UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós Graduação em Engenharia Elétrica

Vítor Gabriel Reis Caitité

# Classificadores de Margem Larga Baseados em Redes Neurais de Camada Oculta Única

Belo Horizonte

2023

Vítor Gabriel Reis Caitité

# Classificadores de Margem Larga Baseados em Redes Neurais de Camada Oculta Única

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Antônio de Pádua Braga
Coorientador: Raul Fonseca Neto

Belo Horizonte
2023

Caitité, Vítor Gabriel Reis.
C137c Classificadores de margem larga baseados em redes neurais de camada oculta única [recurso eletrônico] / Vítor Gabriel Reis Caitité. - 2023.
1 recurso online (78 f. : il., color.) : pdf.

Orientador: Antônio de Pádua Braga.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 76-78.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Redes neurais (Computação) - Teses. 3. Aprendizado do computador - Teses. I. Braga, Antônio de Pádua. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
**ESCOLA DE ENGENHARIA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

## FOLHA DE APROVAÇÃO

## "CLASSIFICADORES DE MARGEM LARGA BASEADOS EM REDES NEURAIS DE CAMADA OCULTA ÚNICA"

**VÍTOR GABRIEL REIS CAITITÉ**

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica. Aprovada em 28 de setembro de 2023. Por:

Prof. Dr. Antônio de Pádua Braga
(UFMG) - Orientador

Prof. Dr. Raul Fonseca Neto
(UFJF) - Co-orientador

Prof. Dr. Cristiano Leite de Castro
(UFMG)

Prof. Dr. Vitor Angelo Maria Ferreira Torres
(UFMG)

Documento assinado eletronicamente por **Antonio de Padua Braga**, **Membro**, em 29/09/2023, às 07:46, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Raul Fonseca Neto**, **Usuário Externo**, em 29/09/2023, às 11:55, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Vitor Angelo Maria Ferreira Torres**, **Professor do Magistério Superior**, em 29/09/2023, às 13:07, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Cristiano Leite de Castro**, **Professor do Magistério Superior**, em 29/09/2023, às 15:35, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2659853** e o código CRC **6F9A9DC6**.

*Dedico este trabalho aos meus pais, Carlos Caitité e Solange Reis, que sempre me incentivaram a buscar voos mais longos.*

# Agradecimentos

Esse trabalho representa a conclusão da jornada de aprendizado mais desafiadora da minha vida até o presente momento. Primeiramente gostaria de agradecer a Deus, que na árdua trajetória até aqui nunca me desamparou.

Em segundo lugar, gostaria de agradecer à minha família, que sempre me deu todo o apoio e as condições necessárias para realizar meus sonhos. Em especial Solange, Caitité, Vinícius e Nelita muito obrigado por tudo e por tanto.

Agradeço ainda à todos os professores que durante essa jornada compartilharam parte do seu conhecimento com tanta maestria. Em especial gostaria de agradecer aos professores Antônio de Pádua Braga e Raul Fonseca Neto que me orientaram durante este trabalho.

Sou grato também a toda equipe do Laboratório de Inteligência Computacional (LITC) que me acompanhou ativamente durante a elaboração deste projeto. Foi um privilégio conviver com vocês e por isso deixo aqui o meu muito obrigado.

Por fim, serei eternamente grato a todos os meus amigos que estiveram comigo durante a elaboração deste trabalho. Fica aqui um obrigado especial ao Diego Maradona, Amanda, Gabriela e Isabela por não me deixarem desistir em momentos de incertezas e ainda me auxiliarem ativamente na revisão deste trabalho.

*Sometimes it is the people who*
*no one imagines anything of,*
*who do the things that no one can imagine.*
Alan Turing

# Resumo

Esta dissertação explorou a relevância dos classificadores de margem larga no campo do aprendizado de máquina. Buscou-se observar algumas características relevantes desses classificadores, como sua capacidade de generalização, robustez a dados ruidosos, interpretabilidade e resistência a *overfitting*. Foram propostos três métodos, baseados em redes neurais de uma única camada oculta, que buscam obter uma margem larga: RP-IMA primal, IM-RBFNN e RP-IMA dual. Esses algoritmos se baseiam no princípio de determinar os pesos da camada escondida da rede de forma não supervisionada, enquanto na camada de saída é empregado um algoritmo de margem incremental. Todos os modelos foram testados em bases sintéticas e bases de *benchmark* e em todos os casos a metodologia de testes utilizada foi a validação cruzada com 10 dobras. Os resultados de medição de margem rígida demonstraram que esses modelos foram capazes de obter margens significativamente maiores em comparação com outros algoritmos, como ELM, RBFNN e ELM dual, respectivamente. Além disso, análises de acurácia dos modelos mostraram uma correlação positiva entre a obtenção de uma margem larga no espaço de características e o desempenho de classificação para os modelos RP-IMA primal e IM-RBFNN. Por fim, uma estratégia de poda de neurônios foi proposta para esses métodos. Os experimentos demonstraram que a poda de neurônios é capaz de reduzir significativamente a arquitetura da rede neural, enquanto mantém um desempenho comparável. Essa abordagem permite obter modelos mais compactos e eficientes sem sacrificar a performance na classificação.

Palavras-chave: classificadores de margem larga; redes neurais; poda de neurônios; classificação de dados tabulares. problemas de classificação binária.

# Abstract

This work explored the relevance of large-margin classifiers in the machine learning field. It was observed some relevant characteristics of these classifiers, such as their generalization capacity, robustness to noisy data, interpretability, and resistance to *overfitting*. Three methods were proposed, based on neural networks with a single hidden layer, which pursues to obtain a large margin: primal RP-IMA, IM-RBFNN, and dual RP-IMA. These algorithms are based on the principle of determining the hidden layer weights of the network in an unsupervised approach and the output layer weights using an incremental margin algorithm. All models were tested on synthetic and *benchmark* datasets, and the methodology used was a 10-fold-cross-validation. The "hard" margin measurement results demonstrated that these models were able to obtain significantly higher margins compared to other algorithms such as ELM, RBFNN, and Dual ELM, respectively. Furthermore, analyses of model accuracy showed a positive correlation between obtaining a large margin in the feature space and classification performance for the primal RP-IMA and IM-RBFNN models. Finally, a neuron pruning strategy was proposed for these methods. The experiments demonstrated that the pruning scheme can significantly reduce neural network architecture while maintaining comparable performance. This approach allows them to obtain more compact and efficient models without reducing classification performance.

*Keywords: large margin classifiers; neural networks; neuron pruning; tabular data classification; binary classification problems.*

# Lista de Ilustrações

# Lista de Tabelas

# Lista de Algoritmos

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| SVM | Support Vector Machine |
| IMA | Incremental Margin Algorithm |
| FMP | Fixed Margin Percetron |
| SLFN | Single-Hidden Layer Feedforward Network |
| ELM | Extreme Learning Machine |
| RBFNN | Radial Basis Function Neural Network |
| kNN | k-Nearest Neighbors |
| RP-IMA | Random Projection Incremental Margin Algorithm |
| IM-RBFNN | Incremental Margin Radial Basis Function Neural Network |

# Sumário

# Capítulo 1

# Introdução

## 1.1 Contextualização

Margem de separação no contexto de classificadores refere-se a valores que determinam o quanto exemplos de duas ou mais classes encontram-se distantes das fronteiras de decisão geradas por um classificador. Normalmente esses valores refletem a distância perpendicular entre a fronteira de separação e as amostras mais próximas de cada classe. Um algoritmo de classificação de margem larga tem como objetivo não apenas encontrar um hiperplano que separe amostras de diferentes classes, mas também fazê-lo com uma grande margem de separação.

Os algoritmos de classificação de margem larga são importantes pois auxiliam na busca por fronteiras de decisão mais precisas (que generalizem bem para dados não vistos) e mitigam o risco de um ajuste excessivo aos dados de treinamento (*overfitting*). Essas características tornam esses algoritmos úteis em muitos cenários de aprendizado de máquina, como em situações onde os dados de treinamento são limitados ou ruidosos.

A Máquina de Vetores Suporte (*Support Vector Machine* - SVM) [Boser et al., 1992] é possivelmente o método mais conhecido quando se trata de classificadores de margem larga. Seu sucesso provavelmente se deve à sua formulação clara de um problema de programação quadrática e ao princípio de adotar um mapeamento implícito de kernel para o espaço de características.

Considerando problemas linearmente separáveis, alguns algoritmos online eficientes que aproximam o hiperplano de margem máxima [Gentile, 2000, Leite e Neto, 2008] também foram propostos na literatura. O objetivo destes algoritmos é construir classificadores com custo computacional menor que a SVM e que fossem capazes de aproximar a solução de margem máxima. Leite e Neto [2008] apresentaram o Algoritmo de Margem Incremental (*Incremental Margin Algorithm* - IMA), que é capaz de obter uma solução de margem larga resolvendo sucessivamente o problema de classificação com tamanhos de margem

crescentes, usando o Perceptron de Margem Fixa (*Fixed Margin Perceptron* - FMP). O FMP é uma extensão do Perceptron de Rosenblatt, que visa encontrar a solução de um problema de aprendizagem linearmente separável dada uma margem fixa.

Com o intuito de gerar classificadores de margem larga baseados em redes neurais de camada escondida única (*Single-Hidden Layer Feedforward Network* - SLFN) este trabalho propõe a utilização do IMA para o treinamento da camada de saída desse tipo de rede. Assim, tem-se uma rede cuja a camada escondida realiza um mapeamento explícito com objetivo de processar os dados de entrada e extrair características relevantes e uma camada de saída que visa gerar um hiperplano classificador com uma margem larga de separação.

Em uma rede SLFN, a camada escondida é composta por um conjunto de neurônios que recebem os valores de entrada ponderados pelos seus respectivos pesos. Cada neurônio na camada escondida recebe os valores de entrada ponderados e os passa por uma função de ativação não linear. A finalidade disso é introduzir não linearidades na rede e permitir que a SLFN seja capaz de aprender representações mais complexas e extrair características importantes dos dados de entrada, capturando padrões e relações entre as variáveis.

Essas características extraídas são então transmitidas à camada de saída, onde são usadas para gerar a saída final da rede neural. Ao se utilizar o IMA para o treinamento dessa camada da rede, espera-se obter um classificador de margem larga que possua uma boa capacidade de generalização, produzindo modelos que generalizem bem para dados novos e sejam menos suscetíveis a influência de *outliers*.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

O objetivo geral desse trabalho é realizar um estudo sobre a maximização de margem em redes neurais *feedforward* de uma camada oculta. Objetiva-se implementar algoritmos de treinamento para essas redes que visem obter uma margem larga de separação na camada de saída.

### 1.2.2 Objetivos Específicos

De maneira detalhada, os objetivos específicos são:

1. Desenvolver um algoritmo de treinamento para uma rede SLFN cujos pesos dos neurônios da camada escondida são escolhidos aleatoriamente e que na camada de saída busque maximizar a margem de decisão no espaço de características.

2. Implementar uma versão em variáveis duais do método proposto no item 1.

3. Desenvolver um algoritmo de treinamento para uma rede SLFN cuja camada oculta é composta de neurônios de funções radiais, e que na camada de saída busque maximizar a margem de decisão no espaço de características.

4. Propor a utilização de um método de poda de neurônios para os algoritmos desenvolvidos nos itens 1 e 3.

## 1.3 Organização do Texto

O restante deste trabalho está organizado da seguinte forma: No capítulo 2 é realizada uma fundamentação teórica dos algoritmos de aprendizado de máquina nos quais os métodos propostos foram embasados. O capítulo 3 descreve as abordagens propostas para a obtenção de classificadores de margem larga baseados em redes neurais *feedforward* de uma única camada oculta. Já o capítulo 4 apresenta os testes realizados e os respectivos resultados obtidos. Por fim, no capítulo 5 são expressas as conclusões e sugestões de trabalhos futuros.

# Capítulo 2

# Referencial Teórico

O presente capítulo busca proporcionar uma contextualização do estudo em questão por meio da exploração de conceitos fundamentais e relevantes no campo. Além disso, são apresentados os métodos presentes na literatura que estabeleceram o alicerce necessário para a construção deste trabalho.

## 2.1 Classificadores de Margem Larga

Desde o desenvolvimento da SVM, uma variedade de técnicas para gerar classificadores de margem larga foram sugeridas. Considerando problemas linearmente separáveis, Freund e Schapire [1999] e Leite e Neto [2008] formularam propostas para classificadores, baseados no algoritmo do perceptron, capazes de encontrar superfícies de separação cuja a margem se aproxime da máxima.

O trabalho de Weinberger e Saul [2009] apresentou uma nova contribuição para o campo, baseada no algoritmo k-Vizinhos Mais Próximos (*k-Nearest Neighbors* - kNN) [Cover e Hart, 1967]. Sua proposta foi aprender uma métrica de distância Mahalanobis para o kNN com o objetivo de que os k vizinhos mais próximos estejam sempre na mesma classe, enquanto amostras de diferentes classes sejam separadas por uma margem ampla [Weinberger e Saul, 2009]. Ainda com base no kNN, Domeniconi et al. [2005] propuseram aplicar a fronteira de decisão das SVMs para induzir uma métrica de distância localmente adaptativa para o kNN. Os autores mostraram que esse método é capaz de aumentar a margem no espaço ponderado onde ocorre a classificação.

Outra abordagem importante usada para construir classificadores de margem larga é estudar o problema a partir de uma formulação geométrica. O estudo realizado por Bennett e Bredensteiner [2000] introduziu uma interpretação geométrica da SVM no contexto de classificação. Uma das contribuições mais significativas deste trabalho reside na capacidade de proporcionar uma compreensão intuitiva e visual dos conceitos fundamentais da SVM, utilizando uma explicação simplificada baseada em fechos convexos.

Além disso, Torres et al. [2015] apresentaram um classificador baseado apenas na estrutura dos dados representada pelo Grafo de Gabriel [Gabriel e Sokal, 1969] para minimizar o erro do conjunto de treinamento e maximizar a margem de separação entre as classes. Nesse método, a superfície de separação resulta de um modelo de misturas de hiperplanos cujos parâmetros são obtidos do próprio grafo, mais especificamente dos padrões localizados próximos à margem de decisão. Com base no Grafo de Gabriel, existem também outras abordagens que implementam classificadores de margem larga. Uma estratégia possível é, por exemplo, a aplicação do kNN com k=1, porém considerando apenas o subconjunto de vetores geométricos de margem (obtidos do grafo associado às informações de rótulo) como conjunto de treinamento [Arias-Garcia et al., 2020].

Recentemente, o tópico de margem larga também tem sido estudado no contexto do aprendizado profundo. Nesse cenário, uma abordagem concentra-se apenas na margem da camada de saída. Por exemplo, Zhang et al. [2015] propuseram o uso de uma SVM na camada final da rede, em vez da ativação *softmax* tradicional. Em outra perspectiva, Elsayed et al. [2018] relataram como abordagens clássicas de maximização de margem não são apropriadas para modelos profundos, apenas para os rasos. Diferentemente da primeira abordagem,l que se limita a aplicar a maximização de margem apenas na camada de saída, neste trabalho os autores propuseram uma função de perda na qual uma margem ampla pode ser aplicada em qualquer conjunto de camadas [Elsayed et al., 2018].

## 2.2 Perceptron de Margem Fixa

O perceptron de margem fixa (*Fixed Margin Perceptron* - FMP) se trata de uma extensão do Perceptron de Rosenblatt [1958], que visa encontrar a solução de um problema de aprendizagem linearmente separável dada uma margem fixa.

No trabalho de Leite e Neto [2008] foi apresentada a definição para o FMP considerando problemas de classificação binária em que as classes são rotuladas como 1 ou $-1$. De acordo com essa formulação, dada uma amostra de treinamento $(\mathbf{x}_i, y_i)$ e uma margem fixa $\gamma_f$, um erro ocorre se $y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b) < \gamma_f \|\boldsymbol{\beta}\|$, sendo $\boldsymbol{\beta}$ o vetor normal ao hiperplano de separação e $b$ o termo *bias*. Assim, a função de erro foi definida por Leite e Neto [2008] como mostrado na Equação 2.1:

$$J(\boldsymbol{\beta}) = \sum_{(\mathbf{x}_i, y_i) \in M} (\gamma_f \|\boldsymbol{\beta}\| - y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + b)), \tag{2.1}$$

onde M é um subgrupo dos dados de treinamento Z, tal que:

$$M = (\mathbf{x}_i, y_i) \in Z | y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + b) < \gamma_f \|\boldsymbol{\beta}\|. \tag{2.2}$$

Baseado então na regra de atualização de pesos do perceptron e adicionado um termo relacionado a margem fixa, estabeleceu-se a regra de atualização mostrada na Equação 2.3:

$$\begin{aligned} &\boldsymbol{\beta}^{t+1} \leftarrow \boldsymbol{\beta}^t \psi_t + \eta y_i \mathbf{x}_i \text{ tal que :} \\ &\psi_t = \begin{cases} 1 - \frac{\eta\gamma_f}{\|\boldsymbol{\beta}^t\|} & \text{se } \|\boldsymbol{\beta}^t\| \neq 0 \\ 1 & \text{caso contrário,} \end{cases} \\ &b^{t+1} \leftarrow b^t + \eta y_i, \end{aligned} \tag{2.3}$$

tal que $\eta > 0$ é a taxa de aprendizado.

Uma limitação do algoritmo FMP, definido em variáveis primais, é ser aplicável apenas a problemas linearmente separáveis. Para lidar com isso, o trabalho de Villela et al. [2016] propôs adaptar e usar a solução de margem flexível introduzida em Schölkopf et al. [2002]. Assim, uma variável de folga, representada por $\alpha_i\lambda$, é utilizada em cada verificação de viabilidade [Villela et al., 2016]:

$$y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + b) \geq \gamma_f \|\boldsymbol{\beta}\| - \alpha_i\lambda, \tag{2.4}$$

tal que $\lambda$ é um hiper-parâmetro e $\alpha_i$ é o $i$-ésimo componente do vetor de multiplicadores $\boldsymbol{\alpha}$ associado à amostra $(\mathbf{x}_i, y_i)$. Assim como $\boldsymbol{\beta}$, $\alpha_i$ é atualizado toda vez que ocorre um erro:

$$\begin{aligned} &\boldsymbol{\alpha}^{t+1} \leftarrow \boldsymbol{\alpha}^t(1 - \frac{\eta\gamma_f}{\|\boldsymbol{\beta}\|}), \\ &\alpha_i^{t+1} \leftarrow \alpha_i^{t+1} + \eta \cdot 1. \end{aligned} \tag{2.5}$$

No Algoritmo 1 encontra-se o pseudocódigo do FMP considerando a estratégia de flexibilidade de margem apresentada acima.

## 2.3 Algoritmo de Margem Incremental

A utilização do algoritmo FMP sozinho não possui grande relevância prática. Isso porque é difícil saber a priori bons valores de margem fixa para se utilizar em cada problema. Assim, Leite e Neto [2008] propuseram o Algoritmo de Margem Incremental (*Incremental Margin Algorithm* - IMA). A proposta deste algoritmo é executar o FMP sucessivamente, aumentando o tamanho da margem em cada iteração. O algoritmo permanece neste *loop* até que o FMP falhe em convergir em um número máximo de iterações ou atinja um número desejado de execuções do algoritmo FMP.

A formulação para o problema de maximização de margem, proposta por Leite e Neto [2008], é desenvolvida a partir da observação de que, uma vez obtida a margem máxima, pontos ou vetores de suporte de classes opostas estão à mesma distância do hiperplano separador [Villela et al., 2013].

**Algoritmo 1** Perceptron de Margem Fixa (FMP)

**Hiperparâmetros**: passo de atualização utilizado pelo FMP ($\eta$), termo constante da variável de folga ($\lambda$), número de atualizações máximo (T_MAX) e margem fixa ($\gamma_f$)
**Entrada:** *dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$, pesos iniciais ($\boldsymbol{\beta}_{init}$) e termo bias inicial ($b_{init}$).
**Saída:** pesos encontrados.
$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_{init}$;
$b \leftarrow b_{init}$;
$t \leftarrow 0$;
$last\_t \leftarrow -1$;
$\alpha[1, ..., N] \leftarrow 0$;
**while** t < T_MAX and last_t ≠ t **do**
    $last\_t \leftarrow t$;
    **for** i = 1, ..., N **do**
        **if** $y_i(\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle + b) < \gamma_f \|\boldsymbol{\beta}\| - \alpha_i \lambda$ **then**
            **if** $\|\boldsymbol{\beta}\| \neq 0$ **then**
                $\psi \leftarrow (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|)$;
            **else**
                $\psi \leftarrow 1$;
            **end if**
            $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}\psi + \eta y_i \mathbf{x}_i$;
            $b \leftarrow b + \eta y_i$;
            $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} \cdot \psi$;
            $\alpha_i \leftarrow \alpha_i + \eta \cdot 1$;
            $t \leftarrow t + 1$ ;
        **end if**
    **end for**
**end while**
**return:** pesos aprendidos $\boldsymbol{\beta}$.

A partir desta observação foi desenvolvida a regra para atualização da margem mostrada na Equação 2.6:

$$\begin{aligned} \gamma_f &= max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1+\delta)\gamma_f), \\ \gamma^+(\boldsymbol{\beta}) &= min\{y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|, \forall y_i = +1\}, \\ \gamma^-(\boldsymbol{\beta}) &= min\{y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|, \forall y_i = -1\}, \end{aligned} \tag{2.6}$$

sendo que $\delta \in (0, 1)$ é um incremento de margem mínimo definido como um hiperparâmetro. Com esta regra de atualização, é possível garantir que o valor da margem fixa passado ao FMP seja sempre maior que o anterior.

Um pseudocódigo do IMA é apresentado no Algoritmo 2.

**Algoritmo 2** Algoritmo de Margem Incremental

1: **Hiperparâmetros**: passo de atualização utilizado pelo FMP ($\eta$), incremento mínimo da margem a cada iteração do IMA ($\delta$), termo constante da variável de folga ($\lambda$), número de atualizações máximo (T_MAX).
2: **Entrada:** *dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$
3: **Saída:** pesos treinados.
4: $\boldsymbol{\beta}[1, ..., \tilde{\mathrm{N}}] \leftarrow 0$;
5: $\gamma_f \leftarrow 0$;
6: **while** FMP convergir em no máximo T_MAX atualizações **do**
7: $\quad \boldsymbol{\beta} \leftarrow \mathrm{FMP}((\mathbf{X}, \mathbf{y}), \boldsymbol{\beta}, \gamma_f, \eta, \mathrm{T\_MAX})$
8: $\quad \gamma_f = max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1 + \delta)\gamma_f)$
9: **end while**
10: **return:** pesos aprendidos $\boldsymbol{\beta}$.

## 2.4 IMA e FMP com Norma Arbitrária

O método FMP apresentado anteriormente se trata de um algoritmo para resolução de problemas de classificação binária para qualquer margem $L_2$ fixa. O trabalho de Villela et al. [2016] apresenta uma generalização do FMP para normas arbitrárias. Este algoritmo, chamado *Fixed p-Margin Perceptron* ($\mathrm{FMP}_p$), resolve problemas de classificação binária para qualquer margem $L_p$ fixa com $p \geq 1$, incluindo $p = \infty$. O $\mathrm{FMP}_p$ também pode ser combinado com o algoritmo de margem incremental, que neste caso é chamado de *Incremental p-Margin Algorithm* ($\mathrm{IMA}_p$).

A partir deste ponto, ao se referir aos métodos FMP e IMA considerando a norma $L_2$ (apresentados nas seções 2.2 e 2.3, respectivamente) serão utilizados os termos $\mathrm{FMP}_2$ e $\mathrm{IMA}_2$.

Como $\mathrm{FMP}_2$, o $\mathrm{FMP}_p$ atualiza $\boldsymbol{\beta}$ e $b$ toda vez que um erro ocorre, as diferenças são as equações para verificação de erro e atualização dos pesos. Essas equações são generalizadas para resolver o problema considerando uma margem $p$ fixa. Assim, um erro ocorre se $y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b) < \gamma_f \|\boldsymbol{\beta}\|_q$ onde $\|\cdot\|_q$ é a norma conjugada, e $p$ e $q$ satisfazem a condição: $1/p + 1/q = 1$. Já a regra de correção, para $1 < q < \infty$, foi definida por Villela et al. [2016] como:

$$\begin{aligned} \boldsymbol{\beta}^{t+1} &\leftarrow \boldsymbol{\beta}^t - \eta(\gamma_f \|\boldsymbol{\beta}^t\|_q^{1-q} |\boldsymbol{\beta}^t|^{q-1} \mathrm{sign}(\boldsymbol{\beta}^t) - y_i \mathbf{x}_i), \\ b^{t+1} &\leftarrow b^t + \eta y_i. \end{aligned} \tag{2.7}$$

Considerando, por exemplo, um classificador capaz de gerar um hiperplano separador com margem $L_\infty$, tem-se $p = \infty$ (o que torna $q = 1$) e a regra de atualização de $\beta$ é simplificada para:

$$\boldsymbol{\beta}^{t+1} \leftarrow \boldsymbol{\beta}^t - \eta(\gamma_f \mathrm{sign}(\boldsymbol{\beta}^t) - y_i \mathbf{x}_i). \tag{2.8}$$

Da mesma forma, quando p=1 deseja-se encontrar um hiperplano com margem $L_1$, o que implica na minimização da norma $L_\infty$ do vetor $\beta$. Para esse caso, onde $q = \infty$, o trabalho de Villela et al. [2016] desenvolveu a regra de atualização definida pela Equação 2.9:

$$\beta_i^{t+1} = \begin{cases} \beta_j^t - \eta(\gamma_f \cdot \text{sign}(\beta_j^t)/n^t - y_i x_{ij}) & \text{se } |\beta_j| = \|\boldsymbol{\beta}\|_\infty \\ \beta_j^t + \eta(y_i x_{ij}) & \text{se } |\beta_j| < \|\boldsymbol{\beta}\|_\infty, \end{cases} \tag{2.9}$$

sendo que $n^t$ refere-s a cardinalidade do conjunto $N^t = \{k \in \{1, ..., d\} : |\beta_j| = \|\boldsymbol{\beta}\|_\infty\}$.

Por fim, a última diferença entre $\text{IMA}_p$ e $\text{IMA}_2$ é que a regra de atualização de margem usa $\|\boldsymbol{\beta}\|_q$ em vez de $\|\boldsymbol{\beta}\|_2$:

$$\begin{aligned} &\gamma_f = max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1+\delta)\gamma_f), \text{ sendo :} \\ &\gamma^+(\boldsymbol{\beta}) = min\{y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = +1\}, \\ &\gamma^-(\boldsymbol{\beta}) = min\{y_i(\mathbf{x}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = -1\}. \end{aligned} \tag{2.10}$$

## 2.5 Perceptron de Margem Fixa Dual

Em otimização, a formulação de um problema pode ser dada em termos de suas variáveis primais ou em termos de suas variáveis duais. A formulação primal é a formulação original do problema, enquanto a dual é uma reformulação do problema primal que permite analisar as condições de otimalidade do problema original de uma maneira diferente. O problema dual envolve a maximização ou minimização de uma função sujeita a restrições em termos das variáveis duais. As variáveis duais estão associadas às restrições do problema primal e fornecem uma medida da sensibilidade do problema primal diante de mudanças nas restrições.

A relação entre os problemas primal e dual é descrita pelo teorema da dualidade da programação linear [Neumann et al., 1944]. De acordo com esse teorema, descrito em Goldfarb e Todd [1989], se o problema primal ou o problema dual tem uma solução ótima finita, então o outro também tem e o valor ótimo dos problemas são iguais. Além disso, o teorema também diz que se um dos problemas tiver um valor de função objetivo ilimitado, o outro não terá solução viável.

