Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Programa de Pós-Graduação em Modelagem Computacional

Johnathan Mayke Melo Neto

Modelos Híbridos para Construção de Redes Neurais Artificiais via Programação Genética Cartesiana

Juiz de Fora

2019

Johnathan Mayke Melo Neto

Modelos Híbridos para Construção de Redes Neurais Artificiais via Programação Genética Cartesiana

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas Computacionais Aplicados, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional.

Orientador: Prof. D.Sc. Heder Soares Bernardino

Coorientador: Prof. D.Sc. Helio José Corrêa Barbosa

Juiz de Fora

2019

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Melo Neto, Johnathan Mayke.
Modelos Híbridos para Construção de Redes Neurais Artificiais via Programação Genética Cartesiana / Johnathan Mayke Melo Neto. – 2019.
105 f. : il.

Orientador: Prof. D.Sc. Heder Soares Bernardino
Coorientador: Prof. D.Sc. Helio José Corrêa Barbosa
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Modelagem Computacional, 2019.

1. Hibridização. 2. Neuroevolução. 3. Classificação. I. Bernardino, Heder Soares, orient. II. Barbosa, Hélio José Corrêa, coorient. III. Título.

Johnathan Mayke Melo Neto

Modelos Híbridos para Construção de Redes Neurais Artificiais via Programação Genética Cartesiana

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas Computacionais Aplicados, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional.

Aprovada em: 22 de Fevereiro de 2019

BANCA EXAMINADORA

Prof. D.Sc. Heder Soares Bernardino - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Helio José Corrêa Barbosa - Coorientador
Universidade Federal de Juiz de Fora
Laboratório Nacional de Computação Científica

Prof. D.Sc. Leonardo Goliatt da Fonseca
Universidade Federal de Juiz de Fora

Prof. D.Sc. Douglas Adriano Augusto
Fundação Oswaldo Cruz

Dedico este trabalho à minha família, fonte inesgotável de amor.

AGRADECIMENTOS

A Deus, pela sabedoria a mim concedida para realizar este trabalho.

Aos meus pais Joseli e Andréia, minha irmã Júhlya, e minha avó Rayl, por todo amor e paciência para comigo ao longo dessa jornada.

Aos amigos e irmãos na fé, pelas orações.

Aos meus orientadores, Prof. Heder e Prof. Helio, pelos valiosos conselhos e pelo exemplo de profissionalismo durante todo o desenvolvimento deste trabalho.

A todos os professores que fizeram parte da minha trajetória acadêmica, desde o ensino básico até a pós-graduação.

Ao PPGMC/UFJF, por fornecer toda a estrutura para a realização deste trabalho.

À FAPEMIG, CAPES, PPGMC/UFJF e CNPq pelo apoio financeiro.

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

"*When a new building block is discovered,*
the result is usually a range of innovations."
John Henry Holland

RESUMO

Na área de aprendizado de máquina, o problema de classificação de dados consiste em rotular corretamente instâncias desconhecidas com base nos rótulos de um conjunto de instâncias conhecidas. Um importante método para a resolução de problemas de classificação de dados é denominado redes neurais artificiais (RNAs). As RNAs são métodos computacionais bioinspirados, cujos parâmetros devem ser corretamente ajustados a fim de resolver uma dada tarefa de aprendizado. A topologia e os pesos são parâmetros determinantes para o desempenho desses métodos. Apesar da importância das RNAs em situações práticas, e dos diversos trabalhos disponíveis na literatura, o ajuste de seus parâmetros ainda é considerado um problema atual. Portanto, o desenvolvimento de estratégias para auxiliar os usuários durante a modelagem das RNAs é relevante. Uma dessas estratégias consiste em utilizar algoritmos evolutivos (EAs) para otimizar os parâmetros das RNAs. A combinação de RNAs e EAs é denominada neuroevolução. Este trabalho propõe novos métodos neuroevolutivos híbridos baseados em programação genética cartesiana (CGP) para a construção de RNAs. Os métodos realizam o desacoplamento dos processos de otimização da topologia e dos pesos para gerar as RNAs. A otimização da topologia é feita pela CGP. Para a otimização dos pesos, duas técnicas são utilizadas separadamente: (i) evolução diferencial (DE), e (ii) *backpropagation* (BP). Nos experimentos computacionais, os modelos gerados foram aplicados a dezoito problemas de classificação, utilizando bases de dados *benchmark* da literatura. Há experimentos com bases balanceadas e outros com bases desbalanceadas. Os modelos foram submetidos a três estudos de desempenho. O primeiro estudo avaliou o desempenho dos modelos utilizando a acurácia como função objetivo. O segundo estudo avaliou o desempenho dos modelos utilizando o erro quadrático médio como função objetivo. O terceiro estudo utilizou os modelos para avaliar o desempenho de quatro funções objetivo distintas: acurácia, G-mean, F_β-score, e área abaixo da curva ROC. Os resultados mostraram a superioridade das propostas quando comparadas a técnicas alternativas existentes na literatura.

Palavras-chave: Neuroevolução. Hibridização. Programação Genética Cartesiana. Classificação de Dados.

ABSTRACT

In machine learning, the problem of data classification consists of correctly labeling unknown instances based on the labels of a set of known instances. An important method for solving data classification problems is called artificial neural networks (ANNs). ANNs are bioinspired computational methods, whose parameters must be correctly adjusted in order to solve a given learning task. Topology and weights are determining parameters for the performance of these methods. Despite the significance of ANNs in practical situations, and the several works available in the literature, to adjust its parameters remains as a current problem. Hence, the development of strategies to assist users during ANN modeling is relevant. One of these strategies is to use evolutionary algorithms (EAs) to optimize ANN parameters. The combination of ANNs and EAs is called neuroevolution. This work proposes new hybrid neuroevolutionary methods based on cartesian genetic programming (CGP) for the construction of ANNs. The methods carry out a decoupled optimization of the topology and the weights to generate the ANNs. Topology optimization is performed by CGP. For the weights optimization, two techniques are used separately: (i) differential evolution (DE), and (ii) backpropagation (BP). In the computational experiments these models were applied to eighteen data classification problems using benchmark datasets from the literature. There are experiments with balanced datasets and others with unbalanced datasets. The models were submitted to three performance studies. The first study evaluated the performance of the models using accuracy as the objective function. The second one evaluated the performance of the models using the mean square error as the objective function. The third study used the models to evaluate the performance of four distinct objective functions: accuracy, G-mean, F_β-score, and area under the ROC curve. The results showed the superiority of the proposals when compared to alternative techniques from the literature.

Key-words: Neuroevolution. Hybridization. Cartesian Genetic Programming. Data Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	Área Abaixo da Curva ROC
CGP	Programação Genética Cartesiana
CGPANN	Programação Genética Cartesiana de Redes Neurais Artificiais
DE	Evolução Diferencial
EE	Esforço de Avaliação
ES	Estratégia Evolutiva
FNR	Taxa de Falsos Negativos
FPR	Taxa de Falsos Positivos
GP	Programação Genética
RNA	Rede Neural Artificial
ROC	Característica de Operação do Receptor
TNR	Taxa de Verdadeiros Negativos
TPR	Taxa de Verdadeiros Positivos
TRN	Conjunto de Treinamento
TST	Conjunto de Teste
VLD	Conjunto de Validação

SUMÁRIO

1 INTRODUÇÃO

O presente capítulo apresenta uma visão geral desta dissertação, apresentando a motivação, os objetivos do trabalho, as principais contribuições realizadas e um resumo dos próximos capítulos. A Seção 1.1 introduz toda motivação necessária para uma compreensão inicial da dissertação. A Seção 1.2 traz os objetivos gerais e específicos do trabalho. A Seção 1.3 descreve as principais contribuições realizadas. Por fim, a Seção 1.4 apresenta uma visão geral dos próximos capítulos.

1.1 MOTIVAÇÃO

O aprendizado de máquina[1] é uma área da inteligência artificial que consiste no desenvolvimento de modelos computacionais capazes de otimizar certo critério de desempenho, melhorando automaticamente a qualidade de suas soluções a partir de dados de exemplo ou experiências passadas, sem que os modelos sejam programados explicitamente [2]. Considere um modelo composto por certa quantidade de parâmetros. O aprendizado desse modelo consiste na execução de um algoritmo para ajustar seus parâmetros, utilizando dados de treinamento ou experiências anteriores. Com esses ajustes, o desempenho do modelo tende a melhorar progressivamente. Em geral, os modelos podem ser classificados como preditivos ou descritivos. Os modelos preditivos são usados para fazer previsões no futuro, e os modelos descritivos são usados quando se quer obter conhecimento dos dados. Nas últimas décadas, a área de aprendizado de máquina tem ganhado destaque na literatura [2, 3], o que tem causado o desenvolvimento de algoritmos cada vez mais eficientes para realizar tarefas complexas de aprendizado. Alguns exemplos de tarefas de aprendizado são: previsão de acidentes de trânsito [4], detecção de doenças cancerígenas [5], sistema de direção autônoma de carros [6], e reconhecimento facial [7].

O problema de classificação de dados é uma área do aprendizado de máquina que consiste em rotular corretamente instâncias desconhecidas com base nos rótulos de um conjunto de instâncias conhecidas. Um importante método para a resolução de problemas de classificação é denominado redes neurais artificiais (RNAs) [8]. As RNAs são modelos computacionais bioinspirados, capazes de realizar com sucesso tarefas complexas de aprendizado, considerando que a quantidade de dados disponíveis para treinamento seja suficientemente grande. Apesar da importância das RNAs em situações práticas e dos diversos trabalhos disponíveis na literatura, o ajuste correto de seus parâmetros ainda é considerado um problema atual e relevante. A topologia, os pesos associados às conexões, e as funções de transferência das RNAs são parâmetros determinantes para o desempenho desses modelos [9]. No entanto, encontrar os valores ideais para esses parâmetros é uma tarefa trabalhosa, que muitas vezes depende do conhecimento de um projetista humano.

[1] Do inglês, *Machine Learning*.

Por isso, diversas estratégias têm sido desenvolvidas para garantir um ajuste automático dos parâmetros das RNAs.

Uma dessas estratégias consiste em utilizar algoritmos evolutivos (EAs)[2] [10] para otimizar os parâmetros das RNAs. Os EAs são métodos estocásticos de otimização baseados na evolução darwiniana. Essa combinação de RNAs e EAs é denominada neuroevolução [11], e suas principais características são descritas a seguir [12]. A neuroevolução pode ser aplicada à criação de redes *feedforward* e redes recorrentes, bem como pode ser utilizada em tarefas de aprendizado supervisionado e aprendizado por reforço. Além disso, esses algoritmos não impõem restrições quanto à diferenciabilidade da função objetivo. Por fim, a neuroevolução pode ser usada para determinar as funções de transferência de cada nó, bem como pode ser aplicada para ajustar os pesos e a topologia das redes. Por essas razões, essa estratégia tem se destacado frente às demais técnicas de treinamento de RNAs.

Com base nessa estratégia, Khan et al. [13] desenvolveu um modelo neuroevolutivo denominado programação genética cartesiana de redes neurais artificiais (CGPANN)[3]. Esse modelo é baseado na aplicação da programação genética cartesiana (CGP)[4] [14] para o treinamento de RNAs. A estrutura da CGP é baseada em grafos e, portanto, é adequada para representar RNAs. Na CGPANN, as redes são codificadas e evoluídas utilizando uma representação adaptada da CGP, que adiciona genes de peso à representação original. Essa abordagem neuroevolutiva tem sido utilizada em diversas aplicações, e.g., sistema de controle de pêndulo [13, 15], previsão de demanda de clientes [16], detecção de doenças [17], e reconstrução de sinais de áudio [18]. Esses trabalhos mostram que a CGPANN é um modelo neuroevolutivo altamente competitivo se comparado a algoritmos alternativos, como NEAT, ESP, SANE, CoSyne, e outros [13].

Entretanto, análises mais detalhadas realizadas por Turner [12] mostram o baixo desempenho da CGPANN quando aplicada ao domínio de classificação, indicando que esse modelo não é recomendado para lidar com tais tarefas. Duas hipóteses para justificar esse baixo desempenho podem ser levantadas. A primeira hipótese é que o modelo CGPANN utiliza a mesma técnica de otimização da CGP, que é voltada para a resolução de problemas discretos (e.g., projeto de circuitos digitais [14]). Contudo, esse esquema de otimização é aplicado na CGPANN tanto para valores discretos (e.g., topologia) quanto para valores contínuos (e.g., pesos). Diante desse cenário, levanta-se a hipótese de que o modelo CGPANN aplicado ao espaço de pesos pode levar a resultados sub-ótimos, visto que a otimização adotada não é apropriada para operar sobre variáveis de projeto contínuas. A segunda hipótese se refere ao fato da CGPANN otimizar simultaneamente a topologia e os pesos da rede. Isso também pode degradar o desempenho do modelo, visto que a configuração da rede é alterada como um todo. Melhores soluções poderiam ser obtidas

[2] Do inglês, *Evolutionary Algorithms.*
[3] Do inglês, *Cartesian Genetic Programming of Artificial Neural Networks.*
[4] Do inglês, *Cartesian Genetic Programming.*

buscando um conjunto de pesos que melhor se adapte a uma topologia específica.

Dada a relevância desse problema, o desenvolvimento de soluções que busquem corrigir as limitações da CGPANN diagnosticadas pela literatura é importante. Essa tarefa constitui a base de investigação desse trabalho. Para isso, são propostas formulações híbridas combinando a CGP com outras técnicas de otimização contínua.

Na literatura, o uso de estratégias híbridas demonstra ser uma abordagem promissora em muitos domínios. Os modelos híbridos frequentemente mostram desempenho superior aos modelos isolados (não-híbridos), apresentando alto potencial e competitividade. Em geral, a hibridização consiste no uso conjunto de métodos distintos para melhorar o desempenho das soluções de um determinado problema [19]. Como cada método é adequado para uma gama específica de cenários, a aplicabilidade dos métodos híbridos é ampliada, visto que podem complementar suas características e potencialmente melhorar o desempenho. Trabalhos recentes aplicaram estratégias híbridas combinando a CGP com outras técnicas de otimização e obtiveram resultados promissores. Alguns desses trabalhos podem ser encontrados em [20, 21, 22, 23].

Nesta dissertação, os modelos híbridos desenvolvidos realizam a evolução desacoplada da topologia e dos pesos para gerar as RNAs. A otimização da topologia é feita pela CGP. Para a otimização dos pesos, duas técnicas são utilizadas separadamente: (i) evolução diferencial (DE)[5], e (ii) *backpropagation* (BP). A combinação dos modelos CGP e DE será denominada CGPDE. Analogamente, a combinação dos modelos CGP e BP será denominada CGPBP. Essas hibridizações visam fornecer melhores classificadores, pois se baseiam no uso de algoritmos mais apropriados para cada tipo de parâmetro. A CGP é adequada para evoluir a topologia de grafos, enquanto o DE/BP são projetados para resolver problemas de otimização contínua. Além disso, ressalta-se que o DE apresenta aplicabilidade mais geral que o BP, visto que o BP requer a diferenciabilidade da função objetivo e das funções de transferência dos nós, enquanto o DE não faz essa exigência.

1.2 OBJETIVOS

O objetivo geral desta dissertação consiste no desenvolvimento e análise de modelos híbridos para construção de redes neurais artificiais via programação genética cartesiana. Para clarificar as etapas do estudo realizado, são listados a seguir os objetivos específicos. São eles:

- Desenvolver os modelos neuroevolutivos híbridos, por meio da combinação de duas técnicas distintas: (i) uma técnica de otimização discreta para ajuste da topologia (CGP), e (ii) uma técnica de otimização contínua para ajuste dos pesos (DE ou BP).

[5] Do inglês, *Differential Evolution.*

- Investigar o desempenho dos modelos híbridos no contexto de classificação de dados, comparando os resultados obtidos com o modelo de referência CGPANN e com modelos de classificação tradicionais da literatura.

- Analisar as topologias de rede geradas pelos modelos híbridos, verificando a relação entre as configurações topológicas geradas e o desempenho dos modelos.

- Avaliar o comportamento dos modelos híbridos quando otimizados a partir de funções objetivo distintas, estudando o desempenho dessas funções quando aplicadas a problemas de dados desbalanceados. As funções objetivo utilizadas são: (i) acurácia, (ii) erro quadrático médio, (iii) G-mean, (iv) F_β-score, e (v) área abaixo da curva ROC[6] (AUC)[7].

1.3 CONTRIBUIÇÕES

Ao longo da dissertação, algumas contribuições foram realizadas nos campos da neuroevolução e do aprendizado supervisionado. Entre as principais contribuições, destaca-se a introdução dos modelos CGPDE e CGPBP, que são novas abordagens para treinamento de redes neurais artificiais. Esses modelos se apresentam como alternativas promissoras em relação às estratégias de otimização encontradas na literatura. Além disso, este trabalho formula estratégias híbridas para o desacoplamento do processo de treinamento das redes neurais artificiais. Esse desacoplamento consiste em utilizar técnicas de otimização adequadas para cada tipo de parâmetro das redes. Para o ajuste de parâmetros discretos (topologia), é utilizada uma técnica de otimização discreta (CGP). Para o ajuste de parâmetros contínuos (pesos), são utilizadas técnicas de otimização contínua (DE ou BP). Outra importante contribuição deste trabalho é a apresentação de um estudo empírico acerca da influência das funções objetivo acurácia, G-mean, F_β-score e AUC sobre o desempenho geral dos modelos, sobretudo em cenários em que se depara com problemas de classificação envolvendo dados desbalanceados. Adicionalmente, este trabalho introduz uma métrica denominada Esforço de Avaliação, que quantifica o volume de avaliações da função objetivo realizadas sobre os diferentes conjuntos de dados, a fim de permitir comparações mais justas entre os algoritmos durante os testes de desempenho. Por último, destaca-se a disponibilização de todo código-fonte desenvolvido. Essa disponibilização permite a reprodutibilidade dos resultados, o uso dos modelos pela comunidade científica, e a constante manutenção e melhoria dos códigos.

---

[6] Do inglês, *Receiver Operating Characteristic*.
[7] Do inglês, *Area Under the ROC Curve*.

1.4 VISÃO GERAL DA DISSERTAÇÃO

Esta seção fornece uma visão geral dos próximos capítulos da dissertação.

O **Capítulo 2** apresenta os conceitos relativos ao aprendizado supervisionado. Esse assunto é abordado sob a perspectiva dos problemas de classificação, do aprendizado com dados desbalanceados, e das métricas de avaliação dos classificadores.

O **Capítulo 3** aborda os conceitos fundamentais da neuroevolução. Para isso, são introduzidas as redes neurais artificiais e os algoritmos evolutivos. A formulação de um método clássico de otimização de redes neurais denominado *backpropagation* é apresentada. Por fim, são discutidos os princípios e as características da neuroevolução.

O **Capítulo 4** descreve o modelo neuroevolutivo denominado CGPANN, que é o algoritmo-base para o desenvolvimento dos modelos híbridos propostos nesta dissertação. Primeiramente são introduzidos os conceitos relativos à programação genética e à programação genética cartesiana. Em seguida, o modelo CGPANN é descrito em detalhes.

O **Capítulo 5** aborda o referencial teórico básico relativo à evolução diferencial (DE). A compreensão acerca do funcionamento do DE é necessária, visto que este é um dos componentes utilizados nas formulações híbridas desenvolvidas aqui.

O **Capítulo 6** fornece uma descrição completa dos novos modelos neuroevolutivos híbridos propostos na dissertação. Inicialmente, são discutidas as motivações para a elaboração dessas formulações. São apresentadas as variantes de cada um dos modelos híbridos, com suas características e pseudocódigos. Por fim, é também introduzida uma métrica denominada Esforço de Avaliação.

O **Capítulo 7** apresenta todo o arcabouço experimental realizado para investigar e analisar o desempenho dos modelos híbridos. Os modelos propostos são comparados a algoritmos conhecidos na literatura. Testes estatísticos de significância são empregados para análise e discussão dos resultados.

Finalmente, o **Capítulo 8** traz as conclusões finais da dissertação, juntamente com algumas propostas de trabalhos futuros.

2 APRENDIZADO SUPERVISIONADO

O desenvolvimento de algoritmos voltados para problemas de aprendizado supervisionado tem ganhado destaque na área de aprendizado de máquina. Nesse capítulo, os conceitos relativos ao aprendizado supervisionado são apresentados. Esse aprendizado engloba diferentes tarefas práticas, sendo uma delas a classificação de dados. Considerando o contexto de aplicação dos modelos propostos neste trabalho, um enfoque maior é dado aos problemas de classificação e às métricas de avaliação dos classificadores.

O texto se encontra organizado da seguinte forma. A Seção 2.1 descreve a formulação geral do problema de aprendizado supervisionado. A apresentação do aprendizado com dados desbalanceados no contexto de classificação é fornecida na Seção 2.2. Na Seção 2.3 são descritas as medidas tipicamente utilizadas para avaliar o desempenho de classificadores. Por fim, as conclusões do capítulo são encontradas na Seção 2.4.

2.1 FORMULAÇÃO GERAL

A formulação geral do aprendizado supervisionado, descrita por Vapnik [24], é apresentada nesta seção. Inicialmente, três componentes abstratos são definidos:

- **Gerador de dados**: produz vetores aleatórios $\vec{x} \in X$ obtidos independentemente a partir de uma densidade de probabilidade $p(\vec{x})$ fixa e desconhecida.
- **Supervisor**: retorna um valor de saída $y \in Y$ para todo o vetor de entrada $\vec{x}$ de acordo com uma densidade condicional $p(y|\vec{x})$ fixa e desconhecida.
- **Máquina de aprendizado**: implementa um conjunto de funções $f : X \mapsto Y$ que mapeiam vetores do espaço de entrada X para o espaço de saída Y.

O problema de aprendizado supervisionado consiste em selecionar uma função f, fornecida pela máquina de aprendizado, que melhor aproxima a resposta y do supervisor, baseado no conjunto finito de vetores $\vec{x}$ produzidos pelo gerador de dados [24].

A qualidade dessa aproximação é medida por uma função objetivo $L(y, f(\vec{x}))$ que leva em consideração a resposta y para um dado $\vec{x}$ e a resposta $f(\vec{x})$ fornecida pela máquina de aprendizado. Portanto, o objetivo do aprendizado supervisionado é encontrar a função ótima $f_0(\vec{x})$ que minimiza a função objetivo sobre o conjunto de funções $f : X \mapsto Y$, quando a probabilidade $p(\vec{x}, y)$ é desconhecida mas os dados de treinamento $(\vec{x}_i, y_i)\, \forall i$ estão disponíveis [25]. Cada par $(\vec{x}_i, y_i)$ é chamado de instância. Dessa forma, denomina-se base de dados o conjunto finito de instâncias geradas de forma independente e identicamente distribuída a partir de uma densidade de probabilidade conjunta $p(\vec{x}, y) = p(\vec{x}) \cdot p(y|\vec{x})$.

Uma base de dados genérica formada por N instâncias pode ser descrita como

$$\{(\vec{x}_i, y_i) \in X \times Y \mid i = 1, 2, ..., N\} \ . \tag{2.1}$$

O aprendizado é denominado supervisionado devido à presença da variável de saída y para orientar o processo de aprendizagem [3]. A formulação acima engloba muitas tarefas práticas de aprendizado, e.g., regressão e classificação [2]. Em tarefas de classificação, considera-se que $\vec{x} \in \mathbb{R}^D$, em que as componentes do vetor de entrada $\vec{x} = \left\{x^{(1)}, ..., x^{(D)}\right\}$ são denominadas de atributos, e representam características específicas da base de dados. Além disso, as saídas assumem valores simbólicos discretos tais que $y \in \{0, 1, ..., M-1\}$, correspondendo às M classes distintas. Assim, cada função arbitrária $f(\vec{x})$ é chamada de classificador, e pertence ao conjunto de funções $f : \mathbb{R}^D \mapsto \{0, 1, ..., M-1\}$ que definem regras de decisão capazes de dividir o espaço de entrada em M regiões disjuntas.

A título de exemplo, considere um problema de classificação de vinhos cuja base de dados fictícia está ilustrada na Tabela 1.

Tabela 1 – Exemplo fictício de classificação de vinhos. Cada linha corresponde a uma instância da base de dados. As três primeiras colunas correspondem aos atributos das instâncias. A última coluna corresponde à classe das instâncias (0: vinho branco; 1: vinho tinto). A última instância apresenta classe desconhecida, e portanto pode ser classificada por meio de técnicas de aprendizado supervisionado.

Álcool (%)	**Magnésio (%)**	**Total de Fenois**	**Classe**
12.3	1.10	2	0
6.8	0.04	9	1
10.1	0.95	3	0
14.0	1.33	2	?

Essa base possui três atributos obtidos através de medições laboratoriais: (i) porcentagem de álcool, (ii) porcentagem de magnésio, e (iii) total de fenois. Nesse exemplo, apenas duas classes de vinhos estão sendo consideradas, cujos valores simbólicos pertencem ao conjunto $\{0, 1\}$, sendo a classe 0 correspondente aos vinhos brancos, e a classe 1 correspondente aos vinhos tintos. No caso de classificação binária, costuma-se denominar a classe 0 como classe negativa, e a classe 1 como classe positiva [2]. A notação referente à primeira instância é $(\vec{x}_1, y_1)$ tal que $\vec{x}_1 = \{12.3,\ 1.10,\ 2\}$ e $y_1 = 0$. A notação das demais instâncias segue padrão similar.

Como ilustrado na tabela, existem 3 instâncias cujas classes são conhecidas e 1 instância cuja classe é desconhecida. Nesse sentido, o objetivo do aprendizado supervisionado consiste em obter um classificador que realize o mapeamento dos atributos das 3 primeiras instâncias a suas classes correspondentes, por meio da otimização de uma dada função objetivo. Em seguida, esse classificador é aplicado aos atributos da última

instância, a fim de avaliar seu desempenho de classificação, visto que essa instância não foi considerada durante o processo de otimização.

O conjunto de instâncias utilizado para estimar a função de classificação é denominado conjunto de treinamento. O conjunto de instâncias utilizado para avaliar o desempenho da função de classificação estimada é denominado conjunto de teste. A utilização do conjunto de teste é importante para avaliar a capacidade de generalização do classificador, i.e., avaliar a capacidade do classificador em classificar corretamente instâncias não consideradas durante o processo de treinamento. Em geral, quanto maior o conjunto de treinamento, mais precisa é a estimação da função de classificação [9]. Entretanto, uma baixa capacidade de generalização é observada quando o classificador é treinado em excesso, ou quando a complexidade da função estimada é maior que a real complexidade do problema em questão [26].

O aprendizado supervisionado é utilizado em inúmeras áreas da ciência, finanças, e indústrias. Alguns exemplos práticos de problemas de aprendizado supervisionado são [3]:

- Prever se um paciente hospitalizado devido a um ataque cardíaco terá um segundo ataque, baseado em medidas demográficas, alimentares e clínicas.
- Identificar os números de um código postal escrito à mão, a partir de uma imagem digitalizada.
- Estimar o preço de uma ação futura com base nas medidas de desempenho da empresa e em dados econômicos.

2.2 APRENDIZADO COM DADOS DESBALANCEADOS

Em diversas aplicações de aprendizado supervisionado, existem diferenças significativas entre as proporções das instâncias presentes nas diferentes classes de uma base [27]. Essa situação é conhecida como desbalanceamento de classes. Alguns exemplos de problemas desbalanceados são [25, 9]:

- Sistemas de inspeção automática que monitoram produtos em uma linha de produção. Nesses casos, a quantidade de produtos defeituosos é significativamente menor que a quantidade de produtos não-defeituosos.
- Detecção de fraudes em cartão de crédito, em que a quantidade de transações fraudulentas é muito menor que a quantidade de transações legítimas.
- Estudo sobre uma doença rara em uma dada população. A proporção de pessoas doentes encontradas é muito menor que a proporção de pessoas sadias.

O grau de balanceamento de uma classe C é definido como a razão entre a quantidade de instâncias pertencentes à classe C e a quantidade total de instâncias da base. Portanto, uma base é perfeitamente balanceada quando suas classes possuem grau de balanceamento igual ao inverso da quantidade total de classes. Em problemas de natureza desbalanceada, o grau de balanceamento da classe minoritária é baixo. Apesar das ocorrências infrequentes, uma classificação correta da classe minoritária geralmente tem valor maior que uma classificação correta da classe majoritária, devido ao interesse em capturar os eventos raros.

No entanto, os algoritmos de aprendizado supervisionado tradicionais apresentam dificuldades em discriminar corretamente as classes raras presentes na base de dados. Em geral, esses algoritmos tendem a produzir classificadores que favorecem as classes mais frequentes, resultando em baixas taxas de reconhecimento para os grupos raros [25]. Por exemplo, se apenas 0.1% das transações de cartão de crédito são fraudulentas, então um modelo que visa a minimização de uma função objetivo baseada na taxa de erros poderia, ingenuamente, classificar todas as transações como legítimas, obtendo porcentagem de acerto de 99.9%. Embora tal desempenho pareça satisfatório, esse classificador falhou em detectar todas as atividades fraudulentas, que eram justamente as instâncias de interesse a serem detectadas.

Diversas abordagens tem sido propostas na literatura para contornar esse problema. Segundo López et al. [27], essas abordagens se dividem em dois grupos: (i) abordagens internas, que criam novos algoritmos ou modificam os existentes levando em consideração o desequilíbrio de classes; e (ii) abordagens externas, que pré-processam os dados a fim de diminuir o efeito do desbalanceamento de classes. Mais detalhes sobre o problema de aprendizado com dados desbalanceados podem ser encontrados em [9, 25, 27].

2.3 MÉTRICAS DE AVALIAÇÃO

Um classificador pode ser avaliado por meio da distinção, em relação a cada classe, dos erros e acertos das suas previsões em relação aos valores esperados. Nesse sentido, o desempenho de classificação pode ser descrito por uma tabela denominada matriz de confusão [28]. Nessa matriz, cada elemento $\epsilon_{i,j}$ representa a quantidade de instâncias classificadas como i quando sua verdadeira classe é j. A Figura 1 ilustra a matriz de confusão para um classificador binário.

		Real (+)	**Real** (−)
Predição	(+)	Verdadeiro Positivo (TP)	Falso Positivo (FP)
	(−)	Falso Negativo (FN)	Verdadeiro Negativo (TN)

Figura 1 – Matriz de confusão para um classificador binário.

A diagonal principal indica a quantidade de instâncias classificadas corretamente: verdadeiros positivos (TP) e verdadeiros negativos (TN). Por outro lado, os elementos fora da diagonal principal indicam a quantidade de erros: falsos positivos (FP) e falsos negativos (FN). Os valores presentes na matriz podem ser expressos em termos de porcentagens, que avaliam o desempenho de classificação das classes positivas e negativas separadamente [9], conforme indicado nas Equações 2.1–2.5 da Tabela 2.

Tabela 2 – Métricas de desempenho de modelos de classificação.

