RAFAEL DA VEIGA CABRAL

PROGRAMAÇÃO GENÉTICA BASEADA EM ÁRVORES PARA CLASSIFICAÇÃO COM UMA CLASSE COM ÊNFASE NA GERAÇÃO DE ANOMALIAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Eduardo J. Spinosa

CURITIBA

2011

Cabral, Rafael da Veiga
Programação genética baseada em árvores para classificação com uma classe com enfase na geração de anomalias / Rafael da Veiga Cabral. – Curitiba, 2011.
94f. : il., tabs.

Impresso.
Dissertação (mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Eduardo Jaques Spinosa

1. Programação genética (Computação). 2. Algoritmos geneticos. I. Spinosa, Eduardo Jaques. II. Título.

CDD: 006.31

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Rafael da Veiga Cabral, avaliamos o trabalho intitulado, *"PROGRAMAÇÃO GENÉTICA BASEADA EM ÁRVORES PARA CLASSIFICAÇÃO COM UMA CLASSE COM ÊNFASE NA GERAÇÃO DE ANOMALIAS"*, cuja defesa foi realizada no dia 29 de agosto de 2011, às 14:00 horas, no Departamento de informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 29 de agosto de 2011.

Prof. Dr. Eduardo Jaques Spinosa
DINF/UFPR – Orientador

Profa. Dra. Myriam Regattieri De Biase da Silva Delgado
UTFPR– Membro Externo

Profa. Dra. Aurora Trindad Ramirez Pozo
DINF/UFPR– Membro Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

Agradeço primeiramente a **Deus** por ter cumprido mais uma importante etapa em minha vida e, principalmente aos meus pais Orlando Onofre Cabral e Erica da Veiga Cabral, pois sem eles esta vitória não seria possível. Ao meu orientador Eduardo J. Spinosa pela orientação deste trabalho, compreensão e, principalmente por ensinar-me a ver o mundo sob o olhar científico. À minha primeira orientadora professora Aurora Pozo, por acreditar em mim, pelas orientações iniciais e por proporcionar-me à orientação do professor Eduardo. À minha querida esposa Talita Wajczyk pela colaboração com este trabalho e pelo apoio e compreensão. Também agradeço a András Bánhalmi pelo auxílio na reprodução dos resultados de sua pesquisa. Por fim, dedico este trabalho de mestrado aos meus avós Leonel Cabral *in memoriam*, Mafalda Maria Cabral, Oliveira Gabriel da Veiga *in memoriam* e Rosa Valentina da Veiga e em especial, à minha avó Rosa pela presença na minha defesa.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

Acc *Accuracy* - taxa geral de acerto na classificação de exemplos

ANOVA *Analysis of Variance*

AUC *Area Under the ROC Curve*

BBA *Balanced Block Algorithm*

BNN *Bottleneck Neural Network*

Curv Parâmetro que estabelece a curvatura das anomalias em relação a classe normal

CSD *Class Separation Distance*

dF Fator multiplicador do raio da hiperesfera que envolve a classe normal usado para remover anomalias

dR Fator multiplicador do raio da hiperesfera que envolve a classe normal usado para gerar anomalias

Dist Parâmetro que estabelece a distância das anomalias em relação a classe normal

DBA2 *Distribution Based Artificial Anomaly*

DPLGP *Dynamic Page-based Linear GP*

DSS *Dynamic Subset Selection*

GMM *Gaussian Mixture Model*

HNIS *Hybrid Neuro-Immune System*

LMF *Local Membership Function*

LVQ *Learning Vector Quantization*

MLP *Multi Layer Perceptron*

OCC Classificação com Uma Classe (*One-Class Classification*)

OCGP *One-Class Genetic Programming*

OCGPC *One-Class Genetic Programming Clustering*

PCA *Principal Component Analysis*

PG Programação Genética

PGout Valor de retorno da função discriminante que compõe o classificador quando aplicada aos valores dos atributos do exemplo

RNS *Real-Valued Negative Selection*

ROC *Receiver Operation Characteristics*

SOM *Self-Organizing Maps*

SVDD *Support Vector Data Description*

SVM *Support Vector Machine*

TreeOCGP Abordagem de PG proposta no presente trabalho para resolução do problema OCC

TNr *True Negative rate* - taxa de acerto para a classe anormal

TPr *True Positive rate* - taxa de acerto para a classe normal

VN Verdadeiro Negativo - número de exemplos da classe de anormal classificados corretamente

VP Verdadeiro Positivo - número de exemplos da classe de normal classificados corretamente

WMW Wilcoxon-Mann-Whitney

RESUMO

A PG (Programação Genética) é aplicada com sucesso em Classificação. Entretanto, a pesquisa voltada à aplicação de PG para OCC (*One-Class Classification*) encontra-se em estágios iniciais, pois os poucos trabalhos relacionados existentes estão repletos de mudanças que não foram individualmente avaliadas e cujos propósitos estão em resolver problemas provenientes do método de geração de anomalias empregado e também na redução do tempo computacional da etapa de treino.

Nesse contexto, é notável que para tornar a PG em algoritmo de excelência para OCC o primeiro passo é avaliar sua abordagem convencional para o problema, algo que ainda não foi realizado, e objetivo central do presente trabalho, pois a introdução de novas ideias somente se justifica ao se conhecer as limitações e os resultados obtidos pelo algoritmo convencional. Contudo, a aplicação de PG para OCC requer que o problema seja transformado em classificação binária, cujas duas classes que compõem o conjunto de dados de treino são compostas por exemplos de perfil normal e anormal. Porém, em diversos problemas de OCC é impraticável obter exemplos anormais, por isto neste trabalho enfatiza-se a avaliação de algoritmos para gerar exemplos anormais, algo que também ainda não foi realizado para PG.

Entre os algoritmos de geração de anomalia estudados, selecionou-se o método proposto por Bánhalmi et al. [6] que baseia-se em exemplos mais distantes da classe normal, o algoritmo baseado no conceito de hiperesfera proposto por Tax e Duin [38] e a técnica RNS (*Real-valued Negative Selection*) inspirada em sistemas imunológicos proposta por Gonzáles et al. [19] [20]. Um estudo comparativo entre eles foi realizado, para avaliar o desempenho de classificação obtido por um classificador induzido por PG convencional sob uma abordagem de classificação binária. Verificou-se que o método proposto por Bánhalmi et al. possibilitou a obtenção dos melhores resultados.

O algoritmo de geração de anomalias com melhor avaliação foi empregado aos demais experimentos do presente trabalho, entre eles, um estudo comparativo entre a PG proposta no presente trabalho e as abordagens da literatura para OCC. Nesse experimento, verificou-se que a PG proposta neste trabalho obteve melhores resultados de classificação em dois problemas OCC, em outros três obteve desempenho similar e em um deles foi inferior. Portanto, atesta-se a hipótese de que é possível resolver OCC usando um algoritmo de PG convencional utilizando o algoritmo adequado para geração de anomalias.

O impacto de certos parâmetros da PG também foi avaliado. Entre eles o tamanho da população, que apresentou maior impacto no desempenho de classificação em um problema OCC comparado a diferentes ajustes no tamanho da árvore e na taxa de mutação. Além disso, diferentes funções de aptidão também foram experimentadas. Verificou-se que a função composta pela média das taxas individuais de acerto em cada classe apresentou melhor desempenho de classificação OCC quando comparada ao uso da métrica AUC (*Area Under the Receiver Operation Characteristic Curve*). A taxa WMW (*Wilcoxon-Mann-Whitney*), considerada um estimador da AUC com custo computacional inferior, também foi aplicada como função de aptidão e apresentou resultado semelhante ao uso da AUC.

ABSTRACT

Genetic Programming (GP) has been successfully applied to classification problems. However, there are only a few works related to applying GP for *One-Class Classification* (OCC) and these approaches introduce several changes in order to cope with issues related to the anomaly generation approach employed and also for decreasing the GP training time. In addition to that, the impact of each of these features was not properly assessed yet.

In this context, it is notable that for turning GP into leading approach for OCC a first step is needed, which is to assess the standard GP algorithm, something that has not yet been done and is the main objective in the present work. Thus, introducing new ideas is acceptable whenever results and limitations of the standard approach is known. Nevertheless, in order to solve OCC by GP, the problem must be transformed to binary classification with a training dataset composed of two classes: one representing normal samples and another composed of anomaly samples. However, since it is impractical to obtain anomaly samples for many OCC problems, the present work emphasizes the evaluation of anomaly samples generation algorithms, something that has also not been done for GP yet.

Among the anomaly detection methods studied, three were selected for comparison: the algorithm proposed by Bánhalmi et al. [6], based on normal class distribution boundary, the algorithm based on the hypersphere concept proposed by Tax and Duin [38] and the Real-valued Negative Selection (RNS) technique inspired by immune systems and proposed by Gonzáles et al. [19] [20]. The classification performance obtained by applying these techniques with GP for OCC was compared and the best results were achieved using the method based on the normal class distribution proposed by Bánhalmi et al, which was also employed for the other experiments in the present work.

Experiments were performed in order to compare the standard GP approach proposed in this work with the state of the art GP algorithm for OCC. According to our results, the proposed technique performed better in two OCC problems and achieved similar results in three others, performing worse in only one OCC problem. Therefore, the present work demonstrates that the hypothesis that a standard GP approach with an appropriated anomaly generation method can achieve competitive results in solving the OCC problem is true.

The influence of some GP configuration parameters was also assessed. The population size had a stronger impact on the classification performance in one OCC problem than the tree size and the mutation rate. Moreover, different GP fitness functions were also evaluated. The function composed of individual class classification rates achieved the best OCC classification performance when compared to the *Area Under the Receiver Operation Characteristic Curve* (AUC). The *Wilcoxon-Mann-Whitney* (WMW) metric, which is considered to be an AUC estimator with lower computational cost, was also evaluated as fitness function and its results were similar to those of the AUC.

CAPÍTULO 1

INTRODUÇÃO

A inteligência humana requer a habilidade de Classificação, seja para reconhecer pessoas, lugares, situações, objetos etc. Aprendizado de Máquina é uma das áreas da Inteligência Artificial cujo desafio é desenvolver técnicas para que o computador realize, entre outras tarefas, classificação eficiente por meio de aprendizado indutivo. Neste caso, um conjunto de dados composto por diferentes classes de exemplos é utilizado em um processo iterativo chamado treinamento, no qual o algoritmo indutivo desenvolve as regras necessárias para identificar exemplos não usados na fase de treino. Ademais, dotar o computador com habilidades cognitivas possibilita a resolução de problemas de classificação que estão além das habilidades humanas, uma vez que é necessário extrair conhecimento de conjuntos de dados complexos e extensos [28].

As aplicações de classificação nas quais o uso de um sistema computacional é necessário, incluem por exemplo, a detecção de uso fraudulento em redes de computador com a finalidade de identificar tentativas de intrusão ou uso indevido [15] [35]. Utilizar Classificação para distinguir o comportamento normal do falho em máquinas é outro exemplo [36]. De fato essas são instâncias de um problema específico chamado Classificação com Uma Classe (*One-Class Classification*) (OCC). A característica que diferencia esse tipo de problema de classificação é a presença de exemplos de apenas uma das classes para realizar indução do classificador, geralmente aqueles que representam o comportamento normal.

Para obtenção de exemplos do comportamento anormal em problemas reais, seria necessário danificar o sistema ou induzi-lo ao comportamento falho, o que inviabilizaria o uso de técnicas de classificação convencionais. Segundo Tax [36], há algoritmos que realizam aprendizado apenas sob o conjunto normal, de modo a construir regras que consideram como anormais aqueles exemplos que não se assemelham à classe conhecida.

Outra estratégia é gerar exemplos artificiais que diferem do padrão normal para treinar o classificador. Essa abordagem viabiliza a aplicação de algoritmos de classificação convencionais como a Programação Genética (PG), uma vez que o problema de OCC é transformado em um problema de classificação binário.

Observa-se que a PG é aplicada com sucesso em problemas de classificação multiclasse. De acordo com Espejo et al. [14], entre as características que destacam a PG para esse tipo de problema de Aprendizado de Máquina de outros tipos de algoritmos estão: a seleção natural de atributos do conjunto de dados mais relevantes durante o processo evolucionário, a possibilidade de uso de diversos modelos de representação da solução no bloco evolutivo da PG, o controle da generalização usando o número de gerações e o fato de que o resultado final é uma população de classificadores que podem ser usados em conjunto para prover mais confiabilidade ao processo de classificação.

Apesar dos benefícios, há apenas dois trabalhos relacionados à PG para OCC que utilizam a abordagem de geração de anomalias para treinar o classificador no contexto de classificação binária, ambos de mesma autoria [10] [11]. Nesses trabalhos, várias mudanças em relação ao algoritmo padrão de PG foram propostas, algumas com objetivo de resolver problemas causados pelo método de geração de anomalias, tais como a mistura de anomalias artificiais com exemplos da classe normal e o desequilíbrio entre o número de exemplos de ambas as classes no conjunto de dados de treino. Outros artefatos utilizados estão relacionados com a redução do tempo computacional da etapa de treino, entre os quais o uso da PG linear e uma técnica baseada no conceito de memória hierárquica para centrar o treinamento da PG em amostras do conjunto original de treino consideradas mais relevantes [10].

A diversidade das mudanças propostas e a ausência de outros trabalhos comparativos dificultam a avaliação dos resultados, méritos e limitações da atual abordagem com relação ao algoritmo padrão de PG para problemas OCC em que alcançar melhor generalização é mais crucial que o tempo de treinamento. Neste contexto, o presente trabalho visa, entre outros, ao estudo e à experimentação de abordagens de geração de anomalias que possibilitam a indução de um classificador PG padrão e livre de mudanças cujos indivíduos

da população são representados na forma de árvore, conforme o algoritmo de PG de propósito geral proposto por Koza [23].

Nesse ponto destaca-se uma importante contribuição do presente trabalho para elaboração de estratégias de resolução de problemas OCC, pois não existem estudos comparativos entre as técnicas de geração de anomalias existentes e deste modo se estabelece uma nova área de pesquisa. O estudo e desenvolvimento desses métodos é também fundamental, pois viabiliza aplicação de algoritmos gerais de classificação binária como a PG, principalmente em casos práticos em que não é possível obter exemplos reais de comportamento anormal.

Também é objeto de estudo do presente trabalho avaliar determinados ajustes de parâmetros da PG e funções de aptidão que podem influenciar o desempenho final de classificação, outra importante contribuição, pois não há estudos relacionados que explorem aspectos internos da PG quando aplicada a problemas OCC. Por fim, realiza-se um estudo comparativo entre a abordagem proposta e o trabalho que representa o estado da arte na resolução de problemas OCC utilizando PG e outros tipos de classificadores.

No Capítulo 2 introduz-se o conceito de classificação e apresenta-se o problema de OCC. Na Seção 2.3 são apresentados algoritmos para gerar anomalias usadas para treinar classificadores a fim de resolver o problema na forma de classificação binária. Na Seção 2.4 mencionam-se exemplos de aplicações reais de OCC. No Capítulo 3, introduz-se a PG e apresenta-se sua aplicação para classificação geral, sendo descritas, na Seção 3.2.1, a maneira como a PG identifica exemplos das classes e, na Seção 3.2.2, as funções de aptidão utilizadas. Na Seção 3.3 há uma revisão dos principais trabalhos envolvendo PG para OCC. No Capítulo 4, discute-se a proposta de trabalho, suas motivações e, no Capítulo 5, a metodologia, os resultados e a análise dos experimentos realizados. No Capítulo 6, por fim, apresentam-se as considerações finais e os trabalhos futuros.

CAPÍTULO 2

CLASSIFICAÇÃO COM UMA CLASSE

2.1 Introdução

O aprendizado indutivo caracteriza uma das formas da inteligência humana, pois viabiliza a formação de conceitos a partir de experiências vivenciadas no passado e no presente para que possa ser feita a aplicação do conhecimento de maneira inteligente no futuro. Dotar o computador com capacidades cognitivas como o aprendizado indutivo é objetivo de Aprendizado de Máquina em Inteligência Artificial. Quanto ao Aprendizado de Máquina Supervisionado, o problema de Classificação consiste em elaborar algoritmos capazes de induzir classificadores usando exemplos de padrões conhecidos para posteriormente reconhecer ou distinguir exemplos desconhecidos com alto nível de acurácia [28].

Nesse contexto, formalmente definindo o problema de classificação, para induzir um classificador o algoritmo recebe um conjunto E de exemplos conhecidos como entrada, na forma $< e_i, y_j >$, onde cada exemplo e_i representa-se na forma vetorial
$e_i = \{Atributo_1, Atributo_2, ..., Atributo_i\}$ sendo que, os componentes do vetor pode ser de tipos discretos ou contínuos e $y_j \in \{C_1, C_2, ..., C_j\}$ representando a classe ou rótulo de cada exemplo. Como saída do algoritmo têm-se o classificador na forma de um mapeamento $F(x) = y$. Sendo assim, o classificador F deve retornar uma das classes $\{C_1, C_2, ..., C_j\}$ para exemplos x desconhecidos, ou seja, classificá-los.

No processo de indução de classificadores um aspecto fundamental é a formação de conceito, pois se o classificador induzido não obteve aprendizado não será capaz de classificar corretamente exemplos não utilizados na fase de treinamento ou desconhecidos [36]. O desempenho do classificador é quantizado durante a fase de testes. Esta etapa consiste na aplicação do classificador sob um conjunto de exemplos de teste E' de modo que cada exemplo deste não esteja no conjunto E utilizado inicialmente para treinar o classificador. Portanto, quanto maior o acerto na classificação dos exemplos de E, melhor o aprendizado

do classificador e como consequência, maiores chances do classificador identificar exemplos E' corretamente.

O objetivo dos algoritmos para resolução de problemas de Classificação multiclasse é naturalmente criar classificadores capazes de identificar exemplos das várias classes para os quais foram treinados. Entretanto, qual será a resposta do classificador para um exemplo que não pertença a nenhuma das classes de exemplo usadas no treino? A resposta está em um tipo especial de classificação chamado Classificação com Uma Classe, tema central do presente trabalho e que será abordado na Seção 2.2.

2.2 Classificação com uma classe

A partir de um conjunto de valores, que são mais frequentes para as variáveis que representam o funcionamento de uma máquina ou sistema complexo, é possível extrair conhecimento ou reconhecer padrões de funcionamento. Um classificador multiclasse poderia ser empregado em tal caso para reconhecer os diferentes padrões, mas o que se deseja é reconhecer também quando tal máquina comporta-se de maneira nova ou estranha aos padrões conhecidos. Neste caso, um novo paradigma de classificação faz-se necessário, em que o classificador induzido é capaz de distinguir um conjunto de exemplos de comportamento normal de outros estranhos ou anormais. Este é um problema típico OCC. Na Figura 2.1, apresenta-se a diferença entre os modelos de Classificação Multiclasse e OCC.

O problema OCC tem sido referenciado na literatura de Aprendizado de Máquina como “Detecção de Novidade”, “Aprendizado de Conceito” e ainda aparece em outros trabalhos como “Detecção de Anomalia” [15]. As diferentes nomenclaturas, segundo Tax [36], são caracterizadas pelo tipo de aplicação na qual se instancia o problema OCC.

Uma característica que diferencia o problema OCC em Aprendizado de Máquina é a ausência de exemplos anormais para treinar o classificador. Esse aspecto aumenta a dificuldade em obter um grau adequado de generalização, pois apenas tendo conhecimento do conjunto normal faz com que a indução do classificador esteja mais suscetível ao problema de superajuste. Neste caso o classificador imita o conjunto normal e novos exemplos deste tipo terão grandes chances de serem classificados como anormais, uma vez que são

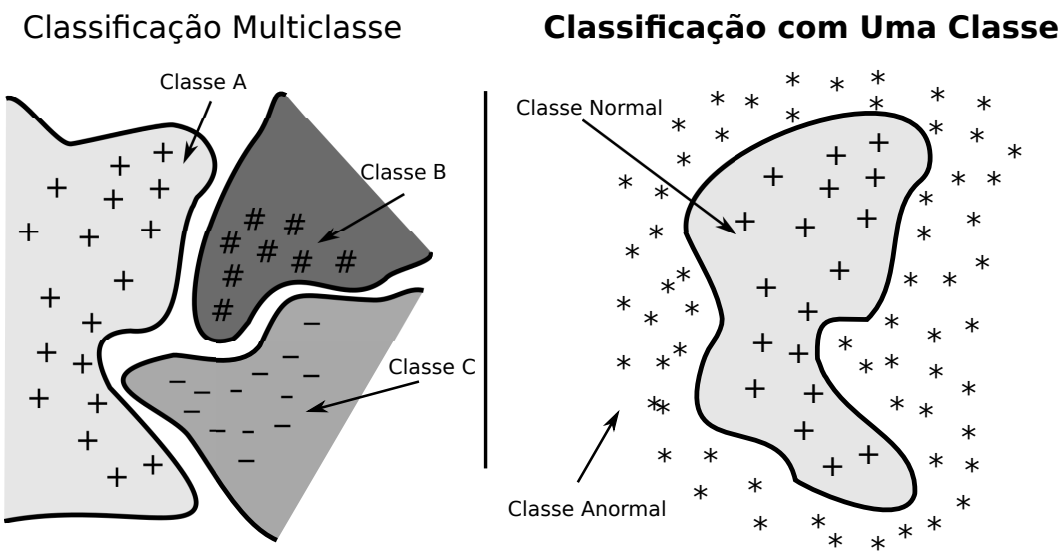


Figura 2.1: Classificação Multiclasse *vs.* Classificação com Uma Classe.

desconhecidos [36].

No caso citado anteriormente cujo intuito é aplicar Classificação para identificar comportamento normal ou anormal de uma máquina ou sistema, seria necessário danificar a máquina ou forçar o sistema a funcionar de modo anormal para que o problema fosse tratado como sendo de classificação binária. Uma estratégia para possibilitar a indução de um classificador binário para esse tipo de problema OCC é produzir exemplos artificiais cujos valores das variáveis ou características diferem do padrão de comportamento normal. Tax considera que essa abordagem escala "pobremente"em conjuntos de dados com alta dimensão [36] — que possuem muitos atributos. Entretanto, a geração de anomalias artificiais viabiliza a aplicação de heurísticas como a PG para esse tipo de problema, o objeto de estudo no presente trabalho.

Segundo Tax [36], há outros algoritmos para OCC que não tratam do problema como classificação binária usando anomalias artificiais. Estes dividem-se em métodos baseados em densidade, reconstrução ou fronteira de decisão. Entre os baseados em densidade estão o estimador Parzen e o *Gaussian Mixture Model* (GMM) [36]. A aplicação efetiva destes requer grandes quantidades de exemplos de comportamento normal para que a densidade seja estimada de maneira precisa, o que na prática pode ser inviável. Entre os métodos baseados em reconstrução estão o K-médias, *Learning Vector Quantization* (LVQ) e *Self-Organizing Maps* (SOM). Por outro lado, segundo Tax [36], a dificuldade

de utilizar as técnicas reconstrutivas se apresenta quando o conjunto de dados normal é irregular, dificultando a construção de um modelo descritivo para o conjunto. Por fim, entre os algoritmos baseados em fronteira estão o algoritmo do Vizinho mais próximo e o *Support Vector Data Description* (SVDD) [36]. A dificuldade destes algoritmos, está segundo Tax [36], em se estabelecer características da fronteira como a distância entre os exemplos normais e anormais, esse é o ajuste-chave para obter generalização adequada. Das dificuldades mencionadas nesses tipos de classificadores surge uma das motivações do presente trabalho, que é estudar técnicas de geração de anomalias, apresentadas na seção 2.3, para aplicar PG de modo a resolver o problema OCC na forma de classificação binária.

2.3 Abordagem baseada na geração de anomalias

A abordagem baseada na geração de anomalias artificiais visa suprir a necessidade de contraexemplos para que o classificador seja induzido, a fim de resolver o problema OCC como um problema classificação binária[1]. Deste modo, tem-se no conjunto de dados para treinar o classificador, duas classes de exemplos: uma composta pelos exemplos normais e a outra pelos exemplos gerados de modo artificial.

De acordo com Fan et al. [15], uma maneira simples de gerar anomalias é atribuir um valor aleatório para cada atributo do conjunto de dados respeitando o domínio de definição do atributo. Para Fan et al. [15], as anomalias produzidas dessa maneira estarão uniformemente distribuídas ao longo do domínio dos atributos. Entretanto, essa abordagem *ad-hoc* pode não oferecer condições de aprendizado ao classificador para que desenvolva regras de classificação para evitar problemas de superajuste, pois na produção dos contraexemplos não são considerados aspectos do conjunto normal como a sua distribuição e os seus limites no espaço do domínio dos atributos. Nesta Seção estão apresentadas técnicas para gerar anomalias que se baseiam em características do conjunto normal.

[1]Na prática pode haver problemas OCC em que há disponibilidade de contraexemplos reais e que podem ser usados para induzir o classificador. Entretanto, a análise exclusiva desse tipo específico de problema está fora do escopo do presente trabalho.

2.3.1 Geração de anomalias baseada em valores máximos dos atributos

Wang et al. [39] apresentaram um método para geração artificial de anomalias que consiste em selecionar aleatoriamente dois atributos A_x e A_y do conjunto normal e selecionar o maior valor de ambos atributos ($A_{max,x}$ e $A_{max,y}$) entre os exemplos. Posteriormente, selecionar aleatoriamente um exemplo e substituir seus valores A_x e A_y por $A_{max,x}$ e $A_{max,y}$. Se esse procedimento for repetido j vezes, j anomalias serão geradas. Além da geração artificial de anomalias, no trabalho de Wang et al. [39], demonstrou-se que é possível otimizar a configuração de determinados parâmetros do algoritmo SVDD executando o treinamento M vezes utilizando validação cruzada.

O treinamento do classificador SVDD foi realizado utilizando um conjunto de dados em que as anomalias eram compostas por exemplos reais do conjunto de dados não pertencentes à classe normal e a exemplos artificiais gerados pelo método proposto. Por meio dos resultados do experimento, observou-se que o SVDD apresentou diminuição na taxa de erro de classificação quando utilizadas as anomalias artificiais e que, quanto maior o número delas no conjunto de dados de treinamento, menor a taxa de erro de classificação do classificador SVDD induzido.

2.3.2 Geração de anomalias usando os exemplos mais distantes

Bánhalmi et al. [6] propuseram um método para geração de anomalias que consiste em, inicialmente, identificar os exemplos da fronteira da distribuição do conjunto normal no espaço de atributos. São considerados exemplos da fronteira aqueles que são linearmente separáveis dos demais exemplos. Para tanto, utiliza-se o algoritmo *hard-margin Support Vector Machine* (SVM) que visa, em um processo de otimização, encontrar um hiperplano que separa um exemplo normal dos demais. Se esse processo falha, considera-se que o exemplo não está na fronteira, caso contrário, considera-se que ele está na extremidade da distribuição do conjunto normal, e portanto pertencente à fronteira.

No segundo passo do método, para cada exemplo x do conjunto normal é gerada uma

anomalia y utilizando um movimento de translação de modo que y é deslocado partindo de x. O movimento de translação é definido pela Equação $y = (1 + T(x, x_b)/||v||) + x$, em que x_b é o exemplo da fronteira que está mais próximo de x e v representa a orientação da translação, sendo que $v = x_b - x$ e a função $T(x, x_b)$ representa a distância envolvida na translação de y e é dada pela Equação 2.1:

$$T(x, x_b) = \frac{dist}{dist \cdot curv + CosAngle(x, x_b)} \tag{2.1}$$

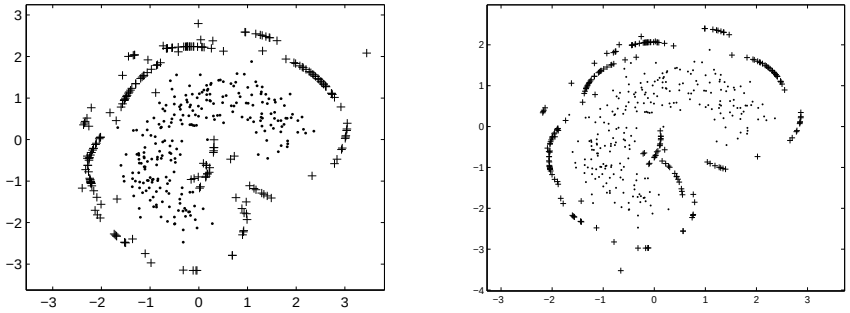
em que:

$$CosAngle(x, x_b) = \frac{x_{b,center} \cdot (x - x_b)}{||x_{b,center}||||x - x_b||} \tag{2.2}$$

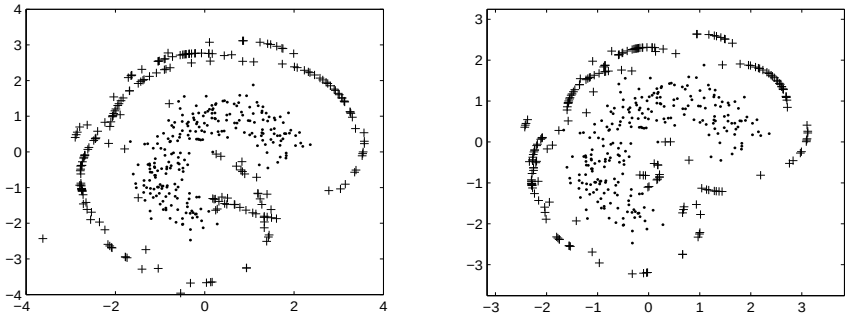
Em que $x_{b,center}$ representa o vetor normal do hiperplano encontrado pelo *hard-margin* SVM no primeiro passo para identificar x_b como sendo um exemplo da fronteira do conjunto normal.

