Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Allan Danilo de Lima

Detecção de Alterações Respiratórias na Sarcoidose Através da Técnica de Oscilações Forçadas e Algoritmos de Aprendizado de Máquinas

Rio de Janeiro

2020

Allan Danilo de Lima

Detecção de Alterações Respiratórias na Sarcoidose Através da Técnica de Oscilações Forçadas e Algoritmos de Aprendizado de Máquinas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes.

Orientador: Prof. Jorge Luís Machado do Amaral, DSc

Coorientador: Prof. Pedro Lopes de Melo, DSc

Rio de Janeiro

2020

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

L732 Lima, Allan Danilo de.
Detecção de alterações respiratórias na sarcoidose através da técnica de oscilações forçadas e algoritmos de aprendizado de máquinas / Allan Danilo de Lima. – 2020.
111f.

Orientador: Jorge Luís Machado do Amaral.
Coorientador: Pedro Lopes de Melo.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Aprendizado do computador - Teses. 3. Algorítmos - Teses. 4. Insuficiência respiratória - Teses. I. Amaral, Jorge Luís Machado do. II. Melo, Pedro Lopes de. III. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. IV. Título.

CDU 004.891

Bibliotecária: Júlia Vieira – CRB7/6022

_____ _____

Assinatura Data

Allan Danilo de Lima

Detecção de Alterações Respiratórias na Sarcoidose Através da Técnica de Oscilações Forçadas e Algoritmos de Aprendizado de Máquinas

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes.

Aprovado em: 10 de novembro de 2020

Banca Examinadora:

Prof. Jorge Luís Machado do Amaral, DSc (Orientador)

Faculdade de Engenharia - UERJ

Prof. Pedro Lopes de Melo, DSc (Coorientador)

Instituto de Biologia - UERJ

Prof. Douglas Mota Dias, DSc

Faculdade de Engenharia - UERJ

Prof. Arthur de Sá Ferreira, DSc

Centro Universitário Augusto Motta - UNISUAM

Rio de Janeiro

2020

AGRADECIMENTOS

Ao meu orientador Professor Jorge Luís Machado do Amaral pelo estímulo, pela confiança e pelo tempo dedicado em me orientar.

Ao meu coorientador Professor Pedro Lopes de Melo pela orientação neste trabalho e, antes disto, por ter me incentivado a seguir na área acadêmica.

Ao Professor José Franco Machado do Amaral, pela influência e incentivo a seguir nesta linha de pesquisa.

Aos Professores que examinaram esta dissertação, Douglas Mota Dias e Arthur de Sá Ferreira, pelas contribuições ao texto final.

Ao Juan, fonte de inspiração e companheiro de todas as horas.

À minha mãe, Izabel, por ter me apoiado em mais esta jornada.

Aos colegas do programa, Elan, George, José Rodrigo e Erito que estiveram comigo nos piores e melhores momentos.

Aos demais professores do PEL que contribuíram para minha formação e que assim se fazem presentes neste resultado.

Aos membros do Laboratório de Instrumentação Biomédica da UERJ e demais envolvidos, pelos exames realizados com a Técnica de Oscilações Forçadas.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

RESUMO

LIMA, Allan Danilo de. *Detecção de Alterações Respiratórias na Sarcoidose Através da Técnica de Oscilações Forçadas e Algoritmos de Aprendizado de Máquinas*. 111 f. Dissertação (Mestrado em Engenharia Eletrônica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2020.

Nesta dissertação são desenvolvidos diversos classificadores baseados em aprendizado de máquina para auxiliar no diagnóstico de alterações respiratórias associadas à sarcoidose, a partir de resultados provenientes da Técnica de Oscilações Forçadas (do inglês *Forced Oscillation Technique*, ou FOT), que é um método não invasivo utilizado para avaliação da mecânica pulmonar. Além de resultados acurados, há especial interesse na interpretabilidade dos mesmos, portanto são utilizadas diversas formas de Programação Genética, devido à classificação ser feita por meio de expressões inteligíveis. É usada a sua forma tradicional em árvores, a orientada à gramática, e também a linear com inspiração quântica (PGLIQ). A fim de verificar se os resultados interpretáveis são competitivos, os mesmos são comparados ao desempenho com K-Vizinhos mais Próximos, *Support Vector Machine*, *AdaBoost*, *Random Forest*, *LightGBM*, *XGBoost* e Regressor Logístico. Para trazer ainda mais interpretabilidade, os experimentos também são realizados com atributos "fuzzificados". Ainda é experimentado um método de seleção de atributos e outro de aumento sintético do conjunto de dados. Quanto à PGLIQ, além de utilizá-la diretamente para a classificação, também é experimentado usá-la para construir atributos a serem utilizados por outros métodos. Os dados utilizados para alimentar os classificadores são provenientes da execução da FOT em 72 indivíduos, sendo 25 saudáveis e 47 diagnosticados com sarcoidose. Nesses últimos, foi verificado pela espirometria que 24 apresentaram condições normais e 23 mostraram alterações respiratórias. Os resultados obtidos apresentam alta acurácia (AUC ≥ 0.90) nas três análises realizadas (controle $\times$ indivíduos com sarcoidose, controle $\times$ indivíduos com sarcoidose e espirometria alterada e controle $\times$ indivíduos com sarcoidose e espirometria normal).

Palavras-chave: Sarcoidose; Técnica de oscilações forçadas; FOT; Aprendizado de máquina; AUC.

ABSTRACT

LIMA, Allan Danilo de. *Detection of respiratory changes in sarcoidosis by forced oscillation technique and machine learning algorithms.* 111 f. Dissertação (Mestrado em Engenharia Eletrônica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2020.

In this work, many machine learning classifiers are developed to assist the diagnosis of respiratory changes associated to sarcoidosis, based on results from the Forced Oscillation Technique (FOT), which is a non-invasive method used to assess pulmonary mechanics. In addition to accurate results, there is a special interest in their interpretability, so several forms of Genetic Programming are used since the classification is made with intelligible expressions. Its traditional tree form, grammar-based form, and also quantum-inspired linear (QILGP) are used. To check if interpretable results are competitive, they are compared to performance with K-Nearest Neighbors, Support Vector Machine, AdaBoost, Random Forest, LightGBM, XGBoost, and Logistic Regressor. To bring even more interpretability, the experiments are also performed with fuzzy features. The use of a feature selection technique and a synthetic oversampling technique is also tested. Besides using QILGP directly for classification, it is also tested to use it to build features to be used by other methods. The data used to feed the classifiers come from the FOT exams in 72 individuals, from which 25 were healthy and 47 were diagnosed with sarcoidosis. Among the latter, it was verified by spirometry that 24 showed normal conditions and 23 showed respiratory changes. The results achieved high accuracy (AUC $\geq$ 0.90) in the three analyzes performed (controls vs. individuals with sarcoidosis, controls vs. individuals with sarcoidosis and altered spirometry, and controls vs. individuals with sarcoidosis and normal spirometry).

Keywords: Sarcoidosis; Forced Oscillation Technique; FOT; Machine Learning; AUC.

LISTA DE FIGURAS

Figura 1 Diagrama básico da FOT. 21
Figura 2 Exemplos de modelos do sistema respiratório. 23
Figura 3 Diagrama do problema do aprendizado. 27
Figura 4 Matriz de confusão. 28
Figura 5 Exemplo de curva ROC. 29
Figura 6 Exemplo de hiperplano ótimo de separação em problema bidimensional. ... 31
Figura 7 Exemplo de Árvore de Decisão. 33
Figura 8 Exemplo de conjunto de regras. 41
Figura 9 Exemplo de mapeamento genótipo-fenótipo em GE. 42
Figura 10 Diagrama de um Sistema *Fuzzy* Baseado em Regras. 44
Figura 11 Exemplo de Árvore de Padrões *Fuzzy*. 45
Figura 12 Esquema de busca *backward* em exemplo com quatro atributos. 48
Figura 13 Representação do vetor de registradores. 51
Figura 14 Representação do indivíduo clássico. 53
Figura 15 Exemplo de mapeamento genótipo-fenótipo com execução em PGLIQ. 54
Figura 16 Representação do indivíduo quântico. 55
Figura 17 Exemplo de construção de atributos com PGLIQ. 63
Figura 18 Fluxograma do modelo proposto. 66
Figura 19 Esquema de fuzzificação. 68
Figura 20 Gramática utilizada pela GE. 70
Figura 21 Diagrama de caixa de cada atributo. 75
Figura 21 Diagrama de caixa de cada atributo. 76
Figura 21 Diagrama de caixa de cada atributo. 77
Figura 22 Desempenho individual dos atributos. 84
Figura 23 Desempenho com o uso direto de classificadores. 85
Figura 24 Exemplo de indivíduo final na PGLIQ. 87
Figura 25 Desempenho com seleção de atributos. 89
Figura 26 Representação dos dados com três atributos. 93
Figura 27 Desempenho com aumento de dados. 94

Figura 28 Desempenho com construção de atributos. 95
Figura 29 Exemplo de indivíduo final na construção de atributos com PGLIQ. 96
Figura 30 Gráficos 3D com atributos construídos para LGR. 99
Figura 31 Gráficos 2D com atributos construídos para LGR. 100
Figura 32 Melhores resultados..... 101

LISTA DE TABELAS

Tabela 1 Parâmetros fornecidos pela FOT. 22
Tabela 2 Parâmetros do modelo eRIC. 25
Tabela 3 Exemplo de conjunto de funções. 36
Tabela 4 Algoritmo Evolutivo da PGLIQ. 40
Tabela 5 Operadores de agregação. 44
Tabela 6 Operadores de média. 44
Tabela 7 Algoritmo evolutivo da construção de atributos com PG. 49
Tabela 8 Descrição das funções. 52
Tabela 9 Conjunto de funções. 53
Tabela 10 Algoritmo Evolutivo da PGLIQ implementada. 58
Tabela 11 Algoritmo Evolutivo da Construção de Atributos com PGLIQ. 64
Tabela 12 Valor-p de cada atributo na análise controle × indivíduos com sarcoidose. . 78
Tabela 13 Atributos fuzzificados com diferenças significativas entre os conjuntos controle e sarcoidose, com espirometria normal ou alterada. 79
Tabela 14 Quantidade de valores diferentes de zero em cada atributo fuzzificado. 81
Tabela 15 Atributos fuzzificados com diferenças significativas entre os conjuntos controle e sarcoidose. 82
Tabela 16 Probabilidades finais médias dos melhores atributos fuzzificados na PGLIQ. 88
Tabela 17 Percentual de aparição dos atributos normalizados. 90
Tabela 18 Maiores percentuais de aparição dos atributos fuzzificados. 91

LISTA DE SIGLAS

AE	Algoritmos Evolutivos
AI	*Artificial Intelligence*
APF	Árvore de Padrões *Fuzzy*
API	*Application Programming Interface*
AUC	*Area under the curve*
DPOC	Doença pulmonar obstrutiva crônica
DTC	*Decision Tree Classifier*
eRIC	Resistência-Inertância-Complacência estendido
FFT	*Fast Fourier Transform*
FOT	*Forced Oscillation Technique*
FPR	*False positive rate*
GE	*Grammatical Evolution*
KNN	*K-nearest neighbors*
ML	*Machine Learning*
PG	Programação Genética
PGLIQ	Programação Genética Linear com Inspiração Quântica
PNT	Pneumotacógrafo
RF	*Random Forests*
ROC	*Receiver Operating Characteristics*
SMOTE	*Synthetic Minority Oversampling Technique*
SVM	*Support Vector Machine*
TP	Transdutor de pressão
TPR	*True positive rate*

SUMÁRIO

	INTRODUÇÃO	14
1	**SARCOIDOSE**	18
2	**TÉCNICA DE OSCILAÇÕES FORÇADAS**	19
3	**MODELOS DO SISTEMA RESPIRATÓRIO**	23
4	**PANORAMA DE APRENDIZADO DE MÁQUINA**	26
4.1	O Problema do Aprendizado	26
4.2	Conceitos Básicos	27
4.3	K-Vizinhos mais Próximos	30
4.4	*Support Vector Machine*	31
4.5	*Ensemble Methods*	31
4.6	Regressor Logístico	33
4.7	Programação Genética	34
4.7.1	Origem e Conceitos Básicos	34
4.7.2	Programação Genética Linear com Inspiração Quântica	35
4.7.2.1	Conceitos Básicos de Computação Quântica	35
4.7.2.2	Descrição Geral	36
4.7.3	Programação Genética Orientada à Gramática	39
4.8	Árvores de Padrões *Fuzzy*	43
4.8.1	Conceitos Básicos de Lógica *Fuzzy*	43
4.8.2	Descrição Geral	44
4.9	Redução da Dimensionalidade	45
4.9.1	Seleção de Atributos	46
4.9.2	Construção de Atributos	47
4.10	Aumento de Dados	49
5	**PROGRAMAÇÃO GENÉTICA LINEAR COM INSPIRAÇÃO QUÂNTICA APLICADA À SíNTESE DE ÁRVORES DE PADRÕES *FUZZY***	51

5.1 Registradores 51
5.2 Representação 52
5.2.1 Indivíduo Clássico 52
5.2.2 Indivíduo Quântico 53
5.3 Aptidão 56
5.4 Atualização dos *qudits* 56
5.5 População 57
5.6 Evolução 57
5.6.1 Inicialização da População Quântica 57
5.6.2 Inicialização da População Clássica 58
5.6.3 Execução das Gerações 59
5.6.4 Atualização dos genes quânticos 59
5.6.5 Avaliação 60

6 **CONSTRUÇÃO DE ATRIBUTOS COM PROGRAMAÇÃO GENÉTICA LINEAR COM INSPIRAÇÃO QUÂNTICA** 62
6.1 Atributos Construídos 62
6.2 Aptidão 63
6.3 Evolução 64

7 **MODELO PROPOSTO** 65
7.1 Normalização 66
7.2 Fuzzificação 67
7.3 Classificadores 68
7.3.1 Algoritmos Básicos 68
7.3.2 Programação Genética 69
7.3.3 Evolução Gramatical 70
7.3.4 Programação Genética Linear com Inspiração Quântica 70
7.4 Seleção de Atributos 71
7.5 Aumento de Dados 72
7.6 Construção de Atributos 72

8 **ESTUDOS DE CASO** 74
8.1 Descrição do Conjunto de Dados 74

8.2 Experimento Individual dos Atributos 82
8.3 Experimento com o Uso Direto de Classificadores 83
8.4 Experimento com Seleção de Atributos 89
8.5 Experimento com Aumento de Dados 92
8.6 Experimento com Construção de Atributos 94
8.7 Principais Resultados 97

CONCLUSÃO 102

REFERÊNCIAS 104

INTRODUÇÃO

A sarcoidose é uma doença granulomatosa multissistêmica, localizada no tórax em 84% dos casos [1], e que afeta pessoas em todo o mundo [2]. Seu diagnóstico costuma precisar de biópsia, e acaba sendo tardio na maioria dos casos, sendo que em muitas situações o paciente pode receber erroneamente o diagnóstico e até o tratamento para tuberculose [3], o que gera riscos que não devem ser desprezados [4].

A espirometria é o método não invasivo padrão utilizado para avaliar a obstrução das vias respiratórias, no entanto, exige grandes e coordenados esforços inspiratórios e expiratórios dos pacientes e, portanto, não é adequada para bebês, crianças pequenas ou pessoas com doenças graves [5].

Outro método não invasivo, que é executado durante respiração espontânea, exigindo somente cooperação passiva dos pacientes, é a Técnica de Oscilações Forçadas (do inglês *Forced Oscillation Technique*, ou FOT) [6], que foi apresentada para descrever a impedância do sistema respiratório. Essa impedância pode ser interpretada fisiologicamente através de modelos da mecânica pulmonar [5], como por exemplo o modelo resistência-inertância-complacência estendido (eRIC) [7], que representa a impedância do sistema respiratório por dois resistores, um capacitor e um indutor.

A FOT complementa as técnicas clássicas ao proporcionar novos parâmetros para análise permitindo uma avaliação mais detalhada [8] [9], podendo auxiliar no diagnóstico tanto da sarcoidose quanto de outras doenças respiratórias. Entretanto, ainda é uma técnica em fase experimental. Para propiciar seu uso clínico, é necessário identificar os parâmetros mais apropriados ao diagnóstico de cada doença respiratória individualmente [8]. Assim, têm sido realizados diversos estudos para analisar, através da FOT, anomalias na mecânica respiratória referentes à sarcoidose [10] [11], dentre outras enfermidades.

Apesar da FOT ser uma técnica com execução fácil, analisar o sistema respiratório interpretando curvas de resistência e reatância e os parâmetros derivados dessas curvas, percebendo anomalias específicas de cada doença, pode ser uma tarefa difícil que demande treinamento e experiência de profissionais de saúde [12]. Assim, mostra-se uma boa opção usar algoritmos de aprendizado de máquina para gerar resultados mais facilmente interpretáveis.

Aprendizado de Máquina (do inglês *Machine Learning*, ou ML) é um campo da

Inteligência Artificial (do inglês *Artificial Intelligence*, ou AI) que dá aos computadores a capacidade de aprender, sem serem programados de forma explícita para tal [13]. Suas metodologias são utilizadas principalmente em problemas que não possuem solução determinística, usando-se de dados para que os algoritmos descubram de forma automática a relação entre os mesmos.

Nos últimos anos, o uso de ML na medicina passou a ser mais frequente, contribuindo para o diagnóstico de doenças, análise de dados, planejamento de tratamentos, etc. [14]. Seus algoritmos são aplicados ao estudo de uma imensidão de enfermidades, como por exemplo diabetes [15], doenças do coração [16], diversos tipos de câncer [17] e claro, doenças respiratórias [18].

Diversos trabalhos têm relatado que é viável utilizar os parâmetros obtidos pela FOT para aplicar algoritmos de aprendizado de máquina no estudo e auxílio ao diagnóstico de doenças respiratórias. O artigo [19] desenvolve um sistema de apoio à decisão clínica para auxiliar o diagnóstico da doença pulmonar obstrutiva crônica, enquanto que um posterior [20] propõe classificadores para categorizar o grau de obstrução das vias respiratórias em pacientes com essa doença. Já o artigo [12] aplica aprendizado de máquina para aumentar a acurácia da FOT no diagnóstico de anomalias respiratórias precoces em fumantes. Há também o artigo [21] que apresenta classificadores automáticos para o diagnóstico de obstrução das vias aéreas em pacientes com asma. Por fim, o trabalho mais recente na literatura envolvendo aprendizado de máquina e os atributos fornecidos pela FOT [22] elabora um sistema de apoio ao diagnóstico diferencial de asma e doenças respiratórias restritivas, utilizando diversos algoritmos e, particularmente, um classificador *neuro-fuzzy*, que apresenta regras simples para explicar a saída obtida, tornando mais fácil a tarefa de interpretação dos resultados.

No entanto, ainda não existem trabalhos na literatura utilizando os métodos de aprendizado de máquina no aprimoramento do diagnóstico de problemas respiratórios decorrentes da sarcoidose. Este projeto visa desenvolver classificadores que apresentem bom desempenho nessa tarefa, e também que possuam resultados interpretáveis. A interpretabilidade do modelo é importante no estudo das doenças, pois aumenta a compreensão de como foi feita a discriminação da alteração respiratória. Assim, será explorada Programação Genética (PG), devido à classificação ser feita por meio de expressões inteligíveis. Será utilizada a sua forma tradicional em árvores, a orientada à gramática, e

também a linear com inspiração quântica (PGLIQ). Para verificar se os resultados interpretáveis são competitivos, diversos outros algoritmos serão utilizados em comparação. Para trazer ainda mais interpretabilidade, também serão explorados sistemas *Fuzzy*, porque eles permitem que as informações obtidas a partir de uma base de dados sejam representadas com termos linguísticos compreensíveis. Entretanto, como é esperado que o número de atributos de entrada seja grande, não será desenvolvido um Sistema *Fuzzy* Baseado em Regras, que necessitaria de muitas regras, e consequentemente prejudicaria a interpretabilidade do modelo. Assim, serão usadas Árvores de Padrões *Fuzzy* (APFs), sintetizadas pela PG em suas diversas formas. Uma última abordagem que também será explorada utiliza a PGLIQ para construir novos atributos, por meio de combinações dos atributos originais, a fim de empregá-los em um outro classificador.

Objetivo

Esta dissertação propõe elaborar um sistema de classificação que receba como entrada as medidas coletadas de um paciente através da FOT, e retorne o diagnóstico acerca da presença ou ausência de alterações respiratórias associadas à sarcoidose. Os dados para o aprendizado são fornecidos pelo Laboratório de Instrumentação Biomédica, desta mesma instituição de ensino, e foram obtidos através da FOT de pacientes com e sem sarcoidose. Serão desenvolvidos classificadores baseados em ML para discriminar os diferentes grupos e também serão explorados modelos interpretáveis para aprender sobre os atributos mais importantes nessa tarefa.

Organização da Dissertação

Esta dissertação está organizada da seguinte forma:

- Os Capítulos 1 a 4 apresentam uma revisão bibliográfica, começando pela sarcoidose. A seguir, a FOT é descrita em detalhes com seus procedimentos e atributos resultantes. Logo depois, são descritos modelos de pulmão, a partir dos quais são apresentados ainda outros atributos usados neste trabalho. Por fim, o Capítulo 4 encerra essa parte inicial fazendo uma revisão geral de ML, e são apresentados alguns algoritmos utilizados;
- Nos Capítulos 5 e 6 são descritos novos modelos de aplicação da PGLIQ, começando

pelo seu uso para sintetizar APFs, e a seguir para construir atributos para outros algoritmos;

- O Capítulo 7 detalha todas as características do modelo proposto neste trabalho, isto é, como os dados são tratados, quais os algoritmos utilizados, quais as implementações desses algoritmos e como os principais hiperparâmetros de cada um deles são escolhidos;

- Por fim, o Capítulo 8 apresenta os estudos de caso para o modelo proposto. É feita ainda uma descrição estatística acerca dos dados utilizados nesta dissertação e também é avaliada a capacidade de cada atributo realizar individualmente a classificação.

1 SARCOIDOSE

A sarcoidose é uma doença inflamatória caracterizada pela presença de granulomas[1], que podem aparecer em praticamente qualquer órgão [23] , embora o pulmão seja o local mais comum. Mais de 150 anos depois de sua primeira descrição clínica [24], a causa da sarcoidose permanece desconhecida e seu tratamento é geralmente insatisfatório [25].

Desenvolve-se principalmente antes dos 50 anos, com uma incidência maior na faixa de 20 a 40 anos [26]. A ocorrência no mundo varia, sendo que, nos Estados Unidos, a incidência em afro-americanos é de 35,5 casos a cada 100 mil habitantes, enquanto que em caucasianos é de apenas 10,9 casos por 100 mil [27]. Ainda não é completamente conhecida a causa da sarcoidose, como uma infecção ou um corpo estranho, nem a explicação para sua ampla variabilidade fenotípica [28]. Como é uma doença com grande heterogeneidade, presume-se que haja uma variedade de fatores desencadeantes. Há também indicação de predisposição genética, pois 4% dos casos são familiares [29].

O diagnóstico da sarcoidose é feito quando a probabilidade de diagnósticos alternativos que expliquem a inflamação granulomatosa se torna muito pequena para justificar mais investigação [30]. Assim, não há um teste específico para o diagnóstico da sarcoidose, e é impossível excluir completamente outros diagnósticos [2]. Em média, os pacientes manifestam sintomas por três meses e necessitam de ao menos três consultas médicas antes do diagnóstico [31].

[1] nódulos de caráter inflamatório

2 TÉCNICA DE OSCILAÇÕES FORÇADAS

A Técnica de Oscilações Forçadas (FOT) [6] se trata de um método não invasivo, executado durante respiração espontânea, que consiste em aplicar-se oscilações com baixo nível de pressão e de amplitude no sistema respiratório de uma pessoa utilizando-se um dispositivo externo. Essa técnica foi apresentada para descrever a impedância do sistema respiratório e suas duas componentes: a resistência (R_{rs}) e a reatância (X_{rs}) [32].

Impedância é o termo que descreve a relação expressa no domínio da frequência entre a entrada e a saída de um sistema dinâmico linear [33]. Por exemplo, impedância elétrica é uma função complexa no domínio da frequência dada pela razão entre tensão (saída) e corrente (entrada). Similarmente, quando se trata de impedância mecânica, a entrada é a velocidade e a saída é a força. Seguindo nessa mesma analogia, ao se analisar a mecânica dos pulmões os equivalentes à corrente e tensão elétricas serão fluxo de ar e pressão, respectivamente. Representando pressão e volume em função do tempo como $P(t)$ e $V(t)$, nesta ordem, o fluxo de ar será $\dot{V}(t)$. Assim, a impedância mecânica do sistema respiratório pode ser escrita como a Equação 1.

$$Z_{rs}(\omega) = \frac{P(\omega)}{\dot{V}(\omega)} \tag{1}$$

Em que $\omega = 2\pi f$ é a frequência angular, $P(\omega)$ e $\dot{V}(\omega)$ são as transformadas rápidas de Fourier (do inglês *fast Fourier transform*, ou FFT) de $P(t)$ e $\dot{V}(t)$, respectivamente, e $Z_{rs}(\omega)$ é a impedância respiratória.

A Z_{rs} corresponde à carga mecânica total concedida pelo sistema respiratório, ou seja, à capacidade do sistema em se opor à passagem de fluxo aéreo, o que envolve as propriedades de resistência, complacência e inertância [32]. Sua forma cartesiana é exibida na Equação 2.

$$Z_{rs}(\omega) = R_{rs}(\omega) + jX_{rs}(\omega) \tag{2}$$

Onde são apresentadas a resistência (R_{rs}) e a reatância (X_{rs}), que em módulo se relacionam com a impedância conforme a Equação 3.

$$|Z_{rs}| = \sqrt{|R_{rs}|^2 + |X_{rs}|^2} \tag{3}$$

A componente R_{rs} engloba as perdas por atrito que ocorrem durante o fluxo de ar e as resistências associadas ao tecido pulmonar, à parede torácica e à glote[2] [9]. E a componente X_{rs} representa o armazenamento de energia no sistema respiratório [32], sendo a energia potencial associada à complacência dinâmica ($Cdin$) e a energia cinética à inertância (I), de acordo com a Equação 4.

$$X_{rs} = \omega I - \frac{1}{\omega C_{din}} \tag{4}$$

A complacência pulmonar é definida como a razão da diferença de volume dos pulmões pela diferença na pressão transpulmonar [34]. O seu inverso é a elastância, que é uma característica do sistema respiratório que existe devido às propriedades elásticas do tecido pulmonar. Essas propriedades permitem que os pulmões se inflem durante a inspiração e retornem ao seu formato original após a expiração. Assim, pode-se entender que se algum fator diminuir a complacência, haverá dificuldade de inspiração, e se diminuir a elastância, haverá dificuldade de expiração.

Pode-se definir a inertância como a resistência ao movimento do sistema respiratório, isto é, ao deslocamento das massas de ar nas vias aéreas. Seus efeitos são desprezíveis nas frequências respiratórias habitualmente atingidas na respiração espontânea e na ventilação mecânica [35]. Assim, analisando-se a Equação 4, percebe-se que em baixas frequências, a reatância terá um valor negativo. Conforme se aumentar a frequência, o termo com a inertância se tornará relevante até que atingirá um ponto tal que os dois termos se anulem e a reatância seja zero [36]. Nesse ponto, é definida a frequência de ressonância (Fr), que é encontrada com a Equação 5. Acima desta frequência, o termo com a inertância predomina e a reatância passa a ser positiva.

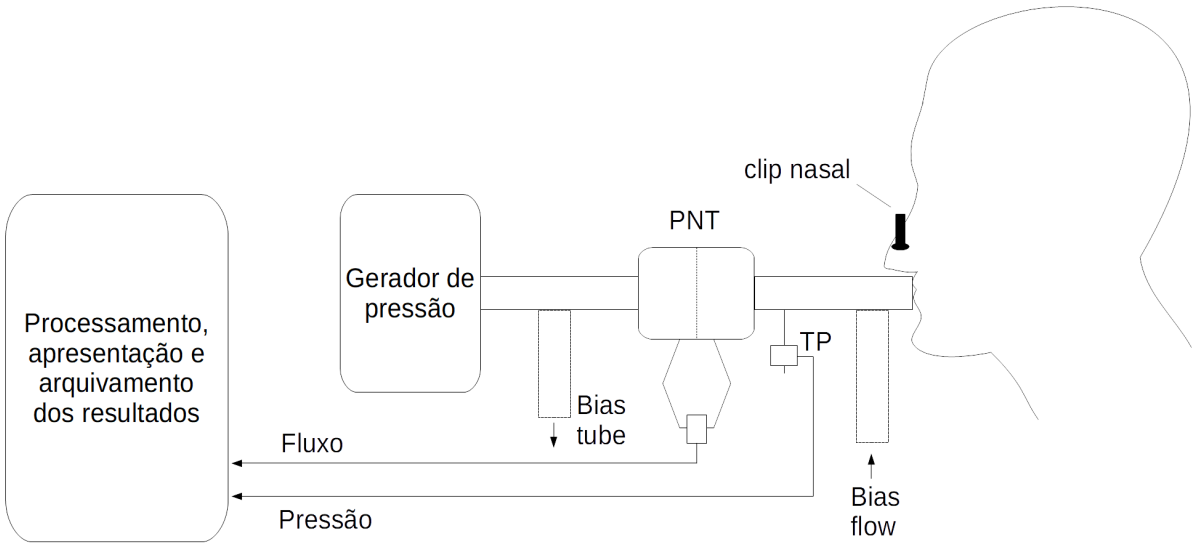
$$\begin{aligned} \omega_r I &= \frac{1}{\omega_r C_{din}}, \quad \omega_r = 2\pi f_r \\ f_r &= \frac{1}{2\pi\sqrt{I C_{din}}} \end{aligned} \tag{5}$$

A árvore traqueobrônquica é a estrutura responsável por levar o ar aspirado até os pulmões. É formada por padrões geométricos complexos, o que permite possíveis não-homogeneidades no sistema respiratório [37]. O processo de movimentação de ar entrando e saindo dos pulmões é chamado de ventilação, que não é igualmente distribuída mesmo

[2]estrutura localizada na parte final da laringe

em pulmões saudáveis, piorando de acordo com certas doenças [38].

Figura 1: Diagrama básico da FOT.



Fonte: Adaptado de [32]

A Figura 1 apresenta de forma simplificada um sistema usado para realizar a FOT [32]. Pequenos sinais senoidais de pressão (P) com frequências múltiplas de 2 na faixa de 4–32 Hz são aplicados por meio de um dispositivo externo à entrada do sistema respiratório de um paciente, que se mantém sentado, utilizando um clipe nasal e respirando espontaneamente. A pressão aplicada é medida pelo transdutor de pressão TP e o fluxo de ar induzido por ela ($\dot{V}$) é medido pelo pneumotacógrafo PNT. Assim, é possível usar a transformada de Fourier para estimar a impedância respiratória, e são geradas curvas de resistência e reatância em função da frequência. Realiza-se uma regressão linear na curva de resistência na faixa de 4–16 Hz, a fim de estimar a resistência na frequência zero, chamada de resistência no intercepto ($R0$), e também para calcular a inclinação dessa curva (S) e a resistência média (Rm) nessa faixa. Na prática, associa-se $R0$ e S à resistência total e à falta de homogeneidade do sistema respiratório, respectivamente, e Rm à resistência das vias aéreas centrais [39]. A complacência dinâmica ($Cdin$) é calculada a partir da reatância obtida a 4 Hz [40], utilizando-se a Equação 4, desprezando-se o valor da inertância. Essa mesma frequência é utilizada para calcular o valor absoluto da impedância respiratória ($Z4$), um parâmetro associado ao trabalho realizado pelos músculos respiratórios para superar cargas resistivas e elásticas, de forma a permitir o

fluxo de ar no sistema respiratório [41]. A reatância média (Xm), também associada à falta de homogeneidade do sistema respiratório, é calculada através da curva de reatância baseando-se em toda a faixa de frequência estudada (4–32 Hz) [42]. Por fim, calcula-se a área sob a parte negativa da curva de reatância (Ax), ou seja, entre 4 Hz e Fr, que reflete a variação do nível de obstrução nas vias aéreas periféricas [36].

A resistência medida em baixa frequência é associada à resistência total das vias aéreas, enquanto que a resistência medida em alta frequência indica a resistência central das vias aéreas, e portanto, a diferença entre elas é considerada um índice da resistência de vias aéreas menores [43] [44]. Assim, os demais parâmetros fornecidos pela FOT para este trabalho são a resistência a 4 Hz ($R4$), a resistência a 20 Hz ($R20$) e a diferença entre essas duas ($R4 - R20$).

A Tabela 1 resume os parâmetros fornecidos pela FOT, apresentados nos parágrafos anteriores, que são utilizados neste trabalho.