Eq.	**Métrica**	**Formulação**
(2.1)	Taxa de Verdadeiros Positivos (TPR) ou *Recall* (R)	$\frac{TP}{TP+FN}$
(2.2)	Taxa de Verdadeiros Negativos (TNR)	$\frac{TN}{TN+FP}$
(2.3)	Taxa de Falsos Positivos (FPR)	$\frac{FP}{TN+FP}$
(2.4)	Taxa de Falsos Negativos (FNR)	$\frac{FN}{TP+FN}$
(2.5)	Precisão (Pr)	$\frac{TP}{TP+FP}$
(2.6)	Acurácia	$\frac{TP+TN}{TP+TN+FP+FN}$
(2.7)	G-mean	$\sqrt{TPR \cdot TNR}$
(2.8)	F_β-score	$\frac{(1+\beta)\cdot R\cdot Pr}{\beta^2\cdot R+Pr}$

Esses valores são descritos a seguir.

A taxa de verdadeiros positivos (TPR), ou *recall* (R) (Eq. 2.1), indica a razão entre a quantidade de instâncias positivas corretamente classificadas e a quantidade total de instâncias positivas. A TPR responde a seguinte questão: dentre todas as instâncias positivas, qual a porcentagem classificada corretamente?

A taxa de verdadeiros negativos (TNR) (Eq. 2.2) indica a razão entre a quantidade de instâncias negativas corretamente classificadas e a quantidade total de instâncias negativas. A TNR responde a seguinte questão: dentre todas as instâncias negativas, qual a porcentagem classificada corretamente?

A taxa de falsos positivos (FPR) (Eq. 2.3) indica a razão entre a quantidade de instâncias negativas incorretamente classificadas como positivas e a quantidade total de instâncias negativas. A FPR responde a seguinte questão: dentre todas as instâncias negativas, qual a porcentagem classificada incorretamente?

A taxa de falsos negativos (FNR) (Eq. 2.4) indica a razão entre a quantidade de instâncias positivas incorretamente classificadas como negativas e a quantidade total de instâncias positivas. A FNR responde a seguinte questão: dentre todas as instâncias positivas, qual a porcentagem classificada incorretamente?

A precisão (Pr) (Eq. 2.5) indica a razão entre a quantidade de instâncias positivas corretamente classificadas e a quantidade total de instâncias classificadas como positivas pelo classificador. A precisão responde a seguinte questão: dentre todas as instâncias classificadas como positivas pelo classificador, qual a porcentagem classificada corretamente?

Além das taxas de erro e acerto para cada classe, esta seção destaca outras métricas comumente adotadas para avaliação de desempenho de classificadores. Primeiramente, são introduzidas cinco métricas de classificação: (i) acurácia, (ii) G-mean, (iii) F_β-score, (iv) erro quadrático médio, e (v) entropia cruzada. Em seguida, é introduzida uma métrica de ranqueamento denominada área abaixo da curva ROC.

As métricas de classificação exigem a rotulação das instâncias como positivas ou negativas, e são descritas a seguir. A acurácia (Eq. 2.6) corresponde à razão entre a quantidade de instâncias corretamente classificadas e a quantidade total de instâncias. A G-mean [29] (Eq. 2.7) corresponde à média geométrica entre a taxa de verdadeiros positivos (TPR) e a taxa de verdadeiros negativos (TNR), e mede o desempenho equilibrado de um classificador considerando as taxas de acerto de ambas as classes. A F_1-score [9] corresponde à média harmônica entre *recall* (R) e precisão (Pr). De forma genérica, essa métrica assume a notação F_β-score (Eq. 2.8), em que o parâmetro β ajusta a importância relativa entre R e Pr.

As métricas G-mean e F_β-score têm sido utilizadas para fornecer avaliações mais adequadas em problemas de natureza desbalanceada [30], visto que representam a capacidade do modelo em associar corretamente instâncias da classe minoritária ou consideram com mesma relevância a discriminação das instâncias pertencentes às classes positiva e negativa [25].

As formulações do erro quadrático médio (MSE) e da entropia cruzada (CE) são descritas, respectivamente, como

$$MSE(f, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2 \ , \tag{2.9}$$

$$CE(f, y) = -\sum_{i=1}^{N} y_i \cdot \ln\left(f(\vec{x}_i)\right) \ , \tag{2.10}$$

em que y_i é a classe da i-ésima instância, $f(\vec{x}_i)$ é o valor de saída gerado pelo classificador f ao avaliar a i-ésima instância, e N é o número de instâncias avaliadas.

O MSE quantifica a diferença quadrática média entre os valores estimados e os valores reais das instâncias. Dessa forma, quanto menor o MSE de um classificador, mais próximos estão os valores estimados em relação aos valores reais. Por outro lado, a métrica CE é utilizada para quantificar a diferença entre duas distribuições de probabilidade. Dessa forma, o valor de CE aumenta à medida que as probabilidades previstas pelo classificador divergem das classes reais das instâncias.

Devido à diferenciabilidade do MSE e CE, essas medidas são frequentemente utilizadas como funções objetivo a serem minimizadas pelos modelos de classificação tradicionais, cuja otimização é baseada na direção do vetor gradiente [31].

Além das métricas de classificação supracitadas, é introduzida a seguir uma métrica de ranqueamento denominada área abaixo da curva ROC (AUC). Diferentemente das métricas anteriores, essa medida não exige a rotulação das instâncias. Para compreensão da AUC, primeiramente faz-se necessária a introdução da curva ROC, que foi originalmente proposta na teoria de detecção de sinais [32] e tem sido empregada na área de aprendizado de máquina para análise e seleção de modelos [28].

A curva ROC é um gráfico bidimensional que representa a taxa de verdadeiros positivos (TPR) em função da taxa de falsos positivos (FPR) utilizando diversas regras de decisão [28]. Cada regra de decisão pode ser descrita por um valor de limiar de decisão θ, que divide as instâncias entre positivas e negativas, gerando um par ordenado $(FPR, TPR)_\theta$. Portanto, cada limiar de decisão θ representa um ponto no plano ROC.

Para cada instância $\vec{x}_i$, o classificador produz uma saída $f(\vec{x}_i)$ de valor contínuo dentro de um intervalo $[L, U]$, que representa o grau de pertinência da instância à classe positiva. Quanto mais próximo o valor de saída de uma instância se encontra de U, maior a chance dela pertencer à classe positiva. Analogamente, quanto mais próximo o valor de saída se encontra de L, maior a chance da instância pertencer à classe negativa. Os intervalos mais comuns na literatura são $[L = -1, U = 1]$ e $[L = 0, U = 1]$. Considerando um intervalo $[L, U]$ e um valor de limiar de decisão θ ($L \leq \theta \leq U$), a j-ésima instância com valor de saída $f(\vec{x}_j)$ tal que $\theta \leq f(\vec{x}_j) \leq U$ é classificada como pertencente à classe positiva. Analogamente, a k-ésima instância com valor de saída $f(\vec{x}_k)$ tal que $L \leq f(\vec{x}_k) < \theta$ é classificada como pertencente à classe negativa.

Dessa forma, a curva ROC é obtida ao se variar o limiar de decisão θ sobre toda a faixa de saída $[L, U]$, traçando os valores de FPR e TPR para cada limiar [9]. A Figura 2 ilustra um exemplo de curva ROC.

A área abaixo da curva ROC (AUC) é um indicador de desempenho de um algoritmo de classificação em relação à curva ROC, e sua formulação discreta [33] é dada por

$$AUC(f) = \frac{1}{PQ}\left(\sum_{j=1}^{P}\sum_{k=1}^{Q} G\left(f(\vec{x}_j^{+}) - f(\vec{x}_k^{-})\right)\right) \quad , \tag{2.11}$$

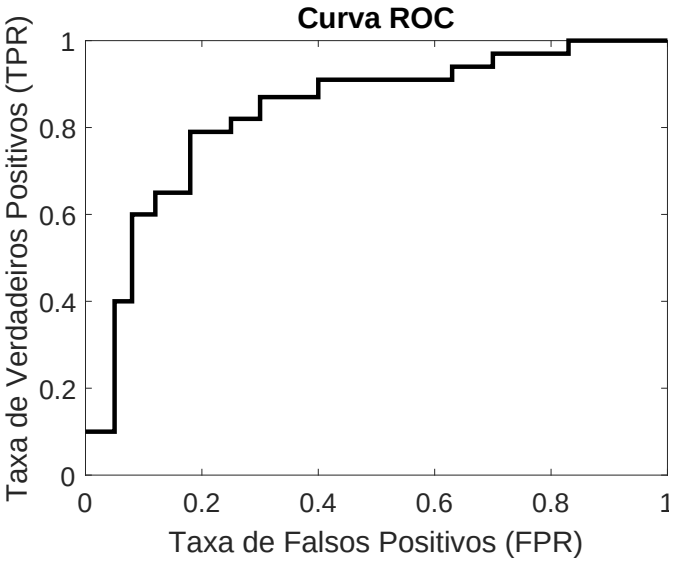


Figura 2 – Exemplo de uma curva ROC.

em que P e Q são as quantidades totais de instâncias positivas e negativas, respectivamente; $\vec{x}^+$ e $\vec{x}^-$ são os vetores de instâncias positivas e negativas, respectivamente; $f(\vec{x}_i)$ é o valor de saída do classificador f para a i-ésima instância; e $G(z)$ é definido como

$$G(z) = \begin{cases} 0 & \text{se } z < 0 \\ 0.5 & \text{se } z = 0 \\ 1 & \text{se } z > 0 \ . \end{cases} \tag{2.12}$$

A AUC de um classificador é equivalente à probabilidade do classificador produzir um valor de saída maior para uma instância positiva escolhida aleatoriamente do que para uma instância negativa escolhida aleatoriamente [28]. Portanto, um classificador que maximiza a AUC não exige o estabelecimento de rótulos para as instâncias, pois seu objetivo é ranqueá-las corretamente, i.e., gerar valores de saída maiores para as instâncias positivas do que para as instâncias negativas. Mais detalhes sobre a curva ROC e a AUC podem ser encontrados em [28].

2.4 CONCLUSÕES DO CAPÍTULO

Este capítulo introduziu os conceitos relativos ao aprendizado supervisionado, dando ênfase às tarefas de classificação de dados. Foi apresentado o problema de classes desbalanceadas, comumente encontrado em diversos cenários reais. No escopo das métricas de desempenho, foram descritas as formulações das principais formas utilizadas para avaliação e seleção de modelos. Os conceitos aqui apresentados são fundamentais para se compreender o contexto de aplicação deste trabalho, que consiste no desenvolvimento de modelos híbridos de aprendizado aplicados à classificação de dados.

3 NEUROEVOLUÇÃO

A combinação de algoritmos evolutivos (EAs) e redes neurais artificiais (RNAs), denominada neuroevolução, é uma sub-área do aprendizado de máquina que aplica EAs para o treinamento de RNAs [12]. O presente capítulo tem como objetivo introduzir os conceitos relativos à neuroevolução. Esse tema é de fundamental relevância, visto que os modelos de aprendizado propostos neste trabalho são baseados em técnicas neuroevolutivas.

O texto se encontra organizado da seguinte forma. A Seção 3.1 apresenta as redes neurais artificiais, incluindo seu método clássico de treinamento denominado *backpropagation*. Na Seção 3.2, são descritos os princípios básicos da computação evolutiva. Em seguida, a Seção 3.3 introduz os conceitos relativos à neuroevolução e suas características. Finalmente, a Seção 3.4 apresenta as principais conclusões do capítulo.

3.1 REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais (RNAs) são modelos computacionais inspirados na biologia estrutural e comportamental dos neurônios presentes no sistema nervoso animal [9]. De forma análoga aos neurônios biológicos, esses modelos são formados por unidades interconectadas capazes de receber, processar e transmitir informações.

A unidade básica de uma RNA é o neurônio artificial, também denominado de nó, ilustrado na Figura 3.

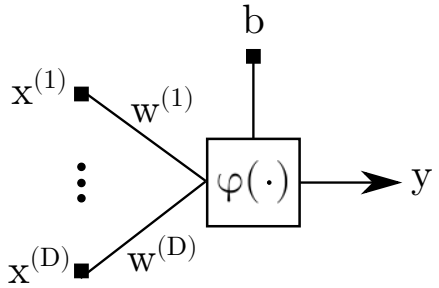


Figura 3 – Unidade básica de uma RNA: neurônio artificial.

As entradas $x^{(1)}, \ldots, x^{(D)}$ correspondem aos valores de entrada provenientes de nós anteriores. Os pesos $w^{(1)}, \ldots, w^{(D)}$ correspondem às forças de conexão das sinapses entre os nós, i.e., indicam a relevância de cada valor de entrada. A função $\varphi(\cdot)$ corresponde à função de ativação do nó, sendo comumente utilizadas as funções do tipo sigmoide logística ou tangente hiperbólica [34]. O *bias* b corresponde a um valor-limiar interno incluído como forma de generalização do modelo. Por fim, a saída y corresponde ao valor de saída do nó,

que é transmitido aos nós subsequentes da rede. A saída de cada nó é calculada como

$$y = \varphi\left(\sum_{r=1}^{D} w^{(r)} x^{(r)} + b\right) \ , \tag{3.1}$$

e a função de ativação do tipo sigmoide logística é dada por

$$\varphi(s) = \frac{1}{1+e^{-s}} \ . \tag{3.2}$$

Diversas configurações podem ser obtidas por meio da organização espacial dos nós de uma RNA. Uma configuração típica é denominada *multilayer feedforward* totalmente conectada [2], cujas características são: (i) os nós são dispostos em camadas; (ii) os nós podem se conectar apenas aos nós da camada seguinte; (iii) cada nó se conecta necessariamente a todos os nós da camada seguinte.

A Figura 4 ilustra uma RNA *multilayer feedforward* totalmente conectada com D nós na camada de entrada (nós de entrada), H nós na camada intermediária (nós intermediários), e 1 nó na camada de saída (nó de saída).

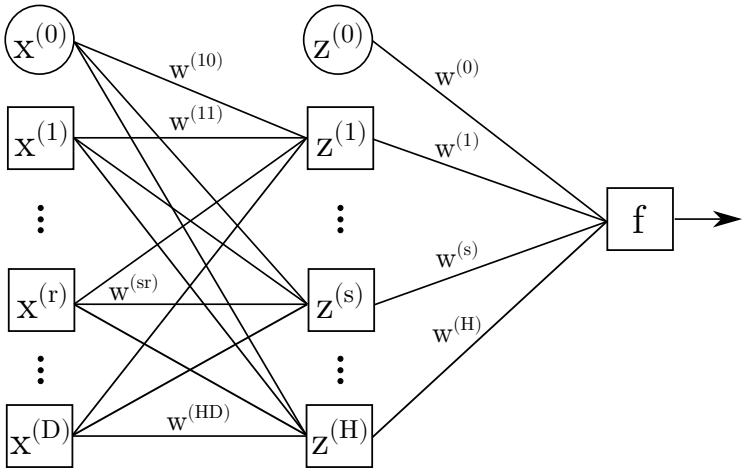


Figura 4 – Exemplo de uma RNA do tipo *multilayer feedforward* totalmente conectada com D nós na camada de entrada (e 1 *bias*), H nós na camada intermediária (e 1 *bias*), e 1 nó na camada de saída.

Os termos $x^{(0)}$ e $z^{(0)}$ representam os valores de *bias*, considerados como nós extras presentes nas camadas de entrada e intermediária, respectivamente. O termo $w^{(sr)}$ representa o peso entre o nó intermediário s ($1 \leq s \leq H$) e o nó de entrada r ($1 \leq r \leq D$). Similarmente, o termo $w^{(s)}$ representa o peso entre o nó de saída e o nó intermediário s. O valor obtido pelo nó intermediário s é dado por

$$z^{(s)} = \varphi\left(u^{(s)}\right) = \varphi\left(\sum_{r=0}^{D} w^{(sr)}\, x^{(r)}\right) \ , \tag{3.3}$$

em que $\vec{x} \in \mathbb{R}^D$ é o vetor de entrada do problema de aprendizado tal que $\vec{x} = \{x^{(1)}, \ldots, x^{(D)}\}$; e $\varphi(\cdot)$ é a função de ativação. Uma vez calculados os valores dos nós

intermediários, o valor do nó de saída da rede é dado por

$$f = \varphi(v) = \varphi\left(\sum_{s=0}^{H} w^{(s)}\, z^{(s)}\right) \quad . \tag{3.4}$$

As RNAs são comumente aplicadas a problemas de regressão e classificação. Em problemas de classificação binária, a rede pode apresentar 1 ou 2 nós de saída, a depender do esquema de classificação adotado. Já em problemas de classificação com M classes ($M > 2$), a rede apresenta necessariamente M nós de saída [3]. No caso da rede apresentar apenas um nó de saída, o valor gerado representa o grau de pertinência da instância em relação à classe positiva. Por outro lado, no caso da rede apresentar dois ou mais nós de saída, cada saída está associada a uma classe, e portanto o valor gerado em cada nó de saída representa o grau de pertinência da instância em relação à classe do nó correspondente. Além disso, vale lembrar que a configuração genérica de uma RNA admite uma quantidade arbitrária de camadas intermediárias, embora um número excessivo de camadas tende a dificultar o treinamento da rede [2].

O processo de treinamento das RNAs consiste no ajuste de seus parâmetros numéricos, de modo a produzir funções de aprendizado capazes de mapear corretamente os dados de entrada em relação aos dados de saída esperados. Esses parâmetros correspondem aos pesos das conexões, que são atualizados visando a minimização (ou maximização) de uma dada função objetivo. Os parâmetros não-numéricos (e.g., funções de ativação e topologia) geralmente são mantidos fixos. Existem diversas técnicas para o treinamento de RNAs. Uma delas, denominada *backpropagation*, é apresentada em detalhes a seguir.

3.1.1 Backpropagation

O algoritmo *backpropagation* (BP) [35, 36] é uma técnica de otimização tradicionalmente utilizada para atualização dos pesos das RNAs. Sua regra de aprendizado é baseada no gradiente descendente (GD) [37], que é um método de otimização iterativa de primeira ordem, cujos passos de atualização dos parâmetros são dados na direção contrária ao vetor gradiente da função objetivo.

Existem diversas variantes do método GD, sendo as principais: (i) *full-batch* GD, (ii) *mini-batch* GD, e (iii) *stochastic* GD. Seja um conjunto de treinamento com N instâncias. No modo *full-batch* GD, o vetor gradiente da função objetivo é calculado utilizando as N instâncias do conjunto de treinamento, e em seguida os pesos são atualizados. No modo *mini-batch* GD, o vetor gradiente é calculado utilizando k instâncias aleatórias ($1 < k < N$) do conjunto de treinamento para a atualização dos pesos, sendo as k instâncias reamostradas a cada passo do processo de otimização. Por fim, no modo *stochastic* GD, o vetor gradiente é calculado utilizando apenas 1 instância aleatória do conjunto de treinamento para a atualização dos pesos, sendo a instância reamostrada a cada passo do processo de otimização. Dada a relevância do algoritmo *full-batch* GD neste

trabalho, sua formulação é apresentada a seguir, seguindo a mesma notação adotada pela Figura 4.

Seja um problema de classificação binária cujo conjunto de treinamento é dado por

$$T = \{(\vec{x}_i, y_i) \in \mathbb{R}^D \times \{0,1\} \,|\, i = 1, \ldots, N\} \ , \tag{3.5}$$

em que $\vec{x}_i \in \mathbb{R}^D$ é o vetor de entrada da i-ésima instância tal que $\vec{x}_i = \{x_i^{(1)}, \ldots, x_i^{(D)}\}$, e y_i é a classe da i-ésima instância tal que $y \in \{0,1\}$. Uma vez obtido o valor de saída $f_i = f(\vec{x}_i)$ para cada instância $\vec{x}_i$ (vide Equação 3.4), o erro de classificação da rede em relação à instância $\vec{x}_i$ pode ser calculado como $e_i = y_i - f_i$. Com base nessa expressão, a função objetivo $L(\vec{w})$ pode ser definida como

$$L(\vec{w}) \ = \ \frac{1}{2}\sum_{i=1}^{N} e_i^2 \ = \ \frac{1}{2}\sum_{i=1}^{N} (y_i - f_i)^2 \ \ \forall\, (\vec{x}_i, y_i) \in T \ , \tag{3.6}$$

correspondendo ao somatório dos erros quadráticos sobre todo o conjunto T, e onde $\vec{w}$ representa o vetor que armazena todos os parâmetros numéricos da rede (i.e., os pesos).

Os parâmetros são inicializados com valores aleatórios e, a cada época (iteração), são atualizados no sentido oposto ao vetor gradiente, conforme

$$\vec{w} \leftarrow \vec{w} - \eta\, \nabla L \ , \tag{3.7}$$

onde ∇L é o vetor gradiente da função objetivo em relação ao vetor $\vec{w}$, e η é uma constante positiva denominada taxa de aprendizado, que indica o fator de atualização dos parâmetros aplicado a cada época. Cada componente de ∇L é calculada pela derivada parcial da função objetivo em relação a cada um dos pesos $w^{(l)}$, ou seja,

$$\frac{\partial L}{\partial w^{(l)}} \ = \ \frac{1}{2}\sum_{i=1}^{N} \frac{\partial e_i^2}{\partial w^{(l)}} \ = \ \sum_{i=1}^{N} e_i \, \frac{\partial e_i}{\partial w^{(l)}} \ . \tag{3.8}$$

Por fim, existem duas maneiras distintas utilizadas para calcular a derivada parcial $\dfrac{\partial e_i}{\partial w^{(l)}}$: (i) se $w^{(l)}$ corresponde a um peso arbitrário $w^{(s)}$ da camada de saída, então a derivada é obtida por

$$\frac{\partial e_i}{\partial w^{(l)}} \ = \ \frac{\partial e_i}{\partial w^{(s)}} \ = \ \frac{\partial e_i}{\partial f_i}\,\frac{\partial f_i}{\partial v_i}\,\frac{\partial v_i}{\partial w^{(s)}} \ = \ -\varphi'\left(v_i\right) \cdot z_i^{(s)} \ , \tag{3.9}$$

e (ii) se $w^{(l)}$ corresponde a um peso arbitrário $w^{(sr)}$ da camada intermediária, então a derivada é obtida por

$$\frac{\partial e_i}{\partial w^{(l)}} \ = \ \frac{\partial e_i}{\partial w^{(sr)}} \ = \ \frac{\partial e_i}{\partial z_i^{(s)}}\,\frac{\partial z_i^{(s)}}{\partial u_i^{(s)}}\,\frac{\partial u_i^{(s)}}{\partial w^{(sr)}} \ = \ -\varphi'(v_i) \cdot w^{(s)} \cdot \varphi'\left(u_i^{(s)}\right) \cdot x_i^{(r)} \ . \tag{3.10}$$

3.1.2 Características das redes neurais artificiais

As RNAs apresentam diversas características relevantes que possibilitaram seu amplo uso na literatura. Algumas dessas características são destacadas a seguir [9]:

- RNAs *multilayer feedforward* com pelo menos uma camada intermediária são aproximadores universais, i.e., podem ser utilizadas para aproximar qualquer função alvo que satisfaça os critérios especificados pelo Teorema de Aproximação Universal [38].
- O método gradiente descendente frequentemente converge para algum mínimo local. Uma maneira de escapar desses mínimos é adicionando um termo de momento à expressão de atualização dos pesos [39].
- Após a obtenção do modelo final de aprendizado por meio do processo de treinamento, as instâncias de teste são processadas rapidamente.

3.2 COMPUTAÇÃO EVOLUTIVA

A computação evolutiva é uma área da computação inspirada no processo de evolução natural [40]. Os princípios fundamentais da computação evolutiva são baseados na preservação dos indivíduos mais aptos e na criação de novos indivíduos por tentativa e erro [41]. Uma visão geral dos algoritmos evolutivos (EAs) é descrita a seguir.

3.2.1 Visão geral da computação evolutiva

No contexto biológico da evolução, seja uma população de indivíduos que vive em um dado ambiente. Essa população está sujeita a uma pressão seletiva, que consiste na sobrevivência dos indivíduos mais aptos e na eliminação dos indivíduos menos aptos. Esse mecanismo de seleção natural gera uma elevação na aptidão média da população, cujos indivíduos serão capazes de transmitir suas características aos seus descendentes. Fazendo a correspondência desses conceitos para a computação evolutiva, o ambiente corresponde ao problema a ser resolvido, os indivíduos correspondem às soluções candidatas, e a aptidão de um indivíduo corresponde à qualidade de uma solução candidata frente ao problema em questão. A aptidão é usualmente ligada à função objetivo a ser otimizada. Além disso, os indivíduos são normalmente criados de forma aleatória.

O Algoritmo 1 descreve o esquema geral de um algoritmo evolutivo. Inicialmente, a população é inicializada com indivíduos aleatórios. Com base em seus valores de aptidão, os indivíduos são selecionados para criação de um novo conjunto de indivíduos, por meio da aplicação de operadores genéticos de recombinação e mutação. Após o uso desses operadores, o conjunto aumentado, composto pelos novos indivíduos e pelos indivíduos já presentes na população, competirá pela sobrevivência. Essa competição selecionará os

Algoritmo 1: Esquema geral de um algoritmo evolutivo

```
1 Inicializa população com indivíduos aleatórios
2 Avalia cada indivíduo
3 while condição de parada não for satisfeita do
4     Seleciona os indivíduos-pais
5     Aplica recombinação nos indivíduos-pais
6     Aplica mutação nos indivíduos-filhos
7     Avalia os novos indivíduos-filhos criados
8     Seleciona os indivíduos sobreviventes para a próxima geração
```

indivíduos sobreviventes para a próxima geração com base em critérios de seleção, e.g., aptidão e idade. O processo descrito acima é iterativo, e se repete até atingir a condição de parada predefinida.

A computação evolutiva é uma técnica essencialmente estocástica [42]. A função de aptidão representa uma estimação heurística da qualidade das soluções, e o processo de busca é guiado pelos operadores de seleção e variação [41]. Durante a seleção, os indivíduos mais aptos têm maior probabilidade de serem selecionados. No entanto, até mesmo os indivíduos menos aptos têm alguma chance de se tornarem pais e sobreviventes. Essa característica evita o elitismo e permite uma exploração mais ampla no espaço de busca das soluções. Durante a variação, i.e., recombinação e mutação, a escolha de quais informações serão combinadas e modificadas também é aleatória. Essa estocasticidade garante uma busca mais abrangente em relação aos métodos de otimização determinísticos tradicionais, aumentando a probabilidade de escape de soluções ótimas locais.

3.2.2 Componentes dos algoritmos evolutivos

Os principais componentes dos algoritmos evolutivos são [41, 43]: (i) representação, (ii) função de aptidão, (iii) população, (iv) mecanismo de seleção de pais, (v) operadores de variação, (vi) mecanismo de seleção de sobreviventes, (vii) inicialização, e (viii) condição de parada. Cada componente é brevemente descrito a seguir.

A representação dos indivíduos consiste no mapeamento das soluções originais de um problema de otimização (fenótipos) para estruturas de dados com capacidade de serem interpretadas e manipuladas pelo algoritmo evolutivo (genótipos). A transformação fenótipo-genótipo é denominada codificação. Já a transformação inversa, genótipo-fenótipo, é denominada decodificação. A título de exemplo, seja um problema de otimização de números inteiros. Para serem manipulados pelo algoritmo evolutivo, os números inteiros podem ser representados em codificação binária. Dessa forma, o número 18 seria visto como fenótipo, e 10010 como seu genótipo correspondente. A escolha da representação dos indivíduos depende da técnica evolutiva utilizada e do problema a ser resolvido.

A função de aptidão tem o papel de impor os requerimentos sobre os quais os

indivíduos devem se adaptar [41]. Do ponto de vista evolutivo, a função de aptidão representa a medida de qualidade dos indivíduos, sendo o critério responsável pela seleção e reprodução. Do ponto de vista do problema de otimização, a função de aptidão muitas vezes representa a função objetivo a ser otimizada. A título de exemplo, em um problema de minimização de $f : \mathbb{Z} \mapsto \mathbb{Z}$ tal que $f(x) = x^2$, a aptidão do genótipo binário 10010 corresponde ao quadrado de seu respectivo fenótipo, i.e., $18^2 = 324$.

A população é um conjunto finito de indivíduos, e constitui a unidade sobre a qual ocorre a evolução [41]. Uma população é definida pela quantidade de indivíduos nela presentes, sendo essa quantidade um parâmetro dos algoritmos evolutivos. A diversidade de uma população é uma medida referente à quantidade de soluções distintas existentes, e pode ser medida em termos dos valores de aptidão, dos fenótipos ou dos genótipos. Em geral, populações pouco diversificadas tendem à estagnação prematura, encontrando dificuldades para explorar novas regiões do espaço de busca.

O mecanismo de seleção de pais tem como objetivo selecionar os indivíduos com base em seus valores de aptidão, a fim de permitir que os melhores indivíduos se tornem os pais da próxima geração. Os indivíduos selecionados por meio desse mecanismo se submetem aos operadores de variação para a criação dos indivíduos filhos. Esse mecanismo de seleção é um dos responsáveis pela pressão seletiva, que eleva a aptidão média da população. Devido ao caráter estocástico dos algoritmos evolutivos, o mecanismo de seleção de pais é probabilístico, i.e., todos os indivíduos da população tem probabilidade maior ou igual a zero de serem selecionados. Em geral, os indivíduos mais aptos possuem maiores chances de serem selecionados em relação aos indivíduos menos aptos.

Os operadores de variação têm o papel de criar novos indivíduos a partir dos indivíduos existentes. Esses operadores podem ser divididos em dois grupos: mutação e recombinação. A mutação é um operador unário aplicado a um indivíduo pai, que gera um novo indivíduo filho por meio de alterações aleatórias em segmentos aleatórios de seu genótipo. O operador de mutação pode ser visto como uma pertubação aplicada às soluções candidatas. Por outro lado, a recombinação é um operador aplicado a dois ou mais indivíduos pais, que geram indivíduos filhos por meio da combinação de segmentos aleatórios de seus genótipos. A recombinação parte do princípio que agregar características de indivíduos distintos pode produzir filhos que carregam as características individuais de cada pai.

O mecanismo de seleção de sobreviventes tem como objetivo selecionar os indivíduos com base em seus valores de aptidão e/ou idade, a fim de permitir que os indivíduos selecionados sobrevivam para a próxima geração. O critério de seleção baseado na aptidão busca preservar os melhores indivíduos, elevando a aptidão média da população. Por outro lado, o critério de seleção baseado na idade busca renovar os genótipos da população, preservando apenas os indivíduos filhos. Um mecanismo de seleção de sobreviventes é

denominado elitista quando garante que uma proporção dos indivíduos mais aptos sempre sobreviva para a próxima geração. Os mecanismos elitistas devem ser aplicados de tal forma que garantam a manutenção da diversidade da população.

A inicialização consiste em criar os indivíduos da primeira geração. Em geral, esse processo é aleatório. Alternativamente, uma inicialização baseada em heurísticas específicas do problema em questão pode ser aplicada, a fim de criar indivíduos cujos valores de aptidão sejam maiores que a aptidão de um indivíduo criado aleatoriamente. Contudo, simplesmente obter melhores indivíduos pode acabar resultando em convergência prematura e, portanto, a amostragem deve ser feita de forma a ampliar o espaço de busca.

