tailor-made*' criado especialmente para a tarefa de aprendizado de regras. Este operador seleciona automaticamente um atributo descritor e muda para um nodo chamado *any*, o que equivale a retirar o atributo da regra. Por exemplo, a regra:

If sepal_length=0.5 and sepal_width between (1, 1.50) and **petal_length** = **50** then class = Iris-setosa

Depois de sofrer ação do operador '*tailor-made*' pode ser modificador por:

If sepal_length=0.5 and sepal_width between (1, 1.50) and **any** then class = Iris-setosa.

3) Criação de uma função específica para avaliação de cada regra. A função de *fitness* do LOGENPRO é baseado na estrutura *support-confidence framework* [Agrawal 1993]. A função considera suporte (medida de cobertura da regra) e fator de confidência (é a confidência do conseqüente que é verdadeira sobre os antecedentes).

4.1 Gramática do LOGENPRO

No LOGENPRO, a gramática orienta como a estrutura deve ser evoluída e especifica a estrutura da regra, que tem a forma "**if** antecedente **then** consequente". Como o formato da regra muda de acordo com o problema, o sistema usa conhecimento de domínio, na qual, um formato para cada regra é escrita de tal forma que melhor se adapte ao domínio. Entretanto, de maneira geral, a parte antecedente é um conjunto de atributos descritores e o conseqüente é formado pelo atributo objetivo [Wong 2000].

A gramática existente permite uma representação de conhecimento muito flexível para a representação do formato da regra. LOGENPRO permite muitas variações na descrição dos atributos e regras com diferentes formatos.

Para ilustrar o uso da gramática para representar um formato de regra adequado, considere uma base de exemplo com 4 atributos [Wong 2000]. O objetivo é aprender regras sobre o atributo: attr4, que é booleano. O atributo attr1 é nominal e seu conteúdo com 0, 1 ou 2. O atributo attr2 é contínuo entre 0-2000 e pode ser categorizado em 'high', 'medium' ou 'low'. O domínio de attr3 é idêntico ao attr2 e, por isso, é possível para a regra compará-los.

Um exemplo para esta base é dado na Tabela 4.1. Os símbolos erc1, erc2, erc3, boolean_erc, e category_erc na gramática são constantes aleatórias efêmeras (ERCs). Cada ERC tem seu alcance para instanciação. Erc1 é um do conjunto {0, 1, 2}, erc2 e erc3 esta entre 0-200, boolean_erc pode ser 'T' ou 'F', category_erc pode ser 'high', 'medium' ou 'low'.

TABELA 4.1: UM EXEMPLO PARA GRAMÁTICA DO LOGENPRO [WONG 2000]

Linha			
1	start	->	[if], antes, [, then], consq, [.].
2	antes	->	attr1, [and], attr2, [and], attr3.
3	attr1	->	[any]
4	attr1	->	attr1_descriptor.
5	attr2	->	[any]
6	attr2	->	attr2_descriptor.
7	attr3	->	[any]
8	attr3	->	attr3_descriptor.
9	attr1_descriptor	->	[attr1 =], erc1.
10	attr2_descriptor	->	[attr2 is], category_erc.
11	attr2_descriptor	->	[attr2 between], erc2, erc3.
12	attr3_descriptor	->	[attr3], comparator, attr3_term.
13	comparator	->	[=].
14	comparator	->	[!=].
15	comparator	->	[<=].
16	comparator	->	[>=].
17	comparator	->	[<].
18	comparator	->	[>].
19	attr3_term	->	attr2.
20	attr3_term	->	erc3
21	consq	->	attr4_descriptor.
22	attr4_descriptor	->	[attr4 =], boolean_erc.
23	erc1	->	{member(?a, [0, 1, 2])}, [?a].
24	erc2	->	{random(0, 200, ?a)}, [?a].
25	erc3	->	{random(0, 200, ?a)}, [?a].
26	category_erc	->	{member(?a, [high, medium, low])}, [?a].
27	boolean_erc	->	{member(?a, [t, f])}, [?a] }.

Na Tabela 4.1, o símbolo '*any*' serve como 'não importa' na regra. Em outras palavras, um atributo não será considerado na regra se o atributo descrito for '*any*'. Na gramática, cada atributo pode ser descrito pelo descritor na regra, ou pelo '*any*' sendo este ignorado na regra. O atributo 'attr1' tem somente uma forma (linha 9). O atributo 'attr2' pode ter duas formas de descritores. Pode ser descrito por um intervalo (linha 11) ou pela categoria que ele pertence (linha 10). O atributo 'attr3' pode ser especificado por um comparador (linha 12), especificando se o descritor pode ser comparado com 'attr2' ou a uma constante. Esta gramática produz regras do tipo apresentado na Figura 4.1.

```
If attr1 = 0 and attr2 between 50 180 and any, then attr4 = T
If attr1 = 2 and attr2 is high and attr3 != 50, then attr4 = T
If attr1 = 1 and any and attr3 >= attr2, then attr4 = F
```

FIGURA 4.1: EXEMPLO DE REGRAS DO LOGENPRO

4.2 Diferenças entre os sistemas

O propósito deste trabalho é um pouco diferente do LOGENPRO. A seguir encontram-se algumas diferenças entre as duas abordagens:

1) O sistema GPSQL Miner não usa as medidas de confidência e suporte para cálculo do fitness, o fitness é calculado com base na precisão do classificador;

2) As regras do LOGENPRO estão no formato *if-then*; enquanto o GPSQL Miner propõe regras num formato SQL;

3) LOGENPRO representa cada regra como um indivíduo, enquanto o GPSQL Miner representa um indivíduo como um classificador completo (conjunto de regras concatenadas) para todas as classes.

4) A gramática do LOGENPRO produz regras com número fixo de atributos necessitando do operador *tailor-made*. Já o GPSQL Miner produz uma gramática para a

geração de classificadores formado por regras com um, dois, ..., N atributos, onde N é o número total de atributos preditores.

5) O sistema GPSQL Miner cria automaticamente a gramática de acordo com as informações contidas no banco de dados, a partir da indicação inicial de quais atributos serão utilizados. A gramática é criada de modo a ser otimizada para cada problema uma vez que a criação é baseada nos dados contidos no banco de dados, entretanto, o usuário pode modificar o arquivo de gramática com ajuda de um assistente para modificação da gramática. Para assistir a modificação da gramática, um arquivo de análise é criado automaticamente, o que facilita a construção adequada da gramática para cada problema. Diferentemente do sistema LOGENPRO que não provê ferramentas de auxílio para criação da gramática.

4.3 Discussão do Capítulo

Neste capítulo foi apresentado o LOGENPRO, um sistema similar ao trabalho que esta sendo proposto. Uma breve descrição do sistema LOGENPRO e suas características para aprendizado de regras foram apresentados.

Uma breve definição da gramática utilizada pelo LOGENPRO foi apresentada juntamente com um exemplo e o formato da regra gerada pelo sistema. Sendo citadas depois disto, diferenças entre o LOGENPRO e o sistema GPSQL Miner.

No próximo capítulo, será definido o sistema proposto para a tarefa de *Data Mining*: GPSQL Miner.

5. GPSQL Miner

O nome GPSQL Miner é a conseqüência de suas características principais: GP indica o uso do paradigma GGP; SQL enfatiza o uso de comandos SQL para representar o classificador; e Miner enfatiza a aplicação central do sistema..

Devido à necessidade de ferramentas que minerem grande bases de dados, este sistema visa ajudar nas decisões estratégicas, auxiliando na busca de informações novas e potencialmente úteis. Para atingir este objetivo, GPSQL Miner tem o objetivo de ser uma ferramenta genérica para *Data Mining* em banco de dados relacionais. É através da tarefa de classificação juntamente com SGBD que o sistema atinge este objetivo.

Um dos outros objetivos do sistema, é fornecer o máximo de automação para que o usuário aproveite o seu tempo para validação e análise das informações encontradas. Apenas com algumas informações o sistema começa a tarefa de classificação, como a indicação do atributo objetivo e dos atributos descritores. Para isso, o sistema utiliza-se do SGBD para pesquisar informações sobre os atributos e criar automaticamente uma gramática adequada para cada base de dados.

A partir desta gramática e do paradigma GGP é que o sistema começa a tarefa de classificação. Cada classificador é representado como um indivíduo cujo formato é definido pela gramática. É através da aplicação de operadores genéticos, utilizados conforme a gramática, que o sistema cria gerações de populações de indivíduos na busca do melhor classificador.

Neste capítulo, especificamente na próximo Seção é descrito um formato diferente para o classificador, a divisão do sistema em módulos, descrição detalhada de cada módulo e outras funcionalidades.

5.1 Classificador SQL

Este trabalho propõe um novo formato para os classificadores. Pelo fato das informações utilizadas para a mineração estarem em um banco de dados relacional o novo formato de cada regra do classificador lembra o comando SQL: SELECT.

A Figura 5.1 mostra o formato geral de uma regra. O caracter 'S' é uma abreviação da palavra SELECT, enquanto o 'W' é abreviação de WHERE. Entretanto, não é necessário ter a parte FROM do comando em cada regra, o que permite ao sistema criar um indivíduo menor.

```
S <valor_goal> W <atributos de predição com operadores>
```

FIGURA 5.1: FORMATO GERAL DAS REGRAS

Um conjunto de regras concatenadas forma o classificador completo. Na Figura 5.2 podemos ver o formato geral para o classificador. A primeira, segunda e a quarta linhas descrevem três regras. Já na terceira linha, as reticências '...' indicam que o classificador pode ter várias regras concatenadas.

```
S <valor_goal> W <predicting attributes>
S <valor_goal> W <predicting attributes>
...
S <valor_goal> W <predicting attributes>
S <valor_goal>
```

FIGURA 5.2: FORMATO GERAL PARA O CLASSIFICADOR

Na Figura 5.3 encontra-se um classificador com as quatro regras descritas pelo formato da Figura 5.1. A primeira regra de 5.3, que é "S 'Não' W País = 'Alemanha'", pode ser interpretada como: "classifica-se o atributo objetivo como 'Não' para todas as tuplas em que o 'País' é 'Alemanha'". Lembrando que o atributo objetivo é a coluna 'Compra' informando se o cliente comprou ou não o livro.

```
S    'Não'   W    País = 'Alemanha'
S    'Sim'   W    País = 'Inglaterra'
S    'Sim'   W    País = 'França' and Idade <= 25
S    'Não'   W    País = 'França' and Idade > 25
```

FIGURA 5.3: EXEMPLO DE CLASSIFICADORES EM FORMATO SQL

Existem os classificadores com regras ordenadas e regras não ordenadas. Neste trabalho os classificadores são formados por regras ordenadas, isto é, a ordem de cada regra dentro de um classificador é importante. Os registros classificados pelas regras anteriores não estarão disponíveis para a classificação das demais regras. Por exemplo, a primeira regra seleciona todos os registros que tiverem a coluna 'País' como 'Alemanha'. Para a avaliação da segunda regra, os registros classificados pela primeira regra não estão disponíveis, ou seja, a segunda regra apenas seleciona os registros de consumidores que não moram na 'Alemanha'. Em outras palavras, a presença de duas regras concatenadas funciona de modo semelhante a um comando 'if-then' concatenado com um comando 'else if-then'.

Como a ordem de cada regra é importante, um classificador pode ter uma última regra como sendo uma regra *default*. Na segunda linha da Figura 5.4 tem-se o exemplo de uma regra *default* dentro do classificador. Esta regra contém apenas o string 'S', o valor de uma classe (no exemplo 'Sim') e não possui o string 'W'. Esta regra classificaria todos os registros ainda não selecionados para a classe informada.

```
S    'Não'   W    País = 'Alemanha'
S    'Sim'
```

FIGURA 5.4: EXEMPLO REGRA COM DEFAULT

Uma conseqüência natural da ordem do classificador é que, cada registro é classificado em uma única classe. E, se for utilizado a regra *default*, todos os registros serão classificados. Neste trabalho, todos os classificadores terão uma regra *default*.

5.2 Módulos Principais

O nome GPSQL Miner é devido às suas principais características: *GP* indica que usa o paradigma Programação Genética orientada à gramática; *SQL* aponta que o sistema usa o formato de um comando SQL para representar o classificador, além de acessar um banco de dados relacional; *Miner* lembra que a aplicação central do sistema é *Data Mining*.

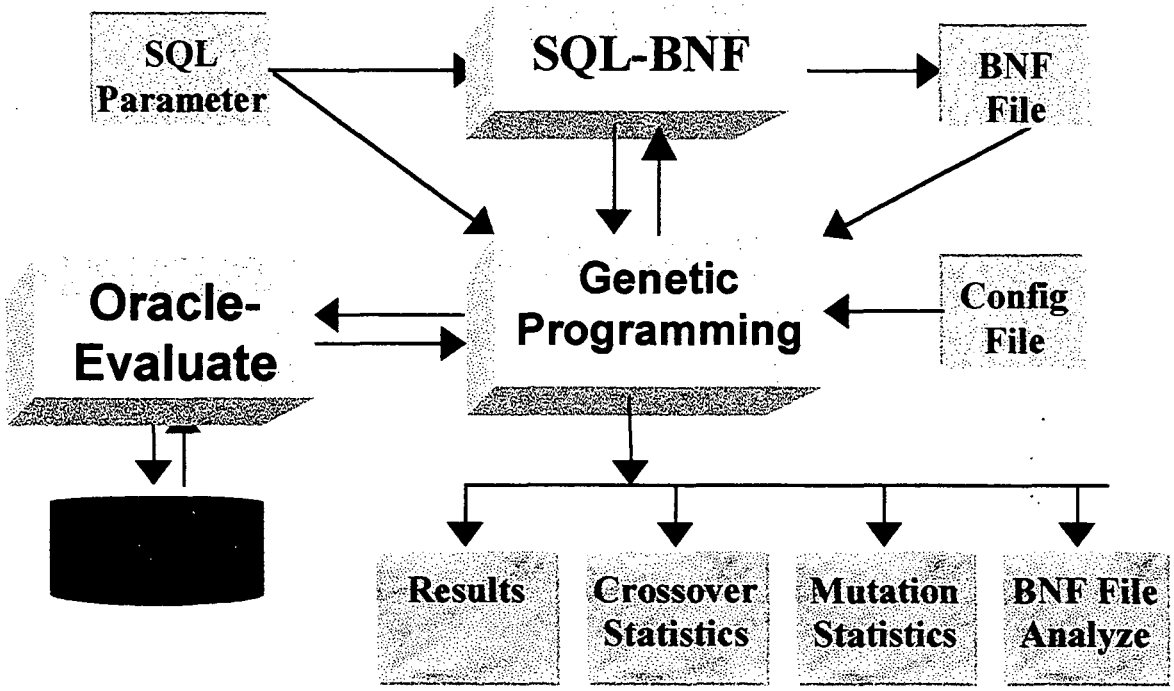

FIGURA 5.5: SISTEMA GPSQL MINER

Para alcançar os objetivos mencionados acima, GPSQL Miner possui três módulos centrais: SQL-BNF, *Grammar Genetic Programming* (GGP) e Oracle-Evaluate, conforme se pode observar na Figura 5.5.

- SQL-BNF: A ferramenta inicialmente lê o arquivo de configuração SQL Parameter (*.par). Este processo é essencial porque define o objetivo da tarefa de classificação e o conhecimento de entrada. Neste arquivo são declarados as tabelas e atributos que serão utilizados para a busca. Usando o

arquivo SQL Parameter o sistema lê informações de cada coluna no banco de dados criando o arquivo BNF com a gramática específica a ser utilizada para o problema em questão.

- GGP: Após ler o arquivo dos principais parâmetros do módulo GGP (*.gpm) e o arquivo BNF, o módulo cria a população inicial com N indivíduos (cada indivíduo é um classificador). Logo após, inicia-se um ciclo onde GGP usa os operadores genéticos para criar uma nova população. A cada iteração o módulo avalia cada indivíduo e seleciona o indivíduo com melhor *fitness*. Se o melhor indivíduo é a solução, o ciclo termina e a solução é selecionada. Caso contrário, o sistema aplica novamente os operadores para criação de uma nova população até a solução ser encontrada ou um outro critério de parada ser alcançado.

- Oracle-Evaluation: Este módulo é responsável pela avaliação do *fitness* e outros acessos ao banco de dados. O *fitness* é baseado na precisão do classificador, que é a porcentagem do conjunto de teste classificado corretamente. A avaliação de *fitness* é um processo simples, uma vez que o indivíduo é baseado em comandos SQL.

A seguir os módulos serão descritos em maiores detalhes.

5.3 Módulo SQL BNF

Este módulo é responsável pela leitura das informações que estão num arquivo SQL Parameter e que serão utilizadas para a descoberta do conhecimento; e, também, é responsável pela criação de um arquivo BNF de acordo com as informações contidas no banco de dados. A seguir descrevem-se detalhes deste processo.

SQL Parameter

GPSQL Miner inicialmente lê o arquivo SQL Parameter que define o objetivo da tarefa de classificação e a linguagem de entrada. Esta linguagem de entrada permite ao usuário declarar quais tabelas e colunas serão usadas para a construção das regras. Deste modo, a linguagem de entrada atua como restrição para o processo de busca.

As informações do arquivo SQL Parameter permite ao sistema otimizar os comandos SQL para avaliação de cada classificador. Um arquivo SQL Parameter possui 3 seções obrigatórias (Figura 5.6): 'Tables' (linha 1), 'Goal' (atributo objetivo - Linha 6), e 'Columns' (Linha 10) e uma que é opcional: 'TrainTestTables' (Linha 3).

```
1     //Tables
2     Alias -> Nome_Tabela
3     //TRAINTESTTABLES
4     Train  -> Nome_Tabela_Test_Run_X
5     Test -> Nome_Tabela_Test_Run_X
6     //Goal
7     Column -> Nome_Coluna_Goal
8     Table  -> Alias_Coluna_Goal
9     Type   -> Tipo_da_coluna_Goal
10    //Columns
11    Column -> Alias_Tabela.Nome_Coluna_1
12    Table  -> Alias_Adicionais_Where_1
13    where  -> {Joins_adicionais}
```

FIGURA 5.6: FORMATO SQL PARAMETER

A seção 'Tables' contém o nome de todas as tabelas a serem utilizadas. O nome de cada tabela esta à direita e o *alias* (apelido da tabela) à esquerda. *Alias* é o "apelido" dado a uma tabela dentro de um comando SQL como forma de otimização na hora de manipulação de suas colunas. As colunas de cada tabela são referenciadas com o *alias* da tabela como forma de garantir que o sistema possa manipular colunas de mesmo nome de tabelas diferentes. Uma outra vantagem da utilização do *alias* é que uma mesma tabela pode ser referenciada duas vezes em um mesmo comando SQL. Não existe limite para o número de tabelas, a única restrição é que o *alias* deve ser único.

A seção 'Goal', linhas de seis a nove da Figura 5.6, contém dados do único atributo objetivo que um problema pode ter. São indicados o nome do atributo, o *alias* de sua tabela e seu tipo que pode ser opcional. Nota-se que os valores das classes são omitidos pois os mesmos serão pesquisados pelo sistema no banco de dados. O tipo do atributo objetivo pode ser numérico ou caracter. A tabela da coluna objetivo, será chamada de tabela principal para o problema.

A última seção 'Columns' declara todas as colunas utilizadas que são os chamados atributos de predição. Para cada coluna, algumas informações são necessárias: nome, apelido das tabelas e os *joins adicionais* informados na cláusula 'where'. A cláusula 'where' indica o *join* (colunas para o relacionamento) entre a tabela do atributo objetivo com a tabela da coluna em questão. A cláusula 'where' é opcional e pode ser omitida se a coluna em questão estiver na tabela principal, e por outro lado, pode conter informações a serem adicionadas na cláusula 'where'. Esta informação é necessária para otimizar os comandos SQL usados para avaliar o classificador, e será acrescentada toda vez que a coluna for utilizada.

A coluna deve ser informada com referência à tabela que ela pertence, ou seja, com o *alias* a esquerda da coluna separados por um ponto. A informação da coluna com seu *alias* é importante pois permite que o classificador não necessite da cláusula 'from' e o sistema possa otimizar o comando SQL na avaliação do classificador.

Podem-se informar as tabelas de treinamento e de teste da tabela principal para cada *run* na seção 'TrainTestTables'. Caso as tabelas não sejam informadas, o sistema separa a tabela principal original nas tabelas de treinamento e de teste. A separação é feita de acordo com o parâmetro que indica o percentual da tabela de treinamento. Por exemplo, se for informado 90% para a tabela de treinamento, 90% dos registros da tabela principal original são separados para a tabela de treinamento, enquanto 10% são para a tabela de teste.

Arquivo BNF

Depois de ler o arquivo SQL Parameter e acessar o banco de dados, o sistema produz o arquivo BNF contendo a definição da gramática para o problema em questão. A existência da gramática juntamente com o acesso ao banco de dados permitem ao sistema GPSQL Miner automatizar o processo. O arquivo BNF é responsável pela limitação do espaço de busca, formato dos classificadores e através deste arquivo o sistema restringe as operações genéticas.