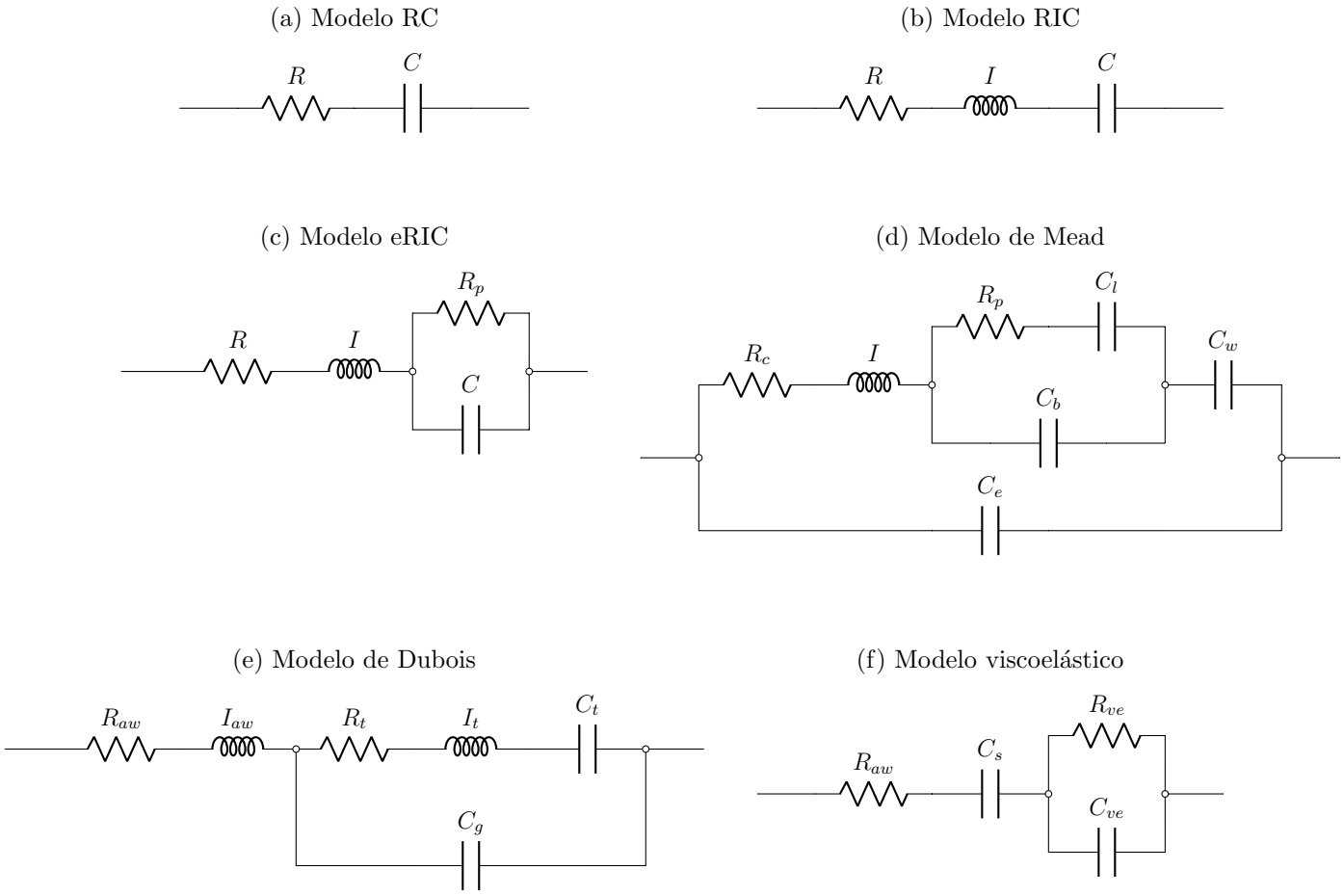
Tabela 1: Parâmetros fornecidos pela FOT.

Parâmetro	**Descrição**
R0	Resistência no intercepto
S	Inclinação da curva de resistência
Rm	Resistência média na faixa de 4–16 Hz
R4	Resistência a 4 Hz
R20	Resistência a 20 Hz
R4-R20	Diferença entre R4 e R20
Xm	Reatância média na faixa de 4–32 Hz
Fr	Frequência de ressonância
Cdin	Complacência dinâmica
Ax	Área da parte negativa da curva de reatância
Z4	Impedância a 4 Hz

3 MODELOS DO SISTEMA RESPIRATÓRIO

As curvas de impedância fornecidas pela FOT podem ser facilmente analisadas usando-se conceitos de engenharia, a fim de serem correlacionadas com modelos compostos por componentes elétricos, que são análogos à resistência, à inertância e à complacência do sistema respiratório. Assim, utilizando processamento computacional, pode-se estimar os valores dos componentes desses modelos, que poderão ser úteis para o diagnóstico de diversas doenças [7]. Vários trabalhos já vêm sendo realizados nesta linha, como por exemplo, associando parâmetros dos modelos com anormalidades na silicose [45], evidenciando que os modelos podem auxiliar no diagnóstico precoce da doença pulmonar obstrutiva crônica (DPOC) [46] e usando os parâmetros para detectar obstrução leve na asma [47].

Figura 2: Exemplos de modelos do sistema respiratório.



Fonte: Adaptado de [48]

A Figura 2 exibe alguns exemplos de modelos do sistema respiratório desenvolvidos ao longo dos anos.

No modelo RC (Figura 2a), o sistema respiratório é representado como um resistor em série com um capacitor, no qual a capacitância C reflete as propriedades elásticas dos pulmões e da parede torácica, e a resistência R engloba a resistência ao fluxo de ar nas vias aéreas e as propriedades viscosas do tecido pulmonar [49]. Esse modelo pode ser utilizado para descrever a mecânica pulmonar durante a respiração em repouso, mas é insuficiente para análises em esforço respiratório. Isso foi corrigido no modelo RIC (Figura 2b), que acrescenta um indutor em série, correlacionando sua indutância com a inertância pulmonar. Pelo pequeno número de parâmetros, é um modelo útil para estudos simples, entretanto, por ser de compartimento único não é capaz de ser usado para investigar as não homogeneidades da mecânica pulmonar.

A consideração de mais propriedades do sistema respiratório aumenta significativamente o número de parâmetros dos modelos. Mead [50] propôs um modelo de sete parâmetros (Figura 2d), que permite a simulação de diferentes influências na mecânica respiratória, possibilitando a investigação de diferentes causas de obstruções das vias aéreas e também a avaliação da influência dos equipamentos nas medições [49]. Esse modelo representa a inertância do sistema respiratório (I), as resistências central (R_c) e periférica (R_p) e as complacências dos pulmões (C_l), das paredes torácicas (C_w), dos brônquios (C_b) e extratorácica (C_e).

O modelo sugerido por Bates et al. [51] (Figura 2f) sugere que as propriedades viscoelásticas[3] da parede torácica são as principais responsáveis pelo comportamento mecânico do sistema respiratório. A resistência R_{aw} representa a resistência das vias aéreas, C_s é a complacência estática do sistema respiratório, e R_{ve} e C_{ve} estão relacionados às propriedades viscoelásticas dos tecidos [52].

Dubois [6] propôs dividir as propriedades das vias aéreas, dos tecidos e dos alvéolos em diferentes repartições. Assim, o modelo da Figura 2e apresenta uma seção contendo a resistência e a inertância das vias aéreas (R_{aw} e I_{aw}), outra com as dos tecidos (R_t e I_t) e uma com as dos alvéolos (R_g e I_g) [7].

O modelo RIC estendido (eRIC) foi introduzido em [48] e pode ser visto na Figura 2c. Consiste em uma extensão do modelo RIC ao adicionar a resistência periférica (R_p)

[3] simultaneamente elásticas e viscosas

e associá-la em paralelo com a complacência, o que permitiu obter uma dependência da impedância com a frequência mais próxima da observada em dados reais. Esse modelo pode ser interpretado como uma simplificação do modelo de DuBois (com I_t tendendo a zero e C_t tendendo ao infinito) ou do modelo de Mead (com C_l, C_w tendendo ao infinito e C_e tendendo a zero).

Para cada um dos modelos citados, é calculada analiticamente a impedância equivalente ao circuito respectivo. Por exemplo, para o modelo eRIC a impedância é calculada segundo a Equação 6, sendo ω a frequência angular em radianos/segundo. Assim, para cada modelo, os valores dos parâmetros precisam ser determinados de forma a minimizar o erro entre a impedância medida em frequências discretas e o seu respectivo resultado analítico. Esse processo de otimização é chamado de estimativa de parâmetros [53].

$$Z = R + \frac{R_p}{1 + (\omega R_p C)^2} + j\left(\omega I - \frac{\omega R_p^2 C}{1 + (\omega R_p C)^2}\right) \tag{6}$$

A análise feita em [7] compara os erros de estimativa de parâmetros para os modelos citados neste trabalho. Esse estudo utilizou dados provenientes de adultos saudáveis e outros diagnosticados com alguma doença respiratória do tipo obstrutiva ou restritiva, incluindo bronquiectasia, DPOC, asma e sarcoidose, e também de crianças saudáveis e com asma. Verificou-se que o modelo de Mead apresentou o menor número de erros de estimativa para esses conjuntos de dados. Entretanto, certos parâmetros estimados apresentaram alguns valores muito distantes das faixas normais, enquanto que o modelo eRIC alcançou uma média de erros próxima ao modelo de Mead, sem esse problema. Além disso, analisando adultos saudáveis e enfermos, a combinação de dois ou mais parâmetros do modelo eRIC apresentou acurácia na discriminação entre saúde e certas doenças.

Assim, o modelo eRIC foi escolhido para esse trabalho, e os seus parâmetros podem ser resumidos segundo a Tabela 2.

Tabela 2: Parâmetros do modelo eRIC.

Parâmetro	**Descrição**
ReRIC	Resistência central
RpeRIC	Resistência periférica
RteRIC	Resistência total (ReRIC + RpeRIC)
IeRIC	Inertância do sistema respiratório
CeRIC	Complacência do sistema respiratório

4 PANORAMA DE APRENDIZADO DE MÁQUINA

Neste capítulo são revisados alguns conhecimentos básicos acerca de aprendizado de máquina, que serão necessários para o entendimento dos próximos capítulos. Também são apresentados diversos métodos, já muito utilizados na literatura, que também serão aproveitados por este trabalho, a fim de comparar o seu desempenho com o de métodos novos que serão apresentados nos capítulos seguintes.

4.1 O Problema do Aprendizado

Aprender é o que traz flexibilidade às nossas vidas. O fato de reconhecer que já estivemos em determinada situação anteriormente, reexecutar uma ação específica porque ela havia funcionado antes, e ao não funcionar novamente, tentar algo diferente. E, mais essencialmente, perceber alguma semelhança em situações distintas e aplicar as mesmas resoluções, ao que definimos como generalização, é o que torna o aprendizado útil. Assim, o conceito-chave é que se aprende com a experiência [54]. Pode-se dizer que um programa de computador aprende com uma experiência E em relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho nas tarefas T, medido por P, melhorar com a experiência E [13]. Resumidamente, um problema de aprendizado estará definido ao identificarmos a classe de tarefas, a medida do desempenho a ser aprimorada e a fonte da experiência.

Em geral, os problemas de aprendizado podem ser divididos em [55]:

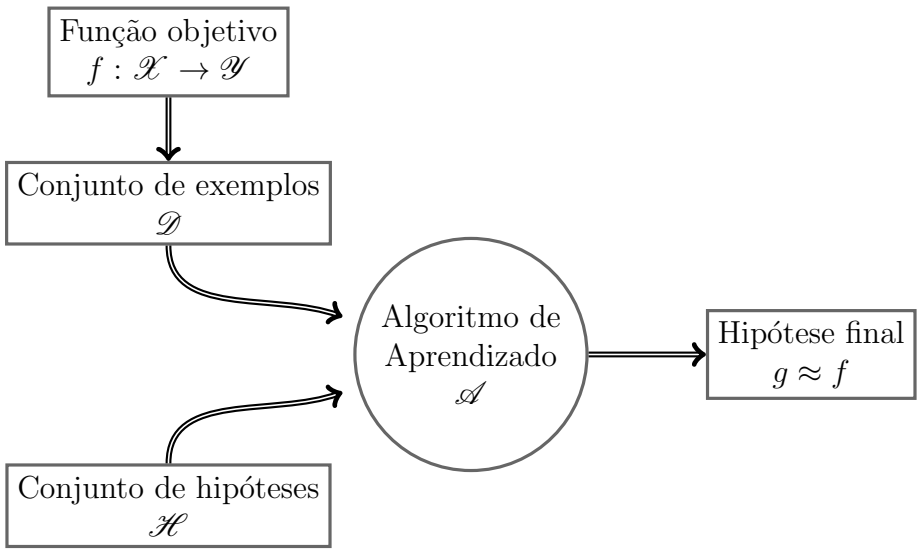
- Aprendizado supervisionado: cada instância do conjunto de dados de entrada está identificada com um rótulo, e o algoritmo busca generalizar, de forma a rotular corretamente futuras entradas.

- Aprendizado não-supervisionado: não há rótulos, e o algoritmo busca identificar padrões nos dados de entrada, de forma a separá-los em grupos que contenham similaridades.

- Aprendizado por reforço: o algoritmo é informado se a resposta está errada, mas não especificamente o quão errada está. Assim, é necessário explorar diferentes possibilidades até encontrar a resposta certa.

O formalismo do problema de aprendizado supervisionado, que é o fruto de interesse deste trabalho, pode ser visto na Figura 3. Definimos a função ideal como $f : \mathscr{X} \rightarrow \mathscr{Y}$, em que $\mathscr{X}$ é o conjunto de todas as possíveis entradas $\mathbf{x}$ e $\mathscr{Y}$ é o conjunto das possíveis saídas. $\mathscr{D}$ é um conjunto de dados de entrada rotulados $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_M)$, também chamados de amostras ou instâncias, em que $y_m = f(\mathbf{x}_m)$, para $m = 1, ..., M$. Cada entrada $\mathbf{x}_m$ pode ser representada por um vetor da forma $(x_{m,1}, x_{m,2}, ..., x_{M,N}$, em que N é o número de atributos de cada amostra. Assim, pode-se usar a notação x_{mn}, para se referir ao n-ésimo atributo da instância $\mathbf{x}_m$ [56]. Resumindo, um algoritmo de aprendizado $\mathscr{A}$ vai usar o conjunto $\mathscr{D}$ para encontrar em um conjunto de hipóteses $\mathscr{H}$, a função $g : \mathscr{X} \rightarrow \mathscr{Y}$, que é a mais próxima de f [57].

Figura 3: Diagrama do problema do aprendizado.



Fonte: Adaptado de [57]

4.2 Conceitos Básicos

Definindo um modelo simples baseado na Figura 3, temos que $\mathscr{X} = \mathbb{R}^d$ é um espaço euclidiano de d dimensões, e $\mathscr{Y}$ é um conjunto de valores discretos, denotando que temos um problema de classificação. Sendo $h \in \mathscr{H}$, é necessário definir uma métrica que quantifique o quanto h se aproxima de f, de forma a identificar a hipótese final g que é a mais próxima possível de f. E ao se tratar da medida de desempenho do modelo, afeta diretamente o processo de aprendizagem, pois diferentes métricas podem levar a diferentes escolhas da hipótese final g, ainda que f e $\mathscr{D}$ sejam os mesmos [57].

Em problemas de classificação, um método pelo qual se podem obter diversas métricas, é chamado de matriz de confusão. Consiste em uma matriz quadrada de ordem c, sendo c o número de classes possíveis, na qual as linhas representam as saídas previstas pelo modelo utilizado e as colunas os valores-alvo [54]. Para um modelo em que $\mathscr{Y} = \{-1, +1\}$, ou seja, de classificação binária, a matriz de confusão é vista na Figura 4, na qual a observação correta da classe +1 indica um verdadeiro positivo (do inglês *true positive*, ou TP) e a incorreta indica um falso positivo (do inglês *false positive*, ou FP), enquanto que a observação correta da classe −1 indica um verdadeiro negativo (do inglês *true negative*, ou TN) e a incorreta um falso negativo (do inglês *false negative*, ou FN).

Figura 4: Matriz de confusão.

		f	
		+1	-1
h	+1	Verdadeiros positivos	Falsos positivos
	-1	Falsos negativos	Verdadeiros negativos

Fonte: Adaptado de [57]

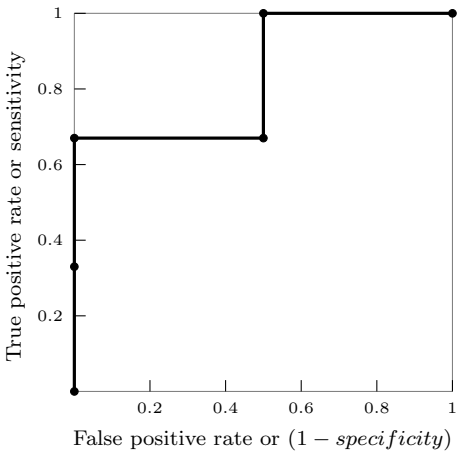
A fim de comparar modelos distintos, ou ainda um mesmo modelo com diferentes parâmetros, é útil realizar uma análise ROC (do inglês *Receiver Operating Characteristics*) [54]. Essa análise examina a relação entre sensibilidade[4] e especificidade[5] de um classificador binário, através de um gráfico bidimensional. Convencionalmente, a taxa de verdadeiros positivos (do inglês *true positive rate*, ou TPR), isto é, a sensibilidade é plotada no eixo das ordenadas, e a taxa de falsos positivos (do inglês *false positive rate*, ou FPR), ou $(1 - specificity)$ é plotada no eixo das abcissas [58]. A Figura 5 apresenta um exemplo de curva ROC.

A curva ROC pode ser resumida pela área sob a curva (do inglês *area under the curve*, ou AUC) [59], facilitando a comparação de diferentes classificadores, pois a AUC pode ser usada como métrica de desempenho de cada classificador [60], e tem uma importante propriedade estatística: ao se ordenar de forma decrescente as saídas de um modelo, e escolher aleatoriamente duas saídas rotuladas com classes diferentes, a AUC

[4] definida como a taxa de acerto dentre o que é realmente da classe +1, isto é, $\frac{TP}{TP+FN}$

[5] definida como a taxa de acerto dentre o que é realmente da classe −1, isto é, $\frac{TN}{TN+FP}$

Figura 5: Exemplo de curva ROC.



pode ser interpretada como a probabilidade da instância rotulada com a classe +1 estar em uma posição acima da rotulada com a classe −1 [61].

Em termos de diagnóstico médico, a AUC pode medir a capacidade de um modelo em discriminar se uma condição está presente ou não. Um modelo com AUC ≥ 0.90 é considerado de alta acurácia, e outro com $0.70 \leq \text{AUC} < 0.90$ é considerado de moderada acurácia [22].

Por fim, um último conceito básico a ser explanado nesta seção, consiste na necessidade de se dividir o conjunto de dados a ser usado no modelo. Conforme foi exposto na Seção 4.1, a generalização é o que torna o aprendizado útil. Assim, para garantir que um classificador tenha boa capacidade de generalização, é imprescindível testá-lo em um conjunto diferente daquele que foi usado para o seu aprendizado. Desta forma, tem-se um conjunto de treinamento, usado para treinar o modelo, e um conjunto de teste, usado para legitimar o resultado final. Também é usual utilizar um conjunto de validação para acompanhar o desempenho do aprendizado [54].

Usualmente, deseja-se usar o máximo possível de dados para treinar o modelo e também a maior quantidade disponível para testar sua capacidade de generalização. Não sendo plausível usar todos para treino ou todos para teste, alguns procedimentos podem ser seguidos para a separação dos dados. A escolha de um método ou de outro, e até mesmo a combinação de alguns, depende da natureza do problema e da quantidade de amostras disponíveis. Abaixo estão listados alguns exemplos desses recursos [62]:

- Validação cruzada com k pastas: o conjunto de dados é dividido aleatoriamente em k partes. Retira-se uma das partes, treina-se o modelo com as $k - 1$ restantes e

calcula-se o erro na parte retirada. Assim, repete-se k vezes, a cada vez retirando uma parte diferente e estima-se o erro final a partir da média dos resultados.

- *Leave-one-out*: um caso particular de validação cruzada em que o valor k é igual ao número de amostras no conjunto de dados.

- *Bootstrap*: Consiste em se amostrar n vezes com repetição o conjunto de dados, a fim de gerar o conjunto de treinamento, sendo n o número de instâncias do conjunto original. Assim, haverá elementos repetidos no conjunto de treino, e a probabilidade de uma amostra não ser escolhida é dada por $(1 - \frac{1}{n})^n \approx e^{-1} = 0.368$. Portanto, aproximadamente 37% dos dados não serão escolhidos. Essa parte é chamada de *out-of-bag*, e é usada como conjunto de teste.

4.3 K-Vizinhos mais Próximos

Um dos métodos mais simples e elegantes para aprendizado de máquina é o classificador dos K-vizinhos mais próximos (do inglês *K-nearest neighbors*, ou KNN) [62]. Esse algoritmo considera que todas as instâncias equivalem a pontos em um espaço de n dimensões, sendo n o número de atributos de cada instância. Por ser uma técnica baseada em instâncias, a etapa de treinamento consiste simplesmente em coletar um conjunto de instâncias rotuladas [21]. Assim, quando uma amostra x_q precisa ser classificada, a simplicidade desse método pode ser resumida como: se não há um modelo que descreva os dados, então o algoritmo busca as K amostras de treinamento mais próximas, e a classificação é realizada com a classe majoritária entre essas K instâncias [54].

Formalmente podemos representar uma determinada instância por um vetor da forma $(a_1(x), a_2(x), ..., a_n(x))$, em que $a_r(x)$ é o valor do r-ésimo atributo da instância $\mathbf{x}$. E usualmente, os K vizinhos mais próximos de uma instância a ser classificada são calculados em termos da distância euclidiana, segundo a Equação 7, em que $d(x_i, x_j)$ é a distância entre as instâncias x_i e x_j [13].

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{7}$$

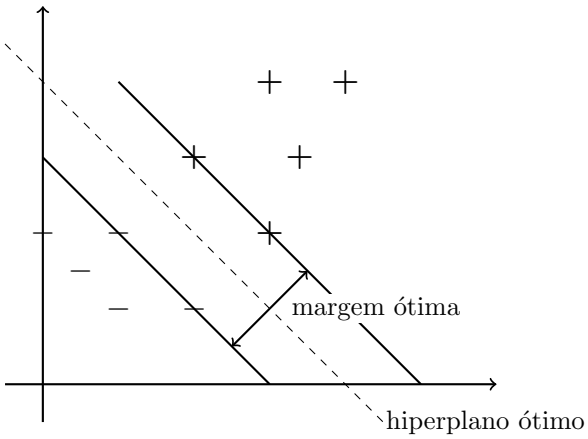
Este método pode ter problemas com a dimensionalidade, pois o custo computacional aumenta conforme aumentar o número de dimensões, e também à medida que crescer

o número de amostras de treino, dado que é necessário calcular as distâncias para todas as instâncias, a fim de definir quais estão mais próximas [54].

4.4 *Support Vector Machine*

O algoritmo *Support Vector Machine* (SVM) consiste em estabelecer entre as classes um hiperplano linear de separação com margem máxima, tal como exemplificado na Figura 6, com o objetivo de generalizar o aprendizado. Caso os dados não sejam separáveis, o conjunto de entrada de m dimensões é mapeado de forma não-linear em um espaço de l dimensões, sendo $l \geq m$, no qual as classes sejam mais facilmente separáveis [63].

Figura 6: Exemplo de hiperplano ótimo de separação em problema bidimensional.



Fonte: Adaptado de [64]

4.5 *Ensemble Methods*

O uso de *ensemble methods* em aprendizado de máquina consiste em se criar um conjunto de classificadores individuais e então combiná-los, de forma que o seu desempenho em grupo seja superior ao de um único estimador [65]. Algumas dessas ferramentas são o *bagging* (acrônimo de **b**ootstrap **agg**regat**ing**) [66], o *boosting* [67] e as florestas aleatórias (do inglês *random forests*, ou RF) [68].

O *bagging* se baseia em construir diferentes conjuntos de instâncias, a partir do conjunto de treinamento, usando a estratégia *bootstrap*, vista na Seção 4.2. A seguir, para um problema de classificação, cada conjunto é treinado separadamente com um mesmo algoritmo de classificação, gerando diferentes classificadores. Assim, para classificar de-

terminada amostra, são utilizados todos os estimadores, e a classe é definida por voto majoritário dentre suas saídas [62].

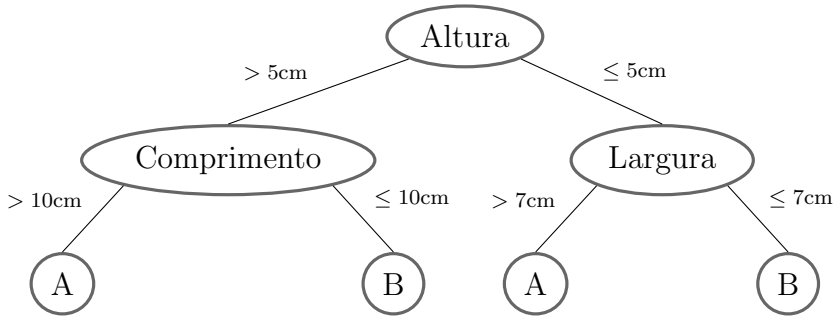
Uma outra ferramenta, o *boosting*, foi criada com o intuito de impulsionar o desempenho de um algoritmo considerado 'fraco', partindo do princípio que é mais fácil criar vários classificadores com baixa acurácia do que um altamente acurado [69]. Dentre aperfeiçoamentos posteriores, destaca-se o *Adaboost* (acrônimo de **Ada**ptive **Boost**ing) [70].

Tal como o *bagging*, o *AdaBoost* gera um conjunto de classificadores, baseados em algum algoritmo, executados em amostragens diferentes feitas com o conjunto de treinamento, e define seu resultado a partir de uma combinação dos resultados de cada estimador. Entretanto, o *bagging* pode gerar esses classificadores paralelamente, enquanto que o *AdaBoost* necessita gerá-los de forma sequencial [71]. Isso porque, partindo inicialmente de uma amostragem com distribuição uniforme do conjunto de treinamento, a cada turno, quando um estimador fraco é criado, a partir do seu resultado são fornecidos pesos às instâncias, de maneira que aquelas classificadas erroneamente apareçam com mais frequências nas amostragens seguintes. Outra diferença é que, ao final, para se obter o resultado dado pelo conjunto de classificadores, os resultados de cada classificador são combinados de forma ponderada. Por exemplo, um estimador de acurácia 0.5 tem peso zero, enquanto que um abaixo disso tem peso negativo [69].

Um algoritmo bastante utilizado como estimador base no *bagging* e no *AdaBoost* é o das árvores de decisão. Essas árvores possuem, por convenção, o nó raiz na parte superior, que é conectado por ramificações sucessivas a outros nós, até chegar aos nós terminais, de onde não saem mais ramificações. A classificação é feita partindo-se do nó raiz, e em cada nó é feito um teste em uma determinada variável e os ramos representam o caminho a seguir, dependendo do resultado desses testes. O processo segue até chegar a um nó terminal, que possuirá um rótulo de categoria, e então designa-se o mesmo à instância tratada [55]. Um exemplo é apresentado na Figura 7, no qual as classes possíveis são A e B e os atributos em questão são altura, comprimento e largura.

Um outro *ensemble method*, que constrói conjuntos usando especificamente árvores de decisão como algoritmo base, é o das florestas aleatórias. Esse método alcança aleatoriedade suficiente para criar árvores bem variadas por meio de dois aspectos principais. O primeiro consiste em gerar um conjunto de treinamento diferente para cada árvore, da mesma forma que o *bagging*. Já o segundo consiste em limitar as opções de uma árvore ao

Figura 7: Exemplo de Árvore de Decisão.



Fonte: Adaptado de [55]

ser construída, de forma que ao se gerar cada nó, a escolha não é feita dentre todos os atributos, mas somente dentre um subconjunto aleatório deles. Essa característica permite lidar facilmente com conjuntos de dados que possuam um grande número de atributos. Em problemas de classificação, da mesma forma que o *bagging*, o resultado final é definido por meio de voto majoritário dentre todos os estimadores [54].

O método das árvores de decisão é bastante simples e com resultados fáceis de se interpretar. Essa característica é perdida ao usá-lo em conjunto com o *bagging*, o *AdaBoost* ou as florestas aleatórias, devido à grande quantidade de árvores que serão geradas, mas se ganha um melhor desempenho em termos de acurácia [66].

4.6 Regressor Logístico

A regressão logística é uma das ferramentas de modelagem mais utilizadas em pesquisas médicas, sendo usada para descrever a relação entre uma saída dicotômica[6] e uma ou mais variáveis de entrada [72]. É particularmente apropriada para modelos que envolvam, por exemplo, o diagnóstico (doente ou saudável) de alguma doença, sendo, portanto, bastante utilizada em estudos na área de saúde [73]. Suas hipóteses são definidas conforme a Equação 8, em que θ é chamada de função logística, sendo definida na Equação 9.

$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) \tag{8}$$

[6]algo dividido em duas partes, em que tudo pertence a uma parte ou a outra, e sendo mutuamente exclusivo

$$\theta(s) = \frac{e^s}{1 + e^s} \tag{9}$$

Formalmente, pode-se definir a função objetivo f da regressão logística como a probabilidade da classe $+1$ ocorrer, dada uma entrada $\mathbf{x}$, conforme pode ser visto na Equação 10. De forma que, se por exemplo, em determinada amostra, a probabilidade de ocorrer a classe $+1$ é de 0.85, então a de ocorrer a classe -1 será de 0.15, de acordo com o que está exposto na Equação 11.

$$f(\mathbf{x}) = P[y = +1 \mid \mathbf{x}] \tag{10}$$

$$P(y \mid \mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{se } y = +1; \\ 1 - f(\mathbf{x}), & \text{se } y = -1. \end{cases} \tag{11}$$

4.7 Programação Genética

4.7.1 Origem e Conceitos Básicos

A teoria da evolução das espécies [74] apresentou o conceito de evolução através da seleção natural, um procedimento pelo qual mutações genéticas aleatórias ocorridas em um indivíduo podem facilitar a sua sobrevivência, o que, consequentemente, aumenta suas chances de gerar descendentes, preservando assim essas mutações para as próximas gerações.

A computação evolutiva é uma área da inteligência computacional composta por algoritmos de otimização inspirados na seleção natural, conhecidos como Algoritmos Evolutivos (AE). Nos AEs, as soluções possíveis de um problema são representadas como indivíduos em uma população. Uma função de aptidão é escolhida para qualificar os indivíduos da geração atual, a fim de reproduzi-los para a próxima geração, de acordo com o princípio de sobrevivência do mais apto.

Programação Genética (PG) [75] é um método utilizado para construir programas, que dependendo da aplicação, podem ser usados como classificadores ou regressores. Em aprendizado de máquina, a PG se enquadra na família dos AEs, sendo cada programa considerado como um indivíduo, cuja aptidão depende da execução desse programa. A

representação mais comum para um indivíduo em PG é feita em forma de árvore, de modo que os nós terminais (folhas) representem as variáveis e parâmetros utilizados, e os nós internos as funções que operam as folhas. Entretanto, outras formas de representação têm se tornado populares, tais como grafos, listas e gramáticas [76]. Em cada caso, a representação manipulável computacionalmente do programa é chamada de genótipo, e a sua interpretação, mais compreensível ao usuário, é chamada de fenótipo.

De forma sucinta, o processo seguido pela PG consiste em gerar aleatoriamente uma população inicial, e evolui-la através de gerações até que um critério de parada seja atingido, como por exemplo, um indivíduo ótimo foi encontrado ou um número máximo de gerações foi atingido. Cada geração consiste em avaliar a aptidão de cada indivíduo e selecionar probabilisticamente alguns deles para aplicar operadores genéticos gerando descendentes.

A aplicação mais comum da PG em ML é realizar aprendizado supervisionado para regressão simbólica ou classificação (binária ou multiclasse), mas já vem sendo empregada em diversos tipos de problemas, obtendo resultados competitivos com os alcançados pelo homem, como por exemplo, projetar circuitos elétricos, antenas, sistemas mecânicos etc [77].

4.7.2 Programação Genética Linear com Inspiração Quântica

4.7.2.1 Conceitos Básicos de Computação Quântica

A computação quântica é uma área de estudos que se baseia em características da mecânica quântica, tais como incerteza, interferência e superposição, a fim de usar métodos basicamente diferentes dos convencionais para processar informações.

Na computação, a menor unidade de informação é o *bit*, que possui dois valores possíveis: 0 ou 1. O seu equivalente na computação quântica é chamado de *qubit*, que pode estar no estado '1', no '0', ou em uma superposição dos dois [78]. Ao ser observado, o *qubit* colapsa para somente um estado clássico. Sua representação é dada por

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

em que $|\psi\rangle$ é o estado do *qubit* e α e β especificam as amplitudes de probabilidade dos estados correspondentes. $|\alpha|^2$ dá a probabilidade de que o *qubit* seja encontrado no estado '0' e $|\beta|^2$ dá a probabilidade de que o *qubit* seja encontrado no estado '1'.

Generalizando, num sistema quântico de d níveis a unidade básica de informação é o *qudit*, que pode assumir qualquer um dos d estados, ou uma superposição deles. A representação desse *qudit* é dada por

$$|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$$

em que $|\psi\rangle$ é o estado do *qudit* e $|\alpha_i|^2$ dá a probabilidade de que o ele seja encontrado no estado i.

4.7.2.2 Descrição Geral

Dada a dificuldade de se desenvolver o *hardware* necessário para computadores quânticos, foi criada a ideia de computação com inspiração quântica [79], em que são criados algoritmos clássicos, aproveitando as características da mecânica quântica, citadas na Seção 4.7.2.1. Um exemplo é a Programação Genética Linear com Inspiração Quântica (PGLIQ) [80], que aplicou o paradigma de inspiração quântica à PG, para aprendizado supervisionado.

A PGLIQ evolui programas para a plataforma Intel x86, utilizando instruções da sua unidade de ponto flutuante, que possui oito registradores (ST(i), em que $i = 0, 1, ..., 7$). Além dos registradores, também podem ser usados dados da memória principal (m).

Na representação da PGLIQ há um "indivíduo quântico", cujo conceito engloba todos os programas possíveis no espaço de busca predefinido. Esse indivíduo quântico, ao ser observado, colapsa para um "indivíduo clássico", representado por algum programa em particular.

Nesse modelo, cada função é representada por um "token de função" (TF). A Tabela 3 exibe um exemplo de um conjunto de funções predefinido para algum problema específico, com seus respectivos TFs.

Tabela 3: Exemplo de conjunto de funções.

Instrução	**Descrição**	**Token de função**
NOP	Nenhuma operação	0
FADD m	$ST(0) \leftarrow ST(0) + m$	1
FSUB ST(0), ST(i)	$ST(0) \leftarrow ST(0) - ST(i)$	2
FMUL m	$ST(0) \leftarrow ST(0) \times m$	3
FXCH ST(i)	$ST(0) \rightleftarrows ST(i)$	4

Percebe-se que os argumentos das funções são somente m, i ou nenhum. Assim, como as funções contém no máximo um terminal, é necessário apenas um "token de terminal" (TT). No caso do TF ser 0 (instrução 'NOP'), o TT será simplesmente ignorado pelo modelo.