A condição de parada determina quando o algoritmo evolutivo encerrará sua execução. As principais condições de parada são [44]: (i) tempo máximo de execução, (ii) quantidade máxima de avaliações da função objetivo, (iii) quantidade máxima de gerações, (iv) aptidão mínima alcançada, (v) quantidade de gerações passadas sem melhoria significativa na aptidão do melhor indivíduo, e (vi) diversidade da população atingir valor predefinido.

A literatura relacionada à computação evolutiva apresenta uma grande quantidade de algoritmos desenvolvidos. Alguns dos mais conhecidos são os algoritmos genéticos [43, 45], programação genética [46], programação genética cartesiana [1], evolução diferencial [47], estratégias evolutivas [48], e programação evolutiva [49]. Mais detalhes sobre a computação evolutiva podem ser encontrados em [41, 42, 50].

3.3 NEUROEVOLUÇÃO

A neuroevolução [11, 51, 52] consiste na aplicação de algoritmos evolutivos para o treinamento de redes neurais artificiais. Esta seção aborda em detalhes os princípios da neuroevolução e suas principais características.

3.3.1 Princípios da neuroevolução

Como visto na Seção 3.1, as RNAs podem ser descritas pela sua topologia, pelos pesos de suas conexões, e pelas funções de transferência de seus nós. Nesse sentido, o escopo da neuroevolução consiste em treinar uma dada RNA por meio da manipulação de sua topologia e/ou seus pesos e/ou suas funções de transferência. Essa capacidade de manipulação permitiu o desenvolvimento de diversas técnicas neuroevolutivas, que são capazes de otimizar diferentes componentes das RNAs. A seguir é discutido como cada componente pode ser manipulado pela neuroevolução. Em seguida, são apresentados os principais esquemas de representação das técnicas neuroevolutivas.

3.3.1.1 *Evolução dos pesos*

A evolução dos pesos de uma RNA é realizada por meio da aplicação de um determinado algoritmo evolutivo para otimização dos pesos e *bias* da rede. Essa abordagem consiste em duas etapas principais [52]: (i) escolha da forma de representação dos pesos, e (ii) escolha do processo evolutivo a ser aplicado nos indivíduos. As primeiras técnicas neuroevolutivas desenvolvidas se concentravam na evolução dos pesos sobre uma topologia fixa [53]. Nesse caso, cada conjunto de pesos de uma RNA descreve um indivíduo distinto da população do algoritmo evolutivo. Esse indivíduo é tipicamente representado como um vetor de valores reais. Além disso, a topologia da RNA deve ser escolhida a priori pelo usuário, e mantida constante durante toda a execução do algoritmo. A título de exemplo, seja a RNA ilustrada na Figura 4. O genótipo de pesos para essa rede pode ser representado como um vetor de valores reais, conforme

$$\{w^{(10)}, w^{(11)}, \ldots, w^{(sr)}, w^{(HD)}, w^{(0)}, w^{(1)}, \ldots, w^{(s)}, w^{(H)}\} \ , \tag{3.11}$$

em que cada vetor representa um indivíduo da população do algoritmo evolutivo.

As técnicas neuroevolutivas para evolução de pesos utilizando topologia fixa apresentam diversas características importantes, sendo algumas delas destacadas a seguir. Primeiramente, essas técnicas não exigem a diferenciabilidade nem da função objetivo nem das funções de transferência. Essa característica abre espaço para a utilização de uma ampla gama de funções distintas, além de permitir que o treinamento de redes com muitas camadas intermediárias ocorra sem dificuldades. Por outro lado, os métodos determinísticos baseados no cálculo do vetor gradiente (e.g., *backpropagation*) exigem a diferenciabilidade dessas funções, e encontram dificuldades em lidar com redes de muitas camadas, sobretudo devido ao efeito conhecido como *vanishing/exploding gradient problem* [54].

Além disso, as técnicas neuroevolutivas são menos dependentes dos pesos iniciais, e são capazes de escapar de ótimos locais devido ao seu caráter estocástico. Por outro lado, os métodos determinísticos tendem a ficar presos em ótimos locais, e são altamente dependentes da inicialização dos pesos [12].

Vale ressaltar que todos os métodos (estocásticos ou determinísticos) que otimizam exclusivamente os pesos de uma RNA executam o processo de busca considerando uma região restrita do espaço de soluções possíveis. Essa restrição ocorre justamente por se manter uma topologia fixa durante todo o processo de otimização. Segundo Hoekstra [51], a escolha da topologia exerce grande influência na eficácia das técnicas neuroevolutivas que otimizam os pesos de redes com topologia fixa. Dessa forma, essas técnicas exigem que o usuário experimente diversas topologias, ou utilize informações específicas do problema para selecionar uma topologia adequada [12].

3.3.1.2 *Evolução da topologia*

A literatura relacionada à neuroevolução indica que tanto a topologia quanto os pesos são importantes no treinamento das RNAs [11]. Em teoria, uma RNA pode ser entendida como um espaço de topologias e pesos. Nessa concepção, fixar uma topologia limita a busca a apenas um subconjunto do espaço de pesos contido no espaço de topologia mais amplo [12]. Essa limitação é uma desvantagem dos métodos que manipulam apenas os pesos da rede. Dessa forma, as técnicas que evoluem a topologia são consideradas benéficas, visto que ajustam a topologia para uma dada tarefa, sem que o usuário necessite conhecer a priori uma topologia adequada. Durante o processo de busca, essas técnicas são capazes de estabelecer relações entre topologias e pesos, com potencial de criarem RNAs incomuns, que normalmente não seriam concebidas por um projetista humano.

A evolução da topologia pode ser aplicada de diversas maneiras. Uma delas consiste em utilizar operadores de mutação que adicionem ou removam os nós e conexões das redes de forma incremental. Dessa forma, as topologias vão gradualmente aumentando ou diminuindo ao longo da evolução. Devido a esse caráter incremental, as topologias geradas por esse processo de busca tendem a ficar presas em bacias de atração de ótimos locais [55].

De maneira geral, os processos de evolução não impõem nenhum tipo de restrição quanto às topologias criadas. No entanto, existem problemas que explicitamente proíbem a utilização de determinadas topologias de rede. Nesses casos, uma ou mais restrições podem ser impostas. Por exemplo, em problemas que não admitem redes recorrentes (i.e., redes com conexões retroalimentadas), pode-se considerar restringir a busca permitindo apenas redes cujas topologias sejam do tipo *feedforward*.

3.3.1.3 *Evolução das funções de transferência*

Grande parte dos métodos tradicionais de treinamento de RNAs utilizam um único tipo de função de ativação em todos os nós da rede. Além disso, esses métodos costumam impor restrições quanto ao tipo de função de ativação utilizada, visto que exigem sua diferenciabilidade. De acordo com Duch & Jankowski [56, 57], o tipo de função de transferência utilizada pelos nós influencia fortemente o desempenho de uma RNA. Portanto, as restrições impostas sobre as funções de ativação podem impactar negativamente o desempenho desses métodos.

Alternativamente, a neuroevolução é capaz de manipular a função de ativação de cada nó da rede, eliminando todas as restrições impostas pelos métodos tradicionais. Isso significa que cada nó pode ter sua própria função de ativação, sem exigência quanto à sua diferenciabilidade. Para isso, genes adicionais são incluídos no genótipo, descrevendo a função de transferência de cada nó. Com o uso de técnicas neuroevolutivas dessa natureza,

RNAs heterogêneas podem ser criadas, possibilitando uma busca no espaço das funções de transferência dos nós, e selecionando as funções adequadas para cada tarefa específica. A evolução das funções de transferência pode ocorrer tanto por meio da seleção dentro de um conjunto de funções predefinidas, quanto por meio da construção de novas funções a partir de um conjunto predefinido de operadores básicos.

3.3.1.4 *Evolução das regras de aprendizado*

Além da capacidade de manipular os componentes apresentados previamente, as técnicas neuroevolutivas também são capazes de evoluir as regras de aprendizado utilizadas para treinar as RNAs [52]. Dessa forma, cada rede pode adaptar sua própria regra de aprendizado de acordo com sua topologia e com a tarefa a ser executada. Em outras palavras, uma RNA deve aprender sua regra de aprendizado dinamicamente, em vez do usuário projetá-la e fixá-la manualmente. Isso pode ser feito com o uso de genótipos para codificar propriedades dinâmicas dos nós, permitindo que eles se adaptem durante sua vida útil [58]. A evolução das regras de aprendizado apresentam duas aplicações principais [12]: (i) descobrir novas regras de aprendizado, e (ii) permitir o aprendizado contínuo.

3.3.1.5 *Representação*

Os esquemas de representação utilizados pelas técnicas neuroevolutivas descrevem como (e quais) os componentes das RNAs são representados como genótipos de um algoritmo evolutivo. Esses esquemas podem ser classificados em relação ao grau e ao tipo de representação [12].

Com relação ao grau de representação, os esquemas podem ser divididos em [12]: (i) representação completa, e (ii) representação parcial. Na representação completa, cada genótipo descreve uma RNA completa. Esse genótipo pode assumir a forma de um vetor de pesos para uma topologia fixa de rede, ou pode apresentar uma descrição completa dos pesos, da topologia, e das funções dos nós. Na representação parcial, cada genótipo descreve um subcomponente de uma RNA, e.g., um nó ou peso individual. As RNAs completas são então construídas por meio da combinação de genótipos individuais.

Com relação ao tipo de representação, os esquemas podem ser divididos em [12]: (i) representação direta, (ii) representação indireta, e (iii) representação de desenvolvimento. Essa classificação descreve como cada genótipo é transformado em sua RNA correspondente. Na representação direta, o genótipo e o fenótipo são idênticos. Por exemplo, na evolução dos pesos de uma rede, o genótipo pode ser descrito como um vetor de valores reais, em que cada valor representa um peso. Dessa forma, para criar o fenótipo, esses valores são diretamente aplicados às conexões correspondentes, e portanto nenhuma decodificação é necessária. Na representação indireta, o genótipo deve ser decodificado para o componente que ele descreve. Por exemplo, se um genótipo descreve uma função matemática usada

para atribuir pesos a uma determinada RNA, então esses pesos não podem ser diretamente obtidos a partir do genótipo, mas calculados indiretamente por meio desse genótipo. Por fim, a representação de desenvolvimento foi criada para contornar algumas desvantagens das representações anteriores, que enfrentam dificuldades quando grandes RNAs precisam ser construídas. Assim, essa representação tenta resolver esse problema assumindo que redes maiores são construídas a partir de componentes menores. Dessa forma, seu genótipo descreve regras para o crescimento e adaptação das RNAs.

3.3.2 Características da neuroevolução

As características da neuroevolução são descritas a seguir. Primeiramente, tais técnicas podem ser aplicadas a uma grande quantidade de problemas distintos, tanto relacionados ao aprendizado supervisionado quanto ao aprendizado por reforço [12].

A segunda característica da neuroevolução é a independência do treinamento em relação à topologia, i.e., as técnicas neuroevolutivas não são limitadas pela topologia da RNA que está sendo treinada. Isso ocorre porque os métodos de treinamento neuroevolutivos não dependem do conhecimento acerca do estado interno da rede para executar sua evolução [12]. A única informação exigida é o comportamento geral da rede, por meio da avaliação de sua aptidão. Devido a essa característica, a neuroevolução pode atuar na otimização de RNAs de quaisquer topologias, como as RNAs recorrentes e as RNAs profundas. As RNAs recorrentes permitem conexões retroalimentadas, e as RNAs profundas são constituídas por muitas camadas intermediárias. Métodos determinísticos frequentemente encontram dificuldades para treinar esses tipos de rede [54, 59], o que caracteriza certa vantagem da neuroevolução frente a esses métodos.

Devido a esse caráter distribuído, os mecanismos de treinamento dos algoritmos neuroevolutivos podem ser facilmente paralelizados por meio de técnicas de computação paralela e distribuída, contribuindo para reduzir o tempo global de treinamento das redes. Outra característica da neuroevolução, e que foi citada anteriormente, é sua capacidade de treinar RNAs heterogêneas por meio da evolução de diferentes funções de transferência aplicadas aos nós da rede. Ao contrário dos métodos determinísticos, as técnicas neuroevolutivas podem facilmente lidar com RNAs heterogêneas, sem exigência quanto à diferenciabilidade das funções de transferência envolvidas.

Além disso, a neuroevolução apresenta a capacidade de manter a melhor solução atual durante o treinamento. Essa característica também é encontrada em outros métodos de treinamento de RNAs, como o *backpropagation*. A título de exemplo, considere uma aplicação que possua algum tipo de restrição de tempo para obtenção de uma solução. Nesse caso, é possível treinar uma RNA por meio da neuroevolução e obter alguma solução dentro desse intervalo de tempo. Findo o tempo estipulado, usa-se a melhor solução encontrada até então. Alguns métodos de treinamento de RNAs não possuem

essa propriedade [59], e portanto esses algoritmos ficam sem solução caso não haja tempo suficiente para concluírem suas execuções.

Por fim, outra característica da neuroevolução é sua capacidade de adaptar e encontrar topologias adequadas durante o treinamento. Essa característica integra um dos princípios da neuroevolução descritos anteriormente. A adaptação da topologia tem como principal benefício retirar a responsabilidade do usuário em escolher manualmente uma topologia adequada, e permitir que o método de busca automaticamente encontre a topologia que melhor atenda aos requisitos da tarefa em questão. Segundo Turner [12], existem duas abordagens para esse fim: (i) abordagens construtivas e (ii) abordagens destrutivas. As abordagens construtivas iniciam a execução com uma pequena quantidade de nós intermediários, tipicamente zero ou um, e construtivamente adicionam nós de forma incremental durante a busca. Essas abordagens são consideradas adequadas pois promovem o uso de um tamanho mínimo de rede, auxiliando na generalização da RNA, e mantendo baixa a dimensionalidade de busca. Por outro lado, as abordagens destrutivas iniciam a execução com uma grande quantidade de nós intermediários, e destrutivamente removem os nós considerados desnecessários. As abordagens destrutivas reduzem a complexidade da rede, o tempo de treinamento, e também contribuem para a generalização [12]. Devido ao caráter incremental da adição ou remoção dos nós, ambas as abordagens de busca tendem a se estagnar em topologias ótimas locais [55].

Diversos métodos neuroevolutivos podem ser encontrados na literatura. Alguns deles são: CGPANN [13], CoSyNE [60], COVNET [61], CNE [53], e NEAT [62]. O método CGPANN é a base deste trabalho, e será detalhado no Capítulo 4.

3.4 CONCLUSÕES DO CAPÍTULO

Esse capítulo introduziu a neuroevolução, iniciando com uma breve discussão acerca de seus componentes básicos, i.e., redes neurais artificiais e computação evolutiva. Foram descritos os aspectos estruturais e comportamentais das redes neurais artificiais, incluindo a formulação de seu método de treinamento tradicional, denominado *backpropagation*. Os princípios básicos da computação evolutiva também foram discutidos.

Em seguida, foram apresentados os princípios relativos à neuroevolução, juntamente com suas principais características. Como visto, as técnicas neuroevolutivas se destacam pela capacidade de ajustar a topologia, os pesos, as funções de transferência, e as regras de aprendizado das RNAs. Tais características diferem dos métodos determinísticos tradicionais, que tipicamente ajustam apenas os pesos das RNAs, mantendo os demais componentes fixos durante todo o processo de treinamento. O entendimento da neuroevolução se faz necessário para guiar a leitura dos próximos capítulos, que descrevem o modelo neuroevolutivo denominado CGPANN e introduzem os novos modelos neuroevolutivos híbridos propostos nesta dissertação.

4 PROGRAMAÇÃO GENÉTICA CARTESIANA DE REDES NEURAIS ARTIFICIAIS

A programação genética cartesiana de redes neurais artificiais (CGPANN) é uma extensão da programação genética cartesiana (CGP) dedicada à construção e ao treinamento de redes neurais artificiais. Por sua vez, a CGP é uma forma de programação genética (GP)[1]. O presente capítulo tem como principal objetivo introduzir a CGPANN, que é o algoritmo neuroevolutivo base para o desenvolvimento dos modelos híbridos deste trabalho.

O texto se encontra organizado da seguinte maneira. Primeiramente, são apresentados os conceitos fundamentais da GP na Seção 4.1, e da CGP na Seção 4.2. Em seguida, o modelo CGPANN é descrito em detalhes na Seção 4.3. Por fim, a Seção 4.4 conclui o capítulo com um resumo dos principais pontos abordados.

4.1 PROGRAMAÇÃO GENÉTICA

A GP [46, 63] é um tipo especial de algoritmo evolutivo desenvolvido para geração automática de programas. A partir de uma declaração dos requisitos necessários para a resolução de um dado problema, criam-se programas sem a programação explícita da estrutura da solução pelo usuário. Desde 1958, diversos trabalhos baseados nesse princípio têm sido desenvolvidos [64, 65, 66], contudo a técnica começou a se tornar amplamente conhecida em 1992, após a publicação de Koza [46].

De forma geral, a GP evolui uma população de programas de computador que, com o passar das gerações, estocasticamente se transformam em novos programas, potencialmente melhores que os programas das gerações anteriores. A qualidade de um programa depende de seu comportamento frente ao problema de interesse, i.e., depende de sua aptidão ao resolver o problema em questão. Os programas em GP são normalmente representados como estruturas de árvore. Nas formas mais avançadas, os programas são compostos de múltiplos componentes (e.g., sub-rotinas), representados como um conjunto de árvores agrupadas sob um nó raiz [63]. Para a criação de novos programas, os operadores genéticos de recombinação e mutação são aplicados aos programas existentes, gerando novas soluções a partir de suas sub-árvores e de seus nós. Mais detalhes sobre a programação genética podem ser encontrados em [46, 63].

[1] Do inglês, *Genetic Programming*.

4.2 PROGRAMAÇÃO GENÉTICA CARTESIANA

A programação genética cartesiana (CGP) [1, 67] é uma forma de GP desenvolvida por Miller et al. [14], originalmente proposta para evolução de circuitos digitais. O termo cartesiana é utilizado pois os programas são representados sob a forma de uma grade bidimensional de nós, indexados por suas coordenadas cartesianas. Esta seção descreve o esquema de representação da CGP, seu processo evolutivo, e suas principais características.

4.2.1 Representação

Cada programa da CGP representa um grafo acíclico direcionado, sendo seus nós organizados originalmente em r linhas e c colunas, formando uma grade bidimensional retangular de $r \times c$ nós, como ilustrado na Figura 5.

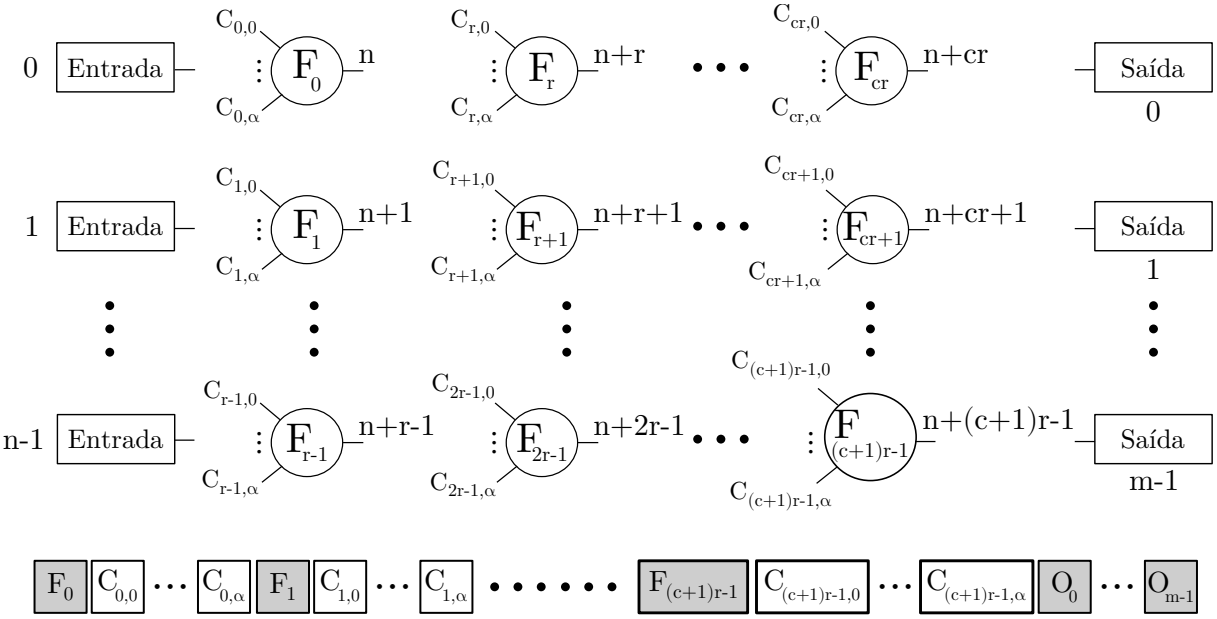


Figura 5 – Configuração padrão da CGP, representada por uma grade de nós com r linhas, c colunas, n entradas, m saídas, e α entradas por nó. O genótipo correspondente está ilustrado abaixo do grafo. Adaptado de [1].

Cada nó pode receber suas entradas apenas de nós de colunas anteriores, ou das entradas do programa. O parâmetro denominado *levels-back* determina a quantidade máxima de colunas imediatamente anteriores, cujos nós nelas presentes podem se conectar ao nó da coluna em questão. Por exemplo, se *levels-back* $= 2$, um nó só poderá receber como entradas os valores dos nós localizados nas duas colunas imediatamente anteriores à sua coluna. Esse parâmetro não impede os nós de se conectarem às entradas do programa.

A topologia de grade bidimensional, contudo, é desnecessariamente restritiva, dado que uma configuração mais geral pode ser obtida ao dispor os nós em uma única linha, com número de colunas igual ao número total de nós, e com *levels-back* igual ao número

de colunas. Assim, cada nó é capaz de se conectar com qualquer outro nó que se encontre à sua esquerda e com as entradas do programa.

A Figura 6 ilustra um exemplo de configuração não-restritiva da CGP.

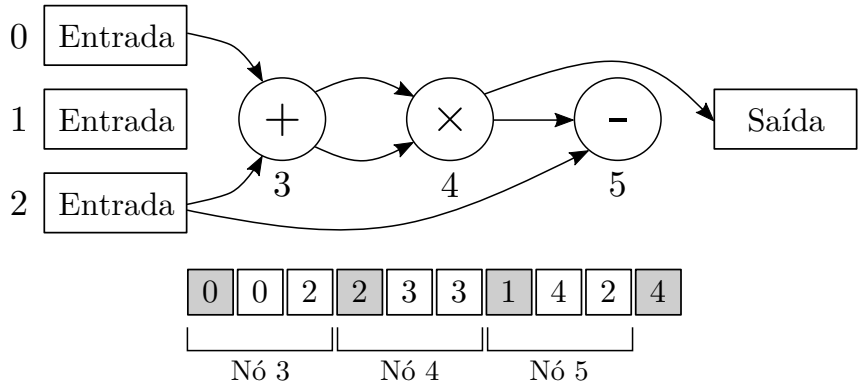


Figura 6 – Exemplo de uma configuração não-restritiva da CGP com três entradas, três nós, e uma saída. O Nó 5 está inativo. O genótipo correspondente está ilustrado abaixo do grafo: as três primeiras caixas hachuradas representam as funções de transferência dos respectivos nós conforme a tabela de consulta (0: adição, 2: multiplicação, 1: subtração); as caixas em branco representam os genes de conexão de cada nó; e a última caixa hachurada se refere à conexão de saída. A equação matemática resultante do grafo deste exemplo é $(I_0 + I_2)^2$, sendo I_0 e I_2 os nós de entrada 0 e 2, respectivamente.

O genótipo é composto por genes de função, genes de conexão, e genes de saída. Os genes de função descrevem a funcionalidade de cada nó, utilizando uma tabela de consulta para fazer a correspondência entre o código e sua função, e.g., o código "0" corresponde à operação de adição. Os genes de conexão definem de onde cada nó obtém suas entradas. Os genes de saída determinam quais nós são as saídas do programa. Os nós que não contribuem para o cálculo de pelo menos uma das saídas são denominados inativos, e portanto não precisam ser calculados durante a execução computacional do grafo.

O modelo CGP apresenta esquema de codificação direta, pois não requer a remoção dos nós inativos para sua execução. No entanto, devido à economia de tempo computacional, os nós inativos são geralmente removidos antes da execução, por meio da determinação recursiva dos nós ativos do grafo. De acordo com Turner [12], o tempo computacional necessário para a determinação dos nós ativos da CGP é muito menor que o tempo total de execução do algoritmo.

4.2.2 Esquema de evolução

A CGP utiliza, em sua versão original, o esquema de evolução elitista denominado Estratégia Evolutiva (ES) $(\mu + \lambda)$, ou simplesmente $(\mu + \lambda)$-ES [48], conforme descrito no Algoritmo 2. Inicialmente, $\mu + \lambda$ indivíduos são aleatoriamente gerados, atribuindo-se para cada gene um valor aleatório válido. Os μ melhores indivíduos são selecionados para

a próxima geração (denominados indivíduos-pais). Cada um dos μ indivíduos-pais cria λ/μ cópias de si próprio, e aplica mutação nestas cópias (denominadas indivíduos-filhos). Portanto, λ indivíduos-filhos são criados. A população da próxima geração é composta pelos μ indivíduos-pais e pelos λ indivíduos-filhos criados. Os processos de seleção e mutação se repetem até se atingir algum critério de parada predefinido. Ao final da evolução, o melhor indivíduo é escolhido como solução do problema. A Figura 7 ilustra o esquema $(\mu + \lambda)$-ES para $\mu = 1$ e $\lambda = 4$, que é uma parametrização comumente adotada na literatura [14, 67].

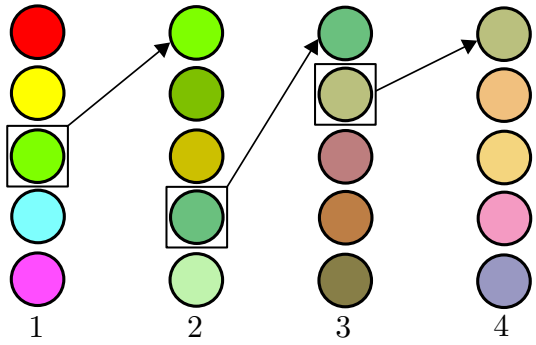


Figura 7 – Ilustração do esquema de evolução $(1 + 4)$-ES. Cada indivíduo é representado por um círculo. Inicialmente, 5 indivíduos são aleatoriamente gerados, como ilustrado na coluna 1. O melhor dentre eles, em destaque na figura, é selecionado para a próxima geração (indivíduo-pai), e os outros são eliminados. O indivíduo-pai cria 4 cópias de si próprio, e aplica mutação nestas cópias (indivíduos-filhos), representada pela coluna 2. Os processos de seleção e mutação se repetem até se atingir algum critério de parada predefinido. Ao final da evolução, o melhor indivíduo é escolhido como solução do problema.

A mutação pode ser do tipo ponto ou do tipo probabilística. Na mutação do tipo ponto, a quantidade de genes que sofrem mutação é previamente estabelecida pelo usuário. Assim, são selecionados aleatoriamente os genes do indivíduo até a quantidade máxima preestabelecida, e estes sofrem mutação, atribuindo-se para cada gene selecionado um valor aleatório válido. Por outro lado, na mutação do tipo probabilística, todo gene pode sofrer mutação com uma dada probabilidade, i.e., para cada gene é gerado um número aleatório no intervalo $[0, 1)$, e se este número for menor que a taxa de mutação predefinida, o gene sofrerá mutação. Nota-se que a mutação do tipo probabilística permite uma busca mais flexível no espaço de soluções [12], pois ao longo das execuções podem ocorrer mutações em nenhum, em alguns, ou em todos os genes do indivíduo, ao passo que a mutação do tipo ponto define a quantidade exata de genes que sofrerão mutação, restringindo de certa forma a busca por diferentes indivíduos. Nesta dissertação, a mutação do tipo probabilística é utilizada.

Algoritmo 2: $(\mu + \lambda)$-ES: Pseudocódigo [48]

```
1  Inicializa número de indivíduos-pais (μ) e indivíduos-filhos (λ)
2  P ← { }
3  for μ + λ vezes do
4      P ← P ∪ {novo indivíduo aleatório}
5  best ← □
6  while critério de parada não for atingido do
7      for cada indivíduo P_i ∈ P do
8          if best = □ ∨ fitness(P_i) > fitness(best) then
9              best ← P_i
10     Q ← os μ indivíduos de P que possuem as melhores aptidões
11     P ← Q
12     for cada indivíduo Q_j ∈ Q do
13         for λ/μ vezes do
14             P ← P ∪ {mutate(copy(Q_j))}
15 return best
```

4.2.3 Parâmetros

A CGP requer que os valores de alguns parâmetros sejam definidos pelo usuário. Tais parâmetros controlam o processo evolutivo, a topologia dos grafos, e as operações disponíveis para os nós. Portanto, a escolha dos parâmetros é fundamental para o funcionamento adequado do algoritmo. Para a execução da CGP em sua configuração não-restritiva (utilizando uma linha e várias colunas), os seguintes parâmetros devem ser predefinidos [1, 12]:

- Número máximo de nós;
- Número máximo de entradas por nó;
- Conjunto de funções disponíveis para os nós;
- Número de indivíduos-pais (μ);
- Número de indivíduos-filhos gerados a partir dos indivíduos-pais (λ);
- Critério de parada;
- Tipo de mutação;
- Taxa de mutação, que no caso da mutação do tipo ponto determina a proporção de genes que sofrerão mutação, e no caso da mutação do tipo probabilística determina a probabilidade de cada gene sofrer mutação.

4.2.4 Características

Algumas características da CGP descritas em [12] são aqui destacadas. A primeira característica é a reusabilidade dos valores calculados internamente, i.e., a saída de um nó pode ser utilizada por qualquer outro nó no programa. Nessa direção, a CGP também apresenta a capacidade de lidar com problemas de múltiplas entradas e múltiplas saídas (MIMO)[2], devido à sua configuração topológica flexível. Como muitos problemas do mundo real tipicamente envolvem tarefas do tipo MIMO, a CGP pode ser uma alternativa a se considerar ao lidar com tais problemas. A segunda característica da CGP é a capacidade dos indivíduos melhorarem a qualidade de suas soluções ao longo das gerações sem que isto implique necessariamente em um crescimento exacerbado no tamanho dos indivíduos. Essa característica é comumente conhecida na literatura como *resilience to bloat*, e ocorre justamente devido ao estabelecimento de um limite máximo para a quantidade de nós permitidos no fenótipo.