Para criar o arquivo BNF, o sistema pesquisa: as classes do atributo objetivo, os tipos e os valores mais freqüentes de cada coluna (se não for informado). Ou seja, com a informação da coluna objetivo o sistema pesquisa o seu tipo (numérico ou caracter) e os valores (classes) contidos na base de dados para a coluna. Para cada atributo preditor, o sistema pesquisa o tipo do atributo e os seus valores registrados.

Todos os arquivos BNF criados têm um formato geral conforme a Figura 5.7. Os números à esquerda apenas indicam o número de cada linha não existindo no arquivo original.

```
1 #productions
2 S       -> <classifier>
3 <classifier> -> <r> ... <r> S <goal>
4 <r>     -> S <goal> W <cond>
5 <r>     -> S <goal> W <cond> AND <cond>
6 ...
7 <goal>   -> goal_val1 | goal_val2 | ... | goal_valG
8 <cond> -> column1 <rel_column1>
9 <rel_column1> -> <Noperator> <value_column1>
10 <value_column1> -> column1_val1 |... | column1_valn
11 <value_column1> -> random ( MIN, MAX)
12 <cond> -> column2 <Coperator> <value_column2>
13 <value_column2> -> column2_val1 |... | column2_valn
14 <cond> -> column3 = <value_column3>
15 <value_column3> -> column3_val1 | column3_val2
16 <Noperator> -> = | != | > | < | >= | <=
17 <Coperator> -> = | !=
```

FIGURA 5.7: FORMATO GENÉRICO DO ARQUIVO BNF

Como uma forma de convenção, todos os não terminais estão entre os sinas < e >, por exemplo, na Figura 5.7 podemos citar: <classifier>, <cond>. Existe pelo menos uma linha para cada não terminal. Cada linha é uma produção e o não terminal fica à esquerda do sinal '->', e o restante da linha pertencerá aos nodos filhos. Os nodos filhos são criados de modo que cada nodo contenha um não terminal ou uma string formando um terminal.

Na continuação, descreve-se sucintamente cada linha do arquivo BNF genérico (Figura 5.7).

A primeira linha do arquivo apenas indica o cabeçalho do arquivo. A segunda linha mostra o nodo inicial (*root*), todo classificador é criado começando pelo não terminal <classifier>.

A terceira linha indica um classificador composto por várias regras. Abaixo do não terminal <classifier> serão gerados vários nodos filhos a partir da string "<r> ... <r> S <goal>", aonde <r> é um não terminal que indica uma regra; os três pontinhos '...' indicam que podem existir vários nodos, cada um contendo um não terminal <r>. O sistema gera CL+CO regras, aonde CL é o número total de classes e CO é o número de colunas. Aconselha-se alterar o número de regras de acordo com o número de regras necessárias para cada problema, pois o número CL+CO não tem nenhum rigor científico. No final do classificador existe uma regra *default*: "S <goal>".

O formato das diferentes regras é apresentado nas linhas 4 a 6. Na 4ª linha, tem-se uma regra com um atributo de predição (que será gerado a partir do não terminal <cond>); a quinta linha ilustra uma regra com 2 atributos de predição. O sistema cria N linhas para regras contendo de 1 a N colunas, onde N é o número de atributos de predição. Todas as regras começam com o não terminal <r>. O sistema não permite a criação de uma regra em que uma mesma coluna seja utilizada mais de uma vez. Com isso, na regra que contém N atributos de predição, todas as colunas estarão indicadas uma única vez.

A sétima linha é gerada com todos os valores possíveis (classes) da coluna <goal>. Estes valores são pesquisados automaticamente pelo sistema

Pelo menos três linhas são criadas para cada atributo numérico, como mostra as linhas de 8 a 10. O sistema cria uma linha (10) com todos os valores mais freqüentes para a coluna, e uma linha (11) com a função *random* com os argumentos MIN e MAX; na qual MIN é o valor mínimo encontrado para a coluna e MAX é o maior valor, ambos pesquisados utilizando o SGBD. Na criação do indivíduo, a função *random*(min, max) é substituída por um valor entre a MIN e MAX gerado aleatoriamente pelo sistema.

Para cada atributo do tipo char, são criadas duas linhas (12 e 13). As colunas do tipo caracter podem utilizar dois operadores: =; !=, que foram gerados pelo não terminal <Coperator>. Apenas estes operadores são utilizados pois não existe a necessidade de comparações como: "Coluna_A > 'Casa'".

Para cada coluna booleana ou que possua dois valores possíveis, podendo ser do tipo numérico ou char, o sistema gera duas linhas (14 e 15) com apenas um operador '=' deixado fixo.

Finalmente a linha 16 indica os operadores permitidos para colunas numéricas. A linha 17 indica os operadores para colunas do tipo char.

Para colunas que possuem o mesmo tipo e os mesmos valores, o sistema permite que estas colunas sejam comparadas entre si. Uma outra observação relevante é que o sistema cria um arquivo texto para cada gramática podendo ser alterada.

O GPSQL Miner baseado no arquivo BNF cria um classificador completo como mostra a Figura 5.8.

```
S 'Desktop' W P.IDADE < 23 AND RES.PAIS = 'Alemanha'
S 'Laptop' W COM.PAIS != 'Portugal' AND P.IDADE < 34 AND P.GENERO = 'M' AND RES.PAIS != 'Franca'
S 'Desktop' W COM.PAIS = RES.PAIS AND P.IDADE between 21 and 23
S 'Desktop' W RES.PAIS != 'Itália'
S 'Desktop' W P.GENERO = 'F'
S 'Laptop' W P.IDADE <= 18
S 'Desktop'
```

FIGURA 5.8: CLASSIFICADOR PARA COMPUTADOR.BNF

O classificador (Figure 5.8) tem 6 regras, cada uma formada por dois componentes: 'S' e 'W'. Após o caracter 'S' segue o valor do objetivo para a regra. (Ex.: 'Desktop' para a primeira regra e 'Laptop' para a segunda). Após o componente 'W' são indicados os atributos de predição (Ex.: "COM.PAIS = RES.PAIS AND P.IDADE between 21 and 23" para a 3ª regra).

A primeira regra do classificador possui dois atributos de predição ou duas colunas: 'P.IDADE' e 'RES.PAIS'. É importante lembrar que todas as colunas têm o *alias* de sua tabela especificado junto com a coluna no arquivo SQL Parameter.

A última regra: "S 'Desktop'" é uma regra *default*, ou seja, uma vez que não exista nenhuma restrição na regra, os dados que não foram selecionados nas regras anteriores serão selecionados e classificados para a classe "Desktop".

5.4 Módulo GGP

O módulo GGP é responsável por achar os melhores classificadores para cada problema. Primeiramente, o sistema lê o arquivo de configurações para inicialização da população e começa o processo evolutivo.

O processo evolutivo consiste num laço através do qual o sistema cria novas gerações mediante as operações genéticas. Após um critério de término, o indivíduo com menor *fitness* é selecionado como candidato à solução do problema. Cada geração é uma população de indivíduos, e cada indivíduo é um classificador completo definido em forma de árvore de derivação.

Internamente o sistema representa cada indivíduo como uma árvore de derivação, conforme ilustra a Figura 5.9.

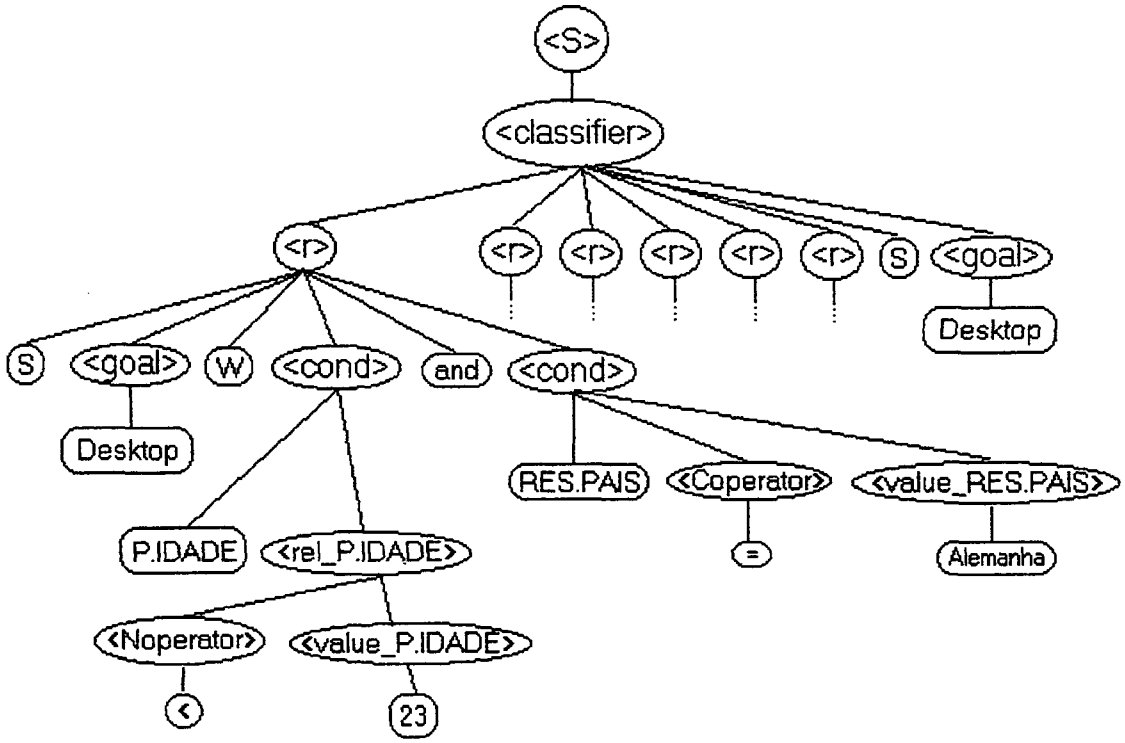

FIGURA 5.9: EXEMPLO DE ÁRVORE DE DERIVAÇÃO

Algoritmo Geral

Após a leitura de parâmetros, o sistema executa os seguintes passos com um conjunto fixo de parâmetros. Durante esta execução, o sistema se utiliza do módulo Oracle-Evaluate para as operações que acessam ao banco de dados.

a) Separação dos registros nas **tabelas de treinamento e de teste**

b) **Criação da população inicial** – Criação de N classificadores de acordo com o arquivo BNF. Onde, N é um número definido no arquivo de parâmetros

c) Avaliação de *fitness* de cada indivíduo

d) Seleção do indivíduo com menor *fitness* para ser o **melhor indivíduo (*Best*).** O *fitness* é 1 menos a precisão do classificador na base de treinamento.

e) Selecionar "*Best Test*" como sendo o *Best*

f) Enquanto algum **critério de parada** for satisfeito:
- *fitness* do *Best* > 0 ou
- *fitness* do *Best* > Precisão (Anexo C - *Precision*) ou
- *accuracy* do *Best* na base de teste for igual a 100 % ou

- Se o sistema estiver a mais de X gerações sem mudar o indivíduo *Best* (Anexo C - Generations_equal) ou
- o número máximo de gerações (Anexo C - Number_of_generations) for alcançado
faça:

1. Se for escolhido a estratégia elitista, inclua o *Best* na nova geração.
2. Repita até criar uma nova geração com o mesmo número de indivíduos
    i. Escolher um operador genético: *crossover*, mutação ou reprodução
    ii. Selecionar um indivíduo para mutação ou 2 indivíduos para a operação.
    iii. Aplicar o operador genético
    iv. Calcular o *fitness* do(s) indivíduo(s) criado(s)
    v. Adicionar o(s) indivíduo(s) criado(s) na nova geração
3. Selecionar o indivíduo de menor *fitness*.
4. Se o *fitness* for menor ou igual do que o *fitness* do *Best* então o *Best* passa ser o indivíduo selecionado.
5. Se a *accuracy* do *Best* na base de teste for maior do que o *Best Test* então ***Best Test*** passa a ser o indivíduo *Best*
6. Acrescente um ao número da geração

g) Indicar os melhores indivíduos de todas as gerações
h) Resumir e salvar os **resultados obtidos**

A execução dos passos acima corresponde a um *run*. O sistema executa estes passos N vezes (N é o número de *runs* definido no arquivo de configuração - Anexo C - Number_of_runs) mantendo os outros parâmetros constantes. A cada *run*, todos os resultados obtidos, como totais, médias, *accuracy*, entre outros, são armazenados em arquivos textos.

A seguir, serão descritas em detalhes as execuções de alguns dos passos realizados pela GGP.

TABELA DE TESTE E TREINAMENTO

O sistema utiliza a taxa de erro como um dos critérios para avaliar a qualidade das regras de classificação encontradas (definida no Capítulo 2). Para avaliação da taxa de erro, propõem-se em dividir a base de dados em duas partes distintas: uma para teste e outra para treinamento. E no início de cada *run*, o sistema cria e transfere os registros da tabela principal para estas tabelas com o auxílio das informações contidas no SQL Parameter e algumas funcionalidades do módulo Oracle-Evaluate.

A primeira tabela é criada para conter os registros de treinamento e tem o nome formado pela concatenação do Nome da tabela 'Goal' com o string "_testing" seguido do ID da conexão no Oracle, ou seja, todos os nomes têm o seguinte formato: "NomeTabelaGoal_trainingID". Esta tabela contém a coluna objetivo e todas as colunas que estão na mesma tabela da coluna objetivo. Além destas colunas, uma coluna: 'Keycolumn' é criada. A coluna 'Keycolumn' é uma coluna auxiliar criada pelo sistema que contém um número único e seqüencial para cada linha da tabela. Com esse número, o sistema tem condições de saber quais os registros foram selecionados de cada tabela ou não.

A segunda tabela tem o nome com o formato: "NomeTabelaGoal_testingID". Existe uma semelhança na formação do nome se comparada com a primeira tabela, a diferença é o string "testing", que indica que é uma tabela que conterá os dados de teste. Esta tabela conterá as mesmas colunas da primeira tabela.

O sistema GPSQL Miner divide os registros da tabela principal em duas novas tabelas através de três opções. Nas três opções, tem-se a criação das tabelas de Treinamento e de Teste no início de cada *run*.

A primeira opção divide a tabela principal de acordo com uma porcentagem sem preocupação das classes. O usuário especifica qual a porcentagem para o conjunto de Treinamento (Train%). E no início de cada *run*, após a criação da tabela de treinamento o sistema inclui aleatoriamente Train% dos registros da tabela principal na tabela criada.

Depois, cria-se a tabela de testes com os registros ainda não selecionados da tabela principal que representa (100- Train)% dos registros da tabela principal. Nesta seleção não existe a preocupação de se manter a porcentagem de registros por classe.

A segunda opção é a divisão da tabela principal de acordo com uma porcentagem e mantendo a taxa de cada classe. Primeiramente, o sistema separa as classes (C1, C2, .., CN) e calcula a porcentagem de registros para cada classe (PC1, PC2, .., PCN). Logo após, o sistema seleciona Train% dos registros da tabela principal para a tabela criada. A seleção destes registros é feita de modo que a porcentagem de cada classe (PC1, ..., PCN) possa ser mantida. Os demais registros são inseridos na tabela de Teste, ou seja, (100- Train)% dos registros da tabela principal.

Na terceira opção, o usuário pode informar as tabelas de treinamento e de teste de cada *run*, especificados na seção 'TrainTestTables' do arquivo SQL Parameter. Nesta opção, no início de cada *run* as tabelas de treinamento e de teste são carregadas com os registros das tabelas indicadas em 'TrainTestTables'.

Após a criação das tabelas de treinamento e de teste para cada *run*, o sistema cria arquivos textos contendo os registros de cada tabela. A partir destes arquivos, é possível saber exatamente quais registros foram utilizados em cada tabela a cada *run* e quais poderão ser utilizado por outros sistemas para comparação ou facilmente transferidos para o banco de dados.

CRIAÇÃO DA POPULAÇÃO INICIAL

Após a criação das tabelas de treinamento e de teste tem-se a formação da população inicial. Os indivíduos são criados respeitando a profundidade mínima e máxima da árvore, e o número de indivíduos da população (parâmetros definidos no arquivo *.gpm). A população inicial forma a geração zero.

Os indivíduos são criados de acordo com a gramática definida no arquivo BNF. Se for utilizado o formato padrão da gramática criado pelo sistema podemos verificar que a

população inicial é criada com certa diversidade. Isto foi observado através dos arquivos de análise da população, pois são raras as populações criadas que possuam indivíduos iguais.

MELHOR INDIVÍDUO

Após a criação da população inicial, todos os indivíduos da população são avaliados. O indivíduo que tiver o menor *fitness* será o *Best* indivíduo, ou melhor indivíduo. Se houver mais de um indivíduo com o mesmo *fitness*, será escolhido o classificador com menor número de folhas pois acredita ser o mais compreensível [Lim 1999].

Após a escolha do primeiro melhor indivíduo, o sistema entra em um laço para criação de uma nova geração de indivíduos. A cada execução do laço, o sistema cria a nova geração utilizando as operações genéticas. No final do laço, o sistema separa o indivíduo que tiver menor *fitness* como sendo o melhor indivíduo da geração. E assim, com a conclusão do laço, o sistema seleciona o indivíduo de melhor *fitness* da última geração. Se existir mais de um indivíduo com mesmo *fitness* o sistema seleciona todos estes indivíduos, porém, o indivíduo de menor tamanho é indicado como sendo o melhor indivíduo de todas as gerações.

Dentre os melhores indivíduos de todas as gerações, o sistema separa o indivíduo que tiver a menor taxa de erro, denominado indivíduo com melhor precisão na base de teste (*Best-Test*). A taxa de erro é calculada apenas nos melhores indivíduos como uma forma de diminuir o esforço computacional.

No final de cada *run* o sistema selecionará um indivíduo com o menor *fitness* (*Best*) e um indivíduo com a menor taxa de erro (*Best-Test*).

CRITÉRIO DE PARADA

São considerados os seguintes critérios para que o sistema termine o laço para gerar uma nova geração:

- Se o *fitness* do melhor indivíduo for igual a zero ou o *fitness* do indivíduo for menor do que a Precisão estabelecida como parâmetro.

- Se a taxa de erro do melhor indivíduo for igual a zero.

- Se o sistema atingir o número máximo de iterações definido como parâmetro ou

- Se o sistema está a mais de GE gerações sem encontrar um indivíduo com *fitness* menor do que o melhor indivíduo. Onde GE é o número *Generation Equal* definido nos parâmetros.

MÉTODO DE SELEÇÃO

O método de seleção utilizado pelo sistema é o torneio (definido na Seção 3.4). Com uma pequena diferença, o sistema GPSQL Miner selecionará o indivíduo que tiver o menor *fitness* para as operações genéticas.

RESULTADOS OBTIDOS

No final de vários *runs* com os mesmos parâmetros, vários resultados são selecionados ou resumidos. Segue-se abaixo a definição e conteúdo dos arquivos criados.

Arquivo de Resultados: Este arquivo contém as seguintes informações: Número de *Runs*; número de registros da tabela de treinamento; número de registros da tabela de teste; média das últimas gerações em que o sistema encontrou os melhores indivíduos; média do *fitness* mínimo, médio e máximo; média do número de melhores indivíduos por *run*; tempo médio de execução de um *run*.

Ainda dentro do arquivo de resultados existem as informações em relação aos melhores indivíduos: média da precisão do *Best* indivíduo na base de treinamento; média da precisão do *Best* indivíduo na base de teste; média da precisão do *Best-Test* indivíduo na base de treinamento; média da precisão do *Best-Test* indivíduo na base de teste; média do

número de *Best* indivíduos de cada *run*; número médio das regras, profundidade média e tamanho médio de cada classificador.

Arquivo com o *Best* e *Best-Test* indivíduos de cada *run*. Este arquivo contém os melhores indivíduos de modo legível para o usuário, seu *fitness*, total de registros classificados pelo indivíduo separados por classe.

Parâmetros

O arquivo de configurações contém os principais parâmetros da Programação Genética e tem o nome no formato: '*.gpm'; mais detalhes de cada parâmetro podem ser encontrados na Seção 3.7 ou no Anexo C. Além dos parâmetros já citados alguns parâmetros adicionais foram utilizados:

Número de *Runs* – Número de vezes que o sistema irá executar um conjunto de parâmetros. Após o término destas execuções, o sistema criará um resumo com todos os resultados.

Generation Equal (GE) – Se o sistema estiver a GE gerações sem achar um indivíduo com ótimo *fitness* o sistema irá parar o *run* atual.

Como método de inicialização foi utilizado *Grow*, definido na Seção 3.7.

Estratégia elitista – Aceita os valores Y/N. Onde, Y indica que o sistema sempre irá reproduzir o indivíduo de melhor *fitness* para a geração seguinte. Já a opção N não assegura a preservação do melhor indivíduo.

Tipo de *fitness* – Foram disponibilizados dois tipos de *fitness*. O primeiro *fitness* é igual a um menos a precisão (ou a taxa de erro do classificador), o segundo é calculado como: *fitness* = 1,01 – precisão. Mesmo que o sistema tenha encontrando um classificador

com 100% de precisão na base de treinamento, o segundo *fitness* permite encontrar outros classificadores que também atinjam 100% de precisão e dentre estes selecionar o classificador que tenha uma menor taxa de erro da base de teste.