De acordo com Bánhalmi et al. [6], é necessário fazer uma verificação adicional para assegurar que a anomalia y está distante dos exemplos da classe normal. Para isso, é utilizado novamente o algoritmo *hard-margin* SVM e deste modo se não há um hiperplano que separa y dos exemplos vizinhos do conjunto normal, y é descartado e outro contraexemplo é gerado utilizando outro exemplo x_b da fronteira que esteja mais próximo x.

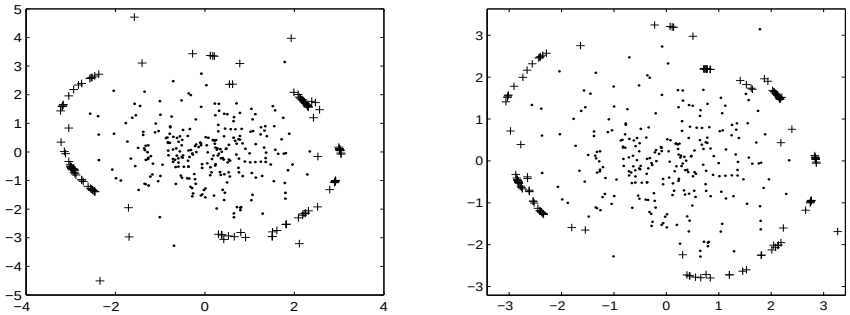
Na Equação 2.1 são utilizados os parâmetros *Dist* e *Curv* que possibilitam regular a distância e a orientação dos contraexemplos y gerados. Na Figura 2.2, apresentam-se exemplos de geração de anomalias usando diferentes ajustes para *Dist* e *Curv* na geração de anomalias para um conjunto de dados cuja a distribuição dos exemplos normais apresenta-se na forma de banana e outro na forma Gaussiana. As bibliotecas (*toolbox*) para Matlab PRTools e DDtools foram usadas para gerar 250 exemplos para esses conjuntos de dados compostos por dois atributos. As anomalias foram geradas utilizando a implementação em Matlab do método disponível em: [5]. Observa-se para conjunto banana que o valor $Curv = 1.0$ possibilita que as anomalias estejam dispostas de modo a melhor contornar os exemplos normais.



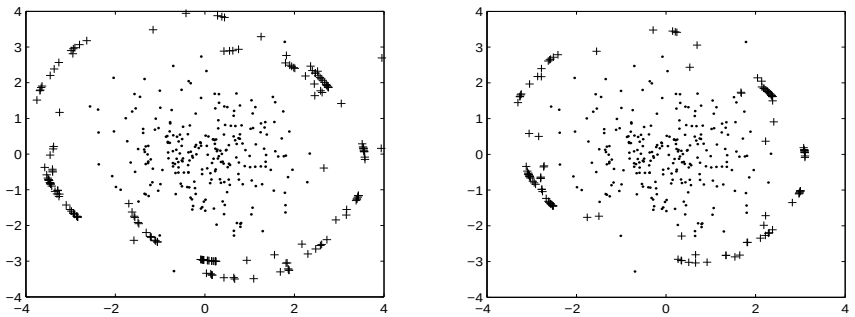
(a) Dist=1.0,Curv=0.5 (b) Dist=1.0,Curv=1.0



(c) Dist=3.0,Curv=0.5 (d) Dist=3.0,Curv=1.0



(e) Dist=1.0,Curv=0.5 (f) Dist=1.0,Curv=1.0



(g) Dist=3.0,Curv=0.5 (h) Dist=3.0,Curv=1.0

Figura 2.2: *Exemplos do efeito na geração de anomalias variando os parâmetros Dist e Curv para os conjunto de dados na forma de banana e na forma Gaussiana.*

Em seus experimentos, Bánhalmi et al. [6], treinaram o algoritmo ν-SVM sobre 10 diferentes conjuntos de dados utilizando o método de geração de anomalias por eles proposto. Também treinaram os algoritmos para OCC *One-Class* SVM e GMM que utilizam apenas os exemplos da classe normal para induzir o classificador. Os resultados desse experimento mostraram que as taxas de acerto global e de acerto para exemplos do conjunto normal foram em geral melhores para o algoritmo ν-SVM treinado como um classificador binário, usando o método de geração de anomalias proposto.

2.3.3 Geração de anomalias baseada na distribuição de exemplos

Fan et al. [15] propuseram um método de geração de anomalias para aplicação em intrusão em redes de computador chamado *Distribution Based Artificial Anomaly* (DBA2). Esse método baseia-se na distribuição do conjunto normal no espaço de atributos R^n para gerar as anomalias. De acordo com Fan et al. [15], para o classificador conseguir boa generalização é necessário gerar anomalias próximas à fronteira que separa o conjunto conhecido das anomalias e para isso uma heurística simples seria alterar o valor de um atributo qualquer de um exemplo conhecido mantendo o mesmo valor para as demais características. Todavia, o conjunto normal pode ocupar regiões no espaço de atributos mais densas e outras menos, o que segundo Fan et al. [15], pode causar o problema de superajuste se utilizadas as anomalias geradas via heurística simples. Com a finalidade de evitar esse problema e para melhor identificar a fronteira que separa o conjunto normal, o DBA2 identifica as regiões esparsas do conjunto normal e agrupa essas regiões para formar uma região única que caracteriza a ocupação do conjunto no espaço de atributos. Então a geração de anomalias é realizada de modo a contornar essa região. Na Figura 2.3 [15], está apresentada a ideia intuitiva da proposta de geração de anomalias utilizando o DBA2. As anomalias são representadas por $\times$ e o conjunto normal representado pelo espaço interno da região curvada em um espaço de atributos R^2.

O algoritmo DBA2 é apresentado em (Algoritmo 1). Para cada um dos atributos do conjunto de dados de treino identifica-se o conjunto V_f de valores pouco frequentes e o número de ocorrências $countV_{max}$ do valor mais frequente. O número $countV$ de

Figura 2.3: Anomalias geradas em volta da fronteira do conjunto normal [15].

ocorrências de cada valor infrequentes v de f é contabilizado e para cada v seleciona-se aleatoriamente $countV_{max} - countV$ exemplos do conjunto normal e cada um destes tem seu valor v_f de f substituído por v' tal modo que $v' \neq v \wedge v' \neq v_f$. Os exemplos do conjunto normal modificados durante o algoritmo irão compor o conjunto final de anomalias artificiais D'.

Uma das vantagens do uso do DBA2, de acordo Fan et al. [15], é a possibilidade de estimar o número de anomalias a serem gerados, pois, se cada atributo f tiver $|V_f|$ valores infrequentes, sendo $countV_{max}(f)$ as ocorrências do valor mais frequente, então o número máximo de anomalias a ser gerado corresponde a $\sum_{f \in F} |V_f| \times countV_{max}(f)$.

Os experimentos realizados para detecção de anomalias em redes de computador mostram que o DBA2 foi melhor do que a geração de anomalias simples e a abordagem de mudança de apenas uma dos valores de um atributo selecionado aleatoriamente comentados no início da Seção 2.3.

2.3.4 Geração de anomalias usando hiperesfera

Tax [36] introduziu o algoritmo SVDD para OCC. O SVDD calcula uma hiperesfera que engloba o conjunto normal por meio de vetores suporte que são exemplos do conjunto que representam as extremidades da hiperesfera, e aqueles fora desta hiperesfera serão classificados pelo SVDD como anormais.

De acordo com Tax e Duin [38], para melhor descrever o conjunto normal usando o

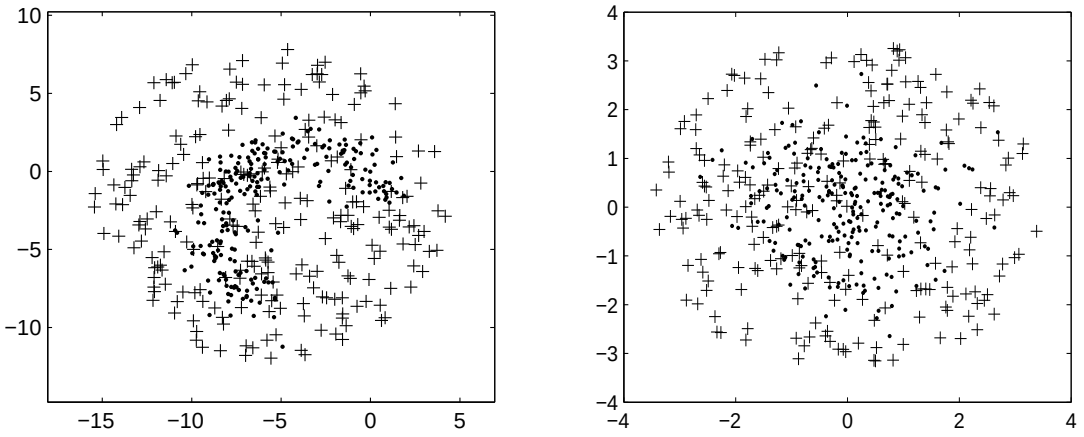
Algoritmo 1 DBA2

Procedimento DBA2(D)
  F =conjunto de todos os atributos de D
  V_f =conjunto de valores únicos dos atributos $f \in F$
  $D' = \varnothing$
  Para Todos $f \in F$ **faça**
    $countV_{max}$ = número de ocorrências do valor mais frequente em V_f
    Para Todos $v \in V_f$ **faça**
      $countV$ = número de ocorrências de v em D
      Enquanto $countV < i \leq countV_{max}$ **faça**
        d = um valor $d \in D$ selecionado aleatoriamente
        v_f = um valor do atributo f para d
        substitua v_f com um valor v' selecionado aleatoriamente para criar d' sujeito a: $v' \neq v \wedge v' \neq v_f$
        $D' \leftarrow D' \cup \{d'\}$
      Fim Enquanto
    Fim Para
  Fim Para
  Retorne D'
Fim Procedimento

modelo de hiperesfera do SVDD foram introduzidas funções *kernel*, porém é necessário ajustar os parâmetro s — largura da função — e C que representa um valor de equilíbrio. Todavia, para um melhor ajuste de s e C, é necessário definir um critério de erro composto pelas taxas de erro na classificação de exemplos normais e anomalias. Tax e Duin [38] elaboraram um método para gerar anomalias artificiais que possibilita estimar essas taxas. O método consiste em usar distribuição Gaussiana n-dimensional para gerar as anomalias de maneira uniforme dentro e fora da hiperesfera que engloba o conjunto normal. A hiperesfera é calculada usando os vetores suporte gerados pelo próprio método SVDD tendo o conjunto normal como entrada.

Na Figura 2.4, apresentam-se dois exemplos de geração de anomalias artificiais usando o conceito de hiperesfera proposto por Tax e Duin [38]. As ferramentas PRtools e DDTools [12] [37] para o *software* Matlab [2] foram utilizadas para criar dois conjuntos de dados de dois atributos contendo 250 exemplos. Uma deles com distribuição na forma de banana e a outra na forma Gaussiana. Outras 250 anomalias foram geradas para cada conjunto usando o método proposto.

Segundo Tax e Duin [38], por meio da otimização dos parâmetros s e C usando anoma-



(a) Classe normal na forma de banana e hiperesfera de anomalias.

(b) Classe normal na forma de distribuição Gaussiana e hiperesfera de anomalias.

Figura 2.4: *Exemplos de anomalias (+) geradas pelo método proposto por Tax e Duin.*

lias artificiais baseadas em hiperesfera o SVDD se mostrou equiparável ao método OCC baseado em Mistura de Gaussianas. De acordo com Tax e Duin [38], outros classificadores podem ser treinados utilizando os contraexemplos gerados pelo método de geração de anomalias baseado em hiperesfera. Entretanto, sugere-se que uma quantidade suficiente de anomalias seja gerada para cobrir a classe normal no espaço ocupado por todos os atributos e isto resultará em um desequilíbrio entre o número de exemplos de uma classe e da outra. Também recomenda-se que o classificador consiga tratar o problema de sobreposição de classes — há exemplos de uma das classes no domínio da outra —, pois haverá anomalias dentro e fora da hiperesfera que delimita as classes. Essa técnica foi utilizada com sucesso em OCC usando PG e foi apresentada na Seção 3.3.

2.3.5 Geração de anomalias inspirada em sistemas imunológicos

Gonzales et al. [19] [20] propuseram uma abordagem voltada para a detecção de anomalias que consiste em induzir um classificador utilizando exemplos normais e anomalias geradas por meio do algoritmo imunológico *Real-Valued Negative Selection* (RNS), baseado nos princípios da distância entre células *self* e *non-self*. A entrada do RNS é um conjunto de vetores $n-$dimensionais que correspondem às células *self*. A saída é um número predefinido de detectores (anticorpos) que são *a priori* gerados aleatoriamente e

o seu posicionamento no espaço é alterado iterativamente a fim de maximizar a cobertura do espaço *non-self*. Isto é obtido afastando os detectores dos pontos *self* e mantendo-os separados entre si.

Um detector (anticorpo) é definido por um vetor n-dimensional que corresponde ao seu centro e por um valor real r que corresponde ao seu raio de detecção. Deste modo, o detector pode ser visto como uma hiperesfera no espaço R^n. Utilizando o raio r determinam-se os k vizinhos *self* e quantos destes estão próximos do detector d por meio da Equação 2.3 onde $||.||$ é a norma euclidiana.

$$\mu_d(x) = -e^{\frac{||d-r||^2}{2r^2}} \tag{2.3}$$

Cada detector d possui um atributo idade que é inicializado em 0 e incrementado a cada iteração. Durante o processo o detector é substituído por outro gerado aleatoriamente se alcançar idade t e não conseguir se distanciar do espaço *self*. De acordo com Gonzales et al. [19], para o RNS convergir de maneira estável é necessário usar o parâmetro η_i que é decrementado a cada iteração i conforme a Equação 2.4 e de modo η_i tenda a zero ($\lim_{i\to\infty} \eta_i = 0$) onde η_0 é o valor inicial da taxa de adaptação e π controla o decréscimo. O critério de parada é o número de iterações do algoritmo. O algoritmo RNS é descrito em Algoritmo 2:

$$\eta_i \to \eta_0 e^{\frac{-i}{\pi}} \tag{2.4}$$

Gonzales et al. [20] utilizaram uma rede neural multicamadas treinada com *backpropagation* para induzir um classificador usando exemplos normais e as anomalias geradas usando RNS. A combinação do RNS com a rede neural foi chamada de *Hybrid Neuro-Immune System* (HNIS). Conforme os resultados dos experimentos, a abordagem HNIS mostrou-se equiparável em comparação com a rede neural auto-organizável SOM para uma aplicação real em detecção de anomalias (MIT Darpa 98 e 99).

Algoritmo 2 RNS

Procedimento Real-Valued-Negative-Selection(r, η, t, k)
    r : raio de detecção
    η : taxa de adaptação
    t : idade do detector
    k : número de vizinhos do detector
    Gere aleatoriamente uma população de detectores
    Enquanto Critério de parada não satisfeito **faça**
        Para Todos detectores d **faça**
            $NearCells = k$vizinhos de d do conjunto *self*
            $NearCells$ é ordenada com relação a distância para d
            $NearestSelf =$ a média de $NearCells$
            Se $distance(d, NearestSelf) < r$ **então**
                $dir = \frac{\sum_{c \in NearCells}(d-c)}{|NearcCells|}$
                Se idade de $d > t$ **então** ▷ detector é velho
                    substitua d por outro gerado aleatoriamente
                Senão
                    Incremente a idade de d
                    $d = d + n * dir$
                Fim Se
            Senão
                idade de $d = 0$
                $dir = \frac{\sum_{d' \in Detectors} \mu_d(d')(d-d')}{\sum_{d' \in Detectors} \mu_d(d')}$
                $d = d + n * dir$
            Fim Se
        Fim Para
    Fim Enquanto
Fim Procedimento

2.4 Aplicações reais de classificação com uma classe

O problema OCC não é exclusividade de apenas uma área, ao contrário, trabalhos que resolvem esse tipo de problema têm aparecido em Biologia, Análise de Imagens, Segurança da Informação, Mineração de Dados, Detecção de Falhas em máquinas e Finanças. Zhong et al. [42] empregaram métodos OCC para identificar notícias que têm maior relevância para o mercado acionário durante operação dos mercados de Shangai e Shenzhen. Em seus experimentos observaram que técnicas de classificação binária para esse tipo de problema obtiveram desempenho inferior. Bánhalmi et al. [4] utilizaram vários métodos OCC para a tarefa de classificação de proteínas e concluíram ser uma alternativa viável se comparada ao uso comum de classificação binária para o mesmo problema. Munroe et al.

[29] aplicaram com sucesso o algoritmo para OCC *k-Nearest Neighbour* para o reconhecimento de automóveis em imagens de modo a conseguir identificá-los dos demais tipos de veículos. Manevitz [27] aplicou uma versão modificada do algoritmo *one-class* SVM em conjuntos de dados da fonte *Reuters* da área de recuperação de informação e realizou comparativo com outras técnicas para OCC e constatou que sua proposta de SVM é equiparável a um algoritmo de rede neural específico para esse tipo problema. Shin et al. [34] também propuseram uma técnica *one-class* SVM para detecção de falhas em máquinas que possuem componentes rotacionais com o objetivo de induzir um classificador a partir de um conjunto de dados que expressam o comportamento normal e determinar se há movimentos oscilatórios anormais que podem levar à falha da máquina. O algoritmo proposto apresentou melhores resultados comparados à uma rede neural *Multi Layer Perceptron* (MLP) [34]. No contexto de detecção de anomalia inclui-se também o trabalho de Fan et al. [15] para detecção de ataques desconhecidos em redes de computador, cujo algoritmo de geração de anomalias DBA2 foi apresentado na Seção 2.3.3.

2.5 Considerações finais

O problema OCC caracteriza-se pela disponibilidade de apenas um conjunto de exemplos e com estes deseja-se induzir um classificador capaz de identificar exemplos desconhecidos como pertencentes ou não à classe de exemplos disponível. Neste contexto, uma das estratégias é tratar o problema como sendo de classificação binária por meio da geração de anomalias. Na Seção 2.3 foram apresentados diversos métodos para esse fim.

Também foram mencionadas várias instâncias de problemas reais OCC na Seção 2.4. Porém, observa-se que diferentes técnicas de classificação foram utilizadas e a PG, amplamente aplicada em problemas de Aprendizado de Máquina, parece estar em estágios iniciais de pesquisa para esse tipo problema. No Capítulo 3 serão discutidas a PG para classificação e sua aplicabilidade em OCC.

CAPÍTULO 3

PROGRAMAÇÃO GENÉTICA PARA CLASSIFICAÇÃO

3.1 Introdução

Programação Genética é uma técnica para elaborar programas de computador de maneira evolutiva para resolução de problemas complexos [23]. Uma população de indivíduos é criada aleatoriamente e eles são avaliados quanto à proximidade da solução desejada. A função de aptidão (*fitness*) é utilizada para computar esse valor. Conforme os princípios da seleção natural de Darwin [23], os indivíduos mais bem adaptados — com maior valor de aptidão — têm maiores chances de sobrevivência e por isso se reproduzem e propagam seu material genético para as próximas gerações. Na PG os indivíduos com maior valor de aptidão têm mais chances de serem selecionados para aplicação de operadores genéticos a fim de compor a população da nova geração.

Os operadores genéticos básicos da PG são os de reprodução, cruzamento e mutação[1]. A reprodução copia o indivíduo para a próxima geração mantendo o material genético. O cruzamento implica na combinação de partes de dois programas para a formação de outros dois indivíduos para a nova população. Esse operador visa criar melhores soluções a partir das já existentes, de modo a explorar o espaço de busca de todas as soluções possíveis. O operador de mutação altera partes internas da estrutura do programa para aumentar a diversidade na população. Deste modo, evita a convergência prematura ou o estacionamento em um ponto do espaço de busca em que se encontram programas subótimos.

A PG em sua forma canônica, como introduzida por Koza [23], utiliza populações de indivíduos compostas por expressões representadas em forma de árvore. Para isso, faz-se necessário definir o conjunto de operadores e o conjunto de terminais que irão

[1]Existem outros tipos e formas melhoradas desses operadores, porém o estudo deles não é alvo do presente trabalho.

compor os programas. Na Figura 3.1, são apresentados exemplos de indivíduos e as operações genéticas de cruzamento e mutação. Observa-se que os programas são compostos pelo conjunto de operadores $\{\times, -, +, \div\}$ e pelo conjunto de terminais $\{f1, f2, f3\}$. O cruzamento é realizado entre dois programas que resolvem respectivamente as equações $y_1(f1, f2, f3) = (f2 \times f1) - (f2 + f3)$ e $y_2(f1, f2, f3) = \frac{f1}{(f1-f3)+(f2\times f2)}$, onde $f1, f2$ e $f3$ são variáveis de entrada dos programas, e o cruzamento consiste em trocar as subárvores $f2 + f3$ e $f1 - f3$ dos programas para formar os indivíduos $y_{1'}(f1, f2, f3) = (f2 \times f1) - (f1 - f3)$ e $y_{2'}(f1, f2, f3) = \frac{f1}{(f2+f3)+(f2\times f2)}$. A mutação do indivíduo $y_{1'}(f1, f2, f3)$ consiste em trocar as subárvores $f2 + f1$ e $f2 + f3$ para formar o novo indivíduo $y_{1'}(f1, f2, f3) = (f1 - f3) - (f2 \times f1)$.

3.2 Programação Genética para classificação

Em problemas de classificação, os indivíduos da população da PG são tratados como programas classificadores. O procedimento de indução acontece durante o processo evolutivo da PG, no qual cada classificador é executado para cada exemplo do conjunto de treinamento tendo como entrada os valores dos atributos dos exemplos, linha 7 do Algoritmo 3. O desempenho de cada classificador é avaliado através da Função de Aptidão, linha 9 do Algoritmo 3, e os indivíduos mais bem adaptados têm mais chances de serem propagados para as próximas gerações por meio dos operadores genéticos, linhas 11 e 12 do Algoritmo 3.

Após a obtenção do melhor classificador, cujo o valor de aptidão é o melhor na última geração[2], realiza-se o procedimento de teste para avaliar o classificador na classificação de exemplos do conjunto de teste, Procedimento *TESTE_PG* do Algoritmo 3. Para isso, o classificador ótimo da última geração é executado para cada exemplo do conjunto de testes, linhas 21 e 22 do Algoritmo 3, e em seguida suas taxas de acerto e erro em classificação para as classes são computadas e retornadas, linhas 24 e 25 do Algoritmo 3.

Um detalhe importante é o modo como os programas irão computar os valores dos atri-

[2]O número de gerações é normalmente usado como critério de parada, porém outros critérios podem ser estabelecidos.

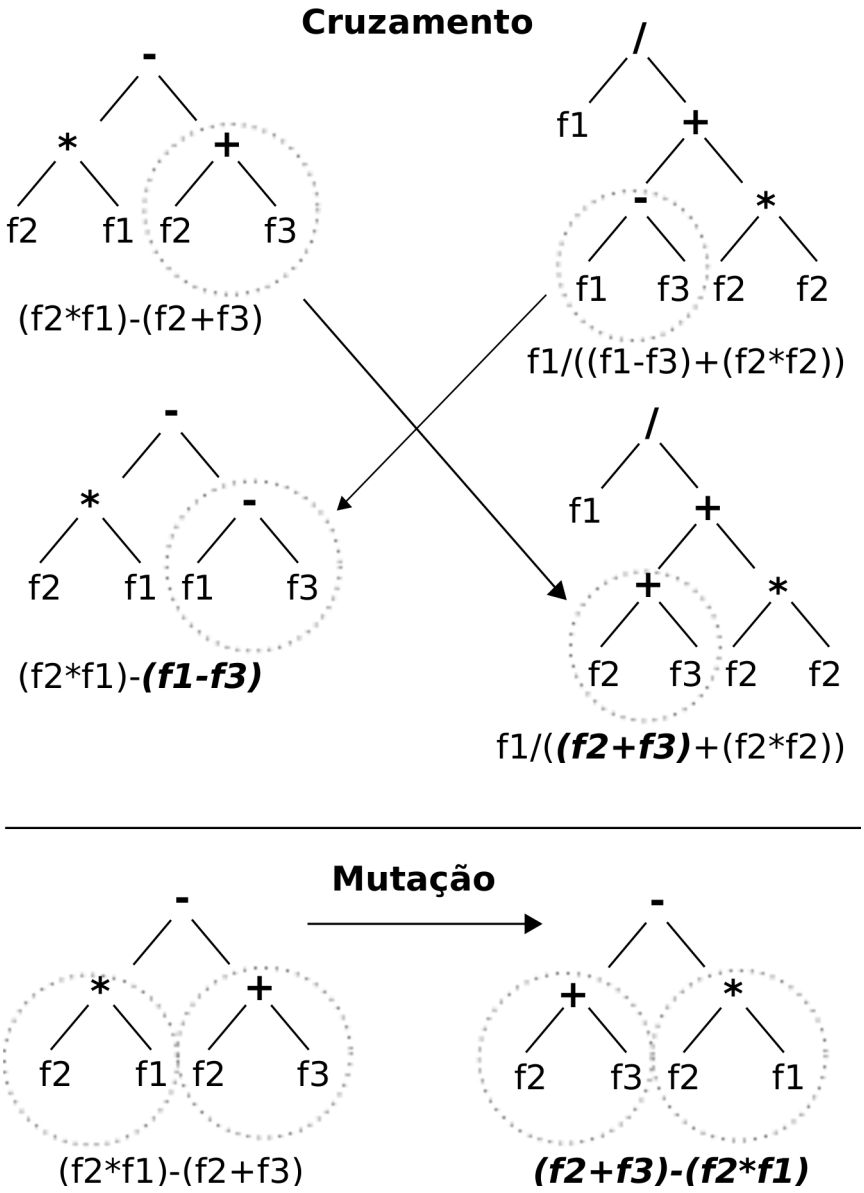


Figura 3.1: Exemplo de aplicação de operadores genéticos de cruzamento e mutação.

butos para ao final decidirem sobre qual classe pertence o exemplo. Para isto é necessário estabelecer o modelo de representação da solução utilizado para realizar a classificação. Os mais utilizados em PG, segundo Espejo et al. [14], são as Árvores de Decisão (*Decision Trees*), Regras de Classificação (*Classification Rules*) e as Funções Discriminantes (*Discriminant Functions*). No presente trabalho destacam-se as Funções Discriminantes que consistem em expressões matemáticas compostas por diferentes tipos de operadores e nas quais as variáveis são os atributos do conjunto de dados. O valor de saída da Função Discriminante será utilizado para decidir sobre qual classe pertence o exemplo. Mais detalhes sobre esse mapeamento encontram-se na Seção 3.2.1. Os programas da PG apre-

Algoritmo 3 Procedimento de Indução e Teste dos classificadores PG

```
1: Procedimento TREINAMENTO_PG( T )
2:     Inicialize a população de programas classificadores P
3:     Leia o conjunto de treino T
4:     Enquanto Critério de para não satisfeito ou (numGeracao < Total_Geracoes) faça
5:         Para Todos classificadores p em P faça
6:             Para Todos exemplos t em T faça
7:                 Execute o classificador p usando t como entrada
8:             Fim Para
9:             Registre o valor de aptidão f_p de p
10:        Fim Para
11:        Aplique operadores genéticos Cruzamento, Mutação etc, conforme valores de aptidão.
12:        Atualize a população P
13:        Incremente numGeracao
14:    Fim Enquanto
15:    Retorne classificador p com melhor f_p de P
16: Fim Procedimento
17:
18: Procedimento TESTE_PG( Classificador_Otimo, T )
19:    Leia o conjunto de teste T
20:    Para Todos exemplos t em T faça
21:        Execute o Classificador_Otimo usando t como entrada
22:        Verifique se a classe de t é a mesma retornada por Classificador_Otimo
23:    Fim Para
24:    Contabilize taxas de acertos e erros das classes
25:    Retorne as taxas de acertos e erros das classes
26: Fim Procedimento
```

sentados na Figura 3.1 são expressos na forma de Funções Discriminantes. Considera-se que esse é um modelo de representação da solução adequado por ser simples de computar e também pela sua ampla aplicabilidade em processamento de imagens e reconhecimentos de padrões, por isto, viabiliza também sua aplicação em diversos tipos de problemas OCC.

