UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

ANGÉLICA LOURENÇO OLIVEIRA

Perceptron Dilatação-Erosão Linear com Treinamento Baseado em Otimização DC: Aplicações em Problemas de Regressão e Classificação

Campinas

2023

Angélica Lourenço Oliveira

Perceptron Dilatação-Erosão Linear com Treinamento Baseado em Otimização DC: Aplicações em Problemas de Regressão e Classificação

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutora em Matemática Aplicada.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita

Este trabalho corresponde à versão final da Tese defendida pela aluna Angélica Lourenço Oliveira e orientada pelo Prof. Dr. Marcos Eduardo Ribeiro do Valle Mesquita .

Campinas

2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

OL4p

Oliveira, Angélica Lourenço, 1993-
Perceptron Dilatação-Erosão Linear com treinamento baseado em otimização DC : aplicações em problemas de regressão e classificação / Angélica Lourenço Oliveira. – Campinas, SP : [s.n.], 2023.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Morfologia matemática. 2. Redes neurais morfológicas. 3. Aprendizagem supervisionada (Aprendizado do computador). 4. Otimização matemática. I. Mesquita, Marcos Eduardo Ribeiro do Valle, 1979-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações Complementares

Título em outro idioma: Linear Dilation-Erosion Perceptron with training based on DC optimization : applications in regression and classification problems
Palavras-chave em inglês:
Mathematical morphology
Morphological neural networks
Supervised learning (Machine learning)
Mathematical optimization
Área de concentração: Matemática Aplicada
Titulação: Doutora em Matemática Aplicada
Banca examinadora:
Marcos Eduardo Ribeiro do Valle Mesquita [Orientador]
Peter Sussner
Francisco de Assis Magalhães Gomes Neto
Guilherme de Alencar Barreto
Geovani Nunes Grapiglia
Data de defesa: 26-01-2023
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-8689-8522
- Currículo Lattes do autor: http://lattes.cnpq.br/5580156622219200

Tese de Doutorado defendida em 26 de janeiro de 2023 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARCOS EDUARDO RIBEIRO DO VALLE MESQUITA

Prof(a). Dr(a). PETER SUSSNER

Prof(a). Dr(a). FRANCISCO DE ASSIS MAGALHÃES GOMES NETO

Prof(a). Dr(a). GUILHERME DE ALENCAR BARRETO

Prof(a). Dr(a). GEOVANI NUNES GRAPIGLIA

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

A todos que me acompanharam nesta jornada...

Agradecimentos

Primeiramente, quero agradecer ao meu esposo Reginaldo por todo apoio e suporte para não permitir que eu desistisse do mundo acadêmico.

Agradeço ao criador do universo por permitir que eu chegasse a este momento.

Um agradecimento mais que especial ao meu orientador Marcos Valle pela excelente orientação, apoio, suporte, conhecimentos, paciência e por me ouvir e entender os momentos que passei durante o doutorado.

Aos professores membros da banca por aceitarem o convite de fazer parte. Agradeço desde já por todas observações, correções e sugestões feitas para este trabalho.

Agradeço a todos os meus amigos do doutorado, em especial ao pessoal da salinha 1B (e agregados) pelos cafés, apoio e bate-papos.

Agradeço à minha mãe que, mesmo sem saber como funciona o meio acadêmico, sempre reconheceu meu esforço e me defendeu quando mais precisei. Não é simplesmente "só estudar".

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

Resumo

A morfologia matemática (MM) é uma teoria de operadores não lineares usada para o processamento e análise de imagens. Redes neurais morfológicas são redes neurais artificiais cujos neurônios computam operadores morfológicos. Dilatações e erosões são operadores elementares da MM. Do ponto de vista algébrico, uma dilatação e uma erosão são operadores que comutam respectivamente com as operações de supremo e ínfimo. Esta tese apresenta uma rede neural morfológica híbrida denominada *perceptron dilatação-erosão linear* (ℓ-DEP). Os modelos ℓ-DEP são dados aplicando transformações lineares antes de computar uma dilatação e uma erosão. Em seguida, é feita uma combinação convexa dessa composição de operadores lineares e morfológicos. Um modelo ℓ-DEP produz uma função linear por partes contínua e, portanto, é um aproximador universal. Nesta tese, aplicamos o ℓ-DEP para tarefas de regressão e de classificação. Apresentamos o treinamento de uma rede ℓ-DEP como um problema de otimização côncava-convexa, também conhecido com problema de otimização de diferença de funções convexas (otimização DC). Usando diversos problemas de regressão e também de classificação binária, comparamos o desempenho dos modelos preditivos ℓ-DEP com outras técnicas de aprendizado de máquinas. Os experimentos computacionais apoiam a potencial aplicação do modelo ℓ-DEP proposto para tarefas genéricas de regressão e de classificação binária.

Palavras-chave: Morfologia matemática; rede neural morfológica; otimização DC; regressão; classificação binária.

Abstract

Mathematical morphology (MM) is a theory of nonlinear operators used for image processing and analysis. Morphological neural networks are artificial neural networks whose neurons compute morphological operators. Dilations and erosions are elementary operators of MM. From an algebraic point of view, a dilation and an erosion are operators that commute respectively with the operations of supremum and infimum. This thesis presents a hybrid morphological neural network named *linear dilation-erosion perceptron* (ℓ-DEP). The ℓ-DEP models are given by applying linear transformations before computing a dilation and an erosion. Then, a convex combination of this composition of linear and morphological operators is made. An ℓ-DEP model produces a continuous piecewise linear function and is a universal approximator. In this thesis, we apply ℓ-DEP for regression and classification tasks. We present the training of a ℓ-DEP network as a concave-convex optimization problem, also known as a DC optimization problem. Using several regression and binary classification problems, we compared the performance of ℓ-DEP predictive models with other machine learning techniques. Computational experiments support the potential application of the proposed ℓ-DEP model to generic regression and binary classification tasks.

Keywords: Mathematical morphology, morphological neural network, DC optimization, regression, binary classification.

Lista de ilustrações

Lista de tabelas

Lista de símbolos

$\mathbb{R}$	Conjunto dos números reais
$\bar{\mathbb{R}}$	Conjunto dos números reais estendidos
$\mathbb{R}^n$	Conjunto dos vetores $\mathbf{x} = [x_1 \cdots x_n]^T$ com $x_i \in \mathbb{R}, \forall i = 1, \ldots, n$
$\bar{\mathbb{R}}^n$	Conjunto dos vetores $\mathbf{x} = [x_1 \cdots x_n]^T$ com $x_i \in \bar{\mathbb{R}}, \forall i = 1, \ldots, n$
$\mathbb{R}^{m \times n}$	Conjunto das matrizes reais com m linhas e n colunas
$\|\cdot\|_p$	Norma-p vetorial
$\max\{\cdot\}$	Valor máximo
$\arg\max\{\cdot\}$	Argumento do valor máximo
$\sup\{\cdot\}$	Supremo
$\inf\{\cdot\}$	Ínfimo
$\leqslant$	Relação de ordem menor ou igual usual em $\mathbb{R}$
$\geqslant$	Relação de ordem maior ou igual usual em $\mathbb{R}$
$\preceq$	Relação de ordem genérica
$\langle \cdot, \cdot \rangle$	Produto interno
$\mathbf{dom} f$	Domínio de uma função f
$\partial f(\mathbf{x})$	Conjunto dos subgradientes de f em $\mathbf{x}$
f^*	Conjugada da função f
f^{**}	Biconjugada da função f
Γ_0	Espaço das funções próprias semicontínuas inferiormente
$\mathcal{P}_{\mathcal{V}}(X, r)$	Partição de Voronoi de X com r centroides

Lista de algoritmos

Sumário

Introdução

As técnicas de aprendizado de máquina (AM) desempenham um papel importante no reconhecimento de padrões e na computação flexível. As redes neurais, inspiradas no sistema nervoso biológico, estão entre as técnicas de aprendizado de máquina mais eficientes usadas para resolver tarefas de reconhecimento de padrões, incluindo, por exemplo, visão computacional e processamento de linguagem natural [1]. Existem muitos modelos de redes neurais interessantes e eficazes, tais como *perceptrons* de múltiplas camadas, máquinas de vetores de suporte, redes neurais convolucionais e redes neurais recorrentes [2–4]. Neste texto nos concentraremos nos modelos relevantes para o desenvolvimento e avaliação da rede neural artificial denominada *perceptron* dilatação-erosão linear, usada para tarefas de regressão e de classificação.

Introdução às redes neurais clássicas

Em primeiro lugar, é amplamente conhecido que o *perceptron*, introduzido por Rosenblatt no final dos anos 1950, pode ser usado para tarefas de classificação binária [5]. Além disso, Rosenblatt propôs uma regra de aprendizagem que converge sempre que as amostras das duas classes são linearmente separáveis. A rede *perceptron* multicamadas (MLP, *multilayer perceptrons*), com pelo menos uma camada oculta e treinada usando algoritmos de retropropagação, supera as limitações do *perceptron* de Rosenblatt para tarefas de classificação em que os dados não são linearmente separáveis [2, 6]. Além das tarefas de classificação, tarefas de regressão também podem ser resolvidas utilizando a rede MLP.

As máquinas de vetores de suporte (SVM, *support vector machines*), desenvolvidas em 1992 por Vapnik e colaboradores, são algoritmos para classificação e regressão fortemente baseada na teoria do aprendizado estatístico e otimização quadrática. Os classificadores de vetores de suporte (SVCs, *support vector classifiers*), assim como o *perceptron*, preveem a classe de uma amostra calculando uma soma ponderada. Porém, o treinamento de um SVC é realizado maximizando a margem de separação entre as duas classes [2]. Os regressores de vetores de suporte (SVRs, *support vector regressors*) são usados para resolver problemas de regressão e seu treinamento é feito minimizando a margem em que os dados de treinamento podem estar [6].

O truque do *kernel* pode ser utilizado de modo a melhorar o desempenho dos SVMs. O truque do *kernel* consiste em mapear os dados no espaço de características a outro espaço de maior dimensão de forma a tentar tornar os dados linearmente separáveis e então aplicar a técnica padrão para dados linearmente separáveis. O desempenho do SVR e

do SVC aumenta significativamente usando o truque do *kernel*, permitindo que os modelos SVR e SVC sejam efetivamente aplicados a problemas de regressão e de classificação não lineares, respectivamente. Os SVCs, usando o truque do *kernel*, também superam as limitações dos *perceptrons* para tarefas de classificação binária.

Introdução às redes neurais morfológicas

No final dos anos 1980, Ritter e colaboradores desenvolveram a chamada álgebra de imagens como uma tentativa de fornecer uma estrutura unificada para técnicas de processamento e análise de imagens [7]. Uma teoria não linear amplamente utilizada para processamento e análise de imagens é a denominada morfologia matemática [8, 9], cujas operações elementares são dilatações e erosões. De forma geral, uma rede neural morfológica é uma rede neural em cujos neurônios são definidos utilizando operações morfológicas.

Em meados dos anos 1990, ao substituir o produto escalar linear usual no modelo *perceptron* de Rosenblatt por uma operação correspondente baseada em reticulados e usada para definir operadores morfológicos, Sussner e Ritter estabeleceram uma nova classe de modelos denominados *perceptrons* morfológicos [10, 11]. Usando a álgebra de imagens, os operadores lineares e morfológicos são definidos de forma análoga.

Os *perceptrons* morfológicos e as memórias associativas morfológicas foram as primeiras redes neurais morfológicas cujas unidades de processamento, os neurônios, realizam dilatações e erosões [10, 12]. Em termos gerais, os neurônios morfológicos são obtidos substituindo o produto escalar usual por um máximo de somas ou o mínimo de somas.

Um *perceptron* baseado em dilatação classifica um padrão calculando o máximo das entradas somadas aos pesos sinápticos correspondentes. Dualmente, a função de decisão de um *perceptron* morfológico baseado na erosão é dada pelo mínimo das entradas adicionadas aos pesos sinápticos. Uma rede neural morfológica com arquitetura semelhante a rede MLP, em que os neurônios lineares são substituídos por neurônios morfológicos baseados exclusivamente ou em dilatação ou em erosão, não tem capacidade de aproximação universal, conforme mostrado pelos autores em [13]. No entanto, assim como o MLP, um *perceptron* morfológico de várias camadas (MLMP, *multilayer morphological perceptron*), com pelo menos uma camada oculta, com neurônios baseados em dilatação e também em erosão, pode teoricamente resolver qualquer tarefa de classificação binária [14].

Do ponto de vista geométrico, um MLMP separa as duas classes incluindo os padrões em hiper-caixas [14, 15]. Com isso, alguns pesquisadores vêm desenvolvendo algoritmos de treinamento para tarefas de classificação em que a estrutura da rede cresce adicionando hiper-caixas para ajustar os dados de treinamento [14–16]. Um ponto negativo

desses modelos é que eles podem superajustar os dados de treinamento.

Devido às operações de máximo e mínimo, as redes neurais morfológicas são geralmente mais baratas computacionalmente do que os modelos tradicionais. No entanto, treinar redes neurais morfológicas costuma ser um grande desafio devido à não diferenciabilidade das operações baseadas em reticulados [17], dificultando, por exemplo, o uso de algoritmos baseados em gradiente. Neste texto apresentaremos um modelo morfológico em que essa dificuldade não seja um problema.

Introdução às redes morfológicas híbridas

Alguns trabalhos recentes sugerem que redes neurais morfológicas com apenas neurônios morfológicos tem capacidade limitada para alguns tipos de tarefas em AM. Investigações vêm sendo feitas para analisar a capacidade de aproximador universal de redes neurais morfológicas.Visando encontrar alternativas para melhorar a capacidade e o desempenho das redes neurais morfológicas, alguns autores vêm fazendo o uso de neurônios lineares juntamente com neurônios morfológicos nas arquiteturas das redes neurais morfológicas. Recentemente, redes neurais híbridas, que utilizam de neurônios morfológicos e lineares, vêm sendo estudadas e estão apresentando bons resultados em AM [18–20].

Neste trabalho, vamos apresentar modelos com arquiteturas semelhantes, porém com diferentes formas de treinamento.

Em 2013, Goodfellow e outros introduziram uma classe de redes neurais para tarefas de classificação denominadas redes *maxout* [21]. As redes *maxout* produzem funções contínuas lineares por partes e, assim como as redes neurais tradicionais, estas redes são geralmente treinadas usando o método gradiente descendente estocástico (SGD, *stochastic gradient descent*) [21, 22].

Em 2017, Araújo apresentou uma classe particular de redes neurais morfológicas com estrutura multicamada, chamada de rede neural de dilatação-erosão (DENN), para lidar com problemas de classificação binária [23]. A rede DENN possui unidades morfológicas que possuem uma combinação balanceada entre as operações elementares da morfologia matemática. O treinamento da rede DENN corresponde a um processo de aprendizagem apresentado em [24] por Pessoa e Maragos baseado em gradiente e posição (*ranking*).

Em [25], Mondal e colaboradores propuseram uma rede neural híbrida denominada rede morfológica densa (MDN, *morphological dense network*), cujos blocos de processamento são compostos por operações morfológicas e as saídas são formadas por neurônios lineares. Apesar de ser efetivamente um modelo híbrido, esse conjunto de operação é denominada bloco morfológico por Mondal et. al. Um bloco morfológico consiste

em uma camada de dilatações e erosões em paralelo, seguida por uma combinação linear. Os autores apresentaram resultados empíricos sugestivos sobre o fato de qualquer função poder ser aproximada por uma rede MDN, com erro de aproximação diminuindo com o aumento do número de neurônios morfológicos. Posteriormente, em [20] eles provaram que as redes morfológicas densas com duas camadas podem aproximar funções contínuas em conjuntos compactos arbitrários. O treinamento destas redes apresentado pelos autores envolve o uso do método *back-propagation* com otimizador Adam [26].

Em 2019, Hernández et al. [27] propuseram duas redes morfológicas híbridas capazes de extrair características em tarefas de classificação não linear. Tais redes são denominadas Rede Neural Morfológica-Linear (MLNN, *morphological-linear neural network*), que consiste em uma camada oculta de neurônios morfológicos e uma camada de saída de perceptrons clássicos e a Rede Neural Linear-Morfológica (LMNN, *linear-morphological neural network*), que é composta por uma ou várias camadas de perceptrons seguida por uma camada de saída de neurônios morfológicos. Hernández et al. usaram o método gradiente descendente estocástico (SGD, *stochastic gradient descent*) para treinar suas redes neurais morfológicas híbridas [27].

Utilizando o método de quadrados mínimos e aprendizado extremo [28], Sussner e Campiotti [19] propuseram o treinamento de uma rede neural morfológica híbrida (HLM-ELM, *hybrid linear morphological extreme learning machine*) equipada tanto com operações morfológicas quanto neurônios clássicos com função de ativação sigmoidal. Tal rede generaliza o modelo perceptron multicamada (MLP) com uma única camada oculta de neurônios. O treinamento desta rede, utilizando a técnica de aprendizado extremo, rendeu bons resultados para tarefas de classificação.

Introdução à rede híbrida ℓ-DEP

Utilizando o método SGD, Araújo [29] propôs o treinamento de uma rede neural morfológica denominada *perceptron* dilatação-erosão (DEP, *dilation-erosion perceptron*). Um modelo DEP é uma rede neural morfológica híbrida obtida por uma combinação convexa de um *perceptron* morfológico baseado em dilatação e de um *perceptron* morfológico baseado em erosão.

Em [30], Charisopoulos e Maragos formularam o treinamento do modelo DEP como um problema de otimização côncavo-convexo que, em certo sentido, se assemelha ao treinamento de um SVC linear. Por se tratar de um operador crescente, este modelo também pressupõe uma relação de ordem parcial entre entradas e saídas. Felizmente, pode-se contornar este problema adicionando neurônios que realizam anti-dilatações e anti-erosões [16]. Por exemplo, considerando a importância das estruturas dendríticas, Ritter e Urcid apresentaram um único neurônio morfológico que pode ser utilizados para contornar as limitações do modelo DEP [31].

Uma proposta para contornar as limitações do modelo DEP, foi apresentada recentemente por Valle em [32]. Tal proposta consiste em utilizar uma ordem reduzida no treinamento do modelo DEP. Usando conceitos da morfologia matemática com valores vetoriais, Valle propôs o *perceptron* dilatação-erosão reduzido (r-DEP, *reduced dilation-erosion perceptron*), que supera as limitações do *perceptron* morfológico. Em poucas palavras, o r-DEP primeiro transforma o vetor de entrada em outro vetor de características. O novo vetor de características é então classificado usando uma combinação linear do *perceptron* baseado em dilatação e do *perceptron* baseado em erosão, que correspondem ao modelo DEP treinado utilizando um procedimento côncavo-convexo.

Infelizmente, escolher um mapeamento adequado é a tarefa mais desafiadora para projetar um modelo r-DEP eficiente. Como solução, neste trabalho propomos os chamados *perceptrons* dilatação-erosão linear (ℓ-DEP, *linear dilation-erosion perceptron*), considerando transformações lineares ao invés de mapeamentos arbitrários [33]. O ℓ-DEP está intimamente relacionado a uma das duas redes neurais morfológicas híbridas investigadas por Hernández et al. para classificação de *big data* [27], a saber, a rede LMNN. Portanto, diferente do modelos MDN e HML-ELM, o ℓ-DEP é obtido primeiro aplicando transformações lineares nos dados e depois os operações elementares da MM em reticulados completos.

Assim como muitas redes neurais tradicionais, o modelo ℓ-DEP é um aproximador universal, ou seja, ele pode aproximar uma função contínua dentro de qualquer precisão desejada em uma região compacta em um espaço euclidiano [34]. Do ponto de vista matemático, o aproximador universal produzido pelo modelo ℓ-DEP é uma função linear contínua por partes [33, 35].

Contribuições e organização da Tese

Neste trabalho, faremos uma investigação de diferentes métodos de treinamento do modelo ℓ-DEP. Apresentaremos aplicações em aprendizado de máquinas supervisionado, a saber, classificação e regressão. Os treinamentos serão formulados utilizando otimização DC. Testes computacionais comparando nosso modelo com outras técnicas clássicas de aprendizado de máquinas mostram o potencial desta nova rede neural morfológica.

É listado a seguir algumas das principais contribuições desta tese:

- Apresentação de um novo modelo de aprendizado de máquinas supervisionado, dado pela combinação convexa entre operadores morfológicos aplicados em dados mapeados por transformações lineares.
- Apresentação do modelo como uma rede neural morfológica híbrida cuja saída é dada pela diferença entre dois operadores morfológicos convexos, seguida da aplicação de

uma função de ativação.

- Conclusão de que a rede neural morfológica híbrida é um aproximador universal, isto é, pode aproximar qualquer função contínua definida em um compacto.
- Treinamento da rede neural morfológica híbrida com uma abordagem incluindo algoritmos de Otimização DC.
- Aplicação da rede neural morfológica híbrida em diversas tarefas de aprendizado de máquina supervisionado (regressão e classificação).

Este trabalho está dividido da seguinte forma. Nos três primeiros capítulos é fornecida uma base matemática necessária para entender os modelos apresentados nesta tese. No quarto e no quinto capítulo são formulados os treinamentos e as avaliações do modelo ℓ-DEP para tarefas de regressão e de classificação, respectivamente. No sexto capítulo são apresentadas as conclusões e algumas considerações finais a respeito do que foi estudado nesta tese. A seguir são fornecidos mais detalhes sobre o que será apresentado em cada um destes capítulos.

No primeiro capítulo, serão analisados alguns conceitos básicos sobre a otimização convexa e otimização DC, incluindo definições e propriedades de funções convexas e funções DC. Serão apresentados dois algoritmos de otimização DC, a saber o algoritmo de diferença de convexas (DCA, *DC algorithm*) e o procedimento convexo-côncavo (CCP, *concave-convex procedure*), que são necessários para as abordagens de treinamento dos modelos apresentados neste trabalho.

No segundo capítulo, primeiramente apresentaremos alguns conceitos básicos sobre aprendizado de máquinas. Em particular, falaremos sobre modelos preditivos para tarefas de classificação e de regressão, que são as categorias de problemas objeto de estudo desta tese. Também apresentaremos alguns métodos de avaliação de modelos preditivos. Em seguida, apresentaremos algumas técnicas clássicas de aprendizado de máquinas, a saber as máquinas de vetores de suporte e as redes neurais artificiais. Em especial, abordaremos algumas características das redes neurais artificiais tradicionais vistas como aproximadores universais.

Em seguida, no Capítulo 3, falaremos sobre morfologia matemática e suas principais operações. Apresentaremos as chamadas redes neurais morfológicas e o operador dilatação-erosão linear. Veremos que o operador dilatação-erosão linear é um aproximador universal e que ele define a rede neural morfológica híbrida *perceptron* dilatação-erosão linear (ℓ-DEP).

No Capítulo 4 vamos revisar o algoritmo DCA aplicado para treinar o modelo ℓ-DEP para tarefas de regressão (ℓ-DER). Formulamos também o treinamento do modelo ℓ-

DEP como um procedimento CCP [36]. Além disso, avaliamos e comparamos o desempenho do modelo ℓ-DER usando vários problemas de regressão.

No Capítulo 5 formularemos o treinamento da rede ℓ-DEP, para tarefas de classificação, utilizando um procedimento concavo-convexo. Avaliamos e comparamos o desempenho do modelo ℓ-DEP para tarefas de classificação usando vários problemas de classificação binária.

Por fim, finalizamos este trabalho apresentando algumas considerações finais no Capítulo 6, destacando a potencial aplicação do modelo ℓ-DEP proposto para tarefas genéricas de regressão e de classificação binária.

1 Conceitos Básicos em Otimização DC

Programação Matemática, também conhecida como Otimização Matemática, refere-se a modelos matemáticos pelos quais podemos encontrar uma variável que melhor se ajusta a um determinado conjunto de critérios e restrições. A forma genérica de um problema de otimização é dada pelo Problema (1.1), em que $f : \mathcal{V} \to \mathbb{R}$ é a função objetivo e $\mathcal{V}$ o conjunto factível, que é o conjunto de pontos viáveis para o problema [37].

$$\begin{aligned} \textbf{minimizar} \quad & f(\boldsymbol{\alpha}) \\ \text{s.a.} \quad & \boldsymbol{\alpha} \in \mathcal{V}, \end{aligned} \tag{1.1}$$

O objetivo de um problema de otimização também pode ser dado por maximizar uma função objetivo restrita ao conjunto factível. Um problema de maximização pode ser reescrito como um problema de minimização realizando manipulações na função objetivo. De fato, maximizar uma função f é equivalente a minimizar o inverso aditivo de f, ou seja, minimizar a função $-f$. Logo, sem perda de generalidade, consideraremos apenas casos em que o objetivo é minimizar uma função.

Um elemento no conjunto de pontos viáveis que minimiza a função objetivo é denominado minimizador global de f. Se um elemento minimiza f em apenas uma região do conjunto viável, então ele é dito minimizador local. Encontrar um minimizador pode não ser uma tarefa simples ou pode até ser impossível. No entanto, para algumas funções com características especiais, tais como as funções convexas, é possível facilitar a busca por um minimizador da função objetivo [37–39].

A seguir revisaremos os conceitos de conjunto convexo e funções convexas.

Definição 1.1. (Conjunto convexo) *Seja $\mathcal{C} \subset \mathbb{R}^n$. Dizemos que $\mathcal{C}$ é um conjunto convexo se para todo $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{C}$ o elemento $\mathbf{z} = \lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}$, para todo $\lambda \in [0,1]$, também pertence ao conjunto $\mathcal{C}$.*

Em outras palavras, $\mathcal{C}$ é um conjunto convexo se, para cada par de elementos em $\mathcal{C}$, o segmento de reta que os une está inteiramente contido em $\mathcal{C}$.

Definição 1.2. (Função convexa) *Seja $\mathcal{C} \subset \mathbb{R}^n$ um conjunto convexo. Dizemos que a função $f : \mathcal{C} \subset \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$ é uma função convexa se, para todo $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{C}$, vale a desigualdade*

$$f(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}) \leqslant \lambda f(\boldsymbol{\alpha}) + (1-\lambda) f(\boldsymbol{\beta}),$$

para todo $\lambda \in [0,1]$. Se a igualdade não for satisfeita na inequação acima, para $\lambda \in (0,1)$, então dizemos que f é estritamente convexa.

Observe que, na definição acima, a função convexa f pode assumir valores em $\{-\infty, +\infty\}$. Nos casos em que possa ocorrer indeterminação nas operações entre $-\infty$ e $+\infty$, por exemplo, $(+\infty) \pm (-\infty)$, consideramos que a desigualdade é inválida. Além disso, pode-se utilizar o conceito de funções próprias, que serão vistas mais adiante, em que essa indeterminação é descartada.

Em problemas de otimização em que a função objetivo é uma função convexa e o conjunto de pontos viáveis é um conjunto convexo, estamos tratando de um ramo especial da otimização denominada Otimização Convexa. Suas aplicações envolvem vários problemas práticos que podem ser resolvidos explorando as propriedades das funções convexas.

Para uma função real escalar é fácil observar que ela é convexa se seu gráfico possui concavidade voltada para cima em todo ponto de seu domínio. Para funções vetoriais esse conceito pode ser analisado com o uso da matriz Hessiana. Se a matriz Hessiana é semi-definida positiva em todos os pontos do domínio, então a função é convexa. No contexto da otimização, sempre será possível encontrar um minimizador para uma função convexa que satisfaça as condições de restrições viáveis. Desta forma, por enquanto vamos nos referir a apenas problemas de minimização convexa, que é, de forma geral, o objeto de estudo da Otimização Convexa.

Na próxima seção, apresentaremos observações e algumas propriedades das funções convexas.

1.1 Algumas Propriedades das Funções Convexas

Algumas das vantagens em trabalhar com otimização convexa são as seguintes: todo minimizador local de uma função convexa também será um minimizador global; o conjunto dos minimizadores é um conjunto convexo e se f for estritamente convexa, então o minimizador é único [38]. Outras propriedades das funções convexas úteis para este trabalho são apresentadas a seguir.

Proposição 1.1. *Sejam $f, g : \mathcal{C} \subset \mathbb{R}^n \to \bar{\mathbb{R}} = \mathbb{R} \cup \{\infty, -\infty\}$, funções convexas definidas no conjunto convexo $\mathcal{C}$, então $f + g$ é também uma função convexa.*

Proposição 1.2. *Seja $f : \mathcal{C} \subset \mathbb{R}^n \to \bar{\mathbb{R}}$ uma função convexa definida no convexo $\mathcal{C}$. Se para todo $\boldsymbol{\alpha} \in \mathcal{C}$ vale $f(\boldsymbol{\alpha}) \geqslant 0$, então f^2 é também uma função convexa.*

Demonstração. Primeiramente, como f é uma função convexa, então para $\lambda \in [0, 1]$ tem-se que $f(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}) \leqslant \lambda f(\boldsymbol{\alpha}) + (1-\lambda) f(\boldsymbol{\beta})$. Como $\mathcal{C}$ é convexo, então $\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta} \in \mathcal{C}$, logo $0 \leqslant f(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}) \leqslant \lambda f(\boldsymbol{\alpha}) + (1-\lambda) f(\boldsymbol{\beta})$, pois f é não negativa. Disso, tem-se

que

$$\begin{aligned}
f^2(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}) =& (f(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}))^2 \\
\leqslant& (\lambda f(\boldsymbol{\alpha}) + (1-\lambda)f(\boldsymbol{\beta}))^2 \\
=& \lambda^2 f^2(\boldsymbol{\alpha}) + (1-\lambda)^2 f^2(\boldsymbol{\beta}) \\
& + 2\lambda(1-\lambda)f(\boldsymbol{\alpha})f(\boldsymbol{\beta}) \\
=& \lambda^2 f^2(\boldsymbol{\alpha}) + (1-\lambda)(1-\lambda)f^2(\boldsymbol{\beta}) \\
& + 2\lambda(1-\lambda)f(\boldsymbol{\alpha})f(\boldsymbol{\beta}) \\
=& \lambda^2 f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& - \lambda(1-\lambda)f^2(\boldsymbol{\beta}) + 2\lambda(1-\lambda)f(\boldsymbol{\alpha})f(\boldsymbol{\beta}) \\
=& \lambda^2 f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& + \lambda(1-\lambda)f(\boldsymbol{\beta})(-f(\boldsymbol{\beta}) + 2f(\boldsymbol{\alpha})) \\
=& [\lambda f^2(\boldsymbol{\alpha}) - \lambda f^2(\boldsymbol{\alpha})] + \lambda^2 f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& + \lambda(1-\lambda)f(\boldsymbol{\beta})(-f(\boldsymbol{\beta}) + 2f(\boldsymbol{\alpha})) \\
=& \lambda f^2(\boldsymbol{\alpha}) - \lambda(1-\lambda)f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& + \lambda(1-\lambda)(-f^2(\boldsymbol{\beta}) + 2f(\boldsymbol{\alpha})f(\boldsymbol{\beta})) \\
=& \lambda f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& - \lambda(1-\lambda)(f^2(\boldsymbol{\alpha}) + f^2(\boldsymbol{\beta}) - 2f(\boldsymbol{\alpha})f(\boldsymbol{\beta})) \\
=& \lambda f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}) \\
& - \lambda(1-\lambda)(f(\boldsymbol{\alpha}) - f(\boldsymbol{\beta}))^2.
\end{aligned}$$

Como $\lambda(1-\lambda)$ é não negativo, então

$$f^2(\lambda\boldsymbol{\alpha} + (1-\lambda)\boldsymbol{\beta}) \leqslant \lambda f^2(\boldsymbol{\alpha}) + (1-\lambda)f^2(\boldsymbol{\beta}).$$

Portanto, f^2 é uma função convexa. □

Além da propriedade de convexidade, outra propriedade importante e útil para elaboração de métodos para resolução dos problemas de otimização convexa é a diferenciabilidade. A Otimização pode ser dividida em otimização diferenciável e otimização não diferenciável. A otimização diferenciável remete ao caso em que as funções que definem o problema são diferenciáveis em todos os pontos de seu domínio. Neste caso estamos tratando de otimização suave. A otimização não diferenciável é a que inclui os problemas cujas funções não possuem derivada em alguns pontos de seu domínio. Neste caso, trata-se da otimização não suave.

Os métodos clássicos de otimização convexa, tais como os métodos Método de Gradientes e Método de Newton [40], envolvem direções de descida que demandam a diferenciabilidade das funções. Com isso, pode se pensar que não seja possível utilizar

técnicas baseadas em derivadas em problemas de otimização não suave. No entanto, são bastante utilizadas essas técnicas em problemas de otimização não suave em que o conjunto de pontos de não diferenciabilidade da função tem medida nula.

Os conceitos de subgradiente e subdiferenciabilidade são amplamente utilizados em otimização não suave. Apesar de não ser possível definir o gradiente de funções não suaves em seus pontos singulares, ainda é possível definir subgradientes nestes pontos. Os subgradientes, que generalizam a noção de gradientes, são bem definidos mesmo para funções convexas não suaves.

Definição 1.3. (Subgradiente e subdiferencial) *Seja $f : \mathcal{C} \subset \mathbb{R}^n \to \bar{\mathbb{R}}$ uma função convexa. Um subgradiente de f num ponto $\boldsymbol{\alpha} \in \mathcal{C}$ é um vetor $\boldsymbol{\beta} \in \mathbb{R}^n$ tal que*

$$f(\mathbf{z}) \geqslant f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} - \boldsymbol{\alpha} \rangle, \ \forall \mathbf{z} \in \mathbb{R}^n. \tag{1.2}$$

O conjunto de todos os subgradientes de f em $\boldsymbol{\alpha}$, denotado por $\partial f(\boldsymbol{\alpha})$, é chamado subdiferencial de f em $\boldsymbol{\alpha}$. Formalmente, o subdiferencial de f em $\boldsymbol{\alpha}$ é

$$\partial f(\boldsymbol{\alpha}) := \{\boldsymbol{\beta} \in \mathbb{R}^n; f(\mathbf{z}) \geqslant f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} - \boldsymbol{\alpha} \rangle, \ \forall \mathbf{z} \in \mathbb{R}^n\}. \tag{1.3}$$

De (1.2), para qualquer $\boldsymbol{\beta} \in \partial f(\boldsymbol{\alpha})$, a função afim definida por $\ell(\mathbf{z}) := f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} - \boldsymbol{\alpha} \rangle$, para todo $\mathbf{z} \in \mathbb{R}^n$, é uma subaproximação para f em $\boldsymbol{\alpha}$. Geometricamente, ℓ determina um hiperplano tangente à função convexa f em $\boldsymbol{\alpha}$. Além disso, o gráfico de f está sempre acima do gráfico de ℓ, ou seja, acima do plano tangente. Observe que se a função f for suave, então podemos substituir o vetor $\boldsymbol{\beta}$ pelo vetor gradiente de f em $\boldsymbol{\alpha}$ e, com isso, obtemos a aproximação de Taylor de primeira de ordem para a função f em $\boldsymbol{\alpha}$. Pelas propriedades dos vetores gradientes, neste caso, o vetor $\boldsymbol{\beta}$ será único. No entanto, se a função f for não suave, então não podemos garantir a unicidade do vetor $\boldsymbol{\beta}$. Logo, é formado um conjunto de subgradientes da função f no ponto $\boldsymbol{\alpha}$. Esse conjunto é denominado subdiferencial de f em $\boldsymbol{\alpha}$.

Na Figura 1.1 é apresentada uma ilustração para exemplificar o conceito de subgradientes e subaproximações. O produto interno utilizado é o produto interno usual. A função do exemplo é dada por $f(\boldsymbol{\alpha}) = |\boldsymbol{\alpha}| + \dfrac{1}{2}$ e seu gráfico é dado pelas linhas tracejadas. O único ponto de não diferenciabilidade de f é o ponto $\boldsymbol{\alpha}_0 = 0$. O subdiferencial de f no ponto $\boldsymbol{\alpha}_0 = 0$ é dado por $\partial f(0) = [-1, 1]$, ou seja, para todo $\boldsymbol{\beta} \in [-1, 1]$ o gráfico da função afim $\ell(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}_0) + \boldsymbol{\beta}^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)$ está sempre abaixo do gráfico da função f. Para quaisquer outros valores de $\boldsymbol{\alpha}_0$ o subgradiente é único. Neste caso, o subdiferencial de f será $\partial f(\boldsymbol{\alpha}_0) = \{-1\}$ se $\boldsymbol{\alpha}_0 < 0$ e será $\partial f(\boldsymbol{\alpha}_0) = \{+1\}$ se $\boldsymbol{\alpha}_0 > 0$.

Encontrar subgradientes de uma função convexa pode não ser uma tarefa fácil. Felizmente, alguns resultados facilitam a determinação de subgradientes. Por exemplo, a desigualdade de Fenchel-Young produz uma maneira eficiente de encontrar subgradientes

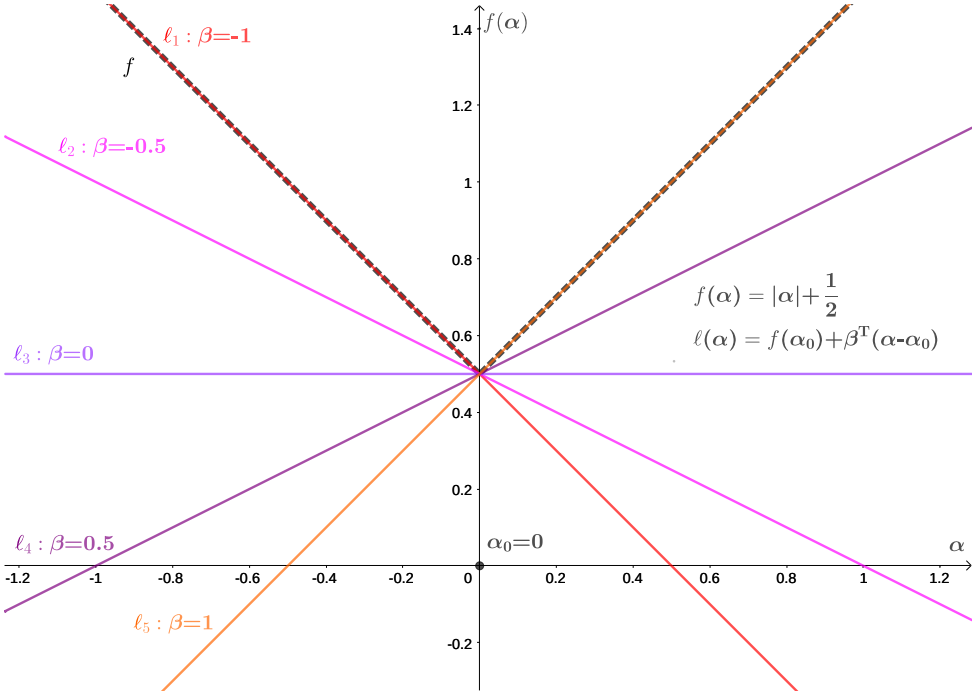


Figura 1.1 – Subaproximações da função $f(\boldsymbol{\alpha}) = |\boldsymbol{\alpha}| + \dfrac{1}{2}$ utilizando subgradientes $\boldsymbol{\beta}$ no conjunto $\{-1, -0.5, 0, 0.5, 1\}$.

de uma ampla classe de funções convexas. Para apresentar a desigualdade Fenchel-Young, precisamos definir mais alguns conceitos.

Definição 1.4. (Função própria) *Dada uma função convexa $f : \mathcal{C} \subset \mathbb{R}^n \to \bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, dizemos que f é uma função própria se f assume pelo menos um valor finito e não assume nenhum valor igual a $-\infty$.*

Em outras palavras, $f(\boldsymbol{\alpha}) < +\infty$ para algum $\boldsymbol{\alpha} \in \mathcal{C}$, isto é, $f \neq +\infty$ e $f(\boldsymbol{\alpha}) > -\infty$ para todo $\boldsymbol{\alpha} \in \mathcal{C}$. Ou ainda, tem-se que $f(\mathcal{C}) \subseteq (-\infty, +\infty]$ e $f(\boldsymbol{\alpha}) \in \mathbb{R}$, para algum $\boldsymbol{\alpha} \in \mathcal{C}$.

Definição 1.5. (Função conjugada) *Dada uma função $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, sua conjugada é a função $f^* : \mathbb{R}^n \to \bar{\mathbb{R}}$ definida por*

$$f^*(\boldsymbol{\beta}) = \sup_{\boldsymbol{\alpha} \in \mathbb{R}^n} \{\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle - f(\boldsymbol{\alpha})\}, \ \forall \boldsymbol{\beta} \in \mathbb{R}^n. \tag{1.4}$$

Definição 1.6. (Funções semicontínuas inferiormente) *Uma função convexa f é uma função semicontínua inferiormente (lsc, lower semicontinuous) se $f^{**} = f$.*

O conjunto[1] de todas as funções próprias semicontínuas inferiormente em $\mathcal{C}$ é denotado por $\Gamma_0(\mathcal{C})$. Se $\mathcal{C} = \mathbb{R}^n$, ou quando não houver dúvidas sobre o conjunto convexo que é o domínio das funções convexas, denotaremos $\Gamma_0(\mathcal{C})$ apenas por Γ_0.

[1] Em algumas obras da literatura tal conjunto é definido como um cone convexo das funções não identicamente igual a $+\infty$ e assumindo valores diferentes de $-\infty$ [41].

Proposição 1.3. (Desigualdade Fenchel-Young) *Sejam $\mathcal{C}$ um conjunto convexo e $f : \mathcal{C} \to \bar{\mathbb{R}}$ uma função própria, então*

$$f(\boldsymbol{\alpha}) + f^*(\boldsymbol{\beta}) \geqslant \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle, \tag{1.5}$$

vale para quaisquer $\boldsymbol{\beta} \in \mathbb{R}^n$ e $\boldsymbol{\alpha} \in \mathcal{C}$. Além disso, a igualdade vale se, e somente se, $\boldsymbol{\beta} \in \partial f(\boldsymbol{\alpha})$.

Demonstração. Sejam $\boldsymbol{\alpha} \in \mathcal{C}$ e $\boldsymbol{\beta} \in \mathbb{R}^n$. Como f é própria, ela assume valor finito e $-f(\mathbf{z}) \neq +\infty$, para todo $\mathbf{z} \in \mathbb{R}^n$. Logo,

$$\begin{aligned} f(\boldsymbol{\alpha}) + f^*(\boldsymbol{\beta}) &= f(\boldsymbol{\alpha}) + \sup_{\mathbf{z} \in \mathbb{R}^n} \{\langle \mathbf{z}, \boldsymbol{\beta} \rangle - f(\mathbf{z})\} \\ &\geqslant f(\boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle - f(\boldsymbol{\alpha}) \\ &= \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle, \end{aligned}$$

e, portanto, fica provada a desigualdade.

Provaremos agora a igualdade. Para isso, basta mostrar que se $\boldsymbol{\beta} \in \partial f(\boldsymbol{\alpha})$, então $f(\boldsymbol{\alpha}) + f^*(\boldsymbol{\beta}) \leqslant \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$.

Por definição, $\boldsymbol{\beta} \in \partial f(\boldsymbol{\alpha})$ se, e somente se, para todo $\mathbf{z} \in \mathbb{R}^n$ vale $f(\mathbf{z}) \geqslant f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} - \boldsymbol{\alpha} \rangle$. Mas, $f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} - \boldsymbol{\alpha} \rangle = f(\boldsymbol{\alpha}) + \langle \boldsymbol{\beta}, \mathbf{z} \rangle - \langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle$. Isso implica que

$$\langle \boldsymbol{\beta}, \mathbf{z} \rangle - f(\mathbf{z}) \leqslant \langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle - f(\boldsymbol{\alpha}), \forall \mathbf{z} \in \mathbb{R}^n.$$

Logo, $\sup_{\mathbf{z} \in \mathbb{R}^n} \{\langle \mathbf{z}, \boldsymbol{\beta} \rangle - f(\mathbf{z})\} \leqslant \langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle - f(\boldsymbol{\alpha})$. Ou seja,

$$\langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle - f(\boldsymbol{\alpha}) \geqslant \sup_{\mathbf{z} \in \mathbb{R}^n} \{\langle \mathbf{z}, \boldsymbol{\beta} \rangle - f(\mathbf{z})\} = f^*(\boldsymbol{\beta}),$$

que equivale a

$$f(\boldsymbol{\alpha}) + f^*(\boldsymbol{\beta}) \leqslant \langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle$$

Portanto, $f(\boldsymbol{\alpha}) + f^*(\boldsymbol{\beta}) = \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$, concluindo a demonstração. □

O resultado em que a desigualdade de Fenchel-Young vale como igualdade é conhecido como Teorema do Subgradiente Conjugado. Embora este teorema não necessite da condição da função ser lsc, ele é bastante útil para encontrar subgradientes de funções em Γ_0. Esta observação é importante quando estamos tratando de algoritmos de otimização baseados em subgradientes e de otimização DC. Otimização DC, a grosso modo, são problemas de otimização em que as funções são escritas como uma diferença de funções convexas.

A observação feita anteriormente ficará mais clara quando introduzirmos conceitos da otimização DC, que serão apresentados na próxima seção.

1.2 Funções Definidas como uma Diferença de Funções Convexas

Até aqui, vimos algumas propriedades interessantes a respeito de funções convexas. Tais propriedades são fundamentais na otimização convexa. Infelizmente, muitos dos problemas reais de otimização são não convexos [42, 43]. Por esse motivo, existe uma busca crescente por métodos de otimização não convexa, que é um grande desafio da programação não convexa [43]. Nas últimas três décadas, as abordagens principais que tratam de problemas de otimização global e de otimização local têm sido bastante estudadas. Mesmo que muitas funções que descrevem fenômenos reais não sejam convexas, existem métodos que usam fortemente a teoria de otimização convexa para resolver os problemas de otimização não convexos. Um exemplo são os problemas que envolvem funções que podem ser escritas como uma diferença de funções convexas. Tais problemas são denominados problemas de Otimização DC ou Programação DC.

A otimização DC visa otimizar a diferença de duas funções convexas, uma ampla classe de funções que podem não ser convexas, mas que desfrutam de propriedades interessantes e úteis [44, 45]. Conforme observado por Shen et al. [36], aplicações de otimização DC incluem processamento de sinal, aprendizado de máquina, visão computacional e estatística.

Uma importante propriedade que podemos observar sobre as funções DC é que qualquer problema de otimização contínua sobre um conjunto compacto pode ser resolvido utilizando um problema de otimização DC. Neste trabalho isso será implicitamente apresentado, pois veremos que funções contínuas definidas em um conjunto compacto podem ser aproximadas por funções contínuas lineares por partes e estas também podem ser escritas como a diferença entre duas funções convexas [34].

A seguir são apresentados alguns conceitos sobre as funções DC. Posteriormente, abordaremos as duas categorias de problemas de otimização DC considerados neste trabalho.

Definição 1.7. (Função DC) *Seja $f : \mathcal{C} \to \mathbb{R}$ uma função definida em um conjunto convexo $\mathcal{C}$. Dizemos que f é uma função DC em $\mathcal{C}$ se existem funções convexas $g, h : \mathcal{C} \to \mathbb{R}$, tais que f pode ser escrita como*

$$f(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}), \quad \forall \boldsymbol{\alpha} \in \mathcal{C}. \tag{1.6}$$

A escrita $f := g - h$ é chamada decomposição DC de f e as funções g e h são chamadas componentes DC de f.

Como existem infinitas funções convexas, e dado que a soma de funções convexas resulta numa função convexa, então um resultado importante sobre as funções DC é o seguinte.

Proposição 1.4. *Existem infinitas decomposições DC para uma única função DC.*

Demonstração. Seja $f : \mathcal{C} \to \mathbb{R}$ uma função DC e g e h suas funções componentes DC. Tome ϕ uma função convexa qualquer, tal que $\mathbf{dom}\, f \subset \mathbf{dom}\, \phi$. Então, para todo $\boldsymbol{\alpha} \in \mathbf{dom}\, f$, tem-se que $f(\boldsymbol{\alpha}) = (g-h)(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) + \phi(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}) - \phi(\boldsymbol{\alpha}) = (g+\phi)(\boldsymbol{\alpha}) - (h+\phi)(\boldsymbol{\alpha})$. Como ϕ é uma função convexa genérica e a soma de funções convexas resulta em uma função convexa, segue o resultado. □

Os resultados anteriores nos permitem provar a seguinte propriedade a respeito das funções DC.