Existem alguns outros parâmetros para indicar se o sistema deve criar alguns arquivos para avaliação. Estes arquivos, de origem estatística ou não, auxiliam na condução dos experimentos e são citados a seguir:

Salvar a População (Sim/Não) – Se o sistema deve ou não salvar todos os indivíduos de uma população em arquivos separados para cada geração. Serão salvos o string do classificador completo e seu *fitness*.

Criar arquivo de Análise da População (Sim/Não) – Se o sistema deve ou não criar um arquivo para cada *run* contendo informações dos indivíduos iguais de cada geração.

Arquivo de *Crossover* (Sim/Não) – Se o sistema deve ou não criar um arquivo com os detalhes de todos os *crossover's* com objetivo para análise de cada operação e da própria gramática. Com este arquivo é possível acompanhar os detalhes de cada indivíduo e das sub-árvores trocadas.

Arquivo de estatística de *Crossover* (Sim/Não) – Se o sistema deve ou não criar um arquivo com as informações de todos os *crossover's* com objetivo de análise estatísticas. O arquivo contém informações como: os indivíduos envolvidos, o nodo não terminal em que foi feita a operação, o *fitness* antes e depois do *crossover*.

Arquivo de estatística de Mutação (Sim/Não) – Se o sistema deve ou não criar um arquivo com as informações de todas as mutações com objetivo de análise estatísticas. O arquivo informa o indivíduo envolvido, o nodo não terminal em que foi feita a operação, a nova sub-árvore criada, o *fitness* antes e depois do *crossover*.

Arquivo de estatística dos Operadores Genéticos (Sim/Não) – Se o sistema deve ou não criar um arquivo contendo informações de todas as operações genéticas para avaliação da efetividade das operações. Uma informação deste arquivo é a diferença de *fitness* dos indivíduos antes e depois de cada operação.

Arquivo de Saída (Sim/Não) – Se o sistema deve ou não gerar um arquivo de saída contendo os parâmetros utilizados para o *run* e os detalhes dos melhores indivíduos

5.5 Módulo Oracle-Evaluate

Este módulo é responsável pela avaliação de cada indivíduo através do acesso ao SGBD Oracle; e, também, é responsável por todos os outros acessos ao Oracle. A seguir descrevem-se algumas observações relevantes para apresentação do módulo.

Como a base de dados está em um banco relacional instalado em um servidor, o mesmo pode ser acessado por vários usuários que tenham direito de acesso. Por isso, o sistema não pode acessar os dados de modo exclusivo e, além disso, o sistema deve prever o acesso às informações para várias instâncias do sistema e de outros ao mesmo tempo.

Um outro fato a ser considerado é que o sistema pode estar acessando diretamente uma base com informações originais e que não devem ser alteradas. Qualquer alteração para avaliação do sistema deve ser feita de modo que não afete os dados originais.

A partir destas observações, o sistema possui algumas características como: 1) Para cada *run* o sistema criará uma tabela com os dados de treinamento, uma com os dados de teste e uma última tabela de resultados para auxiliar no cálculo do *fitness*; 2) Para cada instância do sistema, as tabelas serão criadas com nomes de acordo com a base, tipo e o ID da conexão, que é um número interno dado pelo Oracle para cada conexão e que é único. A estas tabelas chamaremos de tabelas auxiliares.

As tabelas auxiliares são utilizadas pelo módulo para avaliação do *fitness*. O *fitness* é baseado na *accuracy*, que é a porcentagem de registros da base de teste classificados corretamente. Para ser mais preciso, o *fitness* é calculado através da seguinte fórmula:

Fitness = 1 – (No. Registros classificadores corretamente / No. Total de registros da base de treinamento).

Para a avaliação do *fitness*, cada regra dos indivíduos é transformada em um comando SQL e executado separadamente. No final da transformação de cada regra, o sistema lê a tabela de resultados para o cálculo do *fitness*.

A seguir, será descrita a transformação de um classificador completo (Figura 5.10) em um comando SQL (Figura 5.11) e outros passos para a avaliação de *fitness*.

```
S <valor_goal> W <predicting attributes>
S <valor_goal> W <predicting attributes>
...
S <valor_goal> W <predicting attributes>
S <valor_goal>
```

FIGURA 5.10: FORMATO GERAL DE CLASSIFICADOR

O sistema percorre a árvore de derivação para cada indivíduo e transforma cada regra separadamente. Cada regra do classificador (uma linha da Figura 5.10) é convertida para um comando *Insert* (comando SQL que insere dados em uma tabela) sendo executado em seguida. O formato deste comando *Insert* tem o formato conforme a Figura 5.11.

```
Insert into gpsql_resultID
Select Número_regra, <valor_goal>, <coluna_goal>, Apelido_ TabelaGoal.KEYCOLUMN
From TabelaGoal_ID Apelido_TabelaGoal, Tabela2_ID Apelido_Tabela2, ...
Where <predicting attributes>
and not exists(select 1 from gpsql_resultID where KEYCOLUMN = Alias_TabelaGoal.KEYCOLUMN)
```

FIGURA 5.11: FORMATO GERAL PARA COMANDO INSERT

Na Figura 5.11, 'KEYCOLUMN' é uma coluna da tabela principal com um valor único para cada registro da tabela, criada para que o sistema possa ter a certeza de que o registro não seja selecionado por mais de uma regra, ou seja, é uma forma do sistema

trabalhar com regras ordenadas. A tabela 'gpsql_resultID' é a tabela de resultados (sua definição esta no Anexo B) que conterá os registros selecionados de cada regra. Os <predicting attributes> é a junção do string 'where' da regra (<predicting attributes> da Figura 5.10) e todos os strings das cláusulas 'where' (contidas no arquivo SQL Parameter) de cada coluna que aparece na regra em questão.

Após a criação do primeiro *Insert* o sistema executa-o, fazendo com que os registros da tabela principal sejam selecionados na tabela de resultado 'gpsql_resultID' que até o momento estava sem nenhum registro. A segunda regra é convertida para outro comando *Insert* que não considera os dados já inseridos pela primeira regra e em seguida executa-o. E sucessivamente para as demais regras. Na última regra, a regra *default* "S <valor_goal>" será convertida para o comando *Insert* cujo formato esta na Figura 5.11. Nota-se que o string "and not exists(select 1 from gpsql_resultID where KEYCOLUMN = Alias_TabelaGoal.KEYCOLUMN)" é a restrição da cláusula 'where' que não seleciona os registros que já foram inseridos na tabela de resultados. Este string é o mesmo que é inserido na conversão da 2ª à última regra, não sendo necessário para a primeira regra.

Após a transformação e a execução de cada regra, todos os registros da tabela original de treinamento estarão na tabela de resultados (gpsql_resultID). Através do comando SQL (Figura 5.12) que seleciona o total de registros classificados corretamente e o comando para selecionar o total da tabela de treinamento (Figura 5.13) o sistema seleciona as informações suficientes para o cálculo do *fitness*.

```
select count(1) from gpsql_resultID where class = classified
```

FIGURA 5.12: SQL PARA SELEÇÃO DO TOTAL CLASSIFICADO CORRETAMENTE

```
select count(1) from Compra_trainingID
```

FIGURA 5.13: EXEMPLO DE SELEÇÃO DE TOTAL DA TABELA DE TREINAMENTO

5.6 Outras funcionalidades

O sistema possui algumas características adicionais:

Pode criar os seguintes arquivos:

- *Generation* – Contém todos os indivíduos da geração;
- *Analyse Population* – Indica quais indivíduos estão repetidos e o total por geração;
- Arquivo de *Crossover* – Mostra os detalhes dos dois indivíduos durante o crossover, indicando os pontos que foram trocados pelo operador genético;
- *Output File* – Para cada *run*, é criado um arquivo com os melhores indivíduos de cada geração;
- Tabelas utilizadas – Para cada *run*, criam-se um arquivo contendo os registros da tabela de treinamento utilizados e um outro arquivo contendo os registros da tabela de testes;
- Arquivos de estatísticas: *Crossover*, Mutação e Todas as operações. Estes arquivos facilitam a visualização de cada operação, bem como permite verificar estatisticamente os pontos de *crossover* ou de mutação;
- Arquivo de análise da gramática – Tem como objetivo facilitar o melhoramento do arquivo BNF. Agrupa as produções que podem ser geradas para cada não terminal, juntamente com a percentagem para escolha de cada produção;

- Arquivo com faixa de *Fitness* - A cada geração, os indivíduos e seus *fitness* são agrupados por faixa de valores e mostrados de uma forma 'visual' para que o usuário tenha uma idéia do *fitness* da geração de modo rápido. Esta forma 'visual' foi obtida apenas utilizando-se de recursos do modo texto.

O sistema também pode executar várias vezes variando automaticamente os parâmetros. Os parâmetros que podem ser variados são: Tamanho da população, Tamanho do Torneio, Taxa de *Crossover* e Taxa de Mutação. Por exemplo, pode-se pedir para o sistema executar automaticamente um *run* com a população 300 e uma outra vez com a população 600, mantendo-se os demais parâmetros constantes.

Podem-se executar várias instâncias do sistema GPSQL Miner para a mesma base de dados sem ter problemas de concorrência. As tabelas de resultado criadas especialmente para cada problema possuem o ID da conexão Oracle, com isso, o sistema pode ter dados originais em um único local que podem ser utilizados por outras execuções do sistema ou até outros programas.

Os arquivos textos de saída criados pelo sistema estão divididos em diretórios criados de acordo com os parâmetros utilizados, uma forma de facilitar o controle dos resultados. Na Tabela 5.1 têm-se os sub-diretórios criados. A Tabela 5.2 possui os arquivos de resultados criados no sub-diretório #6 da Tabela 5.1.

TABELA 5.1: SUB-DIRETÓRIOS CRIADOS

#	Sub-Diretório	Descrição
1	Results	Sub-diretório criado abaixo do diretório aonde se encontra o sistema
2	Results\<SQLParameter>	<SQLParameter> é o nome do arquivo SQL Parameter utilizado (Ex.: bcw.par).
3	..\Run<R>Gen<G>GenE<GE>Min<Mi> Max<M>Cross<C>Train<T><MC>	Sub-diretório criado abaixo do sub-diretório #2. O seu nome é formado com os valores dos parâmetros que não são variados pelo sistema. Aonde: <R> - número do run; <G> - Número máximo de Gerações <GE> - *Generation Equal* <Mi> - Tamanho mínimo da árvore <M> - Tamanho máximo da árvore <C> - Tamanho máximo de uma árvore durante um *crossover* <T> - Porcentagem conjunto Treinamento <MC> - manter a porcentagem por classe no conjunto de treinamento (Y/N) ?
4	..\Init<I>Fit<F>Tour<T><E>	Sub-diretório criado abaixo do sub-diretório #3, aonde: <I> - método de inicialização <F> - Tipo de *Fitness* <T> - Número de indivíduos para o Torneio <E> - Utilizar *elitisty* (Y/N) ?
5	..\<BNF_File>	Sub-diretório criado abaixo do #4 com o nome do arquivo BNF utilizado
6	..\ Pop<P>	Sub-diretório criado abaixo do #5, aonde <P> é o tamanho da população. Neste sub-diretório conterá o resumo dos parâmetros e resultados obtidos.
7	..\Best	Sub-diretório criado abaixo do #6. Contém arquivos para cada *run* com informações dos melhores indivíduos de cada geração.
8	..\Fitness	Sub-diretório criado abaixo do #6. Contém arquivos para cada *run* com informações dos *fitness* de cada geração separados por faixa de valores
9	..\Output	Sub-diretório criado abaixo do #6. Contém arquivos para cada *run* com os melhores classificadores SQL
10	..\Population	Sub-diretório criado abaixo do #6. Contém os arquivos Analyse Population para cada *run*
11	..\Statistic	Sub-diretório criado abaixo do #6. Contém os arquivos de estatísticas separados por *run*.
12	..\Tables	Sub-diretório criado abaixo do #6. Contém os arquivos das tabelas utilizadas para cada *run*

TABELA 5.2: ARQUIVOS DE RESULTADOS

Nome do Arquivo	Descrição
Cros<C>Mut<M>_allbest.txt	Todos as informações dos melhores indivíduos de cada geração de todos os *runs*
Cros<C>Mut<M>_analyse_<BNF_File>	Arquivo com a análise do arquivo BNF. Aonde: <BNF_File> é o nome do arquivo.
Cros<C>Mut<M>_<SQLParameter>	Cópia dos SQL Parameter utilizado . Aonde: <SQLParameter> - nome do arquivo
Cros<C>Mut<M>_<BNF_File>	Cópia do arquivo BNF utilizado. Aonde: <BNF_File> - nome do arquivo
Cros<C>Mut<M>_config.gpm	Arquivo com as configurações para GP
Cros<C>Mut<M>_ gpsql.bnf	Arquivo BNF criado pelo sistema para o problema em questão
Cros<C>Mut<M>_ result.run	Resumo dos resultados de cada *run*
Cros<C>Mut<M>_ result.txt	Resumo do arquivo (..result.run), ou seja, o resumo de todos os run com os parâmetros que estão indicados no nome dos diretórios
Cros<C>Mut<M>_ result_best.txt	Os melhores classificadores SQL de cada *run*, os indivíduos Best e Best-Test

Observação: <C> - Taxa de Crossover e <M> - Taxa de Mutação

5.7 Discussão do Capítulo

Neste capítulo, foram apresentadas todas as características do sistema GPSQL Miner para que o mesmo possa selecionar os melhores classificadores para cada problema. Foram citadas todas as suas funcionalidades divididas em módulos. Primeiramente, foi apresentado o novo formato SQL para representar os classificadores. Em seguida, foram citados os objetivos gerais de cada módulo e apresentados em detalhes. Por último, foram apresentados algumas características adicionais do sistema.

No próximo capítulo, será citado um exemplo do sistema como uma forma de apresentar o seu funcionamento e suas potencialidades.

6. Exemplo de Execução

Esta seção apresenta em maiores detalhes as funcionalidades de GPSQL Miner mediante um exemplo hipotético. O problema consiste em saber informações sobre os clientes que compram computadores *Desktop, Laptop* e *Handheld* (computadores de mão) da empresa fictícia MCRSIS Informática. Deseja-se saber o perfil dos clientes que compram cada tipo de computador, para isso, têm-se informações como dados pessoais do cliente, o país de sua residência e de seu trabalho.

A empresa mantém um pequeno cadastro de seus clientes em três tabelas, cujo relacionamentos estão indicados na Figura 6.1. Existe entre estas tabelas, a coluna 'Computador' da Tabela 'Compra' que armazena o computador que o cliente adquiriu. Ou seja, a coluna 'Computador' é o atributo objetivo, a tabela 'Compra' é a tabela principal. Os valores desta coluna: 'Desktop', 'Laptop' e 'Handheld' são as classes. A seguir, estão descritas as tabelas e outras colunas (atributos de predição) que estão no banco de dados Oracle.

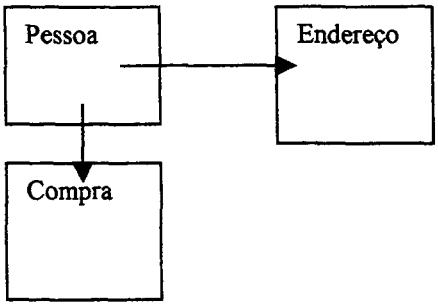

FIGURA 6.1: RELACIONAMENTO ENTRE AS TABELAS

A tabela 'Pessoa' (Tabela 6.1) possui informações gerais sobre os seus clientes. A tabela possui 4 colunas 'Id' (código que identifica o cliente), 'Nome', 'Gênero' (M – Masculino; F – Feminino) e 'Idade' (em anos).

A tabela 'Endereco' (Tabela 6.2) possui os endereços residencial e comercial de cada cliente. As tabelas possuem as colunas: 'Id' (Código do Cliente), 'Tipo' (R –

Residencial; C – Comercial) e 'País' (país da residência ou escritório). Esta tabela pode conter duas informações diferentes para cada cliente: o seu endereço comercial e o endereço residencial.

Tabela 'Compra' (Tabela 6.3) possui a descrição de quais computadores cada cliente comprou. As colunas da tabela são: 'Id' (Código do Cliente), 'Mes' (Mês da Compra), 'Computador' (indica o computador que o cliente adquiriu: Desktop, Laptop ou Handheld).

Nas Tabelas 6.1, 6.2 e 6.3 estão todos os registros de cada tabela.

TABELA 6.1: TABELA PESSOA

ID	NOME	Gênero	IDADE
1	Pessoa 1	M	20
2	Pessoa 2	M	16
3	Pessoa 3	F	18
4	Pessoa 4	F	29
5	Pessoa 5	F	25
6	Pessoa 6	M	16
7	Pessoa 7	M	15
8	Pessoa 8	F	13
9	Pessoa 9	F	29
10	Pessoa 10	M	50

TABELA 6.2: TABELA ENDERECO

Id	Tipo	Pais
1	R	Itália
2	R	Inglaterra
3	R	França
4	R	Inglaterra
5	R	França
6	R	Portugal
7	R	Alemanha
8	R	Inglaterra
9	R	França
10	R	Alemanha
1	C	França
2	C	Inglaterra
3	C	França
4	C	Inglaterra
5	C	França
6	C	Alemanha
7	C	Alemanha
8	C	Alemanha
9	C	França
10	C	França

TABELA 6.3: TABELA COMPRA

ID	Mes	Computador
1	MAY	Desktop
1	MAY	Desktop
1	FEB	Desktop
2	JAN	Desktop
2	JAN	Desktop
3	AUG	Desktop
3	FEB	Desktop
4	DEC	Laptop
4	JAN	Desktop
5	OCT	Laptop
5	JAN	Desktop
5	JAN	Desktop
5	OCT	Laptop
5	OCT	Laptop
5	NOV	Laptop
6	JUL	Handheld
6	AUG	Handheld
6	JAN	Handheld
6	NOV	Laptop
7	DEC	Desktop
7	NOV	Laptop
7	JAN	Desktop
8	NOV	Laptop
8	NOV	Laptop
8	JUL	Handheld
9	SEP	Desktop
9	JAN	Desktop
10	MAR	Handheld
10	MAR	Handheld
10	SEP	Handheld

Nota-se que a organização dos dados poderia ser otimizada ou o formato melhorado, contudo, os dados foram organizados para melhor entendimento das potencialidades do sistema.

6.1 SQL Parameter

A Figura 6.2 mostra o arquivo SQL Parameter para o problema. A primeira seção (//Tables) indica todas as tabelas que serão utilizadas e seu alias. Na segunda linha

pode-se observar a indicação da tabela 'Pessoa' com o alias 'P'. Ou seja, a indicação do alias 'P', geralmente descrita por 'P.', referência a um atributo da tabela 'Pessoa'.

```
//Tables
P -> Pessoa
Com -> Endereco
Res -> Endereco
C -> Compra
//Goal
Column -> Computador
Table -> C
//Columns
Column -> P.GENERO
Table -> C
where -> P.ID = C.ID
Column -> P.IDADE
Table -> C
where -> P.ID = C.ID
Column -> Res.Pais
Table -> C
where -> Res.ID = C.ID and Res.Tipo = 'R'
Column -> Com.Pais
Table -> C
where -> Com.ID = C.ID and Com.Tipo = 'C'
Column -> C.Mes
Column -> C.ID
```

FIGURA 6.2: COMPUTADOR.PAR - SQL PARAMETER PARA A BASE COMPUTADOR

O problema tem como atributo objetivo: 'Computador', da Tabela 'Compra' (definido na seção 'Goal'). A coluna 'Computador' aceita os valores das classes, porém, nota-se que as três não foram informadas, pois estes valores são pesquisados no banco de dados pelo sistema, de forma a permitir uma maior automatização do sistema.

As colunas de predição serão: 'Gênero' e 'Idade' da tabela 'Pessoa'; 'País' da tabela 'Endereço' e 'Mes' da tabela 'Compra'. Nota-se que algumas colunas não serão utilizadas para a descoberta de conhecimento como, por exemplo, a coluna 'Nome' da tabela 'Pessoa'. Tais colunas não necessitam serem informadas no arquivo. Outras como as colunas Id's serão utilizadas apenas para o relacionamento entre as tabelas.

Um outro fato é que a tabela 'Endereco' contem os endereços residencial e/ou comercial de cada cliente. Cada cliente pode ter um registro nesta tabela para o endereço residencial, na qual a coluna Tipo vale 'R' e um segundo registro contendo o seu endereço comercial (Tipo = 'C'). Por isso, a tabela deve ser informada duas vezes no arquivo, uma

com apelido 'Com' (abreviatura de Comercial) e outra de apelido 'Res' (abreviatura de Residencial) (3ª e 4ª linha da Figura 5.4).

Na seção 'Columns' foram informadas as colunas Gênero, Idade, País da Residência (coluna Res.Pais), País do trabalho (coluna Com.Pais), Mês da Compra e a coluna ID. Nota-se que a coluna 'ID' não é importante para a descoberta de conhecimento, mas é importante para o relacionamento entre as tabelas.

