Relatório Anual - Iniciação Científica

Redes neurais distribuídas para classificação de arritmias

Bolsista: Lucca Gamballi
Orientador: Magno Teófilo Madeira da Silva
Coorientador: Renato Candido

Processo Nº: 139071/2021-0
Período de vigência: 01/09/2021 a 31/08/2022

Universidade de São Paulo (USP)
Escola Politécnica
Departamento de Engenharia de Sistemas Eletrônicos (PSI)
Av. Prof. Luciano Gualberto, 158, trav. 3
São Paulo - SP
05508-010

São Paulo

2022

Sumário

1 Resumo do Projeto de Pesquisa

Neste projeto, propõe-se o uso de redes neurais distribuídas para a classificação de arritmias cardíacas. Normalmente, o diagnóstico dessas doenças demanda muito tempo do cardiologista, além disso as características que compõem o eletrocardiograma (ECG) dificultam o trabalho do especialista. Existem diversos métodos computadorizados para identificação de arritmias a partir do ECG, contudo as taxas de erro são elevadas. Devido ao crescimento da área de aprendizado de máquina, a pesquisa sobre a automatização do diagnóstico de arritmias cardíacas ganhou relevância na comunidade científica. Em aplicações médicas, é comum que os dados não possam ser centralizados devido a limitações computacionais, então o treinamento utilizando redes neurais distribuídas tem crescido. A ideia por trás do uso dessas redes neurais é dividir os dados usados no treinamento entre várias máquinas que se comunicam utilizando uma topologia pré-estabelecida, disso surge a proposta deste projeto. Serão utilizadas, em particular, redes perceptron multicamada (*multilayer perceptron* - MLP).

1.1 Justificativa

O diagnóstico de doenças cardiovasculares concentra grande parte das pesquisas envolvendo a análise do eletrocardiograma. Muitas doenças cardíacas, incluindo infarto do miocárdio, bloqueio auriculoventricular, taquicardia ventricular, fibrilação atrial, entre outras, podem ser diagnosticadas a partir da análise de sinais de ECG [1]. Para se obter o diagnóstico automático dessas doenças, técnicas de aprendizado de máquina têm sido empregadas. A utilização dessas técnicas na análise do eletrocardiograma não é recente, porém devido às altas taxas de erros dos métodos computadorizados [1, 2, 3], os diagnósticos ainda são feitos por especialistas na prática. Essa alta taxa de erro em conjunto com o avanço da área de aprendizado de máquina fez com que a pesquisa em diagnóstico automático de doenças cardíacas conquistasse grande interrese.

A principal abordagem em aprendizado de máquina é o treinamento supervisionado, em que um modelo é ajustado para mapear exemplos de entrada à rótulos pertencentes a um banco de dados de treinamento. Em geral, o treinamento é realizado em uma única máquina, o que pode não ser adequado quando se trabalha com uma grande quantidade de dados ou quando a privacidade é importante [4, 5]. Atualmente, a aplicação de técnicas de aprendizado de máquina na medicina, por exemplo, é dificultada pela disponibilidade limitada de conjuntos de dados para treinamento e teste dos algoritmos. Isso ocorre devido à ausência de registros médicos eletrônicos padronizados e devido aos requisitos legais e éticos estritos para proteger a privacidade do paciente. Para garantir a privacidade do paciente e, ao mesmo tempo, promover pesquisas científicas com grandes conjuntos de dados, a busca por soluções que atendem simultaneamente as demandas de proteção e utilização dos dados tem crescido [4, 5, 6, 7]. Nesse contexto é inserido este projeto. A ideia é dividir os dados de treinamento entre diversas máquinas capazes de se comunicar a fim de superar as dificuldades de privacidade e disponibilidade limitante de dados.

1.2 O sistema cardiovascular e o eletrocardiograma

O coração é responsável pela entrega de sangue para os tecidos, fornecendo nutrientes essenciais às células e removendo os dejetos metabólicos. Para isso, ele funciona como uma bomba sanguínea que distribui os nutrientes para o corpo e retira os produtos de excreção das células do organismo. De acordo com a Organização Mundial de Saúde, as doenças cardiovasculares são a principal causa de morte no mundo [8]. O diagnóstico dessas doenças depende de uma análise precisa do sinal de ECG, que registra a ativadade elétrica do coração. O sinal de ECG é obtido com o auxílio de eletrodos, que são colocados nos membros ou no toráx do paciente. Esses eletrodos são responsáveis por captar o potencial elétrico produzido pelos músculos do coração. Um batimento cardíaco usual possui a forma de onda apresentada na Figura 1.

As condições fisícas do paciente determinam as características do sinal de ECG. Dessa maneira, pacientes diferentes com a mesma doença podem apresentar sinais de ECG com características distintas, ao passo que doenças diferentes podem resultar em sinais de ECG semelhantes [8]. Para detectar alterações no batimento cardíaco, o sinal de ECG deve ser analisado, o que demanda muito tempo do cardiologista e se torna inviável

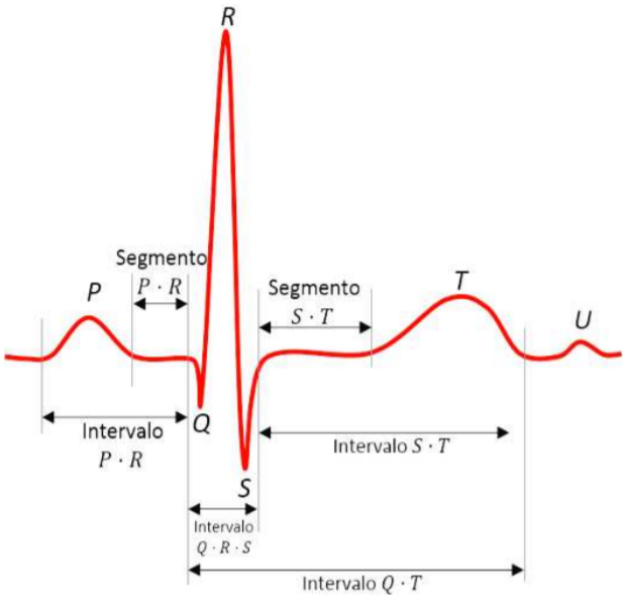


Figura 1: Batimento padrão de um ECG. Fonte: [9]

em casos de urgência. Para suprir essa dificuldade, métodos assistidos por computador têm sido amplamente estudados, como no caso da utilização de redes neurais para classificação de arritmias.

1.3 Atividades Realizadas

Como forma de preparação para a classificação de arritmias cardíacas de forma distribuída, primeiramente foram estudados conceitos de processamento distribuído de sinais e fundamentos de filtragem adaptativa [10, 11, 12]. Em seguida, analisou-se qual a melhor forma de estender as técnicas estudadas para o âmbito das redes neurais, em particular das redes MLP. Com isso, foi possível implementar algoritmos de aprendizado de máquina para resolução de problemas, adotando uma abordagem distribuída. A príncipio, foi resolvido um problema de classificação de imagens, sendo que ele buscava classificar figuras geométricas, e então foi proposta uma solução distribuída para a classificação de arritmias usando redes MLP.