Segundo Loveard e Cielsielski [26], a PG tem sido amplamente aplicada em problemas de classificação de diversas áreas devido à sua flexibilidade, pois em seu bloco evolutivo podem ser utilizados diversos modelos para representar os classificadores e, se observados como programas de computador, apresentam expressividade não encontrada em outros algoritmos como árvores de decisão, classificadores estatísticos e redes neurais. Além disso, de acordo com Loveard e Cielsielski [26], se o tempo de treinamento do classificador

PG não for um problema, é possível estender gradualmente o número de gerações utilizadas até que haja condição de sobreajuste. A partir desse ponto reduz-se o número de gerações até que a generalização desejada seja obtida. Outra característica interessante da PG para classificação é a seleção natural dos atributos do conjunto de exemplos que são mais relevantes, deste modo, criam-se classificadores simplificados e mais fáceis de interpretar [14]. Para tanto, o processo evolutivo da PG irá realizar pressão seletiva sobre os indivíduos que utilizam atributos mais relevantes como entrada, pois estes terão mais chances de apresentar melhores valores de aptidão do que outros programas que possuem como variáveis de entrada atributos pouco relevantes para caracterizar as classes. Outra vantagem da PG é que ao final de sua execução tem-se uma população de indivíduos classificadores que podem ser utilizados em um esquema de votação, tornando o resultado final da classificação ainda mais confiável [10] [11] [26].

No entanto, umas das desvantagens da PG é que para avaliar o desempenho dos indivíduos na fase de treino é necessário executar cada indivíduo p da população sob cada exemplo e do conjunto de dados durante cada uma das g gerações. Isto é, são necessárias $p \times e \times g$ execuções. Nesse sentido, há pesquisa e desenvolvimento de outros tipos de PG, como por exemplo a PG Linear, que utiliza estruturas de dados mais simples como listas de instruções para representar e processar os indivíduos da população. Esta simplicidade possibilita que os programas sejam executados de maneira otimizada e usando menos recursos de memória [35]. Todavia, o presente trabalho centra o uso na PG que representa os programas da população na forma de árvores, conforme Figura 3.1, uma vez que este é o tipo precursor de PG introduzido por Koza [23] e no atual estado da arte não há relatos sobre aplicabilidade desse tipo de PG para o problema OCC.

Na Seção 3.2.1, apresentam-se métodos utilizados para traduzir o valor de saída da função discriminante em rótulos de classes. Na Seção 3.2.2, descreve-se como os indivíduos da população são avaliados na tarefa de classificação por meio da função de aptidão.

3.2.1 Transformando valores em classes

Se o valor retornado por um programa da PG (*PGout*), para um dado exemplo do conjunto de dados, for positivo, este é classificado como pertencendo a uma das classes, caso contrário, valores de *PGout* negativos correspondem a exemplos da outra classe. Essa regra de tradução é geralmente empregada em PG para classificar os exemplos em um problema de classificação binário em que os indivíduos da PG são expressões matemáticas compostas por operadores aritméticos e as variáveis de entrada são os atributos do conjunto de dados. Isto é, os programas da PG são Funções Discriminantes. Na Tabela 3.1, apresenta-se a aplicação dessa regra para classificar 4 exemplos sintéticos de atributos $\{A1, A2, A3\}$ como pertencendo às classes 0 ou 1 utilizando o programa discriminante F.

Tabela 3.1: Classificação binária de 4 exemplos para a Função Discriminante $(A2 * A1) - (A1 - A3)$

Exemplos					
	A1	A2	A3	*PGout*	Classe
1	1,5	1,5	2,0	**-1,25**	**0**
2	2,5	2,0	1,0	**2,0**	**1**
3	1,3	2,0	3,0	**-1,7**	**0**
4	3,0	2,0	1,5	**1,5**	**1**

Segundo Loveard e Cielsielski [26], para um problema de classificação onde há n classes (multiclasse) é possível aplicar decomposição binária e então $n - 1$ problemas de classificação binária são resolvidos utilizando a regra de tradução anteriormente citada para decidir sobre a classe dos exemplos. A principal desvantagem é o custo computacional se o problema de classificação possuir muitas classes. Outras maneiras de realizar a tradução dos valores *PGout* em classes foram propostos. Loveard e Cielsielski [26] mencionaram o uso de intervalos fixos onde cada intervalo corresponde a uma classe. Entretanto, de acordo com os autores, essa abordagem requer conhecimento *a priori* do problema, pois se forem estabelecidos intervalos arbitrários muitas gerações serão necessárias para treinar os classificadores.

Alternativamente foi proposto o uso de intervalos dinâmicos para cada classe em que os programas da população da PG são executados tendo como entrada um subconjunto do conjunto de exemplos de treino. Em seguida os valores de *PGout* são processados por

um outro algoritmo que identifica os intervalos onde há maior concentração de valores *PGout* para cada classe. Então esses intervalos são utilizados para classificar os exemplos para o restante dos exemplos do conjunto de treino e em cada n gerações os intervalos são computados novamente [26].

Também foi proposta a enumeração de classes na qual a Função Discriminante é também composta por variáveis que representam os atributos e constantes que representam as classes, e o valor de retorno do programa é sempre uma destas constantes [26]. Outra proposta é o uso do acúmulo de evidências em que durante a computação da Função Discriminante valores de pontuação são incrementados ou decrementados a cada classe, e ao final o exemplo é classificado como pertencendo à classe com maior pontuação [26]. Observa-se que ambas as propostas se assemelham mais às Árvores de Decisão que Funções Discriminantes e por isto requerem operadores genéticos especiais para manter a consistência dos indivíduos em caso de mutação ou cruzamento.

Smart et al. [41] propuseram o uso de distribuição Gaussiana dos valores de *PGout* para cada classe de exemplos. Se a distância entre as Gaussianas é máxima ou a área de sobreposição entre elas é mínima utiliza-se a distância da média da Gaussiana para decidir sobre a classe do exemplo. Portanto, quanto mais próximo o valor de *PGout* estiver da média da Gaussiana de uma das classes, maior a possibilidade do exemplo pertencer àquela classe.

Em todos esses trabalhos voltados à classificação multiclasse há um senso comum de que para classificação binária é possível utilizar o mapeamento explicado no início da presente seção em que os valores positivos de *PGout* correspondem a uma classe e os negativos a outra sem perda de desempenho. Portanto, também aplica-se no problema OCC cujo conjunto de dados é composto por exemplos da classe normal e anomalias artificiais.

3.2.2 Funções de aptidão

A Função de Aptidão (*fitness*) é utilizada para avaliar o desempenho dos indivíduos da PG. Aqueles com melhores valores de aptidão têm mais chances de propagar material genético para as próximas gerações. Na tarefa de classificação, a Função de Aptidão conduzirá o processo evolutivo de busca da PG pelo classificador ótimo em um espaço de busca de classificadores. Neste contexto, uma estratégia para avaliação de desempenho presente em vários trabalhos [14], é a de usar como Função de Aptidão a taxa de acerto global de classificação, o *Accuracy*. Esta taxa representa a proporção de exemplos corretamente classificados em relação ao total de exemplos do conjunto de dados e para um problema de classificação binário, é dada pela Equação 3.1. Verdadeiro Positivo - número de exemplos da classe de normal classificados corretamente (VP) é o número de exemplos corretamente classificados no conjunto de P exemplos positivos e Verdadeiro Negativo - número de exemplos da classe de anormal classificados corretamente (VN) representa o número de exemplos corretamente classificados no conjunto de N exemplos negativos.

$$F = \frac{(VP + VN)}{(P + N)} \tag{3.1}$$

Para tornar a avaliação dos classificadores mais precisa outros fatores têm sido empregados na Função de Aptidão [3]. Hennessy et al. [22] propuseram o uso do "*certainty*" que visa à valorização dos classificadores mais confiáveis. Isto é, os classificadores cujos valores de *PGout* afastam-se ao máximo do valor de limiar 0 em um problema de classificação binário cujo método de tradução dos valores de *PGout* em classe é aquele apresentado na Seção 3.2.1. Em outro trabalho a cada exemplo do conjunto de dados de treino é atribuído um peso, que inicialmente é igual para todos os exemplos. À medida que a PG itera entre as gerações os exemplos classificados incorretamente com mais frequência têm seu peso atualizado para valores maiores [13].

Patterson e Zhang [31] propuseram funções de aptidão que visam amenizar o problema

[3]Há trabalhos de PG para classificação em que utiliza-se otimização multiobjetivo. Por isto utiliza-se mais de um objetivo a ser maximizado ou minimizado na Função de Aptidão, entre eles o *Accuracy*. Otimização multiobjetivo não é foco do presente trabalho.

de desequilíbrio de classes. As funções são dadas pelas equações 3.2 e 3.3. $\frac{VP}{P}$ é a taxa de exemplos da classe minoritária P corretamente classificados e $\frac{VN}{N}$ é a taxa de exemplos da classe maioritária N corretamente classificados. Ambas as taxas são divididas por dois, a fim de manter o valor de aptidão no intervalo $[0, 1]$. As taxas são elevadas ao quadrado na Equação 3.3 para penalizar indivíduos com baixo desempenho na classificação.

$$F1.1 = \frac{(\frac{VP}{P}) + (\frac{VN}{N})}{2} \tag{3.2}$$

$$F1.2 = \frac{(\frac{VP}{P})^2 + (\frac{VN}{N})^2}{2} \tag{3.3}$$

Constata-se nos experimentos de Patterson e Zhang [31] que ambas as funções de aptidão melhoraram o desempenho para a classe minoritária, porém reduzindo consideravelmente a taxa de acerto para a classe maioritária, fazendo cair drasticamente o acerto global de classificação. Portanto, observa-se que essas funções de aptidão possibilitam a indução de um classificador que seja adequado para ambas as classes independente do número de exemplos em cada conjunto de treinamento.

Bhowan et al. [7] também propuseram e avaliaram quatro novas funções de aptidão com objetivo de melhorar as taxas de acerto tanto na classe minoritária quanto na classe maioritária. A primeira usa a média geométrica entre as taxas $\frac{VP}{P}$ e $\frac{VN}{N}$ como apresentado na Equação 3.4.

$$F2.1 = \sqrt{(\frac{VP}{P}) \times (\frac{VN}{N})} \tag{3.4}$$

A média geométrica tem a propriedade de ser igual a zero sempre que um dos termos valer zero. De acordo com Bhowan et al. [7], isto faz a função de aptidão penalizar os classificadores que obtêm zero de taxa de classificação em uma das classes.

A segunda função de aptidão apresentada na Equação 3.5, baseia-se na Equação 3.2 e adiciona outro objetivo que é a própria taxa de acerto global. A ideia é que as três taxas sejam igualmente consideradas durante o processo evolutivo. Deste modo, o desempenho geral de classificação não é comprometido pela taxa de classificação de somente uma das classes.

$$F2.2 = (\frac{VP}{P}) + (\frac{VN}{N}) + (\frac{VP + VN}{P + N}) \tag{3.5}$$

A terceira função de aptidão apresentada na Equação 3.6 é composta pelas taxas de acerto na classificação de exemplos dos conjuntos P e N e também por outros dois fatores que quantificam de modo normalizado o intervalo de valores $PGout$ cujos exemplos dos conjuntos P e N foram incorretamente classificados. De acordo com Bhowan et al. [7], a ideia é valorizar classificadores com modelos de classificação mais concisos.

$$F2.3 = (\frac{VP}{P}) + (\frac{VN}{N}) + (1 - Rng_P) + (1 - Rng_N) \tag{3.6}$$

Em que:

$$Rng_C = \frac{PGout_C^{max} - PGout_C^{min}}{2} \tag{3.7}$$

Rng_C está entre (0 e 1) para todos os exemplos incorretamente classificados para a classe C, onde $PGout_C^{max}$ é o maior valor de $PGout$ do classificador para todos os exemplos incorretamente classificados da classe C e $PGout_C^{min}$ é o menor $PGout$ do classificador para todos os exemplos incorretamente classificados da classe C.

A quarta função de aptidão apresentada na Equação 3.8 é baseada na estatística Wilcoxon-Mann-Whitney (WMW) que é conhecida como um estimador para área sob a curva *Receiver Operation Characteristics* (ROC)[4].

$$F2.4 = \frac{\sum_{i=0}^{P} \sum_{j=0}^{N} I(x_i, x_j)}{N \times P} \tag{3.8}$$

Em que:

$$I(x, y) = \begin{cases} 1 & \text{se } x > 0 \text{ and } x > y \\ 0 & \text{caso contrário.} \end{cases} \tag{3.9}$$

A função $I(x, y)$ realiza a comparação entre as saídas de $PGout$ dos exemplos de cada uma das classes. A restrição $x > 0$ indica que o exemplo x_i da classe minoritária, neste

[4]A *Area Under the ROC Curve* (AUC) é área sob a curva ROC e que possibilita visualizar o desempenho do classificador em termos de custo e benefício entre acertos na classificação de uma classe e erros na outra [16].

caso da classe P, foi classificado corretamente e $x > y$ indica que o valor *PGout* do exemplo y_i, neste caso da classe N, é maior que a *PGout* de x_i. Considerando que a classe minoritária é representada pelos valores positivos de *PGout* e a maioritária pelos negativos. Deste modo, serão considerados melhores os classificadores que acertam mais na classificação de exemplos da classe minoritária e mantêm os valores de *PGout* bem separados embora possam haver erros na classificação da classe maioritária. Por isso, segundo Bhowan et al., os classificadores que apresentam valores altos para essa função de aptidão possuem modelos de classificação mais concisos.

Em seus experimentos Bhowan et al. [7] constaram que o uso da Equação 3.8 apresentou a menor diferença entre as taxas de acerto de classificação para a classe minoritária e maioritária e com maior desempenho na classe minoritária comparadas ao uso do *Accuracy* e das outras funções de aptidão propostas. As funções de aptidão representadas na equações 3.3 e 3.4 também apresentaram melhora no desempenho da classe da minoritária e a função de aptidão apresentada na Equação 3.5 promoveu aumento no desempenho da classe maioritária.

Em um trabalho posterior sobre classificação de imagens usando PG, Bhowan et al. [8] propuseram uma nova função de aptidão baseada na medida estatística *correlation ratio*, apresentada na Equação 3.10, que serve como indicador da capacidade que um classificador tem de separar classes. O objetivo desta função de aptidão é prover classificadores que obtêm bom desempenho para ambas as classes com perda mínima na taxa de acerto global, o *Accuracy*.

$$r = \sqrt{\frac{\sum_{c=1}^{M} N_c(\bar{\mu_c} - \bar{\mu})^2}{\sum_{c=1}^{M}\sum_{i=1}^{N_c}(PGout_{ci} - \bar{\mu})^2}} \tag{3.10}$$

$\bar{\mu_c}$ é a média aritmética dos valores de *PGout* para todos os exemplos N_c da classe c entre M classes e $\bar{\mu}$ é a média da *PGout* para todos os exemplos. Entretanto segundo Bhowan et al. [8], não basta que as classes estejam bem separadas, mas também que os valores *PGout* estejam dentro dos intervalos estabelecidos para cada classe, deste modo fazendo com que haja alta taxa de acerto em ambas as classes. Para isso, a função de aptidão foi redefinida para usar o *correlation ratio*, conforme Equação 3.11.

$$Correlation = r + I(\bar{\mu_p}, \bar{\mu_n}) \tag{3.11}$$

A função $I(.)$ resulta em 1 caso as média μ_p e μ_n dos valores de *PGout* retornados pelo classificador para os exemplos da classes P e N estejam nos respectivos intervalos de *PGout* determinados para cada classe. Caso contrário a função retorna 0. Desta maneira o valor de aptidão representado na Equação 3.11 varia entre 0 e 2 sendo os mais próximos de 2 os classificadores ótimos e ruins os próximos de 0. Bowman et al. [8] constataram que a função de aptidão apresentada na Equação 3.11 obteve resultados de classificação ligeiramente melhores que a função de aptidão apresentada na Equação 3.2 e com custo computacional semelhante.

3.3 Programação Genética para classificação com uma classe

Curry utilizou PG Linear para resolver o problema OCC de modo a induzir um conjunto de classificadores aptos a distinguir exemplos do conjunto de exemplos considerado normal e anomalias [10]. O problema foi resolvido utilizando anomalias artificiais geradas pelo método apresentado na Seção 2.3.4 de modo a transformá-lo em um prolema de classificação binária. Em paralelo ao processo evolutivo da PG utilizou-se o algoritmo *Balanced Block Algorithm* (BBA) para realizar amostragem no conjunto de dados de treino de modo a focar o aprendizado em exemplos mais difíceis e menos experimentados e também de reduzir o tempo computacional necessário para o treinamento [10]. Na Figura 3.2, apresenta-se como BBA foi aplicado no conjunto de dados de treino e verifica-se que a amostragem é realizada em níveis, segundo Curry, para imitar o processo de memória hierárquica.

Na PG Linear utilizada no trabalho do Curry os indivíduos da população são expressos na forma de listas de instruções e são computadas de maneira sequencial. As instruções são representadas na forma (*opcode operando*), sendo os *opcodes* pertencentes ao conjunto de operadores e os *operandos* pertencentes ao conjunto de terminais da PG. Além disso, as instruções manipulam conteúdo de um conjunto de registradores $\{R[0], ..., R[k]\}$ e o

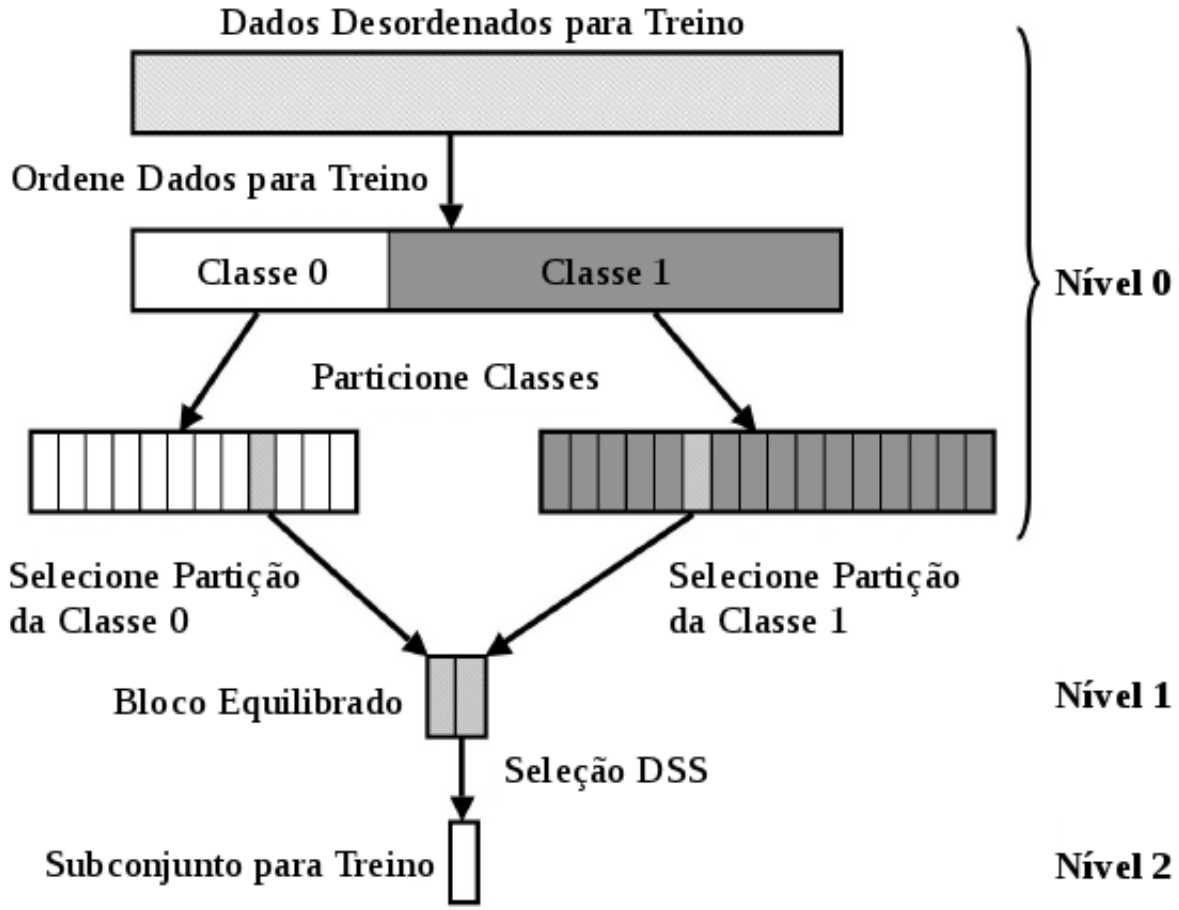


Figura 3.2: Estrutura do BBA [10]

resultado final é armazenado no registrador $R[0]$.

As instruções são agrupadas em páginas para tornar a operação genética de cruzamento menos destrutiva, pois o cruzamento consiste em trocas simples de páginas entre indivíduos. Outra característica da PG Linear utilizada naquele trabalho é o número variável de instruções por páginas utilizado para facilitar a seleção de uma quantidade específica de instruções por página na inicialização e por isto, esse tipo de PG é chamado de *Dynamic Page-based Linear GP* (DPLGP).

Apesar de não ter sido mencionada a motivação para o uso da DPLGP para o problema OCC, verifica-se que sua aplicação é voltada a problemas de tempo real em que há recursos computacionais limitados para o treinamento do classificador [35]. Isto se deve ao fato de que os indivíduos são representados como páginas de instruções e, portanto, apresentando pouca complexidade.

Após a geração das anomalias artificiais, a classe normal e a classe anormal composta

pelas anomalias são particionadas conforme nível 0 da Figura 3.2. Posteriormente inicia-se o treinamento, então o algoritmo de amostragem *Dynamic Subset Selection* (DSS) seleciona uma partição de cada classe de modo que as mais difíceis de classificar ou menos utilizadas até então tenham mais chances de serem selecionadas. O resultado é o bloco de treino do nível 1 conforme Figura 3.2. O DSS é aplicado novamente sob o bloco do nível 1 para selecionar exemplos normais e anormais que irão compor o bloco equilibrado que será utilizado no treinamento, nível 2 da Figura 3.2.

No próximo passo, utiliza-se a estratégia de campeonato que consiste em selecionar aleatoriamente n indivíduos que irão competir e os campeões ou considerados mais bem adaptados dentre os escolhidos serão utilizados via operadores genéticos para atualizar a população da PG, isto é, compor a nova geração de indivíduos. Para isso, todo o programa do campeonato é executado para cada exemplo do bloco equilibrado, nível 2 da Figura 3.2, e o conjunto de valores de saída *PGout* é registrado. Nesse ponto, segundo Curry [10], em um problema OCC as anomalias tendem a possuir valores de *PGout* tanto positivos como negativos, inviabilizando a tradução de valores em classes, como apresentado na Seção 3.2.1.

Curry também considera que exemplos da classe normal terão valores de *PGout* agrupados em um intervalo comum. Como os valores de *PGout* estarão dispostos de modo irregular Curry propôs usar a medida *Class Separation Distance* (CSD) para identificar o melhor agrupamento da classe normal. Deste modo, os valores de *PGout* são agrupados conforme a classe e os grupos são comparados aos pares utilizando a CSD como apresentada na Equação 3.12, onde μ_P e μ_N representam respectivamente os valores *PGout* médios do grupo normal P e do anormal N e σ_P e σ_N os respectivos desvios padrão de cada classe. O grupo com maior valor de $CSD_{P/N}$ será considerado o melhor agrupamento para a classe normal.

$$CSD_{P/N} = \frac{|\mu_P - \mu_N|}{\sqrt{\sigma_P^2 - \sigma_N^2}}, \tag{3.12}$$

Curry utilizou uma função de aptidão composta por vários objetivos. Um deles é a maximização do valor CSD, ou seja, privilegiar os indivíduo cujos os valores de *PGout*

para os exemplos da classe normal estão mais destacados dos exemplos da classe anormal. Outro objetivo é maximizar a quantidade de valores *PGout* que aparecem no melhor agrupamento da classe normal identificado via CSD, isto é, o indivíduo cujos valores de *PGout* dos exemplos da classe normal estão mais concentrados. O terceiro objetivo é privilegiar programas que possuem os melhores agrupamentos distintos de valores *PGout* da classe normal, fortalecendo a diversidade de classificadores pela minimização da sobreposição de regiões da classe normal. O quarto objetivo é minimizar o tamanho do programa quanto ao número de instruções visando à obtenção de melhor generalização.

Como há quatro objetivos concorrentes, foi utilizado o conceito de *Ranking* de Pareto para comparar e definir os campeões do torneio. Os dois programas menos dominados do *Ranking* são utilizados para criar novos indivíduos via operadores genéticos e estes substituem os perdedores do campeonato atualizando a população. Os programas vencedores também são utilizados para atualizar os fatores dificuldade e idade dos exemplos do bloco equilibrado para influenciar a seleção de novos exemplos pelo DSS na próxima iteração do algoritmo.

A proposta do Curry também visa manter conjuntos de classificadores para cada partição do conjunto de exemplos normais, nível 0 da Figura 3.2, de modo a decompor o problema de classificação. Para identificar os melhores programas para classificar a partição, todos são executados para os exemplos do bloco do nível 1 da Figura 3.2. Então, para cada programa a medida CSD é utilizada novamente para identificar o melhor agrupamento de valores *PGout* para os exemplos da classe normal e a média μ e desvio σ do agrupamento são utilizados em uma função chamada *Local Membership Function* (LMF), ilustrada Figura 3.3. O valor *PGout* retornado pelo classificador para um exemplo é um dos argumentos dessa função, apresentada na Equação 3.13, e o seu valor de saída ($confidence_i$) é comparado com o valor de limiar (*cutoff*) e caso este seja maior que o limiar o exemplo é considerado como pertencente a classe normal.

$$confidence_i = exp(-\frac{(PGout(i) - \mu)^2}{2 \times \sigma^2}) \tag{3.13}$$

Nesta etapa, os programas também são avaliados utilizando quatro objetivos, os mes-

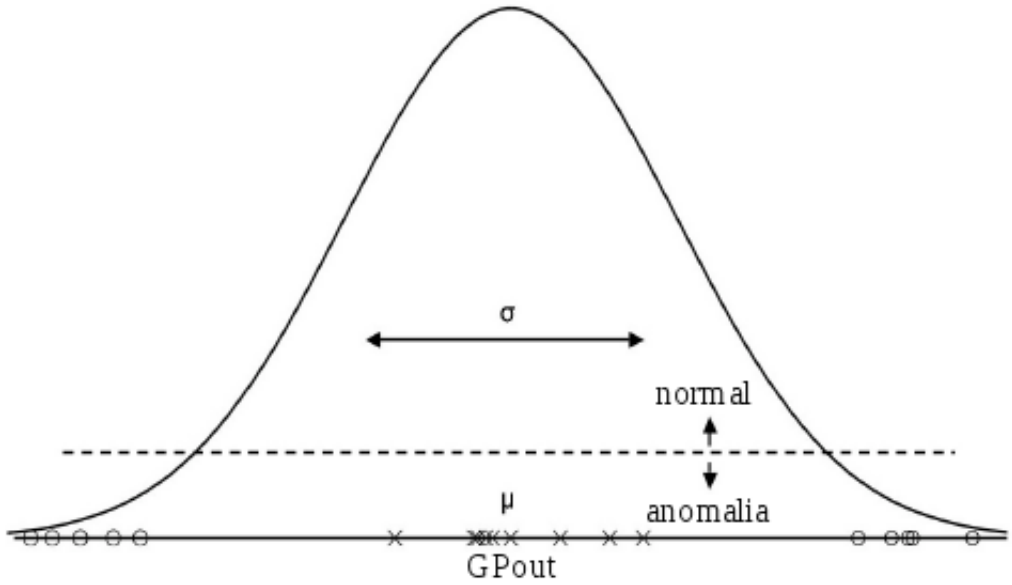


Figura 3.3: *Local Membership Function* [10]

mos da fase anterior, com exceção a maximização do CSD que é substituída pela minimização da taxa de erro de classificação do programa. Os programas da população são comparados utilizando o conceito de Dominância de Pareto e os programas não dominados, isto é, que compõem a Fronteira de Pareto, atualizam o conjunto de classificadores da partição usando a seguinte regra: se a Fronteira de Pareto é maior que o conjunto de programas classificadores da partição, os programas da fronteira são analisados entre si em termos de similaridade e aqueles mais distintos completam o conjunto de programas classificadores da partição. Caso contrário, se a Fronteira de Pareto for menor, os conjuntos de programas da partição são organizados segundo o conceito de *Ranking* de Pareto explicado anteriormente, e os últimos do *Ranking* são substituídos pelos da Fronteira de Pareto.

Uma vez que o conjunto de programas da partição é atualizado, eles são executados para os exemplos do bloco nível 1 e utiliza-se novamente a LMF para computar as taxa de erro dos exemplos normais e anormais e estas atualizam a dificuldade das partições de cada classe do bloco nível 1. As partições utilizadas no bloco do nível 1 têm sua idade incrementada e as demais são ajustadas para o valor zero. Deste modo, na próxima formação de bloco nível 1 o DSS irá selecionar as partições mais difíceis de classificar - altas taxas de erro - e menos utilizadas. No último passo do algoritmo os programas

especializados na classificação de cada partição de exemplos normais são executados sobre o conjunto de treino inteiro e aqueles com altas taxas de erro de classificação global são removidos. Esta taxa de erro é também computada utilizando a LMF e os classificadores duplicados são removidos do conjunto final de programas classificadores.