Com relação a algoritmos de aprendizado, a formulação dual pode ser útil para problemas de classificação ou regressão em que os dados de entrada são transformados em características através de uma função de kernel. Nesses casos, a formulação dual permite que a função de kernel seja expressa em termos das variáveis duais, o que pode ser mais eficiente do que a expressão em termos das variáveis primais.

Como explicado por Leite e Neto [2008], para expressar o algoritmo FMP em variáveis duais, o vetor de pesos $\boldsymbol{\beta}$ pode ser escrito de acordo com a Equação 2.11. Essa definição de $\boldsymbol{\beta}$ foi extraída do estudo realizado por Boser et al. [1992] e descreve o hiperplano

separador como uma soma ponderada dos padrões de suporte, ou seja, aqueles para os quais $\alpha_i \neq 0$.

$$\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i. \tag{2.11}$$

Com isso a função de decisão desse algoritmo torna-se:

$$\begin{aligned} f(\mathbf{x}_j) &= sign(\boldsymbol{\beta}^T \mathbf{x}_j + b), \\ &= sign\left[\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right)^T \mathbf{x}_j + b\right], \\ &= sign\left[\sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j) + b\right]. \end{aligned} \tag{2.12}$$

A partir dessa formulação e considerando o critério de margem fixa estabelecido anteriormente, o seguinte critério de viabilidade é utilizado a cada verificação realizada durante o treinamento dessa rede:

$$y_i \left[\sum_{j=1}^{N} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + b\right] \geq \gamma_f \|\boldsymbol{\beta}\|. \tag{2.13}$$

Observando a Equação 2.13, percebe-se que ainda é necessário a definição de $\|\boldsymbol{\beta}\|$ em variáveis duais, o que pode ser feito como mostrado na Equação 2.14 [Leite e Neto, 2008].

$$\begin{aligned} \|\boldsymbol{\beta}\| &= \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j\right)^{1/2}, \\ &= \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j)\right)^{1/2}, \\ &= \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i y_i \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j)\right)^{1/2}. \end{aligned} \tag{2.14}$$

Além disso, para melhorar a precisão do algoritmo, torná-lo adequado ao processo de aprendizado online e reduzir o esforço computacional necessário para calcular o valor da norma, é viável utilizar uma abordagem aproximada na atualização de $\|\boldsymbol{\beta}\|$. Essa abordagem, mostrada em Villela et al. [2011], leva em consideração apenas o efeito da modificação após cada atualização do multiplicador $\alpha_i$, como mostrado na Equação 2.15.

$$\|\boldsymbol{\beta}\| \cong \left(\|\boldsymbol{\beta}\|^2 + \sum_{j=1}^{N} (\Delta\alpha_i y_i) \cdot \alpha_j y_j \cdot k(\mathbf{x}_i, \mathbf{x}_j)\right)^{1/2}. \tag{2.15}$$

Por fim, de acordo com Leite e Neto [2008], a cada ocorrência de um erro durante o processo de treinamento, o valor do respectivo multiplicador $\alpha_i$ é atualizado pela expressão:

$$\alpha_i = \alpha_i + \eta \cdot 1, \tag{2.16}$$

após a realização de um escalonamento do vetor $\boldsymbol{\alpha}$, dada pela Equação 2.17.

$$\boldsymbol{\alpha} = \boldsymbol{\alpha} \cdot (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|). \tag{2.17}$$

Considerando a formulação realizada acima, o pseudocódigo do Perceptron de Margem Fixa Dual está mostrado no Algoritmo 3.

**Algoritmo 3** Perceptron de Margem Fixa Dual

**Hiperparâmetros**: passo de atualização utilizado pelo FMP ($\eta$), número de atualizações máximo (T_MAX) e margem fixa ($\gamma_f$)
**Entrada:** *dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$, $\boldsymbol{\alpha}_{init}$ e termo bias inicial ($b_{init}$).
**Saída:** $\boldsymbol{\alpha}$.
$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_{init}$;
$b \leftarrow b_{init}$;
$t \leftarrow 0$;
$last\_t \leftarrow -1$;
**while** t $<$ T_MAX and last_t $\neq$ t **do**
    $last\_t \leftarrow t$
    $\|\boldsymbol{\beta}\| \leftarrow (\sum_j^N \sum_i^N \alpha_i y_i \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i))^{1/2}$
    **for** i = 1, ..., N **do**
        **if** $y_i \sum_{j=1}^N \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + b < \gamma_f \|\boldsymbol{\beta}\|$ **then**
            **if** $\|\boldsymbol{\beta}\| \neq 0$ **then**
                $\psi \leftarrow (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|)$;
            **else**
                $\psi \leftarrow 1$;
            **end if**
            $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} \cdot \psi$;
            $\alpha_i \leftarrow \alpha_i + \eta \cdot 1$;
            $b \leftarrow b + \eta y_i$
            $\|\boldsymbol{\beta}\| \leftarrow (\|\boldsymbol{\beta}\|^2 + \sum_{j=1}^N (\Delta\alpha_i y_i) \cdot \alpha_j y_j \cdot k(\mathbf{x}_i, \mathbf{x}_j))^{1/2}$
            $t \leftarrow t + 1$
        **end if**
    **end for**
**end while**
**return:** vetor de multiplicadores $\boldsymbol{\alpha}$ e termo bias $b$.

## 2.6 Máquinas de Vetores de Suporte

Uma Máquina de Vetores de Suporte (*Support Vector Machine* - SVM), proposta pelos trabalhos de Boser et al. [1992] e Cortes e Vapnik [1995], consiste em um algoritmo de

aprendizado de máquina utilizado para problemas de classificação e regressão. Considerando problemas de classificação o objetivo da SVM é encontrar um hiperplano que maximize a margem entre as classes, ou seja, a distância mínima entre o hiperplano e as amostras mais próximas de cada classe. A margem é importante porque quanto maior ela for, maior será a capacidade de generalização do modelo para dados não vistos.

Em casos em que as classes não podem ser separadas por um hiperplano, é necessário usar uma função de kernel para transformar os dados para um espaço de dimensão superior, onde a separação se torna possível. A função de kernel pode ser linear, polinomial, gaussiana, entre outras. A escolha da função de kernel e seus hiperparâmetros é um dos aspectos cruciais para o bom desempenho do modelo, já que uma escolha inadequada pode levar a *overfitting* ou *underfitting*.

Outro aspecto importante em SVMs é a escolha do parâmetro de regularização (normalmente chamado de C), que controla o *trade-off* entre a maximização da margem e a minimização do erro de classificação. Valores baixos de C flexibilizam a margem permitindo fronteiras de decisão com erros (isto é chamado de *soft margin*), enquanto valores altos de C geram maior penalização no erro de classificação, fazendo o algoritmo priorizar a separação completa entre classes.

Como descrito por Haykin e Network [2004], considerando um problema de classificação binária, em que as classes são rotuladas como 1 ou $-1$, dado um conjunto de treinamento cuja cada amostra consiste em um par entrada-saída $(\mathbf{x}_i, y_i)$, o problema de otimização a ser resolvido pela SVM é dado por:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{sujeito a:} \quad & y_i(\mathbf{w}^T\psi(\mathbf{x}_i) + b) \geq 1 - \xi_i \qquad \text{para i = 1, ..., N;} \\ & \xi_i \geq 0 \qquad \text{para i = 1, ..., N,} \end{aligned} \tag{2.18}$$

sendo que $\xi_i$ são variáveis de folga, $\mathbf{w}$ é o vetor de pesos a ser encontrado e $\psi(\mathbf{x}_i)$ consiste em uma função aplicada ao espaço de entrada com intuito de projetar a amostra $\mathbf{x}_i$ em um espaço de características com melhor separabilidade.

Utilizando o método dos multiplicadores de Lagrange, esse problema de otimização pode ser formulado na sua versão dual, como apresentado na Equação 2.19.

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j (\psi(\mathbf{x}_i)\cdot\psi(\mathbf{x}_j)) \\ \text{sujeito a:} \quad & \sum_{i=1}^{N} y_i\alpha_i = 0; \\ & 0 \leq \alpha_i \leq C \qquad \text{para todo i.} \end{aligned} \tag{2.19}$$

O produto $(\psi(\mathbf{x}_i) \cdot \psi(\mathbf{x}_j))$ pode ser substituído por uma função kernel $k$, que satisfaça: $k(\mathbf{x}_i, \mathbf{x}_j) = \psi(\mathbf{x}_i) \cdot \psi(\mathbf{x}_j)$. Assim, o problema de otimização a ser resolvido é:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{sujeito a:} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0; \\ & 0 \leq \alpha_i \leq C \qquad \text{para todo i,} \end{aligned} \tag{2.20}$$

em que $\alpha_i$ é o multiplicador de Lagrange.

Como expresso no trabalho de Horta [2015], o vetor de pesos ótimo ($\mathbf{w}^*$) e o termo bias ótimo ($b^*$) são apresentados, respectivamente, nas Equações 2.21 e 2.22.

$$\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \psi(\mathbf{x}_i), \tag{2.21}$$

$$b^* = -\frac{1}{2} \left[ \max_{\{i|y_i=-1\}} \left( \sum_{j=1}^{N_{SV}} y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right) + \min_{\{i|y_i=+1\}} \left( \sum_{j=1}^{N_{SV}} y_j \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right) \right]. \tag{2.22}$$

Os valores $\alpha_i^*$ podem ser obtidos utilizando programação quadrática para solucionar o problema de otimização expresso em 2.20. O termo $N_{SV}$ refere-se ao número de vetores de suporte (amostras associadas a multiplicadores de Lagrange que estão na faixa $0 < \alpha_i \leq C$).

A função de decisão da SVM para uma amostra de entrada $\mathbf{x}$ está expressa na Equação 2.23. Dessa equação é possível notar que as amostras de entrada que não são vetores de suporte não possuem influência na função de decisão da SVM [Semolini et al., 2002].

$$f(x) = sign \left( \sum_{i=1}^{N_{SV}} y_i \alpha_i^* k(\mathbf{x}_i, \mathbf{x}) + b^* \right) \tag{2.23}$$

Por fim, pode-se dizer que SVM é uma técnica poderosa para problemas de classificação e regressão e oferece uma combinação de simplicidade conceitual e bom desempenho em uma ampla variedade de aplicações. No entanto, é importante uma escolha adequada dos hiperparâmetros e da função de kernel para evitar *overfitting* ou *underfitting*. Além disso, por utilizar programação quadrática para resolver seu problema de otimização, é necessário considerar a escalabilidade computacional em grandes conjuntos de dados.

Enquanto a SVM realiza um mapeamento implícito dos dados de entrada, nas seções seguintes são apresentadas redes de uma única camada escondida baseadas em um mapeamento explícito. A principal diferença entre o mapeamento explícito e o mapeamento implícito é que o primeiro realiza uma transformação explícita dos dados para um espaço

de maior dimensionalidade (com o objetivo de tornar os dados linearmente separáveis no novo espaço). Já o segundo utiliza uma técnica conhecida como *kernel trick* para operar implicitamente no espaço de maior dimensionalidade, sem a necessidade de calcular a transformação completa.

## 2.7 Redes Neurais *Feedforward* de Camada Oculta Única

As *Single-Hidden Layer Feedforward Networks* (SLFN) são redes neurais compostas por uma camada de entrada, que recebe os dados do ambiente, uma camada oculta, que visa realizar uma transformação não linear nos dados e finalmente uma camada de saída, cujo objetivo é gerar o resultado final da rede para uma entrada.

Como este trabalho busca lidar com problemas de classificação binária, a saída das redes estudadas aqui para uma amostra de entrada será a categorização da mesma em uma das classes. Para esta categoria de problemas, é comum recorrer ao uso de redes SLFN cuja a determinação da classe de saída envolve o cálculo da soma ponderada das saídas dos neurônios na camada oculta, seguido pela aplicação de uma função de ativação degrau.

Estas redes SLFN podem ser treinadas em duas etapas, primeiramente é realizada uma transformação não linear das amostras para um espaço de maior dimensão que o espaço de entrada e posteriormente os pesos da camada de saída são definidos pela solução de um problema de mínimos quadrados Huang et al. [2004].

A vantagem dessa abordagem é a simplicidade e interpretabilidade do modelo resultante. Os pesos da camada de saída podem ser interpretados diretamente como coeficientes que multiplicam as saídas dos neurônios da camada oculta. Isso facilita a interpretação dos pesos e o entendimento do impacto de cada neurônio na saída da rede.

Nas seções a seguir são apresentadas dois exemplos de redes SLFN. A seção 2.8 mostra uma rede com uma camada oculta configurada com pesos aleatórios, enquanto a seção 2.9 apresenta uma rede cujos neurônios na camada oculta implementam funções de base radial.

## 2.8 Máquinas de Aprendizado Extremo

De acordo com Huang et al. [2006], uma Máquinas de Aprendizado Extremo (*Extreme Learning Machine* - ELM) se constitui como um método de treinamento simples para as SLFNs com camada de saída linear, composto basicamente pelas seguintes três etapas:

1. Gerar aleatoriamente os pesos de entrada $\mathbf{w}_i$ e os termos *bias* $b_i$, i = 1, ..., $\tilde{\text{N}}$, sendo $\tilde{\text{N}}$ o número de neurônios da camada escondida.

2. Calcular a matriz de mapeamento $\mathbf{H}$ (saída da camada escondida), tal que:

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{x}_1) \\ \vdots \\ h(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{Nx\tilde{N}}, \tag{2.24}$$

   dado que N é a quantidade de amostras de treinamento, $\mathbf{x}_i$ (com $i = 1, ..., N$) são as amostras e $g(\cdot)$ é a função de ativação utilizada.

3. Calcular os pesos de saída $\boldsymbol{\beta}$, tal que:

$$\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{Y}, \tag{2.25}$$

   sendo $\mathbf{H}^{+}$ a pseudoinversa de Moore-Penrose de $\mathbf{H}$, e $\mathbf{Y} = [y_1, \dots, y_N]^T$ é o vetor contendo as saídas correspondentes às N amostras de treinamento.

No caso de um problema de classificação binária onde as classes são rotuladas como 1 ou $-1$, a saída de uma ELM, com $\tilde{\text{N}}$ neurônios na camada escondida, para uma amostra $\mathbf{x}$ pode ser dada por:

$$\hat{y}_i = sign(\sum_{i=1}^{\tilde{\text{N}}} \beta_i h_i(\mathbf{x})), \tag{2.26}$$

sendo a função *sign* definida como mostrado na equação 2.27.

$$sign(x) = \begin{cases} -1 & \text{se } x < 0 \\ +1 & \text{caso contrário,} \end{cases} \tag{2.27}$$

A Figura 1 apresenta a topologia típica da ELM quando aplicada a problemas de classificação binária.

Entre as vantagens proeminentes das redes ELM, destacam-se sua simplicidade, facilidade de implementação e eficiência computacional. No entanto, uma preocupação conhecida em relação às ELMs é a alta probabilidade de ocorrer *overfitting*. Isso se deve ao fato de que, devido à projeção em um espaço de alta dimensão, a rede naturalmente tende a se tornar superdimensionada [Silvestre, 2014].

## 2.9 Redes Neurais de Base Radial

As redes neurais com função de base radial (*Radial Basis Function Neural Network* - RBFNN) na sua forma básica possuem uma arquitetura *feedforward* composta por três

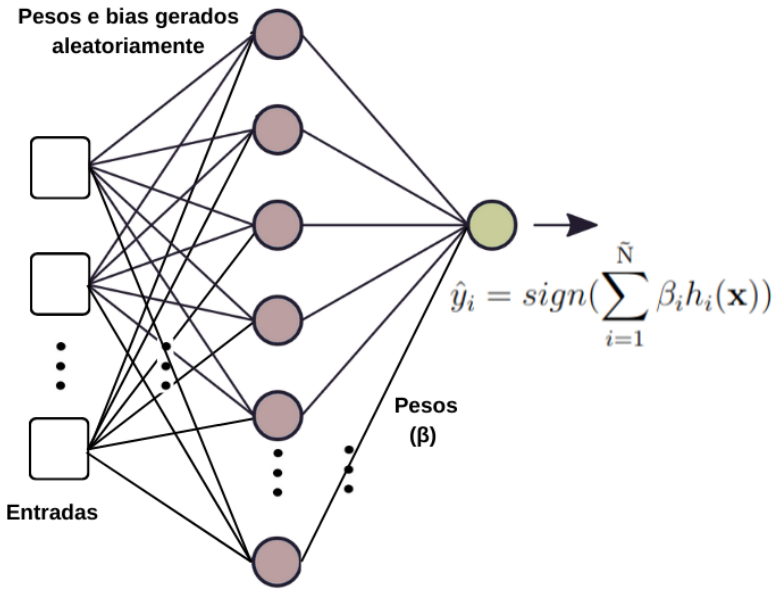


Figura 1 – Topologia típica de uma ELM.

camadas (camada de entrada, camada escondida única e camada de saída). Cada neurônio da camada escondida dessa rede implementa uma função de base radial e o objetivo da camada de saída é combinar linearmente as respostas desses neurônios. Dessa forma, as RBFNNs também se constituem como mais um exemplo de SLFN com camada de saída linear.

A função gaussiana é tipicamente utilizada como função ativação dos neurônios da camada escondida e é dada por [Bishop et al., 1995]:

$$h_i(\mathbf{x}_j) = \exp \frac{-\|\mathbf{x}_j - \boldsymbol{\mu}_i\|}{2\sigma_i^2}, \tag{2.28}$$

sendo que $\mathbf{x}_j$ é um vetor de entrada, $\boldsymbol{\mu}_i$ é o vetor que determina o centro para a função $h_i$ e $\sigma_i$ é um parâmetro que controla a largura (ou raio) da função gaussiana.

O processo de treinamento de uma RBFNN pode ser realizado em duas etapas e começa com a definição dos parâmetros das funções radiais ($\boldsymbol{\mu}$ e $\sigma$). Esses parâmetros são definidos na primeira etapa por meio de métodos não supervisionados. Em seguida, na segunda etapa, ocorre o ajuste dos pesos dos neurônios da camada de saída. Espera-se que a saída da camada oculta seja linearmente separável, assim os pesos de saída podem ser otimizados utilizando modelos lineares [Braga et al., 2000].

Para a definição dos centros das funções de base radial implementadas pelos neurônios na camada escondida é muito comum a utilização de técnicas de agrupamento, como por exemplo o k-means. Esse algoritmo, proposto por McQueen [1967], busca agrupar os dados em K grupos distintos com base na similaridade. Considerando a distância como medida de similaridade utilizada, o objetivo do k-means é atribuir cada ponto a um dos K

grupos de forma que a soma dos quadrados das distâncias entre cada ponto e o centroide do seu grupo seja minimizada. Nesse caso, os dados de entrada são usados para identificar os centros dos neurônios na camada escondida.

Para calcular o parâmetro $\sigma$ dos neurônios da camada escondida existem diferentes técnicas. Uma muito comum é definir esse parâmetro como a média das distâncias entre o centro do neurônio e as outras amostras.

Considerando o k-means para determinar os centros das funções radiais implementadas pelos neurônios, uma possível técnica de treinamento da RBFNN pode ser composto pelas seguintes etapas:

1. Uma vez escolhido um valor para K, que representa o número de neurônios desejados na camada escondida, o primeiro passo é definir os centros das funções de base radial geradas por esses neurônios. Esses parâmetros são determinados como sendo os centroides identificados pelo algoritmo k-means.

2. Calcular o parâmetro que controla a largura da função radial implementada por cada neurônio da camada escondida.

3. Calcular a matriz de mapeamento H:

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \dots & h_{\tilde{N}}(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \dots & h_{\tilde{N}}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & h_2(\mathbf{x}_N) & \dots & h_{\tilde{N}}(\mathbf{x}_N) \end{bmatrix}_{NxK}, \tag{2.29}$$

   dado que N é a quantidade de amostras de treinamento, $\mathbf{x}_i$ (com $i = 1, ..., N$) são as amostras e $h_i(x)$ é a função de ativação utilizada.

4. Calcular os pesos de saída $\boldsymbol{\beta}$, tal que:

$$\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{Y}, \tag{2.30}$$

   sendo $\mathbf{H}^{+}$ a pseudoinversa de Moore-Penrose de $\mathbf{H}$, e $\mathbf{Y} = [y_1, \dots, y_N]^T$ é o vetor contendo as saídas correspondentes às N amostras de treinamento.

Assim, o treinamento desta rede consiste em basicamente definir os parâmetros da função de ativação aplicada na camada escondida, e após isso calcular os pesos da camada de saída linear pela solução de um problema de mínimos quadrados. Considerando um problema de classificação binária, a saída dessa rede é calculada da mesma forma que na rede ELM, como mostrado na Equação 2.26.

A arquitetura de uma RBFNN para problemas binários, mostrada na Figura 2, assemelha-se àquela definida para a ELM. No entanto, uma distinção importante reside

no fato de que, enquanto os parâmetros da camada oculta da ELM são os pesos definidos aleatoriamente, na RBFNN os parâmetros da camada oculta são os componentes $\boldsymbol{\mu}$ e $\sigma$ da função de base radial implementada por cada neurônio.

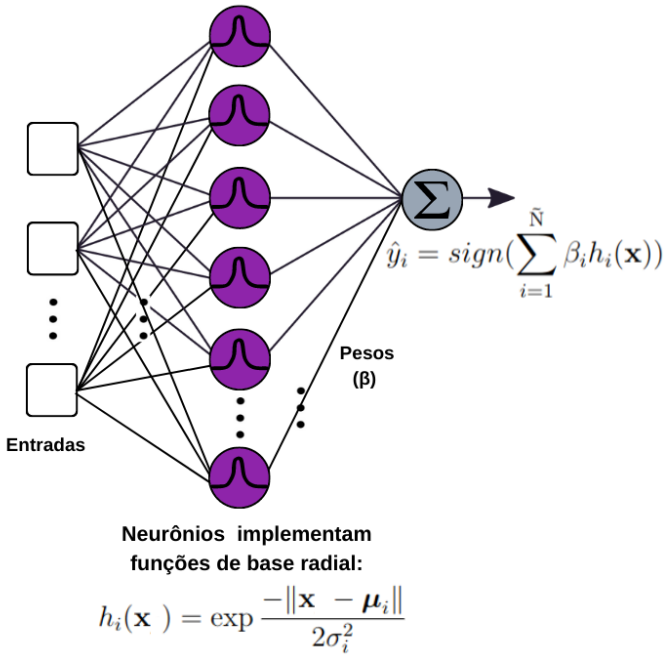


Figura 2 – Topologia típica de uma RBFNN.

# Capítulo 3

# Métodos Propostos

O objetivo deste capítulo é apresentar novos métodos de treinamento para classificadores baseados em redes neurais de camada oculta única. Esses métodos visam a obtenção de uma margem larga de separação entre as classes na camada de saída da rede. Ao longo deste capítulo, tem-se uma descrição de cada método.

## 3.1 Algoritmo de Margem Incremental com Projeção Aleatória

O primeiro método proposto por este trabalho, chamado *Random Projection Incremental Margin Algorithm* (RP-IMA), se trata de um algoritmo de treinamento para uma rede SLFN com projeção aleatória. O RP-IMA propõe a utilização de uma projeção aleatória (na camada oculta de uma SLFN) juntamente com um algoritmo de margem incremental para o cálculo dos pesos do neurônio de saída da SLFN. O neurônio de saída da rede obtém um resultado de margem larga para a camada de saída encontrando sucessivamente a solução para um problema de classificação considerando valores de margem crescentes.

Os pesos da camada oculta do RP-IMA são selecionados aleatoriamente. O algoritmo inicializa um conjunto de pesos aleatórios para as conexões entre a camada de entrada e a camada oculta. Esses pesos são obtidos a partir de uma distribuição uniforme no intervalo $[-1, 1]$. Uma vez que uma função de mapeamento explícito é definida, os pesos da camada de saída são treinados em um espaço de características utilizando o IMA.

O objetivo do método proposto é utilizar a projeção aleatória para melhorar a separabilidade do problema e então aplicar um algoritmo de margem incremental para obter um classificador de margem larga. Ao maximizar a margem de decisão no espaço de características, espera-se obter um classificador com melhor capacidade de generalização e mais robusto, tendo seu desempenho menos afetado por variações na arquitetura da rede

(o que é um problema na implementação original da ELM).

As vantagens de se utilizar a projeção aleatória como a forma de realizar o mapeamento explícito são sua simplicidade, baixo custo de treinamento e o fato dela não requerer nenhum hiperparâmetro adicional ao número de neurônios ocultos para realizar o mapeamento.

De maneira mais formal, o RP-IMA consiste basicamente em duas fases:

1. Execução das duas primeiras etapas do treinamento da ELM conforme descrito na seção anterior: atribuir aleatoriamente os pesos e termos *bias* da camada escondida e calcular a matriz de mapeamento $\mathbf{H}$ (conforme a Equação 2.24).

2. Aplicação do algoritmo de margem incremental (IMA primal) com o esquema de margem flexível, usando o conjunto ($\mathbf{H}$, $\mathbf{Y}$) como dados de entrada. O $\boldsymbol{\beta}$ final, retornado do IMA, será o vetor de pesos de saída $\boldsymbol{\beta}$ do RP-IMA.

O Algoritmo 4 mostra o pseudocódigo do RP-IMA implementado nesse trabalho. Esse algoritmo é apresentado considerando uma formulação genérica para a resolução de problemas de classificação binária de margem larga com norma arbitrária.

Após a conclusão do treinamento de todos os parâmetros da rede neural, basta aplicar a Equação 3.1 para realizar a classificação de um nova amostra de entrada $\mathbf{x}$.

$$\hat{y}_i = sign\left(\sum_{i=1}^{\tilde{\mathrm{N}}} \beta_i h_i(\mathbf{x}) + b\right), \tag{3.1}$$

onde $\tilde{\mathrm{N}}$ denota o número de neurônios na camada oculta, $\beta_i$ corresponde ao peso de saída associado ao neurônio oculto de índice $i$, $h_i(\mathbf{x})$ representa a transformação efetuada por cada neurônio na camada escondida e $b$ é o termo *bias* do neurônio de saída.

## 3.2 Rede Neural de Base Radial com Algoritmo de Margem Incremental

Outra abordagem estudada neste trabalho foi utilizar o algoritmo de margem incremental em conjunto com outro tipo de rede SLFN, que não tivesse os pesos da camada escondida definidos aleatoriamente. Assim nessa seção será apresentado um método que utiliza o algoritmo de margem incremental para treinar os neurônios da camada de saída de uma RBFNN com objetivo de obter uma margem de separação maior nesse espaço.

Esse novo método de treinamento será chamado aqui de *Incremental Margin Radial Basis Function Neural Network* (IM-RBFNN). Assim como no método anterior, a

**Algoritmo 4** RP-IMA$_p$

**Hiperparâmetros**: Número de neurônios da camada escondida ($\tilde{\text{N}}$), passo de atualização utilizado pelo FMP ($\eta$), incremento mínimo da margem ($\delta$), termo constante da variável de folga ($\lambda$), número de atualizações máximo (T_MAX).
**Entrada:** *Dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$
**Saída:** Rede neural com pesos definidos.
Gerar aleatoriamente os pesos de entrada $\mathbf{w}_i$ e o termo *bias* $b_i$, i = 1, ..., $\tilde{\text{N}}$, sendo $\tilde{\text{N}}$ o número de neurônios da camada escondida.