A terceira característica é a capacidade de se escapar de ótimos locais devida às mutações em nós inativos. Quando os nós dos indivíduos permanecem inativos após as mutações, alteram-se os genótipos e conservam-se os fenótipos. Portanto, embora as soluções percorram novas regiões do espaço de genótipos, tais deslocamentos não alteram seus valores de aptidão. Posteriormente, caso tais nós se tornem ativos após novas mutações, existirão indivíduos com diferentes fenótipos entre si, tornando o escape de ótimos locais mais provável de ocorrer, devido a uma melhor exploração do espaço busca por soluções candidatas distintas. Essa característica é conhecida na literatura como *neutral genetic drift*. Por fim, a quarta característica da CGP é sua vasta aplicabilidade em problemas de naturezas distintas, como projeto de circuitos digitais [14], processamento de imagens [68], classificação de imagens mamográficas [69], tolerância a falhas de sensores [70], e criação de artes visuais [71].

Em geral, a literatura não apresenta de forma explícita as desvantagens do método CGP. No entanto, alguns pontos podem ser destacados. Primeiramente, a CGP tradicional faz uso da mutação como o único operador genético, o que significa que o método não utiliza nenhum tipo de recombinação para a troca de informações genéticas entre indivíduos. De acordo com Yazdani & Shanbehzadeh [23], esse fato pode comprometer sua capacidade de exploração do espaço de busca. Já outros autores argumentam que a ausência de recombinação pode ser considerada uma característica da CGP, e não propriamente uma desvantagem [67]. Em segundo lugar, a CGP foi projetada para problemas de otimização discreta [1]. Portanto, o método pode potencialmente produzir resultados sub-ótimos quando aplicado a problemas de natureza contínua. Mais detalhes sobre as características, vantagens, e aplicações da CGP podem ser encontrados em [1, 12]

[2] Do inglês, *Multiple-Input and Multiple-Output*.

4.3 PROGRAMAÇÃO GENÉTICA CARTESIANA DE REDES NEURAIS

A programação genética cartesiana de redes neurais artificiais (CGPANN) [13, 72] é uma técnica neuroevolutiva proposta por Khan et al. [13], baseada na extensão da CGP para construção e treinamento de redes neurais artificiais. A CGPANN é o algoritmo-base para o desenvolvimento dos modelos híbridos deste trabalho. Esta seção descreve o esquema de representação da CGPANN, seu processo evolutivo, e suas principais características.

4.3.1 Representação

A CGPANN estende a CGP através da inclusão de genes de peso para cada gene de conexão do genótipo. A inclusão de pesos para cada conexão, juntamente com a possibilidade de uso de funções de transferência típicas de RNAs (e.g., sigmoide logística), tornam essa técnica adequada para a codificação de RNAs. Dessa forma, cada genótipo é representado como um vetor de valores inteiros e reais. Os valores inteiros definem as funções de transferência, as conexões entre os nós, e as saídas do programa. Os valores reais descrevem os pesos de cada conexão. A Figura 8 ilustra um exemplo de grafo da CGPANN. Note que nenhum gene de peso é atribuído aos genes de saída, uma vez que o papel dos genes de saída é indicar quais nós serão utilizados como saídas do programa. Todos os mecanismos de inicialização e mutação descritos para a CGP são também utilizados na CGPANN. Entretanto, diferentemente dos genes que assumem valores inteiros, os genes de peso são inicializados com valores reais aleatórios uniformemente distribuído num dado intervalo (e.g., $[-5, +5]$). De forma similar, a mutação nos genes de peso é realizada atribuindo-se um valor real aleatório válido dentro de um intervalo predefinido.

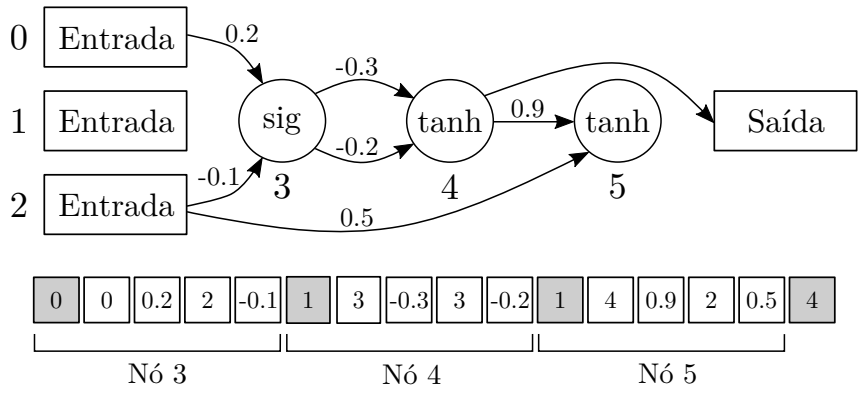


Figura 8 – Exemplo de uma configuração não-restritiva da CGPANN com três entradas, três nós, e uma saída. O Nó 5 está inativo. O genótipo correspondente está ilustrado abaixo do grafo. As três primeiras caixas hachuradas representam as funções de transferência dos respectivos nós conforme a tabela de consulta (0: logística sigmoide, 1: tangente hiperbólica); as caixas em branco representam os genes de conexão seguidos de seus pesos correspondentes; e a última caixa hachurada se refere à conexão de saída. Note que a única diferença em relação à CGP é a inclusão dos genes de peso para cada conexão.

4.3.2 Esquemas de classificação

A CGPANN e os demais modelos neuroevolutivos deste trabalho são utilizados como classificadores de dados, e podem adotar dois esquemas de classificação distintos, aqui denominados Esquema A e Esquema B. Em ambos os esquemas as entradas são tratadas da mesma forma. enquanto as saídas são tratadas diferentemente. Os esquemas são descritos a seguir.

4.3.2.1 *Esquema de Classificação A*

No Esquema de Classificação A, os indivíduos são criados de acordo com as características da base de dados considerada. O número de nós de entrada da rede corresponde ao número de atributos da base, e a quantidade de nós de saída da rede corresponde à quantidade de classes distintas da base.

Seja uma base de dados contendo D atributos e M classes. Portanto, cada indivíduo utilizado para classificação dessa base possuirá, necessariamente, D nós de entrada e M nós de saída. Nesse sentido, cada nó de saída $j \in \{0, 1, \ldots, M-1\}$ do indivíduo pode ser interpretado como uma função de pertinência $\sigma_j : \mathbb{R}^D \mapsto [L, U]$, que mapeia o vetor de entrada $\vec{x} \in \mathbb{R}^D$ para um escalar real dentro do intervalo fechado definido pelos limites L e U. A função σ_j representa o grau de pertinência da instância em relação à classe j. Dessa forma, a regra de decisão para classificação consiste em comparar as M funções de pertinência σ_j. A instância é classificada de acordo com o índice do nó de saída que apresentar o maior valor:

$$f(\vec{x}) = \underset{j}{\operatorname{argmax}}\, \sigma_j(\vec{x}) \ . \tag{4.1}$$

A título de exemplo, seja a classificação de uma instância $\vec{x}_i$ cuja base de dados possua três classes. Logo, cada indivíduo dessa população possui três nós de saída. Assim, se os valores das três saídas de um determinado indivíduo são $\sigma_0(\vec{x}_i) = 0.25$, $\sigma_1(\vec{x}_i) = 0.34$, e $\sigma_2(\vec{x}_i) = 0.09$, então a instância $\vec{x}_i$ é rotulada como "classe 1", i.e. $f(\vec{x}_i) = \{1\}$, pois a saída produzida pela função de pertinência σ_1 apresentou o maior valor. Nota-se que o Esquema de Classificação A pode ser aplicado tanto para classificação binária ($M = 2$) quanto para classificação multi-classe ($M > 2$).

4.3.2.2 *Esquema de Classificação B*

No Esquema de Classificação B, de forma similar ao esquema anterior, o número de nós de entrada da rede corresponde ao número de atributos da base de dados. Entretanto, cada rede possui apenas um nó de saída. Esse esquema é aplicado exclusivamente a problemas de classificação binária. Seja uma base de dados contendo D atributos e 2 classes. Neste caso, os indivíduos utilizados para classificação possuirão, necessariamente, D nós de entrada e 1 nó de saída. Assim, o nó de saída pode ser interpretado como uma

função de pertinência $\sigma : \mathbb{R}^D \mapsto [L, U]$, que mapeia o vetor de entrada $\vec{x} \in \mathbb{R}^D$ num real pertencente ao intervalo fechado definido pelos limites L e U. A função σ representa o grau de pertinência da instância em relação à classe positiva. Dessa forma, a regra de decisão para classificação consiste em definir um valor de limiar θ ($L \leq \theta \leq U$). A i-ésima instância com valor de saída $\sigma(\vec{x}_i)$ tal que $\theta \leq \sigma(\vec{x}_i) \leq U$ é classificada como pertencente à classe positiva. Analogamente, a j-ésima instância com valor de saída $\sigma(\vec{x}_j)$ tal que $L \leq \sigma(\vec{x}_j) < \theta$ é classificada como pertencente à classe negativa. Resumindo:

$$f(\vec{x}) = \begin{cases} 1 & \text{se } \theta \leq \sigma(\vec{x}) \leq U \\ 0 & \text{se } L \leq \sigma(\vec{x}) < \theta \ . \end{cases} \tag{4.2}$$

4.3.3 Esquema de seleção do indivíduo-final

O método CGPANN utiliza o mesmo esquema de evolução elitista da CGP, denominado $(\mu + \lambda)$-ES. Aqui, considera-se $\mu = 1$. Dado que este trabalho tem como foco a resolução de problemas de classificação, existe a necessidade de se garantir boa capacidade de generalização. Para isso, foram feitas modificações no esquema de seleção do indivíduo-final do processo evolutivo. Entende-se por indivíduo-final o indivíduo gerado e selecionado pelo algoritmo para classificar o conjunto de teste.

A base de dados é particionada em três conjuntos distintos: treinamento (TRN), validação (VLD), e teste (TST). O conjunto de treinamento é utilizado para avaliar a aptidão (função objetivo) dos indivíduos para a seleção do indivíduo-pai. O conjunto de validação é utilizado para evitar o sobreajuste dos dados, selecionando um indivíduo com potencial de classificar corretamente dados desconhecidos, i.e., ainda não vistos pelo modelo. O conjunto de teste é utilizado ao final da execução do algoritmo, para avaliar o desempenho do indivíduo-final selecionado pelo processo evolutivo. Sem perda de generalidade, na CGPANN visa-se a maximização da função objetivo.

O Algoritmo 3 descreve o pseudocódigo do esquema utilizado para a CGPANN. Antes da evolução, um indivíduo especial α é criado (linha 4), cuja função é armazenar o melhor indivíduo ao longo das gerações. Durante a evolução, cada indivíduo-filho avalia sua aptidão utilizando o conjunto de validação (linha 8). Se a aptidão do indivíduo-filho é maior que a aptidão do indivíduo especial α, então este indivíduo-filho é armazenado como indivíduo especial α (linhas 9–10). Para seleção do indivíduo-pai, é escolhido o indivíduo que apresentar a melhor aptidão sobre o conjunto de treinamento (linha 11). Os demais indivíduos são descartados. Para criar os λ indivíduos-filhos, o indivíduo-pai cria λ cópias de si próprio, e aplica mutação nessas cópias (linhas 12–13). As mutações podem afetar todos os genes do genótipo. Portanto, os indivíduos-filhos podem sofrer alterações em sua topologia, nas funções de seus nós, e nos pesos de suas conexões. Esse ciclo evolutivo se repete até se atingir a quantidade máxima de gerações predefinida. Após a evolução, o indivíduo especial α é selecionado como indivíduo-final para classificar o conjunto de teste,

utilizando uma ou mais métricas de interesse. Tais métricas determinarão o desempenho final do algoritmo (linha 14).

Algoritmo 3: CGPANN: Pseudocódigo

```
1  Inicializa parâmetros (λ, MaxGen, taxa de mutação, etc.)
2  Cria 1 + λ indivíduos diferentes
3  Avalia a função objetivo do indivíduo-pai (TRN)
4  Cria um indivíduo especial α para armazenar o melhor indivíduo
5  for gen = 1 to MaxGen do
6      for i = 1 to λ do
7          Avalia a função objetivo do indivíduo-filho i (TRN)
8          Avalia a função objetivo do indivíduo-filho i (VLD)
9          if Função obj. do indivíduo-filho i ≥ Função obj. do indivíduo α (VLD)
            then
10             Indivíduo α ← Indivíduo-filho i
11     Seleciona o melhor indivíduo dentre a população 1 + λ para a próxima geração
        (denominado indivíduo-pai) (TRN)
12     Cria λ cópias do indivíduo-pai selecionado (denominadas indivíduos-filhos)
13     Aplica mutação nos genes dos indivíduos-filhos
14 Avalia o indivíduo α utilizando uma ou mais métricas (TST)
```

4.3.4 Características

Por ser uma extensão da CGP, a CGPANN goza de todos os benefícios originais já estabelecidos pela teoria da CGP anteriormente mencionados, que incluem: a capacidade de lidar com problemas de múltiplas entradas e múltiplas saídas; a resiliência ao fenômeno de inchamento (*bloat*); a capacidade de se escapar de ótimos locais devido às mutações em nós inativos (*neutral genetic drift*); e a reutilização de valores calculados internamente.

Além disso, a CGPANN apresenta outras características relevantes, descritas em [12] e destacadas a seguir. A primeira característica é a forma não-convencional como os nós da CGPANN são organizados. Essa organização difere das estruturas tradicionais de rede, que são comumente construídas utilizando quantidades fixas de camada e de nós-por-camada, restringindo a maneira como os nós podem ser conectados (e.g., *multilayer perceptron* [73]). A estrutura flexível da CGPANN, juntamente com seu mecanismo de mutação, permitem a geração de redes diversificadas entre si, e como consequência contribuem para uma busca não-local no espaço das soluções.

A segunda característica da CGPANN é sua capacidade de lidar com restrições de topologia, impostas pela natureza do problema ou por decisão de projeto. Quando a representação original sob a forma de grade bidimensional é utilizada (definindo linhas, colunas, e *levels-back*), a CGPANN é capaz de restringir a busca por topologias indesejadas. Essa propriedade é útil para, por exemplo, a implementação de circuitos digitais ou

para a utilização de computação paralela no processamento das redes. Essas aplicações frequentemente impõem restrições sobre as arquiteturas de rede permitidas, a fim de se atingir determinados critérios de projeto ou de desempenho.

A terceira característica é a capacidade da CGPANN de permitir estruturas recorrentes em seus grafos, i.e., configurações topológicas cíclicas. Tais estruturas são muito utilizadas em problemas de natureza temporal, que necessitam de propriedades de memória e retroalimentação para seu funcionamento adequado. Em sua forma original, a CGPANN não permite estruturas recorrentes, mas pode ser facilmente estendida para tal. Essa propriedade também pode ser encontrada em outras técnicas neuroevolutivas [74, 75].

Finalmente, a quarta característica da CGPANN é a ampla gama de aplicações encontradas na literatura que utilizam essa técnica, e.g., controle de pêndulo invertido [13], reconstrução de sinais de áudio [76], detecção de câncer de mama [77], classificação de arritmia [78], e previsão na requisição de futuros clientes para alocação dinâmica de recursos em *cloud data center* [79]. Mais detalhes sobre as características, vantagens, e aplicações da CGPANN podem ser encontrados em [12, 72].

4.4 CONCLUSÕES DO CAPÍTULO

O presente capítulo teve como objetivo introduzir a técnica neuroevolutiva CGPANN, que é o algoritmo-base para o desenvolvimento dos modelos híbridos deste trabalho. Foram apresentados os conceitos básicos da GP e CGP. Por fim, a CGPANN foi descrita em detalhes, com seu esquema de representação, seu processo evolutivo, parâmetros, e suas principais características.

5 EVOLUÇÃO DIFERENCIAL

Este capítulo aborda o referencial teórico relativo à evolução diferencial (DE). A compreensão acerca do funcionamento do DE faz-se necessária na medida em que este é um dos componentes utilizados nas formulações híbridas desenvolvidas neste trabalho.

O capítulo se encontra organizado da seguinte forma. Inicialmente, na Seção 5.1, são introduzidos os conceitos básicos e a formulação clássica do DE. Em seguida, a Seção 5.2 descreve as principais características deste método. A Seção 5.3 apresenta algumas variantes do DE presentes na literatura. Finalmente, uma conclusão do capítulo é dada na Seção 5.4.

5.1 CONCEITOS BÁSICOS E FORMULAÇÃO

A evolução diferencial (DE), proposta em 1995 por Storn & Price [47], é uma técnica estocástica de otimização contínua, largamente utilizada na literatura pelo seu bom desempenho e por sua simplicidade de implementação. A principal característica do DE, e que o distingue dos demais algoritmos evolutivos, é seu esquema de criação de novas soluções baseado na diferença ponderada de vetores [80].

De forma genérica, o DE busca encontrar uma solução ótima global no espaço $\mathbb{R}^D$. As soluções são representadas por uma população de vetores D-dimensionais $\vec{x}_i = \left\{x_i^{(1)}, \ldots, x_i^{(D)}\right\}, i = 1, \ldots, DePop$, sendo $DePop$ a quantidade de vetores da população. Cada vetor $\vec{x}_i$, também conhecido como indivíduo, forma uma solução-candidata para o problema de otimização. As sucessivas iterações do algoritmo são denominadas gerações, denotadas por $t = 1, \ldots, DeIter$, sendo $DeIter$ a quantidade máxima de gerações. Dessa forma, a j-ésima componente do i-ésimo vetor da geração t é denotada por $x_{i,t}^{(j)}$.

O Algoritmo 4 descreve o pseudocódigo da versão clássica do DE, denominada DE/rand/1/bin. Sem perda de generalidade, o método é formulado visando a maximização da função objetivo. Os vetores são inicializados com números reais aleatórios dentro de um intervalo predefinido (linha 1). Em seguida, a aptidão de toda a população é calculada computando-se a função objetivo $f : \mathbb{R}^D \mapsto \mathbb{R}$ para cada vetor (linha 2). A cada iteração, um vetor $\vec{v}_i$ é criado a partir da mutação de três vetores aleatórios distintos $\vec{x}_{r_1}$, $\vec{x}_{r_2}$, e $\vec{x}_{r_3}$ tomados da população atual (linhas 5–10). A função random_integer(a, b) retorna um inteiro entre a e b, inclusive $(a, b \in \mathbb{N})$. Após a mutação, um vetor $\vec{u}_i$ (*trial vector*) é criado por meio da recombinação de componentes do vetor $\vec{x}_i$ (*target vector*) e de componentes do vetor $\vec{v}_i$ (*donor vector*), em que o parâmetro CR determina a taxa de recombinação dos vetores (linhas 13–17). A função random_float(a, b) retorna um valor real entre a e b, inclusive $(a, b \in \mathbb{R})$. Por fim, são comparados os valores de função objetivo dos vetores $\vec{u}_i$ e $\vec{x}_i$, sendo selecionado para a próxima geração o indivíduo mais apto (linhas 18–21).

O algoritmo encerra sua execução quando o critério de parada for satisfeito. Tipicamente, é adotado um dos seguintes critérios: (i) quando a quantidade de gerações atingir um valor predefinido, (ii) quando o valor da função objetivo do melhor indivíduo não se alterar significativamente ao longo das sucessivas gerações, ou (iii) quando o valor da função objetivo do melhor indivíduo atingir um valor predefinido. No Algoritmo 4, foi exemplificado o critério de parada baseado na quantidade máxima de gerações (linha 3).

Algoritmo 4: Evolução Diferencial: Pseudocódigo da versão DE/rand/1/bin

```
1  Inicializa a população de vetores (indivíduos)
   x_i = {x_i^(1), ..., x_i^(D)} ∀ i, i = 1, ..., DePop, em que D é o tamanho de cada vetor, e
   DePop é o número de vetores da população
2  Avalia a população: f(x_i) ∀ i, i = 1, ..., DePop
3  for t = 1 to DeIter do
4      for i = 1 to DePop do
5          Seleciona três vetores aleatórios dentre a população (r_1 ≠ r_2 ≠ r_3 ≠ i):
6          r_1 = random_integer(1, DePop)
7          r_2 = random_integer(1, DePop)
8          r_3 = random_integer(1, DePop)
9          Cria vetor v_i via mutação:
10         v_{i,t+1} ← x_{r_1,t} + F · (x_{r_2,t} − x_{r_3,t})
11         Seleciona uma componente aleatória j_random:
12         j_random = random_integer(1, D)
13         for j = 1 to D do
14             if (random_float(0, 1) < CR) ∨ (j = j_random) then
15                 u_{i,t+1}^(j) ← v_{i,t+1}^(j)
16             else
17                 u_{i,t+1}^(j) ← x_{i,t}^(j)
18         if f(u_{i,t+1}) ≥ f(x_{i,t}) then
19             x_{i,t+1} ← u_{i,t+1}
20         else
21             x_{i,t+1} ← x_{i,t}
```

5.2 CARACTERÍSTICAS

Desde sua concepção, o DE tem sido largamente utilizado em uma diversa gama de aplicações, e.g., projeto de antenas [81], fusão de imagens multi-foco [82], reconhecimento facial [83], e controle de frequência de carga em sistemas de potência [84]. As principais características do DE que justificam sua ampla aceitação são descritas a seguir.

Se comparado a outras meta-heurísticas, o DE é uma técnica relativamente simples de se implementar. Isso permite que pesquisadores de outras áreas, não familiarizados com ambientes de programação, possam implementar e ajustar esse algoritmo para resolver

seus problemas de interesse [80]. No entanto, tal simplicidade não implica em baixo desempenho, visto que diversos estudos [85, 86] indicam a superioridade do DE em relação a outras técnicas de otimização contínua, como PSO [87], CPSO-H [88], MA-S2 [89] e G3-PCX [90].

Outra importante característica do DE é sua pequena quantidade de parâmetros numéricos de controle ($DePop$, CR, e F). Como visto, $DePop$ é a quantidade de vetores da população, $CR \in [0, 1]$ é a taxa de recombinação, e F é o fator de escala que determina o tamanho do passo a ser dado na direção definida pelo vetor-diferença. Embora um ajuste fino em tais parâmetros melhore significativamente a qualidade das soluções, o DE se mostra robusto na medida em que pequenas variações nos valores desses parâmetros não provocam grandes variações no desempenho do algoritmo [91]. Mais detalhes sobre o funcionamento do DE, suas características e aplicações podem ser encontrados em [80, 91].

5.3 VARIANTES

A literatura tem mostrado uma ampla aceitação do DE em inúmeras áreas do conhecimento, o que tem resultado no surgimento de diversas variantes da versão clássica do algoritmo, com o objetivo de melhorar seu desempenho e ampliar sua aplicabilidade em cenários complexos [80]. Costuma-se denotar as diversas variantes do DE como DE/x/y/z, em que x especifica o vetor-base que sofre mutação, y determina o número de vetores-diferença utilizados na etapa de mutação, e z indica o esquema de recombinação adotado. A versão clássica do DE, descrita na Seção 5.1, é denotada por DE/rand/1/bin. Essa versão é utilizada nesta dissertação. Além dessa versão, a Tabela 3 ilustra outras variantes do DE e seus respectivos esquemas de mutação diferencial.

Tabela 3 – Algumas variantes do algoritmo de evolução diferencial.

Notação	**Mutação Diferencial**
DE/rand/1/bin	$\vec{v}_{i,t+1} = \vec{x}_{r_1,t} + F \cdot (\vec{x}_{r_2,t} - \vec{x}_{r_3,t})$
DE/best/1/bin	$\vec{v}_{i,t+1} = \vec{x}_{best,t} + F \cdot (\vec{x}_{r_1,t} - \vec{x}_{r_2,t})$
DE/mean/1/bin	$\vec{v}_{i,t+1} = \sum_{k=1}^{N} \vec{x}_{k,t} + F \cdot (\vec{x}_{r_1,t} - \vec{x}_{r_2,t})$
DE/rand-to-best/1/bin	$\vec{v}_{i,t+1} = \vec{x}_{r_1,t} + \lambda \cdot (\vec{x}_{best,t} - \vec{x}_{r_1,t}) + F \cdot (\vec{x}_{r_2,t} - \vec{x}_{r_3,t})$
DE/target-to-best/1/bin	$\vec{v}_{i,t+1} = \vec{x}_{i,t} + \lambda \cdot (\vec{x}_{best,t} - \vec{x}_{i,t}) + F \cdot (\vec{x}_{r_1,t} - \vec{x}_{r_2,t})$
DE/rand/2/bin	$\vec{v}_{i,t+1} = \vec{x}_{r_1,t} + \lambda \cdot (\vec{x}_{r_2,t} - \vec{x}_{r_3,t}) + F \cdot (\vec{x}_{r_4,t} - \vec{x}_{r_5,t})$

5.4 CONCLUSÕES DO CAPÍTULO

Neste capítulo, a técnica estocástica de otimização real denominada evolução diferencial (DE) foi apresentada. Foi vista em detalhes a formulação clássica do DE, cuja criação de novas soluções é baseada na adição de um vetor-base com a diferença ponderada de dois vetores distintos, seguido de um processo de recombinação de vetores. As principais características do DE foram descritas, justificando sua relevância em diversos trabalhos presentes na literatura. Por fim, algumas de suas variantes mais comuns foram brevemente abordadas. A compreensão dessa técnica se faz necessária devido a sua utilização nas formulações híbridas propostas neste trabalho.

6 MÉTODOS PROPOSTOS

A análise do modelo CGPANN realizada por Turner [12] demonstrou que o desempenho desse modelo foi insatisfatório quando aplicado a problemas de classificação. Neste capítulo, novos algoritmos de aprendizado são propostos com o objetivo de melhorar o desempenho da CGPANN nesse domínio de aplicação. A estratégia adotada no desenvolvimento dos algoritmos consistiu em reformular o treinamento padrão da CGPANN, partindo-se da ideia de se incorporar técnicas mais adequadas para a otimização dos genes que representam os pesos da RNA. Com base nesse ponto de vista, duas técnicas foram incorporadas à CGP: (i) evolução diferencial (DE), e (ii) *backpropagation* (BP). A hibridização dos modelos CGP e DE é denominada CGPDE. Analogamente, a combinação dos modelos CGP e BP é denominada CGPBP. Na parte final deste capítulo, é proposta uma métrica denominada Esforço de Avaliação, cujo papel é quantificar o volume de avaliações executadas pelos modelos considerando os dados de treinamento e validação, possibilitando comparações mais justas durante os testes de desempenho.

O texto se encontra organizado da seguinte maneira. A Seção 6.1 apresenta a motivação para o desenvolvimento dos algoritmos. Os modelos híbridos CGPDE e CGPBP são introduzidos, respectivamente, nas Seções 6.2 e 6.3. A Seção 6.4 levanta algumas considerações sobre os modelos propostos. A métrica denominada Esforço de Avaliação é descrita na Seção 6.5. Por último, a Seção 6.6 traz as conclusões do capítulo.

6.1 MOTIVAÇÃO

A CGP foi originalmente desenvolvida como uma técnica voltada para a otimização de problemas discretos [1]. Como visto no Capítulo 4, a posterior adaptação da CGP para representação de redes neurais artificiais (CGPANN) foi proposta por meio da inclusão de genes para representar os pesos da RNA, descritos por valores contínuos. Entretanto, o esquema de otimização adotado pela CGPANN é aplicado tanto para valores discretos quanto para valores contínuos [13]. Diante desse cenário, é razoável se levantar a hipótese de que o modelo CGPANN aplicado ao espaço de pesos pode levar a resultados sub-ótimos [92], visto que a otimização adotada não é apropriada para parâmetros de natureza contínua.

De fato, análises realizadas por Turner [12] demonstram o baixo desempenho do modelo CGPANN quando aplicado a problemas de classificação. Conforme Turner [12] elucida em seu trabalho,

> "[...] *it appears conclusive that CGPANN performs poorly in the domain of classification. This result applies generally in the field of classification and specifically compared to other methods for training ANNs.*"

Dada a relevância do problema, o desenvolvimento de soluções que busquem corrigir as deficiências da CGPANN diagnosticadas pela literatura é importante. Essa tarefa constitui a base de investigação desse trabalho. Para isso, são propostas formulações híbridas combinando a CGP com outras técnicas de otimização contínua. Nessas formulações, a CGP é responsável pela evolução da topologia das redes, i.e., genes de conexão e genes de saída. Por outro lado, a otimização dos genes de peso fica a cargo das técnicas de otimização contínua DE e BP. Sem perda de generalidade, todos os modelos são tratados visando a maximização da função objetivo. Vale ressaltar que os genes relativos às funções de transferência são fixados pelo usuário a priori e são mantidos constantes ao longo do processo evolutivo. Isso significa que, embora estejam presentes no genótipo, esses genes não são modificados pelos operadores de variação. Os detalhes sobre as formulações híbridas com o DE (CGPDE) e com o BP (CGPBP) são descritos a seguir.

6.2 MODELOS HÍBRIDOS BASEADOS EM CGP e DE

A CGP é um método voltado para otimização de problemas discretos. Por outro lado, a evolução diferencial (DE) é uma técnica estocástica de otimização em espaços contínuos. Na presente proposta, o modelo híbrido baseado na combinação dos modelos CGP e DE busca desacoplar os processos de otimização discreta e contínua. Sendo assim, a CGP é responsável pela evolução da topologia e o DE é responsável pela otimização dos pesos.

Cada indivíduo do modelo CGPDE é representado por duas estruturas de dados distintas: (i) genótipo completo, e (ii) vetor auxiliar de pesos. A Figura 9 ilustra ambas estruturas.

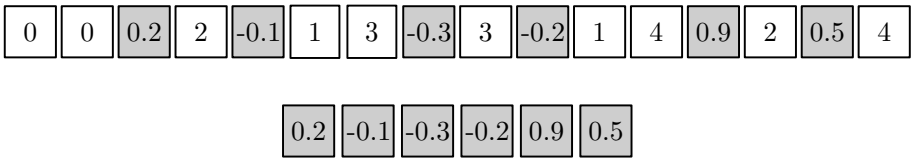


Figura 9 – Exemplo das duas estruturas de dados utilizadas por cada indivíduo do modelo CGPDE: genótipo completo (acima) e vetor auxiliar de pesos (abaixo). O genótipo completo contém todos os genes da rede (i.e., genes de conexão, peso, função e saída), e corresponde à representação vista na Subseção 4.3.1. Esse genótipo completo é utilizado para evolução da topologia e para avaliação da aptidão do indivíduo. O vetor auxiliar de pesos contém apenas os genes de peso da rede. Esse vetor auxiliar é utilizado para evolução dos pesos durante a execução do DE. Os genes de pesos dessas duas estruturas são correspondentes, i.e., quando os valores de peso se modificam no vetor auxiliar, eles também são modificados no genótipo completo.

O genótipo completo contém todos os genes da rede (i.e., genes de conexão, peso, função e saída), e corresponde à representação vista na Subseção 4.3.1. Esse genótipo

completo é utilizado para evolução da topologia e para avaliação da aptidão do indivíduo. O vetor auxiliar de pesos contém apenas os genes de peso da rede. Esse vetor auxiliar é utilizado para evolução dos pesos durante a execução do DE. Os genes de pesos dessas duas estruturas são correspondentes, i.e., quando os valores de peso são alterados no vetor auxiliar, eles também são modificados no genótipo completo.