A Seção 2 trata do treinamento distribuído. Nela, encontram-se sua motivação, definição e implementação considerando redes neurais. Enquanto que, a Seção 3 versa sobre as métricas de desempenho adotadas neste trabalho. Por sua vez, a Seção 4 apresenta o problema de classificação de figuras geométricas assim como propostas de solução. A Seção 5 expõe o problema de classificação de arritmias cardíacas e suas características, além de propor uma solução distribuída para esse problema.

Por fim, as conclusões do trabalho estão mostradas na Seção 6. Afim de apresentar um relatório de atividades conmpleto, a descrição do Perceptron de Rosenblatt e da rede MLP estão apresentadas nos Apêndices A e B.

Resultados dessa pesquisa foram aceitos para apresentação no Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT) a ser realizado em setembro de 2022. Os artigos

- Lucca Gamballi, Daniel Gilio Tiglea, Renato Candido, Mangno T. M. Silva: Treinamento distribuído de redes MLP para classificação de figuras geométricas, páginas 1-2.
- Lucca Gamballi, Daniel G. Tiglea, Renato Candido, Mangno T. M. Silva: Redes MLP distribuídas para classificação arritmias cardíacas, páginas 1-5.

encontram-se nos Apêndices C e D deste relatório.

2 Treinamento Distribuído de redes MLP

No treinamento distribuído de redes neurais, cada máquina treina um modelo local utilizando apenas uma parte dos dados e troca informações acerca de seu modelo com máquinas vizinhas, sem compartilhar os dados de treinamento. Para comunicação entre as máquinas considera-se uma topologia pré-definida. Assim, pode-se obter um único modelo global utilizando todos os dados de treinamento sem que uma única máquina tenha acesso a todos eles [5].

A computação paralela é um dos métodos mais populares para resolver problemas que lidam com privacidade e volume muito grande de dados. Os algoritmos de computação paralela existentes são, de forma geral, projetados para grafos com topologia de mestre-escravo [13]. Nesse tipo de grafo um mestre está conectado a vários escravos, conforme o modelo representado na Figura 2a. O nó central, o mestre, recebe informações de todo os outros nós, os escravos. Após realizar algumas operações ele transmite os resultados aos escravos. No contexto das redes neurais cada nó representa uma rede neural e as informações transmitidas ao mestre são os pesos das redes, assim como a informação transmitida pelo mestre é a atualização dos parâmetros das redes.

Essa topologia apresenta alguns problemas. É possível que o mestre fique congestionado e sobrecarregado, pois cada rede neural precisa se comunicar com o mestre ao término de uma iteração de treinamento [14]. Esse problema pode ser grave quando a largura da banda de comunicação é baixa ou a latência é muito alta. Para contornar essa situação é proposta uma topologia descentralizada, onde não há agente central e cada agente se comunica apenas com seus vizinhos. Um esquema dessa topologia está representado na Figura 2b. Nessa topologia é considerado um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, em que $\mathcal{V} = \{1,2,3,\ldots,\mathcal{V}\}$ representa o conjunto dos $\mathcal{V}$ nós do grafo, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ representa o conjunto de arestas do grafo e $\mathcal{H} \in \mathbb{M}_{\mathcal{V}}(\mathbb{R})$ representa a matriz de adjacência do grafo. O conjunto de nós com o qual o nó i consegue se comunicar (incluindo o próprio nó i) é chamado de vizinhança do nó i e denotado por $\mathcal{N}_i$, tendo cardinalidade $|\mathcal{N}_i|$. Quando o nó i consegue se comunicar com o nó j temos os elementos da matriz de adjacência $h_{ij} = h_{ji} = 1$ e a aresta (i, j) pertencente ao conjunto $\mathcal{E}$. Caso o nó i não consiga se comunicar com o nó j temos os elementos da matriz de adjacência $h_{ij} = h_{ji} = 0$ e a aresta (i, j) fora do conjunto $\mathcal{E}$ [15]. Além disso, podem ser atribuídos pesos às conexões entre os nós, representando a importância da conexão entre eles. É conveniente definir a matriz de conexão entre os nós como

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1\mathcal{V}} \\ c_{21} & c_{22} & c_{23} & \cdots & c_{2\mathcal{V}} \\ c_{31} & c_{32} & c_{33} & \cdots & c_{3\mathcal{V}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{\mathcal{V}1} & c_{\mathcal{V}2} & c_{\mathcal{V}3} & \cdots & c_{\mathcal{V}\mathcal{V}} \end{bmatrix}, \tag{1}$$

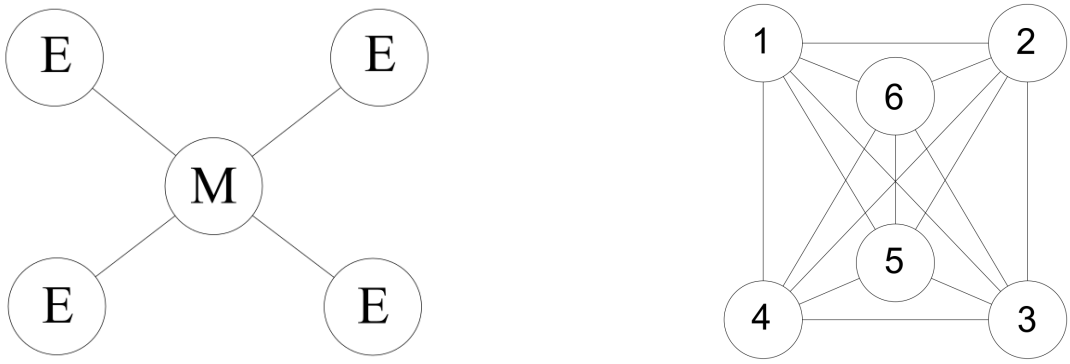
em que o elemento c_{ij} representa o peso da conexão entre o nó i e o nó j. Existem diversas formas de determinar os elentos da matriz $\mathbf{C}$. Dentre as mais utilizadas na literatura, podem-se citar a regra Uniforme [16], Laplaciana [17], Metropolis [4, 18], entre outras. Existem também algoritmos adaptativos para determinação de cada c_{ij}. Na literatura, os mais comuns são os algoritmos ACW (*adaptative combination weights*) [19, 20] e o algortimo dos mínimos quadrados (*least squares* - LS) [12].

Na estratégia Uniforme, cada nó calcula a média simples das estimativas locais de seus vizinhos, ou seja,

$$c_{ij} = \begin{cases} \frac{1}{|\mathcal{N}_i|}, & \text{se } j \in \mathcal{N}_i \\ 0, & \text{se } j \notin \mathcal{N}_i \end{cases}, \tag{2}$$

enquanto na regra Metropolis os pesos são definidos por

$$c_{ij} = \begin{cases} \frac{1}{\max(|\mathcal{N}_i|,|\mathcal{N}_j|)}, & \text{se } j \in \mathcal{N}_i \text{ e } i \neq j, \\ 0, & \text{se } j \notin \mathcal{N}_i, \\ 1 - \sum_{k \in \mathcal{N}_i} a_{ki}, & \text{se } i = j \end{cases}, \tag{3}$$



(a) Exemplo de topologia mestre-escravo.

(b) Exemplo de topologia descentralizada.