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{x}_1) \\ \vdots \\ h(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{Nx\tilde{N}};$$

$\boldsymbol{\beta}[1, ..., \tilde{\text{N}}] \leftarrow 0$;
$b \leftarrow 0$;
$\gamma_f \leftarrow 0$
$t \leftarrow 0$;
IMA_iterations $\leftarrow 0$;
**while** $t < T\_MAX$ **do** ▷ Equivalente a: **while** FMP$_p$ convergir em menos de T_MAX atualizações **do**
    $t \leftarrow 0$;
    $last\_t \leftarrow -1$;
    $\alpha[1, ..., N] \leftarrow 0$;
    **while** t < T_MAX and last_t ≠ t **do** ▷ Este bloco **while** corresponde ao FMP.
        $last\_t \leftarrow t$;
        **for** i = 1, ..., N **do**
            **if** $y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b) < \gamma_f \|\boldsymbol{\beta}\|_q - \alpha_i \lambda$ **then** ▷ Verificação de erro
                **if** $\|\boldsymbol{\beta}\|_q \neq 0$ **then**
                    $\psi \leftarrow (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|_q)$;
                **else**
                    $\psi \leftarrow 1$;
                **end if**
                $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}\psi + \eta y_i \mathbf{h}_i$; ▷ Atualização dos pesos de saída
                $b \leftarrow b + \eta y_i$ ; ▷ Atualização do termo bias
                $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} \cdot \psi$;
                $\alpha_i \leftarrow \alpha_i + \eta \cdot 1$; ▷ Atualização da variável de folga para $h_i$
                $t \leftarrow t + 1$ ;
            **end if**
        **end for**
    **end while**
    $\gamma^+(\boldsymbol{\beta}) = min\{y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = +1\}$ ▷ Margem para a classe +1
    $\gamma^-(\boldsymbol{\beta}) = min\{y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = -1\}$ ▷ Margem para a classe -1
    $\gamma_f = max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1 + \delta)\gamma_f)$; ▷ Nova margem fixa
    IMA_iterations $\leftarrow$ IMA_iterations + 1;
**end while**
**return:** Pesos aprendidos da rede neural.

camada escondida do IM-RBFNN é responsável por realizar a transformação não-linear dos dados de entrada para um espaço de características de maior dimensão. Cada neurônio na camada escondida implementa uma função de base radial (que neste trabalho será uma gaussiana). Dessa forma, com a projeção para um novo espaço definida, os pesos da camada de saída são treinados utilizando o IMA.

O algoritmo proposto apresenta as seguintes etapas de forma itemizada:

1. Definição dos centros das funções de base radial implementadas pelos neurônios da camada escondida. Isso pode ser feito com a utilização de um algoritmo de agrupamento. Neste trabalho os centros foram definidos como os centroides dos agrupamentos encontrados pelo algoritmo *k-means*.

2. Cálculo dos parâmetros $\sigma$ das funções de base radial implementadas pelos neurônios da camada escondida. Nesta implementação, para cada neurônio esse parâmetro foi definido como a média das distâncias euclidianas das amostras do agrupamento até o centro da função radial implementada pelo neurônio.

3. Cálculo da matriz de mapeamento H, explicitada na Equação 2.29. A função de base radial utilizada aqui foi a gaussiana, mostrada na Equação 2.28.

4. Aplicação da versão primal do algoritmo de margem incremental (IMA) com o esquema de margem flexível, usando o conjunto ($\mathbf{H}$, $\mathbf{Y}$) como dados de entrada. O $\boldsymbol{\beta}$ final, retornado pelo IMA, será o vetor de pesos de saída $\boldsymbol{\beta}$ do IM-RBFNN.

Uma vez determinados todos os parâmetros dessa rede, o resultado de classificação para uma amostra de entrada $\mathbf{x}$ é dado por:

$$\begin{aligned} \hat{y}_i &= sign\left(\sum_{i=1}^{\tilde{\mathrm{N}}} \beta_i h_i(\mathbf{x}) + b\right), \\ &= sign\left(\sum_{i=1}^{\tilde{\mathrm{N}}} \beta_i \cdot \exp\frac{-\|\mathbf{x}-\boldsymbol{\mu}_i\|}{2\sigma_i^2} + b\right), \end{aligned} \tag{3.2}$$

onde $\tilde{\mathrm{N}}$ denota o número de neurônios na camada oculta, $\beta_i$ corresponde ao peso de saída associado ao neurônio oculto de índice $i$, $\boldsymbol{\mu}_i$ e $\sigma_i$ são, respectivamente, o centro e o termo que controla a largura da função gaussiana implementada pelo i-ésimo neurônio da camada escondida e $b$ é o termo *bias* do neurônio de saída.

O Algoritmo 5 apresenta o pseudocódigo do IM-RBFNN implementado nesse trabalho. Este algoritmo é apresentado considerando uma formulação genérica para a resolução de problemas de classificação binária de margem larga com norma arbitrária.

**Algoritmo 5** IM-RBFNN$_p$

1: **Hiperparâmetros**: Número de neurônios da camada escondida ($\tilde{\text{N}}$), passo de atualização utilizado pelo FMP ($\eta$), incremento mínimo da margem ($\delta$), termo constante da variável de folga ($\lambda$), número de atualizações máximo (T_MAX).
2: **Entrada:** *Dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$
3: **Saída:** Rede neural com pesos definidos.
4: Rodar o *K-means* e determinar cada $\mu_i$, $i = 1, ..., \tilde{\text{N}}$, como o valor de cada centroide.
5: Determinar $\sigma_i$, $i = 1, ..., \tilde{\text{N}}$, como a distância média entre os pontos no agrupamento e o respectivo centro.
6: $\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \dots & h_{\tilde{N}}(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \dots & h_{\tilde{N}}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & h_2(\mathbf{x}_N) & \dots & h_{\tilde{N}}(\mathbf{x}_N) \end{bmatrix}_{NxN}$, sendo $h_i(\mathbf{x}_j) = \exp \frac{-\|\mathbf{x}_j - \boldsymbol{\mu}_i\|}{2\sigma_i^2}$;
7: $\boldsymbol{\beta}[1, ..., \tilde{\text{N}}] \leftarrow 0$;
8: $b \leftarrow 0$;
9: $\gamma_f \leftarrow 0$
10: $t \leftarrow 0$;
11: IMA_iterations $\leftarrow 0$;
12: **while** $t < T\_MAX$ **do** ▷ Equivalente a: **while** FMP$_p$ convergir em menos de T_MAX atualizações **do**
13: $t \leftarrow 0$;
14: $last\_t \leftarrow -1$;
15: $\alpha[1, ..., N] \leftarrow 0$;
16: **while** t < T_MAX and last_t ≠ t **do** ▷ Este bloco **while** corresponde ao FMP.
17: $last\_t \leftarrow t$;
18: **for** i = 1, ..., N **do**
19: **if** $y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b) < \gamma_f \|\boldsymbol{\beta}\|_q - \alpha_i \lambda$ **then** ▷ Verificação de erro
20: **if** $\|\boldsymbol{\beta}\|_q \neq 0$ **then**
21: $\psi \leftarrow (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|_q)$;
22: **else**
23: $\psi \leftarrow 1$;
24: **end if**
25: $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}\psi + \eta y_i \mathbf{h}_i$; ▷ Atualização dos pesos de saída
26: $b \leftarrow b + \eta y_i$ ; ▷ Atualização do termo bias
27: $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} \cdot \psi$;
28: $\alpha_i \leftarrow \alpha_i + \eta \cdot 1$; ▷ Atualização da variável de folga para $h_i$
29: $t \leftarrow t + 1$ ;
30: **end if**
31: **end for**
32: **end while**
33: $\gamma^+(\boldsymbol{\beta}) = min\{y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = +1\}$ ▷ Margem para a classe +1
34: $\gamma^-(\boldsymbol{\beta}) = min\{y_i(\mathbf{h}_i \cdot \boldsymbol{\beta} + b)/\|\boldsymbol{\beta}\|_q, \forall y_i = -1\}$ ▷ Margem para a classe -1
35: $\gamma_f = max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1 + \delta)\gamma_f)$; ▷ Nova margem fixa
36: IMA_iterations ← IMA_iterations + 1;
37: **end while**
38: **return:** Pesos aprendidos da rede neural.

## 3.3 Poda de Neurônios

Métodos de poda de neurônios são técnicas utilizadas para reduzir o tamanho de uma rede neural, removendo neurônios e conexões considerados menos importantes. A poda de neurônios visa reduzir a complexidade e o custo computacional de uma rede neural, mantendo um bom desempenho.

Existem diversas técnicas e critérios para a poda de neurônios. Aqui será utilizada uma abordagem de poda iterativa baseada na magnitude dos pesos de saída, ou seja, os neurônios da camada escondida associados a pesos de saída com magnitudes significativamente reduzidas são considerados menos importantes e são removidos da rede.

Conforme mostrado por Villela et al. [2016], a formulação primal $\text{IMA}_\infty$ leva a soluções onde o vetor $\boldsymbol{\beta}$ é mais esparso do que em outras formulações. Da mesma forma, espera-se que os métodos $\text{RP-IMA}_\infty$ e $\text{IM-RBFNN}_\infty$ (implementações que visam obter um hiperplano de margem $L_\infty$, ou seja que minimizem o valor da norma $L_1$ do vetor $\boldsymbol{\beta}$) resultem em um vetor de pesos de saída mais esparso do que em outras formulações.

Com base nisso, propõe-se aqui o seguinte método de poda para essas arquiteturas de rede: a cada iteração dos métodos $\text{RP-IMA}_\infty$ e $\text{IM-RBFNN}_\infty$, verifica-se quais componentes do vetor de saída $\boldsymbol{\beta}$ têm valores absolutos abaixo do limite definido por $\rho \cdot \max_i |\beta_i|$, onde $\rho$ é um hiperparâmetro. Esses componentes e seus respectivos neurônios ocultos são eliminados, o que corresponde à remoção dos vetores coluna da matriz $\mathbf{H}$ associados aos elementos $\beta_i$ eliminados.

Resumidamente, após cada execução do algoritmo perceptron de margem fixa, elimina-se os neurônios da camada oculta cujos pesos $\beta_i$ associados são menores que um percentual do valor absoluto do peso de maior magnitude. Esse percentual é definido através do hiperparâmetro $\rho$.

## 3.4 Versão Dual do Algoritmo de Margem Incremental com Projeção Aleatória

Com a intenção de gerar uma nova rede na qual a camada de saída tenha a capacidade de classificação de uma máquina de *kernel*, foi desenvolvido o algoritmo RP-IMA Dual. Diferentemente do método apresentado na seção anterior, essa versão dual utiliza o algoritmo IMA em conjunto com o FMP dual para o treinamento dos pesos da camada de saída da rede.

Assim, o treinamento da rede RP-IMA Dual para um dataset de entrada $(\mathbf{X}, \mathbf{y})$ pode ser definido pela execução das seguintes etapas:

1. Atribuição aleatória dos pesos e dos termos *bias* da camada escondida e cálculo da

matriz de mapeamento $\mathbf{H}$, como indicado pela Equação 2.24.

2. Cálculo da matriz de *kernel* $\boldsymbol{K}$. Para esse trabalho foi utilizado o *kernel* Gaussiano, ou seja:

$$K_{ij} = \exp -\frac{\|\mathbf{h}_i \cdot \mathbf{h}_j\|^2}{2\sigma^2} \tag{3.3}$$

   Essa matriz será utilizada pelo FMP para os cálculos de $\|\boldsymbol{\beta}\|$ (Equações 2.14 e 2.15) e do critério de viabilidade (Equação 2.13). A matriz de *kernel* $\boldsymbol{K}$ pode ser precomputada, evitando assim a necessidade dela ser recalculada a cada execução algoritmo FMP dual.

3. Sucessiva aplicação do algoritmo de FMP dual, considerando o conjunto $((\mathbf{H}, \mathbf{y}))$ como dados de entrada. Isso pode ser feito enquanto o método convergir em até uma quantidade definida de atualizações dos multiplicadores $\alpha_i$ ou até atingir um número desejado de iterações. Entre cada execução do FMP dual a margem é atualizada de acordo com a Equação 2.10.

De maneira mais minuciosa, o Algoritmo 6 apresenta passo a passo a sequência de operações executadas pelo método RP-IMA Dual proposto. Esse algoritmo encapsula o processo que combina a técnica de projeção aleatória com a abordagem de margem incremental (IMA) na sua versão dual.

Dispondo de todos os parâmetros dessa rede treinados, o resultado de classificação para uma amostra de entrada $\mathbf{x}$ é dado por:

$$\begin{aligned} \hat{y}_i &= sign\left(\sum_{i=1}^{\mathrm{N}} \alpha_i y_i K_i(h(\mathbf{x}), \mathbf{h}_i)\right), \\ &= sign\left(\sum_{i=1}^{\mathrm{N}} \alpha_i y_i \exp\frac{-\|h(\mathbf{x}) - \mathbf{h}_i\|}{2\sigma_i^2}\right), \end{aligned} \tag{3.4}$$

sendo que $\mathbf{h}_i$ é a linha $i$ da matriz de mapeamento $\mathbf{H}$ (definida durante o treinamento), $h(\mathbf{x})$ representa o mapeamento realizado pela camada oculta para a nova amostra $\mathbf{x}$, $\alpha_i$ e $y_i$ são, respectivamente, o multiplicador relacionado à i-ésima amostra de treinamento e o rótulo dessa amostra, e N equivale ao número de amostras de treinamento.

Apesar de nesse trabalho ter-se utilizado um *kernel* gaussiano (também conhecido como RBF), outros tipos de *kernel* podem ser utilizados, como por exemplo, o *kernel* linear e o *kernel* polinomial, cujas respectivas funções estão mostradas nas Equações 3.5 e 3.6.

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2 \tag{3.5}$$

**Algoritmo 6** RP-IMA Dual

**Hiperparâmetros**: número de neurônios da camada escondida ($\tilde{\text{N}}$), passo de atualização utilizado pelo FMP ($\eta$), incremento mínimo da margem a cada iteração do IMA ($\delta$), número de atualizações máximo (T_MAX).
**Entrada:** *Dataset* de treinamento com N amostras $\{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbf{R}^n, y_i \in \{-1, 1\}, i = 1, ..., N\}$
**Saída:** Rede neural com pesos definidos.
Gerar aleatoriamente os pesos de entrada $\mathbf{w}_i$ e o termo *bias* $b_i$, i = 1, ..., $\tilde{\text{N}}$, sendo $\tilde{\text{N}}$ o número de neurônios da camada escondida;
Calcular a matriz de mapeamento $\mathbf{H}$, sendo cada elemento $h_{ij} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j)$;
Calcular matriz de *kernel* $\mathbf{K}$, sendo cada elemento $k_{ij} = \exp\left(-\frac{\|\mathbf{h}_i - \mathbf{h}_j\|}{2\sigma^2}\right)$;
$\boldsymbol{\alpha}[1, ..., N] \leftarrow 0$;
$\gamma_f \leftarrow 0$;
**while** $t < T\_MAX$ **do** ▷ Equivalente a: **while** FMP Dual convergir em menos de T_MAX atualizações **do**
    $t \leftarrow 0$;
    $last\_t \leftarrow -1$;
    **while** t < T_MAX and last_t ≠ t **do**
        $last\_t \leftarrow t$
        $\|\boldsymbol{\beta}\| \leftarrow (\sum_j^N \sum_i^N \alpha_i y_i \alpha_j y_j k(\mathbf{h}_j, \mathbf{h}_i))^{1/2}$ ▷ Norma do vetor de pesos de saída
        **for** i = 1, ..., N **do**
            **if** $y_i \sum_{j=1}^N \alpha_j y_j k(\mathbf{h}_j, \mathbf{h}_i) < \gamma_f \|\boldsymbol{\beta}\|$ **then** ▷ Verificação de erro
                **if** $\|\boldsymbol{\beta}\| \neq 0$ **then**
                    $\psi \leftarrow (1 - (\eta\gamma_f)/\|\boldsymbol{\beta}\|)$;
                **else**
                    $\psi \leftarrow 1$;
                **end if**
                $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} \cdot \psi$;
                $\alpha_i \leftarrow \alpha_i + \eta \cdot 1$;
                $\|\boldsymbol{\beta}\| \leftarrow \|\boldsymbol{\beta}\|^2 (\sum_{j=1}^N (\Delta\alpha_i y_i) \cdot \alpha_j y_j \cdot k(\mathbf{h}_i, \mathbf{h}_j))^{1/2}$ ▷ Atualização de $\|\boldsymbol{\beta}\|$
                $t \leftarrow t + 1$
            **end if**
        **end for**
    **end while**
    $\gamma^+(\boldsymbol{\beta}) = min\{y_i (\sum_{j=1}^N \alpha_j y_j k(\mathbf{h}_i, \mathbf{h}_j))/\|\boldsymbol{\beta}\|, \forall y_i = +1\}$ ▷ Margem para a classe +1
    $\gamma^-(\boldsymbol{\beta}) = min\{y_i (\sum_{j=1}^N \alpha_j y_j k(\mathbf{h}_i, \mathbf{h}_j))/\|\boldsymbol{\beta}\|, \forall y_i = -1\}$ ▷ Margem para a classe -1
    $\gamma_f = max((\gamma^+(\boldsymbol{\beta}) + \gamma^-(\boldsymbol{\beta}))/2, (1 + \delta)\gamma_f)$; ▷ Nova margem fixa
**end while**
**return:** Pesos aprendidos da rede neural.

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^d \tag{3.6}$$

Um detalhe importante é que para a utilização dos *kernels* polinomial ou gaussiano é necessário definir mais um hiperparâmetro. No caso do *kernel* polinomial é necessária a

definição do grau do polinômio ($d$), e no caso do *kernel* RBF deve-se determinar o valor do parâmetro $\sigma$, que controla a largura da função gaussiana.

Outra questão explorada aqui diz respeito a flexibilização da margem para este algoritmo dual. Sabe-se que é possível garantir a positividade da matriz *kernel* ao aumentar de forma apropriada o valor da sua diagonal principal, como $\boldsymbol{K} = \boldsymbol{K} + \lambda \boldsymbol{I}$. Contudo, essa operação está também diretamente relacionada à imposição de uma margem flexível na construção do classificador [Villela et al., 2011]. Dessa maneira, quanto maior o valor do parâmetro $\lambda$, maior será a flexibilização da margem aplicada.

Além disso, adição do termo $\lambda I$ à matriz *kernel* é também uma técnica de regularização que tem o efeito de controlar a complexidade do modelo, promovendo uma melhor generalização. Ao aumentar o valor de $\lambda$, aumenta-se a importância da regularização em relação à função de perda durante o treinamento. Isso resulta em uma penalidade maior para os parâmetros do modelo, incentivando-os a ter valores menores e, assim, evitando ajustes excessivos aos dados de treinamento.

Por fim, com o intuito de poder observar a influência de se obter uma margem larga na última camada e comparar a performance do RP-IMA Dual com um algoritmo semelhante, foi implementada ainda nesse trabalho uma versão dual da ELM. Esse algoritmo será detalhado na próxima seção.

## 3.5 Versão Dual de uma Máquina de Aprendizado Extremo

Esse método foi escolhido para servir de comparação para o RP-IMA Dual exatamente por possuir características semelhantes a ele, porém sem se preocupar diretamente com a obtenção de uma margem larga.

A versão dual da ELM implementada aqui, assim como o RP-IMA Dual, aplica duas não linearidades aos dados: a projeção aleatória com função de ativação sigmoidal e a aplicação de um *kernel* (neste trabalho será utilizado um gaussiano).

Assim, como na ELM tradicional, os primeiros passos desse algoritmo (que a partir de agora será chamado de ELM Dual) são gerar aleatoriamente os pesos da camada escondida e então calcular a matriz de mapeamento H, definida na Equação 2.24.

Após esse passo, é então introduzido o cálculo da matriz de *kernel* $\mathbf{K}$. Como será utilizado aqui um *kernel* gaussiano, então os elementos $k_{ij}$ dessa matriz são calculados por: $k_{ij} = \exp\left(-\frac{\|\mathbf{h}_i - \mathbf{h}_j\|}{2\sigma^2}\right)$, sendo $\mathbf{h}_i$ e $\mathbf{h}_j$ vetores linha da matriz $\mathbf{H}$.

Com a introdução dessa matriz K, tem-se agora que a saída da rede é dada por: $\boldsymbol{y} = \boldsymbol{K}^T\boldsymbol{\alpha}$, onde $\boldsymbol{\alpha}$ é o vetor de variáveis duais. Dessa forma, esse vetor pode ser calculado

por:

$$\boldsymbol{\alpha} = (\boldsymbol{K}^T\boldsymbol{K} + \lambda I)^{-1}\boldsymbol{K}^T \cdot \boldsymbol{y} \tag{3.7}$$

Devido a $\boldsymbol{K} \in \mathbb{R}^{nxn}$ ser uma matriz simétrica de posto n, tem-se que:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda I)^{+}\boldsymbol{y}, \tag{3.8}$$

sendo $(\boldsymbol{K} + \lambda I)^{+}$ a pseudo inversa de Moore-Penrose de $(\boldsymbol{K} + \lambda I)$.

Uma vez definido o vetor de multiplicadores $\boldsymbol{\alpha}$, o treinamento está finalizado. A avaliação de uma nova amostra $\boldsymbol{x}^*$ submetida a essa rede é feita como mostrado pela Equação 3.9 a seguir:

$$F(\boldsymbol{x}^*) = \sum_{i=1}^{n} \alpha_i \exp\left(-\frac{\|\mathbf{h}_i - \mathbf{h}^*\|}{2\sigma^2}\right), \tag{3.9}$$

onde $\mathbf{h}^* = g(\boldsymbol{W} \cdot \boldsymbol{x}^* + b)$, sendo $\boldsymbol{W}$ a matriz de pesos randômicos definida durante o treinamento.

# Capítulo 4

# Experimentos e Resultados

O objetivo deste capítulo é apresentar a metodologia dos experimentos realizados neste trabalho e analisar os resultados obtidos a partir de testes com bases de dados sintéticas e reais, com o intuito de avaliar o desempenho dos métodos propostos neste trabalho.

A avaliação dos modelos abrangeu dois aspectos-chave: a acurácia e o tamanho da margem. A acurácia oferece uma análise do desempenho geral dos modelos, enquanto a avaliação do tamanho da margem mensura a eficácia dos algoritmos propostos na otimização da margem na camada de saída.

## 4.1 Resultados do RP-IMA$_2$ Definido em Variáveis Primais

### 4.1.1 Experimento com Bases Sintéticas

Inicialmente foram utilizadas bases sintéticas de 2 dimensões, com o objetivo de se verificar visualmente a superfície de separação gerada por um modelo RP-IMA$_2$ comparada a gerada por um modelo ELM. Quatro *datasets* foram avaliados neste experimento, cada um com 1000 amostras (500 de cada classe).

Para reduzir o efeito da aleatoriedade inerente da ELM e RP-IMA$_2$ na avaliação dos modelos, ambos os métodos utilizaram os mesmos pesos aleatórios dos neurônios ocultos. Ou seja, em cada teste realizado, os pesos ocultos gerados aleatoriamente para a ELM foram replicados para o RP-IMA$_2$. Assim, os dois modelos possuem a mesma matriz H. Para esses testes, foram gerados modelos utilizando 100 neurônios na camada oculta, tendo como função de ativação a tangente hiperbólica.

Os outros hiperparâmetros usados para o RP-IMA$_2$ foram: $\eta = 0.1$ (taxa de aprendizado), $\lambda = 0.1$ (termo constante das variáveis de folga) e $\delta = 10^{-3}$ (parâmetro que

define o incremento mínimo de margem). Também, como critério de parada do algoritmo, foi definido um máximo de 10.000 atualizações do vetor $\boldsymbol{\beta}$ e permitiu-se a execução de no máximo 20 iterações do RP-IMA$_2$. Esses hiperparâmetros foram utilizados para os modelos de todas as bases de dados testadas.

Além do resultado visual da superfície de separação, os modelos também foram avaliados quanto a acurácia e o tamanho da margem de separação obtida. Esses resultados estão expressos em tabelas, que apresentam o valor médio obtido de uma validação cruzada com 10 dobras. Os valores entre parênteses nas tabelas representam os respectivos desvios padrão.

Os valores de tamanho da margem mostrados neste trabalho são na verdade correspondentes à margem geométrica $\gamma$, que é definida por:

$$\gamma = \min_{i=1,\ldots,N}(\gamma_i), \text{tal que } \gamma_i = y_i(\boldsymbol{\beta}^t\mathbf{x}_i + b)/\|\boldsymbol{\beta}\|. \tag{4.1}$$

Este valor foi calculado no espaço de características, ou seja, usando os vetores linha $\mathbf{h}_i$ da matriz $\mathbf{H}$. Desta forma, tem-se:

$$\gamma = \min_{i=1,\ldots,N}(\gamma_i), \text{tal que } \gamma_i = y_i(\boldsymbol{\beta}^t\mathbf{h}_i + b)/\|\boldsymbol{\beta}\|, \tag{4.2}$$

sendo $\boldsymbol{\beta}$ o vetor de pesos da camada de saída da ELM ou RP-IMA$_2$.

A Figura 3 mostra as bordas de decisão obtidas a partir dos dois modelos (ELM à esquerda e RP-IMA$_2$ à direita) para cada um dos conjuntos de dados gerados. Ao se analisar essa figura nota-se que, em vários pontos, os modelos RP-IMA$_2$ resultaram em melhores margens de separação entre as classes. Além disso, enquanto os modelos ELM apresentaram limites de decisão com sinais claros de *overfitting*, os modelos RP-IMA$_2$ produziram fronteiras de separação mais suaves, as quais conseguiram capturar de forma mais precisa os padrões exibidos pelos dados.

A Tabela 1 apresenta os resultados de acurácia e tamanho da margem obtidos para o RP-IMA$_2$ e ELM. Devido à utilização de bases sintéticas simples já era esperado ambos os modelos alcançarem resultados de acurácia estatisticamente similares e próximos de 100%. Assim, o resultado mais importante aqui é observar que, em todos os casos, os modelos RP-IMA$_2$ resultaram em margens de separação significativamente maiores que as geradas por modelos ELM.

A Figura 4 mostra a evolução do valor da margem de acordo com o número de iterações do RP-IMA$_2$. Os valores considerados aqui se referem a uma margem rígida, ou seja, seu valor reflete a distância entre a fronteira de decisão e o dado mais próximo a ela. Uma observação importante é que, embora em problemas linearmente separáveis o IMA sempre produza resultados de margem rígida mais altos a cada iteração, o mesmo não ocorre necessariamente ao se utilizar o algoritmo com esquema de flexibilização de margem.