Duas variantes baseadas na estratégia híbrida combinando CGP e DE são desenvolvidas. A primeira variante, denominada CGPDE-IN, executa os modelos CGP e DE de modo intercalado, i.e., o DE otimiza os pesos do melhor indivíduo de cada geração da CGP. A segunda variante, denominada CGPDE-OUT, executa o DE somente após o término da CGP. Detalhes sobre a implementação de cada variante são descritos a seguir.

6.2.1 Modelo CGPDE-IN

O modelo CGPDE-IN é a variante do esquema CGPDE que combina os modelos CGP e DE de modo intercalado, conforme descrito pelo Algoritmo 5. O termo "IN" se refere ao fato do DE ser executado **dentro** da estrutura de repetição principal da CGP.

Algoritmo 5: CGPDE-IN: Pseudocódigo

```
1  Inicializa parâmetros (λ, CgpGen, taxa de mutação, CR, F, etc.)
2  Cria 1 + λ indivíduos diferentes
3  Avalia a função objetivo do indivíduo-pai (TRN)
4  Cria um indivíduo especial α para armazenar o melhor indivíduo
5  for gen = 1 to CgpGen do
6      for i = 1 to λ do
7          Avalia a função objetivo do indivíduo-filho i (TRN)
8      Seleciona o melhor indivíduo-filho dentre os λ (denominado β) (TRN)
9      DE evolui pesos do indivíduo-filho β, mantendo topologia fixa (TRN)
10     Indivíduo-filho β ← Melhor indivíduo da população do DE (TRN)
11     Avalia a função objetivo do indivíduo-filho β (VLD)
12     if Função obj. do indivíduo-filho β ≥ Função obj. do indivíduo α (VLD) then
13         Indivíduo α ← Indivíduo-filho β
14     Seleciona o melhor indivíduo dentre a população 1 + λ para a próxima geração
         (denominado indivíduo-pai) (TRN)
15     Cria λ cópias do indivíduo-pai selecionado (denominadas indivíduos-filhos)
16     Aplica mutação na topologia dos indivíduos-filhos, mantendo pesos fixos
17 Avalia o indivíduo α utilizando uma ou mais métricas (TST)
```

Antes da evolução, um indivíduo especial α é criado (linha 4), cuja função é armazenar o melhor indivíduo ao longo das gerações. A cada geração, o melhor indivíduo-filho é selecionado utilizando o conjunto de treinamento (linhas 6–8). Em seguida, o DE

evolui os genes de peso desse indivíduo, mantendo fixos os genes referentes à sua topologia (linha 9). Para isso, é executada a versão DE/rand/1/bin, descrita no Capítulo 5. A população inicial do DE é formada por $DePop$ indivíduos, contendo o melhor indivíduo-filho selecionado e outros $DePop - 1$ indivíduos. Esses $DePop - 1$ indivíduos possuem a mesma topologia do indivíduo-filho, contudo seus genes de peso são gerados aleatoriamente (com distribuição uniforme) dentro de um intervalo. Durante a execução do DE, cada um dos $DePop$ indivíduos otimiza os pesos da rede utilizando o vetor auxiliar de pesos. Após $DeIter$ iterações, o DE retorna o indivíduo que apresentar a melhor aptidão avaliada utilizando o conjunto de treinamento (linha 10). A aptidão desse indivíduo retornado pelo DE é avaliada utilizando o conjunto de validação (linha 11). Se sua aptidão for maior que a aptidão do indivíduo especial α, então o indivíduo retornado pelo DE é armazenado como indivíduo especial α (linhas 12–13).

Em seguida, o mecanismo $(1 + \lambda)$-ES é empregado. Para seleção do indivíduo-pai, é escolhido o indivíduo que apresentar a melhor aptidão sobre o conjunto de treinamento (linha 14). Os demais indivíduos são descartados. Para criar os λ indivíduos-filhos, o indivíduo-pai cria λ cópias de si próprio, e aplica mutação nessas cópias (linhas 15–16). Entretanto, diferentemente da CGPANN, aqui a mutação é aplicada somente na topologia dos indivíduos, i.e., os genes de conexão e os genes de saída são modificados conforme o esquema de mutação adotado, mas os genes de peso permanecem inalterados. Esse ciclo evolutivo se repete até se atingir o número máximo de gerações predefinido. Após a evolução, o indivíduo especial α é selecionado como indivíduo-final para classificar o conjunto de teste, utilizando uma ou mais métricas de interesse. Tais métricas determinarão o desempenho final do algoritmo (linha 17).

Nota-se que o DE opera sobre uma topologia fixa a cada iteração da CGP (linha 9) para ajustar os pesos da rede, enquanto o modelo original CGPANN modifica simultaneamente ambos. Portanto, espera-se que essa abordagem híbrida forneça melhores classificadores, visto que permite que um indivíduo de cada geração encontre o conjunto de pesos que melhor se adapte à sua topologia.

6.2.2 Modelo CGPDE–OUT

O modelo CGPDE-OUT é a variante do esquema CGPDE que combina os modelos CGP e DE de modo separado, conforme descrito no Algoritmo 6. O termo “OUT” se refere ao fato do DE ser executado **fora** da estrutura de repetição principal da CGP.

Algoritmo 6: CGPDE-OUT: Pseudocódigo

```
1  Inicializa parâmetros (λ, CgpGen, taxa de mutação, CR, F, etc.)
2  Cria 1 + λ indivíduos diferentes
3  Avalia a função objetivo do indivíduo-pai (TRN)
4  Cria um indivíduo especial α para armazenar o melhor indivíduo
5  for gen = 1 to CgpGen do
6      for i = 1 to λ do
7          Avalia a função objetivo do indivíduo-filho i (TRN)
8          Avalia a função objetivo do indivíduo-filho i (VLD)
9          if Função obj. do indivíduo-filho i ≥ Função obj. do indivíduo α (VLD)
            then
10             Indivíduo α ← Indivíduo-filho i
11     Seleciona o melhor indivíduo dentre a população 1 + λ para a próxima geração
        (denominado indivíduo-pai) (TRN)
12     Cria λ cópias do indivíduo-pai selecionado (denominadas indivíduos-filhos)
13     Aplica mutação na topologia dos indivíduos-filhos, mantendo pesos fixos
14 DE evolui pesos do indivíduo α, mantendo topologia fixa (TRN)
15 Dentre a população evoluída pelo DE, seleciona o indivíduo com melhor função
     objetivo: TRN (versão CGPDE-OUT-T) ou VLD (versão CGPDE-OUT-V)
16 Avalia o indivíduo selecionado utilizando uma ou mais métricas (TST)
```

Antes da evolução, um indivíduo especial α é criado (linha 4), cuja função é armazenar o melhor indivíduo ao longo das gerações. Durante a evolução, cada indivíduo-filho avalia sua aptidão utilizando o conjunto de validação (linha 8). Se a aptidão do indivíduo-filho é maior que a aptidão do indivíduo especial α, então este indivíduo-filho é armazenado como indivíduo especial α (linhas 9–10). Para seleção do indivíduo-pai, é escolhido o indivíduo que apresentar a melhor aptidão sobre o conjunto de treinamento (linha 11). Os demais indivíduos são descartados. Para criar os λ indivíduos-filhos, o indivíduo-pai cria λ cópias de si próprio, e aplica mutação nessas cópias (linhas 12–13). Entretanto, diferentemente da CGPANN, aqui a mutação é aplicada somente na topologia dos indivíduos, i.e., os genes de conexão e os genes de saída são modificados conforme o esquema de mutação adotado, mas os genes de peso permanecem inalterados. Esse ciclo evolutivo se repete até se atingir o número máximo de gerações predefinido.

Após a evolução realizada pela CGP, os pesos do indivíduo especial α são otimizados pelo DE, mantendo a topologia fixa (linha 14). Para isso, é executada a versão DE/rand/1/bin, descrita no Capítulo 5. A população inicial do DE é formada por $DePop$ indivíduos, contendo o indivíduo especial α e outros $DePop - 1$ indivíduos. Esses $DePop - 1$ indivíduos possuem a mesma topologia do indivíduo especial α, contudo

seus genes de peso são gerados aleatoriamente (com distribuição uniforme) dentro de um intervalo. Durante a execução do DE, cada um dos *DePop* indivíduos otimiza os pesos da rede utilizando o vetor auxiliar de pesos. Após *DeIter* iterações, o DE retorna o indivíduo que apresentar a melhor aptidão (linha 15). Nesse ponto, duas políticas são propostas para selecionar o melhor indivíduo dentre os presentes na população do DE:

- CGPDE-OUT-T: seleciona o indivíduo que apresentar a melhor aptidão utilizando o conjunto de **treinamento**; ou
- CGPDE-OUT-V: seleciona o indivíduo que apresentar a melhor aptidão utilizando o conjunto de **validação**.

O indivíduo selecionado é utilizado para classificar o conjunto de teste, utilizando uma ou mais métricas de interesse. Tais métricas determinarão o desempenho final do algoritmo (linha 16). Em resumo, a principal motivação para o desenvolvimento dessa variante é utilizar a CGP para gerar topologias diversificadas, e apenas após encontrar uma topologia razoável, aplica-se o DE para otimizar os pesos que se ajustem a essa configuração topológica particular. Dado o exposto, espera-se que a política CGPDE-OUT-V forneça melhores classificadores que a política CGPDE-OUT-T, visto que a seleção utilizando o conjunto de validação pode melhorar a capacidade de generalização do modelo.

6.3 MODELOS HÍBRIDOS BASEADOS EM CGP e BP

O *backpropagation* é uma técnica determinística de otimização em espaços contínuos, largamente utilizada na literatura para o ajuste de pesos de RNAs em tarefas de aprendizado supervisionado [93]. O modelo híbrido baseado na combinação dos modelos CGP e BP busca desacoplar os processos de otimização discreta e contínua. Sendo assim, a CGP é responsável pela evolução da topologia e o BP é responsável pela otimização dos pesos. Cada indivíduo do modelo CGPBP é representado conforme o esquema visto na Subseção 4.3.1, cujo genótipo contém todos os genes da rede (i.e., genes de conexão, peso, função e saída).

De forma similar à CGPDE, duas variantes baseadas na estratégia híbrida combinando CGP e BP são desenvolvidas. A primeira variante, denominada CGPBP-IN, executa os modelos CGP e BP de modo intercalado, i.e., o BP otimiza os pesos do melhor indivíduo de cada geração da CGP. A segunda variante, denominada CGPBP-OUT, executa o BP somente após o término da CGP. Os detalhes sobre cada variante são descritos a seguir.

6.3.1 Modelo CGPBP-IN

O modelo CGPBP-IN é a variante do esquema CGPBP que combina os modelos CGP e BP de modo intercalado, conforme descrito pelo Algoritmo 7. O termo “IN” se

refere ao fato do BP ser executado **dentro** da estrutura de repetição principal da CGP.

Algoritmo 7: CGPBP-IN: Pseudocódigo

```
1  Inicializa parâmetros (λ, CgpGen, BpEpochs, taxa de mutação, etc.)
2  Cria 1 + λ indivíduos diferentes
3  Avalia a função objetivo do indivíduo-pai (TRN)
4  Cria um indivíduo especial α para armazenar o melhor indivíduo
5  for gen = 1 to CgpGen do
6      for i = 1 to λ do
7          Avalia a função objetivo do indivíduo-filho i (TRN)
8      Seleciona o melhor indivíduo-filho dentre os λ (denominado β) (TRN)
9      BP otimiza pesos do indivíduo-filho β, mantendo topologia fixa (TRN)
10     Avalia a função objetivo do indivíduo-filho β (VLD)
11     if Função obj. do indivíduo-filho β ≥ Função obj. do indivíduo α (VLD) then
12         Indivíduo α ← Indivíduo-filho β
13     Seleciona o melhor indivíduo dentre a população 1 + λ para a próxima geração
         (denominado indivíduo-pai) (TRN)
14     Cria λ cópias do indivíduo-pai selecionado (denominadas indivíduos-filhos)
15     Aplica mutação na topologia dos indivíduos-filhos, mantendo pesos fixos
16 Avalia o indivíduo α utilizando uma ou mais métricas (TST)
```

Antes da evolução, um indivíduo especial α é criado (linha 4), cuja função é armazenar o melhor indivíduo ao longo das gerações. A cada geração, o melhor indivíduo-filho é selecionado utilizando o conjunto de treinamento (linhas 6–8). Em seguida, o BP otimiza os pesos do indivíduo selecionado, mantendo sua topologia fixa (linha 9). Para a execução do BP, foi utilizada a variante *Full-Batch* GD, descrita no Capítulo 3. Nessa variante, os pesos são atualizados após o cálculo da função objetivo sobre todo o conjunto de treinamento. Isso significa que os pesos são atualizados apenas uma vez a cada época. O *Full-Batch* GD foi escolhido por ser um algoritmo clássico para ajuste de pesos. Após a execução do BP, a aptidão do indivíduo otimizado pelo BP é avaliada utilizando o conjunto de validação (linha 10). Se sua aptidão for maior que a aptidão do indivíduo especial α, então o indivíduo otimizado pelo BP é armazenado como indivíduo especial α (linhas 11–12).

Em seguida, o mecanismo $(1 + \lambda)$-ES é empregado. Para seleção do indivíduo-pai, é escolhido o indivíduo que apresentar a melhor aptidão sobre o conjunto de treinamento (linha 13). Os demais indivíduos são descartados. Para criar os λ indivíduos-filhos, o indivíduo-pai cria λ cópias de si próprio, e aplica mutação nessas cópias (linhas 14–15). Entretanto, diferentemente da CGPANN, aqui a mutação é aplicada somente na topologia dos indivíduos, i.e., os genes de conexão e os genes de saída são modificados conforme o esquema de mutação adotado, mas os genes de peso permanecem inalterados. Esse ciclo evolutivo se repete até se atingir o número máximo de gerações predefinido. Após a evolução, o indivíduo especial α é selecionado como indivíduo-final para classificar o

conjunto de teste, utilizando uma ou mais métricas de interesse. Tais métricas determinarão o desempenho final do algoritmo (linha 16).

Nota-se que o BP opera sobre uma topologia fixa a cada iteração da CGP (linha 9) para ajustar os pesos da rede, enquanto o modelo original CGPANN modifica simultaneamente ambos. Portanto, espera-se que essa abordagem híbrida forneça melhores classificadores, visto que permite que um indivíduo de cada geração encontre o conjunto de pesos que melhor se adapte à sua topologia.

6.3.2 Modelo CGPBP-OUT

O modelo CGPBP-OUT é a variante do esquema CGPBP que combina os modelos CGP e BP de modo separado, conforme descrito no Algoritmo 8. O termo "OUT" se refere ao fato do BP ser executado **fora** da estrutura de repetição principal da CGP.

Algoritmo 8: CGPBP-OUT: Pseudocódigo

```
1  Inicializa parâmetros (λ, CgpGen, BpEpochs, taxa de mutação, etc.)
2  Cria 1 + λ indivíduos diferentes
3  Avalia a função objetivo do indivíduo-pai (TRN)
4  Cria um indivíduo especial α para armazenar o melhor indivíduo
5  for gen = 1 to CgpGen do
6      for i = 1 to λ do
7          Avalia a função objetivo do indivíduo-filho i (TRN)
8          Avalia a função objetivo do indivíduo-filho i (VLD)
9          if Função obj. do indivíduo-filho i ≥ Função obj. do indivíduo α (VLD)
            then
10             Indivíduo α ← Indivíduo-filho i
11     Seleciona o melhor indivíduo dentre a população 1 + λ para a próxima geração
        (denominado indivíduo-pai) (TRN)
12     Cria λ cópias do indivíduo-pai selecionado (denominadas indivíduos-filhos)
13     Aplica mutação na topologia dos indivíduos-filhos, mantendo pesos fixos
14 BP otimiza pesos do indivíduo α, mantendo topologia fixa (TRN+VLD)
15 Avalia o indivíduo α utilizando uma ou mais métricas (TST)
```

Antes da evolução, um indivíduo especial α é criado (linha 4), cuja função é armazenar o melhor indivíduo ao longo das gerações. Durante a evolução, cada indivíduo-filho avalia sua aptidão utilizando o conjunto de validação (linha 8). Se a aptidão do indivíduo-filho é maior que a aptidão do indivíduo especial α, então este indivíduo-filho é armazenado como indivíduo especial α (linhas 9–10). Para seleção do indivíduo-pai, é escolhido o indivíduo que apresentar a melhor aptidão sobre o conjunto de treinamento (linha 11). Os demais indivíduos são descartados. Para criar os λ indivíduos-filhos, o indivíduo-pai cria λ cópias de si próprio, e aplica mutação nessas cópias (linhas 12–13). Entretanto, diferentemente da CGPANN, aqui a mutação é aplicada somente na topologia

dos indivíduos, i.e., os genes de conexão e os genes de saída são modificados conforme o esquema de mutação adotado, mas os genes de peso permanecem inalterados. Esse ciclo evolutivo se repete até se atingir o número máximo de gerações predefinido.

Após a evolução realizada pela CGP, os pesos do indivíduo especial α são otimizados pelo BP, mantendo a topologia fixa (linha 14). O BP é executado utilizando a união dos conjuntos de treinamento e validação, a fim de aumentar o número de amostras a serem treinadas pelo algoritmo. Para a execução do BP, foi utilizada a variante *Full-Batch* GD, descrita no Capítulo 3. Nessa variante, os pesos são atualizados após o cálculo da função objetivo para todo o conjunto de dados (treinamento + validação). Isso significa que os pesos são atualizados apenas uma vez a cada época. O *Full-Batch* GD foi escolhido por ser um algoritmo clássico para ajuste de pesos. Após a execução do BP, o indivíduo α é utilizado para classificar o conjunto de teste, utilizando uma ou mais métricas de interesse. Tais métricas determinarão o desempenho final do algoritmo (linha 15).

Em resumo, a principal motivação para o desenvolvimento dessa variante é utilizar a CGP para gerar topologias diversificadas, e apenas após encontrar uma topologia razoável, aplica-se o BP para otimizar os pesos que se ajustem a essa configuração topológica particular.

6.4 DISCUSSÃO

Os modelos híbridos propostos neste trabalho se baseiam fundamentalmente na mesma estratégia, i.e., desacoplar os processos de otimização da topologia e dos pesos das redes. Devido a isso, é notável a semelhança nas descrições dos modelos, diferenciando-se basicamente em relação às técnicas utilizadas para a otimização dos pesos. As variantes CGPDE utilizam a evolução diferencial, enquanto as variantes CGPBP utilizam o *backpropagation*.

Embora aparentemente pequena, essa diferença entre os modelos desencadeia uma importante consequência prática. Os métodos CGP e DE são meta-heurísticas evolutivas cujos indivíduos são selecionados com base em seus valores de aptidão, calculados utilizando uma função objetivo qualquer. Dessa forma, as variantes CGPDE são capazes de lidar com quaisquer funções objetivo, inclusive funções não-diferenciáveis.

Por outro lado, o BP é uma técnica determinística baseada no cálculo do gradiente e, portanto, é exigida a diferenciabilidade de sua função objetivo. Dessa forma, as variantes CGPBP apresentam aplicabilidade restrita, pois admitem apenas funções diferenciáveis em seu processo de otimização. Uma forma de relaxar tal restrição seria utilizar duas funções objetivo distintas: uma função objetivo necessariamente diferenciável para o BP, e outra função objetivo (sem restrição) para o método CGP. No entanto, tal estratégia não é comumente encontrada na literatura. Portanto, este trabalho opta pela via conservadora

e utiliza a mesma função objetivo nos dois métodos incorporados a cada modelo híbrido.

Outra fonte de possível não-diferenciabilidade ocorre na função de ativação de cada neurônio. Esse caso é análogo ao explicado anteriormente, i.e., as variantes CGPDE são capazes de lidar com quaisquer funções de ativação, inclusive funções não-diferenciáveis. Por outro lado, as variantes CGPBP exigem a diferenciabilidade dessas funções.

6.5 ESFORÇO DE AVALIAÇÃO

Várias avaliações da função objetivo são executadas ao longo do processo evolutivo. Com o intuito de realizar comparações justas entre os algoritmos, é proposta uma métrica denominada Esforço de Avaliação (EE), que quantifica o "volume" de avaliações da função objetivo realizadas sobre os diferentes conjuntos de dados. Nessa métrica, uma avaliação da função objetivo sobre o conjunto de treinamento é contabilizada como "esforço unitário", e uma avaliação da função objetivo sobre o conjunto de validação é contabilizada como "esforço p", em que p é a razão entre os tamanhos dos conjuntos de validação e treinamento. Dessa forma, o EE tenta capturar a quantidade total de instâncias utilizadas para o cálculo da função objetivo.

A título de exemplo, considere o seguinte cenário. Durante a execução de um dado algoritmo, a função objetivo foi avaliada quinze vezes utilizando o conjunto de treinamento, e dez vezes utilizando o conjunto de validação, sendo o conjunto de treinamento duas vezes maior que o conjunto de validação (i.e., $p = 0.5$). Nesse caso, portanto, o esforço de avaliação é dado por: $(15 \times 1) + (10 \times 0.5) = 20$. Nos experimentos deste trabalho, os conjuntos de validação e treinamento são formados por, respectivamente, 2 e 7 partes, sendo que cada parte contém a mesma quantidade de instâncias, conforme descrito na Seção 7.2. Portanto, $p = 2/7$.

Na Tabela 4, as Equações 6.1–6.5 descrevem, respectivamente, o EE dos modelos CGPANN, CGPDE-IN, CGPDE-OUT, CGPBP-IN, e CGPBP-OUT, em que: $CgpGen$ é a quantidade máxima de gerações; λ é a quantidade de indivíduos-filhos do esquema $(1 + \lambda)$-ES; $DeIter$ é a quantidade máxima de iterações do DE; $DePop$ é a quantidade de indivíduos da população do DE; e $BpEpochs$ é a quantidade máxima de épocas executadas pelo BP.

Tabela 4 – Esforço de avaliação dos modelos neuroevolutivos.

Eq.	Esforço de Avaliação
(6.1)	$EE_{CGPANN} = CgpGen \cdot \lambda + \frac{2}{7} \cdot CgpGen \cdot \lambda$
(6.2)	$EE_{CGPDE-IN} = CgpGen \cdot (\lambda + DeIter \cdot DePop) + \frac{2}{7} \cdot CgpGen$
(6.3)	$EE_{CGPDE-OUT} = CgpGen \cdot \lambda + \frac{2}{7} \cdot CgpGen \cdot \lambda + DeIter \cdot DePop$
(6.4)	$EE_{CGPBP-IN} = CgpGen \cdot (\lambda + BpEpochs) + \frac{2}{7} \cdot CgpGen$
(6.5)	$EE_{CGPBP-OUT} = CgpGen \cdot \lambda + \frac{2}{7} \cdot CgpGen \cdot \lambda + BpEpochs + \frac{2}{7} \cdot BpEpochs$

6.6 CONCLUSÕES DO CAPÍTULO

Novos modelos neuroevolutivos híbridos para lidar com o problema de aprendizado supervisionado no contexto de classificação foram apresentados: (i) CGPDE e (ii) CGPBP. Ambos possuem estratégias de desacoplamento entre os processos de otimização discreta (topologia) e contínua (pesos), visando melhorar o desempenho de classificação em relação ao modelo de referência (CGPANN).

Em ambos os modelos propostos, a CGP é utilizada para evolução das topologias de rede. O modelo CGPDE é baseado na utilização da evolução diferencial para a otimização dos pesos. Duas variantes CGPDE foram apresentadas: CGPDE-IN e CGPDE-OUT. A variante CGPDE-IN executa os métodos CGP e DE de modo intercalado, enquanto a variante CGPDE-OUT executa o DE somente após o término da CGP. Analogamente, o modelo CGPBP é baseado na utilização do *backpropagation* para a otimização dos pesos. Duas variantes CGPBP foram apresentadas: CGPBP-IN e CGPBP-OUT. A variante CGPBP-IN executa os métodos CGP e BP de modo intercalado, enquanto a variante CGPBP-OUT executa o BP somente após o término da CGP.

Na parte final do capítulo, uma métrica denominada Esforço de Avaliação foi apresentada, cujo papel é quantificar o volume de avaliações executadas pelos modelos, a fim de possibilitar comparações mais justas durante os testes de desempenho.

7 EXPERIMENTOS COMPUTACIONAIS

O presente capítulo apresenta os experimentos computacionais realizados utilizando os modelos híbridos propostos neste trabalho. Pretende-se analisar o comportamento desses modelos sob diferentes funções objetivo e compará-los a métodos conhecidos na literatura quando aplicados a problemas de classificação.

O texto se encontra organizado da seguinte forma. A Seção 7.1 descreve as bases de dados utilizadas nos experimentos. As formas adotadas para divisão dos dados e para comparação dos modelos são apresentadas na Seção 7.2. A Seção 7.3 introduz o perfil de desempenho [94, 95], utilizado como instrumento de visualização e análise de resultados. Os algoritmos utilizados nos experimentos são descritos na Seção 7.4. Os experimentos computacionais são apresentados nas Seções 7.5, 7.6, e 7.7. Esses experimentos são divididos com base nas funções objetivo utilizadas pelos classificadores. Nessa divisão, a Seção 7.5 realiza o estudo dos modelos utilizando a acurácia como função objetivo. A Seção 7.6 realiza o estudo dos modelos utilizando o erro quadrático médio como função objetivo. A Seção 7.7 utiliza os novos modelos híbridos para estudar o desempenho de diversas funções objetivo em cenários de classificação desbalanceada. Por fim, uma discussão geral acerca dos resultados obtidos é realizada na Seção 7.8, que também traz as conclusões do capítulo.

7.1 BASES DE DADOS

As bases de dados utilizadas nos experimentos foram retiradas do repositório UCI [96], e estão descritas nas Tabelas 5 e 6. As bases da Tabela 5 foram utilizadas nos experimentos das Seções 7.5 e 7.6. As bases da Tabela 6 foram utilizadas nos experimentos em cenários desbalanceados da Seção 7.7. Em todas as bases de dados descritas, cada atributo foi normalizado linearmente para o intervalo $[0, 1]$.

Tabela 5 – Características das bases de dados utilizadas nos experimentos de classificação das Seções 7.5 e 7.6: número de instâncias, número de atributos, e número de classes.

Base de dados	**Instâncias**	**Atributos**	**Classes**
Glass Identification	214	9	6
Pima Indians Diabetes	768	8	2
Iris	150	4	3
Wisconsin Breast Cancer [97]	683	9	2
Ionosphere	351	34	2
Vertebral Column	310	6	3
Wine	178	13	3

Tabela 6 – Características das bases de dados utilizadas nos experimentos de classificação binária em cenários desbalanceados da Seção 7.7: número de instâncias, número de atributos, e grau de balanceamento.

Base de dados	**Instâncias**	**Atributos**	**Grau de balanceamento** [%]
Pima Indians Diabetes	768	8	34.90
Wine (c3 vs. all)	178	13	26.97
Vertebral (dh vs. all)	310	6	19.35
Ecoli (imu vs. all)	336	7	10.42
Ecoli (om vs. all)	336	7	5.95
Yeast (vac vs. all)	1484	8	2.02
Yeast (me1 vs. all)	1484	8	2.96
Yeast (me2 vs. all)	1484	8	3.44
Glass (hl vs. all)	214	9	13.55
Glass (vw vs. all)	214	9	7.94
Glass (cnt vs. all)	214	9	6.07

Na Tabela 6, as bases que originalmente possuíam mais de duas classes foram reduzidas a problemas de classificação binária, rotulando uma única classe como positiva e todas as outras classes como negativa. Essa abordagem é conhecida como *one-versus-all* ou *one-against-rest* [9]. Utilizou-se esse esquema para gerar variações binárias dos problemas que em suas formas originais eram multi-classe. Por exemplo, a base de dados *Wine*, que originalmente possuía 3 classes distintas (“c1”, “c2”, “c3”), foi transformada em uma base binária ao rotular a classe “c3” como positiva e as outras duas classes como negativa. Dessa forma, é criada a base de dados denominada “Wine (c3 vs. all)”.

Além disso, a Tabela 6 apresenta, para cada base, o grau de balanceamento da classe positiva, i.e., a razão entre a quantidade de instâncias da classe positiva e a quantidade total de instâncias. Bases com diferentes graus de balanceamento foram escolhidas para avaliar o desempenho de classificação em problemas de naturezas distintas.

7.2 DESCRIÇÃO GERAL DOS EXPERIMENTOS

Todos os experimentos computacionais realizados neste capítulo aplicam o método estatístico de reamostragem denominado validação cruzada 10-*fold* estratificada [98]. Essa técnica é utilizada para avaliar e comparar o desempenho de algoritmos de aprendizado de máquina, verificando a capacidade de generalização dos modelos, i.e., a capacidade de classificar corretamente as instâncias que não estavam presentes durante a etapa de treinamento. Para tal, a base é dividida em três conjuntos distintos: treinamento, validação, e teste.

Esse esquema de divisão é exemplificado pela Figura 10 e descrito a seguir. As instâncias da base são distribuídas aleatoriamente em 10 partes iguais (*folds*), mantendo-se

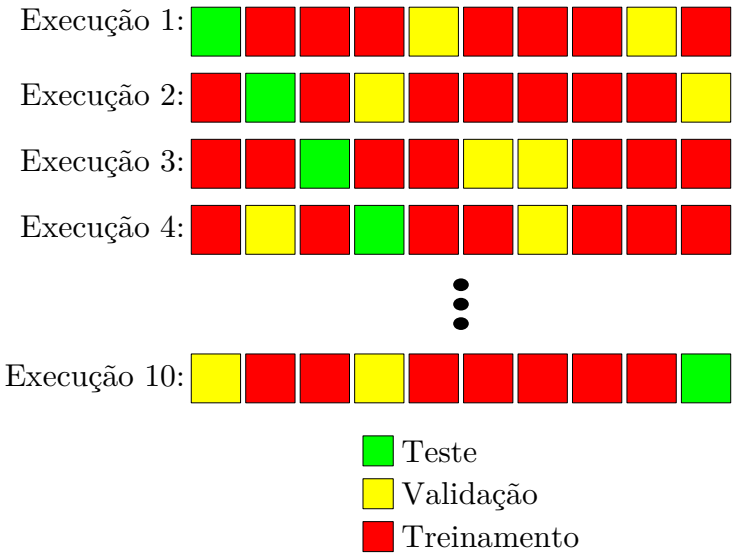


Figura 10 – Validação cruzada 10-*fold* estratificada. A cada execução, as instâncias da base são distribuídas aleatoriamente em 10 partes iguais (*folds*), mantendo-se a mesma proporção de classes em cada parte. Inicialmente (primeira linha da figura), a parte #1 é reservada para o conjunto de teste (em verde), e as 9 partes restantes são distribuídas aleatoriamente: 7 partes para o conjunto de treinamento (em vermelho) e 2 partes para o conjunto de validação (em amarelo). Formados os conjuntos, o algoritmo realiza sua primeira execução completa. Na próxima execução do algoritmo (segunda linha da figura), a parte #2 é reservada para o conjunto de teste, enquanto as partes restantes são divididas aleatoriamente entre os conjuntos de treinamento (7 partes) e validação (2 partes). Esse procedimento é executado 10 vezes, sendo que a cada execução uma parte diferente é utilizada como conjunto de teste, e as partes restantes divididas aleatoriamente entre os conjuntos de treinamento e validação. Neste trabalho, para aumentar o número de resultados a serem analisados, o esquema de validação cruzada foi realizado três vezes, resultando em 30 execuções completas.

a mesma proporção de classes em cada parte. Inicialmente (primeira linha da figura), a parte #1 é reservada para o conjunto de teste (em verde), e as 9 partes restantes são distribuídas aleatoriamente: 7 partes para o conjunto de treinamento (em vermelho) e 2 partes para o conjunto de validação (em amarelo). Formados os conjuntos, o algoritmo realiza sua primeira execução completa.