Figura 2: Tipos de topologias.

ao passo que na estratégia ACW os pesos são calculados por

$$c_{ij}(n) = \begin{cases} \frac{\hat{\sigma}_{ij}^{-2}(n+1)}{\sum_{k\in\mathcal{N}_i}\hat{\sigma}_{ik}^{-2}(n+1)}, & \text{se } j \in \mathcal{N}_i, \\ 0, & \text{caso contrário} \end{cases}, \tag{4}$$

em que $\hat{\sigma}_{ij}^2$ representa a estimativa de σ_{ij}^2 calculada no nó j e é atualizada com base em

$$\hat{\sigma}_{ij}^{-2}(n+1) = (1 - v_k)\hat{\sigma}_{ij}^{-2}(n) + v_k\|\boldsymbol{\theta}\|^2, \tag{5}$$

com $v_k > 0$ para todo $k \in \mathcal{V}$. Por motivos de estabilidade numérica do algoritmo, sugere-se subistituir $\hat{\sigma}_{ij}^{-2}(n)$ por $[\delta + \hat{\sigma}_{ij}^{-2}(n)]^{-1}$, em que δ é uma constante positiva pequena, 10^{-8} por exemplo. Por fim, no caso das redes neurais, $\boldsymbol{\theta}$ pode representar uma de três opções: a diferença entre a estimativa do nó i e o resultado esperado do nó j, a diferença entre as matrizes de pesos (vetorizadas) da camada de saída dos nós i e j e a diferença entre a concatenação das matrizes de pesos (vetorizadas) de todas as camadas dos nós i e j. Cabe observar que na literatura não há uma forma adaptativa para o treinamento dos pesos de conexões de forma distribuída. Essas propostas foram geradas neste trabalho. Na seção seguinte, detalha-se o treinamento distribuído de redes MLP.

2.1 Atualizando os pesos das redes distribuídas

Consideremos um problema de aprendizagem supervisionada com um conjunto de treinamento $\mathcal{D}$ de cardinalidade N. A ideia do treinamento distribuído de redes MLP consiste em dividir $\mathcal{D}$ em V subconjuntos disjuntos de cardinalidade N_i, denotados por $\mathcal{D}_i$, $i = 1,2,\ldots,V$, tal que $\sum_{i=1}^{V} N_i = N$. Os exemplos do subconjunto $\mathcal{D}_i$ são utilizados no treinamento da MLP da máquina i. As V máquinas consideradas se comunicam por meio de uma determinada topologia, como a da Figura 3. Nessa figura, cada máquina corresponde a um nó do grafo e as arestas indicam a existência de uma conexão entre determinados pares de máquinas. A *vizinhança* do nó i, denotada por $\mathcal{V}_i$, é o conjunto de nós com os quais ele consegue se comunicar diretamente (incluindo ele próprio).

O objetivo de cada nó i é treinar uma rede MLP utilizando o subconjunto $\mathcal{D}_i$ a fim de minimizar uma função custo, cuja forma é comum a todos os nós do grafo. Considerando um problema de classificação com C classes, é comum adotar como função custo a entropia cruzada categórica ponderada. Assim, a MLP da i-ésima máquina minimiza [21]

$$J_{\text{ECCP}_i} = -\sum_{n_i=1}^{N_i}\sum_{\ell=1}^{C} p_{i,\ell}\; d_{i,\ell}(n_i)\;\ln[y_{i,\ell}(n_i)], \tag{6}$$

em que $y_{i,\ell}(n_i)$ é a ℓ-ésima saída da rede para o n_i-ésimo dado de treinamento com rótulo $d_{i,\ell}(n_i)$, $\ell = 1,2,\ldots,C$ e $n_i = 1,2,\ldots,N_i$. Por fim, os pesos $p_{i,\ell}$ são utilizados em (6) para conferir uma ponderação maior

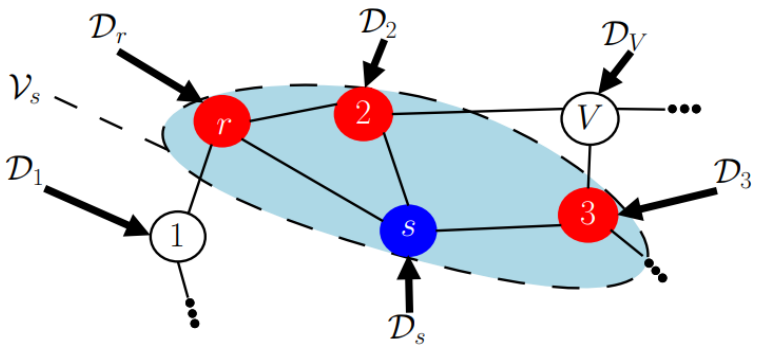


Figura 3: Exemplo de uma rede de difusão, em que cada nó $i \in \{1,2,\ldots,V\}$ treina uma rede MLP com base no subconjunto de treinamento $\mathcal{D}_i$. Neste exemplo, a vizinhança do nó s é $\mathcal{V}_s = \{2,3,r,s\}$.

a classes minoritárias [21]. Inspirando-se na heurística de [22], considerou-se neste trabalho o peso definido por

$$p_{i,\ell} = \left\lceil 10 \frac{N_i}{CB_{i,\ell}} \right\rceil, \tag{7}$$

em que $B_{i,\ell}$ é o número de amostras da classe ℓ em $\mathcal{D}_i$.

A matriz de pesos (incluindo os *biases*) da k-ésima camada da MLP do nó i, denotada por $\mathbf{W}_i^{[k]}$, pode ser atualizada com o algoritmo *backpropagation* [23, 24]. Para garantir a privacidade dos dados, cada nó i pode trocar informações com a sua vizinhança acerca de seu próprio modelo, mas não sobre seus dados de treinamento. Assim, a matriz $\mathbf{W}_i^{[k]}$ é compartilhada com os nós vizinhos após cada atualização. A ordem em que essas operações são realizadas dá origem a dois tipos de configuração: adaptação seguida da combinação (ATC – *adapt-then-combine*) e combinação seguida da adaptação (CTA – *combine-then-adapt*) [5]. Por simplicidade, neste trabalho será considerada apenas a configuração ATC, embora resultados semelhantes possam ser obtidos com a configuração CTA. Assim, a regra para atualização de $\mathbf{W}_i^{[k]}$ pode ser descrita por [5]

$$\mathbf{\Psi}_i^{[k]}(m+1) = \mathbf{W}_i^{[k]}(m) - \eta \frac{\partial J_{\mathrm{ECCP}_i}}{\partial \mathbf{W}_i^{[k]}(m)} \tag{8}$$

$$\mathbf{W}_i^{[k]}(m+1) = \sum_{j \in \mathcal{V}_i} c_{ij} \mathbf{\Psi}_i^{[k]}(m+1), \tag{9}$$

em que $\mathbf{\Psi}_i^{[k]}$ é uma estimativa intermediária de mesma dimensão que $\mathbf{W}_i^{[k]}$, η é o passo de adaptação, m é a iteração do algoritmo e $c_{ij} \geq 0$ são pesos de combinação, tal que $c_{ij} = 0$ se as máquinas i e j não estão diretamente conectadas e $\sum_{j=1}^{\mathcal{V}} c_{ij} = 1$ para todo i, j. Existem diversas regras possíveis para a escolha desses pesos. Vale ressaltar que cada estratégia para determinar os elementos da matriz $\mathbf{C}$ resulta em uma matriz diferente para o mesmo grafo. Isso tem grande influência na taxa de convergência do algoritmo. Pode-se demonstrar que, adotando-se a abordagem cooperativa e inicializando-se as redes MLP distribuídas da mesma maneira, o treinamento distribuído pode levar a um único modelo global [4, 5]. Nesse caso, o desempenho como um todo da redes MLP distribuídas é equivalente ao obtido com uma única rede MLP treinada com o conjunto $\mathcal{D}$, sem que nenhuma máquina tenha acesso a todos os elementos desse conjunto simultaneamente [5].