No exemplo citado, a coluna 'Com.Pais' tem como tabela adicional a tabela 'C' (*alias* da tabela Compras). Ou seja, toda vez que a coluna 'Com.Pais' fizer parte de uma regra, ela deve obrigatoriamente adicionar ao comando SQL a tabela Compras e o string: "Com.ID = C.ID and Com.Tipo = 'C'" na cláusula 'where' do comando SQL. A parte "Com.ID = C.ID" indica o *join* entre as colunas 'ID' das tabelas 'C' e 'Com'; a parte "Com.Tipo = 'C'" indica que só serão considerados os registros da tabela 'Comercial' (de *alias* 'Com') que possuem o valor 'C' na coluna 'Tipo'.

Na última linha da Figura 6.2, a coluna 'C.ID' é indicada sem informações da tabela e cláusula *where*. O motivo da não indicação de tabelas é porque a coluna não necessita de outras tabelas para ser utilizada, pois se encontra na mesma tabela do atributo objetivo.

6.2 Arquivo BNF

Na Figura 6.3, pode-se observar um exemplo de um arquivo BNF gerado pelo sistema a partir do arquivo SQL Parameter da Figura 6.2. Os números do lado esquerdo apenas indicam a linha e não existem no arquivo original. Abaixo se discutem alguns detalhes sobre o arquivo criado.

```
1     #productions
2     S      -> <classifier>
3     <classifier> -> <r> <r> <r> <r> <r> S <goal>
4     <r> -> S <goal> W <cond>
5     <r> -> S <goal> W <cond> AND <cond>
6     <r> -> S <goal> W <cond> AND <cond> AND <cond>
7     <r> -> S <goal> W <cond> AND <cond> AND <cond> AND <cond>
8     <goal> -> 'Desktop' | 'HandHeld' | 'Laptop'
9     <cond> -> P.GENERO = <value_P.GENERO>
10    <value_P.GENERO> -> 'F' | 'M'
11    <cond> -> P.IDADE <rel_P.IDADE>
12    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
13    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
14    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
15    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
16    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
17    <rel_P.IDADE> -> <Noperator> <value_P.IDADE>
18    <rel_P.IDADE> -> between <value_P.IDADE> and <value_P.IDADE>
19    <value_P.IDADE> -> 21 | 34 | 18 | 23 | 55 | 30 | 25 | 20
20    <cond> -> RES.PAIS <Coperator> <value_RES.PAIS>
21    <value_RES.PAIS> -> 'França' | 'Alemanha' | 'Inglaterra' | 'Itália' | 'Portugal' | COM.PAIS
22    <cond> -> COM.PAIS <Coperator> <value_COM.PAIS>
23    <value_COM.PAIS> -> 'França' | 'Alemanha' | 'Inglaterra' | 'Itália' | 'Portugal' | RES.PAIS
24    <cond> -> C.MES <Coperator> <value_C.MES>
25    <value_C.MES> -> 'JAN' | 'NOV' | 'OCT' | 'AUG' | 'DEC' | 'SEP' | 'MAR' | 'MAY' | 'JUL' | 'FEB'
26    <Coperator>  -> = | !=
27    <Noperator>  -> = | != | > | < | >= | <=
```

FIGURA 6.3: ARQUIVO COMPUTADOR.BNF PARA DATABASE COMPUTADOR

A partir do arquivo "computador.bnf" (Figura 6.3), o sistema gera apenas classificadores com cinco regras, o que pode ser verificado pela existência de cinco não terminais <r> na linha 3. Cada regra pode ter de um (linha 4) a quatro (linha 7) atributos de predição, sendo que a chance de se ter uma regra com 2 atributos é a mesma chance do sistema criar uma regra com 1, 3 ou 4 atributos, ou seja, uma opção em quatro que representa 25%. Nota-se que o número máximo de atributos é 4 (linha 7) pois é o número de colunas utilizadas para predição.

Na linha 8 temos as três classes do problema: 'Desktop', 'HandHeld' e 'Laptop'. Existem 4 colunas de predição, todas começando com o não terminal <cond> (linhas 9, 11, 20, 22).

Na linha 9 e 10 temos um exemplo de linhas criadas para uma coluna com 2 valores ('F' - Feminino e 'M' - masculino).

Como exemplo de uma coluna numérica tem-se nas linhas 11 a 19 o conjunto de terminais e não terminais para a coluna 'Idade'. Nota-se que as linhas 12 a 17 são todas iguais por causa da utilização do operador 'between' (linha 18). Na construção de uma árvore de derivação, se um nó não terminal <rel_P.IDADE> for gerado, a escolha para os seus nodos filhos deve ser entre os strings que estão à direita das produções das linhas 17 e 18, respectivamente "<Noperator> <value_P.IDADE>" ou "between <value_P.IDADE> and <value_P.IDADE>". Como esta escolha é aleatória, a probabilidade de se gerar sub-árvores para cada das produções do tipo 'P.Idade = 23', 'P.Idade != 23', 'P.Idade < 23', 'P.Idade > 23', 'P.Idade >= 23', 'P.Idade <= 23' ou 'P.Idade between 23 and 30' é igual a 14.28% para todos (resultado de 100 divididos por 7), ou seja, a probabilidade de se gerar um nodo com algum dos operados aritméticos é igual à probabilidade da geração de um nodo com o comando 'between'. A informação desta percentagem pode ser encontrada no arquivo de análise do arquivo BNF.

Nas linhas 20 e 22 tem-se o exemplo de 2 colunas do tipo char. As linhas 21 e 23 possuem os valores possíveis para cada coluna. Nota-se que na linha 21 existe como não terminal a coluna 'COM.PAIS', ou seja, a coluna 'RES.PAIS' poderá ser comparada diretamente com a coluna 'COM.PAIS'.

Na linha 24 tem-se a coluna 'Mes' do tipo char. Os valores possíveis para esta coluna estão na linha 25. Nota-se que os valores são abreviações dos meses em inglês.

O GPSQL Miner baseado no arquivo BNF (Figura 6.3) pode criar um classificador completo como mostra a Figura 6.4. Para esta representação apenas os não terminais do classificador são informados.

```
S 'Desktop' W RES.PAIS = 'Itália'
S 'Laptop' W C.MES = 'OCT' AND P.GENERO = 'F'
S 'Laptop' W P.IDADE <= 25 AND C.MES = 'NOV' AND RES.PAIS != 'Itália'
S 'Desktop' W COM.PAIS = RES.PAIS
S 'Desktop' W COM.PAIS = 'Portugal' AND P.GENERO = 'F' AND COM.PAIS = 'Portugal'
S 'Handheld'
```

FIGURA 6.4: CLASSIFICADOR PARA COMPUTADOR.BNF

O classificador (Figure 6.4) possui seis regras, cada uma formada pelos componentes: 'S' e 'W'. Após o caracter 'S' tem-se o valor da classe para a regra. (Ex.: 'Desktop' para a primeira regra e 'Laptop' para a Segunda). O componente 'W' é seguido pelos atributos de predição (Ex.: "P.IDADE <= 25 AND C.MES = 'NOV' AND RES.PAIS != 'Itália'" para a 3ª regra).

A primeira regra do classificador tem um atributo de predição ou uma coluna 'RES.PAIS'. É importante lembrar que todas as colunas têm o *alias* de sua tabela e este é especificado juntamente com a coluna no arquivo SQL Parameter (Figura 6.2).

A última regra: "S 'Handheld'" é uma regra *default*, ou seja, uma vez que não exista alguma restrição à regra, os dados que não foram selecionados nas regras anteriores serão selecionados e classificados para a classe "Handheld".

Internamente, o classificador da Figura 6.4 esta na forma de árvore de derivação conforme a Figura 6.5 que mostra parte do classificador em questão.

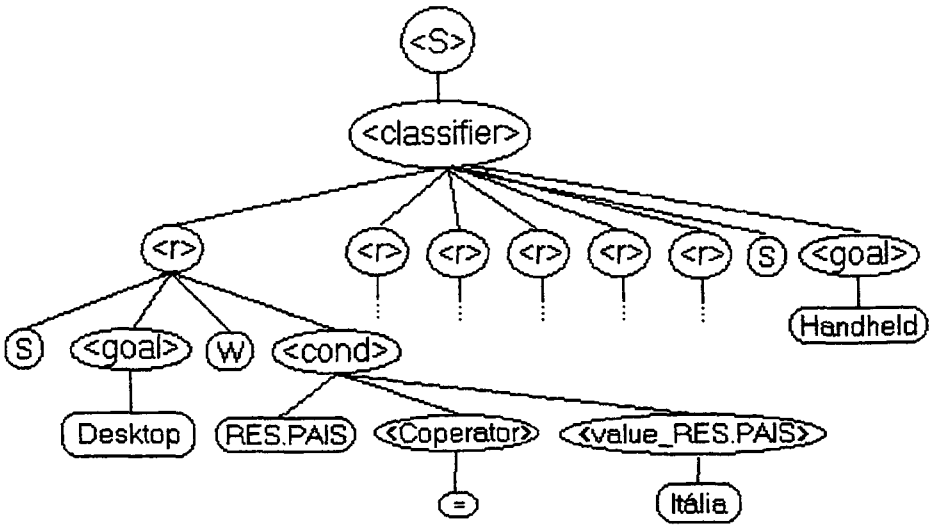

FIGURA 6.5: ÁRVORE DE DERIVAÇÃO PARA O CLASSIFICADOR FIGURA 6.4

6.3 GGP

Foi criado um arquivo texto (computador.gpm) para conter os principais parâmetros genéticos utilizados. Na Tabela 6.4 temos os parâmetros e os valores utilizados para a o exemplo em questão.

TABELA 6.4: COMPUTADOR.GPM

Parâmetros	Valores
Number of runs	5
Number of generations	500
Generations equal	20
Population size (begin)	500
Tournament size	3
Initialization method	2
Minimum tree height	3
Maximum tree height	30
Maximum crossover height	60
Crossover rate	50
Mutation rate	40
Type Fitness	3
Elitist strategy (Y/N/B)	N
Training Set	60
Separate Train class (Y/N)	Y
Interval Fitness	0.01
BNF file	Computador.bnf
SQL Parameter file	Computador.par
Precision	0.001
Save Population (Y/N)	N
Analyze Population (Y/N)	Y
Save Crossover file (Y/N)	N
Operators Statistic (Y/N)	N
Crossover Statistic (Y/N)	N
Mutation Statistic (Y/N)	N
Output File (Y/N)	Y

Depois da definição dos arquivos SQL Parameter e BNF, o sistema utiliza os parâmetros contidos no arquivo de configuração (*.gpm) para início da execução do módulo GGP. Após a leitura e a partir da tabela principal: 'Compra', o sistema cria duas tabelas auxiliares: COMPRA_training4766 (Tabela 6.5) e COMPRA_testing4766 (Figura 6.6). Obs.: O número 4766 é um número da conexão no Oracle.

TABELA 6.5: COLUNAS DA TABELA COMPRA_TRAINING4766

Nome	Tipo
KEYCOLUMN	NUMBER(10)
MES	CHAR(3)
ID	NUMBER(38)
COMPUTADOR	CHAR(9)

TABELA 6.6: COLUNAS DA TABELA COMPRA_TESTING4766

Nome	Tipo
KEYCOLUMN	NUMBER(10)
MES	CHAR(3)
ID	NUMBER(38)
COMPUTADOR	CHAR(9)

De acordo com os parâmetros, foi escolhida a divisão em 60% (Parâmetro "Separate Train Class" igual a 'Y' e "Training Set" igual a 60, da Tabela 6.4) na tabela de treinamento mantendo a percentagem de cada classe. Como a tabela principal possui trinta registros no total, a tabela de treinamento COMPRA_training4766 terá dezessete registros (aproximadamente 60% do total de registros da tabela original) e a tabela de teste COMPRA_testing4766 terá treze registros (40 % do total de registros da tabela original).

A Tabela 6.7 possui a quantidade e a porcentagem de registros na tabela principal. Como a porcentagem de cada classe deve ser mantida na tabela de treinamento, a classe 'Desktop' deve ter 46.67% dos registros nesta tabela, o que equivale a aproximadamente oito registros. Já a classe 'Handhled' deve conter quatro registros (23.33%) e a 'Laptop' deve conter 5 registros (30%). A Tabela 6.8 mostra os registros da tabela de treinamento de um *run* do sistema; o resumo do conteúdo da tabela de treinamento esta na Tabela 6.9, e a Tabela 6.10 contém os registros da tabela de teste. As porcentagens de registros para cada classe não são exatamente iguais se for comparado à tabela original (Compra) com a tabela de treinamento (COMPRA_training4766). Esta diferença é devido ao arredondamento, por exemplo, não é possível colocar 8,4 registros da classe 'Desktop' na tabela de treinamento.

TABELA 6.7: QUANTIDADE DE REGISTROS POR CLASSE DA TABELA COMPRA

COMPUTADOR	Quantidade	%
Desktop	14	46.67
Handheld	7	23.33
Laptop	9	30

TABELA 6.8: REGISTROS DA TABELA COMPRA_TRAINING4766

KEYCOLUMN	MES	ID COMPUTADOR
1	MAY	1 Desktop
2	JAN	2 Desktop
3	AUG	3 Desktop
4	FEB	3 Desktop
5	JAN	4 Desktop
6	JAN	5 Desktop
7	JAN	5 Desktop
8	JAN	7 Desktop
9	JUL	6 Handheld
10	AUG	6 Handheld
11	MAR	10 Handheld
12	SEP	10 Handheld
13	DEC	4 Laptop
14	OCT	5 Laptop
15	NOV	6 Laptop
16	NOV	7 Laptop
17	NOV	8 Laptop

TABELA 6.9: QUANTIDADE DE REGISTROS POR CLASSE DA TABELA COMPRA_TRAINING4766

Computador	Quantidade	%
Desktop	8	47.06
Handheld	4	23.53
Laptop	5	29.41

TABELA 6.10: REGISTROS DA TABELA COMPRA_TESTING4766

KEYCOLUMN	MES	ID	COMPUTADOR
1	MAY	1	Desktop
2	FEB	1	Desktop
3	JAN	2	Desktop
4	OCT	5	Laptop
5	OCT	5	Laptop
6	NOV	5	Laptop
7	JAN	6	Handheld
8	DEC	7	Desktop
9	NOV	8	Laptop
10	JUL	8	Handheld
11	SEP	9	Desktop
12	JAN	9	Desktop
13	MAR	10	Handheld

Como exemplo, pode-se citar a conversão do classificador da Figura 6.5 para o problema dos computadores. A primeira regra: "S 'Desktop' W RES.PAIS = 'Itália'" será transformada no comando da Figura 6.6.

Na figura 6.6, a coluna 'C.Computador' é o atributo objetivo do problema em questão. Esta coluna está sendo selecionada para ser comparada com o valor predito: "Desktop" e, posteriormente o sistema comparará estes dois valores para ver se a regra classificou o registro corretamente ou não. O número 4766 é o número da conexão (ID) no Oracle. Nota-se que o string "AND Res.ID = C.ID and Res.Tipo = 'R'" foi inserido pois o mesmo foi indicado na cláusula 'where' da coluna 'Res.Pais' (indicado no arquivo SQL Parameter - Figura 6.2).

```
insert into gpsql_result4766
select   1, 'Desktop' , C.Computador, C. KEYCOLUMN
from     Compra_training4766 C, Endereco Res
where    Res.PAIS = 'Itália'  AND Res.ID = C.ID and Res.Tipo = 'R'
```

FIGURA 6.6: EXEMPLO DE INSERT PARA 1^{A}. REGRA

Observa-se, que a Tabela 'Endereco' com o apelido 'Com' não foi inserida por não existir no classificador alguma coluna que exija esta tabela, no caso, a coluna: 'Com.Pais'. A não inclusão desta tabela melhora a performance do sistema, uma vez que diminui uma tabela na seleção das informações, ou seja, há uma otimização do *join*.

Após a criação do primeiro *Insert* o sistema executa-o, fazendo com que os registros dos clientes que residem na 'Itália' sejam classificados para a classe 'Desktop' e selecionados para a tabela de resultado 'gpsql_result4766'.

A segunda regra (Figura 6.4) é convertida para outro comando *Insert* que não considera os dados já inseridos pela primeira regra e em seguida executa-o. E assim, o sistema continua para as demais regras.

Na sexta regra da Figura 6.4, a regra *default* "S 'Handheld'" será convertida para o comando *Insert* da Figura 6.7. Nota-se que o string "not exists(select 1 from gpsql_result4766 where KEYCOLUMN = C.KEYCOLUMN)" é a parte da cláusula 'where' que não seleciona os registros que foram inseridos anteriormente na tabela de resultados. Este string será inserido na conversão da segunda à quinta regra.

Após a transformação e a execução de cada regra, todos os registros da tabela original de treinamento estão na tabela de resultados (gpsql_result4766). O cálculo de *fitness* baseia-se nas informações contidas nesta tabela. O cálculo é feito através do comando SQL da Figura 6.8 que seleciona o total de registros classificados corretamente e do comando para selecionar o total da tabela de treinamento (Figura 6.9).

```
insert into gpsql_result4766
select   6, 'Handheld' , C.Computador, C. KEYCOLUMN
from     Compra_training4766 C
where    not exists( select 1 from gpsql_result4766 where KEYCOLUMN = C.KEYCOLUMN)
```

FIGURA 6.7: INSERT PARA REGRA DEFAULT

```
select count(1) from gpsql_result4766 where class = classified
```

FIGURA 6.8: SQL PARA SELEÇÃO DO TOTAL CLASSIFICADO CORRETAMENTE

```
select count(1) from Compra_training4766
```

FIGURA 6.9: EXEMPLO DE SELEÇÃO DE TOTAL DA TABELA DE TREINAMENTO

6.4 Apresentação de Resultados

Durante e após a execução do módulo GGP vários arquivos são criados como resultado para o problema de computadores. Primeiramente o sistema cria os sub-diretórios para conter os resultados, na Tabela 6.11 existe um resumo dos sub-diretórios criados.

TABELA 6.11: SUB-DIRETÓRIOS CRIADOS PARA DATABASE COMPUTADOR

#	Sub-Diretório	Descrição
1	Results	Sub-diretório criado abaixo do diretório aonde se encontra o sistema
2	Results\computador.par	Segundo sub-diretório criado com o nome do SQL Parameter abaixo do sub-diretório #1
3	..\Run5Gen500GenE20Min3Max30Cross60Train60Y	Sub-diretório criado abaixo do #2.
4	..\Init2Fit3Tour3N	Sub-diretório criado abaixo do sub-diretório #3
5	..\Computador.bnf	Sub-diretório criado abaixo do #4 com o nome do arquivo BNF utilizado
6	..\Pop1000	Sub-diretório criado abaixo do #5 com o tamanho da população

Os arquivos de saída são criados abaixo do sub-diretórios criados. A lista dos arquivos está na Tabela 6.12. Os nomes dos arquivos começam por "Cros40Mut50_", que é um string formado pela taxa de crossover e mutação.

TABELA 6.12: EXEMPLO DE ARQUIVOS DE RESULTADOS

Nome do Arquivo
Cros40Mut50_allbest.txt
Cros40Mut50_analyse_computador.bnf
Cros40Mut50_computador.par
Cros40Mut50_computador.bnf
Cros40Mut50_config.gpm
Cros40Mut50_gpsql.bnf
Cros40Mut50_result.run
Cros40Mut50_result.txt
Cros40Mut50_result_best.txt

Um dos arquivos que traz as informações mais interessantes é o "Cros40Mut50_result.txt". Além de trazer os parâmetros utilizados, o arquivo contém um resumo dos resultados como mostra a Tabela 6.13.

O arquivo "Cros40Mut50_result.run" contém resultados detalhados de cada *run*, a Tabela 6.14 mostra algumas informações presentes no arquivo.

A cada *run*, também são selecionados os melhores indivíduos, com menor *fitness*, num formato para facilitar a sua compreensão e utilização, como mostra a Figura 6.4. A Figura 6.10 tem um exemplo do formato e as informações adicionais para o

classificador de melhor *fitness* do *run* 5. Estas informações estão contidas no arquivo "Cros40Mut50_result_best.txt".