Para testar o desempenho de classificação do conjunto final de programas classificadores utiliza-se votação. Porém, o número de votos necessários para classificar os exemplos do conjunto de teste depende do nível de minimização das taxas de erros na classe desejada. Minimização máxima implica na computação dos votos de todos os programas classificadores. Ao final as taxas de erro em cada classe e o *Accuracy* são computados. O algoritmo proposto por Curry para o problema OCC está apresentado no Algoritmo 4.

O Algoritmo 4 proposto foi experimentado em três conjuntos de dados e comparado com o algoritmo ν-SVM com e sem o uso de anomalias artificiais. Em termos de desempenho de classificação o Algoritmo 4 mostrou-se equiparável ao uso do ν-SVM sem anomalias e superior comparado ao mesmo algoritmo executado de modo a resolver um problema de classificação binário com a presença de anomalias artificiais.

Curry propôs mudanças no Algoritmo 4 em um trabalho posterior [11], entre elas o uso da técnica de agrupamento *Subtractive Clustering* para identificar o intervalo de valores de *PGout* mais denso e independente de classe para a aplicação da LMF, Equação 3.13. Essa abordagem substitui a identificação do melhor grupo de valores de *PGout* para exemplos da classe normal com o uso da medida CSD, Equação 3.12.

Outra mudança importante foi a modificação do raio R da hiperesfera calculada para gerar as anomalias no método apresentado na Seção 2.3.4. Este parâmetro é automaticamente identificado por meio do maior intervalo de valores dentre os atributos do conjunto de exemplos da classe normal. Segundo Curry, se houver um atributo onde o intervalo é muito maior que os demais a hiperesfera calculada será muito maior do que o necessário gerando anomalias artificiais com pouca relevância. Para resolver esse problema foi empregado um vetor $\bar{R}$ composto dos intervalo de valores para cada atributo de modo à calcular uma hiperesfera mais representativa da classe normal e portanto possibilitando que anomalias de melhor qualidade sejam geradas.

Algoritmo 4 DPLGP com o uso do BBA e do DSS

1: Gere anomalias artificiais
2: Particione as classes normal e de anomalias (BBA nível 0)
3: **Enquanto** Nível 1 término == Falso **faça**
4: Use o DSS para selecionar uma partição da classe normal e outra da anormal (BBA nível 1)
5: **Enquanto** Nível 2 término == Falso **faça**
6: Use o DSS para selecionar exemplos normais e anormais (BBA nível 2)
7: Selecione programas da PG para o campeonato
8: **Para Todos** Programas do Campeonato **faça**
9: **Para Todos** Exemplos do bloco nível 2 **faça**
10: Execute o programa para o exemplo e registre o valor *PGout*
11: **Fim Para**
12: Determine μ e σ^2 para cada grupo de valores *PGout*
13: Determine o melhor grupo de valores *PGout* para classe normal e os grupos vizinhos de anomalias utilizando CSD
14: Registre os objetivos para função de aptidão: o CSD do melhor grupo de exemplos normais, o número de exemplos normais presentes no grupo, o número de instruções do programa e o *class overlap*
15: **Fim Para**
16: Utilize *Ranking* de Pareto para determinar os vencedores do campeonato nos objetivos da função de aptidão
17: Aplique operadores genéticos nos dois primeiros colocados e atualize a população da PG
18: Atualize a dificuldade e idade dos exemplos do bloco do nível 2 do BBA
19: **Fim Enquanto**
20: **Para Todos** Programas da População **faça**
21: **Para Todos** Exemplos do bloco nível 1 **faça**
22: Execute o programa para o exemplo e registre o valor *PGout*
23: **Fim Para**
24: Determine μ e σ^2 para cada grupo de valores *PGout*
25: Determine o melhor grupo de valores *PGout* para classe normal e os grupos vizinhos de anomalias utilizando CSD
26: Registre os objetivos da função de aptidão: taxa de erro de classificação de exemplos usando a LMF sob o melhor grupo de exemplos normais, o número de exemplos normais presentes no grupo, o número de instruções do programa e o *class overlap*
27: **Fim Para**
28: Determine os programas da fronteira de Pareto utilizando objetivos da função de aptidão
29: Utilize a fronteira de Pareto para atualizar o conjunto de classificadores da partição de conjuntos normal do bloco do nível 1
30: Atualize a dificuldade e a idade das partições das classes no bloco nível 1
31: **Fim Enquanto**
32: Remova programas dos conjuntos de classificadores das partições da classe normal que apresentam baixo desempenho em classificação ou são iguais

O novo algoritmo foi experimentado em seis conjuntos de dados e comparado com o algoritmo ν-SVM sem o uso de anomalias e com uma rede neural *Bottleneck Neural Network* (BNN). Em geral mostrou-se, em termos de desempenho de classificação, ligeiramente melhor que o trabalho original, Algoritmo 4, melhor que o ν-SVM e equiparável à BNN.

3.4 Considerações finais

No presente Capítulo, foi apresentada a PG, uma meta-heurística para buscar a solução ótima para problemas complexos em um espaço de soluções, neste caso, o classificador com melhor desempenho de classificação em um espaço de busca composto por classificadores. O bloco evolutivo da PG para classificação foi apresentado na Seção 3.2, bem como suas vantagens e desvantagens. Nas seções 3.2.1 e 3.2.2 foram discutidos temas especificamente relacionados à classificação, como os métodos de tradução de valores de saída dos classificadores em classe. Também apresentaram-se as Funções de Aptidão, necessárias para guiar a PG na busca pelo melhor classificador.

Na Seção 3.3 foi apresentado um trabalho de PG Linear para OCC, porém constatou-se que muitas mudanças foram introduzidas para melhorar o desempenho computacional na etapa de treino do classificador, deste modo, dificultando a análise dos pontos fortes e fracos do algoritmo em termos de distinguir corretamente exemplos desconhecidos das classes normal e anormal. Além do mais, constatou-se a inexistência de estudos do algoritmo de PG padrão para OCC. No Capítulo 4, discutem-se em detalhes essas lacunas e apresenta-se a abordagem de PG proposta no presente trabalho.

CAPÍTULO 4

ABORDAGEM PROPOSTA

4.1 Motivação

A PG é aplicada com sucesso em vários problemas de classificação, entretanto, verifica-se na literatura que há apenas dois trabalhos relacionados à PG voltados para OCC e oriundos dos mesmos autores: *One-Class Genetic Programming* (OCGP) [10] e *One-Class Genetic Programming Clustering* (OCGPC) [11]. Como discutido no Capítulo 3 e Seção 3.3, algumas ideias foram por eles propostas para tratar de problemas específicos da abordagem utilizada para resolver OCC; outras, com a finalidade de reduzir o tempo computacional gasto na etapa de treino do classificador PG.

Consequentemente a diversidade de mudanças empregadas torna a abordagem de PG proposta específica e repleta de etapas cujo impacto individual ainda não foi avaliado experimentalmente. Todavia, é notável que para tornar a PG um algoritmo de excelência para OCC o primeiro passo é realizar a avaliação de sua abordagem convencional, algo que ainda não foi realizado e é objetivo central do presente trabalho, pois, a introdução de novas ideias, aparatos e sua avaliação, somente se justificaria por meio dos resultados desse estudo. Nas Seções 4.2 e 4.3, trata-se individualmente de aspectos importantes e as escolhas realizadas para compor a abordagem convencional de PG proposta.

4.2 Geração de anomalias

Como discutido no Capitulo 3 e Seção 3.2, para resolver o problema OCC utilizando classificadores como a PG, é necessário transformá-lo em um problema de classificação binária. Deste modo, o conjunto de dados utilizado para treinar o classificador PG é composto por uma classe de exemplos considerados normais cujo perfil é conhecido e por outra classe composta por exemplos anormais. Nesse contexto, faz-se necessário um

algoritmo para gerar tais exemplos, principalmente em problemas práticos de OCC em que não é possível obter esse perfil. Outrora, em detecção de falhas em máquina, por exemplo, seria necessário danificá-la ou induzi-la ao comportamento falho para obter exemplos anormais. Além disso, há trabalhos que mostram que o uso de anomalias artificiais não apenas viabiliza a aplicação de vários tipos de classificadores mas que também apresenta resultados melhores quando comparados a algoritmos que usam apenas exemplos normais em seu processo de aprendizado [6] [19].

Uma das estratégias mais simples para gerar anomalias foi apresentada na Seção 2.3.1 do Capítulo 2. O algoritmo baseia-se nos valores máximos dos atributos dos exemplos normais para gerar os anormais. Por considerar apenas dois atributos, o método proposto gera exemplos que podem não corresponder a anomalias de fato, principalmente em problemas com muitos atributos.

Outro algoritmo de geração de anomalias discutido na Seção 2.3.3, baseia-se na distribuição dos exemplos normais no espaço de atributos. Esse método apresenta melhores resultados para detecção de intrusão em rede de computador comparado a outras técnicas aleatórias. Porém, seu problema é o uso da frequência de valores dos atributos para exemplos normais e sendo assim, é aplicado somente em problemas de classificação com atributos discretos.

O algoritmo que se baseia no conceito de hiperesfera apresentado na Seção 2.3.4, o método que utiliza exemplos normais distantes apresentado na Seção 2.3.2 e a técnica RNS baseada em princípios imunológicos apresentada na Seção 2.3.5 não possuem os problemas anteriormente citados e são avaliados experimentalmente no presente trabalho de modo a produzir exemplos para composição da classe que representa o perfil anormal. Nesse contexto, um importante relato é a inexistência de trabalhos na literatura que avaliem diferentes métodos de geração de anomalias para treinar classificadores OCC — esta é outra importante contribuição do presente trabalho.

Em específico, o método baseado em hiperesfera foi utilizado no trabalho relacionado proposto por Curry OCGP [10] e OCGPC [11]. Entretanto, seu uso resulta em conjuntos de dados desequilibrados, pois Tax recomenda que muitas anomalias sejam geradas para

que os exemplos normais estejam cobertos em todas as direções no espaço de atributos [38]. A estratégia de Curry para evitar esse desequilíbrio está no uso do algoritmo BBA, pois, a cada amostragem realizada pela técnica, uma partição é composta pela mesma quantidade de exemplos normais e anormais e o treinamento da PG é realizado sob as partições. Para o presente trabalho é proposto que seja gerada a mesma quantidade de exemplos da classe normal existente no conjunto de dados, pois não há estudos comprovando que, de fato, a quantidade de exemplos de perfil anormal utilizados no conjunto de treino tenha impacto direto no desempenho final de classificação.

Outra característica do uso do método baseado em hiperesfera é a mistura de exemplos normais e anomalias compartilhando a mesma região no espaço de atributos, e isto, de acordo com Curry, dificulta o estabelecimento de um valor de limiar usado na PG para determinar a qual classe pertence o exemplo em avaliação [10]. Observa-se também que anomalias misturadas com exemplos normais podem não oferecer condições que caracterizem propriedades da classe normal fundamentais no processo indutivo do classificador como a sua distribuição e seus limites no espaço de atributos. Portanto, a fim de tratar a condição de mistura de exemplos, de modo a manter a PG livre de mudanças, propõe-se, para esse método de geração de anomalias, a aplicação de uma etapa adicional de filtragem de anomalias que será tratada em detalhes na Seção 4.2.1.

A respeito do algoritmo que se baseia nos exemplos normais mais distantes apresentado na Seção 2.3.2, verifica-se que não causa conjuntos de dados de treino desequilibrados, pois uma anomalia é gerada para cada exemplo da classe normal e também que não há condição de mistura de exemplos, pois como discutido na Seção 2.3.2 e demonstrado na Figura 2.2 os exemplos anormais são posicionados de modo a ocupar o espaço externo da classe normal. Portanto, observa-se que exemplos produzidos por esse algoritmo têm o benefício de não requererem mudanças em aspectos internos da PG e etapas de tratamento adicional

Com relação ao método RNS, também objeto de avaliação no presente trabalho, verifica-se que as anomalias, neste caso referenciadas como detectores, conforme discutido na Seção 2.3.5, tendem durante a execução do algoritmo a afastarem-se dos exemplos

normais, minimizando a condição de mistura de exemplos. A questão de desequilíbrio no conjunto de dados de treino entre as classes pode ser tratada ajustando o número de detectores a serem gerados. Por esses fatores, nenhum ajuste adicional foi proposto para o RNS.

4.2.1 Proposta de ajuste no algoritmo baseado em hiperesfera

O algoritmo que se baseia no conceito de hiperesfera descrito na Seção 2.3.4, calcula na primeira etapa a hiperesfera que envolve a classe de exemplos normais. A hiperesfera é composta por um vetor que corresponde à posição central dos exemplos normais no espaço de atributos e também por um raio que é a distância euclidiana entre o centro e os limites da classe normal. Na segunda etapa, os exemplos anormais são gerados dentro da hiperesfera com distribuição uniforme. O algoritmo possui o parâmetro dR que é o fator multiplicador usado para redimensionar o raio da hiperesfera, em que 1.0 significa manter o tamanho do raio original calculado na primeira etapa, 1.1 significa aumentar o raio em 10%, 1.6 significa aumentar o raio de geração das anomalias em 60% e assim por diante.

Ao aumentar o raio da hiperesfera, verifica-se a possibilidade de gerar anomalias fora da distribuição de exemplos normais. Deste modo, com o objetivo de evitar o problema de mistura de exemplos das diferentes classes, propõe-se neste trabalho, a geração de anomalias com o raio aumentado ($dR > 1.0$) e uma etapa adicional para remoção das anomalias geradas dentro da hiperesfera que envolve os exemplos da classe normal. Assim, descartam-se os exemplos cuja distância euclidiana do vetor centro da classe normal é menor que o raio original da hiperesfera calculado na primeira etapa do algoritmo. Verifica-se, portanto, que os exemplos não removidos de acordo com essa regra estarão fora da classe normal e de modo não misturado.

Também observa-se que o raio usado para remover exemplos pode ser aumentado de modo que restem apenas exemplos mais afastados da classe normal. Neste contexto, a fim de avaliar experimentalmente diferentes conjuntos de exemplos anormais, propõe-se o uso de dois parâmetros: o próprio dR usado para gerar as anomalias e o parâmetro dF, que é o fator multiplicador do raio original e usado para estabelecer o raio de remoção

das anomalias. Verifica-se que, para que restem anomalias ao final da remoção, a regra $dR > dF$ deve ser respeitada. Logo, se dR_1 e dF_1 são maiores que dR_2 e dF_2, as anomalias geradas utilizando a configuração dR_1 e dF_1 estarão mais afastadas da classe normal do que aquelas geradas utilizando dR_2 e dF_2.

Apesar de existir uma relação de proporção entre dR e dF, se desconhece a maneira como usá-la para determinar o número exato de anomalias a serem geradas tal que as internas sejam removidas e o número de restantes seja igual à quantidade desejada para o conjunto de treino. Por isso, recomenda-se que o processo de geração seguido da remoção de anomalias seja executado tantas vezes quanto necessário usando diferentes valores de dR e dF para que o número de exemplos desejado seja obtido.

4.3 Aspectos da abordagem de PG proposta

O modelo de representação utilizado para expressar as regras de classificação que compõem cada indivíduo da população de classificadores da PG proposta no presente trabalho é a Função Discriminante. Deste modo, como discutido na Seção 3.2, o classificador é uma expressão matemática composta por operadores aritméticos cujas variáveis são atributos do conjunto de dados. A Função Discriminante também foi utilizada na PG proposta por Curry OCGP [10] e OCGPC [11], e é comumente empregada em outros algoritmos de PG para problemas contínuos e de reconhecimento de padrões em áreas como processamento de imagens e diagnóstico médico.

No presente trabalho, a expressão matemática que compõe o classificador é representada na forma de árvore. Essa estrutura de dados foi proposta por Koza para codificar o genótipo do indivíduo, possibilitando que a operação genética de cruzamento seja realizada trocando subárvores de diferentes indivíduos e a mutação trocando subárvores do mesmo indivíduo, conforme Seção 3.2 e Figural 3.1.

A proposta de Curry utiliza a PG Linear, cuja principal característica é codificar os classificadores na forma de lista de instruções. Essa é uma estrutura de dados mais simples que a de árvore, favorecendo o baixo custo computacional para processar os classificadores e realizar operações genéticas sob eles. Entretanto, apesar deste benefício, observa-se que

há poucos trabalhos [35] verificando a influência desse tipo de PG no desempenho de classificação comparados à PG baseada em árvores como originalmente proposta por Koza [23]. Relata-se também a inexistência de estudos diretamente relacionados com OCC e argumenta-se que se o classificador consegue obter o grau de generalização adequado é desnecessário retreiná-lo, assim minimizando o impacto do tempo computacional gasto na etapa de treino na resolução de OCC por meio da PG. Por esses motivos e a fim de manter compatibilidade com o objetivo central do presente trabalho, que é avaliar a PG convencional para OCC, usar-se-á a abordagem baseada em árvores.

O valor *PGout*, como discutido na Seção 3.2.1, é o resultado da aplicação de um exemplo sob a expressão aritmética que compõe o classificador. Através de um valor de limiar determina-se a classe do exemplo. Em uma abordagem convencional de PG para classificação binária é comum utilizar zero como valor de limiar. Deste modo, exemplos cujos valores *PGout* são positivos, serão considerados como pertencentes a uma das classes e aqueles com valores negativos como pertencentes a outra classe.

Como observado na Seção 3.3, em OCC o método de geração de anomalias empregado pode causar a condição de mistura de exemplos dificultando o estabelecimento do valor de limiar para a PG. Curry propôs uma técnica de agrupamento nos valores *PGout* para identificar o grupo de valores que melhor descreve os exemplos normais e este foi usado de modo que exemplos com valores *PGout* mais próximos do valor médio do grupo tenham mais chances de serem classificados como normais e os mais distantes como anomalias. Constata-se que essa foi outra mudança no modo convencional de resolver OCC como classificação binária usando PG cujo impacto também não foi devidamente avaliado.

Com discutido na Seção 4.2, a mistura de exemplos normais e anormais ocupando a mesma região no espaço de atributos pode dificultar o estabelecimento de um valor de limiar de classificação e associa-se essa condição ao método de geração de anomalias empregado. Observa-se que o algoritmos de geração de anomalias, discutidos na Seção 4.2 e objetos de avaliação neste trabalho, não têm como consequência a mistura de exemplos ou esta condição é mínima em seu uso. Inclusive para o método baseado em hiperesfera, se propôs a filtragem das anomalias que estejam posicionadas entre os exemplos da classe

normal, conforme apresentado na Seção 4.2.1. Deste modo, ao contrário dos algoritmos OCGP e OCGPC, mantém-se a abordagem proposta livre de mudanças em relação a esse aspecto. Sendo assim, propõe-se o uso da regra expressa na Equação 4.1 para classificar exemplos:

$$\text{classe} = \begin{cases} \text{normal}, & \text{se } PGout \geq 0 \\ \text{anomalia}, & \text{caso contrário.} \end{cases} \tag{4.1}$$

Então, programas classificadores da população da PG traduzem os valores dos atributos dos exemplos por meio de sua árvore de expressão para o valor único *PGout* que é usado para determinar a classe do exemplo usando zero como valor limiar. Na Seção 4.3.1, a seguir, discute-se outro importante aspecto da PG para OCC que é a função de aptidão utilizada.

4.3.1 Função de aptidão

A função de aptidão é um importante componente da PG, pois possibilita avaliar o desempenho de cada indivíduo de sua população mediante o estabelecimento de um ou vários objetivos a serem otimizados. As soluções que apresentam os melhores valores aptidão têm mais chances de serem selecionadas para compor novos indivíduos para a próxima geração, por isto a função de aptidão direciona o processo de otimização da PG na busca das melhores soluções. Na PG proposta por Curry para OCC, conforme Seção 3.3, a aptidão de cada classificador é avaliada de acordo com quatro objetivos distintos:

- Maximizar o valor CSD para o grupo de valores que melhor caracteriza os exemplos da classe normal;
- Maximizar a quantidade de valores *PGout* no grupo de valores que melhor caracteriza os exemplos da classe normal;
- Minimizar a sobreposição de valores *PGout* de diferentes classes;
- Minimizar o número de instruções no programa.

Observa-se que Curry não apresenta evidências que esses objetivos estão diretamente relacionados com métricas de desempenho de classificação, pois visam a melhoria de características internas do classificador, que segundo o autor, podem ao final resultar em classificadores que apresentam melhores índices de acerto na identificação de exemplos. Nesse contexto, não foram apresentados resultados sobre o impacto individual de cada objetivo, no entanto, foram adotados como critério de seleção em uma importante etapa da PG, pois indivíduos da nova geração serão constituídos de material genético dos selecionados de acordo com a função de aptidão. Como o problema em questão é de classificação, questiona-se o fato de não terem sido utilizados como objetivo métricas para mensurar a aptidão do classificador ao identificar corretamente exemplos nesta importante etapa da PG, que é definição dos indivíduos que farão parte da próxima geração.

Na Seção 3.2.2 foram estudadas funções de aptidão da PG utilizadas em problemas de classificação, verifica-se que em todas levam-se em consideração métricas de desempenho de classificação e não apenas características do classificador. A taxa geral de classificação *Accuracy*, por exemplo, é frequentemente usada para avaliar a aptidão do classificador em técnicas evolutivas e uma vantagem em seu uso é o baixo custo computacional envolvido [7], [31]. Para a PG proposta neste trabalho deseja-se manter esse requisito e ter equilíbrio entre as taxas de classificação de cada classe de modo a penalizar classificadores que apresentam desempenho baixo para classificar corretamente exemplos normais ou anormais. Neste sentido, propõe-se o uso da função de aptidão representada na Equação 4.2, que atende ao objetivo proposto e também foi empregada em problemas OCC cujos conjuntos de dados de treino estão desequilibrados [31].

$$Fitness = \frac{(\frac{VP}{P})^2 + (\frac{VN}{N})^2}{2} \tag{4.2}$$

Para o presente trabalho $\frac{VP}{P}$ é a taxa de exemplos normais classificados corretamente e $\frac{VN}{N}$ representa a taxa de exemplos anormais classificados corretamente. Ambas estão elevadas ao quadrado para penalizar baixo desempenho de classificação por classe e também divididas por dois para enquadrar o valor de aptidão resultante no intervalo $[0, 1]$.

4.4 Considerações finais

Neste Capítulo foram abordados aspectos relacionados à geração de anomalias e também relacionados à abordagem de PG proposta. No Capítulo 5 serão apresentados a metodologia experimental, detalhes da implementação da PG e os resultados obtidos na sua aplicação usando anomalias geradas pelos algoritmos comentados previamente. Há também na Seção 5.4 um estudo comparativo avaliando a proposta de PG do presente trabalho para OCC diante do principal trabalho da área proposto por Curry [11]. Em outro experimento na Seção 5.7 avaliam-se os resultados da variação dos parâmetros da PG considerados mais relevantes para OCC.

CAPÍTULO 5

AVALIAÇÃO EXPERIMENTAL

5.1 Introdução

No presente Capítulo apresentar-se-ão os experimentos realizados com a abordagem de PG proposta para avaliar seu desempenho de classificação em problemas de OCC. Como discutido na Seção 4.2, um importante aspecto ao resolver OCC como classificação binária está na geração de anomalias. Por isso, primeiro são realizados experimentos com a PG proposta utilizando exemplos de perfil anormal, gerados por cada algoritmo discutido na Seção 4.2. Para cada método, escolheu-se uma variação de valores em seus parâmetros de modo que a influência de seu ajuste também seja avaliada. Ao final desta etapa experimental, os melhores resultados obtidos pela PG para os algoritmos de geração de anomalias são comparados. O método e a respectiva configuração de parâmetros que proporciona em geral os melhores resultados é escolhido para ser aplicado nos demais experimentos em que são avaliados outros aspectos da PG proposta. A metodologia empregada nessa etapa é explicada em detalhes na Seção 5.3.

Na segunda etapa experimental, realiza-se um estudo comparativo entre a abordagem de PG proposta no presente trabalho e o algoritmo de PG do atual estado da arte para OCC proposto por Curry e apresentado na Seção 3.3. A metodologia experimental empregada por Curry foi reproduzida e a PG proposta no presente trabalho foi executada sob as mesmas condições para viabilizar a comparação. Inclusive, elabora-se a curva ROC utilizada para visualizar o desempenho de classificação e também aplicada como aparato comparativo no trabalho de Curry. A metodologia e os resultados são apresentados em detalhes na Seção 5.4.

Na terceira etapa experimental realiza-se um estudo para avaliar determinadas funções de aptidão da PG e também comparar com aquela proposta na Seção 4.3.1, Equação 4.2 e utilizada nos demais experimentos. Em específico, há a hipótese de que melhores

resultados podem ser obtidos ao utilizar métricas que exibem o desempenho geral de classificação como função de aptidão. Neste caso, avalia-se a métrica AUC e a WMW que corresponde ao valor estimado da AUC. A metodologia e os resultados para avaliar as funções de aptidão comentadas são apresentados em detalhes na Seção 5.5.

Na quarta etapa experimental, realiza-se outro estudo comparativo para avaliar o desempenho de classificação em OCC obtido pela PG proposta e outros tipos de classificadores como ν-SVM e a Rede Neural Multicamada, cujas implementações estão disponíveis no *software* Weka [21]. A metodologia e os resultados obtidos neste estudo são apresentados na Seção 5.6.

Por fim, realiza-se na última etapa uma análise experimental do impacto da variação dos parâmetros da PG proposta que são considerados mais influentes nos resultados de classificação em OCC. A metodologia e os resultados obtidos com esse experimento são apresentados na Seção 5.7.

A metodologia experimental é diferenciada em determinadas etapas experimentais visando à comparação com os resultados obtidos em outros trabalhos. Entretanto, há itens da metodologia que são comuns a várias etapas. Estes são apresentados a seguir, na Seção 5.2.

5.2 Metodologia geral

A implementação da proposta usou o algoritmo PG padrão baseado em árvores disponível no *framework* C++ OpenBeagle [9]. Uma configuração padrão da PG foi estabelecida para experimentos cujo principal aspecto em avaliação não é a própria parametrização da PG, e apresenta-se na Tabela 5.1. Independentemente da configuração, a PG foi executada 30 vezes em cada configuração experimental avaliada.

A configuração apresentada na Tabela 5.1 foi determinada empiricamente, pois verifica-se na literatura que não há estudos avaliando o impacto dos parâmetros em problemas de classificação e não há consenso dada a diversidade de trabalhos existentes. Deste modo, o tamanho da população foi ajustado para 10 vezes o número de gerações, pois verifica-se que o tamanho da população determina o espaço de busca em que a PG atuará para

Tabela 5.1: Configuração de parâmetros da PG

Parâmetro	Valor
Inicialização	*Half-and-Half*
Tamanho da População	2.000
Gerações	200
Prob. Cruzamento, Mutação	85%, 15%
Tam. Campeonato	4
Tam. Min. Arvore	4
Tam. Max. Arvore	12
Conjunto Operadores	$+, -, /, *, sin, cos, log, exp$
Conjunto Terminais	atributos do conjunto de dados

obter o melhor classificador e então estabeleceu-se um valor proporcionalmente maior ao de gerações. A mutação possui o objetivo de manter a diversidade genética da população ao longo das gerações, porém se for predominantemente aplicada há deterioração do material genético desenvolvido ao longo do processo. Sendo assim, estabeleceu-se 15% de chances de sua aplicação, enquanto a operação de cruzamento possui 85% de chances. Foi estipulado o torneio como método de seleção, pois este visa controlar a pressão seletiva para que o processo evolucionário não estacione em soluções subótimas. Já o tamanho da árvore pode estar relacionado com o número de atributos do problema de classificação. Neste caso, notavelmente verifica-se que se o número de dimensões do problema é alto e o tamanho da árvore é baixo, não haverá nós suficientes na árvore para representar ao menos os atributos mais importantes. Deste modo, o valor estabelecido varia de 4 a 12 — triplicando o valor mínimo — e também considerando que árvores de tamanho excessivo podem inviabilizar os experimentos em função do tempo computacional usado pela PG na etapa de treino. O conjunto de operadores é composto de operações aritméticas básicas também presentes no trabalho de Curry e, adicionalmente, pelos operadores sin, cos, log e exp.

O algoritmo de PG resulta em uma população de classificadores e no presente trabalho o desempenho final de classificação da PG foi avaliado para o classificador com melhor valor de aptidão na etapa de treino ao longo de todas as gerações. As métricas utilizadas para avaliar o desempenho geral de classificação foram a taxa geral de acerto *Accuracy* - taxa geral de acerto na classificação de exemplos (Acc) e a AUC.

A *AUC* é o valor da área sob a curva ROC cujos pontos são obtidos computando a taxa de acerto para uma das classes e a taxa de erro para outra ao usar diferentes valores de limiar. Em modelos de classificação binários cujo classificador provê a probabilidade do exemplo pertencer a uma das classes, o valor limiar de classificação é variado entre 0 e 1 [16]. Na PG proposta, como discutido na Seção 4.3, o exemplo é identificado por meio do valor *PGout* resultante da aplicação do classificador sob o exemplo. Neste modelo de classificação a curva ROC é determinada variando o valor de limiar entre os valores mínimo e máximo de *PGout* resultantes da aplicação dos exemplos do conjunto de teste no classificador. No presente trabalho, a *AUC* é calculada de modo a somar os trapezoides existentes abaixo dos pontos da curva [16].