Proposição 1.5. *Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função DC com componentes DC não negativas. A função f^2 é também uma função DC.*

Demonstração. Considere uma decomposição DC da função f com as funções componentes g e h, ou seja, $f = g - h$. Logo, para todo $\boldsymbol{\alpha} \in \mathbb{R}^n$ tem-se que

$$\begin{aligned}
f^2(\boldsymbol{\alpha}) &= [g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha})]^2 \\
&= g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha}) - 2g(\boldsymbol{\alpha})h(\boldsymbol{\alpha}) \\
&= g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha}) - 2g(\boldsymbol{\alpha})h(\boldsymbol{\alpha}) + [g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha})] - [g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha})] \\
&= 2[g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha})] - [g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha}) + 2g(\boldsymbol{\alpha})h(\boldsymbol{\alpha})] \\
&= 2[g^2(\boldsymbol{\alpha}) + h^2(\boldsymbol{\alpha})] - [g(\boldsymbol{\alpha}) + h(\boldsymbol{\alpha})]^2 \\
&= G(\boldsymbol{\alpha}) - H(\boldsymbol{\alpha}) \\
&= (G-H)(\boldsymbol{\alpha}),
\end{aligned}$$

com $G = 2(g^2 + h^2)$ e $H = (g+h)^2$. Pelas Proposições 1.1 e 1.2, como g e h são funções convexas e não negativas, então G e H são funções convexas. Logo, f^2 é uma função DC. □

Ainda é possível melhorar o resultado da Proposição 1.5 relaxando as hipóteses e combinando-o com o resultado da Proposição 1.4. De fato, seja f uma função DC com componentes DC g e h, não necessariamente não negativas. Dado que existem infinitas decomposições DC para uma única função DC, então, se for possível encontrar uma função convexa ϕ tal que $g + \phi$ e $h + \phi$ se tornem funções não negativas, então f^2 ainda será uma função DC, pois $f = g - h = (g+\phi) - (h+\phi)$, resultando na seguinte proposição.

Proposição 1.6. *Seja $f : \mathbb{R}^n \to \mathbb{R}$ um função DC com componentes conexas g e h. Se f admite decomposição com componentes DC não negativas, então, a função f^2 é também uma função DC.*

No caso da Proposição 1.6, a decomposição será dada por $f^2 = G - H$, com $G = 2(g+\phi)^2 + 2(h+\phi)^2$ e $H = (g+h+2\phi)^2$, sendo ϕ uma função convexa que faz com que a função DC f admita componentes convexas não negativas.

Neste texto, nos concentraremos nos dois seguintes problemas de otimização DC: problema de otimização DC irrestrito, formulado como:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{minimize}}\, f(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}), \tag{1.7}$$

e problema de otimização DC restrito, formulado como

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{minimize}}\, & f(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}) \\ \text{s.a.} \quad & F(\boldsymbol{\alpha}) = G(\boldsymbol{\alpha}) - H(\boldsymbol{\alpha}) \leqslant 0, \end{aligned} \tag{1.8}$$

em que $g, h : \mathbb{R}^n \to \mathbb{R}$, $G, H : \mathbb{R}^n \to \mathbb{R}^m$ são todas funções convexas. As funções G e H são convexas se suas funções coordenadas $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$ e $h_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$, respectivamente, também são convexas.

Consideraremos dois métodos para resolver os problemas de otimização DC apresentados anteriormente. O primeiro método, que resolve o problema DC irrestrito (1.7), é o algoritmo de otimização diferença de convexas (DCA, *difference of convex algorithm*). O segundo, que resolve o problema restrito (1.8), é o algoritmo denominado procedimento convexo-côncavo (CCP, *convex-concave procedure*). Em termos gerais, os métodos de otimização DC aproveitam algumas propriedades que envolvem a convexidade das componentes g e h da função objetivo f e das componentes G e H das funções de restrição F.

1.3 Algoritmo de Otimização Diferença de Convexas

Em [46], Pham Dinh Tao e El Bernoussi Souad introduziram o algoritmo diferença de funções convexas (DCA, *difference of convex functions algorithm*) estendendo o algoritmo de gradientes, usado para minimização de funções convexas, para otimização DC. Desde 1994, em um trabalho conjunto entre Le Thi Hoai An e Pham Dinh Tao, o algoritmo DCA vem sendo desenvolvido e aprimorado, tanto nos aspectos teóricos quanto computacionais.

Em termos gerais, o DCA é um método de otimização de subgradiente usado para resolver problemas de otimização DC irrestritos. Além disso, é baseado na otimização local e na otimização DC dual, que em alguns casos é mais simples de resolver do que otimização DC primal. A seguir, veremos que ao tratar de componentes DC em Γ_0, é possível mostrar uma perfeita simetria existente entre os problemas de otimização DC primal e dual [41, 46]. De modo a apresentar tal simetria, inclui-se na definição de uma função DC f que as componentes g e h consideradas também sejam funções próprias e

semicontínuas inferiormente, ou seja, que $g, h \in \Gamma_0$. Os problemas de otimização DC primal e dual são apresentados a seguir.

Considere um problema DC irrestrito dado por (1.7), que pode ser alternativamente escrito como:

$$\inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{f(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}); g, h \in \Gamma_0\}. \tag{1.9}$$

Um problema DC, restrito a um subconjunto convexo $\mathcal{C} \subset \mathbb{R}^n$, definido por $\inf_{\boldsymbol{\alpha}\in\mathcal{C}}\{f(\boldsymbol{\alpha}); g, h \in \Gamma_0\}$ pode ser transformado num problema no formato do Problema (1.9) [43]. Neste caso, basta incluir a função indicadora em $\mathcal{C}$ na componente DC g. A função indicadora $\mathcal{X}_\mathcal{C}$ em um conjunto $\mathcal{C}$ é definida como (1.10).

$$\mathcal{X}_\mathcal{C}(\boldsymbol{\alpha}) = \begin{cases} 0, & \text{, se } \boldsymbol{\alpha} \in \mathcal{C}, \\ +\infty, & cc. \end{cases} \tag{1.10}$$

O problema DC restrito é então convertido no problema DC irrestrito dado por $\inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{f(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}) + \mathcal{X}_\mathcal{C}(\boldsymbol{\alpha}); g, h \in \Gamma_0\}$.

O Problema (1.9) é a forma padrão de um problema de otimização DC primal. A forma padrão do problema de otimização DC dual para (1.9) é dada pelo seguinte problema:

$$\inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{f^*(\boldsymbol{\beta}) = h^*(\boldsymbol{\beta}) - g^*(\boldsymbol{\beta})\}. \tag{1.11}$$

Considere um problema de otimização e suponha que a solução para este problema seja conhecida. Se a solução para um outro pode ser obtida trivialmente a partir da solução do primeiro problema, então dizemos que estes dois problemas são equivalentes. Sejam λ e λ^* os valores ótimos dos problemas (1.9) e (1.11), respectivamente. Neste caso, podemos dizer que os problemas (1.9) e (1.11) são equivalentes, uma vez que $\lambda = \lambda^*$. Formalmente, temos o seguinte resultado:

Proposição 1.7. *Resolver o problema* (1.9) *é equivalente a resolver o problema* (1.11).

Demonstração. Como $h \in \Gamma_0$, então $h^{**} = h$. Daí, pela definição de função conjugada,

$h(\boldsymbol{\alpha}) = h^{**}(\boldsymbol{\alpha}) = \sup_{\boldsymbol{\beta}\in\mathbb{R}^n}\{\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle - h^*(\boldsymbol{\beta})\},\ \forall\boldsymbol{\alpha}\in\mathbb{R}^n$. Logo,

$$\begin{aligned}
\lambda &= \inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha})\}\\
&= \inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{g(\boldsymbol{\alpha}) - \sup_{\boldsymbol{\beta}\in\mathbb{R}^n}\{\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle - h^*(\boldsymbol{\beta})\}\\
&= \inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{g(\boldsymbol{\alpha}) + \inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{-\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle + h^*(\boldsymbol{\beta})\}\}\\
&= \inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{g(\boldsymbol{\alpha}) - \langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle + h^*(\boldsymbol{\beta})\}\\
&= \inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{h^*(\boldsymbol{\beta}) + g(\boldsymbol{\alpha}) - \langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle\}\\
&= \inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{h^*(\boldsymbol{\beta}) + \inf_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{g(\boldsymbol{\alpha}) - \langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle\}\}\\
&= \inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{h^*(\boldsymbol{\beta}) - \sup_{\boldsymbol{\alpha}\in\mathbb{R}^n}\{\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle - g(\boldsymbol{\alpha})\}\}\\
&= \inf_{\boldsymbol{\beta}\in\mathbb{R}^n}\{h^*(\boldsymbol{\beta}) - g^*(\boldsymbol{\beta})\}\\
&= \lambda^*
\end{aligned}$$

□

Este resultado é também chamado perfeita simetria entre os problemas DC primal e dual. As condições de otimalidade necessárias para a solução de um problema DC primal são dadas em (1.12), em que $\boldsymbol{\alpha}^*$ é denominado[2] ponto crítico de $f = g - h$. Tais condições vêm da simetria existente entre os problemas DC primal e dual [41].

$$\partial h(\boldsymbol{\alpha}^*) \cap \partial g(\boldsymbol{\alpha}^*) \neq \varnothing \quad \text{e} \quad \partial g(\boldsymbol{\alpha}^*) \supset \partial h(\boldsymbol{\alpha}^*) \tag{1.12}$$

A fim de introduzir o algoritmo DCA, considere $\{\boldsymbol{\alpha}_t\}$ uma sequência convergente, sendo $\boldsymbol{\alpha}^*$ o seu limite, ou seja, $\boldsymbol{\alpha}_t \to \boldsymbol{\alpha}^*$. Como h é uma função convexa, então para cada $\boldsymbol{\alpha}_t$ tem-se que $\partial h(\boldsymbol{\alpha}_t) \neq \varnothing$. Logo, para cada elemento $\boldsymbol{\alpha}_t$ existe um elemento $\boldsymbol{\beta}_t \in \partial h(\boldsymbol{\alpha}_t)$. A sequência $\{\boldsymbol{\beta}_t\}$ é uma sequência limitada e seus pontos de acumulação pertencem a $\partial h(\boldsymbol{\alpha}^*)$ (Ver Proposição 2.5 de [47]). Isto é um resultado importante para a formulação do método DCA descrito pelo Algoritmo 1.

O algoritmo DCA possui uma propriedade chamada aproximação convexa sucessiva (SCA, *successive convex approximation*), pois ele consiste em uma aproximação iterativa de um problema DC por uma sequência de problemas convexos que serão resolvidos por algoritmos de otimização convexa apropriados. Em particular, DCA é uma abordagem global para programação convexa, ou seja, ela converge para pontos ótimos de problemas convexos reformulados como problemas DC. Consequentemente, o algoritmo DCA pode ser usado para construir algoritmos customizados eficientes para resolver problemas convexos gerados por DCA.

[2] Estas condições também são chamadas de **condições de KKT generalizada** para o problema primal [41].

Algoritmo 1: DCA

Entrada: Funções convexas: $g, h \in \Gamma_0$
Saída: $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ (soluções primal e dual)
Inicializar: $\boldsymbol{\alpha}_0 \in \mathbb{R}^n, t = 0$
repita
  Calcule $\boldsymbol{\beta}_t \in \partial h(\boldsymbol{\alpha}_t)$
  Calcule $\boldsymbol{\alpha}_{t+1} \in \partial g^*(\boldsymbol{\beta}_t)$
  $t = t + 1$
até *convergir*;
retorna $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}_t$ *e* $\boldsymbol{\beta}^* = \boldsymbol{\beta}_{t-1}$

Ambos problemas DC, tanto o primal quanto o dual, podem ser resolvidos utilizando o algoritmo DCA descrito no Algoritmo 1. Em termos gerais, o Algoritmo 1 busca duas sequências convergentes $\{\boldsymbol{\alpha}_t\}$ e $\{\boldsymbol{\beta}_t\}$, de modo que seus pontos de acumulação sejam aproximações para as soluções locais dos problemas primal e dual, respectivamente. Em outras palavras, o algoritmo produz duas sequências $\{\boldsymbol{\alpha}_t\}$ e $\{\boldsymbol{\beta}_t\}$ tais que $\{f_k = f(\boldsymbol{\alpha}_t)\}$ e $\{f_k^* = f^*(\boldsymbol{\beta}_t)\}$ sejam sequências decrescentes.

Um dos desafios do Algoritmo 1 é encontrar os subgradientes $\boldsymbol{\beta}_t \in \partial h(\boldsymbol{\alpha}_t)$ e $\boldsymbol{\alpha}_{t+1} \in \partial g^*(\boldsymbol{\beta}_t)$ em cada iteração. Visto que resolver o problema primal é equivalente a resolver o problema dual, a escolha de $\boldsymbol{\beta}_t$ será arbitrária e a escolha de $\boldsymbol{\alpha}_{t+1}$ será dada pela resolução de um problema de otimização que usa as propriedades de funções conjugadas.

Se $g \in \Gamma_0(\mathcal{C})$ então $g^{**} = g$, com $\mathcal{C} \subset \mathbb{R}^n$. Do teorema do subgradiente conjugado, as seguintes equivalências valem para qualquer $t \in \mathbb{N}$:

$$\begin{aligned}\boldsymbol{\alpha}_{t+1} \in \partial g^*(\boldsymbol{\beta}_t) &\Leftrightarrow g^*(\boldsymbol{\beta}_t) + g(\boldsymbol{\alpha}_{t+1}) = \langle \boldsymbol{\alpha}_{t+1}, \boldsymbol{\beta}_t \rangle \\ &\Leftrightarrow \boldsymbol{\beta}_t \in \partial g(\boldsymbol{\alpha}_{t+1}) \\ &\Leftrightarrow g(\boldsymbol{\alpha}_{t+1}) - \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha}_{t+1} \rangle \leqslant g(\boldsymbol{\alpha}) - \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} \rangle, \forall \boldsymbol{\alpha} \in \mathcal{C}.\end{aligned}$$

Além disso, as seguintes equivalências são válidas:

$$\begin{aligned}\underset{\boldsymbol{\alpha}\in\mathcal{C}}{\text{minimize}}\{g(\boldsymbol{\alpha}) - \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} \rangle\} &\equiv \underset{\boldsymbol{\alpha}\in\mathcal{C}}{\text{minimize}}\{g(\boldsymbol{\alpha}) - \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} \rangle - h(\boldsymbol{\alpha}_t) + \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha}_t \rangle\} \\ &\equiv \underset{\boldsymbol{\alpha}\in\mathcal{C}}{\text{minimize}}\{g(\boldsymbol{\alpha}) - [h(\boldsymbol{\alpha}_t) + \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle]\}.\end{aligned}$$

Portanto, a sequência $\{\boldsymbol{\alpha}_t\}$ pode ser encontrada resolvendo, em cada iteração t, o problema de otimização convexa dado por:

$$\boldsymbol{\alpha}_{t+1} = \underset{\boldsymbol{\alpha}\in\mathcal{C}}{\arg\min}\{g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}_t) + \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} - \boldsymbol{\alpha}_t \rangle\} \tag{1.13}$$

Esse problema pode ser entendido geometricamente como uma aproximação por funções afins dos termos convexos h da função objetivo. Com isso, o subproblema resultante é um problema de otimização convexa, pois a função objetivo torna-se uma

soma entre uma função convexa e uma função afim e, portanto, uma função convexa. Embora tenhamos falado apenas de problemas irrestritos, essa abordagem pode ser também utilizada em problemas DC com restrições. Neste caso, a aproximação por funções afins satisfaz as mesmas propriedades aplicadas à função objetivo do problema irrestrito.

O limite da sequência $\{\boldsymbol{\alpha}_t\}$ obtida pela resolução de problemas do tipo (1.13) é uma aproximação para a solução do problema primal. O DCA converge com um ponto inicial arbitrário, pois é um método de descida sem busca de direção, com convergência para um ponto crítico de f. A análise de convergência e outras propriedades do Algoritmo 1 podem ser encontradas em [46, 48].

Existem algumas características-chave a respeito do algoritmo DCA que são interessantes. Uma delas é que a construção do DCA é baseada nas funções componentes DC g e h de f, mas não é necessária a própria função f. Visto que uma função DC f possui infinitas decomposições DC, isso tem implicações cruciais para a qualidade das soluções do DCA, por exemplo, velocidade de convergência, robustez, eficiência, globalidade das soluções computadas. Desta forma, é muito importante estudar várias decomposições DC equivalentes de um problema DC.

Na Figura 1.2 é apresentado um exemplo da aplicação do método DCA para minimização de uma função DC $f : \mathbb{R} \to \mathbb{R}$ dada por $f(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^4 - 2\boldsymbol{\alpha}^2$, tomando funções componentes dadas por $g(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^4 + 2$ e $h(\boldsymbol{\alpha}) = 2\boldsymbol{\alpha}^2 + 2$. A função $-\bar{h}$ é uma subaproximação de $-h$ e $\bar{f}$ é a aproximação convexa de f no ponto $\boldsymbol{\alpha}_t$. O minimizador de f na iteração t é $\boldsymbol{\alpha}_{t+1}$ e será o elemento utilizado na aproximação convexa de f na próxima iteração. No exemplo apresentado nos gráficos da Figura 1.2, o método converge para $\boldsymbol{\alpha}^* = -1$ se o ponto inicial for $\boldsymbol{\alpha}_0 < 0$. O método irá convergir para $\boldsymbol{\alpha}^* = 1$ se o ponto inicial for $\boldsymbol{\alpha}_0 > 0$. Com exceção do ponto $\boldsymbol{\alpha}_0 = 0$, o método convergirá para um mínimo local de f.

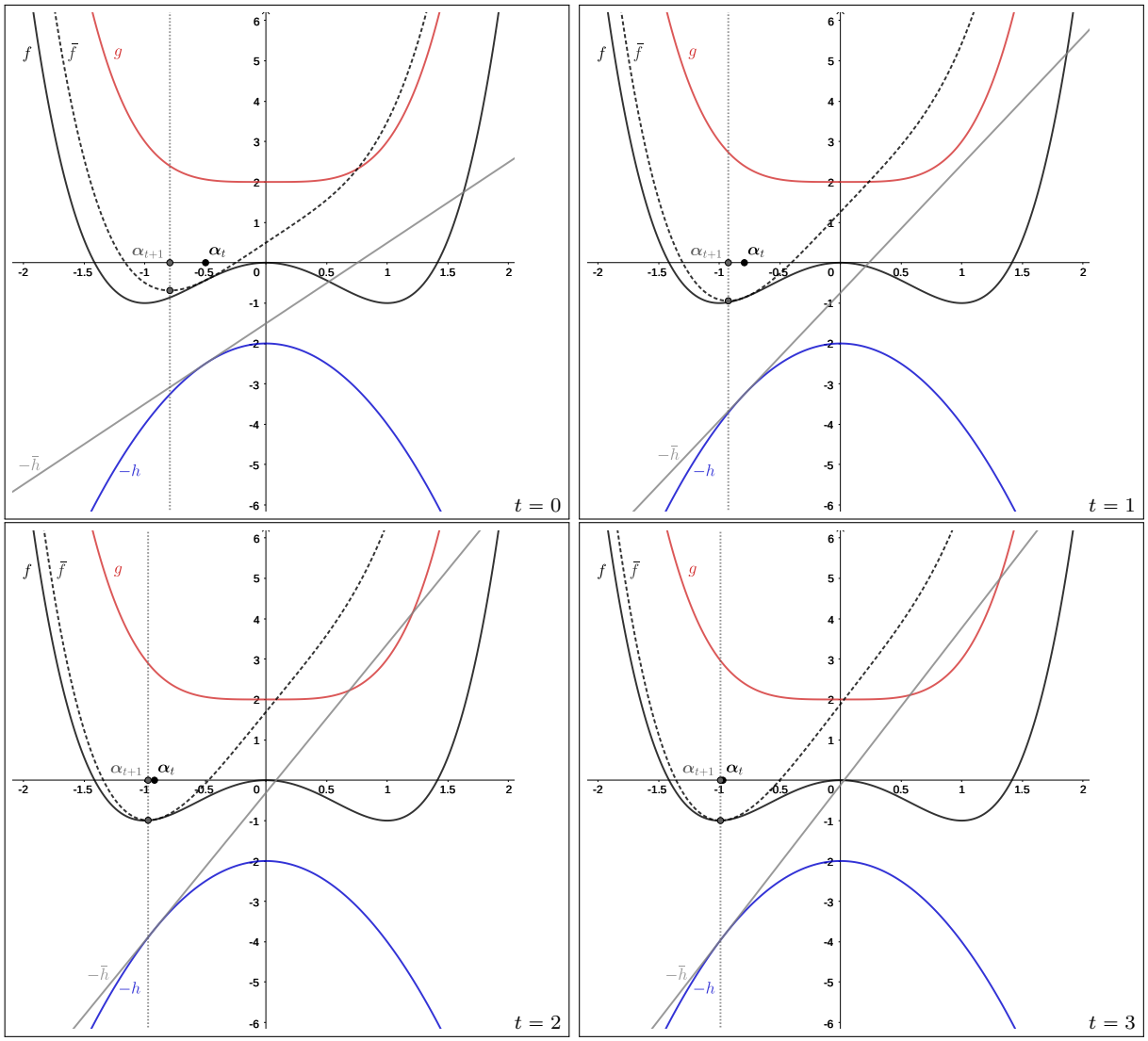


Figura 1.2 – Ilustração, para a função $f(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^4 - 2\boldsymbol{\alpha}^2$, do método DCA convergindo em 4 iterações: $t = 0, 1, 2, 3$.

As aproximações convexas das funções DC podem levar a soluções dos subproblemas convexos (1.13) que não são mínimos locais. Um exemplo concreto disso é quando o termo côncavo $-h$ for diferenciável e o ponto inicial for um maximizador local da função f e também um minimizador de h. Neste ponto inicial a função afim que tangencia h tem inclinação nula. Com isso, o subproblema convexo estaciona em um mínimo local que, neste caso, seria exatamente um maximizador de f. Isso ocorre pois a aproximação convexa de f neste ponto assume o valor mínimo.

Na Figura 1.3 é apresentada uma ilustração das observações feitas no parágrafo anterior utilizando a função apresentada na Figura 1.2. Como ambas componentes convexas são diferenciáveis, então o subdiferencial de h em qualquer ponto $\boldsymbol{\alpha}_0$ é dado pelo gradiente $\nabla h(\boldsymbol{\alpha}_0) = 4\boldsymbol{\alpha}_0$. Logo, a aproximação convexa de f denotada por $\bar{f}$ e representada com linha tracejada na Figura 1.2, é dada por $\bar{f}(\boldsymbol{\alpha}) = g(\boldsymbol{\alpha}) - h(\boldsymbol{\alpha}_0) - \nabla h(\boldsymbol{\alpha}_0)(\boldsymbol{\alpha} - \boldsymbol{\alpha}_0)$. Tomando $\boldsymbol{\alpha}_0 = 0$, o mínimo de $\bar{f}$ ocorre exatamente no ponto inicial $\boldsymbol{\alpha}_0$, que não é um minimizador

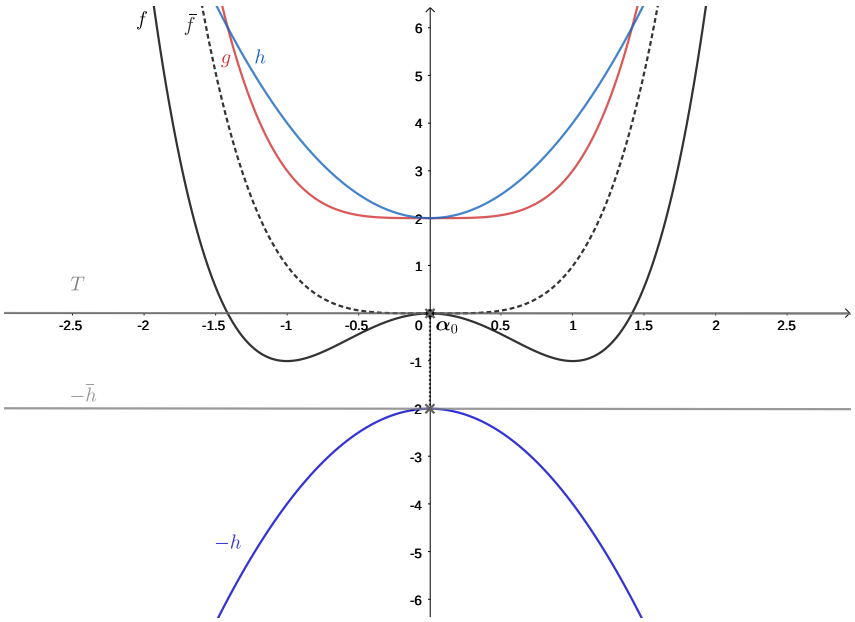


Figura 1.3 – Aproximação convexa de uma função DC num ponto inicial estacionário.

local de f. Portanto, o método irá estacionar neste ponto, que é maximizador local de f.

Com isso, é fácil perceber que, mesmo com um ponto inicial factível, é possível que o método convirja para um ponto que não seja um minimizador local. Desta forma, fica claro que a solução final depende, de fato, do ponto de inicialização.

1.4 Algoritmo de Otimização Convexa-Côncava

A metodologia de majoração-minimização (MM) é um procedimento que visa resolver problemas difíceis de minimização através de problemas mais fáceis usando uma sequência de superaproximações das funções que definem o problema. Esta etapa é denominada majoração. A grosso modo, o algoritmo denominado procedimento côncavo-convexo (CCP, *concave-convex procedure*) [49] é uma generalização de algoritmos de majoração-minimização que utiliza aproximações afins visando aproximar funções DC por funções convexas.

Em sua primeira formulação, o CCP foi usado para construir um sistema dinâmico iterativo de tempo discreto, em que é garantida a convergência em otimização global de problemas com funções cuja Hessiana é limitada. Quando isso ocorre é sempre possível decompor as funções em uma soma de uma função convexa e uma função côncava (Teorema 1, [49]). Esse procedimento se mostrou muito útil em otimização local, no sentido de que muitos problemas de otimização podem ser reformulados e resolvidos utilizando as técnicas de resolução do CCP [49].

Em problemas de otimização, uma condição necessária para utilizar o CCP é a necessidade de as funções poderem ser reescritas como uma soma de uma função convexa

e uma função côncava. Essa soma pode ser vista como uma diferença de funções convexas. Tais informações são úteis na teoria de otimização DC, pois os métodos CCP podem resolver problemas de otimização não convexa sempre que o objetivo e as restrições forem funções DC. A heurística CCP pode ser vista como uma metodologia que usa ferramentas de otimização convexa, que encontra um ótimo local de problemas de otimização DC por meio de uma sequência de subproblemas convexos [39].

De modo a realizar uma interpretação geométrica deste método, considere o problema DC restrito dado por (1.8). A cada iteração do método, é resolvido um subproblema convexo construído da seguinte forma.

Podemos ver a diferença entre as funções convexas g e h como uma soma entre uma função convexa g e uma função côncava $-h$. O mesmo pode ser feito com a soma entre as funções G e H. Os termos côncavos, $-h$ no objetivo e $-H$ nas restrições, são aproximados por funções majorantes afins. Tanto a função objetivo quanto a função da restrição tornam-se uma soma entre uma função convexa e uma afim, resultando numa função convexa. Com isso, o subproblema resultante é um problema de otimização convexa. Logo, o problema resultante pode ser resolvido usando técnicas clássicas de otimização convexa.

O procedimento apresentado acima é repetido até que um critério de parada seja satisfeito. A solução do subproblema em cada iteração é utilizada na iteração seguinte e será útil na linearização dos termos côncavos das funções DC utilizando propriedades dos subgradientes.

As condições gerais originais sob as quais o método CCP pode ser aplicado e a prova de sua convergência podem ser encontradas em [49].

Formalmente, o algoritmo que descreve o método CCP para um problema de otimização DC restrito dado por 1.8 é apresentado no Algoritmo 2. Aqui, as funções $G, H : \mathbb{R}^n \to \mathbb{R}^m$ com valor vetorial são escritas da seguinte forma

$$G(\boldsymbol{\alpha}) = (g_1(\boldsymbol{\alpha}), ..., g_m(\boldsymbol{\alpha})) \quad \text{e} \quad H(\boldsymbol{\alpha}) = (h_1(\boldsymbol{\alpha}), ..., h_m(\boldsymbol{\alpha})).$$

A função objetivo é identificadas por $f \equiv f_0 := g_0 - h_0$, com, $g_0 \equiv g$ e $h_0 \equiv h$

O método CCP é considerado para problemas de otimização DC restrita, ou seja, além da aproximação convexa feita na função objetivo, também são feitas aproximações convexas no conjunto de pontos viáveis. No CCP, o majorante convexo que aproxima dos termos côncavos são funções afins. Isto é, as funções h_i são subaproximadas por funções afins.

Pelo Algoritmo 2, que descreve o CCP, é possível ver que os termos côncavos das funções DC são linearizados explicitamente usando subgradientes. No caso em que as funções são diferenciáveis, podemos considerar apenas os gradientes das funções.

Algoritmo 2: CCP

Entrada: Funções Convexas: $g_0, \ldots, g_m$ e $h_0, \ldots, h_m$. Taxa ϵ
Saída: $\boldsymbol{\alpha}^*$ (Solução)
Inicializar: $\boldsymbol{\alpha}_0 \in \mathbb{R}^n$ e $t = 1$
repita
  Calcule $\boldsymbol{\beta}_i \in \partial h_i(\boldsymbol{\alpha}_{t-1})$, $\forall i = 0, \ldots, m$.
  Calcule $\boldsymbol{\alpha}_t$ solução de (1.14)
  $t = t + 1$
até *convergir conforme* (1.15);
retorna $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}_t$

A solução $\boldsymbol{\alpha}$ do problema DC restrito dado por (1.8) é encontrada resolvendo um problema de otimização convexa restrita, utilizando a técnica de majoração dos termos côncavos das funções DC. Este método constrói uma sequência $\{\boldsymbol{\alpha}_t\}$ que converge para um ponto crítico do problema primal. A cada iteração t, $\{\boldsymbol{\alpha}_t\}$ é solução do problema de otimização convexa dado por (1.14), em que $\boldsymbol{\beta}_i$ são os subgradientes em $\boldsymbol{\alpha}_{t-1}$ (solução da iteração anterior) dos termos côncavos da função objetivo e das restrições

$$\begin{aligned} \underset{(\boldsymbol{\alpha},\mathbf{s})\in\mathbb{R}^{n+m}}{\text{minimize}} \quad & g_0(\boldsymbol{\alpha}) - h_0(\boldsymbol{\alpha}_{t-1}) - \langle \boldsymbol{\beta}_0, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1} \rangle \\ \text{s.a.} \quad & g_i(\boldsymbol{\alpha}) \leqslant h_i(\boldsymbol{\alpha}_{t-1}) + \langle \boldsymbol{\beta}_i, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1} \rangle, \quad \forall i = 1, \ldots, m. \end{aligned} \tag{1.14}$$

A convergência do método é atingida quando a melhoria de uma iteração para outra for suficientemente pequena ou quando for atingido um número máximo de iterações. O critério de parada assumido em [50], e que também assumiremos neste trabalho, é dado por (1.15), sendo ϵ a taxa usada para avaliar a melhoria de uma iteração para outra.

$$|f_0(\boldsymbol{\alpha}_{t-1}) - f_0(\boldsymbol{\alpha}_t)| \leqslant \epsilon \tag{1.15}$$

Conforme a abordagem apresentada em [41], os subproblemas do método CCP podem ser derivados do DCA usando a ideia principal de linearizar a parte côncava das decomposições DC usando as propriedades de subgradientes. A cada iteração do CCP, é resolvido um problema DC restrito. Neste caso, o CCP pode ser visto como um caso especial do algoritmo DCA aplicado a problemas de otimização restrita. Diferente do DCA, abordagem apresentada em [50], na abordagem utilizando o CCP existe apenas uma visão puramente primal, no sentido de não recorrer a funções conjugadas e, consequentemente, não referenciar o problema DC dual.

1.5 Programação Convexa e Côncava-Convexa Disciplinada

A programação convexa disciplinada DCP é uma técnica para construir e resolver problemas de otimização convexa de forma automatizada, em que é imposto

um sistema de regras pré-estabelecidas para a construção de expressões matemáticas com curvaturas conhecidas. Essa metodologia é amplamente utilizada por bibliotecas de otimização convexa como `CVX`, `CVXPY` e `Convex.jl` [51].

A metodologia DCP não constrói simplesmente restrições e funções objetivo. Nesta metodologia são analisadas a convexidade de funções e/ou restrições já existentes. É realizada uma verificação se as funções envolvidas satisfazem um conjunto de regras e de propriedades de convexidade. Tais regras e propriedades, que já são conhecidas, são combinadas e manipuladas de tal forma que a análise convexa garanta resultados a partir de seus princípios básicos.

O conjunto de regras estabelecidas pelo DCP pode ser dividido nas quatro categorias a seguir, que podem ser encontradas em detalhe em [39]:

- *regras de nível superior* - estabelece a estrutura dos tipos de problemas, das restrições e das expressões e afirmações constantes.
- *regras de livre produto* - regem as somas e produtos entre expressões convexas, côncavas e afins.
- *regras de sinal* - regem as condições suficientes para estabelecer se as expressões são ou não convexas, côncavas ou afins.
- *regras de composição* - garantias sobre composições de aplicações lineares e não lineares.

Em termos matemáticos, o DCP lida com problemas de otimização do seguinte tipo

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{minimize/maximize}} \quad & f(\boldsymbol{\alpha}) \\ \text{s.a.} \quad & g_i(\boldsymbol{\alpha}) \simeq h_i(\boldsymbol{\alpha}), \quad \forall i = 1, \ldots, m, \end{aligned} \tag{1.16}$$

sendo conhecidas as curvaturas das funções $f, g_1, ..., g_m, h_1, ..., h_m$ e $\simeq$ uma expressão de desigualdade ou igualdade. A referência [39] fornece mais detalhes sobre programação convexa disciplinada e as condições que devem ser satisfeitas a respeito das regras apresentadas anteriormente.

Combinar o CCP com as ideias da metodologia disciplinada do DCP resultou em um procedimento denominado programação côncava-convexa disciplinada (DCCP, *disciplined concave-convex programming*). Esta combinação permitiu que o DCP lidasse com problemas de otimização DC.

Finalmente, um problema DCCP é semelhante a (1.16), mas com algumas versões relaxadas das regras de nível superior, o que permite generalizar o uso do DCP para problemas de otimização DC com o uso do CCP.

Abaixo, apresentaremos algumas das regras de nível superior relaxadas para formar os problemas DCCP.

1. Se (1.16) for um problema de minimização, então f deve ser convexo.
2. Se (1.16) for um problema de maximização, então f deve ser côncavo.
3. Se $\simeq$ for igual a $\leqslant$, então g_i deve ser convexo e h_i deve ser côncavo para todos $i = 1, \ldots, m$.
4. Se $\simeq$ for igual a $\geqslant$, então g_i deve ser côncavo e h_i deve ser convexo para todos $i = 1, \ldots, m$.
5. Se $\simeq$ for igual a $=$, então g_i e h_i devem ser afins para todos $i = 1, \ldots, m$.

As regras acima são relaxadas, de modo a generalizar o uso de método de otimização DCP para problemas de otimização DC. Por exemplo, diferentemente do DCP, em problemas de otimização DCCP, podemos minimizar uma função côncava, sujeita a restrições de igualdade não-convexas e restrições de desigualdade não-convexas entre expressões convexas e côncavas. O relaxamento dessas condições nos permite resolver problemas não convexos.

Como no DCP, as curvaturas da função objetivo e as funções de restrições são previamente conhecidas em um problema DCCP. Conhecendo as curvaturas, o DCCP pode lidar com problemas não convexos, desde que as funções sejam descritas como funções DC de maneira disciplinada. Por fim, o Algoritmo 2 pode ser usado para resolver o problema DC disciplinadamente.

O procedimento CCP pode resolver problemas que envolvem modelos de aprendizado de máquinas cuja função de decisão é uma função DC. No próximo capítulo apresentaremos um modelo de aprendizado de máquinas cujo treinamento pode ser feito utilizando a otimização DC.

2 Conceitos Básicos em Aprendizado de Máquinas

2.1 Introdução ao Aprendizado de Máquinas

Inspirada em habilidades do sistema nervoso biológico, tais como aprender e raciocinar, a inteligência artificial (AI, *artificial intelligence*) é um campo da Ciência cujo objetivo é criar máquinas que aprendam a realizar tarefas complexas de forma automatizada, isto é, aprender sem serem explicitamente programadas [52]. Aprendizado de Máquinas (ML, *machine learning*) é o campo de estudo da Inteligência Artificial e Ciência de Dados em que as máquinas conseguem aprender. As técnicas de ML são métodos criados para que sistemas computacionais possam aprender com os dados, reconhecer padrões e tomar decisões com o mínimo de intervenção humana possível. Ou seja, ML nada mais é do que a maneira com que os sistemas aprendem e adquirem a forma de inteligência.

O processo de aprendizagem deve possuir um objetivo bem definido. Utilizando dados disponíveis, o aprendizado deve seguir uma sequência de ações lógicas de modo a alcançar tal objetivo. Esse objetivo pode ser alcançado por meio de implementações de *softwares* que possam aprender de forma autônoma a partir de métodos de aprendizagem. Em um contexto mais atualizado, um *software* aprende com a experiência $\mathcal{E}$, em relação a alguma tarefa $\mathcal{T}$ e alguma medida de desempenho, se seu desempenho em $\mathcal{T}$ melhora com a experiência $\mathcal{E}$ [53, 54].

Os métodos de ML se dividem em três categorias principais: aprendizado supervisionado, não supervisionado e aprendizado por esforço (semi-supervisionado) [55]. Neste trabalho focaremos no aprendizado supervisionado. A grosso modo, o aprendizado supervisionado utiliza dados de treinamento que possuem um elemento preditivo. Com isso, é possível encontrar um modelo que mapeia um elemento de entrada em um elemento de saída, que possivelmente será igual ao elemento preditivo. Neste caso, estamos no contexto de modelos preditivos. Estes modelos são divididos em duas classes: modelos de regressão e modelos de classificação.

Na literatura, é dito que os dados de entrada pertencem a um espaço vetorial denominado espaço de características (*feature space*) e são geralmente chamados de variáveis independentes, variáveis explicativas ou ainda variáveis preditoras. Os dados de saída são os elementos preditivos, denominados variáveis dependentes. Tais dados dependem das variáveis preditoras.

2.1.1 Modelos Preditivos de Regressão e Classificação

Os modelos que serão apresentados neste trabalho pertencem à categoria de aprendizado supervisionado. Como modelos preditivos, esses modelos são geralmente apresentados como um problema matemático em que se busca encontrar uma função de previsão, que associa o conjunto das variáveis independentes a um conjunto de variáveis dependentes. Em termos matemáticos, se tomarmos X sendo o conjunto das variáveis independentes e Y o conjunto das variáveis dependentes, no contexto de aprendizado supervisionado, busca-se o seguinte: encontrar parâmetros que definem uma função pré determinada g que satisfaça $y_i \approx g(\mathbf{x}_i)$, em que $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1, ..., m\} \subseteq X \times Y$ é o conjunto dos dados de amostras disponível para o processo de aprendizagem. Esse processo de aprendizagem é comumente chamado treinamento.

Dizemos que a função g que define um modelo é construída com a ajuda de um professor. Isso vem do fato de já termos uma relação pré-estabelecida entre os dados de entrada e saída, que é utilizada para ensinar o modelo para predições de dados de entrada que ainda não foram apresentados ao modelo. As duas classes de modelos preditivos bastante conhecidas que utilizam técnicas de aprendizado de máquina, e que vamos utilizar neste texto, são os modelos do tipo preditivos denominados Regressão e Classificação. Em tarefas de regressão é feita a previsão de uma quantidade. Em tarefas de classificação é feita a previsão de uma qualidade, também conhecida como rótulo. Neste caso, dizemos que os dados são rotulados.

Modelos preditivos do tipo Regressão são usados para resolver problemas de ajuste de curvas cuja metodologia é encontrar uma função que melhor se adapte a uma curva específica em relação a um conjunto de dados. Este ajuste pode ser útil em previsões ou estimativas fora do conjunto de dados.

Modelos preditivos do tipo Classificação são usados para categorizar informações com base em um conjunto de dados históricos e podem, por exemplo, ser úteis no fornecimento de análises para responder perguntas na tomada de decisão. Quando os dados do problema de classificação pertencem a apenas duas classes, ou seja, quando o rótulo pode assumir apenas dois valores, dizemos que o problema se trata de uma classificação binária. Neste texto nos limitaremos a problemas de classificação binária. No entanto, as técnicas apresentadas podem ser estendidas a problemas de classificação multi-classe, utilizando outras técnicas de ML como, por exemplo, a técnica um-contra-todos [56]. A técnica um-contra-todos consiste em fazer diversos treinamentos utilizando técnicas de classificação binária sendo que, em cada treinamento, o conjunto de treinamento é dividido em dois conjuntos: o primeiro conjunto é composto pelos dados de uma classe e o segundo conjunto é composto pelos dados das demais classes.

Embora os dados normalmente sejam numéricos, em princípio, eles podem

ser qualitativos. No entanto, em alguns casos, eles podem ser transformados em valores numéricos usando técnicas de mineração de dados [57]. Resumidamente, modelos de regressão são considerados no contexto em que as variáveis dependentes Y são contínuas, e os modelos de classificação no contexto em que as variáveis dependentes Y são discretas, isto é, quando Y é um conjunto finito.

2.1.2 Medidas para Avaliação de Modelos Preditivos

De modo a apresentar a formulação de algumas medidas de avaliação, considere um conjunto de amostras $\mathcal{T} = \{(\mathbf{x}_i, y_i), i = 1, ..., m\} \subseteq \mathcal{X} \times \mathcal{Y}$. Tome $X = \{\mathbf{x}_j\}_{j=1}^m$, $\mathbf{y} = \{y_j\}_{j=1}^m$ e $g : \mathcal{X} \to \mathcal{Y}$ a função que descreve um modelo preditivo. O valor y_j^p predito através do modelo preditivo g, para um elemento $\mathbf{x}_j$, é dado por $y_j^p = g(\mathbf{x}_j)$ e, neste caso, tomamos $\mathbf{y}^p = g(X)$.

Medidas para Modelos de Regressão

Para problemas de regressão, existem muitas medidas de avaliação. Dentre os modelos avaliados, o melhor modelo será aquele que conseguir melhor descrever o comportamento dos dados de uma amostra de forma semelhante ao comportamento real dos dados.

Uma das medidas mais básicas é o erro médio absoluto (MAE, *Mean Absolute Error*), que é dado pela média do erro que cada elemento da amostra tem em relação à curva de regressão. Tal erro é calculado pelo valor absoluto da diferença entre o valor real e o valor predito. A formulação do MAE é dada pela equação (2.1).

$$MAE(\mathcal{T}, g) = \frac{1}{m} \sum_{j=1}^{m} |y_j - y_j^p|. \tag{2.1}$$

Quanto menor o valor obtido pelo MAE, mais bem ajustado foi o regressor.

Outra medida bastante popular é a medida definida pelo erro quadrático médio (MSE, *Mean Squared Error*). A formulação do MSE é dada pela equação (2.2).

$$MSE(\mathcal{T}, g) = \frac{1}{m} \sum_{j=1}^{m} (y_j^p - y_j)^2 \tag{2.2}$$

O MAE é uma medida mais robusta em relação ao MSE devido à sua interpretação mais intuitiva e por trabalhar nas mesmas unidades dos dados. Uma desvantagem da medida MSE é que ela é mais sensível em dados que possuem anomalias, tais como os *outliers*, dado que o erro em relação a um *outlier* pode aumentar o valor do MSE.

Uma forma alternativa de avaliar o desempenho dos modelos é utilizando a razão entre o MSE do regressor a ser avaliado e o MSE de um regressor base, que não pode possuir MSE nulo[1]. Ou seja, um MSE relativo, conforme apresentado abaixo

$$MSE_R(\mathcal{T}, g, g_b) = \frac{MSE(\mathcal{T}, g)}{MSE(\mathcal{T}, g_b)} = \frac{\sum_{i=1}^{m}(y_j^p - y_j)^2}{\sum_{i=1}^{m}(y_j^b - y_j)^2} \tag{2.3}$$

O modelo base que utilizaremos foi inspirado no modelo denominado *Dummy*, cuja predição é fixa e definida de acordo com alguma estratégia. Alguns exemplos de estratégia a ser utilizada para gerar as previsões são: média, que prevê sempre a média do conjunto de treinamento; mediana, que prevê sempre a mediana do conjunto de treinamento; quantil, que prevê sempre um quantil especificado do conjunto de treinamento; constante, que prevê sempre um valor constante fornecido.

Neste texto, utilizaremos a estratégia da média. Ou seja, o valor predito y_j^b pelo preditor base, para qualquer elemento $\mathbf{x}$, será dado por

$$y^b = g_b(\mathbf{x}) = \frac{1}{m}\sum_{j=1}^{m} y_j.$$

Denotamos $\mathbf{y}^b = g_b(X)$.

Curiosamente, ao utilizar a estratégia da média, é possível determinar um valor denominado coeficiente de regularização, conhecido também como R^2, definido na equação (2.5). Neste caso, o MSE relativo (2.3) será dado por uma medida denominada fração de variância inexplicada (FVU, *fraction of variance unexplained*).

$$FVU(\mathcal{T}, g) = \frac{\sum_{i=1}^{m}(y_j^p - y_j)^2}{\sum_{i=1}^{m}(\bar{y} - y_j)^2} \tag{2.4}$$

O R^2, também conhecido como coeficiente de determinação, é uma medida estatística que define o quão próximos os dados estão da hipersuperfície de regressão ajustada.

$$R^2 = 1 - FVU(\mathcal{T}, g) \tag{2.5}$$

Observe que quando R^2 é maior que 0, então o modelo de ajuste se mostrou melhor do que o modelo base. Se R^2 for igual a 1, como o MSE do modelo base é não

[1] Neste caso, se o MSE do regressor base for nulo, então cada termo da soma que define o MSE é zero. Logo o modelo base se ajusta perfeitamente aos dados.

nulo, então o MSE do modelo ajustado é nulo, ou seja, o modelo se ajustou perfeitamente aos dados. Por outro lado, um valor de R^2 menor ou igual a zero significa que o modelo ajustado é tão ruim quanto, ou ainda pior que o modelo base. Se o valor assumido for nulo, então a razão entre o MSE do modelo a ser avaliado e o modelo base é igual a 1. Ou seja, os dois modelos possuem o mesmo valor para o MSE, o que significa que o ajuste do modelo foi tão ruim quanto predizer todos os dados com um mesmo valor.

Métricas para Modelos de Classificação

Em problemas de classificação binária, é usual utilizar-se de rotulação da classe de maior interesse sendo a classe positiva ($\mathcal{P}$) e a de menor interesse a classe negativa ($\mathcal{N}$). Neste caso, seja um dado de treinamento cuja classe predita foi $\mathcal{N}$, mas sua classe verdadeira é $\mathcal{P}$, então dizemos ter um falso negativo (FN). Caso contrário, se a predição foi $\mathcal{P}$, então tem-se um verdadeiro positivo. O mesmo vale se acontecer o contrário, ou seja, um dado cuja classe predita seja $\mathcal{P}$, mas sua classe verdadeira é $\mathcal{N}$, então tem-se falso positivo (FP). Caso contrário, se a predição foi $\mathcal{N}$, então tem-se um verdadeiro negativo.

Uma medida básica de desempenho é a acurácia. A acurácia nada mais é do que uma taxa real dos acertos em relação às predições feitas. Essa medida para avaliação de modelos para tarefas de classificação pode não ser adequada para dados com classes desbalanceadas, isto é, dados em que o número de elementos de uma classe é bem maior do que o número de elementos de outra classe. Neste caso, por exemplo, a taxa de acerto geralmente pode ser alta, mas a taxa de erro, mesmo sendo pequena, pode não ser o objetivo desejado. Para dados desbalanceados, uma boa medida de avaliação para modelos de classificação é a Medida-F, definida em (2.7), também conhecida como F_1Score que calcula a média harmônica entre duas outras medidas denominadas precisão e revocação, definidas em (2.6).

A precisão é a razão entre a quantidade dos dados classificados corretamente como positivos (verdadeiros positivos) e a quantidade total dos dados classificados como positivos, isto é, tanto os verdadeiros positivos quanto os falsos positivos. Neste caso, só são considerados os exemplos que são de fato classificados pelo modelo na classe positiva. Esta medida é útil quando predizer um valor como FP for considerado mais prejudicial do que predizer como um FN.

A revocação é a razão entre os dados que foram corretamente classificados como positivos e a quantidade de elementos que são verdadeiramente positivos, isto é, os verdadeiros positivos e os falso negativos. Neste caso, só são considerados os exemplos que são de fato da classe positiva. Diferente da precisão, esta medida é útil quando predizer um valor como FN for considerado mais prejudicial do que predizer como um FP.

$$Pre = \frac{VP}{VP + FP} \qquad Rev = \frac{VP}{VP + FN} \tag{2.6}$$

$$F_1Score = 2\frac{Pre * Rev}{Pre + Rev} \tag{2.7}$$

A medida F_1Score é uma maneira de observar as medidas precisão e revocação simultaneamente. Obter um valor de F_1Score baixo pode indicar que uma das duas medidas é baixa.