3 Métricas de desempenho

Neste projeto, foram consideradas como métricas de desempenho a Sensibilidade (Se), Precisão (P), F1-*Score* (F1) e Coeficiente de Correlação de Matthews (MCC). A sensibilidade é o número de verdadeiros positivos corretamente classificados sobre o número de amostras com rótulos positivos, ao passo que a precisão é a proporção entre os verdadeiros positivos e todos os positivos detectados. O F1-*Score* é a média harmônica entre a sensibilidade e a precisão. Essas três métricas estão limitadas entre 0 e 1, ou em porcentagem, 0%

a 100%. Quanto mais próximo de 1 significa uma melhor classificação. O MCC trata a classe verdadeira e classe predita como duas variáveis e calcula o coeficiente de correlação entre elas. Quanto maior a correlação, melhor a predição. O MCC está sempre entre -1 e 1, em que 1 indica correlação positiva perfeita e -1 correlação negativa perfeita. Caso o resultado seja 0, a classificação equivale ao lançamento de uma moeda.

A sensibilidade pode ser calculada por

$$Se = \frac{VP}{VP+FN}, \tag{10}$$

ao passo que a precisão pode ser calculada por

$$P = \frac{VP}{VP+FP}. \tag{11}$$

O F1-*score*, por sua vez, é calculado como

$$F1 = 2\frac{Se \times P}{Se + P}. \tag{12}$$

Por fim, o MCC é calculado por

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}}. \tag{13}$$

Nessas definições, VP representa o número de verdadeiros positivos, VN representa o número de verdadeiros negativos, FP representa o número de falsos positivos e FN representa o número de falsos negativos.

4 Classificação de figuras geométricas

Nesta seção, são mostrados os resultados de classificação de figuras geométricas. A seção está dividida em 3 subseções. Na Subseção 4.1, descreve-se o conjunto de dados. Na Subseção 4.2 descreve-se os cenários de testes e simulações. Por fim, na Subseção 4.3 são apresentados os resultados para a classificação de figuras geométricas.

4.1 Conjunto de dados

O conjunto de dados utilizado é constituído de imagens em preto e branco com dimensões 30×30 criadas aleatoriamente com auxílio da biblioteca PILLOW [25]. Cada imagem apresenta uma de três possíveis figuras geométricas: quadrados (Q), retângulos (R) ou circunferências (C). A espessura do traço pode ser fina ou grossa, determinada aleatoriamente.

O banco de dados apresenta 10000 imagens de cada figura para o treino, assim como 10000 imagens de cada figura para o teste. A Figura 4 mostra um exemplo de amostra para cada uma das figuras possíveis.

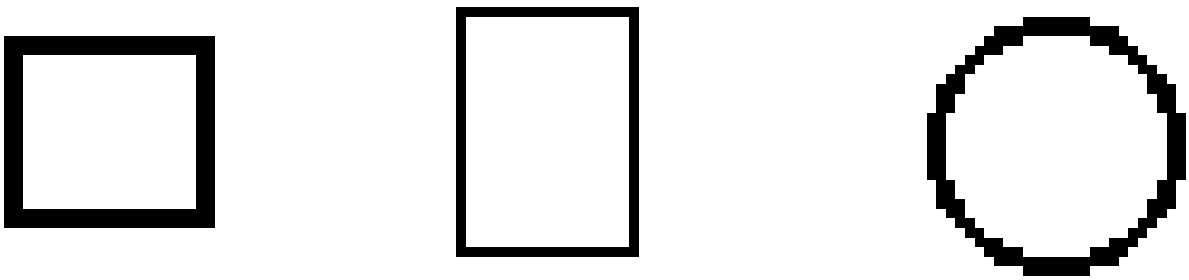


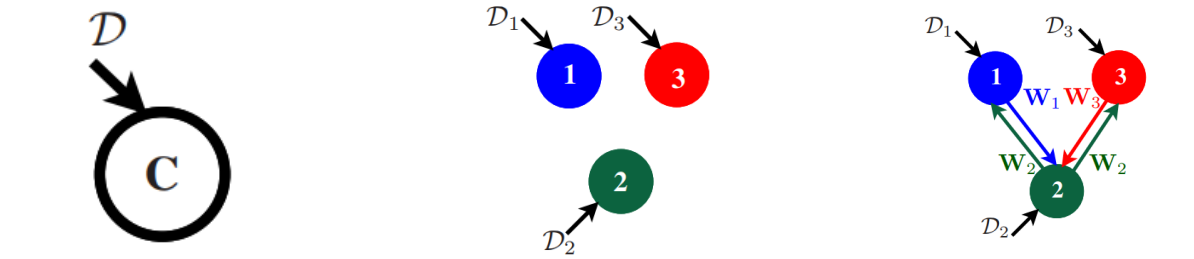
Figura 4: Exemplos de imagens do conjunto de figuras.

4.2 Descrições dos cenários de simulação

Este conjunto de dados foi submetido a um teste cujo objetivo é comparar o desempenho do treinamento distribuído com o desempenho dos treinamentos centralizado e individual. Para isso, foram considerados dois cenários: um com classes balanceadas e outro com classes desbalanceadas, como mostrado na Tabela 1. Vale ressaltar que no treinamento centralizado apenas uma rede é treinada usando todo o conjunto de treinamento, ao passo que no treinamento individual cada rede é treinada apenas com seu próprio subconjunto, sem que ocorra comunicação. Por fim, no treinamento distribuído cada rede é treinada apenas com seu próprio subconjunto ocorrendo comunicação. A Figura 5 ilustra o funcionamento de cada tipo de treinamento considerado.

		Quadrado	Retângulo	Circunferência
Balanceado	Treino	5000	5000	5000
	Teste	5000	5000	5000
Desbalanceado	Treino	5940	689	2819
	Teste	6347	1390	2371

Tabela 1: Número de amostras para cada figura.



(a) Treinamento centralizado.

(b) Treinamento individual.

(c) Treinamento distribuído.

Figura 5: Formas de treinamento.

Um outro aspecto importante a ser definido é a escolha da topologia. Neste caso, foram consideradas duas topologias, sendo que a primeira contém três redes MLP enquanto a segunda possui seis redes MLP. A Figura 6 ilustra as duas topologias escolhidas. Quanto aos parâmetros das redes MLP, todas as redes possuem os mesmos parâmetros que estão representados na Tabela 2.

(a) Topologia com 3 redes.

(b) Topologia com 6 redes.

Figura 6: Topologias consideradas.

4.3 Resultados

Nas Tabelas 3 e 4, são mostradas as métricas apresentadas na Seção 3 alcançadas com cada abordagem para os dados de teste no cenário balanceado e desbalanceado, respectivamente. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, sendo que nos casos distribuído e individual, considerou-se a média das métricas das máquinas.