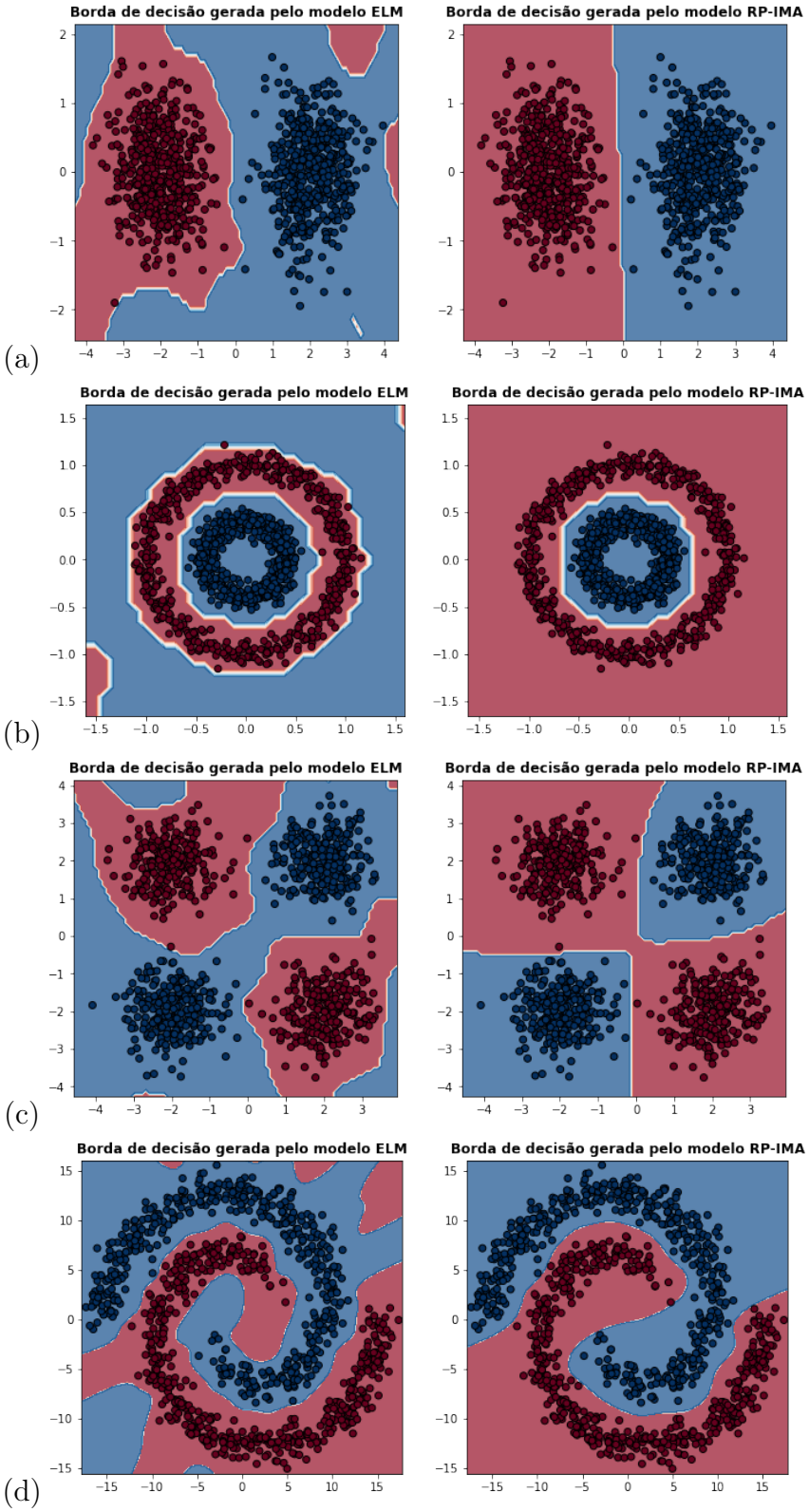


Figura 3 – Superfície de classificação gerada pela ELM (à esquerda) e pelo RP-IMA$_2$ (à direita). *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

Contudo, é possível perceber por esses gráficos uma evolução consistente do tamanho da margem principalmente durante as primeiras iterações.

Tabela 1 – Resultados obtidos para as bases sintéticas.

| Algoritmo | Acurácia | Margem Média |
|---|---|---|
| *Gaussian blobs:* | | |
| ELM | 99.5 (0.5) | $5.0 \cdot 10^{-8}$ ($4.9 \cdot 10^{-8}$) |
| RP-IMA | 100.0 (0.0) | 0.5796 (0.1006) |
| *Xor:* | | |
| ELM | 99.7 (0.6) | $1.7 \cdot 10^{-7}$ ($1.4 \cdot 10^{-7}$) |
| RP-IMA | 99.7 (0.6) | 0.0673 (0.0109) |
| *Circles:* | | |
| ELM | 99.8 (0.6) | $8.3 \cdot 10^{-13}$ ($5.3 \cdot 10^{-13}$) |
| RP-IMA | 100.0 (0.0) | 0.0254 (0.0030) |
| *Spirals:* | | |
| ELM | 99.6 (0.7) | 0.0003 (0.0003) |
| RP-IMA | 99.8 (0.4) | 0.0274 (0.0069) |

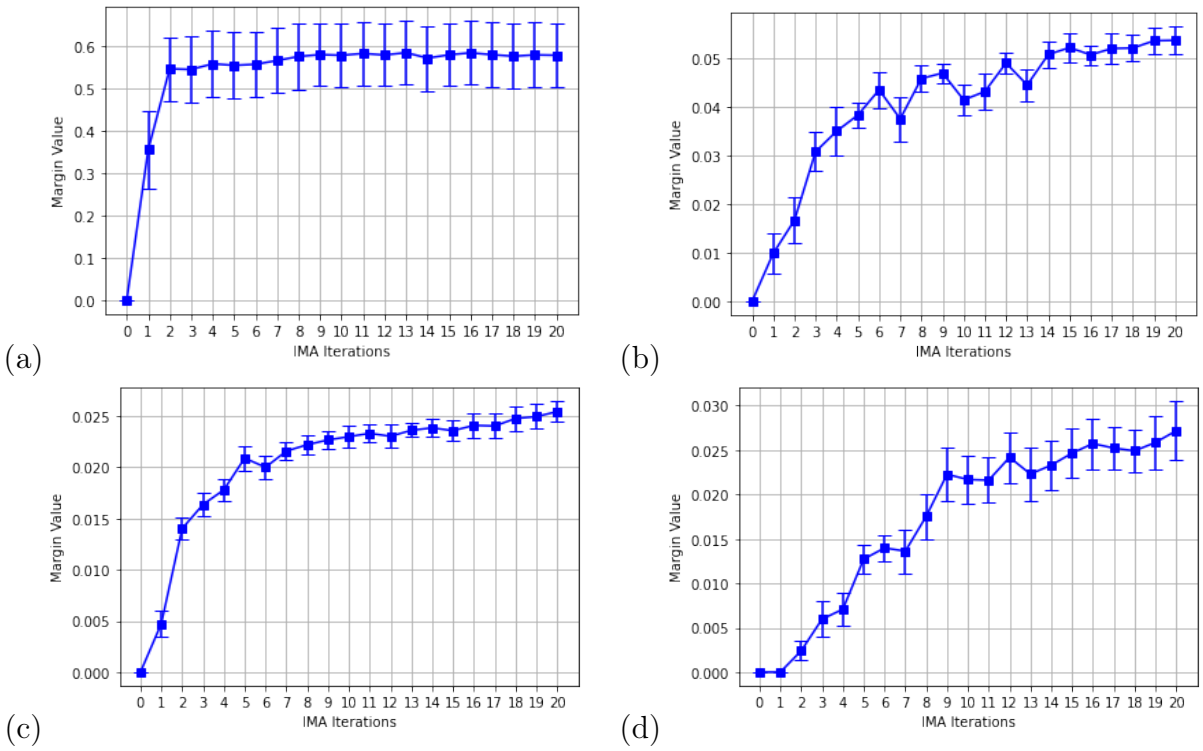


Figura 4 – Margem geométrica calculada a cada iteração do RP-IMA$_2$ para *datasets* gerados sinteticamente. **a** Gaussian blobs. **b** Xor. **c** Circles. **d** Spirals.

### 4.1.2 Experimento com Bases de *Benchmark*

Os conjuntos de dados usados neste experimento foram obtidos do repositório de aprendizado de máquina da UCI [Dua e Graff, 2017], e estão apresentados na Tabela 2. Alguns problemas, como o Iris, foram adaptados para ter apenas duas classes. A Tabela 2 detalha informações básicas sobre os conjuntos de dados utilizados, como o número de características (F), a quantidade de amostras de cada classe e o total de amostras.

Neste experimento, modelos RP-IMA$_2$ foram comparados aos modelos ELM e SVM utilizando *kernel* RBF. Todos os conjuntos de dados foram normalizados para o

Tabela 2 – *Datasets* utilizados nos experimentos.

| *Dataset* | Sigla | F | Amostras | | |
|---|---|---|---|---|---|
| | | | +1 | -1 | Total |
| Iris | IRI | 4 | 50 | 100 | 150 |
| Synthetic | SYN | 60 | 300 | 300 | 600 |
| Robot | ROB | 90 | 24 | 93 | 117 |
| Mushroom | MUS | 22 | 2156 | 3488 | 5644 |
| Ionosphere | ION | 34 | 225 | 126 | 351 |
| Banknote | BAN | 4 | 610 | 762 | 1372 |
| Wine | WIN | 13 | 107 | 71 | 178 |
| WDBC | WBC | 30 | 212 | 357 | 569 |
| Sonar | SON | 60 | 97 | 111 | 208 |
| Spam | SPA | 57 | 1813 | 2788 | 4601 |
| Transfusion | TRA | 4 | 178 | 570 | 748 |
| Breast Cancer | BCA | 9 | 239 | 444 | 683 |
| Australian Cr. | ACR | 14 | 307 | 383 | 690 |
| Haberman | HAB | 3 | 225 | 81 | 306 |
| Heart | HEA | 13 | 150 | 120 | 270 |
| Diabetes | DIA | 8 | 268 | 500 | 768 |
| Mammographic | MAM | 5 | 403 | 427 | 830 |

intervalo [0, 1] antes do treinamento e os hiperparâmetros do RP-IMA$_2$ utilizados foram: $\eta = 0.1$, $\delta = 10^{-3}$ e T_MAX = 10.000 (quantidade máxima de atualizações do vetor de pesos de saída). Além disso, permitiu-se a operação de no máximo 20 execuções de incremento de margem. Para selecionar os hiperparâmetros $\lambda$ do RP-IMA$_2$ e C do SVM (parâmetro de regularização) utilizou-se um *grid search cross-validation*. O coeficiente do *kernel* $\gamma$ do SVM foi definido como $\gamma = 1/(n_{features} \cdot var(X))$, onde $n_{features}$ é o número de características das entradas X e $var(X)$ é a variância de X. Para cada conjunto de dados, testou-se os modelos ELM e RP-IMA$_2$ com três quantidades diferentes de neurônios ocultos, dadas por N, N/2 e N/3, onde N é igual ao total de amostras em cada conjunto de dados, limitado por 1000. A imposição desse limite tem o objetivo de prevenir possíveis problemas relacionados ao consumo de memória. Em todos os experimentos, a tangente hiperbólica foi utilizada como função de ativação dos neurônios ocultos. Os pesos aleatórios dos neurônios da camada oculta selecionados para a ELM e para o RP-IMA foram os mesmos.

Assim como no experimento anterior, os resultados de acurácia e tamanho da margem de separação obtida estão expressos em tabelas, que apresentam o valor médio obtido de uma validação cruzada estratificada (as dobras preservam a porcentagem de amostras para cada classe) com 10 dobras. Os valores entre parênteses nas tabelas representam os respectivos desvios padrão.

A Tabela 3, apresenta os resultados de acurácia. Além disso, a partir dos resultados de acurácia média obtidos por cada algoritmo em cada um dos problemas foi possível gerar o gráfico *boxplot* expresso na Figura 5. Este gráfico foi gerado com base nos resultados médios de acurácia de cada algoritmo na resolução de todos os problemas testados, permitindo assim a observação do comportamento médio dos algoritmos para diferentes problemas.

A partir da Tabela 3 e da Figura 5, é possível observar que, na maioria dos casos, o RP-IMA$_2$ obteve resultados de acurácia média mais altos do que os alcançados pelo ELM convencional. Além disso, em comparação com os resultados alcançados por modelos ELM, nota-se que os resultados obtidos pelos modelos RP-IMA$_2$ são mais similares aos obtidos pelo SVM.

Tabela 3 – Resultados de acurácia obtidos a partir de uma validação cruzada com 10 dobras.

| Set | N | RP-IMA | | | ELM | | | SVM |
|---|---|---|---|---|---|---|---|---|
| | | N nodes | N/2 nodes | N/3 nodes | N nodes | N/2 nodes | N/3 nodes | |
| IRI | 150 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 99.33 (2.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| SYN | 600 | 99.50 (0.76) | 98.83 (1.83) | 98.50 (2.03) | 67.50 (6.72) | 94.59 (3.80) | 97.83 (1.83) | 100.00 (0.00) |
| ROB | 117 | 81.29 (8.85) | 80.38 (5.17) | 79.55 (3.76) | 63.03 (16.48) | 66.52 (11.53) | 68.11 (13.41) | 87.12 (4.05) |
| MUS | 1000 | 99.98 (0.05) | 99.98 (0.05) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 99.72 (0.16) |
| ION | 351 | 94.01 (3.93) | 91.44 (5.58) | 93.15 (4.09) | 73.22 (4.79) | 83.77 (4.57) | 86.61 (6.39) | 95.16 (3.84) |
| BAN | 1000 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 93.80 (2.76) | 99.56 (0.48) | 100.00 (0.00) | 100.00 (0.00) |
| WIN | 178 | 97.22 (3.73) | 97.78 (2.78) | 97.78 (3.69) | 74.67 (11.94) | 96.60 (0.00) | 98.33 (2.55) | 98.33 (3.56) |
| WBC | 569 | 97.36 (1.97) | 97.19 (1.79) | 97.19 (1.79) | 72.40 (5.99) | 92.79 (3.66) | 93.85 (3.26) | 98.42 (2.14) |
| SON | 208 | 69.29 (14.36) | 70.31 (13.89) | 68.83 (15.30) | 61.71 (17.37) | 64.00 (13.10) | 64.98 (16.55) | 69.26 (13.47) |
| SPA | 1000 | 93.47 (0.93) | 92.33 (3.09) | 91.83 (2.72) | 89.55 (1.89) | 91.09 (2.94) | 91.23 (2.55) | 93.15 (2.17) |
| TRA | 748 | 76.74 (2.53) | 77.27 (2.39) | 77.41 (2.50) | 57.90 (10.35) | 58.17 (9.85) | 60.71 (10.45) | 76.47 (0.84) |
| BCR | 683 | 96.50 (2.53) | 96.35 (2.78) | 96.35 (3.06) | 86.38 (2.85) | 79.94 (2.93) | 88.89 (5.10) | 97.23 (2.20) |
| ACR | 690 | 84.49 (3.84) | 85.94 (3.04) | 85.65 (4.65) | 52.61 (5.54) | 76.81 (6.48) | 81.16 (5.46) | 86.38 (3.50) |
| HAB | 306 | 74.84 (2.87) | 74.51 (3.20) | 73.96 (2.43) | 52.61 (6.48) | 58.23 (17.16) | 69.31 (8.18) | 75.51 (2.49) |
| HEA | 270 | 79.26 (5.79) | 81.11 (5.84) | 82.22 (6.37) | 57.78 (9.40) | 69.63 (6.37) | 78.52 (6.15) | 83.33 (5.04) |
| DIA | 768 | 77.61 (2.37) | 74.87 (3.31) | 75.78 (3.62) | 55.05 (4.95) | 67.07 (4.77) | 72.27 (3.72) | 77.47 (3.74) |
| MAM | 830 | 81.69 (5.25) | 81.93 (5.36) | 80.48 (6.48) | 72.17 (4.89) | 71.81 (3.54) | 73.13 (3.77) | 81.45 (4.64) |

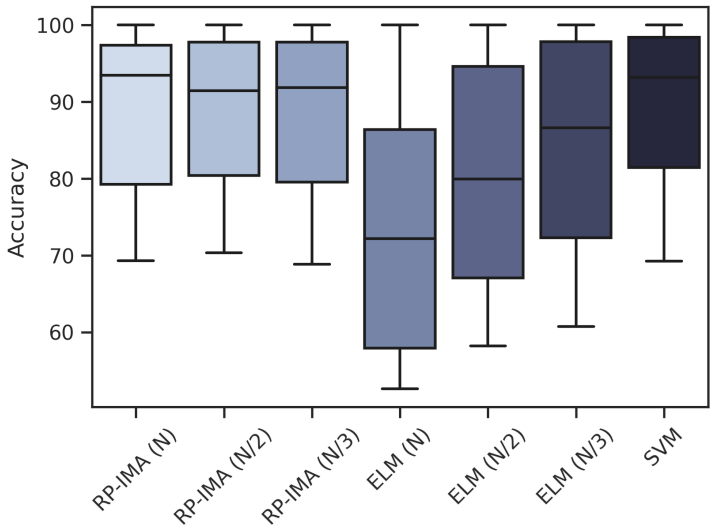


Figura 5 – Gráfico *boxplot* do comportamento médio de acurácia dos algoritmos na resolução de todos os problemas.

A partir da Figura 5, também é possível notar que em comparação com modelos ELM, os modelos RP-IMA$_2$ apresentaram uma menor variação no desempenho de acurácia para os três diferentes números de neurônios ocultos testados. Uma indicação que a ELM é mais suscetível a *overfitting* do que RP-IMA$_2$ é o fato dos modelos ELM com maior número de neurônios apresentarem o pior comportamento geral.

Para quatro conjuntos de dados que resultaram em uma média de acurácia de treinamento de 100%, também foi calculado o valor de margem geométrica no espaço de características. A Tabela 4 apresenta os valores médios obtidos para a margem em uma validação cruzada de 10 dobras. Os valores entre parênteses representam os respectivos desvios padrão. Em todos os quatro conjuntos de dados e considerando qualquer um dos números de neurônios testados, o RP-IMA$_2$ resultou em valores de margem significativamente maiores do que o método ELM convencional.

Tabela 4 – Valores médios de margem obtidos a partir da validação cruzada de 10 dobras.

| *Dataset* | Algoritmo | Neurônios Intermediários | Margem Média |
|---|---|---|---|
| IRI | RP-IMA | 150 | 1.1003 (0.0723) |
| IRI | RP-IMA | 75 | 0.8330 (0.1013) |
| IRI | RP-IMA | 50 | 0.6467 (0.0707) |
| IRI | ELM | 150 | $1.9 \cdot 10^{-5}$ ($4.9 \cdot 10^{-6}$) |
| IRI | ELM | 100 | 0.0002 (0.0001) |
| IRI | ELM | 50 | 0.0008 (0.0002) |
| SYN | RP-IMA | 600 | 0.5442 (0.0507) |
| SYN | RP-IMA | 300 | 0.2327 (0.0481) |
| SYN | RP-IMA | 200 | 0.1179 (0.0476) |
| SYN | ELM | 600 | 0.0123 (0.0019) |
| SYN | ELM | 400 | 0.0020 (0.0017) |
| SYN | ELM | 200 | 0.0012 (0.0014) |
| MUS | RP-IMA | 1000 | 2.1258 (0.0575) |
| MUS | RP-IMA | 500 | 1.2434 (0.1036) |
| MUS | RP-IMA | 333 | 0.8499 (0.0851) |
| MUS | ELM | 1000 | 0.7108 (0.2945) |
| MUS | ELM | 500 | 0.5683 (0.2226) |
| MUS | ELM | 333 | 0.2860 (0.1405) |
| BAN | RP-IMA | 1000 | 0.1413 (0.0053) |
| BAN | RP-IMA | 500 | 0.1008 (0.0043) |
| BAN | RP-IMA | 333 | 0.0812 (0.0055) |
| BAN | ELM | 1000 | $7.2 \cdot 10^{-12}$ ($6.2 \cdot 10^{-6}$) |
| BAN | ELM | 500 | $1.9 \cdot 10^{-9}$ ($3.8 \cdot 10^{-10}$) |
| BAN | ELM | 333 | $2.4 \cdot 10^{-8}$ ($4.3 \cdot 10^{-9}$) |

Para os *datasets* da Tabela 4 buscou-se ainda visualizar a evolução do valor de margem de acordo com o número de iterações do RP-IMA$_2$. Nos gráficos da Figura 6 cada ponto representa o valor médio de margem obtido em cada iteração do RP-IMA$_2$, e as barras de erro representam o erro padrão. Para cada *dataset*, foram testados apenas os modelos da Tabela 4 com maior quantidade de neurônios. Novamente, foi possível observar um crescimento consistente da margem, principalmente durante as primeiras iterações do algoritmo.

## 4.2 Teste de Esparsidade para os Modelos RP-IMA

Em [Villela et al., 2016], foi demonstrado que a formulação $L_\infty$ do IMA gera um vetor $\boldsymbol{\beta}$ com medidas de esparsidade significativamente mais altas do que as outras formulações. Para verificar se isso também é verdade no caso do RP-IMA, as formulações

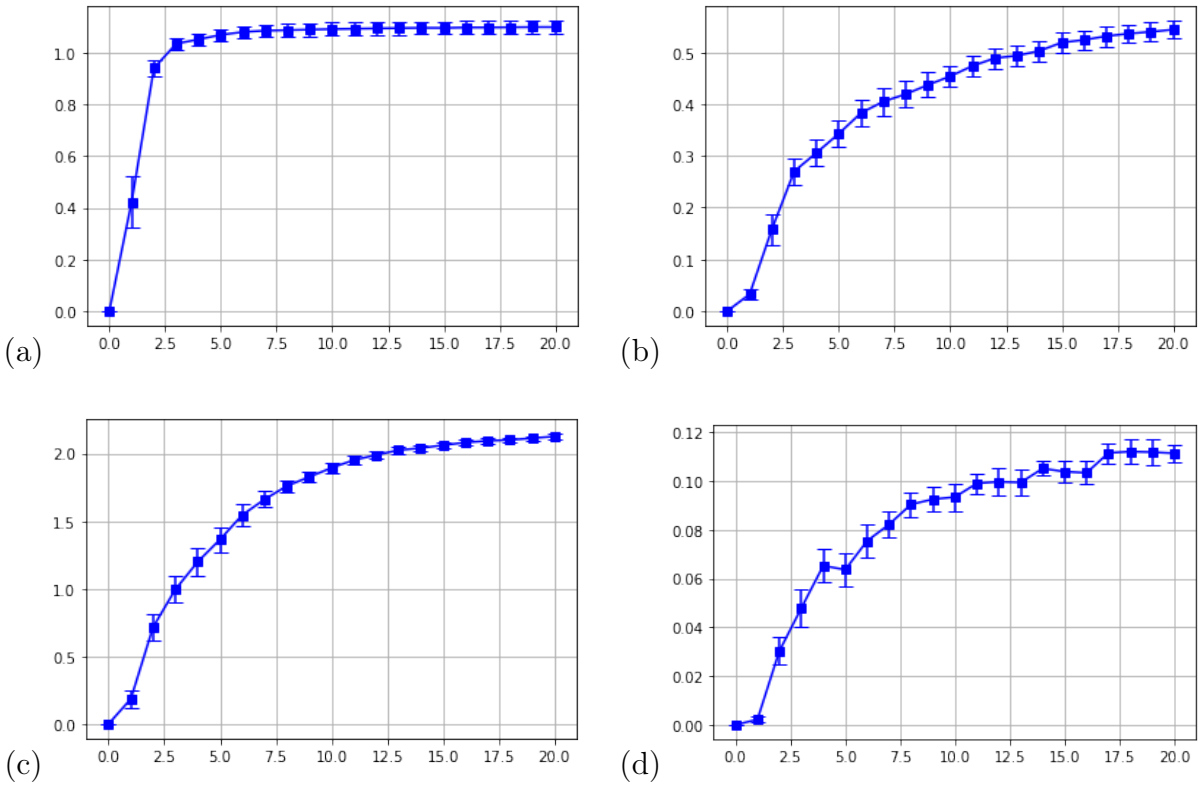


Figura 6 – Margem geométrica calculada a cada iteração do RP-IMA$_2$. **a** Iris. **b** Synthetic control chart time series. **c** Mushroom. **d** Banknote.

$L_1$, $L_\infty$ e $L_2$ do método foram aplicadas novamente aos conjuntos de dados na Tabela 2. Todos os conjuntos de dados foram escalados para o intervalo [0, 1] antes do treinamento.

Para medir a esparsidade de cada solução, foi calculado o número de componentes do vetor $\boldsymbol{\beta}$ que possuem valor absoluto maior do que o limiar $\rho \cdot \max_i |\boldsymbol{\beta}_i|$ [Villela et al., 2016]. Quanto menor o valor obtido, mais esparsa é a solução. O vetor $\boldsymbol{\beta}$ foi normalizado antes dessa medida de esparsidade ser calculada.

A Tabela 5 apresenta a medida de esparsidade (considerando diferentes valores de $\rho$) e os resultados de acurácia obtidos para cada algoritmo. Esses resultados são valores médios obtidos a partir de uma validação cruzada estratificada de 10 dobras. Os valores entre parênteses representam os desvios padrão. Para garantir comparações justas, selecionou-se os mesmos 10 subconjuntos de dados para validação cruzada em todos os algoritmos, para cada conjunto de dados. Além disso, os pesos ocultos gerados aleatoriamente para ELM, RP-IMA$_1$, RP-IMA2 e RP-IMA$\infty$ foram os mesmos. Da mesma forma, o número de neurônios ocultos foi o mesmo para todos os métodos e igual ao número de amostras de cada conjunto de dados (limitado a 1000).

Para todas as formulações do RP-IMA, os hiperparâmetros foram definidos como $\eta = 0.1$, $\delta = 10^{-3}$. O valor do hiperparâmetro $\lambda$ foi selecionado para cada conjunto de dados por meio da técnica *grid search cross-validation*. Como critério de parada foi definido

um máximo de 10000 atualizações do vetor $\boldsymbol{\beta}$ ou 20 iterações do RP-IMA.