Portanto, na PGLIQ cada "gene clássico" é composto por dois tokens, sendo um TF e um TT, e consequentemente são necessários $(L \times 2)$ tokens para compor um indivíduo clássico, sendo L o tamanho máximo predefinido pelo usuário para um programa. Formalmente, todos os programas possuem o mesmo tamanho L, entretanto, devido à disponibilidade da operação 'NOP', o comprimento efetivo de um programa é variável. Além disso, é possível que determinadas linhas, chamadas de introns, não influenciem no resultado, o que também diminui o tamanho efetivo de um programa.

A PGLIQ usa o *qudit* como unidade básica de informação. Assim, d representa a cardinalidade do *token* que terá seu valor definido ao se observar o *qudit* referente.

Cada "gene quântico" é formado por um *qudit* de função (QF) e dois *qudits* de terminal, dado que existem dois tipos de terminais. O QF é constituído pela superposição das funções predefinidas por um conjunto de funções. O primeiro *qudit* de terminal (QT_{Reg}) é formado pela superposição de todos os índices i, referentes aos registradores 'ST(i)'. E o segundo (QT_{mem}) é formado pela superposição de todas as posições de memória possíveis de serem escolhidas (informação predefinida pelo usuário).

O processo de formação de um gene clássico a partir da observação de um gene quântico é feito em três passos. Primeiro, observa-se QF e o seu valor é conferido ao TF. A seguir, de acordo com o valor de TF, é definido se será observado QT_{Reg} ou QT_{mem}. Por fim, é observado o QT definido, e o seu valor é conferido ao TT.

Para um exemplo que tenha como conjunto de funções o conteúdo da Tabela 3, suponha-se que o valor observado no QF foi '2'. É atribuído esse valor ao TF, e será observado, portanto, QT_{Reg}. Se for observado '1', esse valor será conferido ao TT, e à vista disso o gene clássico será (2, 1), que se refere à instrução 'FSUB ST(0), ST(1)'.

A avaliação de um indivíduo clássico começa ao se gerar a sequência de instruções equivalentes aos genes clássicos. O programa resultante é executado usando dados de entrada fornecidos pelo usuário. Como o aprendizado é supervisionado, os dados de saída são comparados com os valores esperados, e a aptidão do indivíduo é medida empregando alguma função predefinida.

A evolução de uma população na PGLIQ não faz uso de mutação ou cruzamento. É utilizado um operador quântico que atua direto nas probabilidades de um *qudit*. Esse operador, nomeado "operador P", inicialmente incrementa uma certa probabilidade do *qudit*. A seguir, normaliza as demais probabilidades, de forma a manter o somatório igual a 1.

O incremento é feito da seguinte forma

$$p_i \leftarrow p_i + s(1-p_i),\ 0 < s < 1$$

em que s é o passo de incremento. Ao se multiplicar esse passo por $(1-p_i)$, tem-se que o tamanho efetivo do incremento será maior quando p_i for pequena, e menor quando for grande. Desta forma, garante-se que p_i nunca atinja 1.

Apesar de conceitualmente ser permitido ao valor de s estar próximo de 1, na prática é necessário que o seu valor seja bem pequeno, próximo a 0. Caso contrário, o modelo irá convergir para um ótimo local. Com s pequeno, os indivíduos das primeiras gerações serão muito diferentes, garantindo a diversidade da população. Entretanto, isso também faz com que a convergência ocorra somente após muitas gerações, quando os melhores indivíduos passam a mudar muito pouco. Com isso, verifica-se que na PGLIQ é fundamental um s muito pequeno e um número de gerações muito grande. Portanto, para não desencadear num gasto computacional excessivo, são usadas populações bem pequenas.

Estruturalmente, a PGLIQ possui uma população clássica C e outra quântica Q, ambas com M indivíduos cada. Há ainda M indivíduos clássicos auxiliares C^{obs}, resultantes da observação dos M indivíduos quânticos no início de cada geração. Após essa observação inicial, a próxima etapa da geração consiste em ordenar conjuntamente os indivíduos da população clássica com os recentemente observados, baseando-se nos seus valores de aptidão. Dessa forma, os M melhores dentre os $2M$ permanecem na população clássica, ordenados a partir da melhor aptidão. A seguir, o operador P é aplicado nos indivíduos Q_k, tomando por base os indivíduos C_k, em que $1 \leq k \leq M$. Com isso, aumenta-se a probabilidade de que os próximos indivíduos observados sejam parecidos com os melhores até então, e a evolução ocorre. Vale salientar que a cada geração, é possível que indivíduos que já estavam dentre os melhores na geração anterior permaneçam na mesma ou em uma posição diferente. No caso de estarem em outra posição, vão influ-

enciar indivíduos quânticos diferentes, através do operador P. Dessa forma, estará sendo usado o conceito de interferência da mecânica quântica.

Para evitar problemas com sobreajuste (do inglês *overfitting*), os dados de entrada geralmente são separados em um conjunto de treinamento e outro de validação. A evolução é conduzida pelo conjunto de treinamento. Uma cópia do melhor indivíduo até então é guardada em C_m após verificar com os dois conjuntos se ele é mesmo o melhor. Inicialmente, é atribuído o valor infinito à aptidão de C_m nos conjuntos de treinamento e validação. Com o passar das gerações, sempre que um indivíduo da geração atual tiver uma aptidão melhor do que C_m, será calculada a sua aptidão no conjunto de validação. Caso essa aptidão supere a de C_m no mesmo conjunto, C_m será substituído por esse indivíduo, que passa a ser o melhor até então. Além disso, também é usual reservar um conjunto de teste, que não será utilizado em momento algum durante a evolução. Só será utilizado após o término, a fim de verificar a capacidade de generalização do programa final.

A Tabela 4 resume todos os processos descritos. Tem-se que g é a geração atual e G é o número máximo de gerações que serão executadas até que a aptidão do programa seja zero, considerando que a função de aptidão foi predefinida para minimização, como por exemplo, o erro médio quadrático.

4.7.3 Programação Genética Orientada à Gramática

Uma gramática pode ser entendida como um conjunto de regras que regula a criação de uma estrutura complexa com base em pequenas partes. Uma sentença gramatical é gerada a partir de uma sequência de escolhas feitas usando as regras de uma gramática. Se as regras forem alteradas, as sentenças derivadas também podem mudar. Assim, usar uma gramática diferente pode fazer com que um modelo apresente resultados muito diferentes, embora os fundamentos sejam os mesmos. Essa flexibilidade permite aplicar as gramáticas a um grande número de problemas, tornando-as muito úteis como parte de uma ferramenta de computação evolucionária [81].

O método de programação genética orientada à gramática mais amplamente utilizado é a Evolução Gramatical (do inglês *Grammatical Evolution*, ou GE) [82]. No princípio da pesquisa em PG, as gramáticas faziam parte de um pequeno segmento, mas o seu papel se expandiu no decorrer dos anos, de forma que a GE é agora um dos métodos de PG

Tabela 4: Algoritmo Evolutivo da PGLIQ.

Algoritmo 1: Pseudocódigo

```
1  para i ← 1 até M faça
2  |  Inicia indivíduo Q_i
3  |  Inicia indivíduo C_i a partir da observação de Q_i
4  |  Avalia C_i no conjunto de treinamento
5  fim
6  C_m.aptidao_Tr ← ∞
7  C_m.aptidao_V ← ∞
8  g ← 1
9  enquanto g < G e (C_m.aptidao_Tr > 0 ou C_m.aptidao_V > 0) faça
10 |  para i ← 1 até M faça
11 |  |  Atualiza indivíduo C_i^obs a partir da observação de Q_i
12 |  |  Avalia C_i^obs no conjunto de treinamento
13 |  fim
14 |  Ordena os indivíduos C em conjunto com os C^obs
15 |  para i ← 1 até M faça
16 |  |  Aplica operador P em Q_i baseando-se em C_i
17 |  fim
18 |  para i ← 1 até M faça
19 |  |  se C_i.aptidao_Tr < C_m.aptidao_Tr então
20 |  |  |  Avalia C_i no conjunto de validação
21 |  |  |  se C_i.aptidao_V < C_m.aptidao_V então
22 |  |  |  |  Substitui C_m por C_i
23 |  |  |  fim
24 |  |  fim
25 |  fim
26 |  g ← g + 1
27 fim
```

Fonte: Adaptado de [80].

mais aplicados [83].

O formalismo de Backus-Naur é uma notação usada para caracterizar gramaticalmente uma linguagem. Nessa notação, uma gramática é representada conforme a tupla $\{N, T, P, S\}$, em que N e T são conjuntos de não-terminais e terminais, respectivamente, P é um conjunto de regras e S é um símbolo inicial, sendo $S \in N$. Em resumo, símbolos terminais são aqueles que não podem ser substituídos, e símbolos não-terminais são os que devem ser substituídos por outros símbolos, terminais ou não. Cada regra é composta por um não-terminal, seguido pelo símbolo `::=`, sucedido por uma lista de escolhas separadas pelo símbolo |. Essas escolhas podem ser constituídas por qualquer combinação de terminais ou não-terminais. Um exemplo de gramática é apresentado na Figura 8, que possui

um conjunto de quatro regras, a primeira com três opções de escolha, a segunda com quatro, a terceira com uma e a quarta com três. Um fato importante a ser observado é que a primeira regra, que apresenta as alternativas para substituir o não-terminal `<expr>`, contém este mesmo não-terminal em duas opções, evidenciando um poderoso aspecto da GE que é a recursividade [84].

Figura 8: Exemplo de conjunto de regras.

```
N = {expr, op, pre-op}
T = {+, -, *, /, minimo, maximo, X}
S = {<expr>}
P =
(I)   <expr> ::= <expr> <op> <expr>          (0)
               | <pre-op> ( <expr> )         (1)
               | <var>                       (2)

(II)  <op> ::= +                             (0)
             | -                             (1)
             | *                             (2)
             | /                             (3)

(III) <pre-op> ::= sqrt                      (0)

(IV)  <var> ::= X1                           (0)
              | X2                           (1)
              | X3                           (2)
```

Fonte: Adaptado de [82].

Na GE, o cromossomo tem uma quantidade de genes que pode variar no decorrer da evolução por meio de operações de cruzamento. Cada gene é representado por um número usualmente inteiro, sendo o genótipo composto por uma cadeia linear deles. Já o fenótipo é retratado por uma expressão que pode ser executada diretamente para se verificar o resultado do indivíduo. Cabe citar também que uma operação de mutação consiste em se alterar aleatoriamente o valor de um gene.

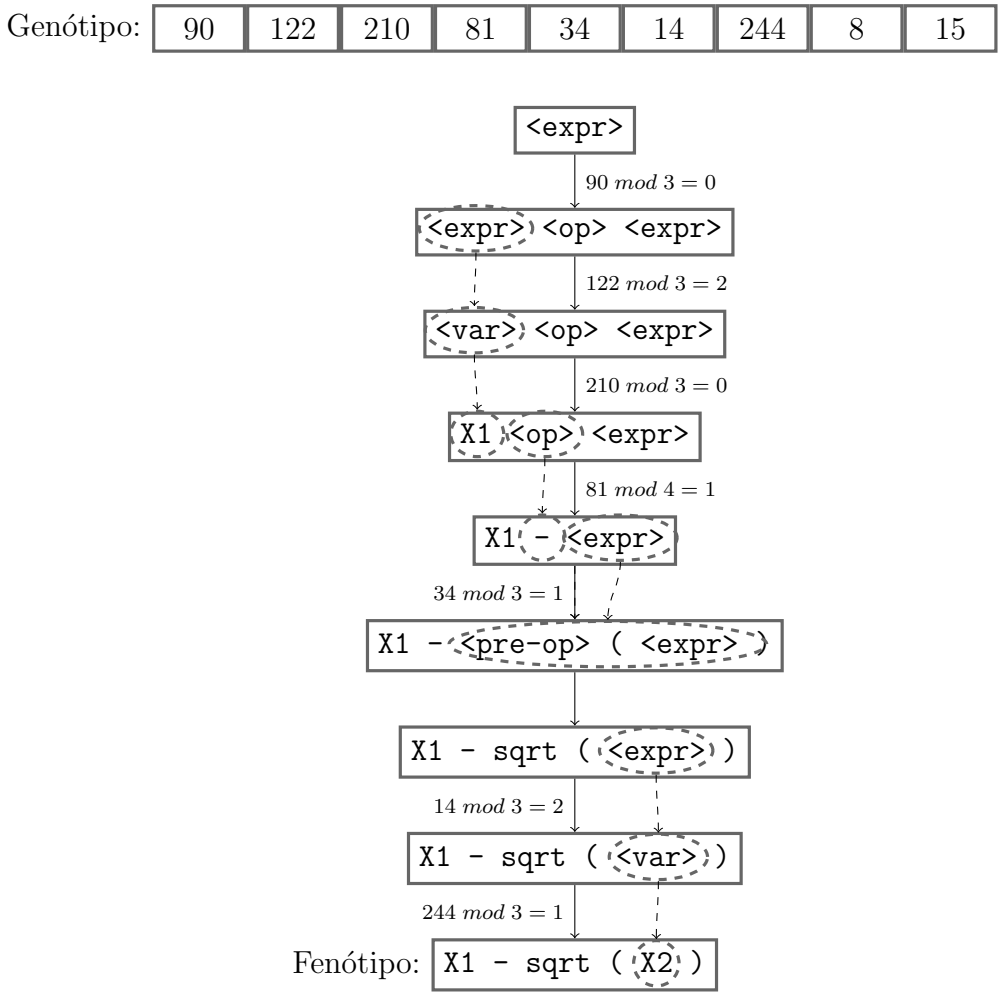
O mapeamento genótipo-fenótipo é feito com o operador módulo[7], de forma a identificar a opção a ser escolhida em cada regra gramatical, tal como a Equação 12, em que g é o valor inteiro do gene, r o número de opções da regra e c a escolha a ser feita.

$$c = g \bmod r \tag{12}$$

[7]retorna o resto da divisão inteira

A Figura 9 apresenta como exemplo um cromossomo com nove genes, cada um contendo um valor inteiro de no máximo 8 bits. É apresentado cada passo do seu respectivo mapeamento genótipo-fenótipo, baseado na gramática da Figura 8. Desta forma, partindo-se do símbolo inicial `<expr>`, tem-se que a regra referente a esse símbolo é a regra I, que possui três opções. Sendo o primeiro gene igual a 90, o resultado de 90 *mod* 3 é zero, o que leva a se escolher a opção (0), que é `<expr> <op> <expr>`. Assim, segue-se até se obter uma expressão composta somente de terminais, o que foi atingido antes mesmo de se utilizar todos os nove genes. Esse exemplo também salienta que a regra III substitui `<pre-op>` por `sqrt`, sem utilizar nenhum gene, pois se trata da única opção dessa regra.

Figura 9: Exemplo de mapeamento genótipo-fenótipo em GE.



Durante um mapeamento, caso ainda haja não-terminais após a utilização do último gene, a operação de *wrapping* pode ser aplicada. Essa operação consiste em prosseguir com o mapeamento, reutilizando os genes, a partir do primeiro, até que uma expressão fechada seja obtida. Entretanto, devido à recursividade é necessário limitar o número de

vezes que isso pode ocorrer. Por exemplo, partindo de `<expr>`, se o resultado do módulo der sempre zero, esse símbolo será seguidamente substituído por `<expr> <op> <expr>`, apenas aumentando o tamanho da expressão, sem nunca fechá-la. Estabelecendo um limite, ao atingi-lo o indivíduo será considerado inválido.

4.8 Árvores de Padrões *Fuzzy*

4.8.1 Conceitos Básicos de Lógica *Fuzzy*

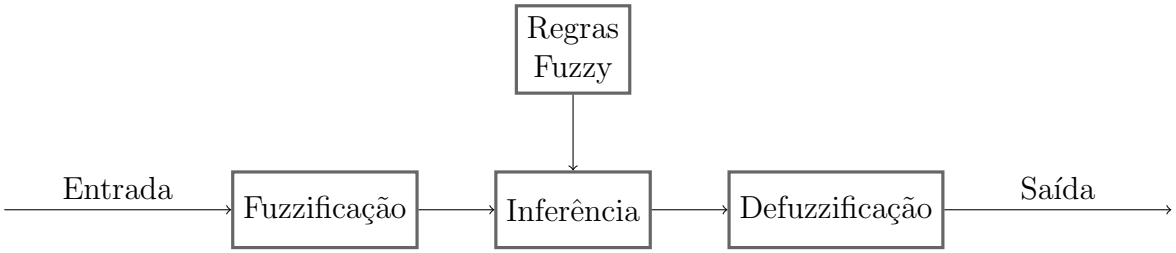
Diferentemente da lógica clássica, que só permite modos de raciocínio binários, como por exemplo, 0 ou 1, verdadeiro ou falso, alto ou baixo etc, a lógica *fuzzy* possibilita modos de raciocínio aproximados, isto é, inúmeros valores entre 0 e 1. A representação linguística também é aproximada, como, por exemplo, muito verdade, pouco falso, meio verdade etc [85].

Além disso, na lógica *fuzzy* um mesmo elemento pode participar de mais de um conjunto. Assim, por exemplo, se na lógica clássica se estabelece que uma pessoa é considerada de meia-idade se tiver entre 35 e 55 anos, alguém com 34 anos vai ser considerado totalmente fora deste conjunto. Já na lógica *fuzzy*, essa mesma pessoa pode ser considerada, por exemplo, 10 % jovem e 80 % de meia-idade. Para definir tais fatores, usa-se o grau de pertinência, que é representado graficamente por sua respectiva função de pertinência, cuja ordenada varia entre os valores lógicos de 0 a 1. Em um sistema *fuzzy*, todas as variáveis (de entrada e de saída) são representadas por suas correspondentes funções de pertinência.

Após definir todas as variáveis de interesse e estipular suas respectivas funções de pertinência, estabelecem-se regras linguísticas do tipo "se x é A então y é B", que expressam a influência das variáveis de entrada nas variáveis de saída como, por exemplo, "se *distribuição de chuva* é boa, então *variação da produção* é zero" [86].

Ao fim, pode-se elaborar um Sistema *Fuzzy* Baseado em Regras (SFBR), que é composto de três partes, conforme pode ser visto na Figura 10. As variáveis de entrada passam pela fuzzificação, sendo convertidas para seus respectivos graus de pertinência nos conjuntos *fuzzy*. Na inferência, as regras *fuzzy* são aplicadas nas variáveis de entrada fuzzificadas, gerando variáveis de saída também fuzzificadas, que vão passar pela defuzzificação.

Figura 10: Diagrama de um Sistema *Fuzzy* Baseado em Regras.



4.8.2 Descrição Geral

As Árvores de Padrões *Fuzzy* (APFs) foram criadas para evitar o aumento exponencial de regras em SFBRs, conforme se aumenta o número de entradas. Essas árvores possuem em seus nós internos diferentes operadores *fuzzy*, e em seus nós terminais os atributos de entrada fuzzificados. Tal como árvores de decisão, APFs são usadas em problemas de classificação [87].

Tabela 5: Operadores de agregação.

t-norm		**t-conorm**	
Mínimo	$\min(a, b)$	Máximo	$\max(a, b)$
Algébrico	$a \times b$	Algébrico	$a + b - a \times b$
Lukasiewicz	$\max(a - 1 + b, 0)$	Lukasiewicz	$\min(a + b, 1)$
Einstein	$\frac{a \times b}{2-(a+b-a \times b)}$	Einstein	$\frac{a+b}{1+a \times b)}$

Diversos tipos de operadores podem ser usados em APFs. Um deles é o que abrange operadores de agregação, que podem ser *t-norms* ou *t-conorms*. O primeiro envolve operadores baseados no conectivo lógico AND, e o segundo aqueles baseados no conectivo OR. Ambos podem ser vistos na Tabela 5. Outro tipo de operadores são os de média, como o WA (do inglês *weighted average*) e o OWA(do inglês *ordered weighted average*), cujas expressões podem ser vistas na Tabela 6. Em ambas as tabelas, a e b são as entradas dos operadores, e $0 < \alpha < 1$.

Tabela 6: Operadores de média.

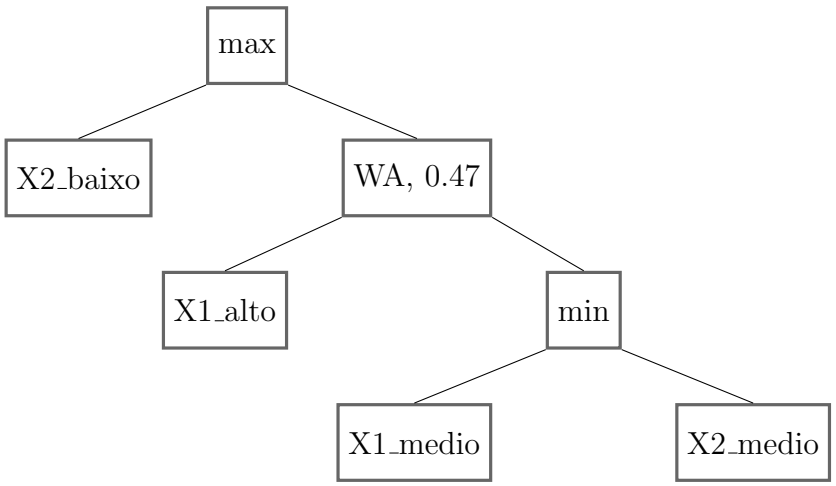
Representação	**Significado**	**Expressão**
$\text{WA}(a, b, \alpha)$	Média ponderada	$\alpha \times a + (1 - \alpha) \times b$
$\text{OWA}(a, b, \alpha)$	Média ponderada ordenada	$\alpha \times \max(a, b) + (1 - \alpha) \times \min(a, b)$

Há também operadores de concentração e de diluição, que recebem somente uma

entrada, respectivamente reduzindo e aumentando o seu grau de pertinência. O concentrador mais simples resulta no quadrado do valor de entrada, enquanto que o diluidor mais simples resulta na raiz quadrada da entrada.

A Figura 11 mostra um exemplo de APF para um problema com dois atributos ($X1$ e $X2$) que foram fuzzificados em três conjuntos (alto, médio e baixo) cada um. Ao se atribuir dados aos terminais, o valor resultante dessa árvore representa o grau de pertinência desses dados a uma determinada classe do problema. Assim, quanto mais próximo o valor estiver de 1, mais provável é que esses dados pertençam à classe, e quanto mais próximo de 0, mais improvável. Para problemas de classificação binária, é possível usar somente uma árvore para classificar os dados. Assim, quanto mais próximo o valor que resultar da árvore estiver de 1, mais provável é que pertença a uma certa classe, e quanto mais próximo de zero mais provável que pertença à outra classe. É usual estabelecer uma fronteira de decisão em 0.5.

Figura 11: Exemplo de Árvore de Padrões *Fuzzy*.



Generalizando para problemas multiclasse, é criada uma árvore para cada classe, e a classificação de uma amostra é feita atribuindo-se a ela a classe na qual tiver um valor de pertinência maior na sua respectiva árvore.

4.9 Redução da Dimensionalidade

Em aprendizado de máquina há um conceito chamado de maldição da dimensionalidade que se refere ao fato de que, usualmente, conforme se aumenta o número de dimensões da entrada, mais amostras são necessárias para garantir que o algoritmo seja

capaz de generalizar [54]. Se não há a possibilidade de se obter uma maior quantidade de dados, ou se ao fazê-lo incorrer-se-ia em gastos significantes, uma etapa pode ser executada antes do treinamento do modelo com o objetivo de se reduzir a dimensão do problema.

Algo que pode ser feito nesta etapa prévia é a seleção de atributos, por meio de métodos automáticos, conforme algumas opções que serão apresentadas na seção seguinte.Outra possibilidade consiste em se construir novos atributos a partir da combinação dos atributos originais, algo que pode ser entendido como uma transformação do espaço original de atributos em outro que o algoritmo possa explorar melhor [88].

4.9.1 Seleção de Atributos

A seleção de atributos é uma metodologia pela qual é selecionado um subconjunto de atributos, de modo que o espaço de atributos é reduzido, sendo esse subconjunto capaz de representar a informação contida no conjunto original, de acordo com algum critério pré-determinado, como por exemplo a acurácia na classificação [89].

As estratégias de seleção essencialmente se dividem em filtros, *wrappers* e métodos embutidos (do inglês *embedded methods*) [90]. Na abordagem por filtros os atributos são avaliados e ordenados a partir de características próprias dos dados, tais como coeficientes de correlação e testes estatísticos clássicos (teste T, teste F, teste qui-quadrado de Pearson, etc) [19]. Enquanto que no tipo *wrapper*, o próprio classificador é usado para avaliar um subconjunto, como por exemplo, fornecendo a medida de acurácia no conjunto de treinamento ao se utilizar somente os atributos presentes em determinado subconjunto [90]. Em geral, a abordagem por *wrappers* alcança melhores resultados que a por filtros porque otimiza a própria medida de desempenho do aprendizado ao remover atributos, entretanto o tempo gasto na etapa de seleção é muito maior que o dos filtros [89]. Por fim, nos métodos embutidos a seleção de atributos é feita durante o próprio processo de treinamento do modelo, sendo uma abordagem específica de certos algoritmos de aprendizado. Um exemplo é o das árvores de decisão, que são construídas de forma que, ao se gerar cada nó, a escolha é feita em um subconjunto de atributos.

A metodologia *wrapper* oferece uma maneira simples e poderosa para selecionar atributos, independentemente do algoritmo de aprendizado utilizado. Entretanto, para evitar alto gasto computacional devem ser utilizadas estratégias eficientes para buscar os subconjuntos que serão avaliados [90]. Isso porque, sendo n o número de atributos no

conjunto original, o espaço de busca por um subconjunto apropriado tem tamanho 2^n. Assim, conforme se aumenta o valor de n, é inviabilizada uma busca exaustiva, e outros métodos de busca devem ser utilizados, tais como os sequenciais e as metaheurísticas.

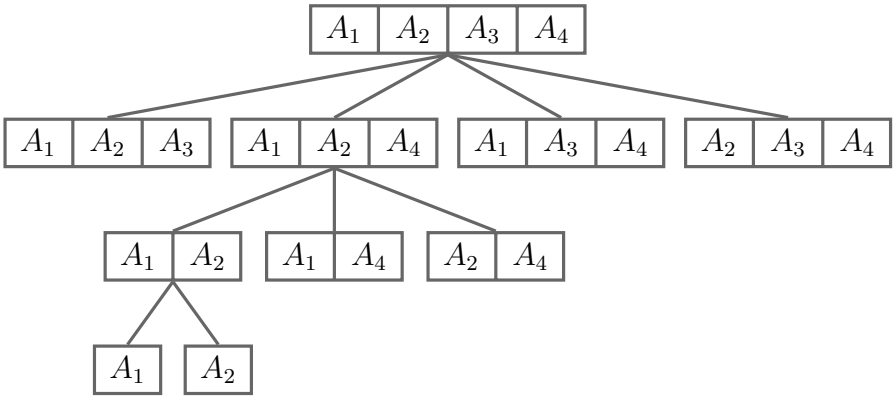
Os métodos de busca sequencial são bastante simples. No método *backward*, a busca pelo subconjunto ótimo começa com todos os atributos, e a cada iteração é eliminado aquele cuja retirada apresentar a menor perda de informação. Já o método *forward* inicia a busca verificando qual o melhor atributo sozinho, e a cada iteração acrescenta aquele que apresentar o maior ganho de informação. Ambos os algoritmos de busca são gulosos, ou seja, podem ficar presos em ótimos locais, mas podem diminuir problemas com *overfitting* [90]. O método *backward* pode ser útil para manter atributos de baixa importância isolada, mas que auxiliem no desempenho do modelo, caso estejam combinados com outros atributos. Enquanto que o *forward* pode ser predisposto a manter no subconjunto final um atributo que tenha um bom desempenho isolado, mas que não acrescente muita informação ao ser combinado com outros [91].

A Figura 12 esquematiza como é feita a busca *backward* em um exemplo cujos atributos são A_1, A_2, A_3 e A_4. Inicialmente é medido o desempenho com todos os atributos. A seguir, o comportamento é verificado em quatro subconjuntos distintos, cada um deles sem um atributo diferente. Observa-se que o melhor desempenho ocorre ao se retirar o atributo A_3. Logo após são analisados mais três subconjuntos, cada um retirando um dos três atributos restantes. Nesta iteração, constata-se o melhor desempenho ao ser retirado A_4. Por fim, examina-se A_1 e A_2 separadamente. Nesse exemplo, o espaço de busca pelo melhor subconjunto tem tamanho 10, enquanto que na busca exaustiva teria tamanho 16. Essa diferença se torna muito superior para maiores quantidades de atributos.

4.9.2 Construção de Atributos

A construção de atributos é um processo pelo qual novos atributos são criados e definidos de alguma forma, como por exemplo, expressões aritméticas dos atributos originais. É uma metodologia que pode alterar completamente o espaço de atributos, ao criar novos conjuntos que representarão o problema como um todo, ou simplesmente aumentá-lo, ao usar os novos atributos como complemento aos originais. Neste último caso, pode ser usado para descobrir informações ocultas acerca das relações entre os atributos, enriquecendo o espaço de representação, ao contrário da seleção que geralmente

Figura 12: Esquema de busca *backward* em exemplo com quatro atributos.



Fonte: Adaptado de [91].

é usada para simplificar a representação em problemas que contém mais informação que o necessário. Por outro lado, é possível que alguns recursos construídos não sejam tão úteis, sendo mais indicado removê-los. Assim, é comum ver o uso combinado de da construção e da seleção de atributos [89].

Um método útil para automatizar a construção de atributos é a PG, que é capaz de sintetizar funções sem muitas suposições prévias acerca de sua forma [88]. Sua representação baseada em árvores é uma forma natural de se representar um atributo construído, em que as folhas podem ser ocupadas pelos atributos originais e também por constantes, e os nós internos podem ser, por exemplo, funções aritméticas [92]. Além disso, a PG é capaz de evoluir um atributo para um algoritmo específico medindo diretamente o seu impacto no desempenho desse algoritmo [88].

Há dois contextos principais para se utilizar PG na construção de atributos. No primeiro é construído uma única árvore a qual representa um único atributo que, em geral, costuma não ser suficiente para representar um problema como um todo, exceto em casos muito simples. Assim, usualmente se usa esse novo atributo em conjunto com os originais, aumentando a dimensionalidade dos dados. Vale ressaltar que em problemas que originalmente já possuem uma grande dimensão, aumentar um atributo pode não alterar o desempenho [92]. No segundo contexto, várias árvores de PG são utilizadas para se construir múltiplos atributos, que podem ser utilizados em conjunto com os atributos originais, ou em substituição a eles.

A Tabela 7 exemplifica como é um algoritmo baseado em PG para construir atributos. Neste caso, M é o tamanho da população e G o número de gerações. Cada indivíduo

I_i pode possuir apenas uma árvore, ou seja, construir um único atributo, ou possuir m árvores, sendo capaz de construir m atributos. Por sua vez, a aptidão de cada indivíduo pode ser calculada empregando-se os atributos construídos por esse indivíduo em um outro método, como por exemplo o KNN. Assim, a métrica de desempenho do indivíduo, como por exemplo a AUC, será tomada a partir do resultado desse método, usando-se os atributos construídos.

Tabela 7: Algoritmo evolutivo da construção de atributos com PG.

Algoritmo 2: Pseudocódigo

```
1  para i ← 1 até M faça
2  |  Inicia indivíduo I_i
3  fim
4  melhor_ind ← I_1
5  g ← 1
6  enquanto g < G e (melhor_ind.aptidao_Tr < 1 faça
7  |  para i ← 1 até M faça
8  |  |  Calcula os atributos construídos pelo indivíduo I_i no conjunto de treino
9  |  |  Calcula a aptidão do indivíduo I_i
10 |  |  Atualiza o melhor_ind, se I_i for melhor
11 |  fim
12 |  Seleciona indivíduos usando torneio
13 |  A partir dos selecionados cria novos indivíduos usando mutação ou
   |   cruzamento
14 |  Substitui novos indivíduos pelos antigos na população
15 |  g ← g + 1
16 fim
```

Fonte: Adaptado de [92].

4.10 Aumento de Dados

Conjuntos de dados desbalanceados, isto é, aqueles nos quais as classes não estão representadas em quantidades aproximadamente iguais, podem trazer mais dificuldade à tarefa do aprendizado, porque o modelo pode tender a classificar os dados com a classe majoritária. Uma das formas de se equilibrar um conjunto desbalanceado é utilizando a técnica SMOTE (do inglês *Synthetic Minority Oversampling Technique*), que sintetiza novas amostras com a classe minoritária [93]. Para tal são adicionadas amostras sintéticas entre cada amostra original da classe minoritária e os seus k vizinhos mais próximos da mesma classe. Essas amostras são inseridas nos segmentos de reta que unem duas amos-

tras originais. Um número aleatório entre 0 e 1 é multiplicado pelo comprimento de cada segmento, definindo uma distância da amostra original a que cada amostra sintética será inserida. O valor k é predefinido, entretanto dependendo do tamanho da sobreamostragem, menos vizinhos podem ser necessários, e é aleatória a escolha de quais usar. Por exemplo, se for definido $k = 4$, mas a sobreamostragem for de 300%, então apenas 3 vizinhos de cada amostra serão usados.

Particularmente neste trabalho, como a quantidade dados é muito pequena, o uso do SMOTE na síntese de novas amostras tem por objetivo não somente equilibrar as classes, mas também aumentar a quantidade de dados, alimentando mais os algoritmos. Todavia, cabe salientar que o SMOTE é aplicado somente em dados de treinamento, e assim, as métricas dos resultados, isto é, nos conjuntos de teste, são tomadas somente dos dados originais.

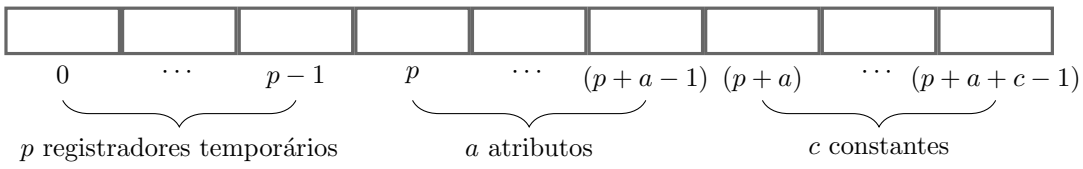
5 PROGRAMAÇÃO GENÉTICA LINEAR COM INSPIRAÇÃO QUÂNTICA APLICADA À SÍNTESE DE ÁRVORES DE PADRÕES *FUZZY*

Este capítulo consiste em apresentar o desenvolvimento de uma aplicação da PGLIQ, retratada na Seção 4.7.2, à síntese de APFs, assunto tratado na Seção 4.8.

