

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
ENGENHARIA MECÂNICA – DEM**

MATHEUS JANCZKOWSKI FOGAÇA

**APLICAÇÃO DO MÉTODO LINEAR DISCRIMINANT ANALYSIS (LDA) EM
REDES NEURAS ARTIFICIAIS PARA DETECÇÃO DE DANO EM MATERIAL
COMPÓSITO**

JOINVILLE

2021

MATHEUS JANCZKOWSKI FOGAÇA

**APLICAÇÃO DO MÉTODO LINEAR DISCRIMINANT ANALYSIS (LDA) EM
REDES NEURAS ARTIFICIAIS PARA DETECÇÃO DE DANO EM MATERIAL
COMPÓSITO**

Trabalho apresentado ao Departamento de Engenharia Mecânica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Ricardo de Medeiros

JOINVILLE

2021

MATHEUS JANCZKOWSKI FOGAÇA

**APLICAÇÃO DO MÉTODO LINEAR DISCRIMINANT ANALYSIS (LDA) EM
REDES NEURAS ARTIFICIAIS PARA DETECÇÃO DE DANO EM MATERIAL
COMPÓSITO**

Trabalho apresentado ao Departamento de Engenharia Mecânica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Ricardo de Medeiros

BANCA EXAMINADORA:

Prof. Dr. Ricardo de Medeiros
Universidade do Estado de Santa Catarina

Membros:

Prof. Assoc. Pablo Andrés Muñoz-Rojas
Universidade do Estado de Santa Catarina

Me. Pedro Almeida Reis
Universidade do Estado de Santa Catarina

Joinville, 30 de agosto de 2021

Aos que nunca desistiram de pesquisar a
verdade através do método e do árduo esforço.

AGRADECIMENTOS

Nesta longa lista de agradecimentos, desejo começar por Deus, que, em sua infinita bondade, tem me agraciado com a força e persistência para buscar e seguir em frente com esta pesquisa, a qual, com Sua, benção rendeu frutos. Agradeço também às mulheres da minha vida - a minha *Matka*, a minha *Schatzi* e a minha *Oma Wiala* - pelo infindável apoio e terna compreensão ao longo de toda esta trajetória.

Agradeço enormemente aos Professores Ricardo de Medeiros e Eduardo Lenz, verdadeiros Mestres, que me orientaram no caminho deste trabalho e que, em verdade, tornaram tudo isto possível. Sou eternamente grato ao Professor Rafael Furlanetto pela forma como me fez ver a beleza da Matemática e como me imbuíu de uma visão crítica, ampla e filosófica da busca científica, sem a qual, nada disto teria tornado-se na boa videira.

RESUMO

Numa busca cada vez mais acelerada por meios tecnológicos que sejam capazes de monitorar e determinar a integridade física de componentes e estruturas, num ambiente cada vez mais dedicado à otimização e à procura de soluções menos onerosas, este trabalho surge para aplicar redes neurais artificiais no estudo da detecção de dano em material compósito a partir de dados de ensaio dinâmico, que consiste da função de resposta à frequência (FRF). Em específico, a contribuição principal é comparar a performance e aplicabilidade de dois métodos para compressão de dados - LDA (Linear Discriminant Analysis) e PCA (Principal Component Analysis), ou uma combinação dos dois. Para isto, é analisado o desempenho de um conjunto de redes neurais artificiais alternando entre métodos para reduzir a dimensionalidade do conjunto de dados. Foi utilizada a biblioteca Flux da linguagem de programação Julia para simular variações de hiperparâmetros, como função de ativação, topologia da rede e regime de classificação, para que fossem organizados arranjos destes para experienciar a variação na resposta aos métodos de compressão de dados. Foi desenvolvida metodologia que consiste na geração de 500 modelos de cada topologia para avaliar a convergência e geração de funções de massa de probabilidade (PMFs) de acurácia e erro no treinamento. Os quais corroboram uma abordagem quantitativa e justificam a determinação de qual é o melhor arranjo para cada problema classificatório, sendo este arranjo invariante à fração de amostras fornecidas durante o treinamento. De modo que a metodologia apresentada é eficaz como critério de criação e escolha entre topologias e configurações de hiperparâmetros. Bem como é mostrado que o PCA é superior ao LDA na aplicação deste trabalho. Por fim, são discutidas as potencialidades e limitações da metodologia para uso em sistemas de diagnóstico de falha.

Palavras-chave: Redes Neurais Artificiais. Material Compósito. Análise Modal. PCA. LDA. Machine Learning.

ABSTRACT

In an ever-accelerating search for technology means that can monitor and assess the physical integrity of components and structures, in an environment where optimization and less costly solutions are demanded, this work applies artificial neural networks to detect damage in composite materials, using data acquired through dynamic and modal analysis, which consists of frequency response functions (FRF). The main objective of this work is to compare the performance and applicability of two data compressing methods - LDA (Linear Discriminant Analysis) and PCA (Principal Component Analysis), or a combination of both. For this purpose, the performance of an array of artificial neural networks alternating among dimensionality reduction methods is analysed. The Flux library of the Julia programming language was used to simulate variations of hyperparameters, like activation function, network topology and classification regime, to set ensembles of them to track the variation of the response due to the data compression techniques. A methodology was developed, that consists of training 500 models of each topology, to evaluate the convergence of probabilistic mass functions (PMFs) of accuracy and loss during training. Which form a quantitative approach that justifies the determination of which array of hyperparameters is the best one for classification problem. As this study shows, the choice of hyperparameters is invariant with respect to the ratio of samples destined to training. Hence the presented methodology proves to be effective as a generative and choice criterion among topologies and different configurations. It shows the superiority as well of PCA above LDA on the application of this work. The potentialities and limitations of the methodology for the use in failure diagnose systems are also discussed.

Keywords: Artificial Neural Networks. Composite Material. Modal Analysis. PCA. LDA. Machine Learning.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação esquemática de um material compósito laminado com as orientações das camadas indicadas.	15
Figura 2 – Exemplo de discretização de uma viga engastada com três graus de liberdade e os três respectivos modos de vibrar.	16
Figura 3 – Representação esquemática do modelo de perceptron.	18
Figura 4 – Representação esquemática de uma rede neural 30/8/3/2. Os pontos vermelhos representam os neurônios; enquanto as linhas roxas, as ligações interneurais.	19
Figura 5 – Curvas, equações e derivadas de funções de ativação.	21
Figura 6 – Comparação do valor da função objetivo ao longo das iterações entre o otimizador Adam e seus pares.	25
Figura 7 – Visão esquemática de uma matriz de confusão.	28
Figura 8 – Exemplo de fronteiras de decisão lineares num conjunto de dados.	32
Figura 9 – Eixos principais num cluster.	34
Figura 10 – Casos em que a ortogonalidade do PCA induz ao erro.	36
Figura 11 – Caso em que a dados com pouca variância ao longo do eixo principal induzem o PCA ao erro.	37
Figura 12 – Diferenças entre as projeções utilizando PCA (à esquerda) e utilizando LDA (à direita).	40
Figura 13 – Vistas lateral e frontal das placas, com destaque para a região enegrecida pela presença da tira de Teflon.	42
Figura 14 – Representação esquemática da placa.	42
Figura 15 – FRF de uma amostra com dano D1.	43
Figura 16 – Parametrizações de topologias num gráfico cuja abscissa corresponde à posição de cada camada e a ordenada corresponde ao número de neurônios em cada camada.	49
Figura 17 – Exemplo de PMF de acurácia.	50
Figura 18 – Exemplo de PMF de erros no treinamento que converge para uma distribuição ao longo das 500 realizações no gráfico a) e a PMF já convergida em b). A ordem de magnitude está na legenda em a).	51
Figura 19 – Parametrizações das topologias com duas camadas interna, 3 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	66
Figura 20 – Parametrizações das topologias com duas camadas internas, 2 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	68

Figura 21 – Parametrizações das topologias com duas camadas internas, um neurônio de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	70
Figura 22 – Parametrizações das topologias com duas camadas internas, 3 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	72
Figura 23 – Parametrizações das topologias com duas camadas internas, 2 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	74
Figura 24 – Parametrizações das topologias com duas camadas internas, um neurônio de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	76
Figura 25 – Fronteiras de decisão do conjunto de dados utilizando a divisão de 75% para treinamento e utilizando apenas o LDA e dois discriminantes lineares. . . .	77
Figura 26 – Fronteiras de decisão do conjunto de dados utilizando a divisão de 75% para treinamento e utilizando PCA seguido de LDA e dois discriminantes lineares. . . .	77

LISTA DE TABELAS

Tabela 1 – Casos de classificação e a resposta do neurônio utilizando apenas um neurônio na camada de <i>output</i>	44
Tabela 2 – Casos de classificação e a resposta dos neurônios utilizando dois neurônios na camada de <i>output</i>	44
Tabela 3 – Casos de classificação e a resposta dos neurônios utilizando quatro neurônios na camada de <i>output</i>	44
Tabela 4 – As quatro divisões do conjunto de dados original e suas respectivas frações de amostras para cada <i>data set</i>	46
Tabela 5 – Características e estatísticas de desempenho de topologias com resultados notáveis.	55
Tabela 6 – Topologias com uma camada interna e 2 neurônios de output. PCA somente foi usado para compressão dos dados.	64
Tabela 7 – Topologias com uma camada interna e 4 neurônios de output. PCA somente foi usado para compressão dos dados.	64
Tabela 8 – Topologias com duas camadas internas, 3 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	65
Tabela 9 – Topologias com duas camadas internas, 2 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	67
Tabela 10 – Topologias com duas camadas internas, um neurônio de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.	69
Tabela 11 – Topologias com duas camadas internas, 3 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	71
Tabela 12 – Topologias com duas camadas internas, 2 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	73
Tabela 13 – Topologias com duas camadas internas, um neurônio de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.	75

SUMÁRIO

1	INTRODUÇÃO	12
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	MATERIAIS COMPÓSITOS	14
2.2	FRF E ANÁLISE MODAL	15
3	REDES NEURAIS ARTIFICIAIS	17
3.1	MODELO DE PERCEPTRON	17
3.2	REDES	19
3.3	FUNÇÃO DE ATIVAÇÃO	21
3.4	FUNÇÃO CUSTO	22
3.5	OTIMIZADORES	24
3.6	DIVISÃO DO CONJUNTO DE DADOS ORIGINAL	26
3.7	MÉTRICAS DE DESEMPENHO	27
4	MÉTODOS DE REDUÇÃO DE DIMENSIONALIDADE E CLASSIFICAÇÃO	29
4.1	MÉTODOS LINEARES	30
4.2	MÉTODOS CLASSIFICADORES	31
4.3	ANÁLISE DO COMPONENTE PRINCIPAL	34
4.4	ANÁLISE DE DISCRIMINANTES LINEARES	37
4.4.1	Fronteiras de decisão de LDA	40
5	MATERIAIS E MÉTODOS	41
5.1	PROCEDIMENTO EXPERIMENTAL	41
5.1.1	Desenvolvimento das placas de material compósito	41
5.1.2	Ensaio de vibrações	41
5.2	DEFINIÇÃO DO PROBLEMA DE CLASSIFICAÇÃO	44
5.3	CONFIGURAÇÃO DE HIPERPARÂMETROS	45
5.3.1	Hiperparâmetros fixos	45
5.3.2	Hiperparâmetros variáveis	45
5.3.2.1	<i>Divisão do conjunto de dados</i>	45
5.3.2.2	<i>Aplicação dos métodos compressivos</i>	46
5.3.2.3	<i>Configuração das topologias</i>	47
5.3.2.4	<i>Funções de ativação</i>	49
5.3.2.5	<i>Realizações</i>	49
5.3.2.6	<i>Pontuação</i>	50
6	RESULTADOS	52
7	CONCLUSÃO	58

REFERÊNCIAS	60
ANEXO A – TOPOLOGIAS SIMULADAS	64
ANEXO B – FRONTEIRAS DE DECISÃO	77

1 INTRODUÇÃO

A indústria do século 21 e, por consequência, a Engenharia Mecânica estão numa busca cada vez mais acelerada por técnicas e métodos que permitam construir e manter peças e sistemas funcionando com a melhor performance possível, *no estado da arte*, e com o menor custo. Nesta procura por otimização, destaca-se o aumento da razão entre resistência mecânica e massa, que é uma linha de pesquisa de elevada importância na indústria aeronáutica e aeroespacial. Sendo que os materiais compósitos são fronteira de avanço para esta questão e já vem sendo utilizados, segundo McMANUS (2021), para projeto de fuselagem de aeronaves BWB (Blended Wing Body), ou seja, em que o corpo do avião é a própria asa. Isto ocorre pois compósito de fibra de carbono com resina polimérica, por exemplo, tem a capacidade de resistir a gradientes de pressão muito maiores em grandes altitudes, ao passo que apresenta uma baixa densidade e pode substituir as estruturas tradicionais, como as feitas em alumínio.

A problemática dessas estruturas otimizadas e dos materiais compósitos reside na predição e prevenção da falha, uma vez que o excesso de material foi extinto e que esses materiais, conforme Karsh, Mukhopadhyay e Dey (2018), tem mecanismos de falha diferentes e complexos, como a delaminação. Assim, surge a necessidade de métodos que possam monitorar esses componentes periódica ou continuamente e, na sequência, outros métodos que sejam capazes de interpretar e avaliar os dados coletados numa resposta classificatória a respeito do estado dessas peças - se intactas ou se danificadas e quanto danificadas.

Como a falha pode ser indetectável na inspeção visual, seja realizada pela visão humana seja com o uso de câmeras ou outras tecnologias dependentes de algum tipo de radiação, o dano pode ser encontrado e medido de forma indireta, isto é, pela medição de algum comportamento do sistema que seja influenciado pela presença de alguma condição não tolerada do ponto de vista de engenharia. Um meio particularmente interessante de avaliar isso é pela medição das vibrações da peça quando submetida a alguma excitação, posto que o modo de vibrar é função das características geométricas e físicas. Matematicamente, esta avaliação pode ser feita com o emprego de função de resposta em frequência (FRF, *frequency response function* em inglês), que mede a amplitude da oscilação no domínio da frequência.

A simples construção dessa função ainda não permite determinar a integridade de um objeto, visto que as diferenças no comportamento à vibração entre peças iguais, intactas ou com danos distintos, podem ser sutis e não-lineares e, portanto, de difícil avaliação direta ou com algoritmos fixos. Há, neste sentido, a demanda por um método cuja tomada de decisão e classificação seja capaz de *aprender* e moldar-se a um conjunto de dados para inferir previsões em outros, de modo que seja capaz de notar sutilezas e superar não-linearidades. Essas habilidades podem ser encontradas em Redes Neurais Artificiais (RNA), que são constituídas por diversos pontos, chamados de neurônios, que recebem informação, realizam uma operação matemática e enviam esta nova unidade de informação para os neurônios subsequentes, numa forma semelhante ao que acontece nas conexões neurais de um cérebro. Desta forma, o conjunto organizado de

neurônios consiste numa função que, dado um conjunto de dados de entrada, gera uma resposta numérica. Esta técnica já foi utilizada para previsão de falha, (JUNIOR; ALMEIDA; GOMES, 2020); para detecção de delaminação em compósito, (GOMES et al., 2019); para detecção de trinca, (PEREIRA et al., 2020).

Logo, percebe-se que as RNAs são capazes de lidar com problemas de diferentes naturezas e que apresentam não-linearidades, e que são aptas para o escopo de engenharia que será avaliado neste trabalho. Há, no entanto, uma miríade de opções de configurações de parâmetros *estruturais* da *arquitetura* de uma rede neural artificial, parâmetros estes que são denominados de *hiperparâmetros*, que informam como o modelo é construído e organizado; como os neurônios serão conectados; como a rede será otimizada, entre outras questões pertinentes e importantes para que o método seja efetivo e acurado.

É justamente nesta questão - configuração de hiperparâmetros - em que está o cerne desta pesquisa, uma vez que se deseja buscar o melhor arranjo para que a função construída pela rede seja capaz de gerar as maiores acurácias possíveis em diferentes conjuntos de dados. Nesse âmbito, encontra-se o ajuste do número de neurônios na camada de entrada da rede neural artificial, camada esta que se comporta como o receptor do domínio da função. Evidencia-se que este arranjo tem um grande impacto sobre a performance do modelo, posto que há um balanço entre o número mínimo de variáveis que um conjunto de dados precisa para ser coerentemente expresso e um número máximo que levará à dificuldades na otimização da rede e na sua baixa capacidade de generalização. Dessa forma, como os dados obtidos experimentalmente têm muitas variáveis, que são as discretizações da FRF, há a necessidade de uma técnica que seja capaz de diminuir a dimensionalidade do conjunto de dados sem que uma quantidade substancial de informação seja perdida no processo, de modo que a rede seja ainda capaz de realizar a sua atividade classificatória sem o prejuízo da grande dimensionalidade.

Para que esta tarefa seja cumprida, existem diversos métodos. Focar-se-á, porém, em dois deles - PCA (*Principal Component Analysis*, do inglês para Análise do Componente Principal) e LDA (*Linear Discriminant Analysis*, do inglês para Análise do Discriminante Linear), que são exemplos conhecidos e estabelecidos de técnicas lineares, ou seja, que lançam mão de transformações lineares para realizar a redução de dimensão. Existe, contudo, mesmo na literatura especializada, poucos exemplos de estudos que tenham conduzido um escrutínio para averiguar em qual domínio uma técnica se sobrepõe a outra. Sendo que este será o objetivo deste trabalho: desenvolver uma metodologia que seja capaz de avaliar qual técnica deve ser utilizada com determinado tipo de dado ou se pode ser usada uma simbiose de ambas, de modo que uma seja empregada na sequência da outra.

Este trabalho será conduzido sobre um conjunto de dados retirados do ensaio de vibração na condição livre-livre de placas de compósito de fibra de vidro com matriz de resina epóxi, experimento este que foi feito no trabalho de Reis (2020). Destaca-se, em consequente, que este trabalho é capaz de investigar nos conceitos discorridos - materiais compósitos; teste por ensaio de vibrações; redução de dimensionalidade e otimização de hiperparâmetros de RNAs.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo descrever e apresentar ao leitor os conceitos que definem a parte experimental da pesquisa. Abordar-se-á brevemente desde os materiais compósitos até a fundamentação teórica dos ensaios de vibração e da FRF.

2.1 MATERIAIS COMPÓSITOS

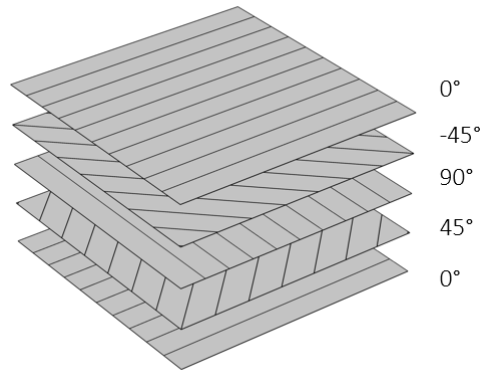
Materiais compósitos, muitas vezes chamados apenas de compósitos, são aqueles constituídos de uma estrutura coesa, porém não homogênea, de dois ou mais materiais de naturezas distintas (PUTTEGOWDA; RANGAPPA; JAWAID, 2018). Os compósitos podem dividir-se basicamente em dois constituintes - fibra e matriz -, sendo primeira a que costuma garantir a maior parte da resistência mecânica prometida pelo material; e segunda responsável por manter as fibras no lugar e na ordenação desejada, ou seja, é a matriz que liga uma fibra à adjacência, destaca-se que é a matriz que costuma trazer características secundárias de interesse, como menor densidade, resistência à corrosão, entre outras.

Uma das formas de organização das fibras e da matriz, de particular interesse para este trabalho, é o laminado, em que as fibras são dispostas paralelas num plano e o espaço restante entre aquelas é preenchido com a matriz. Esses planos, conhecidos por camadas ou lâminas, são dispostos uns sobre os outros, por quantas camadas forem requeridas pelo projeto. Atenta-se para o fato que, tomando o material compósito final, pode-se determinar um sistema de referência global e, a partir deste, prescrever a orientação de cada camada, ou seja, o ângulo de direção das fibras em relação à referência, a exemplo da Fig. 1.

Seguindo a hipótese de carregamento coplanar a cada lâmina e de que a fibra seja delgada e mais resiste mecanicamente que a matriz, é possível perceber que a direção de maior resistência à tração será a paralela à direção da fibra. A natureza não homogênea do compósito, por outro lado, induzirá a diferentes respostas de deformação, assim, pode-se esperar relações constitutivas além da isotropia, e relações estas dependentes da configuração das orientações das camadas, de modo que simetrias constitutivas de interesse podem ser obtidas pela manipulação do arranjo de orientações das lâminas (LEAL, 2005).

No âmbito da falha, pode-se citar diversos casos particulares. O foco, entretanto, será na delaminação, que é o objetivo de estudo neste trabalho e que é considerado o modo de falha mais crítico para compósitos (SRIDHARAN, 2008). A delaminação ocorre quando duas camadas adjacentes se separam em algum ponto por ação de carregamentos cisalhantes extremos ou por defeitos de fabricação, como corte e furação (REIS, 2020). Um dos fatores que levam ao perigo intrínseco a esse mecanismo de falha é a baixa visibilidade.

Figura 1 – Representação esquemática de um material compósito laminado com as orientações das camadas indicadas.



Fonte: Soami (2018)

2.2 FRF E ANÁLISE MODAL

Seja um sistema oscilante definido como o conjunto de um ou mais componentes que é capaz de oscilar mecanicamente quando excitado por uma força. A vibração, ou seja, esta oscilação, pode ser percebida pela mudança de posição, de velocidade ou da aceleração de um ponto numa coordenada no espaço, sendo que cada variação na direção de uma coordenada é definida como grau de liberdade (KELLY, 2012). Um sistema como as placas de compósito utilizadas neste trabalho é classificado como contínuo, pois pode ter infinitos graus de liberdade ao longo de toda a área. Para fins de experimentação e de modelagem, por outro lado, é deveras importante que uma simplificação seja feita, de modo que a complexidade de infinitos graus de liberdade do fenômeno possa ser reduzida para alguns graus convenientemente posicionados ao longo da placa. Processo este de simplificação conhecido como discretização, o qual é ilustrado para uma viga engastada na Fig. 2.

O fenômeno de vibrações pode ser modelado utilizando uma equação diferencial linear de segunda ordem. Essa equação na forma para mais de n graus de liberdade e em sua construção matricial, logo a variável dependente, $x(t)$, é um vetor de deslocamentos, isto é, na j -ésima posição desse vetor, está o deslocamento em função do tempo, t , no j -ésimo grau de liberdade. O qual é a solução para

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \vec{\mathbf{F}}(t), \quad (1)$$

ao passo que \mathbf{M} , \mathbf{C} e \mathbf{K} representam as matrizes de massa, de amortecimento e de rigidez do sistema; enquanto $\vec{\mathbf{F}}(t)$ é um vetor das forças presentes em cada grau de liberdade.

Conforme Kelly (2012), a equação (1) pode ser resolvida de duas formas: se a matriz de amortecimento puder ser representada como uma combinação linear das matrizes de massa e de rigidez, pode-se usar amortecimento proporcional para simplificar e desacoplar a equação em

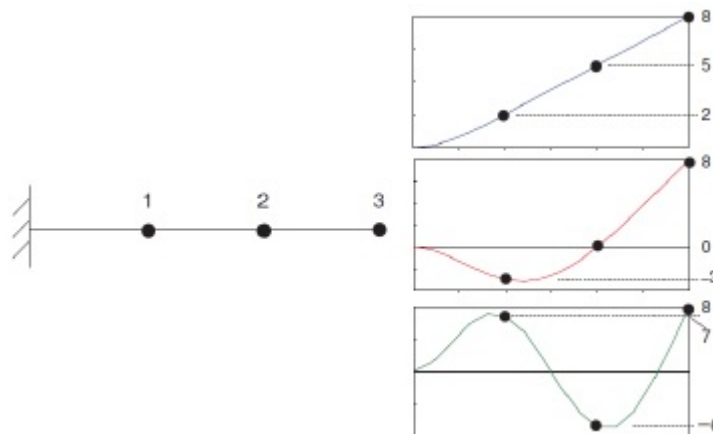
questão; se C não for passível de representação por essa combinação linear, cabe o uso de outra mudança de variável para o vetor $x(t)$. De qualquer forma, a solução será em termos da solução do problema de autovetores de $M^{-1}K$, e esse é um resultado importante, pois esses autovetores formam uma matriz que é capaz de mudar a base de (1) para uma base em que as matrizes M , C e K sejam diagonais e, portanto, os graus de liberdade sejam representados em termos de uma base em que estão desacoplados entre si. Cada autovetor também determina um modo de vibrar, que é o princípio da análise modal.