Também foram consideradas na avaliação do desempenho de classificação as taxas individuais de acerto em cada classe: a taxa de acerto para exemplos da classe normal *True Positive rate* - taxa de acerto para a classe normal (TPr) e a taxa de acerto para exemplos da classe anormal *True Negative rate* - taxa de acerto para a classe anormal (TNr). As taxas *Acc*, TPr e TNr são calculadas das seguintes maneiras:

$$Acc = \frac{\text{Exemplos corretamente classficados}}{\text{Total de Exemplos}},$$
$$TPr = \frac{\text{Normais corretamente classificados}}{\text{Total de Normais}},$$
$$TNr = \frac{\text{Anormais corretamente classificados}}{\text{Total de Anormais}}.$$

No presente trabalho também utiliza-se em alguns experimentos uma medida de disparidade entre as taxas de acerto das classes ($|TPr - TNr|$). Quando a disparidade é alta o classificador considera com alta probabilidade quaisquer exemplos como pertencentes a uma das classes e, consequentemente, apresenta desempenho de classificação inferior para a outra classe. Neste caso, a disparidade revela que o grau de generalização adequado não foi obtido na etapa de treino. Embora as taxas gerais *Acc* e *AUC* sejam semelhantes em diferentes configurações experimentais, aquela que proporcionar ao classificador menor disparidade terá apresentando melhor grau de generalização e, por esse motivo, no presente trabalho consideramos tal configuração como superior em termos de desempenho de classificação. Além disso, quando necessário, aplicam-se os testes estatísticos

não-paramétricos de Kruskall-Willis [24], Friedman [17] e o teste *post-hoc* Wilcoxon *signed rank test* [40] sob correção de Bonferroni [18] para avaliar o nível de significância p da diferença estatística entre os valores das taxas de classificação resultantes das configurações experimentadas. O software R [32] contém a implementação dos testes estatísticos citados e foi utilizado no presente trabalho.

5.3 Comparação entre os métodos de geração de anomalia

Os seguintes conjuntos de dados foram utilizados na avaliação dos métodos de geração de anomalias: Silence, Unvoiced, Ball B., Water P., OC507, OC511, OC514, OC598 e OC620. Bánhalmi et al. [6] utilizaram esses conjuntos para testar o método de geração de anomalias por eles proposto no treinamento de outros tipos de classificadores[1]. Bánhalmi et al. [6] elaboraram os conjuntos Silence, Unvoiced com base no banco de dados *Mel Filter Bank* criado a partir de exemplos da fala humana. Os demais conjuntos foram obtidos de fontes externas [6].

Os conjuntos de dados escolhidos para os experimentos desta Seção e a metodologia utilizada para prepará-los são os mesmos utilizados no trabalho de Bánhalmi et al. [6], pois os resultados apresentados naquele estudo puderam ser devidamente reproduzidos e isto possibilitou maior confiabilidade na avaliação dos métodos de geração de anomalias utilizando a PG proposta no presente trabalho. Outro benefício dessa escolha, foi a possibilidade de comparar diretamente o desempenho de classificação OCC da PG com o desempenho dos outros tipos de algoritmos avaliados por Bánhalmi et al [6].

Os conjuntos Silence, Unvoiced, Ball B. e Water P., conforme Bánhalmi et al. [6], já possuem subconjuntos predefinidos para treino e apresentam-se na Tabela 5.2. Já para os conjuntos de dados OC507, OC511, OC514, OC598 e OC620, utilizou-se validação cruzada 5-*fold* para separar os exemplos utilizados nas etapas de treino e teste, conforme Bánhalmi et al. [6]. Mais detalhes sobre estes conjuntos são apresentados na Tabela 5.3.

Todos os conjuntos são compostos por duas classes. Uma delas foi considerada como

[1] O conjunto de dados OC589 não foi incluso na avaliação porque não foi possível reproduzir os resultados obtidos por Bánhalmi et al. [6] para este conjunto usando classificadores implementados no *software* Weka [21].

Tabela 5.2: Conjuntos de dados utilizados para avaliar os métodos de geração de anomalia

Conjunto	**Silence**		**Unvoiced**		**Ball B.**		**Water P.**	
Atributos	12		12		26		26	
	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste
Normal	301	1.412	405	1.877	56	32	224	96
Anomalia	1.713	1.713	2.282	2.266	88	128	320	320

Tabela 5.3: Conjuntos de dados utilizados para avaliar os métodos de geração de anomalia

Conjunto	**OC507**	**OC511**	**OC514**	**OC598**	**OC620**
Atributos	13	5	278	21	10
Normal	163	126	236	300	4.490
Anomalia	163	126	236	300	4.490
Anomalia Teste	160	125	235	300	2.800

normal e usada para gerar os exemplos anormais. A outra foi utilizada no conjunto de dados de teste como anomalia real. O número de anomalias geradas para o treino equivale ao número de exemplos normais existentes no conjunto de dados. Além disto, todos os conjuntos foram preprocessados usando um passo de normalização seguido por *Principal Component Analysis* (PCA), conforme Bánhalmi et al. [6].

5.3.1 Algoritmo de geração de anomalias usando exemplos distantes

A técnica proposta por Bánhalmi et al. [6] tem dois parâmetros importantes: a distância das anomalias (*Dist*) e a curvatura (*Curv*). Como discutido na Seção 2.3.2 e apresentado na Figura 2.2, *Dist* define a distância a que as anomalias estarão dos exemplos normais e *Curv* define como as anomalias irão contornar os exemplos normais. Deste modo, para avaliar o desempenho da PG proposta, experimentos foram realizados com a seguinte variação de parâmetros: $(Dist = 1,0,\ Curv = 0,5)$, $(Dist = 1,0,\ Curv = 1,0)$, $(Dist = 3,0,\ Curv = 0,5)$ e $(Dist = 3,0,\ Curv = 3.0)$ para os conjuntos de dados comentados na Seção 5.3 e os resultados apresentados na Tabela 5.4. Nesses experimentos utilizou-se a configuração de parâmetros padrão para a PG definida na Seção 5.2, Tabela 5.1.

A Tabela 5.5 apresenta a disparidade entre as taxas de acerto das classes ($|TPr - TNr|$)

Tabela 5.4: Resultados da aplicação da PG para variação dos parâmetros *Dist* e *Curv* (média ± desvio padrão)

	$Dist = 1,0,\ Curv = 0,5$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$83,8 \pm 0,8$	$87,3 \pm 1,5$	$80,4 \pm 2,1$	$88,7 \pm 2,8$
Unvoiced	$73,2 \pm 1,6$	$85,5 \pm 2,0$	$61,0 \pm 3,7$	$80,2 \pm 4,2$
Ball B.	$62,4 \pm 12,4$	$91,6 \pm 4,8$	$55,1 \pm 15,5$	$68,2 \pm 13,1$
Water P.	$86,7 \pm 8,7$	$85,6 \pm 4,8$	$87,0 \pm 12,0$	$88,2 \pm 10,0$
OC 507	$62,4 \pm 5,6$	$83,9 \pm 7,3$	$40,9 \pm 9,8$	$65,2 \pm 7,8$
OC 511	$83,7 \pm 5,0$	$90,0 \pm 6,5$	$77,3 \pm 8,5$	$85,0 \pm 5,3$
OC 514	$88,5 \pm 3,4$	$98,3 \pm 2,4$	$78,8 \pm 6,7$	$80,1 \pm 5,3$
OC 598	$66,7 \pm 6,6$	$83,0 \pm 6,2$	$50,4 \pm 15,3$	$68,8 \pm 8,8$
OC 620	$85,4 \pm 2,1$	$96,0 \pm 1,1$	$74,8 \pm 4,7$	$88,4 \pm 6,0$
	$Dist = 1,0,\ Curv = 1,0$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$72,6 \pm 5,8$	$83,7 \pm 6,7$	$61,5 \pm 17,7$	$75,2 \pm 7,8$
Unvoiced	$60,3 \pm 1,9$	$91,9 \pm 1,9$	$28,6 \pm 4,9$	$60,4 \pm 7,4$
Ball B.	$58,8 \pm 10,0$	$91,8 \pm 4,9$	$50,6 \pm 12,6$	$68,7 \pm 12,8$
Water P.	$61,5 \pm 11$	$86,8 \pm 6,1$	$53,9 \pm 15,3$	$72,3 \pm 10,1$
OC 507	$54,6 \pm 5,3$	$85,1 \pm 8,2$	$24,2 \pm 10,9$	$57,3 \pm 9,7$
OC 511	$67,8 \pm 9,1$	$79,3 \pm 8,2$	$56,3 \pm 19,1$	$68,3 \pm 12,6$
OC 514	$54,3 \pm 3,9$	$47,1 \pm 8,3$	$61,4 \pm 5,7$	$50,4 \pm 5,8$
OC 598	$60,6 \pm 5,2$	$89,7 \pm 5,0$	$31,5 \pm 11,5$	$61,5 \pm 9,7$
OC 620	$88,7 \pm 1,9$	$91,8 \pm 1,5$	$85,5 \pm 3,8$	$89,5 \pm 3,4$
	$Dist = 3,0,\ Curv = 0,5$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$80,5 \pm 1,2$	$93,7 \pm 1,4$	$67,2 \pm 3,2$	$89,9 \pm 3,3$
Unvoiced	$67,0 \pm 1,6$	$94,7 \pm 1,0$	$39,4 \pm 3,9$	$81,7 \pm 3,1$
Ball B.	$74,5 \pm 17,5$	$90,8 \pm 5,1$	$70,4 \pm 21,7$	$79,5 \pm 14,4$
Water P.	$89,5 \pm 5,6$	$89,5 \pm 3,6$	$89,5 \pm 7,2$	$92,5 \pm 5,6$
OC 507	$64,0 \pm 5,1$	$88,5 \pm 6,2$	$39,5 \pm 10,1$	$67,7 \pm 6,8$
OC 511	$85,2 \pm 5,2$	$94,8 \pm 3,4$	$75,6 \pm 9,7$	$86,6 \pm 5,7$
OC 514	$93,4 \pm 3,7$	$99,8 \pm 1,3$	$87,0 \pm 7,1$	$87,1 \pm 7,0$
OC 598	$74,2 \pm 4,7$	$91,2 \pm 4,9$	$57,3 \pm 9,0$	$79,1 \pm 5,7$
OC 620	$80,1 \pm 3,0$	$97,4 \pm 0,8$	$62,7 \pm 6,4$	$87,9 \pm 6,5$
	$Dist = 3,0,\ Curv = 1,0$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$80,2 \pm 2,3$	$79,8 \pm 2,1$	$80,5 \pm 5,0$	$84,0 \pm 3,5$
Unvoiced	$63,2 \pm 4,0$	$87,9 \pm 6,1$	$38,4 \pm 13,6$	$64,0 \pm 6,7$
Ball B.	$55,6 \pm 9,1$	$91,4 \pm 4,8$	$46,7 \pm 11,3$	$65,1 \pm 12,0$
Water P.	$74,4 \pm 13,7$	$85,6 \pm 6,5$	$71,0 \pm 19,3$	$79,7 \pm 10,6$
OC 507	$60,3 \pm 6,6$	$83,1 \pm 8,3$	$37,4 \pm 13,0$	$62,5 \pm 9,6$
OC 511	$73,9 \pm 8,1$	$80,8 \pm 6,9$	$67,1 \pm 15,5$	$75,2 \pm 10,9$
OC 514	$58,0 \pm 3,9$	$54,5 \pm 6,6$	$61,5 \pm 4,3$	$54,1 \pm 5,1$
OC 598	$62,4 \pm 4,9$	$87,1 \pm 5,5$	$37,8 \pm 11,0$	$62,9 \pm 8,9$
OC 620	$88,2 \pm 2,0$	$92,7 \pm 1,6$	$83,7 \pm 4,0$	$89,1 \pm 3,4$

para as diferentes configurações de $Curv$ e $Dist$. Essa tabela foi usada para auxiliar a análise em casos em que Acc e AUC de diferentes configurações são semelhantes, porém em uma delas há maior disparidade entre as taxas de classificação individuais.

	Dist=1,0 Curv=0,5	*Dist=1,0 Curv=1,0*	*Dist=3,0 Curv=0,5*	*Dist=3,0 Curv=1,0*
Silence	6,9	22,2	26,5	0,7
Unvoiced	24,5	63,3	55,3	49,5
Ball B.	36,5	41,2	20,4	44,7
Water P.	1,4	32,9	0,0	14,6
OC 507	43,0	60,9	49,0	45,7
OC 511	12,7	23,0	19,2	13,7
OC 514	19,5	14,3	12,8	7,0
OC 598	32,6	58,2	33,9	49,3
OC 620	21,2	6,3	34,7	9,0

Tabela 5.5: Disparidade entre as taxas TPr e TNr

Conforme resultados apresentados na Tabela 5.4, para o conjunto de dados Silence, a configuração padrão ($Dist = 1,0,\ Curv = 0,5$) apresentou melhor desempenho para a taxa geral Acc e a segunda menor disparidade conforme Tabela 5.5. No conjunto Silence observa-se também que, ocorreu diferença estatística entre pelo menos um par de configurações para as taxas Acc e AUC com significância de 99% ($p = 7,96.10^{-19}, p = 3,03.10^{-17}$), considerando a aplicação do teste estatístico não paramétrico de Kruskal-Wallis. Apesar do desempenho para a taxa AUC ter sido semelhante entre as configurações ($Dist = 1,0,\ Curv = 0,5$) e ($Dist = 3,0,\ Curv = 0,5$), verifica-se que, ocorreu diferença estatística para a medida Acc na fase *post-hoc* do teste estatístico entre as configurações citadas com significância de 99% ($p = 1,09.10^{-5}$), considerando a aplicação do teste não-paramétrico *Wilcoxon signed rank test* sob a correção de Bonferroni. A configuração padrão também apresentou a maior taxa Acc e a menor disparidade para o conjunto Unvoiced e neste aplicam-se as mesmas observações feitas para o conjunto Silence em termos de diferença estatística.

Para os conjuntos de dados Water P., OC514 e OC598, o ajuste ($Dist = 3,0,\ Curv = 0,5$) apresentou melhor desempenho de classificação observando as medidas gerais de acerto Acc e AUC. As diferenças estatísticas para esses conjuntos em Acc e AUC fo-

ram respectivamente com significância de 99% entre pelo menos um par de configurações conforme teste de Kruskal-Wallis ($p = 9,88.10^{-13}, p = 7,60.10^{-12}$), ($p = 2,21.10^{-87}, p = 4,68.1^{-64}$) e ($p = 8,54.10^{-59}, p = 3,12.10^{-52}$). Observa-se também que, para o conjunto Water P. não ocorreu diferença estatística significativa ($p = 1.0$) para a taxa *Acc* entre as configurações ($Dist = 1,0,\ Curv = 0,5$) e ($Dist = 3,0,\ Curv = 0,5$), entretanto, ocorreu zero disparidade com altas taxas de classificação variando de bom para ótimo no ajuste ($Dist = 3,0,\ Curv = 0,5$).

Nos conjuntos Ball B., OC507 e OC511, o ajuste ($Dist = 3,0,\ Curv = 0,5$) teve resultados ligeiramente melhores em termos de acerto geral *Acc* e AUC, sendo que para Ball B. houve também a menor disparidade. As diferenças estatísticas para esses conjuntos em *Acc* e AUC foram respectivamente com significância de 99%, conforme teste de Kruskal-Wallis: ($p = 2,08.10^{-06}, p = 0,0002$), ($p = 5,69.10^{-25}, p = 8,49.10^{-23}$) e ($p = 1.42.10^{-67}, p = 7,02.10^{-54}$).

No conjunto OC620 ocorreu valores semelhantes para as medidas *Acc* e AUC utilizando as configurações ($Dist = 1,0,\ Curv = 1,0$) e ($Dist = 3,0,\ Curv = 1,0$). Também verificou-se que ocorreu diferença estatística entre pelo menos um par de ajustes para a medida *Acc* com 99% ($p = 2,28.10^{-83}$) de significância, conforme teste de Kruskal-Wallis. O mesmo não se repetiu para AUC, pois ($p = 0.65$), entretanto, considera-se o ajuste ($Dist = 1,0,\ Curv = 1,0$) melhor por apresentar maiores valores para as taxas *Acc* e AUC e a menor disparidade conforme Tabela 5.5.

De modo geral, verificou-se que ocorreu diferença estatística entre pelo menos um par de ajustes considerando todos os conjuntos de dados para a taxa *Acc* com significância de 99% ($p = 0,0026$), conforme a aplicação do teste não-paramétrico de Friedman. Apesar disto, observou-se na fase *post-hoc* do teste estatístico que, não houve diferença estatística ($p = 1.0$) entre as configurações ($Dist = 1,0,\ Curv = 0,5$) e ($Dist = 3,0,\ Curv = 0,5$), considerando a aplicação do teste não-paramétrico *Wilcoxon signed rank test* sob a correção de Bonferroni. Já para a taxa AUC, também ocorreu diferença entre todas as configurações com significância de 99% ($p = 0,0029$) e, além disto, na fase *post-hoc* ocorreu diferença estatística entre as configurações citadas com significância de 95,3%

($p = 0,047$). Portanto, sem desconsiderar a configuração padrão, observa-se que ao utilizar ($Dist = 3,0$, $Curv = 0,5$) geralmente ocorreu melhor desempenho geral de classificação. Deste modo revelando que, há melhor condição de aprendizado para PG quando anomalias estão mais distantes dos limites da distribuição do conjunto normal. Todavia, os valores da taxa TPr são, na maioria dos casos, maiores que os da taxa TNr, significando que é mais fácil para o classificador PG classificar corretamente exemplos normais do que anormais.

A Tabela 5.6 foi extraída do trabalho de Bánhalmi et al. [6] e uma coluna com os resultados da PG foi adicionada com resultados que proveem da configuração ($Dist = 1,0$, $Curv = 0,5$) respeitando as configurações usadas naquele trabalho. Deste modo, é possível comparar diretamente resultados obtidos por outros classificadores naquele trabalho com os resultados obtidos pela PG neste. Os algoritmos *One-class* SVM e GMM fazem uso somente de exemplos normais na fase de treino [6]. O ν-SVM e a PG proposta foram treinados na forma de classificador binário. Bánhalmi et al. adotaram a taxa de acerto geral (*Accuracy*) em sua análise como critério comparativo e, para comparar os resultados por eles apresentados com os da PG proposta neste trabalho, o mesmo critério foi adotado para os resultados apresentados na Tabela 5.6.

Tabela 5.6: "Melhores resultados usando diferentes modelos em tarefas OCC: taxas gerais de classificação, taxas de acerto em normais e de erro em anormais" [6].

Accuracy/ TP/FP(%)	exemplos/ atributos	*One-class* SVM	GMM	ν-SVM	PG
Silence	301/12	82,4/89,7/24,8	**85,8**/82,5/36,8	85,6/83,8/12,4	83,8/87,3/19,6
Unvoiced	405/12	78,6/86,3/29,0	78,1/82,9/26,6	**82,9**/80,0/14,4	73,2/85,5/39,0
Ball B.	56/26	99,3/96,9/0,0	96,3/81,3/0,0	**100,0**/100,0/0,0	62,4/91,6/44,9
Water P.	224/26	94,0/88,5/4,4	94,9/87,5/4,4	**95,7**/94,8/4,1	86,7/85,6/13,0
OC 507	163/13	67,2/78,8/44,4	60,3/26,9/6,3	**67,5**/77,5/42,5	62,4/83,9/59,1
OC 511	126/5	85,2/88,8/18,4	78,8/80,0/22,4	**86,8**/92,8/19,2	83,7/90,0/22,7
OC 514	236/278	70,9/82,6/40,9	67,7/53,2/17,9	71,9/ 68,3/34,4	**88,5**/98,3/21,2
OC 598	300/21	**77,8**/76,0/20,3	50,5/1,0/0,0	77,7/86,3/31,0	66,7/83,0/49,6
OC 620	4490/10	**86,3**/89,6/17,0	84,9/93,1/23,2	84,6/91,1/22,1	85,4/96,0/25,2

A Tabela 5.6 mostra que a PG obteve valor superior de *Accuracy* e FP (taxa de erro em anormais) inferior em relação aos valores obtidos por *One-class* SVM para o conjunto de dados Silence. Para o problema Unvoiced, a PG obteve desempenho inferior

aos demais em relação as taxas *Accuracy* e FP. O mesmo ocorreu para o conjunto Ball B., porém observa-se que os outros algoritmos obtiveram 0% de erro para identificar anormais. Neste caso, a PG foi exclusivamente inferior, um indício de que possa requerer para alguns problemas maiores quantidades de exemplos no conjunto de treino, uma vez que Ball B. contém o menor número de exemplos em relação ao número de atributos existentes no conjunto de dados. Para o problema Water P., o desempenho da PG foi inferior aos demais em relação a todas as taxas, porém alcançou $86,7\%$ na taxa geral de acerto e apresentou $13,0\%$ de erro para acerto em anormais, demonstrando ter desenvolvido aprendizado em termos de OCC. Isto não ocorreu para conjunto OC507 em que a PG obteve $59,1\%$ de erro ao identificar anormais. Para o problema OC511, a PG obteve desempenho similar a GMM nas taxas de *Accuracy* e FP e para OC598 desempenho inferior comparado a *One-class* SVM e ν-SVM. No conjunto de dados OC514 houve destaque da PG para a taxa de acerto geral e uma das taxas mais baixas de erro ao identificar anormais. Para o conjunto OC598, a PG não desenvolveu aprendizado para OCC, pois apresentou taxa de erro de $49,6\%$. Em relação ao conjunto OC620, a PG obteve desempenho similar de classificação aos algoritmos GMM e ν-SVM em relação a todas as taxas.

Considerando os melhores resultados da Tabela 5.4, observa-se que a PG proposta treinada com anomalias geradas pelo algoritmo proposto por Bánhalmi et al. obteve desempenho superior a $70,0\%$ em OCC para as todas taxas (Acc, AUC, TPr e TNr) nos conjuntos de dados Silence, Ball B., Water P., OC511, OC514 e OC620, e que ao usar a configuração ($Dist = 3,0,\ Curv = 0,5$) há melhora no resultado para alguns conjuntos de dados. A PG também apresentou o melhor resultado para o conjuntos de dados OC514 nesse ajuste, e para OC620 usando a configuração ($Dist = 1,0,\ Curv = 1,0$). Portanto, verificou-se que com o uso da geração de anomalias estudada, a PG mostra-se competitiva em OCC inclusive se comparada a outros algoritmos como ν-SVM, *One-class* SVM e GMM.

5.3.2 Algoritmo de geração de anomalias usando hiperesfera

O algoritmo, como descrito na Seção 2.3.4, calcula no primeiro passo uma hiperesfera que envolve os exemplos da classe normal no espaço de atributos e posteriormente gera exemplos distribuídos uniformemente dentro dela. A hiperesfera é composta por um vetor centro e outro valor escalar que corresponde ao seu raio, e que é resultante da composição dos vetores suporte calculados através do SVDD na primeira etapa do algoritmo.

Para evitar o problema de mistura de exemplos da classe normal e anomalias, foi proposta na Seção 2.3.4 a geração de exemplos com base em uma hiperesfera aumentada e remoção dos exemplos que estão dentro da hiperesfera original. Utilizando essa abordagem, dois conjuntos de anomalias foram geradas usando diferentes configurações dos parâmetros multiplicadores dos raios das hiperesferas de geração e remoção de anormais: $(dR = 1,1,\ dF = 1,0)$ e $(dR = 1,6,\ dF = 1,5)$. Deste modo, deseja-se verificar o desempenho da classificação obtido pela PG proposta quando as anomalias são geradas próximas aos exemplos da classe normal e também quando estão posicionadas de modo mais afastado.

Nesses experimentos foram utilizados os mesmos conjuntos de dados e metodologia empregados na Seção 5.3.1. Os resultados dos experimentos para os dois conjuntos de anomalias são apresentados na Tabela 5.7. O algoritmo proposto por Tax e Duin [38] foi implementado no *toolbox DD_tools* para o *software* Matlab, e disponível em [37]. O procedimento adicional de remoção de anomalia proposto no presente trabalho e descrito na Seção 4.2.1 foi também implementado em Matlab. Nesses experimentos utilizou-se a configuração de parâmetros padrão para a PG definida na Seção 5.2, Tabela 5.1.

Considerando os resultados obtidos apresentados na Tabela 5.7, para os conjunto de dados Silence, Unvoiced, OC507 e OC598 a configuração $(dR = 1,1,\ dF = 1,0)$ apresentou melhores taxas gerais de classificação (Acc e AUC). Verifica-se também que para os valores destas medidas ocorreu diferença estatística com significância maior que 95%, pois para os conjuntos citados: $(p = 8,12.10^{-11}, p = 0,0007)$, $(p = 4,83.10^{-10}, p = 0,048)$,$(p = 2,21.10^{-19}, p = 8,58.10^{-5})$ e $(p = 1,39.10^{-20}, p = 3,4.10^{-7})$, considerando o teste estatístico não-paramétrico *Wilcoxon signed rank test*. Para o conjunto OC511 o mesmo não

Tabela 5.7: Resultados da aplicação da PG para variação dos parâmetros dR e dF (média $\pm$ desvio padrão)

	$dR = 1,1,\ dF = 1,0$			
Taxas (%)	*Acc*	TPr	TNr	*AUC*
Silence	$72,5 \pm 2,5$	$95,5 \pm 1,0$	$53,5 \pm 5,3$	$89,3 \pm 1,5$
Unvoiced	$57,3 \pm 2,1$	$96,9 \pm 0,8$	$24,5 \pm 4,4$	$79,5 \pm 6,3$
Ball B.	$70,1 \pm 25,2$	$97,1 \pm 7,1$	$68,5 \pm 26,9$	$83,6 \pm 18,1$
Water P.	$54,0 \pm 11,4$	$99,8 \pm 0,6$	$40,3 \pm 14,9$	$67,4 \pm 24,3$
OC 507	$69,6 \pm 5,5$	$89,8 \pm 5,8$	$49,4 \pm 10,4$	$76,3 \pm 6,7$
OC 511	$83,7 \pm 4,8$	$93,0 \pm 4,0$	$74,4 \pm 9,0$	$85,5 \pm 5,6$
OC 514	$59,7 \pm 4,7$	$100 \pm 0,0$	$19,5 \pm 9,4$	$24,0 \pm 15,1$
OC 598	$69,9 \pm 4,7$	$98,2 \pm 1,8$	$41,6 \pm 9,4$	$79,4 \pm 5,7$
OC 620	$56,9 \pm 1,5$	$99,7 \pm 0,2$	$14,1 \pm 3,0$	$76,7 \pm 19,7$
	$dR = 1,6,\ dF = 1,5$			
Taxas (%)	*Acc*	TPr	TNr	*AUC*
Silence	$63,8 \pm 2,8$	$97,7 \pm 1,0$	$35,8 \pm 5,7$	$83,9 \pm 11,3$
Unvoiced	$51,5 \pm 2,2$	$98,8 \pm 0,5$	$12,4 \pm 4,3$	$75,8 \pm 11,2$
Ball B.	$77,4 \pm 17,4$	$99,6 \pm 2,3$	$76,0 \pm 18,5$	$88,8 \pm 16,1$
Water P.	$52,5 \pm 12,3$	$99,9 \pm 0,3$	$38,3 \pm 16,0$	$74,4 \pm 26,1$
OC 507	$63,2 \pm 5,5$	$95,0 \pm 4,6$	$31,5 \pm 12,9$	$73,0 \pm 7,6$
OC 511	$81,1 \pm 1,7$	$96,3 \pm 1,2$	$66,0 \pm 3,3$	$85,6 \pm 2,4$
OC 514	$60,1 \pm 4,7$	$100,0 \pm 0,0$	$20,1 \pm 9,4$	$24,2 \pm 14,8$
OC 598	$62,8 \pm 5,8$	$99,1 \pm 1,1$	$26,5 \pm 11,6$	$75,5 \pm 7,0$
OC 620	$54,9 \pm 1,2$	$99,9 \pm 0,2$	$10,0 \pm 2,5$	$79,3 \pm 19,8$

ocorre para a taxa AUC, pois ($p = 0.92$), entretanto, a configuração ($dR = 1,1,\ dF = 1,0$) apresenta menor disparidade para as taxas de acertos individuais das classes (TPr e TNr).

Para o conjunto de dados Water P. o valor AUC foi maior no ajuste ($dR = 1,6,\ dF = 1,5$) havendo ocorrência de diferença estatística com significância de 96,1% ($p = 0,039$) entre os valores da medida. Por outro lado, o mesmo não ocorreu entre as medidas de acerto individuais das classes, pois ($p = 0,64, p = 0.68$) para TPr e TNr. Para o conjunto OC514 a diferença estatística não foi significante em ambas as configurações do método de geração de anomalias, pois ($p = 0.48, p = 1.0, p = 0.47, p = 0.78$) respectivamente para *Acc*, TPr, TNr e AUC.