Tabela 5 – Resultados de esparsidade.

| *Dataset* | Neurônios | Método | Medida de Esparsidade ($\rho \cdot 100\%$) | | | | Acurácia (%) |
|---|---|---|---|---|---|---|---|
| | | | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ | |
| IRI | 150 | RP-IMA$_1$ | 145.1 (1.1) | 147.4 (1.6) | 149.8 (0.60) | 150.0 (0.0) | 100.00 (0.00) |
| | | RP-IMA$_2$ | 97.1 (4.9) | 121.9 (4.1) | 147.6 (1.5) | 149.5 (0.8) | 100.00 (0.00) |
| | | RP-IMA$_\infty$ | 10.5 (4.4) | 13.0 (5.7) | 24.0 (14.8) | 106.8 (27.8) | 100.00 (0.00) |
| SYN | 600 | RP-IMA$_1$ | 532.5 (15.7) | 566.9 (8.1) | 595.9 (2.1) | 600.0 (0.0) | 99.50 (1.07) |
| | | RP-IMA$_2$ | 253.3 (49.1) | 417.2 (36.1) | 580.5 (4.6) | 597.8 (1.2) | 99.50 (1.07) |
| | | RP-IMA$_\infty$ | 76.4 (18.1) | 107.5 (17.5) | 184.1 (30.8) | 520.1 (20.8) | 99.67 (0.67) |
| ROB | 117 | RP-IMA$_1$ | 101.4 (12.2) | 109.5 (5.3) | 116.3 (0.9) | 116.9 (0.3) | 81.14 (7.45) |
| | | RP-IMA$_2$ | 54.7 (15.9) | 91.6 (7.5) | 113.9 (1.8) | 116.8 (0.6) | 82.05 (6.90) |
| | | RP-IMA$_\infty$ | 13.4 (11.0) | 27.0 (19.9) | 97.2 (9.3) | 115.0 (1.3) | 76.89 (5.70) |
| MUS | 1000 | RP-IMA$_1$ | 876.5 (18.0) | 936.7 (10.7) | 992.4 (2.3) | 999.3 (0.9) | 100.00 (0.00) |
| | | RP-IMA$_2$ | 450.8 (57.6) | 696.8 (35.2) | 969.1 (5.3) | 996.8 (2.1) | 100.00 (0.00) |
| | | RP-IMA$_\infty$ | 77.6 (16.7) | 116.3 (19.3) | 196.5 (35.2) | 779.5 (87.1) | 100.00 (0.00) |
| ION | 351 | RP-IMA$_1$ | 286.7 (15.6) | 319.0 (8.8) | 348.0 (1.7) | 350.7 (0.4) | 90.87 (5.84) |
| | | RP-IMA$_2$ | 205.5 (23.0) | 277.3 (15.7) | 345.2 (4.2) | 350.1 (0.9) | 91.16 (6.06) |
| | | RP-IMA$_\infty$ | 175.5 (23.2) | 245.3 (17.4) | 332.5 (6.0) | 348.8 (1.5) | 90.58 (6.89) |
| BAN | 1000 | RP-IMA$_1$ | 846.5 (20.8) | 924.1 (10.5) | 992.4 (3.0) | 999.10 (1.1) | 99.64 (0.59) |
| | | RP-IMA$_2$ | 376.6 (87.7) | 669.6 (57.5) | 965.9 (7.8) | 996.5 (1.2) | 100.00 (0.00) |
| | | RP-IMA$_\infty$ | 194.5 (46.7) | 359.0 (51.2) | 535.9 (73.2) | 717.7 (97.0) | 100.00 (0.00) |
| WIN | 178 | RP-IMA$_1$ | 141.3 (8.3) | 159.2 (4.4) | 175.6 (1.7) | 177.6 (0.5) | 98.30 (2.60) |
| | | RP-IMA$_2$ | 100.4 (8.7) | 136.9 (4.3) | 173.4 (1.9) | 177.5 (0.5) | 97.75 (2.76) |
| | | RP-IMA$_\infty$ | 74.1 (14.5) | 103.9 (16.1) | 162.2 (7.1) | 176.1 (1.6) | 98.30 (2.60) |
| WBC | 569 | RP-IMA$_1$ | 489.4 (13.8) | 529.9 (7.6) | 565.2 (2.0) | 568.2 (1.0) | 96.49 (2.35) |
| | | RP-IMA$_2$ | 307.6 (35.3) | 435.3 (22.5) | 556.8 (3.4) | 567.6 (1.2) | 96.84 (1.31) |
| | | RP-IMA$_\infty$ | 150.5 (43.0) | 218.2 (51.5) | 410.3 (62.8) | 547.3 (13.8) | 97.36 (1.62) |
| SON | 208 | RP-IMA$_1$ | 156.7 (9.9) | 182.7 (6.6) | 205.4 (1.8) | 207.8 (0.4) | 81.62 (9.35) |
| | | RP-IMA$_2$ | 118.3 (12.1) | 163.0 (8.8) | 203.3 (2.6) | 207.1 (0.9) | 80.64 (9.93) |
| | | RP-IMA$_\infty$ | 105.4 (15.2) | 144.4 (11.5) | 197.0 (3.9) | 207.0 (0.6) | 78.26 (8.43) |
| SPA | 1000 | RP-IMA$_1$ | 182.2 (279.8) | 240.5 (367.8) | 293.7 (448.6) | 299.3 (457.2) | 68.79 (12.60) |
| | | RP-IMA$_2$ | 131.2 (201.6) | 206.6 (315.8) | 290.8 (444.2) | 299.1 (456.9) | 70.33 (14.87) |
| | | RP-IMA$_\infty$ | 125.1 (192.5) | 198.5 (303.6) | 287.2 (438.7) | 299.0 (456.7) | 70.10 (14.51) |
| TRA | 748 | RP-IMA$_1$ | 572.3 (53.4) | 659.1 (29.2) | 739.8 (4.8) | 747.5 (0.7) | 76.61 (2.60) |
| | | RP-IMA$_2$ | 227.7 (56.1) | 426.8 (50.1) | 714.3 (7.5) | 744.0 (1.8) | 76.61 (1.31) |
| | | RP-IMA$_\infty$ | 178.4 (187.4) | 273.6 (256.8) | 602.1 (144.7) | 731.2 (16.6) | 76.21 (0.90) |
| BCR | 683 | RP-IMA$_1$ | 591.4 (28.3) | 638.9 (16.6) | 679.3 (2.6) | 682.7 (0.5) | 97.22 (2.50) |
| | | RP-IMA$_2$ | 262.9 (79.3) | 438.0 (66.8) | 659.0 (8.1) | 680.3 (1.6) | 97.66 (1.88) |
| | | RP-IMA$_\infty$ | 48.6 (18.6) | 82.6 (31.8) | 356.5 (123.5) | 639.5 (22.0) | 97.07 (1.74) |
| ACR | 690 | RP-IMA$_1$ | 675.0 (61.7) | 831.7 (30.6) | 984.2 (4.1) | 998.5 (1.4) | 86.52 (3.37) |
| | | RP-IMA$_2$ | 433.3 (50.5) | 692.3 (30.1) | 968.6 (8.0) | 997.2 (1.5) | 86.09 (2.69) |
| | | RP-IMA$_\infty$ | 352.4 (38.9) | 580.9 (33.4) | 918.2 (18.5) | 991.6 (3.0) | 85.36 (2.46) |
| HAB | 306 | RP-IMA$_1$ | 230.3 (18.8) | 265.9 (13.7) | 302.4 (2.1) | 305.9 (0.3) | 72.19 (4.91) |
| | | RP-IMA$_2$ | 152.9 (29.9) | 225.6 (23.6) | 297.6 (3.2) | 297.6 (3.2) | 74.51 (2.83) |
| | | RP-IMA$_\infty$ | 154.4 (25.9) | 224.0 (17.9) | 297.7 (3.4) | 305.7 (0.5) | 73.53 (0.94) |
| HEA | 270 | RP-IMA$_1$ | 226.4 (8.3) | 249.8 (5.2) | 268.7 (1.1) | 269.9 (0.3) | 77.78 (5.97) |
| | | RP-IMA$_2$ | 130.4 (16.4) | 196.1 (8.7) | 260.8 (2.8) | 269.4 (0.7) | 79.63 (7.27) |
| | | RP-IMA$_\infty$ | 76.7 (13.0) | 122.2 (13.9) | 235.9 (12.8) | 265.9 (2.4) | 78.52 (8.25) |
| DIA | 768 | RP-IMA$_1$ | 556.8 (97.5) | 659.2 (52.6) | 756.2 (5.0) | 767.6 (0.5) | 71.62 (4.16) |
| | | RP-IMA$_2$ | 234.6 (45.6) | 440.8 (43.6) | 736.4 (4.4) | 764.2 (1.5) | 73.70 (2.51) |
| | | RP-IMA$_\infty$ | 138.6 (35.7) | 251.8 (38.3) | 612.0 (34.4) | 750.0 (6.8) | 75.13 (2.86) |
| MAM | 830 | RP-IMA$_1$ | 660.2 (27.9) | 744.2 (17.3) | 821.0 (3.0) | 828.9 (1.0) | 80.48 (5.67) |
| | | RP-IMA$_2$ | 496.1 (36.2) | 661.9 (20.3) | 812.6 (5.4) | 828.4 (1.1) | 83.25 (5.18) |
| | | RP-IMA$_\infty$ | 375.1 (49.60) | 510.3 (48.5) | 719.1 (30.6) | 819.5 (3.9) | 82.29 (4.57) |

A partir da Tabela 5, é possível notar que, embora os métodos apresentem resultados de acurácia muitas vezes semelhantes, o RP-IMA$_\infty$ resultou em soluções significativamente mais esparsas para o vetor $\boldsymbol{\beta}$. Isso sugere que um método de poda na arquitetura do RP-IMA$_\infty$ pode ser efetivo para durante o treinamento realizar a remoção de um alto número de conexões menos relevantes da rede.

A Figura 7 apresenta a comparação da esparsidade entre RP-IMA$_1$, RP-IMA$_2$ e RP-IMA$_\infty$. Essa figura mostra as magnitudes normalizadas de todos os pesos da rede de cada método para quatro conjuntos de dados diferentes. Como pode ser observado, a esparsidade é maior nas soluções do RP-IMA$_\infty$, uma vez que resultou em mais pesos com magnitudes nulas (ou muito pequenas).

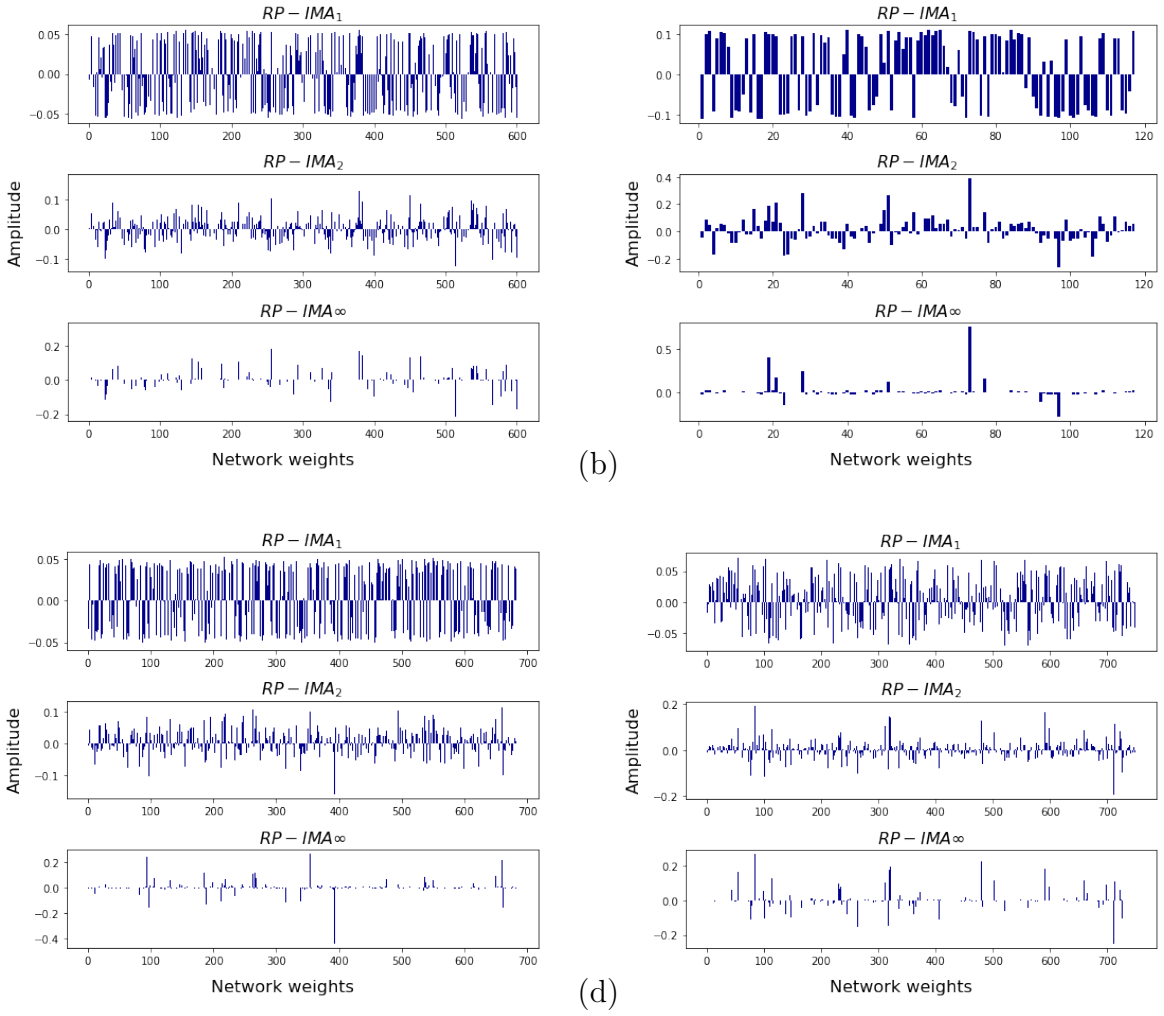


Figura 7 – Comparação da esparsidade entre as soluções de rede RP-IMA$_1$, RP-IMA$_2$ e RP-IMA$_\infty$. **a** Synthetic control chart time series. **b** Robot **c** Breast. **d** Transfusion.

A capacidade do RP-IMA$_\infty$ de gerar soluções mais esparsas é significativa em uma variedade de cenários. Por exemplo, modelos esparsos são altamente interpretáveis, facilitando a compreensão do impacto de cada variável nas previsões do modelo. Além disso, como mencionado anteriormente, essa característica possibilita a poda de neurônios,

reduzindo o número de operações computacionais necessárias e resultando em economia de recursos de armazenamento.

## 4.3 Resultados do Modelo RP-IMA$_\infty$ com Método de Poda

Na seção anterior, observamos que o RP-IMA$_\infty$ gera um vetor de pesos de saída $\boldsymbol{\beta}$ mais esparso do que as outras formulações. Assim, esse método foi integrado com a estratégia de poda de neurônios apresentada na seção 3.3.

Para avaliar este algoritmo, aplicamos o RP-IMA$_\infty$ com o método de poda aos conjuntos de dados da Tabela 2. A mesma metodologia e hiperparâmetros dos testes anteriores foi aplicada. O número de neurônios ocultos selecionados e as acurácias obtidas para cada conjunto de dados são apresentados na Tabela 6 e na Tabela 7, respectivamente.

Tabela 6 – Número de neurônios selecionados pelos modelos RP-IMA$_\infty$ com método de poda.

| Set | Neurônios iniciais | Número final de neurônios do RP-IMA$_\infty$ com poda | | | |
|---|---|---|---|---|---|
| | | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ |
| IRI | 150 | 6,1 (2.17) | 12.3 (5.3) | 23.3 (10.2) | 62.4 (14.8) |
| SYN | 600 | 91.8 (17.8) | 167.4 (33.5) | 277.1 (29.6) | 425.6 (33.9) |
| ROB | 117 | 11.6 (7.2) | 23.2 (20.9) | 63.4 (19.5) | 104.7 (10.5) |
| MUS | 1000 | 85.1 (14.8) | 162.5 (19.9) | 250.4 (30.3) | 490.6 (55.0) |
| ION | 351 | 109.7 (15.5) | 170.4 (16.9) | 267.8 (13.0) | 330.8 (5.1) |
| BAN | 1000 | 168.7 (35.9) | 358.9 (136.0) | 554.1 (144.2) | 671.3 (167.3) |
| WIN | 178 | 55.0 (6.7) | 78.9 (6.0) | 113.8 (13.9) | 153.0 (5.5) |
| WBC | 569 | 111.8 (15.0) | 171.2 (28.5) | 268.5 (36.2) | 420.0 (43.8) |
| SON | 208 | 68.7 (17.7) | 104.7 (11.8) | 149.1 (10.7) | 178.2 (9.5) |
| SPA | 1000 | 188.2 (26.2) | 380.2 (19.6) | 752.1 (17.1) | 940.8 (4.3) |
| TRA | 748 | 9.4 (6.7) | 24.6 (22.6) | 419.5 (257.2) | 641.7 (131.7) |
| BCR | 683 | 13.2 (7.2) | 18.9 (7.9) | 97.3 (34.8) | 445.7 (33.6) |
| ACR | 690 | 93.0 (31.2) | 179.0 (57.7) | 399.8 (69.6) | 598.0 (27.2) |
| HAB | 306 | 20.6 (21.2) | 39.8 (40.8) | 181.4 (40.4) | 286.0 (16.5) |
| HEA | 270 | 29.7 (9.1) | 58.2 (16.8) | 142.1 (25.3) | 230.2 (11.7) |
| DIA | 768 | 94.6 (15.1) | 162.9 (25.3) | 386.1 (75.1) | 604.3 (42.4) |
| MAM | 830 | 261.1 (48.0) | 382.2 (52.5) | 541.7 (43.6) | 716.5 (27.9) |

A Tabela 6 mostra que o método de poda proposto gera arquiteturas de rede com uma quantidade significativamente menor de neurônios na camada oculta, em comparação com o RP-IMA sem poda.

Além disso, a partir da Tabela 7, foi possível observar que RP-IMA$_\infty$ com método de poda alcançar um desempenho médio de acurácia similar ao do RP-IMA$_2$. Isso é mais evidente no gráfico da Figura 8. Esta figura apresenta, por meio de um gráfico de *boxplot*, o comportamento médio de acurácia do método RP-IMA$_\infty$ com poda em comparação com o ELM e o RP-IMA$_2$ na resolução de todos os problemas. Observando a Figura 8, também é possível notar que, em comparação com o ELM, o RP-IMA$_\infty$ com método de poda apresenta melhor desempenho médio de acurácia para todos os valores de $\rho$ testados.

Tabela 7 – Resultados de acurácia dos modelos RP-IMA$_\infty$, ELM and RP-IMA$_2$.

| *Dataset* | RP-IMA$_\infty$ com poda | | | | ELM | RP-IMA$_2$ |
|---|---|---|---|---|---|---|
| | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ | | |
| IRI | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 98.67 (2.67) | 100.00 (0.00) |
| SYN | 99.50 (0.76) | 99.67 (0.67) | 99.67 (0.67) | 99.67 (0.67) | 81.50 (6.77) | 99.67 (0.67) |
| ROB | 85.53 (8.49) | 85.53 (5.81) | 83.79 (5.81) | 84.70 (9.05) | 57.80 (20.71) | 83.71 (7.01) |
| MUS | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| ION | 90.31 (5.75) | 90.30 (6.55) | 89.45 (5.73) | 91.44 (7.48) | 72.62 (8.46) | 90.30 (6.30) |
| BAN | 99.64 (0.88) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 96.36 (1.56) | 99.78 (0.66) |
| WIN | 97.75 (2.76) | 96.63 (2.75) | 96.63 (2.75) | 97.19 (2.81) | 76.83 (7.86) | 98.30 (2.60) |
| WBC | 95.96 (2.22) | 97.19 (1.95) | 96.66 (2.28) | 96.84 (2.04) | 69.44 (6.98) | 96.48 (2.08) |
| SON | 79.14 (10.48) | 82.52 (11.08) | 82.05 (9.89) | 82.07 (10.77) | 70.69 (4.37) | 81.60 (8.43) |
| SPA | 91.07 (2.06) | 92.59 (1,09) | 91.13 (1.96) | 91.85 (1.15) | 90.76 (0.73) | 91.35 (1.37) |
| TRA | 74.75 (9.16) | 77.14 (1.87) | 78.08 (2.11) | 77.02 (2.65) | 69.77 (4.89) | 77.28 (1.49) |
| BCR | 96.92 (1.67) | 96.92 (1.91) | 97.07 (1.97) | 96.92 (1.80) | 85.20 (3.77) | 97.51 (2.28) |
| ACR | 84.93 (2.44) | 85.65 (2.71) | 85.65 (3.14) | 85.65 (3.40) | 52.90 (4.31) | 85.51 (3.24) |
| HAB | 73.20 (1.18) | 73.19 (1.42) | 73.53 (0.94) | 73.20 (1.18) | 66.66 (6.21) | 72.55 (2.96) |
| HEA | 83.33 (6.47) | 83.33 (7.81) | 81.48 (8.11) | 82.96 (8.48) | 60.37 (9.23) | 81.85 (8.68) |
| DIA | 76.96 (5.17) | 75.38 (5.46) | 76.95 (5.06) | 75.65 (4.41) | 55.98 (3.81) | 76.17 (4.53) |
| MAM | 81.93 (5.98) | 80.84 (5.01) | 82.53 (4.97) | 82.89 (4.53) | 73.61 (4.22) | 82.77 (4.48) |

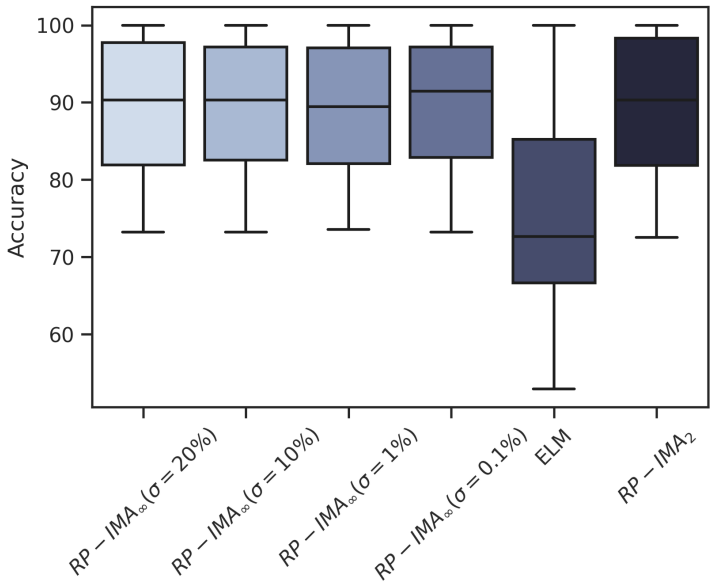


Figura 8 – Gráfico *boxplot* do comportamento médio de acurácia dos algoritmos RP-IMA$_\infty$ com método de poda, ELM e RP-IMA$_2$ na resolução de todos os problemas.

## 4.4 Resultados do Modelo IM-RBFNN$_2$

Assim como no caso do RP-IMA primal, o método IM-RBFNN será testado primeiramente considerando sua versão que busca um hiperplano separador com margem $L_2$.

### 4.4.1 Experimento com Bases Sintéticas

Utilizando as mesmas bases sintéticas de 2 dimensões descritas na seção 4.1, primeiramente buscou-se verificar visualmente as superfícies de separação geradas pelos

métodos RBFNN tradicional e IM-RBFNN$_2$.

Para esses testes, foram gerados modelos utilizando 100 neurônios na camada oculta, tendo como função de ativação a função gaussiana. Para cada neurônio o centro da função RBF foi definido como o centroide do respectivo agrupamento encontrado pelo algoritmo *k-means* (com $k = 100$). Já o parâmetro que controla a largura da gaussiana foi definido como a média das distâncias euclidianas das amostras até o centro da função radial implementada pelo neurônio.

Já os hiperparâmetros específicos do IM-RBFNN$_2$ foram fixados em: $\eta = 0.1$, $\lambda = 0.1$ e $\delta = 10^{-3}$. Também, como critério de parada do algoritmo, foi definido um máximo de 10.000 atualizações do vetor $\boldsymbol{\beta}$ e permitiu-se a execução de no máximo 20 iterações de incremento de margem.

Além do resultado visual da superfície de separação, os modelos também foram avaliados quanto a acurácia e a margem geométrica de separação obtida (essa margem foi medida com relação ao espaço de características). Esses resultados estão expressos em tabelas, que apresentam o valor médio obtido de uma validação cruzada com 10 dobras. Os valores entre parênteses nas tabelas representam os respectivos desvios padrão.

A Figura 9 mostra as bordas de decisão obtidas a partir dos dois modelos (RBFNN à esquerda e IM-RBFNN$_2$ à direita) para cada um dos conjuntos de dados gerados. Nessa figura é possível notar que o efeito de *overfitting* está mais presente nos modelos RBFNN e que para a maioria das bases sintéticas os modelos IM-RBFNN$_2$ geraram superfícies de separação mais suavizadas. Contudo, é importante observar que, no teste realizado com a base de dados sintética "*Spirals*", essa suavização levou o modelo IM-RBFNN$_2$ a não conseguir capturar determinados padrões nos dados de treinamento, resultando na classificação incorreta de algumas amostras.

A Tabela 8 apresenta os resultados de acurácia e valor da margem geométrica obtidos para os modelos RBFNN e IM-RBFNN$_2$. Percebe-se que os modelos obtiveram resultados de acurácia estatisticamente similares e próximos de 100%, isso era esperado visto que se tratam de problemas simples e sem sobreposição de instancias de diferentes classes.

Com relação a margem de separação, para a maioria dos *datasets* nota-se que os modelos IM-RBFNN$_2$ obtiveram resultados médios melhores que os obtidos pelos modelos RBFNN. A exceção fica para a base "*Spirals*", onde o modelo IM-RBFNN$_2$ com os parâmetros definidos anteriormente não conseguiu classificar corretamente todos os dados de treinamento e por isso a Tabela 8 apresenta seu valor de margem zerado. Esse resultado se deve ao fato de que a margem calculada aqui é uma margem rígida (normalmente chamada de margem *hard*), que não leva em conta nenhuma flexibilidade nas fronteiras de decisão e exige que todas as instâncias de treinamento sejam classificadas corretamente.

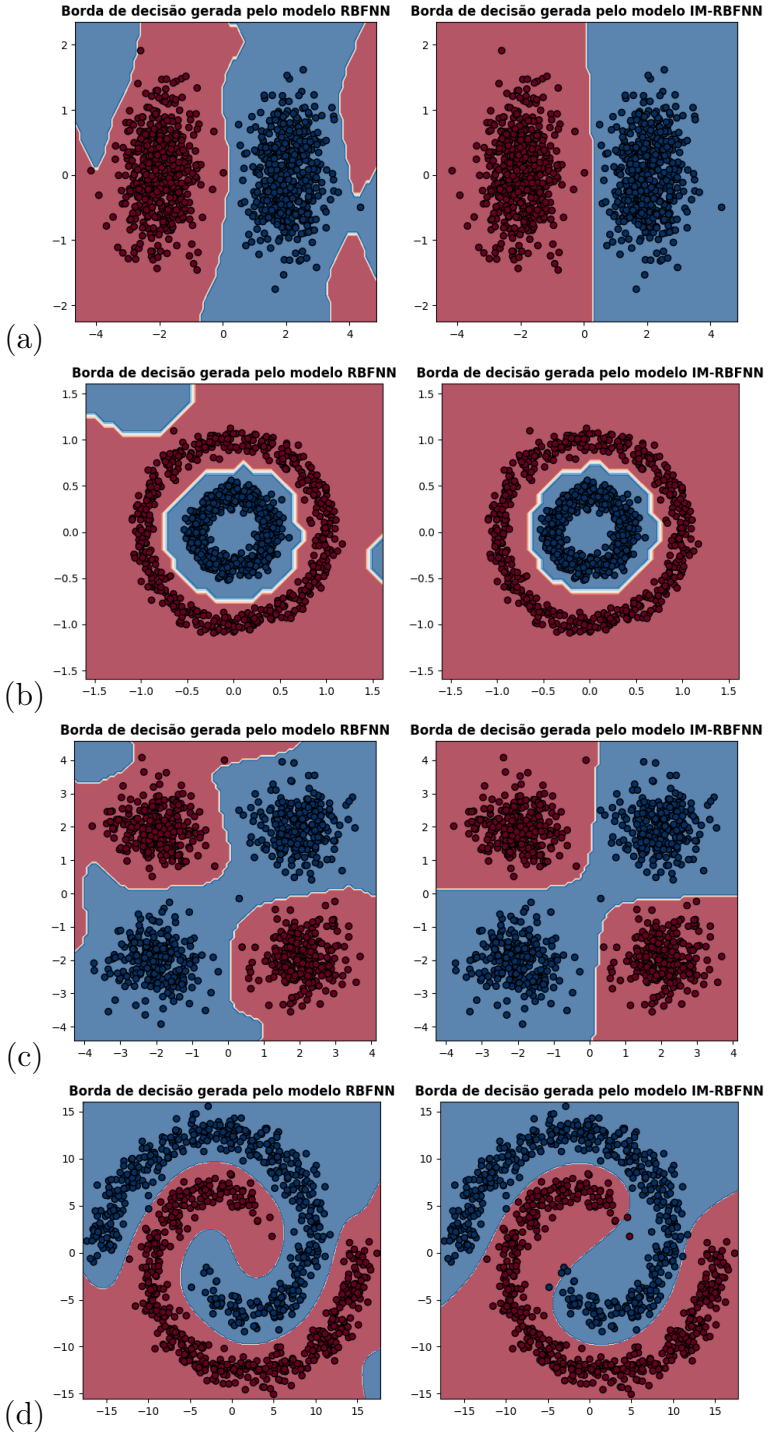


Figura 9 – Superfície de classificação gerada pela RBFNN (à esquerda) e pelo IM-RBFNN (à direita). *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

Para as três primeiras bases mostradas na Tabela 8 (onde os modelos obtiveram 100% de acurácia de treinamento), a Figura 4 mostra a evolução do valor da margem geométrica (calculada no espaço de características) de acordo com o número de iterações do IM-RBFNN$_2$. Destaca-se aqui que apesar de se utilizar modelos IM-RBFNN$_2$ com flexibilização de margem, os resultados mostrados nos gráficos da Figura 10 foram medidos considerando uma margem rígida. Assim, apesar do valor de margem fixa sempre aumentar

Tabela 8 – Resultados obtidos para as bases sintéticas.

| Algoritmo | Acurácia | Margem Média |
|---|---|---|
| *Gaussian blobs:* | | |
| RBFNN | 99.4 (0.7) | $3.7 \cdot 10^{-13}$ $(2.6 \cdot 10^{-13})$ |
| IM-RBFNN | 100.0 (0.0) | $4.5 \cdot 10^{-1}$ $(4.7 \cdot 10^{-2})$ |
| *Xor:* | | |
| RBFNN | 99.9 (0.3) | $6.0 \cdot 10^{-13}$ $(3.7 \cdot 10^{-13})$ |
| IM-RBFNN | 100.0 (0.0) | $2.7 \cdot 10^{-2}$ $(1.3 \cdot 10^{-2})$ |
| *Circles:* | | |
| RBFNN | 100.0 (0.0) | $84 \cdot 10^{-13}$ $(3.0 \cdot 10^{-13})$ |
| IM-RBFNN | 100.0 (0.0) | $2.9 \cdot 10^{-1}$ $(3.3 \cdot 10^{-2})$ |
| *Spirals:* | | |
| RBFNN | 99.9 (0.3) | $1.04 \cdot 10^{-14}$ $(3.1 \cdot 10^{-14})$ |
| IM-RBFNN | 99.0 (0.8) | 0.0 (0.0) |

a cada iteração do IM-RBFNN, pode acontecer do valor de margem *hard* diminuir de uma iteração para outra devido a flexibilização. Contudo, é possível perceber por esses gráficos uma evolução consistente do tamanho da margem, especialmente nas iterações iniciais do algoritmo.