A solução comentada anterior é dada em função do tempo, já que a variável independente em (1) é o tempo, t . Esse domínio pode ser modificado para o da frequência, por meio da Transformada de Fourier. E essa transformação é de suma importância, pois, assim, é possível identificar frequências de ressonância, que são críticas na integridade estrutural de um sistema e também fornecem informações sobre os modos de vibrar e sobre características físicas do conjunto, que é justamente o interesse. Logo a função de resposta à frequência, FRF, nesse domínio, pode ser definida, segundo Grenier (2020), como

$$h = \frac{x_i(\omega)}{F_j(\omega)} = H_{i,j}, \quad (2)$$

em que h é a FRF; $x_i(\omega)$, o deslocamento no i -ésimo grau de liberdade; $F_j(\omega)$, a força de excitação no j -ésimo grau de liberdade. Ao passo que H é a matriz de todas as FRFs, de modo que, $H_{i,j}$ corresponde à FRF no i -ésimo grau de liberdade considerando a excitação no j -ésimo grau de liberdade. Destaca-se que a FRF pode ser definida como o quociente da velocidade ou da aceleração pela excitação; não apenas pelo deslocamento. Consoante a Grenier (2020), a instrumentação da experimentação é fundamental para que os resultados de (2) tenham repetibilidade satisfatória.

Figura 2 – Exemplo de discretização de uma viga engastada com três graus de liberdade e os três respectivos modos de vibrar.



Fonte: Avitabile (2017)

3 REDES NEURAIIS ARTIFICIAIS

Embora a ordem natural na sequência do procedimento experimental seja a redução da dimensionalidade dos dados coletados, é de interesse que o conceito de Redes Neurais Artificiais seja aprofundado antes, para que as razões para a compressão dos dados seja apresentada e compreendida, uma vez que os métodos de compressão são o objetivo principal e final deste trabalho, de maneira que a contribuição deste é a compreensão da fronteira de aplicação de cada método ou de combinações destes. Logo, este capítulo abordará dos parâmetros aos hiperparâmetros e os conceitos de RNA.

As redes neurais artificiais são modelos matemáticos, uma desambiguação aqui, porém, deve ser traçada: neste trabalho e na literatura de forma geral, especialmente em língua inglesa, *modelo* corresponde a cada rede neural individualmente treinada, visto que uma arquitetura de rede com uma composição de hiperparâmetros pode ser treinada por n vezes, gerando, pois, n modelos. E, como explicado anteriormente, as redes neurais artificiais são funções que ligam um domínio a um contradomínio numéricos. Dessa forma, as informações devem ser convertidas por algum artifício de lógica numa informação puramente numérica.

As RNAs são inspiradas nos exemplos naturais de conexões neurais, uma vez que há um mimetismo na construção e organização, de modo que os pesos de cada neurônio artificial imitam os elementos químicos na sinapse de um neurônio natural (AGGARWAL, 2018). Isso ocorre pois, da mesma forma que as substâncias numa sinapse induzem um sinal elétrico no neurônio, a informação anterior associada dos pesos produz um *input*, uma entrada, para a função de ativação do neurônio na RNA.

3.1 MODELO DE PERCEPTRON

Segundo Reis (2020), o modelo de perceptron foi proposto em 1958 por Rosenblatt, e consiste de um ente, chamado de neurônio, que recebe um vetor de informações de input e gera um número como seu *output* utilizando a seguinte regra

$$y = P(\vec{x}) = f(\vec{x}^T \vec{w} + b), \quad (3)$$

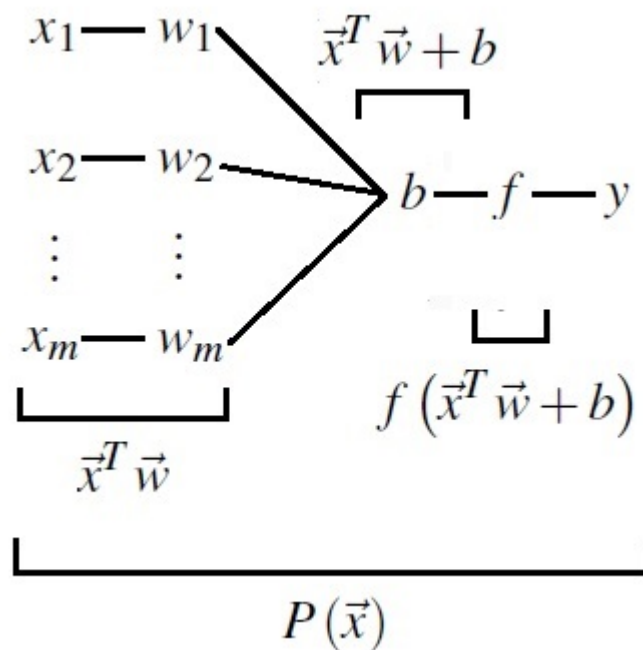
tal que y é o *output* do neurônio; P , a função construída com o neurônio; f , a função de ativação; \vec{x} , o vetor de *input*; \vec{w} , o vetor de pesos; b , o *bias* (do inglês para viés). A partir da equação (3), percebe-se que a função, ou o modelo, construído com o neurônio correlaciona um domínio de dimensão, m , do vetor de *input*, \vec{x} , com um contradomínio de dimensão unitária, logo $P : A^m \rightarrow A$, em que A é o conjunto numérico de interesse na aplicação. Na Figura 3, pode-se ver esquematicamente como ocorre o processo de cálculo descrito matematicamente em (3), sendo que, em cada colchete, está uma das operações presentes nessa equação.

O vetor de pesos e o bias são constantes e são uma característica do neurônio. Como a informação que entra neste é uma combinação linear do *input* com o vetor de pesos, representada

pela multiplicação em (3), os pesos podem ser interpretados como uma importância ou uma modulação da informação precursora; ao passo que o bias pode amplificar ou atenuar o sinal recebido em sua totalidade. Ambos são ajustados por um processo que reduz o erro da resposta da rede sobre e comparada com um conjunto de dados de *treinamento*. Justamente por esse ajuste na fase de treinamento, os pesos e o bias são denominados como *parâmetros*, para que sejam distinguidos dos *hiperparâmetros*, que são configurações mais gerais da arquitetura da rede neural e que não são modificadas na fase de treinamento.

Já a função de ativação, f , é do tipo $f : A \rightarrow A$, em que A é o mesmo conjunto numérico descrito anteriormente. Esta função tem o objetivo de modular o resultado da operação com os pesos e da adição do bias, posto que a imagem da função de ativação é a imagem, isto é, o *output* do neurônio. Assim, f pode ser escolhida convenientemente para que o comportamento da resposta seja previsível ou esteja confinada num intervalo. Conforme Silva, Spatti e Flauzino (2010b), as funções de ativação podem ser divididas entre as totalmente diferenciáveis e as parcialmente diferenciáveis, ou seja, em qual é a extensão do intervalo em que se pode derivar coerentemente a função. Essa questão é de suma importância, posto que grande parte dos algoritmos de otimização são dependentes do cálculo do gradiente, como será visto ao longo deste texto na seção sobre otimizadores.

Figura 3 – Representação esquemática do modelo de perceptron.



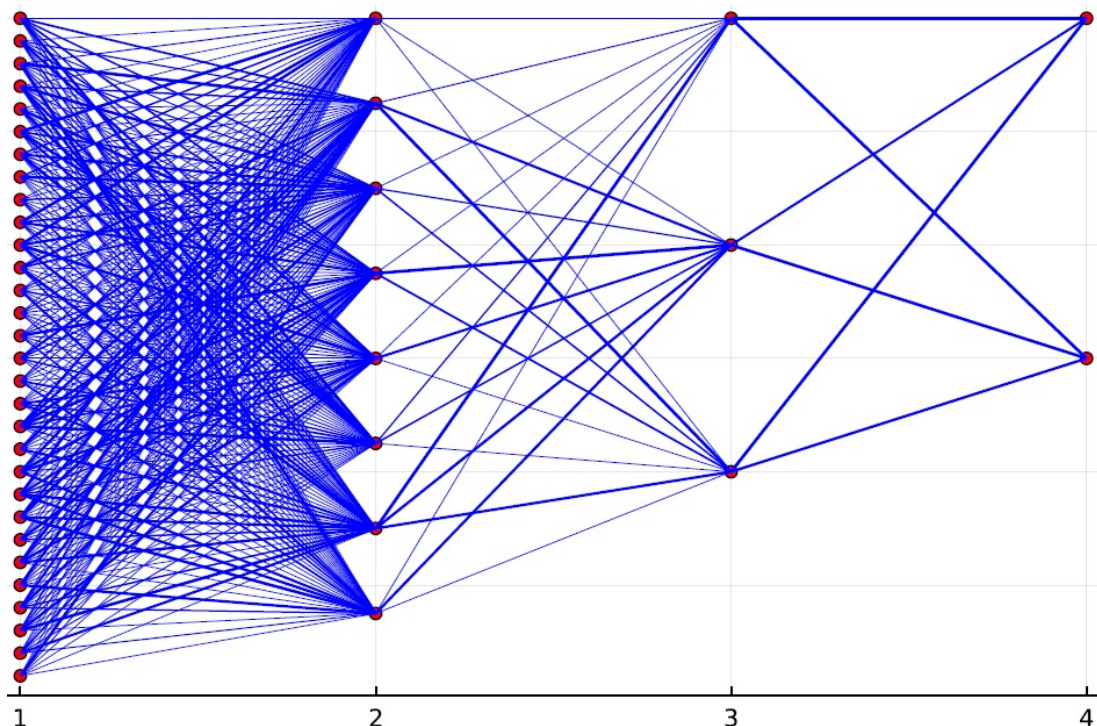
Fonte: Produção do autor.

3.2 REDES

O problema XOR (*OR-exclusive*, do inglês para ou e exclusivo), que consiste em uma resposta positiva para entradas diferentes e uma resposta negativa para entradas iguais, que é um caso específico do problema de classificar pontos num hipercubo. Este é um exemplo de problema em que as classes não são linearmente separáveis, pois uma única fronteira de decisão linear não é capaz de separar as duas classes. Segundo Haykin (2007), um único neurônio não é capaz de superar esta não linearidade e, portanto, para que se encontre uma solução, um neurônio a mais deve ser colocado *na sequência* do neurônio já existente, para que um forneça a não-linearidade necessária ao classificador, ou seja, uma rede foi formada. Em verdade, um tipo de rede específico - com mais de uma camada interna.

Numa rede neural artificial, como já explicado, a informação *flui* da entrada até a saída, sendo que, na primeira, a informação crua é fornecida e, na segunda, o resultado da classificação é obtido. Os neurônios que compõe essa rede, contudo, podem ser divididos e agrupados em segmentos transversais, de forma que os neurônios de uma camada ligam-se com os de outra. Percebe-se, em conseqüente, que o modelo unidimensional da seção anterior ganhou mais uma dimensão - os neurônios podem ser divididos em fatias ao longo da direção de propagação da informação e cada fatia pode ter um ou mais neurônios. Por conveniência, a propagação de informação é simbolizada por um eixo horizontal, enquanto que os neurônios numa mesma camada são dispostos nela paralelamente ao eixo vertical, como se pode verificar na Fig. 4.

Figura 4 – Representação esquemática de uma rede neural 30/8/3/2. Os pontos vermelhos representam os neurônios; enquanto as linhas roxas, as ligações interneurais.



Fonte: Produção do autor.

Tendo sido explicado o conceito de *camada*, é de interesse que haja uma diferenciação entre dois tipos de camadas: as que estão na extremidade e as intermediárias. As primeiras podem ser melhor denominadas de *camada de input* e de *camada de output*, o número de neurônios naquela é determinado pela dimensionalidade do conjunto de dados; enquanto o número de neurônios da camada de *output* é definido pela forma com que é parametrizada a classificação, como veremos posteriormente neste trabalho. As camadas intermediárias, no entanto, são uma questão a parte - tanto a quantidade delas quanto o número de neurônios em cada é algo que depende de quem constrói a rede neural e configuram, portanto, esses dados como hiperparâmetros.

A *arquitetura* ou a *topologia* são termos que nomeiam uma configuração que informa o número de camadas de uma rede neural, a quantidade de neurônios em cada uma delas e como essas camadas interligam-se. As topologias podem ser classificadas em grupos, dos quais existem muitos, mas, neste trabalho, o foco será em arquiteturas: FF (*Feed-Forward*, que são topologias em que o número de neurônios em cada camada diminui monotonicamente ou permanece constante da camada de *input* para a de *output*, isto é, a rede consegue ser encapsulada num retângulo se for plotada como na Fig. 4, de modo que a altura desse retângulo é determinada pela quantidade de neurônios na camada de *input*; e DFF (*Deep Feedforward*), que são aquelas em que o número de neurônios numa camada intermediária pode ser maior que o das extremidades. Essas duas famílias de topologias produzem resultados muito diferentes, haja vista ao fato de como dividem não-linearidades e ligações especializadas, os *branches*. Para que uma arquitetura seja representada por texto, convêm-se utilizar a forma $\rho_1/\rho_2/\dots/\rho_K$, em que ρ_i representa o número de neurônios na *i*-ésima camada, lembrando-se que *i* é um contador no sistema global de contagem de camadas, de modo que as camadas de *input* e de *output* são contadas também e na ordem de sua posição, isto é, a de *input* em $i = 1$ e a de *output* em $i = K$, sendo *K* o número total de camadas da rede neural artificial.

Na literatura, encontra-se uma confusão em torno dos termos DFF e FF, sendo trocados entre si muitas vezes. Neste texto, entretanto, a terminologia será usada como descrita acima, uma vez as duas famílias dividem a característica da informação seguir sempre na direção *input-output*, mas a DFF pode ter um alargamento do número de neurônios nas camadas intermediárias, o que faz uso do termo *deep*, pois esta topologia aprofunda-se na direção transversal, assim tendo a semântica inerente à terminologia preservada e satisfeita.



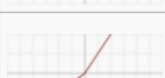

Ainda no tema das topologias, existem diversas formas de uma camada ligar-se com as outras; neste trabalho, porém, o foco será sobre topologias do tipo *densa*, isto é, aquelas em que todos os neurônios da *k*-ésima camada conectam-se com todos os neurônios da (*k*-1)-ésima camada. Este tipo de conexão torna o cálculo do gradiente no processo de otimização computacionalmente custoso, já que todas as conexões entre neurônios de camadas adjacentes são ativas. A vantagem, no entanto, está que essa enorme quantidade de ligações ativas permite a visualização da qualidade das ligações no contexto geral da rede e permite a comparação entre o uso de diferentes números de camadas internas.

3.3 FUNÇÃO DE ATIVAÇÃO

As funções de ativação são outro hiperparâmetro e são uma característica intrínseca à rede neural, posto que são elas que controlam o intervalo em que se encontra o *output* do neurônio. Há uma extensa gama de funções, porém, serão abordados aqui as opções que melhor se enquadraram nesta pesquisa, que são a sigmóide; a Leaky ReLU (*Parametric Rectified Linear Unit*, do inglês para unidade linear retificada paramétrica) e a ELU (*Exponential Linear Unit*, do inglês para unidade linear exponencial). Além do comportamento diferenciável ou não que já foi descrito, as funções de ativação também podem ser lineares, como a ReLU (*Rectified Linear Unit*, do inglês para unidade linear retificada), a Leaky ReLU e em parte da ELU, cujas linearidades podem ser observadas nas curvas apresentadas na Fig. 5. As funções de ativação também podem restringir o intervalo de *output*, como o exemplo clássico da sigmóide, em que aquele está entre 0 e 1; há, entretanto, casos em que apenas um segmento do domínio da função tem a imagem restrita, como é o caso do segmento negativo dos domínios da ReLU e da ELU.

Antes que uma ou mais funções de ativação sejam escolhidas, é fundamental observar em que parte do domínio o *input* está disperso e qual é a distância entre os centróides desses *inputs*. De modo que, quando há uma proximidade muito grande, deve-se optar por funções que sejam capazes de separar os *outputs*; ao passo que, quando há uma distância considerável, pode-se optar por funções que restrinjam o intervalo. Outra consideração é por funções que não ignorem diversos *inputs* num mesmo segmento, como a ReLU faz com todos os valores negativos, que é uma das causas da dificuldade de propagação de informação ao longo da rede neural, conforme Sharma (2017), e causa também da inabilidade de o processo classificatório ser bem sucedido, posto que parte da informação necessária é ignorada ou tratada de forma igualitária.

Figura 5 – Curvas, equações e derivadas de funções de ativação.

Sigmóide		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
ReLU		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky ReLU		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
ELU		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Fonte: (SHARMA, 2017)

Uma função muito utilizada é a *softmax*, que não deve ser confundida com uma função de ativação ordinária, pois não é utilizada em cada neurônio de uma camada, e, sim, em todos os neurônios da camada (geralmente de *output*) numa única vez. A *softmax* calcula uma distribuição

de probabilidade dos *inputs*, de forma que cada *input* é normalizado para a sua probabilidade por

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^m e^{x_j}}, \quad (4)$$

em que os termos x representam as componentes do vetor de *input* da função softmax e os termos y correspondem às componentes do vetor de *output* daquela. Percebe-se, pois, que a soma dos *outputs* é unitária. Esta função é muito utilizada para problemas de classificação com duas ou mais classes, em que a dimensão do *output* é igual ao número daquelas e, portanto, os componentes da representação da classe serão restritos entre 0 e 1.

Por uma questão construtiva da biblioteca Flux da linguagem de programação Julia, as ligações entre duas camadas devem ter todas a mesma função de ativação.

3.4 FUNÇÃO CUSTO

Estando os conceitos da rede neural artificial e de sua construção bem sedimentados, é o momento de definir uma métrica que seja capaz de medir o erro da classificação em relação à resposta real. A função custo, ou de erro, faz exatamente isso por meio de uma lei de função, lei esta que é mais um hiperparâmetro a ser definido. Como a imagem do modelo de RNA é numérica e como pode ser interpretada como um vetor, a intuição leva inicialmente à aferição da distância euclidiana entre o vetor resultado do modelo e o vetor classificação real. Em verdade, muitas funções seguem nesse caminho ou em variações deste, como é o caso da MAE (*Mean Absolute Error*, do inglês para erro absoluto médio) e da MSE (*Mean Squared Error*, do inglês para erro quadrático médio). As equações para esses erros, segundo Krzyk (2018), são definidas da seguinte forma

$$E_{MAE} = \frac{1}{m} \sum_{j=1}^m |y_{p,j} - y_{r,j}|, \quad (5)$$

$$E_{MSE} = \frac{1}{m} \sum_{j=1}^m (y_{p,j} - y_{r,j})^2, \quad (6)$$

tal que E está para o erro; \mathbf{y}_p , o vetor de predições; \mathbf{y}_r , o vetor esperado e real; e o índice após a letra que caracteriza se predição ou se esperado informa a componente dos vetores em análise. Vê-se, por exemplo, que, se for efetuada a raiz quadrada do somatório da equação (6), ter-se-á a distância euclidiana entre a previsão do modelo e o ponto que representa a realidade.

Há, no entanto, outras formas de se calcular o erro que se baseiam em ideias estatísticas e da teoria da informação; não na geometria euclidiana. Uma dessas propostas, segundo Brownlee (2019), e que é muito usada para RNAs no âmbito de classificação é a *cross-entropy* (do inglês

para entropia cruzada), que mede a entropia entre duas distribuições de probabilidade discretas (DPD). O objetivo dessa função custo é medir o quanto a DPD da previsão é diferente daquela do vetor de classificações esperadas. Para que esta métrica seja consistente, é aplicada a entropia de informação, que equivale ao logaritmo da probabilidade de um evento acontecer, e o logaritmo é usado para que os eventos mais raros (com probabilidade menor) sejam mais penalizados, (BROWNLEE, 2019). Logo, para que o *output* da rede neural seja uma DPD, aquele deve ser transformado numa inicialmente, o que já acontece caso a função softmax, como na equação (4); caso esta não seja utilizada naturalmente na estrutura da RNA, porém, deve-se aplicar (4) para que a natureza do vetor de *output* seja probabilística, (KOECH, 2020), e como faz sentido com a natureza probabilística já elucidada da cross-entropy

$$E_{CE} = - \sum_{j=1}^m \mathbf{y}_{r,i} \log \mathbf{y}_{p,i}. \quad (7)$$

Como probabilidade é, por definição, um valor entre 0 e 1, o logaritmo disso resulta num valor negativo; o erro, por outro lado, não é de interesse que seja negativo, pois é uma medida análoga à distância. Assim, é colocado o sinal negativo anterior ao somatório em (7), para que o valor de E_{CE} seja sempre positivo. Consoante Brownlee (2019), o logaritmo costuma ser calculado em base 2, pois esta é a base do sistema binário, o que torna a resposta do cálculo uma referência em *bits*, as unidades de informação nesse sistema.

Foi explanado sobre a natureza probabilística dos dados que são inseridos na *cross-entropy* e como o vetor de *output* da rede deve ser transformado numa DPD caso já não o tenha sido feito pelo uso da softmax. Não foi discorrido ainda, entretanto, sobre como o vetor de classificação esperado ou real deve ser montado para enquadrar-se neste contexto. Em verdade, isto pode ser feito de uma forma muito conveniente: para um problema de n classes, o *output* da rede deve ter dimensão igual a n , e, para a classificação da i -ésima classe, a i -ésima posição do vetor de *output* deve ser igual a 1, ao passo que todas as outras componentes devem ser nulas. Além de demarcar de forma visível a classe a que se pertence uma amostra pela posição do único elemento diferente de zero na componente cujo número seja igual ao número de ordenação da classe, há uma interpretação probabilística para esse arranjo - para esta i -ésima classe, a i -ésima componente do vetor ser unitária significa que a probabilidade de uma amostra ser dessa classe é de 1, enquanto que a probabilidade de pertencer a outras classes é nula, já que as outras componentes do vetor são nulas. Outra característica inerente a uma DPD é que, se a soma do vetor de classificação com esse arranjo for somado, o resultado será unitário, que é justamente uma das obrigatoriedades dentro de uma distribuição de probabilidade.

Destaca-se também que, neste estudo e em vários outros, a função custo *cross-entropy* é muito utilizada, posto que produz resultados muito melhores que a MSE, por exemplo. A *cross-entropy*, quando aliada a otimizadores baseados em gradiente, gera modelos que tem capacidades de generalização muito maiores. A sua limitação, contudo, está naquilo que traz as suas vantagens - a natureza de comparação probabilística - uma vez que *outputs* com dimensão

unitária não são capazes de avaliar-se, pois, nesse caso, a soma do vetor de *output* não será necessariamente unitária, o que impede a construção de uma DPD.