5.1 Registradores

A implementação da PGLIQ produzida para este trabalho trabalha com dados armazenados em um vetor, que funciona como um conjunto de registradores. Esse vetor pode ser dividido em três partes diferentes, tal como visto na Figura 13. A primeira consiste em registradores temporários, que sempre recebem zero como valores iniciais a cada execução, e são reservados para se receber resultados parciais, podendo ser alterados conforme se executa cada linha de um programa. Já a segunda e a terceira permanecem inalteradas durante a execução, servindo apenas para fornecer entradas. Na segunda os registradores são inicializados com os valores dos atributos da amostra para a qual se deseja obter uma saída. Por fim, na terceira são registradas as constantes que também poderão servir de entradas para o programa. Sendo p o número de registradores temporários, a o número de atributos e c o número de constantes, o vetor de registradores tem tamanho $(p+a+c)$, sendo inicializado com 0 nas posições 0 a $(p-1)$, com os valores dos atributos nas posições p a $(p+a-1)$ e com os valores das constantes nas posições $(p+a)$ a $(p+a+c-1)$.

Figura 13: Representação do vetor de registradores.



O número de registradores temporários é um hiperparâmetro definido pelo usuário, que está relacionado com a diversidade na população de programas. Como cada programa pode ser interpretado como um árvore, um número maior de registradores temporários permite maior liberdade na criação de ramificações para as árvores. A segunda parte do vetor tem tamanho igual ao número de atributos das amostras, e a terceira tem tamanho

igual ao número de constantes definidas. A quantidade e o valor das contantes é outro hiperparâmetro definido pelo usuário.

Sendo R um vetor de registradores, a Tabela 8 descreve algumas funções que envolvem operadores *fuzzy* e os argumentos necessários para cada uma delas. Esses argumentos definem os índices do vetor R. Assim, o índice i pode assumir qualquer valor entre 0 e $(p-1)$, o índice j pode assumir qualquer valor entre 0 e $(p+a+c-1)$ e o índice k pode assumir qualquer valor entre $(p+a)$ e $(p+a+c-1)$.

Tabela 8: Descrição das funções.

Função	**Descrição**	**Argumentos**
NOP	Nenhuma operação	-
WA(i, j, k)	R(i) ← R(k) × R(i) + (1 - R(k)) × R(j)	i, j, k
OWA(i, j, k)	R(i) ← R(k) × max(R(i),R(j)) + (1 - R(k)) × min(R(i),R(j))	i, j, k
min(i, j)	R(i) ← min(R(i),R(j))	i, j
max(i, j)	R(i) ← max(R(i),R(j))	i, j
dilator(i, j)	R(i) ← $\sqrt{\text{R(j)}}$	i, j
concentrator(i, j)	R(i) ← R(j)^2	i, j

Ao se executar um programa para determinada amostra, a saída referente a essa amostra será o valor obtido na posição $R(0)$ após a execução do programa.

5.2 Representação

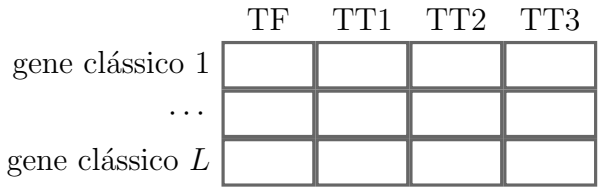
Conforme apresentado na Seção 4.7.2.2, na PGLIQ há um "indivíduo quântico", cujo conceito engloba todos os programas possíveis no espaço de busca, que ao ser observado, colapsa para um "indivíduo clássico", representando algum programa em particular.

5.2.1 Indivíduo Clássico

O indivíduo clássico, isto é um programa observado, pode ser representado como uma matriz, tal como apresentado na Figura 14. Cada linha dessa matriz é um "gene clássico", e corresponde a uma linha do programa. A primeira coluna indica o "token de função" (TF), e as três seguintes indicam os "tokens de terminal" (TTs), cada um caracterizando um dos argumentos das funções, conforme apresentado na Tabela 8, em que cada função tem no máximo três argumentos. Portanto, um indivíduo clássico é representado por uma matriz de tamanho $L \times 4$, em que L é o número de linhas do programa. Essa representação é o genótipo, e o conjunto equivalente de instruções é o

fenótipo, sendo a execução do programa realizada desde a primeira até a última linha. Vale salientar que a disponibilidade do uso da função 'NOP' permite que, apesar de uma população de indivíduos ter o mesmo comprimento L, na prática o comprimento efetivo é diferente.

Figura 14: Representação do indivíduo clássico.



Por exemplo, a Tabela 9 exibe um conjunto de funções predefinido para algum problema específico, com seus respectivos TFs. Para um programa baseado nesse conjunto, nas linhas cuja primeira coluna possua o valor 0, a função respectiva será 'NOP', e as colunas seguintes, ou seja os TTs, serão ignoradas. Da mesma forma, caso o TF seja 3, 4, 5 ou 6, o terceiro TT será ignorado. Somente caso o TF seja 1 ou 2, todos os três TTs serão considerados. Assim, um exemplo desse mapeamento genótipo-fenótipo pode ser visto na Figura 15.

Tabela 9: Conjunto de funções.

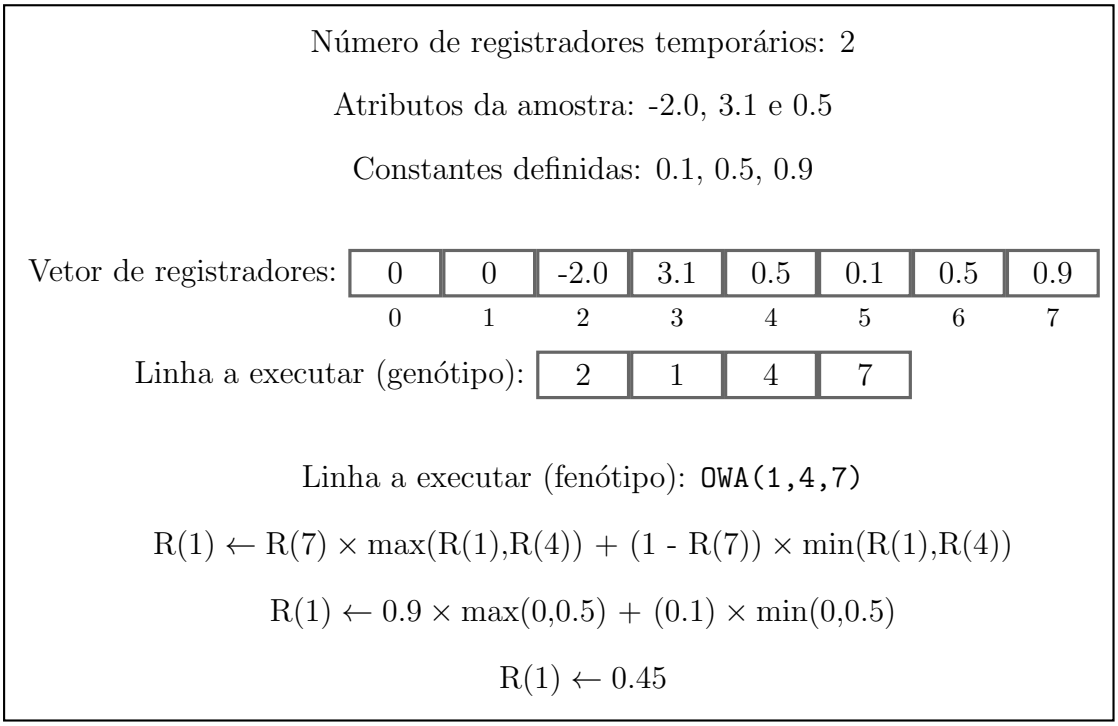
Função	**Token de função**
NOP	0
WA(i, j, k)	1
OWA(i, j, k)	2
min(i, j)	3
max(i, j)	4
dilator(i, j)	5
concentrator(i, j)	6

5.2.2 Indivíduo Quântico

Tal como descrito na Seção 4.7.2.2, a PGLIQ usa o *qudit* como unidade básica de informação, em que d caracteriza a cardinalidade do *token* que terá seu valor definido ao se observar o *qudit* concernente.

Cada "gene quântico" é formado por um *qudit* de função (QF), três *qudits* de terminal (QT), dado que existem três tipos de terminais, e mais um *qudit* extra (QE). Cada gene clássico é observado a partir de um gene quântico diferente, assim para formar um

Figura 15: Exemplo de mapeamento genótipo-fenótipo com execução em PGLIQ.

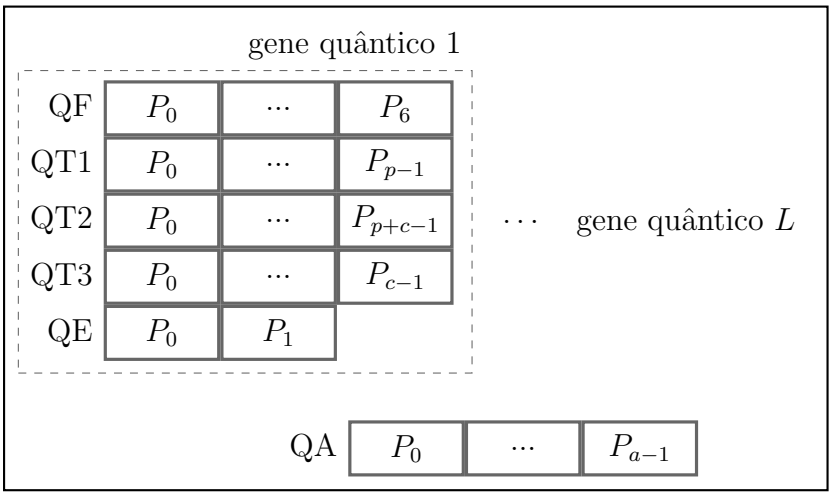


indivíduo clássico com L genes clássicos, é necessário que o indivíduo quântico respectivo possua L genes quânticos. Cada indivíduo quântico possui ainda um *qudit* de atributos (QA) próprio, que funciona como uma implementação de método embutido (do inglês *embedded methods*) de seleção atributos.

A representação de um invidíduo quântico é apresentada na Figura 16. O QF é constituído pela superposição das funções predefinidas por um conjunto de funções. Seguindo o exemplo da Tabela 9, QF é composto por sete probabilidades (P_0 a P_6). O primeiro *qudit* de terminal (QT_1) é formado pela sobreposição de todos os índices i, referentes aos p registradores temporários, ou seja, é composto por p probabilidades (P_0 a P_{p-1}). O segundo *qudit* de terminal (QT_2) é formado pela superposição dos índices referentes aos registradores temporários e às c posições com constantes no vetor de registradores, ou seja, é composto por $(p+c)$ probabilidades (P_0 a P_{p+c-1}). O terceiro *qudit* de terminal (QT_3) é formado pela sobreposição de todos os índices k, referentes às posições com constantes no vetor de registradores, ou seja, é composto por c probabilidades (P_0 a P_{c-1}). O QE é formado pela superposição de apenas dois estados ('0' e '1'), sendo usado na primeira fase do método embutido de seleção de atributos, ou seja, é composto por apenas duas probabilidades (P_0 e P_1). Por fim o QA é formado pela sobreposição dos índices referentes às a posições com os atributos no vetor de registradores, ou seja, é com-

posto por a probabilidades (P_0 a P_{a-1}), e consiste na segunda fase do método embutido de seleção de atributos.

Figura 16: Representação do indivíduo quântico.



A observação de QE define como será observado o TT_2. Caso QE observe o estado '0', o TT_2 será observado a partir de QA, selecionando um atributo, ou será observado a partir de QT_2, selecionando um registrador temporário ou uma constante, caso QE observe o estado '1'. Assim funciona o método embutido de seleção de atributos, pois tal como visto na Figura 16, QA é único para todo o indivíduo quântico, diferentemente dos demais *qudits*, que são distintos para cada gene. Dessa forma, todos os atributos de um indivíduo clássico são observados a partir da distribuição de probabilidades presente em QA.

O método de composição de um gene clássico com base na observação de um gene quântico, considerando o conjunto de funções da Tabela 9 pode ser resumido como a seguir:

1. Observa-se QF, atribuindo-se o seu resultado ao TF. Caso seja zero, finaliza-se a formação do gene clássico;

2. Observa-se QT_1, atribuindo-se o seu resultado ao TT_1;

3. Observa-se QE. Caso seja '0', observa-se QA, atribuindo-se o seu resultado ao TT_2, e caso seja '1', o mesmo será atribuído a partir da observação de QT_2;

4. Caso TF seja 1 ou 2, observa-se QT_3, atribuindo-se o seu resultado ao TT_3.

5.3 Aptidão

Dado um conjunto de dados, para cada amostra um vetor de registradores será inicializado e o indivíduo clássico será percorrido, linha por linha, executando-se suas respectivas instruções, de forma que ao final do programa a saída referente à cada amostra será tomada na posição 0 do vetor. Assim, ao se completar essa tarefa, tem-se um conjunto de saídas, cada uma referente a uma amostra do conjunto de dados, e pode-se calcular uma métrica de desempenho em comparação com as etiquetas dos dados. Usualmente foi calculada a AUC, que é a grandeza que se deseja maximizar neste trabalho, mas outras métricas podem ser utilizadas.

Para amostras provenientes de conjuntos *fuzzy* e funções que envolvem operadores *fuzzy*, o resultado final será sempre um valor entre 0 e 1. Em problemas de classificação binária, esse resultado pode ser interpretado como a probabilidade da classe de saída ser '1', e usado diretamente na confecção da curva ROC, a fim de calcular a AUC. Já para amostras normalizadas com média 0 e desvio-padrão 1 e funções aritméticas, o resultado pode ser positivo ou negativo e ter módulo maior que 1. Assim, faz-se necessário usar alguma transformação para se obter a probabilidade, como por exemplo a função sigmoide ou a função *softmax*.

5.4 Atualização dos *qudits*

Cada *qudit* se trata de um conjunto de probabilidades p_i, cuja soma é igual a 1. A evolução de uma população na PGLIQ não se usa de mutação ou cruzamento, como outras formas de PG, mas se dá a partir da atualização dos *qudits*. Isso é feito com um operador quântico, denominado "operador P", que consiste em se incrementar uma determinada probabilidade, e ajustar todas as demais, de forma a se manter o critério de soma igual a 1.

Uma determinada probabilidade p_i é incrementada conforme a Equação 13, em que s é o passo de incremento, sendo $0 < s < 1$. Essa equação permite que probabilidades pequenas tenham um incremento maior do que as grandes.

$$p_i \leftarrow p_i + s(1-p_i) \tag{13}$$

5.5 População

Estruturalmente, a PGLIQ possui uma população clássica C e outra quântica Q, ambas com M indivíduos cada. Outros M indivíduos clássicos C^{obs} são resultantes da observação dos M indivíduos quânticos no início de cada geração. Após essa observação inicial, é feita uma ordenação conjunta dos indivíduos da população clássica com os recentemente observados, de acordo com seus valores de aptidão. Assim, os M melhores permanecem na população clássica e os M piores serão substituídos na geração seguinte. Então, o operador P é aplicado nos indivíduos Q_k, tomando por base os indivíduos C_k, em que $1 \leq k \leq M$. Assim, aumenta-se a probabilidade de que os próximos indivíduos observados sejam parecidos com os melhores até então, e a evolução ocorre. Para que o modelo não convirja rapidamente ficando preso em um ótimo local, é necessário que o passo de incremento seja bem pequeno.

5.6 Evolução

O algoritmo evolutivo implementado neste trabalho pode ser visto na Tabela 10, e segue basicamente o pseudocódigo já apresentado na Tabela 4, com as exceções de que não há uma divisão em conjuntos de treinamento e de validação, além de buscar maximizar a medida de desempenho.

Para evitar sobreajuste (do inglês *overfitting*), é essencial que haja uma validação interna no decorrer da evolução, no entanto como os dados disponíveis para este trabalho são em pequena quantidade, optou-se por usar a validação cruzada (do inglês *cross-validation*), conforme será descrito nas Seção 5.6.5.

5.6.1 Inicialização da População Quântica

Nesta etapa, cada *qudit* de cada um dos L genes quânticos de cada um dos M indivíduos quânticos é inicializado. Também o é o *qudit* de atributos (QA) de cada indivíduo quântico. Em geral, essa inicialização é feita atribuindo-se um mesmo valor $1/d$ a todas as probabilidades p_i de um dado *qudit*, em que d é a sua cardinalidade. Assim, por exemplo, em um problema com dez atributos, as probabilidades em cada QA serão inicializadas com 1/10. A única exceção ocorre com os QFs, em que a probabilidade referente à função 'NOP' é inicializada usualmente com um valor muito maior que as

Tabela 10: Algoritmo Evolutivo da PGLIQ implementada.

Algoritmo 3: Pseudocódigo

```
1  para i ← 1 até M faça
2  |  Inicia indivíduo Q_i
3  |  Inicia indivíduo C_i a partir da observação de Q_i
4  |  Avalia C_i nas pastas de treinamento
5  fim
6  C_m.aptidao_Tr ← 0
7  C_m.aptidao_V ← 0
8  g ← 1
9  enquanto g < G e C_m.aptidao_Tr < 1 e C_m.aptidao_V < 1 faça
10 |  para i ← 1 até M faça
11 |  |  Atualiza indivíduo C_i^obs a partir da observação de Q_i
12 |  |  Avalia C_i^obs nas pastas de treinamento
13 |  fim
14 |  Ordena os indivíduos C em conjunto com os C^obs
15 |  para i ← 1 até M faça
16 |  |  Aplica operador P em Q_i baseando-se em C_i
17 |  fim
18 |  para i ← 1 até M faça
19 |  |  se C_i.aptidao_Tr > C_m.aptidao_Tr então
20 |  |  |  Avalia C_i nas pastas de validação
21 |  |  |  se C_i.aptidao_V > C_m.aptidao_V então
22 |  |  |  |  Substitui C_m por C_i
23 |  |  |  fim
24 |  |  fim
25 |  fim
26 |  g ← g + 1
27 fim
```

Fonte: Adaptado de [80].

demais, por exemplo 0.9, o que levaria as demais probabilidades a serem inicializadas com $0.1/(d-1)$. Isso é feito para que nas gerações iniciais, os programas tenham um tamanho efetivo bem menor que o comprimento máximo, o que permite um maior controle da evolução. Tal procedimento emula o que é feito na PG clássica, em que geralmente as árvores nas primeiras gerações são bem menores que no final.

5.6.2 Inicialização da População Clássica

Após a inicialização da população quântica, cada um dos L genes clássicos de cada um dos M indivíduos clássicos C_i $(1 < i < M)$ será inicializado a partir da observação

do seu respectivo gene quântico no referente indivíduo quântico Q_i, conforme descrito na Seção 5.2.2. Da mesma forma, a partir da geração 2, cada gene clássico de cada indivíduo clássico C_i^{obs} será observado.

5.6.3 Execução das Gerações

Após a inicialização das populações, o algoritmo inicia a execução das gerações. Essa execução prosseguirá até se atinja um número máximo de gerações, pré-definido pelo usuário, ou que o desempenho do melhor indivíduo atinja o valor máximo considerando todo o conjunto de treinamento.

A cada geração, após a observação da população C_i^{obs}, seus desempenhos serão avaliados. Caso dois ou mais indivíduos possuam o mesmo valor de desempenho entre si ou com algum indivíduo da população C_i será averiguado qual deles possui o menor comprimento efetivo (número de linhas diferentes de 'NOP'), e ao maior ou aos maiores será atribuído valor zero à aptidão. A seguir, ambas as populações serão ordenadas em conjunto de acordo com as medidas de desempenho e os M melhores seguirão para a próxima geração na população C_i.

Também a cada geração a população Q_i é atualizada através do operador P de acordo com a população C_i.

Por fim, avalia-se se algum novo indivíduo supera o melhor até então, e o armazena em C_m. É necessário armazenar esse indivíduo separadamente e não somente considerar o melhor indivíduo da população C_i ao fim da evolução, pois a ordenação na população C_i se dá apenas pelo desempenho no treinamento, enquanto que C_m recebe o indivíduo que em algum momento teve o melhor desempenho no treinamento e que se manteve até o final de evolução com o melhor desempenho na validação.

5.6.4 Atualização dos genes quânticos

A cada geração se utiliza o passo s para incrementar nos *qudits* as probabilidades referentes aos *tokens* nos indivíduos clássicos. Cada gene quântico é atualizado de acordo com o respectivo gene clássico. Considerando o conjunto de funções da Tabela 9, em caso de que em um gene clássico o TF seja 0, isto é caso represente a função 'NOP', somente QF será atualizado no gene quântico referente, ou seja os demais *qudits* não serão atualizados, pois os *tokens* de terminal respectivos não tem importância. Seguindo essa

linha de atualizar somente os *qudits* de acordo com a "efetividade" dos *tokens* respectivos, caso TF seja diferente de zero, além de QF as seguintes atualizações serão feitas:

1. QT_1 será atualizado de acordo com TT_1;
2. QE terá a probabilidade referente ao estado '0' incrementada se TT_2 se referir a uma posição de atributo e a referente ao estado '1' se TT_2 se referir a um registrador temporário ou a uma constante;
3. QT_2 só será atualizado se o TT_2 se referir a um registrador temporário ou a uma constante;
4. QA só será atualizado se o TT_2 se referir a uma posição de atributo;
5. QT_3 só será atualizado se o gene clássico respectivo tiver TF igual a 1 ou 2, pois TT_3 só é válido para as funções 'WA' e 'OWA'.

Vale lembrar que o QA é único por indivíduo quântico, enquanto que os demais *qudits* são únicos por gene quântico. Assim, geralmente ocorre de em um indivíduo clássico muitas linhas utilizarem atributos. Dessa forma, um QA pode ser atualizado muitas vezes em uma mesma geração, o que pode levar a uma convergência muito rápida. Para resolver isso, o passo de incremento de QA deve ser muito menor do que s.

5.6.5 Avaliação

Na primeira geração, a população clássica C_i é avaliada no treinamento, e a partir da segunda geração, a população clássica C_i^{obs} é que é avaliada. Essa avaliação se dá da seguinte forma:

1. O conjunto de treinamento é dividido em k pastas, e são separadas as k combinações de $k-1$ pastas;
2. Cada indivíduo é treinado e avaliado k vezes, cada qual com uma dessas combinações;
3. O desempenho de cada indivíduo no treinamento é tomado como a média dentre as k avaliações.

A cada geração, todos os indivíduos C_i que tiverem um desempenho no treinamento melhor que o desempenho no treinamento do melhor indivíduo até então serão avaliados na validação. Isso é feito realizando-se a avaliação em cada uma das k pastas, e tomando-se a média como o desempenho na validação. Assim, caso algum indivíduo supere o desempenho na validação do melhor indivíduo até então, esse será considerado o novo melhor indivíduo.

6 CONSTRUÇÃO DE ATRIBUTOS COM PROGRAMAÇÃO GENÉTICA LINEAR COM INSPIRAÇÃO QUÂNTICA

Este capítulo consiste em apresentar o desenvolvimento de uma outra aplicação da PGLIQ, retratada na Seção 4.7.2, que consiste em usá-la para a construção de atributos, tal como foi realizado com PG e apresentado na Seção 4.9.2.

A maioria dos detalhes apresentados no Capítulo 5 também se aplicam neste, não sendo necessária a sua reapresentação. As pequenas diferenças e novas informações serão tratadas nas seções que seguem.

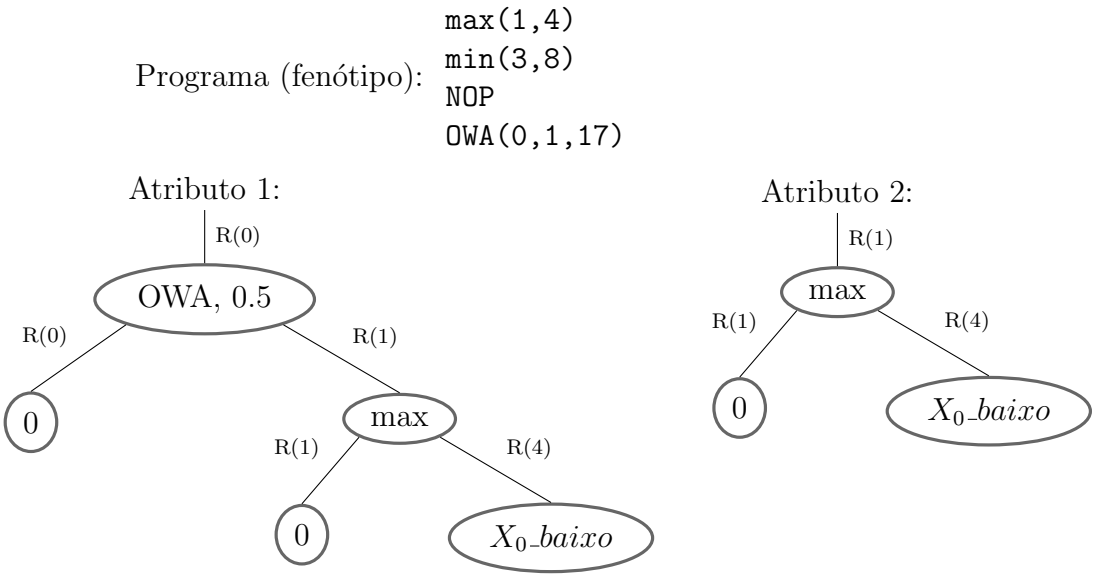
6.1 Atributos Construídos

Conforme retratado na Seção 5.3, em um classificador baseado em PGLIQ a saída do programa executado com determinada amostra pode ser interpretada diretamente como a probabilidade daquela amostra ser da classe '1', ou pode ser necessário transformar tal valor para que seja interpretado dessa forma. Usando a PGLIQ para construir atributos, a saída do programa será um atributo construído ou um conjunto deles. Tal como descrito na Seção 5.1, a saída do programa é tomada na posição 0 do vetor de registradores após a execução do programa. Ao se construir um atributo, o mesmo pode ser tomado de forma idêntica. E ao se construir m atributos, estes serão tomados nas posições 0 a $m-1$ do vetor de registradores. Assim, é necessário que o número de registradores temporários do vetor seja maior ou igual ao número de atributos que se deseja construir. Da mesma forma, o número de linhas do programa também deve ser maior ou igual ao número de atributos a construir, pois cada linha pode alterar somente uma posição do vetor de registradores.

A Figura 17 exibe um exemplo com o fenótipo de um programa de quatro linhas para a construção de dois atributos, sendo estes representados como árvores. Trata-se de um exemplo com APFs, que usa funções já descritas anteriormente na Tabela 8 e atributos fuzzificados, mas da mesma forma também poderia se tratar de outras árvores, com atributos base e funções aritméticas. As arestas indicam as posições do vetor R de registradores e os nós o seu conteúdo. Esse vetor possui quatro registradores temporários (0 a 3), sendo inicializadas com zero. Assim, os atributos (X_i) são inicializados a partir da posição 4, e na posição 17 há a constante 0.5. Como são dois atributos a construir, o primeiro é tomado na posição $R(0)$ e o segundo em $R(1)$. Dessa forma, a linha `min(3,8)`

se trata de um íntron, pois a posição 3 não influencia os resultados. Esse exemplo também é útil para mostrar que a parte ou o todo de um atributo pode ser uma subárvore de outro atributo.

Figura 17: Exemplo de construção de atributos com PGLIQ.



6.2 Aptidão

A principal diferença entre os usos da PGLIQ para classificação e para construção de atributos está no cálculo da aptidão. Enquanto que no uso para classificação a medida de desempenho é calculada diretamente sobre as saídas da PGLIQ, para a construção essas saídas são usadas em um classificador à parte e a partir das saídas deste é calculada a medida de desempenho. Assim, em cada geração, para cada indivíduo são calculados os atributos construídos, sendo estes utilizados em um classificador pré-definido, a fim de se obter a aptidão daquele indivíduo naquela geração.

Caso um atributo seja construído a partir de uma saída constante, ele não irá contribuir para a classificação. Assim, é estabelecido um critério conforme a Equação 14, em que $fitness$ é a aptidão realmente atribuída ao indivíduo e $raw\_fitness$ é a que foi calculada em um classificador pré-definido. O número de atributos construídos é n e c é o número de atributos constantes. Vale observar que, se todos os atributos forem constantes, todo o indivíduo é invalidado, pois $fitness$ recebe valor zero.

$$fitness \leftarrow raw\_fitness(1-\frac{c}{n}) \tag{14}$$

Voltando ao exemplo da Figura 17, caso a posição $R(4)$ tivesse uma constante qualquer ao invés do atributo X_0_baixo, ambos os atributos construídos seriam constantes.

6.3 Evolução

O algoritmo evolutivo implementado para construção de atributos pode ser visto na Tabela 11, e é quase idêntico ao pseudocódigo apresentado na Tabela 10, com o acréscimo das linhas referentes à construção de atributos antes de cada avaliação.

Tabela 11: Algoritmo Evolutivo da Construção de Atributos com PGLIQ.

Algoritmo 4:

para $i \leftarrow 1$ *até* M **faça**
  Inicia indivíduo Q_i
  Inicia indivíduo C_i a partir da observação de Q_i
  Calcula os atributos construídos por C_i nas pastas de treinamento
  Avalia C_i nas pastas de treinamento usando classificador pré-definido
fim
$C_m.aptidao\_Tr \leftarrow 0$
$C_m.aptidao\_V \leftarrow 0$
$g \leftarrow 1$
enquanto $g < G$ *e* $C_m.aptidao\_Tr < 1$ *e* $C_m.aptidao\_V < 1$ **faça**
  para $i \leftarrow 1$ *até* M **faça**
    Atualiza indivíduo C_i^{obs} a partir da observação de Q_i
    Calcula os atributos construídos por C_i^{obs} nas pastas de treinamento
    Avalia C_i^{obs} nas pastas de treinamento usando classificador pré-definido
  fim
  Ordena os indivíduos C em conjunto com os C^{obs}
  para $i \leftarrow 1$ *até* M **faça**
    Aplica operador P em Q_i baseando-se em C_i
  fim
  para $i \leftarrow 1$ *até* M **faça**
    se $C_i.aptidao\_Tr > C_m.aptidao\_Tr$ **então**
      Calcula os atributos construídos por C_i nas pastas de validação
      Avalia C_i nas pastas de validação usando classificador pré-definido
      se $C_i.aptidao\_V > C_m.aptidao\_V$ **então**
        Substitui C_m por C_i
      fim
    fim
  fim
  $g \leftarrow g + 1$
fim

7 MODELO PROPOSTO

Este capítulo apresenta o modelo elaborado neste projeto, que procura identificar alterações respiratórias decorrentes da sarcoidose. Além de uma classificação acurada, busca-se interpretabilidade nos resultados, seja pela seleção ou pela construção, que permitem distinguir os atributos mais importantes, ou seja pelos algoritmos interpretáveis, os quais permitem um maior entendimento dea forma que a classificação é feita ao descrever como os atributos se relacionam até a saída do modelo.

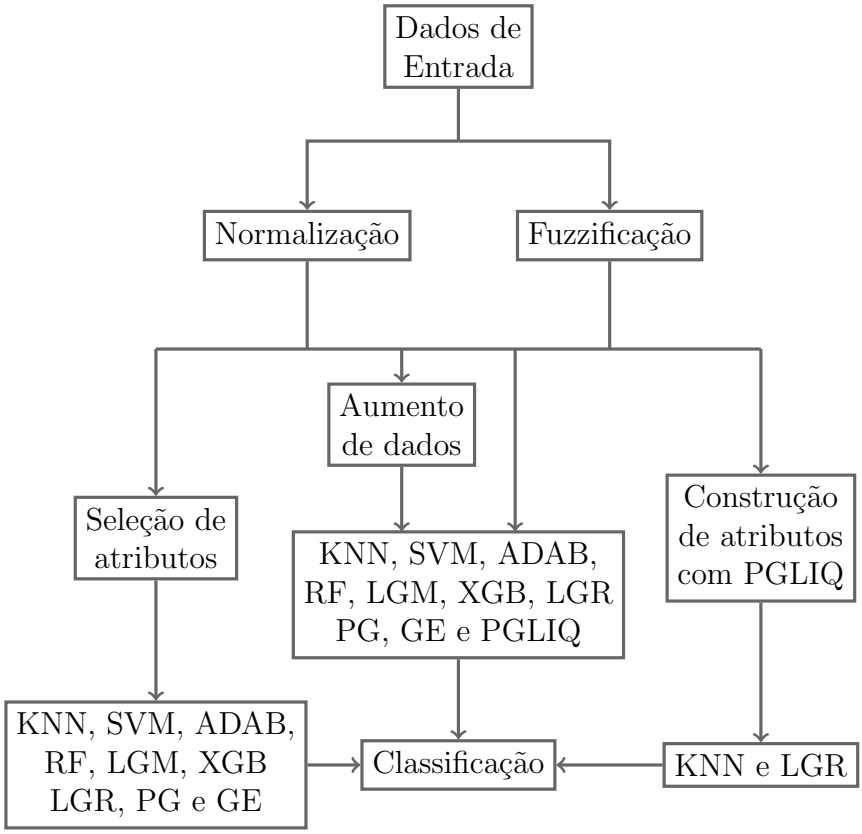
O modelo é resumido segundo o fluxograma da Figura 18, no qual se pode notar que, seja com os dados de entrada 'fuzzificados' ou simplesmente normalizados, a classificação pode ser realizada de quatro formas distintas:

- Utilizando-se diretamente um algoritmo de classificação;
- Selecionando os melhores atributos, e a seguir utilizando um algoritmo de classificação;
- Aumentando artificialmente o número de amostras de treinamento, e a seguir utilizando um algoritmo de classificação;
- Construindo atributos para determinados algoritmos de classificação, e a seguir utilizando-os para classificar.