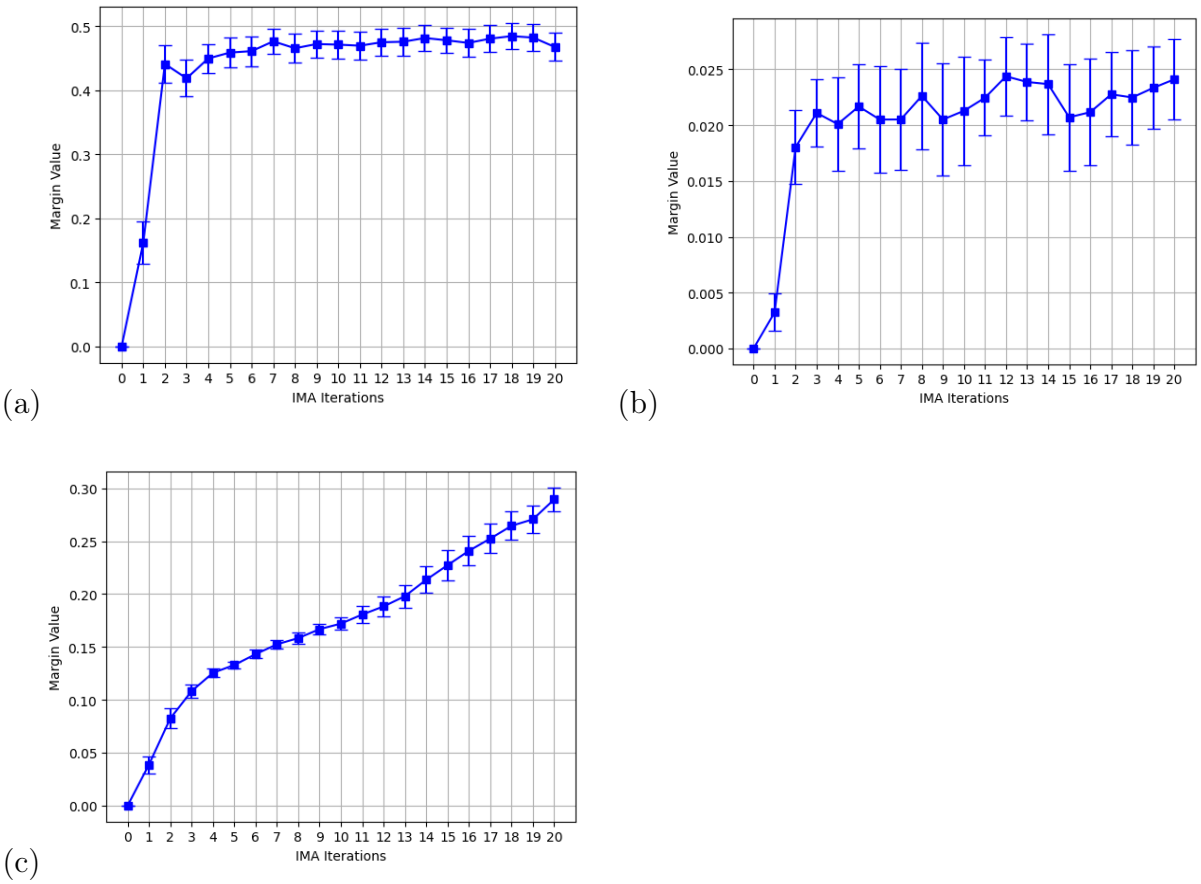


Figura 10 – Margem geométrica calculada a cada iteração do IM-RBFNN para *datasets* gerados sinteticamente. **a** *Gaussian blobs*. **b** *Xor*. **c** *Circles*.

### 4.4.2 Experimento com Bases de *Benchmark*

Novamente, os *datasets* utilizados neste experimento foram aqueles descritos na Tabela 2. Aqui foram comparados os métodos IM-RBFNN$_2$, RBFNN e SVM com *kernel* RBF. Todos os conjuntos de dados foram normalizados para o intervalo [0, 1] antes do treinamento e os hiperparâmetros do IM-RBFNN$_2$ utilizados foram: $\eta = 0.1$, $\delta = 10^{-3}$ e T_MAX = 10.000. Além disso, permitiu-se a operação de no máximo 20 incrementos de margem. Para selecionar os hiperparâmetros $\lambda$ do IM-RBFNN$_2$ e C do SVM (parâmetro de regularização) utilizou-se a técnica *grid search cross-validation*. O coeficiente do *kernel* $\gamma$ do SVM foi definido como $\gamma = 1/(n_{features} \cdot var(X))$, onde $n_{features}$ é o número de características da base de treinamento X e $var(X)$ é a variância de X.

Com objetivo de avaliar o desempenho dos algoritmos para diferentes arquiteturas de rede, testou-se os métodos RBFNN e IM-RBFNN$_2$ com três quantidades diferentes de neurônios ocultos, dadas por N/3, N/5 e N/7, onde N é igual ao total de amostras em cada conjunto de dados, limitado por 1000. Um detalhe crucial a ser considerado é que, dado que esses métodos empregaram um algoritmo de agrupamento para determinar os parâmetros dos neurônios na camada oculta, é imperativo escolher um número de neurônios que evite a formação de grupos vazios ao realizar o agrupamento. Por esse motivo, as quantidades de neurônios da camada escondida testadas aqui foram menores que as testadas na seção 4.1.

Os resultados de acurácia e tamanho da margem de separação obtida estão expressos em tabelas, que apresentam o valor médio obtido de uma validação cruzada estratificada (as dobras preservam a porcentagem de amostras para cada classe) com 10 dobras. Os valores entre parênteses nas tabelas representam os respectivos desvios padrão. A cada iteração da validação cruzada, para evitar os efeitos da aleatoriedade advinda do *k-means* na avaliação dos modelos, os valores de centros e larguras das gaussianas foram os mesmos definidos para os neurônios da camada oculta dos modelos RBFNN e IM-RBFNN$_2$.

A Tabela 9, apresenta os resultados de acurácia. Além disso, a partir dos resultados de acurácia média obtidos por cada algoritmo em cada um dos problemas foi possível gerar o gráfico *boxplot* expresso na Figura 11. O objetivo da geração desse gráfico foi conseguir observar o comportamento médio dos algoritmos na resolução de todos os problemas estudados neste trabalho.

Com base nos dados apresentados na Tabela 9, é evidente que, na maioria dos problemas, os modelos IM-RBFNN$_2$ superaram os modelos RBFNN em termos de acurácia média, quando se compara o desempenho entre arquiteturas com o mesmo número de neurônios.

Observando o gráfico 11 nota-se que todos os 3 modelos IM-RBFNN$_2$ apresentam maior mediana de acurácia que os modelos RBFNN, o que pode indicar uma tendência

Tabela 9 – Resultados de acurácia obtidos a partir de uma validação cruzada com 10 dobras.

| Set | N | IM-RBFNN$_2$ | | | RBFNN | | | SVM |
|---|---|---|---|---|---|---|---|---|
| | | N/3 nós | N/5 nós | N/7 nós | N/3 nós | N/5 nós | N/7 nós | |
| IRI | 150 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| SYN | 600 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 99.83 (0.50) | 99.98 (0.06) | 100.00 (0.00) |
| ROB | 117 | 89.92 (10.40) | 79.55 (3.76) | 80.45 (5.07) | 69.47 (11.49) | 77.88 (5.15) | 79.55 (3.76) | 87.12 (4.05) |
| MUS | 1000 | 100.00 (0.00) | 99.95 (0.20) | 99.95 (0.20) | 100.00 (0.00) | 99.96 (0.10) | 99.95 (0.10) | 99.72 (0.16) |
| ION | 351 | 94.01 (3.72) | 94.29 (4.61) | 92.58 (5.31) | 91.44 (4.96) | 90.86 (6.36) | 89.17 (4.21) | 95.16 (3.84) |
| BAN | 1000 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 99.85 (0.29) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| WIN | 178 | 96.63 (4.46) | 97.19 (3.75) | 96.08 (3.57) | 98.89 (3.33) | 96.67 (3.56) | 95.46 (4.56) | 98.33 (3.56) |
| WBC | 569 | 97.01 (1.37) | 96.84 (1.53) | 95.96 (2.08) | 96.31 (2.65) | 97.37 (1.96) | 96.84 (1.05) | 98.42 (2.14) |
| SON | 208 | 80.21 (7.53) | 78.26 (8.21) | 75.83 (9.99) | 82.12 (9.93) | 81.19 (8.56) | 76.38 (9.06) | 69.26 (13.47) |
| SPA | 1000 | 92.28 (0.83) | 92.57 (1.33) | 92.04 (1.61) | 92.81 (1.02) | 92.91 (1.01) | 92.15 (1.10) | 93.15 (2.17) |
| TRA | 748 | 77.69 (2.78) | 77.41 (1.05) | 77.27 (0.81) | 71.51 (3.37) | 73.67 (9.85) | 76.20 (10.45) | 76.47 (0.84) |
| BCR | 683 | 96.49 (1.63) | 96.63 (1.87) | 96.78 (1.95) | 91.94 (2.59) | 96.33 (2.48) | 96.48 (1.77) | 97.23 (2.20) |
| ACR | 690 | 85.36 (3.14) | 85.65 (3.00) | 86.38 (3.38) | 83.91 (1.99) | 84.78 (3.56) | 85.80 (4.14) | 86.38 (3.50) |
| HAB | 306 | 73.87 (6.56) | 73.19 (3.53) | 75.83 (7.63) | 68.28 (6.25) | 69.31 (5.90) | 71.89 (5.88) | 75.51 (2.49) |
| HEA | 270 | 84.44 (6.37) | 82.96 (6.24) | 85.93 (7.55) | 81.11 (6.30) | 82.59 (7.23) | 81.48 (6.63) | 83.33 (5.04) |
| DIA | 768 | 76.57 (4.52) | 77.61 (3.70) | 76.57 (3.65) | 72.26 (3.39) | 74.47 (4.68) | 75.65 (4.21) | 77.47 (3.74) |
| MAM | 830 | 83.86 (3.58) | 84.10 (4.30) | 83.01 (4.33) | 76.99 (5.74) | 78.80 (5.71) | 79.28 (4.72) | 81.45 (4.64) |

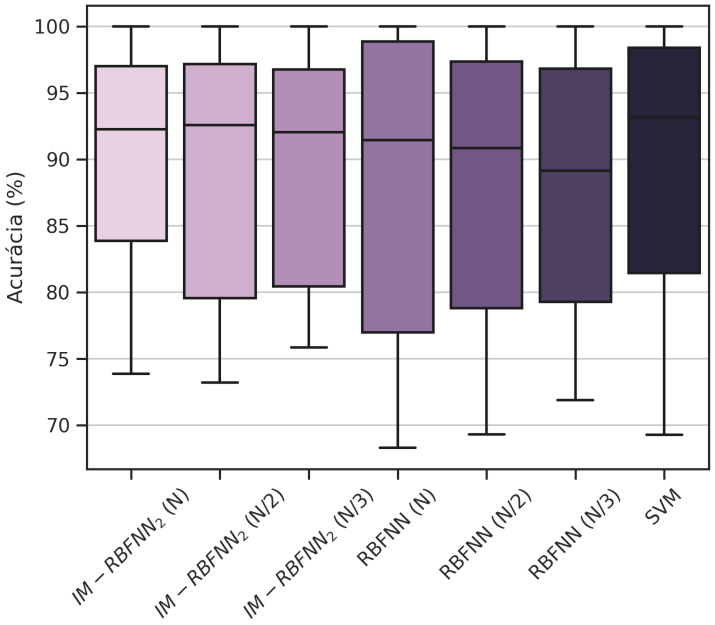


Figura 11 – Gráfico *boxplot* do comportamento médio de acurácia dos algoritmos na resolução de todos os problemas.

geral de melhor desempenho. O maior valor de mediana foi atingido pelo modelo SVM com kernel RBF. Contudo, nota-se que o modelo IM-RBFNN$_2$ com N neurônios foi o que obteve menor amplitude entre o primeiro e o terceiro quartil, o que indica menor variabilidade nos resultados e maior consistência. Além disso, os modelos IM-RBFNN$_2$ foram aqueles com maiores valores mínimos no gráfico.

Para quatro conjuntos de dados que resultaram em uma média de acurácia de treinamento de 100%, também foi calculado o valor de margem geométrica no espaço de características. A Tabela 10 apresenta os valores médios obtidos para a margem em uma va-

lidação cruzada de 10 dobras, com os respectivos desvios padrão indicados entre parênteses. Observou-se que, em todos os quatro conjuntos de dados e considerando diferentes números de neurônios testados, o método IM-RBFNN$_2$ demonstrou consistentemente valores de margem significativamente maiores em comparação ao método RBFNN convencional.

Tabela 10 – Valores médios de margem obtidos a partir da validação cruzada de 10 dobras.

| *Dataset* | Algoritmo | Neurônios Intermediários | Margem Média |
|---|---|---|---|
| IRI | IM-RBFNN$_2$ | 50 | 1.0000 (0.0266) |
| IRI | IM-RBFNN$_2$ | 30 | 0.7713 (0.0254) |
| IRI | IM-RBFNN$_2$ | 21 | 0.6507 (0.0220) |
| IRI | RBFNN | 50 | $2.3 \cdot 10^{-5}$ ($9.0 \cdot 10^{-6}$) |
| IRI | RBFNN | 30 | 0.0002 (0.0001) |
| IRI | RBFNN | 21 | 0.0017 (0.0007) |
| SYN | IM-RBFNN$_2$ | 200 | 0.1317 (0.0145) |
| SYN | IM-RBFNN$_2$ | 120 | 0.1250 (0.0170) |
| SYN | IM-RBFNN$_2$ | 85 | 0.1311 (0.0180) |
| SYN | RBFNN | 200 | 0.0009 (0.0002) |
| SYN | RBFNN | 120 | 0.0011 (0.0003) |
| SYN | RBFNN | 85 | 0.0010 (0.0005) |
| MUS | IM-RBFNN$_2$ | 333 | 0.0369 (0.0071) |
| MUS | IM-RBFNN$_2$ | 200 | 0.0405 (0.0144) |
| MUS | IM-RBFNN$_2$ | 142 | 0.0256 (0.0090) |
| MUS | RBFNN | 333 | 0.0028 (0.0006) |
| MUS | RBFNN | 200 | 0.0013 (0.0009) |
| MUS | RBFNN | 142 | 0.0004 (0.0008) |
| BAN | IM-RBFNN$_2$ | 333 | 0.0782 (0.0036) |
| BAN | IM-RBFNN$_2$ | 200 | 0.0638 (0.0022) |
| BAN | IM-RBFNN$_2$ | 142 | 0.0525 (0.0018) |
| BAN | RBFNN | 333 | $9.8 \cdot 10^{-12}$ ($2.1 \cdot 10^{-12}$) |
| BAN | RBFNN | 200 | $5.0 \cdot 10^{-9}$ ($1.0 \cdot 10^{-9}$) |
| BAN | RBFNN | 142 | $4.8 \cdot 10^{-8}$ ($1.1 \cdot 10^{-8}$) |

Para os *datasets* da Tabela 10 buscou-se ainda visualizar a evolução do valor de margem de acordo com o número de iterações do IM-RBFNN$_2$. Os gráficos mostrados na Figura 12 apresentam o valor médio da margem obtido em cada iteração do algoritmo IM-RBFNN$_2$, sendo que as barras de erro representam o erro padrão. Foram gerados esses gráficos apenas para os modelos da Tabela 10 com a maior quantidade de neurônios, para cada conjunto de dados. Mais uma vez, foi observado um aumento consistente da margem ao longo das iterações do algoritmo, especialmente durante as primeiras iterações.

## 4.5 Teste de Esparsidade para os Modelos IM-RBFNN

Assim como realizado para os modelos RP-IMA$_p$, também foi testada a esparsidade para as formulações IM-RBFNN$_1$, IM-RBFNN$_2$ e IM-RBFNN$_\infty$. A mesma medida de esparsidade utilizada na seção 4.2 foi utilizada aqui.

A Tabela 11 apresenta os resultados de esparsidade (considerando diferentes valores de $\rho$) e acurácia obtidos para cada algoritmo. Esses resultados são valores médios obtidos a partir de uma validação cruzada estratificada de 10 dobras. Os valores entre parênteses

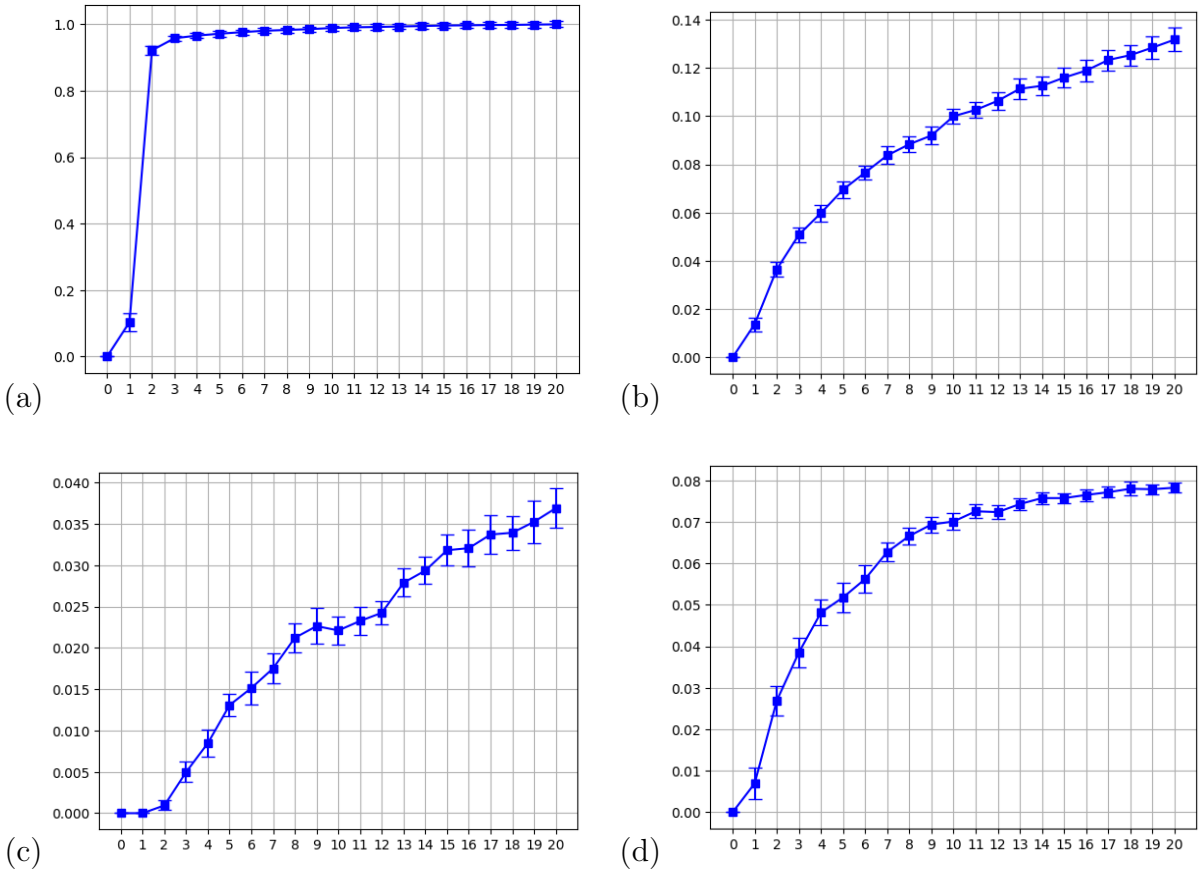


Figura 12 – Margem geométrica calculada a cada iteração do IM-RBFNN$_2$. **a** *Iris*. **b** *Synthetic control chart time series*. **c** *Mushroom*. **d** *Banknote*.

representam os desvios padrão. Para evitar efeitos de aleatoriedade, os parâmetros relativos aos neurônios da camada escondida foram os mesmos para os modelos IM-RBFNN$_1$, IM-RBFNN$_2$ e IM-RBFNN$_\infty$.

Para todas as formulações do IM-RBFNN$_p$, os hiperparâmetros foram definidos como $\eta = 0.1$, $\delta = 10^{-3}$. O valor do hiperparâmetro $\lambda$ foi selecionado para cada conjunto de dados por meio da técnica *grid search cross-validation*. Como critério de parada foi definido um máximo de 10000 atualizações do vetor $\boldsymbol{\beta}$ ou 20 iterações do algoritmo.

A partir da Tabela 11, nota-se que os modelos IM-RBFNN$_\infty$ obtiveram soluções mais esparsas para todos os problemas testados. Além disso, os resultados de acurácia obtidos por esses métodos não apresentaram diferenças significativas de comportamento quando comparado com os resultados obtidos pelo IM-RBFNN$_2$. Assim, o método IM-RBFNN$_\infty$ apresentou soluções mais esparsas mantendo um bom desempenho com relação às demais formulações.

Já método IM-RBFNN$_1$, na maioria dos problemas obteve soluções menos esparsas e além disso para algumas bases de dados, como ROB, WIN, HEA e MAM, esse algoritmo apresentou resultados de acurácia significativamente menores que os obtidos pelos demais

Tabela 11 – Resultados de esparsidade.