3.5 OTIMIZADORES

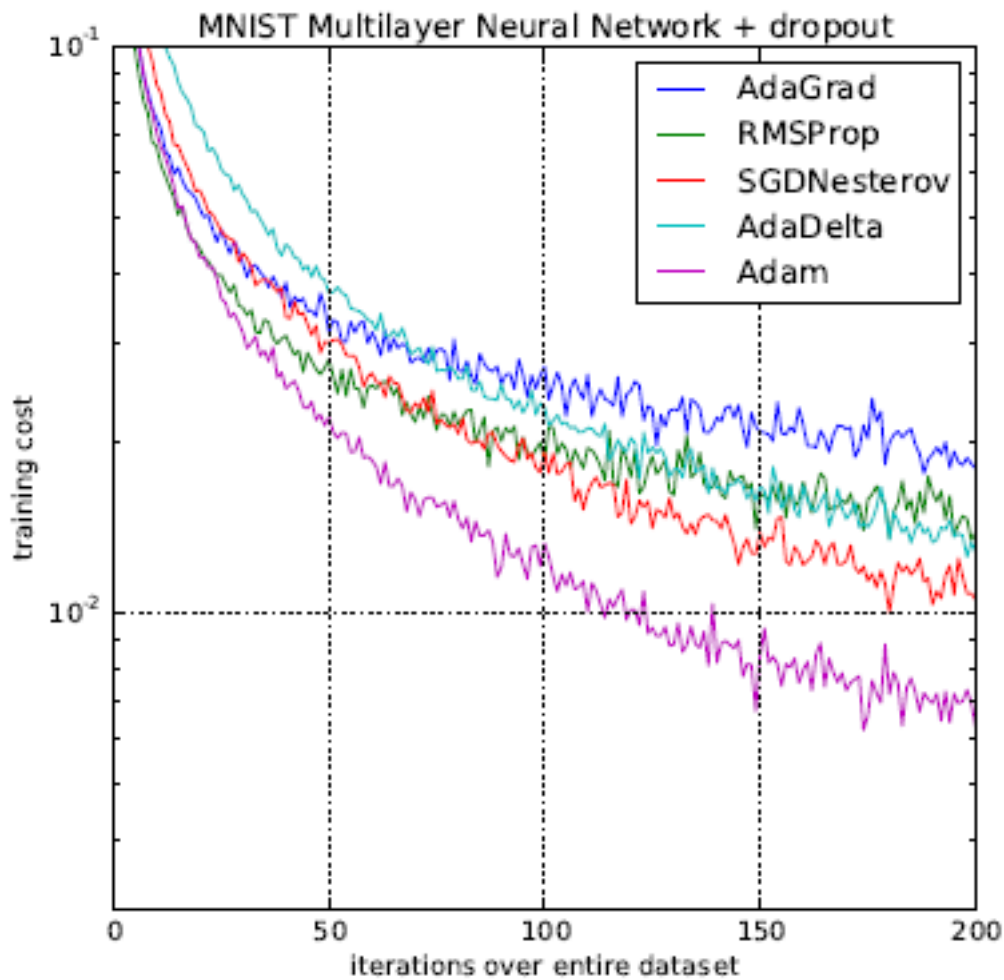
Se uma função custo ou erro mede o quanto a previsão do modelo está longe da resposta real, minimizar seu valor é necessário, pois isso implica numa melhor performance da rede neural no conjunto de dados em que ela está sendo comparada. A resposta da rede passa a ser dependente apenas dos parâmetros, que são os pesos e biases, posto que os hiperparâmetros estão fixos. Logo, deve haver uma metodologia de alterar os parâmetros para que o *output* do modelo fique cada vez mais próximo do *output* desejado, o que, por consequência, diminui o valor da função custo. Essa metodologia é chamada de *otimização* e é performada por um *otimizador*, que, por meio de um algoritmo, percorre o espaço vetorial dos pesos e biases e encontra um ponto, isto é, um arranjo daqueles, em que a função custo é mínima, seja local seja globalmente. A partir disso, no contexto de otimização, a função custo troca de nome para *função objetivo*, pois minimizá-la é o foco e a demanda do otimizador.

Este processo de percorrer o espaço dos parâmetros na busca pela melhor configuração é chamado também de *treinamento*, e o conjunto de dados que fornece o *input* para a rede e de onde se tira o vetor de *output* desejado para a comparação é denotado por *data set de treinamento* (*data set*, do inglês para conjunto de dados). Existem diversas famílias de estratégias para resolver o problema de otimização, sendo duas muito difundidas a PSO (*Particle Swarm Optimization*, do inglês para Otimização por Enxame de Partículas) e GD (*Gradient Descent*, do inglês para Descida por Gradiente), embora, no contexto de RNA, a última opção seja a mais difundida para a otimização de parâmetros. Há, todavia, exemplos do uso de PSO na otimização de hiperparâmetros, como em Voltz (2019). Os métodos do tipo PSO utilizam a dinâmica de partículas para *explorar* o espaço de parâmetros; enquanto os métodos do tipo GD calculam o gradiente da função objetivo e percorrem a partir de um ponto para outro ponto numa direção que é modificada ou não a partir do gradiente e no sentido oposto a este por uma distância especificada ou calculada a cada iteração, distância esta, η , chamada de *learning rate* (do inglês para taxa de aprendizado). A direção oposta ao gradiente é minimizante e isto é um conhecido resultado provado por derivadas direcionais.

Dos otimizadores que utilizam o gradiente e que são usados em RNAs, pode-se citar Nesterov, RMSProp, Adam e suas variantes. Neste estudo, porém, o Adam (*Adaptive Movement Estimation*, do inglês para Estimativa Adaptativa de Movimento) foi usado em todas as simulações, uma vez que, como explicitado em Brownlee (2021), este otimizador é mais rápido, computacionalmente menos custoso e mais preciso que seus pares. Sendo que a superioridade do Adam é visível na Fig. 6, pois este encontrou um mínimo menor e mais rapidamente.

A performance notável do Adam é creditada à abordagem de gerar diferentes passos para cada variável do gradiente da função objetiva, ou seja, para cada parâmetro, de modo que cada

Figura 6 – Comparação do valor da função objetivo ao longo das iterações entre o otimizador Adam e seus pares.



Fonte: (BOCK; GOPPOLD; WEIß, 2008)

um destes percorra uma distância diferente na direção minimizante, bem como a um cálculo diferenciado desta. Como o gradiente tem uma grande variabilidade no espaço de parâmetros, (BUDUMA, 2017), há a demanda por técnicas matemáticas que mitiguem as flutuações do gradiente, de modo que a minimização não permaneça num caminho muito irregular quando pode seguir por um mais direto. Esse *amortecimento* é feito por conceitos análogos à inércia, tanto é que são chamados de primeiro e segundo momentos de gradiente, que computam e memorizam o valor e o quadrado desse valor do gradiente respectivamente ao longo das iterações. De forma que a direção de minimização não será mais puramente a oposta ao gradiente; mas a oposta à razão desses dois momentos, que são corrigidos com base em dois hiperparâmetros: os decaimentos de momento.

Segundo Brownlee (2017a), o otimizador Adam foi testado empiricamente nos algoritmos de regressão logística para o reconhecimento de dígitos no conjunto de dados do MNIST (*Modified National Institute of Standards and Technology*, do inglês para Instituto Nacional Modificado para Padrões e Tecnologia) e para a análise de emoção e expressão no conjunto de

dados da IMDB (*Internet Movie Database*, do inglês para Base de Dados de Filmes da Internet). No curso da Universidade de Stanford em *deep learning* intitulado de “CS231n: Convolutional Neural Networks for Visual Recognition”, desenvolvido por Andrej Karpathy et al (S.l), a escolha do Adam é padrão, embora deva-se testar também o algoritmo SGD+Nesterov Momentum.

3.6 DIVISÃO DO CONJUNTO DE DADOS ORIGINAL

Como já vem sendo introduzido ao longo deste capítulo, a meta por trás de uma rede neural artificial é classificar, realizar uma previsão ou ajustar uma curva para um conjunto de dados. Assim, o conceito de conjunto de dados precisa ser esmiuçado e, para tal, dois termos necessitam de elucidação: amostra e variável. Este último já foi comentado, posto que a variável é o equivalente a uma componente do vetor de *input* e, além disso, o número de variáveis corresponde à dimensionalidade do conjunto de dados. A nomenclatura *amostra*, contudo, precisa ser definida ainda: a qual é uma célula consistente do conjunto de dados global, consistente porque tem a mesma dimensão deste; no contexto de experimentação, cada amostra representa cada medição; no âmbito deste trabalho, cada amostra equivale a cada uma das 73 placas de compósito. Uma forma costumeira na literatura de se organizar um conjunto de dados é a matricial, em que, para cada linha, está uma amostra e, para cada coluna, está uma variável.

Para se avaliar a efetividade de uma RNA na sua atividade classificatória, esta avaliação deve ser feita num *data set* que não foi utilizado para o treinamento do modelo, isto é, a otimização dos parâmetros, para que a prova seja tirada num campo de dados a que a rede não foi exposta antes, pois, assim, há de se averiguar se o treinamento levou o modelo a ter sucesso apenas nesta fase ou se ganhou capacidade de generalizar sua habilidade de decisão, que é o verdadeiro interesse. Dessa maneira, o conjunto de dados original, ou bruto, deve ser dividido para que a rede seja exposta a apenas uma parte daquele durante o treinamento e que, na sequência, seja colocada à prova noutra ou noutras seções do conjunto original. A utilização de uma parte dos dados brutos é intuitiva, pois são conhecidas as classificações reais das amostras que caíam nessa separação. Evidencia-se, porém, que a divisão é feita por amostras; não por variáveis, logo, uma quantidade de amostras será segregada para cada divisão do conjunto original.

Existem diversas metodologias para levar a cabo a segregação das amostras. Entre estas, está a validação cruzada em k partições, que, segundo Brownlee (2018), consiste na separação dos dados crus em k conjuntos independentes, o que resultará em k análises, posto que, a cada uma destas, a RNA será treinada utilizando as outras partições para ser testada no i -ésimo conjunto. Ao final das k análises, a média da performance será avaliada. Essa metodologia tem a vantagem de verificar a capacidade de generalização de uma arquitetura em particular; é, no entanto, muito desvantajosa para *data sets* com muitas amostras, haja vista ao custo computacional elevado.

Outra forma de fazer a separação é pela construção dos *data sets* de treinamento, validação e teste, sendo que o primeiro é intuitivo sobre qual é a sua função; é, contudo, nos dois últimos

em que se encontra uma questão mais profunda. Como explica Brownlee (2017b), o *data set* de validação tem a função de medir a assertividade do modelo que teve seus parâmetros ajustados no *data set* de treinamento, e, com esta medição, procede-se a ajustar os hiperparâmetros da RNA; ao passo, que o *data set* de teste tem a função de medir a assertividade por uma última vez, depois das correções e melhoramentos nos hiperparâmetros. Logo, percebe-se que existe uma relação de aprimoramento nesta abordagem, de maneira que o treinamento otimiza os parâmetros e o *data set* de validação é usado para corrigir e evoluir arquiteturas utilizando a performance de outras como critério e direção.

É importante salientar que, independentemente da forma com que os *data sets* são divididos, deve haver um cuidado para que cada um seja homogêneo na composição em relação aos demais, de modo que haja uma proporção semelhante, senão igual, de amostras de cada classe em cada conjunto. Se existem anomalias nos experimentos ou na construção das amostras, estas devem ser dispersas nos conjuntos com base naquelas. Porque, caso contrário, as medidas de assertividade serão enviesadas por disparidades na dispersão dessas amostras com características diferentes. O que pode levar, além da medida de desempenho enviesada, a uma baixa capacidade de generalização e até confusão ou desfalque no processo de aprendizagem da RNA.

3.7 MÉTRICAS DE DESEMPENHO

Já foi comentado que a meta é construir RNAs que sejam capazes de produzir classificações e previsões assertivas, deve-se, portanto, definir parâmetros matemáticos que sejam capazes de atribuir valores para a qualidade do desempenho de uma rede e, sobretudo, que essa atribuição seja consistentemente comparável, a isso se dá o nome de *métrica*. Antes de construir estas métricas, é preciso que se esclareça que estas só fazem sentido num *data set* específico, pois se está comparando a predição do modelo com o valor real, ambos os quais estão contidos num conjunto de dados. Outro ponto importante é que as métricas revelam fenômenos, a saber: *overfitting*, quando os parâmetros são excessivamente ajustados para um *data set* e são vulneráveis e demasiadamente sensíveis, até ao ruído; *underfitting*, o oposto do fenômeno anterior, ocorre quando o ajuste dos parâmetros é tão grosseiro que a capacidade de discernimento do modelo torna-se grosseira e, por consequência, este é incapaz de classificar sequer as diferenças mais óbvias; capacidade de generalização, é a habilidade de o modelo ter previsões acuradas em outros conjuntos de dados que não fizeram parte do seu treinamento.

A melhor forma de organizar a resposta de um modelo é por uma matriz de confusão, que relaciona, em suas linhas, as previsões com, em suas colunas, as respostas reais. Logo, a matriz de confusão é do tipo quadrada com dimensões $n \times n$, em que n representa o número de classes. Assim, as previsões acertadas estão na diagonal principal e, por conseguinte, a acurácia, que é a razão dos acertos pelo total de previsões, será igual à quociente da soma da diagonal principal pela soma de toda a matriz de confusão. Uma visão esquemática de uma matriz de confusão pode ser contemplada na Fig. 7, em que $q_{i,j}$ representa a quantidade de amostras da

classe j que foram classificadas como da classe i . Em vermelho, as previsões errôneas; em verde, as acertadas. Assim, a acurácia num *data set* que não o de treinamento pode ser usada como uma forma de medir a capacidade de generalização de um modelo, pois maiores acurácias significam que o modelo foi capaz de extrapolar o seu critério para outros conjuntos de dados sobre os quais não foi treinado.

Figura 7 – Visão esquemática de uma matriz de confusão.

	$y_{r,1}$	$y_{r,2}$...	$y_{r,m}$
$y_{p,1}$	$q_{1,1}$	$q_{1,2}$		$q_{1,m}$
$y_{p,2}$	$q_{2,1}$	$q_{2,2}$		$q_{2,m}$
...				
$y_{p,m}$	$q_{m,1}$	$q_{m,2}$		$q_{m,m}$

Fonte: Produção do autor.

A função custo no final do processo de otimização também é um fator de desempenho, posto que é um indicativo da acurácia do modelo no *data set* de treinamento e pois, se o seu valor for baixo enquanto as acurácias do modelo forem baixas noutros *data sets*, é sinal de que está acontecendo *overfitting*, uma vez que o modelo foi muito bem ajustado para os dados de treinamento, mas é incapaz de estender esse ajuste para outros conjuntos, em virtude de sua sensibilidade. Por outro lado, quando o valor da função custo é alto demais, significa que houve um *underfitting*, já que o processo de otimização foi incapaz de minimizar o erro para que os parâmetros fizessem o modelo se encaixar às classificações esperadas.

4 MÉTODOS DE REDUÇÃO DE DIMENSIONALIDADE E CLASSIFICAÇÃO

Embora os métodos de redução de dimensionalidade são uma etapa de pré-processamento dos dados antes que estes sejam fornecidos à RNA, é de interesse do ponto de vista didático que o leitor ambiente-se com os conceitos da construção dos modelos, de sua otimização e de sua verificação, para que, então, seja capaz compreender os caminhos percorridos por esta pesquisa e suas justificativas. Bem como foi capaz essa introdução de já lançar luz sobre os hiperparâmetros que foram empregados nas simulações.

Foi mostrado que a camada de *input* da rede neural artificial tem o seu tamanho igual à dimensão dos dados, ou seja, igual ao número de variáveis e, portanto, igual ao número de colunas na matriz de dados, conforme a convenção adotada na seção 3.6. Logo, o tamanho da camada de *input* implica diretamente na quantidade de ligações que a primeira camada intermediária fará com aquela, o que implica na quantidade de ligações como um todo. Foi visto que o número de parâmetros é diretamente proporcional ao número de conexões interneurais, por essa lógica, uma camada de *input* muito grande acarreta num número grande de parâmetros a serem otimizados, o que, por consequência, ocasiona um custo computacional alto e a diminuição da efetividade do otimizador, de modo que a performance global da rede possa ser negativamente afetada. Em verdade, é por isso que surge a demanda para que a camada de entrada seja encolhida e, portanto, aparece a necessidade de que os dados tenham sua dimensionalidade diminuída.

No âmbito da classificação, existem ainda problemas causados pela dimensão demasiadamente grande (acima de 100 variáveis) que estão além da construção e treinamento de um modelo em si. Essa problemática é chamada de *curse of dimensionality* (do inglês para a maldição da dimensionalidade) e, conforme Arun (2020), manifesta-se de duas formas: esparsidade dos dados, que compreende a dificuldade de sintetizar num subconjunto do conjunto original todas as combinações de variáveis, o que leva o treinamento a ser feito somente sobre as combinações mais frequentes e, portanto, leva ao prejuízo da capacidade de generalização e ao *overfitting*. E a outra forma é a concentração à distância, que se manifesta na medida da distância entre amostras de uma classe a partir do centróide desta, de tal forma que, quanto mais se avança nas dimensões, mais concentradas ficam as amostras entre si, o que acarreta numa dificuldade de segregação para métodos que dependem de uma métrica de distância para escrutinar e classificar. Essa aglutinação, entretanto, prejudica não apenas as técnicas que utilizam este tipo de mapeamento; mas também, a classificação por RNAs, posto que o valor da variável nessa dimensão com aglutinação é muito semelhante entre as amostras, tornando assim muito confuso o trabalho de segregação da rede neural, haja vista que mesmo funções de ativação não-lineares conseguem afastar satisfatoriamente os *inputs*, o que produz o *underfitting*. Visto isso, fica evidente que a praga da dimensionalidade pode gerar tanto *over* quanto *underfitting*.

Com essas duas motivações, particularmente sendo a última apresentada a mais importante segundo Bishop et al. (1995), deve-se proceder à redução da dimensionalidade, que também pode ser chamada de compressão dos dados. As técnicas para realizar isso podem ser classifica-

das como se segue - quanto a sua linearidade e quanto a sua natureza. Como o objetivo deste trabalho é traçar uma comparação entre PCA e LDA, estes serão o foco nas seções subsequentes. A próxima seção será sobre uma característica comum aos dois métodos, a linearidade; enquanto a subsequente a esta tratará sobre o que diferencia uma técnica da outra.

4.1 MÉTODOS LINEARES

Para compreender matematicamente o processo de redução de dimensionalidade, pode-se fazer uso de conceitos da Álgebra Linear, como mudança de espaço vetorial e transformação linear. Já foi lançada a fundação em termos de análise de como o modelo de RNA é basicamente uma função que conecta um domínio a um contradomínio por meio de sua imagem, e já foi percorrido sobre a natureza do domínio, no formato A^m , em que A é o conjunto numérico em que estão os dados. Assim, os experimentos de vibrações coletam dados de deslocamento ou de velocidade ou de aceleração, como é o caso nesta pesquisa, assim faz sentido que estas medições estejam no conjunto dos números reais, logo $A = \mathfrak{R}$. Pelo fruto da mesma comparação, sabe-se então que o domínio é, em termos da Álgebra Linear, um espaço vetorial cuja base é os números reais com uma dimensão igual a m , ou simplesmente \mathfrak{R}^m . Em consequente, a redução de dimensionalidade pode ser interpretada como uma mudança desse espaço original para um novo espaço vetorial de dimensão menor, denotada por n , ou em notação matemática $T : \mathfrak{R}^m \rightarrow \mathfrak{R}^n$, tal que T é a transformação que realiza essa mudança.

Ainda na esfera da Álgebra Linear, pode-se analisar propriedades da transformação T , as quais devem configurá-la como uma transformação linear, pois esta é a forma mais simples de realizar uma mudança de base. Essa simplicidade, em verdade, é consequência de como pode ser construída uma transformação linear: por meio de uma mera multiplicação matricial,

$$T(\mathbf{x}) = \mathbf{x}\mathbf{B}, \tag{8}$$

tal que \mathbf{B} é a matriz de transformação, especificamente uma matriz $m \times n$; \mathbf{x} , a matriz dos dados, ou seja, uma matriz $k \times m$, de modo que possui k amostras, uma para cada linha, e m variáveis, uma para coluna, como já foi convencionado anteriormente. Ao contrário da representação comum de transformações lineares, em que a matriz de transformação está à esquerda do *input*, \mathbf{x} , a ordem será invertida, posto que a ordem apresentada na equação (8) é a utilizada nos métodos LDA e PCA. Haja vista que a simplicidade dos métodos lineares está na sua natureza, de uma mera multiplicação matricial, o que imperará e aparatará os métodos dentre si será a forma como a matriz \mathbf{B} é calculada. Isto, em verdade, é função do que a técnica valoriza mais e, portanto, o que deseja preservar com maior cuidado.

4.2 MÉTODOS CLASSIFICADORES

Existem métodos que são orientados à classificação, de modo que, para o desenvolvimento de seu algoritmo, deve-se fornecer a classe a que pertence cada amostra e, por uma particularidade intrínseca da técnica, esta informação é computada e condiciona-a à distribuição das amostras em suas respectivas classificações. Este conhecimento também pode ser usado pelo método para inferir classificações, de forma que aquele seja não apenas para reduzir dimensionalidade; mas, sobretudo, para classificar um conjunto de dados. O procedimento para que isso seja realizado é semelhante ao que é feito com redes neurais artificiais: parte do conjunto de dados original é subdividido em um *data set* de treinamento e noutro de teste; o método é então *treinado* no primeiro e, na sequência, é testado no último, para que se veja se é capaz de classificar com sucesso. Embora não se deva confundir com o procedimento para construção de uma RNA, posto que são entes completamente distintos, mesmo no aspecto da *aprendizagem*, pois modelos de RNA aprendem pela otimização de parâmetros num processo iterativo de redução de erro, como já foi descrito; enquanto métodos de compressão de dados com orientação classificatória têm um algoritmo fechado que analisa propriedades estatísticas dos dados e das relações entre estes.

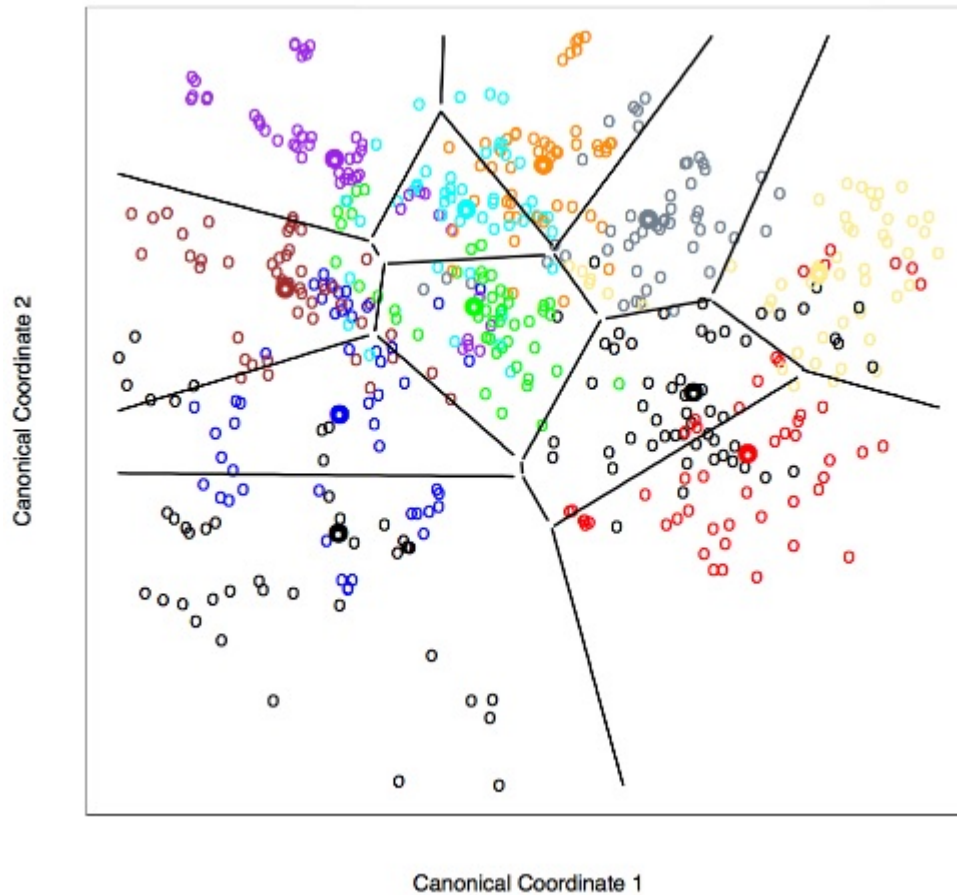
No processo de mudar os dados para um novo espaço vetorial, métodos classificadores tenderão à agrupar as amostras com base nas classificações que aqueles forneceram. Este agrupamento pode ser mapeado por alguma métrica, de tal modo que fronteiras sejam traçadas entre as classes, o que, por consequência, delimita regiões, ou *loci* (plural para *locus*), no espaço para cada classe. Destaca-se que a informação necessária para a formação dessas fronteiras e para a alocação das regiões é um atributo da fase de treinamento. Assim, fica claro que verificar os *loci* em que as amostras não utilizadas para o treinamento foram depositadas configura um meio de avaliar se o método é capaz de classificar acuradamente.