Conforme está apresentado no Capítulo 8, este trabalho executa três tipos de análise com o conjunto de dados. Para cada uma, a fim de realizar uma comparação entre os desempenhos dos diferentes métodos no modelo proposto, é necessário estabelecer um protocolo. Assim, cada base é dividida em dez pastas, possibilitando dez experimentos, cada um utilizando nove pastas para treino e uma para teste. Todos os métodos utilizam os mesmos conjuntos de treino e de teste. A métrica de desempenho de cada método é a AUC, que é calculada conforme a seguir:

1. Cada experimento realiza uma "busca em grade" pelos melhores hiperparâmetros de cada método. Essa busca é feita com a função `GridSearchCV`, da biblioteca *Scikit-learn* para *Python* [94]. Tal função realiza uma busca exaustiva por hiperparâmetros especificados para um estimador, utilizando validação cruzada. Assim, em cada experimento, o conjunto de treino é dividido em dez pastas, e faz-se a busca em

Figura 18: Fluxograma do modelo proposto.



grade pelos melhores hiperparâmetros, que após serem definidos, são aplicados no conjunto de teste;

2. Ao fim de cada experimento, são tomados do respectivo conjunto de teste os rótulos das amostras e também a probabilidade de cada amostra ser da classe '1'. Com isso, ao fim dos dez experimentos junta-se esses resultados e confecciona-se a curva ROC, tomando-se a AUC referente àquele método naquela base de dados.

7.1 Normalização

Como os limites inferior e superior no conjunto de dados variam muito de um atributo para outro, é necessário normalizar os dados para deixar todos os atributos na mesma escala. Uma das técnicas mais comuns para normalização de dados é a *z-score*, que possui média zero e variância unitária, sendo definida conforme a Equação 15, em que z_i é o valor *z-score* referente à amostra x_i, $\bar{x}$ é a média das amostras e σ é o desvio-padrão.

$$z_i = \frac{x_i - \bar{x}}{\sigma} \tag{15}$$

A *z-score*, por definição, descreve a localização de uma amostra x_i em relação à média em unidades de desvio-padrão. Isso significa que um *z-score* negativo indica que a amostra está abaixo da média, enquanto que um positivo está acima. Além disso, um *z-score* de valor absoluto unitário indica que a amostra está um desvio-padrão acima ou abaixo da média [95].

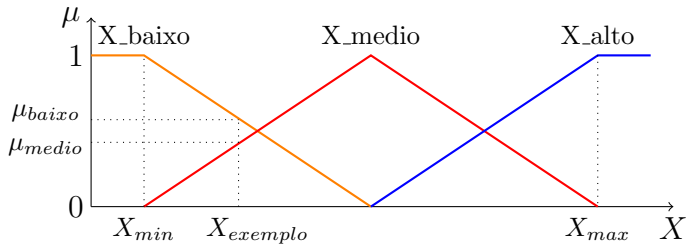
7.2 Fuzzificação

O principal objetivo para se criar uma etapa de "fuzzificação" neste trabalho foi a necessidade de se adequar os dados de entrada, de forma que pudessem ser utilizados em um modelo que gerasse árvores de padrões *fuzzy* (APFs). Entretanto, como se trata apenas de uma outra representação dos dados, pode ser usada com qualquer outro modelo. Além disso, ao "fuzzificar" os dados, todos os atributos ficarão na mesma escala, não sendo necessário normalizar antes de usar com outros algoritmos.

A fuzzificação neste trabalho gera um número de atributos três vezes maior que o original, assim nos algoritmos que trabalham bem com um número grande de atributos pode haver melhoria no desempenho. Outrossim, nos experimentos que selecionam os melhores atributos, o fato deles estarem "fuzzificados" pode aumentar a interpretabilidade dos resultados.

O esquema de fuzzificação utilizado é bem simples e pode ser visto na Figura 19, em que X é um atributo a ser fuzzificado, X_{max} é o seu maior valor presente no conjunto de dados e X_{min} é o menor. O atributo X é fuzzificado em três conjuntos *fuzzy*, representados por X_baixo, X_medio e X_alto. As funções de pertinência são triangulares, e os respectivos valores de pertinência são tomados a partir do eixo vertical, enquanto que o eixo horizontal representa os valores reais do atributo. Assim, por exemplo, o maior valor presente no conjunto de dados tem pertinência 1 no conjunto X_alto e 0 nos demais, enquanto que o menor valor tem pertinência 1 no conjunto X_baixo e 0 nos outros. Todos os outros dados têm pertinência maior que zero em dois conjuntos vizinhos e igual a zero no terceiro, como por exemplo a amostra $X_{exemplo}$, que possui pertinência $\mu_{baixo} > 0$ no conjunto X_baixo, $\mu_{medio} > 0$ em X_medio e 0 em X_alto. Uma exceção é quando houver uma amostra cujo valor de X seja exatamente $\frac{X_{max}-X_{min}}{2}$, quando então tem pertinência 1 no conjunto X_medio e 0 nos outros.

Figura 19: Esquema de fuzzificação.



7.3 Classificadores

7.3.1 Algoritmos Básicos

Foram implementados classificadores utilizando-se os seguintes algoritmos, a partir de funções da biblioteca *Scikit-learn* [94]: K-Vizinhos mais Próximos (KNN), *Support Vector Machine* (SVM), *AdaBoost* (ADAB), *Random Forest* (RF), *LightGBM* (LGM), *XGBoost* (XGB) e Regressor Logístico (LGR).

Para cada classificador, em cada experimento o método de busca em grade verifica todas as combinações possíveis dos hiperparâmetros abaixo listados, a fim de escolher os melhores a usar no conjunto de teste. Como se trata de uma busca exaustiva, foram variados somente os atributos mais comuns. No caso do *AdaBoost*, que utiliza o classificador por árvores de decisão (do inglês *Decision Tree Classifier*, ou DTC) como estimador base, também são combinados hiperparâmetros deste classificador.

- K-Vizinhos mais Próximos
  - n_neighbors: {1, 3, 5, 7, 9, 11, 13}
- *Support Vector Machine*
  - C: {1, 2, 5, 7, 10, 50, 100, 200, 400}
  - gamma: {0.001, 0.01, 0.05, 0.1, 1}
- *AdaBoost*
  - n_estimators: {10, 30, 60, 100, 200, 400}
  - max_depth (DTC): {1, 2, 3, 4, 5, 10, 15, 30, 60}
- *Random Forest*
  - n_estimators: {10, 30, 60, 100, 200, 400}
  - max_depth: {1, 2, 3, 4, 5, 10, 15, 30, 60}

- *LightGBM*
  - max_depth: {1, 2, 3, 4, 5, 10, 15, 30, 60}
  - n_estimators: {10, 30, 60, 100, 200, 400}
- *XGBoost*
  - max_depth: {1, 2, 3, 4, 5, 10, 15, 30, 60}
  - n_estimators: {10, 30, 60, 100, 200, 400}
- Regressor Logístico
  - C: {0.001, 0.01, 0.1, 1, 10, 100, 1000}

7.3.2 Programação Genética

A classificação com PG neste trabalho foi feita com a biblioteca *gplearn*, que implementa PG em *Python*, usando uma API (*Application Programming Interface*) inspirada e compatível com *Scikit-learn* [96]. A função para classificação nesta biblioteca transforma diretamente a saída da árvore em uma probabilidade por meio da função sigmoide, portanto ela foi usada somente com os atributos normalizados. Com os atributos fuzzificados, foi utilizada a função para regressão do *gplearn*, que apresenta como saída um valor entre 0 e 1, já que nesse caso são usados somente operadores *fuzzy*. Assim, essa saída é interpretada como a possibilidade da amostra ser da classe 1, sem a necessidade da função sigmoide.

Com os dados normalizados, as árvores foram criadas com funções aritméticas simples ('add', 'sub', 'mul' e 'div'). Já para os dados fuzzificados, foram implementadas funções com operadores *fuzzy* ('WA', 'OWA', 'max', 'min', 'concentrator' e 'dilator'), cujas descrições foram apresentadas na Tabela 8, no Capítulo 5. Vale salientar que, com a PG, diferentemente da GE e da PGLIQ, não foi possível permitir que as entradas das funções WA e OWA, referentes ao fator de multiplicação, recebessem somente constantes. É permitido que recebam também atributos e até mesmo resultados de outras funções.

A busca em grade foi feita com os dois principais hiperparâmetros, conforme a seguir:

- *PG*
  - population_size: {100, 300, 500, 1000, 1500, 2000, 3000}
  - generations: {5, 10, 15, 20, 30, 40}

7.3.3 Evolução Gramatical

A classificação com GE neste trabalho foi feita com a biblioteca *ponyGE2* [84], que implementa GE em *Python*. Entretanto, como essa biblioteca não é compatível com *Scikit-learn* [96], foi necessário construir um arcabouço para compatibilizá-la, e então permitir a busca em grade de seus hiperparâmetros usando `GridSearchCV`.

A gramática definida para uso neste trabalho pode ser vista na Figura 20, em que as regras (I) a (IV) são utilizadas nos experimentos com atributos normalizados, e as regras (V) a (X) naqueles com atributos fuzzificados.

Figura 20: Gramática utilizada pela GE.

```
(I)   <e>   ::= <op>(<e>, <e>) | <op>(<e>, <c>) | x[<idx>]
(II)  <op>  ::= add | mul | sub | pdiv
(III) <idx> ::= GE_RANGE:dataset_n_vars
(IV)  <c>   ::= 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9

(V)    <e>   ::= <f1>(<e>, <e>, <c>) | <f2>(<e>, <e>) | <f3>(<e>)
                 | x[<idx>] | <c>
(VI)   <f1>  ::= WA | OWA
(VII)  <f2>  ::= minimum | maximum
(VIII) <f3>  ::= dilator | concentrator
(IX)   <idx> ::= GE_RANGE:dataset_n_vars
(X)    <c>   ::= 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9
```

Foi definido o valor 5 ao hiperparâmetro que se refere à profundidade máxima das árvores, para garantir simplicidade nos resultados, facilitando a interpretação. Para os demais hiperparâmetros foram utilizados os seus valores padrão, exceto nos dois principais, nos quais foi feita busca em grade conforme a seguir:

- *GE*
  - population_size: {100, 200, 500, 1000, 3000}
  - generations: {50, 100, 200}

7.3.4 Programação Genética Linear com Inspiração Quântica

Um classificador baseado em PGLIQ foi implementado em *Python*, usando uma API (*Application Programming Interface*) inspirada e compatível com *Scikit-learn* [96]. Com entradas fuzzificadas e operadores *fuzzy* no `function_set`, o resultado são APFs, cuja saída é um valor entre 0 e 1, interpretado como a probabilidade da amostra de entrada

ser da classe 1. Entretanto também podem ser usadas entradas normalizadas e outro tipo de funções, por exemplo aritméticas. Neste caso, a saída passa pela função sigmoide para definir a probabilidade da amostra ser da classe.

Os hiperparâmetros do classificador são os seguintes, sendo a busca em grade realizada com `individual_length` e `generations`:

- `population_size`: tamanho M de cada população (C, C^{obs} e Q, conforme apresentadas no Capítulo 5), ou seja, o número de indivíduos. Usa-se um valor padrão igual a 6;
- `individual_length`: número de linhas L de cada indivíduo. A busca em grade foi feita com os valores {4, 8, 16, 32};
- `step`: passo de incremento s de todos os indivíduos quânticos, exceto QA, sendo o passo deste definido em $s/50$. Usa-se um valor padrão igual a 0.004;
- `generations`: número máximo de gerações. A busca em grade foi feita com os valores {10000, 20000, 30000};
- `const_set`: conjunto de constantes disponíveis para uso nas funções. Usa-se um valor padrão igual a [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9];
- `function_set`: conjunto de funções disponíveis. Nos experimentos com atributos fuzzificados, usa-se o valor padrão ['WA', 'OWA', 'min', 'max', 'dilator', 'concentrator'] e nos demais experimentos, o padrão é ['add', 'sub', 'mul', 'div'];
- `metric`: métrica de desempenho. Usa-se como padrão a AUC;
- `n_reg`: número de registradores temporários no vetor, que serão usados para receber resultados. Fora da busca em grade foram feitos experimentos em quantidade não-massiva, e estimou-se um valor padrão igual a 6.

7.4 Seleção de Atributos

A seleção de atributos foi feita com a função `RFE`, da biblioteca *Scikit-learn* [94], que utiliza eliminação recursiva de atributos. O estimador utilizado é a função `LinearSVC`, da mesma biblioteca. É feita a eliminação de um atributo por vez (`step=1`), e com cada classificador apresentado na Seção 7.3.1 em conjunto com a busca em grade pelos seus hiperparâmetros é feita uma busca acerca do número de atributos a selecionar, que varia de 1 até o número total de atributos menos um, exceto para os algoritmos evolutivos, que devido ao tempo de execução, são buscadas apenas três quantidades diferentes. Isso é

resumido conforme a seguir:

- `n_features_to_select`
  - atributos normalizados: {1, 2, ..., 14, 15} ou {4,8,12}
  - atributos fuzzificados: {1, 2, ..., 46, 47} ou {12, 18, 24}

7.5 Aumento de Dados

O aumento de dados foi feito com a função `SMOTE`, da biblioteca *Imbalanced-learn*, que utiliza as técnicas descritas na Seção 4.10. Conforme será descrito na Seção 8.1, foram feitos três tipos de análise nos estudos de caso, todos utilizando uma classe com 25 dados. Na primeira análise, essa classe é estudada com outra que possui 47 dados, na segunda com uma de 23 dados e na terceira com outra de 24 dados. Percebe-se que a primeira análise possui uma quantidade pequena de dados e classes muito desequilibradas. Já as outras duas possuem classes equilibradas, mas ainda menos dados no total. Assim, na primeira análise o aumento de dados foi feito de forma a equilibrar as classes, ou seja, somente os dados da classe menos frequente foram aumentados. Dessa forma, ao invés de 72 dados, os classificadores utilizam 94 dados, sendo 22 deles sintéticos. Já na segunda e na terceira, os dados foram aumentados de ambas as classes, e com cada classificador apresentado na Seção 7.3.1 em conjunto com a busca em grade pelos seus hiperparâmetros foi feita uma busca acerca de quanto aumentar os dados, conforme a seguir. A primeira opção consiste em apenas igualar a quantidade de cada classe, já que uma delas possui 25 dados originais. Já as outras expandem a quantidade total de dados para 60, 80 e até 100.

- `sampling_strategy`: {{0: 25, 1: 25}, {0: 30, 1: 30}, {0: 40, 1: 40}, {0: 50, 1: 50}}

7.6 Construção de Atributos

Foi implementado em *Python* um construtor de atributos baseado em PGLIQ, usando uma API inspirada e compatível com *Scikit-learn* [96].

Os hiperparâmetros do construtor são exatamente os mesmos do classificador, já apresentado na Seção 7.3.4, acrescidos de mais dois, conforme a seguir:

- `model_fitness`: modelo que realiza a classificação utilizando os atributos construídos, a fim de calcular a métrica de desempenho. Usa-se como padrão a função `LogisticRegression`, com os valores padrão de seus hiperparâmetros;
- `n_new_features`: número de atributos a construir. A busca em grade foi feita com os valores {2, 3};.

Diferentemente do classificador baseado em PGLIQ, não é feita a busca em grade com `generations`, utilizando-se o valor 10000. Optou-se por isso, pois além do tempo de execução de uma geração ser maior, devido à forma como é calculada a aptidão, ainda foi incluso na busca em grade o hiperparâmetro que se refere ao número de atributos a construir. Por outro lado, manteve-se na busca em grade o hiperparâmetro `individual_length`, com as opções {4, 8, 16}, pois trata-se de um hiperparâmetro de vital importância. Além disso, cabe ressaltar que agora o hiperparâmetro `metric` se refere à métrica de desempenho que será medida a partir do resultado do classificador definido por `model_fitness`.

8 ESTUDOS DE CASO

Este capítulo inicia com uma descrição estatística dos dados utilizados neste trabalho. A seguir, são apresentados resultados de experimentos realizados a fim de testar os algoritmos expostos nos capítulos anteriores.

8.1 Descrição do Conjunto de Dados

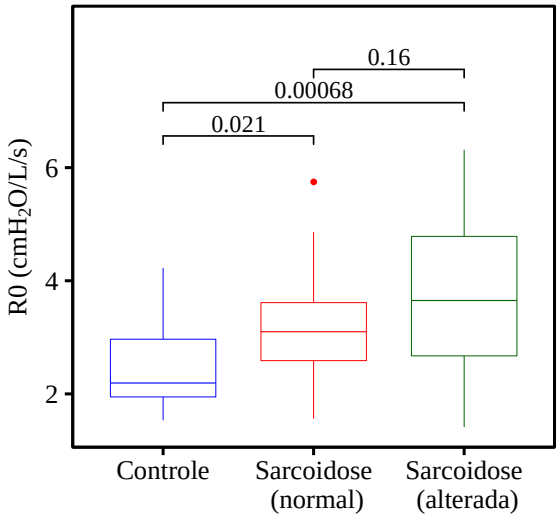
Os dados utilizados neste trabalho foram obtidos por um sistema capaz de realizar a FOT, tal como já mostrado na Figura 1, desenvolvido no Laboratório de Instrumentação Biomédica, desta mesma instituição de ensino (LIB/UERJ). O exame com cada indivíduo foi repetido três vezes, e cada dado usado neste trabalho é o resultado da média dessas três medidas. Participaram da coleta de dados 25 indivíduos saudáveis, que compõem o grupo de controle e 47 diagnosticados com sarcoidose. Nesses últimos, foi verificado pela espirometria que 24 apresentaram condições normais e 23 mostraram alterações respiratórias. E os atributos referentes ao modelo eRIC foram estimados usando-se o programa ModeLIB, também desenvolvido pelo LIB/UERJ. Assim, tem-se um conjunto com 72 dados e 16 atributos (conforme já mostrados na Tabela 1 e na Tabela 2), e cada experimento será feito com três tipos de análise:

- Grupo de controle × indivíduos com sarcoidose: 72 dados.
- Grupo de controle × indivíduos com sarcoidose e espirometria alterada: 48 dados.
- Grupo de controle × indivíduos com sarcoidose e espirometria normal: 49 dados.

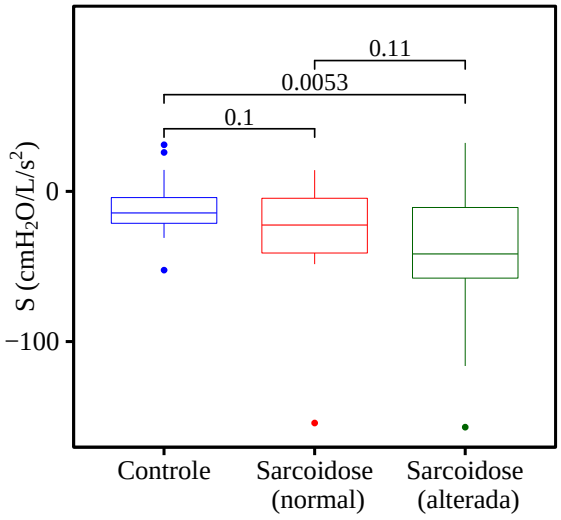
Na Figura 21 são apresentados os diagramas de caixa (do inglês *boxplots*). Cada caixa é limitada entre o primeiro quartil e o terceiro quartil, ou seja, engloba 50% dos dados, e a mediana é indicada entre eles. Saindo de cada quartil há os "fios de bigode"(do inglês *whiskers*), que indicam o valor mínimo e o valor máximo, exceto quando estes distem do primeiro quartil e do terceiro quartil, respectivamente, mais de 150% da distância entre os quartis. Os dados que se enquadrem nesses casos são considerados *outliers* e representados por pontos. Os diagramas de caixa caracterizam a variação de cada atributo, de forma que se possa compará-la entre as três diferentes classes de dados. O eixo vertical representa a variação dos atributos, e para cada um foram traçadas três caixas: uma com

Figura 21: Diagrama de caixa de cada atributo.

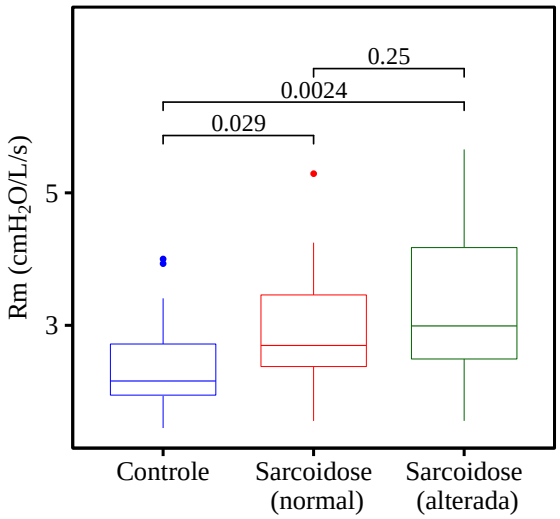
(a) Resistência no Intercepto



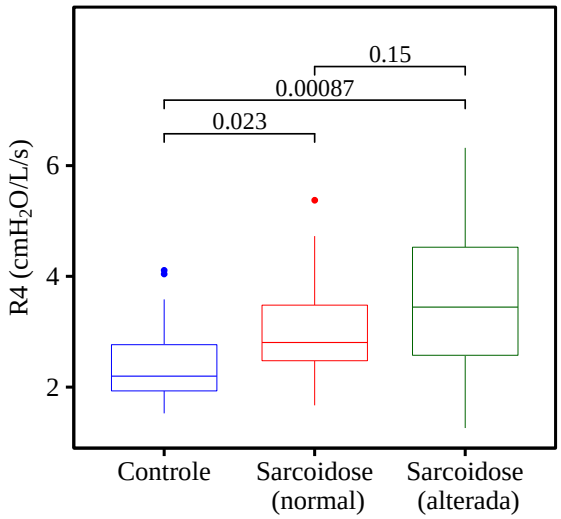
(b) Inclinação da curva de resistência



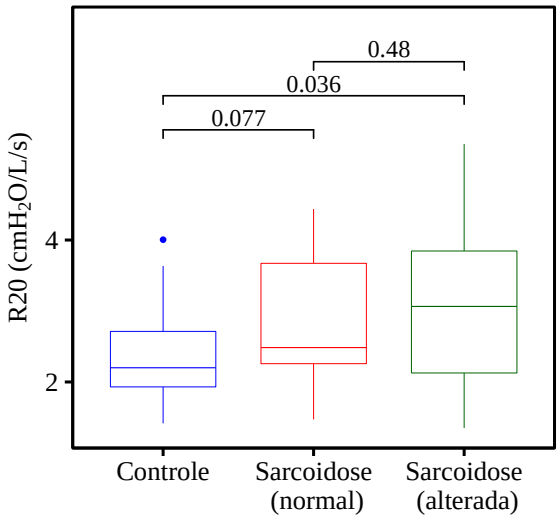
(c) Resistência média na faixa de 4-16 Hz



(d) Resistência a 4 Hz



(e) Resistência a 20 Hz



(f) Diferença entre R4 e R20

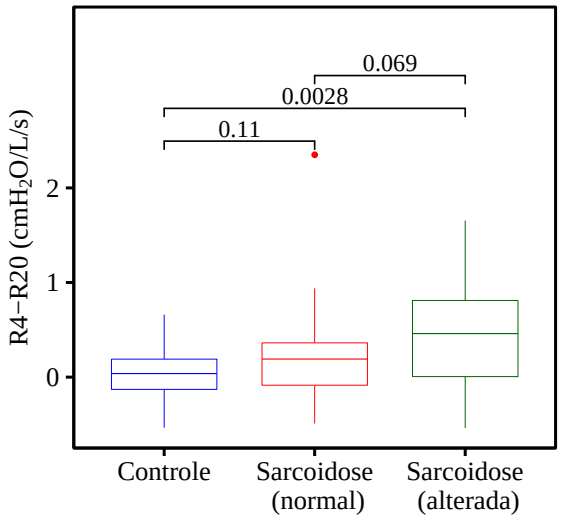
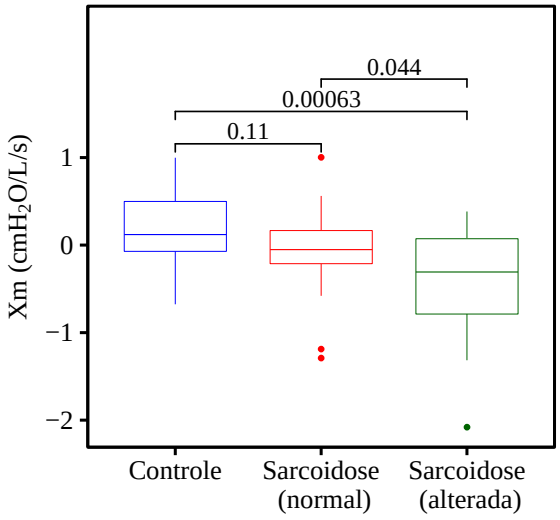
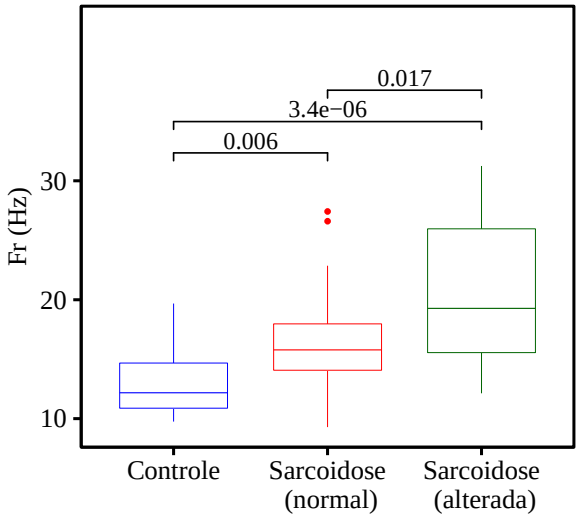


Figura 21: Diagrama de caixa de cada atributo.

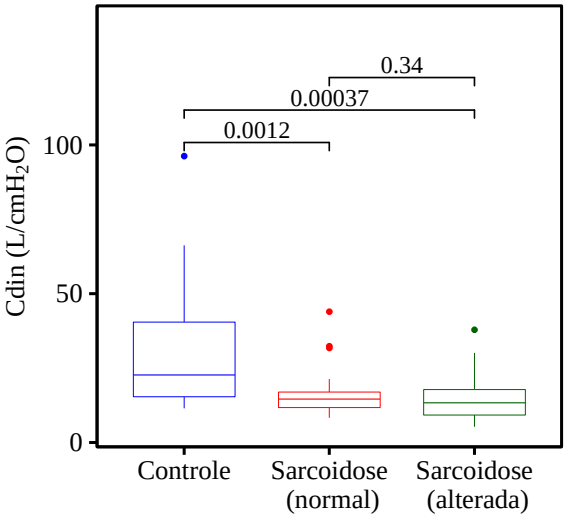
(g) Reatância média na faixa de 4-32 Hz



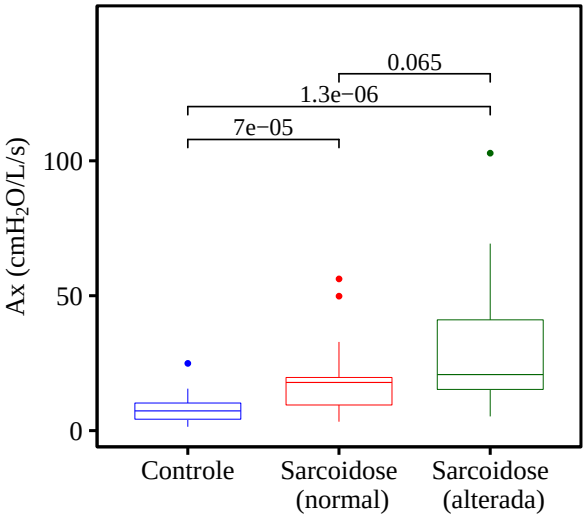
(h) Frequência de ressonância



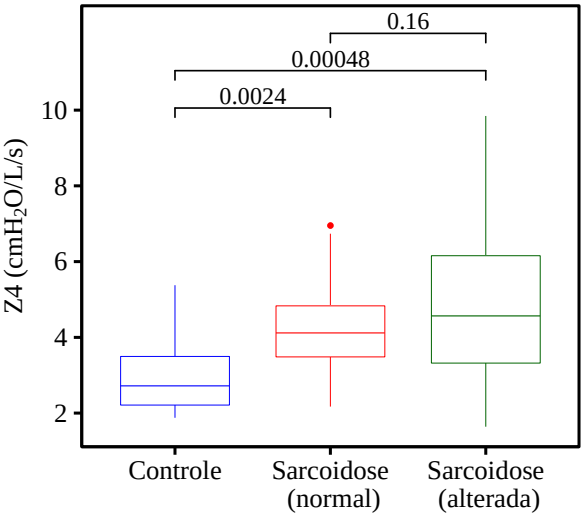
(i) Complacência dinâmica



(j) Área da parte negativa da curva de reatância



(k) Impedância a 4 Hz



(l) Resistência central

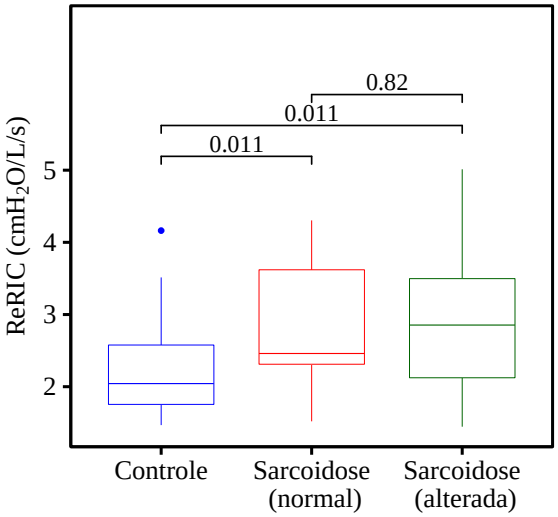
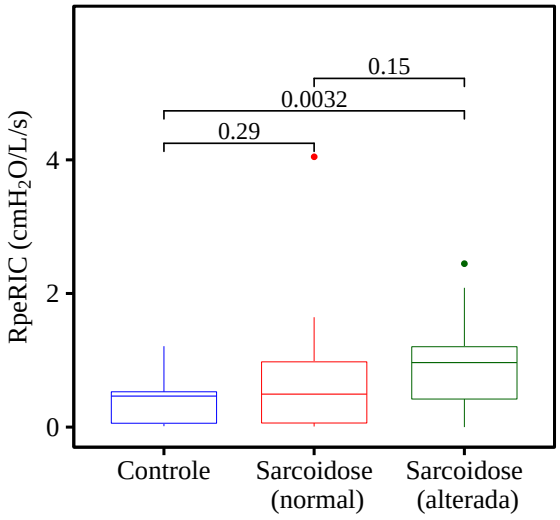
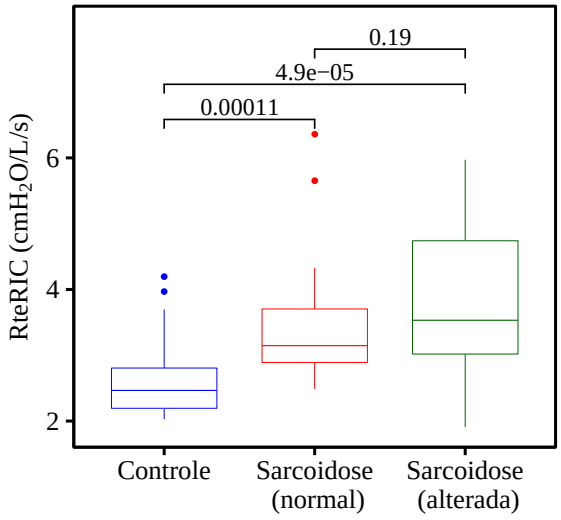


Figura 21: Diagrama de caixa de cada atributo.

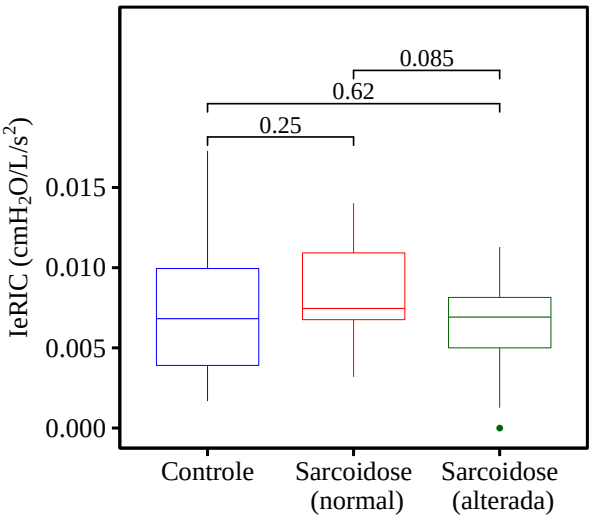
(m) Resistência periférica



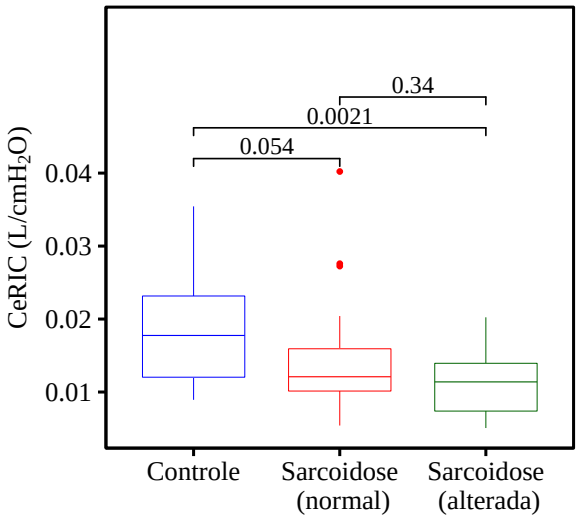
(n) Resistência total (ReRIC + RpeRIC)



(o) Inertância do sistema respiratório



(p) Complacência do sistema respiratório



as amostras referentes ao grupo de controle e duas com aquelas pertencentes aos pacientes diagnosticados com sarcoidose, sendo um grupo com espirometria normal e outro com espirometria alterada. Entre os grupos, dois a dois, é apresentado o valor-p, calculado por meio do teste de Wilcoxon-Mann-Whitney.

Analisando os indivíduos com sarcoidose, não foram encontradas diferenças significativas ($p > 0.05$) entre os grupos com espirometria normal e alterada na maioria dos atributos, exceto Xm e Fr.