2.2 Algumas Técnicas Clássicas de Aprendizado de Máquinas

Existem na literatura muitas técnicas de aprendizado de máquinas para tarefas de classificação e de regressão. Nesta seção, falaremos sobre as máquinas de vetores de suporte e as redes neurais artificiais, úteis para o desenvolvimento deste trabalho.

2.2.1 Máquinas de Vetores de Suporte

As máquinas de vetores de suporte (SVMs, *support vector machines*), desenvolvidas por Vapnik e colaboradores, são um método de aprendizado baseado fortemente no aprendizado estatístico [58].

Os classificadores de vetores de suporte (SVCs, *support vector classifiers*) superam as limitações dos *perceptrons* de Rosenblatt para tarefas de classificação binária. Um SVC para classificação binária prevê a classe de uma amostra calculando uma soma ponderada no espaço de características. O treinamento de um SVC é realizado maximizando a margem de separação entre os dados das duas classes, com rótulos representados no conjunto $\{-1, +1\}$.

No caso de um SVC linear, utiliza-se como superfície de decisão um hiperplano separador. A equação de um hiperplano separador, em um espaço de dimensão n, é dada por $\mathbf{w}^T\mathbf{x} + b = 0$, com $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$ e $b \in \mathbb{R}$. A função $g : \mathbb{R}^n \to \mathbb{R}$, dada por $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, seguida da função sinal definida na equação (2.8), é utilizada para classificar um novo dado $\mathbf{x}$

$$f(a) = \begin{cases} +1, \text{ se } a \geqslant 0 \\ -1, \text{ se } a < 0. \end{cases} \tag{2.8}$$

Os classificadores SVC com margens rígidas utilizam o conceito de dados linearmente separáveis e superfícies lineares de decisão. Porém, nem sempre é possível separar o conjunto de dados em seu respectivo grupo de classes por um hiperplano. Neste caso, para encontrar um classificador que ainda seja linear, o SVC com margens flexíveis é utilizado. No entanto, na prática, os dados não são linearmente separáveis e a superfície de decisão linear ótima para o problema pode não nos dar uma boa classificação dos dados. O desempenho de classificação de um SVC aumenta significativamente usando o truque do *kernel*, que consiste em mapear os dados a um novo espaço de características, que

possui um produto interno, de tal forma que nesse novo espaço os dados sejam linearmente separáveis. O truque do *kernel* permite que um modelo SVC seja efetivamente aplicado a problemas de classificação não linear.

Ao mapear os dados a um novo espaço de características de maior dimensão, utilizando um mapeamento $\phi : \mathbb{R}^n \to \mathbb{R}^r$ $(r \geqslant n)$ que define um produto interno $\phi(\mathbf{x})^T\phi(\mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y})$, o treinamento pode ser feito resolvendo um problema de otimização quadrática dado pelo Problema (2.9),

$$\begin{aligned} \underset{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{r+1+m}}{\text{minimize}} \quad & \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + \boldsymbol{\xi}^T\mathbf{p} \\ s.a. \quad & y_j(\mathbf{w}^T\phi(\mathbf{x}_j) + b) \geqslant 1 - \xi_j, j = 1, ..., m \\ & \boldsymbol{\xi} \geqslant 0, \end{aligned} \tag{2.9}$$

em que $\mathbf{w} \in \mathbb{R}^r, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^m$ e $\mathbf{p} = (p, ..., p) \in \mathbb{R}^m$, com p sendo um parâmetro de penalização especificado pelo usuário, que controla a flexibilização do problema de minimizar o erro de treinamento e maximizar a margem de separação.

O Problema (2.9) é denominado problema primal. A construção de seu dual, dado pelo Problema (2.10), é feita usando uma função Lagrangiana desenvolvida a partir da função objetivo e das restrições correspondentes

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T K\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbb{1} \\ s.a. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leqslant \boldsymbol{\alpha} \leqslant \mathbf{p}. \end{aligned} \tag{2.10}$$

As componentes do vetor $\mathbf{p}$ são iguais à constante p, parâmetro de penalização especificado pelo usuário. Os elementos α_j são os multiplicadores de Lagrange, K é uma matriz quadrada com entradas dadas por $\{k_{i,j} = y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)\}$, com $\kappa(\mathbf{x}_i, \mathbf{x}_j) \equiv \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$ e $\mathbb{1}$ é um vetor de entradas unitárias.

A formulação via multiplicadores de Lagrange do problema de otimização, usado no treinamento de um SVC, depende apenas de função *kernel* κ utilizada. Isso quer dizer que não é necessário conhecer a função ϕ que mapeia os dados para o novo espaço de características. Os vetores de suporte são os elementos $\mathbf{x}_j$ tais que $\alpha_j > 0$. A classificação de um novo elemento depende apenas de função *kernel* definida no espaço de características e não da função utilizada para fazer o mapeamento.

Além dos SVCs, cujo foco está em resolver problemas de classificação, o SVM também pode ser usado para resolver problemas de regressão. Tais métodos são chamados regressores de vetores de suporte (SVRs, *support vector regressors*). Um SVR linear é treinado utilizando um conjunto de amostras $\{(\mathbf{x}_j, y_j)\}_{j=1}^m \subseteq \mathbb{R}^n \times \mathbb{R}$, cujo objetivo é encontrar elementos $\mathbf{w}$ e b que definem um hiperplano $H : \mathbf{w}^T\mathbf{x} + b = 0$, tal que os dados de amostra estejam o mais próximo possível deste hiperplano. Sua formulação utiliza-se

de uma função denominada função de perda ϵ-insensível desenvolvida por Vapnik [58]. A função de perda ϵ-insensível é dada por $|y - g(\mathbf{x})|_\epsilon = \max(0, |y - g(\mathbf{x})| - \epsilon)$, sendo ϵ um parâmetro predeterminado que define uma tolerância responsável pela região entre dois hiperplanos paralelos a H, diferindo em deslocamento.

Considerando um modelo de regressão linear, no qual a dependência de um escalar observável y a um elemento $\mathbf{x}$ é descrita por $y = g(\mathbf{x}) := \mathbf{w}^T\mathbf{x} + b$, em que os parâmetros $\mathbf{w}$ e b são ambos desconhecidos, então o treinamento de um SVR pode ser feito minimizando $\frac{1}{2}\mathbf{w}^T\mathbf{w} + p\sum_{j=1}^{m} |y - g(\mathbf{x})|_\epsilon)$, que é equivalente a resolver o problema quadrático convexo definido no Problema (2.11).

$$
\begin{aligned}
\underset{(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}')\in\mathbb{R}^{n+1+2m}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + p\sum_{j=1}^{m}(\xi_j + \xi_j') \\
s.a. \quad & y_j - g(\mathbf{x}_j) \leqslant \epsilon + \xi_j, \quad j = 1, \ldots, m \\
& g(\mathbf{x}_j) - y_j \leqslant \epsilon + \xi_j', \quad j = 1, \ldots, m \\
& \boldsymbol{\xi}, \boldsymbol{\xi}' \geqslant 0.
\end{aligned}
\tag{2.11}
$$

Se um determinado dado pertence à margem definida pelos hiperplanos paralelos a H, então ele não contribuirá com nenhuma perda para a função objetivo, logo não carregará qualquer informação sobre a posição do hiperplano.

O parâmetro p é uma constante que determina a compensação entre o erro de treinamento e o termo de penalização. As restrições indicam que os dados devem estar o mais próximo possível do hiperplano H. As variáveis de folga $\boldsymbol{\xi}$ e $\boldsymbol{\xi}'$ foram incluídas a fim de permitir que elementos estejam fora da margem definida pela tolerância ϵ. Elas também descrevem uma função de perda, representada pela soma na função objetivo, responsável pela minimização do erro de treinamento. Qualquer erro de treinamento $|y_j - g(\mathbf{x}_j)|$ menor que ϵ não requer ξ_j ou ξ_j' diferentes de zero.

Assim como nos SVCs, o algoritmo de regressão SVR pode ser desenvolvido estimando funções lineares e escrito em termos de produtos internos para generalizar para o caso não linear. O truque do *kernel* também pode ser usado no desenvolvimento de um SVR [59]. Neste caso, é feito um mapeamento dos dados a um novo espaço de características, que possui um produto interno, de tal forma que nesse novo espaço seja possível encontrar um hiperplano em que os dados mapeados estejam o mais próximos possível deste hiperplano.

Similar à construção do problema dual no treinamento dos SVCs com truque do *kernel*, no problema de otimização dual utilizado para treinar um SVR, a ideia é construir a função Lagrangiana a partir da função objetivo e das restrições correspondentes [59]. Os vetores de suporte e os multiplicadores de Lagrange para um problema de regressão

são então obtidos como soluções de um problema de programação quadrática, dual ao Problema (2.11), dado pelo Problema (2.12), com K' uma matriz quadrada com entradas dadas por $K'_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Neste caso, os vetores de suporte são os elementos $\mathbf{x}_j$ da amostra de treinamento associados aos multiplicadores de Lagrange (α_j ou α'_j) não nulos.

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\alpha}'\in\mathbb{R}^m}{\text{minimize}} \quad & \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T K'(\boldsymbol{\alpha}-\boldsymbol{\alpha}') + \epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}')^T\mathbb{1} - \mathbf{y}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}') \\ s.a. \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}')^T\mathbb{1} = 0 \\ & 0 \leqslant \boldsymbol{\alpha}, \boldsymbol{\alpha}' \leqslant \mathbf{p}. \end{aligned} \tag{2.12}$$

A estimativa y de um novo elemento $\mathbf{x}$ é dada por

$$y = f(g(\mathbf{x})) = f\left(\sum_{j\in I} w_j \kappa(\mathbf{x}, \mathbf{x}_j) + b\right), \tag{2.13}$$

sendo f a função identidade, no caso de tarefas de regressão, ou a função sinal, no caso de tarefas de classificação, I o conjunto de índices relativos aos N vetores de suporte da respectiva tarefa e w_j o parâmetro encontrado ao resolver os problemas de otimização relacionado [59], sendo, $w_j = \alpha'_j - \alpha_j$, para o caso de regressão e $w_j = \alpha_j y_j$, para o caso de classificação. O termo b pode ser calculado indiretamente usando a média

$$b = \frac{1}{N}\sum_{i\in I}\left(y_i - \sum_{j\in I} w_j \kappa(\mathbf{x}_j, \mathbf{x}_i)\right).$$

Alguns exemplos de funções *kernel* incluem o *kernel* linear e o *kernel* gaussiano definidos na Tabela 1. Comumente, o parâmetro γ é tomado sendo $1/(2\sigma^2)$, com σ sendo o desvio padrão entre os dados do conjunto de treinamento. O *kernel* gaussiano produz uma função de base radial (RBF, *radial-basis function*).

Kernel	Lei de Formação
Linear	$\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$
Gaussiano	$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\right)$

Tabela 1 – Exemplos de funções *kernel*

2.2.2 Redes Neurais Artificiais

Inspiradas no cérebro humano, as redes neurais artificiais (RNA) estão entre as técnicas de aprendizado de máquina mais eficientes usadas para resolver tarefas de reconhecimento de padrões, incluindo, por exemplo, visão computacional e processamento

de linguagem natural [1]. As RNAs são úteis para modelar a maneira pela qual o cérebro executa uma tarefa ou função específica de interesse [6].

As arquiteturas de RNA que consideraremos neste trabalho consistem em uma camada de entrada, uma camada de saída e, eventualmente, camadas ocultas. Os dados de entrada são convertidos em dados de saída através de um processo definido utilizando as informações das camadas que contêm seus pesos sinápticos. As camadas são compostas de neurônios artificiais.

Em um neurônio artificial, os sinais de entrada das sinapses interagem e são agregados em um único valor. A expressão que descreve como um neurônio manipula a entrada dos neurônios é também conhecida como função de agregação. Uma função denominada função de ativação realiza o cálculo da saída final. Um neurônio artificial também pode receber e agregar saídas de outros neurônios.

Um exemplo clássico de um neurônio artificial é apresentado na Figura 2.1, com função de agregação linear dada por $g(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$.

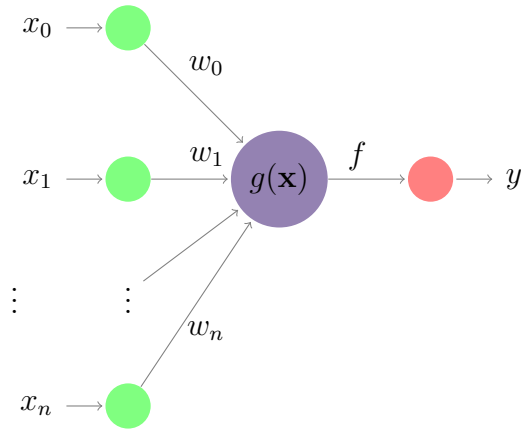


Figura 2.1 – Neurônio Artificial Clássico

Dados os sinais de entrada de um neurônio $\mathbf{x} := (x_0, x_1, ..., x_n) \in \mathbb{R}^{n+1}$, é feita uma soma ponderada utilizando os pesos sinápticos $\mathbf{w} = (w_0, w_1, ..., w_n) \in \mathbb{R}^{n+1}$. A função de agregação g é dada pela soma ponderada entre os sinais e os pesos. Em seguida, é aplicada uma função de ativação na combinação linear. Por fim, a saída y é o resultado da combinação linear seguida da aplicação da função de ativação, ou seja,

$$y = f\left(g(\mathbf{x})\right) = f\left(\mathbf{x}^T\mathbf{w}\right). \tag{2.14}$$

Outros exemplos de neurônios artificiais podem ser obtidos ao trocar a operação de combinação linear por outras operações matemáticas.

Comumente o sinal de entrada x_0 é fixado ao valor $+1$ e o peso sináptico w_0 é tratado como o que chamamos de *bias*. O bias tem o efeito de aumentar ou diminuir a resposta da função de ativação f, dependendo se é positiva ou negativa, respectivamente.

A arquitetura das RNAs depende da forma como os neurônios se conectam. Existem dois tipos clássicos de arquitetura de redes neurais: as redes neurais progressivas (*feedforward*), redes neurais cuja estrutura não possui ciclos em suas conexões; e as redes neurais recorrentes, redes que possuem pelo menos um ciclo em suas conexões. Neste trabalho, vamos nos restringir apenas a redes neurais do tipo *feedforward*.

O *perceptron* de Rosenblatt é o exemplo mais simples de uma rede neural com arquitetura *feedforward* e que possui um único neurônio. Ele foi introduzido por Rosenblatt no final dos anos 1950 com aplicações em tarefas de classificação binária [5]. Na Figura 2.1 é apresentado o *perceptron* de Rosenblatt sendo $g(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$, com $\mathbf{x} := (x_1, ..., x_n)$ e $\mathbf{w} := (w_1, ..., w_n)$ em $\mathbb{R}^n$, $b \in \mathbb{R}$ e f a função definida por $f(a) = 1$ se $a \geqslant 0$ e $f(a) = 0$ se $a < 0$.

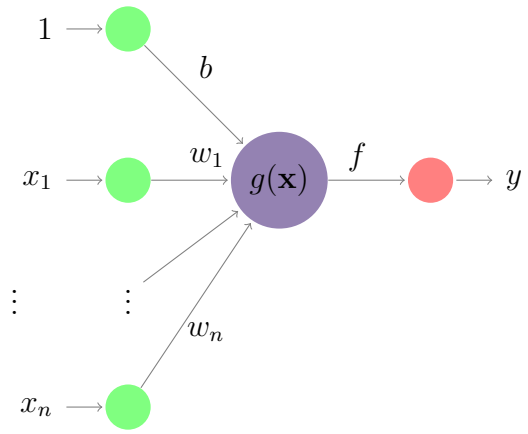


Figura 2.2 – *Perceptron* de Rosenblatt.

Precisamente, em problemas de classificação, o *perceptron* de Rosenblatt produz a classe 1 se a soma ponderada $\mathbf{x}^T\mathbf{w}$ for maior ou igual a $-b$ e retorna a classe 0 caso contrário. Além disso, Rosenblatt propôs um simples e eficiente algoritmo de treinamento que converge sempre que as amostras das duas classes são linearmente separáveis [6]. No entanto, para tarefas de classificação em que os dados não podem ser separados linearmente, o algoritmo proposto por Rosenblatt não se mostrou eficiente. As limitações do *perceptron* de camada única foi superada com a criação dos *perceptrons* com multicamadas. Um *Perceptron* Multicamadas (MLP, *multi-layer perceptron*) é um *perceptron* que possui mais de uma camada oculta. Na Figura 2.3 é apresentado um exemplo de uma rede MLP. Quando um MLP possui apenas uma camada oculta, dizemos que ele é um SLP (*single hidden-layer perceptron*).

As operações realizadas em cada neurônio de uma rede MLP podem ser definidas utilizando as mesmas operações realizadas nos *perceptrons* de camada única.

Considerando uma rede MLP com $t_c \in \mathbb{N}$ camadas ocultas, para cada camada $t = 1, 2, ..., t_c$ e para cada neurônio $j = 1, 2, ..., j_t$, sendo j_t a quantidade de neurônios da

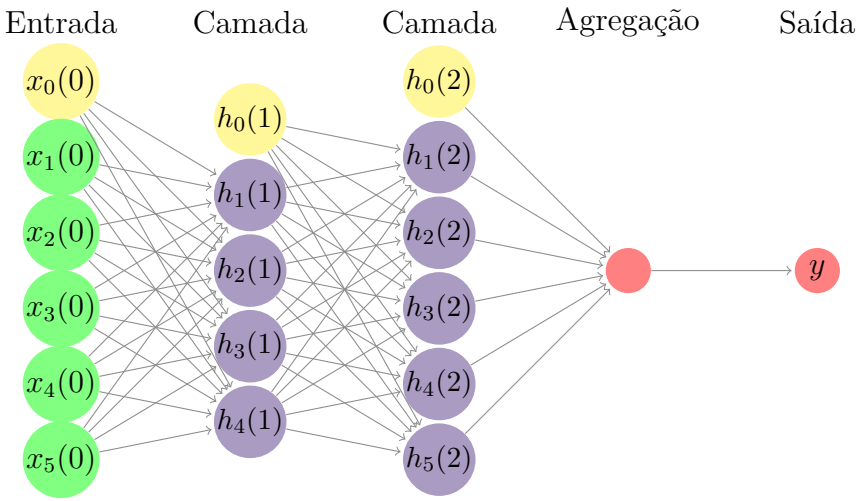


Figura 2.3 – MLP com duas camadas ocultas.

camada t, podemos tomar

$$h_j(t) = f_t((\mathbf{h}(t-1))^T \mathbf{w}_j(t)),$$

com $\mathbf{h}(0) = \mathbf{x}(0)$, $x_0(0) = 1$, $h_0(t) = 1$, $\mathbf{w}_j(t)$ os pesos da camada t para o neurônio j, f_t uma função de ativação definida para cada neurônio da camada e por fim, a saída $y = \sum_{j=1}^{j_{t_c}} h_j(t_c - 1) w_j(t_c)$.

Existem vários exemplos de funções de ativação. Algumas mais clássicas e bastante utilizadas são apresentadas na Tabela 2.

Nome	**Lei de formação**
Linear	$f(a) = a$
ReLu	$f(a) = \max(0, a)$
Sigmoid	$\sigma(a) = \dfrac{1}{1+e^{-a}}$
Sinal	$f(a) = \begin{cases} +1, \text{ se } a \geqslant 0 \\ -1, \text{ se } a < 0 \end{cases}$
Limiar ou degrau	$f(a) = \begin{cases} 1, \text{ se } a \geqslant 0 \\ 0, \text{ se } a < 0 \end{cases}$
Tangente hiperbólica	$tanh(a) = \dfrac{2}{1+e^{-2a}} - 1$

Tabela 2 – Exemplos de funções de ativação.

É comum na área de ML os modelos terem suas arquiteturas apresentadas com formas semelhantes. O que os diferenciam são os seus processos de treinamento.

O MLP foi um dos responsáveis pelo impulsionamento do uso em larga escala do aprendizado profundo (DL, *deep learning*). Em meados da década de 1980, foi introduzida uma descrição dos algoritmos de treinamento de retropropagação [60]. O MLP treinado usando algoritmos de retropropagação supera as limitações do *perceptron* de Rosenblatt para tarefas de classificação em que os dados não podem ser separados linearmente [6]. O termo retropropagação vem do fato de que o algoritmo depende da retropropagação de erros para realizar o ajuste de peso das camadas intermediárias onde as propriedades da regra da cadeia são utilizadas.

As propriedades das funções de ativação estão diretamente ligadas à eficiência do treinamento da rede. Por exemplo, no método de retropropagação, a regra da cadeia impõe que as funções sejam deriváveis em quase todo seu domínio. Neste caso, se uma função de ativação não for derivável, o método pode não conseguir atualizar os pesos no processo de treinamento.

Um otimizador comumente utilizado no treinamento de redes neurais artificiais é o otimizador Adam (*Adaptive Moment Estimation*), um algoritmo para otimização baseada em gradiente estocástico de primeira ordem. A taxa de aprendizado η (*learning rate)* deste otimizador determina o tamanho dos passos para atingir o mínimo local da função objetivo do problema. Outros parâmetros deste otimizador são β_1 e β_2, para os quais, em tarefas de aprendizado de máquinas, uma boa escolha a se tomar são 0.9 e 0.999, respectivamente [26].

Modelos de aprendizado de máquinas podem superajustar aos dados de treinamento, diminuindo as chances de generalização do modelo. Isto é, obter ótimos resultados de avaliação no conjunto de treinamento, no entanto, não conseguir ser um bom preditor em outros conjuntos de dados. Nestes casos, é comum utilizar um método de regularização que inclui na função objetivo um termo que penaliza os coeficientes do modelo. Algumas técnicas de regularização comuns são as regularizações R_1, R_2 e Elastic Net, que é uma combinação entre as regularizações R_1, R_2 [61]. Técnicas de regularização são úteis no processo de generalização das redes neurais.

Paralelamente às redes neurais, as SVMs são algoritmos de aprendizado de máquina que talvez tenham o mais elegante de todos os métodos de aprendizado utilizando *kernel* [3]. Uma vez treinado o modelo preditivo utilizando o truque do *kernel* e encontrado os vetores de suporte e seus respectivos multiplicadores de Lagrange, as SVMs podem ser vistas como uma rede de função de base radial (RBF, *radial-basis function*) [3]. Elas podem ser definidas a partir dos princípios das redes *feedforward* com uma única camada oculta de unidades não lineares (neurônios com funções não lineares) cuja arquitetura pode ser construída conforme o exemplo apresentado na Figura 2.4, com camada de produtos

internos *kernels* $h_j = \kappa(\mathbf{x}, \mathbf{x}_j)$ e pesos w_j, definido conforme a categoria da tarefa, para todo $j \in I = \{1, \ldots, m_s\}$.

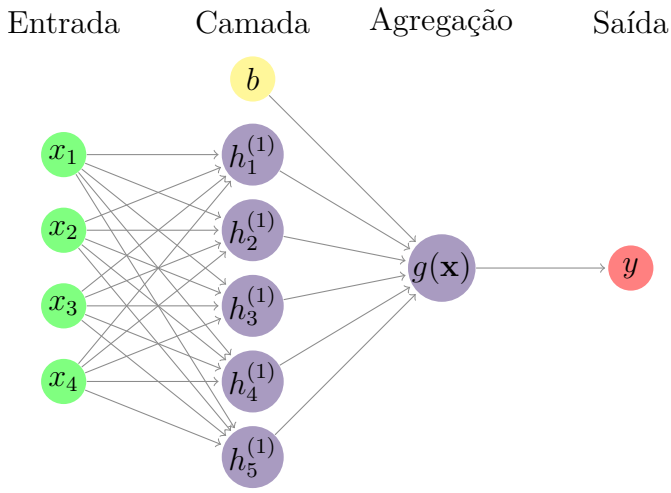


Figura 2.4 – Exemplo da estrutura de uma rede neural SVM.

Na Figura 2.5 e na Figura 2.6 são apresentados alguns exemplos de redes neurais aplicadas a problemas de regressão e classificação. Para problemas de regressão, é mostrado: a rede SLP com função de ativação ReLU, o SVR linear e gaussiano. Além dessas técnicas, para problemas de classificação, incluímos o *perceptron* de Rosenblatt. Graficamente, é possível ver que o MLP (função de ativação ReLU), SVR e SVC com *kernel* se ajustam melhor aos problemas e os modelos lineares não se mostraram eficientes.

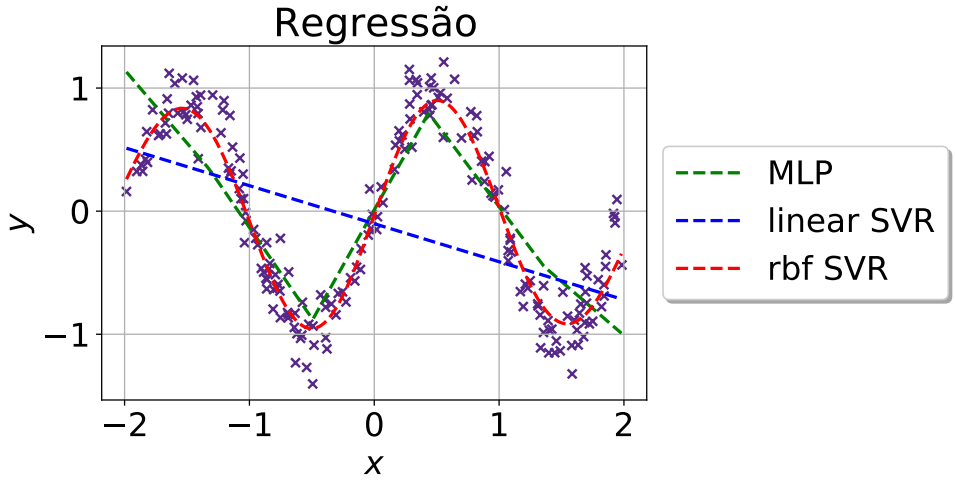


Figura 2.5 – Exemplos de modelos de Regressão.

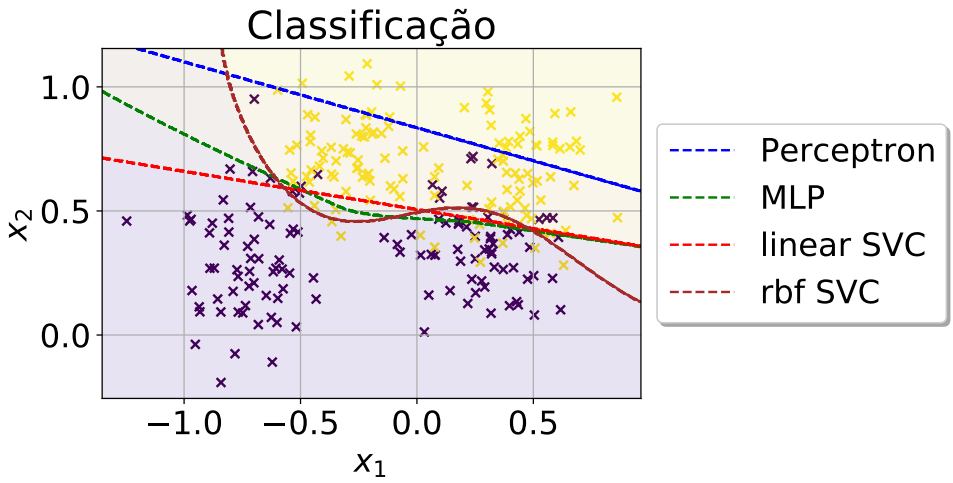


Figura 2.6 – Exemplos de modelos de Classificação.

2.3 Redes Neurais Artificiais como Aproximadores Universais

As principais descobertas em relação às redes neurais artificiais e aproximações de funções foram feitas em 1989. O mais famoso resultado, estabelecido por Cybenko [62], diz que uma rede neural, com uma única camada oculta e com função de ativação sigmoidal contínua, pode aproximar arbitrariamente qualquer função real contínua definida num hipercubo unitário. As funções do tipo sigmoidal são funções σ que satisfazem as seguintes propriedades: $\sigma(a) = 1$, quando a tende a $+\infty$ e $\sigma(a) = 0$, quando a tende a $-\infty$. Em [63], Hornik e outros encontraram resultados semelhantes estabelecendo rigorosamente que as redes *feedforward* multicamadas com apenas uma camada oculta, para funções de ativação sigmoidal não decrescente, são capazes de aproximar qualquer função contínua de um espaço n-dimensional em outro espaço m-dimensional. Essas aproximações podem ser feitas com qualquer grau de precisão desejado, desde que haja um número suficiente de neurônios ocultos disponíveis. Tais resultados são conhecidos como Teorema da Aproximação Universal e são consequências do Teorema da Aproximação de Stone-Weierstrass [64].

Resultados semelhantes para outras categorias de funções de ativação foram estabelecidos e, embora os resultados mencionados acima tenham sido obtidos para funções do tipo sigmoidal, Cybenko definiu uma classe de funções de ativação para as quais estes resultados ainda são válidos. Tais funções são denominadas funções discriminatórias[2].

As redes *feedforward* multicamadas são uma classe de aproximadores universais e o desempenho destas redes está diretamente ligada ao uso das funções de ativação. A função de aproximação obtida por uma rede neural com função ativação sigmoidal é uma

[2] Omitiremos uma explanação a respeito das funções discriminatórias dado que os termos que a definem fogem do escopo deste trabalho. Mais informações a respeito destas funções podem ser encontradas em [62].

função suave. Já para a função de ativação ReLU, a função aproximada obtida não é uma função suave. A utilização da função ReLU como função de ativação produz uma função de aproximação que é contínua linear por partes. A grosso modo, uma função contínua linear por partes é uma função cujo gráfico é composto por vários hiperplanos.

No Capítulo 3 falaremos com mais detalhes sobre as funções lineares por partes, um conceito bastante importante para os resultados apresentados neste texto.

3 Morfologia Matemática e Redes Neurais Morfológicas

3.1 Conceitos básicos em Morfologia Matemática e Teoria dos Reticulados

No final dos anos 1980, Ritter e colaboradores desenvolveram a chamada álgebra de imagens como uma tentativa de fornecer uma estrutura unificada para técnicas de processamento e análise de imagens [7]. Usando a estrutura da álgebra de imagens, operadores lineares e operadores morfológicos são definidos analogamente. Nesse mesmo período, nos trabalhos de Ritter e Sussner, ao substituir o produto escalar linear usual no modelo *perceptron* de Rosenblatt com a operação correspondente baseada em reticulado, usada para definir operadores morfológicos, surgiu o *perceptron* morfológico [10, 11]. Recentemente, surgiram muitos outros modelos de aprendizado de máquinas que utilizam operações baseadas em reticulados [16, 19, 24, 29, 32]. Neste trabalho introduziremos o regressor dilatação-erosão linear (ℓ-DER) para tarefas de regressão e também introduziremos o *perceptron* dilatação-erosão linear (ℓ-DEP) para tarefas de classificação. Mas antes, apresentaremos a base matemática necessária para entender tais modelos. Iniciaremos falando sobre alguns conceitos básicos da Morfologia Matemática e Teoria dos Reticulados.

A morfologia matemática é uma teoria não linear amplamente utilizada para processamento e análise de imagens [65, 66]. Os operadores elementares da morfologia matemática, denominados dilatação e erosão, comutam com as operações básicas em reticulados. Do ponto de vista teórico, a morfologia matemática é muito bem definida em reticulados completos. Desta forma, nesta seção apresentaremos alguns dos conceitos básicos da teoria de reticulados para a compreensão da teoria e modelos que serão apresentados neste trabalho.

Definição 3.1. (Conjunto parcialmente ordenado) *Um conjunto não vazio $\mathbb{L}$ equipado com uma relação binária $\leq$ é um conjunto parcialmente ordenado, denotado por $(\mathbb{L}, \leq)$, se as seguintes propriedades forem satisfeitas:*

- *reflexiva:* $\mathbf{x} \leq \mathbf{x}, \quad \forall \mathbf{x} \in \mathbb{L}$
- *anti-simétrica:* $\mathbf{x} \leq \mathbf{y} \quad e \quad \mathbf{y} \leq \mathbf{x}$ *implica* $\mathbf{x} = \mathbf{y}, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{L}$
- *transitiva:* $\mathbf{x} \leq \mathbf{y} \quad e \quad \mathbf{y} \leq \mathbf{z}$ *implica* $\mathbf{x} \leq \mathbf{z}, \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{L}$.

É fácil ver que o conjunto $(\mathbb{R}, \leqslant)$, com $\leqslant$ sendo a desigualdade usual no conjunto dos números reais, é um conjunto parcialmente ordenado. Observe que neste caso apresentamos um exemplo de ordem parcial para um conjunto unidimensional. Na definição a seguir, apresentamos um exemplo de ordem parcial para um conjunto com elementos n-dimensionais.

Exemplo 3.1. *Sejam $\mathbb{L}$ um conjunto parcialmente ordenado e X um subconjunto de $\mathbb{L}^n$ munido da relação $\preceq$ definida por $\mathbf{x} \preceq \mathbf{y}$ se, e somente se, $x_k \leqslant y_k$, para todo $k = 1, \ldots, n$. A relação $\preceq$ é uma ordem parcial.*

Seja $\mathbb{L}$ um conjunto parcialmente ordenado munido de uma ordem parcial $\preceq$. O supremo e o ínfimo de um conjunto $X \subseteq \mathbb{L}$ são denotados, respectivamente, por $\sup X$ e $\inf X$. O ínfimo (maior limitante inferior) de X satisfaz $\inf X \preceq \mathbf{x}$, para todo $\mathbf{x} \in X$. Analogamente, o supremo (menor limitante superior) de X satisfaz $\mathbf{x} \preceq \sup X$, para todo $\mathbf{x} \in X$.

Definição 3.2. (Reticulado) *Um conjunto parcialmente ordenado $\mathbb{L}$ é um reticulado se todo subconjunto finito e não vazio de $\mathbb{L}$ tem um ínfimo e um supremo em $\mathbb{L}$.*

Definição 3.3. (Reticulado completo) *Um conjunto parcialmente ordenado $\mathbb{L}$ em que cada subconjunto X de $\mathbb{L}$ possui um supremo e um ínfimo em $\mathbb{L}$ é chamado reticulado completo.*

Vimos que $(\mathbb{R}, \leqslant)$ é um conjunto parcialmente ordenado. Pode-se mostrar que $(\mathbb{R}, \leqslant)$ é um reticulado, porém não é um reticulado completo. Com efeito, todos os subconjuntos finitos de $\mathbb{R}$ possuem um ínfimo e um supremo em relação à ordem usual. No entanto, existem subconjuntos em que o ínfimo ou o supremo não pertencem a $\mathbb{R}$. Por exemplo, o conjunto $(0, +\infty) \subset \mathbb{R}$ não possui supremo em $\mathbb{R}$. Por outro lado, denotando por $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ o conjunto dos números reais estendido, então $\bar{\mathbb{R}}$ possui supremo e ínfimo para qualquer subconjunto. Portanto, o conjunto $(\bar{\mathbb{R}}, \leqslant)$ é um reticulado completo. E mais, o produto cartesiano $\bar{\mathbb{R}}^n$ é um reticulado completo com a ordenação parcial dada por $\mathbf{x} \preceq \mathbf{y} \Leftrightarrow x_i \leqslant y_i$, para $i = 1, \ldots, n$. Denotaremos esse conjunto parcialmente ordenado por $(\bar{\mathbb{R}}^n, \leqslant)$.

Teoricamente, quando estamos tratando de reticulados completos, algumas operações precisam ser explicitamente definidas, por exemplo, a soma ou subtração entre valores do tipo $\pm\infty$ em $\bar{\mathbb{R}}$. Para isso, podemos considerar estruturas algébricas equipadas com as operações $(\sup, +)$ ou $(\inf, +')$. Neste caso, tem-se:

$$\infty + (-\infty) = (-\infty) + \infty = +\infty$$
$$\infty +' (-\infty) = (-\infty) +' \infty = -\infty$$

Neste tese, na pártica nos limitaremos a conjuntos com apenas valores reais. Portanto, tais operação não serão problema para os modelos baseados em operações em reticulados completos que apresentaremos.

Até aqui, apresentamos alguns conceitos básicos da Teoria de Reticulados. Os operadores dilatação e erosão comutam com as operações de supremo e ínfimo de um reticulado completo. Formalmente, tais operadores são definidos da seguinte forma.

Definição 3.4. (Dilatação e erosão) *Dados dois reticulados completos* $\mathbb{L}$ *e* $\mathbb{M}$*, o par de operadores* $\delta : \mathbb{L} \to \mathbb{M}$ *e* $\varepsilon : \mathbb{L} \to \mathbb{M}$ *são denominados dilatação e erosão se satisfazem, respectivamente, as seguintes identidades para todo* $X \subset \mathbb{L}$ *[67]:*

$$\delta\left(\sup X\right) = \sup_{\mathbf{x}\in X}\{\delta(\mathbf{x})\} \quad e \quad \varepsilon\left(\inf X\right) = \inf_{\mathbf{x}\in X}\{\varepsilon(\mathbf{x})\}. \tag{3.1}$$

Similarmente, uma anti-dilatação e uma anti-erosão são definidos da seguinte forma

Definição 3.5. (Anti-dilatação e anti-erosão) *Dados dois reticulados completos* $\mathbb{L}$ *e* $\mathbb{M}$*, o par de operadores* $\bar{\varepsilon}, \bar{\delta} : \mathbb{L} \to \mathbb{M}$*, então* $\bar{\delta}$ *e* $\bar{\varepsilon}$ *são denominados anti-dilatação e anti-erosão se satisfazem, respectivamente, as seguintes identidades para todo* $X \subset \mathbb{L}$ *[68]:*

$$\bar{\delta}\left(\sup X\right) = \inf_{\mathbf{x}\in X}\{\delta(\mathbf{x})\} \quad e \quad \bar{\varepsilon}\left(\inf X\right) = \sup_{\mathbf{x}\in X}\{\varepsilon(\mathbf{x})\}. \tag{3.2}$$

Exemplo 3.2. *Dados* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$*, os operadores* $\delta_{\mathbf{a}}, \varepsilon_{\mathbf{b}} : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ *definidos por*

$$\delta_{\mathbf{a}}(\mathbf{x}) = \max_{j=1,\ldots,n}\{a_j + x_j\} \quad e \quad \varepsilon_{\mathbf{b}}(\mathbf{x}) = \min_{j=1,\ldots,n}\{b_j + x_j\}, \tag{3.3}$$

para todo $\mathbf{x} \in \bar{\mathbb{R}}^n$ *são, respectivamente, uma dilatação e uma erosão [16].*

Definição 3.6. (Adjunção) *Dado dois reticulados completos* $\mathbb{L}$ *e* $\mathbb{M}$*. O par de operadores* (ε, δ)*, com* $\varepsilon : \mathbb{L} \to \mathbb{M}$ *e* $\delta : \mathbb{M} \to \mathbb{L}$*, é denominado uma adjunção se*

$$\delta(x) \preceq y \Leftrightarrow x \preceq \varepsilon(y)$$

para todo $y \in \mathbb{L}$ *e* $x \in \mathbb{M}$*.*

A adjunção relaciona os operadores de dilatação e erosão por meio de uma dualidade. Se um par de operadores formam uma adjunção, então dizemos que tais operadores são adjuntos. Se ε e δ são adjuntos, então ε é uma erosão e δ é uma dilatação. Para uma dilatação arbitrária δ, existe uma única erosão ε tal que (ε, δ) é uma adjunção. Dualmente, para uma erosão arbitrária ε, existe uma única dilatação δ tal que (ε, δ) é uma adjunção [69, 70].

A adjunção não é a única relação de dualidade entre erosões e dilatações. Uma outra relação que existe é com respeito a negação, isto é, os operadores δ e ε podem estar relacionados por meio de uma relação de dualidade baseada no conceito de negação [8]. Para entender o conceito de negação, é preciso definir uma bijeção involutiva.

Definição 3.7. (Negação em reticulados completos) *Sejam $\mathbb{L}$ e $\mathbb{M}$ reticulados completos. Uma bijeção $\nu : \mathbb{L} \to \mathbb{L}$ é involutiva se ela também for a sua própria inversa, isto é, para todo $x \in \mathbb{L}$ vale $\nu(\nu(x)) = x$. Uma involução ν é negação se ν reverte a ordem parcial de $\mathbb{L}$.*

Considere $\nu_{\mathbb{L}}$ e $\nu_{\mathbb{M}}$ negações em $\mathbb{L}$ e $\mathbb{M}$, respectivamente. A aplicação $\Phi^{\nu} : \mathbb{L} \to \mathbb{M}$ dada por $\Phi^{\nu} = \nu_{\mathbb{L}} \Phi \nu_{\mathbb{M}}$ é uma negação de $\Phi : \mathbb{L} \to \mathbb{M}$, com respeito a $\nu_{\mathbb{L}}$ e $\nu_{\mathbb{M}}$.

Exemplo 3.3. *Considere o reticulado completo $\bar{\mathbb{R}}$. O elemento $x^* \in \mathbb{R}$ definido em (3.4) é uma negação de x em $\bar{\mathbb{R}}$. O elemento $\mathbf{x}^* \in \bar{\mathbb{R}}^n$ definido por $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ é uma negação em $\bar{\mathbb{R}}^n$.*

$$x^* = \begin{cases} -x, & x \notin \{-\infty, +\infty\} \\ -\infty, & x = +\infty \\ +\infty, & x = -\infty. \end{cases} \tag{3.4}$$

Exemplo 3.4. *Dados $\mathbf{a}$ e $\mathbf{b}$ em $\mathbb{R}^n$, os operadores $\bar{\delta}_{\mathbf{a}}, \bar{\varepsilon}_{\mathbf{b}} : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ dados por*

$$\bar{\delta}_{\mathbf{a}}(\mathbf{x}) = \min_{j=1,...,n} \{a_j + x_j^*\} \quad e \quad \bar{\varepsilon}_{\mathbf{b}}(\mathbf{x}) = \max_{j=1,...,n} \{b_j + x_j^*\}, \tag{3.5}$$

para todo $\mathbf{x} \in \bar{\mathbb{R}}^n$ são, respectivamente, uma anti-dilatação e uma anti-erosão.

Considere uma dilatação δ, uma erosão ε e suas respectivas negações δ^{ν} e ε^{ν}. A negação de uma dilatação corresponde a uma erosão e a negação de uma erosão corresponde a uma dilatação. De forma geral, se (ε, δ) corresponde à uma adjunção, então o par $(\varepsilon^{\nu}, \delta^{\nu})$ é também uma adjunção [8, 71]. Considere os operadores morfológicos definidos em (3.2) e a negação em $\bar{\mathbb{R}}^n$ apresentada em (3.4). Então, pode-se mostrar que

$$\delta_{\mathbf{a}}^*(\mathbf{x}) = (\delta_{\mathbf{a}}(\mathbf{x}^*))^* = \varepsilon_{\mathbf{a}^*}(\mathbf{x}) \quad e \quad \varepsilon_{\mathbf{b}}^*(\mathbf{x}) = (\varepsilon_{\mathbf{b}}(\mathbf{x}^*))^* = \delta_{\mathbf{b}^*}(\mathbf{x}). \tag{3.6}$$

Os operadores $\delta_{\mathbf{a}}$ e $\varepsilon_{\mathbf{b}}$ do Exemplo (3.2) representam, respectivamente, uma dilatação e uma erosão definidas do reticulado completo $\bar{\mathbb{R}}^n$ ao reticulado completo $\bar{\mathbb{R}}$. Além disso, $\bar{\delta}_{\mathbf{a}}$ e $\bar{\varepsilon}_{\mathbf{b}}$ representam, respectivamente, uma anti-dilatação e uma anti-erosão definidas do reticulado completo $\bar{\mathbb{R}}^n$ ao reticulado completo $\bar{\mathbb{R}}$. Embora estejamos apresentando a definição utilizando a reta real estendida, neste trabalho aplicaremos os operadores apenas em dados com valores reais e finitos.

As operações de dilatação e erosão podem ser estendidas para conjuntos mais abstratos usando o conceito de ordem reduzida [72]. Recentemente, usando conceitos de ordem reduzida, Valle apresentou modelos de redes neurais morfológicas baseadas em ordens reduzidas supervisionadas [32]. Operadores como dilatação reduzida e erosão reduzida são operadores utilizados na definição destas redes neurais morfológicas.

Sejam $\mathbb{V}$ e $\mathbb{W}$ conjuntos arbitrários, não necessariamente reticulados completos. Além disso, sejam $\rho : \mathbb{V} \to \mathbb{L}$ e $\sigma : \mathbb{W} \to \mathbb{M}$ mapeamentos sobrejetivos, em que $\mathbb{L}$ e $\mathbb{M}$

são ambos reticulados completos. Um operador $\delta^r : \mathbb{V} \to \mathbb{W}$ é uma dilatação reduzida se existir uma dilatação $\delta : \mathbb{L} \to \mathbb{M}$ tal que

$$\sigma\left(\delta^r(\mathbf{x})\right) = \delta\left(\rho(\mathbf{x})\right), \quad \forall \mathbf{x} \in \mathbb{V}. \tag{3.7}$$

Além disso, um operador $\varepsilon^r : \mathbb{V} \to \mathbb{W}$ é uma erosão reduzida se existir uma erosão $\varepsilon : \mathbb{L} \to \mathbb{M}$ tal que

$$\sigma\left(\varepsilon^r(\mathbf{x})\right) = \varepsilon\left(\rho(\mathbf{x})\right), \quad \forall \mathbf{x} \in \mathbb{V}. \tag{3.8}$$

Operadores morfológicos reduzidos são ferramentas úteis na busca de esquemas de ordenação apropriados para dados de valor vetorial. Tais conceitos têm sido efetivamente usados para processar imagens coloridas e hiper-espectrais [73, 74]. Por exemplo, embora $(\bar{\mathbb{R}}^n, \leqslant)$ seja um reticulado completo, a ordem parcial $\leqslant$ definida não leva em consideração a possível relação entre as componentes do vetor, e isso pode resultar no chamado problema das "cores falsas" na morfologia multi-valorada [75]. Com isso surge a necessidade do uso de ordens parciais mais apropriadas.

Outras propriedades básicas da MM baseadas em reticulados podem ser encontradas em [8, 69].

3.2 Redes Neurais Morfológicas

Em termos gerais, redes neurais morfológicas (MNNs, *morphological neural networks*) são redes neurais cujas operações feitas nos neurônios são dadas por uma operações da morfologia matemática [16]. Os *perceptrons* morfológicos, introduzidos por Ritter e Sussner em meados de 1990 para tarefas de classificação binária, foram uma das primeiras MNNs [10]. Em poucas palavras, os *perceptrons* morfológicos são obtidos substituindo no *perceptron* de Rosenblatt a transformação afim usual $A(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, para todo $\mathbf{x} \in \mathbb{R}^n$, por um operador morfológico elementar de dilatação ou erosão.

Formalmente, os dois *perceptrons* morfológicos são dados pelas equações $y = f\big(\delta_{\mathbf{a}}(\mathbf{x})\big) \equiv f\delta_{\mathbf{a}}(\mathbf{x})$ e $y = f\big(\varepsilon_{\mathbf{b}}(\mathbf{x})\big) \equiv f\varepsilon_{\mathbf{b}}(\mathbf{x})$, para todo $\mathbf{x} \in \bar{\mathbb{R}}^n$, em que f denota uma função de ativação. Especificamente, o *perceptron* baseado em dilatação e o *perceptron* baseado em erosão são dados, respectivamente, por

$$y = f\left(\max_{j=1:n}\{a_j + x_j\}\right) \quad \text{e} \quad y = f\left(\min_{j=1:n}\{b_j + x_j\}\right), \quad \mathbf{x} \in \bar{\mathbb{R}}^n, \tag{3.9}$$

em que $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$

Para simplificar, neste trabalho consideraremos a função de ativação dos *perceptrons* morfológicos sendo a função sinal, ou seja,

$$f(x) = \begin{cases} +1, & x \geqslant 0 \\ -1, & x < 0. \end{cases}$$

Observe que um *perceptron* morfológico é dado pela composição $f\delta_{\mathbf{a}}$ ou pela composição $f\varepsilon_{\mathbf{b}}$, em que $\delta_{\mathbf{a}} : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ e $\varepsilon_{\mathbf{b}} : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ denotam, respectivamente, a dilatação e a erosão dadas em (3.3), para vetores $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$. Em consequência da operação de máximo, um *perceptron* baseado em dilatação dado por $y = f\delta_{\mathbf{a}}$ favorece a classe positiva cujo rótulo é $+1$. Dualmente, um *perceptron* baseado em erosão definido por $y = f\varepsilon_{\mathbf{b}}$ favorece a classe negativa, ou seja, o rótulo de classe -1, devido à operação de mínimo [32].