As fronteiras do parágrafo anterior são formalmente chamadas de *fronteiras de decisão* (tradução livre do inglês para *decision boundaries*). A forma dessas fronteiras é uma característica do método: nas técnicas que podem ser descritas pela seção anterior, por exemplo, as fronteiras de decisão são linhas, planos ou hiperplanos, dependendo da dimensionalidade do espaço vetorial projetado, o que é comum a estas três entidades geométricas é a curvatura nula, que é consequência da linearidade das técnicas na matriz de dados de *input*, a exemplo da Fig. 8. É um resultado conhecido de outras áreas da Engenharia e da Matemática Aplicada que representar ou reduzir fenômenos por modelos lineares é uma tarefa complexa, quando não impossível, haja vista à complexidade e a não-linearidades intrínsecas ao fenômeno, o que leva ao erro, tolerável ou não, caso a abordagem linear seja mantida.

Analogamente, percebe-se que confinar dados em regiões linearmente delimitadas pode induzir ao erro, uma vez que sequer conjuntos não conexos podem ser segregados coerentemente, pois é definido que um conjunto conexo é aquele em que todos seus pontos devem ser ligados por uma poligonal, que é, essencialmente, uma série de linhas conectadas uma após a outra. Visto isso, é razoável reexaminar que métodos lineares tenham erros na tarefa de separar e classificar dados

frutos de experimentos de natureza não linear. Isto implica numa seara de hipóteses estatísticas simplificadoras dos dados para que tais métodos sejam aplicados com segurança.

Figura 8 – Exemplo de fronteiras de decisão lineares num conjunto de dados.



Fonte: Hastie, Tibshirani e Friedman (2009)

O mapeamento que leva ao conceito de fronteira e que permite a distribuição dos *loci* é uma medida de distância, o que implica, portanto, na avaliação de uma distância a partir de uma referência. Esta sendo, em consequente, o centro de cada classe, pois, intuitivamente, a probabilidade de uma amostra pertencer a uma determinada classe é inversamente proporcional à distância entre o centro da classe e si. É natural que esse centro seja o centróide da distribuição de pontos de uma classe, já que o centróide de uma distribuição discreta é a média aritmética entre as coordenadas dos pontos. Segundo Prabhakaran (2019), o cálculo da distância euclidiana entre um ponto correspondente a uma amostra e o centróide da classe é uma métrica enganosa para distribuições de dados cujas variáveis tenham algum tipo de correlação, e isso acontece porque a métrica euclidiana é meramente a medida do afastamento entre dois pontos no espaço como se as coordenadas daqueles, ou seja, suas variáveis, não tivessem correlação alguma entre si, o que impede essa métrica de identificar *clusters*, isto é, agrupamentos concentrados de pontos e de medir o afastamento de um ponto não apenas do centróide mas de todo o conjunto.

Para que essa deficiência na análise fosse mitigada, Prasanta Mahalanobis propôs em 1936 uma métrica que considerasse a covariância entre os pontos, técnica esta que ficou conhecida

como distância de Mahalanobis, em homenagem a seu criador. A qual é definida por

$$d = \sqrt{(\vec{x} - \vec{\mu})^T \mathbf{S}^{-1} (\vec{x} - \vec{\mu})}, \quad (9)$$

em que d representa a distância; \vec{x} , o ponto de que se deseja calcular a distância a partir do centróide; $\vec{\mu}$, o centróide da distribuição; \mathbf{S} , a matriz de covariância da distribuição. Consoante com Prabhakaran (2019), a primeira multiplicação, da esquerda para a direita, na equação (9) equivale, basicamente, a uma extensão para um espaço multivariado da normalização de uma distribuição pela média e pelo desvio padrão, pois a subtração do ponto pelo centróide deixa a distribuição com média nula, enquanto a divisão pelo desvio padrão torna a variância da distribuição unitária. Analogia esta que faz sentido mesmo que a matriz \mathbf{S} seja de covariância, o que é coerente porque o desvio padrão é a raiz quadrada da variância. Logo, por esta operação, as variáveis foram descorrelacionadas, de modo que possa ser feita a segunda multiplicação. Com este último resultado, percebe-se que, caso a distribuição fosse com variáveis sem correlação alguma e a variância em cada uma dessas variáveis fosse unitária, a matriz \mathbf{S} seria a matriz identidade e, assim, a equação (9) e a distância de Mahalanobis, por consequência, reduzir-se-ia à distância euclidiana, ou seja, a métrica Mahalanobisiana é uma generalização da euclidiana para distribuições em que há correlação entre variáveis.

Como os métodos classificadores atuam como um modelo em si próprios, já que segregam e agrupam amostras, suas respostas podem facilitar a tarefa dos modelos de RNA ou podem fazer justamente o contrário. A diferença está no quanto os *loci* estão afastados, posto que agrupamentos muito próximos implicam em variáveis com pequena amplitude de variação entre as amostras, o que dificulta a separação na rede, sobretudo pela função de ativação; e a acurácia do método também influencia, haja vista que amostras erroneamente classificadas e, em consequência, posicionadas numa região de outra classe induzem a RNA ao erro. Logo, é crucial que as condições do *data set* sejam favoráveis a aplicação dessas técnicas, para que, ao menos, acurácias de 100% ou muito próximas disso sejam atingidas. Essas condições compreendem a validade das hipóteses sobre a distribuição dos dados e a quantidade de amostras, para que o método seja coerente com a natureza das informações e seja exposto a várias combinações possíveis de variáveis, a fim de reconhecer e aprender as nuances dentro de cada classe.

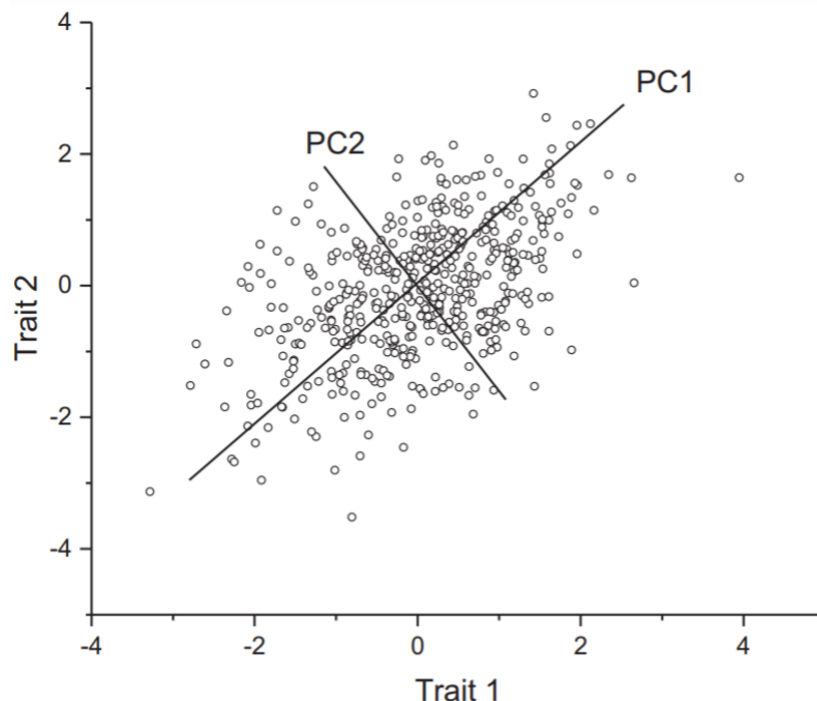
Apesar desta seção tratar sobre métodos classificadores, é importante que seja feito um adendo sobre métodos que não fazem classificação. A missão destes, portanto, é apenas a redução de dimensionalidade, o que os leva a desconsiderar a presença de classes e a procurar por outras características das amostras para atuar sobre e preservar em seus algoritmos. Dessa forma, essas técnicas não precisam de um treinamento, nem sua acurácia precisa ser monitorada, já que sequer há sentido em medir acurácia de um método que não emite classificação. Logo, esses métodos são menos prejudicados quando o número de amostras é pequeno comparado com o número de combinações de variáveis a que um modelo precisa ser exposto para ser acurado, o que os torna menos suscetíveis a induzir ao erro uma RNA na escassez de amostras.

4.3 ANÁLISE DO COMPONENTE PRINCIPAL

A análise do componente principal, ou simplesmente PCA, é um método linear e não classificador; é, no entanto, muito utilizado nas aplicações de RNA, posto que seus resultados são interpretáveis, conforme Doring (2018a); proporciona bons resultados, mesmo com dados não-normalmente distribuídos, segundo Jolliffe e Cadima (2016); é facilmente implementado e customizado. Consoante Jaadi (2021), o objetivo do PCA é manter a representação mais acurada possível dos dados, mas tolerando uma perda de informação para que aquela seja simplificada, o que é feito segundo Jolliffe e Cadima (2016) pela construção de um novo espaço vetorial cujas bases sejam ortogonais e a variância seja maximizada.

Conservar a variabilidade dos dados é um critério chave, pois é capaz de evitar que aqueles sejam condensados, de forma que sejam difíceis de se separar, e identifica *clusters* pelos seus eixos principais, como pode-se ver na Fig. 9. O meio pelo qual se avalia a variabilidade é pelo parâmetro estatístico da variância, o que implica, no contexto de análise multivariada, numa matriz de covariância, que continuará sendo representada por S como na seção anterior.

Figura 9 – Eixos principais num cluster.



Fonte: Kamperis (2021)

Conforme dedução omitida aqui, mas detalhada em Jolliffe e Cadima (2016), a maximização da variância é feita por um problema de autovetores e autovalores da matriz de covariância, de modo que aqueles representam os eixos principais de variância e estes a fração deste parâmetro estatístico no âmbito de todo o conjunto de dados. A abordagem por autovetores, pela sua ortogonalidade intrínseca, acarreta que, quando utilizados como base para o novo espaço vetorial, produzirão variáveis que não são correlacionadas entre si. Dessa forma, o PCA tem uma estrutura

análoga à resolução da equação de vibrações, a equação (1) deste trabalho, por um problema de autovetores, uma vez que estes geram um novo sistema de coordenadas independentes e seus autovalores correlacionam uma quantidade de interesse - a variância no âmbito dos dados e a frequência natural na esfera de vibrações.

Segundo Jolliffe e Cadima (2016), o PCA é equivalente ao SVD (*Singular Value Decomposition*, do inglês para Decomposição pelo Valor Singular) da matriz de dados centrada pela coluna e normalizados pelo desvio padrão, X^* . Sendo que as componentes desta, x^* , são definidas como segue

$$x_{i,j}^* = \frac{x_{i,j} - \bar{x}_j}{\sigma_j}, \quad (10)$$

tal que $x_{i,j}$ corresponde à componente da matriz de dados; \bar{x}_j , a média da j -ésima coluna desta matriz; σ_j , o desvio padrão desta mesma coluna. Assim, para que a equivalência com o método SVD seja traçada, a matriz de covariância amostral pode ser descrita por

$$S = \frac{X^{*T} X^*}{k-1}, \quad (11)$$

em que k é novamente o número de amostras e T indica a transposição da matriz. Na sequência, calcula-se os autovetores e autovalores dessa matriz de covariância amostral e ordena-os pela ordem decrescente de seus autovalores. Como os autovalores representam a variância que cada autovetor tomou do conjunto de dados original, os maiores autovalores indicam as maiores variância e, de fato, maximizá-la é o objetivo do método, o que justifica a ordenação descrita. Neste ponto, cada autovetor, \vec{e} , forma uma parte da base da matriz de mudança de base e é chamado, portanto, de *componente principal*. A base é construída da seguinte forma

$$B = [\vec{e}_1 \ \vec{e}_2 \ \dots \ \vec{e}_n]. \quad (12)$$

Na qual, os índices dos componentes principais, e , indicam a ordem destes na base, embora sua ordenação seja em ordem decrescente pelo critério do autovalor. O índice n informa que a mudança de espaço vetorial é para um com dimensão igual a n , que, por sua vez, é menor que a dimensão original m . A escolha do valor de n é feita pelo usuário que manipula o algoritmo do PCA; deve-se, porém, recordar que a razão de variância preservada pode ser calculada por

$$\eta = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_n}{\lambda_1 + \lambda_2 + \dots + \lambda_n + \dots + \lambda_m}, \quad (13)$$

tal que λ representa um autovalor e os índices são correspondentes aos índices da equação (12). Pela equação (13), o usuário pode encontrar n para que um *eta* mínimo seja obedecido. Por fim, os dados são transferidos para o novo espaço vetorial pela seguinte transformação

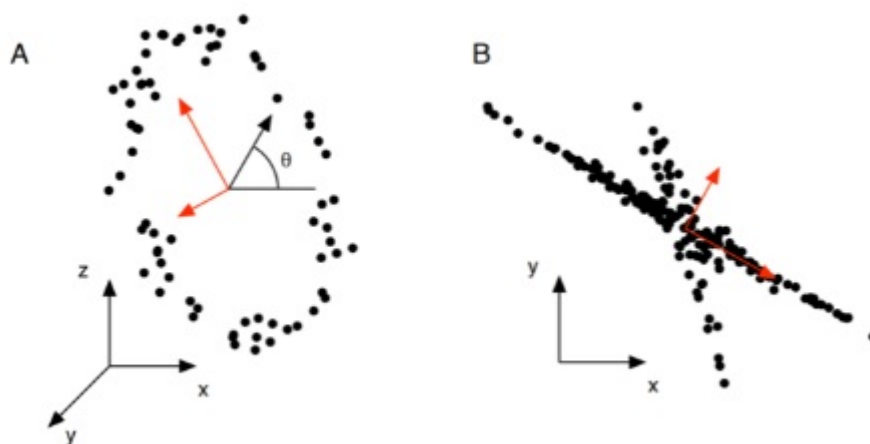
$$X_c = B^T X^{*T}. \quad (14)$$

A transformação é feita sobre a matriz de dados normalizada pois, segundo Jolliffe e Cadima (2016), a normalização pelo desvio padrão amostral colunar garante que as variáveis sejam restringidas por uma escala, de modo que, mesmo que os dados forem coletados em unidades de medida diferentes (o que não é um problema matematicamente; mas, sim, do ponto de vista físico), o desvio padrão há de traçar uma escala que retifique a discrepância. Sendo isto de grande importância, posto que, consoante Jaadi (2021), variáveis com alta variância frente ao conjunto destas *atraem* os componentes principais para si, de maneira que uma variável de um experimento que tenha uma variância falsamente alta por um erro de escala torna-se importante na métrica do método, mesmo não sendo.

Kamperis (2021) evidencia ainda que os testes de esfericidade de Bartlett, que identifica se existe relações entre as variáveis, o que é necessário haja vista à natureza do PCA de construir sua análise sobre a correlação entre as variáveis. E a referência cita ainda o teste de Kaiser-Meyer-Olkin (KMO), cujo escopo é avaliar a proporção de variância nas variáveis que é comum a todas, o que indica fatores não descritos ou não esperados; quanto maior for essa proporção, menos adequado é o conjunto de dados para uma análise utilizando PCA. Kamperis (2021) destacou também a necessidade de verificar se as relações entre as variáveis são lineares, já que isto é uma hipótese para a utilização do PCA. Caso não sejam, deve-se usar o Kernel PCA, KPCA, ou outras técnicas que lidam com dados não-linearmente correlacionados.

Conforme Kamperis (2021), outra limitação está na ortogonalidade da base do PCA, visto que, dependendo do contexto, pode haver uma base melhor simplesmente, como é o caso da Fig. 10, em que há um caso de um componente principal que aponta para ortogonalmente para fora da distribuição na imagem B e outro caso, na imagem A, que mostra a trajetória discretizada de uma pessoa numa roda-gigante, cujas posições são uma função não-linear da fase de giro.

Figura 10 – Casos em que a ortogonalidade do PCA induz ao erro.

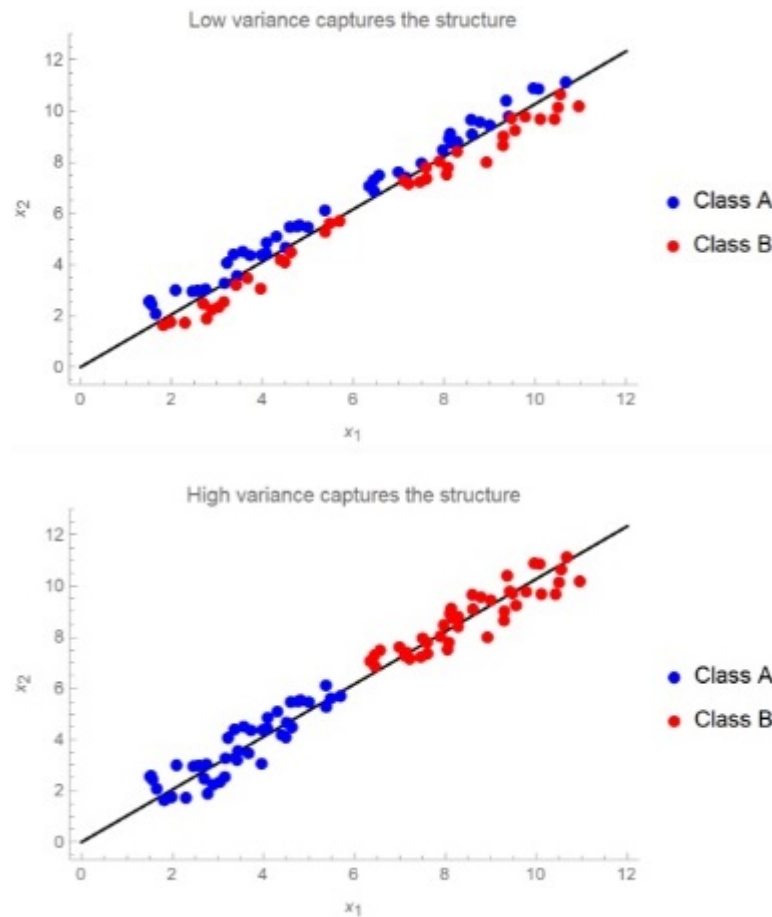


Fonte: Shlens (2014)

A preferência pela variância como critério, consoante Kamperis (2021), também pode fazer equivocados os resultados do PCA, pois estruturas de dados podem distribuir-se com baixa

variância e, ainda assim, com classes bem segmentadas, como é o caso da Fig. 11, em que duas classes estão dispostas com uma variância pequena ao longo de um mesmo eixo.

Figura 11 – Caso em que a dados com pouca variância ao longo do eixo principal induzem o PCA ao erro.



Fonte: Kamperis (2021)

4.4 ANÁLISE DE DISCRIMINANTES LINEARES

A Análise de Discriminantes Lineares, abreviado para LDA, é um método linear e classificatório. O seu critério, conforme Maklin (2019), é a maximização do afastamento dos centróides das classes e a minimização da variância em cada classe. Ou seja, a meta do LDA é, por meio de uma transformação linear, mudar os dados para um espaço vetorial em que as classes estejam mais afastadas e distintamente bem definidas umas das outras, ao passo que as amostras fiquem agrupadas o quanto mais próximas possível dos centróides de suas respectivas classes, o que facilita a atividade classificatória, seja do método seja de outros modelos que usem os dados pré-processados por LDA como *input*.

Doring (2018b) citou que as hipóteses do LDA sobre os dados são: estes são dispersos por uma distribuição normal; as matrizes de covariância de cada classe são iguais entre si. Estas hipóteses são tomadas para que a construção estatística do método pela regra de Bayes seja

simplificada, uma vez que esta é utilizada para criar uma funções que informem cada uma se uma amostra se encontra na sua respectiva classe, sendo aquelas denominadas de funções discriminantes. Como estas são calculadas utilizando a matriz de covariância da classe e uma densidade dos dados condicionados nessa mesma classe, usar estas simplificações permite que as funções discriminantes passem a ser função apenas do vetor de dados de uma amostra a ser testada e, em verdade, linear neste vetor.

Uma maneira de mensurar o afastamento de uma distribuição de pontos de um único ponto é por meio de uma matriz de covariância (DORING, 2018b). Logo, pode-se medir o afastamento das classes entre si pela covariância de seus centróides de um centróide global dos dados, matriz esta que será representada por S_B , que será chamada de matriz de covariância interclasse e que, segundo Raschka (2014), será definida por

$$S_B = \sum_{i=1}^c \frac{N_i}{N} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_g) (\boldsymbol{\mu}_i - \boldsymbol{\mu}_g)^T, \quad (15)$$

tal que c representa o número de classes dos dados; N_i , o número de amostras na i -ésima classe; $\boldsymbol{\mu}_i$, o centróide dessa classe; $\boldsymbol{\mu}_g$, o centróide global dos dados. Consoante Raschka (2014) e Doring (2018b), a razão N_i/N tem a função de trazer um senso de escala ao cálculo, pois correlaciona a importância da matriz de covariância do centróide de cada classe com a densidade de amostras presente naquela.

Por outro lado, a forma de mensurar a dispersão das amostras de cada classe é pela matriz de covariância de cada uma destas, a qual chamada de intra-classe e denotada por S_W é calculada da forma

$$S_W = \sum_{i=1}^c \frac{1}{N - c} \sum_{j=1}^{N_i} (\mathbf{x}_j - \boldsymbol{\mu}_i) (\mathbf{x}_j - \boldsymbol{\mu}_i)^T, \quad (16)$$

em que \mathbf{x}_j está para a j -ésima amostra da i -ésima classe. Destaca-se que cada parcela da matriz intraclasse é dividida por $N - c$, o número de graus de liberdade, para evitar que o estimador tenha algum viés (DORING, 2018b).