Na próxima execução do algoritmo (segunda linha da figura), a parte #2 é reservada para o conjunto de teste, enquanto as partes restantes são divididas aleatoriamente entre os conjuntos de treinamento (7 partes) e validação (2 partes). Esse procedimento é executado 10 vezes, sendo que a cada execução uma parte diferente é utilizada como conjunto de teste, e as partes restantes divididas aleatoriamente entre os conjuntos de treinamento e validação. Para uma comparação justa, todos os algoritmos que estão sendo avaliados utilizam os mesmos conjuntos de treinamento, validação e teste.

Como as técnicas de busca aqui implementadas possuem natureza estocástica, é desejável que a quantidade de resultados obtidos seja grande o suficiente para uma análise estatística mais assertiva. Para isso, o esquema de validação cruzada descrito acima é realizado três vezes. Portanto, 30 execuções completas são realizadas.

Para efeitos de comparação entre os modelos, uma prática comumente adotada na literatura consiste em calcular, para cada modelo, o valor médio obtido ao classificar os conjuntos de teste das 30 execuções, utilizando uma dada métrica de desempenho. Além disso, foram realizadas análises estatísticas comparativas via teste-t pareado unilateral à direita [99] utilizando o *Statistics and Machine Learning Toolbox* do MATLAB [100]. Nessas análises, considerando a comparação entre dois métodos A e B, as seguintes hipóteses foram formuladas:

- Hipótese nula (H_0): o valor médio do método A é **igual** ao valor médio do método B.
- Hipótese alternativa (H_1): o valor médio do método A é **maior** que o valor médio do método B.

As bibliotecas *CGP-Library* [101] e *Fast Artificial Neural Network Library* (FANN) [102] foram utilizadas para o desenvolvimento dos modelos híbridos. As implementações foram realizadas em linguagem C. A biblioteca *CGP-Library* foi utilizada para o desenvolvimento das funções relacionadas ao método CGP dos modelos CGPDE e CGPBP. Por outro lado, a biblioteca FANN foi utilizada para o desenvolvimento das funções relacionadas ao método BP do modelo CGPBP. Todos os modelos neuroevolutivos implementados adotaram a configuração topológica não-restritiva descrita no Capítulo 4.

O código-fonte referente ao modelo CGPDE está disponível em `https://github.com/johnathanmelo/cgpde-lib`. O código-fonte referente ao modelo CGPBP está disponível em `https://github.com/johnathanmelo/cgpbp-lib`. Finalmente, o código-fonte referente aos experimentos em cenários desbalanceados apresentados na Seção 7.7 está disponível em `https://github.com/johnathanmelo/cgpde-lib-imbalanced`.

7.3 PERFIL DE DESEMPENHO

O perfil de desempenho [94, 95] é um instrumento de visualização e análise utilizado para apresentar os resultados de uma forma compacta, sobretudo quando existe uma grande quantidade de métodos e sobretudo de problemas a serem considerados durante os experimentos. Considerando o conjunto S de métodos s_i, $i \in \{1, \ldots, n_s\}$, e o conjunto P de problemas p_j, $j \in \{1, \ldots, n_p\}$, pode-se definir o índice de desempenho $t_{p,s}$ como $t_{p,s} = 1 - m_{p,s}$, em que $m_{p,s} \in [0, 1)$ e é considerado aqui como a mediana de um conjunto de resultados avaliados sob uma dada métrica (e.g., acurácia) quando o método $s \in S$ é aplicado ao problema $p \in P$. Ou seja, considerando um problema de maximização, quanto

menor o valor de $t_{p,s}$, **maior** a mediana dos resultados, e portanto **melhor** o desempenho do método. A razão de desempenho $r_{p,s}$ é definida como

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}} \ , \tag{7.1}$$

que descreve o fator de desempenho relativo do método $s \in S$ em relação ao método de melhor desempenho, quando aplicados ao problema $p \in P$. Utilizando a razão de desempenho $r_{p,s}$, e denotando a cardinalidade de um conjunto A por $|A|$, temos que

$$\rho_s(\tau) = \frac{1}{n_p}|\{p \in P : r_{p,s} \leq \tau\}| \ , \tag{7.2}$$

em que $\rho_s(\tau)$ é a probabilidade da razão de desempenho $r_{p,s}$ do método $s \in S$ estar dentro de um fator τ da melhor razão possível, sendo $1 \leq \tau \leq \tau_{max}$ [95]. Em outras palavras, $\rho_s(\tau)$ descreve a fração dos problemas em P tal que o método $s \in S$ é capaz de resolver em um fator $\tau \in [1, \tau_{max}]$ em relação ao melhor desempenho observado. Por exemplo, $\rho_s(1)$ representa a fração de problemas em P nos quais o método s apresenta o melhor desempenho entre todos os métodos em S. Dessa forma, ao variar o fator τ, o perfil de desempenho $\rho_s : [1, \tau_{max}] \mapsto [0, 1]$ pode ser traçado. Portanto, a área abaixo da curva ρ_s é um indicador geral de desempenho do método s no conjunto de problemas P, i.e., quanto maior a área, mais eficiente o método. Mais detalhes sobre os perfis de desempenho podem ser encontrados em [94, 95].

7.4 ALGORITMOS CONSIDERADOS

Os seguintes métodos de classificação foram utilizados ao longo dos experimentos: CGPDE-IN, CGPDE-OUT (versões T e V), CGPBP-IN, CGPBP-OUT, CGPANN [13], e *Gradient Descent Backpropagation Feedforward Network* (BP) [103]. Os métodos CGPDE-IN, CGPDE-OUT, CGPBP-IN, e CGPBP-OUT estão descritos no Capítulo 6. O método CGPANN foi executado utilizando a biblioteca *CGP-Library* [101], e foi incluído nos experimentos por ser o algoritmo-base dos modelos híbridos propostos. O método BP foi executado utilizando o pacote *Neural Network Toolbox* do MATLAB [104, 105], e foi incluído por ser um algoritmo de classificação baseado em RNAs tradicionalmente utilizado como referência comparativa [12, 106, 107, 108].

Os parâmetros utilizados pelos métodos neuroevolutivos estão descritos na Tabela 7. Para uma comparação justa, as gerações da CGP, as iterações do DE, a população do DE, e as épocas do BP foram escolhidas de tal forma que o Esforço de Avaliação dos métodos fossem similares entre si (vide Tabela 4). Os valores de F e CR foram definidos com base na recomendação de [109], e os demais parâmetros foram definidos utilizando [12] como referência. Uma vez estabelecidos tais valores, nenhuma otimização paramétrica foi realizada. Optou-se por manter as funções de transferência constantes, visto que permitir a evolução dessas funções incluiria mais um grau de liberdade aos algoritmos, e portanto

Tabela 7 – Parâmetros utilizados pelos algoritmos neuroevolutivos.

Parâmetro	**CGPANN**	**CGPDE**		**CGPBP**	
		IN	**OUT**	**IN**	**OUT**
Gerações CGP	50000	64	40000	64	40000
Iterações DE	-	400	2570	-	-
População DE	-	10	20	-	-
F	-	0.70	0.70	-	-
CR	-	0.90	0.90	-	-
Épocas BP	-	-	-	4000	40000
Esquema de mutação	probabilística				
Taxa de mutação	0.05				
Função de transferência	sigmoide logística				
Estratégia evolutiva	(1+4)-ES				
Número de nós	500				
Número de entradas por nó	20				
Intervalo inicial dos pesos	[-5,+5]				

dificultaria uma análise mais assertiva acerca da real influência da otimização dos pesos, que é a proposta central deste trabalho. O algoritmo BP utilizou a configuração padrão do MATLAB, descrita em [104, 105], com 500 neurônios na camada escondida. Para os experimentos da Seção 7.5, o BP utilizou a função objetivo do tipo entropia cruzada. Para os experimentos da Seção 7.6, o BP utilizou a função objetivo do tipo MSE.

7.5 ESTUDO DE DESEMPENHO DOS ALGORITMOS PROPOSTOS USANDO ACURÁCIA

A acurácia (Eq. 2.6) é comumente utilizada na avaliação e seleção de modelos de classificação [9, 12, 25]. Nos estudos desta seção, os modelos neuroevolutivos utilizaram a acurácia como função objetivo e como métrica avaliadora dos conjuntos de teste. As bases de dados apresentadas na Tabela 5 foram utilizadas para os experimentos, sendo estes realizados conforme descrito na Seção 7.2. Os modelos neuroevolutivos considerados nesse estudo adotam o Esquema de Classificação A (vide Subseção 4.3.2). Além disso, vale destacar que os modelos híbridos CGPBP não foram analisados, visto que exigem a diferenciabilidade da função objetivo [36] e, portanto, não são capazes de otimizar diretamente a acurácia.

7.5.1 Análise comparativa

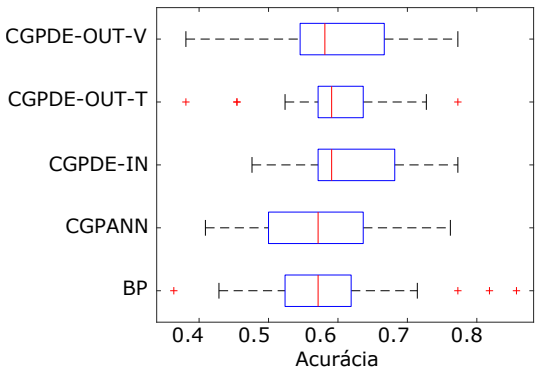
Na presente análise, foram comparados os algoritmos CGPDE-IN, CGPDE-OUT (versões T e V), CGPANN, e BP. Embora o BP utilize entropia cruzada como função objetivo, esse método foi incluído por ser uma referência comparativa tradicionalmente

utilizada na literatura [12, 106, 107, 108]. Para cada base, as acurácias obtidas a partir dos 30 conjuntos de teste estão ilustradas nos diagramas de caixa da Figura 11. Em geral, pode-se notar que as medianas dos resultados obtidos pelos métodos híbridos são superiores às encontradas pelos métodos BP e CGPANN. Além disso, a variabilidade dos valores de acurácia é mais proeminente no método CGPANN (e.g., nas bases *Diabetes*, *Iris*, e *Wine*). Já em outras bases, diferentes métodos apresentaram distribuição e amplitude interquartílica igualmente amplas.

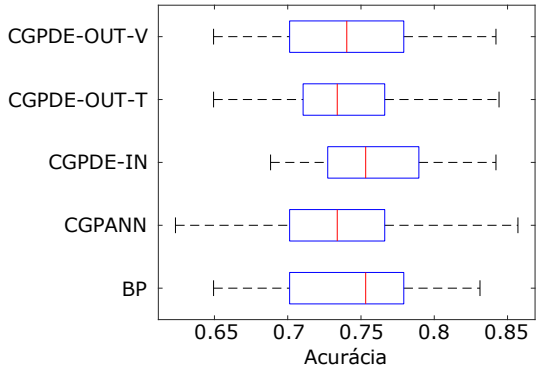
A Tabela 8 apresenta a média e o desvio padrão dos valores de acurácia dos conjuntos de teste, juntamente com análises estatísticas comparando o método de maior valor médio contra os demais métodos, para cada base. Como pode ser visto, em cinco das sete bases utilizadas, o método CGPDE-IN superou os demais métodos, indicando superioridade em relação aos métodos CGPANN (em todas as bases), CGPDE-OUT-T (nas bases *Diabetes* e *Wine*), e CGPDE-OUT-V (na base *Wine*), conforme representado pelos valores marcados com asterisco. Já para a base *Cancer*, o método CGPDE-OUT-V superou a acurácia média dos outros classificadores, sendo estatisticamente superior aos métodos BP e CGPANN. Para a base *Ionosphere*, o método CGPDE-OUT-T apresentou desempenho superior, com evidência estatística de que é melhor que o método CGPANN.

Tabela 8 – Acurácia dos conjuntos de teste: média e desvio padrão. Células hachuradas indicam o maior valor médio para cada base. Valores com asterisco indicam a rejeição da hipótese nula (H_0) a um nível de significância de 5% contra o método de maior valor médio, utilizando o teste-t pareado unilateral à direita.

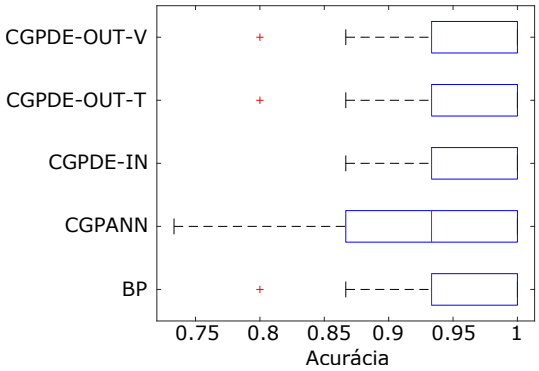
Base de dados	**BP**	**CGPANN**	**CGPDE IN**	**CGPDE OUT-T**	**CGPDE OUT-V**
Glass	0.5812 (0.1085)	0.5731* (0.0937)	0.6198 (0.0769)	0.5977 (0.0874)	0.6058 (0.0923)
Diabetes	0.7427 (0.0496)	0.7361* (0.0580)	0.7549 (0.0409)	0.7379* (0.0461)	0.7454 (0.0475)
Iris	0.9355 (0.0510)	0.9178* (0.0693)	0.9489 (0.0516)	0.9444 (0.0556)	0.9467 (0.0537)
Cancer	0.9532* (0.0241)	0.9546* (0.0263)	0.9609 (0.0257)	0.9644 (0.0240)	0.9648 (0.0246)
Ionosphere	0.8785 (0.0477)	0.8433* (0.0550)	0.8927 (0.0435)	0.8936 (0.0444)	0.8927 (0.0402)
Vertebral	0.7882 (0.0654)	0.7151* (0.0749)	0.8204 (0.0798)	0.8097 (0.0716)	0.8000 (0.0755)
Wine	0.9402 (0.0582)	0.8578* (0.1072)	0.9533 (0.0444)	0.9045* (0.0765)	0.9138* (0.0598)



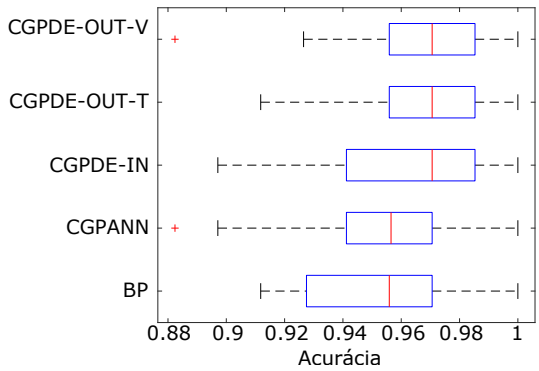
(a) Glass



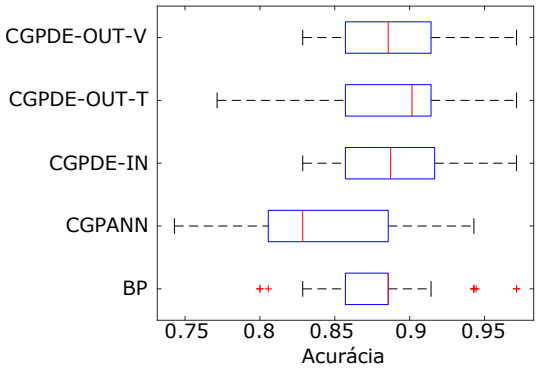
(b) Diabetes



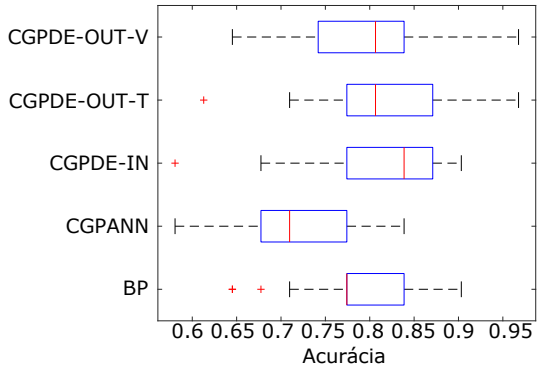
(c) Iris



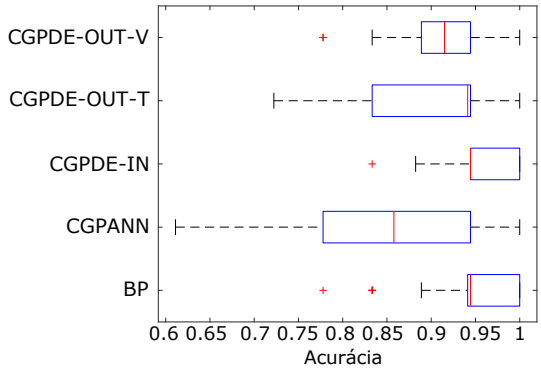
(d) Cancer



(e) Ionosphere



(f) Vertebral



(g) Wine

Figura 11 – Análise comparativa: diagramas de caixa referentes aos valores de acurácia dos 30 conjuntos de teste de cada algoritmo para as bases de dados da Tabela 5.

Tabela 9 – Comparação entre os métodos híbridos e o método CGPANN: p-valor obtido pelo teste-t pareado unilateral à direita dos modelos híbridos contra o método CGPANN. Células hachuradas indicam a rejeição da hipótese nula (H_0) a um nível de significância de 5%.

Base de dados	**CGPDE-IN**	**CGPDE-OUT-T**	**CGPDE-OUT-V**
Glass	0.0044	0.1020	0.0330
Diabetes	0.0478	0.4374	0.2298
Iris	0.0102	0.0314	0.0201
Cancer	0.0888	0.0310	0.0253
Ionosphere	<0.0001	<0.0001	<0.0001
Vertebral	<0.0001	<0.0001	<0.0001
Wine	<0.0001	0.0300	0.0095

Analisando especificamente os métodos híbridos em relação ao método CGPANN, a Tabela 9 apresenta o p-valor obtido pelo teste-t pareado unilateral à direita dos modelos híbridos contra o método CGPANN. Nota-se que os modelos rejeitaram a hipótese nula a um nível de significância de 5% para a grande maioria das bases, sugerindo que os modelos propostos são superiores ao modelo CGPANN.

Em resumo, pode-se concluir que os métodos neuroevolutivos híbridos propostos exibiram maior acurácia média em relação aos métodos BP e CGPANN para todas as bases de dados, além de fornecerem evidência de superioridade sobre o método CGPANN. Ao se comparar apenas as duas versões do método CGPDE-OUT, o uso dos dados de validação para escolha do melhor indivíduo (CGPDE-OUT-V) mostrou melhor desempenho do que usando dados de treinamento (CGPDE-OUT-T), como esperado. Finalmente, o algoritmo CGPDE-IN produziu melhores resultados em comparação às demais versões híbridas, apresentando maiores valores médios e melhores resultados estatísticos para a maioria das bases, sugerindo que a evolução intercalada da topologia e dos pesos utilizando CGP e DE gera redes com boa capacidade de generalização para problemas de classificação.

7.5.2 Análise para bases de dados reduzidas

Para verificação da robustez dos modelos híbridos, é realizada uma análise de desempenho dos modelos neuroevolutivos ao lidar com bases de dados com número reduzido de instâncias. Foram comparados os algoritmos CGPDE-IN, CGPDE-OUT (versões T e V), e CGPANN. Para cada uma das bases, foram escolhidas instâncias aleatórias até uma porcentagem determinada, mantendo a mesma proporção de classes da base original, e os algoritmos foram executados sobre essa porção reduzida de instâncias. Isso significa que todo o processo de validação cruzada foi realizado sobre essa porção reduzida de instâncias, e portanto o custo computacional para essas bases foi menor. As porcentagens aqui utilizadas são 25%, 50%, 75% e 100% das bases de dados originais, e os valores médios das acurácias dos 30 conjuntos de teste de cada algoritmo estão ilustrados

nos gráficos da Figura 12.

Na maioria dos cenários, a acurácia tende a diminuir quando o número de instâncias é reduzido. Esse fato é esperado, visto que uma pequena quantidade de observações durante o treinamento compromete a capacidade de generalização dos modelos, principalmente devido à falta de instâncias representativas da totalidade dos dados [9]. Os resultados mostram que o método CGPDE-IN apresentou a menor variação de desempenho, com altos valores médios de acurácia na maior parte dos problemas. Isso indica maior robustez em comparação aos demais métodos. As duas versões do método CGPDE-OUT apresentaram valores de acurácia média similares entre si ao variar a quantidade de instâncias utilizadas. Por outro lado, o método CGPANN apresentou as piores acurácias médias em todos os experimentos. Destaca-se, portanto, a superioridade dos modelos híbridos, sobretudo do método CGPDE-IN, em relação ao modelo CGPANN.

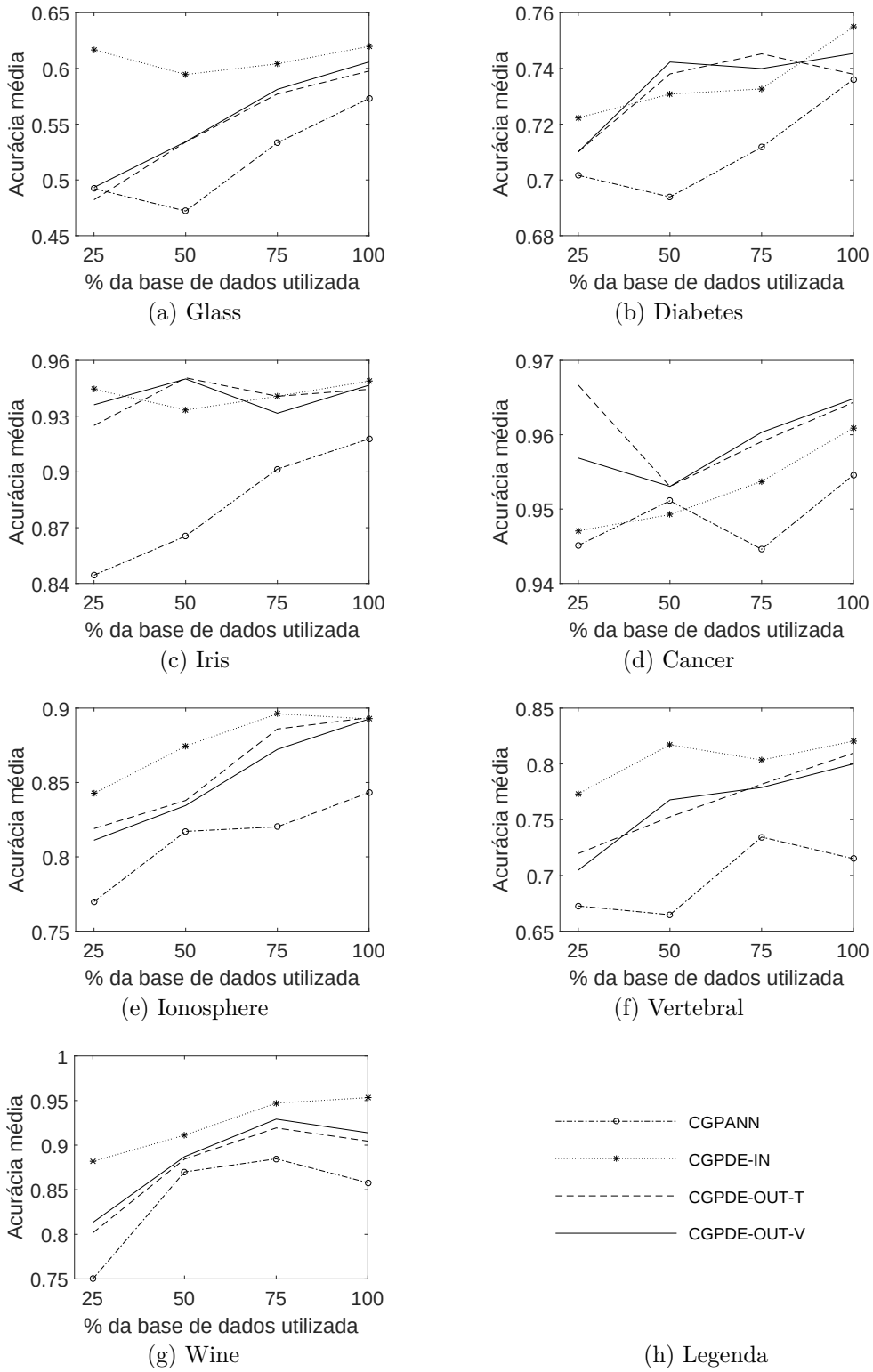
7.5.3 Análise estrutural das redes

Um estudo acerca da estrutura (topologia) das redes finais é realizado nesta seção. Entende-se por redes finais as redes geradas e selecionadas pelos modelos para serem avaliadas sobre os conjuntos de teste. As análises se concentram na forma como os nós estão organizados, referente ao número de nós ativos e à profundidade das redes. Os nós ativos são os nós que participam do cálculo de pelo menos uma saída da rede, e a profundidade de uma rede é entendida como o maior número de nós ativos conectados desde as entradas até as saídas.

Os métodos CGPDE-IN, CGPDE-OUT-V (aqui denominado CGPDE-OUT), e CGPANN foram comparados. O método CGPDE-OUT-T não foi incluído nessa análise, uma vez que o mesmo apresentou desempenho inferior à versão CGPDE-OUT-V nos experimentos da Seção 7.5.1. Para cada execução foram calculados: a acurácia do conjunto de teste, o número de nós ativos e a profundidade da rede final.

As Tabelas 10 e 11 ilustram, respectivamente, os valores médios dos nós ativos e os valores médios de profundidade das redes finais das 30 execuções, juntamente com os respectivos desvios-padrões. Observa-se que o método CGPANN apresentou menores valores médios de nós ativos e de profundidade para a maioria das bases. Além disso, em geral, os métodos apresentaram desvios-padrões relativamente elevados, tanto referente aos nós ativos quanto à profundidade. Tal fato sugere que as estruturas de rede geradas a cada execução foram bem distintas entre si, não se estabelecendo uma regularidade estrutural explícita. Essa irregularidade reforça o caráter estocástico dos modelos neuroevolutivos.

Um métrica interessante surge ao se dividir o valor médio de nós ativos pelo valor médio de profundidade das redes finais. Essa métrica pode ser interpretada como a quantidade média de nós ativos por camada, caso todas as camadas apresentassem a mesma quantidade de nós ativos. A Tabela 12 ilustra essa métrica. Nota-se a similaridade dos



(a) Glass (b) Diabetes (c) Iris (d) Cancer (e) Ionosphere (f) Vertebral (g) Wine (h) Legenda

Figura 12 – Análise para bases de dados reduzidas: acurácia média dos conjuntos de teste, utilizando 25%, 50%, 75%, e 100% de instâncias das bases de dados da Tabela 5.

valores obtidos pelos métodos, em que a maior variação foi a verificada na base *Vertebral*, cujos valores máximo e mínimo foram, respectivamente, 3.87 e 3.64. Essas similaridades indicam que, embora as redes finais geradas a cada execução sejam estruturalmente distintas entre si, os algoritmos preservam certa proporcionalidade entre os valores médios de nós ativos e de profundidade.

Tabela 10 – Quantidade de **nós ativos** das redes finais para as bases da Tabela 5: média e desvio padrão. Células hachuradas indicam o menor valor médio para cada base.

Base de dados	CGPANN	CGPDE-IN	CGPDE-OUT
Glass	375 (049)	370 (052)	372 (048)
Diabetes	269 (101)	306 (105)	303 (089)
Iris	316 (090)	328 (083)	321 (089)
Cancer	295 (096)	275 (112)	283 (094)
Ionosphere	252 (122)	300 (078)	261 (116)
Vertebral	309 (078)	323 (080)	320 (085)
Wine	315 (084)	302 (084)	272 (104)

Tabela 11 – **Profundidade** das redes finais para as bases da Tabela 5: média e desvio padrão. Células hachuradas indicam o menor valor médio para cada base.

Base de dados	CGPANN	CGPDE-IN	CGPDE-OUT
Glass	86.3 (06.3)	88.0 (06.9)	88.6 (07.3)
Diabetes	74.4 (14.8)	82.3 (17.1)	78.9 (11.0)
Iris	85.5 (12.6)	86.1 (09.9)	86.1 (12.4)
Cancer	79.5 (16.4)	74.2 (18.0)	78.1 (14.3)
Ionosphere	57.1 (20.3)	65.0 (09.5)	57.9 (18.6)
Vertebral	85.0 (09.8)	85.1 (11.1)	82.6 (11.4)
Wine	77.4 (12.8)	76.0 (11.4)	70.5 (16.8)

Tabela 12 – **Quantidade média de nós ativos por camada** (nós ativos/profundidade) das redes finais para as bases da Tabela 5.

Base de dados	CGPANN	CGPDE-IN	CGPDE-OUT
Glass	4.35	4.20	4.20
Diabetes	3.62	3.72	3.84
Iris	3.70	3.81	3.73
Cancer	3.71	3.71	3.62
Ionosphere	4.41	4.62	4.51
Vertebral	3.64	3.80	3.87
Wine	4.07	3.97	3.86

Para avaliar a influência da estrutura das redes no desempenho dos métodos, a Tabela 13 apresenta os coeficientes de correlação linear de Pearson [99] entre os valores de acurácia dos conjuntos de teste e a quantidade de nós ativos das redes finais.

Tabela 13 – Coeficiente de correlação linear de Pearson entre **acurácia** e quantidade de **nós ativos** das redes. Células hachuradas indicam os coeficientes com módulo maior que 0.3.

Base de dados	**CGPANN**	**CGPDE-IN**	**CGPDE-OUT**
Glass	-0.0817	0.0983	0.1157
Diabetes	-0.3060	0.2083	0.0011
Iris	0.2684	0.3004	0.0598
Cancer	0.1664	0.4475	0.0356
Ionosphere	0.0092	0.0940	0.1173
Vertebral	-0.0859	0.0984	-0.1417
Wine	-0.2965	-0.0532	-0.4239

Tabela 14 – Coeficiente de correlação linear de Pearson entre **acurácia** e **profundidade** das redes. Células hachuradas indicam os coeficientes com módulo maior que 0.3.