Parâmetros da rede MLP (classificação por classe)	
Parâmetros com inicialização Glorot	
Função de ativação ReLU para os neurônios das camadas ocultas	
Função de ativação softmax para a camada de saída	
Taxa de aprendizagem	0,001
Número de épocas	1600
Tamanho da camada de entrada	900
Configuração da rede	32-16-3
Função custo	Entropia cruzada categórica ponderada
Algoritmo de otimização	Adam
Parâmetros Adam	$\beta_1 = 0{,}9$ $\beta_2 = 0{,}99$ $\epsilon = 10^{-7}$

Tabela 2: Parâmetros das redes MLP.

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**96,02**	89,40	81,85	88,34	82,45
	P(%)	99,36	89,30	90,81	**99,48**	95,31
	F1(%)	**97,66**	89,12	86,05	93,58	88,37
	MCC	**0,96**	0,84	0,79	0,90	0,83
R	Se(%)	99,34	90,44	89,26	**99,41**	94,56
	P(%)	**95,78**	89,90	79,69	86,82	79,74
	F1(%)	**97,3**	89,97	84,14	92,68	86,35
	MCC	**0,96**	0,85	0,77	0,90	0,81
C	Se(%)	99,90	99,92	99,52	99,94	**99,98**
	P(%)	**100,0**	99,96	99,12	**100,0**	**100,0**
	F1(%)	99,95	99,94	99,31	99,97	**99,99**
	MCC	**1,00**	**1,00**	0,99	**1,00**	**1,00**

Tabela 3: Métricas de desempenho para dados balanceados.

Analisando-se a Tabela 3, nota-se que o treinamento centralizado apresenta as melhores métricas para todas as classes. O treinamento distribuído com a topologia da Figura 6(a) apresenta métricas comparáveis às do caso centralizado para a classe C e um pouco inferiores para as classes Q e R. Comparando com o caso individual, o treinamento distribuído alcançou resultados superiores para a maior parte das métricas (22 de 24 métricas) considerando a mesma topologia. Por fim, os resultados obtidos com o treinamento distribuído para a topologia da Figura 6(a) foram superiores aos da Figura 6(b) para as classes Q e R e próximos para a classe C.

Para o caso desbalanceado (Tabela 4) e topologia da Figura 6(a), o treinamento individual forneceu os melhores resultados, com exceção das métricas da classe C e a precisão da classe R. Nota-se também neste caso que o treinamento distribuído com a topologia da Figura 6(a) superou as métricas do centralizado, com exceção da sensibilidade para a classe Q e precisão para a classe R. O mesmo não é válido para a topologia da Figura 6(b). Por fim, observa-se que o treinamento individual para a topologia da Figura 6(b) obteve resultados piores que os da topologia da Figura 6(a).

Por fim, vale ressaltar que o inesperado desempenho superior do treinamento individual em relação ao distribuído e ao centralizado, observado em alguns casos, será investigado posteriormente.

5 A classificação de arritmias

Nesta seção, discute-se sobre o sistema cardiovascular e o eletrocardiorgrama. Além disso, é proposto o uso de redes MLP para o diagnóstico automático de arritmias. Foram utilizados os sinais de ECG encontrados no banco de dados MIT-BIH *Arrhythmia Database* (MITDB) [26, 27]. A fim de se obter um resultado

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**93,77**	93,54	89,03	93,27	90,45
	P(%)	91,57	**99,80**	98,69	93,48	74,13
	F1(%)	92,66	**96,53**	93,60	93,36	77,39
	MCC	0,80	**0,90**	0,82	0,82	0,63
R	Se(%)	65,06	**98,02**	75,15	71,25	57,64
	P(%)	**71,65**	67,52	43,93	68,75	60,65
	F1(%)	68,20	**78,64**	57,72	69,63	47,27
	MCC	0,63	**0,78**	0,58	0,65	0,44
C	Se(%)	99,63	99,26	98,83	99,72	**99,95**
	P(%)	**100,0**	99,68	98,84	**100,0**	99,42
	F1(%)	99,81	99,47	98,84	**99,86**	99,69
	MCC	**1,00**	0,99	0,99	**1,00**	**1,00**

Tabela 4: Métricas de desempenho para dados desbalanceados.

clinicamente mais realista, os sinais de ECG dos pacientes utilizados no conjunto de treino não estão presentes no conjunto de teste. Realiza-se ainda uma comparação entre os resultados das redes MLP para os casos em que há separação dos pacientes e os casos em que não há separação.

A seção está organizada em 5 subseções. Na Subseção 5.1, é feita uma introdução ao problema. Na Subseção 5.2 é descrito o banco de dados utilizado. Na Subseção 5.3 são descritos os procedimentos de preparação do sinal de ECG para construir os conjuntos de treinamento e teste. Na Subseção 5.4 são descritos os cenários de testes e simulações. Por fim, na Subseção 5.5 são apresentados os resultados de classificação em cada um dos testes realizados.

5.1 Introdução

Por meio do ECG, é possível identificar diversas doenças cardíacas. Dentre elas, a arritmia é a mais comum e corresponde a qualquer distúrbio na taxa, na regularidade e nos locais de origem ou condução dos impulsos elétricos cardíacos. A classificação de batimentos consecutivos em um sinal de ECG é uma etapa fundamental do diagnóstico de arritmias [28]. Como a análise desses batimentos demanda muito tempo do cardiolagista, surgiram os métodos computarizados para identificar automaticamente arritmias.

Em [29] e [30], foram propostas redes MLP para indicar a presença ou não de determinado tipo de arritmia. Para que o sinal de ECG possa ser utilizado como entrada de uma rede MLP, é necesserário que os dados do sinal sejam pré-processados. A etapa inicial do pré-processamento consiste em determinar o complexo QRS, uma vez que utilizando suas variações se torna possível discriminar várias alterações cardíacas. Detectado o complexo QRS, é possível delitmitar os segmentos e intervalos que compõem o batimento cardíaco, o que viabiliza a análise isolada de parâmetros como o intervalo RR ou o segmento ST [31]. Um dos algoritmos mais utilizados na detecção de complexos QRS é o algoritmo de Pan-Tompkins [32]. Ele pode ser usado para separar os batimentos, então a partir desse resultado se torna possível realizar uma série de operações sobre os dados do ECG, como por exemplo calcular o período médio dos batimentos. Assim, pode-se construir o conjunto de dados que constituirão a entrada da rede neural.

5.2 O Banco de Dados

O banco de dados é composto por 48 gravações ambulatoriais de duas derivações que duram trinta minutos. As gravações foram obtidas de pacientes do *Boston's Beth Israel Hospital* entre 1975 e 1979.

As gravações foram digitalizadas a 360 amostras por segundo, com resolução de 11 bits em uma faixa de 10 mV. Dois ou mais cardiologistas anotaram independentemente cada registro e as discordâncias foram resolvidas para obter as anotações de referência legíveis por computador para cada batimento.

5.3 Preparação do Sinal de ECG

Essa etapa consiste em separar os batimentos presentes no sinal de ECG, agrupá-los de acordo com as anotações que lhes foram atribuidas para então formar a entrada da rede neural.