Pela interpretação de Fisher, estatístico que propôs o LDA em 1936, a separação de dados almejada no primeiro parágrafo desta seção pode ser visualizada num novo espaço vetorial, \mathbf{X}_c , construído a partir de uma combinação linear do espaço atual na forma $\mathbf{X}_c = \mathbf{w}^T \mathbf{X}$. Evidencia-se que, por simplicidade, \mathbf{X} representa um vetor com os dados de cada amostra. Dessa forma, define-se o seguinte quociente de Rayleigh, conforme apresentado em Liu e Wasserman (2017)

$$J = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}. \quad (17)$$

Fica claro, portanto, que maximizar este quociente de Rayleigh implica num maior afastamento dos centróides das classes (quanto maior S_B) e numa maior concentração das

amostras de cada classe em torno de seu respectivo centróide. Em verdade, segundo Liu e Wasserman (2017), maximizar a equação (17) equivale a maximizar $w^T S_B w$ com a restrição de $w^T S_W w = 1$, o que acarreta num problema de autovetores generalizado. Logo, o maximizador, denotado por \hat{w} , será o autovetor de $S_W^{-1} S_B$ correspondente ao maior autovalor. É trivial, em consequência, notar que, no caso multivariado, w pode ser alçado para uma matriz de base ortogonal entre si, W , porque é a resolução do problema de autovetores de $S_W^{-1} S_B$, sendo estes ordenados pelos seus autovalores em ordem decrescente. Matriz de mudança de base ortogonal esta que leva a seguinte transformação linear

$$X_c = XW. \quad (18)$$

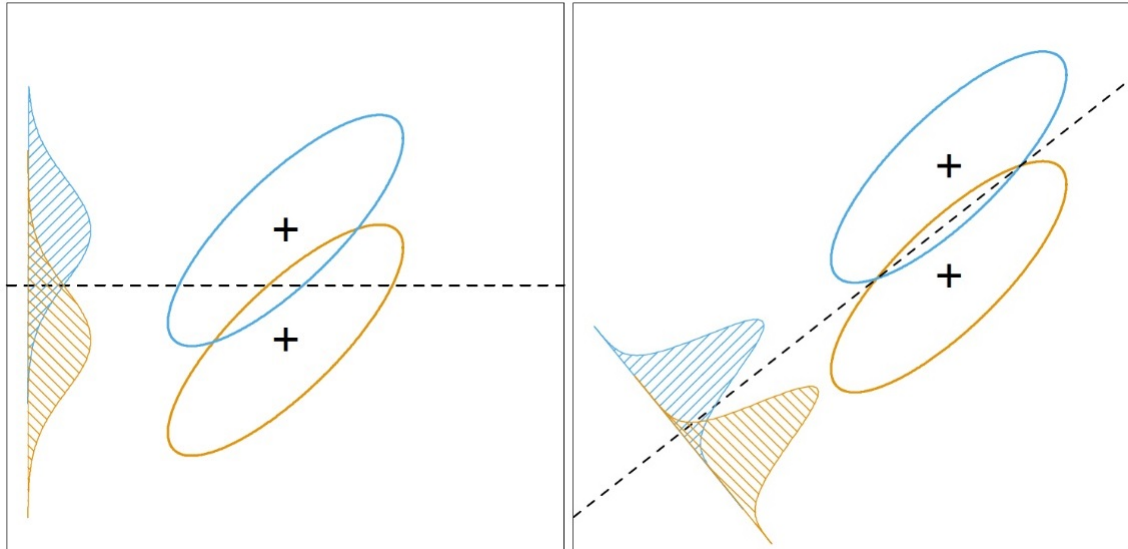
Como uma das hipóteses sobre os dados é que sua distribuição é normal, é importante que sejam centralizados pela média e normalizados pelo desvio padrão, para que tenham, por fim, as características de uma distribuição gaussiana - média nula e variância unitária. Sumarizando, portanto, a rotina do algoritmo de LDA, identifica-se os passos de calcular: as posições dos centróides das classes e o centróide global da distribuição; as matrizes de covariância intra e interclasse; o problema de autovetores e autovalores de $S_W^{-1} S_B$; a ordenação dos autovetores pela ordem decrescente dos autovalores. E, por fim, a construção da matriz de mudança de base W .

Assim como no PCA, retorna, porém, a dúvida em quantos autovetores escolher para formar a matriz W . Ao contrário daquela técnica, há um limite máximo de dimensionalidade da base do novo espaço vetorial construído pelo LDA, que, conforme Doring (2018b) é igual a $c - 1$, sendo isto devido ao fato que o posto do subespaço afim criado a partir dos c centróides é de no máximo $c - 1$. Há a possibilidade, porém, de se escolher menos autovetores ainda, o que é chamado de LDA com posto reduzido, cuja motivação é oriunda da melhora que este procedimento pode trazer para a performance do método sobre modelos na fase de teste que apresentam *overfitting* (DORING, 2018b).

Como o LDA é orientado à classe e é um método *supervisionado* por isso, já que não ignora as classes e seus rótulos (RASCHKA, 2014), tem a possibilidade de superar a dificuldade do PCA, que é uma técnica *não supervisionada*, sobre dados distribuídos com baixa variância ou sobre eixos que não sejam trivialmente encontrados pelo algoritmo de componentes principais, como é possível visualizar na Fig. 12. Raschka (2014) alertou, entretanto, que o PCA costuma superar o LDA em performance quando o número de amostras seja pequeno relativo ao número de variáveis. Steorts (2017) demonstrou o efeito *masking* (do inglês para mascaramento), em que a presença de uma classe não rotulada e, por conseguinte, escondida entre outras classes causa uma piora significativa de performance do LDA no teste, sendo isto comum com 3 ou mais classes. Outra limitação do LDA, segundo Doring (2018b), está quando a quantidade de amostras em cada classe na fase de treinamento não é balanceada, o que levará à incapacidade do método classificar observações de classes que pouco estiveram presentes na criação do modelo. Por outro lado, o LDA pode manter-se robusto e pode produzir bons resultados mesmo quando

as hipóteses dos dados são violadas (RASCHKA, 2014).

Figura 12 – Diferenças entre as projeções utilizando PCA (à esquerda) e utilizando LDA (à direita).



Fonte: Xiaozhou (2020)

4.4.1 Fronteiras de decisão de LDA

Como método classificador que é o LDA, é necessário que seus *loci* sejam determinados pela delimitação de suas fronteiras de decisão, que, como se espera em virtude de ser um método linear, têm curvatura nula. Observando as equações (15) e (16), é possível identificar a relação da técnica com a distância de Mahalanobis; a qual, de fato, é utilizada para mapear o novo espaço vetorial, como é descrito na construção da função discriminante em Liu e Wasserman (2017). Essas funções discriminantes devem ser utilizadas para delimitar as fronteiras, visto que, nestas, o valor daquelas para as duas classes vizinhas é o mesmo. Assim, empregando a semelhança logarítmica e o conceito anterior para as classes adjacentes i e j , encontra-se, conforme Hastie, Tibshirani e Friedman (2009), que o vetor diretor da fronteira entre essas classes deve ser ortogonal a $\mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ e que aquela deve passar pelo ponto médio aos centróides das duas classes, ou seja, em $(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$. Destaca-se que, quando calculadas as direções e pontos de partida das fronteiras, estas devem ser plotadas e suas interseções devem ser avaliadas, de modo que fronteiras que separam ao menos uma classe em comum unam-se para contornar um *locus*. Enquanto fronteiras sem conexão com seus pares num contexto de $c > 2$ devem ser descartadas, visto que estão fora do *locus*.

Para fins ilustrativos, um plot bi ou tridimensional das fronteiras pode ser feito, mesmo que a dimensionalidade do conjunto de dados após o processamento por LDA seja maior que 3. As direções correspondentes aos dois ou três maiores autovalores, portanto, precisam ser escolhidos como bases para este espaço vetorial, posto que são esses autovetores que carregam a maior parte da informação.

5 MATERIAIS E MÉTODOS

A partir dos conceitos abordados nos capítulos anteriores, as configurações e desenvolvimento dos materiais e métodos serão descritos nas seções subsequentes, em ordem paralela à da fundamentação teórica. Neste capítulo, será explicado como o conjunto de dados foi construído e tratado; bem como serão explicitados alguns hiperâmetros que foram mantidos fixos e qual a lógica da variação dos outros. Sendo claro que o objetivo é traçar uma comparação entre o uso, com diferentes razões de amostras para treinamento, de PCA, de LDA e de uma combinação de PCA com LDA na sequência.

5.1 PROCEDIMENTO EXPERIMENTAL

5.1.1 Desenvolvimento das placas de material compósito

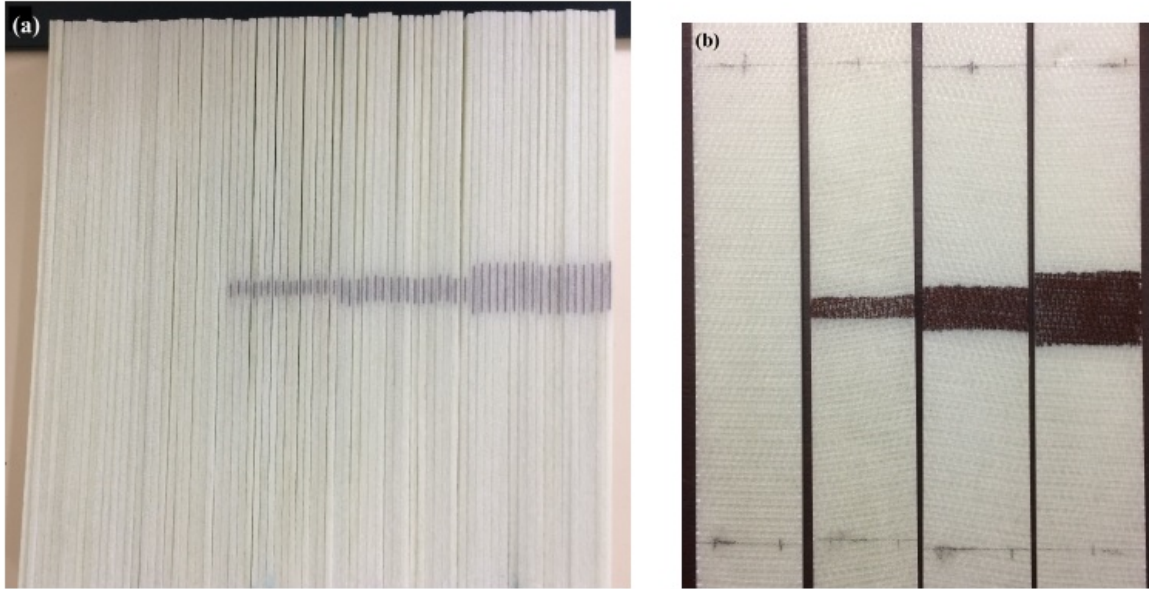
Foram produzidas 73 placas para os experimentos, que, como descrito em Reis (2020), são compostas por 12 lâminas de fibras de vidro com resina epóxi como matriz, sendo todas as camadas orientadas a 0° , $[0]_{12}$. Para a indução de delaminação nas amostras tidas como danificadas, foi inserida uma tira de *Teflon* entre as duas lâminas centrais. Essa tira de Teflon foi cortada e introduzida de modo que separasse completamente essas duas camadas por toda a direção transversal da amostra e por uma extensão longitudinal prescrita, a saber: 5 mm, 10 mm e 19 mm. Sendo que cada uma dessas extensões foram classificadas dentro de um grupo de dano - D1, D2 e D3, respectivamente. Para cada um dos danos anteriores, foram feitas as seguintes quantidades de amostras: 15; 16 e 17, respectivamente, enquanto 25 amostras foram deixadas em sua configuração original e intacta. Na Figura 13, vê-se a presença da tira de Teflon, em seus três distintos tamanhos, e é possível vislumbrar seu efeito na indução de um dano por delaminação.

5.1.2 Ensaio de vibrações

Evidencia-se que, neste trabalho, como descrito em Reis (2020), foi feita uma discretização de dois pontos como se pode ver na Fig. 14, na qual, visualiza-se também o procedimento para a obtenção da componente $H_{1,2}$ da matriz de FRFs. O número de pontos discretos de obtenção de sinal pelo software PULSE de aquisição de dados foi de 6400 num intervalo de frequências de 0 a 3200 Hz. Ao passo que a placa foi deixada numa condição livre-livre e a excitação foi fornecida por um martelo de impacto modelo 8206-003 (sensibilidade 1.14 mV/N) com uma ponteira de alumínio de Brüel & Kjaer (B&K). A medição da aceleração ao longo do experimento foi feita por um acelerômetro modelo 4517-C (sensibilidade de 0.18 pC=m², e massa de 0.6 g).

Os 200 primeiros pontos discretos da FRF, ou seja, depois que estes foram transformados para o domínio da frequência pela Transformada de Fourier, foram descartados. Este descarte é justificado pelo ruído presente nesta região, como pode ser observado na Fig. 15. Foram coletados 4400 pontos adiante dessa região, já que, assim, preserva-se os três máximos da Fig. 15.

Figura 13 – Vistas lateral e frontal das placas, com destaque para a região enegrecida pela presença da tira de Teflon.



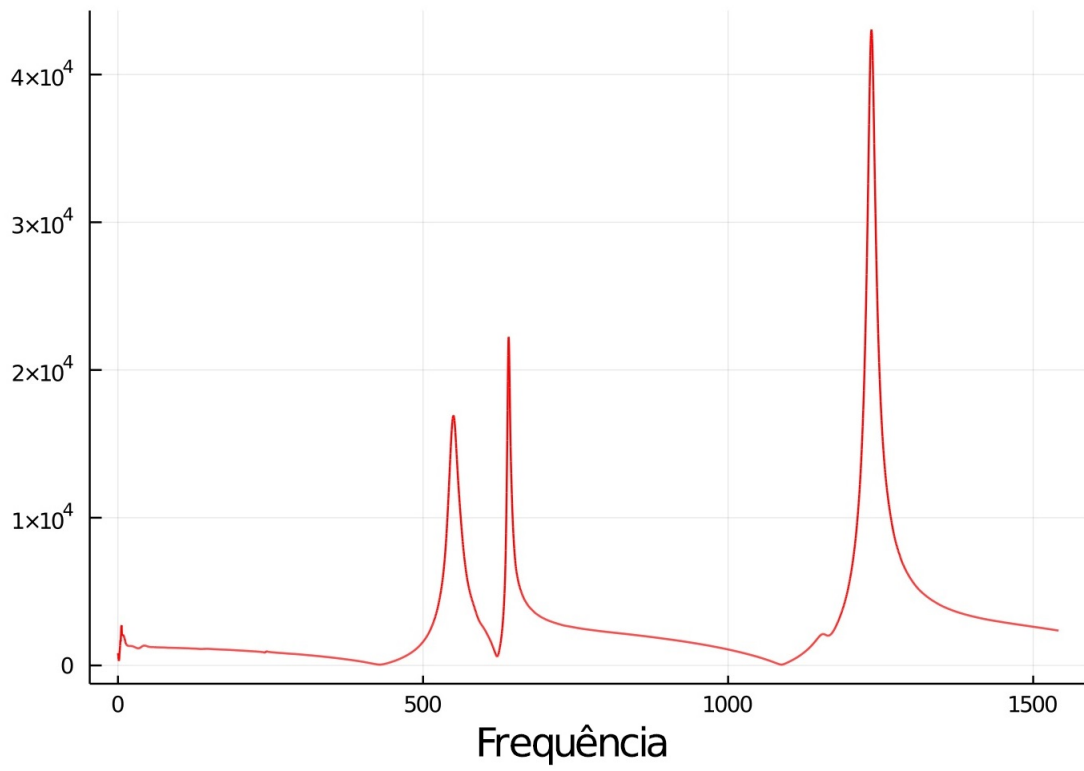
Fonte: Voltz (2019)

Figura 14 – Representação esquemática da placa.



Fonte: Voltz (2019)

Figura 15 – FRF de uma amostra com dano D1.



Fonte: Produção do autor.

5.2 DEFINIÇÃO DO PROBLEMA DE CLASSIFICAÇÃO

Haja vista à forma com que as placas de material compósito foram produzidas, existem duas formas de classificar:

1. Se uma amostra é danificada ou não.
2. Se uma amostra é intacta ou, se for danificada, qual tipo de dano.

Para o primeiro modo de classificar, foram adotados dois caminhos: dois neurônios na camada de *output*; um neurônio na camada de *output*. Enquanto, para o segundo modo, foi selecionado apenas um meio, com 4 neurônios na camada de *output*. A parametrização dos valores esperados de respostas estão descritos nas Tab. 1, 2 e 3.

Tabela 1 – Casos de classificação e a resposta do neurônio utilizando apenas um neurônio na camada de *output*.

Classificação	Neurônio de <i>output</i> I
Intacta	1
Danificada	0

Fonte: Produção do autor.

Tabela 2 – Casos de classificação e a resposta dos neurônios utilizando dois neurônios na camada de *output*.

Classificação	Neurônio de <i>output</i> I	Neurônio de <i>output</i> II
Intacta	1	0
Danificada	0	1

Fonte: Produção do autor.

Tabela 3 – Casos de classificação e a resposta dos neurônios utilizando quatro neurônios na camada de *output*.

Classificação	Neurônio de <i>output</i> I	Neurônio de <i>output</i> II	Neurônio de <i>output</i> III	Neurônio de <i>output</i> IV
Intacta	1	0	0	0
D1	0	1	0	0
D2	0	0	1	0
D3	0	0	0	1

Fonte: Produção do autor.

A forma de construir a resposta nas Tab. 2 e 3 é própria para o uso da função custo *cross-entropy*, uma vez que o valor 1 representa que a probabilidade da amostra pertencer àquela

classe é de 100%, ao passo que é nula de pertencer as outras classes. A escolha da *cross-entropy* deve-se à recomendação da literatura. Evidencia-se, contudo, que, na aplicação das RNAs, os valores 1 foram convertidos para 0,9999 e os valores 0 para 0,0001, conforme a indicação de Brownlee (2019), para que o cálculo do logaritmo dentro da função *cross-entropy* não vá divergir nas proximidades de um argumento nulo. No caso da Tab. 1, como há apenas um neurônio no output, de modo que não se tem como criar uma distribuição de probabilidade com um único valor, a função custo utilizada é a MSE.

5.3 CONFIGURAÇÃO DE HIPERPARÂMETROS

5.3.1 Hiperparâmetros fixos

Com base nos objetivos destacados no primeiro parágrafo deste capítulo, evidencia-se que alguns hiperparâmetros de RNA foram fixados, posto que suas esferas de influência não estão condicionadas ao tratamento dos dados; e, sim, à construção da rede. Dessa forma, buscou-se que esses hiperparâmetros imutados estivessem em consonância com o que recomenda a bibliografia referenciada; estando, portanto, convenientemente introduzidas no capítulo 3, as razões para as escolhas tomadas. Otimizador Adam com taxa de aprendizado, *eta*, de 0,3; taxa de decaimento do primeiro momento, β_1 , de 0,9; taxa de decaimento do segundo momento, β_2 , de 0.999; 5000 iterações de otimização. Enquanto as topologias foram todas do tipo densa.

5.3.2 Hiperparâmetros variáveis

5.3.2.1 Divisão do conjunto de dados

A divisão utilizada dos dados foi em treinamento, validação e teste; tal que as percentagens de amostras para cada um destes conjuntos foram inicialmente: 75%, 15% e 10%, respectivamente. O que é coerente com uma gama de trabalhos, a exemplo de: Looney (1996), que sugeriu 65% para treinamento, 10% para validação e 25% para teste; Silva, Spatti e Flauzino (2010a), que propuseram 60 a 90% para treinamento e validação. Extraíndo uma média dessas proposições, chega-se ao valor utilizado neste trabalho, que, por sua vez, é muito semelhante aos 70% de treinamento, 20% de validação e 10% de teste de Pereira e Bezerra (2007).

Notou-se, porém, que havia a necessidade de experimentar diferentes configurações de divisão de *data sets*. Assim, foram feitas mais segmentações utilizando a fração de amostras do conjunto de treinamento como critério principal, visto que é nesse *data set* em que os modelos realizam seu aprendizado. Logo, foram escolhidos limites acima e abaixo dos 75%, os quais foram de 50% e de 80%; sendo o limite inferior em virtude de que uma rede treinada com menos da metade dos dados torna-se instável, justamente pelo o que se comentou sobre a falta de exposição a um número significativos de possíveis combinações de variáveis; ao passo que o limite superior deve-se à inviabilidade econômica de treinar uma rede que necessita de uma fração tão expressiva dos dados para ser treinada. Por fim, foi selecionado um ponto médio entre

o limite superior e o valor configurado de 75%, para que este intervalo fosse subdividido e para que possíveis nuances na performance das redes fossem detectadas.

Destaca-se, entretanto, que os conjuntos de validação e teste permaneceram com razões proporcionais às configuradas na divisão inicial, de modo que, portanto, o *data set* de validação continuou com 60% das amostras que não foram utilizadas para treinamento e o *data set de teste* com 40% dessas amostras. Sintetiza-se na Tabela 4 as quatro configurações utilizadas.

Tabela 4 – As quatro divisões do conjunto de dados original e suas respectivas frações de amostras para cada *data set*.

Divisão	Treinamento	Validação	Teste
I	50%	30%	20%
II	63%	23%	14%
III	75%	15%	10%
IV	80%	12%	8%

Fonte: Produção do autor.

5.3.2.2 Aplicação dos métodos compressivos

Dos 6400 pontos experimentais coletados, as 200 medições iniciais, no domínio da frequência, do acelerômetro foram descartadas para evitar o prejuízo do ruído presente nessa parte da FRF. 4400 pontos espectrais, então, foram selecionados à frente destes retirados. Para o PCA individualmente, foi selecionado um critério de 98% de manutenção de variância dos dados, o que foi atingido com 30 componentes principais.

O LDA foi implementado considerando duas classes - intacto e danificado - e considerando quatro classes - intacto e os três tipos de dano. Em ambos os arranjos, foi experimentado também o uso de LDA com posto reduzido, ou seja, utilizando menos discriminantes lineares do que o número de classes subtraído de um. Por fim, foi aferida a performance do emprego do LDA na sequência de um PCA como pré-processamento.

Como esses métodos são lineares, é importante que seja definido sobre qual ou quais conjuntos a matriz de transformação será calculada. Para o LDA, é evidente que seja no *data set* de treinamento, haja vista que também é um método classificatório e, portanto, deve ser *treinado*, o que torna a escolha do conjunto de treinamento óbvia, pois é a única parcela dos dados a que se deseja expor todo o aparato classificatório - método compressor e modelo de RNA. Para o PCA, entretanto, o objetivo não é diferente: deve-se evitar *contaminar* os dados de treinamento da rede neural com amostras de outros conjuntos, logo a matriz de covariância deve ser calculada somente sobre as amostras selecionadas para treinar o modelo de rede neural, conforme especificam Karpathy (2017), Ng (2018) e Vidhya (2016). Dessa forma, a matriz de mudança de base do PCA será calculada com essas amostras e será reutilizada para comprimir os *data sets* de validação e de teste.

5.3.2.3 Configuração das topologias

Ainda que os números de neurônios nas camadas de *input* e *output* sejam previamente definidos, este pelo tipo da resposta e aquele pela dimensionalidade final dos dados, surge a necessidade de determinar o número de camadas internas e a quantidade de neurônios em cada uma delas. Sendo definido aquele para as simulações deste trabalho em 1, 2 e 3 camadas internas; enquanto as quantidades de neurônios em cada camada sendo definidas por uma função.

Adota-se para o desenvolvimento do conceito para essa função de distribuição de neurônios a convenção de visualização de uma topologia densa apresentada na seção 3.2, isto é, a que dispõe as camadas ao longo do eixo de propagação da informação, horizontal, e que dispõe os neurônios de cada camada num eixo desta e perpendicular, vertical, ao último. Convencionou-se ainda que a i -ésima camada (conta-se aqui todas as camadas juntas, o que inclui as de *input* e de *output*) está na posição i sobre o eixo de propagação de informação. Seja o comprimento de um vetor colinear ao eixo dessa i -ésima camada, portanto, o número de neurônios desta. Se este procedimento for repetido em todas as camadas, como o número de neurônios é uma quantidade positiva, o resultado será uma distribuição de pontos distantes um do outro com um passo unitário ao longo do eixo horizontal e cuja distância deste informa a quantidade de neurônios nas respectivas camadas. Essa distribuição de pontos pode ser ligada por uma curva, e a função que parametriza essa curva é justamente a função procurada, posto que, se o número da camada for colocado como argumento daquela, o resultado será a quantidade de neurônios.

Visando simplicidade, a curva anteriormente descrita pode ser definida por uma função polinomial, $p(s)$, cujo argumento é o número de ordem da camada, s ,

$$p(s) = a_0 + a_1 (s - c_1)^{b_1} + a_2 (s - c_2)^{b_2} + \dots + a_{q-1} (s - c_{q-1})^{b_{q-1}}, \quad (19)$$

em que a , b e c são constantes. A vantagem imediata desse tipo de parametrização é que pela configuração dos expoentes, b , pode-se modular a forma da curva e, por intermédio dos zeros, c , pode-se alterar a posição de pontos críticos. A segunda e mais importante vantagem para esta discussão, contudo, está no desenvolvimento de uma ferramenta análoga à matriz de Vandermonde, pois é possível que seja construída uma matriz $q \times q$, tal que q é o número de condições prescritas, a exemplo de pontos por onde se deseja que a curva passe e valores para as derivadas primeira e segunda. Dessa maneira, para cada linha, é aplicado em p a localização da condição prescrita e, na j -ésima coluna, tem o valor da $(j-1)$ -ésima potência dessa localização (multiplicada de uma subtração de um expoente caso a prescrição seja sobre uma derivada). Logo, ao resolver o sistema linear dessa matriz de condições prescritas com o vetor dos valores de p aplicado nas localizações das prescrições, ter-se-á um vetor com os coeficientes a da equação (19), de modo que, então, $p(s)$ estará completamente definido.