Para o conjunto OC620 também não houve diferença estatística significante entre as configurações para a medida AUC, pois ($p = 0.44$), no entanto, o maior valor de *Acc* foi obtido no ajuste ($dR = 1,1,\ dF = 1,0$) havendo ocorrência de diferença estatística entre os valores *Acc* com significância de 99% ($p = 3,88.10^{-25}$) e, do mesmo modo para as taxas individuais, pois ($p = 6,26.10^{-8}, p = 1,39.10^{-25}$) para TPr e TNr. Já para o problema

Ball B., há melhora em todas as medidas para a configuração ($dR = 1,6,\ dF = 1,5$), contudo neste problema, os resultados apresentam-se com baixa diferença estatística nas taxas gerais Acc e AUC, pois ($p = 0.53, p = 0.2$).

Portanto, verifica-se de modo geral que, afastar as anomalias da classe normal usando ($dR = 1,6,\ dF = 1,5$) como configuração de parâmetros para o algoritmo baseado em hiperesfera não auxiliou a PG a obter melhor desempenho de classificação. Também relata-se que ocorreu diferença estatística com significância menor que 95% ($p = 0,095$) para a taxa de desempenho geral Acc e o mesmo não ocorreu para a taxa AUC ($p = 0,73$) em ambas as configurações considerando todos os conjuntos avaliados, conforme o teste estatístico não-paramétrico de Friedman. Todavia, destaca-se também que o método de geração de anomalias avaliado possibilitou valores da taxa de acerto em anormais TNr maiores que 70% para os problemas Ball B. e OC511.

5.3.3 Algoritmo de geração de anomalias inspirada em sistemas imunológicos

O algoritmo RNS, que baseia-se em sistemas imunológicos, como descrito na Seção 2.3.5, gera inicialmente exemplos que simulam células detectoras de modo que as componentes do vetor posição estejam uniformemente distribuídas de acordo com os valores dos componentes dos vetores que representam as células normais. Posteriormente, o algoritmo consiste em, um processo de otimização, reposicionar os detectores de modo a maximizar a ocupação do espaço entre as células normais. Para OCC os detectores caracterizam as anomalias e as células normais os exemplos da classe de perfil normal e os valores dos atributos são os componentes do vetor posição. Entre os parâmeros do algoritmo RNS, há o raio R de detecção do detector usado para reposicioná-lo durante a execução do algoritmo e o número K usado para determinar o número de vizinhos normais usados no cálculo do reposicionamento do detector (Seção 2.3.5). Como ambos determinam a posição final dos detectores (anomalias) afetando a qualidade dos exemplos anormais, foram realizados experimentos com as seguintes varições nesses parâmetros: ($R = 0,1,\ K = 1$), ($R = 0,1,\ K = 10$), ($R = 0,5,\ K = 1$) e ($R = 0,5,\ K = 10$). Os demais parâmetros

do RNS como η_0 (taxa inicial de adaptação), τ (controle de decaimento) e número de rodadas foram mantidos nos mesmos valores usados nos experimentos de Gonzales et al. [20], assim: $\eta_0 = 1,0,\ \tau = 1,0$ e número de rodadas 400. A idade t máxima do detector foi ajustada para 10. Os resultados da execução da PG, usando os exemplos gerados por RNS mediante as variações comentadas, são apresentados na Tabela 5.8. Nesses experimentos utilizou-se a configuração de parâmetros-padrão para a PG definida na Seção 5.2, Tabela 5.1.

Considerando os resultados obtidos, apresentados na Tabela 5.8, para o conjunto de dados Silence houve desempenho de classificação semelhante para todas as variações de parâmetros em todas as taxas de classificação. O ajuste ($R = 0,1,\ K = 10$) possibilitou a menor disparidade conforme Tabela 5.9, seguido da configuração ($R = 0,1,\ K = 1$), no entanto o primeiro apresentou o melhor valor AUC entre elas. Verifica-se que houve diferença estatística entre pelo menos um par de configurações nas taxas gerais *Acc* e AUC com significância de 99%, pois ($p = 6,10.10^{-13},\ p = 0,0001$), considerando teste estatístico não-paramétrico de Kruskal-Wallis. Por apresentar menor disparidade e maior valor AUC, o ajuste ($R = 0,1,\ K = 10$) pode ser considerado ligeiramente superior em relação ao ajuste ($R = 0,1,\ K = 1$), pois observa-se que ocorreu diferença estatística entre as configurações com significância de 95,8% ($p = 0,042$) para a taxa AUC, 99% para TPr ($p = 0,001$) e 97% para TNr ($p = 0,03$), considerando o teste estatístico *post-hoc Wilcoxon signed rank test* sob a correção Bonferroni.

Para o problema Unvoiced, o ajuste ($R = 0,1,\ K = 1$) apresentou melhor valor de *Acc*, AUC e TNr e, a menor disparidade conforme Tabela 5.9, seguido da configuração ($R = 0,1,\ K = 10$) que apresentou valores similares em todos esses quesitos. Verifica-se que houve diferença estatística para a taxa *Acc* com significância de 99% ($7,17.10^{-13}$), 99% para TPr ($p = 3,45.10^{-14}$) e também 99% para TNr ($p = 3,52.10^{-14}$) entre pelo menos um par de ajustes, considerando o teste de Kruskal-Wallis. O mesmo não ocorre para AUC, pois ($p = 0.68$). Apesar do ajuste ($R = 0,1,\ K = 1$) ter apresentado melhores valores na maioria dos fatores avaliados, constata-se em análise estatística *post-hoc* que não há diferença estatística significativa quando comparado à configuração ($R = 0,1,\ K =$

Tabela 5.8: Resultados da aplicação da PG para variação dos parâmetros R e K do RNS (média ± desvio padrão)

	$R = 0,1,\ K = 1$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$75,1 \pm 2,2$	$94,0 \pm 1,1$	$59,4 \pm 4,6$	$85,8 \pm 7,7$
Unvoiced	$63,7 \pm 2,1$	$94,7 \pm 0,9$	$37,9 \pm 4,4$	$81,9 \pm 3,1$
Ball B.	$64,4 \pm 21,3$	$70,4 \pm 18,7$	$64,0 \pm 22,9$	$66,6 \pm 18,1$
Water P.	$86,0 \pm 5,9$	$93,5 \pm 5,9$	$83,8 \pm 7,6$	$92,8 \pm 5,2$
OC 507	$64,6 \pm 6,5$	$80,0 \pm 8,0$	$49,2 \pm 11,6$	$69,2 \pm 7,9$
OC 511	$67,5 \pm 6,6$	$79,4 \pm 8,0$	$55,7 \pm 13,8$	$67,6 \pm 10,1$
OC 514	$72,8 \pm 6,9$	$94,1 \pm 3,4$	$51,4 \pm 13,7$	$78,7 \pm 7,7$
OC 598	$71,4 \pm 5,0$	$90,5 \pm 4,3$	$52,4 \pm 9,4$	$76,8 \pm 5,9$
OC 620	$61,2 \pm 5,9$	$99,5 \pm 0,4$	$23,0 \pm 12,1$	$79,9 \pm 18,7$

	$R = 0,1,\ K = 10$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$75,0 \pm 1,8$	$93,9 \pm 1,2$	$59,5 \pm 3,7$	$86,7 \pm 6,2$
Unvoiced	$62,3 \pm 2,1$	$94,8 \pm 0,9$	$35,4 \pm 4,3$	$80,7 \pm 8,2$
Ball B.	$59,3 \pm 19,5$	$70,8 \pm 12,9$	$58,5 \pm 20,6$	$61,5 \pm 15,7$
Water P.	$88,0 \pm 5,8$	$90,1 \pm 4,6$	$87,4 \pm 7,4$	$92,7 \pm 6,7$
OC 507	$64,8 \pm 6,6$	$79,3 \pm 6,3$	$50,4 \pm 12,2$	$69,9 \pm 7,8$
OC 511	$72,4 \pm 7,6$	$80,2 \pm 8,1$	$64,6 \pm 13,4$	$74,4 \pm 10,1$
OC 514	$75,0 \pm 7,2$	$94,4 \pm 3,5$	$55,7 \pm 14,3$	$80,3 \pm 7,9$
OC 598	$72,4 \pm 3,9$	$91,0 \pm 3,6$	$53,8 \pm 7,6$	$77,8 \pm 5,7$
OC 620	$67,2 \pm 1,5$	$98,9 \pm 0,5$	$35,4 \pm 3,1$	$84,3 \pm 13,8$

	$R = 0,5,\ K = 1$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$72,2 \pm 1,7$	$95,6 \pm 0,8$	$52,9 \pm 3,5$	$87,2 \pm 6,6$
Unvoiced	$59,6 \pm 1,6$	$96,5 \pm 0,7$	$29,1 \pm 3,2$	$81,8 \pm 5,2$
Ball B.	$69,4 \pm 17,8$	$72,5 \pm 12,9$	$69,2 \pm 19,0$	$70,6 \pm 15,7$
Water P.	$85,9 \pm 5,4$	$91,6 \pm 3,3$	$84,2 \pm 6,9$	$91,8 \pm 7,0$
OC 507	$68,8 \pm 5,9$	$82,5 \pm 5,5$	$55,1 \pm 11,8$	$73,6 \pm 7,1$
OC 511	$88,3 \pm 5,1$	$91,3 \pm 4,1$	$75,2 \pm 9,0$	$84,9 \pm 5,6$
OC 514	$76,3 \pm 7,3$	$93,8 \pm 3,5$	$58,8 \pm 14,4$	$81,2 \pm 7,9$
OC 598	$71,5 \pm 4,4$	$92,5 \pm 3,4$	$50,6 \pm 7,8$	$78,0 \pm 5,4$
OC 620	$61,9 \pm 1,7$	$99,5 \pm 0,4$	$24,3 \pm 3,5$	$80,5 \pm 13,2$

	$R = 0,5,\ K = 10$			
Taxas (%)	*Acc*	*TPr*	*TNr*	*AUC*
Silence	$70,8 \pm 2,0$	$96,2 \pm 0,8$	$49,9 \pm 4,1$	$89,9 \pm 0,9$
Unvoiced	$59,5 \pm 1,6$	$96,7 \pm 0,6$	$28,6 \pm 3,2$	$81,3 \pm 5,6$
Ball B.	$67,5 \pm 18,7$	$66,7 \pm 12,0$	$67,5 \pm 19,8$	$69,5 \pm 17,8$
Water P.	$86,7 \pm 5,8$	$91,4 \pm 3,0$	$85,3 \pm 7,6$	$91,9 \pm 5,6$
OC 507	$67,3 \pm 6,0$	$79,4 \pm 7,0$	$55,2 \pm 11,3$	$71,3 \pm 7,5$
OC 511	$82,4 \pm 5,2$	$92,7 \pm 3,9$	$72,1 \pm 9,5$	$84,5 \pm 5,6$
OC 514	$73,4 \pm 8,4$	$84,1 \pm 4,1$	$52,8 \pm 16,5$	$78,8 \pm 9,2$
OC 598	$66,1 \pm 6,0$	$90,7 \pm 3,4$	$41,5 \pm 11,3$	$71,5 \pm 7,7$
OC 620	$64,5 \pm 1,5$	$99,3 \pm 0,3$	$29,7 \pm 3,0$	$71,7 \pm 21,3$

10), pois entre os referidos ajustes ocorrem os valores significância ($p = 1.0,\ p = 0.4,\ p = 1.0,\ p = 1.0$), para as respectivas taxas *Acc*, AUC, TPr e TNr.

Já para o conjunto Ball B., o ajuste ($R = 0,5,\ K = 1$) apresentou os melhores valores para todas as taxas e a segunda menor disparidade conforme Tabela 5.9, seguido da configuração ($R = 0,5,\ K = 10$) que apresentou a menor disparidade e valores similares aos demais quesitos. Entretanto, não desconsidera-se as demais configurações, pois ocorreu diferença estatística com significância apenas de 71% ($p = 0,29$) para *Acc*, de 86% ($p = 0,14$) para AUC, de 65% para TPr ($p = 0,35$) e de 73% ($p = 0,27$) para TNr, ou seja, todos valores significância ($p > 0.05$) entre os pares de ajustes, considerando o teste estatístico de Kruskal-Wallis.

Para o problema Water P., o ajuste ($R = 0,1,\ K = 10$) apresentou os maiores valores para as taxas *Acc* e TNr e, também a menor disparidade conforme Tabela 5.9. Entretanto, verifica-se que ocorreu diferença estatística com significância de apenas 53% para AUC, de 79% ($p = 0,21$) para *Acc* e de 91% ($p = 0,09$) para TNr, ou seja, valores significância ($p > 0.05$) entre os pares de ajustes, considerando o teste estatístico de Kruskal-Wallis. Por isso, não desconsidera-se as demais configurações para o conjunto Water P.

Para o conjunto OC507, o ajuste ($R = 0,5,\ K = 1$) apresentou os melhores valores para todas as taxas e a segunda menor disparidade conforme Tabela 5.9, seguido da configuração ($R = 0,5,\ K = 10$) que apresentou a menor disparidade e valores similares aos demais quesitos. Verifica-se também que, ocorreu diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 2,46.10^{-9},\ p = 0,0001,\ p = 1,58.10^{-6},\ p = 5,39.10^{-6}$), respectivamente para as taxas *Acc*, TPr, TNr e AUC. Entretanto, em análise *post-hoc* verifica-se que não ocorre diferença estatística entre os ajustes ($R = 0,5,\ K = 1$) e ($R = 0,5,\ K = 10$), pois ($p = 1.0$) entre eles para todas as medidas.

No conjunto OC511, a configuração ($R = 0,5,\ K = 1$) apresentou os maiores valores para as taxas *Acc*, TNr e AUC seguido do ajuste ($R = 0,5,\ K = 10$) cujos valores das medidas apresentaram-se similares e esta configuração apresentou a menor disparidade conforme Tabela 5.9. Verifica-se também que, ocorreu diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 8,96.10^{-70},\ p = 1,3.10^{-65},\ p =$

$3,98.10^{-35},\ p = 2,67.10^{-58}$), respectivamente para as taxas *Acc*, TPr, TNr e AUC. Entretanto, em análise *post-hoc* verifica-se que não ocorre diferença estatística entre os ajustes ($R = 0,5,\ K = 1$) e ($R = 0,5,\ K = 10$), pois ($p = 1.0$) entre eles para todas as medidas.

Para o problema OC514, ($R = 0,5,\ K = 1$) apresentou os maiores valores para as taxas *Acc*, TNr e AUC e a segunda menor disparidade conforme Tabela 5.9. Nesses quesitos a configuração ($R = 0,1,\ K = 10$) apresentou valores similares. Verifica-se também que, ocorreu diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 0,0003,\ p = 0,0001,\ p = 0,03$), respectivamente para as taxas *Acc*, TNr e AUC. O mesmo não ocorreu para TPr, pois ($p = 0,57$). Todavia, entre as configurações que se destacaram, não ocorreu diferença estatística em análise *post-hoc*, pois ($p = 1.0$) entre elas para todas as medidas.

Para o conjunto OC598, os ajustes ($R = 0,1,\ K = 1$), ($R = 0,1,\ K = 10$) e ($R = 0,5,\ K = 1$) possibilitaram valores similares em termos de *Acc*, TNr, AUC, entre elas ($R = 0,1,\ K = 10$) possibilitou, conforme Tabela 5.9, a menor disparidade e o maior valor para a taxa TNr. Verifica-se também que, ocorreu diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 8,12.10^{-21},\ p = 1,0.10^{-5},\ p = 6,48.10^{-23},\ p = 9,4.10^{-17}$), respectivamente para as taxas *Acc*, TPr, TNr e AUC. Todavia, entre as configurações que se destacaram, não ocorreu diferença estatística em análise *post-hoc*, pois ($p = 1.0$) entre elas para todas as medidas.

Por fim, no conjunto OC620, o ajuste ($R = 0,1,\ K = 10$) apresentou os maiores valores para as taxas *Acc*, TNr e AUC e a menor disparidade conforme Tabela 5.9. Verifica-se também que, ocorreu diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 1,0.10^{-5},\ p = 2,68.10^{-5},\ p = 3,96.10^{-8}$), respectivamente para as taxas *Acc*, TPr e TNr. O mesmo não ocorreu para AUC, pois ($p = 0,47$). Entretanto, confirma-se o destaque para ($R = 0,1,\ K = 10$) para *Acc*, pois verifica-se que ocorreu diferença estatística em análise *post-hoc* cujos valores significância valem ($p = 0,03,\ p = 7,28.10^{-8},\ p = 4,47^{-5}$) em comparação as demais configurações e, o mesmo ocorreu para a taxa TNr cujos valores significância valem ($p = 0,02,\ p = 5,8.10^{-8},\ p = 1,79.10^{-5}$)

em relação aos demais ajustes.

Tabela 5.9: Disparidade entre as taxas TPr e TNr

	R=0,1 K=1	*R=0,1 K=10*	*R=0,5 K=1*	*R=0,5 K=10*
Silence	34,6	34,4	42,7	46,3
Unvoiced	56,8	59,4	67,4	68,1
Ball B.	6,4	12,3	3,3	0,8
Water P.	9,7	2,7	7,4	6,1
OC 507	30,8	28,9	27,4	24,2
OC 511	23,7	15,6	16,1	20,6
OC 514	42,7	38,7	35	31,3
OC 598	38,1	37,2	41,9	49,2
OC 620	76,5	63,5	75,2	69,6

Com relação ao parâmetro K, verifica-se que há 9 incidências de $K = 10$ entre as configurações que apresentaram os maiores valores para as taxas de acerto geral para os 9 conjunto de dados. Isto demonstra a relevância do parâmetro e também porque possibilita que mais exemplos normais sejam considerados no reposicionamento dos detectores (anomalias) durante o processo do RNS. Apesar disso, constata-se também, que o aumento de K pode afetar o tempo computacional do algoritmo ao considerar mais exemplos normais.

Com relação ao raio de detecção R, verifica-se que para alguns conjuntos $R = 0,5$ apresenta exclusivamente os maiores valores para as taxas de acerto geral, isto deve-se ao fato de que, ao usar maiores valores de R, a tendência é que os detectores afastem-se mais ao longo do processo do RNS, nestes casos resultando em anomalias que melhor caracterizam a fronteira entre exemplos normais e anormais no espaço de atributos.

Todavia, de modo geral, não foi possível estabelecer o melhor ajuste de parâmetros para o RNS, pois distintas configurações apresentam desempenho de classificação destacado em diferentes conjuntos de dados. Este fato comprovou-se, pois não ocorreu diferença estatística entre os pares de ajustes para a taxa geral Acc cuja significância foi apenas de 67% ($p = 0,33$) e de 86% ($p = 0,14$) para a medida geral AUC em todos os conjuntos de dados avaliados, considerando a aplicação do teste estatístico não-paramétrico de Friedman.

5.3.4 Definição do melhor método de geração de anomalias e sua configuração

O objetivo da análise realizada nesta Seção é determinar um método de geração de anomalias e sua configuração, tendo em vista os experimentos realizados nas Seções 5.3.1, 5.3.2 e 5.3.3, para serem aplicados juntamente com a abordagem de PG proposta na Seção 4.3 nos demais experimentos do presente Capítulo. Essa escolha consiste em estabelecer o melhor método e sua configuração para cada um dos conjuntos de dados experimentados, então, aquele algoritmo e respectivo ajuste que forem mais vezes definidos como os melhores entre todos os conjuntos de dados são os escolhidos finais.

Considera-se como critério para determinar o melhor método e configuração por conjunto de dados, os maiores valores de desempenho geral de classificação para ambas as taxas Acc e AUC apresentados nas Tabelas 5.4, 5.7 e 5.8. Caso haja concorrência - uma configuração resulta em maior valor para uma das taxas e menor para a outra - aplica-se o um segundo critério para determinar o melhor que é apresentar a menor disparidade entre as taxas TPr e TNr. Em caso de persistência de empate, os métodos e respetivos ajustes em igualdade segundo esses critérios são concorrentemente considerados como os melhores. Deste modo, apresenta-se na Tabela 5.10 o melhor algoritmo de geração de anomalias e respectivo ajuste para cada um dos conjuntos de dados. Observa-se também que, ocorreu diferença estatística entre pelo menos um par de configurações considerando os resultados dos experimentos dos métodos de geração de anomalia avaliados nas Seções 5.3.1, 5.3.2 e 5.3.3 para a taxa Acc e também para a medida AUC com significância de 99%, pois ($p = 0,003$, $p = 0,014$), conforme o teste estatístico não-paramétrico de Friedman.

Observa-se portanto que, para os conjuntos de dados Silence, Unvoiced, Water P., OC514, OC598 e OC620 o algoritmo de geração de anomalias que se baseia na fronteira de exemplos normais desenvolvido por Bánhalmi et al. [6], apresentou exclusivamente os melhores resultados na aplicação da PG, por isto, foi o algoritmo escolhido para os demais experimentos desenvolvidos nas Seções 5.4, 5.5, 5.6 e 5.7. Entre os ajustes da técnica de

Tabela 5.10: Melhores algoritmos de geração de anomalias e respectivas configurações

	Bánhalmi	Tax	Gonzales
Silence	*Curv=1,0* *Dist=0,5*		
Unvoiced	*Curv=1,0* *Dist=0,5*		
Ball B.		*dR=1,6* *dF=1,5*	
Water P.	*Curv=3,0* *Dist=0,5*		
OC507		*dR=1,1* *dF=1,0*	
OC511			*R=0,5* *K=1*
OC514	*Curv=3,0* *Dist=0,5*		
OC598	*Curv=3,0* *Dist=0,5*		
OC620	*Curv=1,0* *Dist=1,0*		

geração de anomalias escolhida, a configuração ($Dist = 3,0,\ Curv = 0,5$) foi, conforme Tabela 5.10, a que predominantemente proporcionou a PG obter os melhores desempenhos de classificação e, por isto, foi escolhida para ser aplicada nos demais experimentos.

Todavia, apesar da predominância de um dos algoritmos verifica-se que o método baseado em hiperesfera obteve o melhor resultado para os conjunto Ball B. e OC507, mostrando que sua aplicabilidade pode para alguns casos oferecer melhores conjuntos de dados de treino. O mesmo ocorreu para o método baseado em sistemas imunológicos RNS para o conjunto OC511.

5.4 Comparação entre a PG proposta e os algoritmos OCGP e OCGPC

Curry propôs o algoritmo OCGP que utiliza princípios de PG para resolver o problema OCC [10], e posteriormente propôs uma modificação no OCGP referenciada como OCGPC [11]. O OCGP está descrito na Seção 3.3, Algoritmo 4 do Capítulo 3. Além da comparação entre OCGP e OCGPC, outros algoritmos também foram avaliados por Curry: a rede

neural BNN e o ν-SVM, ambos treinados sob o mesmo contexto de classificação binária usando uma classe normal e outra composta por anomalias artificiais. O algoritmo *One-class* SVM, que produz um classificador sem a necessidade de exemplos anormais no conjunto de treino, também foi avaliado.

Os conjuntos de dados empregados na avaliação de Curry [11] foram obtidos no banco de dados da UCI [3], são: Adult, Census, Letter Recognition e Mushroom. O conjunto Letter foi avaliado separadamente usando a classe composta por exemplos da letra "A" (Letter-a), letra "E" (Letter-e) e letras vogais (Letter vowell). Adult e Census possuem subconjuntos de treino e teste predefinidos e para os demais utilizou-se 75% dos normais no treino e 25% no teste. A outra classe de exemplos, usada para testar a classificação de anormais, foi incluída no conjunto de teste respeitando essa proporção. Para os conjuntos Adult e Census foram geradas 50.000 anomalias e para os demais conjuntos 10.000, usando o método baseado em hiperesfera (Capítulo 2, Seção 2.3.4).

Os conjuntos de dados empregados nesta Seção experimental são os mesmos utilizados no trabalho de Curry [11], a fim de comparar diretamente os resultados das abordagens de PG OCGP e OCGPC e, dos demais algoritmos apresentados naquele trabalho com os resultados da PG (*TreeOCGP*) proposta neste. Entretanto, utilizou-se o método de geração de anomalias de Bánhalmi et al. [6], pois apresentou melhores resultados conforme Seção 5.3.4, e foi ajustado para a melhor configuração de parâmetros experimentada ($Dist = 3,0,\ Curv = 0,5$).

O número de anomalias gerado para cada conjunto de dados é igual ao número de exemplos existentes na classe normal nos subconjuntos de treino e teste. Além disto, foi aplicada uma operação de normalização seguida da transformação PCA para preprocessar os conjuntos de dados conforme Seção 5.3.1. Mais detalhes referentes à preparação dos subconjuntos de treino e teste para os experimentos discutidos nesta Seção são apresentados na Tabela 5.11.

A *TreeOCGP* proposta utiliza árvores de expressões aritméticas (Funções Discriminates) como indivíduos da população cuja variáveis são numéricas. Entretanto, os conjuntos Adult, Census e Mushroom possuem atributos categóricos e no conjunto Census apenas

Tabela 5.11: Detalhes dos conjuntos de dados utilizados para avaliar a *TreeOCGP*

Conjunto	**Adult**		**Census**		**Letter-vowell**	
Atributos	14		40		16	
Classes	Treino	Teste	Treino	Teste	Treino	Teste
Anomalia	11.206	11.355	8.155	34.947	3.629	4.031
Normal	7.506	3.700	5.472	2.683	2.660	969
Total	18.712	15.055	13.627	37.630	6.289	5.000
Conjunto	**Letter-a**		**Letter-e**		**Mushroom**	
Atributos	16		16		22	
Classes	Treino	Teste	Treino	Teste	Treino	Teste
Anomalia	771	4.803	737	4.808	2.156	872
Normal	574	197	545	192	1.617	539
Total	1.345	5.000	1.282	5.000	3.773	1.411

3 dos 40 atributos são numéricos. Uma alternativa seria utilizar a notação equidistante para converter os atributos categóricos para numéricos [25]. No entanto, isto resultaria em um número excessivo de atributos, principalmente para o conjunto Census. Por isto, foi aplicada a conversão simples em que cada valor categórico é substituído por um número inteiro. A sequência $\{1, 2, 3, ...\}$, por exemplo, para o caso de 3 ou mais valores categóricos. Apesar deste não ser o melhor tratamento para o problema, apresenta resultados compatíveis com aqueles obtidos quando aplicada a notação equidistante [25]. Observa-se que Curry não especificou o tratamento dado para o problema de conversão em seus experimentos [11].

Para comparar diretamente os resultados da *TreeOCGP* com os resultados apresentados no trabalho de Curry [11], faz-se necessário a elaboração de uma curva ROC para representar os resultados do desempenho de classificação da *TreeOCGP* em cada um dos conjunto de dados. Uma possível maneira de obter os pontos dessa curva é calcular os valores (TPr, FPr) para cada valor de limiar entre os valores *PGout* resultantes da aplicação do melhor classificador sob o conjunto de dados de teste em questão. Lembrando que, conforme Seção 3.2.1, o valor de limiar determina se o exemplo será classificado como normal ou anormal. Como a *TreeOCGP* foi executada 30 vezes para cada conjunto de dados, verifica-se ainda a necessidade de produzir uma curva ROC média dentre as 30 possíveis curvas. A computação desta curva é explicada em detalhes por Fawccet [16].

Todavia, no trabalho de Curry a curva ROC é computada de modo diferente, pois

sua abordagem não utiliza valores de limiar de classificação para identificar exemplos. Os algoritmos de PG propostos por Curry, contudo, resultam em múltiplos classificadores e, então obtém-se os valores (TPr, FPr) para cada um dos classificadores pertencentes aos conjuntos soluções das 50 execuções do algoritmo e para a curva ROC final mantém-se apenas os pares (TPr, FPr) não dominados, conforme conceito de Pareto, de modo a maximizar TPr minimizando FPr [11].

A fim de aproximar a metodologia construtiva de Curry para elaborar uma curva ROC semelhante, todos os pares (TPr, FPr) resultantes de cada uma das curvas ROC das 30 execuções da *TreeOCGP* foram usados para compor um conjunto solução intermediário. Posteriormente, aplicou-se também o conceito de não dominância de Pareto para remover os pares dominados (TPr, FPr) do conjunto solução intermediário. O conjunto resultante dessa operação contém os pares (TPr, FPr) da curva ROC final resultante da *TreeOCGP*. Assim, as curvas ROC resultantes da aplicação da *TreeOCGP* para cada problema OCC são apresentadas nas Figuras 5.1 e 5.2. As demais curvas apresentadas nestas figuras foram extraídas do trabalho de Curry [11]. Para esses experimentos utilizou-se a configuração de parâmetros da PG para *TreeOCGP* definida na Seção 5.2, Tabela 5.1.

Primeiramente, verifica-se de maneira geral conforme Figuras 5.1 e 5.2, que as curvas ROC computadas para *TreeOCGP* em todos os conjuntos de dados apresentam formato similar às curvas dos demais algoritmos. Isso indica que a metodologia empregada para gerar as curvas resultantes da *TreeOCGP* é compatível com as geradas por Curry [11] e, portanto, viabilizando a comparação entre os resultados.