Em [25], Mondal e colaboradores propuseram a rede neural morfológica denominada rede morfológica densa (DMN, *dense morphological network*) cuja arquitetura se assemelha a um perceptron de camada única (SLP, *single hidden layer perceptron*). Cada neurônio da rede é composto por um operador morfológico dilatação ou erosão, sendo uma quantidade r_1 de neurônios definidos usando dilatação e uma quantidade r_2 de neurônios definidos utilizando erosão. Por fim, os neurônios de saída da rede são dados por uma combinação linear desses operadores, seguida de uma função de ativação. Para tarefas de classificação binária, a rede DMN é dada da seguinte forma:

$$y = g(\mathbf{x}) \coloneqq f\left(\sum_{i=1}^{r_1} t_i \delta_{\mathbf{a}_i}(\mathbf{x}) + \sum_{j=1}^{r_2} z_j \varepsilon_{\mathbf{b}_j}(\mathbf{x})\right), \tag{3.10}$$

com f uma função de ativação, $t_i, z_j \in \mathbb{R}$ os pesos da i-ésima dilatação e j-ésima erosão, respectivamente. O treinamento dos pesos da combinação e dos operadores morfológicos proposto pelos autores envolve o uso do método *backpropagation* combinado com o otimizador Adam aplicado à entropia cruzada binária, definida em (3.11), como função de perda e o uso da função softmax na última camada como função de ativação.

$$\mathcal{EC}(\mathcal{T}, g) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(g(\mathbf{x}_i)) + (1 - y_i)\log(1 - g(\mathbf{x}_i)), \quad y_i \in \{0, 1\}. \tag{3.11}$$

Recentemente, Sussner e Campiotti [19] propuseram uma nova rede neural morfológica híbrida que é equipada tanto com operadores morfológicos quanto neurônios clássicos. Esta rede generaliza o modelo perceptron de duas camadas, cuja arquitetura é dada por um perceptron multicamada (MLP) com uma única camada oculta de neurônios e duas camadas de pesos. A primeira camada oculta é composta por m neurônios morfológicos e l neurônios clássicos com função de ativação sigmoidal. Nos neurônios morfológicos são computados o ínfimo entre uma erosão e uma anti-dilatação. Para tarefas de classificação binária, podemos definir a rede da seguinte forma:

$$y \coloneqq \underset{c\in\{1,2\}}{\arg\max} \left\{ \sum_{i=1}^{m} u_i^c (\bar{\delta}_{\mathbf{a}_i}(\mathbf{x}) \wedge \varepsilon_{\mathbf{b}_i}(\mathbf{x})) + \sum_{j=1}^{l} v_j^c \sigma(\mathbf{x}^T \mathbf{w}_j + b_j) \right\}, \tag{3.12}$$
$$y \in \{1, 2\},$$

com $u_i^c, v_j^c \in \mathbb{R}$ os pesos dos c-ésimos neurônios da última camada oculta. Esta rede híbrida permitiu os autores treinarem a rede utilizando uma abordagem de aprendizado

extremo, também conhecida como máquinas de aprendizado extremo (ELM, *extreme learning machine*) [28]. Consequência disso, a rede híbrida foi nomeada como *perceptron* morfológico/linear hibrido (HMLP-EL, *hybrid morphological/linear* - EL).

Em resumo, no método de treinamento proposto para a HMLP-EL, os parâmetros $\mathbf{a}_i, \mathbf{b}_i, \mathbf{w}_j$ e b_j são tomados aleatoriamente e busca-se encontrar os parâmetros $\mathbf{z}^c = [\mathbf{u}^c, \mathbf{v}^c]^T \in \mathbb{R}^{m+l}$, com $c \in \{1, 2\}$, no caso de classificação binária. Esta busca envolve problemas de quadrados mínimos, mais especificamente, a resolução do problema $HZ \approx T$, com $Z = [\mathbf{z}^1, \mathbf{z}^2]^T$, $H = (\mathbf{h}_j)_j$ uma matriz com linhas dadas por

$$\mathbf{h}_j = [\bar{\delta}_{\mathbf{a}_1}(\mathbf{x}_j) \wedge \bar{\varepsilon}_{\mathbf{b}_1}(\mathbf{x}_j), \ldots, \bar{\delta}_{\mathbf{a}_m}(\mathbf{x}_j) \wedge \bar{\varepsilon}_{\mathbf{b}_m}(\mathbf{x}_j), \sigma(\mathbf{x}_j^T \mathbf{w}_1 + b_1), \ldots, \sigma(\mathbf{x}_j^T \mathbf{w}_l + b_l)],$$

$T = (\mathbf{t}_j)_j$, com $\mathbf{t}_j = [1; 0]$ se $\mathbf{x}_j$ for da primeira classe ou $\mathbf{t}_j = [0; 1]$ se $\mathbf{x}_j$ for da segunda classe, para cada $\mathbf{x}_j$ do conjunto de treino. A solução do problema $HZ \approx T$ proposta por Sussner e Campiotti faz uso de resoluções de equações normais com inclusão de um termo de regularização C. O tipo de problema é definido levando em consideração a relação entre quantidade de elementos no conjunto de treino (P) e a quantidade de neurônios ocultos da rede ($K = l + m$). Neste trabalho, iremos trabalhar com o caso $K < P$, ou seja, sempre teremos uma quantidade maior de dados do que de neurônios, buscando a resolução do sistema $(CI + H^T H)Z = T$.

Em [29], Araújo propôs uma MNN híbrida denominada *perceptron* dilatação-erosão (DEP, *dilation-erosion perceptron*). Para tarefas de classificação binária, um DEP calcula a combinação convexa de uma dilatação e uma erosão seguida por uma função de ativação. Intuitivamente, o modelo DEP permite um equilíbrio entre as duas classes. Em termos matemáticos, dado $\beta \in [0, 1]$, um modelo DEP é definido por

$$y = f(\tau(\mathbf{x})) \equiv f\tau(\mathbf{x}),$$

em que

$$\tau(\mathbf{x}) = \beta\delta_{\mathbf{a}}(\mathbf{x}) + (1 - \beta)\varepsilon_{\mathbf{b}}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n, \tag{3.13}$$

é a função de decisão do modelo DEP.

Em [30], Charisopoulos e Maragos propuseram um treinamento do modelo DEP de uma maneira semelhante ao treinamento das máquinas de vetores de suporte tradicionais. No treinamento dos operadores morfológicos foi utilizada a otimização DC para determinar a atribuição de peso ideal para um problema de classificação binária, fazendo uso de uma penalização de padrões com maiores chances de serem *outliers*.

Apesar da aplicação bem-sucedida do modelo DEP para predição de séries temporais [29], os classificadores DEP têm uma séria desvantagem: como um modelo baseado em reticulados, o classificador DEP pressupõe uma ordenação parcial no espaço de características, bem como no conjunto das classes. Para contornar este problema, Valle

recentemente propôs o chamado *perceptron* dilatação-erosão reduzido (r-DEP, *reduced dilation-erosion perceptron*) usando conceitos da morfologia matemática de valor vetorial [32]. O classificador r-DEP será definido a seguir.

Seja $\mathbb{V}$ o espaço de entrada e seja $\mathbb{C} = \{c_1, c_2\}$ o conjunto de rótulos de classes de uma tarefa de classificação binária. O espaço de entrada $\mathbb{V}$ é geralmente um subconjunto de $\mathbb{R}^n$, mas podemos considerar espaços de entrada abstratos. Além disso, considere o reticulado completo $\mathbb{L} = \bar{\mathbb{R}}^r$, equipado com a ordenação usual, e denote o espaço de classes positivas e negativas como $\mathbb{M} = \{-1, +1\}$. Dada uma correspondência bijetiva $\sigma : \mathbb{C} \to \mathbb{M}$ e um mapeamento sobrejetivo $\rho : \mathbb{V} \to \bar{\mathbb{R}}^r$, um modelo r-DEP é definido pela equação

$$y = \sigma^{-1}\big(f\big(\tau^r(\mathbf{x})\big)\big) \equiv \sigma^{-1} f \tau^r(\mathbf{x}),$$

em que $\tau^r : \mathbb{V} \to \mathbb{R}$ é a função de decisão dada pela seguinte combinação convexa para $\beta \in [0, 1]$:

$$\tau^r(\mathbf{x}) = \beta\delta_{\mathbf{a}}\big(\rho(\mathbf{x})\big) + (1 - \beta)\varepsilon_{\mathbf{b}}\big(\rho(\mathbf{x})\big), \quad \forall \mathbf{x} \in \mathbb{V}. \tag{3.14}$$

O diagrama apresentado na Figura 3.1 ilustra o comportamento das funções que definem os modelos DEP e r-DEP. A função $f\tau$ define o modelo DEP e a função $\sigma^{-1} f \tau^r$ o modelo r-DEP.

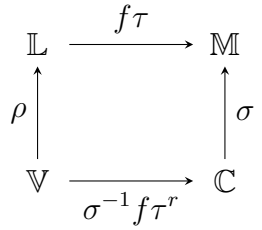


Figura 3.1 – Diagrama comutativo: modelos DEP e r-DEP.

O treinamento de um classificador r-DEP é feito da forma a seguir. Considere um conjunto de treinamento $\mathcal{T} = \{(\mathbf{x}_i, d_i) : i = 1, \ldots, m\} \subseteq \mathbb{V} \times \mathbb{C}$. A partir da equação (3.14), um r-DEP é definido simplesmente treinando um classificador DEP usando os dados de treinamento transformados $\mathcal{T}_r = \{(\rho(\mathbf{x}_i), \sigma(d_i)) : i = 1, \ldots, m\} \subseteq \mathbb{L} \times \mathbb{M}$.

Neste ponto, gostaríamos de lembrar que Charisopoulos e Maragos formularam o treinamento dos *perceptrons* morfológicos, bem como a rede neural morfológica DEP como solução de problemas de programação côncava-convexa (CCP) [30]. Os *perceptrons* baseados em erosão e em dilatação podem ser treinados utilizando o CCP [30, 33]. Combinando os resultados destes treinamentos, Valle apresentou uma forma de treinar os modelos DEP e posteriormente os modelos r-DEP. Treinar o classificador r-DEP usando procedimentos convexo-côncavos rendeu resultados encorajadores em tarefas de classificação [32].

O principal desafio na definição de um classificador r-DEP é como determinar o mapeamento sobrejetivo $\rho : \mathbb{V} \to \bar{\mathbb{R}}^r$. Ao utilizar o mapeamento ρ como transformação

linear, tem-se que ρ satisfaz $\rho(\mathbf{x}) = R\mathbf{x}$, com $R \in \mathbb{R}^{r\times n}$. Nossa proposta é utilizar transformações lineares distintas em cada um dos operadores morfológicos.

Dados mapeamentos lineares ρ_1 e ρ_2, a aplicação de mapeamentos lineares antes da avaliação de um mapeamento morfológico elementar produz

$$\tau^r(\mathbf{x}) = \beta\delta_{\mathbf{c}}\big(\rho_1(\mathbf{x})\big) + (1-\beta)\varepsilon_{\mathbf{d}}\big(\rho_2(\mathbf{x})\big), \quad \forall \mathbf{x} \in \mathbb{V}.$$

Observe que os mapeamentos lineares ρ_1 e ρ_2 satisfazem $\rho_1(\mathbf{x}) = R_1\mathbf{x}$ e $\rho_2(\mathbf{x}) = R_2\mathbf{x}$ para matrizes de valor real $R_1 \in \mathbb{R}^{r_1\times n}$ e $R_2 \in \mathbb{R}^{r_2\times n}$. Definindo $W = \beta R_1 \in \mathbb{R}^{r_1\times n}$, $M = (\beta-1)R_2 \in \mathbb{R}^{r_2\times n}$, $\mathbf{a} = \beta\mathbf{c}$ e $\mathbf{b} = (\beta-1)\mathbf{d}$, a função τ^r pode ser alternativamente escrita como:

$$\begin{aligned}
\tau^r(\mathbf{x}) &= \beta\delta_{\mathbf{c}}(R_1\mathbf{x}) + (1-\beta)\varepsilon_{\mathbf{d}}(R_2\mathbf{x}) \\
&= \beta\max\{R_1\mathbf{x}+\mathbf{c}\} + (1-\beta)\min\{R_2\mathbf{x}+\mathbf{d}\} \\
&= \beta\max\{R_1\mathbf{x}+\mathbf{c}\} + \min\{(1-\beta)R_2\mathbf{x} + (1-\beta)\mathbf{d}\} \\
&= \beta\max\{R_1\mathbf{x}+\mathbf{c}\} + \min\{-(\beta-1)R_2\mathbf{x} - (\beta-1)\mathbf{d}\} \\
&= \max\{\beta R_1\mathbf{x}+\beta\mathbf{c}\} - \max\{(\beta-1)R_2\mathbf{x} + (\beta-1)\mathbf{d}\} \\
&= \max\{W\mathbf{x}+\mathbf{a}\} - \max\{M\mathbf{x}+\mathbf{b}\} \\
&= \delta_{\mathbf{a}}(W\mathbf{x}) - \delta_{\mathbf{b}}(M\mathbf{x})
\end{aligned} \tag{3.15}$$

Definição 3.8. (Operador dilatação-erosão linear) *A função $\tau^\ell : \mathbb{R}^n \to \mathbb{R}$ definida por*

$$\tau^\ell(\mathbf{x}) = \delta_{\mathbf{a}}(W\mathbf{x}) - \delta_{\mathbf{b}}(M\mathbf{x}) \tag{3.16}$$

é denominada operador dilatação-erosão linear.

3.3 Propriedades do Operador Dilatação-Erosão Linear

As funções lineares são funções simples e de fácil implementação computacional. No entanto, o fato de uma única função linear não ser suficiente para obter uma boa aproximação de funções não lineares, a utilização de funções lineares por partes pode ser uma boa ferramenta para melhor aproximar tais funções.

Ao longo dos anos, vários pesquisadores têm buscado uma boa representação das funções contínuas lineares por partes [76–79]. Na literatura são encontrados alguns modelos que incluem funções lineares por partes. Estes modelos podem ser utilizados para uma melhor adaptação das aplicações dos estudos de pesquisadores [80, 81].

Assim como as redes neurais *feedforward*, as funções lineares por partes também podem ser utilizadas para aproximar funções não lineares e desconhecidas, para as quais apenas alguns pontos de amostras estão disponíveis [80]. Com efeito, essa ideia será utilizada para criarmos um modelo de regressão definido utilizando funções lineares por

partes a partir de amostras de treinamento existente, e, posteriormente para a criação de um modelo de classificação.

De modo a desenvolvermos uma teoria a respeito dos modelos que apresentaremos, faremos algumas definições que julgamos necessárias para compreensão do nosso modelo e abordaremos alguns resultados importantes na elaboração deste texto.

Primeiramente, iremos brevemente apresentar alguns conceitos básicos sobre aproximações de funções contínuas por funções contínuas lineares por partes [82], e também apresentaremos algumas representações destas funções.

Uma função afim $\ell : \mathbb{R}^n \to \mathbb{R}$ pode ser definida usando o conceito de hiperplano, com hiper-parâmetros dados por $\boldsymbol{\alpha} = (\mathbf{z}, z_0) \in \mathbb{R}^{n+1}$, em que $\ell(\mathbf{x}; \boldsymbol{\alpha}) = \mathbf{z}^T\mathbf{x} + z_0$. Além disso, tomando $W \in \mathbb{R}^{r_1 \times n}$, $M \in \mathbb{R}^{r_2 \times n}$, $\mathbf{a} \in \mathbb{R}^{r_1}$ e $\mathbf{b} \in \mathbb{R}^{r_2}$, tem-se que $W\mathbf{x} + \mathbf{a}$ e $M\mathbf{x} + \mathbf{b}$ descrevem respectivamente coleções de r_1 e r_2 funções afins.

Existem funções que podem ser representadas por uma coleção de hiperplanos, por exemplo, as funções lineares por partes que serão definidas a seguir. Propriedades sobre essa classe de funções podem ser encontradas em [83].

Definição 3.9. (Funções lineares por partes) *Sejam D um subconjunto convexo de $\mathbb{R}^n$, $g : D \to \mathbb{R}$ e $P = \{D_i \,; i = 1 : K\}$ uma partição de D. Dizemos que g é uma função linear por partes se existe um conjunto de funções $\{g_i\}_{i=1}^K$ tal que cada função $g_i : D_i \to \mathbb{R}$ é uma função afim e vale $g_i(\mathbf{x}) = g(\mathbf{x})$ para todo $\mathbf{x}$ em D_i. As funções $g_i : D_i \to \mathbb{R}$, $i = 1, \ldots K$ são chamadas componentes de g.*

Definição 3.10. (Funções contínuas lineares por partes) *Seja g uma função linear por partes. Se g for uma função contínua, então dizemos que g é uma função contínua linear por partes.*

Exemplo 3.5. *A função $g : \mathbb{R} \to \mathbb{R}$, definida por*

$$g(x) = \begin{cases} 0, & x < 0, \\ x, & x \geqslant 0, \end{cases}$$

é uma função linear por partes com funções componentes dadas por $g_1(x) = 0$ e $g_2(x) = x$, definidas em $D_1 = (-\infty, 0)$ e $D_2 = [0, \infty)$, respectivamente.

Exemplo 3.6. *A função $g : \mathbb{R}^2 \to \mathbb{R}$ definida por*

$$g(x_1, x_2) = \begin{cases} -x_1 - x_2, & x_1 < 0, \\ +x_1 - x_2, & x_1 \geqslant 0, \end{cases}$$

é uma função contínua linear por partes, com funções componentes dadas por $g_1(x_1, x_2) = -x_1 - x_2$ e $g_2(x_1, x_2) = x_1 - x_2$, definidas em $D_1 = (-\infty, 0) \times \mathbb{R}$ e $D_2 = [0, \infty) \times \mathbb{R}$, respectivamente.

O seguinte resultado é uma consequência do Teorema da Aproximação de Stone-Weierstrass [64]. Ele garante que qualquer função contínua definida num compacto pode ser aproximada por uma função contínua linear por partes.

Teorema 3.1 ([21] Proposição 4.2)**.** *Seja D um subconjunto compacto de $\mathbb{R}^n$. Dada uma função contínua $f : D \to \mathbb{R}$ e dado $\epsilon > 0$, então existe uma função contínua linear por partes $\ell : D \to \mathbb{R}$ tal que, para todo $\mathbf{x} \in D$, vale $|f(\mathbf{x}) - \ell(\mathbf{x})| < \epsilon$.*

Assim como muitos resultados matemáticos, o teorema anterior garante apenas existência de uma função contínua linear por partes que é uma boa aproximação para uma função contínua, definida em um compacto. No entanto, tal resultado não fornece uma caracterização para essas funções. Nesta seção, apresentaremos uma caracterização de funções contínuas lineares por partes utilizando conceitos da morfologia matemática definida em reticulados completos.

O próximo teorema, apresentado em [34] e generalizado em [84], nos fornece uma caracterização de funções contínuas lineares por partes [76], usando a teoria de reticulados [78].

Teorema 3.2. *Sejam $D \subseteq \mathbb{R}^n$ um conjunto compacto, $\ell : D \to \mathbb{R}$ uma função contínua linear por partes. Então, existem dois conjuntos de hiper-parâmetros $\{(\mathbf{w}_i, a_i)\}_{i=1}^{r_1}$, $\{(\mathbf{m}_i, b_i)\}_{i=1}^{r_2} \subset \mathbb{R}^{n+1}$ de modo que*

$$\ell(\mathbf{x}) = \max_{i=1,\dots,r_1} \{\mathbf{w}_i^T \mathbf{x} + a_i\} - \max_{i=1,\dots,r_2} \{\mathbf{m}_i^T \mathbf{x} + b_i\} \tag{3.17}$$

O resultado apresentado no Teorema 3.2 é fundamental neste trabalho, pois nos garante que qualquer função contínua linear por partes pode ser representada por uma diferença entre duas funções convexas. Mais ainda, tais funções convexas são definidas utilizando operadores elementares da morfologia matemática. De fato, podemos reescrever ℓ da seguinte forma:

$$\begin{aligned} \ell(\mathbf{x}) &= \max_{i=1,\dots,r_1} \{\mathbf{w}_i^T \mathbf{x} + a_i\} - \max_{i=1,\dots,r_2} \{\mathbf{m}_i^T \mathbf{x} + b_i\} \\ &= \max\{W\mathbf{x} + \mathbf{a}\} - \max\{M\mathbf{x} + \mathbf{b}\} \\ &= \delta_{\mathbf{a}}(W\mathbf{x}) - \delta_{\mathbf{b}}(M\mathbf{x}), \end{aligned} \tag{3.18}$$

com $\mathbf{a} = [a_1 \dots a_{r_1}]^T \in \mathbb{R}^{r_1}$ e $\mathbf{b} = [b_1 \dots b_{r_2}]^T \in \mathbb{R}^{r_2}$, e $\mathbf{w}_i^T$ e $\mathbf{m}_i^T$ as linhas de $W \in \mathbb{R}^{r_1 \times n}$ e $M \in \mathbb{R}^{r_2 \times n}$, respectivamente.

Outras representações para funções lineares por partes podem ser encontradas em [34, 85, 86].

3.4 Operador Dilatação-Erosão Linear como Aproximador Universal

Pelos Teoremas 3.1 e 3.2, concluímos que o operador dilatação-erosão é contínuo linear por partes, τ^ℓ definido em (3.16) é um aproximador universal e, consequentemente, pode aproximar qualquer função contínua definida em um conjunto compacto, bem como pode ser usado para resolver problemas de ML supervisionado.

Exemplo 3.7. *A função do Exemplo 3.6 pode ser reescrita usando o operador τ^ℓ cujos parâmetros são*

$$W = \begin{bmatrix} 0 & -0.5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \end{bmatrix},$$
$$M = \begin{bmatrix} 1 & 0.5 \\ -1 & 0.5 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

De fato, para $\mathbf{x} = (x_1, x_2)$ tem-se

$$\begin{aligned} \max\{W\mathbf{x} + \mathbf{b}\} &= -0.5x_2 \\ \max\{M\mathbf{x} + \mathbf{c}\} &= \max\{x_1 + 0.5x_2, -x_1 + 0.5x_2\} \\ &= \max\{x_1, -x_1\} + 0.5x_2 \\ &= \begin{cases} x_1 + 0.5x_2, & x_1 \leqslant 0 \\ 0.5x_2 - x_1, & x_1 \geqslant 0 \end{cases} \end{aligned}$$

Logo,

$$\begin{aligned} \tau^\ell(\mathbf{x}) &= \max\{W^T\mathbf{x} + \mathbf{b}\} - \max\{M^T\mathbf{x} + \mathbf{c}\} \\ &= \begin{cases} -x_1 - x_2, & x_1 \leqslant 0 \\ -x_2 + x_1, & x_1 \geqslant 0 \end{cases} \end{aligned}$$

Neste caso, os valores para r_1 e r_2 são 1 e 2, respectivamente.

3.5 Rede Neural Morfológica Perceptron Dilatação-Erosão Linear

Do ponto de vista das redes neurais, a Figura 3.2 representa o operador morfológico τ^ℓ como uma rede neural morfológica híbrida em que a saída é definida por

$$y = f\left(\max_{i=1,\ldots,r_1}(\bar{\mathbf{x}}^T\mathbf{u}_i) - \max_{i=1,\ldots,r_1}(\bar{\mathbf{x}}^T\mathbf{v}_i)\right),$$

em que $\bar{\mathbf{x}} = (\mathbf{x}, 1)$ denota os valores de entrada, $\mathbf{u}_i = (\mathbf{w}_i, a_i)$ e $\mathbf{v}_i = (\mathbf{m}_i, b_i)$ os parâmetros ajustáveis e f é a função de ativação. Curiosamente, a rede apresentada na Figura 3.2 aplicada a problemas de classificação, com $r_1 = r_2$, possui uma arquitetura equivalente às redes maxout investigadas por Goodfellow em [21]. Observação semelhante foi feita por Charisopoulos em [30]

Assim como as redes neurais *feedforward*, as redes neurais com estruturas representadas pela rede da Figura 3.2 são aproximadores universais, ou seja, elas podem aproximar uma função contínua dentro de qualquer precisão desejada em uma região compacta de um espaço euclidiano. Neste sentido, podemos resolver modelos de aprendizado de máquinas cuja função de decisão é o operador dilatação-erosão linear.

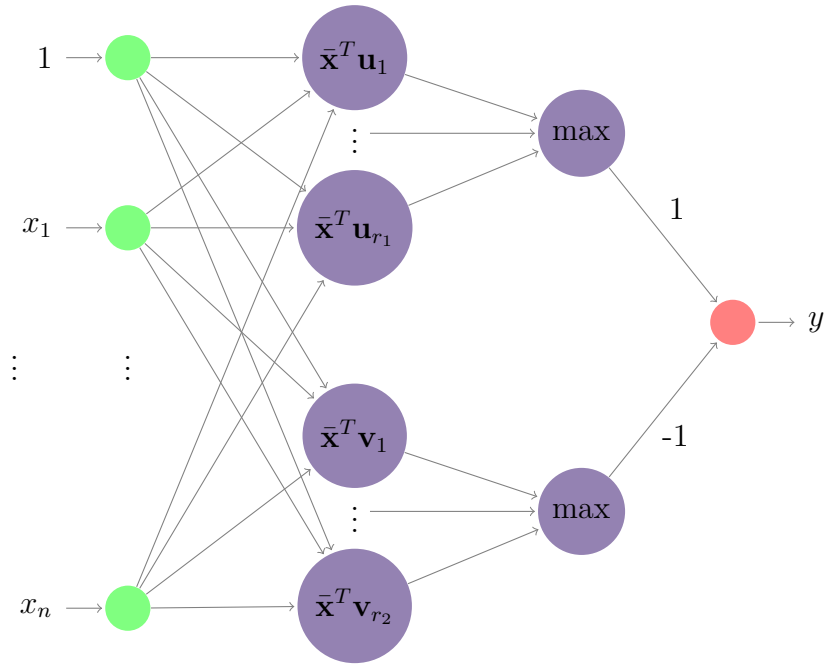


Figura 3.2 – Rede Neural Morfológica ℓ-DEP.

Desta forma, a rede neural morfológica apresentada na Figura 3.2 pode ser aplicada tanto em problemas de regressão quanto de classificação. Os operadores dilatação e erosão que definem os *perceptrons* morfológicos são funções convexas e côncavas, respectivamente. Equivalentemente, o operador dilatação-erosão linear é uma função DC. Logo é possível utilizar o procedimento CCP no treinamento de redes neurais cuja função de decisão é o operador dilatação-erosão linear. Contudo, deve-se realizar uma escolha apropriada da função perda para o tipo de tarefa considerado.

Nos próximos capítulos apresentaremos aplicações do *Perceptron* Dilatação-Erosão Linear (ℓ-DEP, *linear dilation-erosion perceptron*) em tarefas de regressão e também de classificação. Também apresentaremos algumas abordagens de treinamento para esses modelos.

4 Perceptron Dilatação-Erosão Linear Aplicado a Tarefas de Regressão

Nas seções anteriores, como consequência do Teorema 3.2, qualquer função contínua pode ser aproximada arbitrariamente em um compacto pelo operador dilatação-erosão linear τ^ℓ definido em (3.16). Vimos também que as funções lineares por partes podem ser utilizadas para aproximar funções não lineares e desconhecidas, para as quais apenas alguns pontos de amostras estão disponíveis. Como o operador τ^ℓ dado em (3.16) é um aproximador universal, então τ^ℓ também pode ser usado em um modelo preditivo para tarefas de regressão. Em outras palavras, é possível usar o τ^ℓ como função de previsão que mapeia um conjunto de variáveis independentes em $\mathbb{R}^n$ a uma variável dependente em $\mathbb{R}$. Neste caso, nos referimos a $\tau^\ell : \mathbb{R}^n \to \mathbb{R}$ dado por (3.16) como um regressor dilatação-erosão linear (ℓ-DER). A formulação do modelo de regressão ℓ-DER e seu processo de treinamento serão apresentados na seção a seguir.

Neste capítulo serão apresentadas algumas abordagens distintas para treinar um modelo ℓ-DER usando o conjunto de treinamento $\mathcal{T} = \{(\mathbf{x}_i, y_i); i = 1, \ldots, m\} \subset \mathbb{R}^n \times \mathbb{R}$. Precisamente, o objetivo é encontrar os parâmetros de um modelo ℓ-DER de forma que a estimativa $\tau^\ell(\mathbf{x}_i)$ se aproxime da saída desejada y_i de acordo com alguma função perda.

Os parâmetros $(\mathbf{w}_i^T, a_i) \in \mathbb{R}^{n+1}$ e $(\mathbf{m}_j^T, b_j) \in \mathbb{R}^{n+1}$, para $i = 1, \ldots, r_1$ e $j = 1, \ldots, r_2$, que definem τ^ℓ são determinados usando o conjunto de treinamento, minimizando a soma dos quadrados das diferenças entre os valores previstos e desejados, ou seja, o erro quadrático médio (MSE, *mean squared error*). Os parâmetros dos regressores ℓ-DER são as matrizes $W \in \mathbb{R}^{r_1 \times n}$ e $M \in \mathbb{R}^{r_2 \times n}$ e vetores $\mathbf{a} = (a_1, \ldots, a_{r_1}) \in \mathbb{R}^{r_1}$ e $\mathbf{b} = (b_1, \ldots, b_{r_2}) \in \mathbb{R}^{r_2}$. Para simplificar a exposição, os parâmetros do ℓ-DER também são arranjados em um vetor

$$\boldsymbol{\alpha} = (\mathbf{w}_1^T, a_1, \ldots, \mathbf{w}_{r_1}^T, a_{r_1}, \mathbf{m}_1^T, b_1, \ldots, \mathbf{m}_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)}, \tag{4.1}$$

em que $\mathbf{w}_i^T$ e $\mathbf{m}_i^T$ são as linhas de W e de M, respectivamente. Durante o treinamento, a função τ^ℓ do modelo ℓ-DER é interpretada como uma função de seus parâmetros, ou seja, $\tau^\ell(\mathbf{x}) \equiv \tau^\ell(\mathbf{x}; \boldsymbol{\alpha})$.

Neste trabalho, o erro quadrático médio (MSE), amplamente utilizado nesta seção, é definido a seguir usando o conjunto de treinamento $\mathcal{T}$:

$$MSE(\mathcal{T}, \boldsymbol{\alpha}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \tau^\ell(\mathbf{x}_i; \boldsymbol{\alpha}))^2, \tag{4.2}$$

e é considerado como a função perda. Como consequência, os parâmetros do ℓ-DER podem

ser determinados minimizando $MSE(\mathcal{T}, \boldsymbol{\alpha})$, ou seja, resolvendo o problema de otimização:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^{(r_1+r_2)(n+1)}}{\text{minimize}} \frac{1}{m}\sum_{i=1}^{m}(y_i - \tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}))^2. \tag{4.3}$$

A seguir, apresentaremos duas abordagens para resolver o Problema (4.3). A primeira abordagem é uma releitura do que foi apresentado em [87], que utiliza o DCA. A segunda abordagem, parte das nossas contribuições, usa o procedimento côncavo-convexo para resolver o Problema (4.3). Estas abordagens têm forte embasamento na Otimização DC e serão descritas detalhadamente na próxima seção.

Para fins de reprodutibilidade, incluímos na seção de testes computacionais todos os parâmetros utilizados em nossos testes.

4.1 Releitura do Algoritmo Diferença de Convexa: ℓ-DCA

A abordagem de treinamento baseada no DCA é uma releitura do método apresentado em [87]. Começamos notando que τ^{ℓ}, o operador dilatação-erosão linear definido em (3.16), é uma função DC, pois $\delta_{\mathbf{a}}(W\mathbf{x})$ e $\delta_{\mathbf{b}}(M\mathbf{x})$ são ambas funções convexas. É possível escrever o quadrado da diferença $\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i$ como uma função DC para todo $i = 1, \ldots, m$.

Devido ao fato de existirem infinitas decomposições DC para uma única função DC, é possível encontrar uma função convexa ϕ_i tal que $(\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i)^2$ pode ser escrito como uma função DC utilizando as propriedades apresentadas nas proposições da Seção 1.1. De fato, é possível escrever $\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i$ como a seguinte diferença de funções

$$\begin{aligned}\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i &= \tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i + \phi_i(\boldsymbol{\alpha}) - \phi_i(\boldsymbol{\alpha}) \\ &= [\delta_{\mathbf{a}}(W\mathbf{x}_i) - y_i + \phi_i(\boldsymbol{\alpha})] - [\phi_i(\boldsymbol{\alpha}) + \delta_{\mathbf{b}}(M\mathbf{x}_i)] \\ &= g_i(\boldsymbol{\alpha}) - h_i(\boldsymbol{\alpha}),\end{aligned}$$

para qualquer função ϕ_i definida no domínio de τ^{ℓ}. Daí, se $g_i(\boldsymbol{\alpha})$ e $h_i(\boldsymbol{\alpha})$ forem funções convexas e não negativas, então é possível escrever $(\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i)^2$ como uma função DC, conforme apresentado em (4.4). Com efeito, pela proposição 1.5, temos

$$\begin{aligned}(\tau^{\ell}(\mathbf{x}_i;\boldsymbol{\alpha}) - y_i)^2 &= 2\big((\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}))^2 + (\phi_i(\boldsymbol{\alpha}) + \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}))^2\big) \\ &\quad - \big(\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + 2\phi_i(\boldsymbol{\alpha})\big)^2 \\ &= G_i(\boldsymbol{\alpha}) - H_i(\boldsymbol{\alpha})\end{aligned} \tag{4.4}$$

em que $\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) = \delta_{\mathbf{a}}(W\mathbf{x}_i) - y_i$ e $\tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) = \delta_{\mathbf{b}}(M\mathbf{x}_i)$ são funções convexas e

$$G_i(\boldsymbol{\alpha}) = 2\big((\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}))^2 + (\phi_i(\boldsymbol{\alpha}) + \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}))^2\big)$$

e

$$H_i(\boldsymbol{\alpha}) = \big(\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + 2\phi_i(\boldsymbol{\alpha})\big)^2,$$

também são funções convexas.

Tomando $\phi_i(\boldsymbol{\alpha}) = \max\{-\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^1, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle, -\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^2, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle\}$, para um vetor arbitrário $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^{(r_1+r_2)(n+1)}$ e $\boldsymbol{\beta}_i^j \in \partial\tau_j(\mathbf{x}_i : \bar{\boldsymbol{\alpha}})$, $j = 1, 2$, então a não negatividade das funções g_i e h_i é obtida. De fato, pela definição de vetor subgradiente, para todo $\bar{\boldsymbol{\alpha}}$ valem as seguintes implicações:

$$\boldsymbol{\beta}_i^1 \in \partial\big(\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}})\big) \Longrightarrow \tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) - \tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_1^i, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle \geqslant 0$$

e

$$\boldsymbol{\beta}_i^2 \in \partial\big(\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}})\big) \Longrightarrow \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) - \tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^2, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle \geqslant 0.$$

Com isso,

$$\begin{aligned} g_i(\boldsymbol{\alpha}) &= \tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}) \\ &= \tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \max\{-\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^1, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle, -\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^2, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle\} \\ &\geqslant \tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) - \tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^1, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle \\ &\geqslant 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^{(r_1+r_2)(n+1)} \end{aligned}$$

e

$$\begin{aligned} h_i(\boldsymbol{\alpha}) &= \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}) \\ &= \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + \max\{-\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^1, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle, -\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^2, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle\} \\ &\geqslant \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) - \tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle \boldsymbol{\beta}_i^2, \boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\rangle \\ &\geqslant 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^{(r_1+r_2)(n+1)} \end{aligned}$$

Assim, o erro quadrático médio dado em (4.2) admite uma decomposição DC dada por $MSE(\mathcal{T}, \boldsymbol{\alpha}) = G(\boldsymbol{\alpha}) - H(\boldsymbol{\alpha})$, em que

$$G(\boldsymbol{\alpha}) = \frac{2}{m}\sum_{i=1}^{m}\big[(\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}))^2 + (\tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + \phi_i(\boldsymbol{\alpha}))^2\big]$$

e

$$H(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^{m}[\tau_1(\mathbf{x}_i;\boldsymbol{\alpha}) + \tau_2(\mathbf{x}_i;\boldsymbol{\alpha}) + 2\phi_i(\boldsymbol{\alpha})]^2.$$

Como $MSE(\mathcal{T}, \alpha)$ é uma função DC, o DCA pode ser usado para resolver o Problema (4.3).

Como apontado no Capítulo 1, um dos desafios do algoritmo DCA é encontrar os elementos $\boldsymbol{\beta}_t \in \partial H(\boldsymbol{\alpha}_t)$ e $\boldsymbol{\alpha}_{t+1} \in \partial G^*(\boldsymbol{\beta}_t)$ em cada iteração t. Para treinar o ℓ-DER, em particular, a determinação desses elementos será apresentado a seguir.

Pelas propriedades de subgradientes, tem-se que

$$\begin{aligned}\partial H &= \frac{1}{m}\sum_{i=1}^{m}\partial[(\tau_1+\tau_2+2\phi_i)^2]\\ &= \frac{2}{m}\sum_{i=1}^{m}(\tau_1+\tau_2+2\phi_i)(\partial\tau_1+\partial\tau_2+2\partial\phi_i).\end{aligned}$$

Para cada $i=1,\ldots,m$, tome

$$j_{i1} = \underset{j=1,\ldots,r_1}{\arg\max}\{\mathbf{w}_j^T\mathbf{x}_i + a_j\} \quad \text{e} \quad j_{i2} = \underset{j=1,\ldots,r_2}{\arg\max}\{\mathbf{m}_j^T\mathbf{x}_i + b_j\}, \tag{4.5}$$

e defina $\mathbf{v}^{i,s} = (\mathbf{v}_1^{i,s},\ldots,\mathbf{v}_{r_1+r_2}^{i,s}) \in \mathbb{R}^{(r_1+r_2)(n+1)}$ para os índices $i=1,\ldots,m$ e $s=1,\ldots,r_1+r_2$, em que

$$\mathbf{v}_k^{i,s} = \begin{cases}(\mathbf{x}_i,1), & k=s,\\ (\mathbf{0},0), & \text{c.c.},\end{cases} \tag{4.6}$$

com $k=1,\ldots,r_1+r_2$, $\mathbf{0}=(0,\ldots,0)\in\mathbb{R}^n$ e $\mathbf{v}^i = \mathbf{v}^{i,j_{i1}} - \mathbf{v}^{i,r_1+j_{i2}}$. Então, para $\bar{\boldsymbol{\alpha}}$ fixo, tem-se

$$\partial\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) = \mathbf{v}^{i,j_{i1}} \quad \text{e} \quad \partial\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) = \mathbf{v}^{i,r_1+j_{i2}}.$$

Para o cálculo de $\partial\phi_i$, considere

$$\phi_i^j(\boldsymbol{\alpha}) = -\tau_j(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \langle\boldsymbol{\beta}_i^j, \boldsymbol{\alpha}-\bar{\boldsymbol{\alpha}}\rangle, \quad \boldsymbol{\beta}_i^j \in \partial\tau_j(\mathbf{x}_i:\bar{\boldsymbol{\alpha}}), \quad j=1,2,$$

de modo que $\phi_i(\boldsymbol{\alpha}) = \max\limits_{j=1,2}\{\phi_i^j(\boldsymbol{\alpha})\}$. Suponha que $\phi_i(\boldsymbol{\alpha}) = \phi_i^s(\boldsymbol{\alpha})$, para algum $s\in\{1,2\}$, então

$$\partial\phi_i(\boldsymbol{\alpha}) = \partial\phi_i^s(\boldsymbol{\alpha}) = -\boldsymbol{\beta}_i^s \in \partial\tau_s(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}),$$

e, portanto, $\partial\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) + \partial\tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) + 2\partial\phi_i(\bar{\boldsymbol{\alpha}}) = \mathbf{v}^{i,j_{i1}} + \mathbf{v}^{i,r_1+j_{i2}} - 2\boldsymbol{\beta}_i^s$. Logo,

$$\partial H(\bar{\boldsymbol{\alpha}}) = \frac{2}{m}\sum_{i=1}^{m}(\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) + \tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) + 2\phi_i^s(\bar{\boldsymbol{\alpha}}))\big(\mathbf{v}^{i,j_{i1}} + \mathbf{v}^{i,r_1+j_{i2}} - 2\boldsymbol{\beta}_i^s\big). \tag{4.7}$$

Escolhendo $\boldsymbol{\alpha}_t = \bar{\boldsymbol{\alpha}}$ na iteração t, como $\boldsymbol{\beta}_i^s \in \partial\tau_s(\mathbf{x}_i;\bar{\boldsymbol{\alpha}})$, podemos escolher o subgradiente $\boldsymbol{\beta}_t \in \partial H(\boldsymbol{\alpha}_t)$ da seguinte forma,

$$\begin{aligned}\boldsymbol{\beta}_t &= \frac{2}{m}\sum_{i=1}^{m}(\tau_1(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}) - \tau_2(\mathbf{x}_i;\bar{\boldsymbol{\alpha}}))\big(\mathbf{v}^{i,j_{i1}} - \mathbf{v}^{i,r_1+j_{i2}}\big)\\ &= \frac{2}{m}\sum_{i=1}^{m}(\tau^\ell(\mathbf{x}_i;\boldsymbol{\alpha}_t) - y_i)\mathbf{v}^i \quad \in \mathbb{R}^{(r_1+r_2)(n+1)},\end{aligned} \tag{4.8}$$

em que $\mathbf{v}^i = \mathbf{v}^{i,j_{i1}} - \mathbf{v}^{i,r_1+j_{i2}}$, $i=1,\ldots,m$.

Das propriedades das funções conjugadas, $\boldsymbol{\alpha}_{t+1} \in \partial G^*(\boldsymbol{\beta}_t)$ é obtido resolvendo o Problema (4.9), que é um problema de otimização quadrática derivado do Problema (1.13).

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\mathbf{p},\mathbf{q}}{\text{minimize}} \quad & 2\|\mathbf{q}\|_2^2 + 2\|\mathbf{p}\|_2^2 - \langle \boldsymbol{\beta}_t, \boldsymbol{\alpha} \rangle \\
\text{s.a.} \quad & \langle \mathbf{v}^{i,l} - \mathbf{v}^{i,j_{i1}}, \boldsymbol{\alpha} \rangle \leqslant q_i, && l = 1,\ldots,r_1, \quad i = 1,\ldots,m, \\
& \langle \mathbf{v}^{i,l} - \mathbf{v}^{i,r_1+j_{i2}}, \boldsymbol{\alpha} \rangle \leqslant q_i + y_i, && l = 1,\ldots,r_1, \quad i = 1,\ldots,m, \\
& \langle \mathbf{v}^{i,r_1+l} - \mathbf{v}^{i,j_{i1}}, \boldsymbol{\alpha} \rangle \leqslant p_i - y_i, && l = 1,\ldots,r_2, \quad i = 1,\ldots,m, \\
& \langle \mathbf{v}^{i,r_1+l} - \mathbf{v}^{i,r_1+j_{i2}}, \boldsymbol{\alpha} \rangle \leqslant p_i, && l = 1,\ldots,r_2, \quad i = 1,\ldots,m.
\end{aligned} \tag{4.9}$$

Embora o vetor $\boldsymbol{\beta}_t$ seja utilizado para encontrar uma solução do Problema (4.9) em cada iteração, para nosso modelo, na última iteração nos interessa apenas a solução primal $\boldsymbol{\alpha}$ do Algoritmo 1. Isto é, os parâmetros que desejamos obter, apresentados em (4.1), e que definem o modelo ℓ-DER, compõe a solução primal $\boldsymbol{\alpha}$ do problema da última iteração do algoritmo.

O Problema 4.9 pode ser reescrito, utilizando a esparsidade dos elementos $\mathbf{v}^i$, conforme o problema de otimização quadrática com restrições lineares dado pelo Problema (4.10):

$$\begin{aligned}
\underset{W,\mathbf{a},M,\mathbf{b},\mathbf{p}.\mathbf{q}}{\text{minimize}} \quad & 2\|\mathbf{q}\|_2^2 + 2\|\mathbf{p}\|_2^2 - 2\sum_{i=1}^{m} \langle \mathbf{x}_i, \mathbf{w}_{j_{i1}} - \mathbf{m}_{j_{i2}} \rangle e_i \\
\text{s.a.} \quad & \langle \mathbf{x}_i, \mathbf{w}_l - \mathbf{w}_{j_{i1}} \rangle + a_l - a_{j_{i1}} \leqslant q_i, && l = 1,\ldots,r_1, \quad i = 1,\ldots,m, \\
& \langle \mathbf{x}_i, \mathbf{w}_l - \mathbf{m}_{j_{i2}} \rangle + a_l - b_{j_{i2}} \leqslant q_i + y_i, && l = 1,\ldots,r_1, \quad i = 1,\ldots,m, \\
& \langle \mathbf{x}_i, \mathbf{m}_l - \mathbf{w}_{j_{i1}} \rangle + b_l - a_{j_{i1}} \leqslant p_i - y_i, && l = 1,\ldots,r_2, \quad i = 1,\ldots,m, \\
& \langle \mathbf{x}_i, \mathbf{m}_l - \mathbf{m}_{j_{i2}} \rangle + b_l - b_{j_{i2}} \leqslant p_i, && l = 1,\ldots,r_2, \quad i = 1,\ldots,m,
\end{aligned} \tag{4.10}$$

com e_i sendo o erro da iteração t dado por

$$e_i := \tau^{\ell}(\mathbf{x}_i) - y_i = \tau^{\ell}(\mathbf{x}_i; \boldsymbol{\alpha}_t) - y_i, \quad i = 1,\ldots,m. \tag{4.11}$$

De modo a apresentar o critério de parada usado para treinar o ℓ-DER com abordagem DCA, considere em cada iteração t o valor tol_t definido em (4.12). O critério de parada é atingido quando $tol_t \leqslant \epsilon$, para algum $\epsilon > 0$, definido a priori, ou se um número máximo de iterações $t_{\max}$ for atingido.

$$tol_t := \frac{|MSE(\mathcal{T}, \boldsymbol{\alpha}_{t+1}) - MSE(\mathcal{T}, \boldsymbol{\alpha}_t)|}{1 + MSE(\mathcal{T}, \boldsymbol{\alpha}_t)} \tag{4.12}$$

Concluindo, neste trabalho o treinamento do modelo ℓ-DER com a abordagem DCA é feito usando o Algoritmo 1, que é reescrito conforme o Algoritmo 3.

Algoritmo 3: ℓ-DCA: regressão

Entrada: $\mathcal{T}, \epsilon, t_{\max}, r_1, r_2$
Saída: $(W, \mathbf{a}, M, \mathbf{b})$
Inicializar: $(W_0, \mathbf{a}_0, M_0, \mathbf{b}_0), t = 0$
repita
  Compute j_{i1} e j_{i2}, $i = 1, \ldots, m$ conforme (4.5)
  Compute o erro e_i^t conforme (4.11)
  Compute $(W, \mathbf{a}, M, \mathbf{b})$, solução de (4.10)
  Compute tol_t conforme (4.12)
  Incremente $t = t + 1$
até $tol_t \leqslant \epsilon$ *ou* $t_{\max} \leqslant t$;
retorna $(W, \mathbf{a}, M, \mathbf{b})$

4.2 Treinamento Baseado em Programação Côncava-Convexa: ℓ-CCP

Na seção anterior foi feito uma releitura do método DCA aplicado ao problema de ajuste de curva cuja função é contínua linear por partes. Especificamente, apresentamos de forma detalhada como o algoritmo DCA pode ser aplicado para ajustar funções contínuas lineares por partes que podem ser escritas como funções DC. Como consequência, o modelo ℓ-DEP pode ser treinado utilizando essa técnica apresentada por Ho e colaboradores em [87] aplicada na resolução do problema de otimização irrestrita (4.3).