Como a arquitetura da RNA influencia preponderantemente em seu desempenho e como essa arquitetura está sendo condicionada à uma curva parametrizada por uma função, é natural que propriedades geométricas deste não de influenciar o desempenho, a exemplo da curvatura e

da presença ou ausência de pontos críticos. Logo, para que um ponto crítico ou uma dinâmica de curvatura possa ser replicada em mais curvas e para que o espaço entre duas parametrizações possa ser igualmente varrido em busca de padrões de performance, lança-se mão à inspiração pelo método da bisseção. Tal que o intervalo entre duas curvas seja particionado por r vezes, o que levará à existência de $2^r + 1$ curvas no total no final das r partições do intervalo. E, em verdade, cada função gerada pode ser criada linear ou não-linearmente como uma combinação de suas curvas adjacentes,

$$p_{d,r}(s) = v p_{\frac{d}{2},r-1}(s) + w p_{\frac{d}{2}+1,r-1}(s), \quad (20)$$

a forma linear, em que o primeiro índice indica a ordem de posição da curva na etapa de bisseção definida pelo segundo índice; v e w são constantes que somadas igualam a 1, ou seja, a equação (20) trata-se de uma média ponderada das funções polinomiais adjacentes à curva. E a forma não-linear,

$$p_{d,r}(s) = p_{\frac{d}{2},r-1}(s)^f p_{\frac{d}{2}+1,r-1}(s)^g, \quad (21)$$

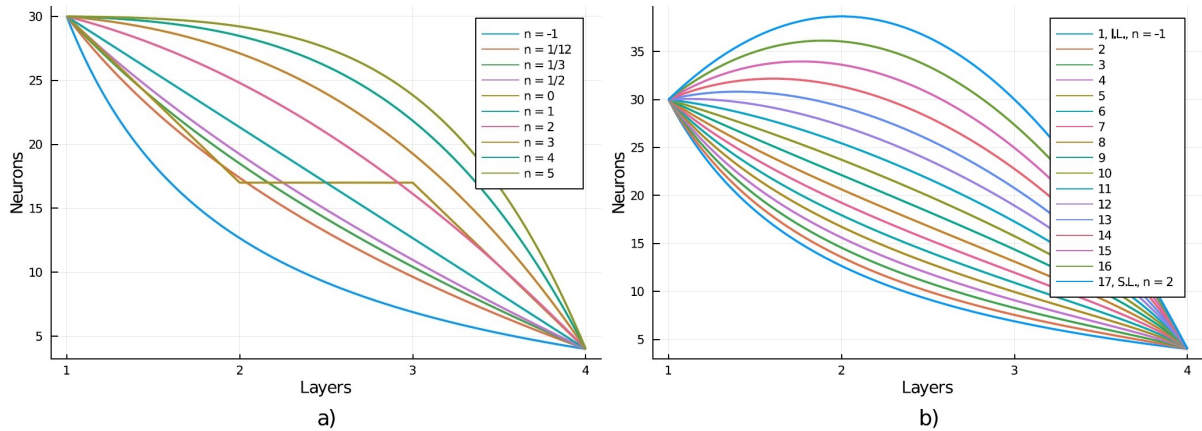
tal que f e g são expoentes e constantes que, novamente, quando somados igualam a 1.

A partir do fato de que a nova curva, definida pela equação (20) ou pela equação (21), está entre duas curvas que a precederam nas etapas de bisseção, é evidente que determinar um *limite inferior* e um *limite superior* é necessário. Serão esses limites, em consequente, que carregarão as dinâmicas desejadas de curvatura e de pontos críticos e/ou prescritos e, para que o intervalo entre esses limites seja coesamente percorrido por um número de trajetórias, são traçadas as curvas de bisseção, que podem ser moduladas linear ou não-linearmente para aproximar-se mais ou menos de um limite em detrimento do outro.

Essa modulação das parametrizações pode ser utilizada para confinar as topologias no retângulo formado pelas redes FF (utilizando a convenção aqui adotada para diferenciar FF de DFF, em que redes FF tem seu número máximo de neurônios por camada limitado superiormente pelo número de neurônios no output) e fazê-las diminuírem monotonicamente, do input ao output, em número de neurônios por camada; ou, por exemplo, é possível que um máximo ou um ponto prescrito sejam colocados acima do retângulo de FF para que sejam criadas então arquiteturas DFF. Na Figura 16, observa-se à direita, no gráfico a, uma série de curvas construídas pela configuração individual do grau polinomial, o qual foi inserido nas legendas dessa figura; ao passo que, no gráfico b, utiliza-se a técnica da equação (21), e é possível perceber como esta é capaz de varrer com sucesso o espectro delimitado pelos limites superior e inferior.

Para o uso como uma topologia, os resultados das parametrizações são avaliados na posição de cada camada e arredondados para um inteiro. Destaca-se ainda que, para as redes Deep Feedforward com apenas uma camada interna, a quantidade de neurônios nesta foi aumentado de um em um até o dobro do número de neurônios de inputs, no caso do PCA, e até o quántuplo desse

Figura 16 – Parametrizações de topologias num gráfico cuja abscissa corresponde à posição de cada camada e a ordenada corresponde ao número de neurônios em cada camada.



Fonte: Produção do autor.

número para o caso do LDA. Há de se notar, entretanto, que a abordagem genética apresentada por Voltz (2019) não foi abandonada completamente; em verdade, uma abordagem evolutiva foi aprimorada, de forma que, pelo desempenho das RNAs no *data set* de validação, foi possível unir e preservar características positivas e eliminar as prejudiciais, como sugere Brownlee (2017b), no uso da fase de validação para a melhora dos hiperparâmetros.

5.3.2.4 Funções de ativação

As funções de ativação utilizadas foram sigmóide para os testes com PCA apenas e sigmóide, Leaky-ReLU e ELU para os experimentos com LDA e PCA com LDA na sequência. Exceto nas topologias com um neurônio de output apenas, todas as arquiteturas apresentavam softmax na última camada e nenhuma função de ativação da penúltima para a última, para garantir estabilidade no cálculo do gradiente. No caso da implementação de LDA e PCA com LDA com mais de uma camada intermediária, as funções de ativação Leaky-ReLU e ELU foram utilizadas apenas na primeira camada intermediária, para que houvesse a convergência para valores entre 0 e 1 nas camadas subsequentes.

5.3.2.5 Realizações

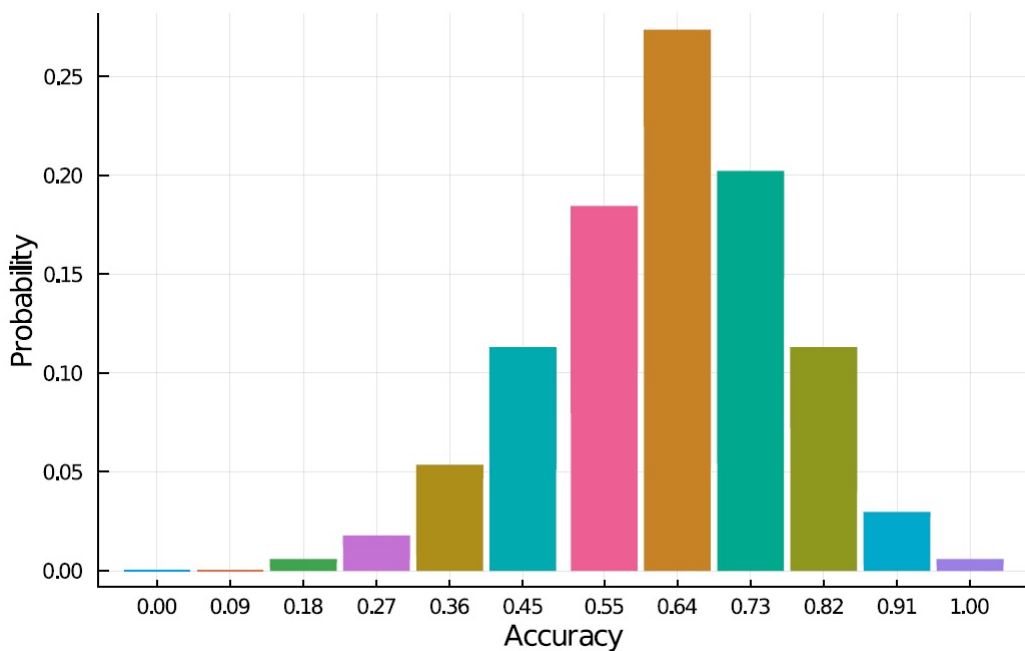
Dado o comportamento probabilístico de uma arquitetura de RNA atingir um desempenho quando seus parâmetros forem inicializados randomicamente, cada uma daquelas foi treinada por 500 vezes, isto é, foram gerados 500 modelos para cada topologia. Neste trabalho, o processo de gerar cada modelo foi chamado de *realização*. O objetivo com este número é verificar a convergência de alguns parâmetros de desempenho, pois, se houver convergência, pode-se construir PMFs (*Probability Mass Functions*, do inglês para Funções de Massa de Probabilidade), que mostram basicamente distribuições de probabilidade discretas. Essas PMFs são usadas, na sequência, como critério de comparação entre as diferentes topologias, o que pode ser ilustrado

por uma PMF de acurácia, como na Fig. 17: se esta tem um máximo, é de interesse que este pico esteja o quanto mais próximo da acurácia de 100% possível; além de que o valor da PMF nesse nível de acurácia é de suma importância também.

Sendo os parâmetros de desempenho mais importantes neste trabalho o erro no treinamento e as acurácias nos *data sets* de validação e treinamento, é trivial que estas já tenham uma quantidade finita, igual a $k + 1$, de valores possíveis de acurácia, em que k é o número de amostras no *data set* em questão. Para o erro no treinamento, entretanto, não é o caso, já que, como o valor da função custo na última iteração de treinamento, pode assumir infinitas possibilidades de resultados. Há, em consequente, a necessidade de dividir esses erros em bandas, as quais foram definidas pela ordem de magnitude, que, por sua vez, tem uma ligação intrínseca com acurácia no conjunto de treinamento, sobretudo se for utilizada a função *cross-entropy*, e permite, portanto, vislumbrar se há over ou underfitting ou nenhum dos dois fenômenos.

Para construir as PMFs, a quantidade de cada modelos em cada banda é dividida pela quantidade total deles, quociente este que justamente imbui o conceito de probabilidade na análise. No caso do erro no treinamento, esse quociente é calculado à cada realização, justamente para que se verifique a convergência, como é visível na Fig. 18.

Figura 17 – Exemplo de PMF de acurácia.

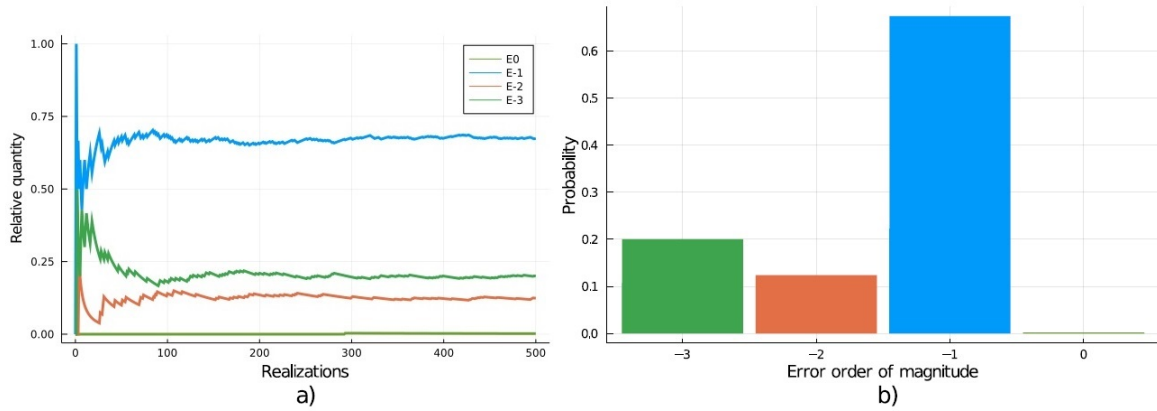


Fonte: Produção do autor.

5.3.2.6 Pontuação

Para que um modelo seja coerentemente comparado com outro no âmbito da habilidade de generalização, é introduzido o parâmetro *score*, que executa a tarefa de pontuar um modelo,

Figura 18 – Exemplo de PMF de erros no treinamento que converge para uma distribuição ao longo das 500 realizações no gráfico a) e a PMF já convergida em b). A ordem de magnitude está na legenda em a).



Fonte: Produção do autor.

para o que usa a seguinte equação

$$score = \frac{k_{tr}a_{tr} + k_v a_v + k_{te} a_{te}}{k_{tr} + k_v + k_{te}}, \quad (22)$$

em que k_{tr} representa o número de amostras no data set de treinamento; k_v , o número de amostras no data set de validação; k_{te} , o número de amostras no data set de teste; a , a acurácia, e os índices seguem a mesma regra que para k . Este conceito pode ser extendido para o contexto de toda uma topologia, que é composta, por sua vez, de 500 modelos, cada um com seu score. O objetivo é, então, criar um score de toda a topologia, de maneira que seja possível comparar arquiteturas entre si. O qual será calculado por

$$Score = \sum_{i=1}^N score_i \gamma_i, \quad (23)$$

tal que N é o número dos maiores scores selecionados; γ_i , o número de modelos de uma topologia com o i -ésimo score. Para este trabalho, foi selecionado um score mínimo de 0,95, que equivale a uma acurácia de 80% nos data sets de validação e de teste. Ou seja, exige-se que, para efeitos de comparação, o modelo atinja pelo menos este nível de assertividade.

6 RESULTADOS

O termo *simulação* é usado neste trabalho para definir um conjunto de topologias com hiperparâmetros em comum. Foram realizadas 320 simulações, posto que foram tomadas as seguintes variações de hiperparâmetros: três opções para a camada de output; com uma e com duas camadas internas; compressão de dados com PCA somente, LDA somente e PCA seguido de LDA; variação de 1 a 3 discriminantes lineares; três funções de ativação, sigmóide, ELU e Leaky ReLU; quatro frações de amostras para treinamento. Todas as combinações destes hiperparâmetros foram experimentadas, sendo que as arquiteturas foram construídas pela técnica de parametrização da subsubseção 5.3.2.3 ou pela aumento de um em um no número de neurônios na camada intermediária para as topologias com apenas uma daquelas. As arquiteturas, por sua vez, simuladas estão descritas no anexo A, de modo que uma tabela com os hiperparâmetros é informada para as topologias com uma camada intermediária e, para duas camadas internas, são fornecidas num gráfico na sequência, além da tabela, as parametrizações que levaram a esta configuração.

O primeiro resultado, que já foi percebido na fase de validação, foi a inabilidade de generalização de redes com um neurônio de output apenas, mesmo que apresentem um erro pequeno no *data set* de treinamento, o que demonstra a tendência ao *overfitting* desta configuração. Em verdade, isto é consequência do uso da função custo MSE, que produz modelos que se ajustam bem às amostras de treinamento mas que, quando aplicados a outros conjuntos, têm resultados muito pobres. No entanto, como já explicado, não é possível que a função *cross-entropy* seja empregada para essa configuração. Ainda na questão sobre quantidade de neurônios de *output*, percebeu-se que a classificação com dois neurônios nesta posição, isto é, apenas se existe ou não dano, foi muito mais acurada do que a classificação com quatro neurônios de output - se intacta ou dano 1, dano 2 ou dano 3. Ou seja, é mais difícil para RNA identificar o tipo de dano do que apenas identificar sua presença.

Sobre as funções de ativação, notou-se que a sigmóide é muito superior à Leaky ReLU e à ELU para esta aplicação, sobretudo no uso de apenas PCA para compressão de dados. A vantagem da sigmóide sobre as outras funções de ativação diminui um pouco no caso do emprego de LDA ou PCA com LDA; não há, contudo, superioridade alguma dessas outras duas funções de ativação, como era de se esperar pela separação e amplificação de amostras de classes diferentes que estejam próximas no espaço de suas variáveis. Para uma comparação quantitativa, no caso do PCA, a utilização de sigmóide levou a dezenas de modelos com acurácias de 100% para cada topologia; ao passo que, com as outras duas funções de ativação, não houve sequer um bom ajuste ao conjunto de treinamento, o que é o sinal de *underfitting*, ocasionado pela impossibilidade de convergência dessas funções. A performance da Leaky ReLU e da ELU melhora caso sejam utilizadas duas camadas internas, de modo que, da primeira para segunda, sejam posicionadas sigmóides para estabilizar a amplificação proporcionada pela Leaky ReLU e pela ELU.

Na questão sobre camadas internas, percebe-se que, para o uso do PCA, uma camada

intermediária apenas com mais neurônios é preferível e garante resultados muito melhores que o uso de mais de uma camada interna. Isto se justifica pela problemática do gradiente que desaparece (*vanishing gradient*), em que este ganha complexidade, aumento de custo computacional no seu cálculo e uma tendência a decair em seu valor com o aumento da não-linearidade causada pelo número excessivo de camadas. Assim, redes com muitas destas tendem ao *underfitting* e à incapacidade de generalização.

No caso do LDA ou de PCA com LDA, por outro lado, a utilização de duas camadas internas produz bons resultados, visto que o número de inputs é baixo nesta situação e a introdução de não-linearidades auxilia a rede a lidar com as não-linearidades presentes em amostras com poucas variáveis, analogamente à solução do problema XOR em Haykin (2007). De fato, não foi incomum o pico na PMF de acurácias deslocar-se para uma acurácia duas ou três vezes maior, como de 25% para 50% ou para 75%.

O número de inputs influencia fortemente o desempenho do LDA ou de sua combinação com PCA. Como foi descrito, foi testado o uso de LDA com posto reduzido, ou seja, o emprego de menos variáveis do que os $c - 1$ discriminantes lineares resultantes do método. Nas simulações, ficou evidente que o desempenho do LDA melhora com 2 variáveis em vez das 3 originalmente calculadas; ao passo que, quando o PCA é feito antes do LDA, o uso de 3 discriminantes lineares traz acurácias maiores. De fato, os resultados conquistados com o PCA feito como pré-processamento para o LDA foram muito superiores, posto que esta configuração não tem a tendência ao *overfitting* que o LDA somente apresentou.

A justificativa para esse *overfitting* é visível nos gráficos com as dispersões dos pontos de cada amostra após a aplicação dos métodos. No caso do uso de somente LDA, é perceptível que todas as amostras de uma classe caíram sobre uma mesma posição para essa classe, enquanto as amostras de validação e teste ficaram dispersas, fora de seus *loci* por muitas vezes, de modo que o próprio método apresentou baixa acurácia e induz, portanto, a RNA ao erro. Este problema decresceu consideravelmente quando PCA foi aplicado nos dados de treinamento numa etapa anterior. Conforme Martínez e Kak (2001), o LDA apresenta um desempenho inferior ao PCA quando o número de amostras por classe é baixo comparado com o número de variáveis, embora não seja fornecido na literatura um critério sólido a respeito. Segundo Raschka (2014), o PCA como uma etapa de pré-processamento pode melhorar a performance do LDA, como de fato aconteceu, e isso se deve ao fato de a dimensionalidade ter sido prévia e drasticamente reduzida.

Na comparação direta de desempenho de redes neurais construídas e treinadas com alguma variante de aplicação de LDA entre aquelas treinadas com PCA somente, percebe-se que estas produziram resultados muito mais satisfatórios, com uma grande folga de vantagem. Esta vantagem só não foi vista na divisão de 63% das amostras para treinamento, em que a performance do LDA e suas combinações equiparou-se com a do PCA. Nesta circunstância de poucas amostras por classe em relação ao número de variáveis (15 amostras de dano 1 frente a 4400 variáveis nos dados *crus*), o LDA tende a levar a rede ao *overfitting*, de maneira que sua capacidade de generalização fica muito prejudicada e tem-se picos da PMF de acurácia em

valores pífios de acurácia, como 25-50 %, contra 70-80 % no uso exclusivo de PCA. Nota-se, porém, um padrão na construção das arquiteturas que geraram os melhores resultados: o PCA produz os modelos mais bem sucedidos quando o número de neurônios na camada interna é consideravelmente maior que o de neurônios na camada de input, a exemplo das topologias descritas no anexo A; enquanto, o LDA traz seus melhores resultados em arquiteturas cuja primeira camada interna tenha o número de neurônios igual ou pouquíssimo maior que o número de neurônios no input, como foi o caso das topologias 3/3/3/4 e 3/4/4/4 e 3/3/4/4, em que estão as quantidades de neurônio em ordem.

Estes padrões se repetem, de fato, em todas as quatro opções de razões de divisão do conjunto de dados original, o que demonstra que existe uma estrutura matemática intrínseca à topologia que propicia um máximo de desempenho de acurácia e capacidade de generalização nessas configurações de arquitetura. Como era de se esperar, entretanto, essa capacidade de generalização diminuiu com a percentagem de amostras destinadas para o treinamento, uma vez que o número de combinações de variáveis a que a rede foi exposta diminuiu proporcionalmente. Essa diminuição da generalização foi percebida no deslocamento do máximo da PMF de acurácias para a esquerda e foi particularmente vista no uso de 50% das amostras para o conjunto de treinamento. Houve uma melhora significativa desta razão para a de 63 %, embora ainda muito menos expressivo que as PMFs de acurácias encontradas para 75% ou 80% dos dados para treinamento, como esperado.

Assim, torna-se oportuno que uma investigação futura seja conduzida neste intervalo entre 63 e 75%. Ainda sobre essas divisões, notou-se uma anomalia no data set com 63% de amostra para treinamento, uma vez que as suas amostras de teste ficaram particularmente bem distribuídos e propiciaram grande facilidade de classificação para as redes, com um máximo de 28 modelos com acurácia igual a 1 para a topologia 3/3/2/2, com sigmóide apenas como função de ativação, treinada a partir de dados que passaram por PCA e LDA. Na tabela a seguir, visualiza-se comparações e as melhores arquiteturas. Nas duas colunas da direita, tem-se, a cada linha, o número de modelos que tiveram acurácia igual a 1 no data set da coluna e em qual valor de acurácia encontra-se o pico da PMF.

Embora o uso de PCA com LDA mostre resultados impressionantes, como os 22 modelos com acurácia igual a 1 no data set de teste, como se pode ver na Tabela 13, deve-se prestar atenção no *Score*, posto que este mede a performance do modelo em todos os conjuntos de dados ao mesmo tempo, de modo que um desempenho ruim num data set penaliza um excelente resultado num outro data set. Essa penalização é feita proporcionalmente ao tamanho do conjunto, como é visível em (22). Assim, o objetivo do *Score* é aferir a habilidade de generalização de uma arquitetura e é por esse critério que a utilização de PCA com LDA cai em desfavor em relação ao uso de PCA somente, por mais tênue que esta diferença tenha ficado nas divisões de treinamento intermediárias, sobretudo a de 63%, ou seja, a divisão II.

Averigua-se, portanto, conforme os gráficos de dispersão com as fronteiras de decisão no anexo B, em que é dado o exemplo com o conjunto com 75% das amostras para treinamento,

Tabela 5 – Características e estatísticas de desempenho de topologias com resultados notáveis.