TABELA 6.13: EXEMPLO RESULTADOS APRESENTADOS NO ARQUIVO

Parâmetro	Valor
Number of runs	5
Training Set	17
Testing Set	13
Last Generation	26
Fitness Minimum	0.0452941
Fitness Average	0.165835
Fitness Maximum	0.762941
Training Accuracy	96.4706 %
Testing Accuracy	90.7692 %
Training Acc - Best Test	96.4706 %
Testing Acc - Best Test	92.3077 %
Number Best Individuals	232
Rules Average	6
Depth Average	5.9458
Size Average	117.772
Number's Equal Individual	92
Time (hh:mm:ss)	0:37:20

TABELA 6.14: ARQUIVO CROS20MUT70_RESULT.RUN PARA DATABASE IRIS

Run	Generation	Training Accuracy	Testing Accuracy	Number Best	Size	BestCount	BestTest Train Accuracy	BestTest Test Accuracy
1	29	100,00	84,62	568	139.867	75	100,00	92,31
2	24	94,12	100,00	1	121.094	93	94,12	100,00
3	20	94,12	100,00	1	116.955	86	94,12	100,00
4	29	100,00	69,23	538	106.963	128	100,00	69,23
5	30	94,12	100,00	1	103.982	78	94,12	100,00

A análise do arquivo BNF se encontra em: "Cros40Mut50_analyse_computador.bnf". Na Figura 6.11, pode-se ver uma parte do conteúdo do arquivo. A primeira linha indica que todo indivíduo começará com o não terminal <classifier>; a segunda linha indica todos os nodos filhos do terminal <classifier>; a terceira linha indica que o não terminal <r> tem como opção uma das quatro produções para formação de seus nodos filhos. Como a seleção é feita de modo aleatório, cada produção tem 25% de chance de ser escolhida (esta porcentagem é indicada antes de cada produção).

Standardized Fitness = 0,0688235

Table: COMPRA_training4766

	Predict			
Columns	Desktop	Handheld	Laptop	Accuracy
Desktop	8	0	0	100 %
Handheld	0	4	0	100 %
Laptop	1	0	4	80 %

Overall accuracy: 94,1176 %

Table: COMPRA_testing4766

	Predict			
Columns	Desktop	Handheld	Laptop	Accuracy
Desktop	6	0	0	100 %
Handheld	0	3	0	100 %
Laptop	0	0	4	100 %

Overall accuracy: 100 %

FIGURA 6.10: SAÍDA PARA O CLASSIFICADOR DE MENOR FITNESS DO RUN 5

```
S -> <classifier>
<classifier> -> <r> <r> <r> <r> <r> S <goal>

<r> -> 4 Productions
        -> (1) 25% S <goal> W <cond>
        -> (1) 25% S <goal> W <cond> AND <cond>
        -> (1) 25% S <goal> W <cond> AND <cond> AND <cond>
        -> (1) 25% S <goal> W <cond> AND <cond> AND <cond> AND <cond>

<goal> -> 3 Productions
        -> (1) 33.33% 'Desktop'
        -> (1) 33.33% 'Handheld'
        -> (1) 33.33% 'Laptop'

<cond> -> 5 Productions
        -> (1) 20% P.GENERO = <value_P.GENERO>
        -> (1) 20% P.IDADE <rel_P.IDADE>
        -> (1) 20% RES.PAIS <Coperator> <value_RES.PAIS>
        -> (1) 20% COM.PAIS <Coperator> <value_COM.PAIS>
        -> (1) 20% C.MES <Coperator> <value_C.MES>

<value_P.GENERO> -> 2 Productions
        -> (1) 50% 'F'
        -> (1) 50% 'M'
```

FIGURA 6.11: PARTE DO ARQUIVO "CROS40MUT50_ANALYSE_COMPUTADOR.BNF"

6.5 Discussão do Capítulo

Foi apresentado uma execução do sistema GPSQL Miner com o objetivo de exemplificar suas principais funcionalidades. A apresentação foi feita com o exemplo do 'Computador' criado especialmente para mostrar as potencialidades do sistema.

Primeiramente, foram definidos os arquivos de entradas necessários para o problema: SQL Parameter (computador.par), Configuração para GGP (computador.gpm). Depois, foi indicado um exemplo de divisão dos registros da tabela principal nas tabelas de treinamento e de teste.

Após a criação das tabelas, foi exemplificada a avaliação das regras do classificador. Mostraram-se os passos para avaliação de cada regra e como o sistema calcula o *fitness* de cada indivíduo. E, finalmente, alguns resultados apresentados pelo sistema.

No próximo capítulo serão detalhados os experimentos realizados para validação do sistema.

7. Experimentos e Resultados Obtidos

O sistema cria automaticamente uma gramática com um formato padrão (Figura 5.7), conforme visto na Seção 5.3. Antes de serem apresentados os experimentos realizados, na Seção 7.1 é descrito a análise de como a forma geral para os arquivos BNF foi obtida. Na seção 7.2 é apresentado um critério para a comparação de resultados entre algoritmos de classificação. Na seção seguinte o sistema GPSQL Miner é comparado com o sistema similar LOGENPRO. E, por último, são apresentados os experimentos feitos para comparação com outros algoritmos de *Data Mining* baseados em Árvore de Decisão, Algoritmos Estatísticos e Redes Neurais.

7.1 Efeito da gramática

Estudos preliminares foram feitos com o propósito de compreender a influência da gramática na performance do sistema. Neste caso, foi verificado empiricamente que as seguintes modificações na gramática tiveram importância significativa:

Número de regras fixas para cada classificador

Umas das primeiras versões para a forma geral foram baseada no fato de que inicialmente não sabemos com exatidão o número de regras necessárias para que o sistema possa classificar corretamente os dados. Partindo deste princípio, foi criada uma forma geral para que a gramática desse gerar um número variável de regras conforme mostra a Figura 7.1.

Alguns testes foram conduzidos, mas notou-se que os classificadores gerados geralmente têm poucas regras. Descobriu-se que a causa era a valorização, pelo sistema,

dos menores classificadores, já que, em caso de empate de precisão o melhor classificador escolhido era o de menor tamanho ou o que tinha menos regras.

```
S          -> <classifier>
<classifier> -> <r> S <goal>
<r>        -> S <goal> W <cond>
<r>        -> S <goal> W <cond> <r>
```

FIGURA 7.1: GRAMÁTICA COM NÚMERO VARIÁVEL DE REGRAS

Com a finalidade de gerar bons classificadores, realizaram-se um conjunto de experimentos com uma gramática mais genérica, isto é, não era pré-definido nem o número de regras nem o número de condições de uma regra individual. Porém, esta gramática genérica não apresentou bons resultados levando ao estabelecimento de uma gramática mais específica. Testes empíricos demonstraram que o estabelecimento de um número fixo de regras produz resultados com mais precisão. Na Figura 7.2 apresenta-se a gramática para gerar um classificador com três regras.

O número de regras necessárias para cada base de dados não pode ser previsto de modo simples. Pode-se gerar uma gramática com um número grande de regras, como por exemplo 50. Porém, um número excessivo de regras para cada classificador implica no sistema dispender mais tempo desnecessário para avaliação da população. Um fato óbvio é: se a base de dados tiver CL classes então o classificador vai ter pelo menos CL regras. Outro fato verificado é que nos testes realizados foram necessárias pelo menos CO regras. Na qual, CL = Número de Classes e CO = número de colunas.

O sistema gera um arquivo BNF com um número fixo de regras para cada classificador. O número de regas geradas é: CL+CO regras. O sistema permite a alteração deste número de regras de modo simples, basta modificar o arquivo BNF que esta em formato texto.

```
S          -> <classifier>
<classifier> -> <r> <r> <r> S <goal>
<r>        -> S <goal> W <cond>
```

FIGURA 7.2: GRAMÁTICA COM NÚMERO FIXO DE REGRAS

Número fixo de condições

Uma condição de uma regra é formada por um atributo descritor, um operador, e um valor (Ex.: Idade = 'Adulta'). Uma gramática geral pode ser projetada de modo que as regras tenham um número variável de condições. Temos um exemplo na Figura 7.3, a primeira e a segunda linha são responsáveis por gerarem regras com várias condições, a terceira linha mostra o começo de uma produção para gerar a condição com a coluna 'Column1'. Entretanto, testes iniciais mostraram não ser essa uma boa solução.

```
<r> -> S <goal> W <cond>
<cond> -> <cond> AND <cond>
<cond> -> Column1 ...
```

FIGURA 7.3: REGRAS COM NÚMERO VARIÁVEL DE CONDIÇÕES

Experimentos empíricos preliminares demonstraram que o sistema pode obter um ganho significativo na eficiência se a gramática já incluir diferentes números de condições, como mostra a Figura 7.4. Neste exemplo, pode-se ter três diferentes regras, cada regra representada pelo não terminal <r> e a condição pelo não terminal <cond>. Na primeira linha da Figura 7.4, as regras geradas por esta produção terão apenas uma condição, a segunda linha com dois <cond> gerará regras com duas condições.

```
<r> -> S <goal> W <cond>
<r> -> S <goal> W <cond> AND <cond>
<r> -> S <goal> W <cond> AND <cond> AND <cond>
<cond> -> Column1 ...
```

FIGURA 7.4: REGRAS COM NUMERO FIXO DE CONDIÇÕES

Além destas modificações na gramática padrão, algumas restrições adicionais garantem que, na criação da população inicial, o atributo (coluna) apareça apenas uma única vez em cada regra. Esta verificação não permite criar condições como: "Column_A = 1 AND Column_A = 2", e assim gerando condições consistentes.

7.2 Critério de Comparação

Freqüentemente existe o interesse na comparação de performance entre dois algoritmos de aprendizado, contudo, não existe um consenso na comunidade de pesquisa de aprendizado de máquina sobre qual o teste mais apropriado para a comparação [Mitchell 1997]. Alguns métodos alternativos são descritos por [Dietterich 1996]. Porém, neste trabalho, é apresentado o critério de comparação descrito por [Mitchell 1997].

Para um melhor entendimento do método de comparação, a descrição do método é feito com um exemplo, maiores detalhes podem ser encontrados em [Mitchell 1997]. Suponha a comparação entre dois algoritmos: A1 e A2. Deseja-se saber qual o melhor algoritmo para o aprendizado de um classificador em um determinado banco de dados. Considerando que as taxa de erros foram estimadas com a utilização do *ten-fold cross-validation* (descrito na seção 7.4).

O algoritmo A1 obteve uma média de taxa de erro de 9,0% (representado por média(A1), ou seja, média(A1) = 9) para as 10 execuções e um desvio padrão igual a 1 (sd(A1) = 1). Já o algoritmo A2 obteve 7,5% (média(A2) = 7,5) como média de taxa de erro com um desvio de 0,8 (sd(A2) = 0,8).

É calculado a diferença entre as médias:

média(A1 – A2) = média(A1) – média(A2) = 9 – 7,5 = 1,5.

Calcula-se a diferença entre os desvios padrões através da fórmula:

sd(A1-A2)=raiz_quadrada((sd(A1)2+sd(A2)2)/2) = raiz_quadrada((1^2+0,8^2)/2) = raiz_quadrada(1,64/2) = raiz_quadrada(0,82) = 0,9055

E por último calcula-se o valor de ad(A1-A2) que é:

ad(A1-A2) = media(A1-A2)/sd(A1-A2)= 1,5 / 0,9055 = 1,6565

E o resultado é avaliado pela condição:

Se ad>0, A2 tem um desempenho melhor que A1,

Se ad>= 2, A2 tem um desempenho melhor que A1 com nível de confiança 95%

Devido à falta dos valores de desvio padrão dos algoritmos utilizados na comparações, este trabalho utilizará outros critérios de comparação descritos nas próximas seções.

7.3 Comparação com LOGENPRO

Experimentos foram conduzidos para verificar o comportamento do GPSQL Miner comparando o sistema similar LOGENPRO. Os resultados detalhados dos outros algoritmos encontram-se em [Wong 2000]. Para permitir a comparação foi utilizada a mesma metodologia empregada nos testes realizados em [Wong 2000].

Foram testados quatro conjuntos de dados diferentes (Íris – uma base de dados real, e três bases artificiais: Monk1, Monk2 e Monk3), extraídos de *UCI Machine Learning Repository* [Blake 1998], que mostram diversidade de tamanho, número de classes, e número e tipos de atributos. Estes conjuntos de dados estão em formato proposicional e foram importados para o banco de dados Oracle para utilização do GPSQL Miner.

O sistema foi executado 25 vezes para cada base de dados. Para cada *run*, os parâmetros foram mantidos constantes. Todos os parâmetros referentes a GGP e que foram utilizados pelo GPSQL Miner encontram-se na Tabela C.1 do Anexo C. Em cada *run* os dados são separados aleatoriamente em uma base de dados de treinamento e uma base de testes conforme um percentual estabelecido para cada base. Os classificadores são construídos na base de treinamento e, no final, a taxa de precisão do classificador é calculada utilizando os registros da base de testes.

Banco de dados Iris Plant

O database 'Iris' tem 3 classes: 'Iris-setosa', 'Iris-versicolor' e 'Iris-viginica' e 150 registros. Em cada rodada, 67% (100 registros) são escolhidos aleatoriamente para compor a base de treinamento e os 50 restantes formam a base de testes.

Os resultados de [Wong 2000] e deste experimento são resumidos na Tabela 7.1 em ordem crescente de precisão; nesta tabela pode-se verificar a precisão de alguns algoritmos. Comparado ao LOGENPRO, o sistema GPSQL Miner é um pouco mais preciso apresentando menos da metade da taxa de erro (ou 5,36% de diferença na precisão). Detalhes de outros sistemas, métodos e seus resultados podem ser encontrados em [Wong 2000].

TABELA 7.1: PRECISÃO PARA DATABASE IRIS

MÉTODO / SISTEMA	TAXA DE ERRO	PRECISÃO
LOGENPRO	8,96 %	91,04 %
C4.5	6,20 %	93,80 %
ID3	5,80 %	94,20 %
Nearest Neighbor	4,00 %	96,00 %
GPSQL Miner	3,60 %	96,40 %
Neural Net	3,30 %	96,70 %

Bancos de dados Monk

Outros experimentos foram conduzidos utilizando o banco de dados Monk [Thrun 1991]. O problema Monk é uma coleção de três problemas de classificação binária

utilizando seis atributos de domínio discreto. Existem três conjuntos de dados para este problema: Monk1, Monk2 e Monk3. Para os três conjuntos utilizados os dados foram divididos em 80% para a base de treinamento e 20% para a base de teste, esta configuração foi utilizada para comparar os resultados do GPSQL Miner com os resultados de [Wong 2000].

O conjunto Monk1 é composto por [Wong 2000]:

- Conjunto de treinamento: 124 exemplos, sendo 62 exemplos positivos e 62 exemplos negativos;
- Conjunto de teste: 432 exemplos, sendo 216 casos positivos e 216 exemplos negativos;
- Não há erro de classificação na base de dados. A base não apresenta ruídos.

A segunda coluna da Tabela 7.2 mostra o resultado dos algoritmos. Apesar do sistema GPSQL-Miner obter 0% de erro em todos os *run*'s na base de treinamento, na base de teste o sistema obteve 0.4%.

O conjunto Monk2 é composto por [Wong 2000]:

- Conjunto de treinamento: 169 exemplos, sendo 105 exemplos positivos e 64 exemplos negativos;
- Conjunto de teste: 332 exemplos, sendo 190 exemplos positivos e 142 negativos;
- Não há erro de classificação na base de dados. A base não apresenta ruídos.

A terceira coluna da Tabela 7.2 mostra os resultados dos algoritmos para a base de dados Monk2. O sistema tem 27.2% de taxa de erro para o conjunto de teste, contra 40% para o seu similar LOGENPRO.

Conjunto Monk3 tem [Wong 2000]:

- Conjunto de treinamento: 122 exemplos com 62 positivos e 60 negativos;
- Conjunto de teste: 432 exemplos, sendo 204 exemplos positivos e 228 exemplos negativo;.

- Existe 5% de erro de classificação (ruído) no conjunto de treinamento.

Tabela 7.2, quarta coluna, mostra o resultados dos algoritmos. Pode-se observar que nas bases de dados Monk2 e Monk3 o GPSQL Miner obteve resultados melhores do que o similar LOGENPRO [Wong 2000]. Enquanto na base Monk1 não observamos uma diferença significativa, uma vez que o LOGENPRO não obteve erro na classificação enquanto o GPSQL Miner obteve um erro de apenas 0,4%.

TABELA 7.2: TAXA DE ERRO DO PROBLEMA MONK (EM %)

MÉTODO / SISTEMA	MONK1	MONK2	MONK3
AQ15-GA	0,0	13,2	0,0
AQ17-DCI	0,0	0,0	5,8
AQR	4,1	20,3	13,0
Assistant Professional	0,0	18,7	0,0
Backpropagation	0,0	0,0	6,9
CN2	0,0	31,0	10,9
GPSQL Miner	**0,4**	**27,2**	**3,5**
ID3	1,4	32,1	5,6
LOGENPRO	**0,0**	**40,0**	**4,6**

Avaliação da Comparação com LOGENPRO

No intuito de estabelecer um critério de comparação entre os sistemas foi feita a média com as quatro taxas de precisão encontradas em cada base de dados. A partir das média de cada sistema (Tabela 7.3) foi calculada a diferença entre os mesmos que foi de 4,69%. Pelo método Tukey [Ruppert 1981] a diferença entre a média de taxa de erros de dois algoritmos pode ser considerada estatisticamente diferente no nível de 10% se a diferença for maior do que 5,8%. Portanto, apesar do sistema GPSQL Miner apresentar resultados aparentemente melhores, o mesmo não pode ser considerado estatisticamente mais preciso do que o LOGENPRO.

TABELA 7.3: PRECISÃO E TAXA DE ERRO

SISTEMA	PRECISÃO MÉDIA	TAXA DE ERRO MÉDIO
GPSQL Miner	91,33 %	8,67 %
LOGENPRO	86,61 %	13,39 %

7.4 Comparação com Diferentes Sistemas

Para uma avaliação mais efetiva do sistema não basta apenas comparar com um sistema similar. Por esta razão, o sistema GPSQL Miner foi comparado com os resultados de outros 33 algoritmos de classificação [Lim 1999]. Dentre os algoritmos, 22 algoritmos são baseados em árvores de decisão, 9 são algoritmos estatísticos e 2 algoritmos de redes neurais. Os algoritmos utilizados na comparação estão melhor definidos na próxima subseção.

As bases de dados testadas apresentam diversidade em termos de precisão de classificação, atributos, classes e outras características [Lim 1999]. Para cada base de dados uma variação foi criada com adicionamento de ruídos, ou seja, atributos foram criados com valores aleatórios. As bases de dados com ruídos foram marcadas com '+' em seus nomes.

Uma breve descrição dos dados e suas características são dadas na Tabela 7.4. Informações mais detalhadas e referências sobre a origem destes dados podem ser encontrados em [Lim 1999].

TABELA 7.4: DESCRIÇÃO SOBRE OS CONJUNTOS DE DADOS ANALISADOS

Nome	Tamanho	Classes	Atributos Contínuos	Atributos Discretos	Breve descrição do Domínio
BCW	683	2	9	0	Diagnóstico de Câncer no seio obtido da
BCW+	683	2	18	0	Universidade de Wisconsin
BLD	345	2	6	0	Diagnóstico de doenças de fígado em homens
BLD+	345	2	15	0	via testes de sangue e consumo de álcool
BOS	506	3	12	1	Análise do custo de vida nos subúrbios de
BOS+	506	3	24	1	Boston
HEA	270	2	7	6	Diagnóstico de doença cardíaca, dados obtidos
HEA+	270	2	17	6	de Cleveland Clinic Foundation
PID	532	2	7	0	Diagnóstico de diabetes de pacientes da Índia
PID+	532	2	15	0	Diagnóstico pelo estado psicológico + testes
SMO	1855	3	3	5	Previsão da atitude de pessoas diante de
SMO+	1855	3	10	5	Restrições a fumantes no local de trabalho.
TAE	151	3	1	4	Avaliação de professores assistentes, usando
TAE+	151	3	6	4	Como dados informações sobre o curso
VOT	435	2	0	16	Classificação de um político como republicano
VOT+	435	2	0	30	ou democrata de acordo com suas votações

Para permitir as comparações, a mesma metodologia de [Lim 1999] foi utilizada. Para as bases de dados a taxa de erro foi estimada usando *ten-fold cross-validation* [Lim 1999]. O conjunto de dados é dividido em dez subconjuntos disjuntos, cada um contendo aproximadamente o mesmo número de registros com aproximadamente a mesma proporção de registros para cada classe. Para cada subconjunto, um classificador é construído usando os registros que não estão dentro dele (isto é, os 9 sub-conjuntos restantes, sendo que o primeiro sub-conjunto é usado como teste). O classificador então é testado com o *withheld subset* para obter uma estimativa *cross-validation* para a taxa de erro. Esta taxa de erro é a média de todas as estimativas *cross-validation.*

Depois da cálculo da taxa de erro, cria-se uma tabela com os resultados. Nesta tabela cria-se uma linha para cada algoritmo, para cada base de dados uma colunas e na célula correspondente a média da taxa de erro. No final usa-se um método estatístico chamado '*randomized block design*' para analisar a tabela. The '*randomized block design*' é também chamado de '*two-way factorial design without replicates*' [Lim 1999]. Todos os parâmetros utilizados para cada base de dados estão no Anexo C.

Algoritmos usados nas comparações

Nesta seção existe uma breve descrição dos algoritmos que serão comparados com o GPSQL Miner. As referências indicadas contêm dados mais detalhados sobre cada algoritmo.

ÁRVORES DE DECISÃO:

- CART – Versão do algoritmo CART [Breiman 1984] tendo um critério especial de escolha de atributos.

- Árvore S-Plus – variante do método CART escrito na linguagem S [Becker 1988]. É descrita em detalhes em [Clark 1993].