Inspirados na metodologia desenvolvida por Charisopoulos e Maragos para treinar *perceptrons* morfológicos [30], propomos agora a reformulação do problema de otimização irrestrita (4.3) como um problema DC restrito. Precisamente, fazendo $\xi_i = y_i - \tau^\ell(\mathbf{x}_i)$, para $i = 1, \ldots, m$, observamos, da definição do $MSE(\mathcal{T}, \boldsymbol{\alpha})$, o problema irrestrito (4.3) corresponde a minimizar $\dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \xi_i^2$ sujeito às restrições $\xi_i = y_i - \tau^\ell(\mathbf{x}_i)$ para todo $i = 1, \ldots, m$. Em outras palavras, o ℓ-DER pode ser treinado resolvendo o seguinte problema:

$$\begin{cases} \underset{W,\mathbf{a},M,\mathbf{b},\boldsymbol{\xi}}{\text{minimize}} & \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \xi_i^2 \\ \text{s.a.} & \delta_{\mathbf{a}}(W\mathbf{x}_i) + \xi_i = \delta_{\mathbf{b}}(M\mathbf{x}_i) + y_i, \quad i = 1, \ldots, m. \end{cases} \tag{4.13}$$

Equivalentemente, podemos transformar cada uma das restrições de igualdade em duas restrições de desigualdades, resultando no problema restrito a seguir

$$\begin{cases} \underset{W,\mathbf{a},M,\mathbf{b},\boldsymbol{\xi}}{\text{minimize}} & \dfrac{1}{m}\displaystyle\sum_{i=1}^{m} \xi_i^2 \\ \text{s.a.} & \max\limits_{l=1,\ldots,r_1} \{\mathbf{w}_l^T \mathbf{x}_i + a_l\} + \xi_i \leqslant \max\limits_{j=1,\ldots,r_2} \{\mathbf{m}_j^T \mathbf{x}_i + b_j\} + y_i, \quad i = 1, \ldots, m. \\ & \max\limits_{l=1,\ldots,r_2} \{\mathbf{m}_l^T \mathbf{x}_i + b_l\} - \xi_i \leqslant \max\limits_{j=1,\ldots,r_1} \{\mathbf{w}_j^T \mathbf{x}_i + a_j\} - y_i, \quad i = 1, \ldots, m. \end{cases} \tag{4.14}$$

Mais ainda, observe que, se o máximo entre um conjunto de elementos for menor ou igual a um valor, então cada elemento deste conjunto será menor ou igual ao majorante. Com base nisso, o problema acima pode ainda ser reformulado conforme o Problema (4.15) dado por

$$\begin{aligned} \underset{W,\mathbf{a},M,\mathbf{b},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m}\xi_i^2 \\ \text{s.a.} \quad & \mathbf{w}_l^T\mathbf{x}_i + a_l + \xi_i \leqslant \max_{j=1,\ldots,r_2}\{\mathbf{m}_j^T\mathbf{x}_i + b_j\} + y_i, \quad i = 1,\ldots,m, \quad l = 1,\ldots,r_1 \\ & \mathbf{m}_l^T\mathbf{x}_i + b_l - \xi_i \leqslant \max_{j=1,\ldots,r_1}\{\mathbf{w}_j^T\mathbf{x}_i + a_j\} - y_i, \quad i = 1,\ldots,m, \quad l = 1,\ldots,r_2 \end{aligned} \tag{4.15}$$

Observe que a função objetivo em (4.15) é uma função quadrática convexa, portanto DC. Além disso, as funções em ambos os lados das restrições têm concavidade conhecida, podendo ser consideradas ambas funções convexas, já que funções afins são tanto convexas quanto côncavas. Portanto, o problema de otimização (4.15) pode ser resolvido usando o Algoritmo 2.

As funções convexas do lado direito das desigualdades são funções que dependem apenas de dois parâmetros, ou de $(M, \mathbf{b})$ ou de $(W, \mathbf{a})$. Desta forma, em cada caso, no Algoritmo CCP, as funções h_i dependem apenas de $\boldsymbol{\alpha}$ e as funções g_i dependem de $\boldsymbol{\alpha}$ e $\boldsymbol{\xi}$, para $i = 1,\ldots,m$ com

$$\begin{aligned} &\boldsymbol{\alpha} = (\mathbf{w}_1^T, a_1, \ldots, \mathbf{w}_{r_1}^T, a_{r_1}, \mathbf{m}_1^T, b_1, \ldots, \mathbf{m}_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)} \\ &\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m) \in \mathbb{R}^m. \end{aligned} \tag{4.16}$$

Neste caso, faz sentido que o cálculo dos subgradientes seja feito somente em relação a $\boldsymbol{\alpha}$. Com isso, as funções g e h apresentadas no método CCP são dadas por

$$\begin{aligned} g_0(\boldsymbol{\alpha},\boldsymbol{\xi}) &= \frac{1}{m}\sum_{i=1}^{m}\xi_i^2, & h_0(\boldsymbol{\alpha}) &= 0, \\ g_{il}(\boldsymbol{\alpha},\boldsymbol{\xi}) &= \mathbf{w}_l^T\mathbf{x}_i + a_l + \xi_i, & h_{i1}(\boldsymbol{\alpha}) &= \max_{j=1,\ldots,r_2}\{\mathbf{m}_j^T\mathbf{x}_i + b_j\} + y_i, \quad i \in \mathcal{I}, l \in \mathcal{L}_1 \\ g_{il}(\boldsymbol{\alpha},\boldsymbol{\xi}) &= \mathbf{m}_l^T\mathbf{x}_i + b_l - \xi_i, & h_{i2}(\boldsymbol{\alpha}) &= \max_{j=1,\ldots,r_1}\{\mathbf{w}_j^T\mathbf{x}_i + a_j\} - y_i, \quad i \in \mathcal{I}, l \in \mathcal{L}_2, \end{aligned}$$

em que $\mathcal{I} = \{1,\ldots,m\}$, $\mathcal{L}_1 = \{1,\ldots,r_1\}$, e $\mathcal{L}_2 = \{1,\ldots,r_2\}$.

Faremos a seguir uma análise a respeito da aplicação do método CCP na resolução do Problema (4.15) utilizando as funções definidas acima, resultando em cada iteração t no problema convexo dado por:

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\xi},\mathbf{p},\mathbf{q}}{\text{minimize}} \quad & g_0(\boldsymbol{\alpha},\boldsymbol{\xi}) \\ \text{s.a.} \quad & g_{il}(\boldsymbol{\alpha},\boldsymbol{\xi}) \leqslant h_{i1}(\boldsymbol{\alpha}_{t-1}) + \langle \boldsymbol{\beta}_{i1}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1}\rangle, \quad i \in \mathcal{I}, l \in \mathcal{L}_1, \\ & g_{il}(\boldsymbol{\alpha},\boldsymbol{\xi}) \leqslant h_{i2}(\boldsymbol{\alpha}_{t-1}) + \langle \boldsymbol{\beta}_{i2}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1}\rangle, \quad i \in \mathcal{I}, l \in \mathcal{L}_2, \end{aligned} \tag{4.17}$$

Para o cálculo de $\boldsymbol{\beta}_{ij}$, como estamos tratando do máximo de uma coleção de funções afins, o cálculo do subgradiente em relação a um elemento se resume ao cálculo de gradiente da função afim Analogamente ao que foi feito em (4.6), tomando em cada iteração t

$$j_{i1} = \underset{j=1,\ldots,r_2}{\arg\max}\{\mathbf{x}_i^T\mathbf{m}_j^{t-1} + b_j^{t-1}\} \quad \text{e} \quad j_{i2} \quad = \underset{j=1,\ldots,r_1}{\arg\max}\{\mathbf{x}_i^T\mathbf{w}_j^{t-1} + a_j^{t-1}\}, \quad \forall i = 1,\ldots,m,$$

então, definindo $\mathbf{v}^{i,s} = (\mathbf{v}_1^{i,s},\ldots,\mathbf{v}_{r_1+r_2}^{i,s}) \in \mathbb{R}^{(r_1+r_2)(n+1)}$ para cada índice $i = 1,\ldots,m$ e $s = 1,\ldots,r_1+r_2$, em que

$$\mathbf{v}_k^{i,s} = \begin{cases} (\mathbf{x}_i, 1), & k = s, \\ (\mathbf{0}, 0), & \text{c.c.}, \end{cases} \tag{4.18}$$

tem-se que $\boldsymbol{\beta}^{ij} \in \partial h_{ij}(\boldsymbol{\alpha}_t)$ será dado por

$$\boldsymbol{\beta}^{i1} = \mathbf{v}^{i,r_1+j_{i1}} \quad \text{e} \quad \boldsymbol{\beta}^{i2} = \mathbf{v}^{i,j_{i2}} \quad \forall\, i = 1,\ldots,m. \tag{4.19}$$

Desta forma, em cada iteração t, para encontrar a solução $\boldsymbol{\alpha}_t$ utilizando o método CCP representado pelo Problema (1.14) e utilizando a esparsidade dos vetores $\boldsymbol{\beta}^{ij}$, o Problema (4.15) pode ser reescrito resultando no seguinte problema quadrático com restrições lineares, portanto, com concavidades conhecidas:

$$\begin{aligned} \underset{W,\mathbf{a},M,\mathbf{b},\boldsymbol{\xi},\mathbf{p},\mathbf{q}}{\text{minimize}} \quad & \frac{1}{m}\sum_{i=1}^{m}\xi_i^2 \\ \text{s.a.} \quad & \langle \mathbf{w}_l - \mathbf{m}_{j_{i1}}, \mathbf{x}_i\rangle + a_l - b_{j_{i1}} \leqslant y_i - \xi_i, \quad i \in \mathcal{I}, l \in \mathcal{L}_1 \\ & \langle \mathbf{m}_l - \mathbf{w}_{j_{i2}}, \mathbf{x}_i\rangle + b_l - a_{j_{i2}} \leqslant \xi_i - y_i, \quad i \in \mathcal{I}, l \in \mathcal{L}_2 \end{aligned} \tag{4.20}$$

em que $\mathcal{I} = \{1,\ldots,m\}$, $\mathcal{L}_1 = \{1,\ldots,r_1\}$, e $\mathcal{L}_2 = \{1,\ldots,r_2\}$, com critério de convergência atingido quando um número máximo de iterações for alcançado ou conforme (1.15), que pode ser escrito como

$$|\|\boldsymbol{\xi}^{t-1}\|_2^2 - \|\boldsymbol{\xi}^{t}\|_2^2 \leqslant m\epsilon \tag{4.21}$$

Portanto, neste trabalho, o treinamento do modelo ℓ-DER com a abordagem CCP é feito usando o Algoritmo 2, reescrito como o Algoritmo 4.

Algoritmo 4: ℓ-CCP: regressão

Entrada: $\mathcal{T}, \epsilon, t_{\max}, r_1, r_2$.
Saída: $\{W, \mathbf{a}, M, \mathbf{b}\}$
Inicializar: $W_0, \mathbf{a}_0, M_0, \mathbf{b}_0, \boldsymbol{\xi}_0$ e $t = 0$
repita
  Compute $W, \mathbf{a}, M, \mathbf{b}, \boldsymbol{\xi}$ solução de (4.20)
  $t = t + 1$
até $t \geqslant t_{max}$ *ou conforme critério de parada* (4.21);
retorna $\{W, \mathbf{a}, M, \mathbf{b}\}$

Sobre a Complexidade dos Algoritmos ℓ-DCA e ℓ-CCP

Em relação à complexidade dos problemas de otimização utilizados para o treinamento dos modelos ℓ-DER, observe que o maior custo está na resolução dos subproblemas de otimização. Observe também que os subproblemas dos algoritmos ℓ-DCA e ℓ-CCP utilizados no treinamento do ℓ-DEP, para tarefas de regressão, são ambos problemas quadráticos com restrições lineares.

O algoritmo ℓ-CCP para tarefas de regressão possui algumas vantagens em relação ao algoritmo ℓ-DCA. Uma das vantagens é uma quantidade menor de restrições. Outra vantagem do algoritmo ℓ-CCP para tarefas de regressão está em computar os vetores $\boldsymbol{\beta}$. O fato do cálculo dos subgradientes no algoritmo ℓ-DCA serem computados em relação a todas as variáveis que definem o vetor $\boldsymbol{\alpha}$ faz com que o vetor obtido seja um vetor não esparso, diferente do que é feito no cálculo dos vetores $\boldsymbol{\beta}$ para a obtenção do algoritmo ℓ-CCP para tarefas de regressão.

Neste trabalho, realizaremos uma análise empírica a respeito da complexidade do algoritmo proposto. Ou seja, os algoritmos serão implementados e comparados com outros métodos.

4.3 Método de Inicialização

É comum inicializar algoritmos de otimização DC com vários pontos iniciais e, posteriormente, escolher aquele cujo valor da função objetivo tenha alcançado o valor mais baixo nas diferentes execuções. Para nossos modelos, o ponto de partida para os problemas de otimização foi produzido utilizando uma estratégia determinística apresentada na Seção 3.4 do artigo [87] que utiliza o método KKZ [88], partições de Voronoi [89] e resoluções de problemas de quadrados mínimos lineares. Tal estratégia será apresentada a seguir.

O método KKZ, introduzido por Katsavounidis, Kuo e Zhang em 1994, é uma técnica muito utilizada em aprendizado não supervisionado. A ideia principal é formar um subconjunto com elementos do conjunto de treinamento, tais que estes elementos estejam distribuídos o mais distante um do outro. Os elementos desse conjunto serão denominados centroides. Com estes elementos é possível fazer um agrupamento dos dados utilizando partições de Voronoi.

A partição de Voronoi consiste em particionar uma região utilizando um conjunto de centroides preestabelecido. Neste trabalho, os centroides utilizados foram os centroides encontrados através do método KKZ. Um elemento da região considerada pertence ao grupo cujo centroide que o define é o mais próximo deste elemento.

No Apêndice A é apresentado o Algoritmo 6 que descreve o método KKZ. Além disso, incluímos um exemplo com uma aplicação deste método juntamente com as

partições de Voronoi.

Para determinar o ponto de partida dos métodos de otimização DC apresentados neste trabalho, em cada subconjunto de cada uma das partições é realizado um ajuste linear minimizando o MSE destes subconjuntos. As soluções desses ajustes compõem o ponto inicial que será escrito rearranjando os elementos das soluções conforme apresentaremos mais adiante.

O método de inicialização apresentado acima é apresentado em termos matemáticos como segue. Sejam

$$\mathcal{P}^1_{\mathcal{V}}(X, r_1) = \bigcup_{j=1}^{r_1} P^1_j \quad \text{e} \quad \mathcal{P}^2_{\mathcal{V}}(X, r_2) = \bigcup_{j=1}^{r_1} P^2_j$$

partições de Voronoi, com r_1 e r_2 as quantidades dos centroides que definem os conjuntos P^1_j e P^2_j, respectivamente. Tais partições foram obtidas utilizando os centroides iniciais

$$\mathbf{x}^1_0 = \arg\max_{\mathbf{x}_i \in X}\{\|\mathbf{x}_i\|\} \quad \text{e} \quad \mathbf{x}^2_0 = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i,$$

respectivamente, através do método KKZ apresentado no Apêndice A. Observe que podemos escrever

$$\begin{aligned}\tau^{\ell}(\mathbf{x}_i; \boldsymbol{\alpha}) - y_i &= \delta_{\mathbf{a}}(W\mathbf{x}_i) - \delta_{\mathbf{b}}(M\mathbf{x}_i)) - y_i \\ &= \left(\delta_{\mathbf{a}}(W\mathbf{x}_i) - \frac{y_i}{2}\right) - \left(\delta_{\mathbf{b}}(M\mathbf{x}_i) + \frac{y_i}{2}\right).\end{aligned}$$

Para o subconjunto P^1_j é realizado um ajuste linear nos dados $\{(\mathbf{x}_i, y_i/2); \mathbf{x}_i \in P^1_j\}$. O ajuste é feito resolvendo o Problema (4.22), cuja solução será indicada por $(\mathbf{w}_j, a_j)$, com $j = 1, \ldots, r_1$,

$$\underset{\mathbf{w},a}{\text{minimize}} \sum_{\mathbf{x}_i \in P^1_j} \left(\mathbf{x}_i^T\mathbf{w} + a - \frac{y_i}{2}\right)^2. \tag{4.22}$$

Para o subconjunto P^2_j, é realizado um ajuste linear nos dados $\{(\mathbf{x}_i, y_i/2); \mathbf{x}_i \in P^2_j\}$ resolvendo o Problema (4.23), cuja solução será indicada por $(\mathbf{m}_j, b_j)$, com $j = 1, \ldots, r_2$,

$$\underset{\mathbf{m},b}{\text{minimize}} \sum_{\mathbf{x}_i \in P^2_j} \left(\mathbf{x}_i^T\mathbf{m} + b + \frac{y_i}{2}\right)^2. \tag{4.23}$$

As soluções de cada problema são arranjadas formando a aproximação inicial $\boldsymbol{\alpha}^0$ da seguinte forma $\boldsymbol{\alpha}^0 = (\mathbf{w}_1^T, a_1, \ldots, \mathbf{w}_{r_1}^T, a_{r_1}, \mathbf{m}_1^T, b_1, \ldots, \mathbf{m}_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)}$.

Os problemas de otimização dados pelos Problemas (4.22) e (4.23) podem ser resolvidos utilizando alguma técnica de resolução de problemas de quadrados mínimos. Neste trabalho utilizamos a decomposições QR.

Para análises futuras, de modo a melhorar o desempenho dos modelos, desejamos testar os modelos com outros pontos iniciais e com isso, analisar possíveis melhoras nas diversas abordagens apresentadas neste trabalho.

Na próxima seção, apresentaremos os experimentos computacionais empregados para avaliar o desempenho dos algoritmos apresentados neste trabalho para resolução de problemas de regressão.

4.4 Experimentos Computacionais

Nesta seção, apresentaremos alguns resultados computacionais sobre o comportamento dos modelos apresentados nesta tese, aplicados à tarefas representadas por dados sintéticos e reais.

Primeiramente, escolhemos alguns conjuntos de dados de tarefas de regressão para explorar e encontrar os melhores parâmetros para o modelo ℓ-DER e também para os modelos MLP, SVC, ℓ-DCA. Para os modelos ℓ-DEP e ℓ-DCA variamos os parâmetros r_1 e r_2. Para o modelo MLP usamos apenas a quantidade de neurônios (#h) na camada oculta. Já para o modelo SVC, usamos a constante penalizadora p e o kernel utilizado. Além desses modelos, incluímos também uma variação dos modelos MAXOUT, MDN e HMLP-EL [19–21], de forma a utilizá-los em tarefas de regressão. Para a rede MAXOUT, consideramos $r_1 = r_2$ tal como proposto em [21], que são as quantidades de neurônios de dilatação e de erosão. No caso da rede MDN, o primeiro bloco possuirá r_1^1 dilatações e r_2^1 erosões. As c saídas do primeiro bloco morfológico da rede se tornam entradas para o segundo bloco morfológico, que possuirá r_1^2 dilatações e r_2^2 erosões e apenas uma saída. No caso da rede HMLP-EL, consideramos tal rede com m neurônios morfológicos e l neurônios lineares. Foram ajustados as quantidades de neurônios m e l e também o termo de regularização.

4.4.1 Busca exaustiva dos melhores parâmetros dos modelos

Para encontrar os melhores valores para os parâmetros dos modelos, utilizamos um método de validação cruzada combinada com uma busca exaustiva dos parâmetros utilizando o `GridSearchCV` do `scikit-learn (sklearn)` [90]. Em todos os *datasets*, fizemos as buscas dos melhores parâmetros para os modelos utilizando validação cruzada, dividindo o conjunto de treino e teste em 5 subconjuntos (`KFold()` com $k = 5$).

A busca fornecida pelo `GridSearchCV` gera exaustivamente candidatos a partir de um conjunto de valores de parâmetros pré estabelecidos. Para cada modelo, apresentaremos em tabelas os parâmetros utilizado nas buscas e seus resultados. Adotamos os candidatos a parâmetros (`tuned_parameters`) conforme apresentado na Tabela 3.

Nas implementações dos modelos, tanto para o modelo ℓ-DCA quanto para o modelo ℓ-DER, resolvemos o Problema (4.20) e o Problema (4.10), respectivamente, utilizando o Algoritmo 4 e o Algoritmo 3, e resolvendo-os por meio do pacote `CVXPY` [51] combinado com o solver `MOSEK` [91]. Para os classificadores SVR e MLP, usamos implementações do `python's scikit-learn (sklearn)` [21,22,90,92]. As implementações dos modelos MDN e HMLP-EL foram baseadas nos códigos disponíveis em [93, 94], sendo eles adaptados para tarefas de regressão. As compilações foram feitas utilizando o otimizador `Adam` e função perda sendo o MSE dado em (4.2).

Tabela 3 – Valores fornecidos para o `tuned_parameters` do GridSearch para o conjunto de dados.

Modelo	**Candidatos**
MLP	#h:=nº de neurônios $\in \{50, 80, 100\}$
SVR	$p \in \{0.1, 1, 10, 100\}$ $\kappa \in \{linear, rbf\}$
ℓ-DCA	$r_1, r_2 \in \{2, 3, 5, 7, 10\}$
ℓ-DEP	$r_1, r_2 \in \{2, 3, 5, 7, 10\}$
MAXOUT	$r_1 = r_2 \in \{2, 3, 5, 7, 10\}$
MDN	$c, r_1^1, r_2^1, r_1^2, r_2^2 \in \{5, 10, 15\}$
HMLP-EL	$m \in \{5, 10, 20\}$ $l \in \{5, 10, 20\}$ $C \in \{0.01, 1, 10\}$

Conjunto de dados sintéticos

Curva modular

A fim de ilustrar o comportamentos dos modelos de regressão mencionados acima, construímos um conjunto de dados sintéticos em formato da letra V, que teria como uma boa função de ajuste a função $f(x) = |x|$, com $x \in [-10, +10]$. Consideramos 400 elementos simétricos no intervalo $[-10, +10]$ para compor a variável X. Para compor a variável y, utilizamos uma amostra de 400 elementos com distribuição uniforme ao longo do intervalo semiaberto $[-1, 1)$ e somamos os valores absolutos dos elementos do intervalo $[-10, +10]$, isto é, $y := |x|$+ruído.

Na Tabela 4 são apresentados os valores da métrica R^2 nos conjuntos de treino e teste obtidos pelos modelos retreinados usando os parâmetros cujo desempenho foi melhor usando validação cuzada e busca exaustiva (GridSearch) no conjunto de treino. A tabela

Tabela 7 apresenta os valores dos parâmetros obtidos pelo GridSearch para cada um dos modelos. Uma ilustração do comportamento de cada um dos classificadores no conjunto de dados sintético é apresentado na Figura 4.1, na Figura 4.2 e na Figura 4.3.

Conforme a Tabela 4, podemos ver que todos os modelos tiveram um ótimo desempenho no treinamento e de teste, obtendo desempenho bastante semelhante, com R^2 acima de 0.94 em todos os casos. Os modelos ℓ-DCA e ℓ-DER foram os únicos com desempenho levemente superior no conjunto de testes em relação ao conjunto de treino. Os demais, apresentaram desempenho no teste levemente inferior em relação ao conjunto de treino.

Nas figuras 4.1, 4.2 e na 4.3 é apresentada a ilustração do desempenho dos modelos aplicados ao problema representado pelos dados sintéticos. A coluna da esquerda fornece os resultados para os dados de treino e a da direita os resultados para os dados de teste.

Tabela 4 – Valor médio da métrica R^2 para o dataset sintético (refit).

	MLP	SVR	ℓ-DCA	ℓ-DER	MAXOUT	MDN	HMLP-EL
Treino	0.958	0.958	0.960	0.943	0.959	0.954	0.961
Teste	0.957	0.957	0.961	0.944	0.960	0.953	0.960

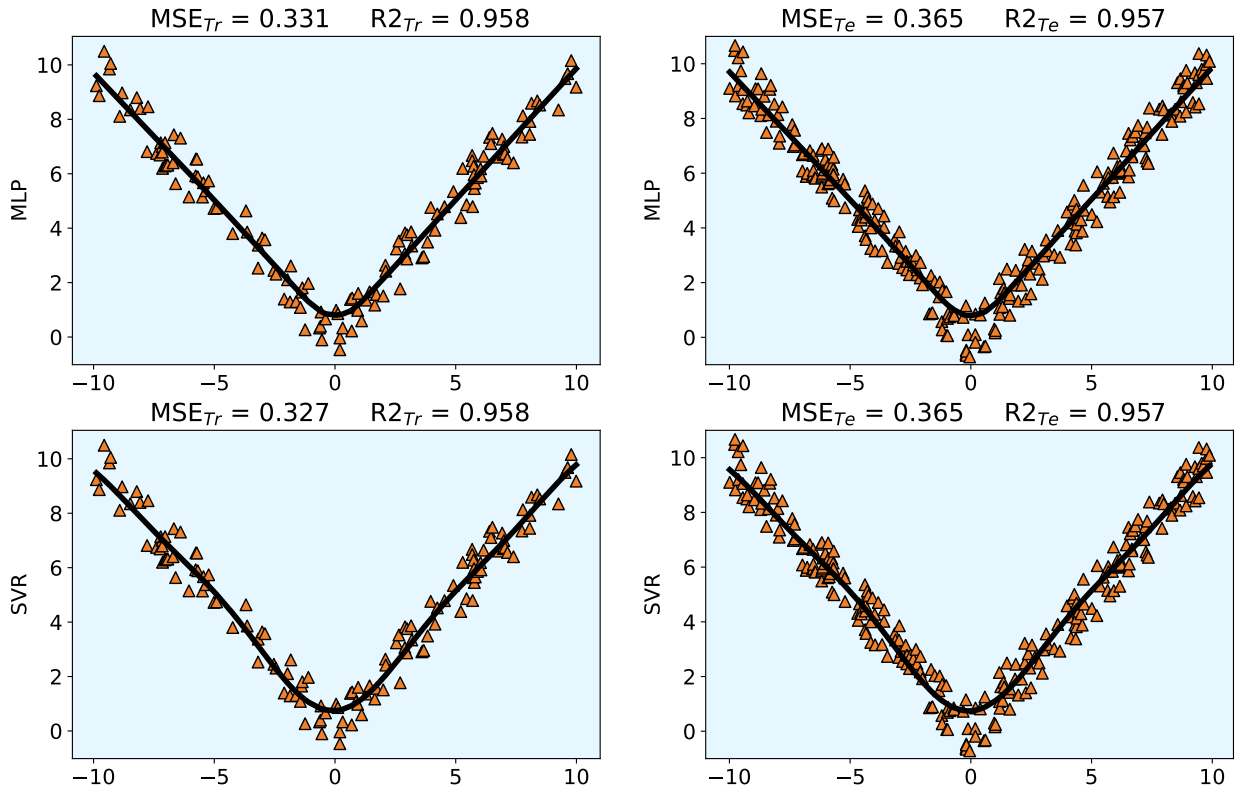


Figura 4.1 – *Dataset* sintético: curva de ajuste e desempenho em relação à métrica MSE dos regressores resultantes treinados utilizando GridSearch no *dataset* sintético.

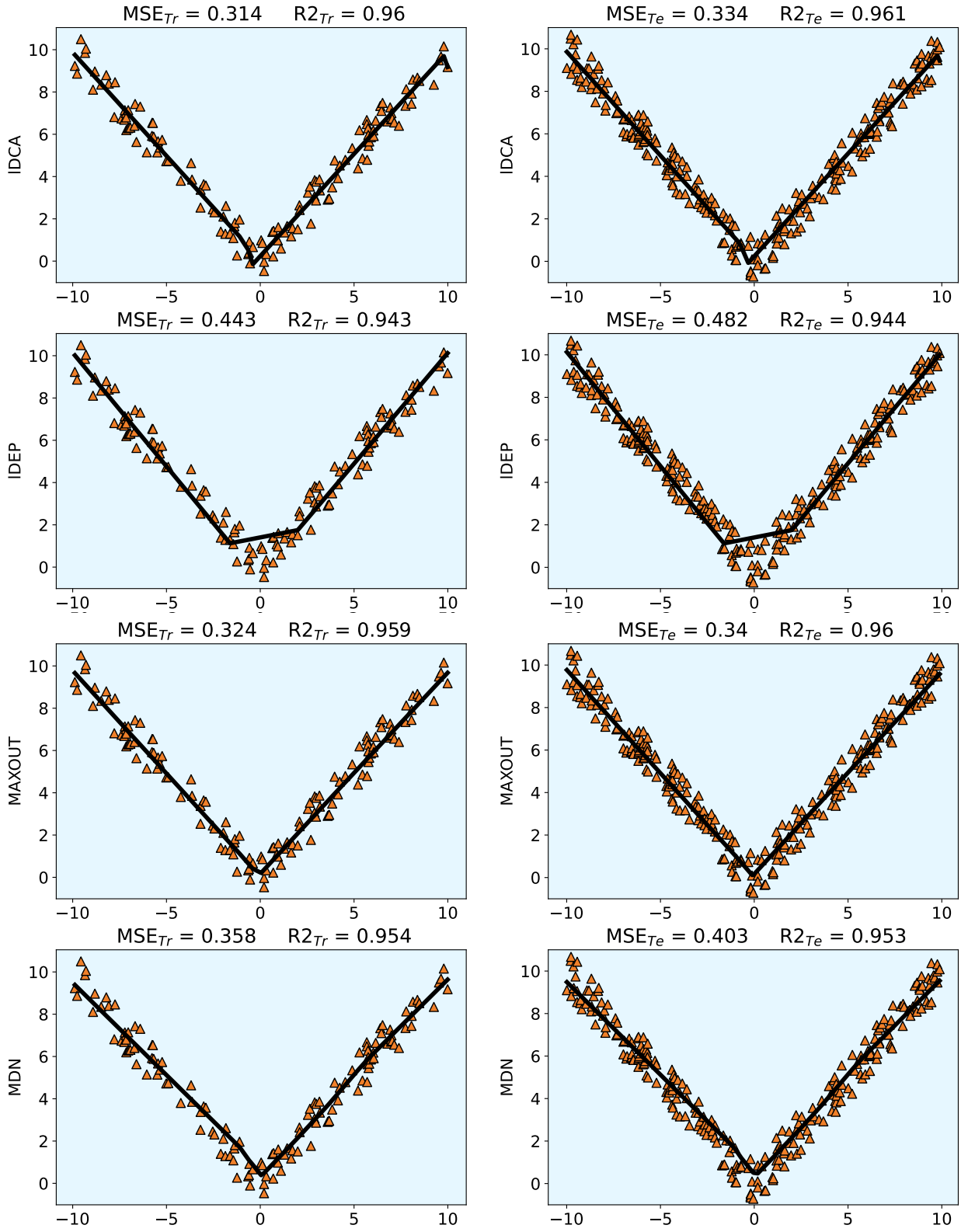


Figura 4.2 – *Dataset* sintético: curva de ajuste e desempenho em relação à métrica MSE dos regressores resultantes treinados utilizando GridSearch no *dataset* sintético.

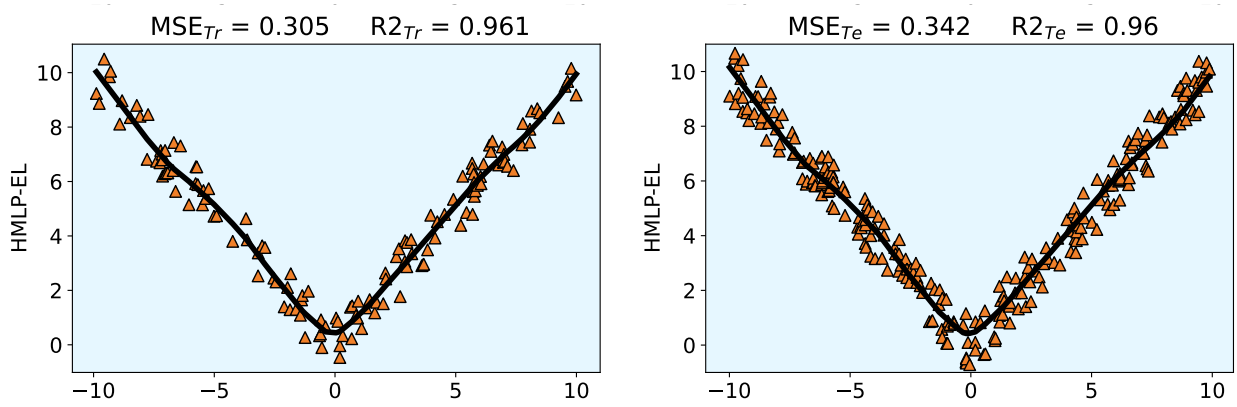


Figura 4.3 – *Dataset* sintético: curva de ajuste e desempenho em relação à métrica MSE dos regressores resultantes treinados utilizando GridSearch no *dataset* sintético.

Conjunto de dados reais

Assim como foi feito anteriormente com o *dataset* sintético, selecionamos alguns conjuntos de dados reais para analisar a escolha dos parâmetros dos modelos de regressão. Faremos a mesma busca de parâmetros para conjuntos de dados reais, isto é, foram utilizados os mesmos conjunto de parâmetros apresentados na Tabela 3. Os parâmetros nos quais os modelos obtiveram melhor desempenho estão apresentados na Tabela 7.

O primeiro *dataset* real que utilizaremos é o conjunto de dados PWlinear e o segundo é o conjunto de dados Pm10. Os resultados obtidos serão apresentados a seguir.

Dataset PWLinear

O *dataset* PWLinear é um conjunto de dados utilizado em [95] para avaliação de modelos de regressão e previsão numérica, usando aprendizado baseado em instância com seleção de comprimento de codificação.

Na Tabela 5 são apresentados os valores da métrica R^2 obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho do modelo foi melhor. O modelo que apresentou melhor desempenho no conjunto de teste foi o SVR, sendo o único modelo a apresentar R^2 acima de 0.9 no conjunto de teste. Os demais modelos apresentaram R^2 abaixo de 0.9 no conjunto de teste, com HMLP-EL e MDN abaixo de 0.8.

Tabela 5 – Valor médio da métrica R^2 para o *dataset* PWLinear (refit).

	MLP	SVR	ℓ-DCA	ℓ-DER	MAXOUT	MDN	HMLP-EL
Treino	0.862	0.985	0.923	0.848	0.915	0.614	0.786
Teste	0.824	0.981	0.879	0.813	0.828	0.623	0.789

Dataset PM10

O *dataset* consiste em uma amostra de 500 observações de um conjunto de dados que se origina de um estudo em que a poluição do ar em uma estrada está relacionada ao volume de tráfego e variáveis meteorológicas, coletados pela Administração de Estradas Públicas da Noruega. A concentração de partículas de diâmetro igual ou inferior a 10 micrômetros (PM10) é uma das medidas mais adotadas para avaliação da qualidade do ar ambiente.

Na Tabela 6 são apresentados os valores da métrica R^2 obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho do modelo foi melhor, apresentados na Tabela 20. De acordo com os resultados apresentados na Tabela 6, podemos ver que este *dataset* não apresenta um bom resultado para nenhum dos modelos estudados tanto no conjunto de treino quanto no conjunto de teste, tendo como menos pior os modelos ℓ-DCA no conjunto de treino e de teste. Apenas ℓ-DCA e o ℓ-DER obtiveram R^2 acima de 0.2 no conjunto de treino. Uma opção para melhorar o desempenho dos modelos para a tarefas proposta, seria fazer uma análise dos dados para identificar quais variáveis são importantes para os modelos para que o mesmo tenha um melhor desempenho.

Tabela 6 – Valor médio da métrica R^2 para o *dataset* PM10 (refit).

	MLP	SVR	ℓ-DCA	ℓ-DER	MAXOUT	MDN	HMLP-EL
Treino	0.118	0.185	0.284	0.225	-0.573	-4.102	0.146
Teste	-0.097	0.118	0.209	0.127	-1.113	-5.337	0.136

Resumo da Busca Exaustiva dos Melhores Parâmetros

A Tabela 7 apresenta um resumo dos resultados dos testes feitos utilizando a busca exaustiva para cada conjunto de dados apresentados nas subseções anteriores. São apresentados os parâmetros do regressor ganhador da busca exaustiva realizada em cada um dos *datasets* para os exemplos de conjuntos de dados.

A Tabela 7 resume o resultado da busca dos melhores hiperparâmetros para os modelos. Precisamente, a segunda coluna lista os hiperparâmetros[1] dos modelos da primeira coluna. O terceiro bloco de colunas mostra os melhores hiperparâmetros encontrados, considerando os quatro *datasets*, "Sintético", "PWLinear" e "PM10". Observando os resultados em cada um dos *datasets*, não é possível identificar em todos os casos um valor único global dos parâmetros para cada um dos modelos. Desta forma, adotaremos os parâmetros conforme apresentados a seguir.

Em relação aos modelos ℓ-DER e ℓ-DCA adotamos os hiperparâmetros $r_1 = r_2 = 10$.Para uma comparação entre os modelos lineares contínuos por partes, consideramos

[1] Aqui, r_1^i e r_2^i simbolizam o número de neurônios de dilatação e erosão, enquanto c é o número de saídas do primeiro bloco morfológico de um MDN [25].

Tabela 7 – Melhores hiperparâmetros para o *dataset* sintético e para os três conjuntos de dados "PWLinear" e "PM10".

Modelo	**Parâmetros**	**Sintético**	**PWLinear**	**PM10**
MLP	$\#h$	80	80	10
SVR	p	10	10	0.1
	kernel	rbf	rbf	linear
ℓ-DCA	r_1	2	2	2
	r_2	7	2	7
ℓ-DER	r_1	3	5	2
	r_2	10	2	7
MAXOUT	$r_1 = r_2$	10	2	2
MDN	c	15	5	10
	r_1^1	5	15	5
	r_2^1	5	10	5
	${r_1}^2$	15	15	10
	${r_2}^2$	10	15	15
HMLP-EL	l	10	20	20
	m	10	20	20
	C	10	1	10

$r_1 = r_2 = 10$ para a rede MAXOUT [21]. Para os modelo HMLP-EL utilizou-se a contante $C = 10$ e a quantidade de camadas morfológicas e lineares iguais a 10. Para o modelo MDN, utilizou-se o mesmo valor para todos os parâmetros, sendo este valor $c = r_1^1 = r_2^1 = r_1^2 = r_2^2 = 15$. Por fim, os hiperparâmetros dos demais modelos são configurados para os valores padrão do `scikit-learn (sklearn)`.

4.4.2 Análise empírica de desempenho em tarefas de regressão

A seguir, avaliaremos computacionalmente o desempenho do classificador proposto ℓ-DER em vários conjuntos de dados do repositório PMLB. Os conjuntos de dados escolhidos têm tamanhos médios e suas dimensões são apresentadas na Tabela 8. Além disso, comparamos o desempenho do regressor proposto com as outras técnicas de regressão utilizadas na subseção anterior. A Tabela 9 resume os hiperparâmetros usados em nossos experimentos computacionais descritos a seguir.

Tabela 8 – Informações sobre os *Datasets* de tarefas de Regressão do PMLB

	Nome	**Instâncias**	**Características**	**Código PMLB**
D1	BODYFAT	252	14	560_bodyfat
D2	CHSCASE_GEYSER1	222	2	712_chscase_geyser1
D3	CPU	209	7	561_cpu
D4	MACHINE_CPU	209	6	230_machine_cpu
D5	PM10	500	7	522_pm10
D6	PWLINEAR	200	10	229_pwLinear
D7	RABE_266	120	2	663_rabe_266
D8	RMFTSA_LADATA	508	10	666_rmftsa_ladata
D9	VISUALIZING_ENVIRONMENTAL	111	3	678_visualizing_environmental
D10	VISUALIZING_GALAXY	323	4	690_visualizing_galaxy

Tabela 9 – Escolha dos valores dos parâmetros utilizados nas implementações.

Modelo	**Parâmetros utilizados**
MLP	$\#h = 100$
SVR	$p = 1$ $kernel = \text{rbf}$
ℓ-DER	$r_1 = r_2 = 10$
ℓ-DCA	$r_1 = r_2 = 10$
MAXOUT	$r_1 = r_2 = 10$
MDN	$r_1^1 = r_2^1 = 15$ $c = 15$ $r_1^2 = r_2^2 = 15$
HMLP-EL	$l = 10$ $m = 10$ $C = 10$

Utilizando a medida de avaliação, valores mais próximos de 1 simbolizam melhor desempenho do modelo. Para o caso da medida MSE, valores mais próximos de 0 simbolizam um melhor ajuste. Ao utilizar a medida de avaliação FVU, também teremos que menores valores simbolizam melhores ajuste. Logo, ao valiar MSE e FVU teremos que menores valores simbolizam melhor desempenho sem que haja resultados negativos.

A Tabela 10 contém a média e o desvio padrão do FVU, ambos obtidos dos regressores usando validação cruzada e busca exaustiva 5-fold. A Tabela 11 contém a média e o desvio padrão do MSE e a Tabela 12 contém a média do MSE normalizada por linha, isto é, normalizada para cada um dos *datasets*. A fórmula para a normalização entre 0 e 1 utilizada foi $X_{norm} = (X - X_{\min})/(X_{\max} - X_{\min})$, em que X_{norm} é o valor normalizado entre 0 e 1, X o valor original do valor a ser normalizado, $X_{\min}$ o valor mínimo do conjunto e $X_{\max}$ o valor máximo do conjunto. Após aplicar essa fórmula para cada ponto de dados de médias, teremos um novo conjunto de dados normalizados entre 0 e 1. A Tabela 13

contém a média do tempo gasto nos treinamentos.

Os boxplots mostrados na Figura 4.4 e na Figura 4.5 representam os *scores* representadas nas tabelas acima, bem como o tempo de execução (em segundos) usado para treinar os regressores. Precisamente, eles foram obtidos, para cada conjunto de dados, calculando os *scores* médios MSE e FVU/R^2 produzidos pelos regressores. O diagrama de Hasse representa o teste de hipótese não paramétrico de Wilcoxon com um nível de confiança de 95%, em que os modelos mais abaixo, considerando as linhas, são os modelos com pior desempenho, de acordo com cada métrica. A não existência de linhas que ligam os modelos, nem por transitividade, indica que não foi possível identificar diferenças significativas em relação as medianas das médias dos scores dos modelos. Consideramos nesse caso, que os modelos são competitivos entre si.

Os *boxplots* apresentados indicam que o modelo ℓ-DER apresenta menor mediana em relação a medida MSE, sendo este valor bem próximo às medianas das médias do MSE para os modelos SVR, ℓ-DCA, MAXOUT e HMLP-EL. Já em relação à medida R^2 a rede MAXOUT obteve melhor mediana das média, seguido dos modelos ℓ-DER e HMLP-EL. Do diagrama apresentado na Figura 4.6, em relação à métrica MSE, o ℓ-DER treinado com o algoritmo ℓ-CCP obteve desempenho superior aos modelos SVR, MDN e MLP, sendo competitivos com os demais modelos. O modelo treinado utilizando o ℓ-DCA obteve desempenho competitivo em relação a todos os demais modelo. O modelo MLP foi o que apresentou pior desempenho em relação à métrica MSE.

Quanto ao tempo de execução do treinamento, o SVR mostrou ser o procedimento mais rápido juntamente com o modelo HMLP-EL. O ℓ-DCA provou ser mais lento do que todos os demais regressores. A Figura 4.5 também fornece uma interpretação visual do resultado do nosso experimento computacional em relação ao tempo de execução do treinamento dos modelos. Embora o modelo SVR tenha obtivo melhor tempo de processamento, o desempenho deste modelo não foi o melhor. Os modelos ℓ-DER e ℓ-DCA, que obtiveram maior tempo de processamento, obtiveram desempenhos superiores ou competitivos aos demais modelos.

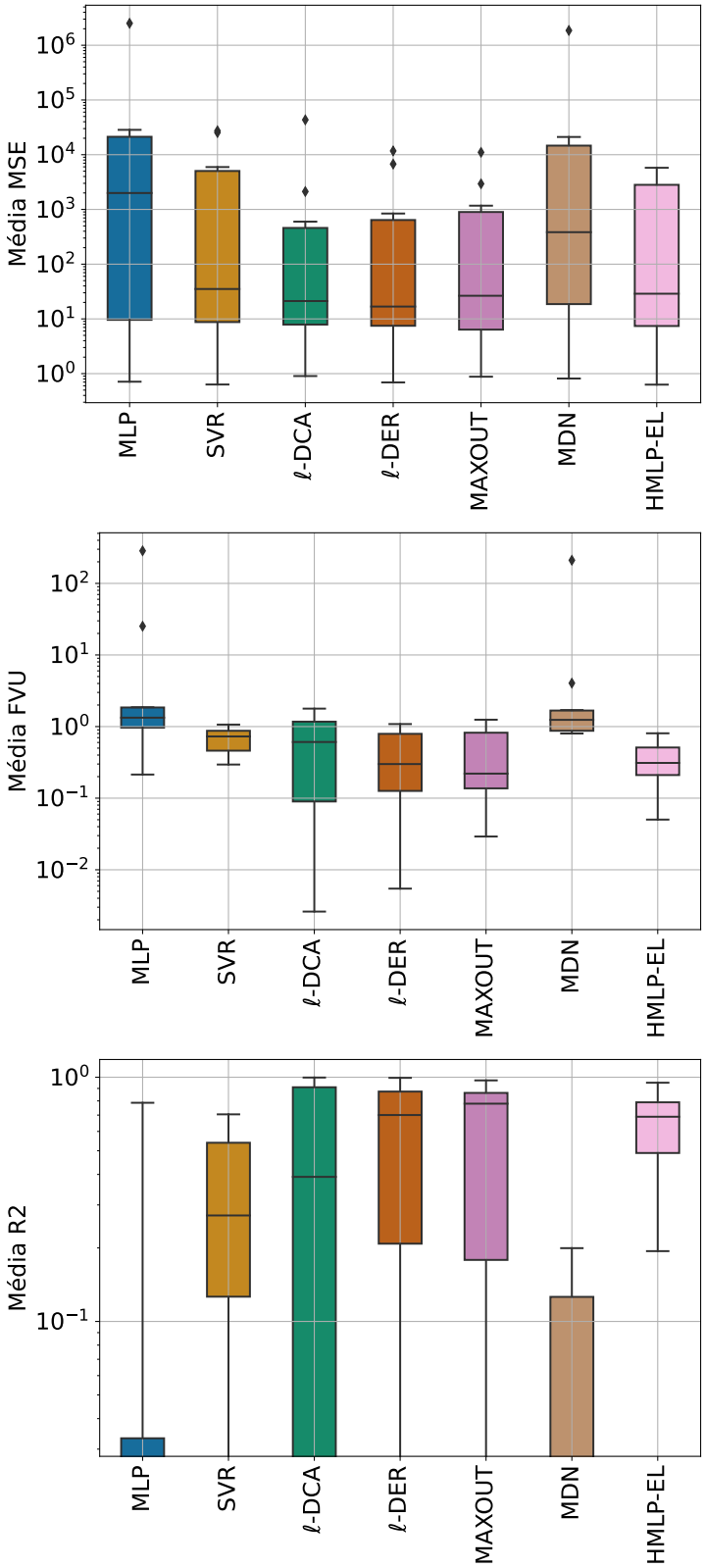


Figura 4.4 – Boxplots da média MSE, FVU e R^2 dos *Datasets* da Tabela 8.

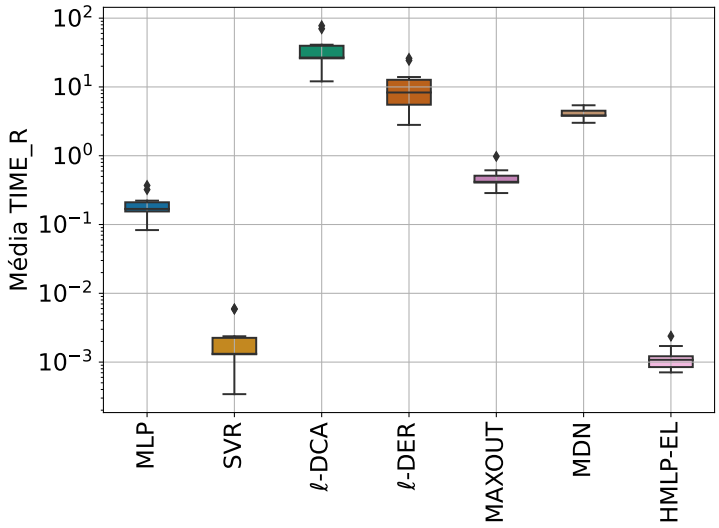
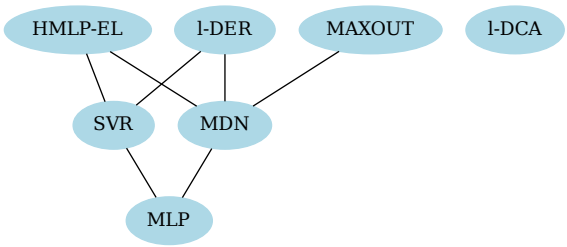


Figura 4.5 – Boxplots do Tempo médio (em segundos) gasto no treinamento nos *Datasets* da Tabela 8.

a) Erro médio quadrático (MSE)



b) Fração de variância inexplicada (FVU)

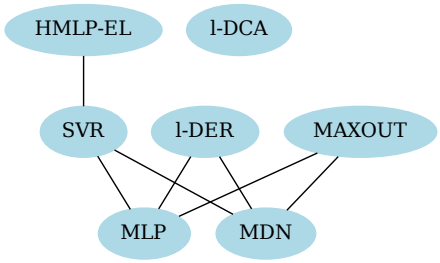


Figura 4.6 – Diagramas de Hasse: Teste de Hipótese de Wilcoxon, com nível de confiança de 95%, com desempenho dos regressores nas métricas MSE e FVU

Tabela 10 – Média o desvio padrão do FVU dos *Datasets* da Tabela 8.