Valendo-se do algoritmo de Pan-Tompkins, foi possível separar os batimentos presentes no banco de dados. Para isso foram excluídos os pacientes 102, 104, 107 e 217 como recomenda a *Association for the Advancement of Medical Instrumentation* (AAMI), haja vista que esses pacientes utilizam marca-passo. A partir dessa separação, foi calculado o período médio dos batimentos. O valor obtido foi de 302,61 amostras, que foi, então, aproximado para 300 amostras. Vale ressaltar que esse valor é apenas uma estimativa do período médio dos batimentos para deterimar o tamanho da entrada da rede, o que é suficiente para essa finalidade. A seguir, os batimentos foram agrupados em suas respectivas classes, a Tabela 5 representa a divisão dos batimentos por classes como proposto pela AAMI.

Classe AAMI	N	S	V
MIT-BIH	Batimento normal (N) Batimento de bloqueio do ramo esquerdo (L) Batimento de bloqueio do ramo esquerdo (R) Batimento de escape atrial (e) Batimento de escape juncional (j)	Batimento atrial prematuro (A) Batimento atrial prematuro aberrante (a) Batimento juncional prematuro (J) Batimento prematuro supraventricular (S)	Contração ventricular prematura (V) Batimento de escape ventricular (E)

Tabela 5: Classes de batimentos de acordo com a AAMI.

Por fim, com os batimentos já agrupados basta organizá-los para formar a entrada da rede neural. Para isso, é conveniente que cada amostra da entrada da rede seja composta por três batimentos. Esses batimentos são o batimento a ser analisado, assim como, os batimentos imediatamente anterior e imediatamente posterior ao que será analisado. Portanto, em uma amostra da entrada são consideradas 900 amostras (três períodos médios) do sinal de ECG, sendo o pico R do batimento a ser analisado o centro das 900 amostras. A Figura 7 representa uma amostra da entrada da rede neural. O batimento a ser analisado nessa amostra da entrada é o batimento central, nela também são considerados os batimentos adjacentes a ele.

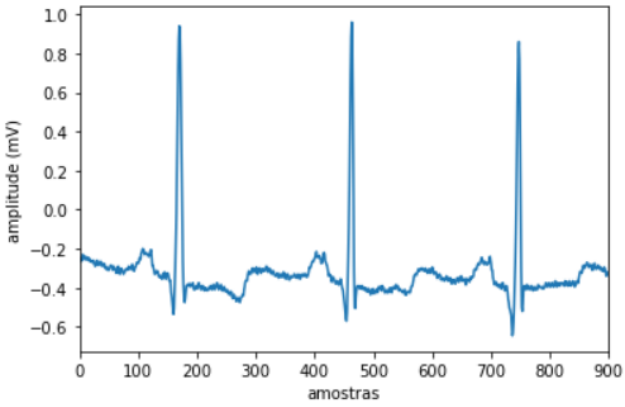


Figura 7: Exemplo de amostra da entrada da rede neural.

Dessa forma, para finalmente construir a entrada da rede neural basta unir todos os agrupamentos de

três batimentos em um grande conjunto. Esse novo conjunto é a entrada da rede e pode ser representado por uma matriz. Cada linha dessa matriz representa um agrupamento de três batimentos, portanto a matriz possui novecentas colunas. Vale ainda ressaltar que o número de linhas representa o número de batimentos considerados no treinamento.

A classificação consiste em identificar a presença ou ausência de arritmia nos batimentos. Contudo, devem ser consideradas as classes específicas de arritmia de cada batimento, respeitando o que foi proposto pela AAMI e exposto na Tabela 5. Os conjuntos de treino e de teste foram formados de modo a tentar reproduzir a realidade clínica, ou seja, há uma divisão de pacientes, 22 pertencem ao conjunto de treino enquanto os outros 22 pertencem ao conjunto de teste. Essa separação obedece ao que foi proposto em [33] e a Tabela 6 mostra quais pacientes pertencem aos conjuntos de treino e teste.

Treino	101 106 108 109 112 114 115 116 118 119 122 124 201 203 205 207 208 209 215 220 223 230
Teste	100 103 105 111 113 117 121 123 200 202 210 212 213 214 219 221 222 228 231 232 233 234

Tabela 6: Divisão dos pacientes entre conjuntos de treino e teste.

5.4 Descrições dos cenários de simulação

Foram considerados dois testes para a classificação de arritmias. O primeiro compara o desempenho do treinamento distribuído, considerando a estratégia Metropolis como regra para os pesos de adaptação, com os treinamentos centralizado e individual. Além disso, foram consideradas duas situações: a situação S_1 em que apenas a classe N foi subamostrada e a a situação S_2 em que todas as classes foram subamostradas. Na Tabela 7, são mostradas a quantidade de dados disponíveis no MITDB e a quantidade de dados usados neste trabalho em cada situação [34].

Classe	Total de batimentos		Quantidade de batimentos utilizados			
			S_1		S_2	
	Treino	Teste	Treino	Teste	Treino	Teste
N	45784	44173	9448	10108	5940	6347
S	940	1837	940	1837	689	1390
V	3783	3218	3783	3218	2819	2371

Tabela 7: Total de batimentos do MITDB e quantidade de batimentos utilizados em cada situação para os conjuntos de treinamento e teste.

O segundo teste visa comparar o desempenho do treinamento distribuído considerando diferentes estratégia de determinação dos pesos de combinação. Foram consideradas as estratégias Metropolis, Uniforme e ACW. Sendo que para o ACW foram consideradas as três possibilidades para o vetor $\boldsymbol{\theta}$, nas quais a diferença entre a estimativa do nó i e o resultado esperado do nó j será representado por ACW_1, a diferença entre as matrizes de pesos da camada de saída dos nós i e j será representada por ACW_2 e a diferença entre a concatenação das matrizes de pesos de todas as camadas dos nós i e j será representada por ACW_3. Por fim, quanto ao número de dados foi considerada a situação S_1.

Para os dois testes foram considerados os mesmos três tipos de treinamento apresentados na Subseção 4.2, assim como foram consideradas as mesmas topologias para os treinamentos individual e distribuído. As Figuras 5 e 6 ilustram respectivamente, os modos de treinamento considerados e as topologias adotadas.

O mesmo é válido para os hiperparâmetros de cada rede MLP a ser treinada. De tal forma, foram considerados os hiperparâmetros expostos na Tabela 2 da Subseção 4.2.

5.5 Resultados

Nas Tabelas 8 e 9 são mostradas as métricas de desempenho alcançadas. Tais métricas representam os resultados alcançados para o primeiro teste para cada topologia e treinamento considerado. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, sendo que no caso individual, considerou-se a média das máquinas.