Ao se comparar o grupo de controle com o grupo com sarcoidose e espirometria normal não foram encontradas diferenças significativas ($p > 0.05$) nos atributos S, R20,

R4-R20, Xm, RpeRIC, IeRIC e CeRIC. Já os atributos Ax e RteRIC apresentaram os melhores valores ($p < 0.001$).

Comparando o grupo de controle com o de sarcoidose e espirometria alterada, somente no parâmetro IeRIC não foram observadas diferenças significativas ($p > 0.05$). Enquanto que os melhores valores ($p < 0.001$) foram encontrados em R0, R4, Xm, Fr, Cdin, Ax, Z4 e RteRIC.

Os diagramas de caixa da Figura 21 apresentam os valores-p entre os conjuntos, dois a dois, entretanto como é do interesse deste trabalho analisar também o grupo de controle × indivíduos com sarcoidose, a Tabela 12 apresenta o valor p de cada atributo entre o conjunto de dados do grupo de controle e a união dos outros dois conjuntos. Percebe-se que há diferenças significativas ($p < 0.05$) entre os conjuntos em todos os atributos, exceto IeRIC. Além disso, os melhores valores ($p < 0.001$) foram encontrados em R0, Fr, Cdin, Ax, Z4 e RteRIC.

Tabela 12: Valor-p de cada atributo na análise controle × indivíduos com sarcoidose.

Controle × Sarcoidose			
Atributo	**Valor-p**	**Atributo**	**Valor-p**
R0	0.00091	Cdin	0.000068
S	0.01	Ax	0.00000027
Rm	0.0024	Z4	0.00014
R4	0.0011	ReRIC	0.0029
R20	0.024	RpeRIC	0.021
R4-R20	0.008	RteRIC	0.0000035
Xm	0.0039	IeRIC	0.7
Fr	0.00002	CeRIC	0.004

Ao fuzzificar os dados, as comparações descritas nos parágrafos anteriores podem ser enriquecidas com novas observações. Como a fuzzificação utilizada neste trabalho triplica o número de atributos, optou-se por não apresentar diagramas de caixa. A Tabela 13 lista para as análises controle × sarcoidose (normal) e controle × sarcoidose (alterada) os atributos que apresentaram diferenças significativas ($p < 0.05$) entre os conjuntos.

Inicialmente cabe destacar que, tal como nas comparações entre os grupos com os atributos base, o número de atributos com diferenças significativas ($p < 0.05$) entre os grupos é bem maior na análise controle × sarcoidose (alterada), já que são esperadas alterações respiratórias mais evidentes.

A partir da Tabela 13 já se pode comentar o quanto os termos *fuzzy* podem con-

Tabela 13: Atributos fuzzificados com diferenças significativas entre os conjuntos controle e sarcoidose, com espirometria normal ou alterada.

Controle × sarcoidose (normal)		**Controle × sarcoidose (alterada)**			
Atributo	**Valor-p**	**Atributo**	**Valor-p**	**Atributo**	**Valor-p**
Ax_baixo	0.00007	Ax_baixo	0.0000068	CeRIC_alto	0.0036
Ax_medio	0.00007	Fr_baixo	0.000012	RpeRIC_baixo	0.0039
RteRIC_baixo	0.00024	Ax_medio	0.000065	Rm_baixo	0.0042
Cdin_baixo	0.0015	RteRIC_baixo	0.00014	S_alto	0.006
RteRIC_medio	0.0023	Fr_alto	0.0003	Rm_alto	0.0077
Z4_baixo	0.003	Cdin_baixo	0.00056	R4-R20_medio	0.0084
Z4_medio	0.0035	Z4_baixo	0.00068	RteRIC_medio	0.009
Fr_baixo	0.0067	Z4_alto	0.00069	Z4_medio	0.011
Cdin_medio	0.0072	Xm_alto	0.0011	Xm_baixo	0.012
ReRIC_baixo	0.019	R0_baixo	0.0016	R4_alto	0.015
Fr_medio	0.022	R4_baixo	0.0019	R4-R20_alto	0.016
R0_baixo	0.028	RteRIC_alto	0.0023	IeRIC_alto	0.016
R4_baixo	0.028	CeRIC_baixo	0.0025	S_baixo	0.016
CeRIC_baixo	0.039	R0_alto	0.003	ReRIC_baixo	0.017
CeRIC_medio	0.049	Cdin_medio	0.0031	CeRIC_medio	0.029
		RpeRIC_medio	0.0032	Fr_medio	0.034
		R4-R20_baixo	0.0034		

tribuir para a interpretabilidade dos resultados. Por exemplo, já se havia notado que o atributo Fr tem importância na discriminação de sarcoidose, seja com espirometria normal ou alterada, entretanto, agora se sabe que Fr_alto tem importância na análise com espirometria alterada, e não tem na normal. Intrinsecamente já se percebe informação contida nesse termo, isto é, que há poucos indivíduos com Fr alto nos grupos de controle e com espirometria normal. De fato, analisando-se o diagrama de caixa de Fr, percebe-se que os seus valores mais altos se concentram no grupo com espirometria alterada.

Outro fato interessante é que diversos atributos fuzzificados possuem um valor-p menor se comparado à mesma análise com seu respectivo atributo base, denotando em uma maior diferença significativa entre os conjuntos. Pode-se citar como exemplos alguns atributos base que não apresentaram diferenças significativas entre os grupos. Na análise controle × sarcoidose (alterada), IeRIC apresentou $p = 0.62$, sendo o pior atributo neste quesito, enquanto que IeRIC_alto apresentou $p = 0.016$ na mesma análise. Da mesma forma, CeRIC apresentou $p = 0.054$ na análise com espirometria normal, enquanto que CeRIC_baixo e CeRIC_medio apresentaram, respectivamente, $p = 0.039$ e $p = 0.049$. Tal

fato pode ser entendido como uma concentração de informações relevantes em determinados atributos fuzzificados, diminuindo ou até eliminando a influência de *outliers*, o que será melhor entendido nos parágrafos seguintes.

A Tabela 14 apresenta a quantidade de valores diferentes de zero nos atributos fuzzificados. Essa tabela pode ter seus valores interpretados como uma medida da quantidade de informação presente em cada atributo fuzzificado. De acordo com o esquema de fuzzificação apresentado na Figura 19, com o conjunto de 72 dados usado neste trabalho já era esperado que os atributos X_medio, em que X é um dos 16 atributos base, tivessem 70 valores diferentes de zero. Isso porque somente o maior e o menor valor de cada atributo não possuem pertinência em X_medio, apresentando pertinência 1 em X_alto e X_baixo, respectivamente, e zero nas demais. E caso os dados estivessem igualmente distribuídos em cada atributo, acima e abaixo de suas respectivas médias, seria esperado que cada X_baixo e cada X_alto tivesse por volta de 36 valores diferentes de zero. Nenhum atributo sequer se aproxima dessa quantidade, principalmente devido às diferenças de distribuição em grupos diferentes (controle e sarcoidose com espirometria normal ou alterada), conforme observado nos diagramas de caixa. E, em casos mais extremos, notadamente RpeRIC, Ax, Cdin, S e R4-R20, isso pode ser explicado ao se observar os seus respectivos diagramas de caixa, nos quais se percebe que esses atributos são os que possuem os *outliers* mais distantes dos demais valores. No caso de S, isso leva a um X_min muito menor que a média do atributo, e então poucas amostras terão pertinência em X_baixo. Nos demais casos citados, *outliers* muito maiores que a média dos atributos levam a X_max demasiadamente elevados, permitindo poucas amostras com pertinência em X_alto. O mesmo ocorre em menor grau com os demais atributos, de forma que muitos dos atributos fuzzificados podem acabar sendo irrelevantes.

Entretanto, tal fato não chega a ser um problema. Pelo contrário, pode ser até útil em modelos com seleção de atributos, ou propriamente em algoritmos com seleção embutida, pois se por um lado um atributo ao ser fuzzificado em outros três pode ter um desses possivelmente irrelevante, a qualidade dos outros dois pode ser superior a do atributo original, pois possuirão menos informação de *outliers*. Por exemplo, no atributo Ax, o valor discrepante mais alto possui pertinência 1 no conjunto Ax_alto e 0 nos demais, enquanto que o restante das amostras terá suas informações concentradas em Ax_baixo e Ax_medio. Mesmo os outros três dados, não tão discrepantes, que possuem pertinência

Tabela 14: Quantidade de valores diferentes de zero em cada atributo fuzzificado.

Atributo	**Qtde**	**Atributo**	**Qtde**	**Atributo**	**Qtde**
R0_baixo	55	R4-R20_medio	70	Z4_alto	11
R0_medio	70	R4-R20_alto	7	ReRIC_baixo	54
R0_alto	17	Xm_baixo	11	ReRIC_medio	70
S_baixo	6	Xm_medio	70	ReRIC_alto	19
S_medio	70	Xm_alto	61	RpeRIC_baixo	69
S_alto	66	Fr_baixo	59	RpeRIC_medio	70
Rm_baixo	55	Fr_medio	70	RpeRIC_alto	3
Rm_medio	70	Fr_alto	13	RteRIC_baixo	58
Rm_alto	17	Cdin_baixo	67	RteRIC_medio	70
R4_baixo	57	Cdin_medio	70	RteRIC_alto	14
R4_medio	70	Cdin_alto	5	IeRIC_baixo	49
R4_alto	15	Ax_baixo	68	IeRIC_medio	70
R20_baixo	53	Ax_medio	70	IeRIC_alto	23
R20_medio	70	Ax_alto	4	CeRIC_baixo	61
R20_alto	19	Z4_baixo	61	CeRIC_medio	70
R4-R20_baixo	65	Z4_medio	70	CeRIC_alto	11

maior que zero em Ax_alto, terão ainda pertinência também em Ax_medio, influenciando os resultados, caso o atributo Ax_alto seja descartado.

De fato, Ax é um atributo que, quanto ao valor-p, se destacou na análise controle × sarcoidose (normal) ($p = 0.00007$) e também na análise controle × sarcoidose (alterada) ($p = 0.0000013$), e ao ser fuzzificado, continua a se destacar com Ax_baixo ($p = 0.00007$ e $p = 0.0000068$, respectivamente) e Ax_medio ($p = 0.00007$ e $p = 0.000065$, respectivamente) em ambas, enquanto que Ax_alto não apresenta diferenças significativas entre os grupos ($p = 0.33$ e $p = 0.069$, respectivamente), mostrando-se um dos piores atributos neste quesito. As mesmas observações podem ser feitas com RteRIC ($p = 0.00011$ e $p = 0.000049$, respectivamente), pois agora temos RteRIC_baixo ($p = 0.00024$ e $p = 0.00014$, respectivamente), com Cdin ($p = 0.0012$ e $p = 0.00037$, respectivamente), porque agora temos Cdin_baixo ($p = 0.0015$ e $p = 0.00056$, respectivamente), dentre outros.

A PGLIQ, ao usar atributos fuzzificados, é um modelo que, além de já possuir um seletor embutido, ainda é capaz de encontrar utilidade para os atributos com muitos zeros. Isto devido à disponibilidade das funções `max` e `OWA`[8], que são capazes de aproveitar somente os valores diferentes de zero presentes nesses atributos.

Por fim, tal como feito para os atributos base, também é interessante analisar o

[8] OWA(x, y, r) = $r \times \max(x, y) + (1 - r) \times \min(x, y)$

Tabela 15: Atributos fuzzificados com diferenças significativas entre os conjuntos controle e sarcoidose.

Controle × Sarcoidose					
Atributo	**Valor-p**	**Atributo**	**Valor-p**	**Atributo**	**Valor-p**
Ax_baixo	0.0000013	CeRIC_baixo	0.0032	CeRIC_medio	0.016
Ax_medio	0.0000054	Rm_baixo	0.0033	Rm_alto	0.02
RteRIC_baixo	0.000013	Fr_alto	0.0043	RpeRIC_baixo	0.022
Fr_baixo	0.000038	Xm_alto	0.005	R4-R20_medio	0.032
Cdin_baixo	0.00012	ReRIC_baixo	0.0057	R4_alto	0.037
Z4_baixo	0.00021	R4-R20_baixo	0.0086	IeRIC_medio	0.043
Cdin_medio	0.0011	Z4_alto	0.0096	R20_baixo	0.044
RteRIC_medio	0.0011	S_alto	0.011	R4-R20_alto	0.045
Z4_medio	0.0016	Fr_medio	0.011	RpeRIC_medio	0.046
R0_baixo	0.0018	RteRIC_alto	0.013	Xm_baixo	0.047
R4_baixo	0.002	R0_alto	0.015		

grupo de controle × indivíduos com sarcoidose. A Tabela 15 apresenta o valor-p dos atributos que apresentaram diferenças significativas ($p < 0.05$) entre o conjunto de dados do grupo de controle e a união dos outros dois conjuntos. Em geral, os atributos apresentados nessa tabela compreendem a união daqueles exibidos na Tabela 13, o que já era esperado, pois esta análise busca uma discriminação mais geral entre controle e sarcoidose, havendo espirometria alterada ou não. De fato, dentre os atributos apresentados na Tabela 13, apenas CeRIC_alto e IeRIC_alto não aparecem agora, e são atributos que já não apresentaram diferenças tão significativas dentre os grupos antes ($p > 0.01$ na análise controle × sarcoidose com espirometria alterada, e $p > 0.05$ na análise controle × sarcoidose com espirometria normal). Além disso, apenas ReRIC_alto aparece agora e não apareceu na Tabela 13. Outrossim, a maioria dos atributos possui um valor-p menor na análise controle × sarcoidose do que nas demais, indicando que esta análise pode ter uma discriminação mais simples das suas amostras.

8.2 Experimento Individual dos Atributos

Cada atributo foi analisado individualmente a fim de testar o seu desempenho na classificação dos grupos. O resultado pode ser visto na Figura 22. Na análise do grupo de controle × indivíduos com sarcoidose os melhores atributos foram Ax e RteRIC, que apresentaram AUC iguais a 0.83 e 0.80, respectivamente. Enquanto que na análise com os indivíduos com sarcoidose e espirometria alterada os melhores atributos foram Fr, Ax e

RteRIC, que apresentaram AUC iguais a 0.87, 0.87 e 0.82, respectivamente. Já na análise com espirometria normal, nenhum atributo apresentou AUC superior a 0.80, sendo os melhores Ax e RteRIC, que apresentaram AUC iguais a 0.79.

Voltando à seção anterior, na análise controle × sarcoidose (normal), os atributos com diferenças mais significativas entre os grupos foram Ax ($p = 0.00071$), RteRIC ($p = 0.0012$), Z4 ($p = 0.0033$) e Cdin ($p = 0.0042$). Já na análise controle × sarcoidose (alterada), foram Fr ($p = 1.1 \times 10^{-5}$), RteRIC ($p = 0.00012$), Ax ($p = 0.00019$) e Z4 ($p = 0.00037$). E na análise controle × sarcoidose, os atributos com maior destaque foram Ax ($p = 1.7 \times 10^{-6}$), Fr ($p = 2.3 \times 10^{-6}$), RteRIC $p = 4.8 \times 10^{-6}$ e Z4 ($p = 3.5 \times 10^{-5}$). Em todos os casos foi observado agora que esses atributos obtiveram os melhores valores de AUC nas suas respectivas análises, o que já era esperado, pois se trata de um experimento individual. Da mesma forma, o atributo IeRIC, que possuía os piores valores-p em todas as análises, agora obteve a segunda menor AUC na análise com espirometria normal e a menor de todas nas outras duas análises.

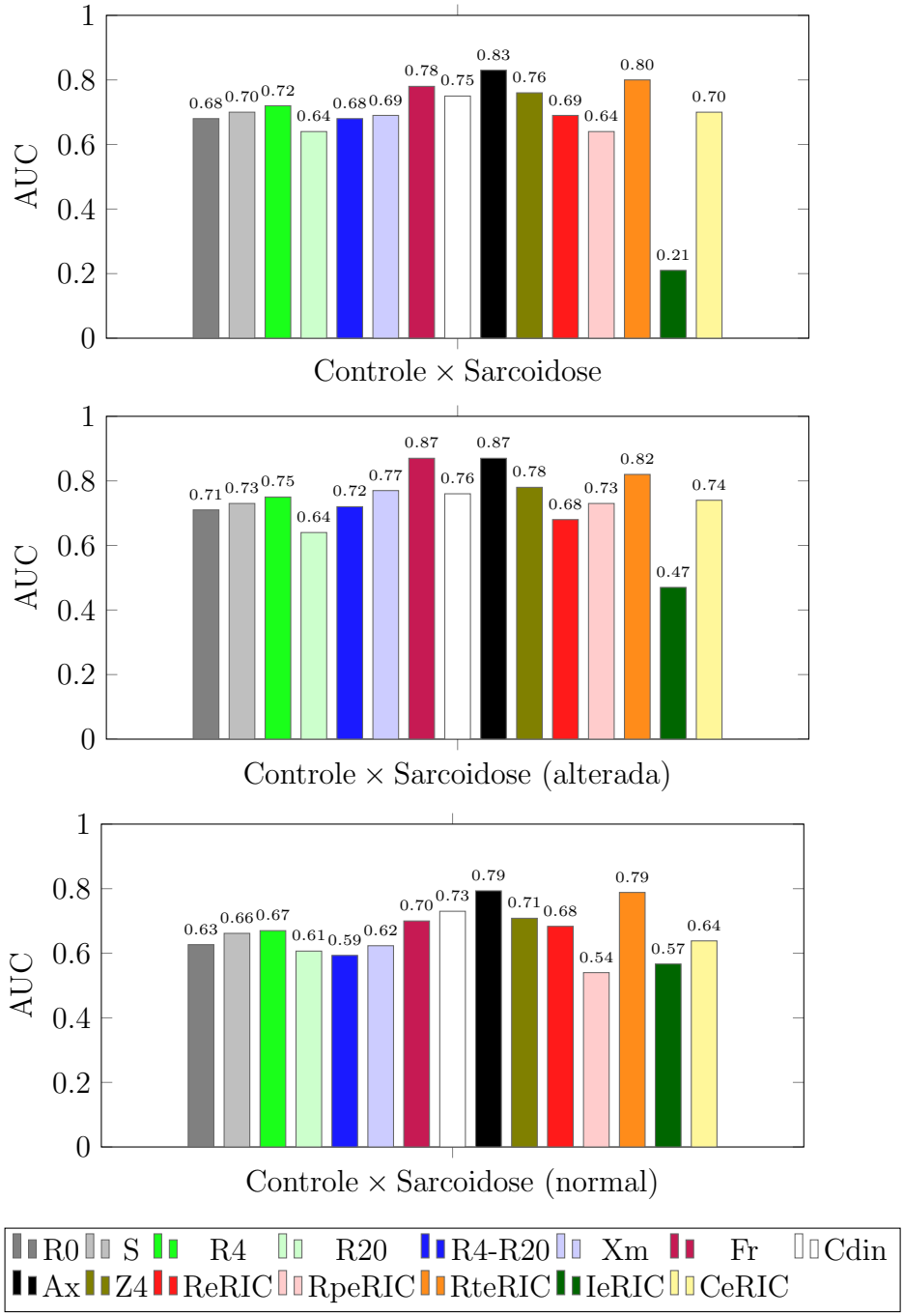
O trabalho mais recente encontrado na literatura [11] a analisar os atributos da FOT em indivíduos com sarcoidose apresentou como resultados que o melhor parâmetro para identificar alteração respiratória devido à sarcoidose em um indivíduo com espirometria alterada foi o Z4 (AUC = 0.84), seguido de R0 e Rm (AUC = 0.83), e em um indivíduo com espirometria normal foi o R0 (AUC = 0.81). Vale destacar que esse trabalho analisou apenas os atributos R0, S, Rm, Xm, Fr, Cdin e Z4. Agora, ao considerar outros atributos, percebe-se que alguns deles se mostram promissores, especialmente Ax e RteRIC.

8.3 Experimento com o Uso Direto de Classificadores

Neste experimento, todos os atributos foram utilizados nos seguintes algoritmos: K-Vizinhos mais Próximos (KNN), *Support Vector Machine* (SVM), *AdaBoost* (ADAB), *Random Forest* (RF), *LightGBM* (LGM), *XGBoost* (XGB), Regressor Logístico (LGR), Programação Genética (PG), Evolução Gramatical (GE) e Programação Genética Linear com Inspiração Quântica (PGLIQ).

O resultado das análises utilizando os dados normalizados, isto é os 16 atributos, pode ser visto na Figura 23a. Na análise do grupo de controle × indivíduos com sarcoidose os melhores resultados foram com XGB, SVM e LGM, que apresentaram AUC

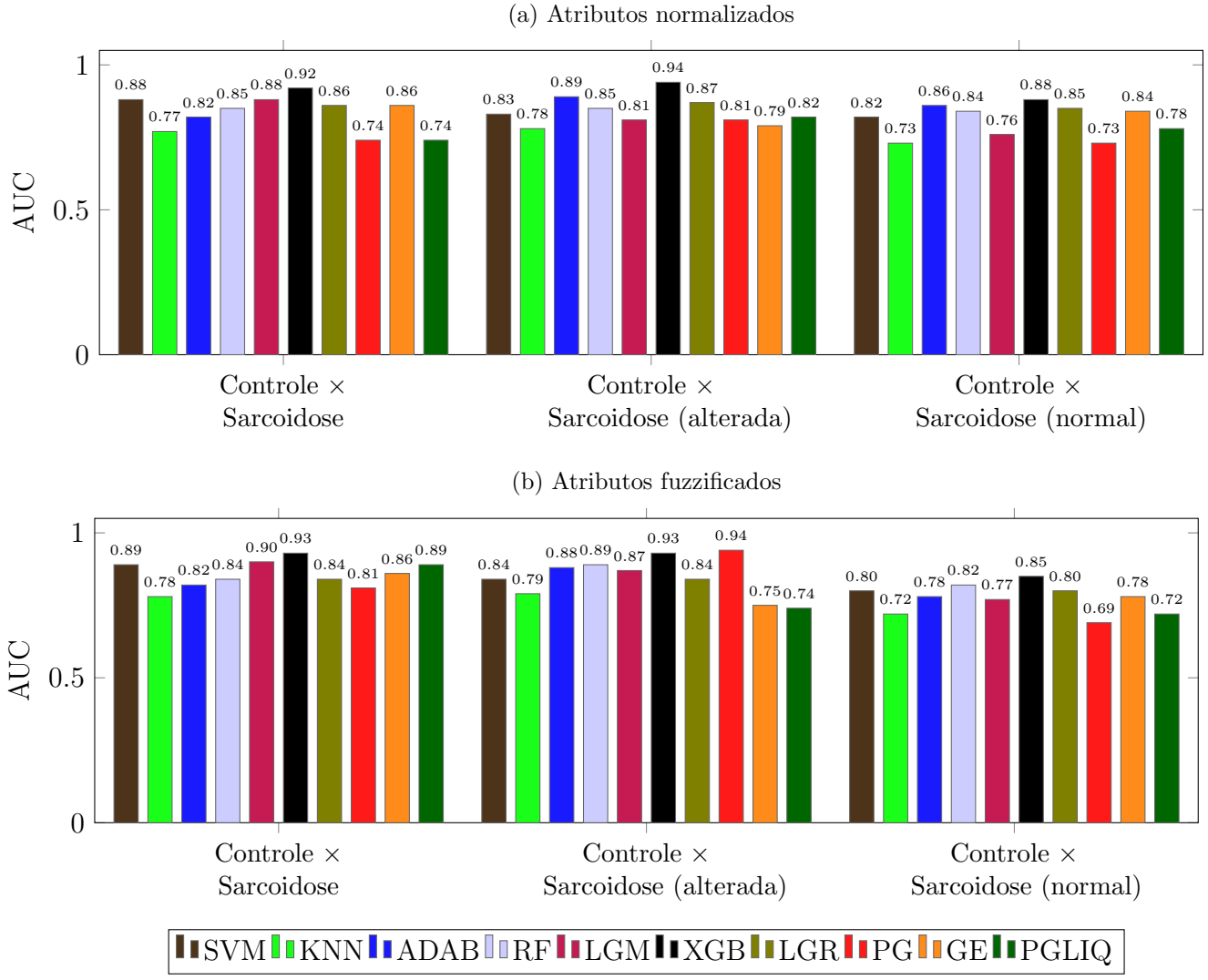
Figura 22: Desempenho individual dos atributos.



iguais a 0.92, 0.88 e 0.88, respectivamente. Enquanto que na análise com os indivíduos com sarcoidose e espirometria alterada os melhores resultados foram com XGB, ADAB e LGR, que apresentaram AUC iguais a 0.94, 0.89 e 0.87, respectivamente. Já na análise com espirometria normal, nenhum algoritmo apresentou AUC superior a 0.90, sendo os melhores XGB e ADAB, que apresentaram AUC iguais a 0.88 e 0.86, respectivamente.

Já o resultado das análises utilizando os dados fuzzificados, isto é os 48 atributos, pode ser visto na Figura 23b. Na análise do grupo de controle × indivíduos com sarcoi-

Figura 23: Desempenho com o uso direto de classificadores.



dose os resultados com XGB, SVM e LGM seguem dentre os melhores, ao apresentarem AUC iguais a 0.93, 0.89 e 0.90, respectivamente, ou seja, tiveram um pequeno aumento em relação ao mesmo experimento com os dados normalizados. Já na análise com os indivíduos com sarcoidose e espirometria alterada o melhor resultado também foi da PG com AUC igual a 0.94, seguido do XGB, ao apresentar AUC igual a 0.93. Vale destacar o aumento de 0.04 no resultado com RF e de 0.06 com LGM. Por fim na análise com espirometria normal, nenhum algoritmo apresentou AUC superior a 0.85, sendo XGB o melhor novamente, ao apresentar AUC igual a 0.85.

Quanto à PGLIQ, vale ressaltar o seu desempenho com os atributos fuzzificados na análise do grupo de controle × indivíduos com sarcoidose ao apresentar AUC igual a 0.89, bem próximo dos melhores desempenhos, com a vantagem de se tratar de um método

interpretável. Entretanto, o mesmo não ocorreu nas demais análises, que não utilizavam todo o conjunto de dados disponível.

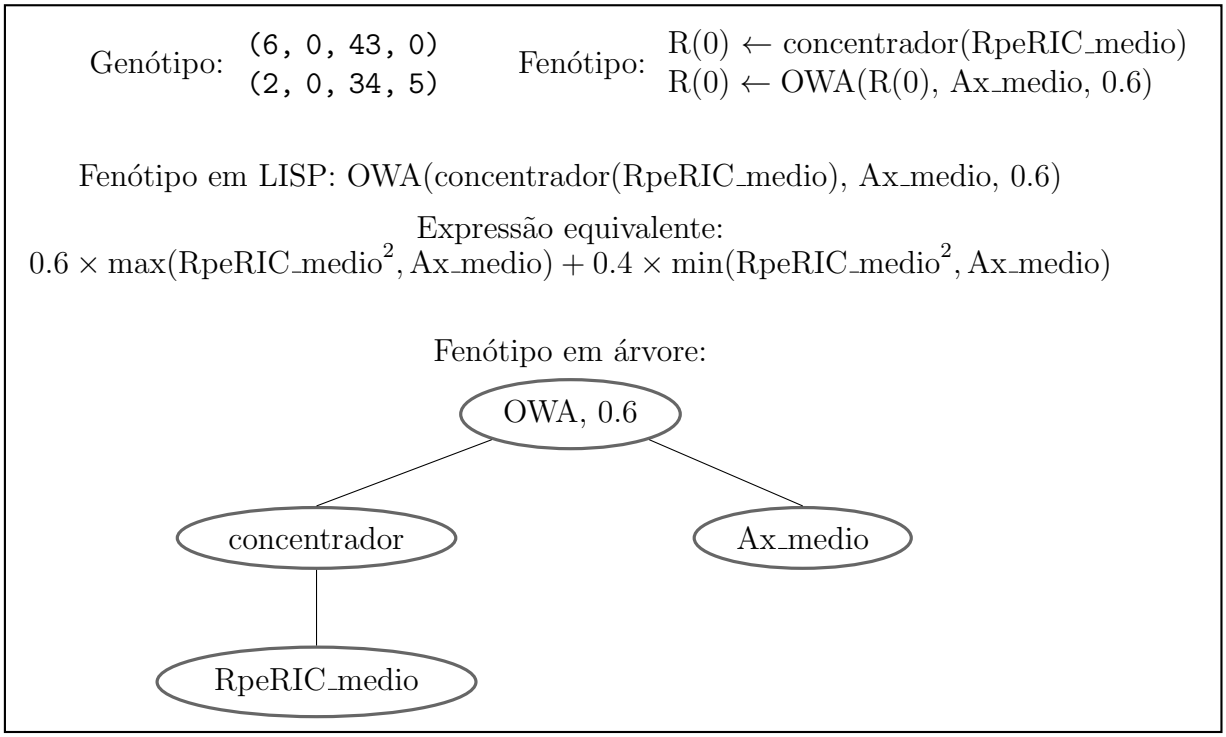
No que tange à interpretabilidade na PGLIQ, a Figura 24 apresenta um exemplo de um indivíduo final. Trata-se de um exemplo real obtido ao final de um experimento da PGLIQ na análise do grupo de controle × indivíduos com sarcoidose, que obteve AUC[9] de 0.93 na busca em grade, ao selecionar `generations = 10000` e `individual_length = 4`. Na figura, o genótipo é apresentado com apenas duas linhas, apesar do comprimento do indivíduo ser igual a 4, porque optou-se por não exibir os íntrons. Essas duas linhas são representadas em uma árvore de profundidade 3, o que facilita muito a interpretação do resultado. Propositadamente foi dada liberdade ao modelo para criar indivíduos bem maiores ao incluir na busca em grade `individual_length` = 4, 8, 16 ou 32, além de um número de registradores temporários[10] fixo igual a 6, entretanto o que se verificou na prática foram indivíduos bem pequenos. Na busca em grade foi selecionado `individual_length` igual a 4 em sete dos dez experimentos da análise do grupo de controle × indivíduos com sarcoidose, igual a 8 em dois e igual a 16 em um. E quanto ao comprimento efetivo dos indivíduos, isto é, sem os íntrons, o número médio de linhas dentre os dez experimentos foi de 2.5.

Ainda sobre o indivíduo da Figura 24, ao se verificar as probabilidades de cada atributo aparecer nele em uma próxima geração encontram-se os valores 29.1%, 26.0% e 22.6% para Ax_medio, RpeRIC_medio e Fr_alto, respectivamente. Para os demais atributos, cada probabilidade é inferior à probabilidade inicial dos mesmos, que é 1/48. Assim, os três atributos citados podem ser vistos como dominantes, entretanto Fr_alto não é utilizado neste indivíduo. Como sua probabilidade é muito alta, pode-se entender que em gerações passadas ele apareceu muitas vezes nesse mesmo indivíduo, sendo posteriormente substituído por um dos outros dois atributos, ou que aparecia em outro indivíduo que era o melhor anteriormente, e que foi superado pelo indivíduo apresentado na figura

[9]Conforme descrito no capítulo anterior, em cada análise os dados são divididos em dez pastas, que permitem a realização de dez experimentos, cada um utilizando nove pastas no treinamento e uma no teste. Por sua vez, em cada experimento, o conjunto de treinamento é subdividido em outras dez subpastas, a fim de realizar a busca em grade pelos hiperparâmetros do modelo. Assim, para cada combinação de hiperparâmetros, cada experimento é executado dez vezes, usando nove subpastas como treinamento e uma como teste. Por fim, a combinação de hiperparâmetros para aquele experimento é escolhida a partir da média na AUC nas dez subpastas de teste, e é medida a capacidade de generalização do modelo ao executá-lo na pasta de teste, que não foi usada em momento algum. O valor referenciado se refere à AUC obtida por meio da execução no conjunto de treinamento usando os melhores hiperparâmetros obtidos pela busca em grade.

[10]Esse hiperparâmetro é responsável por dar maior liberdade à criação de subárvores.

Figura 24: Exemplo de indivíduo final na PGLIQ.



em gerações mais recentes. Seja qual for o caso, percebe-se que um atributo pode ser dominante por diversas gerações, até que encontra outro ainda mais dominante, quando então começa a diminuir a sua frequência de aparecimento, podendo até cessar de aparecer. Entretanto, o fato de Fr_alto apresentar uma grande probabilidade de aparição no melhor indivíduo após 10000 gerações pode indicar que ele é relevante para a análise do grupo de controle × indivíduos com sarcoidose, ainda que ele não apareça no indivíduo final. Neste caso, seria menos relevante que os atributos que efetivamente aparecem no indivíduo, mas isso já é considerado propriamente por ele apresentar uma probabilidade menor que os indivíduos que aparecem. Isso é algo que não poderia ser observado no indivíduo final da PG ou da GE, por exemplo, porque não temos as informações das probabilidades nestes algoritmos.

Assim, a fim de contribuir para a interpretabilidade dos resultados, foram calculadas em cada análise as médias nos dez experimentos das probabilidades finais de cada atributo. A Tabela 16 exibe em ordem decrescente cada atributo cuja probabilidade final média é superior à probabilidade inicial de aparição de cada atributo, isto é, maior que 1/48. Pode-se interpretar isso como o resultado da seleção embutida de atributos feita pela PGLIQ. Na análise do grupo de controle × indivíduos com sarcoidose há menos atributos listados que nas demais, indicando que houve maior convergência na seleção,

Tabela 16: Probabilidades finais médias dos melhores atributos fuzzificados na PGLIQ.