Portanto verifica-se que para o conjunto de dados Adult, a *TreeOCGP* obteve desempenho semelhante a rede neural BNN e superior a ν-SVM e *One-class* SVM e pouco inferior a OCGP, porém inferior a OCGPC. Já para o conjunto Census, a *TreeOCGP* apresentou desempenho similar ao obtido por OCGP e, ligeiramente superior a OCGPC e superior aos demais. Para o problema Letter-vowell, a *TreeOCGP* obteve desempenho similar a OCGP, ligeiramente superior OCGPC e BBN e superior aos demais. Para os conjuntos de dados Letter-a e Letter-e a *TreeOCGP* obteve resultado similar a BNN, porém superior e destacado em relação aos demais.

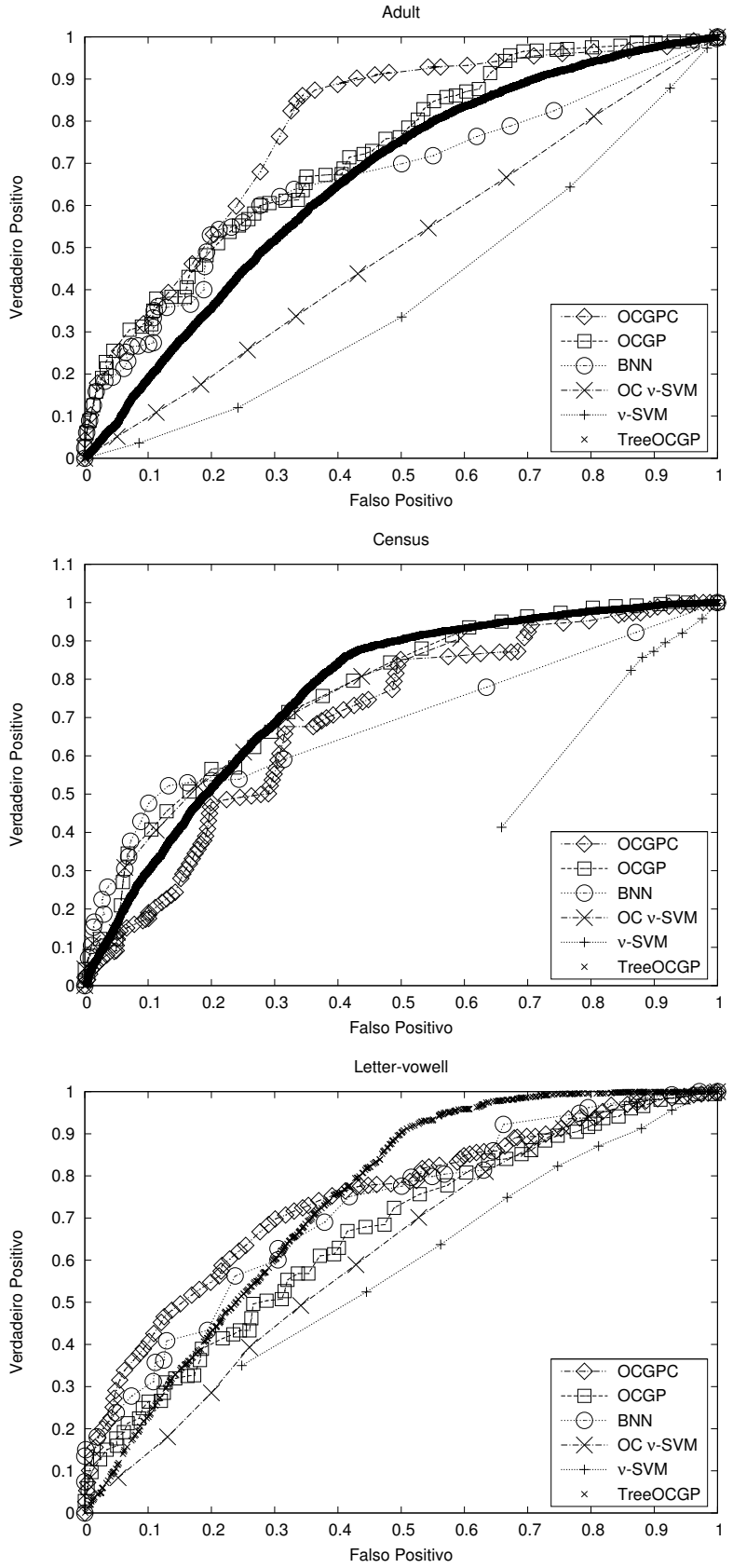


Figura 5.1: Resultados para os conjuntos Adult, Census e Letter-Vowell. TreeOCGP está representada pela linha mais escura que possui alto número de pontos na curva.

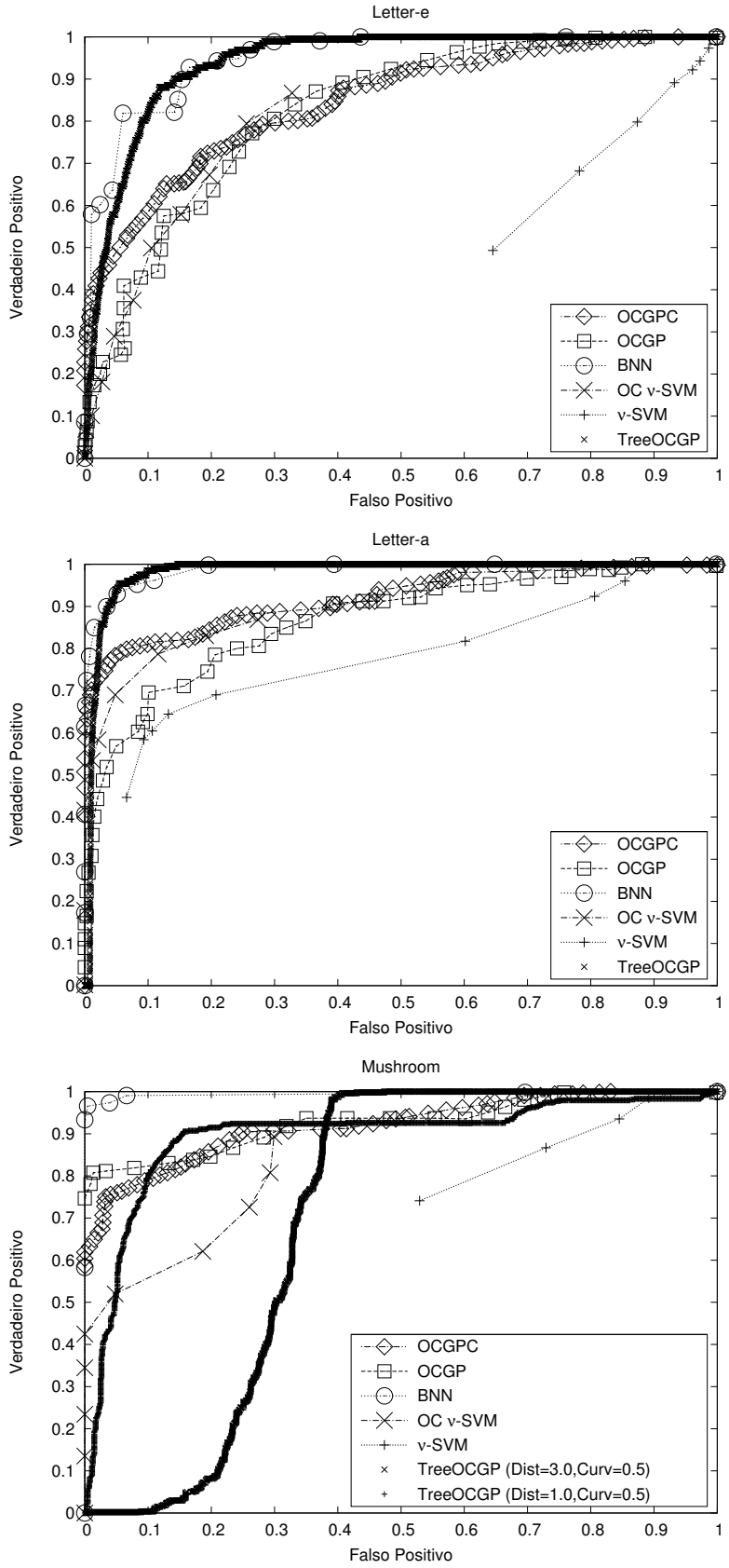


Figura 5.2: Resultados para os conjuntos Letter-e, Letter-a e Mushroom. TreeOCGP está representada pela linha mais escura que possui alto número de pontos na curva.

Em específico, para o conjunto Mushroom, houve desempenho inferior utilizando a configuração de parâmetros para geração de anomalias ($Dist = 3,0,\ Curv = 0,5$). Então, outro experimento foi realizado utilizando a segunda melhor configuração ($Dist = 1,0,\ Curv = 0,5$), conforme Seção 5.3.1. Sob este outro ajuste para gerar anomalias, a *TreeOCGP* obteve melhor desempenho de classificação, pois a curva resultante apresenta maior área, e também passou a apresentar melhorias em relação aos algoritmos ν-SVM e OC ν-SVM e com desempenho pouco inferior a OCGPC. Através desses experimentos com o conjunto Mushroom foi possível constatar a influência da geração de anomalias e sua parametrização nos resultados obtidos pelo classificador *TreeOCGP*.

Em suma, considerando os resultados obtidos, verifica-se que a *TreeOCGP* obteve melhor resultado em alguns problemas e resultado similar em outros comparada às outras abordagens de PG, e em alguns casos comportando-se de maneira equiparável à rede neural BNN. Não foi possível realizar comparação em termos de tempo computacional entre os algoritmos, pois essas informações não foram relatadas por Curry em ambos os trabalhos [10] e [11]. Entretanto, o tempo e o CPU utilizados pela *TreeOCGP* para realizar o treinamento em cada conjunto de dados estão registrados na Tabela 5.12.

Tabela 5.12: Recursos utilizados e tempo de execução da TreeOCGP

	Adult	Census	Letter-a	Letter-e	Letter-vowell	Mushroom
CPU (Clock)	AMD Opt. 2,4 Gghz	AMD Opt. 2,4 Gghz	AMD Opt. 2,8 Gghz	AMD Opt. 2,4 Gghz	AMD Opt. 2,4 Gghz	AMD Opt. 2,4 Gghz
Tempo (Horas:Minutos)	165:46	63:06	5:43	5:47	5:51	15:20

Relata-se também que os algoritmos propostos por Curry computam amostras no conjunto de dados treino, e isto foi desnecessário ao utilizar a *TreeOCGP*. Além disso, o número de exemplos anormais utilizados na etapa de treino foi, na maioria dos casos, na ordem de $1:10$ menor. Também verifica-se que o ajuste ($Dist = 3,0,\ Curv = 0,5$) para gerá-los foi no geral suficiente, deste modo, consolidando o estudo apresentado na Seção 5.3.1.

Outro destaque para a *TreeOCGP* como uma nova abordagem de PG para o problema é que em todos os conjuntos de dados experimentados houve desenvolvimento de regras de

classificação para a distinção de normais e anormais durante o processo evolutivo. Pois, conforme observa-se nas Figuras 5.1 e 5.2, em todos os conjuntos de dados as área das curvas ROC da *TreeOCGP* foram visivelmente superiores a $0,5$, desempenho em que o classificador não desenvolve regras de aprendizado.

5.5 Comparação entre funções de aptidão

Um dos problemas verificados conforme experimentos realizados na Seção 5.3 (Tabelas 5.5 e 5.9) é a dificuldade do classificador PG em obter melhor desempenho na identificação de anomalias, pois na maior parte dos casos estudados há considerável disparidade entre acerto na identificação das classes de exemplos normais e anormais. Uma possível causa pode estar relacionada com a função de aptidão utilizada. A fim de verificar esta hipótese, foram realizados experimentos utilizando a AUC como função de aptidão, pois permite através de sua curva visualizar o desempenho de classificação variando a prioridade de acerto entre as classes. Outra função de aptidão avaliada foi a métrica WMW que representa, de acordo com Bowman [8], um valor estimado da AUC. O cálculo dessa métrica como função de aptidão está apresentado na Seção 3.2.2, Equações 3.10 e 3.11 do Capítulo 3.

Para avaliar o desempenho da PG proposta sob o uso das funções de aptidão AUC e a métrica WMW, os seguintes conjuntos de dados disponíveis na UCI [3] foram utilizados: Glass Identification (Glass), Ionosphere (Ionos), Iris, Sonar e Spectf. A escolha destes conjuntos diferencia-se daqueles empregados nas Seções 5.3 e 5.4, pois nesta Seção experimental não há comparativo direto com resultados de outros trabalhos e, além disto, o objetivo é de ampliar a avaliação da PG proposta para outros problemas OCC.

O método para gerar anomalias empregado é aquele que se baseia nos limites da classe normal usando como parâmetros ($Dist = 3,0,\ Curv = 0,5$), como discutido na Seção 5.3.4. Nestes experimentos utilizou-se a configuração de parâmetros padrão para a PG definida na Seção 5.2, Tabela 5.1. Mais detalhes sobre os conjuntos de dados são apresentados na Tabela 5.13.

Em todos os conjuntos de dados foi utilizada a abordagem de validação cruzada *5-*

Tabela 5.13: Conjuntos de dados

	Glass Id.	**Ionosphere**	**Iris**	**Sonar**	**Spectf**
Atributos	9	34	4	60	44
Classe Normal	*float proc.*, *non-window*	*good*	*versicolor virginica*	*mines*	*abnormal*
Nr. Exemplos Normais	138	225	100	111	212
Nr. Anormais Gerados	138	225	100	111	212
Classe "Anormal"	*non-float* proc.	*bad*	*setosa*	*rocks*	*normal*
Nr. Exemplos "Anormais"	76	126	50	97	55

fold para divisão da quantidade de exemplos usados nas etapas de treino e teste. Um experimento adicional com a função de aptidão composta pela média da taxas TPr e TNr (Equação 4.2) também foi realizado e os resultados obtidos para as diferentes funções de aptidão são apresentados na Tabela 5.14.

Tabela 5.14: Resultados da aplicação da PG para diferentes funções de aptidão (média ± desvio padrão)

Função	AUC				
Taxas (%)	*Acc*	TPr	TNr	AUC	Valor aptidão
Glass	$35,1 \pm 11,1$	$99,0 \pm 2,0$	$12,4 \pm 15,7$	$92,2 \pm 9,6$	$1,00 \pm 0,00$
Ionos	$39,7 \pm 13,0$	$98,3 \pm 3,5$	$18,7 \pm 18,7$	$90,8 \pm 90,8$	$0,98 \pm 0,00$
Iris	$46,5 \pm 23,2$	$99,7 \pm 0,7$	$25,2 \pm 32,8$	$99,7 \pm 0,5$	$1,00 \pm 0,00$
Sonar	$22,4 \pm 6,5$	$98,9 \pm 2,4$	$5,1 \pm 8,4$	$80,8 \pm 25,7$	$1,00 \pm 0,00$
Spectf	$43,3 \pm 0,0$	$100,0 \pm 0,0$	$0,0 \pm 0,0$	$35,6 \pm 19,6$	$1,00 \pm 0,00$
Função	WMW				
Taxas	*Acc*	TPr	TNr	AUC	Valor aptidão
Glass	$37,0 \pm 15,4$	$99,4 \pm 1,1$	$14,8 \pm 21,2$	$93,8 \pm 9,2$	$1,00 \pm 0,00$
Ionos	$42,2 \pm 11,8$	$99,3 \pm 1,1$	$21,8 \pm 16,2$	$92,6 \pm 4,9$	$0,99 \pm 0,00$
Iris	$42,8 \pm 16,1$	$99,9 \pm 0,3$	$19,9 \pm 22,6$	$99,4 \pm 0,9$	$1,00 \pm 0,00$
Sonar	$21,6 \pm 5,3$	$99,8 \pm 0,5$	$3,8 \pm 6,7$	$85,0 \pm 21,1$	$1,00 \pm 0,00$
Spectf	$43,4 \pm 0,2$	$99,9 \pm 0,2$	$0,2 \pm 0,4$	$39,3 \pm 17,5$	$1,00 \pm 0,00$
Função	Média entre TPr e TNr, Equação 4.2				
Taxas	*Acc*	TPr	TNr	AUC	Valor aptidão
Glass	$91,8 \pm 7,9$	$90,9 \pm 9,3$	$92,1 \pm 9,6$	$94,1 \pm 6,4$	$0,97 \pm 0,02$
Ionos	$81,4 \pm 7,2$	$79,3 \pm 6,6$	$82,2 \pm 9,3$	$80,9 \pm 9,4$	$0,81 \pm 0,04$
Iris	$97,4 \pm 2,1$	$95,7 \pm 4,7$	$98,0 \pm 2,2$	$98,6 \pm 2,0$	$0,99 \pm 0,01$
Sonar	$66,7 \pm 13,4$	$80,7 \pm 12,9$	$63,6 \pm 16,9$	$74,2 \pm 11,9$	$0,78 \pm 0,07$
Spectf	$44,3 \pm 3,9$	$85,7 \pm 7,0$	$12,7 \pm 6,1$	$46,3 \pm 9,8$	$0,80 \pm 0,08$

Os valores de aptidão apresentados na 5.14 para cada função de aptidão variam de um valor mínimo $0,0$ ao máximo de $1,0$. Observa-se que para as funções AUC e WMW os valores máximos foram obtidos para todos os conjuntos, porém ocorreram na etapa de teste valores mais baixos em termos de AUC para desempenho de classificação. Além disso, constatou-se baixo desempenho de classificação para identificar anomalias (TNr) em todos os conjuntos de dados na utilização das funções AUC e WMW. Neste caso, observou-se a condição de super generalização, caso em que boa parte dos exemplos é classificada como normal, inclusive os anormais.

Verifica-se também que ocorre diferença estatística com significância de 95% para todas as taxas em todos os conjuntos para pelo menos um par de funções entre as medidas de aptidão avaliadas, pois ($p < 0.05$), considerando o teste estatístico não-paramétrico de Krukal-Wallis. Entretanto, na etapa *post-hoc* não ocorre diferença estatística significante ($p > 0.05$) entre as funções de aptidão AUC e WMW para todas as medidas, conforme o teste *Wilcoxon signed rank test*, por isto, não foi possível estabelecer comparação entre elas. Por outro lado, um indício de que podem apresentar desempenho de classificação similar, uma vez que a WMW é valor estimado da AUC.

Todavia, uma melhor generalização foi obtida no uso da função de aptidão composta pelas médias TPr e TNr, pois foram obtidos valores maiores que 63% para a taxa de acerto para identificar exemplos anormais, a medida TNr, nos conjuntos: Glass, Ionos, Iris e Sonar, e os valores da taxa TPr mantiveram-se maiores que 79% para esses conjuntos. Já para o conjunto Spectf, ocorreu baixo desempenho de classificação para identificar anomalias e isto pode estar relacionado com o grau de dificuldade da PG em identificar os limites da classe de exemplos normais no espaço de atributos. Esta hipótese é melhor verificada na Seção 5.6, pois nela outros algoritmos classificadores são avaliados para o problema Spectf.

Outra observação é que melhores classificadores poderiam ter sido obtidos para os conjuntos de dados Sonar, Spectf e Ionos, cujos valores de aptidão foram respectivamente 0,78, 0,80 e 0,81. Isto pode estar relacionado com a parametrização da PG e esse aspecto será explorado na Seção 5.7. Ademais, observa-se a alta incidência de valores de aptidão

próximos do ótimo (1.0), e isto atesta que a PG funcionou bem enquanto algoritmo de otimização para OCC.

5.6 Comparação entre PG proposta e outros algoritmos de classificação

Um experimento adicional foi realizado afim de comparar a PG proposta (*TreeOCGP*) com outros algoritmos de classificação convencionais. Os conjuntos de dados utilizados nestes experimentos são os mesmos utilizados na Seção 5.5 e a mesma metodologia também foi empregada em sua preparação para as etapas de treino e teste. Os algoritmos comparados com a *TreeOCGP* são o ν-SVM e a Rede Neural Multicamada MLP (*Multilayer Perceptron*) ambos implementados no *software* Weka [21]. O método de geração de anomalias empregado é aquele que se baseia nos exemplos normais mais distantes usando como parâmetros ($Dist = 3,0,\ Curv = 0,5$), como discutido na Seção 5.3.4. Nestes experimentos utilizou-se a configuração de parâmetros padrão para a PG definida na Seção 5.2, Tabela 5.1. Já para os outros algoritmos utilizou-se o ajuste de parâmetros padrão presente em sua implementação no ambiente Weka [21]. O resultado dos experimentos são apresentados na Tabela 5.15.

Conforme Tabela 5.15, verifica-se que a *TreeOCGP* obteve melhor desempenho de classificação geral nas taxas *Acc* e *AUC* para o conjunto Glass e também para identificação de anormais (TNr). Por outro lado, para o conjunto Ionos, o algoritmo ν-SVM obteve o melhor desempenho seguido da rede neural, embora a *TreeOCGP* tenha obtido valores maiores que 80% em termos de *Acc*, *AUC* e TNr. Para o conjuntos Iris, todos os algoritmos apresentaram altas taxas de desempenho de classificação. Isto deve-se ao fato do conjunto possuir classes bem definidas em termos de distribuição dos exemplos no espaço de atributos e também porque há poucos atributos (4) no conjunto de dados. Para o conjunto Sonar, a *TreeOCGP* saiu-se melhor que os demais em todas as taxas, embora a taxa de classificação de anormais tenha sido igual a 63,6%. Por fim, para o conjunto Spectf, a MLP obteve melhor desempenho geral de classificação seguido de ν-SVM, porém

Tabela 5.15: Desempenho de classificação para os algoritmos ν-SVM, MLP e *TreeOCGP* (média ± desvio padrão)

	ν-SVM			
Taxas	*Acc*	TPr	TNr	*AUC*
Glass	$86,8 \pm 8,7$	$88,1 \pm 8,8$	$86,3 \pm 8,7$	$87,2 \pm 8,7$
Ionos	$92,7 \pm 6,6$	$84,4 \pm 9,0$	$95,7 \pm 8,1$	$90,1 \pm 6,3$
Iris	$98,9 \pm 2,8$	$96,0 \pm 3,5$	$100,0 \pm 3,2$	$98,0 \pm 2,7$
Sonar	$35,1 \pm 17,5$	$62,7 \pm 25,4$	$28,9 \pm 21,3$	$45,8 \pm 15,7$
Spectf	$59,0 \pm 20,4$	$90,0 \pm 35,4$	$35,3 \pm 33,8$	$62,6 \pm 20,3$
	Multilayer Perceptron - MLP			
Taxas	*Acc*	TPr	TNr	*AUC*
Glass	$65,4 \pm 23,4$	$91,1 \pm 28,6$	$56,3 \pm 28,0$	$80,2 \pm 23,1$
Ionos	$85,7 \pm 4,1$	$83,6 \pm 5,2$	$86,5 \pm 3,9$	$91,8 \pm 6,3$
Iris	$96,9 \pm 3,7$	$92,0 \pm 5,8$	$98,8 \pm 4,4$	$98,1 \pm 4,0$
Sonar	$46,9 \pm 12,3$	$50,0 \pm 12,2$	$46,2 \pm 12,4$	$51,7 \pm 12,5$
Spectf	$62,3 \pm 18,7$	$88,1 \pm 29,8$	$42,5 \pm 29,8$	$68,3 \pm 18,7$
	TreeOCGP			
Taxas	*Acc*	TPr	TNr	*AUC*
Glass	$91,8 \pm 7,9$	$90,9 \pm 9,3$	$92,1 \pm 9,6$	$94,1 \pm 6,4$
Ionos	$81,4 \pm 7,2$	$79,3 \pm 6,6$	$82,2 \pm 9,3$	$80,9 \pm 9,4$
Iris	$97,4 \pm 2,1$	$95,7 \pm 4,7$	$98,0 \pm 2,2$	$98,6 \pm 2,0$
Sonar	$66,7 \pm 13,4$	$80,7 \pm 12,9$	$63,6 \pm 16,9$	$74,2 \pm 11,9$
Spectf	$44,3 \pm 3,9$	$85,7 \pm 7,0$	$12,7 \pm 6,1$	$46,3 \pm 9,8$

verifica-se que todos os algoritmos obtiveram baixo índice de acerto para identificar anomalias. Acredita-se que isto se relaciona com o fato de o conjunto de dados possuir alta quantidade de atributos (44). O mesmo comentário se aplica ao conjunto Sonar (60).

Em resumo a *TreeOCGP* foi melhor em dois dos cinco conjuntos conforme resultados apresentados na Tabela 5.15, demonstrando ser competível com algoritmos de classificação tradicionais. Para os conjuntos de dados Glass, Ionos e Iris foram obtidas taxas de acerto superiores a 80% para identificar exemplos anormais e para esses conjuntos mantiveram-se equilibradas as disparidades entre TPr e TNr, pois sua diferença foi inferior a 3%.

5.7 Comparação entre variação dos parâmetros da PG proposta

Foram realizados experimentos com o objetivo de verificar o impacto na variação de alguns parâmetros da PG proposta no desempenho de classificação em OCC. Entre os parâmetros escolhidos, estão aqueles que podem potencialmente oferecer restrições severas ao algoritmo: o número de indivíduos da população da PG, o tamanho da árvore que compõe cada classificador da população na forma de função discriminante e a taxa de mutação.

O número de indivíduos na população está relacionado com o tamanho do espaço de busca pelo melhor classificador. Em relação ao tamanho da árvore, verifica-se que quanto maior seu tamanho, maior a quantidade de nós e por sua vez maiores chances de que todos os atributos do conjunto de dados estejam presentes na árvore. Por isso, se este número for insuficiente, atributos importantes do problema de classificação podem ficar fora da composição do classificador levando a baixos índices de acerto. A taxa de mutação é outro fator que auxilia a PG na manutenção da diversidade de classificadores na população, assim evitando que a busca pelo melhor classificador estacione em regiões subótimas do espaço de busca. Altas taxas de mutação, por outro lado, podem deteriorar o material genético transformando o processo de otimização em busca aleatória.

A variação nos parâmetros foi estabelecida a partir da configuração padrão para a PG definida na Seção 5.2, Tabela 5.1. O tamanho da população foi reduzido de 2.000 para 1.000 para verificar se ao utilizar uma população proporcionalmente menor há degradação nos resultados de classificação, considerando que este ajuste possibilita reduzir o tempo computacional utilizado pela PG na etapa de treino. O tamanho máximo da árvore foi alterado de 12 para 24, a fim de verificar a influência no aumento do tamanho da árvore. O taxa de mutação foi aumentada de 15% para 30% a fim de verificar se o valor padrão foi insuficiente para manter a diversidade genética. Apresenta-se na Tabela 5.16 todas variações de parâmetros estabelecidas. O restante da configuração da PG foi mantida como definida na Seção 5.2, Tabela 5.1.

Os conjuntos de dados utilizados são os mesmos utilizados na Seção 5.5 e a mesma metodologia também foi empregada em sua preparação para as etapas de treino e teste. O

Tabela 5.16: Variação de parâmetros da PG

Índice	**População**	**Tamanho da Árvore**	**Taxa de Mutação**	**Taxa de Cruzamento**
A	1.000	12	15	85
B	1.000	24	15	85
C	1.000	12	30	70
D	1.000	24	30	70
E (padrão)	2.000	12	15	85
F	2.000	24	15	85
G	2.000	12	30	70
H	2.000	24	30	70

método de geração de anomalias empregado é aquele que se baseia nos exemplos normais mais distantes usando como parâmetros ($Dist = 3,0,\ Curv = 0,5$), como discutido na Seção 5.3.4. O resultado dos experimentos para cada conjunto de dados utilizando as variações de parâmetro estabelecidas na Tabela 5.16 são apresentados nas Tabelas 5.17, 5.18, 5.19, 5.20 e 5.21.

Tabela 5.17: Variação de parâmetros da PG para o conjunto Glass (média ± desvio padrão)

	Conjunto Glass			
Taxas	*Acc*	TPr	TNr	AUC
A	$92,56 \pm 6,88$	$89,65 \pm 11,75$	$93,60 \pm 8,02$	$93,06 \pm 9,01$
B	$91,77 \pm 6,92$	$89,83 \pm 11,85$	$92,46 \pm 8,45$	$92,99 \pm 7,93$
C	$91,18 \pm 9,48$	$90,44 \pm 10,84$	$91,44 \pm 11,66$	$92,54 \pm 9,59$
D	$90,20 \pm 10,45$	$90,17 \pm 11,37$	$90,21 \pm 13,09$	$92,43 \pm 9,51$
E	$92,38 \pm 7,61$	$90,42 \pm 11,17$	$93,07 \pm 9,21$	$94,29 \pm 6,85$
F	$91,90 \pm 8,65$	$88,37 \pm 12,50$	$93,16 \pm 10,57$	$92,63 \pm 8,94$
G	$92,61 \pm 7,46$	$91,19 \pm 9,60$	$93,11 \pm 9,23$	$94,64 \pm 6,47$
H	$90,58 \pm 9,35$	$88,79 \pm 13,19$	$91,22 \pm 11,66$	$93,15 \pm 8,19$

Para o conjunto Glass ocorreu variação menor que 3% nas taxas gerais *Acc* e AUC entre todas as configurações de PG. Como todas as taxas foram no geral maiores que 90%, verifica-se que não houve maiores dificuldades em relação ao estabelecimento de parâmetros da PG conforme Tabela 5.17. Também observa-se que não houve destaque entre as configurações da PG, pois não ocorreu diferença estatística significante ($p < 0.05$) para as taxas *Acc*, TPr, TNr e AUC, pois ($p = 0,43,\ p = 0.71,\ p = 0.15,\ p = 0.46$), considerando o teste estatístico não-paramétrico de Kruskal-Wallis.