FVU = 1 - $\mathbf{R}^2$	MLP	SVR	l-DCA	l-DER	MAXOUT	MDN	HMLP-EL
D1	1.862 ± 0.249	0.320 ± 0.023	0.102 ± 0.002	0.082 ± 0.064	0.201 ± 0.020	1.596 ± 0.601	0.198 ± 0.003
D2	25.231 ± 0.291	0.295 ± 0.027	0.252 ± 0.008	0.234 ± 0.020	0.239 ± 0.030	4.044 ± 0.200	0.270 ± 0.015
D3	1.175 ± 0.148	1.066 ± 0.007	0.086 ± 0.012	0.354 ± nan	0.042 ± 0.025	0.800 ± 0.110	0.247 ± 0.017
D4	1.123 ± 0.131	1.062 ± 0.002	1.437 ± 1.828	0.245 ± 0.127	0.116 ± 0.013	0.838 ± 0.139	0.173 ± 0.009
D5	0.915 ± 0.125	0.810 ± 0.022	1.159 ± 0.129	0.882 ± 0.033	1.128 ± 0.127	1.047 ± 0.190	0.806 ± 0.041
D6	0.214 ± 0.002	0.421 ± 0.003	0.966 ± 0.117	0.944 ± 0.393	0.202 ± 0.048	0.883 ± 0.073	0.352 ± 0.026
D7	1.489 ± 0.436	0.895 ± 0.083	0.003 ± 0.001	0.005 ± 0.003	0.029 ± 0.003	1.436 ± 0.322	0.050 ± 0.015
D8	0.759 ± 0.371	0.579 ± 0.015	1.176 ± 0.331	0.520 ± 0.079	0.702 ± 0.439	0.871 ± 0.404	0.539 ± 0.162
D9	1.811 ± 0.325	0.785 ± 0.066	1.782 ± 0.643	1.085 ± 0.291	0.861 ± 0.074	1.702 ± 0.330	0.677 ± 0.195
D10	285.043 ± 20.826	0.671 ± 0.007	0.067 ± 0.002	0.090 ± nan	1.247 ± 0.006	210.502 ± 9.140	0.424 ± 0.149
Média	31.962 ± 84.666	0.690 ± 0.269	0.703 ± 0.636	0.444 ± 0.374	0.477 ± 0.441	22.372 ± 62.717	0.374 ± 0.226
Mediana	1.489 ± 0.291	0.690 ± 0.022	0.703 ± 0.117	0.354 ± 0.079	0.239 ± 0.030	1.436 ± 0.322	0.352 ± 0.026

Tabela 11 – Média o desvio padrão do MSE dos *Datasets* da Tabela 8

MSE	MLP	SVR	l-DCA	l-DER
D1	130.127 ± 21.081	22.329 ± 2.269	7.093 ± 0.072	5.643 ± 4.303
D2	4104.385 ± 42.868	48.047 ± 5.446	41.029 ± 2.240	38.112 ± 4.018
D3	26959.362 ± 11939.237	25384.432 ± 14189.703	2124.148 ± 1416.344	11721.656 ± nan
D4	28459.397 ± 4478.830	27335.708 ± 7362.927	43368.856 ± 57086.531	6746.586 ± 4972.392
D5	0.712 ± 0.049	0.633 ± 0.027	0.903 ± 0.038	0.691 ± 0.073
D6	4.246 ± 0.297	8.371 ± 0.746	19.101 ± 0.769	19.082 ± 9.344
D7	3859.538 ± 912.702	2334.144 ± 80.073	6.868 ± 2.116	14.528 ± 9.935
D8	5.479 ± 0.101	4.758 ± 2.356	10.235 ± 7.237	4.094 ± 1.362
D9	22.013 ± 2.041	9.897 ± 3.461	23.285 ± 14.015	13.024 ± 0.023
D10	2517978.676 ± 1226.590	5947.467 ± 502.763	598.277 ± 65.842	841.874 ± nan
Média	258152.394 ± 753349.755	6109.579 ± 10291.571	4619.980 ± 12931.699	1940.529 ± 3820.577
Mediana	3859.538 ± 42.868	48.047 ± 5.446	23.285 ± 7.237	19.082 ± 4.303

MSE	MAXOUT	MDN	HMLP-EL
D1	13.982 ± 1.030	110.719 ± 38.786	13.819 ± 0.624
D2	38.949 ± 5.792	657.568 ± 17.995	43.885 ± 1.499
D3	1171.296 ± 1157.179	18300.162 ± 7909.263	5766.622 ± 2848.657
D4	2938.988 ± 468.176	21089.708 ± 2268.808	4420.241 ± 980.280
D5	0.879 ± 0.039	0.814 ± 0.092	0.629 ± 0.011
D6	3.967 ± 0.629	17.595 ± 2.874	6.966 ± 0.047
D7	76.425 ± 11.392	3731.227 ± 622.286	132.467 ± 47.240
D8	4.879 ± 0.879	6.329 ± 0.058	4.088 ± 0.758
D9	10.857 ± 3.825	21.763 ± 9.835	8.758 ± 4.709
D10	11044.770 ± 757.791	1861535.137 ± 56177.256	3709.065 ± 1047.489
Média	1530.499 ± 3295.147	190547.102 ± 557048.034	1410.654 ± 2160.324
Mediana	38.949 ± 5.792	657.568 ± 38.786	43.885 ± 4.709

Tabela 12 – Média do MSE normalizado dos *Datasets* da Tabela 8.

MSE	MLP	SVR	l-DCA	l-DER	MAXOUT	MDN	HMLP-EL
D1	1.000000	0.134041	0.011651	0.000000	0.066990	0.844093	0.065680
D2	1.000000	0.002443	0.000717	0.000000	0.000206	0.152340	0.001420
D3	1.000000	0.938928	0.036949	0.409118	0.000000	0.664217	0.178196
D4	0.631227	0.603433	1.000000	0.094178	0.000000	0.448943	0.036638
D5	0.301909	0.012771	1.000000	0.225109	0.910495	0.672991	0.000000
D6	0.018449	0.291006	1.000000	0.998722	0.000000	0.900488	0.198140
D7	1.000000	0.604068	0.000000	0.001988	0.018054	0.966696	0.032600
D8	0.226380	0.109099	1.000000	0.001069	0.128720	0.364638	0.000000
D9	0.912437	0.078406	1.000000	0.293693	0.144465	0.895239	0.000000
D10	1.000000	0.002125	0.000000	0.000097	0.004150	0.739235	0.001236
Média	1.000000	0.018302	0.012500	0.002064	0.000467	0.736680	0.000000
Mediana	1.000000	0.007542	0.001094	0.000000	0.005173	0.166253	0.006458

Tabela 13 – Média o desvio padrão do Tempo gasto no treinamento dos *Datasets* da Tabela 8.

Time	MLP	SVR	l-DCA	l-DER	MAXOUT	MDN	HMLP-EL
D1	0.175 ± 0.007	0.002 ± 0.000	35.446 ± 0.212	9.005 ± 2.794	0.448 ± 0.034	3.856 ± 0.062	0.002 ± 0.001
D2	0.222 ± 0.090	0.001 ± 0.000	26.162 ± 4.168	5.994 ± 0.140	0.411 ± 0.047	4.173 ± 0.068	0.001 ± 0.000
D3	0.163 ± 0.013	0.001 ± 0.000	26.837 ± 0.197	13.930 ± nan	0.533 ± 0.192	3.847 ± 0.110	0.001 ± 0.001
D4	0.154 ± 0.007	0.001 ± 0.000	26.680 ± 0.299	8.157 ± 3.912	0.614 ± 0.066	3.830 ± 0.098	0.001 ± 0.001
D5	0.370 ± 0.063	0.006 ± 0.001	70.074 ± 0.016	24.317 ± 14.748	0.424 ± 0.083	5.413 ± 0.017	0.002 ± 0.001
D6	0.157 ± 0.006	0.001 ± 0.000	25.990 ± 0.046	5.359 ± 0.004	0.369 ± 0.000	3.799 ± 0.002	0.001 ± 0.000
D7	0.083 ± 0.025	0.000 ± 0.000	14.407 ± 0.076	3.756 ± 1.040	0.406 ± 0.119	3.055 ± 0.051	0.001 ± 0.000
D8	0.322 ± 0.003	0.006 ± 0.000	77.530 ± 0.460	26.043 ± 15.989	0.412 ± 0.015	5.404 ± 0.018	0.001 ± 0.000
D9	0.089 ± 0.017	0.000 ± 0.000	12.065 ± 2.088	2.808 ± 0.036	0.287 ± 0.000	3.010 ± 0.060	0.001 ± 0.000
D10	0.174 ± 0.002	0.002 ± 0.000	41.099 ± 0.107	8.447 ± nan	0.978 ± 0.026	4.623 ± 0.016	0.001 ± 0.000
Média	0.191 ± 0.087	0.002 ± 0.002	35.629 ± 20.779	10.782 ± 7.790	0.488 ± 0.184	4.101 ± 0.792	0.001 ± 0.000
Mediana	0.174 ± 0.013	0.001 ± 0.000	26.837 ± 0.212	8.447 ± 2.794	0.424 ± 0.047	3.856 ± 0.060	0.001 ± 0.000

5 Perceptron Dilatação-Erosão Linear Aplicado a Tarefas de Classificação

Assim como nos modelos preditivos para tarefas de regressão, o aproximador universal dado pelo operador dilatação-erosão τ^ℓ pode ser usado para tarefas de classificação. O modelo para tarefas de classificação será denominado *perceptron* dilatação-erosão linear (ℓ-DEP, *linear dilation-erosion perceptron*). Pelos teoremas 3.1 e 3.2, um classificador ℓ-DEP pode teoricamente resolver qualquer problema de classificação binária.

Vamos agora apresentar algumas abordagens para o treinamento de um classificador ℓ-DEP.

Sem perda de generalidade, suponha que as classes de estudo em um conjunto de dados para tarefa de classificação binária estejam no conjunto $\{-1, +1\}$. Caso contrario, basta fazer um mapeamento utilizando a função bijetiva σ. De fato, dado um conjunto de treino $\mathcal{T} = \{(\mathbf{x}_i, d_i) : i = 1, \ldots, m\} \subseteq \mathbb{V} \times \mathbb{C}$, com $\mathbb{V}$ sendo o espaço de características e $\mathbb{C}$ o conjunto das classes, então basta fazer $y_i = \sigma(d_i) \in \{-1, +1\}$. A classificação de um elemento $\mathbf{x}$ utilizando um modelo ℓ-DEP é definida em (5.1), com y sendo a classe predita e f sendo a função sinal.

$$y = f\big(\tau^\ell(\mathbf{x})\big). \tag{5.1}$$

Em tarefas de classificação, a função perda usada no treinamento pode descrever o comportamento da classificação utilizando o modelo de predição. Uma função perda bastante utilizada em classificação binária é a denominada *Hinge Loss* definida em (5.2). Intuitivamente, a função Hinge Loss usa o conceito de margem máxima entre os elementos de classes diferentes, similar ao que é feito com o SVM.

$$\mathcal{H}(\mathcal{T}) = \sum_{i=1}^{m} \max\{0, 1 - y_i \tau^\ell(\mathbf{x}_i)\} \tag{5.2}$$

Se o sinal de y_i e $\tau^\ell(\mathbf{x}_i)$ forem os mesmos, o modelo fez uma predição correta de um dado de treino $\mathbf{x}_i$ e $y_i \tau^\ell(\mathbf{x}_i) \geqslant \epsilon > 0$. Incorporando o valor de ϵ na função de predição τ^ℓ, podemos, sem perda de generalidade, reescrever a desigualdade como $y_i \tau^\ell(\mathbf{x}_i) \geqslant 1$. Neste caso, o termo $\max\{0, 1 - y_i \tau^\ell(\mathbf{x}_i)\}$ não irá contribuir no valor de $\mathcal{H}$, pois será nulo. Por outro lado, se os sinais forem distintos, então o modelo fez uma predição incorreta do dado e $y_i \tau^\ell(\mathbf{x}_i) \leqslant 1$, logo, $\max\{0, 1 - y_i \tau^\ell(\mathbf{x}_i)\} \neq 0$. Ou seja, os dados de treinamento em que o modelo faz uma predição incorreta sempre contribuirão para a função $Hinge\,Loss$.

Quanto menor for o valor $\mathcal{H}(\mathcal{T})$, mais correta será a predição do modelo ℓ-DEP no conjunto de treinamento. Desta forma, os parâmetros que definem o modelo ℓ-DEP podem ser determinados minimizando $\mathcal{H}(\mathcal{T})$.

Considerando τ^ℓ como uma função de seus parâmetros $W, \mathbf{a}, M, \mathbf{b}$ que definem o classificador ℓ-DEP, podemos também rearranjá-los como

$$\boldsymbol{\alpha} = (\mathbf{w}_1^T, a_1, \ldots, \mathbf{w}_{r_1}^T, a_{r_1}, \mathbf{m}_1^T, b_1, \ldots, \mathbf{m}_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)}, \tag{5.3}$$

em que $\mathbf{w}_i^T$ e $\mathbf{m}_i^T$ são as linhas de $W \in \mathbb{R}^{r_1 \times n}$ e $M \in \mathbb{R}^{r_2 \times n}$, respectivamente. Com isso, $\tau^\ell(\mathbf{x}_i)$ passa a ser apresentado como $\tau^\ell(\mathbf{x}_i; \boldsymbol{\alpha})$ e a função *Hinge Loss* pode ser utilizada como uma função perda no treinamento do modelo ℓ-DEP, reescrita conforme a equação (5.4).

$$\mathcal{H}(\mathcal{T}, \boldsymbol{\alpha}) = \sum_{i=1}^{m} \max\{0, 1 - y_i \tau^\ell(\mathbf{x}_i; \boldsymbol{\alpha})\} \tag{5.4}$$

Desta forma, os parâmetros do modelo ℓ-DEP podem ser determinados resolvendo o Problema (5.5), com $N = (r_1 + r_2)(n + 1)$,

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\text{minimize}}\ \mathcal{H}(\mathcal{T}, \boldsymbol{\alpha}). \tag{5.5}$$

5.1 Treinamento Baseado em Otimização DC

Inspirados no SVC linear [6] e no procedimento côncavo-convexo desenvolvido por Charisopoulos e Maragos em [30], apresentaremos uma abordagem de treinamento do ℓ-DEP aplicado a problemas de classificação utilizando o procedimento côncavo-convexo [36].

Definimos os conjuntos que contêm os índices das entrada de cada classe por

$$I^+ = \{\, i \,;\, y_i := \sigma(d_i) = +1\} \quad \text{e} \quad I^- = \{\, i \,;\, y_i := \sigma(d_i) = -1\}.$$

Dessa forma, podemos escrever as restrições como

$$\tau^\ell(\mathbf{x}_i, \boldsymbol{\alpha}) \leqslant \xi_i - 1, \text{ para } i \in I^- \quad \text{e} \quad \tau^\ell(\mathbf{x}_i, \boldsymbol{\alpha}) \geqslant 1 - \xi_i, \text{ para } i \in I^+. \tag{5.6}$$

Concluindo, um classificador ℓ-DEP pode ser treinado resolvendo o seguinte problema:

$$\begin{aligned} \underset{W, \mathbf{a}, M, \mathbf{c}, \boldsymbol{\xi}}{\text{minimize}} \quad & \sum_{i=1}^{m} \max(\xi_i, 0) \\ \text{s.a.} \quad & \max_{k=1,\ldots,r_1} (\mathbf{w}_k^T \mathbf{x}_i + a_i) - \xi_i \leqslant \max_{j=1,\ldots,r_2} (\mathbf{m}_j^T \mathbf{x}_i + b_j) - 1, \forall i \in I^- \\ & \max_{j=1,\ldots,r_2} (\mathbf{m}_j^T \mathbf{x}_i + b_j) - \xi_i \leqslant \max_{k=1,\ldots,r_1} (\mathbf{w}_k^T \mathbf{x}_i + a_i) - 1, \ \forall i \in I^+. \end{aligned} \tag{5.7}$$

Semelhante ao que foi feito na abordagem de treinamento do modelo ℓ-DER utilizando o algoritmo CCP, considere

$$
\begin{aligned}
g_0(\boldsymbol{\alpha}, \boldsymbol{\xi}) &= \sum_{i=1}^{m} \max(\xi_i, 0), \qquad h_0(\boldsymbol{\alpha}) = 0, \\
g_{il}(\boldsymbol{\alpha}, \boldsymbol{\xi}) &= \mathbf{w}_l^T \mathbf{x}_i + a_l - \xi_i, \quad h_{i1}(\boldsymbol{\alpha}) = \max_{j=1,\ldots,r_2} \{\mathbf{m}_j^T \mathbf{x}_i + b_j\} - 1, \quad i \in I^-, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad l = 1, \ldots, r_1, \\
g_{il}(\boldsymbol{\alpha}, \boldsymbol{\xi}) &= \mathbf{m}_l^T \mathbf{x}_i + b_l - \xi_i, \quad h_{i2}(\boldsymbol{\alpha}) = \max_{j=1,\ldots,r_1} \{\mathbf{w}_j^T \mathbf{x}_i + a_j\} - 1, \quad i \in I^+, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad l = 1, \ldots, r_2.
\end{aligned}
$$

Linearizando as funções $h_{ij}(\boldsymbol{\alpha})$ utilizando subgradientes $\boldsymbol{\beta}_{ij} \in \partial h_{ij}(\boldsymbol{\alpha})$, para $i = 1, \ldots, n$ e $j = 1, 2$, o Problema (5.7) é resolvido iterativamente solucionando, em cada iteração t, o problema convexo dado pelo Problema (5.8)

$$
\begin{aligned}
\underset{\boldsymbol{\alpha}, \boldsymbol{\xi}, \mathbf{p}}{\text{minimize}} \quad & g_0(\boldsymbol{\alpha}, \boldsymbol{\xi}) \qquad (5.8) \\
\text{s.a.} \quad & g_{il}(\boldsymbol{\alpha}, \boldsymbol{\xi}) \leqslant h_{i1}(\boldsymbol{\alpha}_{t-1}) + \langle \boldsymbol{\beta}_{i1}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1} \rangle, \quad i \in I^-, l = 1, \ldots, r_1, \\
& g_{il}(\boldsymbol{\alpha}, \boldsymbol{\xi}) \leqslant h_{i2}(\boldsymbol{\alpha}_{t-1}) + \langle \boldsymbol{\beta}_{i2}, \boldsymbol{\alpha} - \boldsymbol{\alpha}_{t-1} \rangle, \quad i \in I^+, l = 1, \ldots, r_2. \\
& \qquad (5.9)
\end{aligned}
$$

Numa iteração t, tome

$$
j_{i1} = \underset{j=1,\ldots,r_1}{\arg\max} \{\mathbf{x}_i^T \mathbf{m}_j^{t-1} + b_j^{t-1}\}, \quad i \in I^-,
$$

e

$$
j_{i2} = \underset{j=1,\ldots,r_2}{\arg\max} \{\mathbf{x}_i^T \mathbf{w}_j^{t-1} + a_j^{t-1}\}, \quad i \in I^+,
$$

então, definindo $\mathbf{v}^{i,s} = (\mathbf{v}_1^{i,s}, \ldots, \mathbf{v}_{r_1+r_2}^{i,s}) \in \mathbb{R}^{(r_1+r_2)(n+1)}$ para cada um dos índices $i = 1, \ldots, n$ e $s = 1, \ldots, r_1 + r_2$, em que

$$
\mathbf{v}_k^{i,s} = \begin{cases} (\mathbf{x}_i, 1), & k = s, \\ (\mathbf{0}, 0), & \text{c.c.}, \end{cases} \qquad (5.10)
$$

tem-se que $\boldsymbol{\beta}_{ij} \in \partial h_{ij}(\boldsymbol{\alpha}_t)$ será dado por

$$
\boldsymbol{\beta}_{i1} = \mathbf{v}^{i, r_1 + j_{i1}}, i \in I^- \quad \text{e} \quad \boldsymbol{\beta}_{i2} = \mathbf{v}^{i, j_{i2}}, i \in I^+. \qquad (5.11)
$$

Desta forma, podemos reescrever o Problema (5.8) utilizando os valores encontrados anteriormente. Ou seja, a cada iteração t, o elemento $\boldsymbol{\alpha}_t$ é obtido como a solução do seguinte problema convexo com restrições lineares

$$
\begin{aligned}
\underset{W, \mathbf{a}, M, \mathbf{b}, \boldsymbol{\xi}, \mathbf{p}}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \max(\xi_i, 0) \qquad (5.12) \\
\text{s.a.} \quad & \langle \mathbf{w}_l - \mathbf{m}_{j_{i1}}, \mathbf{x}_i \rangle + a_l - b_{j_{i1}} \leqslant \xi_i - 1, \quad i \in I^-, \ l = 1, \ldots, r_1, \\
& \langle \mathbf{m}_l - \mathbf{w}_{j_{i2}}, \mathbf{x}_i \rangle + b_l - a_{j_{i2}} \leqslant \xi_i - 1, \quad i \in I^+, \ l = 1, \ldots, r_2, \\
& \qquad (5.13)
\end{aligned}
$$

e critério de convergência dado por

$$\frac{1}{m}\left|\sum_{i=1}^{m}\max(\xi_i^{t-1},0)-\sum_{i=1}^{m}\max(\xi_i^{t},0)\right|\leqslant\epsilon. \tag{5.14}$$

Portanto, o modelo ℓ-DEP pode ser treinado utilizando o Algoritmo 5.

Algoritmo 5: ℓ-DEP: classificação

Entrada: $\mathcal{T},\epsilon,t_{\max}$.
Saída: $\{W,\mathbf{a},M,\mathbf{b}\}$
Inicializar: $W_0,\mathbf{a}_0,M_0,\mathbf{b}_0,\boldsymbol{\xi}_0,t=0$
repita
  Compute $W,\mathbf{a},M,\mathbf{b},\boldsymbol{\xi}$ solução de (5.12)
  $t=t+1$
até $t<t_{max}$ *ou convergir conforme* (5.14);
retorna $\{W,\mathbf{a},M,\mathbf{b}\}$

Até o momento apresentamos uma abordagem de treinamento para os modelos ℓ-DEP para tarefas de classificação baseados no algoritmo CCP, um algoritmo de otimização DC restrito. Transformar o problema irrestrito (5.5) num problema restrito nos permitiu utilizar o algoritmo CCP. Desta forma, nos perguntamos se seria possível treinar o modelo ℓ-DEP para tarefas de classificação resolvendo o Problema (5.5) utilizando o DCA. Tal abordagem seria interessante, mas neste capítulo partimos da minimização da função perda *Hinge Loss* que, em princípio, não sabemos se pode ser vista como uma função DC. Podemos apenas afirmar que ela corresponde ao máximo entre funções DC. Ou seja, para utilizar o algoritmo DCA para treinar o modelo ℓ-DEP para tarefas de classificação binária teríamos que utilizar uma outra função perda ou tentar verificar se a função adotada atualmente é uma função DC.

Na próxima seção será utilizada a estratégia de inicialização apresentada no capítulo anterior, aplicada a problemas de classificação. Em seguida, na seção 5.3, apresentaremos alguns experimentos computacionais que demonstram o desempenho do modelo ℓ-DEP em várias tarefas de classificação binária, comparado-o a outros modelos de classificação.

5.2 Método de Inicialização

Inspirados na estratégia apresentada no capítulo anterior para a inicialização de $\boldsymbol{\alpha}^0$ dos modelos para tarefas de regressão, neste capítulo utilizamos uma abordagem semelhante utilizando o método KKZ e partições de Voronoi. A diferença está em determinar, em cada partição, um classificador linear ao invés do ajuste de curvas linear.

Em cada subconjunto P_j^1 e P_j^2 das partições $\mathcal{P}_{\mathcal{V}}(X,r_1)$ e $\mathcal{P}_{\mathcal{V}}(X,r_2)$, respectivamente, busca-se encontrar elementos $(\mathbf{w}_j,a_j)$, $j=1,\ldots,r_1$ e $(\mathbf{m}_j,b_j)$, $j=1,\ldots,r_2$, que

definem os modelos lineares de classificação $g_1(\mathbf{x}) = \mathbf{x}^T\mathbf{w}_j + a_j$ e $g_2(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_j + b_j$, de tal forma que tais elementos, quando rearranjados, formam o ponto inicial $\boldsymbol{\alpha}^0$ dado em (5.15)

$$\boldsymbol{\alpha}^0 = (\mathbf{w}_1^T, a_1, \ldots, \mathbf{w}_{r_1}^T, a_{r_1}, \mathbf{m}_1^T, b_1, \ldots, \mathbf{m}_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)}. \tag{5.15}$$

A classificação y^k para cada elemento $\mathbf{x}$ das partições $\mathcal{P}_{\mathcal{V}}(X, r_k)$, $k \in \{1, 2\}$, é dada por

$$y^k(\mathbf{x}) = \begin{cases} +1, & g_k(\mathbf{x}) \geqslant 0 \\ -1, & \text{c.c.}, \end{cases} \tag{5.16}$$

Os classificadores lineares de cada partição foram tomados sendo o SVC linear. Para subconjuntos das partições com dados de apenas uma classe, foram tomados parâmetros com entradas aleatórias.

5.3 Experimentos Computacionais

Nesta seção, assim como fizemos no capítulo anterior com os modelos de classificação, apresentaremos alguns resultados computacionais sobre o comportamento dos modelos apresentados neste trabalho, aplicados a tarefas representadas por dados sintéticos e reais.

5.3.1 Busca exaustiva dos melhores parâmetros dos modelos

Primeiramente, escolhemos alguns conjuntos de dados de tarefas de classificação binária para explorar e encontrar melhores parâmetros para o modelo ℓ-DEP e também para os modelos DEP, MLP, SVC, MDN e HMLP-EL. Para isso, utilizamos um método de validação cruzada combinada com uma busca exaustiva dos parâmetros utilizando o `GridSearchCV` do `sklearn`. Em todos os *datasets* fizemos as buscas dos melhores parâmetros para os modelos utilizando validação cruzada estratificada, com 5 subconjuntos (`StratifiedKFold()` com $k = 5$). Em relação a separação dos dados de treino e teste dos *datasets*, quando não estiver claramente apresentado, usamos uma proporção estratificada de 30% para o conjunto de teste e os demais 70% para validação e treino.

A busca fornecida pelo `GridSearchCV` gera exaustivamente candidatos a partir de um conjunto de valores de parâmetros pré estabelecidos. Adotamos os candidatos a parâmetros (`tuned_parameters`) apresentados na Tabela 14. Para cada modelo, apresentaremos em tabelas os parâmetros utilizados nas buscas e seus resultados. Em alguns casos, pode ocorrer que, com mais de um conjunto de parâmetros, um modelo obtenha o mesmo desempenho. Neste caso, todos ocupam a mesma posição em *ranking*. Além disso, vale ressaltar que no `GridSearchCV` pode ser feito um retreino utilizando os parâmetros cujo modelo teve melhor desempenho.

O Apêndice B contém as tabelas com os resultados obtidos usando validação cruzada para os parâmetros considerados na Tabela 14. Em cada uma das tabelas contidas no apêndice são apresentadas as combinações dos parâmetros, as médias e os desvios padrão obtidos.

Na implementação do modelo ℓ-DEP, resolvemos o Problema (5.12) utilizando o Algoritmo 5 por meio do pacote `CVXPY` [51], combinado com o solver `MOSEK` [91]. Para os classificadores SVC e MLP, usamos implementações do `sklearn` [21, 22, 90, 92]. A dilatação e a erosão que definem o modelo DEP foram treinadas utilizando o modelo DEP implementado por meio do pacote `CVXPY` [51] que possui uma extensão do método `DCCP`, combinado com o solver `MOSEK`. As implementações dos modelos MDN e HMLP-EL foram baseadas nos códigos disponíveis em [93, 94]. As compilações foram feitas utilizando o otimizador `Adam` e função perda sendo a Entropia Cruzada (3.11).

Tabela 14 – Valores fornecidos para o `tuned_parameters` do GridSearch para os conjuntos de dados sintéticos.

Modelo	**Candidatos**
MLP	$\eta \in \{50, 80, 100\}$
SVC	$p \in \{0.1, 1, 10, 100\}$ $kernel \in \{linear, rbf\}$
DEP	$ref_{\mathbf{r}} \in \{min, max, mean\}$ $C \in \{0.01, 0.1, 1\}$
ℓ-DEP:	$r_1, r_2 \in \{2, 3, 5, 7, 10\}$
MAXOUT MDN	$r_1 = r_2 \in \{2, 3, 5, 7, 10\}$ $c, r_1^1, r_2^1, r_1^2, r_2^2 \in \{5, 10, 15\}$
HMLP-EL	$m, l \in \{5, 10, 20\}$ $c \in \{0.01, 1, 10\}$

Conjunto de dados sintéticos

Dataset de Ripley

O primeiro conjunto de dados utilizado é o *dataset* de Ripley [96]. Este *dataset* contém dados sintéticos com 250 amostras de treinamento de 1000 amostras de teste.

Na Tabela 15 são apresentados os valores da métrica F_1Score obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho modelo foi melhor, cujos parâmetros são os apresentados na Tabela 20. Embora os escores apresentados na tabelas são em relação a medida F_1Score, vale ressaltar que cada classe do conjunto tem

Tabela 15 – F_1Score do Treino e Teste dos classificadores usando os melhores parâmetros encontrados na busca exaustiva para o *dataset* de Ripley (refit).

F_1**Score**	MLP	SVC	DEP	ℓ-DEP	MAXOUT	MDN	HMLP-EL
Treino	0.863	0.874	0.853	0.911	0.786	0.884	0.891
Teste	0.909	0.906	0.866	0.886	0.820	0.922	0.909

uma distribuição bimodal conhecida e a melhor pontuação de acurácia do conjunto de teste é de aproximadamente 0.92. Uma ilustração do comportamento de cada um dos classificadores nos conjuntos de teste é apresentado na Figura 5.1.

Observe que, pelos valores da Tabela 15, o modelo ℓ-DEP obteve melhor F_1score no treinamento. Já no conjunto de teste, o modelo MDN obteve melhor desempenho. Além disso, o modelo ℓ-DEP obteve desempenho superior aos modelos DEP e MAXOUT.

Dataset Double Moons

O Double Moons consiste em um conjunto de dados sintéticos que pode ser gerado utilizando o comando `make_moons` do `sklearn`. O conjunto é formado por duas semicircunferências (luas) que se intercalam. Utilizamos para o conjunto de treino e teste um total de 1000 e 2000 padrões, respectivamente. Incluímos no conjunto ruído gaussiano com desvio padrão $\sigma = 10^{-1}$.

Na Tabela 16 são apresentados os valores da métrica F_1Score obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho do modelo foi melhor, considerando os parâmetros conforme apresentado na Tabela 20. Observe que apenas os modelos ℓ-DEP e SVC conseguiram classificar 100% dos elementos no conjunto de treino. No conjunto de treino e teste todos os modelos obtiveram ótimos desempenhos, exceto o modelo DEP.

Uma ilustração do comportamento de cada um dos classificadores nos conjuntos de teste é apresentado na Figura 5.2.

Tabela 16 – F_1Score do Treino e Teste dos classificadores usando os melhores parâmetros encontrados na busca exaustiva para o *dataset* Double Moons (refit).

F_1**Score**	MLP	SVC	DEP	ℓ-DEP	MAXOUT	MDN	HMLP-EL
Treino	0.986	1.000	0.677	1.000	0.992	0.994	0.997
Teste	0.987	1.000	0.675	0.999	0.994	0.995	0.998

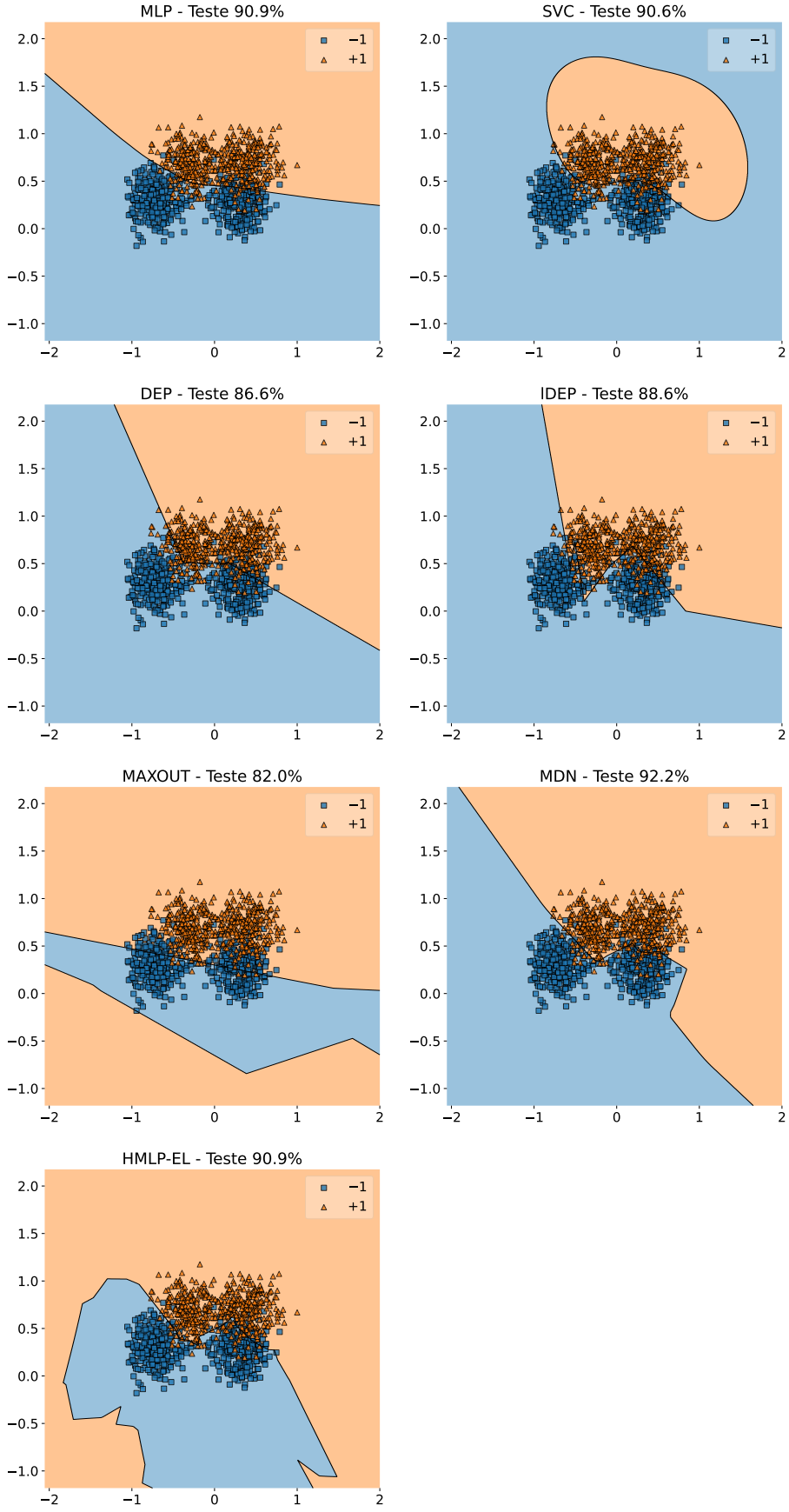


Figura 5.1 – *dataset* de Ripley: superfície de decisão e desempenho em relação à métrica F_1Score (%) dos classificadores resultantes treinados utilizando GridSearch no *dataset* de Ripley.

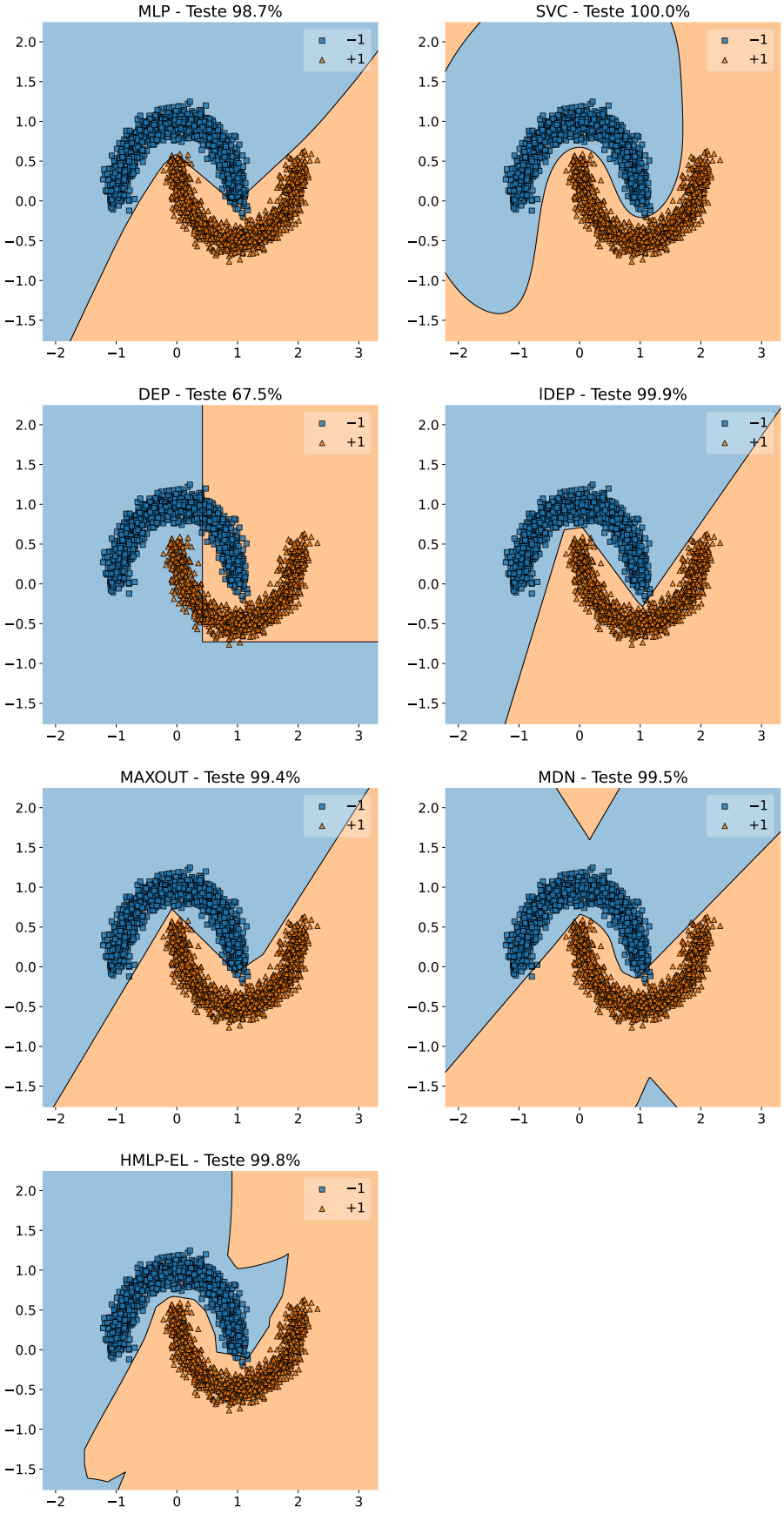


Figura 5.2 – *Dataset* Double Moons: superfície de decisão e desempenho em relação à métrica F_1Score (%) dos classificadores resultantes treinados utilizando GridSearch no *dataset* Double Moons.

Conjunto de dados reais

Na seção anterior apresentamos alguns exemplos da aplicação do GridSeach para encontrar os melhores parâmetros para alguns modelos de classificação. Vimos que o desempenho do modelo ℓ-DEP foi bastante competitivo entre os melhores modelos treinados nos conjuntos de dados sintéticos. Nesta seção, faremos a mesma busca de parâmetros para conjuntos de dados reais. O primeiro *dataset* real que utilizaremos é o conjunto de dados Tic-tac-toe. O segundo é o conjunto de dados Breast Cancer Wisconsin e, por último, o conjunto de dados Hill-Valley.

Dataset Tic-Tac-Toe - Jogo da velha

Esse conjunto contém 958 amostras obtidas de codificações de partidas do conhecido jogo-da-velha. Cada amostra do conjunto contém 9 características pertencentes ao conjunto $\{0, 1, 2\}$, que simbolizam as jogadas feitas pelos jogadores. Este conjunto possui apenas jogos em que houve um vencedor, ou seja, as classes dos dados são vencer ou perder, considerando que "x" foi o primeiro jogador.

A Tabela 17 contém os resultados da métrica F_1Score obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho do modelo foi melhor. Os únicos modelos que obtiveram desempenho acima de 90% foram os modelos SVC, ℓ-DEP e MDN. A rede MAXOUT teve um péssimo desempenho tanto no conjunto de treino quanto no conjunto de teste.

Tabela 17 – F_1Score do Treino e Teste dos classificadores usando os melhores parâmetros encontrados na busca exaustiva para o *dataset* Tic-tac-toe (refit).

F_1**Score**	MLP	SVC	DEP	ℓ-DEP	MAXOUT	MDN	HMLP-EL
Treino	0.883	1.000	0.667	0.914	0.347	0.923	0.744
Teste	0.871	1.000	0.696	0.908	0.312	0.925	0.762

Dataset Breast Cancer Wisconsin

O conjunto de dados Breast Cancer Wisconsin é referente a um problema de classificação binária em que os dados são os registros de acompanhamento de um caso de câncer de mama. Os elementos do conjunto estão relacionados a pacientes atendidos desde 1984 no centro de ciências clínicas da universidade de Wisconsin. O conjunto inclui apenas os casos que apresentam câncer de mama invasivo e nenhuma evidência de metástases à distância no momento do diagnóstico.

A Tabela 18 contém os resultados da métrica F_1Score obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho modelo foi melhor, ou seja, é feito um novo treinamento usando os melhores parâmetros que estão apresentados na Tabela 20. O único modelo que obteve 100% de desempenho foi o modelo ℓ-DEP, tanto no

conjunto de teste quanto no conjunto de treino. O modelo MAXOUT foi o único que não obteve desempenho acima de 90%.

Tabela 18 – F_1Score do Treino e Teste dos classificadores usando os melhores parâmetros encontrados na busca exaustiva para o *dataset* Breast Cancer Wisconsin (refit).

F_1**Score**	MLP	SVC	DEP	ℓ-DEP	MAXOUT	MDN	HMLP-EL
Treino	0.937	0.967	0.916	1.000	0.627	0.909	0.917
Teste	0.965	0.972	0.930	1.000	0.622	0.958	0.930

Dataset Hill-Valley

Este conjunto de dados é composto por elementos que simbolizam uma Colina ou um Vale. Para cada registro, que contém 100 pontos, os elementos podem ser visualizados em um gráfico bidimensional em ordem (um eixo de 1 a 100 e o outro os valores das entradas). Tais pontos criarão uma Colina ou um Vale. A versão utilizada é sem ruído, com conjuntos de treinamento e teste mesclados.

A Tabela 19 contém os resultados da métrica F_1Score obtidos pelo modelo treinado usando os parâmetros para os quais o desempenho do modelo foi melhor, cujos parâmetros são apresentados na Tabela 20. O único modelo que obteve 100% de desempenho foi o modelo ℓ-DEP, tanto no conjunto de teste quanto no conjunto de treino. Além do modelo SVC, nenhum outro modelo obteve desempenho superior a 70% em relação à métrica F_1Score.

Tabela 19 – F_1Score do Treino e Teste dos classificadores usando os melhores parâmetros encontrados na busca exaustiva para o *dataset* Hill-Valley (refit).

F_1**Score**	MLP	SVC	DEP	ℓ-DEP	MAXOUT	MDN	HMLP-EL
Treino	0.666	0.992	0.540	1.000	0.500	0.635	0.593
Teste	0.644	0.990	0.551	1.000	0.469	0.634	0.568

Observações sobre os resultados obtidos com a busca exaustiva

De forma geral, a busca exaustiva mostrou que não é necessário uma grande quantidade de neurônios para as redes aplicadas nos conjuntos de dados reais. Com efeito, o uso de quantidades muito grande de neurônios para as redes pode resultar num sobre-ajuste no conjunto de treino fazendo com que o desempenho fique ruim nos conjuntos de testes.

A Tabela 20 apresentam os melhores parâmetros para cada conjunto de dados utilizadoes na subseção anterior. Com base nos resultados apresentados, em cada um dos *datasets*, não é possível identificar em todos os casos um valor único geral para os hiper-parâmetros para cada um dos modelos.

Tabela 20 – Melhores parâmetros para cada um dos conjunto de dados de Ripley, Double Moons, Tic-Tac-Toe (TTT), Breast Cancer Wisconsin (BCW) e Hill-Valley (HV).

Modelo	**Parâmetros**	**Ripley**	**Moons**	**TTT**	**BCW**	**HV**
MLP	η	100	100	80	50	100
SVC	p	1	10	100	1	1
	kernel	rbf	rbf	rbf	linear	linear
DEP	C	0.1	0.1	1	0.01	1
	ref_r	max	min	max	max	min
ℓ-DEP	r_1	10	10	7	3	7
	r_2	5	10	3	5	10
MAXOUT	r_1	10	7	10	2	2
MDN	c	15	15	15	15	15
	r_1^1	5	15	10	10	5
	r_2^1	15	15	15	10	5
	r_1^2	10	5	15	5	5
	r_2^2	10	15	15	10	10
HMLP-EL	C	0.01	0.01	0.1	10	10
	l	5	5	20	20	5
	m	5	10	20	20	20

5.3.2 Análise empírica de desempenho em tarefas de classificação binária

Na Tabela 20 consta um resumo dos resultados dos testes feitos na seção anterior. São apresentados os parâmetros do classificador ganhador da busca exaustiva realizada em cada um dos *datasets* para os exemplos de conjuntos de dados.

Consideranndo a Tabela 20, observe que em alguns casos, os parâmetros do modelo que obteve melhor resultado dividem o primeiro lugar com outro conjunto de parâmetros. Por exemplo, o modelo MDN, no caso do *dataset* Double Moons, teve desempenho de teste semelhante utilizando parâmetros distintos. O mesmo ocorre no caso de outras redes no *dataset* Breast Cancer Wisconsin.

Para uma melhor análise, o ideal seria buscar os melhores parâmetros para cada *dataset* particularmente, mas iremos buscar um padrão utilizando uma análise empírica baseada na tabela de resultados do GridSearch.

Na Tabela 22 são apresentados os *datasets* utilizados neste trabalho para fazermos a análise empíricas dos modelos de classificação, tal inspiração veio da lista de datasets apresentados em [32]. Em relação ao modelo ℓ-DEP, para todos os conjuntos de dados da Tabela 22, adotamos os mesmos hiperparâmetros $r_1 = r_2 = 10$. Consideramos $r_1 = r_2 = 10$ para a rede MAXOUT [21]. Por fim, os hiperparâmetros dos demais modelos são configurados para os mesmos valores padrões considerados no capítulo anterior. A Tabela 21 resume os hiperparâmetros utilizados em nossos experimentos computacionais.

Tabela 21 – Escolha dos valores dos parâmetros utilizados nas implementações.

Modelo	**Parâmetros utilizados**
MLP	$\#h = 100$
SVC	$p = 1$ $kernel = \text{rbf}$
DEP	$ref_{\mathbf{r}} = \max$ $C = 1$
ℓ-DEP:	$r_1 = r_2 = 10$
MAXOUT:	$r_1 = r_2 = 10$
MDN	$r_1^1 = r_2^1 = 15$ $c = 15$ $r_1^2 = r_2^2 = 15$
HMLP-EL	$m = 10$ $l = 10$ $C = 10$

Tabela 22 – Informações sobre os *Datasets* de tarefas de Classificação do OpenML

	Nome	**Instâncias**	**Características**	**Código OpenML**	**Versão**
D1	ACCUTE INFLAMMATIONS	120	6	acute-inflammations	1
D2	AUSTRALIAN	690	14	Australian	4
D3	BANKNOTE	1372	4	banknote-authentication	1
D4	BLOOD TRANSFUSION	748	4	blood-transfusion-service-center	1
D5	BREAST CANCER WISCONSIN	569	30	wdbc	1
D6	CHESS	3196	36	kr-vs-kp	1
D7	COLIC	368	22	colic	2
D8	CREDIT APPROVAL	690	15	credit-approval	1
D9	CREDIT-G	1000	20	credit-g	1
D10	CYLINDER BANDS	540	37	cylinder-bands	2
D11	DIABETES	768	8	diabetes	1
D12	HABERMAN	306	3	haberman	1
D13	HILL-VALLEY	1212	100	hill-valley	1
D14	IONOSPHERE	351	34	ionosphere	1
D15	MOFN-3-7-10	1324	10	mofn-3-7-10	1
D16	MONKS-2	601	6	monks-problems-2	1
D17	SONAR	208	60	sonar	1
D18	STEEL PLATES FAULT	1941	33	steel-plates-fault	1
D19	THORACIC SURGERY	470	16	thoracic_surgery	1
D20	TIC-TAC-TOE	958	9	tic-tac-toe	1
D21	TITANIC	2201	3	Titanic	2
D22	ILPD	583	10	ilpd	1

Para os dados de tarefas de classificação binária também lidamos com dados faltantes substituindo-os pelo valor mais frequente nos atributos para todos os dados, usando o comando `SimpleImputer()` do `sklearn`. Além disso, de modo a tratar possíveis dados desbalanceados, particionamos o conjunto de dados em conjuntos de treinamento e teste usando o comando `sklearn StratifiedKFold()` com $k = 5$.