- C4.5 – foi usada a versão 8 [Quinlan 1993] [Quinlan 1996], com valores padrão, gerando árvores de decisão e usando o programa do pacote para a geração de conjuntos de regras.

- FACT – é um algoritmo rápido de classificação, detalhado em [Loh 1988]. Duas variantes foram testadas conforme o critério de escolha dos atributos.

- QUEST – este algoritmo é descrito em [Loh 1997]. Quatro variações diferentes foram testadas, conforme o critério de poda e escolha dos atributos. A versão usada foi a 1.7.10.

- IND – este algoritmo é detalhado em [Buntine 1992]. A versão usada é a 2.1, com valores padrão. Quatro variações são analisadas.

- OC1 – Este algoritmo é detalhado em [Murthy 1994]. Três variações são analisadas. A versão usada nos testes foi a 3.

- LMDT – detalhado em [Brodley 1995], usando os valores padrão do programa.

- CAL5 – criado principalmente para valores numéricos, é detalhado em [Muller 1994] [Muller 1997]. Usou-se uma ferramenta do pacote para encontrar os valores ótimos.

- R1 – árvore de decisão baseada na escolha de apenas 1 atributo [Holte 1993].

ALGORITMOS ESTATÍSTICOS

- LDA – análise linear do discriminante. Foi usado o SAS PROC DISCRIM [SAS 1990] com valores padrão.

- QDA – análise quadrática do discriminante. Foi usado o SAS PROC DISCRIM [SAS 1990] com valores padrão.

- NN - implementação do método do vizinho mais próximo do pacote SAS PROC DISCRIM [SAS 1990].

- LDA – *Polytomous Logistic Regression* [Agresti 1990]. Análise logística do discriminante.

- FDA – análise flexível do discriminante [Hastie 1994]. Duas variações de [Friedman 1991] foram analisadas.

- PDA – trata-se de uma variante do algoritmo LDA mostrada em [Hastie 1995]

- MDA – análise mista do discriminante. Implementada usando S-Plus e a biblioteca `mda` [Hastie 1996].

- POL – este é o algoritmo POLYCLASS, detalhado em [Kooperberg 1997]. Indicado por [Lim 1999] como um dos melhores de todos os algoritmos analisados.

REDES NEURAIS

- LVQ – *Learning Vector Quantization*, explicado detalhadamente em [Kohonen 1995]

- RBF – *Radial Basis Function Network* implementada usando [Sarle 1994]. Mais detalhes sobre o algoritmo em [Bishop 1995] e [Ripley 1996].

Resultados obtidos

Os resultados obtidos são muito animadores se considerarmos as médias de taxas de erro. A Tabela 7.5 indica um resumo das médias de taxas de erro de todos os testes realizados com GPSQL Miner e os resultados obtidos de [Lim 1999]. Através da Tabela 7.5 podemos verificar a melhor (segunda coluna) e a pior média de taxa de erro (terceira coluna) alcançada pelos algoritmos. Se compararmos as médias das taxas de erro em cada

base de dados do sistema GPSQL Miner (quarta coluna), pode-se verificar que o sistema esta quase sempre próximo do melhor valor e em nenhum deles próximo do pior resultado. A última coluna indica o *rank* do sistema GPSQL Miner em cada base, o *rank* indica a posição do algoritmo em relação à média da taxa de erro em cada base e será melhor definido na seção "análise de *rank*".

Outro detalhe a ser observado é que, se considerarmos o tamanho da população (última coluna da Tabela 7.5) podemos observar que são populações relativamente pequenas e que poderiam ser aumentadas como uma forma de diminuir a taxa de erro em alguns casos.

TABELA 7.5: RESUMO DOS RESULTADOS PELA TAXA DE ERRO

Banco de dados	Min	Max	GPSQL Miner Taxa de Erro	Perto do Melhor ?	GPSQL Miner Rank
BCW	0,0278	0,0848	0,0306	**Sim**	2
BCW+	0,0293	0,076	0,0307	**Sim**	3
BLD	0,279	0,432	0,288	**Sim**	4
BLD+	0,284	0,441	0,284	**Sim**	1
BOS	0,221	0,314	0,235	**Sim**	6
BOS+	0,225	0,422	0,253		5
HEA	0,141	0,341	0,144	**Sim**	2
HEA+	0,148	0,311	0,155	**Sim**	2
PID	0,208	0,310	0,208	**Sim**	1
PID+	0,205	0,318	0,205	**Sim**	1
SMO	0,304	0,454	0,305	**Sim**	10
SMO+	0,3048	0,445	0,3048	**Sim**	8
TAE	0,325	0,693	0,378		5
TAE+	0,403	0,696	0,403	**Sim**	1
VOT	0,0364	0,0617	0,0364	**Sim**	1,5
VOT+	0,0409	0,0662	0,0409	**Sim**	1
Média			0,2063		3,3

O resumo dos resultados obtidos pelo sistema em cada uma das bases podem ser encontradas nas tabelas do Anexo D. Já os resultados detalhados de algoritmo em cada base de dados podem ser encontrados nas tabelas do Anexo F.

A seguir descreveremos algumas análises mais detalhadas dos resultados obtidos. Como uma forma de validá-los serão aplicadas algumas análises para os resultados usando os mesmos critérios apresentados por [Lim 1999].

INTERVALO DE CONFIANÇA

O intervalo de confiança simultânea para diferenças entre a média de taxa de erro pode ser obtido pelo método Tukey [Ruppert 1981]. Como foi visto anteriormente, de acordo com este método, uma diferença entre a média da taxa de erro de dois algoritmos é estatisticamente diferente ao nível de 10% se eles diferem mais do que 5,8% [Lim 1999].

Na Tabela 7.5, quarta coluna da última linha, pode-se ver a média da taxa de erro de 20,6% para o sistema GPSQL Miner. Através da Tabela E.1, que mostra a média da taxa de erro de todos os algoritmos por ordem crescente, pode-se observar que o sistema GPSQL Miner é estatisticamente diferente dos sistemas: FM2, CAL, NN, FTU, STO, FTL, ST1, T1, QDA e LVQ. Pois as médias destes métodos são maiores do que 26,4 (= 20,6 + 5,8).

ANÁLISE EXPLORATÓRIA PELA TAXA DE ERRO

Neste caso, as taxas de erro de cada algoritmo foram comparadas entre si. A comparação foi feita para verificar se o resultado obtido esta próximo do melhor ou se é o pior resultado.

Antes da análise estatística dos resultados é necessária a apresentação dos resultados da Tabela 7.6 (resultados dos algoritmos baseados em Árvore de Decisão ou regras) e da Tabela 7.7 (resultados dos Algoritmos Estatísticos e Redes Neurais). Os algoritmos (segunda linha nas duas tabelas) estão agrupados de acordo com a sua categoria (Ex.: Estatísticos, Redes Neurais e Árvores & Regras– indicados na primeira linha). As taxas de erro média para os algoritmos em todos as bases de dados estão na terceira linha (Ex.: para GPSQL Miner a taxa média é 0,203).

Vamos chamar de 'Mn' a menor taxa de erro observada para cada base de dados (cada linha equivale a uma base de dados nas Tabelas 7.6 e 7.7). Se um algoritmo tem uma taxa de erro dentro de um erro padrão de 'Mn', pode-se considerar que o algoritmo está perto do melhor, indicado pela letra 'M' nas Tabelas 7.6 e 7.7 (Ex.: O algoritmo C4R esteve próximo do melhor apenas para os testes da base de dados 'BLD').

O erro padrão para cada base de dados é calculado pela fórmula: $\sqrt{\frac{Mn*(1-Mn)}{n}}$. Onde 'n' é igual ao tamanho da base de treinamento, embora as taxas de erro sejam calculados nas bases de teste.

TABELA 7.6: RESULTADOS DE ANÁLISE PARA TAXA DE ERRO

	Tree & Rules																								
Database	C4R	C4T	CAL	FTL	FTU	**GPSQL Miner**	IB	IBO	ICO	IC1	IM	IMO	LMT	OCL	OCM	OCU	QLO	QL1	QUO	QU1	ST1	STO	T1
Média	0,248	0,242	0,261	0,265	0,262	**0,203**	0,253	0,238	0,237	0,253	0,245	0,246	0,248	0,273	0,244	0,253	0,232	0,235	0,249	0,256	0,269	0,27	0,276
#M	4	4	5	7	4	**14**	1	4	1	3	2	4	3	3	4	5	10	9	5	3	4	3	4
#P	0	0	0	1	0	**0**	0	0	0	0	0	0	0	3	1	0	0	0	0	0	1	0	3
BCW						**M**		M						P			M	M					
BCW+						**M**											M	M					P
BLD	M					**M**								M	M								P
BLD+						**M**																	P
BOS	M	M				**M**						M			M								
BOS+		M														M							
HEA				M		**M**							M				M	M					
HEA+				M		**M**							M				M	M					
PID			M	M		**M**								P			M	M	M				
PID+						**M**								P			M	M					
SMO	M	M	M	M	M	**M**	M				M	M		M	M	M	M	M	M		M	M	M
SMO+			M	M	M	**M**			M	M	M		M	M	M	M	M	M	M	M	M	M	M
TAE				P				M															
TAE+						**M**															P		
VOT			M	M	M	**M**		M		M		M				M	M		M	M	M		M
VOT+	M	M	M	M	M	**M**		M		M		M			P	M	M	M	M	M	M	M	M

Ainda das Tabelas 7.6 e 7.7, o algoritmo com a maior taxa de erro é indicado pelo 'P' (Ex.: O algoritmo com a pior taxa de erro para a base 'SMO+' foi o algoritmo 'NN'). O número de vezes que cada algoritmo esteve próximo do melhor esta na quarta linha e é representado pelo símbolo #M. Já o total de vezes que o algoritmo obteve o pior resultado encontra-se na quinta linha da coluna do mesmo e representado pelo símbolo #P.

TABELA 7.7: RESULTADOS DE ANÁLISE PARA TAXA DE ERRO

	Estatístico									Redes Neurais	
Database	FM1	FM2	LDA	LOG	MDA	NN	PDA	POL	QDA	LVQ	RBF
Média	0,249	0,268	0,224	0,228	0,231	0,27	0,232	0,232	0,284	0,291	0,237
#M	8	5	7	6	6	3	7	8	1	2	6
#P	0	2	0	0	1	1	0	0	0	3	0
BCW				M						M	M
BCW+	M	M		M	M					M	M
BLD	M	M						M			
BLD+								M			
BOS	M					M		M		P	M
BOS+	M	P						M			
HEA			M	M	M		M			P	
HEA+			M		M		M			P	
PID	M		M				M				
PID+			M					M			
SMO	M	P		M	M	M	M	M	M		M
SMO+		M	M	M	M	P	M	M			M
TAE						M					
TAE+					M						
VOT	M	M	M		P		M				
VOT+	M	M	M	M			M	M			M

Algumas conclusões podem ser extraídas das Tabelas 7.6 e 7.7:

- O sistema GPSQL Miner obteve a menor média para a taxa de erro. A ordem crescente dos algoritmos de acordo com a média da taxa de erro pode ser encontrada na Tabela E.1 do Anexo E.

- Os algoritmos também podem ser avaliados em função ao número total de marcas 'M' e 'P'. Por este critério, o sistema GPSQL Miner obteve quatorze vezes a marca 'M' e em nenhuma das bases o sistema obteve uma marca 'P'. Na Tabela E.3, Anexo E, pode-se ver o total de 'M' e 'P' em ordem para melhor análise dos algoritmos.

- Nove algoritmos estiveram entre os piores resultados, ou seja, tiveram um ou mais 'P'. A ordem decrescente do número de marcas 'P' (entre parênteses) é:

LVQ(3), OCL(3), T1(3), FM2(2), NN(1), ST1(1), OCM(1), MDA(1), FTL(1)

- A quantidade de 'P's pode indicar um algoritmo com performance inconstante, como é o caso do FM2. O algoritmo esteve perto do melhor em 6 bases de dados, mas obteve dois piores resultados.

- As bases de dados mais fáceis para a classificação são: BCW, BCW+, SMO, SMO+ e VOT+

Pela ordem da taxa de erro podemos concluir que o sistema GPSQL Miner ficou em primeiro lugar com a menor taxa. Considerando o número de vezes (#M) que o sistema ficou entre os melhores, o sistema também ficou em primeiro com dez, sendo que o segundo foi o algoritmo QL0 com oito. Os resultados são melhores observados na Tabela E.3.

ANÁLISE DE RANKS

O *rank* é avaliado para cada base de dados, onde o algoritmo com a menor taxa de erro é o primeiro do *rank*, o segundo com a menor taxa recebe o *rank* dois e assim por diante. Em caso de empate na taxa o *rank* de cada algoritmo será a média dos *ranks*. Nas tabelas do Anexo F podemos ver o *rank* de cada algoritmo separado por base de dados.

Depois de cada algoritmo receber o número do *rank* em cada base foi calculado uma média de todos os *rank* por algoritmo. Na Tabela 7.8, podem-se observar os primeiros algoritmos mostrados por ordem de *rank*. Na Tabela E.2 do Anexo E todos os algoritmos são listados por ordem de *rank*.

TABELA 7.8: OS PRIMEIROS ALGORITMOS POR ORDEM DE RANK

Método	Rank
GPSQL Miner	3,3438
QL0	11,0625
LOG	11,2188
LDA	11,3438
POL	11,7813
PDA	12,7813
FM1	12,8438

Mais uma vez pode-se observar que o sistema GPSQL Miner obteve a primeira posição com *rank* médio de 3,3438 (Tabela 7.8 ou Tabela E.2). O fato de um algoritmo ter uma baixa taxa de erro média indica que o classificador tem uma boa performance na maioria dos testes realizados [Lim 1999].

O número de 'P' (definido nas Tabelas 7.6 e 7.7) juntamente com a média de *rank* pode ser um indicador de instabilidade. Se o algoritmo tiver numa boa posição em relação à média dos *rank* (Tabela E.2) e em algum dos testes obter o pior resultado (#'P' maior do que zero nas Tabelas 7.6 e 7.7) indica que o algoritmo é instável, podendo obter bons resultados na maioria dos casos e péssimos resultados em alguns.

Teste de Friedman [Friedman 1937] é um procedimento padrão para testes estatísticos de significância na diferença de *ranks*. Segundo [Hollander 1999] a diferença nas médias dos *ranks* maior do que 8.7 % é estatisticamente significante ao nível de 10%. Em relação ao *rank*, o sistema GPSQL Miner não é diferente estatisticamente de apenas seis algoritmos: QL0, LOG, LDA, QL1 e PDA, pois possuem média menor do que 13,075 (soma de 4,375 com 8,7).

TEMPO DE EXECUÇÃO DOS SISTEMAS

O sistema GPSQL Miner é capaz de obter a média do tempo de execução para encontrar o melhor classificador de cada base de dados. Contudo, os tempos médios que foram obtidos não podem servir de comparação entre os sistemas devido à arquitetura dos mesmos. O sistema desenvolvido acessa diretamente o SGBD enquanto os outros algoritmos acessam arquivos em formato texto.

Um outro detalhe a ser observado, é que o sistema GPSQL Miner acessa as bases de dados que estão em um servidor. Para os testes, o sistema foi executado em máquinas clientes ligadas a uma rede. Algumas vezes, foi executado diretamente do servidor. Ou seja, temos algumas variáveis que não puderam ser controladas para se ter uma comparação real. Por exemplo, o servidor do banco de dados é também servidor de arquivos e servidor para controlar o acesso à rede. E muitas vezes nos clientes outros sistemas estavam concorrendo para a utilização dos recursos da máquina, além de poder ter outras instâncias do sistema.

Nas tabelas do Anexo D, podem-se ver as médias dos tempos para cada base de dados, juntamente com outros resultados. Acredita-se que o tempo médio de execução para cada base de dados seja menor do que se encontrou. Contudo, mesmo sendo resultados aproximados, pode-se concluir que o sistema não possui uma boa performance se considerado o tempo, podendo até ser um dos piores analisando apenas o tempo de treinamento.

7.5 Discussão do Capítulo

Neste capítulo foi apresentada uma discussão sobre a melhor forma de uma gramática.

Depois, foram mostrados dois tipos de comparações parar validar o sistema GPSQL Miner. Na primeira comparação, foram aplicados testes em quatro bases de dados e os resultados foram comparados com o sistema similar LOGENPRO. Os resultados mostraram que o sistema obteve uma média de taxa de erro um pouco superior ao sistema similar, contudo, não pode ser considerado estatisticamente melhor.

Na segunda comparação foram utilizadas 33 algoritmos e os resultados analisados pelo intervalo de confiança, taxa de erro, *rank* e o tempo de treinamento. Nos três primeiros critérios o GPSQL Miner obteve ótimos resultados, estando sempre entre os

melhores algoritmos, contudo, no último critério podemos dizer que o sistema obteve um resultado ruim.

No próximo capítulo, tem-se a conclusão deste trabalho.

8. Conclusões

Data mining é definido como um processo não trivial de identificar padrões válidos, novos, potencialmente úteis e compreensível em dados. Este trabalho apresenta GPSQL Miner, um sistema que usa o paradigma da GGP para a tarefa de classificação. Sendo uma proposta inicial para problemas de classificação em *Data Mining*.

Baseado nas informações fornecidas pelo usuário e os dados no SGBD, o sistema cria uma gramática adequada a cada domínio. Além de criar automaticamente o arquivo BNF, o sistema permite que usuário altere-o de acordo com a sua experiência fornecendo como auxílio o arquivo de estatísticas BNF. É importante ressaltar que a gramática deve ser específica para cada problema para obter resultados melhores, isto foi alcançado devido direto aos dados pelo SGBD.

A aproximação da GGP provê uma representação de conhecimento poderosa. Um ambiente de banco de dados pode ter muitas tabelas e procurar conceitos sem uma direção é uma tarefa difícil. GPSQL Miner é flexível e permite que o domínio de conhecimento seja facilmente definido e efetivamente utilizado para problemas de classificação. A utilização da gramática com SGBD facilita a automação do processo criando uma gramática de acordo com cada problema. Além de permitir que o usuário especifique informações como a relação entre as tabelas, o sistema procura automaticamente as classes e os tipos e valores das colunas.

Diversos experimentos foram realizados divididos em duas comparações. Na primeira comparação do GPSQL Miner foi utilizado o sistema LOGENPRO. LOGENPRO é um sistema similar que utiliza GGP para *Data Mining*. Os resultados mostraram que o sistema GPSQL Miner obteve uma média de taxa de erro um pouco superior, contudo, não pode ser considerado estatisticamente melhor.

Na segunda comparação foram utilizados 33 algoritmos e os resultados analisados considerando: intervalo de confiança, taxa de erro, *rank* e o tempo de treinamento. Nos três primeiros critérios o GPSQL Miner posiciona-se entre os melhores com ótimos resultados, contudo, quanto ao último critério pode-se dizer que o sistema não obteve bons resultados, mas não pode ser comparado com os demais pelos seguintes motivos: o acesso à base de dados é através do SGBD outros acessam diretamente o arquivo texto; o sistema executa em arquitetura cliente-servidor.

Foram encontrados alguns pontos fortes da ferramenta: a representação no formato de comandos SQL; a utilização do DBMS ajuda na criação de uma gramática adequada a cada problema, além de facilitar e assegurar a segurança dos dados; o uso de GP para a procura pelas soluções mais promissoras com a limitação do espaço de busca pela gramática mostrou ter um bom comportamento para *Data Mining*. A utilização de SGBD com a técnica GGP possibilita automação do processo.

Apesar do sistema não ter sido testado em muitas bases de dados, os resultados dos experimentos demonstram que a ferramenta tem um comportamento muito bom como um sistema de indução. Desta forma GPSQL Miner é indicado para aplicações que exigem precisão na classificação das informações. Acredita-se ser uma promissora ferramenta de *Data Mining*, embora necessite melhorar a sua performance em relação ao tempo. As principais sugestões de melhoramento e observações são:

A aplicação do sistema em um maior número de experimentos. Incluindo base de dados relacionais e/ou base com um número maior de registros.

O principal melhoramento na ferramenta refere-se à performance. Embora o sistema apresente boa precisão para a classificação dos problemas. O sistema demorou alguns dias para achar algumas soluções, sendo praticamente inviável a sua utilização para bases de dados maiores do que 5000 registros.

Exploração dos parâmetros do sistema tais como *crossover* e mutação, esta parte será facilitada com o uso dos arquivos de estatística. A pesquisa de quais são os melhores

parâmetros para cada tipo de problema e a implementação automática de parâmetros facilitariam o trabalho do usuário final uma vez que ele não tem a obrigação de conhecer GGP já que a aplicação do sistema é para uma tarefa de *Data Mining*.

Análise dos arquivos de estatísticas para melhorar pontos de *crossover*. Acredita-se que a escolha do ponto em que cada operação possa ser feita possibilite melhorar o tempo de conversão ou até, em alguns casos, melhorar a precisão.

Bibliografia

[Agrawal 1993] Agrawal, R; Imielinski, T.; Swami, A. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93), 1993, pp. 207-216.