Topologia	Função de ativação	Método compressor	Fração de treinamento	Score	Acurácia na validação	Acurácia no teste
22/46/2	Sigmóide	PCA	50%	0	0 e 0,636	0 e 0,733
22/55/2	Sigmóide	PCA	50%	0,959	0 e 0,636	0 e 0,733
3/7/6/2	Sigmóide	PCA-LDA	50%	0	0 e 0,682	0 e 0,733
22/37/4	Sigmóide	PCA	50%	0	0 e 0,5	0 e 0,6
22/49/4	Sigmóide	PCA	50%	0	0 e 0,591	0 e 0,533
3/5/5/4	Sigmóide	PCA-LDA	50%	0	0 e 0,545	0 e 0,0
26/44/2	Sigmóide	PCA	63%	2,89	0 e 0,647	3 e 0,700
26/56/2	Sigmóide	PCA	63%	5,80	0 e 0,706	4 e 0,700
3/4/4/2	Sigmóide	PCA-LDA	63%	6,71	0 e 0,765	22 e 1,0
25/33/4	Sigmóide	PCA	63%	0,973	1 e 0,529	0 e 0,6
25/46/4	Sigmóide	PCA	63%	0,959	0 e 0,588	0 e 0,8
3/6/6/4	Sigmóide	PCA-LDA	63%	0,959	0 e 0,706	1 e 0,8
30/45/2	Sigmóide	PCA	75%	68,6	10 e 0,727	1 e 0,714
30/53/2	Sigmóide	PCA	75%	63,8	9 e 0,727	2 e 0,714
3/4/4/2	Sigmóide	PCA-LDA	75%	20,1	31 e 0,909	0 e 0,571
30/42/4	Sigmóide	PCA	75%	5,81	4 e 0,636	1 e 0,286
30/54/4	Sigmóide	PCA	75%	5,75	4 e 0,636	0 e 0,571
3/5/5/4	Sigmóide	PCA-LDA	75%	0,959	2 e 0,727	0 e 0,571
31/44/2	Sigmóide	PCA	80%	47,4	9 e 0,667	2 e 0,667
31/55/2	Sigmóide	PCA	80%	49,1	3 e 0,667	0 e 0,667
3/4/4/2	Sigmóide	PCA-LDA	80%	0	0 e 0,667	0 e 0,8
30/37/4	Sigmóide	PCA	80%	5,84	0 e 0,556	2 e 0,8
30/52/4	Sigmóide	PCA	80%	3,92	1 e 0,556	1 e 0,8
3/5/4/4	Sigmóide	PCA-LDA	80%	0	0 e 0,333	0 e 0,0

Fonte: Produção do autor.

que a falha em torno do LDA está na incapacidade, neste caso, de classificar as amostras de forma assertiva antes que a rede neural absorva esses dados. Isto é, como é visível nos gráficos, o LDA mostra-se instável para uma aplicação com uma *convexidade*, como se define neste trabalho a razão de número de amostras por classe pelo número de variáveis, menor que 1. Outro fator que contribui para a instabilidade é a presença de classes ocultas, conforme alerta Steorts (2017), pois a existência desses fatores que não foram contabilizados ou informados confundem totalmente a característica principal do LDA: ser voltado para a visualização e segregação de classes. Em verdade, foram feitos testes para determinar como deveriam ser separadas as amostras de treinamento com respeito à classe, na ocasião da classificação binária - com ou sem dano. Inicialmente, foram separadas em amostras intactas e danificadas; o resultado desta configuração foi muito insatisfatório e enviesado, justamente pela presença das diferentes classes

de danos que não foram informadas. Logo, por toda a extensão do estudo, mesmo no problema de classificação binária, as amostras tiveram que ser informadas como pertencente a sua classe de dano em específico.

Verificou-se que topologias com parametrizações de curvaturas maiores, ou seja, topologias com mais neurônios na camada interna, têm suas PMFs de erro no treinamento convergindo para ordens de magnitudes menores, a saber: -3 e -2, utilizando *cross-entropy*. Em verdade, essas ordens de grandeza são perceptíveis com menos neurônios nessa camada, mas o que diferencia do caso com mais neurônios nessa camada é a PMF de acurácia, visto que, quando a quantidade de neurônios na camada interna se aproxima e passa da quantidade na camada de *input*, o pico da PMF desloca-se para a direita e a capacidade de generalização da rede aumenta. Foi observado também que a quantidade de modelos com acurácia de 100% e o score de topologia oscilam com amplitudes cada vez maiores até chegar no pico de performance para a configuração com uma camada interna e, na sequência, oscilam com valores de score e quantidade de bons modelos cada vez menores. Sendo que as topologias treinadas com PCA apresentaram dois máximos de performance; na Tabela 13, tem-se o segundo pico, que apresentou os melhores resultados. Padrão que foi trocado apenas nas simulações com 80% de amostras para treinamento, de modo que os dois máximos foram colocados nessa tabela.

Na Tabela 13, não estão dispostos os testes com outras funções de ativação, o que, como descrito anteriormente, justifica-se pelos seus péssimos desempenhos de acurácia e de treinamento. A mesma justificativa é dada pela não inclusão dos resultados com LDA somente, haja vista que também foram ruins. Há, no entanto, de se reparar que, além do PCA ter superado o uso de PCA com LDA em todos as quatro opções de divisão do conjunto de dados original, existe um padrão: é evidente que o primeiro e o segundo picos de desempenho no caso de classificação binária mantêm-se em torno de números de neurônio na camada intermediária de 44,75 e 54,75, com desvios padrões de 0,957 e 1,26 respectivamente; ao passo que, para a classificação em quatro classes, percebe-se um pico em torno de 37,25 e outro em torno de 50,25 neurônios por camada, com desvios padrões de 3,69 e 3,50 respectivamente. Logo, é notável que a hierarquia das arquiteturas e, por conseguinte, a escolha das melhores topologias são invariantes com a mudança de fração de amostras para treinamento, visto que os desvios padrões são pequenos quando comparados com os valores médios, o que implica na existência do agrupamento dessa quantidade de neurônios na camada intermediária, que, como visto, é invariante até mesmo ao número de neurônios na camada de *input*.

A variação na dimensionalidade do conjunto de dados fornecido à rede neural quando aplicado o PCA é função direta da quantidade de casos particulares e diferentes de combinações de variáveis. Pois, quanto menor a razão de amostras para o treinamento, menor será a diversidade de casos exposta à rede, de modo que a demanda por PCs para manter o critério de variância diminui, porque a variabilidade dos dados decaiu. Caso as diferentes classes sejam distribuídas da forma mais homogênea possível entre os *data sets* no momento da divisão das amostras, não ocorre o enviesamento do treinamento do modelo para uma classe. O que, por fim, implicará na

invariância da topologia experienciada. Os maiores desvios padrões e distância entre picos para a classificação em quatro classes corroboram que essa forma de classificação é mais instável e custosa para as RNAs, e que o número de outputs deve influir no número de neurônios na camada intermediária, de tal forma que este não aparenta ser invariante àquele.

A invariância também foi notada nas topologias com uso de PCA seguido de LDA, tal que as melhores topologias contam com um número de neurônios levemente maior na primeira camada intermediária quando comparado à quantidade de neurônios na camada de input, sendo que, então, o número de neurônios na segunda camada interna diminui suavemente para a quantidade de neurônios no output. Dessa forma, revela-se, por consequência, que a metodologia de parametrizar as topologias por meio de curvas suaves é eficaz e mais importante ainda: é uma metodologia capaz de identificar configurações ótimas, posto que, com base nas simulações, foi evidente um fenômeno de melhora do desempenho ao longo das topologias até a opção ótima e, a partir da qual, houve uma piora, tal que ficou evidente qual é a melhor arquitetura para cada modo de classificação, em diferentes divisões de data sets.

7 CONCLUSÃO

Conclui-se que o PCA é ainda uma estratégia mais segura e conservadora para a compressão de dados antes da aplicação em RNA, em detrimento do LDA ou de PCA com LDA, na situação de poucas amostras para muitas variáveis. Mostrou-se, porém, que o LDA, com o auxílio do PCA, é uma ferramenta que também produz bons resultados, embora sejam anômalos e de boa qualidade num *data set* apenas. Logo, torna-se oportuno um estudo futuro para investigar qual seria um valor crítico para a convexidade no contexto da aplicação estudada.

Foi comprovado que o uso da sigmóide como função de ativação com número de neurônios de *output* igual ao número de classes do problema, e softmax na última camada sem função de ativação entre aquela e a penúltima, aliada da função *cross-entropy* como função custo e otimizador Adam, é a melhor opção. Revelou-se também que arquiteturas Deep Feedforward, na terminologia aqui utilizada, produziu os melhores resultados. Descobriu-se ainda que redes com apenas uma camada intermediária funcionam melhor com PCA; ao passo que o LDA demanda mais uma camada interna para superar não-linearidades. Conclui-se também que, neste caso, a classificação binária é de maior facilidade para RNAs do que a multivariada.

A metodologia utilizada para construir as topologias pelo uso de parametrizações mostrou-se uma ferramenta poderosa para evidenciar padrões na relação de causalidade entre estrutura neural e desempenho. Assim como o treinamento de 500 modelos para cada topologia destacou a natureza probabilística da resposta daquela quando tendo seus parâmetros inicializados aleatoriamente. De modo que a convergência da razão de cada ordem de magnitude de erro no treinamento é uma ferramenta muito útil para identificar over e underfitting. Em verdade, tão útil quanto as PMFs de acurácia, que, quando utilizadas em simbiose com a PMF de erro no treinamento, indicam pistas para o ajuste de hiperparâmetros da RNA. Destaca-se ainda o papel do cálculo do score da topologia para a comparação justa e coerente com outras arquiteturas, de modo que o melhor arranjo de hiperparâmetros possa ser escolhido para trazer os resultados mais satisfatórios, estáveis e coerentes. Sendo esse score mínimo, ou seja, o score de corte definido pelo usuário para atender às suas necessidades de acurácia mínima.

Mostrou-se, conclusivamente, que a escolha de uma arquitetura é invariante com a fração de amostras que são deixadas para o treinamento, ao passo que as características da topologia são variantes com respeito ao número de neurônios de *output*. Assim, é oportuno que outra pesquisa seja feita com mais tipos de danos para que mais combinações de classificações sejam possíveis, em vez de apenas duas formas. O que há de esclarecer e permitir a modelagem da hipótese da variância da topologia em função do número de *outputs*.

Foi demonstrado que a razão de amostras para treinamento traz melhores resultados quando for igual a 75%, embora seja necessário um estudo futuro para que se encontre uma fração ótima, menor que aquela e maior que 63%, que garanta ainda bons resultados. De forma análoga ao PCA, com uma razão menor, há uma perda de desempenho, mas uma economia na demanda por amostras para treinamento. Logo, define-se o quociente de classificações corretas

pelo número de amostras utilizadas para o treinamento, ou seja, divide-se o bônus pelo ônus, e, analogamente ao LDA, esse quociente deve ser maximizado. Por meio deste trabalho, observa-se que tal tarefa é possível de ser solucionada e encontra-se a motivação para isso.

É visível, por fim, que mesmo com as complexidades dos materiais compósitos, como mecanismo de falha de delaminação e acoplamentos de deslocamento, foi possível construir uma ferramenta robusta o suficiente para prever em todos os casos o estado de integridade estrutural do componente. Sendo esta previsão feita com ensaios não destrutivos e permissíveis de se realizar *in situ*, de modo que há plena viabilidade de aplicação de tais métodos na indústria, para o avanço tecnológico de peças e de estruturas mais otimizadas e constantemente monitoradas.

REFERÊNCIAS

- AGGARWAL, C. C. Neural networks and deep learning. **Springer**, Springer, v. 10, p. 978–3, 2018. Citado na página 17.
- ARUN, K. Understanding curse of dimensionality. Great Learning, 2020. Disponível em: <<https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/>>. Citado na página 29.
- AVITABILE, Peter. **Modal testing: a practitioner’s guide**. [S.l.]: John Wiley & Sons, 2017. Citado na página 16.
- BISHOP, Christopher M et al. **Neural networks for pattern recognition**. [S.l.]: Oxford university press, 1995. Citado na página 29.
- BOCK, Sebastian; GOPPOLD, Josef; WEIß, Martin Georg. An improvement of the convergence proof of the adam-optimizer. [S.l.], Elsevier, 2008. Citado na página 25.
- BROWNLEE, J. Gentle introduction to the adam optimization algorithm for deep learning. Machine Learning Mastery, 2017. Disponível em: <<https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>>. Citado na página 25.
- BROWNLEE, J. **What is the Difference Between Test and Validation Datasets?** 2017. <https://machinelearningmastery.com/difference-test-validation-datasets/>. Accessed 5 March 2021. Citado 2 vezes nas páginas 27 e 49.
- BROWNLEE, J. **A Gentle Introduction to k-fold Cross-Validation**. 2018. <https://machinelearningmastery.com/k-fold-cross-validation/>. Acessado 5 julho 2021. Citado na página 26.
- BROWNLEE, J. A gentle introduction to cross-entropy for machine learning. Machine Learning Mastery, 2019. Disponível em: <<https://machinelearningmastery.com/cross-entropy-for-machine-learning/>>. Citado 3 vezes nas páginas 22, 23 e 45.
- BROWNLEE, J. Code adam optimization algorithm from scratch. Machine Learning Mastery, 2021. Disponível em: <<https://machinelearningmastery.com/adam-optimization-from-scratch/>>. Citado na página 24.
- BUDUMA, Nikhil. **Fundamentals of Deep Learning**. [S.l.]: O’Reilly Media, Inc, 2017. Citado na página 25.
- DORING, Matthias. Dimensionality reduction for visualization and prediction. Data Science Blog, 2018. Disponível em: <<https://www.datascienceblog.net/post/machine-learning/dimensionality-reduction/>>. Citado na página 34.
- DORING, Matthias. Linear, quadratic, and regularized discriminant analysis. Data Science Blog, 2018. Disponível em: <<https://www.datascienceblog.net/post/machine-learning/linear-discriminant-analysis/>>. Citado 3 vezes nas páginas 37, 38 e 39.
- GOMES, Guilherme Ferreira et al. Optimized damage identification in cfrp plates by reduced mode shapes and ga-ann methods. **Engineering Structures**, Elsevier, v. 181, p. 111–123, 2019. Citado na página 13.

- GRENIER, Kevin. What is a frequency response function (frf)? **Simcenter**, Siemens, 2020. Citado na página 16.
- HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2. ed. [S.l.]: Springer, 2009. Citado 2 vezes nas páginas 32 e 40.
- HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007. Citado 2 vezes nas páginas 19 e 53.
- JAADI, Zakaria. A step-by-step explanation of principal component analysis (pca). Built in, 2021. Disponível em: <<https://builtin.com/data-science/step-step-explanation-principal-component-analysis>>. Citado 2 vezes nas páginas 34 e 36.
- JOLLIFFE, Ian T.; CADIMA, Jorge. Principal component analysis: a review and recent developments. Royal Society, 2016. Disponível em: <<https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202>>. Citado 3 vezes nas páginas 34, 35 e 36.
- JUNIOR, Ronny Francis Ribeiro; ALMEIDA, Fabrício Alves de; GOMES, Guilherme Ferreira. Fault classification in three-phase motors based on vibration signal analysis and artificial neural networks. **Neural Computing and Applications**, Springer, v. 32, n. 18, p. 15171–15189, 2020. Citado na página 13.
- KAMPERIS, Stasis. Principal component analysis limitations and how to overcome them. A Blog on Science, 2021. Disponível em: <<https://ekamperi.github.io/mathematics/2021/02/23/pca-limitations.html>>. Citado 3 vezes nas páginas 34, 36 e 37.
- KARPATHY, A. **The vanishing gradient problem**. 2017. <https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b>. Accessed 23 November 2018. Citado na página 46.
- KARSH, PK; MUKHOPADHYAY, T; DEY, S. Spatial vulnerability analysis for the first ply failure strength of composite laminates including effect of delamination. **Composite Structures**, Elsevier, v. 184, p. 554–567, 2018. Citado na página 12.
- KELLY, S. Graham. **Mechanical Vibrations: Theory and applications**, si. Stamford: Cengage Learning, 2012. Citado na página 15.
- KOECH, K. E. Cross-entropy loss function. Towards Data Science, 2020. Disponível em: <<https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>>. Citado na página 23.
- KRZYK, Kamil. Coding deep learning for beginners — linear regression (part 2): Cost function. **Towards Data Science**, 2018. Disponível em: <<https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f>>. Citado na página 22.
- LEAL, R. P. **Introdução à Teoria dos Laminados**: Resumo. [S.l.: s.n.], 2005. Citado na página 14.
- LIU, Han; WASSERMAN, Larry. Linear classification. In: **Lectures of the Course of Statistical Machine Learning of the Carnegie Mellon University**. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 38, 39 e 40.

LOONEY, C G. Advances in feedforward neural networks: demystifying knowledge acquiring black boxes. In: **IEEE Transactions on Knowledge and Data Engineering**. IEEE: [s.n.], 1996. Citado na página 45.

MAKLIN, Cory. Linear discriminant analysis in python. Towards Data Science, 2019. Disponível em: <<https://towardsdatascience.com/linear-discriminant-analysis-in-python-76b8b17817c2>>. Citado na página 37.

MARTÍNEZ, Aleix M.; KAK, Avinash C. Pca versus lda. In: **IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE**. [S.l.: s.n.], 2001. Citado na página 53.

MCMANUS, Brian. The plane that will change travel forever. **Real Engineering Channel**, 2021. Disponível em: <<https://www.youtube.com/watch?v=59A8-rKRr-0&t=1467s>>. Acesso em: 1 ago. 2021. Citado na página 12.

NG, A. **Principle Component Analysis Algorithm**. 2018. <https://www.coursera.org/lecture/machine-learning/principal-component-analysis-algorithm-ZYIPa>. Accessed 10 January 2019. Citado na página 46.

PEREIRA, João Luiz Junho et al. Lichtenberg optimization algorithm applied to crack tip identification in thin plate-like structures. **Engineering Computations**, Emerald Publishing Limited, 2020. Citado na página 13.

PEREIRA, M S; BEZERRA, E M. Utilization of neural networks in prediction of aeronautical compounds behaviour under shearing (in portuguese). In: **Meeting of scientific initiation and post-graduate program of ITA (in Portuguese)**. São José dos Campos, SP, Brazil: [s.n.], 2007. Citado na página 45.

PRABHAKARAN, S. Mahalanobis distance – understanding the math with examples (python). Machine Learning Plus, 2019. Disponível em: <<https://www.machinelearningplus.com/statistics/mahalanobis-distance/>>. Citado 2 vezes nas páginas 32 e 33.

PUTTEGOWDA, Madhu; RANGAPPA, Sanjay; JAWAID, Mohammad. Sustainable composites for aerospace applications. **Woodhead Publishing Series in Composites Science and Engineering**, Elsevier, 2018. Citado na página 14.

RASCHKA, Sebastian. Linear discriminant analysis - bit by bit. Sebastian Raschka Blog, 2014. Disponível em: <https://sebastianraschka.com/Articles/2014_python_lda.html>. Citado 4 vezes nas páginas 38, 39, 40 e 53.

REIS, P. A. **Artificial neural networks applied to structural damage identification using dynamic response and signal processing**. Dissertação (Mestrado) — Mechanical Engineering Graduate Program, Santa Catarina State University, Joinville, Brasil, 2020. Citado 4 vezes nas páginas 13, 14, 17 e 41.

SHARMA, S. Activation functions in neural networks. **Towards Data Science**, 2017. Disponível em: <<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>>. Citado na página 21.

SHLENS, Jonathon. A tutorial on principal component analysis. Google Research, 2014. Disponível em: <<https://arxiv.org/pdf/1404.1100.pdf>>. Citado na página 36.

- SILVA, I; SPATTI, D H; FLAUZINO, R A. **Artificial neural networks for engineering and applied sciences (in Portuguese)**. São Paulo: Artliber, 2010. Citado na página 45.
- SILVA, I. d.; SPATTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas**. 5. ed. São Paulo: Artliber, 2010. v. 23. Citado na página 18.
- SOAMI, Pawan. Introducing the composite materials module. COMSOL, 2018. Disponível em: <<https://www.comsol.com/blogs/introducing-the-composite-materials-module/>>. Citado na página 15.
- SRIDHARAN, S. Delamination behaviour of composites. [S.l.], Elsevier, 2008. Citado na página 14.
- STEORTS, Rebecca C. Classification methods ii: Linear and quadratic discriminant analysis. In: **Lectures of the Course of Data Mining and Machine Learning of the Duke University**. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 39 e 55.
- VIDHYA, T A. **A Practical Guide to Principle Component Analysis (PCA) in R and Python**. 2016. <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/>. Accessed 10 December 2018. Citado na página 46.
- VOLTZ, L. R. **Fault diagnosis in composite structures using artificial neural network and principal component analysis**. Dissertação (Mestrado) — Mechanical Engineering Graduate Program, Santa Catarina State University, Joinville, Brasil, 2019. Citado 3 vezes nas páginas 24, 42 e 49.
- XIAOZHOU, Yang. Linear discriminant analysis, explained. Towards Data Science, 2020. Disponível em: <<https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b>>. Citado na página 40.

ANEXO A – TOPOLOGIAS SIMULADAS

Anexo com as tabelas que documentam todas as topologias simuladas neste trabalho. Para cada coluna, tem-se a opções de hiperparâmetro, sendo que todas as combinações possíveis foram simuladas.

Tabela 6 – Topologias com uma camada interna e 2 neurônios de output. PCA somente foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária	Função de ativação	Número de neurônios de output	Função de ativação
22	ELU	9	Identidade	2	Softmax
26		10			
		11			
		12			
30	LeakyReLU	13			
		:			
31	Sigmóide	56			
		57			
		58			
		59			
		60			

Fonte: Produção do autor.

Tabela 7 – Topologias com uma camada interna e 4 neurônios de output. PCA somente foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária	Função de ativação	Número de neurônios de output	Função de ativação
22	ELU	9	Identidade	4	Softmax
25	LeakyReLU	10			
		:			
30	Sigmóide	59			
		60			

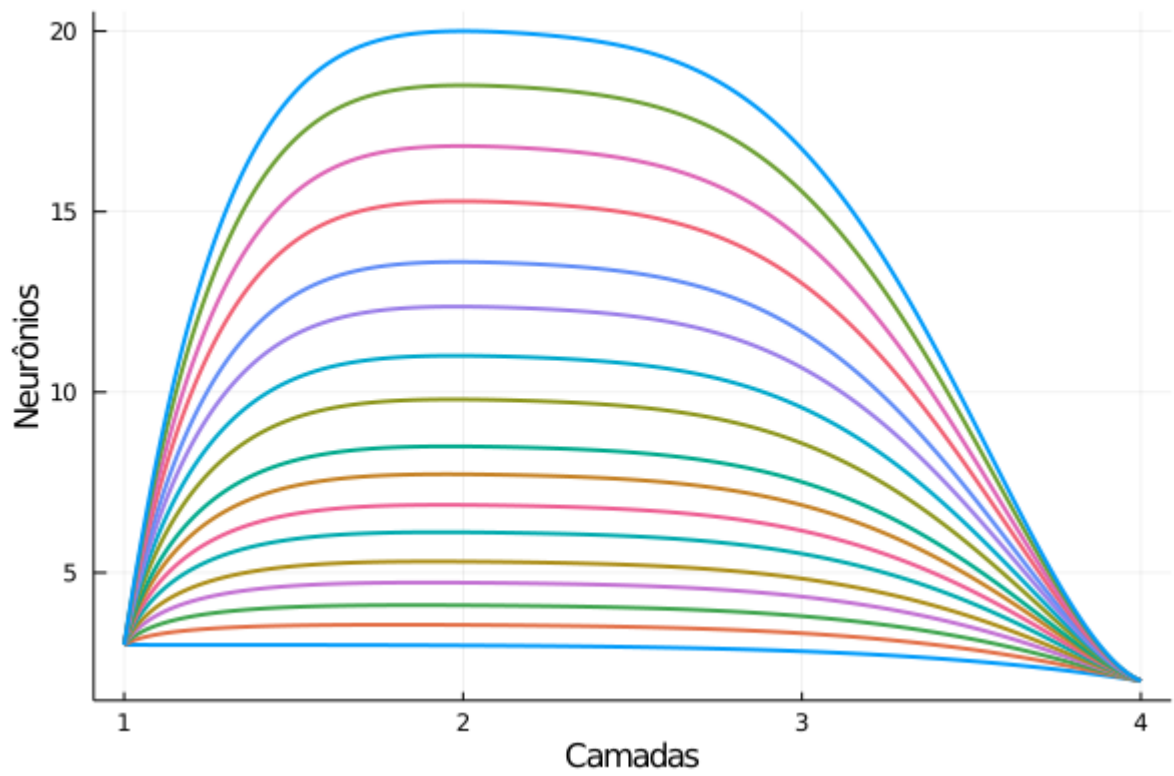
Fonte: Produção do autor.