| Set | Nós | Método | Medida de Esparsidade ($\rho \cdot 100\%$) | | | | Acurácia (%) |
|---|---|---|---|---|---|---|---|
| | | | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ | |
| IRI | 50 | IM-RBFNN$_1$ | 49.9 (0.3) | 49.9 (0.3) | 50.0 (0.0) | 50.0 (0.0) | 100.00 (0.00) |
| | | IM-RBFNN$_2$ | 50.0 (0.0) | 50.0 (0.0) | 50.0 (0.0) | 50.0 (0.0) | 100.00 (0.00) |
| | | IM-RBFNN$_\infty$ | 12.4 (3.9) | 15.9 (4.7) | 27.1 (7.9) | 45.4 (4.5) | 100.00 (0.00) |
| SYN | 200 | IM-RBFNN$_1$ | 177.1 (4.6) | 188.7 (3.7) | 198.8 (0.74) | 200.0 (0.0) | 100.00 (0.00) |
| | | IM-RBFNN$_2$ | 160.9 (3.3) | 179.3 (3.0) | 197.8 (1.5) | 199.6 (0.5) | 100.00 (0.00) |
| | | IM-RBFNN$_\infty$ | 95.7 (11.6) | 108.8 (11.1) | 128.8 (15.8) | 189.6 (5.2) | 100.00 (0.00) |
| ROB | 39 | IM-RBFNN$_1$ | 29.1 (1.3) | 33.5 (1.6) | 38.4 (0.7) | 39.0 (0.0) | 79.55 (3.76) |
| | | IM-RBFNN$_2$ | 27.7 (2.0) | 33.1 (1.4) | 38.3 (0.6) | 38.9 (0.3) | 89.98 (10.40) |
| | | IM-RBFNN$_\infty$ | 22.1 (3.9) | 27.7 (3.6) | 34.9 (5.1) | 38.4 (1.2) | 88.99 (10.69) |
| MUS | 333 | IM-RBFNN$_1$ | 230.1 (13.1) | 284.5 (5.8) | 326.8 (2.9) | 331.7 (1.5) | 100.00 (0.00) |
| | | IM-RBFNN$_2$ | 172.9 (11.2) | 281.2 (4.3) | 326.8 (2.0) | 332.1 (0.7) | 100.00 (0.00) |
| | | IM-RBFNN$_\infty$ | 150.9 (9.6) | 268.5 (9.2) | 310.2 (3.8) | 314.8 (5.4) | 100.00 (0.00) |
| ION | 117 | IM-RBFNN$_1$ | 88.3 (6.5) | 102.8 (3.9) | 115.0 (1.5) | 116.7 (0.5) | 94.29 (3.84) |
| | | IM-RBFNN$_2$ | 68.8 (5.8) | 91.2 (6.0) | 114.2 (1.1) | 116.4 (0.7) | 94.01 (3.72) |
| | | IM-RBFNN$_\infty$ | 46.3 (4.4) | 64.1 (6.9) | 93.7 (6.6) | 113.9 (2.0) | 93.44 (6.00) |
| BAN | 333 | IM-RBFNN$_1$ | 297.2 (6.0) | 315.8 (3.8) | 330.9 (2.0) | 332.9 (0.3) | 99.85 (0.29) |
| | | IM-RBFNN$_2$ | 257.4 (4.7) | 296.5 (2.2) | 329.1 (2.5) | 332.5 (0.7) | 100.00 (0.00) |
| | | IM-RBFNN$_\infty$ | 117.9 (8.4) | 143.1 (10.7) | 171.3 (14.2) | 281.9 (16.4) | 100.00 (0.00) |
| WIN | 59 | IM-RBFNN$_1$ | 54.3 (3.2) | 57.3 (1.6) | 58.8 (0.4) | 59.0 (0.0) | 93.30 (7.37) |
| | | IM-RBFNN$_2$ | 49.2 (2.2) | 54.7 (1.7) | 58.4 (0.7) | 58.9 (0.3) | 96.63 (4.46) |
| | | IM-RBFNN$_\infty$ | 41.2 (3.1) | 47.2 (3.5) | 56.7 (2.2) | 58.6 (0.7) | 97.2 (3.75) |
| WBC | 189 | IM-RBFNN$_1$ | 156.9 (6.4) | 171.7 (3.2) | 187.7 (0.8) | 189.0 (0.0) | 95.95 (1.60) |
| | | IM-RBFNN$_2$ | 109.2 (3.6) | 145.2 (3.8) | 184.1 (1.9) | 188.7 (0.5) | 97.01 (1.37) |
| | | IM-RBFNN$_\infty$ | 79.0 (5.2) | 103.9 (6.1) | 161.8 (5.8) | 183.5 (2.0) | 96.48 (1.94) |
| SON | 69 | IM-RBFNN$_1$ | 56.0 (2.5) | 62.8 (1.3) | 68.0 (1.0) | 69.0 (0.0) | 80.21 (8.39) |
| | | IM-RBFNN$_2$ | 47.8 (4.5) | 58.5 (2.5) | 68.7 (0.5) | 69.0 (0.0) | 80.21 (7.53) |
| | | IM-RBFNN$_\infty$ | 42.2 (3.0) | 53.3 (3.4) | 66.5 (1.6) | 68.7 (0.5) | 79.33 (6.41) |
| SPA | 333 | IM-RBFNN$_1$ | 225.9 (7.8) | 275.2 (4.5) | 324.7 (3.0) | 332.1 (1.0) | 91.83 (1.21) |
| | | IM-RBFNN$_2$ | 205.8 (9.1) | 266.6 (6.9) | 326.2 (1.7) | 332.2 (0.7) | 92.28 (0.83) |
| | | IM-RBFNN$_\infty$ | 201.8 (5.5) | 262.1 (4.2) | 323.0 (1.9) | 330.5 (0.8) | 92.02 (1.12) |
| TRA | 249 | IM-RBFNN$_1$ | 175.0 (16.8) | 211.1 (12.6) | 245.8 (1.9) | 248.8 (0.6) | 77.67 (3.10) |
| | | IM-RBFNN$_2$ | 116.5 (13.7) | 172.3 (11.4) | 239.3 (2.0) | 248.2 (0.6) | 77.69 (2.78) |
| | | IM-RBFNN$_\infty$ | 19.3 (7.6) | 32.7 (10.3) | 171.6 (23.1) | 239.2 (3.4) | 79.15 (3.09) |
| BCR | 227 | IM-RBFNN$_1$ | 136.3 (17.1) | 181.8 (9.7) | 223.0 (2.2) | 225.9 (1.1) | 96.19 (2.47) |
| | | IM-RBFNN$_2$ | 137.4 (11.4) | 181.0 (6.1) | 223.0 (1.7) | 226.3 (0.9) | 96.49 (1.63) |
| | | IM-RBFNN$_\infty$ | 48.8 (7.7) | 69.8 (15.4) | 197.2 (10.6) | 223.3 (1.6) | 96.78 (1.95) |
| ACR | 230 | IM-RBFNN$_1$ | 195.4 (8.5) | 210.9 (5.3) | 228.5 (0.9) | 229.9 (0.3) | 83.77 (4.71) |
| | | IM-RBFNN$_2$ | 147.1 (7.6) | 186.3 (4.2) | 226.0 (2.0) | 229.7 (0.5) | 85.36 (3.14) |
| | | IM-RBFNN$_\infty$ | 101.4 (11.7) | 143.7 (16.3) | 214.1 (3.9) | 227.6 (1.5) | 85.80 (2.81) |
| HAB | 102 | IM-RBFNN$_1$ | 83.9 (7.1) | 92.5 (5.3) | 101.2 (1.0) | 101.9 (0.3) | 73.87 (6.56) |
| | | IM-RBFNN$_2$ | 74.9 (4.1) | 89.3 (2.6) | 100.7 (0.9) | 102.0 (0.0) | 73.55 (3.52) |
| | | IM-RBFNN$_\infty$ | 57.0 (19.0) | 76.3 (12.1) | 98.6 (1.7) | 101.5 (0.7) | 72.56 (5.01) |
| HEA | 90 | IM-RBFNN$_1$ | 84.6 (3.3) | 87.3 (1.8) | 89.6 (0.8) | 90.0 (0.0) | 74.81 (14.33) |
| | | IM-RBFNN$_2$ | 51.8 (3.7) | 71.0 (3.8) | 88.1 (0.9) | 88.9 (0.4) | 84.44 (6.37) |
| | | IM-RBFNN$_\infty$ | 31.8 (4.3) | 46.7 (4.3) | 79.9 (2.5) | 88.8 (0.9) | 83.70 (6.24) |
| DIA | 256 | IM-RBFNN$_1$ | 197.1 (14.7) | 225.7 (10.3) | 253.0 (1.3) | 255.6 (0.7) | 74.75 (5.88) |
| | | IM-RBFNN$_2$ | 136.9 (17.3) | 198.1 (9.7) | 250.2 (2.6) | 255.4 (1.0) | 76.57 (4.52) |
| | | IM-RBFNN$_\infty$ | 113.6 (14.7) | 165.8 (12.9) | 242.4 (4.8) | 254.4 (1.1) | 77.35 (3.98) |
| MAM | 276 | IM-RBFNN$_1$ | 228.4 (12.4) | 252.9 (8.3) | 273.0 (2.1) | 275.1 (0.5) | 75.18 (8.64) |
| | | IM-RBFNN$_2$ | 136.7 (17.4) | 214.2 (14.4) | 271.3 (1.8) | 275.2 (0.6) | 83.86 (3.58) |
| | | IM-RBFNN$_\infty$ | 69.2 (7.1) | 111.1 (10.8) | 230.5 (19.9) | 269.4 (3.5) | 83.49 (4.41) |

métodos.

A Figura 13 apresenta a comparação da esparsidade entre IM-RBFNN$_1$, IM-RBFNN$_2$ e IM-RBFNN$_\infty$. Essa figura mostra as magnitudes normalizadas de todos os pesos da rede de cada método para quatro conjuntos de dados diferentes. Como pode ser observado, a esparsidade é maior nas soluções do IM-RBFNN$_\infty$, uma vez que resultou em mais pesos com magnitudes nulas (ou muito pequenas).

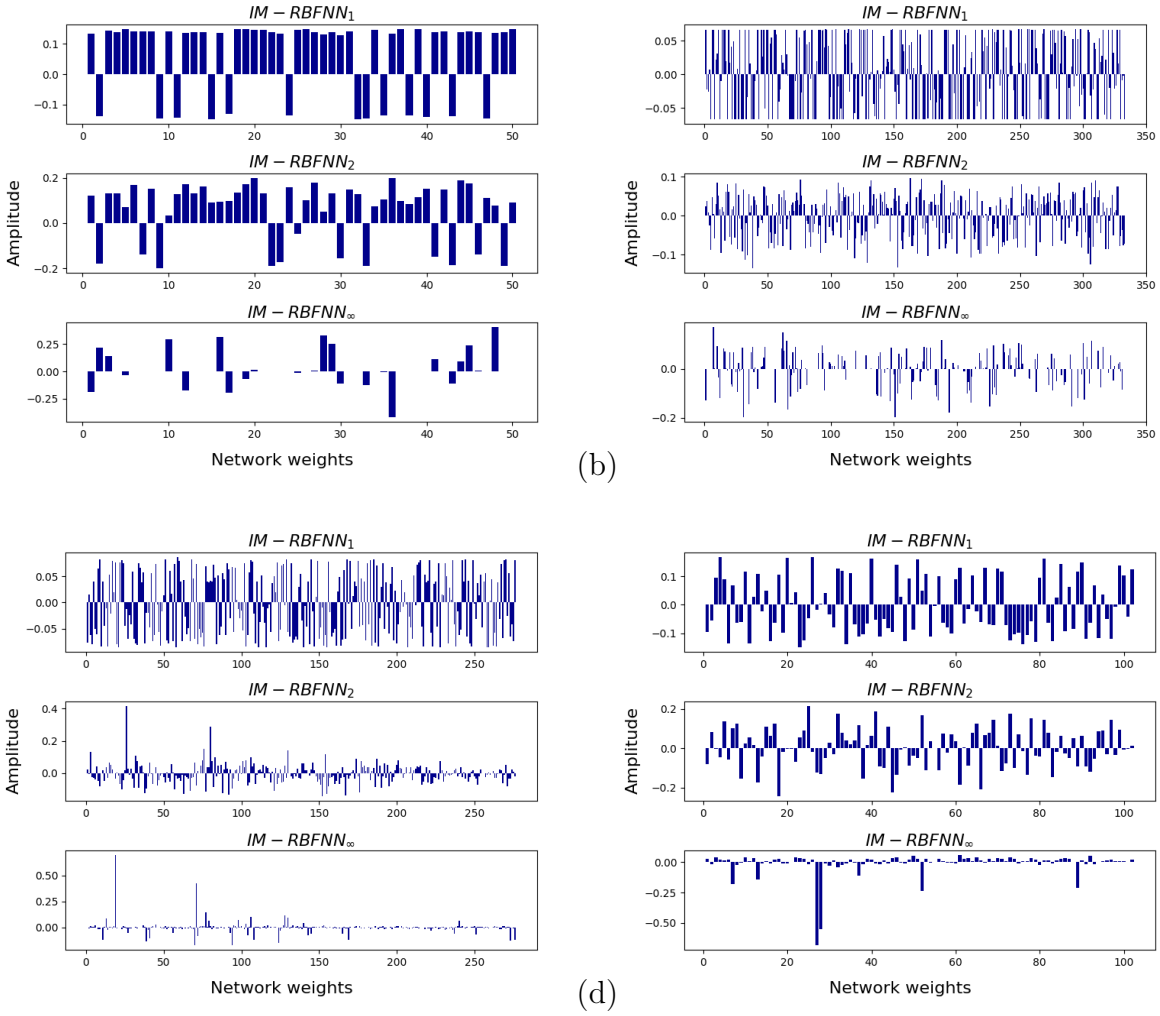


Figura 13 – Comparação da esparsidade entre as soluções de rede IM-RBFNN$_1$, IM-RBFNN$_2$ e IM-RBFNN$_\infty$. **a** *Iris*. **b** *Banknote*. **c** *Mammographic*. **d** *Haberman*.

## 4.6 Resultados do Modelo IM-RBFNN$_\infty$ com Método de Poda

Como mostrado na seção anterior o método IM-RBFNN$_\infty$ gera soluções mais esparsas que as outras formulações estudadas neste trabalho. Assim, esse algoritmo utilizado

juntamente com o método de poda descrito na seção 3.3 foi avaliado aqui. Para isso, o IM-RBFNN$_\infty$ com o método de poda foi aplicado aos conjuntos de dados da Tabela 2.

A mesma metodologia e hiperparâmetros dos testes anteriores foram aplicados. Apenas o parâmetro $\lambda$ (relacionado a flexibilização de margem) foi selecionado através de um *grid search cross-validation*. A Tabela 12 apresenta o número de neurônios ocultos iniciais e a quantidade final selecionada. Essa tabela mostra que o método de poda proposto é capaz de gerar arquiteturas de rede com uma quantidade significativamente menor de neurônios na camada oculta, em comparação com o IM-RBFNN sem poda.

Considerando todos os problemas, para $\rho = 20\%$ alcançou-se uma redução média de 61,03% dos neurônios da camada escondida da rede, para $\rho = 10\%$ esse valor foi de 47,51%, para $\rho = 1\%$ obteve-se uma redução média de 25,98% e finalmente para $\rho = 0.1\%$ eliminou-se na média 7,72% dos nós da camada escondida.

Tabela 12 – Neurônios selecionados pelo IM-RBFNN$_\infty$ com o método de poda.

| *Dataset* | Neurônios iniciais | Neurônios selecionados pelo IM-RBFNN$_\infty$ com poda | | | |
|---|---|---|---|---|---|
| | | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ |
| IRI | 50 | 20.0 (5.4) | 22.0 (5.6) | 30.1 (7.2) | 38.1 (5.8) |
| SYN | 200 | 100.4 (30.0) | 122.9 (13.3) | 159.9 (12.0) | 193.1 (2.3) |
| ROB | 39 | 13.8 (3.2) | 19.3 (1.9) | 29.9 (1.5) | 36.3 (1.3) |
| MUS | 333 | 191.6 (7.4) | 257.9 (5.8) | 305.8 (4.0) | 317.2 (5.4) |
| ION | 117 | 47.9 (3.0) | 62.4 (4.9) | 89.2 (5.2) | 110.4 (2.6) |
| BAN | 333 | 205.9 (7.7) | 237.5 (4.4) | 282.6 (6.4) | 315.2 (4.8) |
| WIN | 59 | 35.0 (3.3) | 43.1 (2.4) | 48.7 (1.4) | 56.8 (0.9) |
| WBC | 189 | 95.5 (5.1) | 122.2 (5.5) | 152.7 (4.2) | 179.3 (3.7) |
| SON | 69 | 28.4 (2.5) | 38.1 (1.8) | 52.7 (1.7) | 65.1 (1.9) |
| SPA | 333 | 152.1 (4.7) | 213.8 (7.9) | 290.3 (5.2) | 324.1 (2.2) |
| TRA | 249 | 13.6 (4.2) | 31.6 (23.6) | 67.8 (14.3) | 196.2 (8.9) |
| BCR | 227 | 60.7 (8.5) | 84.4 (7.5) | 155.7 (8.5) | 212.8 (5.7) |
| ACR | 230 | 118.4 (11.9) | 150.2 (8.3) | 190.9 (4.3) | 218.1 (3.9) |
| HAB | 102 | 23.9 (7.2) | 44.7 (11.6) | 76.5 (11.4) | 96.1 (3.8) |
| HEA | 90 | 26.9 (2.7) | 40.4 (4.0) | 67.0 (6.1) | 86.1 (1.2) |
| DIA | 256 | 79.8 (11.2) | 124.8 (13.4) | 197.0 (5.3) | 238.7 (4.9) |
| MAM | 276 | 32.4 (7.4) | 70.4 (11.1) | 155.8 (21.1) | 235.5 (7.4) |

A Tabela 13 mostra o desempenho com relação a acurácia dos métodos IM-RBFNN$_\infty$ com poda, RBFNN tradicional e IM-RBFNN$_2$. Para evitar efeitos de aleatoriedade decorrentes do *k-means*, todos os algoritmos foram executados utilizando-se a mesma definição dos parâmetros das funções de base radial implementadas pelos neurônios da camada escondida.

A partir dessa tabela foi possível observar que todos os modelos IM-RBFNN$_\infty$ com método de poda mantiveram um bom desempenho médio de acurácia, similar ao do IM-RBFNN$_2$. Isso pode ser visto de maneira mais clara no gráfico da Figura 14. Esta figura apresenta, por meio de um gráfico de boxplot, o comportamento médio do método IM-RBFNN$_\infty$ com poda em comparação com o RBFNN e o IM-RBFNN$_2$ na resolução de todos os problemas.

Tabela 13 – Resultados de Acurácia para os métodos: IM-RBFNN$_\infty$ com poda, RBFNN e IM-RBFNN$_2$.

| *Dataset* | Pruned IM-RBFNN$_\infty$ | | | | IM-RBFNN$_2$ | RBFNN |
|---|---|---|---|---|---|---|
| | $\rho = 20\%$ | $\rho = 10\%$ | $\rho = 1\%$ | $\rho = 0.1\%$ | | |
| IRI | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| SYN | 99.50 (1.07) | 99.50 (1.07) | 99.67 (0.67) | 99.67 (0.67) | 100.00 (0.00) | 100.00 (0.00) |
| ROB | 83.03 (9.15) | 85.53 (9.99) | 83.94 (10.14) | 89.09 (11.20) | 83.11 (9.07) | 71.06 (7.26) |
| MUS | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| ION | 92.29 (4.44) | 93.15 (3.88) | 92.01 (5.40) | 93.15 (4.47) | 92.29 (6.14) | 91.17 (4.69) |
| BAN | 99.93 (0.22) | 100.00 (0.00) | 99.71 (0.88) | 99.71 (0.36) | 100.00 (0.00) | 100.00 (0.00) |
| WIN | 97.19 (3.75) | 96.08 (4.35) | 96.08 (5.01) | 96.08 (3.57) | 97.19 (3.75) | 98.89 (2.22) |
| WBC | 95.26 (2.22) | 94.37 (2.70) | 95.08 (2.05) | 94.73 (2.22) | 96.48 (2.37) | 95.95 (2.39) |
| SON | 77.81 (9.09) | 77.88 (5.80) | 80.24 (7.78) | 78.29 (10.99) | 80.62 (9.90) | 81.62 (10.12) |
| SPA | 90.50 (2.01) | 91.18 (1.82) | 92.13 (0.99) | 91.89 (1.09) | 92.07 (0.93) | 92.87 (0.58) |
| TRA | 76.34 (0.52) | 77.41 (2.60) | 78.48 (2.36) | 77.55 (2.15) | 77.01 (1.27) | 71.11 (2.97) |
| BCR | 97.07 (1.60) | 97.51 (1.86) | 97.22 (1.79) | 96.92 (1.91) | 96.78 (1.58) | 93.40 (3.38) |
| ACR | 85.22 (4.09) | 86.52 (3.95) | 86.23 (3.06) | 85.22 (4.24) | 85.80 (2.58) | 82.61 (2.05) |
| HAB | 74.20 (6.13) | 74.17 (6.43) | 72.91 (4.11) | 73.55 (5.59) | 73.53 (0.94) | 66.65 (5.31) |
| HEA | 80.74 (5.93) | 82.96 (6.02) | 83.33 (7.99) | 84.44 (9.34) | 82.22 (7.37) | 79.26 (7.26) |
| DIA | 76.69 (4.27) | 76.18 (4.60) | 76.83 (4.05) | 77.08 (4.26) | 76.04 (3.76) | 71.49 (3.63) |
| MAM | 82.29 (6.05) | 82.53 (3.29) | 83.73 (5.15) | 83.13 (5.78) | 83.25 (5.10) | 76.87 (5.38) |

Observando a Figura 14, nota-se que o método RBFNN foi o que apresentou maior amplitude entre os quartis, indicando maior variabilidade. Com exceção do modelo IM-RBFNN$\infty$ com $\rho = 20\%$, todos os modelos com poda apresentam uma amplitude entre os quartis muito parecidas com a obtida pelo modelo IM-RBFNN$_2$. O mesmo pode se dizer para a mediana, a diferença da mediana de acurácia entre todos os modelos é inferior a 2%. Os métodos que obtiveram a maior mediana de acurácia (92%) foram o IM-RBFNN$_\infty$ com $\rho = 1\%$ e IM-RBFNN$_2$. Já o método IM-RBFNN$_\infty$ com $\rho = 20\%$ apresentou a pior mediana de acurácia (90.5%).

## 4.7 Resultados do Modelo RP-IMA Definido em Variáveis Duais

Os testes dessa seção seguem a mesma metodologia daqueles realizados na seção 4.1, ou seja foram realizados testes com bases sintéticas e bases de *benchmark*. Todas as bases utilizadas nesta seção tiveram seus dados normalizados antes de cada teste.

### 4.7.1 Experimento com Bases Sintéticas

Novamente, primeiro foram realizados testes com as bases sintéticas de duas dimensões. Na Figura 15 é possível observar as superfícies de separação geradas pelos modelos ELM Dual (à esquerda) e RP-IMA Dual (à direita). Para cada problema, os modelos foram gerados utilizando 100 neurônios com função de ativação sigmoidal na

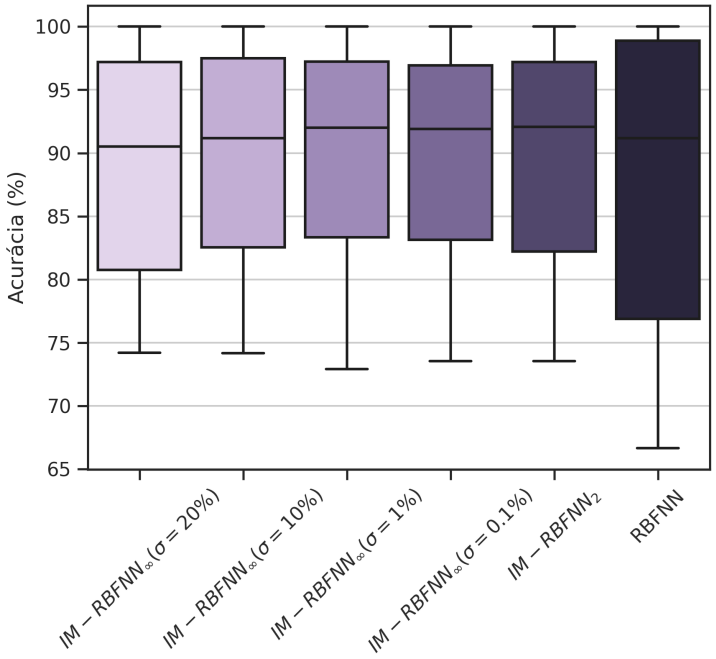


Figura 14 – Gráfico *boxplot* do comportamento médio de acurácia dos algoritmos IM-RBFNN$_\infty$ com método de poda, RBFNN e RP-IMA$_2$ na resolução de todos os problemas.

camada escondida e o tipo de kernel utilizado foi o gaussiano. Para tentar diminuir efeitos de aleatoriedade na comparação entre os métodos, optou-se por selecionar os mesmos pesos aleatórios para os neurônios ocultos de ambos os métodos.

Os outros hiperparâmetros específicos do RP-IMA Dual foram fixados em: $\eta = 0.1$ (taxa de aprendizagem), $\lambda = 0.01$ (fator de regularização) e $\delta = 10^{-3}$ (fator responsável por definir um incremento mínimo da margem a cada iteração). Também, como critério de parada do algoritmo, foi definido um máximo de 10000 atualizações do vetor $\boldsymbol{\beta}$ e permitiu-se a execução de no máximo 20 iterações do algoritmo.

O parâmetro que define a largura de banda do *kernel* RBF ($\sigma$) foi selecionado tanto para o RP-IMA Dual quanto para a ELM Dual através da regra de Silverman [Silverman, 1986]:

$$\sigma = 0.9 \cdot min\left(\hat{\sigma}; \frac{q_3 - q_1}{1.349}\right) n^{-\frac{1}{5}}, \tag{4.3}$$

onde $\hat{\sigma}$ é o desvio padrão dos dados, $n$ é o número de amostras de treinamento e $q_3$ e $q_1$ são, respectivamente, o primeiro e o terceiro quartil calculados a partir dos dados.

Pelas superfícies de decisão mostradas na Figura 15 é possível observar que ambos os métodos geraram superfícies capazes de distinguir entre os dados de treinamento das 2 classes. Além disso, nota-se uma semelhança visual entre as bordas de decisão geradas pelos dois modelos. Diferentemente dos casos estudados nas seções anteriores, neste teste não foi possível perceber que o aumento da margem na camada final do modelo gerou

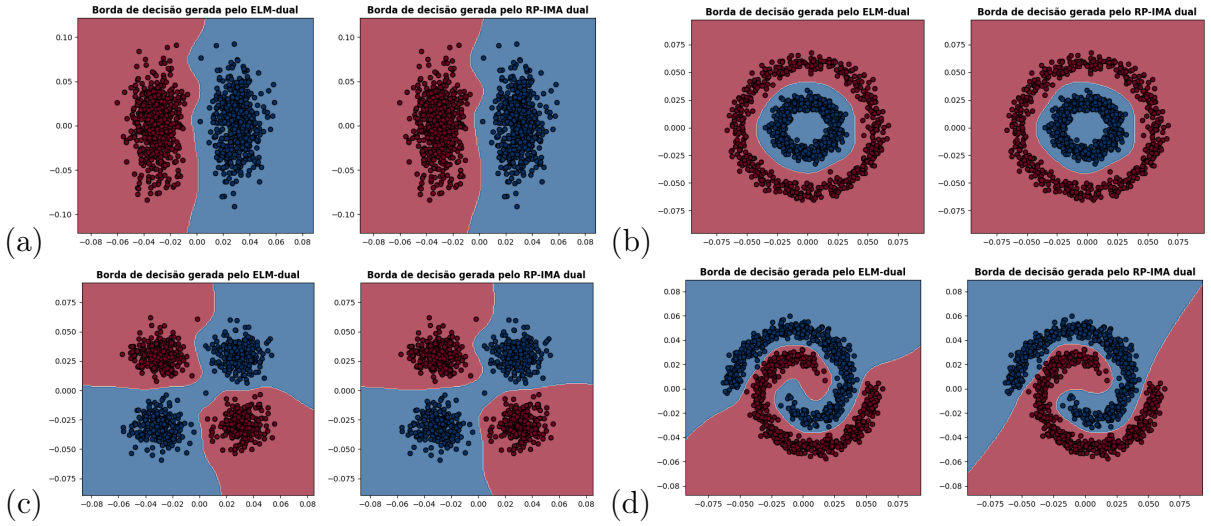


Figura 15 – Superfície de classificação gerada pela ELM Dual (à esquerda) e pelo RP-IMA (à direita). *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

modificações claras nas bordas de decisão dos quatro problemas sintéticos estudados aqui.

Para estudar mais profundamente o comportamento dos métodos para esses problemas foi realizado um teste para medir a acurácia e valor da margem geométrica (com relação ao espaço de características) obtidos pelos modelos. Na Tabela 14 são apresentados esses resultados médios obtidos para essas medidas a partir de uma validação cruzada de 10 dobras. Para esse teste todos os *datasets* foram normalizados para o intervalo [0, 1].

Tabela 14 – Resultados obtidos para as bases sintéticas.

| Algoritmo | Acurácia | Margem Média |
|---|---|---|
| *Gaussian blobs:* | | |
| ELM Dual | 100.0 (0.0) | 0.105 (0.009) |
| RP-IMA Dual | 100.0 (0.0) | 0.170 (0.009) |
| *Xor:* | | |
| ELM Dual | 99.8 (0.4) | 0.065 (0.004) |
| RP-IMA Dual | 99.7 (0.5) | 0.151 (0.009) |
| *Circles:* | | |
| ELM Dual | 100.0 (0.0) | 0.076 (0.005) |
| RP-IMA Dual | 100.0 (0.0) | 0.179 (0.003) |
| *Spirals:* | | |
| ELM dual | 99.9 (0.3) | 0.028 (0.001) |
| RP-IMA Dual | 99.9 (0.2) | 0.076 (0.006) |

Nas Equações 4.4 e 4.5 abaixo estão mostrados como os valores de margem foram computados para cada algoritmo.

- Margem calculada para modelos ELM Dual:

$$\text{margem} = \min_{j=1,\ldots,N} \left( y_j \frac{\sum_{i=1}^{N} \alpha_i K(h(\mathbf{x}_i), h(\mathbf{x}_j))}{\sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(h(\mathbf{x}_i), h(\mathbf{x}_j))}} \right). \tag{4.4}$$

- Margem calculada para modelos RP-IMA Dual:

$$\text{margem} = \min_{j=1,\ldots,N} \left( y_j \frac{\sum_{i=1}^{N} \alpha_i y_i K(h(\mathbf{x}_i), h(\mathbf{x}_j))}{\|\boldsymbol{\beta}\|} \right), \tag{4.5}$$

onde N corresponde ao número de amostras de treinamento, $\alpha_i$ é o multiplicador relacionado a amostra i, $h(\mathbf{x}_i)$ representa a amostra i após a realização da projeção aleatória e aplicação da função de ativação, $K(\cdot, \cdot)$ equivale a função de *kernel* utilizada e $\|\boldsymbol{\beta}\|$ foi calculado de acordo com a Equação 2.14.