		Central	Individual		Distribuído										
					Topologia (a)				Topologia (b)						
			Top. (a)	Top. (b)	Nó 1	Nó 2	Nó 3	Média	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Média
N	Se(%)	85,68	88,22	87,67	86,62	83,97	87,08	85,89	83,59	**91,95**	83,52	83,13	83,68	83,06	84,82
	P(%)	90,99	81,43	80,96	90,45	91,18	91,42	91,02	91,94	90,53	92,03	91,20	**92,30**	90,81	91,47
	F1(%)	88,26	84,68	84,18	86,90	87,43	**89,20**	87,84	87,57	86,02	87,56	86,98	87,78	86,76	87,11
	MCC	0,67	0,51	0,49	0,65	0,65	**0,69**	0,66	0,67	0,64	0,65	0,65	0,65	0,66	0,65
S	Se(%)	66,63	9,05	5,67	66,72	65,61	67,13	66,49	71,58	72,62	67,66	71,08	**73,84**	70,20	71,16
	P(%)	64,69	29,63	22,55	64,86	63,58	**69,89**	66,11	61,11	64,89	58,91	64,71	63,00	66,67	63,22
	F1(%)	65,65	13,82	9,01	65,78	64,58	68,48	66,28	65,93	**68,54**	62,98	67,74	67,99	68,39	66,93
	MCC	0,61	0,10	0,05	0,61	0,59	**0,64**	0,61	0,61	**0,64**	0,58	0,63	**0,64**	**0,64**	0,62
V	Se(%)	90,18	86,46	86,62	**90,87**	90,32	89,62	90,27	90,37	88,10	89,44	88,63	88,13	89,38	89,01
	P(%)	**77,34**	76,72	75,21	75,24	72,23	76,61	74,76	76,79	72,66	72,12	72,33	72,13	73,73	73,29
	F1(%)	83,27	81,42	80,49	**85,32**	80,27	82,72	81,77	83,02	79,64	79,85	79,65	79,33	80,80	80,38
	MCC	**0,79**	0,76	0,75	0,77	0,75	0,78	0,77	0,78	0,74	0,75	0,74	0,74	0,75	0,75

Tabela 8: Métricas de desempenho para o caso S_1.

		Central	Individual		Distribuído										
					Topologia (a)				Topologia (b)						
			Top. (a)	Top. (b)	Nó 1	Nó 2	Nó 3	Média	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Média
N	Se(%)	84,04	83,67	85,56	88,45	88,94	**89,20**	88,86	82,93	84,01	87,96	82,51	86,44	86,75	85,10
	P(%)	**90,01**	78,39	77,72	89,67	88,77	89,80	89,41	83,09	82,99	83,08	82,12	82,78	83,79	82,97
	F1(%)	86,92	80,93	81,43	89,05	88,85	**89,50**	89,13	83,01	83,5	85,45	82,31	84,57	85,25	84,02
	MCC	0,67	0,46	0,46	0,71	0,70	**0,72**	0,71	0,53	0,55	0,59	0,53	0,57	0,60	0,56
S	Se(%)	**67,27**	6,84	6,28	63,56	65,01	67,17	65,25	33,91	33,33	32,20	33,04	31,69	33,18	32,89
	P(%)	67,46	22,80	21,06	72,89	74,14	**75,79**	74,27	49,69	52,78	58,46	44,91	55,00	51,39	52,04
	F1(%)	67,36	5,25	9,38	67,90	69,28	**71,22**	69,47	40,31	40,86	41,53	38,07	40,21	40,33	40,22
	MCC	0,62	0,05	0,04	0,63	0,65	**0,67**	0,65	0,34	0,35	0,37	0,31	0,35	0,34	0,34
V	Se(%)	90,30	89,34	86,02	91,03	88,25	89,51	89,60	**91,58**	89,25	88,58	89,41	90,24	90,05	89,85
	P(%)	76,57	72,76	74,95	81,80	**82,77**	82,52	82,36	76,32	75,80	79,68	78,74	78,62	81,72	78,48
	F1(%)	82,87	80,14	80,07	**86,17**	85,42	85,87	85,82	83,25	81,97	83,89	83,74	84,03	85,68	83,76
	MCC	0,78	0,74	0,73	**0,82**	0,81	0,81	0,81	0,78	0,76	0,79	0,78	0,79	0,81	0,79

Tabela 9: Métricas de desempenho para o caso S_2.

Analisando-se as Tabelas 8 e 9, nota-se que o treinamento individual resultou nas piores métricas tanto no caso S_1 como no caso S_2. Apesar disso, os resultados do treinamento individual são comparáveis aos dos treinamentos centralizado e distribuído no caso S_1 para a sensibilidade da classe N e para precisão, F1-*score* e MCC da classe V. Já no cenário S_2, o treinamento individual produziu desempenhos semelhantes aos observados para os treinamentos centralizado e distribuído apenas para a sensibilidade das classes N e V. Ainda assim, a sensibilidade para a classe N só é comparável aos demais tipos de treinamento considerando-se o treinamento individual com a topologia da Figura 6(b). Em contrapartida, para a sensibilidade da classe V, apenas a topologia da Figura 6(a) leva a um desempenho por parte do treinamento individual comparável aos treinamentos centralizado e distribuído. Cabe ressaltar que em ambos os cenários as métricas obtidas com o treinamento individual para a classe S foram significativamente inferiores àquelas alcançadas com os demais tipos de treinamento, qualquer que seja a topologia considerada.

Ainda com relação à Tabela 8, nota-se que a diferença entre as métricas alcançadas com a média das máquinas no treinamento distribuído e o treinamento centralizado no cenário S_1 é inferior em módulo a 1,5% para a sensibilidade, precisão e F1-*score* de todas as classes, exceto para a precisão e o F1-*score* da classe V. Nesses últimos casos, as métricas obtidas com o treinamento centralizado foram superiores, sendo que o treinamento distribuído considerando-se a topologia da Figura 6(a) apresentou desempenho mais próximo do treinamento centralizado do o caso em que se considera a topologia da Figura 6(b). Analogamente, a diferença entre o MCC alcançado com o treinamento distribuído e aquele atingido com o treinamento centralizado

foi de no máximo 0,02 em módulo, exceto para a classe V considerando-se o treinamento distribuído com a topologia da Figura 6(b). Nesse caso, a diferença foi de 0,04, com o treinamento distribuído apresentando o melhor desempenho.

Realizando-se a mesma análise para o cenário S_2, nota-se que o treinamento distribuído com a topologia da Figura 6(a) levou a resultados superiores aos do treinamento centralizado para todas as métricas, exceto para a precisão da classe N e para a sensibilidade das classes S e V. Por outro lado, nota-se que a adoção da topologia da Figura 6(b) em vez daquela da Figura 6(a) leva a um desempenho consideravelmente inferior para o treinamento distribuído, exceto para a sensibilidade da classe N e para as métricas da classe V. Em particular, nota-se uma degradação significativa para as métricas da classe S, com diferenças superiores a 30% para a sensibilidade, precisão e F1-*score* entre os treinamentos distribuídos considerando-se a topologia da Figura 6(a) e aquela da Figura 6(b). Em comparação com o treinamento centralizado, o treinamento distribuído com a topologia da Figura 6(b) levou a desempenhos melhores apenas para a sensibilidade da classe N e para as métricas da classe V. Uma possível interpretação para isso reside no fato de que quando a rede de computadores apresenta muitas máquinas, a divisão do conjunto de treinamento $\mathcal{D}$ pode levar a subconjuntos $\mathcal{D}_i$ com cardinalidades muito pequenas. Nesse caso, a quantidade de exemplos a que cada máquina tem acesso pode ser insuficiente para a obtenção de um modelo adequado, o que não é mitigado pela etapa de combinação representada pela Equação (9). Cabe notar que a topologia da Figura 6(b) apresenta o dobro do número de máquinas da topologia da Figura 6(a). Além disso, a comparação entre os cenários S_1 e S_2 sugere que esse efeito é agravado pela subamostragem de todas as classes, o que implica ainda menos exemplos disponíveis para o treinamento de cada máquina quando se considera a topologia da Figura 6(b) para o cenário S_2.