Tabela 8 – Topologias com duas camadas internas, 3 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação	
3	ELU	3	Sigmóide	3	Identidade	2	Softmax	
		3		2				
		4		3				
		4		4				
		5		4				
		5		5				
		6		6				
	7	6						
	Leaky-ReLU	8		7				
		8		8				
		10		9				
		11		10				
		12		11				
		14		12				
		Sigmóide		15				13
				17				14
				18				16
20			17					

Fonte: Produção do autor.

Figura 19 – Parametrizações das topologias com duas camadas interna, 3 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.



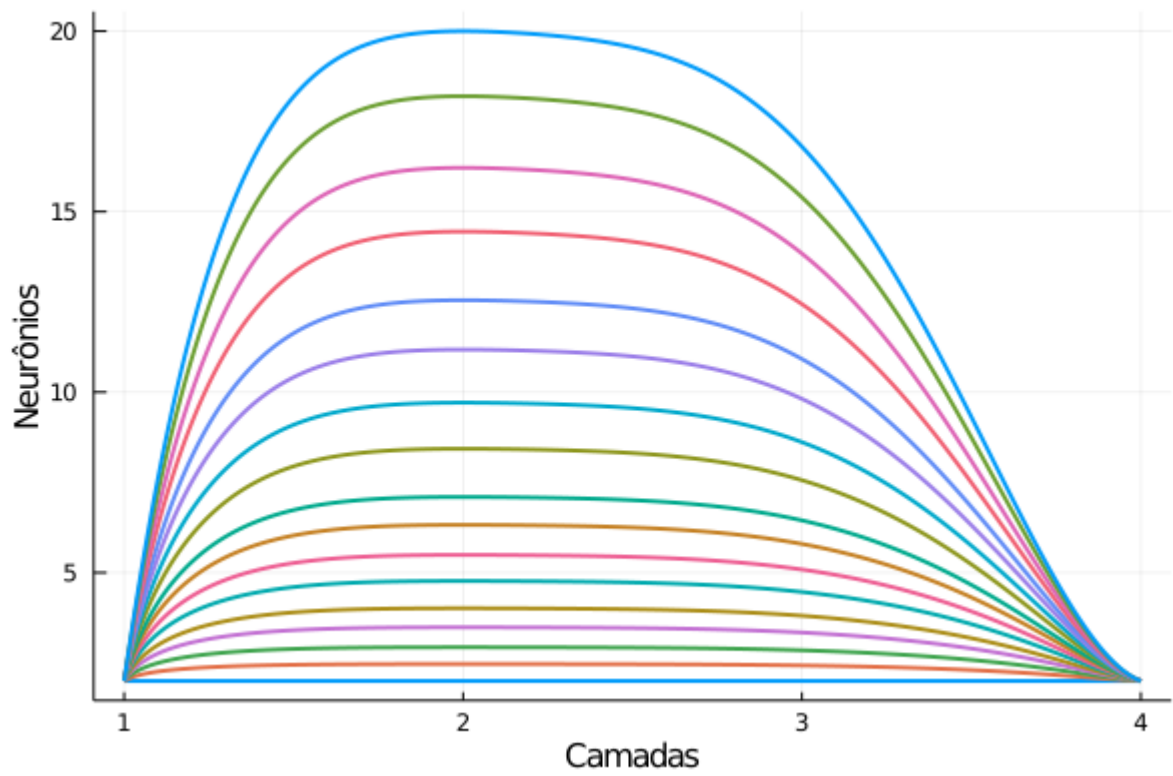
Fonte: Produção do autor.

Tabela 9 – Topologias com duas camadas internas, 2 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação	
2	ELU	2	Sigmóide	2	Identidade	2	Softmax	
		3		2				
		3		3				
		4		3				
		4		4				
		5		4				
	5	5						
	Leaky-ReLU	6		6				
		7		6				
		8		8				
		10		9				
		11		10				
		13		11				
		14		12				
		Sigmóide		16				14
				17				14
				18				15
20			17					

Fonte: Produção do autor.

Figura 20 – Parametrizações das topologias com duas camadas internas, 2 neurônios de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.



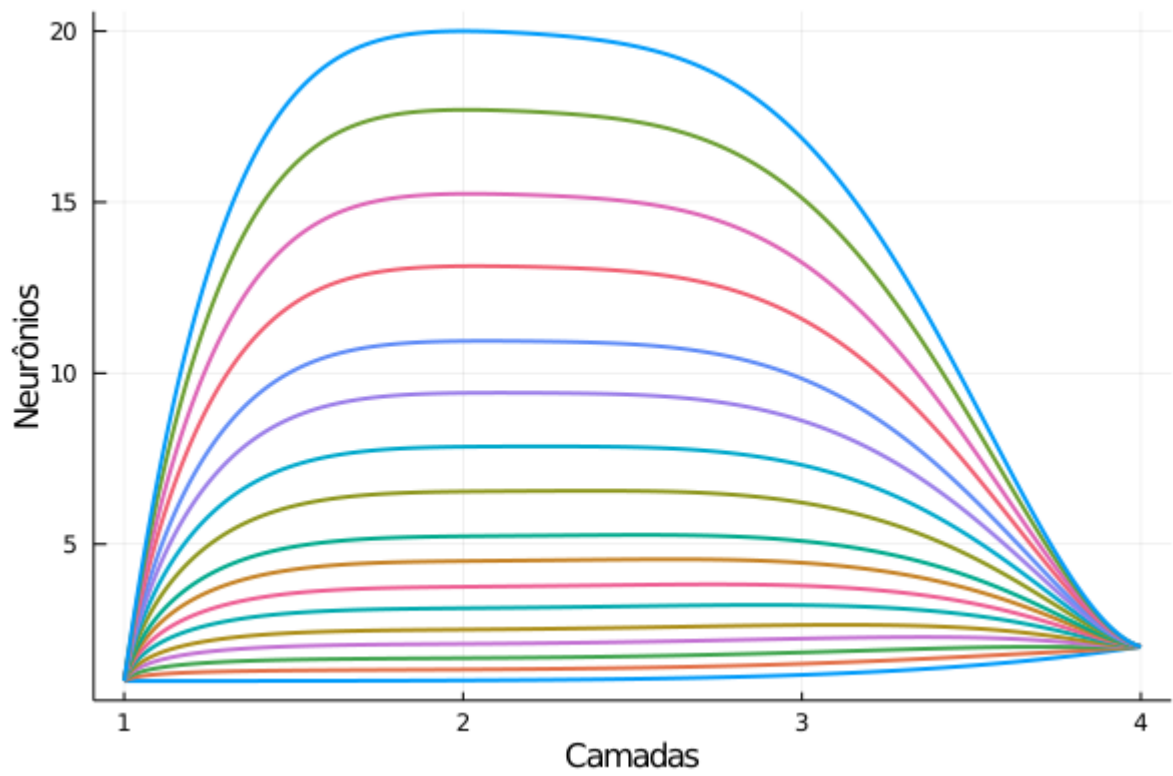
Fonte: Produção do autor.

Tabela 10 – Topologias com duas camadas internas, um neurônio de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação					
1	ELU	1	Sigmóide	1	Identidade	2	Softmax					
		1		2								
		2		2								
		2		3								
		3		3								
		4		3								
		4		4								
		5		4								
		5		5								
		6		5								
	7	6										
	8	7										
	9	9										
	11	10										
	13	12										
	15	13										
	18	15										
	20	17										
		Sigmóide										

Fonte: Produção do autor.

Figura 21 – Parametrizações das topologias com duas camadas internas, um neurônio de input e 2 neurônios de output. PCA com LDA foi usado para compressão dos dados.



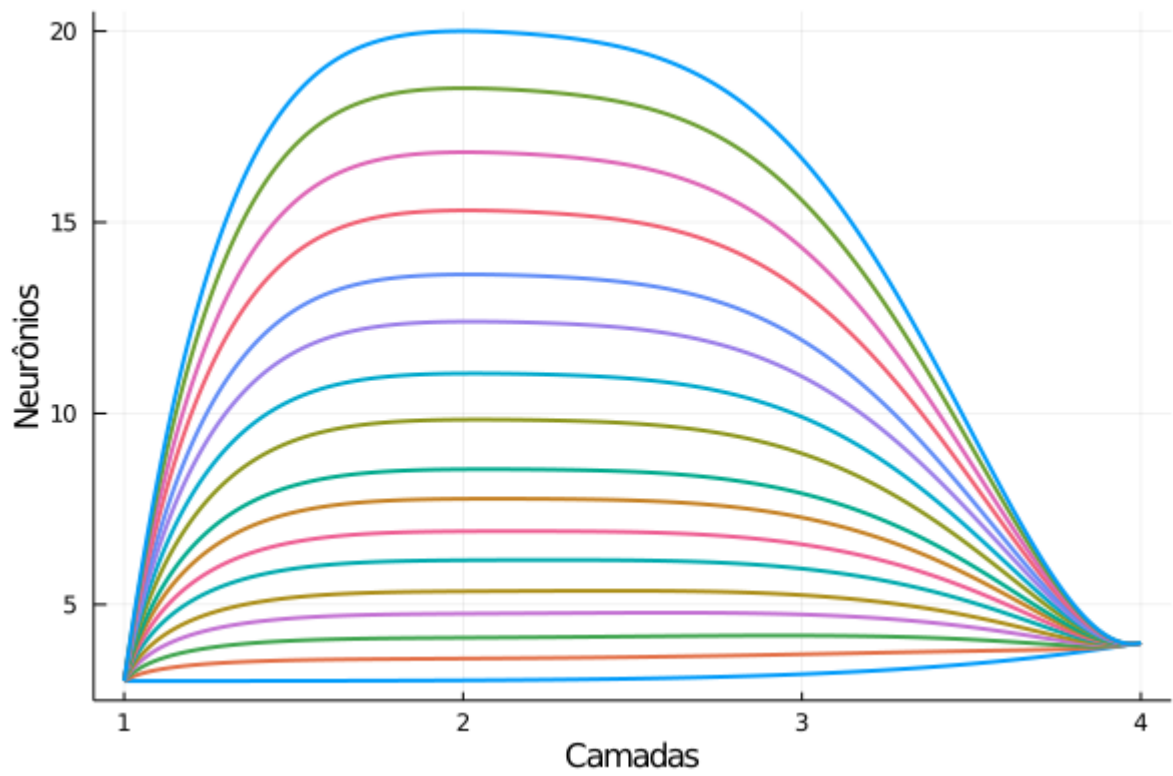
Fonte: Produção do autor.

Tabela 11 – Topologias com duas camadas internas, 3 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação
3	ELU	3	Sigmóide	3	Identidade	4	Softmax
		3		4			
		4		4			
		5		4			
		5		5			
		6		6			
		4		4			
		7		7			
		8		7			
		9		8			
	Sigmóide	10		9			
		11		10			
		12		11			
		14		12			
		15		13			
		17		14			
		19		16			
		20		17			

Fonte: Produção do autor.

Figura 22 – Parametrizações das topologias com duas camadas internas, 3 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.



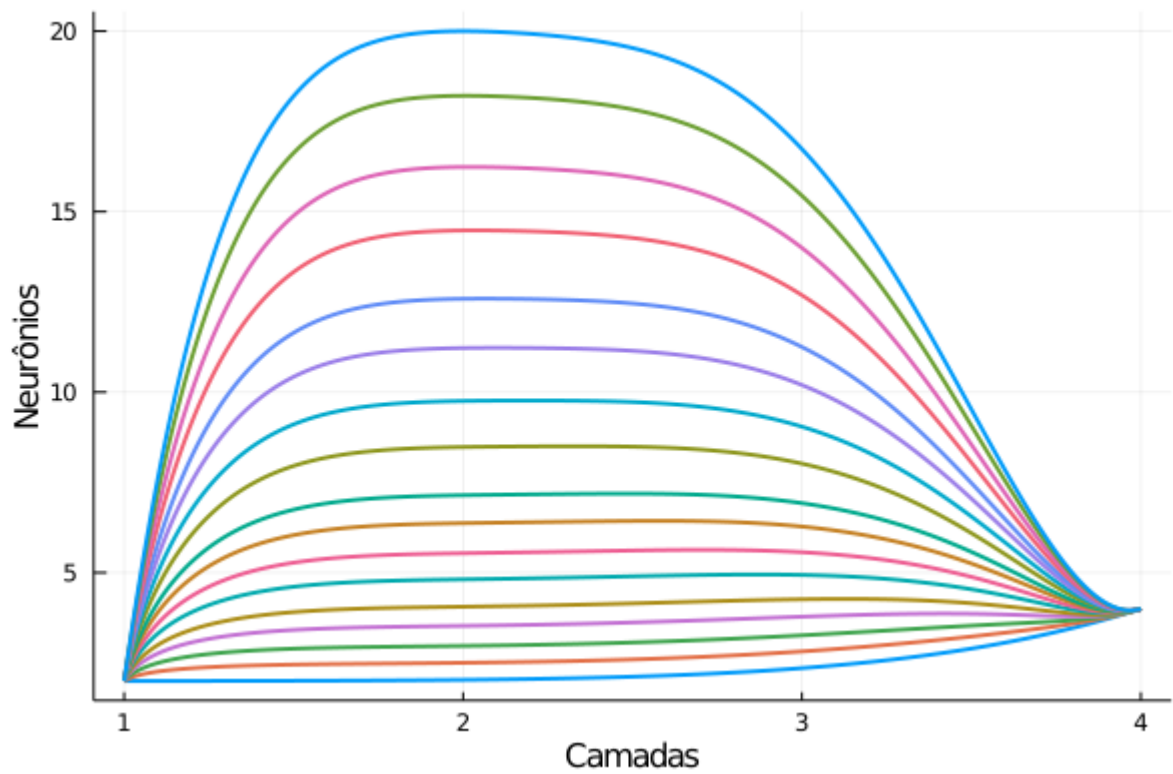
Fonte: Produção do autor.

Tabela 12 – Topologias com duas camadas internas, 2 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação
2	ELU	2	Sigmóide	2	Identidade	4	Softmax
		2		3			
		3		2			
		3		3			
		4		3			
		4		4			
		5		5			
	6	6					
	Leaky-ReLU	7		5			
	7	7					
	8	8					
	10	9					
	11	10					
	13	11					
	Sigmóide	14		13			
	16	14					
	18	15					
20	17						

Fonte: Produção do autor.

Figura 23 – Parametrizações das topologias com duas camadas internas, 2 neurônios de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.



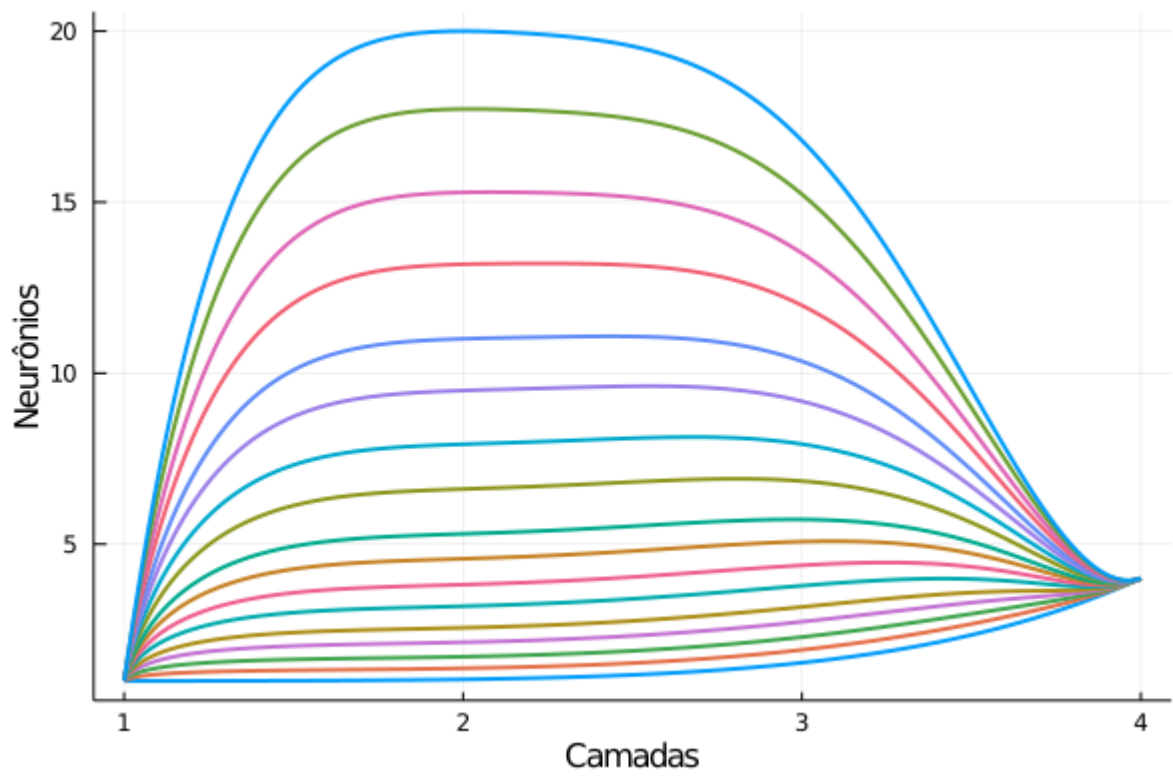
Fonte: Produção do autor.

Tabela 13 – Topologias com duas camadas internas, um neurônio de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.

Número de neurônios de input	Função de ativação	Número de neurônios da camada intermediária I	Função de ativação	Número de neurônios da camada intermediária II	Função de ativação	Número de neurônios de output	Função de ativação
1	ELU	1	Sigmóide	2	Identidade	4	Softmax
		2		2			
		2		3			
		3		3			
		3		4			
		4		3			
		4		4			
		5		4			
		5		5			
	5	6					
	7	7					
	8	8					
	9	9					
	11	10					
	13	12					
	15	14					
	18	15					
	20	17					

Fonte: Produção do autor.

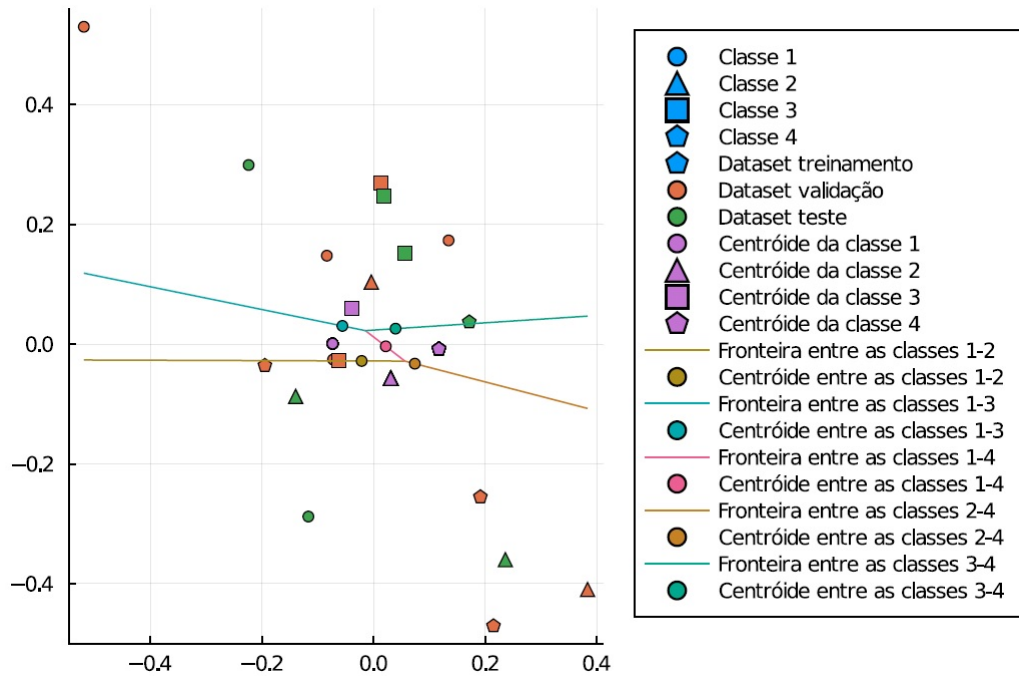
Figura 24 – Parametrizações das topologias com duas camadas internas, um neurônio de input e 4 neurônios de output. PCA com LDA foi usado para compressão dos dados.



Fonte: Produção do autor.

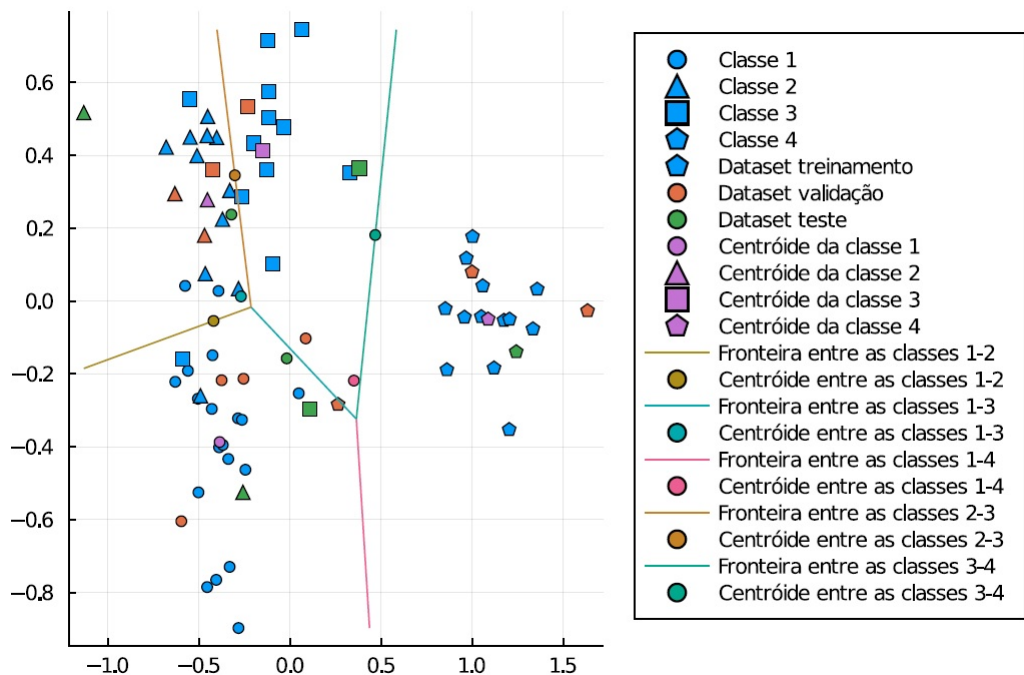
ANEXO B – FRONTEIRAS DE DECISÃO

Figura 25 – Fronteiras de decisão do conjunto de dados utilizando a divisão de 75% para treinamento e utilizando apenas o LDA e dois discriminantes lineares.



Fonte: Produção do autor.

Figura 26 – Fronteiras de decisão do conjunto de dados utilizando a divisão de 75% para treinamento e utilizando PCA seguido de LDA e dois discriminantes lineares.



Fonte: Produção do autor.