Base de dados	**CGPANN**	**CGPDE-IN**	**CGPDE-OUT**
Glass	0.1180	0.0124	0.1634
Diabetes	-0.2427	0.1878	0.0463
Iris	0.2633	0.2098	0.0531
Cancer	0.2025	0.4337	0.0897
Ionosphere	0.0389	0.0946	0.1827
Vertebral	-0.0653	0.2059	-0.1103
Wine	-0.1584	-0.2047	-0.3826

Analogamente, a Tabela 14 apresenta os coeficientes de correlação linear de Pearson entre os valores de acurácia dos conjuntos de teste e a profundidade das redes finais. Os coeficientes foram calculados utilizando o *Statistics and Machine Learning Toolbox* do MATLAB [110]. Os valores podem variar de –1 a +1. Um valor de –1 indica correlação negativa perfeita, enquanto um valor de +1 indica correlação positiva perfeita. Um valor de 0 indica que não há correlação entre os dados.

Em geral, percebe-se a baixa correlação obtida em todos os métodos analisados. Excepcionalmente, para a base *Cancer*, o método CGPDE-IN apresentou coeficientes de correlação positivos e moderados (em torno de +0.44), e para a base *Wine*, o método CGPDE-OUT apresentou coeficientes de correlação negativos e moderados (em torno de -0.40). Visto que a maioria dos coeficientes de correlação analisados estiveram dentro do intervalo $[-0.3, +0.3]$, compreende-se que a quantidade de nós ativos e a profundidade das redes, se analisadas isoladamente, contribuem de forma pouco significativa para o desempenho de classificação dos métodos. Tais resultados sugerem, por conseguinte, que os pesos e a estrutura das redes parecem exercer, de forma conjunta e interdependente, um papel preponderante na acurácia dos modelos.

7.6 ESTUDO DE DESEMPENHO DOS ALGORITMOS PROPOSTOS USANDO ERRO QUADRÁTICO MÉDIO

O erro quadrático médio (MSE) (Eq. 2.9) é comumente utilizado como função objetivo pelos algoritmos de classificação determinísticos, como o *backpropagation* [111]. Nesta seção, os modelos foram executados utilizando o MSE como função objetivo a ser minimizada. Após a execução dos algoritmos, os modelos foram avaliados sobre os conjuntos de teste utilizando a acurácia. Optou-se por utilizar a acurácia para avaliar os modelos visto que, em última análise, a classificação tem como objetivo final identificar a classe correta de cada instância, e não propriamente reduzir o erro quadrático médio. As bases de dados apresentadas na Tabela 5 foram utilizadas para os experimentos, sendo estes realizados conforme descrito na Seção 7.2. Os modelos neuroevolutivos considerados neste estudo adotam o Esquema de Classificação A (vide Subseção 4.3.2).

7.6.1 Análises comparativas

Na presente análise, foram comparados os algoritmos CGPDE-IN, CGPDE-OUT-V (aqui denominado CGPDE-OUT), CGPBP-IN, CGPBP-OUT, CGPANN, e BP. A Tabela 15 apresenta a média e o desvio padrão dos valores de acurácia dos conjuntos de teste, juntamente com análises estatísticas comparando o método de maior valor médio contra os demais métodos, para cada base. Como indica a Tabela 15, cada modelo híbrido apresentou a maior acurácia média em pelo menos uma base. O modelo CGPDE-IN obteve melhor desempenho nas bases *Diabetes* e *Vertebral*. O modelo CGPDE-OUT apresentou melhor desempenho na base *Glass*. O modelo CGPBP-IN obteve melhor acurácia média nas bases *Cancer* e *Ionosphere*. Por fim, o modelo CGPBP-OUT apresentou melhor acurácia média nas bases *Iris* e *Wine*.

De acordo com as indicações em asterisco na tabela, o método BP foi estatisticamente inferior aos modelos de melhor desempenho em todas as bases, se comparado aos modelos de maior acurácia média. O modelo CGPANN foi estatisticamente inferior aos modelos de maior acurácia média em todas as bases, exceto *Diabetes* e *Cancer*. Os modelos CGPDE-IN e CGPDE-OUT apresentaram os melhores desempenhos, pois foram estatisticamente inferiores aos modelos de maior acurácia média em apenas duas bases. Já os modelos CGPBP-IN e CGPBP-OUT foram estatisticamente inferiores aos modelos de maior acurácia média em quatro e três bases, respectivamente.

Percebe-se que os modelos CGPDE são mais estáveis que os modelos CGPBP. Isso significa que mesmo quando os modelos CGPDE são estatisticamente inferiores, eles não apresentam tanta discrepância de desempenho em relação aos modelos de maior acurácia média. Por exemplo, na base *Cancer*, o modelo CGPDE-IN foi estatisticamente inferior ao modelo CGPBP-IN. Contudo, as acurácias médias de ambos modelos foram 0.9619 (CGPDE-IN) e 0.9732 (CGPBP-IN), ou seja, uma diferença de 0.0113 entre os valores.

Tabela 15 – Acurácia dos conjuntos de teste: média e desvio padrão. Células hachuradas indicam o maior valor médio para cada base. Valores com asterisco indicam a rejeição da hipótese nula (H_0) a um nível de significância de 5% contra o método de maior valor médio, utilizando o teste-t pareado unilateral à direita.

Base de dados	**BP**	**CGPANN**	**CGPDE IN**	**CGPDE OUT**	**CGPBP IN**	**CGPBP OUT**
Glass	0.2953* (0.1385)	0.4984* (0.0876)	0.5531 (0.0951)	0.5633 (0.0800)	0.3658* (0.1079)	0.3304* (0.0156)
Diabetes	0.6290* (0.1400)	0.7347 (0.0627)	0.74693 (0.0525)	0.74688 (0.0666)	0.7383 (0.0543)	0.7457 (0.0409)
Iris	0.5156* (0.2426)	0.9156* (0.0552)	0.9311 (0.0593)	0.9444 (0.0528)	0.8511* (0.1461)	0.9533 (0.0468)
Cancer	0.8734* (0.1814)	0.9638 (0.0261)	0.9619* (0.0213)	0.9658 (0.0229)	0.9732 (0.0144)	0.9663* (0.0164)
Ionosphere	0.7076* (0.1871)	0.8642* (0.0680)	0.8927 (0.0573)	0.8823* (0.0493)	0.9117 (0.0521)	0.9098 (0.0436)
Vertebral	0.5247* (0.1669)	0.7215* (0.0930)	0.8183 (0.0539)	0.7860* (0.0731)	0.6667* (0.1231)	0.7817* (0.0627)
Wine	0.5797* 0.2189	0.8765* (0.1107)	0.9254* (0.0627)	0.9398 (0.0560)	0.8925* (0.1348)	0.9494 (0.0432)

Por outro lado, os modelos CGPBP são mais instáveis. Isso significa que existem bases nas quais os modelos CGPBP apresentam maiores valores médios de acurácia. Entretanto, ao analisar as bases nas quais os modelos CGPBP são estatisticamente inferiores, a discrepância em relação aos modelos de maior acurácia média é significativa. Por exemplo, nas bases *Glass* e *Vertebral*, o modelo CGPBP-IN apresentou valores bem abaixo dos valores observados pelos modelos de maior acurácia média. Na base *Glass*, as acurácias médias dos modelos foram 0.5633 (CGPDE-OUT) e 0.3658 (CGPBP-IN), ou seja, uma diferença de 0.1975 entre os valores. Na base *Vertebral*, as acurácias médias dos modelos foram 0.8183 (CGPDE-IN) e 0.6667 (CGPBP-IN), ou seja, uma diferença de 0.1516 entre os valores. Portanto, percebe-se que essas diferenças são cerca de uma ordem de grandeza maior que a diferença observada no caso anterior, reforçando a instabilidade dos modelos CGPBP em relação aos modelos CGPDE.

Analisando especificamente os modelos híbridos em relação ao modelo CGPANN, a Tabela 16 apresenta o p-valor obtido pelo teste-t pareado unilateral à direita dos modelos híbridos contra o modelo CGPANN. Nota-se que os modelos CGPDE-IN, CGPDE-OUT, e CGPBP-OUT rejeitaram a hipótese nula a um nível de significância de 5% em quatro das sete bases. Dentre esses três modelos, o CGPDE-OUT apresentou os menores p-valores, o que indica uma superioridade estatística mais evidente desse modelo em relação ao

Tabela 16 – Comparação entre os métodos híbridos e o método CGPANN: p-valor obtido pelo teste-t pareado unilateral à direita dos modelos híbridos contra o método CGPANN. Células hachuradas indicam a rejeição da hipótese nula (H_0) a um nível de significância de 5%.

Base de dados	**CGPDE IN**	**CGPDE OUT**	**CGPBP IN**	**CGPBP OUT**
Glass	0.0033	0.0004	1	1
Diabetes	0.1419	0.0604	0.4042	0.1863
Iris	0.1466	0.0227	0.9759	0.0022
Cancer	0.6636	0.3150	0.0591	0.3392
Ionosphere	0.0296	0.0993	0.0037	0.0054
Vertebral	<0.0001	0.0006	0.9669	0.0018
Wine	0.0339	0.0050	0.3272	0.0002

modelo CGPANN. O modelo CGPBP-OUT, embora tenha sido estatisticamente superior ao modelo CGPANN em quatro bases, apresentou desempenho insatisfatório na base *Glass*. Por fim, o modelo CGPBP-IN apresentou o pior desempenho dentre os modelos híbridos, rejeitando a hipótese nula em apenas uma base.

Em resumo, pode-se concluir que os modelos CGPDE exibiram melhor desempenho geral que os modelos CGPBP. Algumas hipóteses podem explicar a causa do desempenho inferior observado nos modelos CGPBP. A primeira hipótese é a estagnação em ótimos locais, visto que esses modelos ajustam os pesos utilizando o *backpropagation*, que é uma técnica de otimização determinística. Já os modelos que ajustam os pesos utilizando o DE são menos suscetíveis a estagnações devido à sua estocasticidade. Além disso, em redes com quantidade suficientemente grande de camadas, a técnica *backpropagation* sofre o efeito denominado *vanishing/exploding gradient problem*, em que o gradiente de erro se torna muito menor ou muito maior em magnitude à medida que os erros são propagados ao longo das camadas [54]. Como o ajuste dos pesos depende do cálculo dos erros, esse efeito interfere negativamente na qualidade dos valores de peso. Em contrapartida, os modelos CGPDE não calculam gradientes e, portanto, não sofrem desse efeito.

Ao se comparar isoladamente os modelos CGPDE, nota-se que ambos obtiveram desempenhos similares, com uma pequena vantagem do modelo CGPDE-OUT. Por outro lado, ao se comparar isoladamente os métodos CGPBP, verifica-se uma superioridade estatística do método CGPBP-OUT em relação ao método CGPBP-IN.

O modelo CGPBP-IN apresentou o pior desempenho dentre todos os modelos híbridos propostos. A estagnação em ótimos locais e a constante mudança na topologia das redes parecem exercer, de forma conjunta, uma degradação no desempenho desse modelo. A cada geração, para uma determinada topologia, o *backpropagation* otimiza o conjunto de pesos da rede, levando-os próximo a um mínimo local. No entanto, quando essa topologia é alterada, todo esforço feito para otimização dos pesos é de certa forma perdido, pois a

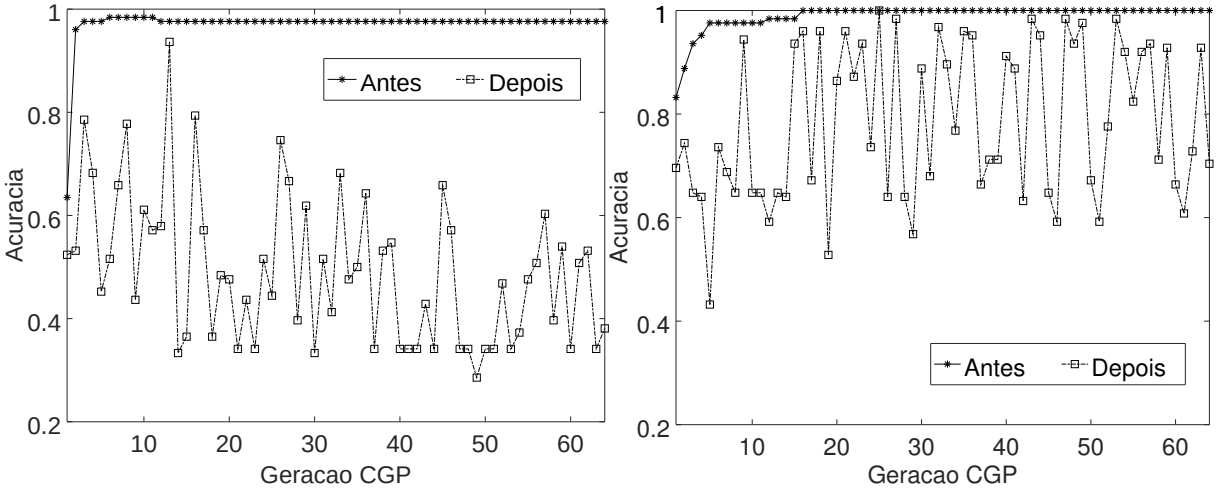
nova topologia exige um novo conjunto de pesos ótimos, que muito provavelmente não estão dentro da mesma bacia de atração de mínimos locais anteriormente visitada.

Em teoria, esse fenômeno não ocorre no modelo CGPDE-IN, visto que a busca feita pelo DE sobre os pesos da rede é mais abrangente, e portanto não fica estagnada em ótimos locais nem dependente de uma topologia específica de rede. Dessa forma, quando a topologia é alterada, a otimização dos pesos realizada pelo DE não é totalmente perdida, visto que o conjunto de pesos da rede não se estagnou em bacias de atração de mínimos locais, devido ao caráter global do mecanismo de busca. Portanto, o conjunto de pesos encontrados pelo DE pode se adequar a novas topologias de rede posteriormente geradas.

Uma vez levantadas essas hipóteses, investigações empíricas mais detalhadas foram realizadas para se explicar a diferença de desempenho observada pelos modelos CGPDE-IN e CGPBP-IN. Para isso, um experimento foi realizado para acompanhar, ao longo das gerações, o desempenho das redes imediatamente antes e imediatamente após suas topologias serem alteradas. Isso permite verificar se os pesos obtidos pelo modelo CGPDE-IN permanecem com bom desempenho após a alteração da topologia e se os pesos obtidos pelo modelo CGPBP-IN apresentam degradação de desempenho após a alteração da topologia.

Este experimento foi realizado como segue. A cada geração, imediatamente antes de se evoluir as topologias, os modelos apresentam 1 indivíduo-pai e λ indivíduos-filhos, idênticos ao indivíduo-pai. No caso do modelo CGPDE-IN, esse ponto pode ser localizado na linha 15 do Algoritmo 5. No caso do modelo CGPBP-IN, esse ponto refere-se à linha 14 do Algoritmo 7. Nesse passo, o indivíduo-pai de cada modelo é avaliado sobre o conjunto de treinamento utilizando a acurácia. Em seguida, o fluxo de execução dos algoritmos continua, e os λ indivíduos-filhos de cada modelo sofrem mutações em suas topologias. Imediatamente após as topologias dos λ indivíduos-filhos serem alteradas, um indivíduo-filho aleatório é escolhido para ser avaliado sobre o conjunto de treinamento utilizando a acurácia. Dessa forma, obtém-se o desempenho anterior e o desempenho posterior à alteração da topologia das redes. Esse processo é realizado ao longo de todas as gerações, em uma única execução do algoritmo. Embora a função objetivo utilizada neste experimento seja o MSE, optou-se por avaliar os indivíduos utilizando a acurácia por ser uma métrica mais realista do ponto de vista prático de classificação.

A Figura 13 ilustra a evolução da acurácia das redes, imediatamente antes e imediatamente após suas topologias serem alteradas.



(a) CGPBP-IN (b) CGPDE-IN

Figura 13 – Análise comparativa entre CGPBP-IN e CGPDE-IN. Evolução da acurácia das redes ao longo das gerações. A cada geração, uma rede aleatória foi escolhida e avaliada imediatamente antes e imediatamente após sua topologia ser alterada, mantendo os pesos fixos.

A Figura 13a refere-se ao modelo CGPBP-IN, e a Figura 13b refere-se ao modelo CGPDE-IN. A base *Wine* foi utilizada para esse experimento. Com base na figura pode-se notar que, imediatamente antes da evolução da topologia, as redes de ambos os modelos apresentaram bom desempenho sobre o conjunto de treinamento. No entanto, após a topologia dessas redes serem alteradas, observa-se que as redes do modelo CGPDE-IN exibiram um desempenho superior em relação ao desempenho das redes do modelo CGPBP-IN. A fim de analisar numericamente esses resultados, a Tabela 17 descreve o valor médio das acurácias indicadas pela Figura 13.

Tabela 17 – Média dos valores de acurácia obtidos ao longo das gerações. A cada geração, uma rede aleatória foi escolhida e avaliada imediatamente antes e imediatamente após sua topologia ser alterada, mantendo os pesos fixos.

	Acurácia Média das Redes	
	ANTES da evolução da topologia	DEPOIS da evolução da topologia
CGPBP-IN	0.9714	0.4911
CGPDE-IN	0.9903	0.7845

Por meio dessa tabela, nota-se que antes da evolução da topologia, ambos os modelos exibiram desempenho médio similar (0.9714 e 0.9903). Entretanto, após as topologias serem alteradas, as redes do modelo CGPDE-IN foram capazes de alcançar desempenho médio de 0.7845, enquanto as redes do modelo CGPBP-IN apresentaram um

desempenho médio de 0.4911. Percebe-se uma diferença significativa em relação a esses desempenhos.

De posse dos resultados da Figura 13 e da Tabela 17, as duas hipóteses levantadas anteriormente foram reforçadas: (i) devido ao caráter global do mecanismo de busca do DE, o conjunto de pesos encontrados no modelo CGPDE se adapta mais facilmente a novas topologias de rede posteriormente geradas; e (ii) devido ao caráter local do mecanismo de busca do BP, o conjunto de pesos encontrados no modelo CGPBP é adequado apenas para a topologia específica na qual a rede foi otimizada, encontrando dificuldades de se obter bom desempenho quando novas topologias de rede são posteriormente geradas.

Essa degradação de desempenho observada no modelo CGPBP-IN é ocasionada justamente devido aos ajustes periódicos e intercalados entre topologias e pesos. Portanto, conclui-se que o modelo CGPBP-OUT não sofre dessa degradação, pois esse modelo executa o *backpropagation* apenas após a evolução completa da topologia. Dessa forma, todo o esforço realizado para a otimização dos pesos da rede não é perdido, visto que a topologia não é posteriormente alterada. Esse fenômeno parece explicar a superioridade estatística do modelo CGPBP-OUT em relação ao modelo CGPBP-IN, cujos resultados foram apresentados nas Tabelas 15 e 16.

Por outro lado, existe uma hipótese que contra-argumenta a análise feita acerca deste último experimento. Essa hipótese alternativa enuncia que o DE, justamente devido ao seu mecanismo global de busca, consegue alcançar valores de peso mais próximos aos ótimos globais para a topologia atual. Dessa forma, esses pesos seriam mais especializados para a topologia sobre a qual o DE foi executado, o que poderia degradar o desempenho da rede após alterações em sua topologia. No entanto, o que se observou no experimento realizado foi uma melhoria de desempenho ao utilizar o DE durante a otimização dos pesos, o que vai na contramão do que foi enunciado por essa hipótese alternativa. De fato, a análise aqui realizada se baseou em apenas uma execução de uma única base de dados e, portanto, novos experimentos são necessários para uma verificação mais ampla e assertiva acerca desta questão.

7.7 ESTUDO DE MÉTRICAS EM CENÁRIOS DESBALANCEADOS

Esta seção se concentra no estudo de desempenho das métricas de classificação e ranqueamento. Tais métricas são utilizadas como funções objetivo pelos modelos neuroevolutivos, sendo aplicadas a problemas de classificação de dados desbalanceados. Vale destacar que os modelos híbridos CGPBP não foram analisados nesta seção, visto que exigem a diferenciabilidade da função objetivo [36] e portanto não são capazes de otimizar diretamente as métricas aqui utilizadas.

7.7.1 Motivação do estudo

Como visto no Capítulo 2, a classificação de dados desbalanceados é um problema largamente abordado na área de aprendizado de máquina, dada a complexidade de se obter modelos capazes de prover bom desempenho de classificação em bases dessa natureza. Diversas métricas têm sido utilizadas para avaliar a qualidade dos classificadores em cenários desbalanceados [25], como G-mean (Eq. 2.7), F_β-score (Eq. 2.8), e AUC (Eq. 2.11). Dado que o processo de busca dos algoritmos de classificação é dependente da função objetivo a ser otimizada, o desenvolvimento de métodos que permitem a otimização de tais métricas é relevante. As técnicas de computação evolutiva são capazes de lidar com funções objetivo não diferenciáveis [12], de modo que a formulação exata de tais métricas pode ser utilizada como função a ser otimizada pelos métodos evolutivos.

Ademais, existem aplicações nas quais o usuário frequentemente não tem uma noção clara de qual função objetivo adotar durante o processo de otimização, visto que o problema em questão não define nenhum critério de desempenho específico. Em cenários desbalanceados, por exemplo, uma escolha incorreta da função objetivo pode comprometer a classificação, visto que o desempenho das métricas é sensível à proporção de classes presentes durante a etapa de treinamento [112]. Portanto, torna-se desejável que a escolha da função objetivo forneça bom desempenho geral para a maioria das métricas existentes. Em outras palavras, espera-se que o classificador produzido durante a otimização de uma determinada métrica apresente bons resultados ao ser avaliado não só sob a métrica que foi otimizada, mas também sob outras métricas distintas.

A principal motivação deste estudo é a ausência de trabalhos na literatura que estudem o impacto de se utilizar uma determinada função objetivo quando a aplicação não exige um critério de desempenho específico. Nessa aplicações, o processo de otimização deve idealmente fornecer desempenho razoável em todas as métricas existentes. Jeni et al. [113] analisou o efeito do desbalanceamento sobre o desempenho de diferentes métricas, utilizando máquinas de vetores de suporte para detecção de unidades de ação facial. No entanto, o estudo não analisou o uso de tais métricas como funções objetivo, nem a influência de tais funções ao avaliarem outras métricas. Suprir tais carências da literatura é justamente o objetivo desta seção, cujos experimentos são descritos a seguir.

7.7.2 Descrição dos experimentos

Os experimentos utilizam os modelos neuroevolutivos CGPANN, CGPDE-IN, e CGPDE-OUT-V (aqui denominado CGPDE-OUT). Esses modelos realizam a otimização de diferentes funções objetivo e são aplicados a problemas de classificação de dados desbalanceados. Os indivíduos finais obtidos pelo processo evolutivo são avaliados sobre os conjuntos de teste por meio do cálculo de diversas métricas. A Tabela 6 descreve as 11 bases de dados utilizadas nos experimentos, sendo esses realizados conforme descrito na

Seção 7.2.

Os algoritmos são executados utilizando quatro funções objetivo distintas: (i) acurácia, (ii) G-mean, (iii) F_β-score, e (iv) área abaixo da curva ROC (AUC). Ao final da etapa de treinamento, cada algoritmo retorna seu melhor indivíduo, que é avaliado sobre os conjuntos de teste utilizando as quatro métricas acima. Por exemplo, o algoritmo CGPDE-IN realiza o processo evolutivo utilizando a AUC como função objetivo, retorna seu melhor indivíduo, e esse indivíduo é avaliado sobre os conjuntos de teste por meio do cálculo da acurácia, G-mean, F_β-score, e AUC. Esse processo é realizado para investigar como um indivíduo, otimizado a partir de uma função objetivo específica, se comporta ao ser avaliado por métricas que não foram consideradas em nenhuma etapa do processo evolutivo. Portanto, existem 12 métodos distintos a serem analisados: 3 algoritmos utilizando 4 métricas como função objetivo. Ademais, existem 44 problemas distintos a serem resolvidos: 11 bases de dados cujos conjuntos de teste são utilizados para avaliação de cada método por meio de 4 métricas.

Os modelos neuroevolutivos considerados neste estudo adotam o Esquema de Classificação B (vide Subseção 4.3.2). Nesse esquema, o valor do nó de saída pode assumir valores dentro do intervalo $[0, 1]$. Para rotular uma dada instância, um limiar de decisão $\theta = 0.5$ é aplicado nos métodos cuja função objetivo é acurácia, G-mean, ou F_β-score.

Entretanto, os métodos que otimizam a AUC não estabelecem um limiar de decisão, uma vez que o objetivo da AUC é ranquear corretamente as instâncias, não havendo necessidade de rotulá-las como positivas ou negativas (vide Eq. 2.11). Portanto, para calcular a acurácia, G-mean, e F_β-score dos conjuntos de teste utilizando os métodos que otimizam a AUC, um limiar de decisão deve ser estabelecido. Para isso, foi traçada a curva ROC considerando apenas o conjunto de treinamento, e o valor-limiar que mais se aproxima do ponto $(0, 1)$ da curva ROC foi definido como limiar de decisão. Essa escolha foi feita pois o valor-limiar desse ponto tende a apresentar bom desempenho de classificação para ambas as classes [28].

Os 12 métodos são avaliados sobre os conjuntos de teste das 11 bases de dados, por meio das métricas acurácia, G-mean, F_β-score e AUC, analisando cada métrica separadamente. Devido ao grande número de resultados a serem analisados, dois índices de avaliação foram propostos para verificar o comportamento geral dos métodos:

- **Á**rea **N**ormalizada **A**baixo do **P**erfil **D**esempenho (ANAPD): a área abaixo do perfil de desempenho (vide Seção 7.3) foi calculada para cada um dos 12 métodos ($n_s = 12$), considerando os 11 problemas relativos ao cálculo de cada métrica ($n_p = 11$). As doze áreas obtidas foram normalizadas em relação ao método de maior área.

- **P**roporção das **V**ezes em que um dado método obteve **M**aior média (PVM): como já visto, cada método é avaliado sobre os conjuntos de teste de uma base utilizando

uma dada métrica, e o valor médio desta métrica, referente às 30 execuções, é obtido. Esse cálculo do valor médio é feito por todos os 12 métodos para as 11 bases, utilizando a mesma métrica para avaliar os conjuntos de teste das 11 bases. Os valores médios obtidos pelos métodos são comparados entre si, em relação a mesma base, e a proporção de vezes que um dado método obteve média maior que os demais métodos, com p-valor < 0.05, é indicada pelo índice PVM. O p-valor foi calculado pelo teste-t pareado unilateral à direita [99] utilizando o *Statistics and Machine Learning Toolbox* do MATLAB.

Para exemplificar o cálculo do PVM, o presente exemplo considera que os conjuntos de teste das 11 bases de dados foram avaliados utilizando a métrica G-mean. Assim, o método CGPDE-IN, utilizando AUC como função objetivo, é avaliado sobre os conjuntos de teste da base *Wine* (utilizando a G-mean), e o valor médio das 30 execuções é obtido. Esse cálculo das médias é feito por todos os 12 métodos para as 11 bases. Neste exemplo fictício, o método CGPDE-IN utilizando AUC como função objetivo, ao ter seus valores médios comparados com os demais métodos, em relação à mesma base, obteve maior média com p-valor < 0.05 em 38 comparações. Dado que cada método compara seus valores médios com os valores médios dos outros 11 métodos, para cada uma das 11 bases, então são realizadas 121 comparações. Portanto, o índice PVM do método aqui exemplificado é $(38/121) \times 100 \approx 31\%$.

7.7.3 Resultados

A Tabela 18 descreve os índices ANAPD e PVM de cada método ao ser avaliado sobre os conjuntos de teste das 11 bases de dados, por meio das métricas acurácia, G-mean, F_β-score e AUC, analisando cada métrica separadamente. De acordo com essa tabela, os métodos que utilizaram a mesma métrica tanto para otimização da função objetivo quanto para avaliação sobre os conjuntos de teste tenderam a apresentar melhores resultados. Tal fato é indicado pela presença de células hachuradas na diagonal principal da tabela, que destacam os maiores valores de ANAPD e PVM de cada coluna. Esse comportamento é esperado, já que esses métodos foram projetados justamente para a otimização da métrica pela qual estão sendo avaliados.

Ao utilizar a acurácia como função objetivo, os métodos exibiram alta acurácia sobre os conjuntos de teste, contudo valores menores para as demais métricas. Isso indica que tais métodos tendem a rotular as instâncias como pertencentes à classe negativa, visto que em bases desbalanceadas essa classe contribui majoritariamente para uma alta acurácia [9], afetando o desempenho das demais métricas ao rotular incorretamente as instâncias verdadeiramente positivas.

Com relação aos métodos que utilizaram a G-mean como função objetivo, o método CGPDE-IN apresentou comportamento superior ao ser avaliado pelas métricas

Tabela 18 – ANAPD e PVM dos 12 métodos neuroevolutivos (linhas) ao serem avaliados sobre os conjuntos de teste das 11 bases de dados da Tabela 6, por meio das métricas Acurácia, G-mean, F_β-score, e AUC (colunas). Células hachuradas indicam os maiores valores de ANAPD e PVM de cada coluna.

		ANAPD (PVM)			
Função Objetivo	Método	Acurácia	G-mean	F_β-score	AUC
Acurácia	CGPANN	0.9893 (36%)	0.9159 (07%)	0.9283 (05%)	0.9746 (26%)
	CGPDE-IN	1 (56%)	0.9387 (13%)	0.9685 (19%)	0.8212 (01%)
	CGPDE-OUT	0.9984 (64%)	0.9496 (11%)	0.9658 (22%)	0.8707 (02%)
G-mean	CGPANN	0.9283 (18%)	0.9566 (26%)	0.9443 (15%)	0.9804 (46%)
	CGPDE-IN	0.9159 (21%)	1 (45%)	1 (35%)	0.9707 (19%)
	CGPDE-OUT	0.9243 (21%)	0.9675 (40%)	0.9816 (27%)	0.8780 (15%)
F_β-score	CGPANN	0.9843 (26%)	0.8540 (09%)	0.8911 (12%)	0.9770 (35%)
	CGPDE-IN	0.9950 (44%)	0.9792 (29%)	0.9975 (42%)	0.9253 (12%)
	CGPDE-OUT	0.9889 (45%)	0.9762 (27%)	0.9982 (33%)	0.9202 (07%)
AUC	CGPANN	0.8514 (07%)	0.9845 (41%)	0.9268 (14%)	0.9649 (33%)
	CGPDE-IN	0.7313 (04%)	0.8171 (31%)	0.8427 (18%)	0.9952 (31%)
	CGPDE-OUT	0.8040 (07%)	0.8925 (36%)	0.9089 (19%)	1 (44%)

G-mean e F_β-score sobre os conjuntos de teste, obtendo um desempenho equilibrado de classificação para ambas as classes. O método CGPANN obteve melhor desempenho ao ser avaliado pela AUC. Entretanto, todos os métodos que otimizaram a G-mean apresentaram baixo desempenho ao serem avaliados pela acurácia, sugerindo que tais métodos, embora classifiquem corretamente a maioria das instâncias positivas, tendem a classificar incorretamente uma porção não desprezível de instâncias negativas. Como as instâncias negativas exercem um papel preponderante para uma alta acurácia, rotular incorretamente tais instâncias afeta sensivelmente o valor dessa métrica.