Finalmente, tendo em vista realizar uma boa avaliação nos dados, sendo ou não desbalanceados, usamos a métrica de avaliação F_1Score para medir o desempenho dos classificadores. Também incluímos nos testes as pontuações da acurácia balanceada para fins de comparação dos resultados.

A Tabela 23 contém a média e o desvio padrão da pontuação F_1Score obtida pelos classificadores usando validação cruzada estratificada de 5-fold. A Tabela 24 contém a média e o desvio padrão da acurácia balanceada obtida dos classificadores usando validação cruzada estratificada de 5-fold. A Tabela 25 contém os tempos necessários para treinar os classificadores para cada conjunto de dados. Os *boxplots* mostrados na Figura 5.4 resumem as pontuações médias do F_1 Score (topo), da acurácia balanceada (centro) e do tempo médio de processamento (abaixo) apresentadas, respectivamente, nas tabelas Tabela 23, Tabela 24 e Tabela 25.

Os resultados apresentados nas tabelas e imagens citadas acima mostraram que o modelo ℓ-DEP apresentou um desempenho competitivo ou superior aos classificadores híbridos morfológicos, considerando tanto o F_1Score quanto à acurácia. Considerando ainda as duas medidas em ambos os casos o modelo ℓ-DEP foi superior aos modelos morfológicos e apresentou um desempenho superior aos classificadores HMLP-EL, MAXOUT e DEP. Os modelos HMLP-EL e SVC foram os modelos com tempo de processamento mais rápido, seguidos do MAXOUT, MLP, DEP, MDN e por fim o ℓ-DEP.

Utilizamos o teste de hipóteses de Wilcoxon, com um nível de confiança de 95%, para comparar o desempenho de cada par de classificadores. O teste confirmou os resultados citados anteriormente e com isso, concluímos que o modelo é em muitos casos superior aos demais modelos. E mais, em relação aos modelos de estado da arte o modelo ℓ-DEP foi competitivo. Isso confirma o potencial do modelo ℓ-DEP em tarefas e classificação.

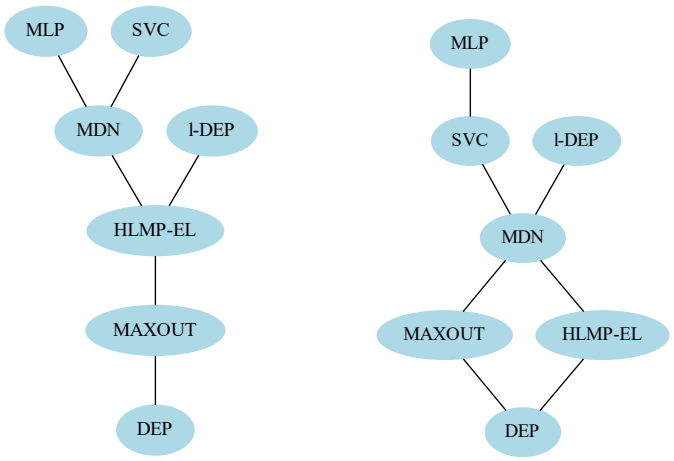


Figura 5.3 – Diagramas de Hasse: Teste de Hipótese de Wilcoxon, com nível de confiança de 95% com desempenho dos classificadores em relação ao F_1Score (esquerdo) e à acurácia balanceada (direito).

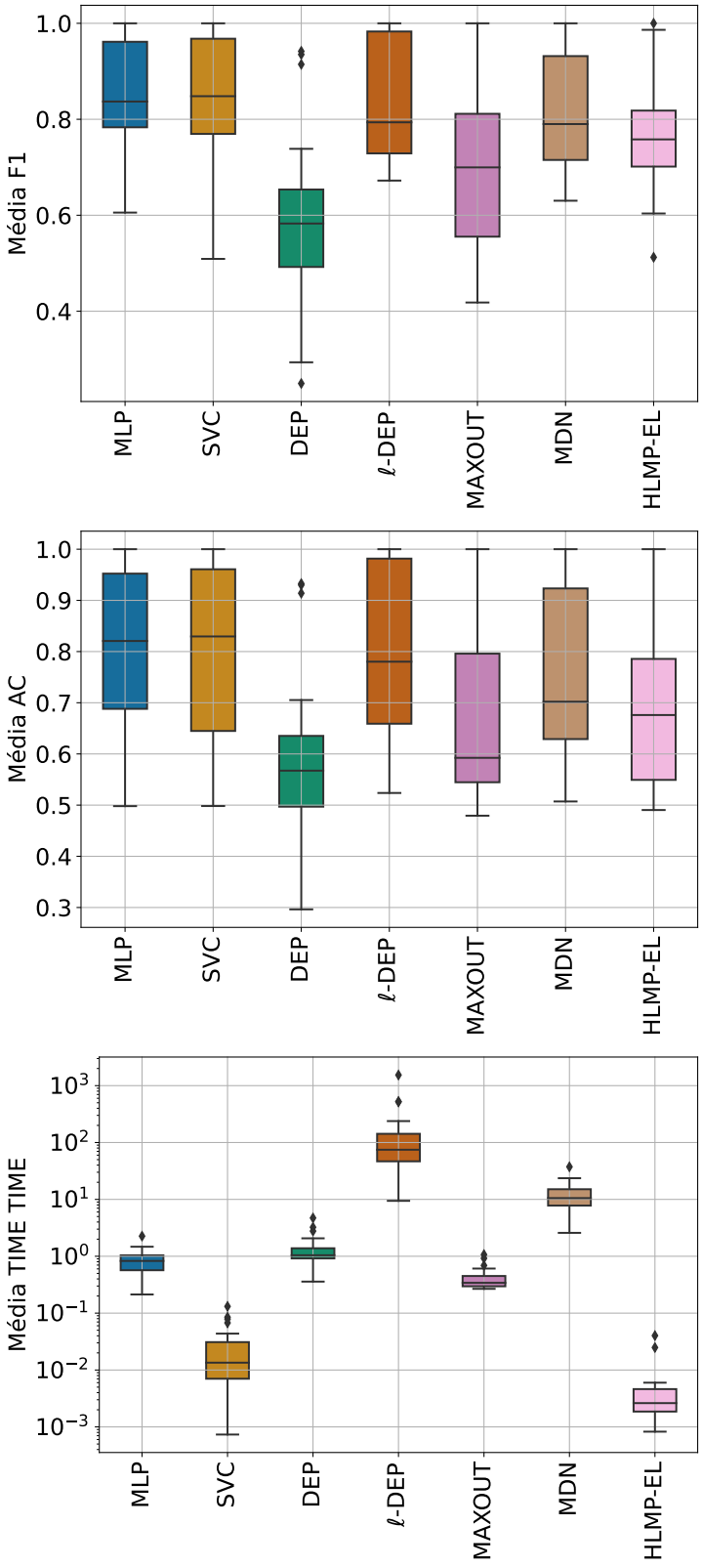


Figura 5.4 – *Boxplot* das médias do F_1Score (F1), da acurácia balanceada (AC) e do Tempo (TIME) necessário para treinar os classificadores binários.

Tabela 23 – Média e desvio padrão do F_1Score dos *Datasets* da Tabela 22.

F_1**Score**	MLP	SVC	DEP	l-DEP	MAXOUT	MDN	HLMP-EL
D1	1.000 ± 0.000	1.000 ± 0.000	0.942 ± 0.081	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
D2	0.846 ± 0.058	0.841 ± 0.047	0.456 ± 0.061	0.722 ± 0.052	0.462 ± 0.035	0.630 ± 0.105	0.635 ± 0.056
D3	0.745 ± 0.049	0.742 ± 0.031	0.738 ± 0.048	0.712 ± 0.045	0.732 ± 0.024	0.722 ± 0.027	0.752 ± 0.026
D4	0.914 ± 0.039	0.940 ± 0.028	0.914 ± 0.042	0.889 ± 0.052	0.418 ± 0.121	0.909 ± 0.013	0.780 ± 0.049
D5	0.818 ± 0.054	0.845 ± 0.040	0.587 ± 0.065	0.764 ± 0.019	0.667 ± 0.208	0.758 ± 0.020	0.726 ± 0.023
D6	0.828 ± 0.009	0.851 ± 0.000	0.294 ± 0.212	0.751 ± 0.026	0.802 ± 0.040	0.853 ± 0.005	0.851 ± 0.000
D7	0.783 ± 0.040	0.785 ± 0.030	0.569 ± 0.044	0.709 ± 0.046	0.543 ± 0.062	0.639 ± 0.055	0.604 ± 0.020
D8	0.977 ± 0.013	0.977 ± 0.018	0.935 ± 0.021	0.954 ± 0.021	0.861 ± 0.051	0.942 ± 0.025	0.923 ± 0.017
D9	0.717 ± 0.028	0.708 ± 0.005	0.362 ± 0.029	0.672 ± 0.033	0.659 ± 0.040	0.693 ± 0.030	0.708 ± 0.013
D10	0.800 ± 0.031	0.715 ± 0.018	0.491 ± 0.062	0.907 ± 0.047	0.782 ± 0.029	0.699 ± 0.035	0.634 ± 0.022
D11	0.872 ± 0.037	0.867 ± 0.039	0.680 ± 0.150	0.778 ± 0.032	0.814 ± 0.052	0.829 ± 0.018	0.820 ± 0.031
D12	0.868 ± 0.038	0.864 ± 0.034	0.630 ± 0.142	0.788 ± 0.031	0.552 ± 0.170	0.801 ± 0.038	0.812 ± 0.032
D13	0.802 ± 0.008	0.774 ± 0.015	0.563 ± 0.041	0.798 ± 0.017	0.735 ± 0.020	0.798 ± 0.015	0.765 ± 0.015
D14	0.762 ± 0.026	0.762 ± 0.019	0.661 ± 0.035	0.680 ± 0.039	0.618 ± 0.166	0.743 ± 0.023	0.741 ± 0.016
D15	0.869 ± 0.019	0.888 ± 0.027	0.615 ± 0.011	0.998 ± 0.005	1.000 ± 0.000	0.779 ± 0.046	0.706 ± 0.021
D16	0.756 ± 0.021	0.768 ± 0.014	0.603 ± 0.102	0.706 ± 0.019	0.620 ± 0.090	0.713 ± 0.031	0.700 ± 0.019
D17	0.606 ± 0.077	0.509 ± 0.027	0.548 ± 0.030	0.816 ± 0.012	0.499 ± 0.001	0.665 ± 0.033	0.512 ± 0.025
D18	1.000 ± 0.000	1.000 ± 0.000	0.616 ± 0.259	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.986 ± 0.009
D19	1.000 ± 0.000	1.000 ± 0.000	0.466 ± 0.038	1.000 ± 0.000	0.999 ± 0.002	0.996 ± 0.006	0.984 ± 0.006
D20	0.999 ± 0.001	0.997 ± 0.002	0.497 ± 0.135	1.000 ± 0.000	0.790 ± 0.123	0.980 ± 0.010	0.786 ± 0.027
D21	0.783 ± 0.009	0.780 ± 0.007	0.249 ± 0.017	0.789 ± 0.006	0.466 ± 0.192	0.782 ± 0.004	0.776 ± 0.009
D22	0.994 ± 0.003	0.984 ± 0.009	0.579 ± 0.116	0.992 ± 0.005	0.565 ± 0.027	0.939 ± 0.014	0.659 ± 0.012
Média	0.852 ± 0.107	0.845 ± 0.122	0.591 ± 0.178	0.838 ± 0.120	0.708 ± 0.183	0.812 ± 0.120	0.766 ± 0.124
Mediana	0.846 ± 0.026	0.845 ± 0.018	0.587 ± 0.061	0.798 ± 0.021	0.708 ± 0.040	0.798 ± 0.023	0.765 ± 0.020

Tabela 24 – Média e desvio padrão da acurácia balanceada dos *Datasets* da Tabela 22.

Acurácia B	MLP	SVC	DEP	l-DEP	MAXOUT	MDN	HLMP-EL
D1	1.000 ± 0.000	1.000 ± 0.000	0.930 ± 0.097	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
D2	0.846 ± 0.055	0.840 ± 0.046	0.457 ± 0.067	0.714 ± 0.050	0.479 ± 0.048	0.628 ± 0.099	0.636 ± 0.056
D3	0.578 ± 0.071	0.564 ± 0.032	0.668 ± 0.040	0.548 ± 0.063	0.534 ± 0.084	0.541 ± 0.067	0.578 ± 0.038
D4	0.889 ± 0.050	0.924 ± 0.034	0.914 ± 0.028	0.873 ± 0.055	0.506 ± 0.006	0.892 ± 0.021	0.741 ± 0.065
D5	0.802 ± 0.069	0.819 ± 0.045	0.597 ± 0.036	0.753 ± 0.032	0.634 ± 0.160	0.716 ± 0.039	0.697 ± 0.034
D6	0.498 ± 0.024	0.500 ± 0.000	0.514 ± 0.021	0.524 ± 0.009	0.501 ± 0.028	0.507 ± 0.016	0.500 ± 0.000
D7	0.777 ± 0.035	0.774 ± 0.032	0.543 ± 0.039	0.702 ± 0.045	0.544 ± 0.027	0.614 ± 0.069	0.559 ± 0.025
D8	0.973 ± 0.017	0.973 ± 0.022	0.933 ± 0.026	0.949 ± 0.022	0.831 ± 0.066	0.934 ± 0.031	0.918 ± 0.013
D9	0.604 ± 0.030	0.498 ± 0.006	0.475 ± 0.031	0.607 ± 0.027	0.547 ± 0.056	0.533 ± 0.032	0.502 ± 0.008
D10	0.783 ± 0.035	0.636 ± 0.014	0.564 ± 0.066	0.901 ± 0.049	0.755 ± 0.054	0.631 ± 0.032	0.490 ± 0.016
D11	0.870 ± 0.039	0.865 ± 0.040	0.705 ± 0.138	0.774 ± 0.035	0.809 ± 0.063	0.824 ± 0.022	0.816 ± 0.030
D12	0.867 ± 0.037	0.866 ± 0.033	0.644 ± 0.140	0.786 ± 0.030	0.553 ± 0.181	0.797 ± 0.036	0.799 ± 0.033
D13	0.648 ± 0.016	0.558 ± 0.036	0.448 ± 0.048	0.646 ± 0.024	0.587 ± 0.103	0.651 ± 0.024	0.546 ± 0.020
D14	0.728 ± 0.024	0.716 ± 0.013	0.608 ± 0.056	0.649 ± 0.052	0.602 ± 0.140	0.694 ± 0.023	0.696 ± 0.026
D15	0.839 ± 0.025	0.844 ± 0.036	0.491 ± 0.011	0.997 ± 0.007	1.000 ± 0.000	0.710 ± 0.078	0.601 ± 0.019
D16	0.688 ± 0.026	0.671 ± 0.019	0.571 ± 0.067	0.651 ± 0.028	0.598 ± 0.024	0.598 ± 0.041	0.533 ± 0.026
D17	0.606 ± 0.077	0.509 ± 0.026	0.548 ± 0.030	0.816 ± 0.012	0.499 ± 0.002	0.665 ± 0.033	0.512 ± 0.025
D18	1.000 ± 0.000	1.000 ± 0.000	0.595 ± 0.051	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.973 ± 0.022
D19	1.000 ± 0.000	1.000 ± 0.000	0.458 ± 0.037	1.000 ± 0.000	0.999 ± 0.001	0.996 ± 0.006	0.986 ± 0.005
D20	0.999 ± 0.002	0.996 ± 0.003	0.514 ± 0.026	1.000 ± 0.000	0.709 ± 0.176	0.978 ± 0.009	0.746 ± 0.028
D21	0.690 ± 0.015	0.698 ± 0.020	0.296 ± 0.016	0.683 ± 0.010	0.547 ± 0.112	0.690 ± 0.019	0.700 ± 0.015
D22	0.994 ± 0.003	0.984 ± 0.009	0.583 ± 0.104	0.992 ± 0.005	0.562 ± 0.037	0.939 ± 0.014	0.656 ± 0.013
Média	0.804 ± 0.153	0.783 ± 0.177	0.594 ± 0.157	0.799 ± 0.159	0.673 ± 0.181	0.752 ± 0.163	0.690 ± 0.162
Mediana	0.804 ± 0.026	0.819 ± 0.022	0.571 ± 0.040	0.786 ± 0.027	0.598 ± 0.054	0.710 ± 0.031	0.690 ± 0.025

Tabela 25 – Média e desvio padrão do Tempo gasto no treinamento dos *Datasets* da Tabela 22.

Tempo (s)	MLP	SVC	DEP	l-DEP	MAXOUT	MDN	HLMP-EL
D1	0.213 ± 0.070	0.001 ± 0.000	0.359 ± 0.038	9.430 ± 0.090	0.342 ± 0.315	2.581 ± 0.111	0.003 ± 0.004
D2	0.293 ± 0.045	0.003 ± 0.000	1.347 ± 0.069	22.530 ± 0.346	0.268 ± 0.011	5.145 ± 0.099	0.001 ± 0.000
D3	0.429 ± 0.088	0.003 ± 0.000	0.924 ± 0.093	25.621 ± 0.836	0.280 ± 0.012	5.853 ± 0.072	0.001 ± 0.000
D4	0.568 ± 0.168	0.003 ± 0.000	0.688 ± 0.014	35.421 ± 0.559	0.294 ± 0.038	6.430 ± 0.243	0.004 ± 0.004
D5	0.509 ± 0.125	0.006 ± 0.000	0.659 ± 0.013	34.470 ± 0.288	0.269 ± 0.006	6.763 ± 0.126	0.002 ± 0.004
D6	0.470 ± 0.025	0.007 ± 0.000	0.750 ± 0.024	44.028 ± 0.331	0.295 ± 0.023	7.613 ± 0.091	0.003 ± 0.004
D7	0.755 ± 0.213	0.013 ± 0.002	0.958 ± 0.015	64.287 ± 0.363	0.286 ± 0.009	8.405 ± 0.209	0.002 ± 0.001
D8	0.837 ± 0.360	0.013 ± 0.018	1.096 ± 0.355	89.856 ± 44.056	0.920 ± 0.828	11.703 ± 6.467	0.040 ± 0.050
D9	0.841 ± 0.159	0.020 ± 0.020	0.940 ± 0.265	56.462 ± 1.472	0.340 ± 0.055	9.363 ± 0.296	0.005 ± 0.005
D10	0.796 ± 0.213	0.013 ± 0.006	0.907 ± 0.047	54.717 ± 0.344	0.451 ± 0.110	9.027 ± 1.702	0.002 ± 0.001
D11	0.747 ± 0.127	0.014 ± 0.004	1.003 ± 0.200	79.347 ± 10.107	0.446 ± 0.269	10.625 ± 0.292	0.002 ± 0.001
D12	0.862 ± 0.172	0.013 ± 0.001	0.913 ± 0.040	73.198 ± 0.669	0.307 ± 0.003	10.453 ± 0.036	0.002 ± 0.001
D13	0.570 ± 0.186	0.016 ± 0.004	1.043 ± 0.033	67.329 ± 1.249	0.358 ± 0.034	11.264 ± 0.389	0.005 ± 0.005
D14	0.862 ± 0.104	0.015 ± 0.000	1.041 ± 0.083	76.127 ± 4.224	0.340 ± 0.023	11.497 ± 0.228	0.003 ± 0.004
D15	1.119 ± 0.181	0.028 ± 0.002	1.124 ± 0.028	99.691 ± 0.880	0.689 ± 0.062	13.240 ± 0.249	0.004 ± 0.004
D16	1.143 ± 0.240	0.032 ± 0.001	1.377 ± 0.055	145.786 ± 2.814	0.302 ± 0.037	8.336 ± 0.911	0.002 ± 0.001
D17	1.004 ± 0.173	0.085 ± 0.002	3.238 ± 0.088	526.019 ± 11.971	0.324 ± 0.004	15.699 ± 0.132	0.006 ± 0.005
D18	1.050 ± 0.266	0.068 ± 0.013	1.385 ± 0.020	156.968 ± 0.665	0.585 ± 0.055	16.987 ± 0.096	0.003 ± 0.004
D19	1.248 ± 0.175	0.008 ± 0.000	1.531 ± 0.038	131.605 ± 0.457	0.608 ± 0.084	17.339 ± 0.111	0.002 ± 0.001
D20	0.819 ± 0.172	0.044 ± 0.002	2.773 ± 0.005	518.576 ± 2.542	0.424 ± 0.012	23.440 ± 0.134	0.006 ± 0.005
D21	1.471 ± 0.262	0.079 ± 0.010	2.069 ± 0.021	238.123 ± 5.707	0.439 ± 0.047	23.587 ± 5.394	0.002 ± 0.001
D22	2.259 ± 0.170	0.131 ± 0.011	4.716 ± 0.401	1537.303 ± 237.182	1.070 ± 1.181	37.390 ± 1.311	0.025 ± 0.042
Média	0.858 ± 0.431	0.028 ± 0.033	1.402 ± 0.977	185.768 ± 325.127	0.438 ± 0.211	12.397 ± 7.608	0.006 ± 0.009
Mediana	0.837 ± 0.172	0.014 ± 0.002	1.043 ± 0.040	76.127 ± 0.880	0.342 ± 0.038	10.625 ± 0.228	0.003 ± 0.004

6 Considerações Finais

Neste trabalho introduzimos a rede neural morfológica *perceptron* de dilatação-erosão linear dada por uma combinação convexa da composição de transformações lineares e duas operações elementares da morfologia matemática. Apresentamos duas aplicações, uma para tarefas de regressão: o regressor ℓ-DEP, o qual denominamos ℓ-DER, e outra para tarefas de classificação: o classificador ℓ-DEP, que decidimos manter denominado por ℓ-DEP.

Especificamente, um regressor ℓ-DER é definido por meio da equação $y = \tau^{\ell}(\mathbf{x})$, onde a função preditiva $\tau^{\ell} : \mathbb{R}^n \to \mathbb{R}$ é definida em (3.16). O classificador ℓ-DEP é definido utilizando um mapeamento bijetivo σ, que toma valores do conjunto de rótulos de classe $\mathbb{C}$ e mapeia ao conjunto $\{+1, -1\}$. A classificação é feita por meio da equação $y = \sigma^{-1} f \tau^{\ell}(\mathbf{x})$ em que τ^{ℓ} é a função de decisão. A função τ^{ℓ} é expressa por meio de matrizes $W \in \mathbb{R}^{r_1 \times n}$ e $M \in \mathbb{R}^{r_2 \times n}$ e vetores $\mathbf{c} \in \mathbb{R}^{r_1}$ e $\mathbf{d} \in \mathbb{R}^{r_2}$. Os parâmetros W, M, $\mathbf{c}$ e $\mathbf{d}$ são auto-ajustados no treinamento dos modelos ℓ-DEP.

Do ponto de vista teórico, a função τ^{ℓ} é uma função linear contínua por partes [34]. Como consequência, um modelo ℓ-DEP pode, em princípio, resolver qualquer problema em que a função τ^{ℓ} possa ser utilizada como função preditiva.

Para o treinamento da rede ℓ-DEP como modelo preditivo de regressão e de classificação, neste trabalho, investigamos um problema de programação côncavo-convexa. O regressor e o classificador são treinados resolvendo o problema de programação convexo-côncavo, inspirado nos trabalhos de Charisopoulos e Maragos [30].

Experimentos computacionais com diversos problemas de regressão e de classificação binária revelaram o desempenho da rede neural morfológica ℓ-DEP proposta, treinada utilizando o procedimento côncavo-convexo. Foram apresentados resultados competitivos com outros modelos preditivos da literatura, mostrando o potencial da rede ℓ-DEP.

Trabalhamos na busca de melhores hiper-parâmetros r_1 e r_2 da função τ^{ℓ}. Para isso, selecionaremos alguns *datasets* e exploramos a rede morfológica de modo a encontrar melhores parâmetros para os modelos preditivos ℓ-DEP.

Os resultados obtidos nessa tese nos renderam alguns trabalhos acadêmicos.

- Resumo - Linear Dilation-Erosion Perceptron for Binary Classification - Encontro Científico de Pós-Graduandos do IMECC (EnCPos 2020) - Boletim Digital - [97]
- Capítulo de Livro - Linear Dilation-Erosion Perceptron Trained Using a Convex-

Concave Procedure - 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020) - Advances in Intelligent Systems and Computing (Springer) - [33]

- Capítulo de Livro - Least-Squares Linear Dilation-Erosion Regressor trained using a Convex-Concave Procedure - 11th Brazilian Conference on Intelligent Systems (BRACIS 2022) - Intelligent Systems (Springer) - [35]

Como trabalhos futuros, pode ser feita uma melhor reflexão sobre os resultados apresentados neste texto com objetivo de identificar melhores aplicações da rede morfológica ℓ-DEP em tarefas diversas. Além disso, pode-se buscar uma melhor identificação e descrição sobre quais os cenários em que nossa abordagem é mais apropriada em relação às outras técnicas já existentes.

Nos subproblemas de otimização dos algoritmos apresentados, pode-se buscar uma solução por meio de métodos de otimização inexata para os problemas quadráticos [98, 99] ou usar resoluções por quadrados mínimos utilizando técnicas de regularização [61].

Referências

1 AURELIEN, G. *Hands–On Machine Learning with Scikit–Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. ed. Sebastopol, California, USA.: O Reilly, 2019. 0–600 p. ISBN 1492032646.

2 VAPNIK, V. N. *The Nature of Statistical Learning Theory*. 2. ed. [S.l.]: Springer, 1999.

3 HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN-10 0132733501. ISBN 978-0-13-273350-2.

4 HOPFIELD, J. J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, v. 79, p. 2554–2558, 1982.

5 ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, v. 65, p. 386–408, 1958.

6 HAYKIN, S. *Neural Networks and Learning Machines*. 3ª edição. ed. New York: Prentice Hall, 2008. ISBN 978-0-13-147139-9.

7 RITTER, G. X.; WILSON, J. N.; DAVIDSON, J. L. Image Algebra: An Overview. *Computer Vision, Graphics, and Image Processing*, v. 49, n. 3, p. 297–331, 1990.

8 HEIJMANS, H. J. A. M. Mathematical Morphology: A Modern Approach in Image Processing Based on Algebra and Geometry. *SIAM Review*, v. 37, n. 1, p. 1–36, 1995.

9 SOILLE, P. *Morphological Image Analysis*. Berlin: Springer Verlag, 1999.

10 RITTER, G. X.; SUSSNER, P. An Introduction to Morphological Neural Networks. In: *Proceedings of the 13th International Conference on Pattern Recognition*. Vienna, Austria: [s.n.], 1996. p. 709–717.

11 RITTER, G. X.; SUSSNER, P. Morphological Perceptrons. In: *ISAS'97, Intelligent Systems and Semiotics*. Gaithersburg, Maryland: [s.n.], 1997.

12 RITTER, G. X.; SUSSNER, P.; DIAZ-DE-LEON, J. L. Morphological associative memories. *IEEE Transactions on Neural Networks*, v. 9, n. 2, p. 281–293, 1998. ISSN 10459227.

13 CHANG, W.; HAMAD, H.; CHUGG, K. M. *Approximation Capabilities of Neural Networks using Morphological Perceptrons and Generalizations*. [S.l.]: arXiv, 2022. ArXiv:2207.07832 [cs].

14 RITTER, G. X.; URCID, G. Lattice Algebra Approach to Single-Neuron Computation. *IEEE Transactions on Neural Networks*, v. 14, n. 2, p. 282–295, 2003.

15 SUSSNER, P. Morphological perceptron learning. In: *IEEE International Symposium on Intelligent Control - Proceedings*. [S.l.]: IEEE, 1998. p. 477–482.

16 SUSSNER, P.; ESMI, E. L. Morphological Perceptrons with Competitive Learning: Lattice-Theoretical Framework and Constructive Learning Algorithm. *Information Sciences*, v. 181, n. 10, p. 1929–1950, 2011.

17 PESSOA, L. F. C.; MARAGOS, P. Neural networks with hybrid morphological/rank/-linear nodes: a unifying framework with applications to handwritten character recognition. *Pattern Recognition*, v. 33, p. 945–960, 2000.

18 HERNáNDEZ, G. et al. Hybrid neural networks for big data classification. *Neurocomputing*, v. 390, p. 327–340, maio 2020. ISSN 0925-2312.

19 SUSSNER, P.; CAMPIOTTI, I. Extreme learning machine for a new hybrid morphological/linear perceptron. *Neural Networks*, v. 123, p. 288–298, mar. 2020. ISSN 0893-6080.

20 MONDAL, R. et al. *Morphological Network: How Far Can We Go with Morphological Neurons?* [S.l.], 2020. ArXiv:1901.00109 [cs, stat] type: article.

21 GOODFELLOW, I. et al. Maxout Networks. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2013. p. 1319–1327.

22 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

23 ARAúJO, R. d. A.; OLIVEIRA, A. L. I.; MEIRA, S. A morphological neural network for binary classification problems. *Engineering Applications of Artificial Intelligence*, v. 65, p. 12–28, out. 2017. ISSN 0952-1976.

24 PESSOA, L.; MARAGOS, P. Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition. *Pattern Recognition*, v. 33, p. 945–960, 2000.

25 MONDAL, R.; SANTRA, S.; CHANDA, B. *Dense Morphological Network: An Universal Function Approximator*. 2019.

26 KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, jan. 2017. ArXiv: 1412.6980.

27 HERNÁNDEZ, G. et al. Hybrid neural networks for big data classification. *Neurocomputing*, Elsevier B.V., v. 390, p. 327–340, 5 2020. ISSN 18728286.

28 HUANG, G.-B.; WANG, D.; LAN, Y. Extreme learning machines: a survey. *Int. J. Machine Learning & Cybernetics*, v. 2, n. 2, p. 107–122, 2011.

29 ARAúJO, R. d. A. A class of hybrid morphological perceptrons with application in time series forecasting. *Knowledge-Based Systems*, v. 24, n. 4, p. 513–529, 2011. ISSN 0950-7051.

30 CHARISOPOULOS, V.; MARAGOS, P. Morphological Perceptrons: Geometry and Training Algorithms. In: *Math. Morph. and Its Appli. to Signal and Image Proc.* [S.l.]: Springer, 2017. p. 3–15.

31 RITTER, G. X.; URCID, G. Lattice Algebra Approach to Single-Neuron Computation. *IEEE Transactions on Neural Networks*, v. 14, n. 2, p. 282–295, 2003.

32 VALLE, M. E. Reduced Dilation-Erosion Perceptron for Binary Classification. *Mathematics*, v. 8, n. 4, p. 512, 2020.

33 OLIVEIRA, A. L.; VALLE, M. E. Linear Dilation-Erosion Perceptron Trained Using a Convex-Concave Procedure. In: ABRAHAM, A. et al. (Ed.). *Proceedings of SoCPaR 2020*. Cham: Springer International Publishing, 2021. (Advances in Intelligent Systems and Computing), p. 245–255. ISBN 978-3-030-73689-7.

34 WANG, S. General constructive representations for continuous piecewise-linear functions. *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 51, n. 9, p. 1889–1896, 2004.

35 OLIVEIRA, A. L.; VALLE, M. E. Least-Squares Linear Dilation-Erosion Regressor Trained Using a Convex-Concave Procedure. In: XAVIER-JUNIOR, J. C.; RIOS, R. A. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2022. (Lecture Notes in Computer Science), p. 16–29. ISBN 978-3-031-21689-3.

36 SHEN, X. et al. Disciplined convex-concave programming. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. [S.l.: s.n.], 2016. p. 1009–1014.

37 FRIEDLANDER, A. *Elementos de programação não linear*. 1. ed. [S.l.]: Editora Unicamp, 1994.

38 MARTíNEZ, J. M.; SANTOS, S. A. *Métodos Computacionais de Otimização*. [S.l.: s.n.], 1995. Versão revisada - Setembro/2020.

39 GRANT, M.; BOYD, S.; YE, Y. Disciplined Convex Programming. In: *Global Optimization*. [S.l.]: Kluwer Academic Publishers, 2006. p. 155–210.

40 LUENBERGER, D. G. *Linear and Nonlinear Programming*. 2. ed. [S.l.]: Addison-Wesley, 1984.

41 THI, H. A. L.; NGAI, H. V.; DINH, T. DC Programming and DCA for General DC Programs. In: *Advances in Intelligent Systems and Computing*. [S.l.: s.n.], 2014. v. 282, p. 15–35. ISBN 978-3-319-06568-7.

42 KUMAR, A. et al. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, v. 56, p. 100693, ago. 2020. ISSN 2210-6502.

43 THI, H. A. L.; DINH, T. P. DC programming and DCA: thirty years of developments. *Mathematical Programming: Series A and B*, v. 169, n. 1, p. 5–68, maio 2018. ISSN 0025-5610.

44 HARTMAN, P. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, v. 9, n. 3, p. 707–713, set. 1959. ISSN 0030-8730. Publisher: Mathematical Sciences Publishers.

45 TUY, H. DC Functions and DC Sets. In: TUY, H. (Ed.). *Convex Analysis and Global Optimization*. Cham: Springer International Publishing, 2016, (Springer Optimization and Its Applications). p. 103–123. ISBN 978-3-319-31484-6. 10.1007/978-3-319-31484-6_4.

46 TAO, P. D.; SOUAD, E. B. Duality in D.C. (Difference of Convex functions) Optimization. Subgradient Methods. In: HOFFMANN, K.-H. et al. (Ed.). *Trends in Mathematical Optimization: 4th French-German Conference on Optimization*. Basel: Birkhäuser, 1988, (International Series of Numerical Mathematics). p. 277–293. ISBN 978-3-0348-9297-1.

47 SILVA, J. C. R. *Método Subgradiente Condicional com Sequência Ergódica*. Tese (Dissertação) — Universidade Federal de Goiás, Goiania, 2011. Instituto de Matemática e Estatística.

48 DINH, T. P.; THI, H. A. L. Convex analysis approach to DC programming: Theory, Algorithm and Applications. *Acta Mathematica Vietnamica*, v. 22, n. 1, p. 289, 1997.

49 YUILLE, A. L.; RANGARAJAN, A. The concave-convex procedure. *Neural Computation*, v. 15, n. 4, p. 915–936, 2003. ISSN 0899-7667.

50 LIPP, T.; BOYD, S. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, v. 17, n. 2, p. 263–287, nov. 2015. ISSN 1573-2924.

51 DIAMOND, S.; BOYD, S. *CVXPY: A Python-Embedded Modeling Language for Convex Optimization*. [S.l.]: -, 2016.

52 SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, jul. 1959. ISSN 0018-8646. Conference Name: IBM Journal of Research and Development.

53 KANG, M.; JAMESON, N. J. Machine Learning: Fundamentals. In: *Prognostics and Health Management of Electronics*. [S.l.]: John Wiley & Sons, Ltd, 2018. p. 85–109. ISBN 978-1-119-51532-6. Section: 4.

54 MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. Google-Books-ID: EoYBngEACAAJ. ISBN 978-0-07-115467-3.

55 ZHANG, Y. *New Advances in Machine Learning*. [S.l.]: BoD – Books on Demand, 2010. Google-Books-ID: XAqhDwAAQBAJ. ISBN 978-953-307-034-6.

56 BISHOP, C. *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006. (Information Science and Statistics). ISBN 978-0-387-31073-2.

57 TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to data mining*. [S.l.: s.n.], 2006. OCLC: 956940574. ISBN 978-0-321-42052-7.

58 VAPNIK, V. N. *Statistical Learning Theory*. New York, NY, USA: John Wiley and Sons, 1998.

59 SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing*, v. 14, n. 3, p. 199–222, 2004.

60 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, out. 1986. ISSN 1476-4687. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 6088 Primary_atype: Research Publisher: Nature Publishing Group.

61 ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, v. 67, n. 2, p. 301–320, 2005. ISSN 1467-9868. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2005.00503.x. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00503.x>.

62 CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, Springer-Verlag, v. 2, n. 4, p. 303–314, 12 1989. ISSN 09324194.

63 HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, jan. 1989. ISSN 0893-6080.

64 STONE, M. H. The Generalized Weierstrass Approximation Theorem. *Mathematics Magazine*, v. 21, n. 4, p. 167–184, 1948. ISSN 0025-570X. Publisher: Mathematical Association of America.

65 HEIJMANS, H. J. A. M. Mathematical Morphology: A Modern Approach in Image Processing Based on Algebra and Geometry. *SIAM Review*, v. 37, n. 1, p. 1–36, 1995.

66 SOILLE, P. *Morphological Image Analysis*. Berlin: Springer Verlag, 1999.

67 HEIJMANS, H. J. A. M.; RONSE, C. The algebraic basis of mathematical morphology I. Dilations and erosions. *Computer Vision, Graphics, and Image Processing*, v. 50, n. 3, p. 245–295, 1990.

68 BANON, G.; BARRERA, J. Decomposition of Mappings between Complete Lattices by Mathematical Morphology, Part 1. General Lattices. *Signal Processing*, v. 30, n. 3, p. 299–327, 1993.

69 HEIJMANS, H. *Morphological Image Operators*. New York, NY: Academic Press, 1994.

70 SERRA, J. *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances*. New York: Academic Press, 1988.

71 VALLE, M. E. *Fundamentos e Aplicações de Memórias Associativas Morfológicas Nebulosas*. Tese (PhD Thesis) — Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil, 2007.

72 GOUTSIAS, J.; HEIJMANS, H. J. A. M.; SIVAKUMAR, K. Morphological Operators for Image Sequences. *Computer vision and image understanding*, v. 62, p. 326–346, 1995.

73 VELASCO-FORERO, S.; ANGULO, J. Supervised Ordering in R^p: Application to Morphological Processing of Hyperspectral Images. *IEEE Transactions on Image Processing*, v. 20, n. 11, p. 3301–3308, nov. 2011.

74 VELASCO-FORERO, S.; ANGULO, J. Vector Ordering and Multispectral Morphological Image Processing. In: CELEBI, M. E.; SMOLKA, B. (Ed.). *Advances in Low-Level Color Image Processing*. Dordrecht: Springer Netherlands, 2014. p. 223–239.

75 SERRA, J. The "False Colour" Problem. In: WILKINSON, M. H.; ROERDINK, J. B. (Ed.). *Mathematical Morphology and Its Application to Signal and Image Processing*. [S.l.]: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5720). p. 13–23.

76 OVCHINNIKOV, S. Max-min representation of piecewise linear functions. *Beitrage zur Algebra und Geometrie*, v. 43, n. 1, p. 297–302, 2002.

77 TARELA, J. M.; ALONSO, E.; MARTíNEZ, M. V. A representation method for PWL functions oriented to parallel processing. *Mathematical and Computer Modelling*, v. 13, n. 10, p. 75 – 83, 1990. ISSN 0895-7177.

78 TARELA, J. M.; MARTíNEZ, M. V. Region configurations for realizability of lattice Piecewise-Linear models. *Mathematical and Computer Modelling*, v. 30, n. 11, p. 17 – 27, 1999. ISSN 0895-7177.

79 OVCHINNIKOV, S. Discrete piecewise linear functions. *European Journal of Combinatorics*, v. 31, n. 5, p. 1283 – 1294, 2010. ISSN 0195-6698.

80 CAMPONOGARA, E.; NAZARI, L. F. Models and Algorithms for Optimal Piecewise-Linear Function Approximation. *Mathematical Problems in Engineering*, v. 2015, p. 876862, jul. 2015. ISSN 1024-123X.

81 REBENNACK, S.; KALLRATH, J. Continuous Piecewise Linear Delta-Approximations for Bivariate and Multivariate Functions. *Journal of Optimization Theory and Applications*, v. 167, n. 1, p. 102–117, 2015. ISSN 1573-2878.

82 ALIPRANTIS, C. D.; HARRIS, D.; TOURKY, R. Continuous piecewise linear functions. *Macroeconomic Dynamics*, v. 10, n. 1, p. 77–99, fev. 2006. ISSN 1469-8056, 1365-1005. Publisher: Cambridge University Press.

83 GOROKHOVIK, V. V.; ZORKO, O. I.; BIRKHOFF, G. Piecewise affine functions and polyhedral sets. *Optimization*, v. 31, n. 3, p. 209–221, jan. 1994. ISSN 0233-1934. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/02331939408844018. Disponível em: <https://doi.org/10.1080/02331939408844018>.

84 WANG, S.; SUN, X. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, v. 51, n. 12, p. 4425–4431, 2005. ISSN 1557-9654.

85 MELZER, D. On the expressibility of piecewise-linear continuous functions as the difference of two piecewise-linear convex functions. *Mathematical Programming Study*, v. 29, p. 118–134, 1986.

86 WEN, C. et al. A compact f-f model of high-dimensional piecewise-linear function over a degenerate intersection. *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 52, n. 4, p. 815–821, 2005.

87 HO, V. T.; THI, H. A. L.; DINH, T. P. DCA-based algorithms for DC fitting. *Journal of Computational and Applied Mathematics*, v. 389, p. 113353, jun. 2021. ISSN 0377-0427.

88 KATSAVOUNIDIS, I.; KUO, C.-C. J.; ZHANG, Z. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, v. 1, n. 10, p. 144–146, out. 1994. ISSN 1558-2361. Conference Name: IEEE Signal Processing Letters.

89 MOLLER, J. *Lectures on Random Voronoi Tessellations*. New York: Springer-Verlag, 1994. (Lecture Notes in Statistics). ISBN 978-0-387-94264-3.

90 PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, v. 12, n. null, p. 2825–2830, nov. 2011. ISSN 1532-4435.

91 APS, M. *MOSEK Optimizer API for Python*. [S.l.]: -, 2020. Release 9.3.10.

92 HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999.

93 BARROS, L. F. D.; VALLE, M. E. V. Um estudo sobre redes neurais morfológicas para classificação de padrões. *In: XXIX CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNICAMP, 2021, Campinas. Anais eletrônicos.*, 1986.

94 CAMPIOTTI, I. *Morphological Perceptron and Hybrid Morphological/Linear Perceptron*. 2022. Disponível em: <https://github.com/israelcamp/hmlp>.

95 FRANK, E. et al. Technical Note: Naive Bayes for Regression. *Machine Learning*, v. 41, n. 1, p. 5–25, out. 2000. ISSN 1573-0565.

96 RIPLEY, B. D. *Pattern Recognition and Neural Networks*. Cambridge: Cambridge University Press, 1996.

97 OLIVEIRA, A.; VALLE, M. *Linear Dilation-Erosion Perceptron for Binary Classification*. EnCPos: [s.n.], 2020.

98 DEMBO, R. S.; EISENSTAT, S. C.; STEIHAUG, T. Inexact Newton Methods. *SIAM Journal on Numerical Analysis*, v. 19, n. 2, p. 400–408, abr. 1982. ISSN 0036-1429. Publisher: Society for Industrial and Applied Mathematics.

99 BELLAVIA, S. Inexact Interior-Point Method. *Journal of Optimization Theory and Applications*, v. 96, n. 1, p. 109–121, jan. 1998. ISSN 1573-2878.

APÊNDICE A – Método KKZ e Partições de Voronoi

No algoritmo a seguir, é apresentado o Método KKZ, isto é, os passos para encontrar os centroides. Tais centroides são utilizados para definir a partição de Voronoi. Em seguida, é apresentado um exemplo da utilização do algoritmo.

Algoritmo 6: KKZ

Entrada: $X, \mathbf{x}_0, k$.
Saída: C (centroides)
Inicializar: $B = X, C = \{\mathbf{x}_0\}$
para $i = 1, \ldots, k$ **faça**
  para cada $\mathbf{b}_j \in B$ **faça**
    $d_j = \min_{\mathbf{c} \in C} \{\|\mathbf{b}_j - \mathbf{c}\|\}$
  fim
  $k = \arg\max_j \{d_j\}$
  $C = C \cup \{\mathbf{b}_k\}$
  $B = B \backslash \{\mathbf{b}_k\}$
fim
retorna C

Para o exemplo a seguir, o ponto inicial será o ponto médio de um conjunto de entrada X. No entanto, pode-se escolher o ponto inicial pertencente ao conjunto X.

Exemplo A.1. *Considere conjunto de dados apresentado na Figura A.1.*

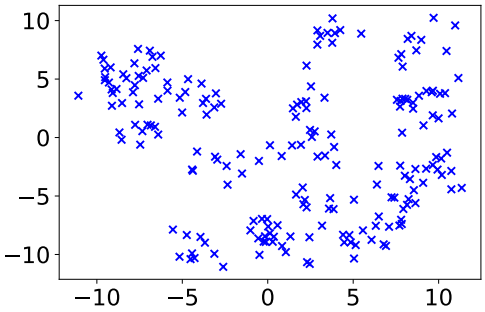


Figura A.1 – Conjunto de dados X.

O método inicia procurando o elemento em X mais distante do ponto inicial. Ao encontrar tal elemento, este passa a pertencer ao conjunto dos centroides C. O próximo passo é encontrar o elemento em X mais distante do conjunto C, este passa a pertencer ao conjunto dos centroides. Esse processo é repetido até encontrar k elementos.

Nas imagens da Figura A.2 é apresentada a construção do conjunto de centroides. Cada imagem representa a i-ésima iteração do algoritmo para um conjunto de controides com 6 *elementos.*

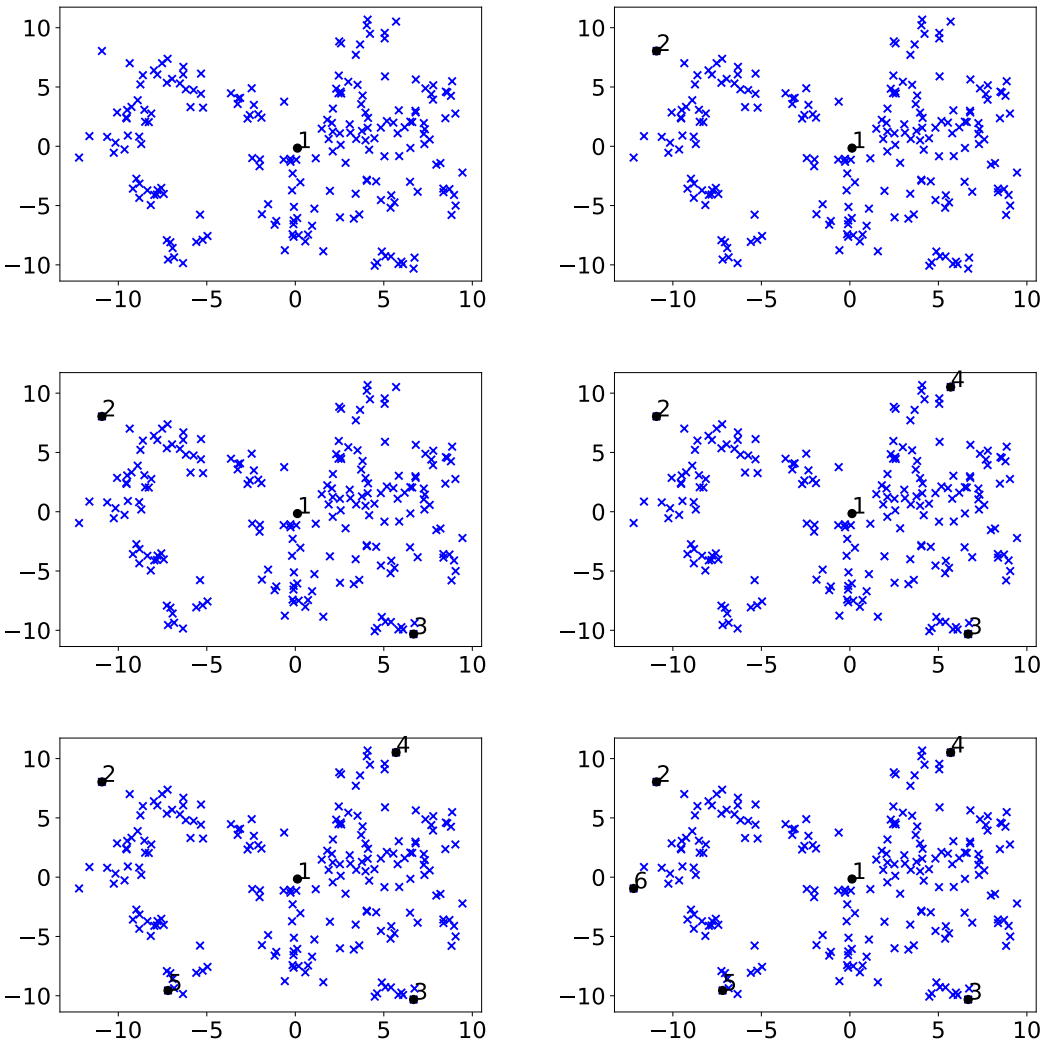


Figura A.2 – Iterações do método KKZ aplicado no conjunto de dados X, com ponto médio inicial e 6 centros.