Pela Tabela 14 nota-se que tanto a ELM Dual quanto o RP-IMA Dual obtiveram acurácias de teste próximas ou iguais a 100% em todas as bases de dados sintéticas. Isso indica que ambos os algoritmos foram capazes de capturar os padrões dos dados e gerar boas superfícies de separação, classificando todos os pontos corretamente. Contudo, para todas as bases de dados, o RP-IMA Dual apresentou margem média superior em comparação com o ELM Dual. Assim pode-se dizer que o RP-IMA Dual criou fronteiras de decisão que são mais bem espaçadas entre as classes, isso é importante para a obtenção de modelos mais robustos em termos de generalização para dados não vistos.

Assim, como nas outras seções, foram gerados também gráficos que mostram a evolução do valor da margem para cada iteração do RP-IMA Dual. Cada ponto nesses gráficos foi calculado de acordo com a equação 4.5, considerando o vetor $\alpha$ correspondente a cada iteração do algoritmo. Na Figura 16 é possível observar novamente uma característica marcante da evolução da margem, que inicialmente tem um crescimento mias rápido, e à medida que os valores aumentam, o crescimento se torna mais lento e a curva se aproxima de um valor limite, a margem máxima.

Por fim, para finalizar os testes dessa seção decidiu-se observar o efeito o parâmetro de regularização ($\lambda$) nos algoritmos duais estudados aqui. Em todos os testes anteriores utilizou-se um valor de 0.01 para esse parâmetro em todos os modelos. Porém, buscando visualizar o comportamento dos modelos para diferentes valores decidiu-se também realizar testes considerando $\lambda = 0$ e $\lambda = 1000$. Todos os outros hiperparâmetros foram mantidos como descrito anteriormente.

Na Figura 17 tem-se as superfícies de separação geradas pelos modelo considerando $\lambda = 0$. É possível observar sinais claros de *overfitting* nos modelos ELM Dual, o que não ocorre para os modelos RP-IMA Dual. Isso demonstra que uma margem larga pode indiretamente ter um efeito semelhante ao da regularização, um classificador de margem larga permite que a solução se afaste dos pontos de treinamento de fronteira e generalize melhor para novos exemplos. Isso é semelhante ao objetivo da regularização, que é evitar o ajuste excessivo aos dados de treinamento e promover a capacidade de generalização.

Um detalhe importante é que, apesar do modelo RP-IMA Dual com $\lambda = 0$ ter tido um bom desempenho para esses problemas sintéticos, o mesmo não ocorrerá sempre.

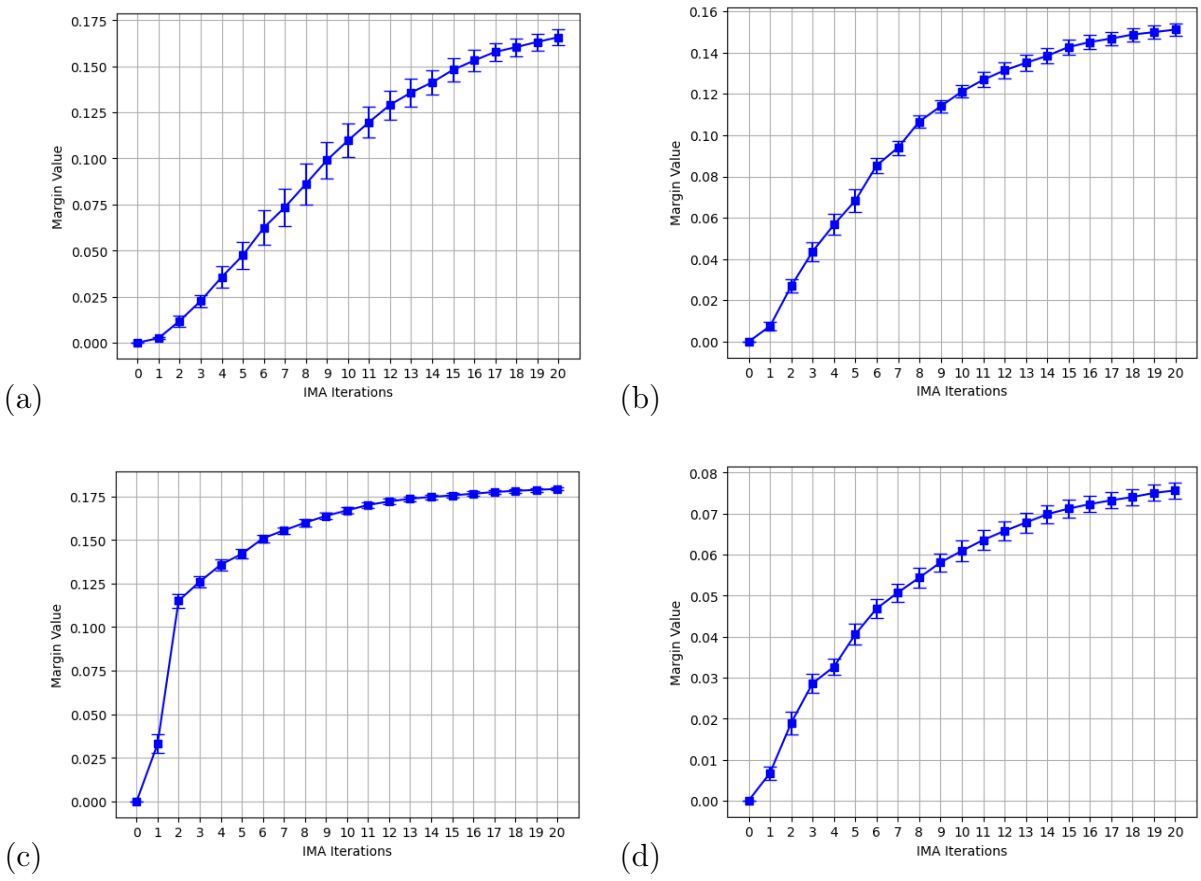


Figura 16 – Margem geométrica calculada a cada iteração do RP-IMA para *datasets* gerados sinteticamente. *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

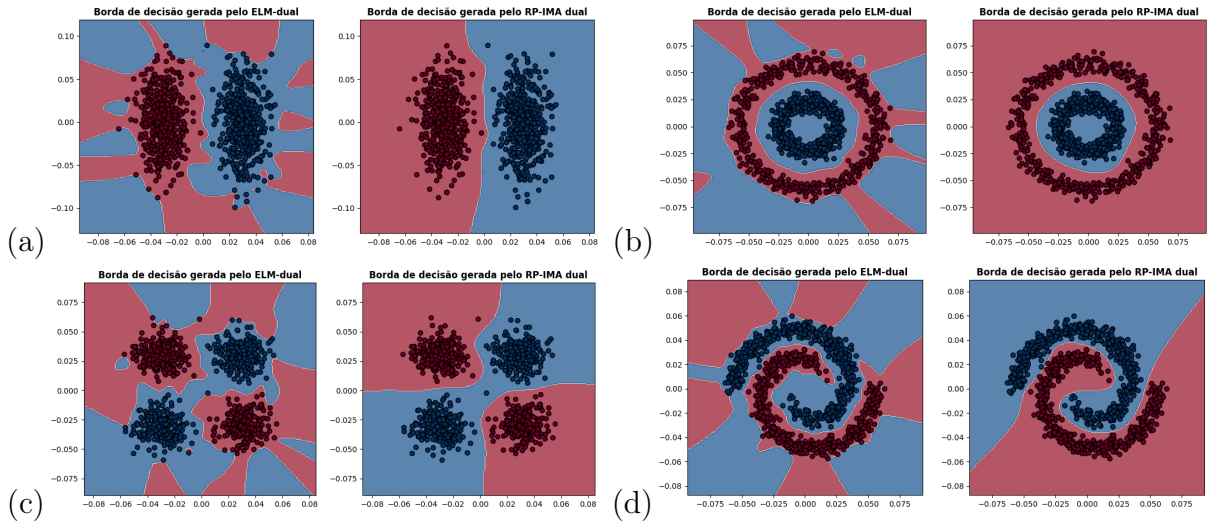


Figura 17 – Superfície de classificação gerada pela ELM Dual (à esquerda) e pelo RP-IMA (à direita), ambos utilizando parâmetro $\lambda = 0$. *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

Isso visto que esse parâmetro está associado a capacidade do modelo de implementar uma margem flexível. Para problemas mais complexos, onde ocorra superposições de amostras

de diferentes classes, esse parâmetro é fundamental para o algoritmo FMP dual conseguir convergir dentro de uma quantidade de iterações viável. Assim, para valores muito baixos ou nulos de $\lambda$, pode ocorrer do algoritmo RP-IMA Dual não conseguir realizar uma boa quantidade de iterações de incremento do valor da margem fixa.

Já na Figura 18 tem-se as superfícies de separação geradas pelos modelo considerando $\lambda = 1000$. Para as bases de dados *Xor* e *Gaussian blobs* ambos os modelos geraram superfícies de separação com boa capacidade de generalização para as amostras dos problemas. Contudo, para as bases nas bases *Circles* e *Spirals* são observados sinais *underfitting* para ambos os modelos, ou seja, eles não conseguiram capturar corretamente alguns padrões e relações nos dados de treinamento. Um alto valor para o parâmetro de regularização pode levar a modelos excessivamente simplificados, limitando assim a flexibilidade dos mesmos para se ajustar aos dados de treinamento.

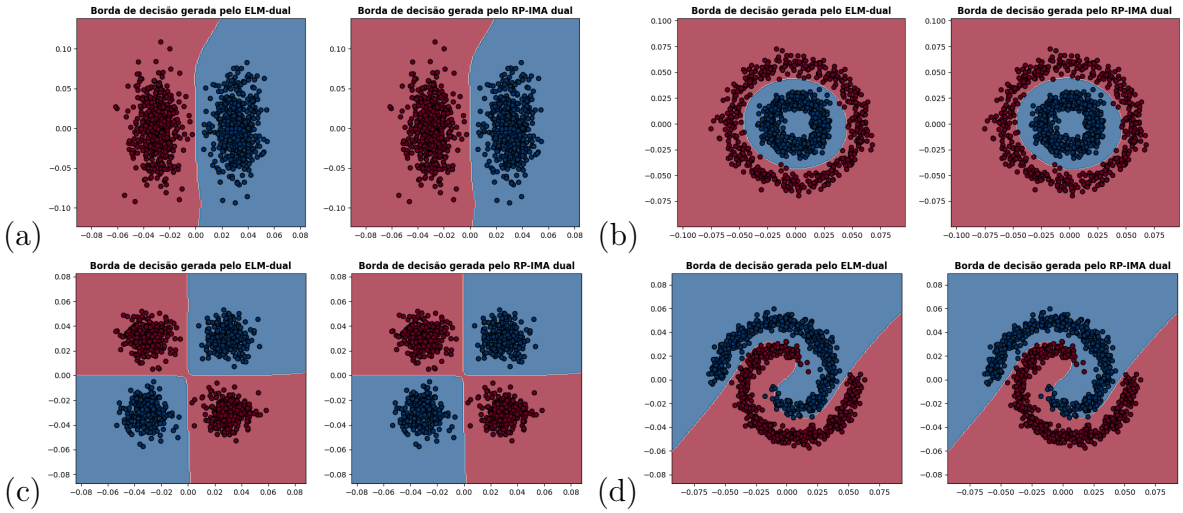


Figura 18 – Superfície de classificação gerada pela ELM Dual (à esquerda) e pelo RP-IMA (à direita), ambos utilizando parâmetro $\lambda = 1000$. *Datasets*: **(a)** *Gaussian blobs*; **(b)** *Circles*; **(c)** *Xor*; **(d)** *Spirals*.

### 4.7.2 Experimento com Bases de *Benchmark*

Mais uma vez, utilizou-se os *datasets* listados na Tabela 2 para conduzir este experimento. Nesta etapa, comparou-se os desempenhos dos métodos RP-IMA Dual, ELM Dual e SVM com *kernel* RBF. Para o método RP-IMA Dual, adotou-se os seguintes hiperparâmetros: $\eta = 0.1$, $\delta = 10^{-3}$ e T_MAX $= 10000$, além de se estabelecer um limite máximo de 20 iterações do algoritmo. A seleção do hiperparâmetro $\lambda$ para o RP-IMA Dual e ELM Dual foi realizada por meio de *grid search cross-validation*, onde foram testados os valores 0, 0.001, 0.01, 1 e 10. A mesma técnica foi utilizada para definição do parâmetro de regularização C da SVM. Já ao parâmetro $\sigma$, que define a largura de banda do *kernel* RBF, foi definido para todos os algoritmos de acordo com a regra de Silverman, expressa na Equação 4.3.

Para avaliar o desempenho dos algoritmos em diferentes arquiteturas de rede, testou-se os métodos RP-IMA Dual e ELM Dual com três quantidades diferentes de neurônios ocultos: N, N/2 e N/3, sendo N o número total de amostras em cada conjunto de dados, limitado a 1000.

Nesta seção, os modelos foram testados e tiveram seus desempenhos avaliados com base na métrica de acurácia. Os resultados foram apresentados em uma tabela que exibe os valores médios obtidos por meio de uma validação cruzada estratificada com 10 dobras. Os desvios padrão correspondentes estão indicados entre parênteses. Em cada iteração da validação cruzada, pesos idênticos foram gerados para os neurônios ocultos dos métodos RP-IMA Dual e ELM Dual.

A Tabela 15, exibe os resultados de acurácia. Adicionalmente, com base nas médias de acurácia obtidas por cada algoritmo em cada problema, gerou-se um gráfico *boxplot*, conforme ilustrado na Figura 19. O objetivo desse gráfico é permitir observar o desempenho médio dos algoritmos na resolução de todos os problemas abordados. Ao se analisar esse gráfico e também a Tabela 15 nota-se uma semelhança de comportamento de acurácia para os modelos RP-IMA Dual e ELM Dual com a mesma quantidade de neurônios.

Tabela 15 – Resultados de acurácia obtidos a partir de uma validação cruzada com 10 dobras.

| *Dataset* | N | Dual RP-IMA | | | Dual ELM | | | SVM |
|---|---|---|---|---|---|---|---|---|
| | | N | N/2 | N/3 | N | N/2 | N/3 | |
| IRI | 150 | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) | 100.00 (0.00) |
| SYN | 600 | 100.00 (0.00) | 100.00 (0.00) | 99.83 (0.50) | 100.00 (0.00) | 100.00 (0.00) | 99.83 (0.50) | 100.00 (0.00) |
| ROB | 117 | 75.98 (6.70) | 88.79 (8.07) | 95.61 (4.40) | 76.82 (5.97) | 87.88 (10.08) | 93.86 (8.11) | 87.12 (4.05) |
| MUS | 1000 | 100.00 (0.00) | 99.98 (0.05) | 100.00 (0.00) | 100.00 (0.00) | 99.98 (0.05) | 100.00 (0.00) | 99.72 (0.16) |
| ION | 351 | 86.59 (5.60) | 87.45 (3.69) | 86.60 (6.29) | 86.59 (5.60) | 87.45 (3.69) | 86.30 (6.88) | 95.16 (3.84) |
| BAN | 1000 | 99.85 (0.29) | 99.93 (0.22) | 99.93 (0.22) | 99.85 (0.29) | 99.93 (0.22) | 99.93 (0.22) | 100.00 (0.00) |
| WIN | 178 | 97.16 (2.85) | 97.16 (3.78) | 96.60 (3.73) | 97.75 (3.72) | 96.60 (4.48) | 97.16 (3.78) | 98.33 (3.56) |
| WBC | 569 | 95.60 (2.26) | 96.31 (1.84) | 96.83 (2.20) | 95.60 (2.26) | 96.31 (1.84) | 96.48 (2.24) | 98.42 (2.14) |
| SON | 208 | 84.62 (8.21) | 84.14 (8.35) | 81.74 (7.36) | 84.62 (8.21) | 84.14 (8.35) | 81.74 (7.36) | 69.26 (13.47) |
| SPA | 1000 | 91.20 (1.66) | 91.65 (1.52) | 92.07 (1.15) | 91.18 (1.72) | 91.85 (1.58) | 92.24 (1.14) | 93.15 (2.17) |
| TRA | 748 | 77.67 (2.27) | 78.87 (3.00) | 79.41 (3.26) | 79.01 (2.27) | 78.74 (2.46) | 79.27 (2.46) | 76.47 (0.84) |
| BCA | 683 | 94.87 (1.51) | 95.46 (1.53) | 95.61 (1.74) | 94.87 (1.51) | 95.46 (1.53) | 96.05 (2.08) | 97.23 (2.20) |
| ACR | 690 | 82.17 (4.10) | 82.90 (4.19) | 83.48 (4.68) | 82.17 (4.10) | 82.75 (4.07) | 82.75 (4.37) | 86.38 (3.50) |
| HAB | 306 | 74.56 (5.21) | 75.53 (5.93) | 73.88 (4.45) | 75.22 (5.67) | 75.19 (6.14) | 75.89 (6.49) | 75.51 (2.49) |
| HEA | 270 | 76.30 (7.26) | 76.67 (7.42) | 76.67 (6.43) | 76.67 (7.04) | 77.41 (8.19) | 78.52 (6.99) | 83.33 (5.04) |
| DIA | 768 | 69.67 (4.16) | 70.70 (4.52) | 68.36 (4.49) | 69.54 (4.09) | 70.97 (4.19) | 71.48 (3.82) | 77.47 (3.74) |
| MAM | 830 | 79.88 (5.90) | 81.33 (5.31) | 82.05 (5.71) | 79.88 (5.44) | 80.24 (5.53) | 81.33 (5.55) | 81.45 (4.64) |

Além disso, para ambos os métodos nota-se sinais de perda de desempenho por *overfitting* à medida que se aumenta a quantidade de neurônios da camada escondida da rede. Pela Figura, percebe-se que os modelos que obtiveram maior mediana de acurácia foram: SVM (93,15%) e ELM Dual e RP-IMA Dual com N/3 neurônios na camada escondida, cujos os valores de mediana foram respectivamente 92,24% e 92,07%. Já os modelos ELM Dual e RP-IMA Dual com maior quantidade de neurônios na camada

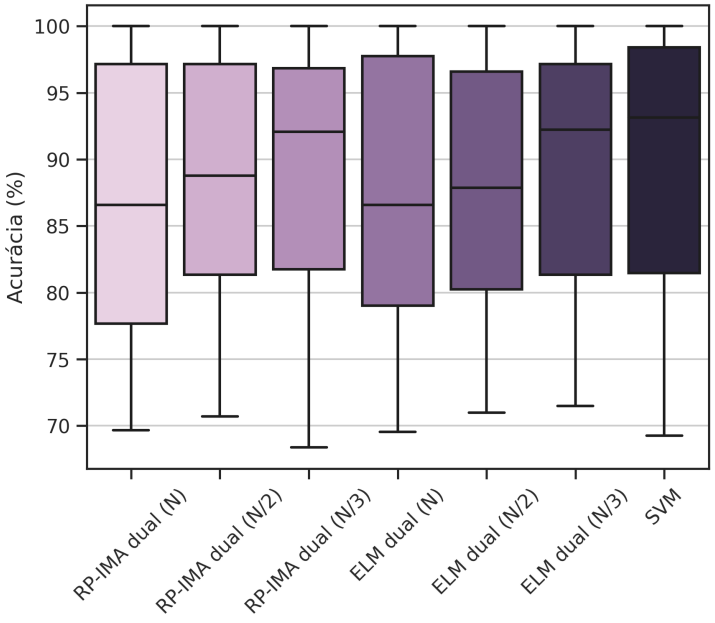


Figura 19 – Gráfico *boxplot* do comportamento médio de acurácia dos algoritmos RP-IMA dual, ELM Dual e SVM na resolução de todos os problemas.

escondida obtiveram menor mediana de acurácia (86.59%) e as maiores variabilidades.

Ao contrário das seções anteriores, aqui não pôde-se constatar uma melhoria significativa no desempenho de acurácia dos modelos ao buscar maximizar a margem na camada de saída. Uma explicação plausível para esse fenômeno pode ser que, em modelos com maior complexidade não linear, ou seja, com mais não linearidades (como é o caso do modelo dual apresentado), a busca pelo aumento da margem apenas na camada de saída pode não ser suficiente para aprimorar o desempenho global do modelo.

# Capítulo 5

# Conclusão

Através deste trabalho, pôde-se perceber mais uma vez que os classificadores de margem larga desempenham um papel relevante no campo do aprendizado de máquina. Eles têm a capacidade de criar fronteiras de separação mais robustas e eficazes entre diferentes classes, permitindo uma melhor generalização e desempenho de modelos.

A importância dos classificadores de margem larga reside na sua capacidade de lidar com dados ruidosos, variações e sobreposições entre classes, muitas vezes evitando o *overfitting* e fornecendo uma margem de erro aceitável para se obter uma classificação correta. Essa resistência ao *overfitting* foi observada tanto visualmente, quanto através de resultados experimentais para todos os três modelos de margem larga propostos neste trabalho.

As medições de margem rígida realizadas, mostraram que o os métodos RP-IMA primal, IM-RBFNN e RP-IMA dual foram capazes de obter margens (no espaço de características) significativamente maiores que as obtidas pelos algoritmos ELM, RBFNN e ELM dual, respectivamente.

A maioria dos resultados de acurácia obtidos para os modelos RP-IMA primal e IM-RBFNN revelam uma correlação entre a obtenção de uma margem larga no espaço de características de uma rede SLFN e o desempenho de classificação do modelo. Isso significa que a busca por estabelecer uma margem maior no espaço de características, reflete-se positivamente no desempenho de classificação dos modelos.

Contudo, não foi possível observar essa relação direta entre a obtenção de uma margem larga no espaço de saída da rede e os resultados de acurácia obtidos pelos modelos duais. O RP-IMA dual e a ELM dual apresentaram resultados de acurácia bastante semelhantes. Uma possível explicação para esse resultado pode estar relacionada ao fato de que essas redes possuem duas não linearidades, ao contrário dos outros modelos que apresentam apenas uma. Essa característica pode tornar a busca por uma margem larga apenas na camada de saída insuficiente para melhorar o desempenho dos modelos. Essas

não linearidades adicionais podem introduzir complexidade adicional ao modelo e exigir a busca por uma margem larga em todas as camadas da rede para otimizar o desempenho de classificação.

No entanto, é importante ressaltar que o desempenho dos modelos de rede neural é influenciado por diversos fatores, como a escolha das funções de ativação, a quantidade de neurônios nas camadas ocultas e a arquitetura geral do modelo. Portanto, uma análise mais aprofundada desses fatores pode ajudar a compreender melhor o comportamento e aprimorar o desempenho do modelo RP-IMA dual.

Outro ponto explorado por esse estudo foi a obtenção de soluções esparsas e a proposta de uma poda de neurônios de redes neurais do tipo SLFN. Existem diversas razões pelas quais soluções esparsas e a poda de neurônios são importantes, uma delas é eficiência computacional (modelos com menos neurônios requerem menor carga computacional necessária para armazenar e manipular o modelo). Outro motivo significativo é a interpretabilidade, soluções esparsas fornecem interpretações mais simples, visto que os coeficientes não nulos correspondem diretamente às características ou variáveis relevantes. Adicionalmente, a generalização é mais um estímulo para se buscar por soluções mais esparsas e arquiteturas com menos neurônios com pouca relevância, visto que soluções que concentram-se nas características ou parâmetros mais significativos podem auxiliar na mitigação da influência de dados ruidosos ou irrelevantes.

Para a maioria dos problemas testados, as formulações considerando a margem $L_\infty$ dos algoritmos RP-IMA e IM-RBFNN mostraram ser capazes de atingir soluções mais esparsas e, ainda assim, manter um desempenho similar à formulação dos métodos considerando a margem $L_2$.

Para trabalhos futuros, uma possível frente que ainda pode ser estudada é a poda de neurônios e seleção de amostras para o método RP-IMA dual. Essa abordagem é interessante para diminuir o custo computacional do método, bem como possivelmente melhorar seu desempenho. Além disso, outra possível melhoria para esse trabalho envolve o teste dos modelos propostos em bases de dados reais e com maior volume de dados.

Outro enfoque importante para futuras pesquisas é explorar diferentes configurações de hiperparâmetros e assim estudar a sensibilidade de cada hiperparâmetro dos métodos propostos. Isso possibilitará uma avaliação da robustez dos algoritmos face às variações em seus parâmetros.

Todas as implementações desenvolvidas durante este trabalho estão disponíveis no repositório `github.com/vcaitite/large-margin-classifiers-based-on-SLFN`.

# Referências

J. Arias-Garcia, A. Mafra, L. Gade, F. Coelho, C. Castro, L. Torres, e A. Braga. Enhancing performance of gabriel graph-based classifiers by a hardware co-processor for embedded system applications. *IEEE Transactions on Industrial Informatics*, 17(2):1186–1196, 2020.

K. P. Bennett e E. J. Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, volume 2000, pages 57–64. Citeseer, 2000.

C. M. Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

B. E. Boser, I. M. Guyon, e V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

A. d. P. Braga, T. B. Ludermir, e A. C. P. d. L. F. Carvalho. Redes neurais artificiais: teoria e aplicações. 2000.

C. Cortes e V. Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

T. Cover e P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964.

C. Domeniconi, D. Gunopulos, e J. Peng. Large margin nearest neighbor classifiers. *IEEE Transactions on Neural Networks*, 16(4):899–909, 2005. doi: 10.1109/TNN.2005.849821.

D. Dua e C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

G. Elsayed, D. Krishnan, H. Mobahi, K. Regan, e S. Bengio. Large margin deep networks for classification. *Advances in neural information processing systems*, 31, 2018.

Y. Freund e R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

K. R. Gabriel e R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic zoology*, 18(3):259–278, 1969.

C. Gentile. A new approximate maximal margin classification algorithm. In T. Leen, T. Dietterich, e V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/2000/file/d072677d210ac4c03ba046120f0802ec-Paper.pdf.

D. Goldfarb e M. J. Todd. Chapter ii linear programming. *Handbooks in Operations Research and Management Science*, 1:73–170, 1989.

S. Haykin e N. Network. A comprehensive foundation. *Neural networks*, 2(2004):41, 2004.

E. G. Horta. Aplicação de máquinas de aprendizado extremo ao problema de aprendizado ativo. 2015.

G.-B. Huang, Q.-Y. Zhu, e C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. Ieee, 2004.

G.-B. Huang, Q.-Y. Zhu, e C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.

S. C. Leite e R. F. Neto. Incremental margin algorithm for large margin classifiers. *Neurocomputing*, 71:1550–1560, 2008.

J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967*, pages 281–297, 1967.

J. v. Neumann, O. Morgenstern, et al. Theory of games and economic behavior. 1944.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

R. Semolini et al. Support vector machines, inferência transdutiva e o problema de classificação. *Campinas, SP*, 2002.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

L. J. Silvestre. Regularização de extreme learning machines: uma abordagem com matrizes de afinidade. 2014.

L. Torres, C. Castro, F. Coelho, F. S. Torres, e A. Braga. Distance-based large margin classifier suitable for integrated circuit implementation. *Electronics Letters*, 51(24): 1967–1969, 2015.

S. M. Villela, A. Xavier, e R. Neto. Seleção de caracter˜ sticas com busca ordenada e clpssificadores de larga margem. *Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computaçao*, 2011.

S. M. Villela, S. Leite, e R. F. Neto. Algoritmo de margem incremental com norma p para classificadores de larga margem. In *XI CBIC-Congresso Brasileiro de Inteligência Computacional, Porto de Galinhas, PE*, 2013.

S. M. Villela, S. de Castro Leite, e R. F. Neto. Incremental p-margin algorithm for classification with arbitrary norm. *Pattern Recognition*, 55:261–272, 2016.

K. Q. Weinberger e L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

S.-X. Zhang, C. Liu, K. Yao, e Y. Gong. Deep neural support vector machines for speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4275–4279, 2015. doi: 10.1109/ICASSP.2015.7178777.