Todavia, as configurações "A","E" e "G" proporcionaram maiores valores para as

taxas gerais *Acc* e *AUC* e, verifica-se que o tamanho de árvore 12 é comum entre elas e isto pode estar relacionado com a quantidade de atributos no conjunto de dados (9), conforme Tabela 5.17. Entre essas configurações, "A" contém apenas 1.000 indivíduos classificadores na população da PG e isto favorece menor tempo computacional na etapa de treino.

Tabela 5.18: Variação de parâmetros da PG para o conjunto Ionos (média ± desvio padrão)

	Conjunto Ionos			
Índice	*Acc*	TPr	TNr	*AUC*
A	$81,73 \pm 6,68$	$80,40 \pm 7,41$	$82,21 \pm 8,68$	$82,16 \pm 9,11$
B	$79,68 \pm 7,84$	$79,75 \pm 7,25$	$79,66 \pm 10,88$	$80,61 \pm 8,94$
C	$81,58 \pm 6,83$	$80,77 \pm 7,12$	$81,87 \pm 9,08$	$83,27 \pm 8,00$
D	$80,33 \pm 7,22$	$79,69 \pm 7,10$	$80,56 \pm 9,65$	$79,75 \pm 8,70$
E	$81,23 \pm 6,49$	$80,96 \pm 6,68$	$81,33 \pm 8,53$	$80,60 \pm 8,64$
F	$82,24 \pm 5,85$	$79,33 \pm 7,09$	$83,28 \pm 7,68$	$80,33 \pm 7,93$
G	$82,81 \pm 5,95$	$81,17 \pm 7,12$	$83,39 \pm 7,76$	$81,72 \pm 8,13$
H	$81,40 \pm 6,81$	$79,14 \pm 8,14$	$82,20 \pm 9,08$	$81,11 \pm 9,45$

Para o conjunto Ionos as observações são semelhantes as feitas para o conjunto Glass, pois há variação menor que 4% nas taxas gerais *Acc* e *AUC* entre todas as configurações de PG. Como todas taxas foram no geral maiores que 80%, verifica-se que não houve maiores dificuldades em relação ao estabelecimento de valores adequados para os parâmetros da PG conforme Tabela 5.18. Também observa-se que ocorreu diferença estatística significante entre pelo menos um par de ajustes ($p < 0.05$) para as taxas *Acc*, TNr e *AUC*, pois ($p = 0,009,\ p = 0.037,\ p = 0.002$) considerando o teste de Kruskal-Wallis. Contudo, a avaliação estatística *post-hoc Wilcoxon signed rank test* sob a correção de Bonferroni revela que a diferença significante ($p < 0.05$) apresenta-se apenas entre as configurações "B" e "G" na taxa *Acc* e, "C" e "F" na taxa *AUC*, cujos os valores de significância são respectivamente ($p = 0,014,\ p = 0,02$).

As configurações "A","C" e "G" proporcionaram maiores valores para as taxas gerais *Acc* e *AUC* e verifica-se que o tamanho de árvore 12 é comum entre elas. Entre essas configurações, "A" contém apenas 1.000 indivíduos classificadores na população da PG e isto favorece menor tempo computacional na etapa de treino.

Para o conjunto Iris as observações também são semelhantes as feitas para o conjunto

Tabela 5.19: Variação de parâmetros da PG para o conjunto Iris (média ± desvio padrão)

	Conjunto Iris			
Índice	*Acc*	TPr	TNr	*AUC*
A	$96,81 \pm 4,43$	$96,23 \pm 4,76$	$97,04 \pm 5,83$	$98,59 \pm 3,19$
B	$96,66 \pm 5,42$	$96,10 \pm 4,86$	$96,88 \pm 7,49$	$98,69 \pm 2,49$
C	$95,84 \pm 6,57$	$96,00 \pm 5,20$	$95,77 \pm 8,75$	$98,06 \pm 5,89$
D	$96,00 \pm 6,22$	$95,37 \pm 5,45$	$96,25 \pm 8,55$	$97,98 \pm 5,48$
E	$96,80 \pm 5,24$	$95,97 \pm 4,64$	$97,13 \pm 7,23$	$98,57 \pm 3,72$
F	$97,12 \pm 2,79$	$95,33 \pm 5,19$	$97,84 \pm 2,94$	$98,69 \pm 2,89$
G	$96,73 \pm 6,20$	$96,13 \pm 4,55$	$96,97 \pm 8,60$	$98,51 \pm 7,31$
H	$95,93 \pm 5,82$	$95,47 \pm 5,33$	$96,12 \pm 7,78$	$98,65 \pm 3,59$

Glass, pois conforme Tabela 5.19 há variação menor que 2% nas taxas gerais *Acc* e *AUC* entre todas as configurações de PG. Como todas as taxas foram no geral maiores que 95%, verifica-se que não houve maiores dificuldades em relação ao estabelecimento de valores adequados para os parâmetros da PG. Além disto, todas as configurações da PG proporcionam taxas de acerto *Acc*, *AUC*, TPr e TNr semelhantes e considerando o seu desvio padrão não foi possível estabelecer qual dos ajustes da PG ofereceu melhores condições de aprendizado. Isto comprova-se também pelo teste estatístico de Kruskal-Wallis, cujos respectivos valores de significância são ($p = 0,81,\ p = 0,67,\ p = 0,46,\ p = 0,68$), comprovando a não ocorrência de diferença estatística ($p < 0.05$).

Contudo, verifica-se que a configuração "A" contém apenas 1.000 indivíduos classificadores na população da PG e também o menor tamanho de árvore (12) e isto favorece menor tempo computacional na etapa de treino.

Tabela 5.20: Variação de parâmetros da PG para o conjunto Sonar (média ± desvio padrão)

	Conjunto Sonar			
Índice	*Acc*	TPr	TNr	*AUC*
A	$67,83 \pm 14,09$	$77,82 \pm 14,66$	$65,57 \pm 17,47$	$74,67 \pm 12,30$
B	$61,74 \pm 14,78$	$81,64 \pm 13,91$	$57,23 \pm 18,86$	$71,87 \pm 12,02$
C	$66,90 \pm 16,17$	$80,24 \pm 13,90$	$63,88 \pm 19,73$	$74,34 \pm 13,77$
D	$64,39 \pm 15,00$	$80,61 \pm 13,23$	$60,71 \pm 19,21$	$73,20 \pm 12,41$
E	$72,57 \pm 12,74$	$78,42 \pm 14,60$	$71,24 \pm 15,87$	$77,28 \pm 11,53$
F	$67,83 \pm 13,25$	$77,12 \pm 15,83$	$65,73 \pm 16,35$	$73,18 \pm 12,01$
G	$72,40 \pm 14,30$	$77,85 \pm 15,61$	$71,16 \pm 17,45$	$77,28 \pm 12,87$
H	$62,56 \pm 15,42$	$79,64 \pm 15,79$	$58,69 \pm 19,46$	$71,00 \pm 13,03$

Para o conjunto Sonar verifica-se que as configurações "E" e "G" proporcionaram

à PG o melhor desempenho de classificação conforme as taxas gerais *Acc* e *AUC*. Isto confirma-se em análise *post-hoc*, pois a configuração "E" apresentou considerável diferença estatística com valor significância ($p = 0,07,\ p = 2,9.10^{-7},\ p = 0,07,\ p = 5,4.10^{-5},\ p = 0,07, p = \ 8,0.10^{-7}$) respectivamente em relação à "A", "B", "C", "D", "F" e "H". Para a configuração "G" também consta-se considerável diferença estatística com valor significância ($p = 0,08,\ p = 2,32.10^{-7},\ p = 0.07,\ p = 1.7.10^{-4},\ p = 0.25,\ p = 6,8.10^{-6}$) em relação as demais configurações na etapa *post-hoc* do teste estatístico não-paramétrico *Wilcoxon signed rank test* sob a correção de Bonferroni. Notavelmente, não ocorre diferença entre "E" e "G", pois entre elas o valor significância vale ($p = 1.0$) para a medida *Acc*. Verifica-se também que no teste estatístico não-paramétrico de Kruskal-Wallis apresentou-se diferença estatística entre pelo menos um par de ajustes com significância de 99% ($p = 2,7.10^{-13},\ p = 6,5.10^{-6}$) para *Acc* e *AUC*.

Ocorre situação semelhante em termos de diferença estatística na taxa *AUC*, pois nela também ocorre significância de 99% ($p = 6,53.10^{-6}$) entre pelo menos um par de ajustes. A configuração "E" possui considerável diferença estatística com valor significância de ($p = 0,01,\ p = 0,09,\ p = 0,02$) em relação às configurações "B", "D" e "G" na etapa *post-hoc* do teste estatístico. A configuração "G" possui também considerável diferença estatística com valor significância de ($p = 0,001,\ p = 0,02,\ p = 0,001$) em relação às configurações "B", "D" e "H" na etapa *post-hoc* do teste estatístico.

As configurações "E" e "G" possuem em comum 2.000 indivíduos na população da PG e tamanho de árvore ajustado para 12. A diferença entre elas está no ajuste da taxa de mutação, porém os resultados são semelhantes em todas as taxas de classificação inclusive o desvio padrão. Por isso, para o conjunto Sonar considera-se que o aumento na taxa de mutação não resultou em melhorias, ao contrário houve redução de desempenho geral de classificação para o experimento "H", mostrando que o ajuste da mutação pode permanecer moderado.

Verificou-se na Tabela 5.20 que houve considerável variação de valores da taxa *Acc* entre as diferentes configurações e que o ajuste no tamanho da população teve maior influência no desempenho geral de classificação. Um indício de que o ajuste do tamanho

da população é um dos mais importantes da PG para OCC. Outra importante constatação dos experimentos realizados com o conjunto Sonar.

Tabela 5.21: Variação de parâmetros da PG para o conjunto Spectf (média ± desvio padrão)

	Conjunto Spectf			
Índice	*Acc*	TPr	TNr	AUC
A	$44,07 \pm 4,08$	$85,76 \pm 6,84$	$12,23 \pm 5,61$	$47,22 \pm 9,53$
B	$43,90 \pm 4,69$	$85,17 \pm 7,21$	$12,38 \pm 7,63$	$46,67 \pm 11,16$
C	$44,98 \pm 3,80$	$87,30 \pm 6,70$	$12,65 \pm 5,65$	$46,51 \pm 12,35$
D	$43,93 \pm 3,95$	$86,41 \pm 7,95$	$11,49 \pm 6,00$	$46,23 \pm 11,43$
E	$44,84 \pm 4,33$	$86,16 \pm 6,47$	$13,28 \pm 6,22$	$47,65 \pm 11,28$
F	$44,69 \pm 4,05$	$85,79 \pm 7,10$	$13,30 \pm 5,81$	$48,30 \pm 10,78$
G	$44,76 \pm 3,99$	$86,98 \pm 6,86$	$12,51 \pm 5,41$	$47,52 \pm 10,56$
H	$44,14 \pm 4,12$	$85,73 \pm 7,30$	$12,39 \pm 6,17$	$45,82 \pm 12,44$

Para o conjunto Spectf as observações são semelhantes às feitas para o conjunto Iris, pois há variação em geral menor que 1% nas taxas *Acc* e AUC entre todas as configurações de PG. Como todas elas proporcionam valores semelhantes e considerando o desvio padrão não foi possível estabelecer as melhores configurações. Isto comprova-se para as taxas TPr, TNr e AUC, cujos respectivos valores de significância são ($p = 0,24,\ p = 0,19,\ p = 0,5$). Deste modo, não ocorre diferença estatística significante ($p < 0.05$) conforme teste de Kruskal-Wallis entre os pares de ajustes da PG nessas medidas. Apesar de verificada a ocorrência de diferença estatística para a taxa AUC, pois ($p = 0.031$), o menor valor de significância ($p = 0.35$) ocorre apenas entre as configurações "C" e "D" no teste estatístico *post-hoc Wilcoxon signed rank test* sob correção de Bonferroni. Observa-se também que a configuração "A" contém apenas 1.000 indivíduos classificadores na população da PG e também o menor tamanho de árvore (12) e isto favorece menor tempo computacional na etapa de treino.

Outra observação para o conjunto Spectf conforme Tabela 5.21 é que ocorrem taxas de acerto menores que 14% para TNr e maiores que 85% para TPr isto caracteriza uma condição de excessiva generalização, pois o classificador tende a aceitar quaisquer exemplos como normais. A dificuldade no estabelecimento de um melhor grau de generalização pode estar mais relacionada com uma possível distribuição irregular dos exemplos normais no espaço de atributos do que a parametrização da PG. Essa condição também é verificada

nos experimentos com outros tipos de classificadores para o conjunto Spectf apresentados na Seção 5.6.

Os experimentos realizados nesta Seção revelaram por meio de experimentos com o conjunto Sonar que o tamanho da população de indivíduos da PG pode exercer maior influência nos resultados de classificação e também que o aumento na taxa de mutação não melhora os resultados. Porém, considera-se de modo geral que não houve configuração da PG cujo resultado em termos de desempenho de classificação fosse destacado em relação às demais configurações. Por isso, em problemas OCC práticos cujo tempo computacional para treinamento do classificador também é crucial, pode-se ajustar os parâmetros dentre os avaliados como a população e o tamanho da árvore para níveis que tornem viável a aplicação da PG proposta no presente trabalho.

CAPÍTULO 6

CONSIDERAÇÕES FINAIS

6.1 Conclusão e principais contribuições

No presente trabalho apresentou-se uma abordagem de PG ainda não avaliada em OCC. O principal trabalho relacionado, estudado na Seção 3.3 e Capítulo 3, introduz uma série de mudanças e novas ideias. Entretanto, como discutido na Seção 4.2 e Capítulo 4, se o método de geração de anomalias adequado for utilizado, uma abordagem mais simples de PG pode também ser aplicada com sucesso ao problema. Por isso, a ênfase do trabalho foi avaliar a PG proposta na presença de conjuntos de dados de treino compostos por anomalias artificiais geradas por diferentes algoritmos.

No Capítulo 5, Seção 5.3, foram apresentados o resultado e a análise comparativa entre o método proposto por Tax e Duin [38] baseado em hiperesfera, o método proposto por Gonzáles et al. [19] e [20] baseado em princípios de sistemas imunológicos e o método proposto por András et al. [6] que se baseia nos limites da classe normal. Todos foram testados com a aplicação da PG proposta em diferentes problemas OCC. O método proposto por András et al. [6] foi, segundo o presente estudo, o melhor avaliado e por isso, aplicado nos demais experimentos. Neste ponto, há uma importante contribuição do presente trabalho para pesquisa em problemas OCC, pois verifica-se a inexistência de estudos comparativos entre métodos de geração de anomalias, uma vez que essa abordagem viabiliza o emprego de algoritmos como a PG, principalmente em casos práticos em que não é possível obter exemplos reais de comportamento anormal.

No Capítulo 5, Seção 5.4, também foi realizado um estudo comparativo entre a PG proposta proposta por Curry [10] e [11] aplicando a mesma metodologia e conjuntos de dados experimentados naquele trabalho. Os resultados atestam que a abordagem de PG proposta apresentou desempenho de classificação similar em alguns problemas e melhor em outros em termos da correta identificação de exemplos normais e anormais

em comparação aos resultados das abordagens de PG propostas por Curry. Neste ponto, verifica-se a hipótese de que a geração de anomalias é crucial e tem influência direta para resolver o problema na forma de classificação binária. Ademais, os resultados atestam o presente estudo como base e ponto de partida para que novas ideias e aparatos do estado da arte da PG possam ser aplicados e seu impacto devidamente avaliado, outra importante contribuição do presente trabalho.

Ainda na parte experimental, Capítulo 5, foram avaliadas funções de aptidão da PG como a AUC e WMW e verificou-se que sua aplicação direta pode não oferecer condições para que a PG desenvolva bons classificadores. Outro experimento adicional foi elaborado para comparar a PG proposta com outros tipos de classificadores bem estabelecidos como ν-SVM e rede neural MLP, e os resultados atestam que a PG apresentou desempenho similar de classificação. Por fim, outro estudo relacionado com a variação de parâmetros da PG considerados mais relevantes para classificação foi realizado e os resultados mostraram que o tamanho da população da PG pode influenciar no desempenho de classificação em alguns casos e, o aumento na taxa de mutação no geral não resulta em melhoria.

6.2 Trabalhos futuros

Existem problemas OCC em que há a ocorrência de exemplos anormais dentro do espaço ocupado pela distribuição dos exemplos da classe normal, ou seja, a existência de *outliers* de fato. Neste caso, não faz sentido remover anomalias artificiais que estão presentes na classe normal. Deste modo, atestando que o método de geração de anomalias proposto por Tax e Duin [38] ainda que não tenha apresentado os melhores os resultados, pode produzir conjuntos de anomalias que oferecem melhores condições de aprendizado do classificador para problemas OCC em que é comum a existência de anomalias dentro do classe normal. Entretanto, é importante ressaltar que aspectos internos da PG devem ser trabalhados, como, por exemplo, o mapeamento de valores *PGout* em classes em que a ocorrência de valores positivos e negativos é comum justificando, portanto, o uso da LFM proposta por Curry [10].

O uso do método de geração de anomalias proposto por Tax e Duin [38], como ob-

servado no Capítulo 2, Seção 2.3.4, resulta em conjuntos de dados desequilibrados. Por isso, é necessário que a função de aptidão da PG esteja apta para que o baixo desempenho na classe minoritária seja igualmente tratada em relação à maioritária. Deste modo, propõe-se que ao utilizar as anomalias geradas pelo método baseado em hiperesfera sejam avaliadas as demais funções de aptidão voltadas a conjuntos de dados desequilibrados apresentadas no Capítulo 3, Seção 3.2.2. Inclui-se nesta avaliação a análise da convergência dos valores aptidão durante a execução da PG que pode auxiliar na otimização dos demais parâmetros como o ajuste do número de gerações e o número de classificadores na população.

Especialmente neste contexto, verifica-se que ao utilizar anomalias geradas pelo método proposto por Tax e Duin [38] faz-se necessário mudanças em aspectos internos da PG, algumas delas propostas por Curry [10]. Por isso, o presente trabalho enfatizou a avaliação de técnicas para gerar anomalias de modo a não alterar características internas do funcionamento da PG enquanto algoritmo para classificação binária.

Outra proposta de trabalho futuro é combinar os métodos de geração de anomalia. Por exemplo, usar o método proposto por Tax e Duin [38] para gerar a hiperesfera de exemplos anormais e posteriormente utilizar o RNS [19] e [20] para otimizar o posicionamento das anomalias dentro da hiperesfera. Ao final, possivelmente haverá anomalias no espaço ocupado por exemplos normais, porém as partes mais irregulares desta classe estarão melhor envolvidas pelas anomalias. Outra estratégia para gerar mais anomalias de melhor qualidade é combinar as classes de exemplos anormais produzidos na execução do método proposto por András [6] usando diferentes ajustes do parâmetro *Dist*.

Um estudo adicional do parâmetro *Dist* também precisa ser realizado, pois verifica-se que seu ajuste pode estar relacionado com a disparidade entre os valores dos atributos do conjunto de dados. Observa-se que para conjuntos nos quais há pouca variação de valores de atributos de um mínimo para máximo, maiores valores de *Dist* podem afastar excessivamente as anomalias da classe normal. Ao contrário, se houver atributos que apresentem alta variação entre valores mínimo e máximo, ajustes reduzidos de *Dist* podem aproximar excessivamente as anomalias da classe normal. Do mesmo modo, o parâmetro

$Curv$ pode ser melhor ajustado se características da distribuição dos valores dos atributos também forem consideradas.

Propõe-se também avaliar a aplicação direta do algoritmo de Bánhalmi apresentado na Seção 2.3.2 para resolver OCC. Uma possível regra para classificar exemplos é: para um exemplo x qualquer calcula-se o exemplo f no limite da classe normal mais próximo de x e os respectivos exemplos n normais mais próximos de f, então verifica-se a separabilidade linear entre x e os normais n utilizando o algoritmo SVM e, caso este processo falhe, considera-se x como exemplo normal e caso contrário uma anomalia.

Em relação à PG, como trabalho futuro, propõe-se a aplicação de uma estratégia de otimização multiobjetivo na abordagem proposta no presente trabalho. Os objetivos a serem otimizados podem ser as taxas de cada classe (TPr e TNr), bem como as taxas gerais de classificação: a área sob a curva ROC e a taxa de acerto *Accuracy*. Dependendo das métricas utilizadas como objetivos a serem otimizados, outra comparação direta com o trabalho proposto por Curry [11] pode ser realizada.

Na PG proposta no presente trabalho a avaliação do desempenho de classificação foi realizada utilizando o indivíduo classificador que obteve o melhor valor aptidão ao longo das gerações. Entretanto, constata-se que uma característica única da PG em relação a outros algoritmos classificadores é a produção de uma população de classificadores. Por que não utilizá-la em um esquema de votação, juntamente com o melhor classificador elaborado para avaliar o desempenho final de classificação?

A distribuição dos exemplos da classe normal pode apresentar-se de diferentes formatos dependendo do problema OCC, como por exemplo, na forma de espiral dupla e anel. Neste caso, propõe-se como outro trabalho futuro verificar se o método de geração de anomalias apresentado na Seção 2.3.2 está apto a gerar anomalias nas fronteiras internas do anel e das espirais, bem como avaliar o desempenho da PG proposta nessas condições.

Com o objetivo de auxiliar a análise dos resultados foram aplicados os testes estatísticos de Kruskal-Wallis, Friedman e Wilcoxon *signed rank test* conforme exposto nas Seções 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.5 e 5.7. Outro teste estatístico amplamente aplicado, porém não utilizado no presente trabalho é o *Analysis of Variance* (ANOVA) [30] [33]. Por isto, sugere-se seu uso em trabalhos futuros para casos em que haja distribuição normal nos resultados avaliados.

Por fim, relata-se que na parte experimental foram usados conjuntos de dados com duas ou mais classes de exemplos que foram modificados para OCC. Devido a isso, propõe-se a avaliação da PG para um problema real de OCC, por exemplo, em detecção de intrusão em redes de computador em que o conjunto de dados *KDD Cup 1999*, disponível em [1], possa ser avaliado.

BIBLIOGRAFIA

[1] KDD Cup 1999 Data. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, Agosto de 2011.

[2] Software Matlab. `http://www.mathworks.com/products/matlab/`, Agosto de 2011.

[3] A. Asuncion e D. J. Newman. Uci repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/mlrepository.html`, Agosto de 2011.

[4] A. Bánhalmi, R. Busa-Fekete, e B. Kégl. A one-class classification approach for protein sequences and structures. *ISBRA '09: Proceedings of the 5th International Symposium on Bioinformatics Research and Applications*, páginas 310–322, 2009.

[5] András Bánhalmi. One-class datasets and resources automatic counter-example generation-based one-class classifier. `http://www.inf.u-szeged.hu/~banhalmi/oneclass/`, Agosto de 2011.

[6] András Bánhalmi, András Kocsor, e Róbert Busa-Fekete. Counter-example generation-based one-class classification. *ECML '07: Proceedings of the 18th European conference on Machine Learning*, páginas 543–550, 2007.

[7] U. Bhowan, M. Zhang, e M. Johnston. Differentiating between individual class performance in genetic programming fitness for classification with unbalanced data. *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, páginas 2802–2809, 2009.

[8] U. Bhowan, M. Zhang, e M. Johnston. Genetic programming for image classification with unbalanced data. *24th International Conference Image and Vision Computing New Zealand*, páginas 316–321, 2009.

[9] C. Cagne e M. Parizeau. OpenBEAGLE: A new versatile C++ framework for evolutionary computations. *Late-Breaking Papers of the 2002 Genetic and Evolutionary Computation Conference (GECCO)*, páginas 161–168, 2002.

[10] Robert Curry e Macolm Heywood. One-class learning with multi-objective genetic programming. *Proceedings of the IEEE Systems, Man and Cybernetics Conference*, páginas 1938–1945, 2007.

[11] Robert Curry e Macolm Heywood. One-class genetic programming. *Lectures Notes in Computer Science*, 5481/2009:1–12, Abril de 2009.

[12] Robert P. W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, e David M. J. Tax. Prtools4: a matlab toolbox for pattern recognition. version 4.1, 2009.

[13] J. Eggermont, A. E. Eiben, e J.I. van Hemert. A comparasion of genetic programming variants for data classification. 1999.

[14] Pedro G. Espejo, Sebastián Ventura, e Francisco Herrera. A survey on the application of genetic programming to classification. *Trans. Sys. Man Cyber Part C*, 40(2):121–144, 2010.

[15] W. Fan, M. Miller, S. Stolfo, W. Lee, e P. Chan. Using artificial anomalies to detect unknown and known network intrusions. *Proceedings of the first IEEE International conference on Data Mining*, páginas 123–130, 2001.

[16] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. Relatório técnico, HP Laboratories, 2004.

[17] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association 47*, 200:675–701, Dezembro de 1937.

[18] Tal Galili. Post hoc analysis for friedmanś test (r code). `http://www.r-bloggers.com/post-hoc-analysis-for-friedman%E2%80%99s-test-r-code/`, Agosto de 2011.

[19] F. A. Gonzáles, D. Dasgupta, e L. F. Nino. Combining negative selection and classification techniques for anomaly detection. *Congress on Evolutionary Computation CEC*, páginas 261–272, 2002.

[20] Fabio A. Gonzáles e Dasgupta Dipankar. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, December de 2003.

[21] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, e Ian H. Witten. The weka data mining software: An update.

[22] Kenneth Hennessy, Michael G. Madden, Jennifer Conroy, e Alan G. Ryder. An improved genetic programming technique for the classification of raman spectra. *Knowledge Based Systems*, 18(4-5):217–224, Agosto de 2005.

[23] John. R. Koza. Genetic programming: on the programming of computers by means of natural selection. *MIT press*, 1992.

[24] Kruskal e Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association 47*, 260:583–621, Dezembro de 1952.

[25] Namgil Lee e Jong-Min. Kim. Conversion of categorical variables into numerical variables via bayesian network classifiers for binary classifications. *Computional Statistcs anda Data Analysis*, 54:1247–1265, 2010.

[26] T. Loveard e V. Cielsielski. Representing classification problems in genetic programming. *In Proceedings of the Congress on Evolutionary Computation*, 2:1070–1077, 2001.

[27] Larry M. Manevitz e Yousef Malik. One-class svms for document classification. *Journal of Machine Learning*, 2:139–154, 2001.

[28] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997.

[29] Daniel T. Munroe e Michel G. Madden. Multi-class and single-class classification approaches to vehicle model recognition from images. *Proceedings of AICS-05: Irish Conference on Artificial Intelligence and Cognitive Science*, 2005.

[30] R. G. O'Brien. A general anova method for robust tests of additive models for variances. *Journal of the American Statistical Association*, 74:877–880, 1979.

[31] Grant Patterson e Mengjie Zhang. Fitness functions in genetic programming for classification with unbalanced data. *Lectures Notes in Artificial Intellingence - Proceedings of 20th Australian joint conference on Advances in artificial intelligence*, páginas 769–775, 2007.

[32] R Development Core Team. R: A language and environment for statistical computing. `http://www.R-project.org`, 2011. ISBN 3-900051-07-0.

[33] H. Scheffé. *The Analysis of Variance*. John Wiley & Sons, 1959.

[34] H. J. Shin, D. Eom, e S. Kim. One-class support vector machines-an application in machine fault detection and classification. *Computers & Industrial Engineering*, 48:395–408, 2005.

[35] D. Song, M. I. Heywood, e A. Nur Zincir-Heywood. A linear genetic programming approach to intrusion detection. *GECCO'03: Proceedings of the 2003 international conference on Genetic and evolutionary computation*, páginas 2325–2336, 2003.

[36] David M. J. Tax. *One-class classification, Concept-learning in absence of counter-examples*. Tese de Doutorado, Technische Universiteit Delf, setembro de 2001.

[37] David M. J. Tax. Ddtools, the data description toolbox for matlab. version 1.7.3, 2009.

[38] David M. J. Tax e Robert P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal for Machine Learning Research*, páginas 155–173, 2001.

[39] C. Wang, Y. Ting, Y. Liu, e G. Hariyanto. A novel approach to generate artificial outliers for support vector data description. *IEEE International Symposium on Industrial Electronics (ISIE 2009)*, páginas 2202–2207, July de 2009.

[40] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin 1*, 6:80–83, Dezembro de 1945.

[41] Mengjie Zhang e Will Smart. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27(11):1266–1274, 2006.

[42] W. Zhongyu, J. Xun, e X. Wang. One-class classification based finance news story recommendation. *Journal of Computacional Information Systems*, 5:1625–1631, 2009.