[Agresti 1990] Agresti, A. Categorical data analysis. John Wiley & Sons, New York, NY, 1990.

[Banzhaf 1998] Banzhaf, W.; Nordin, P.; et al Genetic Programming ~ An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, 1998.

[Becker 1988] Becker, R. A.; Chambers, J. M.; Wilks, A. R. The new S language. Wadsworth, 1988.

[Bishop 1995] Bishop, C. M. Neural networks for pattern recognition. Oxford University Press, New York, NY, 1995.

[Blake 1998] Blake, C. L.; Merz, C.J. UCI Repository of Machine Learning Data Bases. Disponível em http://www.ics.uci.edu/~mlearn/MLRepository.html Irvine, CA: University of California, Departament of Information and Computer Science, 1998.

[Breiman 1984] Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. Classification and regression trees. Chapman and Hall, New York, NY, 1984.

[Brodley 1995] Brodley, C. E.; Utgoff, P. E. Multivariate decision trees. Machine Learning, 1995, pp. 19:45-77.

[Brown 1995] Brown, M.; Watson I.; Filer, N. Separating the cases from the data: towards more flexible case-based reasoning. Proc. 1st Int. Conf. On Case-Based Reasoning (ICCBR-95). LNAI 1010, 1995, pp. 157-169.

[Buntine 1992] BUNTINE, W. Learning classification trees. Statistics and Computing, 1992, pp. 2:63-73.

[Chen 1996] Chen, M. S.; Han, J.; Yu., S. Data Mining: Na Overview from Database Perspective. IEEE Transactions on Knowledge and Data Engineering, 1996, pp. 8:866-883.

[Clark 1993] Clark, L. A.; Pregibon, D. Tree-based models. In CHAMBERS, J. M. & HASTIE, T. J. eds. Statistical models in S. Chapman & Hall, New York, NY, 1993, pp. 377-419.

[Darwin 1859] Darwin, C. On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life. Murray, London, UK, 1859.

[Dietterich 1996] Dietterich, T.G. Proper statistical tests for comparing supervised classification learning algorithms (Technical Report). Department of Computer Science, Oregon State University, Corvallis, OR. 1996.

[Duarte 2001] Duarte, D. Utilizando Técnicas de Programação Lógica Indutiva para Mineração de Banco de Dados Relacional. Dissertação do curso de Mestrado em Informática. Universidade Federal do Paraná, 2001.

[Fayyad 1996] Fayyad, U. M.; Piatesky-Shapiro, G.; Smyth, P. From Data mining to Knowledge Discovery: An overview. AI Magazine, 1996, 17(3), pp.37-54.

[Freitas 1997] Freitas, A. A. A Genetic Programming Framework for two Data Mining Tasks: Classification and Generalized Rule Induction. 1997.

[Freitas 1997b] Freitas, A. A. Generic, set-oriented primitives to support data-parallel knowledge discovery in relational database systems. Ph.D. Thesis. University of Essex, UK. 1997.

[Freitas 1998] Freitas, A. A.; Lavington, S. H. Mining Very Large Database with Parallel Processing. Kluwer Academic Publishers, 1998.

[Friedman 1937] Friedman, M. The use of rans to avoid the assumption of normality implicity in the analysis of variance, Journal of the American Statistical Association, 1937, pp. 32:675-701.

[Friedman 1991] Friedman, J. Multivariate adaptive regression splines (with discussion). Annals of Statistics, 1991, pp. 19:1-141.

[Gustafson 1986] Gustafson, D.; Barrett, W.; Bates, R.; Couch, J. D. Compiler Construction: Theory and Practice. Science Research Assoc., 1986.

[Gritz 1999] Gritz, L. I. Evolutionary Controller Synthesis for 3-D Character Animation. PhD Thesis. Department of Electrical Engineering and Computer Science, The George Washington University, 1999.

[Han 1996] Han, J.; Fu, Y.; Wang, W. et al. DBMiner: A system for mining knowledge in large relational databases. Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining. AAAI Press, 1996, pp. 250-255.

[Hasse 2000] Hasse, M. Mineração de Dados usando Algoritmos Genéticos. Dissertação do curso de Mestrado em Informática. Universidade Federal do Paraná, 2000, pp. 6-7.

[Hastie 1994] Hastie, T.; Buja, A.; Tibshirani, R. Flexible discriminant analysis by optimal scoring. Journal of the American Statistical Association., 1994, 89:1255-1270.

[Hastie 1995] Hastie, T.; Buja, A.; Tibshirani, R. Penalized discriminant analysis. Annals of Statistics, 1995, pp. 23:73-102.

[Hastie 1996] Hastie, T., Tibshirani, R. Discriminant analysis by gaussian mixtures. Journal of the Royal Statistical Society, Series B, 1996, pp. 58:155-176.

[Holland 1992] Holland, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press. Second Edition: MIT Press, 1992.

[Hollander 1999] Hollander, M. e Wolfe, D. A. Nonparametric Statistical Methods. John Wiley & Sons, New York, NY, 2nd edition, 1999.

[Holte 1993] Holte, R. C. Very simple classification rules perform well on most commonly used datasets. Machine Learning, 1993, pp. 11:63-90.

[Imielinski 1996] Imielinski, T.; Virmani, A.; Abdulghani, A. DataMine: application programming interface and query language for database mining. Proc. 2nd Int. Conf. Knowledge Discovery & Data Mining. AAAI Press, 1996, pp. 256-261.

[Ishida 1999] Ishida, C. Y.; Cavalheiro, A. F. BUSINESS INTELLIGENCE Expandindo os Benefícios do ERP. Monografia do curso de especialização MBA em Gestão Empresarial Integrada Através de Sistemas de ERP. Pontifícia Universidade Católica do Paraná, 1999.

[Kohonen 1995] Kohonen, T. Self-organizing maps. Springer-Verlag, Heidelberg, 1995.

[Kooperberg 1997] Kooperberg, C. , BOSE, S. & STONE, C. J. Polychotomous regression. Journal of the American Statistical Association, 1997, pp. 92:117-127.

[Koza 1992] Koza, J. R. Genetic Programming: on the Programming of Computers by Computers by Means of Natural Selection. Cambridge, MA: MIT Press. 1992, pp.72.

[Koza 1994] Koza, J. R. Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge, MA: MIT Press, 1994.

[Lim 1999] Lim, T.-S.; Loh, W.-Y.; Shih, Y.-S. A comparison of prediction accuracy, complexity and training time of 33 old and new classification algorithms in Machine Learning Journal. Kluwer Academic, Boston, 1999.

[Loh 1997] Loh, W.-Y.; Shih, Y.-S. Split selection methods for classification trees. Statistica Sinica, 1997, pp. 7: 815-840.

[Loh 1988] Loh, W.-Y.; Vanichsetakul, N. Tree-structured classification via generalized discriminant analysis (with discussion). Journal of the American Statistical Association, 1988, pp. 83: 715-728.

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

[Mitchell 1997] Mitchell, T.M. Machine Learning. WCB/McGraw-Hill, 1997, pp. 145-152.

[Muller 1994] Muller, W.; Wysotzki, F. Automatic construction of decision trees for classification. Annals of Operations Research, 1994, pp. 52:231-247.

[Muller 1997] Muller, W.; Wysotzki, F. The decision tree algorithm CAL5 based on a statistical approach to its splitting algorithm. In G. Nakhaezadeh & C. C. Taylor, eds. Machine Learning and Statistics: the interface. John Wiley & Sons, New York, NY, 1997, pp. 45-65.

[Murthy 1994] Murthy, S. K.; Kasif, S.; Salzberg, S. A system for induction of oblique decision trees. Journal of Artificial Intelligence Research, 1994, pp. 2:1-33.

[Quinlan 1993] Quinlan, J. R. C4.5: programs for machine learning. San Mateo. Morgan Kaufmann, 1993, pp. 302.

[Quinlan 1996] Quinlan, J. R. Improved use of continuous attributes in C4.5. Journal of Artificial Intelligence Research, 1996, pp. 4:77-90.

[O'Leare 1995] O'Leare, D.E. Some privacy issues in knowledge discovery: the OECD Personal Privacy Guidelines. IEEE Expert. Apr 1995, 10(2), pp. 45-52.

[Ramos 1999] Ramos, P. G. Uma Investigação das Redes Neuro-Fuzzy aplicadas à Mineração de Dados. Trabalho de graduação. Universidade Federal de Pernambuco. 30 de julho de 1999.

[Ripley 1996] RIPLEY, R. D. Pattern recognition and neural networks. Cambridge University Press, Cambridge, 1996.

[Ruppert 1981] Ruppert, G. M. Simultaneous Satatistical Inference. Springer-Verlag, New York, 2nd edition, 1981.

[Sarle 1994] Sarle, W. S. Neural networks and statistical models. In Proceedings of the Nineteenth Annual SAS Users Groups International Conference, Cary, NC, 1994, pp. 1539-1550.

[SAS 1990] SAS Institute Inc. SAS/STAT user's guide, version 6. Volume 1 & 2. SAS Institute Inc, Cary, NC, 1990.

[Shen 1993] Shen W. M. Bayesian probability theory – A general method for machine learning MCCCarnot-101-93. Microelectronics and Computer Technology Corporation, Austin, TX, 1993.

[Teller 1995] Teller, A.; Veloso, M. Program Evolution for Data Mining. The International Journal of Expert Systems, 1995, pp. 8(3):216-236.

[Thrun 1991] Thrun, S. B.; and et al. The Monk's Problems: A Performance Comparison of Different Learning Algoritmos. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[Weiss 1991] S.M. Weiss and C.A. Kulikowski. Computer Systems that Learn. Morgan Kaufmann, 1991.

[Whigham 1995a] Whigham, P. A. Grammatically-based genetic programming. In Rosca, J. P., editor, Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, CA, 1995, pp. 33-41.

[Whigham 1995b] Whigham, P. A. Inductive bias and genetic programming. In Zalzala, A. M. S., editor, First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA, Sheffield, UK. IEE, London, Uk, 1995, pp. 414:461-466.

[Whigham 1996] Whigham, P. A. Grammatical Bias for Evolutionary Learning. PhD thesis. School of Computer Science, University of New South Wales, Australian Defense Force Academy, 1996.

[Wong 2000] Wong, M. L., Leung, K. S. Data mining using grammar based genetic programming and aplications. Kluwer Academic Publishers, 2000.

[Zytkow 1993] Zytkow, J.M. Cognitive autonomy in machine discovery. Machine Learning, 12(1-3), Aug 1993, pp. 7-16.

Anexo A – Tabela Auxiliar

Tabela de Resultado: GPSQL_resultID

É uma tabela de resultados criada para conter os registros classificados da tabela de treinamento e a classe indicada pela classificador. A Tabela A.1 contém as colunas, seus tipos e uma breve descrição.

TABELA A.1: COLUNAS DA TABELA DE RESULTADOS

Coluna	Tipo	Descrição
Rule	Number(10)	Indica qual a regra que classificou o registro
Classified	char(15)	Classe indicada na regra
Class	char(15)	Classe original do registro
Keycolumn	Number(10)	Coluna auxiliar para o sistema não inserir duas vezes o mesmo registro

Anexo B – Definição dos parâmetros

TABELA B.1: DESCRIÇÃO DOS PARÂMETROS PARA GGP

Number_of_runs	Número de vezes que o sistema executara as configurações gerais
Number_of_generations	Número máximo de gerações
Generations_equal	Se o sistema alcançar o número máximo de gerações sem que o fitness tenha melhorado o mesmo cancela a tentativa de conseguir obter uma nova solução
Population_size	Tamanho da população
Tournament_size	Número de indivíduos utilizados para o torneio
Initialization_method	*Full*
Minimum_tree_height	Tamanho mínimo de uma árvore. O indivíduo que for criado com tamanho menor do informado é descartado e um novo indivíduo é criado
Maximum_tree_height	Tamanho máximo de uma árvore. O indivíduo que for criado com tamanho maior do informado é descartado e um novo indivíduo é criado
Maximum_crossover_height	Tamanho máximo que um indivíduo pode ter após uma operação de *crossover*
Crossover_rate	Taxa de *Crossover*
Mutation_rate	Taxa de Mutação
Type_Fitness	1 = 1 – *accuracy* 2 = 1.01 – *accuracy*
Elitist_strategy	Y – Usar estratégia elitista N – Não usar estratégia elitista
Training_Set	Porcentagem de registros que serão utilizados para o conjunto de treinamento
Separate_Train_class	Y – mantém o porcentagem de ocorrência de cada classe N – não mantém o porcentagem de ocorrência de cada classe
Interval_Fitness	Intervalo de *fitness* para mostrar os resultados
BNF_file	Nome do arquivo BNF que será utilizado. Se for informado gpsql.bnf o sistema usa o arquivo criado pelo GPSQLMiner
SQL_Parameter_file	Nome do arquivo SQL Parameter
Precision	O menor valor de *fitness* para que o sistema pare a busca pelo melhor indivíduo.
Save_Population	Criar ou não um arquivo (runR_genG.txt) para cada geração com todos os indivíduos e seus *fitness*
Save_Crossover_file	Criar ou não um arquivo para cada geração com os detalhes dos pontos de *crossover*
Crossover_Estatistic	Criar ou não um arquivo para cada geração com estatísticas dos pontos de *crossover*
Mutation_Estatistic	Criar ou não um arquivo para cada geração com estatísticas dos pontos de mutação
Population_size_(begin)	Tamanho inicial da População que o sistema deve executar os R *runs*
Population_size_(+)	Incremento no Tamanho da População
Population_size_(end)	Tamanho final da População que o sistema deve executar os R *runs*
Tournament_size_(BEGIN)	Tamanho inicial do Torneio que o sistema deve executar os R *runs*
Tournament_size_(+)	Incremento no Torneio
Tournament_size_(end)	Tamanho final do Torneio que o sistema deve executar os R *runs*
Initialization_method_(B)	Tamanho inicial do Método de inicialização que o sistema deve executar os R *runs*
Initialization_method_(+)	Incremento no Método de inicialização

Initialization_method_(E)	Tamanho final do Método de inicialização que o sistema deve executar os R runs
Crossover_rate_(begin)	Tamanho inicial da taxa de *crossover* que o sistema deve executar os R *runs*
Crossover_rate_(+)	Incremento da taxa de *crossover*
Crossover_rate_(end)	Tamanho final da taxa de *crossover* que o sistema deve executar os R *runs*
Mutation_rate_(begin)	Tamanho inicial da taxa de mutação que o sistema deve executar os R *runs*
Mutation_rate_(+)	Incremento na taxa de mutação
Mutation_rate_(end)	Tamanho final da taxa de mutação que o sistema deve executar os R *runs*
Type_Fitness_(BEGIN)	Tamanho inicial do Tipo de *Fitness* que o sistema deve executar os R *runs*
Type_Fitness_(+)	Incremento no Tipo de *Fitness*
Type_Fitness_(end)	Tamanho final do Tipo de *Fitness* que o sistema deve executar os R *runs*

Anexo C – Parâmetros dos Testes

TABELA C.1: PARAMETROS (*.GPM)

	IRIS	Monk1	Monk 2	Monk 3
Number of runs	25	25	25	25
Number of generations	50	50	400	50
Generations equal	20	20	30	10
Population size	300	300	80	500
Tournament size	13	3	2	13
Initialization method	2	2	2	2
Minimum tree height	3	3	3	3
Maximum tree height	30	30	30	30
Maximum crossover height	60	60	60	60
Crossover rate	50	50	10	40
Mutation rate	40	40	80	50
Type Fitness	3	1	1	1
Elitist strategy (Y/N/B)	Y	Y	Y	Y
Training Set	67	80	80	80
Separate Train class(Y/N)	N	Y	Y	Y
Interval Fitness	0.01	0.01	0.01	0.01
BNF file	iris.bnf	monk1.bnf	monk2b.bnf	monk3.bnf
SQL Parameter file	iris.par	monk1.par	monk2.par	monk3.par
Precision	0.001	0.001	0.001	0.01
Save Population (Y/N)	N	N	N	N
Analyze Population (Y/N)	Y	Y	Y	Y
Save Crossover file (Y/N)	N	N	N	N
Operators Statistic (Y/N)	Y	Y	Y	Y
Crossover Statistic (Y/N)	Y	Y	Y	Y
Mutation Statistic (Y/N)	Y	Y	Y	Y
Output File (Y/N)	Y	Y	Y	Y

TABELA C.2: OUTROS PARAMETROS (*.GPM)

	Bcw	bcw+	Bld	Bld+	hea	hea+
Number of runs	10	10	10	10	10	10
Number of generations	400	400	400	200	400	400
Generations equal	60	60	10	20	60	60
Population size	100	100	80	300	100	100
Tournament size	2	2	2	2	2	2
Initialization method	2	2	2	2	2	2
Minimum tree height	3	3	3	3	3	3
Maximum tree height	30	30	30	30	30	30
Maximum crossover height	60	60	60	60	60	60
Crossover rate	20	20	20	30	20	20
Mutation rate	70	70	70	60	70	70
Type Fitness	3	3	1	1	3	3
Elitist strategy (Y/N/B)	Y	Y	N	N	Y	Y
Training Set	90	90	90	90	90	90
Separate Train class(Y/N)	Y	Y	Y	Y	Y	Y
Interval Fitness	0.01	0.01	0.01	0.01	0.01	0.01
BNF file (*.bnf)	bcw2	bcw_noise3	liver3	liver_noise2	Heart	heart_noise2
SQL Parameter file(*.par)	Bcw	bcw_noise	Liver	liver_noise	heart	Heart_noise
Precision	0.001	0.001	0.001	0.001	0.001	0.001
Save Population (Y/N)	N	N	N	N	N	N
Analyze Population (Y/N)	Y	Y	Y	Y	Y	Y
Save Crossover file (Y/N)	N	N	N	N	N	N
Operators Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Crossover Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Mutation Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Output File (Y/N)	Y	Y	Y	Y	Y	Y

TABELA C.3: OUTROS PARAMETROS (*.GPM)

	Smo	Smo+	Tae	Tae+	Vot	Vot+
Number of runs	10	10	10	10	10	10
Number of generations	200	200	400	200	100	100
Generations equal	30	10	40	20	30	30
Population size	300	40	40	300	300	300
Tournament size	3	2	2	3	3	3
Initialization method	2	2	2	2	2	2
Minimum tree height	3	3	3	3	3	3
Maximum tree height	30	30	30	30	30	30
Maximum crossover height	60	60	60	60	60	60
Crossover rate	40	10	10	40	40	40
Mutation rate	50	80	80	50	50	50
Type Fitness	1	1	3	1	1	1
Elitist strategy (Y/N/B)	Y	Y	Y	Y	Y	Y
Training Set	90	90	90	90	90	90
Separate Train class(Y/N)	Y	Y	Y	Y	Y	Y
Interval Fitness	0.01	0.01	0.01	0.01	0.01	0.01
BNF file (*.bnf)	smoke	Smoke_noise	Ta	ta_noise	Voting	voting_noise
SQL Parameter file (*.par)	smoke	Smoke_noise	Ta	ta_noise	voting	voting_noise
Precision	0.001	0.001	0.001	0.001	0.001	0.001
Save Population (Y/N)	N	N	N	N	N	N
Analyze Population (Y/N)	Y	Y	Y	Y	Y	Y
Save Crossover file (Y/N)	N	N	N	N	N	N
Operators Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Crossover Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Mutation Statistic (Y/N)	Y	Y	Y	Y	Y	Y
Output File (Y/N)	Y	Y	Y	Y	Y	Y

TABELA C.4: OUTROS PARAMETROS (*.GPM)

	Bos	Bos+	Pid	Pid+
Number of runs	10	10	10	10
Number of generations	200	200	400	200
Generations equal	10	10	13	10
Population size	400	500	300	300
Tournament size	2	2	2	2
Initialization method	2	2	2	2
Minimum tree height	3	3	3	3
Maximum tree height	30	30	30	30
Maximum crossover height	60	60	60	60
Crossover rate	40	50	40	40
Mutation rate	50	40	50	50
Type Fitness	1	1	1	1
Elitist strategy (Y/N/B)	N	N	N	N
Training Set	90	90	90	90
Separate Train class(Y/N)	Y	Y	Y	Y
Interval Fitness	0.01	0.01	0.01	0.01
BNF file (*.bnf)	house2	house_noise2	pima2	pima_noise
SQL Parameter file (*.par)	house	house_noise	pima	Pima_noise
Precision	0.001	0.001	0.001	0.001
Save Population (Y/N)	N	N	N	N
Analyze Population (Y/N)	Y	Y	Y	Y
Save Crossover file (Y/N)	N	N	N	N
Operators Statistic (Y/N)	Y	Y	Y	Y
Crossover Statistic (Y/N)	Y	Y	Y	Y
Mutation Statistic (Y/N)	Y	Y	Y	Y
Output File (Y/N)	Y	Y	Y	Y