Controle × sarcoidose		**Controle × sarcoidose (alterada)**		**Controle × sarcoidose (normal)**	
Atributo	**Percentual**	**Atributo**	**Percentual**	**Atributo**	**Percentual**
Ax_medio	30.4%	Fr_alto	19.9%	Ax_medio	18.8%
RpeRIC_medio	27.5%	Ax_medio	17.3%	RpeRIC_medio	17.0%
Fr_alto	7.4%	IeRIC_medio	13.1%	IeRIC_medio	5.9%
RteRIC_alto	2.4%	RpeRIC_medio	11.2%	Fr_alto	5.3%
ReRIC_alto	2.2%	Fr_medio	8.1%	Ax_alto	4.9%
		CeRIC_medio	4.1%	RteRIC_medio	4.9%
		RpeRIC_alto	3.4%	R4-R20_baixo	4.0%
		R4_alto	2.2%	R4-R20_alto	3.3%
		Cdin_baixo	2.1%	RteRIC_alto	2.8%
				Z4_alto	2.3%
				RpeRIC_alto	2.3%
				Xm_baixo	2.3%
				R4_alto	2.2%

o que se confirma, pois o resultado da PGLIQ nessa análise foi melhor que nas outras. Comparando a análise do grupo de controle × indivíduos com sarcoidose e espirometria alterada com a do grupo com espirometria normal pode-se fazer as seguintes observações:

- Ax_medio e RpeRIC_medio se destacam em ambas, podendo ser considerados indicadores de sarcoidose, o que se confirma, pois são os atributos que mais se distinguem na análise do grupo de controle × indivíduos com sarcoidose;

- Fr_alto é muito maior na análise com espirometria alterada, podendo ser um discriminante de alterações respiratórias. De fato, segundo a Tabela 13, esse atributo só apresentou diferenças significativas entre os conjuntos controle × sarcoidose (alterada);

- Por outro lado, IeRIC_medio é relevante em ambas as análises, apresentando-se muito maior na análise com espirometria alterada, entretanto só havia apresentado diferenças significativas entre os conjuntos controle × sarcoidose (normal), indicando que apesar desse atributo sozinho não ser um bom discriminante (na análise com espirometria alterada), pode contribuir muito em conjunto com outros;

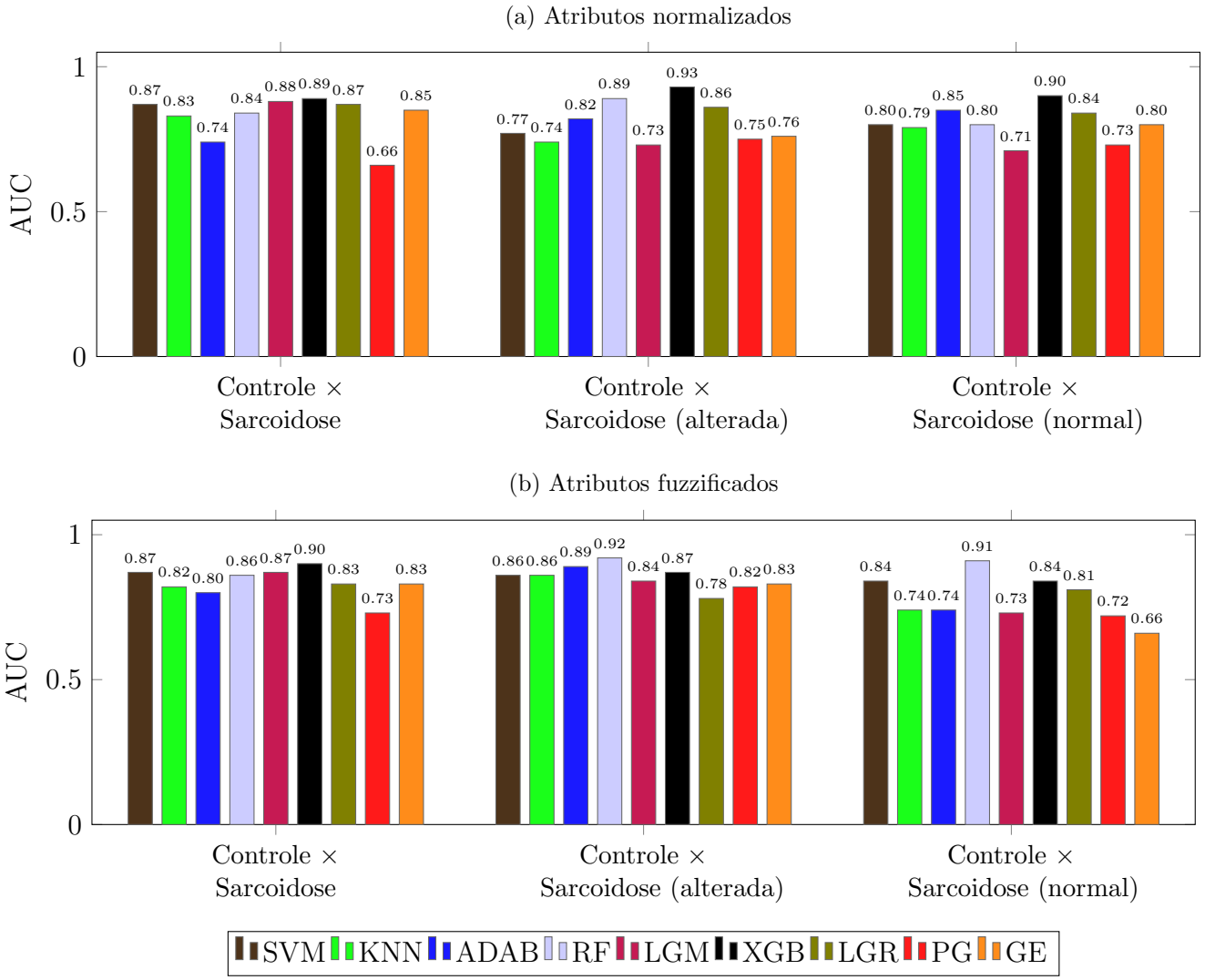
Uma última observação acerca da Tabela 16 é que diversos atributos com muitos zeros, conforme descrito na Tabela 14, se destacaram. Como por exemplo o já citado

Fr_alto, e inclusive o caso mais extremo, RpeRIC_alto, que possui apenas três valores diferentes de zero. Tal como descrito no final da Seção 8.1, isso provavelmente ocorre devido à disponibilidade das funções `max` e `OWA`.

8.4 Experimento com Seleção de Atributos

Neste experimento, foi feita a seleção de atributos para uso nos mesmos algoritmos do experimento da Seção 8.3, exceto a PGLIQ, devido ao tempo de execução, além de já possuir um seletor próprio.

Figura 25: Desempenho com seleção de atributos.



O resultado das análises utilizando seleção de atributos pode ser visto na Figura 25. Em relação ao desempenho com todos os atributos, as principais diferenças se referem ao desempenho do KNN na análise com espirometria alterada e RF com espirometria

normal, ambas usando atributos fuzzificados. Vale ressaltar que é o primeiro caso, cujo desempenho na análise com espirometria normal apresenta AUC igual ou superior a 0.90, o que ocorreu com o XGB usando atributos normalizados e com o RF usando atributos fuzzificados. Os demais resultados se mostram pouco diferentes em relação ao desempenho com todos os atributos.

Este experimento também pode contribuir para a interpretabilidade dos resultados ao se observar quais atributos são selecionados com maior frequência. Conforme citado no capítulo anterior, cada análise é feita com dez experimentos. Assim, com nove algoritmos e três análises cada, foram realizados um total de 270 experimentos para apresentar os resultados da Figura 25a e o mesmo número para a Figura 25b. A Tabela 17 exibe o percentual de aparição de cada atributo normalizado nos 270 experimentos respectivos, enquanto que a Tabela 18 exibe o percentual de aparição dos atributos fuzzificados mais frequentes em cada análise.

Tabela 17: Percentual de aparição dos atributos normalizados.

Atributo	**Controle × sarcoidose**	**Controle × (sarcoidose) (alterada)**	**Controle × (sarcoidose) (normal)**
R0	57%	43%	26%
S	0%	14%	24%
Rm	86%	54%	79%
R4	80%	76%	26%
R20	13%	21%	20%
R4-R20	53%	74%	63%
Xm	73%	69%	76%
Fr	89%	94%	80%
Cdin	80%	90%	84%
Ax	83%	29%	91%
Z4	50%	31%	44%
ReRIC	86%	36%	74%
RpeRIC	49%	66%	49%
RteRIC	100%	94%	99%
IeRIC	54%	61%	59%
CeRIC	71%	64%	80%

Usando-se atributos normalizados, observa-se na Tabela 17 na análise controle × sarcoidose que os mais frequentes foram RteRIC, ReRIC e Rm. Já na análise controle × sarcoidose e espirometria normal, os mais frequentes foram RteRIC, Ax e Cdin, que foram justamente os três melhores no experimento individual. Da mesma forma, na análise con-

Tabela 18: Maiores percentuais de aparição dos atributos fuzzificados.

Controle × sarcoidose		**Controle × sarcoidose (alterada)**		**Controle × sarcoidose (normal)**	
Atributo	**Percentual**	**Atributo**	**Percentual**	**Atributo**	**Percentual**
Fr_baixo	100%	Fr_baixo	99%	RteRIC_baixo	89%
RteRIC_baixo	99%	IeRIC_medio	93%	Ax_baixo	84%
IeRIC_baixo	97%	Fr_alto	86%	Fr_baixo	81%
RteRIC_medio	94%	CeRIC_alto	79%	IeRIC_medio	80%
Fr_alto	93%	IeRIC_baixo	66%	IeRIC_baixo	77%
Ax_baixo	90%	Ax_baixo	63%	CeRIC_alto	77%
IeRIC_medio	90%	CeRIC_medio	49%	Rm_baixo	76%
R0_baixo	83%	Ax_medio	43%	RpeRIC_medio	69%
Ax_medio	77%	RpeRIC_medio	39%	RpeRIC_baixo	67%
Rm_baixo	76%	ReRIC_medio	31%	Xm_baixo	66%
RpeRIC_medio	73%	R4-R20_medio	29%	Ax_medio	56%

trole × sarcoidose e espirometria alterada os mais frequentes foram RteRIC, Fr e Cdin, também dentre os melhores no individual, entretanto o atributo Ax que foi o melhor no individual, agora foi selecionado poucas vezes. Pode-se supor que um atributo, apesar de relevante individualmente, ao ser combinado com outros pouco contribua em conjunto. Pelo contrário, o atributo IeRIC que foi o pior individualmente em todas as análises, agora aparece muito acima dos menos frequentes nos três casos. Vale salientar que o número de atributos selecionados em cada experimento é um hiperparâmetro buscado pela busca em grade, que varia de 1 a 15 nos experimentos com dados normalizados e 1 a 47 naqueles com dados fuzzificados, exceto nos casos da PG e da GE, devido ao tempo de execução. Nestes casos, foram buscados 4, 8 ou 12 atributos normalizados e 12, 18 ou 24 atributos fuzzificados. Dessa forma, sem pré-estabelecer a quantidade de atributos, não são selecionados atributos além do mínimo necessário, de forma que observar a frequência de determinado atributo nos experimentos se torna algo mais relevante para a interpretabilidade dos resultados. Algo a ressalvar, é que o método *backward* pode acabar eliminando um atributo importante no início, quando a contribuição de cada um para o desempenho é menor.

Já nos experimentos com atributos fuzzificados, observa-se na Tabela 18 na análise controle × sarcoidose que os mais frequentes foram Fr_baixo, RteRIC_baixo e IeRIC_baixo. Voltando à Tabela 15, verifica-se que IeRIC_baixo sequer apresenta diferença significativa, o que mais uma vez leva à conclusão que determinados atributos podem ser irrelevantes

sozinhos, mas muito úteis em conjunto com outros. Na análise controle × sarcoidose e espirometria normal, os mais frequentes foram RteRIC_baixo e Ax_baixo, que na Tabela 13 estavam dentre aqueles com diferenças extremamente significativas ($p < 0.001$), seguidos do Fr_baixo, que estava dentre aqueles com diferenças muito significativas ($p < 0.01$). Por fim, na análise controle × sarcoidose e espirometria alterada os mais frequentes foram Fr_baixo, IeRIC_medio e Fr_alto. Voltando à Tabela 13, esse primeiro apresentou diferenças extremamente significativas ($p < 0.001$) e o terceiro apresentou diferenças muito significativas ($p < 0.01$), enquanto que o segundo sequer apresentou diferenças significativas ($p > 0.05$), levando outra vez à conclusão já descrita neste parágrafo.

Ainda sobre a Tabela 18, vale observar que, na análise controle × sarcoidose, muitos atributos apareceram em mais de 90% dos experimentos, enquanto que na análise com espirometria alterada apenas dois e com espirometria normal nenhum. Pode-se concluir que, pela análise controle × sarcoidose se tratar de uma análise menos específica, mais atributos são capazes de contribuir para a discriminação das amostras.

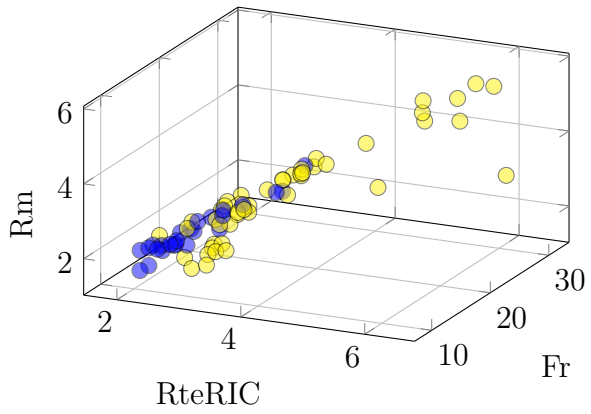
Outra utilidade deste experimento é a possibilidade de verificar visualmente se há diferenças entre os grupos, isto é, se é possível fazer a discriminação visual entre controle e sarcoidose com espirometria normal ou alterada usando os atributos disponíveis. A partir dos resultados apresentados na Tabela 17 e na Tabela 18 foram utilizados os três atributos mais frequentes em cada análise para a criação de gráficos 3D. O resultado pode ser visto na Figura 26. Em todas as análises, apesar da maioria dos dados estar distante daqueles de classes diferentes, muitos estão bem próximos, dificultando a concepção de uma superfície de separação simples. Com base nisso, verifica-se que este problema é difícil de ser resolvido com apenas três dados.

8.5 Experimento com Aumento de Dados

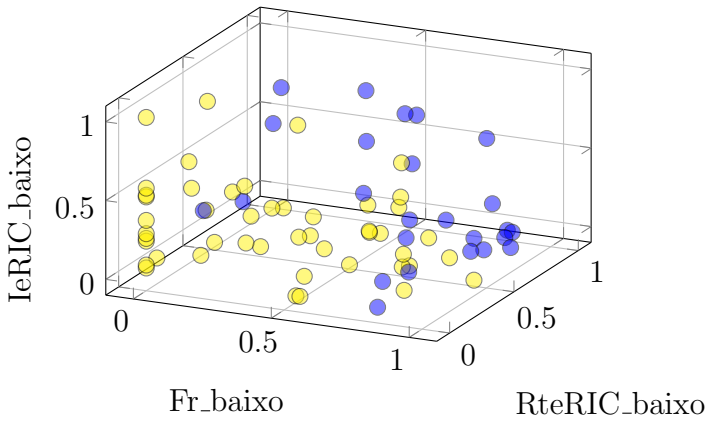
Neste experimento, todos os classificadores utilizados na Seção 8.3 foram usados novamente, desta vez com os dados aumentados. Na análise do grupo de controle × indivíduos com sarcoidose o aumento de dados foi feito de forma a equilibrar as classes, ou seja, ao invés de 72 dados, os classificadores utilizam 94 dados, sendo 22 deles sintéticos, todos pertencentes ao grupo controle. Já na segunda e na terceira análise, os dados foram aumentados de ambas as classes, conforme a busca em grade descrita na Seção 7.5, de forma a utilizar 50, 60, 80 ou 100 dados, metade de cada classe.

Figura 26: Representação dos dados com três atributos.

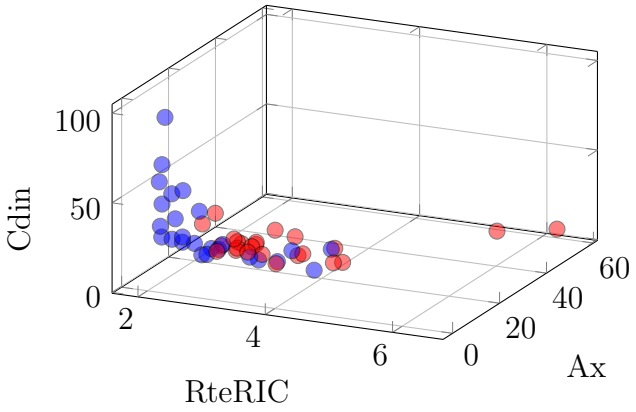
(a) Controle × Sarcoidose, com atributos normalizados.



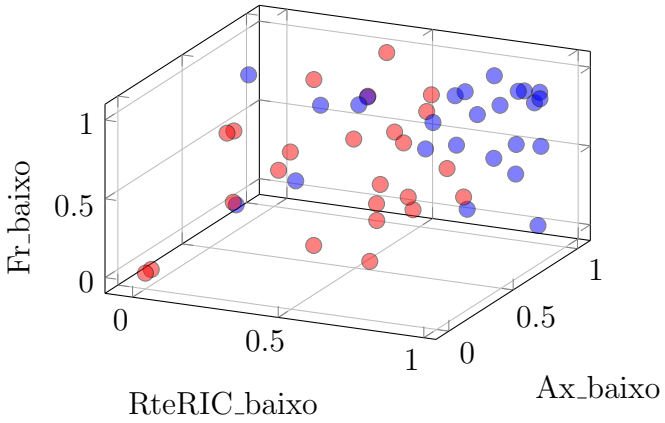
(b) Controle × Sarcoidose, com atributos fuzzificados.



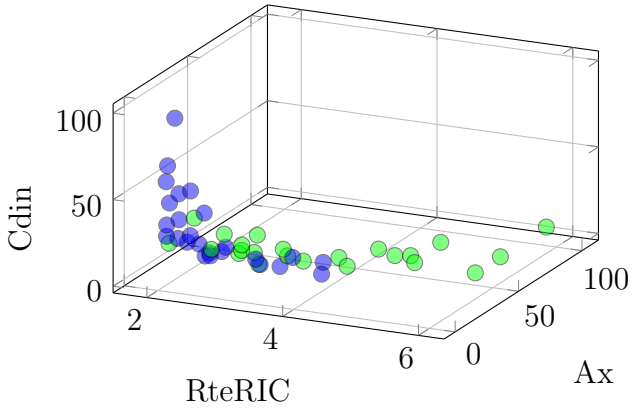
(c) Controle × Sarcoidose (normal), com atributos normalizados.



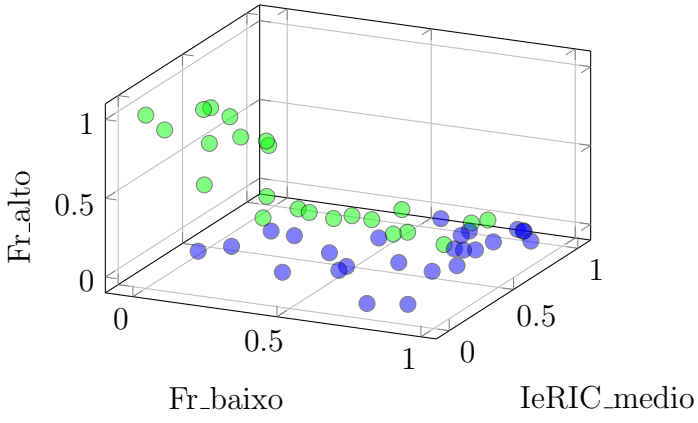
(d) Controle × Sarcoidose (normal), com atributos fuzzificados.



(e) Controle × Sarcoidose (alterada), com atributos normalizados.



(f) Controle × Sarcoidose (alterada), com atributos fuzzificados.



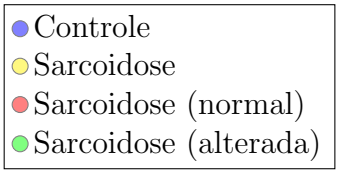
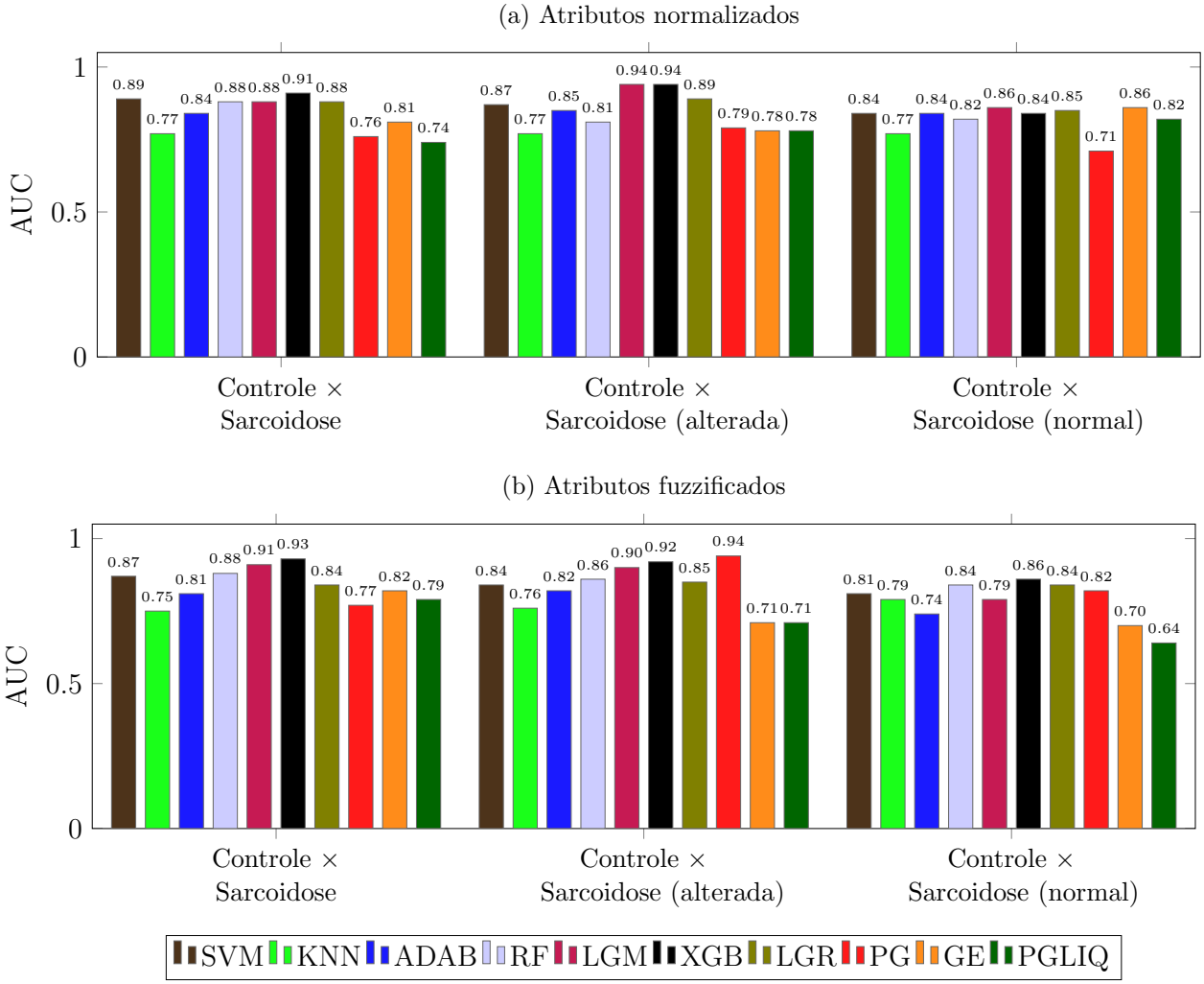


Figura 27: Desempenho com aumento de dados.



O resultado das análises utilizando aumento de dados pode ser visto na Figura 27. Com relação aos experimentos com os dados originais e todos os atributos, visto na Figura 23, poucas diferenças foram observadas. Cabe destacar somente a melhoria de desempenho do algoritmo LGM, que apresentou resultados superiores em todas as análises, especialmente na análise controle × sarcoidose (alterada) com atributos normalizados, que subiu de 0.81 para 0.94, e na análise controle × sarcoidose (normal), também com atributos normalizados, que subiu de 0.76 para 0.86.

8.6 Experimento com Construção de Atributos

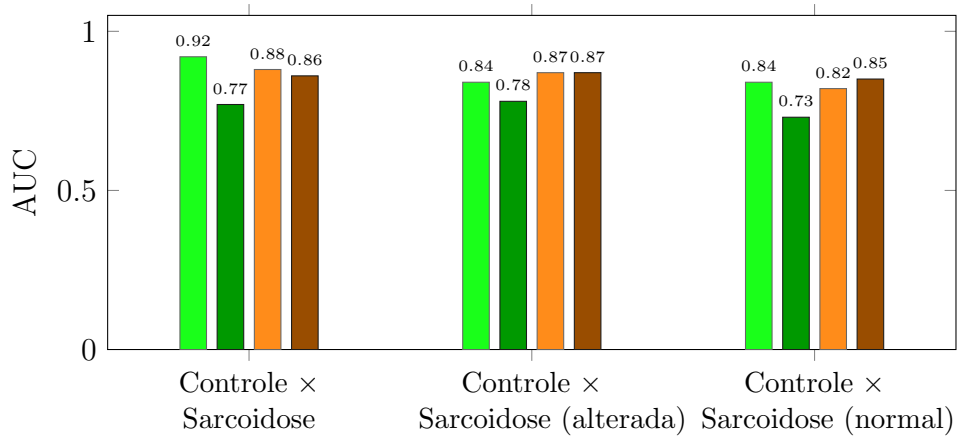
Neste experimento foi a feita a construção de atributos com PGLIQ para uso nos algoritmos KNN e LGR. A principal razão para a escolha desses métodos foi o tempo de execução, já que a cada geração da PGLIQ, para cada indivíduo é necessário executar

o KNN ou LGR, a fim de calcular a aptidão. Por outro lado são algoritmos que costumam trabalhar melhor com poucos atributos, mostrando-se uma escolha adequada, já que são construídos apenas dois ou três atributos, pois do contrário o modelo perderia interpretabilidade.

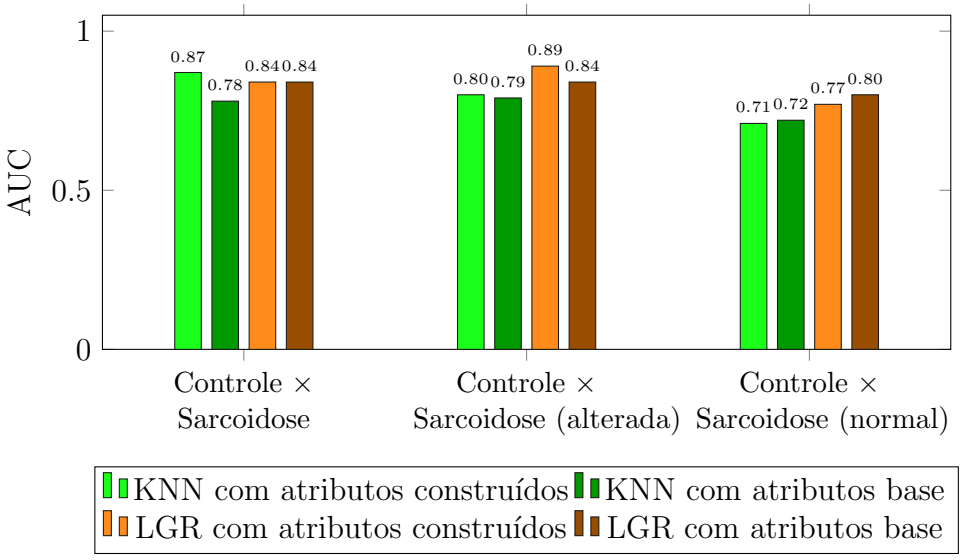
O resultado pode ser visto na Figura 28. Percebe-se que, quanto ao desempenho, não houve grandes melhorias com relação aos experimentos já realizados com LGR, enquanto que com KNN, houve grandes melhoras em alguns casos. Cabe destacar o resultado do KNN na análise controle × sarcoidose com atributos normalizados, que obteve alta acurácia (AUC = 0.92).

Figura 28: Desempenho com construção de atributos.

(a) Atributos normalizados



(b) Atributos fuzzificados

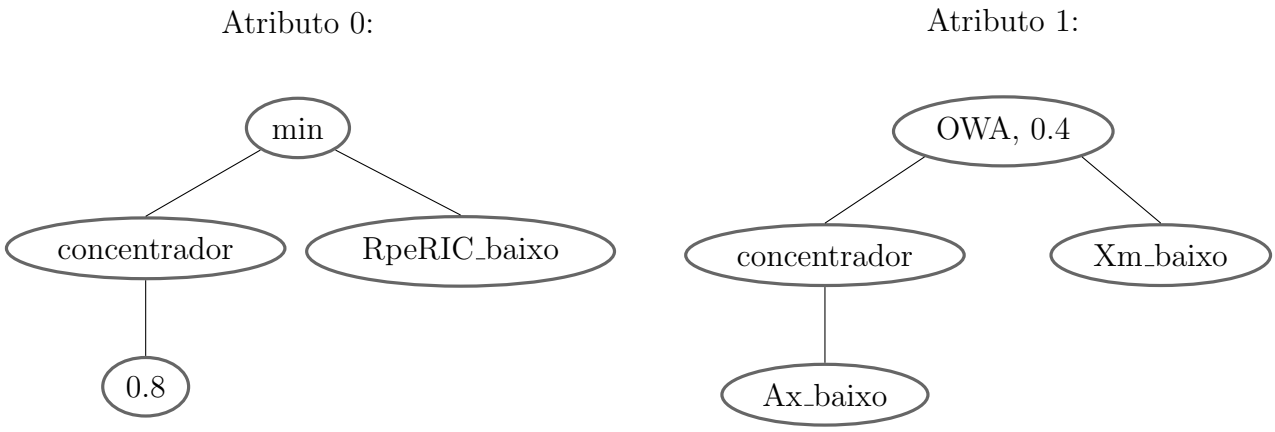


Tal como usar a PGLIQ diretamente para classificação, usá-la agora para construir atributos para outro algoritmo tem por principal objetivo trazer interpretabilidade para

os resultados. Assim, ainda que outros métodos tenham apresentado melhores desempenhos, em comparação direta com os algoritmos interpretáveis (PG, GE e PGLIQ para classificação), a construção com PGLIQ apresentou bons resultados. Obteve o segundo melhor desempenho[11], com AUC de 0.89, na análise controle × sarcoidose (alterada) com LGR e atributos fuzzificados, o melhor desempenho[12], com AUC de 0.92, na análise controle × sarcoidose com atributos normalizados e KNN e o segundo melhor desempenho[13], com AUC de 0.84, na análise controle × sarcoidose (normal) com atributos normalizados e KNN.

A Figura 29 apresenta um exemplo de um indivíduo final na construção de atributos com PGLIQ. Trata-se de um exemplo real obtido ao final de um experimento da PGLIQ na análise do grupo de controle × indivíduos com sarcoidose, que obteve AUC de 0.95 no conjunto de treinamento, após a busca em grade, ao selecionar `generations = 10000`, `individual_length = 8` e `n_new_features = 2`, ou seja, foram construídos dois atributos. Ambos usaram duas linhas cada, portanto as outras quatro linhas do indivíduo eram íntrons.

Figura 29: Exemplo de indivíduo final na construção de atributos com PGLIQ.



Ainda que os atributos sejam construídos a partir de expressões inteligíveis, tal como exemplificado na Figura 29, a interpretabilidade do resultado depende do tamanho dessas expressões. Quanto maior um indivíduo, mais complexas serão as expressões resultantes, e mais difícil será a tarefa de interpretar os resultados. Entretanto, o que

[11] o melhor desempenho foi da PG, com atributos fuzzificados, obtendo AUC de 0.94, conforme visto na Figura 23b

[12] o segundo melhor desempenho foi da PGLIQ, com atributos fuzzificados, obtendo AUC de 0.89, conforme visto na Figura 23b

[13] o melhor desempenho foi da GE, com atributos normalizados e dados aumentados, obtendo AUC de 0.86, conforme visto na Figura 27a

se vê na prática é que os experimentos apresentam resultados simples. Ainda que na busca em grade seja disponibilizado `individual_length` = 4, 8, 16 e até 32, na maioria dos casos o valor selecionado é 4 ou 8. De fato, resultados mais simples têm uma chance maior de apresentar boa capacidade de generalização, e a escolha dos hiperparâmetros por meio da busca em grade ajuda nessa tarefa. Além disso, essas linhas serão usadas para se construir dois ou três atributos, e diversas delas serão ocupadas por íntrons. Assim, considerando a construção para LGR com atributos normalizados, sem dados aumentados, as médias de linhas efetivas (sem íntrons) por atributo foram 2.17, 1.96 e 2.24 nas análises controle × sarcoidose, controle × sarcoidose (normal) e controle × sarcoidose (alterada), respectivamente. Já com atributos fuzzificados, as médias foram 2.61, 3.68 e 2.07, respectivamente.

Por fim, o fato de se construir dois ou três atributos em cada experimento também tem por objetivo a visualização desses novos atributos em duas ou três dimensões. Isso pode ser verificado na Figura 30 e na Figura 31, em que foram plotados os atributos construídos em alguns experimentos. Foi escolhido um resultado de cada análise, de cada quantidade de atributos contruídos (dois ou três) e de cada tipo de construção (com atributos normalizados ou fuzzificados). Inclusive o indivíduo da Figura 29 foi plotado na Figura 31b. Mesmo que a visualização de uma superfície de separação ainda não seja tão simples, agora é mais fácil do que nos gráficos da Figura 26. Isso ocorre principalmente porque dois ou três atributos construídos representam o problema como um todo, enquanto que dois ou três atributos selecionados representam apenas parte dele.