Após encontrar os centroides, eles podem ser usados para encontrar a partição de Voronoi em relação ao conjunto de centroides C. Na Figura A.3 é apresentada a partição de Voronoi referente a C, encontrados utilizando o método KKZ aplicado no conjunto X.

Cada parte da partição é definida utilizando um centroide de C. Assim, cada uma contém os dados que estão mais próximos de seus centroides.

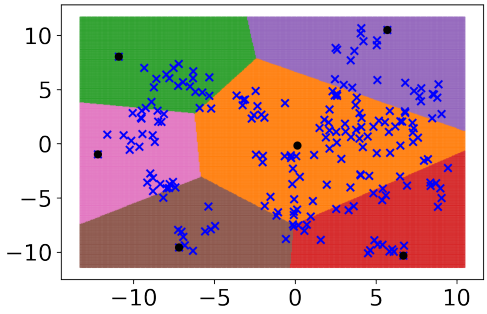


Figura A.3 – Partição de Voronoi do conjunto de centroides C.

APÊNDICE B – Resultados da busca exaustiva dos melhores parâmetros para tarefas de regressão.

B.1 Dataset Curva Modular

Tabela 26 – Média R^2 obtidas pelo modelo MLP usando busca exaustiva para a base de dados Curva Modular.

Rank	#h	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	80	0.954 ± 0.004	0.952 ± 0.008
2	100	0.951 ± 0.005	0.948 ± 0.010
3	50	0.936 ± 0.025	0.928 ± 0.031

Tabela 27 – Média R^2 obtidas pelo modelo SVR usando busca exaustiva para a base de dados Curva Modular.

Rank	C	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	rbf	0.958 ± 0.002	0.945 ± 0.013
2	100	rbf	0.960 ± 0.002	0.945 ± 0.014
3	1	rbf	0.945 ± 0.003	0.931 ± 0.012
4	0.1	rbf	0.744 ± 0.015	0.719 ± 0.016
5	100	linear	-0.028 ± 0.015	-0.120 ± 0.147
6	10	linear	-0.028 ± 0.015	-0.120 ± 0.147
7	1	linear	-0.028 ± 0.015	-0.120 ± 0.147
8	0.1	linear	-0.028 ± 0.015	-0.120 ± 0.147

Tabela 28 – Média R^2 obtidas pelo modelo ℓ-DCA usando busca exaustiva para a base de dados Curva Modular.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	7	2	0.96 ± 0.002	0.952 ± 0.013
2	7	3	0.96 ± 0.002	0.952 ± 0.013
3	5	3	0.96 ± 0.003	0.952 ± 0.013
4	5	2	0.96 ± 0.002	0.952 ± 0.014
5	3	3	0.96 ± 0.003	0.952 ± 0.013
⋮				
20	10	2	0.957 ± 0.005	0.947 ± 0.017
21	10	10	0.901 ± 0.120	0.890 ± 0.117
22	2	10	0.901 ± 0.120	0.889 ± 0.117
23	7	10	0.902 ± 0.120	0.889 ± 0.117
24	3	10	0.902 ± 0.120	0.889 ± 0.117
25	5	10	0.897 ± 0.128	0.885 ± 0.124

Tabela 29 – Média R^2 obtidas pelo modelo ℓ-DEP usando busca exaustiva para a base de dados Curva Modular.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	3	0.943 ± 0.006	0.938 ± 0.008
2	5	3	0.942 ± 0.008	0.936 ± 0.008
3	2	3	0.942 ± 0.008	0.936 ± 0.008
4	7	3	0.942 ± 0.008	0.936 ± 0.008
5	3	3	0.942 ± 0.008	0.936 ± 0.008
⋮				
20	10	5	0.917 ± 0.086	0.897 ± 0.095
21	2	10	0.780 ± 0.325	0.746 ± 0.369
22	3	10	0.776 ± 0.323	0.746 ± 0.369
23	7	10	0.777 ± 0.323	0.746 ± 0.369
24	5	10	0.770 ± 0.320	0.700 ± 0.366
25	10	10	0.778 ± 0.324	0.737 ± 0.364

Tabela 30 – Média R^2 obtidas pelo modelo MAXOUT usando busca exaustiva para a base de dados Curva Modular.

Rank	#h	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	0.959 ± 0.003	0.951 ± 0.014
2	7	0.955 ± 0.006	0.948 ± 0.007
3	2	0.954 ± 0.005	0.946 ± 0.020
4	5	0.953 ± 0.007	0.945 ± 0.017
5	3	0.945 ± 0.025	0.941 ± 0.024

Tabela 31 – Média R^2 obtidas pelo modelo MDN usando busca exaustiva para a base de dados Curva Modular.

Rank	c	c1	c2	r1	r2	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	5	5	15	10	0.953 ± 0.010	0.956 ± 0.010
2	10	5	15	5	15	0.956 ± 0.006	0.956 ± 0.010
3	15	15	5	10	5	0.959 ± 0.003	0.955 ± 0.014
4	10	10	10	5	10	0.957 ± 0.002	0.955 ± 0.013
5	10	15	15	10	15	0.958 ± 0.004	0.955 ± 0.013
⋮							
239	15	5	5	10	5	-2.297 ± 2.082	-1.955 ± 1.802
240	10	10	10	15	5	-1.874 ± 4.417	-2.119 ± 4.331
241	15	10	5	15	5	-2.334 ± 5.051	-2.780 ± 5.135
242	5	15	5	15	5	-2.103 ± 4.968	-3.585 ± 7.972
243	15	10	5	10	5	-4.014 ± 9.193	-3.785 ± 8.687

Tabela 32 – Média R^2 obtidas pelo modelo HMLP-EL usando busca exaustiva para a base de dados Curva Modular.

Rank	C	L	M	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	10	40	0.962 ± 0.002	0.951 ± 0.010
2	10	40	40	0.962 ± 0.002	0.950 ± 0.012
3	10	50	50	0.963 ± 0.002	0.949 ± 0.012
4	10	50	40	0.961 ± 0.003	0.949 ± 0.009
5	10	40	50	0.963 ± 0.002	0.949 ± 0.012
⋮					
32	0.01	10	40	0.970 ± 0.002	0.939 ± 0.017
33	0.01	40	50	0.972 ± 0.002	0.935 ± 0.016
34	0.01	50	40	0.970 ± 0.003	0.934 ± 0.020
35	0.01	10	50	0.971 ± 0.003	0.932 ± 0.017
36	0.01	50	50	0.972 ± 0.003	0.931 ± 0.020

B.2 Dataset PWLinear

Tabela 33 – Média R^2 obtidas pelo modelo MLP usando busca exaustiva para a base de dados PWLinear.

Rank	#h	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	80	0.867 ± 0.006	0.785 ± 0.028
2	100	0.878 ± 0.010	0.784 ± 0.035
3	50	0.815 ± 0.018	0.724 ± 0.041

Tabela 34 – Média R^2 obtidas pelo modelo SVR usando busca exaustiva para a base de dados PWLinear.

Rank	p	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	rbf	0.989 ± 0.003	0.776 ± 0.037
2	100	rbf	1.000 ± 0.000	0.765 ± 0.045
3	1	linear	0.776 ± 0.010	0.740 ± 0.040
4	10	linear	0.775 ± 0.011	0.738 ± 0.039
5	100	linear	0.776 ± 0.012	0.738 ± 0.040
6	0.1	linear	0.761 ± 0.013	0.729 ± 0.040
7	1	rbf	0.801 ± 0.010	0.704 ± 0.030
8	0.1	rbf	0.204 ± 0.008	0.147 ± 0.026

Tabela 35 – Média R^2 obtidas pelo modelo ℓ-DCA usando busca exaustiva para a base de dados PWLinear.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	2	0.921 ± 0.002	0.868 ± 0.009
2	3	2	0.922 ± 0.008	0.845 ± 0.006
3	2	5	0.939 ± 0.002	0.829 ± 0.032
4	7	2	0.938 ± 0.007	0.817 ± 0.031
5	5	2	0.934 ± 0.006	0.811 ± 0.047
6	2	7	0.943 ± 0.006	0.804 ± 0.035
⋮				
20	5	3	0.944 ± 0.002	0.646 ± 0.189
21	2	10	0.950 ± 0.007	0.540 ± 0.459
22	5	10	0.968 ± 0.007	0.488 ± 0.329
23	7	10	0.972 ± 0.006	0.484 ± 0.360
24	10	10	0.975 ± 0.006	0.395 ± 0.490
25	3	10	0.954 ± 0.012	-0.230 ± 1.539

Tabela 36 – Média R^2 obtidas pelo modelo ℓ-DER usando busca exaustiva para a base de dados PWLinear.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	5	0.859 ± 0.037	0.809 ± 0.058
2	2	7	0.865 ± 0.032	0.803 ± 0.039
3	2	3	0.854 ± 0.040	0.803 ± 0.060
4	5	5	0.841 ± 0.029	0.802 ± 0.036
5	5	7	0.845 ± 0.021	0.798 ± 0.039
6	5	3	0.835 ± 0.036	0.797 ± 0.053
⋮				
19	10	2	0.830 ± 0.011	0.759 ± 0.026
20	3	2	0.839 ± 0.022	0.756 ± 0.046
21	10	10	0.878 ± 0.023	0.476 ± 0.551
22	7	10	0.882 ± 0.020	0.006 ± 1.502
23	5	10	0.868 ± 0.013	-0.093 ± 1.771
24	2	10	0.883 ± 0.024	-1.733 ± 5.092
25	3	10	0.880 ± 0.021	-2.048 ± 5.662

Tabela 37 – Média R^2 obtidas pelo modelo MAXOUT usando busca exaustiva para a base de dados PWLinear.

Rank	$r_1 = r_2$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	0.921 ± 0.004	0.870 ± 0.020
2	3	0.929 ± 0.006	0.858 ± 0.015
3	5	0.953 ± 0.003	0.798 ± 0.061
4	10	0.967 ± 0.008	0.779 ± 0.029
5	7	0.966 ± 0.005	0.761 ± 0.034

Tabela 38 – Média R^2 obtidas pelo modelo MDN usando busca exaustiva para a base de dados PWLinear.

Rank	c	c_1	c_2	r_1	r_2	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	5	15	10	15	15	0.659 ± 0.059	0.592 ± 0.049
2	5	10	15	15	15	0.649 ± 0.087	0.582 ± 0.122
3	5	15	15	15	15	0.671 ± 0.090	0.572 ± 0.126
4	5	15	15	5	10	0.604 ± 0.039	0.555 ± 0.078
5	5	5	15	10	15	0.627 ± 0.091	0.549 ± 0.086
⋮							
239	15	15	5	10	5	0.193 ± 0.127	0.150 ± 0.117
240	15	10	5	5	5	0.196 ± 0.072	0.141 ± 0.043
241	10	10	5	5	5	0.190 ± 0.117	0.140 ± 0.150
242	10	5	5	5	5	0.194 ± 0.063	0.139 ± 0.100
243	15	5	5	5	5	0.101 ± 0.070	0.085 ± 0.068

Tabela 39 – Média R^2 obtidas pelo modelo HMLP-EL usando busca exaustiva para a base de dados PWLinear.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	20	20	0.824 ± 0.021	0.734 ± 0.036
2	1	20	10	0.785 ± 0.025	0.725 ± 0.065
3	10	20	20	0.746 ± 0.036	0.701 ± 0.037
4	0.01	20	10	0.793 ± 0.019	0.700 ± 0.039
5	0.01	20	5	0.775 ± 0.030	0.698 ± 0.038
⋮					
20	1	10	20	0.740 ± 0.043	0.573 ± 0.054
21	0.01	5	10	0.696 ± 0.032	0.573 ± 0.160
22	10	10	20	0.646 ± 0.113	0.544 ± 0.169
23	1	5	5	0.527 ± 0.101	0.529 ± 0.084
24	1	5	10	0.587 ± 0.045	0.494 ± 0.055
25	10	5	10	0.542 ± 0.065	0.471 ± 0.068
26	0.01	5	5	0.543 ± 0.109	0.347 ± 0.144
27	10	5	5	0.428 ± 0.138	0.344 ± 0.129

B.3 Dataset PM10

Tabela 40 – Média R^2 obtidas pelo modelo MLP usando busca exaustiva para a base de dados PM10.

Rank	#h	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	100	-0.044 ± 0.163	-0.159 ± 0.158
2	80	-0.149 ± 0.333	-0.343 ± 0.447
3	50	-24.57 ± 47.20	-20.822 ± 39.169

Tabela 41 – Média R^2 obtidas pelo modelo SVR usando busca exaustiva para a base de dados PM10.

Rank	p	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.1	linear	0.171 ± 0.010	0.152 ± 0.024
2	1	linear	0.172 ± 0.010	0.150 ± 0.028
3	100	rbf	0.212 ± 0.013	0.121 ± 0.042
4	10	linear	0.150 ± 0.015	0.105 ± 0.058
5	10	rbf	0.107 ± 0.007	0.047 ± 0.043
6	1	rbf	0.042 ± 0.002	0.006 ± 0.031
7	0.1	rbf	0.014 ± 0.002	-0.008 ± 0.016
8	100	linear	-2.644 ± 1.548	-2.894 ± 1.791

Tabela 42 – Média R^2 obtidas pelo modelo ℓ-DCA usando busca exaustiva para a base de dados PM10.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	7	2	0.321 ± 0.052	0.189 ± 0.088
2	2	5	0.303 ± 0.033	0.188 ± 0.074
3	2	2	0.250 ± 0.052	0.180 ± 0.049
4	7	5	0.368 ± 0.031	0.176 ± 0.058
5	10	5	0.371 ± 0.035	0.174 ± 0.069
⋮				
19	2	10	0.269 ± 0.045	0.085 ± 0.147
20	3	10	0.288 ± 0.049	0.078 ± 0.108
21	10	10	0.346 ± 0.059	0.038 ± 0.096
22	10	7	0.369 ± 0.064	-0.571 ± 1.349
23	2	7	0.299 ± 0.053	-1.027 ± 2.445
24	7	7	0.348 ± 0.050	-3.924 ± 8.029
25	5	7	0.336 ± 0.042	-4.942 ± 9.506

Tabela 43 – Média R^2 obtidas pelo modelo ℓ-DER usando busca exaustiva para a base de dados PM10.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	7	2	0.207 ± 0.028	0.164 ± 0.047
2	10	7	0.214 ± 0.043	0.164 ± 0.047
3	2	7	0.197 ± 0.024	0.161 ± 0.034
4	7	3	0.201 ± 0.034	0.161 ± 0.038
5	7	7	0.206 ± 0.040	0.161 ± 0.044
⋮				
20	5	10	0.200 ± 0.023	0.138 ± 0.058
21	7	5	0.200 ± 0.031	0.137 ± 0.075
22	2	5	0.192 ± 0.025	0.134 ± 0.063
23	5	2	0.203 ± 0.019	0.133 ± 0.057
24	3	5	0.199 ± 0.022	0.123 ± 0.079
25	5	5	0.195 ± 0.028	0.115 ± 0.085

Tabela 44 – Média R^2 obtidas pelo modelo MAXOUT usando busca exaustiva para a base de dados PM10.

Rank	$r_1 = r_2$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	-0.217 ± 0.131	-0.282 ± 0.215
2	5	-0.460 ± 0.492	-0.593 ± 0.863
3	3	-0.482 ± 0.680	-0.627 ± 0.925
4	7	-1.787 ± 2.708	-1.448 ± 1.801
5	10	-3.638 ± 6.768	-3.673 ± 6.730

Tabela 45 – Média R^2 obtidas pelo modelo MDN usando busca exaustiva para a base de dados PM10.

Rank	c	r_1^1	r_2^1	r_1^2	r_2^2	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	5	5	10	15	-1.375 ± 0.460	-1.406 ± 0.502
2	5	5	15	5	15	-1.433 ± 0.860	-1.441 ± 0.891
3	10	5	15	10	15	-1.56 ± 0.909	-1.495 ± 0.792
4	15	10	10	5	15	-1.402 ± 0.289	-1.531 ± 0.541
5	5	10	5	5	10	-1.430 ± 0.666	-1.583 ± 0.957
	⋮						
239	5	5	10	5	5	-297.687 ± 591.474	-288.675 ± 573.731
240	5	5	5	10	5	-327.754 ± 646.985	-427.393 ± 848.283
241	15	5	5	10	5	-526.676 ± 1036.601	-739.28 ± 1464.257
242	5	5	15	5	10	-843.564 ± 1682.131	-822.996 ± 1640.483
243	10	5	5	5	10	-1100.962 ± 2197.79	-889.406 ± 1773.916

Tabela 46 – Média R^2 obtidas pelo modelo HMLP-EL usando busca exaustiva para a base de dados PM10.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	20	20	0.122 ± 0.026	0.047 ± 0.076
2	1	10	20	0.151 ± 0.015	0.033 ± 0.058
3	1	10	10	0.096 ± 0.020	0.030 ± 0.094
4	1	20	20	0.171 ± 0.020	0.021 ± 0.166
5	1	5	10	0.065 ± 0.028	0.013 ± 0.081
	⋮				
22	1	10	5	0.076 ± 0.007	-0.034 ± 0.116
23	10	20	5	0.055 ± 0.019	-0.035 ± 0.080
24	10	5	10	0.032 ± 0.034	-0.035 ± 0.062
25	1	20	10	0.116 ± 0.032	-0.035 ± 0.086
26	0.01	20	10	0.104 ± 0.019	-0.037 ± 0.159
27	0.01	20	20	0.178 ± 0.006	-0.069 ± 0.196

APÊNDICE C – Resultados da busca exaustiva dos melhores parâmetros para tarefas de classificação.

C.1 Dataset de Ripley

Tabela 47 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MLP na base de dados Ripley.

Rank	neurônios	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	100	0.856 ± 0.011	0.855 ± 0.066
2	80	0.846 ± 0.012	0.853 ± 0.069
3	50	0.829 ± 0.010	0.838 ± 0.056

Tabela 48 – Média F_1Score da busca exaustiva e validação cruzada para o modelo SVC na base de dados Ripley.

Rank	C	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	rbf	0.880 ± 0.007	0.878 ± 0.025
2	10	rbf	0.896 ± 0.003	0.878 ± 0.015
3	100	rbf	0.898 ± 0.008	0.873 ± 0.031
4	0.1	rbf	0.872 ± 0.011	0.866 ± 0.037
5	100	linear	0.863 ± 0.018	0.857 ± 0.066
6	1	linear	0.860 ± 0.020	0.855 ± 0.065
7	10	linear	0.854 ± 0.012	0.854 ± 0.053
8	0.1	linear	0.839 ± 0.012	0.844 ± 0.059

Tabela 49 – Média F_1Score da busca exaustiva e validação cruzada para o modelo DEP na base de dados Ripley.

Rank	C	ref	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	max	0.861 ± 0.018	0.849 ± 0.067
2	0.01	mean	0.827 ± 0.049	0.814 ± 0.090
3	0.01	min	0.827 ± 0.050	0.810 ± 0.089
4	0.01	max	0.856 ± 0.017	0.808 ± 0.068
5	0.1	max	0.779 ± 0.171	0.750 ± 0.205
6	1	mean	0.688 ± 0.020	0.688 ± 0.081
7	0.1	mean	0.666 ± 0.014	0.659 ± 0.111
8	0.1	min	0.621 ± 0.070	0.651 ± 0.054
9	1	min	0.526 ± 0.016	0.530 ± 0.079

Tabela 50 – Média F_1Score da busca exaustiva e validação cruzada para o modelo ℓ-DEP na base de dados Ripley.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	5	10	0.896 ± 0.010	0.888 ± 0.021
2	2	10	0.890 ± 0.021	0.883 ± 0.022
3	7	7	0.888 ± 0.018	0.881 ± 0.043
4	7	10	0.899 ± 0.005	0.876 ± 0.026
5	7	3	0.872 ± 0.022	0.872 ± 0.030
	⋮			
20	7	5	0.883 ± 0.015	0.854 ± 0.048
21	2	2	0.867 ± 0.021	0.851 ± 0.070
22	5	5	0.883 ± 0.022	0.846 ± 0.043
23	3	2	0.872 ± 0.013	0.845 ± 0.045
24	10	3	0.884 ± 0.021	0.843 ± 0.071
25	3	5	0.873 ± 0.008	0.842 ± 0.057

Tabela 51 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MAXOUT na base de dados Ripley.

Rank	$r_1 = r_2$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	0.819 ± 0.018	0.828 ± 0.053
2	3	0.777 ± 0.105	0.788 ± 0.076
3	7	0.717 ± 0.173	0.734 ± 0.173
4	5	0.704 ± 0.101	0.650 ± 0.116
5	2	0.618 ± 0.275	0.598 ± 0.269

Tabela 52 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MDN na base de dados Ripley.

Rank	c	r_1^2	r_2^2	r_1^1	r_2^1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	5	15	10	15	0.864 ± 0.014	0.880 ± 0.021
2	15	10	15	5	15	0.875 ± 0.011	0.877 ± 0.035
3	15	15	5	5	5	0.861 ± 0.011	0.875 ± 0.034
4	10	15	15	15	10	0.862 ± 0.013	0.874 ± 0.036
5	5	5	5	10	15	0.852 ± 0.02	0.874 ± 0.05
⋮							
239	15	5	5	15	5	0.862 ± 0.019	0.819 ± 0.061
240	10	5	10	5	5	0.851 ± 0.004	0.817 ± 0.052
241	10	5	10	15	10	0.865 ± 0.019	0.814 ± 0.035
242	5	5	5	5	5	0.834 ± 0.021	0.804 ± 0.080
243	10	5	5	10	10	0.850 ± 0.012	0.796 ± 0.071

Tabela 53 – Média F_1Score da busca exaustiva e validação cruzada para o modelo HMLP-EL na base de dados Ripley.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.01	5	10	0.889 ± 0.006	0.888 ± 0.021
2	0.01	10	10	0.887 ± 0.008	0.888 ± 0.021
3	0.01	20	10	0.887 ± 0.008	0.888 ± 0.021
4	0.01	5	20	0.891 ± 0.006	0.883 ± 0.026
5	0.01	10	20	0.892 ± 0.005	0.883 ± 0.026
⋮					
23	10	20	20	0.847 ± 0.011	0.842 ± 0.067
24	1	5	5	0.820 ± 0.010	0.814 ± 0.069
25	10	10	5	0.723 ± 0.010	0.734 ± 0.065
26	10	20	5	0.732 ± 0.015	0.734 ± 0.065
27	10	5	5	0.688 ± 0.009	0.661 ± 0.074

C.2 Dataset Double Moons

Tabela 54 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MLP na base de dados Double Moons.

Rank	neurônios	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	100	0.959 ± 0.006	0.952 ± 0.028
2	80	0.955 ± 0.009	0.951 ± 0.032
3	50	0.944 ± 0.010	0.942 ± 0.028

Tabela 55 – Média F_1Score da busca exaustiva e validação cruzada para o modelo SVC na base de dados Double Moons.

Rank	p	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	rbf	1.000 ± 0.000	1.000 ± 0.000
2	100	rbf	1.000 ± 0.000	1.000 ± 0.000
3	1	rbf	0.999 ± 0.002	0.997 ± 0.004
4	0.1	rbf	0.988 ± 0.001	0.988 ± 0.010
5	1	linear	0.874 ± 0.009	0.871 ± 0.030
6	10	linear	0.876 ± 0.009	0.871 ± 0.034
7	100	linear	0.876 ± 0.009	0.871 ± 0.034
8	0.1	linear	0.864 ± 0.009	0.863 ± 0.030

Tabela 56 – Média F_1Score da busca exaustiva e validação cruzada para o modelo DEP na base de dados Double Moons.

Rank	C	ref	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.01	min	0.678 ± 0.012	0.676 ± 0.044
2	0.01	mean	0.677 ± 0.012	0.674 ± 0.047
3	0.01	max	0.677 ± 0.014	0.673 ± 0.040
4	1	min	0.671 ± 0.013	0.673 ± 0.049
5	0.1	min	0.671 ± 0.013	0.671 ± 0.051
6	0.1	mean	0.671 ± 0.012	0.671 ± 0.048
7	1	mean	0.672 ± 0.013	0.670 ± 0.053
8	0.1	max	0.591 ± 0.034	0.593 ± 0.049
9	1	max	0.531 ± 0.024	0.527 ± 0.029

Tabela 57 – Média F_1Score da busca exaustiva e validação cruzada para o modelo ℓ-DEP na base de dados Double Moons.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	10	1.000 ± 0.000	1.000 ± 0.000
2	7	10	1.000 ± 0.000	1.000 ± 0.000
3	7	7	1.000 ± 0.000	1.000 ± 0.000
4	5	10	1.000 ± 0.000	1.000 ± 0.000
5	10	7	1.000 ± 0.000	1.000 ± 0.000
⋮				
20	7	2	0.909 ± 0.005	0.904 ± 0.021
21	2	10	0.893 ± 0.008	0.891 ± 0.032
22	2	7	0.888 ± 0.015	0.891 ± 0.029
23	2	3	0.884 ± 0.013	0.883 ± 0.035
24	2	5	0.886 ± 0.016	0.880 ± 0.034
25	2	2	0.876 ± 0.008	0.873 ± 0.034

Tabela 58 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MAXOUT na base de dados Double Moons.

Rank	$r_2 = r_1$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	7	0.974 ± 0.038	0.959 ± 0.072
2	3	0.942 ± 0.047	0.948 ± 0.046
3	10	0.932 ± 0.062	0.938 ± 0.060
4	2	0.855 ± 0.043	0.854 ± 0.047
5	5	0.778 ± 0.390	0.779 ± 0.390

Tabela 59 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MDN na base de dados Double Moons.

Rank	c	r_1^2	r_2^2	r_1^1	r_2^1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	15	15	5	15	0.998 ± 0.002	0.997 ± 0.004
2	10	15	15	5	15	0.995 ± 0.004	0.997 ± 0.004
3	15	15	15	15	10	0.996 ± 0.004	0.997 ± 0.004
4	10	10	15	5	10	0.997 ± 0.004	0.997 ± 0.004
5	15	15	10	10	10	0.998 ± 0.002	0.997 ± 0.004
⋮							
238	5	5	15	5	15	0.984 ± 0.003	0.976 ± 0.023
239	5	5	10	5	5	0.982 ± 0.006	0.976 ± 0.019
240	10	15	10	5	5	0.989 ± 0.009	0.976 ± 0.022
241	15	5	5	15	5	0.983 ± 0.006	0.976 ± 0.021
242	5	5	5	5	5	0.978 ± 0.008	0.973 ± 0.024
243	5	5	15	15	5	0.983 ± 0.011	0.972 ± 0.032

Tabela 60 – Média F_1Score da busca exaustiva e validação cruzada para o modelo HMLP-EL na base de dados Double Moons.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.01	5	10	0.997 ± 0.001	0.997 ± 0.002
2	0.01	5	20	0.997 ± 0.000	0.997 ± 0.002
3	0.01	10	10	0.997 ± 0.001	0.997 ± 0.002
4	0.01	10	20	0.997 ± 0.000	0.997 ± 0.002
5	0.01	20	10	0.997 ± 0.001	0.997 ± 0.002
⋮					
22	10	20	10	0.961 ± 0.004	0.961 ± 0.020
23	10	5	10	0.959 ± 0.003	0.960 ± 0.020
24	10	10	10	0.96 ± 0.003	0.960 ± 0.020
25	10	20	5	0.921 ± 0.004	0.915 ± 0.029
26	10	10	5	0.908 ± 0.005	0.908 ± 0.030
27	10	5	5	0.902 ± 0.006	0.899 ± 0.033

C.3 Dataset Tic-Tac-Toe

Tabela 61 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MLP na base de dados Tic-tac-toe.

Rank	neurônios	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	80	0.88 ± 0.009	0.822 ± 0.014
2	100	0.884 ± 0.006	0.817 ± 0.009
3	50	0.846 ± 0.012	0.802 ± 0.009

Tabela 62 – Média F_1Score da busca exaustiva e validação cruzada para o modelo SVC na base de dados Tic-tac-toe.

Rank	C	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	100	rbf	1.000 ± 0.000	0.994 ± 0.005
2	10	rbf	1.000 ± 0.001	0.993 ± 0.009
3	1	rbf	0.925 ± 0.004	0.884 ± 0.013
4	0.1	linear	0.653 ± 0.001	0.653 ± 0.002
4	0.1	rbf	0.655 ± 0.002	0.653 ± 0.002
4	1	linear	0.653 ± 0.001	0.653 ± 0.002
4	10	linear	0.653 ± 0.001	0.653 ± 0.002
4	100	linear	0.653 ± 0.001	0.653 ± 0.002

Tabela 63 – Média F_1Score da busca exaustiva e validação cruzada para o modelo DEP na base de dados Tic-tac-toe.

Rank	C	ref	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	min	0.671 ± 0.025	0.667 ± 0.023
2	0.1	min	0.615 ± 0.049	0.612 ± 0.073
3	0.1	mean	0.599 ± 0.064	0.587 ± 0.070
4	1	mean	0.593 ± 0.037	0.580 ± 0.044
5	0.01	mean	0.572 ± 0.028	0.555 ± 0.023
6	0.1	min	0.533 ± 0.076	0.550 ± 0.075
7	0.01	min	0.560 ± 0.049	0.541 ± 0.045
8	0.01	min	0.486 ± 0.061	0.467 ± 0.061
9	1	min	0.444 ± 0.105	0.454 ± 0.099

Tabela 64 – Média F_1Score da busca exaustiva e validação cruzada para o modelo ℓ-DEP na base de dados Tic-tac-toe.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	3	7	0.919 ± 0.012	0.883 ± 0.028
2	5	7	0.942 ± 0.018	0.860 ± 0.047
3	2	7	0.909 ± 0.011	0.860 ± 0.025
4	7	7	0.952 ± 0.016	0.849 ± 0.021
5	10	7	0.953 ± 0.021	0.83 ± 0.044
⋮				
19	7	2	0.797 ± 0.010	0.735 ± 0.031
20	5	2	0.777 ± 0.004	0.734 ± 0.018
21	10	3	0.854 ± 0.022	0.731 ± 0.023
22	10	2	0.834 ± 0.010	0.721 ± 0.011
23	3	2	0.760 ± 0.021	0.720 ± 0.025
24	2	2	0.752 ± 0.012	0.714 ± 0.015
25	2	10	0.744 ± 0.03	0.691 ± 0.041

Tabela 65 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MAXOUT na base de dados Tic-tac-toe.

Rank	$r_2 = r_1$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	0.413 ± 0.120	0.418 ± 0.120
2	2	0.347 ± 0.001	0.347 ± 0.002
2	3	0.347 ± 0.001	0.347 ± 0.002
2	5	0.347 ± 0.001	0.347 ± 0.002
2	7	0.347 ± 0.001	0.347 ± 0.002

Tabela 66 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MDN na base de dados Tic-tac-toe.

Rank	c	r_1^2	r_2^2	r_1^1	r_2^1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	10	15	15	15	0.889 ± 0.018	0.875 ± 0.038
2	5	10	15	15	15	0.893 ± 0.018	0.875 ± 0.034
3	5	15	5	10	15	0.883 ± 0.012	0.866 ± 0.012
4	15	15	15	15	15	0.882 ± 0.024	0.865 ± 0.028
5	5	10	15	10	10	0.873 ± 0.024	0.863 ± 0.034
6	10	15	15	15	15	0.878 ± 0.019	0.862 ± 0.025
⋮							
238	10	5	5	5	5	0.767 ± 0.048	0.754 ± 0.048
239	5	5	5	5	5	0.762 ± 0.036	0.753 ± 0.043
240	10	10	5	5	5	0.790 ± 0.036	0.750 ± 0.052
241	15	10	10	5	5	0.755 ± 0.051	0.741 ± 0.044
242	15	5	10	5	5	0.759 ± 0.034	0.739 ± 0.035
243	10	15	5	5	5	0.736 ± 0.057	0.723 ± 0.046

Tabela 67 – Média F_1Score da busca exaustiva e validação cruzada para o modelo HMLP-EL na base de dados Tic-tac-toe.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.01	20	20	0.749 ± 0.002	0.727 ± 0.003
2	0.01	5	20	0.737 ± 0.006	0.720 ± 0.012
3	0.01	10	20	0.735 ± 0.005	0.718 ± 0.009
4	0.01	20	10	0.725 ± 0.003	0.714 ± 0.011
5	1	5	20	0.721 ± 0.004	0.708 ± 0.012
6	1	20	20	0.724 ± 0.004	0.707 ± 0.013
7	1	10	20	0.722 ± 0.003	0.706 ± 0.012
⋮					
21	1	5	10	0.692 ± 0.004	0.681 ± 0.026
22	10	20	5	0.683 ± 0.008	0.680 ± 0.014
23	10	10	5	0.684 ± 0.008	0.680 ± 0.012
24	10	5	5	0.683 ± 0.008	0.680 ± 0.013
25	1	5	5	0.688 ± 0.011	0.680 ± 0.016
26	1	20	5	0.691 ± 0.009	0.680 ± 0.018
27	1	10	5	0.691 ± 0.010	0.677 ± 0.019

C.4 Dataset Breast Cancer Wisconsin

Tabela 68 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MLP na base de dados Breast Cancer Wisconsin.

Rank	neurônios	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	50	0.938 ± 0.009	0.933 ± 0.023
2	80	0.931 ± 0.003	0.930 ± 0.026
3	100	0.932 ± 0.021	0.928 ± 0.024

Tabela 69 – Média F_1Score da busca exaustiva e validação cruzada para o modelo SVC na base de dados Breast Cancer Wisconsin.

Rank	C	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	linear	0.968 ± 0.005	0.953 ± 0.007
2	10	linear	0.971 ± 0.006	0.951 ± 0.014
3	0.1	linear	0.961 ± 0.004	0.946 ± 0.019
3	100	linear	0.966 ± 0.008	0.946 ± 0.017
5	100	rbf	0.939 ± 0.006	0.935 ± 0.027
6	1	rbf	0.918 ± 0.006	0.921 ± 0.031
7	10	rbf	0.925 ± 0.006	0.921 ± 0.028
8	0.1	rbf	0.892 ± 0.008	0.893 ± 0.031

Tabela 70 – Média F_1Score da busca exaustiva e validação cruzada para o modelo DEP na base de dados Breast Cancer Wisconsin.

Rank	C	ref	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	0.1	min	0.927 ± 0.009	0.909 ± 0.017
1	1	min	0.927 ± 0.009	0.909 ± 0.017
3	0.01	min	0.906 ± 0.007	0.907 ± 0.027
3	1	min	0.904 ± 0.007	0.907 ± 0.027
5	0.01	mean	0.905 ± 0.007	0.905 ± 0.029
6	0.1	mean	0.906 ± 0.008	0.905 ± 0.025
6	1	mean	0.906 ± 0.007	0.905 ± 0.025
8	0.01	min	0.916 ± 0.036	0.903 ± 0.021
8	0.1	min	0.906 ± 0.007	0.903 ± 0.026

Tabela 71 – Média F_1Score da busca exaustiva e validação cruzada para o modelo ℓ-DEP na base de dados Breast Cancer Wisconsin.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	5	3	1.000 ± 0.000	0.951 ± 0.009
2	2	2	1.000 ± 0.000	0.949 ± 0.010
2	10	2	1.000 ± 0.000	0.949 ± 0.010
2	7	7	1.000 ± 0.000	0.949 ± 0.010
2	7	5	1.000 ± 0.000	0.949 ± 0.010
2	7	2	1.000 ± 0.000	0.949 ± 0.010
2	5	10	1.000 ± 0.000	0.949 ± 0.010
2	5	7	1.000 ± 0.000	0.949 ± 0.010
2	10	7	1.000 ± 0.000	0.949 ± 0.010
2	5	2	1.000 ± 0.000	0.949 ± 0.010
2	5	5	1.000 ± 0.000	0.949 ± 0.010
2	3	7	1.000 ± 0.000	0.949 ± 0.010
2	3	5	1.000 ± 0.000	0.949 ± 0.010
2	3	3	1.000 ± 0.000	0.949 ± 0.010
2	3	2	1.000 ± 0.000	0.949 ± 0.010
2	2	10	1.000 ± 0.000	0.949 ± 0.010
2	2	7	1.000 ± 0.000	0.949 ± 0.010
2	2	5	1.000 ± 0.000	0.949 ± 0.010
2	2	3	1.000 ± 0.000	0.949 ± 0.010
2	3	10	1.000 ± 0.000	0.949 ± 0.010
21	7	3	1.000 ± 0.000	0.947 ± 0.010
21	10	3	1.000 ± 0.000	0.947 ± 0.010
23	10	5	1.000 ± 0.000	0.946 ± 0.010
24	10	10	1.000 ± 0.000	0.944 ± 0.012
25	7	10	1.000 ± 0.000	0.939 ± 0.005

Tabela 72 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MAXOUT na base de dados Breast Cancer Wisconsin.

Rank	$r_2 = r_1$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	0.627 ± 0.001	0.627 ± 0.004
1	3	0.627 ± 0.001	0.627 ± 0.004
1	5	0.627 ± 0.001	0.627 ± 0.004
1	7	0.627 ± 0.001	0.627 ± 0.004
1	10	0.627 ± 0.001	0.627 ± 0.004

Tabela 73 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MDN na base de dados Breast Cancer Wisconsin.

Rank	c	r_1^2	r_2^2	r_1^1	r_2^1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	10	10	5	10	0.908 ± 0.024	0.923 ± 0.022
2	5	15	5	10	5	0.916 ± 0.011	0.921 ± 0.031
3	5	10	10	5	5	0.905 ± 0.017	0.921 ± 0.030
4	5	10	15	5	10	0.896 ± 0.026	0.907 ± 0.047
5	5	10	10	10	15	0.901 ± 0.044	0.905 ± 0.059
6	15	5	15	10	15	0.889 ± 0.026	0.900 ± 0.026
7	10	15	10	10	15	0.888 ± 0.022	0.900 ± 0.020
8	5	10	15	5	15	0.893 ± 0.026	0.896 ± 0.050
9	5	10	5	5	10	0.902 ± 0.026	0.896 ± 0.038
10	5	5	10	15	15	0.884 ± 0.029	0.895 ± 0.031
⋮							
235	15	15	10	10	5	0.671 ± 0.173	0.671 ± 0.167
236	15	15	15	10	5	0.661 ± 0.202	0.667 ± 0.209
237	10	15	15	15	5	0.668 ± 0.187	0.663 ± 0.188
238	5	15	10	10	5	0.642 ± 0.163	0.654 ± 0.151
239	10	15	15	5	5	0.666 ± 0.207	0.652 ± 0.211
240	15	15	5	15	5	0.656 ± 0.129	0.638 ± 0.134
241	5	5	15	15	5	0.633 ± 0.139	0.622 ± 0.130
242	15	5	15	15	5	0.631 ± 0.124	0.622 ± 0.104
243	5	10	10	15	10	0.623 ± 0.169	0.619 ± 0.179

Tabela 74 – Média F_1Score da busca exaustiva e validação cruzada para o modelo HMLP-EL na base de dados Breast Cancer Wisconsin.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	20	20	0.923 ± 0.008	0.923 ± 0.031
1	10	10	20	0.923 ± 0.009	0.923 ± 0.031
1	10	10	10	0.920 ± 0.007	0.923 ± 0.031
1	10	5	20	0.923 ± 0.009	0.923 ± 0.031
1	10	5	10	0.920 ± 0.007	0.923 ± 0.031
6	0.01	5	20	0.922 ± 0.008	0.921 ± 0.031
6	0.01	10	20	0.922 ± 0.008	0.921 ± 0.031
6	1	20	20	0.923 ± 0.008	0.921 ± 0.031
6	0.01	20	20	0.923 ± 0.008	0.921 ± 0.031
6	1	10	20	0.923 ± 0.008	0.921 ± 0.031
6	1	5	20	0.923 ± 0.008	0.921 ± 0.031
12	1	20	10	0.923 ± 0.007	0.921 ± 0.032
12	1	10	10	0.922 ± 0.008	0.921 ± 0.032
12	1	5	10	0.921 ± 0.009	0.921 ± 0.032
15	10	20	10	0.921 ± 0.008	0.919 ± 0.032
16	0.01	20	10	0.923 ± 0.007	0.919 ± 0.029
16	0.01	10	10	0.923 ± 0.007	0.919 ± 0.029
16	0.01	5	10	0.923 ± 0.007	0.919 ± 0.029
19	1	20	5	0.893 ± 0.007	0.889 ± 0.034
19	0.01	20	5	0.893 ± 0.007	0.889 ± 0.034
19	0.01	10	5	0.893 ± 0.007	0.889 ± 0.034
19	10	20	5	0.893 ± 0.006	0.889 ± 0.034
19	1	10	5	0.893 ± 0.007	0.889 ± 0.034
19	0.01	5	5	0.893 ± 0.007	0.889 ± 0.034
25	1	5	5	0.894 ± 0.008	0.888 ± 0.036
26	10	5	5	0.89 ± 0.006	0.886 ± 0.037
26	10	10	5	0.893 ± 0.007	0.886 ± 0.037

C.5 Dataset Hill-Valley

Tabela 75 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MLP na base de dados Hill-Valley.

Rank	neurônios	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	100	0.612 ± 0.096	0.614 ± 0.082
2	80	0.634 ± 0.061	0.613 ± 0.052
3	50	0.629 ± 0.066	0.611 ± 0.057

Tabela 76 – Média F_1Score da busca exaustiva e validação cruzada para o modelo SVC na base de dados Hill-Valley.

Rank	C	kernel	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	linear	0.993 ± 0.002	0.953 ± 0.011
2	0.1	linear	0.989 ± 0.005	0.951 ± 0.008
3	100	linear	0.993 ± 0.002	0.942 ± 0.013
4	10	linear	0.993 ± 0.002	0.94 ± 0.013
5	100	rbf	0.647 ± 0.006	0.618 ± 0.021
6	10	rbf	0.591 ± 0.005	0.566 ± 0.019
7	1	rbf	0.535 ± 0.004	0.520 ± 0.016
8	0.1	rbf	0.518 ± 0.006	0.501 ± 0.020

Tabela 77 – Média F_1Score da busca exaustiva e validação cruzada para o modelo DEP na base de dados Hill-Valley.

Rank	C	ref	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	1	min	0.538 ± 0.004	0.541 ± 0.018
2	0.1	min	0.540 ± 0.010	0.540 ± 0.015
3	0.1	min	0.541 ± 0.005	0.537 ± 0.016
4	1	min	0.539 ± 0.013	0.535 ± 0.024
5	0.01	min	0.529 ± 0.008	0.534 ± 0.011
6	0.1	mean	0.524 ± 0.005	0.524 ± 0.019
6	1	mean	0.524 ± 0.005	0.524 ± 0.019
8	0.01	mean	0.516 ± 0.005	0.513 ± 0.014
9	0.01	min	0.517 ± 0.005	0.512 ± 0.014

Tabela 78 – Média F_1Score da busca exaustiva e validação cruzada para o modelo ℓ-DEP na base de dados Hill-Valley.

Rank	r_2	r_1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	7	1.000 ± 0.000	0.908 ± 0.014
2	7	10	1.000 ± 0.000	0.901 ± 0.012
3	2	7	1.000 ± 0.000	0.900 ± 0.017
4	10	10	1.000 ± 0.000	0.893 ± 0.025
5	7	2	1.000 ± 0.000	0.893 ± 0.026
6	7	7	1.000 ± 0.000	0.890 ± 0.048
19	7	5	1.000 ± 0.000	0.834 ± 0.031
20	3	3	1.000 ± 0.000	0.829 ± 0.030
21	5	2	1.000 ± 0.000	0.827 ± 0.035
22	5	10	1.000 ± 0.000	0.827 ± 0.027
23	5	7	1.000 ± 0.000	0.826 ± 0.033
24	5	5	1.000 ± 0.000	0.814 ± 0.022
25	5	3	1.000 ± 0.000	0.811 ± 0.012

Tabela 79 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MAXOUT na base de dados Hill-Valley.

Rank	$r_2 = r_1$	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	2	0.500 ± 0.000	0.500 ± 0.001
1	3	0.500 ± 0.000	0.500 ± 0.001
1	5	0.500 ± 0.000	0.500 ± 0.001
1	7	0.500 ± 0.000	0.500 ± 0.001
1	10	0.500 ± 0.000	0.500 ± 0.001

Tabela 80 – Média F_1Score da busca exaustiva e validação cruzada para o modelo MDN na base de dados Hill-Valley.

Rank	c	r_1^2	r_2^2	r_1^1	r_2^1	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	15	5	5	5	10	0.746 ± 0.041	0.734 ± 0.052
2	10	5	5	10	15	0.752 ± 0.045	0.725 ± 0.058
3	5	10	15	5	15	0.719 ± 0.057	0.724 ± 0.045
4	15	5	5	15	15	0.735 ± 0.040	0.723 ± 0.046
5	15	10	5	15	10	0.737 ± 0.069	0.720 ± 0.059
6	10	5	5	10	10	0.735 ± 0.028	0.716 ± 0.018
7	5	15	15	10	15	0.730 ± 0.056	0.714 ± 0.046
8	5	10	10	15	15	0.731 ± 0.051	0.714 ± 0.045
9	10	10	15	10	15	0.748 ± 0.051	0.714 ± 0.058
10	10	5	10	10	10	0.722 ± 0.031	0.712 ± 0.027
⋮							
233	10	15	10	5	5	0.603 ± 0.065	0.587 ± 0.056
234	5	15	15	15	10	0.618 ± 0.060	0.585 ± 0.042
235	15	15	5	10	5	0.589 ± 0.075	0.583 ± 0.078
236	10	15	5	5	5	0.591 ± 0.066	0.582 ± 0.064
237	10	5	15	10	15	0.613 ± 0.062	0.582 ± 0.045
238	5	15	10	5	5	0.578 ± 0.090	0.580 ± 0.092
239	15	10	15	10	10	0.595 ± 0.060	0.577 ± 0.066
240	15	5	15	5	5	0.604 ± 0.070	0.576 ± 0.063
241	10	10	5	10	5	0.593 ± 0.042	0.568 ± 0.030
242	15	10	10	5	5	0.580 ± 0.052	0.564 ± 0.065
243	10	10	15	5	5	0.578 ± 0.052	0.562 ± 0.047

Tabela 81 – Média F_1Score da busca exaustiva e validação cruzada para o modelo HMLP-EL na base de dados Hill-Valley.

Rank	C	l	m	$\mu_{Tr} \pm \sigma_{Tr}$	$\mu_{Te} \pm \sigma_{Te}$
1	10	5	20	0.587 ± 0.031	0.571 ± 0.027
2	10	20	20	0.588 ± 0.033	0.570 ± 0.023
2	10	10	20	0.587 ± 0.032	0.570 ± 0.025
4	0.01	5	20	0.589 ± 0.034	0.569 ± 0.024
4	0.01	10	20	0.589 ± 0.034	0.569 ± 0.024
4	1	20	20	0.589 ± 0.034	0.569 ± 0.023
4	0.01	20	20	0.589 ± 0.034	0.569 ± 0.024
4	1	10	20	0.589 ± 0.033	0.569 ± 0.023
4	1	5	20	0.589 ± 0.033	0.569 ± 0.023
10	10	5	10	0.566 ± 0.021	0.557 ± 0.033
11	10	10	10	0.566 ± 0.020	0.554 ± 0.035
12	10	20	10	0.566 ± 0.020	0.553 ± 0.034
12	1	10	10	0.565 ± 0.020	0.553 ± 0.034
12	1	5	10	0.565 ± 0.020	0.553 ± 0.034
12	0.01	20	10	0.565 ± 0.020	0.553 ± 0.034
12	0.01	10	10	0.565 ± 0.020	0.553 ± 0.034
12	0.01	5	10	0.565 ± 0.020	0.553 ± 0.034
18	1	20	10	0.565 ± 0.020	0.552 ± 0.034
19	10	5	5	0.522 ± 0.007	0.517 ± 0.026
20	10	10	5	0.521 ± 0.007	0.516 ± 0.026
21	10	20	5	0.522 ± 0.007	0.512 ± 0.023
22	1	5	5	0.522 ± 0.006	0.512 ± 0.024
22	1	20	5	0.522 ± 0.006	0.512 ± 0.024
22	0.01	20	5	0.523 ± 0.006	0.512 ± 0.024
22	0.01	10	5	0.523 ± 0.006	0.512 ± 0.024
22	1	10	5	0.522 ± 0.006	0.512 ± 0.024
22	0.01	5	5	0.523 ± 0.006	0.512 ± 0.024