Anexo D – Resultados do sistema GPSQL Miner

TABELA D.1: TABELA DE RESULTADOS

	IRIS	Monk 1	Monk 2	Monk 3
Last Generation	40	10	182	18
Fitness Minimum	0.036	0	0.168284	0.055082
Fitness Average	0.108761	0.179003	0.196654	0.114551
Fitness Maximum	0.7768	0.783548	0.372544	0.613771
Training Accuracy	97.96 %	100 %	83.1716 %	94.6557 %
Testing Accuracy	96 %	99.6111 %	72.1204 %	96.5463 %
Training Acc - Best Test	97.68 %		80.6154 %	
Testing Acc - Best Test	96.4 %	99.71296 %	72.8055 %	97.3796 %
Number Best Individuals	158	1	25	150
Rules Average	8	5	19	12
Depth Average	6	6	6	7.80608
Size Average	173.851	118.653	451.879	393.644
Number's Equal Individual	11	42	4	19
Time (hh:mm:ss)	0:43:19	0:16:42	2:37:36	0:51:13

TABELA D.2: TABELA DE RESULTADOS 2

	BCW	BCW+	BLD	BLD+	Heart	Heart +
Last Generation	226	237	92	165	335	355
Fitness Minimum	0.0257803	0.0332632	0.293763	0.238674	0.103827	0.111235
Fitness Average	0.047513	0.0526759	0.306053	0.257991	0.121436	0.125403
Fitness Maximum	0.59422	0.506224	0.43832	0.503277	0.397243	0.338395
Training Accuracy	98.422 %	97.6737 %	70.6237 %	76.4869 %	90.6173 %	89.8765 %
Testing Accuracy	96.2013 %	95.1719 %	61.2857 %	67.2381 %	81.1111 %	80.3704 %
Training Acc - Best Test	98.1455 %	96.76217 %	66.684 %	73.8438 %	88.3128 %	87.73664 %
Testing Acc - Best Test	96.9366 % (3,0634)	96.93038 % (3,06962)	67.9524 % (32,0476)	71.5714 % (28,4286)	85.5556 % (14,4444)	84.44446 % (15,55554)
Number Best Individuals	45	29	30	15	46	42
Rules Average	11	12	18.9	13	15	15
Depth Average	6	6	6	6.41467	6	6.034
Size Average	379.916	393.798	594.857	481.56	570.751	
Number's Equal Individual	2	2	1	7	3	
Time (hh:mm:ss)	15:7:20	16:3:24	0:57:56	8:26:44	17:45:49	

TABELA D.3: TABELA DE RESULTADOS 3

	Smoking	Smoking+	Tae	Tae+	Vot	Vot+
Last Generation	68	22	167	104	64	79
Fitness Minimum	0.301294	0.303827	0.358028	0.306852	0.0316651	0.0341839
Fitness Average	0.303591	0.306329	0.375588	0.337683	0.0452367	0.0449426
Fitness Maximum	0.599407	0.389811	0.528807	0.66963	0.658372	0.623818
Training Accuracy	69.8706 %	69.6173 %	65.1972 %	69.3148 %	96.8335 %	96.5816 %
Testing Accuracy	69.42 %	69.51 %	54.875 %	55.0417 %	96.3636 %	95.9091 %
Training Acc - Best Test	69.6281 %	69.6119 %	60.27069 %	65.1923 %	96.8335 %	96.5816 %
Testing Acc - Best Test	69.48 %	69.52 %	62.166 %	59.6660%	96.3636 %	95.9091 %
	(30,52)	(30,48)	(37,834)	(40,334)	(3,6364)	(4,0909)
Number Best Individuals	274	37	22	105	160	160
Rules Average	36	28	23	11	16	16
Depth Average	6	6.7025	6	6.14367	5	5
Size Average	1197.88	993.753	508.785	261.19	924.979	857.224
Number's Equal Individual	6	1	1	19	21	23
Time (hh:mm:ss)	43:59:48	1:3:19	1:8:30	7:0:30	4:48:21	6:40:14

TABELA D.4: TABELA DE RESULTADOS 4

	Bos	Bos+	Pid	Pid+
Last Generation	83	110	82	32
Fitness Minimum	0.258006	0.24816	0.189929	0.214524
Fitness Average	0.27743	0.261331	0.207566	0.236405
Fitness Maximum	0.610076	0.604908	0.557869	0.657441
Training Accuracy	74.1994 %	75.184 %	80.6418 %	78.1149 %
Testing Accuracy	74.2929 %	71.5478 %	76.6184 %	75.7568 %
Training Acc - Best Test	73.6944 %	73.7986 %	79.5968 %	77.1735 %
Testing Acc - Best Test	76.4998 %	74.7184 %	79.2431 %	77.7939 %
	(23,5002)	(25,2816)	(20,7569)	(22,2061)
Number Best Individuals	107	168	30	16
Rules Average	12	14	15	15
Depth Average	6.2185	6	6.302	6.25733
Size Average	522.279	727.063	499.052	539.707
Number's Equal Individual	11	17	5	5
Time (hh:mm:ss)	10:38:42	15:38:8	4:39:29	3:4:22

Anexo E – Resultados por Algoritmo

TABELA E.1: RESULTADOS POR ORDEM DE TAXA DE ERRO

Método	Erro	Rank	Categoria
GPSQL Miner	0,206452788	3,34375	Tree & Rules
LDA	0,22399375	11,34375	Estatístico
LOG	0,2279625	11,21875	Estatístico
MDA	0,23106875	16,375	Estatístico
QL0	0,2317375	11,0625	Tree & Rules
POL	0,23185625	11,78125	Estatístico
PDA	0,23215625	12,78125	Estatístico
QL1	0,234775	12,96875	Tree & Rules
RBF	0,23728125	15,875	Redes Neurais
IC0	0,23733125	17,8125	Tree & Rules
IB0	0,23783125	15,9375	Tree & Rules
C4T	0,24206875	15,5625	Tree & Rules
OCM	0,24396875	18,90625	Tree & Rules
IM	0,244525	16,96875	Tree & Rules
IM0	0,24564375	16,625	Tree & Rules
C4R	0,24751875	15,9375	Tree & Rules
LMT	0,24775625	19,375	Tree & Rules
FM1	0,2487375	12,84375	Estatístico
QU0	0,24893125	15,90625	Tree & Rules
IB	0,252525	22,4375	Tree & Rules
OCU	0,25254375	18,21875	Tree & Rules
IC1	0,25294375	18,34375	Tree & Rules
QU1	0,25608125	19,3125	Tree & Rules
CAL	0,260925	22,625	Tree & Rules
FTU	0,2615375	20,65625	Tree & Rules
FTL	0,26523125	15,78125	Tree & Rules
FM2	0,26795625	17,40625	Estatístico
ST1	0,2689625	19,5625	Tree & Rules
NN	0,2701875	25,28125	Estatístico
STO	0,27040625	21,25	Tree & Rules
OCL	0,27280625	24,59375	Tree & Rules
T1	0,2764375	24,6875	Tree & Rules
QDA	0,28376875	26,9375	Estatístico
LVQ	0,2914875	25,28125	Redes Neurais

TABELA E.2: RESULTADOS POR ORDEM DE RANK

Método	Erro	Rank	Categoria
GPSQL Miner	0,206452788	3,34375	Tree & Rules
QL0	0,2317375	11,0625	Tree & Rules
LOG	0,2279625	11,21875	Estatístico
LDA	0,22399375	11,34375	Estatístico
POL	0,23185625	11,78125	Estatístico
PDA	0,23215625	12,78125	Estatístico
FM1	0,2487375	12,84375	Estatístico
QL1	0,234775	12,96875	Tree & Rules
C4T	0,24206875	15,5625	Tree & Rules
FTL	0,26523125	15,78125	Tree & Rules
RBF	0,23728125	15,875	Redes Neurais
QU0	0,24893125	15,90625	Tree & Rules
C4R	0,24751875	15,9375	Tree & Rules
IB0	0,23783125	15,9375	Tree & Rules
MDA	0,23106875	16,375	Estatístico
IM0	0,24564375	16,625	Tree & Rules
IM	0,244525	16,96875	Tree & Rules
FM2	0,26795625	17,40625	Estatístico
IC0	0,23733125	17,8125	Tree & Rules
OCU	0,25254375	18,21875	Tree & Rules
IC1	0,25294375	18,34375	Tree & Rules
OCM	0,24396875	18,90625	Tree & Rules
QU1	0,25608125	19,3125	Tree & Rules
LMT	0,24775625	19,375	Tree & Rules
ST1	0,2689625	19,5625	Tree & Rules
FTU	0,2615375	20,65625	Tree & Rules
STO	0,27040625	21,25	Tree & Rules
IB	0,252525	22,4375	Tree & Rules
CAL	0,260925	22,625	Tree & Rules
OCL	0,27280625	24,59375	Tree & Rules
T1	0,2764375	24,6875	Tree & Rules
NN	0,2701875	25,28125	Estatístico
LVQ	0,2914875	25,28125	Redes Neurais
QDA	0,28376875	26,9375	Estatístico

TABELA E.3: RESULTADOS POR ORDEM DE #P E #M

Algoritmo	#M	#P
GPSQL Miner	14	0
QL0	10	0
FM1	9	0
QL1	9	0
PDA	8	0
POL	8	0
LDA	7	0
LOG	7	0
RBF	6	0
QU0	6	0
C4R	5	0
CAL	5	0
OCU	5	0
C4T	4	0
FTU	4	0
IB0	4	0
IM0	4	0
QU1	4	0
IC1	3	0
LMT	3	0
STO	3	0
IC0	2	0
IM	2	0
QDA	1	0
IB	1	0
FTL	8	1
MDA	7	1
OCM	4	1
ST1	4	1
NN	3	1
FM2	6	2
T1	4	3
OCL	3	3
LVQ	2	3

Anexo F – Resultados por Database

TABELA F.1: RESULTADOS PARA BCW E BLD

	BCW		BCW+		BLD		BLD+	
Métodos	Erro	Rank	Erro	Rank	Erro	Rank	Erro	Rank
				Tree & Rules				
QU0	0,038	11	0,0425	16,5	0,389	28	0,403	30
QU1	0,0454	25	0,0469	20	0,399	29	0,376	26
QL0	0,0308	3,5	0,0293	1,5	0,306	8	0,38	27
QL1	0,0308	3,5	0,0293	1,5	0,32	15	0,369	24
FTU	0,0498	29,5	0,0498	23	0,401	30,5	0,402	29
FTL	0,0439	23	0,0424	15	0,413	32	0,419	31
C4T	0,0425	22	0,0513	25	0,308	9	0,329	10
C4R	0,0395	14,5	0,0395	13	0,292	6	0,32	5,5
IB	0,0424	20	0,06	32	0,328	22	0,34	13,5
IB0	0,0336	5	0,0425	16,5	0,322	17,5	0,333	11
IM	0,0424	20	0,0453	18	0,32	15	0,34	13,5
IM0	0,038	11	0,0409	14	0,312	12	0,336	12
IC0	0,0453	24	0,0542	28	0,327	21	0,327	8
IC1	0,0468	26	0,0541	27	0,319	13	0,313	3
OCU	0,0423	18	0,0584	30	0,35	26	0,347	20
OCL	0,0848	34	0,0526	26	0,296	7	0,328	9
OCM	0,0408	16	0,0585	31	0,279	1	0,367	23
STO	0,0512	31	0,0483	21	0,311	11	0,326	7
ST1	0,0498	29,5	0,0512	24	0,326	19,5	0,351	21
LMT	0,0351	8	0,0468	19	0,322	17,5	0,362	22
CAL	0,0706	32	0,0658	33	0,42	33	0,398	28
T1	0,076	33	0,076	34	0,432	34	0,441	34
GPSQL Miner	**0,030634**	**2**	**0,0306962**	**3**	**0,288095**	**4**	**0,28429**	**1**
				Estatísticos				
LDA	0,0394	13	0,0379	10	0,326	19,5	0,343	16
QDA	0,0481	27	0,0496	22	0,401	30,5	0,435	33
NN	0,0482	28	0,0555	29	0,37	27	0,423	32
LOG	0,0337	6,5	0,0352	9	0,309	10	0,344	18
FM1	0,0366	9	0,035	7,5	0,289	5	0,314	4
FM2	0,038	11	0,0321	4	0,28	2	0,32	5,5
PDA	0,0395	14,5	0,038	11	0,329	23,5	0,343	16
MDA	0,0409	17	0,035	7,5	0,32	15	0,374	25
POL	0,0424	20	0,0383	12	0,286	3	0,286	2
				Redes Neurais				
LVQ	0,0278	1	0,0336	6	0,329	23,5	0,343	16
RBF	0,0337	6,5	0,0322	5	0,33	25	0,346	19

TABELA F.2: RESULTADOS PARA HEA E SMO

	HEART		HEART+		SMOKE		SMOKE+	
Métodos	Erro	Rank	Erro	Rank	Erro	Rank	Erro	Rank
				Tree & Rules				
QU0	0,226	24,5	0,248	28	0,389	28	0,403	30
QU1	0,244	29	0,256	30	0,399	29	0,376	26
QL0	0,152	5,5	0,17	7	0,306	8	0,38	27
QL1	0,152	5,5	0,167	5	0,32	15	0,369	24
FTU	0,233	27,5	0,23	23	0,401	30,5	0,402	29
FTL	0,148	3,5	0,148	1	0,413	32	0,419	31
C4T	0,196	14,5	0,193	12	0,308	9	0,329	10
C4R	0,2	16,5	0,2	15	0,292	6	0,32	5,5
IB	0,222	22,5	0,226	21,5	0,328	22	0,34	13,5
IB0	0,196	14,5	0,189	11	0,322	17,5	0,333	11
IM	0,2	16,5	0,196	13,5	0,32	15	0,34	13,5
IM0	0,204	18	0,211	19	0,312	12	0,336	12
IC0	0,207	19	0,207	17	0,327	21	0,327	8
IC1	0,219	20,5	0,244	27	0,319	13	0,313	3
OCU	0,23	26	0,241	26	0,35	26	0,347	20
OCL	0,189	11	0,226	21,5	0,296	7	0,328	9
OCM	0,222	22,5	0,207	17	0,279	1	0,367	23
STO	0,256	31	0,233	24	0,311	11	0,326	7
ST1	0,233	27,5	0,215	20	0,326	19,5	0,351	21
LMT	0,163	8,5	0,17	7	0,322	17,5	0,362	22
CAL	0,27	32,5	0,256	30	0,42	33	0,398	28
T1	0,27	32,5	0,27	33	0,432	34	0,441	34
GPSQL Miner	**0,144444**	**2**	**0,1555554**	**2**	**0,288095**	**4**	**0,28429**	**1**
				Estatísticos				
LDA	0,141	1	0,156	3	0,326	19,5	0,343	16
QDA	0,248	30	0,259	32	0,401	30,5	0,435	33
NN	0,226	24,5	0,256	30	0,37	27	0,423	32
LOG	0,159	7	0,185	10	0,309	10	0,344	18
FM1	0,193	12,5	0,196	13,5	0,289	5	0,314	4
FM2	0,219	20,5	0,237	25	0,28	2	0,32	5,5
PDA	0,148	3,5	0,163	4	0,329	23,5	0,343	16
MDA	0,163	8,5	0,17	7	0,32	15	0,374	25
POL	0,174	10	0,207	17	0,286	3	0,286	2
				Redes Neurais				
LVQ	0,341	34	0,311	34	0,329	23,5	0,343	16
RBF	0,193	12,5	0,174	9	0,33	25	0,346	19

TABELA F.3: RESULTADOS PARA TAE E VOT

	TAE		TAE+		VOT		VOT+	
Métodos	Erro	Rank	Erro	Rank	Erro	Rank	Erro	Rank
				Tree & Rules				
QU0	0,43	9	0,516	15,5	0,0412	5	0,0412	2
QU1	0,49	17,5	0,516	15,5	0,0435	10	0,0435	6
QL0	0,444	11	0,503	10	0,0364	1,5	0,0503	22
QL1	0,47	15,5	0,51	12,5	0,0503	24,5	0,048	18,5
FTU	0,538	23,5	0,555	23	0,0435	10	0,0435	6
FTL	0,693	34	0,681	32	0,0457	15,5	0,0457	12,5
C4T	0,503	20	0,502	9	0,048	18,5	0,0503	22
C4R	0,583	27	0,529	18	0,0526	29	0,0457	12,5
IB	0,373	4	0,451	4	0,0526	29	0,0554	28
IB0	0,325	1	0,47	7	0,0386	4	0,0506	24
IM	0,538	23,5	0,53	19	0,0503	24,5	0,0594	30
IM0	0,492	19	0,549	22	0,0367	3	0,0457	12,5
IC0	0,372	3	0,543	21	0,048	18,5	0,0548	27
IC1	0,537	22	0,582	26	0,0435	10	0,0457	12,5
OCU	0,451	13	0,65	30	0,0435	10	0,0435	6
OCL	0,59	28	0,596	27	0,0574	32	0,0651	33
OCM	0,418	8	0,563	24	0,058	33	0,0662	34
STO	0,669	31,5	0,682	33	0,05	22	0,05	20
ST1	0,675	33	0,696	34	0,0432	6,5	0,0432	3
LMT	0,47	15,5	0,573	25	0,0483	20	0,0529	25
CAL	0,439	10	0,51	12,5	0,0457	15,5	0,0457	12,5
T1	0,54	25	0,536	20	0,0435	10	0,0435	6
GPSQL Miner	**0,37834**	**5**	**0,40334**	**1**	**0,036364**	**1,5**	**0,04091**	**1**
				Estatísticos				
LDA	0,411	7	0,45	3	0,0458	17	0,0458	16
QDA	0,543	26	0,511	14	0,0549	31	0,0617	32
NN	0,349	2	0,46	6	0,0526	29	0,0577	29
LOG	0,45	12	0,458	5	0,05	22	0,0435	6
FM1	0,636	30	0,628	28	0,0455	13,5	0,0457	12,5
FM2	0,669	31,5	0,642	29	0,0432	6,5	0,048	18,5
PDA	0,49	17,5	0,478	8	0,0455	13,5	0,0455	9
MDA	0,404	6	0,445	2	0,0617	34	0,0545	26
POL	0,53	21	0,518	17	0,0523	26,5	0,0477	17
				Redes Neurais				
LVQ	0,628	29	0,669	31	0,05	22	0,0614	31
RBF	0,464	14	0,504	11	0,0523	26,5	0,0503	22

TABELA F.4: RESULTADOS PARA BOS E PID

	BOS		BOS+		PID		PID+	
Métodos	Erro	Rank	Erro	Rank	Erro	Rank	Erro	Rank
				Tree & Rules				
QU0	0,267	28	0,275	18,5	0,226	8	0,23	12,5
QU1	0,273	31	0,294	27,5	0,23	12	0,23	12,5
QL0	0,268	29	0,282	23,5	0,225	6	0,221	3,5
QL1	0,274	32	0,282	23,5	0,223	5	0,221	3,5
FTU	0,261	25	0,263	11	0,247	25,5	0,258	26
FTL	0,254	19,5	0,254	6	0,221	2,5	0,225	7
C4T	0,221	1	0,225	1	0,242	22	0,252	22
C4R	0,236	7	0,261	9,5	0,227	10	0,233	14,5
IB	0,251	17,5	0,274	17	0,252	31	0,278	32
IB0	0,249	15,5	0,259	8	0,258	32	0,259	27
IM	0,241	10,5	0,258	7	0,233	15	0,246	18
IM0	0,231	5	0,273	15,5	0,246	24	0,265	29
IC0	0,254	19,5	0,264	12	0,237	17,5	0,233	14,5
IC1	0,266	26,5	0,28	22	0,239	21	0,248	19
OCU	0,241	10,5	0,233	2	0,237	17,5	0,256	24,5
OCL	0,272	30	0,358	33	0,31	34	0,318	34
OCM	0,239	8,5	0,267	13	0,247	25,5	0,261	28
STO	0,259	23,5	0,275	18,5	0,237	17,5	0,269	31
ST1	0,255	21	0,261	9,5	0,25	29,5	0,244	17
LMT	0,251	17,5	0,312	30	0,249	28	0,253	23
CAL	0,259	23,5	0,293	26	0,226	8	0,25	20,5
T1	0,287	33	0,287	25	0,25	29,5	0,25	20,5
GPSQL Miner	**0,235002**	**6**	**0,252816**	**5**	**0,207569**	**1**	**0,20539**	**1**
				Estatísticos				
LDA	0,249	15,5	0,279	21	0,221	2,5	0,223	5
QDA	0,266	26,5	0,273	15,5	0,238	20	0,256	24,5
NN	0,227	4	0,335	32	0,295	33	0,313	33
LOG	0,243	12,5	0,268	14	0,23	12	0,226	8
FM1	0,239	8,5	0,243	4	0,222	4	0,224	6
FM2	0,243	12,5	0,422	34	0,248	27	0,228	10
PDA	0,247	14	0,278	20	0,226	8	0,228	10
MDA	0,257	22	0,294	27,5	0,231	14	0,228	10
POL	0,225	2,5	0,239	3	0,237	17,5	0,217	2
				Redes Neurais				
LVQ	0,314	34	0,322	31	0,243	23	0,267	30
RBF	0,225	2,5	0,298	29	0,23	12	0,24	16