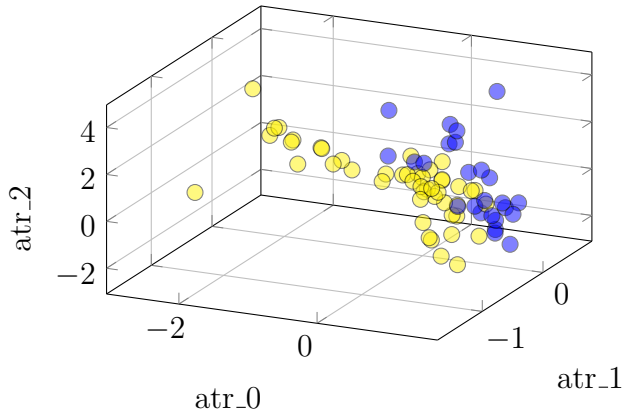
8.7 Principais Resultados

A Figura 32 resume os melhores resultados obtidos nos diferentes experimentos, exceto os da construção de atributos. O experimento 2 se refere ao uso direto dos classificadores, enquanto que o experimento 3 se refere à seleção de atributos, e por fim o experimentou 4 foi o que utilizou dados aumentados sinteticamente. O objetivo é comparar o desempenho dos métodos interpretáveis (PG, GE e PGLIQ) com os demais. Também para comparação, é apresentado em cada caso o melhor resultado no experimento individual, isto é, sem o uso de ML. Quanto à construção de atributos, conforme apresentado na seção anterior, os melhores resultados apresentaram AUC de 0.92, 0.89 e 0.84, respectivamente nas análises controle × sarcoidose, controle × sarcoidose (alterada) e controle ×

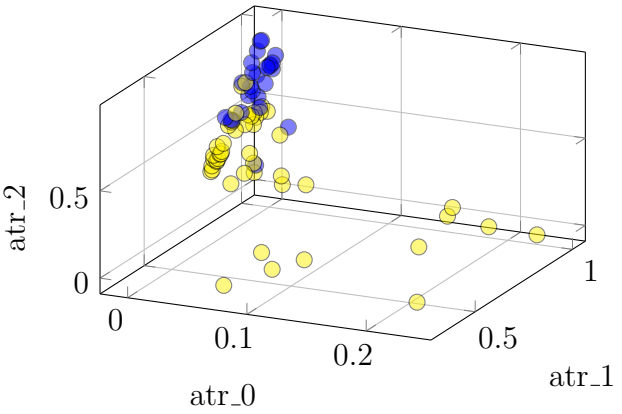
sarcoidose (normal). Em todas as análises, em ao menos um dos experimentos os métodos interpretáveis apresentaram resultados muito próximos aos demais métodos e superiores ao experimento individual.

Figura 30: Gráficos 3D com atributos construídos para LGR.

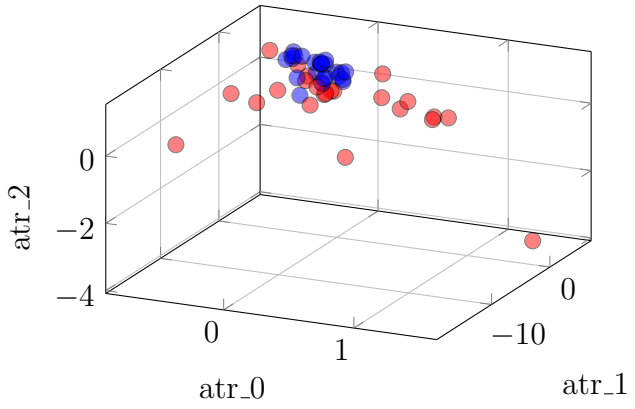
(a) Controle × Sarcoidose, com atributos normalizados.



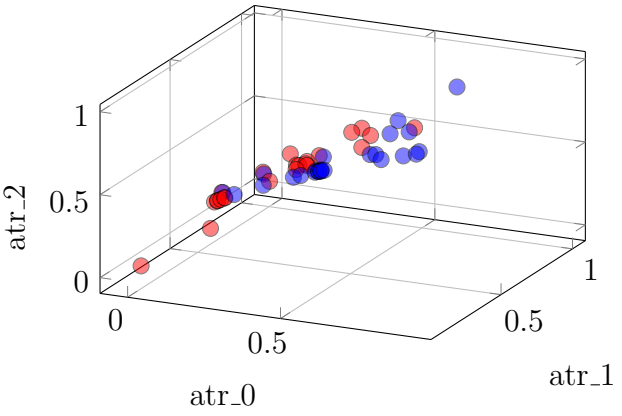
(b) Controle × Sarcoidose, com atributos fuzzificados.



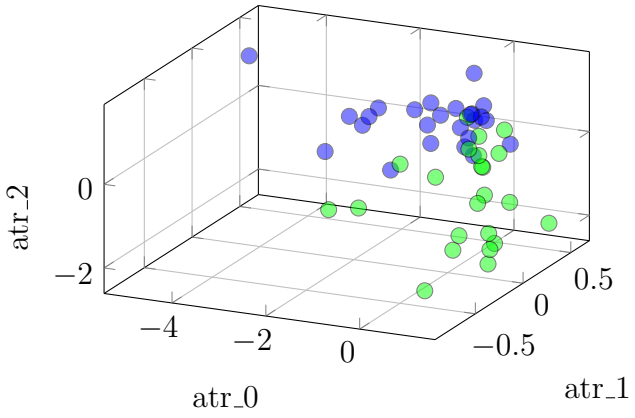
(c) Controle × Sarcoidose (normal), com atributos normalizados.



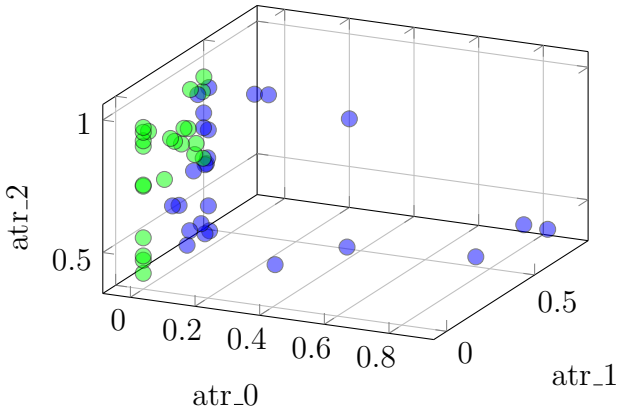
(d) Controle × Sarcoidose (normal), com atributos fuzzificados.



(e) Controle × Sarcoidose (alterada), com atributos normalizados.



(f) Controle × Sarcoidose (alterada), com atributos fuzzificados.



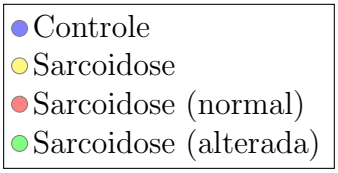
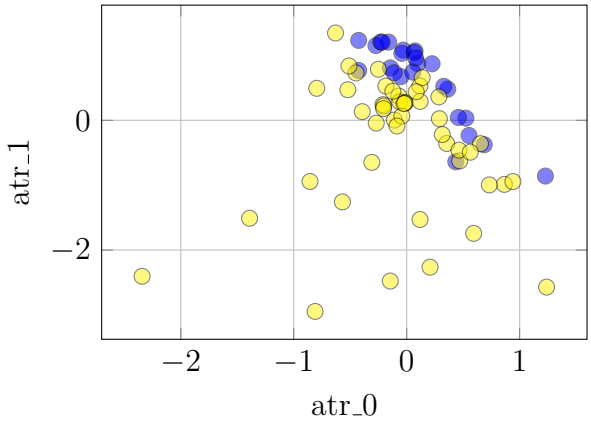
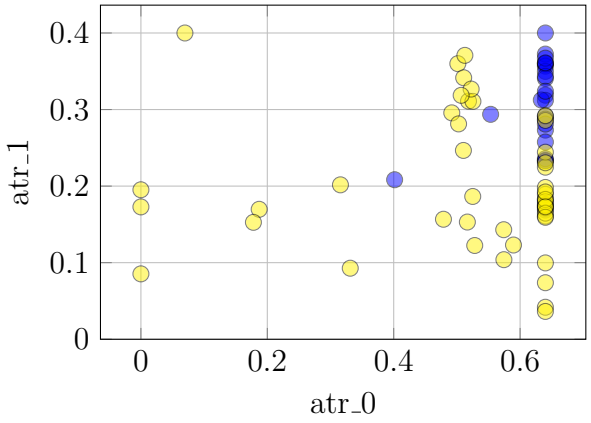


Figura 31: Gráficos 2D com atributos construídos para LGR.

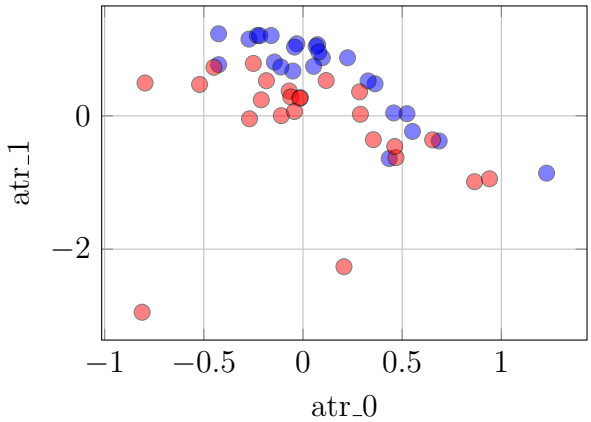
(a) Controle × Sarcoidose, com atributos normalizados.



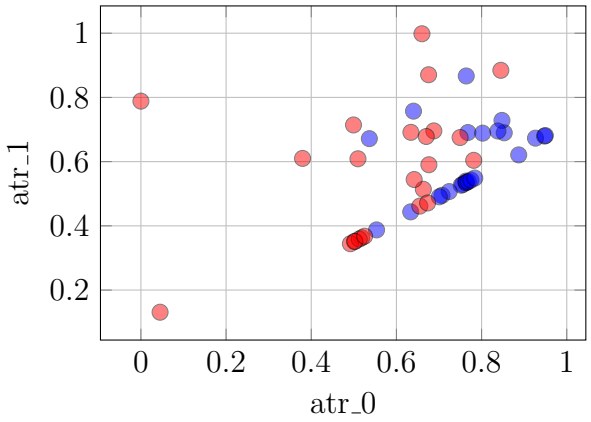
(b) Controle × Sarcoidose, com atributos fuzzificados.



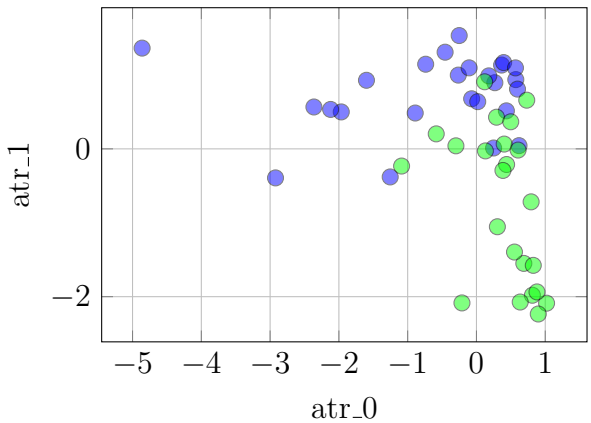
(c) Controle × Sarcoidose (normal), com atributos normalizados.



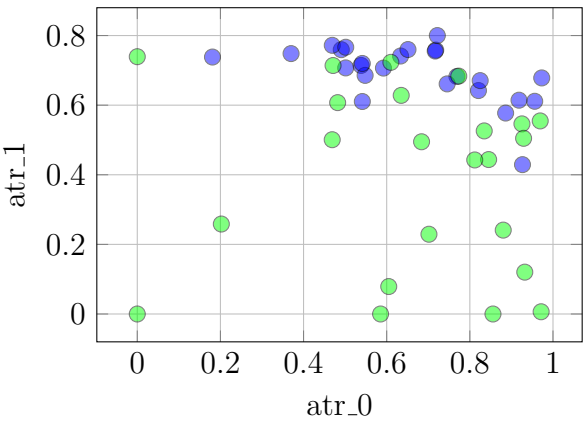
(d) Controle × Sarcoidose (normal), com atributos fuzzificados.



(e) Controle × Sarcoidose (alterada), com atributos normalizados.



(f) Controle × Sarcoidose (alterada), com atributos fuzzificados.



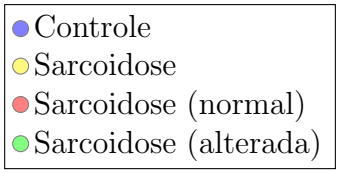
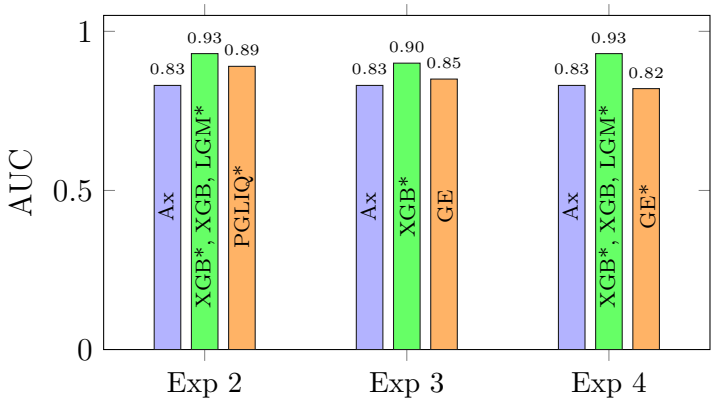
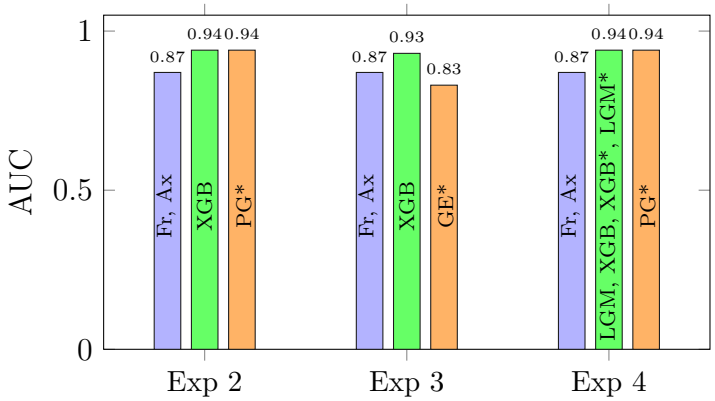


Figura 32: Melhores resultados.

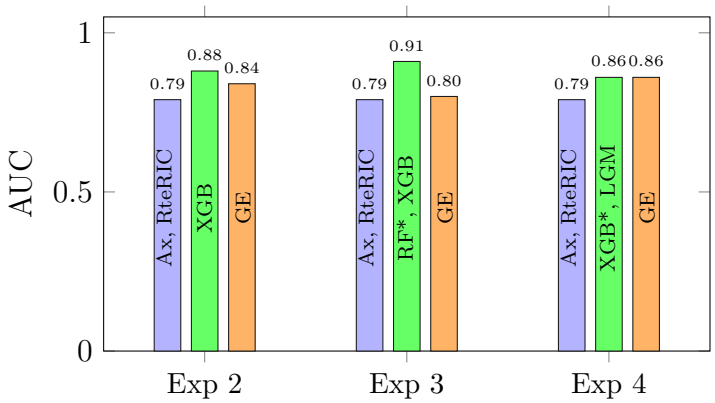
(a) Controle × Sarcoidose

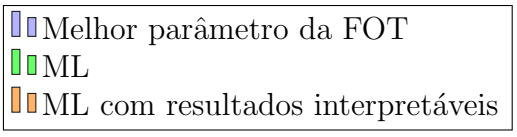


(b) Controle × Sarcoidose com espirometria alterada



(c) Controle × Sarcoidose com espirometria normal





*experimentos com atributos “fuzzificados”

CONCLUSÃO

Inicialmente foi avaliada a capacidade individual de cada atributo na classificação dos grupos, obtendo no máximo AUC igual a 0.83 na análise controle × indivíduos com sarcoidose, 0.87 na análise com espirometria alterada e 0.79 com espirometria normal. Com o uso de classificadores automáticos, os melhores desempenhos subiram para 0.93, 0.94 e 0.88, respectivamente, mostrando o quanto pode ser útil a aplicação de algoritmos de ML para combinar os atributos e consequentemente auxiliar no diagnóstico da sarcoidose. Isso é válido especialmente em pacientes com espirometria normal, pois conforme mostrado, a discriminação é mais difícil, sem qualquer atributo no experimento individual atingindo AUC de 0.80, enquanto que com os classificadores, diversos se mostraram capazes de ultrapassar esse valor. Essa superioridade ao experimento individual também se mantém mesmo ao considerar apenas algoritmos interpretáveis, cujos melhores desempenhos foram 0.89 na análise controle × indivíduos com sarcoidose, 0.94 na análise com espirometria alterada e 0.84 com espirometria normal.

Especificamente sobre a fuzzificação, o seu uso apresentou resultados, em geral, próximos em comparação aos experimentos em que ela não foi usada. Entretanto, há um peso maior na sua importância quanto à interpretabilidade dos resultados, pois conforme foi visto os termos *fuzzy* trazem mais informação intrinsecamente.

Na seleção de atributos o desempenho foi parecido com aquele no uso dos classificadores com todos os atributos, com destaque para a análise com espirometria normal que atingiu AUC de 0.90 com os atributos normalizados e 0.91 com os fuzzificados. Além disso, há o adicional de se observar quais atributos foram selecionados com maior frequência, o que traz interpretabilidade ao estudo da sarcoidose, especialmente ao perceber a contribuição de certos atributos que se mostraram irrelevantes no experimento individual, mas que apareceram diversas vezes na seleção, como por exemplo o IeRIC.

Cabe destacar a PGLIQ como melhor algoritmo interpretável na análise controle × indivíduos com sarcoidose, na qual obteve AUC igual a 0.89 quando usada para classificação. Além de apresentar a cada experimento uma expressão simples de ser interpretada que mostre como a classificação foi feita, com a PGLIQ ainda é possível verificar a distribuição final de probabilidades dos atributos, com o mesmo objetivo que ao observar a frequência com que os atributos aparecem na seleção.

Quando usada para construção de atributos, a PGLIQ também apresentou bons resultados, obtendo no máximo AUC igual a 0.92 na análise controle $\times$ indivíduos com sarcoidose, 0.89 na análise com espirometria alterada e 0.84 com espirometria normal. Além disso, nesses experimentos com construção de atributos, também foi possível visualizar graficamente o problema como um todo em apenas duas ou três dimensões.

Uma proposta para trabalhos futuros consiste em explorar a possibilidade da PGLIQ ser usada para classificação multiclasse. Tal como foi mostrado que diferentes expressões podem ser tomadas a partir de um único indivíduo em PGLIQ para construir diferentes atributos, da mesma forma se mostra viável tomar n expressões, com cada saída representando a possibilidade de uma classe em um problema com n classes. Assim, no caso do conjunto de dados usado neste trabalho, seria possível utilizar um único classificador para discriminar entre controle, sarcoidose com espirometria alterada e sarcoidose com espirometria normal.

Uma outra ideia se resume em utilizar a GE para construir atributos, algo que se mostra viável devido à natureza de suas expressões, com estrutura semelhante à PG ou à PGLIQ, e que ainda não tem sido explorado.

REFERÊNCIAS

[1] ROOS, N. et al. Thoracic sarcoidosis. *Radiologe*, v. 30, p. 581–590, 1990.

[2] BAUGHMAN, R. P.; CULVER, D. A.; JUDSON, M. A. A concise review of pulmonary sarcoidosis. *American Journal of Respiratory and Critical Care Medicine*, v. 183, 2010. Disponível em: <https://doi.org/10.1164/rccm.201006-0865CI>.

[3] RODRIGUES, M. et al. Diagnóstico tardio da sarcoidose é comum no brasil. *J. bras. pneumol.*, v. 39, 2013.

[4] FORGET, E.; MENZIES, D. Anti-infectives - adverse reactions to first-line antituberculosis drugs. *Expert opinion on drug safety*, v. 5, p. 231–49, 04 2006.

[5] NGO, C. et al. The volume-dependent forced oscillation technique. *IFAC-PapersOnLine*, v. 51, p. 373–377, 01 2018.

[6] DUBOIS, A. et al. Oscillation mechanics of lungs and chest in man. *J Appl Physiol*, 1956.

[7] DIONG, B. et al. Modeling human respiratory impedance. *Engineering in Medicine and Biology Magazine, IEEE*, v. 26, p. 48 – 55, 02 2007.

[8] MELO, P. L. et al. Avaliação das características resistivas do sistema respiratório de indivíduos portadores de silicose pela técnica de oscilações forçadas. *Jornal Brasileiro de Pneumologia*, 2006.

[9] MACLEOD, D.; BIRCH, M. Respiratory input impedance measurement: Forced oscillation methods. *Medical & biological engineering & computing*, v. 39, p. 505–16, 09 2001.

[10] MELO, P. L. et al. Evaluation of respiratory mechanics by forced oscillation technique in sarcoidosis. *Chest*, 2002.

[11] FARIA, A. et al. Assessment of respiratory mechanics in patients with sarcoidosis using forced oscillation: Correlations with spirometric and volumetric measurements and diagnostic accuracy. *Respiration*, v. 78, p. 93?104, 2009.

[12] AMARAL, J. et al. An improved method of early diagnosis of smoking-induced respiratory changes using machine learning algorithms. *Computer Methods and Programs in Biomedicine*, v. 112, p. 441–454, 2013.

[13] MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw Hill, 1997.

[14] PAVITHRA, D. A study on machine learning algorithm in medical diagnosis. *International Journal of Advanced Research in Computer Science*, v. 9, p. 42–46, 08 2018.

[15] KAVAKIOTIS, I. et al. Machine learning and data mining methods in diabetes research. *Computational and Structural Biotechnology Journal*, v. 15, 01 2017.

[16] RAMALINGAM, V. V.; DANDAPATH, A.; RAJA, M. Heart disease prediction using machine learning techniques: A survey. *International Journal of Engineering & Technology*, v. 7, p. 684, 03 2018.

[17] SHAMASNEH, A.; OBAIDELLAH, U. Artificial intelligence techniques for cancer detection and classification: Review study. *European Scientific Journal, ESJ*, v. 13, p. 342, 01 2017.

[18] MLODZINSKI, E.; STONE, D.; CELI, L. Machine learning for pulmonary and critical care medicine: A narrative review. *Pulmonary Therapy*, v. 6, 02 2020.

[19] AMARAL, J. et al. Machine learning algorithms and forced oscillation measurements applied to the automatic identification of chronic obstructive pulmonary disease. *Computer Methods and Programs in Biomedicine*, v. 105, p. 183–193, 2012.

[20] AMARAL, J. et al. Machine learning algorithms and forced oscillation measurements to categorize the airway obstruction severity in chronic obstructive pulmonary disease. *Computer Methods and Programs in Biomedicine*, v. 118, p. 186–197, 2015.

[21] AMARAL, J. et al. High-accuracy detection of airway obstruction in asthma using machine learning algorithms and forced oscillation measurements. *Computer Methods and Programs in Biomedicine*, v. 144, p. 113–125, 2017.

[22] AMARAL, J. et al. Differential diagnosis of asthma and restrictive respiratory diseases by combining forced oscillation measurements, machine learning and neuro-fuzzy classifiers. *Medical & Biological Engineering & Computing*, p. 1–19, 08 2020.

[23] GRUNEWALD, J. et al. Sarcoidosis. *Nature Reviews Disease Primers*, v. 5, p. 45, 07 2019.

[24] HUTCHINSON, J. Anomalous disease of skin of fingers (papillary psoriasis?). In: *Illustrations of clinical surgery*. [S.l.]: Churchill, J. and Churchill, A., 1875. p. 42–43.

[25] SPAGNOLO, P. et al. Pulmonary sarcoidosis. *The Lancet. Respiratory medicine*, v. 6, 04 2018.

[26] IANNUZZI, M.; RYBICKI, B.; TEIRSTEIN, A. Sarcoidosis. *New England Journal of Medicine*, v. 357, 2007.

[27] SWIGRIS, J. et al. Sarcoidosis-related mortality in the united states from 1988 to 2007. *Am J Respir Crit Care Med*, v. 183, 2011.

[28] WIJSENBEEK, M.; CULVER, D. Treatment of sarcoidosis. *Clinics in Chest Medicine*, v. 36, p. 751–767, 12 2015.

[29] PRASSE, A. The diagnosis, differential diagnosis, and treatment of sarcoidosis. *Deutsches Arzteblatt international*, v. 113, p. 565–574, 08 2016.

[30] RAMACHANDRAIAH, V.; ARONOW, W.; CHANDY, D. Pulmonary sarcoidosis: an update. *Postgraduate Medicine*, Taylor & Francis, v. 129, n. 1, p. 149–158, 2017. PMID: 27766929. Disponível em: <https://doi.org/10.1080/00325481.2017.1251818>.

[31] JUDSON, M. A. et al. The diagnostic pathway to sarcoidosis. *Chest*, v. 123, p. 406–412, 2003.

[32] MELO, P. Forced oscillation technique in pulmonology practice: principles and examples of potential applications. *Pulmão RJ*, v. 24, p. 42–48, 2015.

[33] BATES, J. H. et al. Oscillation mechanics of the respiratory system. In: _____. *Comprehensive Physiology*. American Cancer Society, 2011. p. 1233–1272. ISBN 9780470650714. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cphy.c100058>.

[34] FRY, D.; HYATT, R. Pulmonary mechanics. a unified analysis of the relationship between pressure, volume and gasflow in the lungs of normal and diseased human subjects. *The American journal of medicine*, v. 29, p. 672–89, 11 1960.

[35] OLIVEIRA, D. C. *Determinação da vida em fadiga de tecido pulmonar*. Dissertação (Mestrado) — Universidade Estadual do Ceará, 2010.

[36] SMITH, H.; REINHOLD, P.; GOLDMAN, M. Forced oscillation technique and impulse oscillometry. *European Respiratory Monograph*, v. 31, 04 2005.

[37] POZIN, N. *Multiscale lung ventilation modeling in health and disease*. Tese (Doutorado) — Université Pierre et Marie Curie, 2017.

[38] KACMAREK, R. M. et al. *Egan's Fundamentals of Respiratory Care*. [S.l.: s.n.], 2017.

[39] CAVALCANTI, J. et al. Using the forced oscillation technique to evaluate bronchodilator response in healthy volunteers and in asthma patients presenting a verified positive response. *Jornal brasileiro de pneumologia*, v. 32, p. 91–8, 04 2006.

[40] COSTA, G. M. d. et al. Bronchodilation in copd: beyond fev - the effect of albuterol on resistive and reactive properties of the respiratory system. *Jornal Brasileiro de Pneumologia*, scielo, v. 35, p. 325 – 333, 04 2009. ISSN 1806-3713.

[41] LIMA, A. et al. Forced oscillations and respiratory system modeling in adults with cystic fibrosis. *Biomedical engineering online*, v. 14, p. 7, 12 2015.

[42] MANGO, A. et al. Changes in respiratory mechanics with increasing degrees of airway obstruction in copd: Detection by forced oscillation technique. *Respiratory medicine*, v. 100, p. 399–410, 03 2006.

[43] KUBOTA, M. et al. Low frequency oscillometry parameters in copd patients are less variable during inspiration than during expiration. *Respiratory physiology & neurobiology*, v. 166, p. 73–9, 05 2009.

[44] SUZUKI, T. et al. Estimation using the impulse oscillation system in patients with pulmonary sarcoidosis. *Sarcoidosis, vasculitis, and diffuse lung diseases: official journal of WASOG / World Association of Sarcoidosis and Other Granulomatous Disorders*, v. 32, p. 144–150, 02 2015.

[45] FARIA, A. et al. Association of respiratory integer and fractional-order models with structural abnormalities in silicosis. *Computer Methods and Programs in Biomedicine*, v. 172, 02 2019.

[46] RIBEIRO, C. et al. Forced oscillation technique for early detection of the effects of smoking and copd: contribution of fractional-order modeling. *International Journal of Chronic Obstructive Pulmonary Disease*, Volume 13, p. 3281–3295, 10 2018.

[47] FARIA, A. et al. Forced oscillation, integer and fractional-order modeling in asthma. *Computer Methods and Programs in Biomedicine*, v. 128, 02 2016.

[48] WOO, T. et al. A comparison of various respiratory system models based on parameter estimates from impulse oscillometry data. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, v. 5, p. 3828–31, 02 2004.

[49] SCHMIDT, M. et al. Computer simulation of the measured respiratory impedance in newborn infants and the effect of the measurement equipment. *Medical engineering & physics*, v. 20, p. 220–8, 05 1998.

[50] MEAD, J. Mechanical properties of lungs. *Physiological Reviews*, v. 41, 04 1961.

[51] BATES, J. et al. Respiratory resistance with histamine challenge by single-breath and forced oscillation methods. *Journal of applied physiology (Bethesda, Md. : 1985)*, v. 61, n. 3, p. 873?880, September 1986. ISSN 8750-7587. Disponível em: <https://doi.org/10.1152/jappl.1986.61.3.873>.

[52] Lutchen, K. R.; Costa, K. D. Physiological interpretations based on lumped element models fit to respiratory impedance data: use of forward-inverse modeling. *IEEE Transactions on Biomedical Engineering*, v. 37, n. 11, p. 1076–1086, 1990.

[53] ASTER, R.; BORCHERS, B.; THURBER, C. *Parameter Estimation and Inverse Problems*. [S.l.: s.n.], 2013.

[54] MARSLAND, S. *Machine Learning: An Algorithmic Perspective, Second Edition*. 2nd. ed. [S.l.]: Chapman & Hall/CRC, 2014. ISBN 1466583282.

[55] DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2. ed. [S.l.]: Wiley, 2001. ISBN 978-0-471-05669-0.

[56] DIETTERICH, T. Ensemble methods in machine learning. In: . [S.l.: s.n.], 2000. p. 1–15. ISBN 3-540-67704-6.

[57] ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. *Learning From Data*. [S.l.]: AMLBook, 2012. ISBN 1600490069.

[58] FLACH, P. Roc analysis. In: _____. [S.l.: s.n.], 2010. p. 869–875. ISBN 9780387307688.

[59] MUSCHELLI, J. Roc and auc with a binary predictor: a potentially misleading metric. *Journal of Classification*, 12 2019.

[60] BRADLEY, A. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, v. 30, p. 1145 – 1159, 11 1996.

[61] FAWCETT, T. Introduction to roc analysis. *Pattern Recognition Letters*, v. 27, p. 861–874, 06 2006.

[62] KUNCHEVA, L. *Combining Pattern Classifiers: Methods and Algorithms*. [S.l.]: Wiley, 2014.

[63] ABE, S. *Support Vector Machines for Pattern Classification*. [S.l.: s.n.], 2010. ISBN 978-1-84996-097-7.

[64] CORTES, C.; VAPNIK, V. Support vector network. *Machine Learning*, v. 20, p. 273–297, 09 1995.

[65] DIETTERICH, T. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, v. 40, 12 2000.

[66] BREIMAN, L. Bagging predictors. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 24, n. 2, ago. 1996. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1018054314350>.

[67] SCHAPIRE, R. E. The strength of weak learnability. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 5, n. 2, p. 197?227, jul. 1990. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1022648800760>.

[68] BREIMAN, L. Random forests. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 45, n. 1, out. 2001. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

[69] SCHAPIRE, R. E. The boosting approach to machine learning: An overview. In: _____. *Nonlinear Estimation and Classification*. [S.l.]: Springer New York, 2003. p. 149–171.

[70] FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (ICML?96), p. 148?156. ISBN 1558604197.

[71] BAUER, E.; KOHAVI, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, v. 36, p. 1–38, 01 1996.

[72] HESS, A.; HESS, J. Logistic regression. *Transfusion*, v. 59, 06 2019.

[73] BOATENG, E. Y.; ABAYE, D. A review of the logistic regression model with emphasis on medical research. *Journal of Data Analysis and Information Processing*, v. 07, p. 190–207, 01 2019.

[74] DARWIN, C. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. [S.l.]: John Murray, 1859.

[75] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. [S.l.]: MIT press, 1992.

[76] LANGDON, W. B.; POLI, R.; MCPHEE, N. F. *A Field Guide to Genetic Programming*. [S.l.]: Lulu Enterprises, 2008.

[77] KOZA, J. R. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 2010.

[78] HEY, T. Quantum computing: an introduction. *Computing & Control Engineering Journal*, 1999.

[79] MOORE, M.; NARAYANAN, A. Quantum-inspired computing. *Computing & Control Engineering Journal*, 1999.

[80] DIAS, D. M. *Programação Genética Linear com Inspiração Quântica*. Tese (Doutorado) — PUC-Rio - Pontifícia Universidade Católica do Rio de Janeiro, 2010.

[81] O'NEILL, M.; RYAN, C. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. [S.l.: s.n.], 2003. ISBN 1402074441.

[82] RYAN, C. et al. Grammatical evolution: Evolving programs for an arbitrary language. In: *Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming*. [S.l.]: Springer-Verlag, 1998. p. 83–95.

[83] MCKAY, R. I. et al. Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, Kluwer Academic Publishers, USA, v. 11, n. 3?4, p. 365?396, set. 2010. ISSN 1389-2576. Disponível em: <https://doi.org/10.1007/s10710-010-9109-y>.

[84] FENTON, M. et al. Ponyge2: Grammatical evolution in python. 03 2017.

[85] GOMIDE, F.; GUDWIN, R. Modelagem, controle, sistemas e lógica fuzzy. *SBA Controle & Automação*, v. 4, p. 97–115, 1994.

[86] TANSCHEIT, R.; VARGENS, J.; VELLASCO, M. Previsão de produção agrícola baseada em regras linguísticas e lógica fuzzy. *Revista Controle & Automação*, v. 14, p. 114–120, 2003.

[87] HUANG, Z.; GEDEON, T.; NIKRAVESH, M. Pattern trees induction: A new machine learning method. *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, v. 16, 2008.

[88] VIRGOLIN, M.; ALDERLIESTEN, T.; BOSMAN, P. A. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation*, Elsevier BV, v. 53, Mar 2020. ISSN 2210-6502. Disponível em: <http://dx.doi.org/10.1016/j.swevo.2019.100640>.

[89] LIU, H.; MOTODA, H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. [S.l.: s.n.], 1998. ISBN 978-1-4613-7622-4.

[90] GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, JMLR.org, v. 3, n. null, p. 1157?1182, mar. 2003. ISSN 1532-4435.

[91] SHARMA, A. et al. A strategy to select suitable physicochemical attributes of amino acids for protein fold recognition. *BMC bioinformatics*, v. 14, p. 233, 07 2013.

[92] TRAN, B.; XUE, B.; ZHANG, M. Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recognition*, v. 93, 05 2019.

[93] CHAWLA, N. et al. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, v. 16, p. 321–357, 01 2002.

[94] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[95] MENDENHALL, W. M.; SINCICH, T. L. *Statistics for Engineering and the Sciences*. 6th. ed. USA: Chapman and Hall, 2015. ISBN 1498728855.

[96] BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. [S.l.: s.n.], 2013. p. 108–122.