A utilização da métrica F_β-score como função objetivo produziu modelos que apresentaram altos valores de F_β-score ao serem avaliados sobre os conjuntos de teste, principalmente o método CGPDE-IN, que obteve melhor desempenho geral.

Finalmente, ao utilizar a AUC como função objetivo, o método CGPDE-OUT apresentou alta AUC ao ser avaliado sobre os conjuntos de teste. Entretanto, os métodos que otimizaram a AUC obtiveram baixos valores de acurácia. Isso indica que tais métodos apresentaram boa capacidade de ranqueamento, i.e., são capazes de gerar valores de saída maiores para as instâncias positivas, e valores menores para as instâncias negativas. Entretanto, ao estabelecer um limiar de decisão para o cálculo das demais métricas, boa parte das instâncias negativas são rotuladas incorretamente, o que contribui para um baixo valor de acurácia, e valores relativamente baixos de G-mean e F_β-score.

Nota-se que, de forma geral, o método CGPDE-IN exibiu bom desempenho para a maioria das métricas, apresentando resultados superiores quando comparado aos demais

métodos. Para investigar o comportamento das soluções considerando exclusivamente o método CGPDE-IN, a Tabela 19 ilustra os índices ANAPD e PVM do método CGPDE-IN utilizando as 4 funções objetivo, portanto resultando em 4 métodos ($n_s = 4$). Tais métodos são avaliados sobre os conjuntos de teste das 11 bases de dados utilizando as 4 métricas, resultando em 44 problemas ($n_p = 44$). Nesse caso, cada método realiza 132 comparações para o cálculo do índice PVM, pois compara seus valores médios com os outros 3 métodos para cada um dos 44 problemas em questão.

Tabela 19 – ANAPD e PVM do método CGPDE-IN ao ser avaliado sobre os conjuntos de teste das 11 bases de dados pelas 4 métricas, utilizando diferentes funções objetivo. Células hachuradas indicam os maiores valores de ANAPD e PVM.

	Função objetivo			
	Acurácia	G-Mean	F_β-score	AUC
ANAPD	0.9669	1	0.9945	0.9263
PVM	18%	30%	33%	23%

De acordo com a Tabela 19, a utilização da G-mean como função objetivo resultou em um melhor índice ANAPD, e a otimização de F_β-score produziu melhor índice PVM. Tais resultados indicam que, quando a G-mean é utilizada como objetivo, o método CGPDE-IN apresenta altos valores de mediana ao ser avaliado sobre os conjuntos de teste pelas quatro métricas. Por outro lado, a otimização de F_β-score gerou valores superiores de média com p-valor < 0.05. Ambas funções objetivo apresentaram desempenho superior em relação ao desempenho obtido ao otimizar as funções acurácia e AUC.

Vale destacar o desempenho relativamente baixo apresentado pelos métodos ao utilizarem a AUC como função objetivo. Em teoria, a métrica AUC é mais apropriada para lidar com problemas desbalanceados, enquanto a acurácia normalmente tende a enviesar o classificador em direção à classe majoritária [9]. Uma análise conjunta das Tabelas 18 e 19 mostra que os métodos que utilizaram a AUC como função objetivo obtiveram bom desempenho ao serem avaliados pela AUC sobre os conjuntos de teste. Entretanto, eles não obtiveram bons resultados ao serem avaliados pelas demais métricas. Isso sugere que a utilização da AUC como função objetivo produz modelos com alta capacidade de ranqueamento e baixa capacidade de classificação. Dessa forma, uma melhoria de desempenho pode ser obtida com uma escolha mais adequada do limiar de decisão, que exerce papel crucial para o desempenho das métricas de classificação.

7.8 CONCLUSÕES DO CAPÍTULO

Neste capítulo, uma investigação empírica foi feita para avaliar comparativamente o comportamento dos modelos híbridos propostos quando aplicados a problemas de classificação de dados. O primeiro estudo utilizou a acurácia como objetivo. Nesse estudo, foram realizadas três análises distintas: (i) análise comparativa dos métodos, (ii) análise

para bases de dados reduzidas, e (iii) análise estrutural das redes produzidas. Os seguintes resultados foram obtidos:

- Os modelos híbridos exibiram maiores valores médios de acurácia em relação aos valores médios obtidos pelos modelos BP e CGPANN.
- O modelo híbrido CGPDE-IN produziu melhores valores médios de acurácia em comparação aos valores médios obtidos pelas demais versões híbridas.
- Ao serem executados sobre bases de dados reduzidas, os modelos híbridos apresentaram maior robustez em relação ao modelo CGPANN.
- A análise estrutural das redes indica que os algoritmos preservam certa proporcionalidade entre os valores médios de nós ativos e de profundidade.
- A quantidade de nós ativos e a profundidade das redes, se analisados isoladamente, contribuem de forma pouco significativa para o desempenho dos métodos.

O segundo estudo utilizou o erro quadrático médio como objetivo. Esse foi o único estudo em que todos os modelos híbridos propostos foram comparados em conjunto. Os seguintes resultados foram obtidos:

- Os modelos híbridos CGPDE exibiram melhor desempenho geral que os modelos híbridos CGPBP.
- Os modelos CGPDE-IN, CGPDE-OUT, e CGPBP-OUT foram estatisticamente superiores ao modelo CGPANN em quatro das sete bases comparadas, com destaque para o modelo CGPDE-OUT.
- Dentre todos os modelos comparados, o BP apresentou o pior desempenho geral.
- Dentre todos os modelos neuroevolutivos comparados, o modelo CGPANN apresentou o pior desempenho geral.
- Dentre os modelos neuroevolutivos híbridos comparados, o modelo CGPBP-IN exibiu o pior desempenho geral
- A estagnação em ótimos locais e a constante mudança na topologia das redes parecem exercer, de forma conjunta, uma degradação no desempenho do modelo CGPBP-IN.

Por fim, o terceiro estudo avaliou o desempenho das métricas acurácia, G-mean, F_β-score, e AUC quando utilizadas como funções objetivo pelos modelos neuroevolutivos. Esse estudo foi aplicado a problemas de classificação de dados desbalanceados. Os seguintes resultados foram obtidos:

- A utilização das métricas G-mean e F_β-score como funções objetivo produzem modelos com capacidade de classificação satisfatória para as instâncias de ambas as classes.
- A utilização da acurácia como função objetivo produz modelos capazes de classificar corretamente a maioria das instâncias negativas (majoritárias), contudo mostraram-se incapazes de detectar corretamente grande parte das instâncias positivas (minoritárias).
- A utilização da AUC como função objetivo produz modelos com boa capacidade de ranqueamento e baixa capacidade de classificação.
- Para melhorar o desempenho de classificação dos modelos que otimizam a AUC, sugere-se definir novas estratégias para estabelecimento do valor de limiar de decisão.
- O modelo híbrido CGPDE-IN exibiu desempenho geral superior quando comparado aos demais modelos, principalmente quando as métricas acurácia, G-mean, e F_β-score são utilizadas como objetivo.

8 CONCLUSÕES E TRABALHOS FUTUROS

A presente dissertação propôs modelos neuroevolutivos híbridos para a construção de redes neurais artificiais (RNAs) via programação genética cartesiana (CGP). A formulação desenvolvida consiste em utilizar técnicas de otimização adequadas para cada tipo de parâmetro das redes. Esses modelos realizam a evolução desacoplada da topologia e dos pesos para a construção das RNAs. Para o ajuste da topologia, é utilizada a CGP, que é uma técnica de otimização discreta. Para o ajuste dos pesos, são utilizadas as seguintes técnicas de otimização contínua: (i) evolução diferencial (DE), e (ii) *backpropagation* (BP). A combinação dos modelos CGP e DE é denominada CGPDE e a combinação dos modelos CGP e BP é denominada CGPBP.

Para cada modelo, foram elaboradas duas variantes, denominadas IN e OUT. Dessa forma, o modelo CGPDE apresenta as variantes CGPDE-IN e CGPDE-OUT, e o modelo CGPBP apresenta as variantes CGPBP-IN e CGPBP-OUT. Nas variantes IN, o ajuste das topologias e dos pesos é realizado de forma intercalada e periódica, i.e., ora as topologias são ajustadas e os pesos são mantidos constantes, ora os pesos são ajustados e as topologias são mantidas constantes. Nas variantes OUT, o ajuste das topologias e dos pesos é realizado de forma separada, i.e., primeiramente ajustam-se as topologias mantendo os pesos fixos, e apenas após encontrar uma topologia razoável, ajustam-se os pesos dessa configuração topológica particular.

Além do desenvolvimento dos modelos híbridos, este trabalho introduziu uma métrica denominada Esforço de Avaliação. Essa métrica tem o propósito de permitir comparações mais justas entre os algoritmos durante os testes de desempenho.

Com relação aos experimentos computacionais realizados, o primeiro estudo utilizou a acurácia como objetivo. Nesse estudo, foram realizadas três análises distintas: (i) análise comparativa dos métodos, (ii) análise para bases de dados reduzidas, e (iii) análise estrutural das redes produzidas. O segundo estudo utilizou o erro quadrático médio como objetivo. Esse foi o único estudo em que todos os modelos híbridos propostos foram comparados em conjunto. Por fim, o terceiro estudo avaliou o desempenho das métricas acurácia, G-mean, F_β-score, e AUC quando utilizadas como funções objetivo pelos modelos neuroevolutivos. Esse estudo foi aplicado a problemas de classificação de dados desbalanceados.

Os experimentos computacionais demonstraram que os modelos híbridos são capazes de produzir soluções competitivas quando comparados a outras técnicas da literatura. Quando a acurácia foi utilizada como função objetivo, o modelo CGPDE-IN se mostrou superior em relação às demais técnicas consideradas aqui, inclusive para bases de dados reduzidas. Quando o erro quadrático médio foi utilizado como função objetivo, verificou-se que os modelos CGPDE exibiram melhor desempenho geral que os modelos CGPBP. O estudo comparativo entre funções objetivo distintas demonstrou que a otimização das

métricas G-mean e F_β-score produziu modelos com melhor capacidade de classificação e de ranqueamento, com destaque para a métrica G-mean. Considerando todos os experimentos realizados, o modelo CGPDE-IN exibiu desempenho superior em relação aos demais modelos comparados, e portanto pode ser considerado o método mais promissor dentre os métodos propostos nesta dissertação. De forma geral, os resultados obtidos indicam a superioridade das propostas híbridas em relação aos modelos CGPANN e BP tomados como bases comparativas.

De acordo com esses resultados, as hipóteses inicialmente estabelecidas foram reforçadas. Verificou-se que a utilização de técnicas apropriadas para a otimização dos genes de peso produzem modelos com melhor desempenho geral de classificação em relação ao modelo de referência CGPANN, que utiliza uma mesma técnica para otimizar parâmetros de naturezas distintas. Ademais, verificou-se que a estratégia de se buscar um conjunto de pesos que melhor se adapte a uma topologia específica se mostrou superior à formulação da CGPANN, que ajusta simultaneamente todos os parâmetros da rede.

As análises estruturais realizadas sugerem que os parâmetros das redes contribuem de forma interdependente para o desempenho de classificação. A importância conjunta que cada parâmetro exerce sobre o desempenho do modelo reafirma a necessidade de se aplicar técnicas adequadas para o ajuste de cada tipo de parâmetro. Essa conclusão valida a estratégia adotada neste trabalho, visto que a realização de ajustes personalizados para cada parâmetro foi a principal motivação para a implementação dos modelos híbridos propostos. Espera-se que as técnicas de hibridização formuladas possam ser aplicadas a tarefas complexas de aprendizado, bem como possam servir de base para o projeto de novos modelos neuroevolutivos. Por fim, sugere-se como propostas de continuidade desse trabalho investir nos seguintes tópicos relacionados ao tema:

- Estudo acerca dos custos computacionais envolvidos durante as execuções dos modelos propostos, comparando-os aos custos computacionais obtidos por outras técnicas.
- Implementação de técnicas de computação de alto desempenho, como paralelismo em GPU, a fim de reduzir os custos computacionais dos algoritmos.
- Extensão dos modelos propostos, permitindo a evolução das funções de transferência das redes. Diversas funções distintas podem ser utilizadas, e.g., ReLU, tangente hiperbólica, gaussiana, e *swish*.
- Aplicação de versões mais recentes e eficientes dos algoritmos DE e BP para o ajuste dos pesos das redes. Algumas versões do DE são JADE [86] e SaDE [114]. Algumas versões do BP são RPROP [115] e SARPROP [116].
- Ampliação das análises comparativas entre as funções objetivo, por meio da comparação entre as funções objetivo utilizadas neste trabalho e outras funções

objetivo. Algumas funções que podem ser analisadas são Log-Loss [117] e Cohen's Kappa [118].

- Definição de novas estratégias para estabelecimento do valor de limiar de decisão dos modelos que otimizam a AUC, a fim de se obter desempenho satisfatório ao serem avaliados utilizando as métricas de classificação.
- Aplicação dos modelos propostos a tarefas mais complexas de aprendizado supervisionado, e.g., reconhecimento de imagens e classificação de vídeos. E por fim, aplicação dos modelos propostos a problemas de regressão.

REFERÊNCIAS

1 MILLER, J. F. *Cartesian Genetic Programming*. [S.l.]: Springer-Verlag Berlin Heidelberg, 2011.

2 ALPAYDIN, E. *Introduction to Machine Learning*. 2. ed. [S.l.]: The MIT Press, 2010.

3 HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference and prediction*. 2. ed. [S.l.]: Springer, 2009.

4 SUN, P.; GUO, G.; YU, R. Traffic crash prediction based on incremental learning algorithm. In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*. [S.l.: s.n.], 2017. p. 182–185.

5 SHETTY, A.; SHAH, V. Survey of cervical cancer prediction using machine learning: A comparative approach. In: *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2018. p. 1–6.

6 OKUYAMA, T.; GONSALVES, T.; UPADHAY, J. Autonomous driving system based on deep q learnig. In: *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. [S.l.: s.n.], 2018. p. 201–205.

7 BABIKER, M. et al. Automated daily human activity recognition for video surveillance using neural network. In: *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*. [S.l.: s.n.], 2017. p. 1–5.

8 ZHANG, G. P. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 30, n. 4, p. 451–462, Nov 2000. ISSN 1094-6977.

9 TAN, P.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

10 GALLETLY, J. Evolutionary algorithms in theory and practice:: Evolution strategies, evolutionary programming, genetic algorithms. *Kybernetes*, v. 27, n. 8, p. 979–980, 1998.

11 FLOREANO, D.; DÜRR, P.; MATTIUSSI, C. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, v. 1, n. 1, p. 47–62, Mar 2008.

12 TURNER, A. *Evolving Artificial Neural Networks using Cartesian Genetic Programming*. Tese (Doutorado) — University of York, 2015.

13 KHAN, M. M.; KHAN, G. M.; MILLER, J. F. Evolution of neural networks using cartesian genetic programming. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8.

14 MILLER, J. F. et al. *Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study*. 1997.

15 KHAN, M. M.; KHAN, G. M.; MILLER, J. F. Evolution of optimal anns for non-linear control problems using cartesian genetic programming. In: *In Proceedings of IEEE International Conference in Artificial Intelligence*. [S.l.: s.n.], 2010.

16 ALI, J. et al. Future clients' requests estimation for dynamic resource allocation in cloud data center using cgpann. In: *2013 12th International Conference on Machine Learning and Applications*. [S.l.: s.n.], 2013. v. 2, p. 331–334.

17 AHMAD, A. M. et al. Breast cancer detection using cartesian genetic programming evolved artificial neural networks. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: [s.n.], 2012. (GECCO '12), p. 1031–1038.

18 KHAN, N. M.; KHAN, G. M. Audio signal reconstruction using cartesian genetic programming evolved artificial neural network (cgpann). In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.: s.n.], 2017. p. 568–573.

19 TALBI, E. A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning. In: _____. *Hybrid Metaheuristics*. [S.l.]: Springer Berlin Heidelberg, 2013. p. 3–76.

20 LUIS, S.; SANTOS, M. dos. On the evolvability of a hybrid ant colony-cartesian genetic programming methodology. In: KRAWIEC, K. et al. (Ed.). *Genetic Programming*. [S.l.]: Springer Berlin Heidelberg, 2013. p. 109–120.

21 NOBILE, M. et al. Reverse engineering of kinetic reaction networks by means of cartesian genetic programming and particle swarm optimization. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2013. p. 1594–1601.

22 ELOLA, A. et al. Hybridizing cartesian genetic programming and harmony search for adaptive feature construction in supervised learning problems. *Applied Soft Computing*, v. 52, p. 760 – 770, 2017. ISSN 1568-4946.

23 YAZDANI, S.; SHANBEHZADEH, J. Balanced cartesian genetic programming via migration and opposition-based learning: application to symbolic regression. *Genetic Programming and Evolvable Machines*, v. 16, n. 2, p. 133–150, Jun 2015.

24 VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Berlin, Heidelberg: Springer-Verlag, 1995.

25 CASTRO, C. L. *Novos critérios para seleção de modelos neurais em problemas de classificação com dados desbalanceados*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Belo Horizonte, Out 2011.

26 UROLAGIN, S.; K.V., P.; REDDY, N. V. S. Generalization capability of artificial neural network incorporated with pruning method. In: THILAGAM, P. S. et al. (Ed.). *Advanced Computing, Networking and Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 171–178. ISBN 978-3-642-29280-4.

27 LóPEZ, V. et al. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, v. 250, p. 113 – 141, 2013.

28 FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters*, v. 27, n. 8, p. 861 – 874, 2006. ROC Analysis in Pattern Recognition.

29 KUBAT, M.; HOLTE, R. C.; MATWIN, S. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, v. 30, n. 2, p. 195–215, Feb 1998. ISSN 1573-0565.

30 SUN, Y.; WONG, A. K. C.; KAMEN, M. S. Classification of imbalanced data: A review. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, v. 23, n. 04, p. 687–719, 2009.

31 GOLIK, P.; DOETSCH, P.; NEY, H. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In: BIMBOT, F. et al. (Ed.). *INTERSPEECH*. [S.l.]: ISCA, 2013.

32 EGAN, J. P. *Signal detection theory and ROC-analysis*. [S.l.]: New York : Academic Press, 1975.

33 YAN, L. et al. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. [S.l.]: AAAI Press, 2003. (ICML'03), p. 848–855.

34 HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

35 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. In: RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). Cambridge, MA, USA: MIT Press, 1986. cap. Learning Internal Representations by Error Propagation, p. 318–362.

36 LECUN, Y. A theoretical framework for back-propagation. In: _____. *Artificial neural networks*. [S.l.]: IEEE Computer Society Press, 1992.

37 LUENBERGER, D. G.; YE, Y. *Linear and Nonlinear Programming*. [S.l.]: Springer Publishing Company, Incorporated, 2015.

38 CSÁJI, B. C. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, Citeseer, v. 24, p. 48, 2001.

39 QIAN, N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, Elsevier Science Ltd., Oxford, UK, v. 12, n. 1, p. 145–151, jan. 1999.

40 DARWIN, C. On the origins of species by means of natural selection. *London: Murray*, v. 247, 1859.

41 EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. [S.l.]: SpringerVerlag, 2003.

42 LUKE, S. *Essentials of Metaheuristics*. second. [S.l.]: Lulu, 2013.

43 GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

44 VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. [S.l.: s.n.], 2016. p. 261–265.

45 HOLLAND, J. H. Genetic algorithms and adaptation. In: _____. *Adaptive Control of Ill-Defined Systems*. Boston, MA: Springer US, 1984. p. 317–333.

46 KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

47 STORN, R.; PRICE, K. *Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI, USA, 1995. TR-95-012.

48 RECHENBERG, I. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Tese (Doutorado) — TU Berlin, 1971.

49 FOGEL, D. B.; FOGEL, L. J. An introduction to evolutionary programming. In: ALLIOT, J.-M. et al. (Ed.). *Artificial Evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 21–33.

50 JONG, K. D. Evolutionary computation: A unified approach. In: *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2012. (GECCO '12), p. 737–750.

51 HOEKSTRA, V. An overview of neuroevolution techniques. 2011.

52 YAO, X. Evolving artificial neural networks. *Proceedings of the IEEE*, v. 87, n. 9, p. 1423–1447, Sept 1999.

53 BELEW, R.; MCINERNEY, J.; SCHRAUDOLPH, N. *Evolving networks: Using the genetic algorithm with connectionist learning*. Cognitive Computer Science Research Group, Computer Science & Engr. Dept., University California, San Diego, 1990.

54 GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. [S.l.]: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256.

55 ANGELINE, P. J.; SAUNDERS, G. M.; POLLACK, J. B. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, v. 5, n. 1, p. 54–65, Jan 1994.

56 DUCH, W.; JANKOWSKI, N. Survey of neural transfer functions. *Neural Computing Surveys*, v. 2, n. 1, p. 163–212, 1999.

57 DUCH, W.; JANKOWSKI, N. Transfer functions: Hidden possibilities for better neural networks. In: *9th European Symposium on Artificial Neural Networks (ESANN)*. [S.l.: s.n.], 2001. p. 81–94.

58 CHALMERS, D. The evolution of learning: An experiment in genetic connectionism. In: *Proceedings of the 1990 Connectionist Summer School*. San Mateo, CA: Morgan Kaufmann, 1992.

59 LUKOŠEVIČIUS, M.; JAEGER, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, Elsevier, v. 3, n. 3, p. 127–149, 2009.

60 GOMEZ, F.; SCHMIDHUBER, J.; MIIKKULAINEN, R. Efficient non-linear control through neuroevolution. In: FÜRNKRANZ, J.; SCHEFFER, T.; SPILIOPOULOU, M. (Ed.). *Machine Learning: ECML 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 654–662.

61 GARCIA-PEDRAJAS, N.; HERVAS-MARTINEZ, C.; MUNOZ-PEREZ, J. Covnet: a cooperative coevolutionary model for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, v. 14, n. 3, p. 575–596, May 2003.

62 STANLEY, K. O.; MIIKKULAINEN, R. Efficient evolution of neural network topologies. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. [S.l.: s.n.], 2002. v. 2, p. 1757–1762.

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

64 FRIEDBERG, R. M. A learning machine: Part i. *IBM Journal of Research and Development*, v. 2, p. 2–13, 1958.

65 SMITH, S. F. *A Learning System Based on Genetic Adaptive Algorithms*. Tese (Doutorado) — University of Pittsburgh, Pittsburgh, PA, USA, 1980.

66 CRAMER, N. L. A representation for the adaptive generation of simple sequential programs. In: *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985. p. 183–187.

67 MILLER, J. F.; THOMSON, P. Cartesian genetic programming. In: POLI, R. et al. (Ed.). *Proceedings of the Third European Conference on Genetic Programming (EuroGP)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. v. 1802, p. 121–132.

68 SEKANINA, L. Image filter design with evolvable hardware. In: CAGNONI, S. et al. (Ed.). *Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 255–266.

69 HOPE, D. C.; MUNDAY, E.; SMITH, S. L. Evolutionary algorithms in the classification of mammograms. In: *2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing*. [S.l.: s.n.], 2007. p. 258–265.

70 BENTLEY, P. J.; LIM, S. L. Fault tolerant fusion of office sensor data using cartesian genetic programming. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.: s.n.], 2017. p. 1–8.

71 ASHMORE, L.; MILLER, J. F. *Evolutionary Art with Cartesian Genetic Programming*. University of York, 2003.

72 TURNER, A. J.; MILLER, J. F. Cartesian genetic programming encoded artificial neural networks: A comparison using three benchmarks. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2013. (GECCO '13), p. 1005–1012.

73 POPESCU, M. C. et al. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, v. 8, n. 7, p. 579–588, 2006.

74 WIELAND, A. P. Evolving neural network controllers for unstable systems. In: *IJCNN-91-Seattle International Joint Conference on Neural Networks*. [S.l.: s.n.], 1991. ii, p. 667–673 vol.2.

75 STANLEY, K. O.; MIIKKULAINEN, R. Efficient evolution of neural network topologies. In: *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*. [S.l.: s.n.], 2002. v. 2, p. 1757–1762.

76 KHAN, N. M.; KHAN, G. M. Audio signal reconstruction using cartesian genetic programming evolved artificial neural network (cgpann). In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.: s.n.], 2017. p. 568–573.

77 AHMAD, A. M.; KHAN, G. M. Bio-signal processing using cartesian genetic programming evolved artificial neural network (cgpann). In: *2012 10th International Conference on Frontiers of Information Technology*. [S.l.: s.n.], 2012. p. 261–268.

78 AHMAD, A. M.; KHAN, G. M.; MAHMUD, S. A. Classification of arrhythmia types using cartesian genetic programming evolved artificial neural networks. In: ILIADIS, L.; PAPADOPOULOS, H.; JAYNE, C. (Ed.). *Engineering Applications of Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 282–291.

79 ALI, J. et al. Future clients' requests estimation for dynamic resource allocation in cloud data center using cgpann. In: *2013 12th International Conference on Machine Learning and Applications*. [S.l.: s.n.], 2013. v. 2, p. 331–334.

80 DAS, S.; SUGANTHAN, P. N. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, v. 15, n. 1, p. 4–31, Feb 2011.

81 WANG, W. et al. Differential evolution algorithm and method of moments for the design of low-rcs antenna. *IEEE Antennas and Wireless Propagation Letters*, v. 9, p. 295–298, April 2010.

82 ASLANTAS, V.; TOPRAK, A. N. Multi focus image fusion by differential evolution algorithm. In: *2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. [S.l.: s.n.], 2014. v. 01, p. 312–317.

83 HUANG, S. Y.; LIN, C. J. Using neural networks with differential evolution learning for face recognition. In: *2014 International Symposium on Computer, Consumer and Control*. [S.l.: s.n.], 2014. p. 376–379.

84 KUMARI, S. et al. Study of load frequency control by using differential evolution algorithm. In: *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*. [S.l.: s.n.], 2016. p. 1–5.

85 DAS, S. et al. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 3, p. 526–553, June 2009.

86 ZHANG, J.; SANDERSON, A. C. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 5, p. 945–958, Oct 2009.

87 KENNEDY, J.; EBERHART, R. C. *Swarm Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN 1-55860-595-9.

88 BERGH, F. van den; ENGELBRECHT, A. P. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 3, p. 225–239, June 2004.

89 ONG, Y. S.; KEANE, A. J. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 2, p. 99–110, April 2004.

90 DEB, K.; ANAND, A.; JOSHI, D. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, v. 10, n. 4, p. 371–395, Dec 2002.

91 QING, A. Basics of differential evolution. In: _____. *Differential Evolution in Electromagnetics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 19–42.

92 NETO, J. M. M.; BERNARDINO, H. S.; BARBOSA, H. J. C. Hybridization of cartesian genetic programming and differential evolution for generating classifiers based on neural networks. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.: s.n.], 2018. p. 1–8.

93 WERBOS, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, v. 78, n. 10, p. 1550–1560, Oct 1990. ISSN 0018-9219.

94 DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, v. 91, n. 2, p. 201–213, Jan 2002.

95 BARBOSA, H. J. C.; BERNARDINO, H. S.; BARRETO, A. M. S. Using performance profiles to analyze the results of the 2006 cec constrained optimization competition. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8.

96 DHEERU, D.; TANISKIDOU, E. K. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

97 MANGASARIAN, O.; SETIONO, R.; WOLBERG, W. H. *Pattern Recognition Via Linear Programming: Theory And Application To Medical Diagnosis*. 1990.

98 MORENO-TORRES, J. G.; SAEZ, J. A.; HERRERA, F. Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, v. 23, n. 8, p. 1304–1312, Aug 2012.

99 BOSLAUGH, S. *Statistics in a Nutshell, 2nd Edition*. [S.l.]: O'Reilly Media, Inc., 2012.

100 MATLAB *Statistics and Machine Learning Toolbox*. 2018. `https://www.mathworks.com/help/stats/ttest.html`.

101 TURNER, A. J.; MILLER, J. F. Introducing a cross platform open source cartesian genetic programming library. *Genetic Programming and Evolvable Machines*, v. 16, n. 1, p. 83–91, Mar 2015.

102 NISSEN, S. Implementation of a Fast Artificial Neural Network Library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU)*, v. 31, 2003.

103 LI, J. et al. Brief introduction of back propagation (bp) neural network algorithm and its improvement. In: JIN, D.; LIN, S. (Ed.). *Advances in Computer Science and Information Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 553–558.

104 MATLAB *Gradient Descent Backpropagation*. 2018. `https://www.mathworks.com/help/nnet/ref/traingd.html`.

105 MATLAB *Pattern Recognition Network*. 2018. `https://www.mathworks.com/help/nnet/ref/patternnet.html`.

106 OONG, T. H.; ISA, N. A. M. Adaptive evolutionary artificial neural networks for pattern classification. *IEEE Transactions on Neural Networks*, v. 22, n. 11, p. 1823–1836, Nov 2011.

107 NETO, J. M. M.; CASTRO, C. L. Otimização da auc via redes neurais evolutivas para classificação de dados desbalanceados. In: *Congresso Brasileiro de Inteligência Computacional*. [S.l.: s.n.], 2017. p. 1–12.

108 LU, B.-L. et al. An efficient multilayer quadratic perceptron for pattern classification and function approximation. In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. [S.l.: s.n.], 1993. v. 2, p. 1385–1388 vol.2.

109 ANGELO, J. S.; KREMPSER, E.; BARBOSA, H. J. C. Differential evolution for bilevel programming. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2013. p. 470–477.

110 MATLAB *Pearson's Linear Correlation Coefficient*. 2018. `https://www.mathworks.com/help/stats/corr.html`.

111 SINGH, S.; SINGH, D. S.; KUMAR, S. Modified mean square error algorithm with reduced cost of training and simulation time for character recognition in backpropagation neural network. In: SATAPATHY, S. C.; UDGATA, S. K.; BISWAL, B. N. (Ed.). *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*. [S.l.]: Springer International Publishing, 2014. p. 137–145. ISBN 978-3-319-02931-3.

112 HOSSIN, M.; M.N, S. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, v. 5, p. 01–11, March 2015.

113 JENI, L. A.; COHN, J. F.; TORRE, F. D. L. Facing imbalanced data–recommendations for the use of performance metrics. In: *2013 Humane Association Conf. on Affective Computing and Intelligent Interaction*. [S.l.: s.n.], 2013. p. 245–251.

114 QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 2, p. 398–417, April 2009.

115 RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: *IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1993. v. 1, p. 586–591.

116 TREADGOLD, N. K.; GEDEON, T. D. The sarprop algorithm: A simulated annealing enhancement to resilient back propagation. In: *Proceedings International Panel Conference on Soft and Intelligent Computing*. [S.l.: s.n.], 1996.

117 de Boer, P. T. et al. A tutorial on the cross-entropy method. *Annals of operations research*, Springer Netherlands, v. 134, n. 1, p. 19–67, 1 2005. ISSN 0254-5330.

118 SIM, J.; WRIGHT, C. C. The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements. *Physical Therapy*, v. 85, n. 3, p. 257–268, 03 2005.