Nas Tabelas 10 e 11 são mostradas as métricas alcançadas com segundo teste para cada topologia e estratégia considerada. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, considerando-se a média das máquinas.

		Metrópolis	Uniforme	ACW (1)	ACW (2)	ACW (3)
N	Se(%)	85,89	85,64	**89,76**	84,17	85,39
	P(%)	**91,02**	88,59	83,53	88,51	86,91
	F1(%)	**87,84**	87,09	86,53	86,28	86,13
	MCC	**0,66**	0,62	0,57	0,61	0,59
S	Se(%)	**66,49**	51,50	16,49	52,22	42,13
	P(%)	**66,11**	65,48	43,10	64,85	55,69
	F1(%)	**66,28**	57,60	23,86	57,79	47,73
	MCC	**0,61**	0,53	0,21	0,53	0,43
V	Se(%)	90,27	90,72	89,34	**90,82**	90,32
	P(%)	74,76	74,02	**79,90**	71,84	74,68
	F1(%)	81,77	81,52	**84,36**	80,21	81,76
	MCC	0,77	0,76	**0,80**	0,75	0,77

Tabela 10: Métricas de desempenho para topologia da Figura 6(a).

Analisando-se as Tabelas 10 e 11, nota-se que a estratégia ACW (1) resultou nas piores métricas para a classe S tanto para a topologia da Figura 6(a) como para a topologia da Figura 6(b). Apesar disso, os resultados da estratégia ACW (1) são comparáveis aos das demais estratégias para a classe N e apresenta as melhores métricas para a classe V com exceção da sensibilidade para o caso da topologia da Figura 6(a). Comparando as diferentes formas de implementação do ACW, nota-se que independente da topologia há uma grande variação entre os resultados. Isso é válido em especial, para a classe S como um todo e para a precisão das classes N e V. Entre as diferentes formas do ACW, a que apresentou resultados mais consistentes foi a estratégia ACW (2). Apesar disso, apresenta métricas inferiores àquelas apresentadas pelo ACW (1) para a classe V. Em relação às demais regras de escolha para os pesos, nota-se que o desempenho da regra adaptativa, de forma geral, foi inferior ao Metropolis e ao modo Uniforme. Cabe ressaltar que isso não é observado para o ACW (2), uma vez que apresenta métricas comparáveis ao modo Uniforme.

		Metrópolis	Uniforme	ACW (1)	ACW (2)	ACW (3)
N	Se(%)	84,82	82,65	85,78	84,34	**86,30**
	P(%)	**91,47**	89,49	83,50	90,28	85,83
	F1(%)	**87,11**	85,93	84,62	87,01	85,20
	MCC	**0,65**	0,61	0,53	0,64	0,54
S	Se(%)	**71,16**	58,47	19,77	63,52	26,29
	P(%)	**63,22**	60,15	40,90	62,87	46,99
	F1(%)	**66,93**	59,27	26,60	63,18	33,44
	MCC	**0,62**	0,54	0,22	0,58	0,29
V	Se(%)	89,01	90,37	**90,44**	89,42	87,81
	P(%)	73,29	71,99	**74,74**	74,50	74,44
	F1(%)	80,38	80,13	**81,50**	81,28	80,57
	MCC	0,75	0,75	**0,77**	**0,77**	0,75

Tabela 11: Métricas de desempenho para topologia da Figura 6(b).

Comparando, agora, o desempenho do método Metropolis com os demais métodos, nota-se que esse foi superior aos demais para a precisão, F1-*score* e MCC para a classe N e para todas as métricas para a classe S, o que é válido para as duas topologias consideradas. Além disso, para a classe V resultou em métricas próximas das maiores métricas alcançadas para essa classe, com exceção da precisão, F1-*score* e MCC no caso da topologia da Figura 6(a). Quanto à regra Uniforme, nota-se um desempenho semelhante ao observado no modo Metropolis, com exceção da classe S que apresentou métricas ligeiramente inferiores.

6 Conclusão

A classificação automática de arritmias utilizando métodos computadorizados é uma área com muito espaço para desenvolvimento. Por sua vez, o treinamento distribuído de redes neurais tem grande potencial para aplicações práticas, uma vez que visa resolver problemas de privacidade que constituem uma preocupação real em diversos tipos de situações.

Neste trabalho, avaliaram-se os efeitos da adoção de treinamentos distribuídos para redes perceptron multicamadas. Constatou-se que com esse tipo de treinamento pode-se alcançar um desempenho comparável com a abordagem centralizada, em que se utiliza uma única máquina para treinar a rede neural com todos os dados do conjunto de treinamento. Nesse caso, pode-se conciliar o desempenho com a proteção à privacidade dos dados.

Além disso, os resultados mostram que, para isso, é necessário adotar uma topologia adequada. Em particular, deve-se garantir que o número de máquinas não seja excessivamente elevado para que a divisão do conjunto de dados entre as máquinas não leve a subconjuntos de treinamento muito pequenos.

Diante dos resultados obtidos, a regra adaptativa ACW não parece ser uma boa opção para o cálculo dos pesos, pelo menos da forma considerada aqui. Em trabalhos futuros, vale a pena implementar outras regras adaptativas como, por exemplo, a dos Mínimos Quadrados de [11]. Regras adaptativas podem ser úteis quando uma das máquinas tem dados mais ruidosos que as outras, o que pode ocorrer em aplicações distribuídas com dados médicos.

Por fim, verificou-se que diferentes regras de atribuição para os pesos de combinação resultam em desempenhos diferentes considerando a mesma topologia. Observou-se com o caso de classificação de arritmias que uma estratégia adaptativa pode apresentar um desempenho inferior ao de uma determinística. Até o momento, a regra Metrópolis destaca-se como a que resulta no melhor desempenho.

A Perceptron de Rosenblatt

O perceptron de Rosenblatt é uma estrutura simples usada na classificação de padrões separáveis linearmente, ou seja, padrões que podem ser separados por um hiperplano. Sua estrutura é baseada em um

único neurônio que possui pesos e *bias* ajustáveis. Os parametros são ajustados por meio de um algoritmo de aprendizado que foi desenvolvido por Rosenblatt. Além disso, Rosenblatt provou que se o conjunto de dados usado para treinar o perceptron for obtido de duas classes separáveis linearmente, então o algoritmo converge e o perceptron define uma fronteira de decisão na forma de um hiperplano entre as classes. Dessa forma, é possível classificar cada elemento do conjunto de dados como pertencente a uma das duas classes. Nos subapêndices seguintes, são descritos a construção e o funcionamento do perceptron de Rosenblatt, bem como o algoritmo de treinamento. Mais detalhes da implementação e de experimentos envolvendo o perceptron de Rosenblatt podem ser encontrados em [35, 36].

A.1 Construção e Funcionamento

O perceptron de Rosenblatt é construído a partir de um neurônio não linear, cujo modelo consiste em uma combinação linear dos dados de entrada, seguida de uma função não linear. Os dados de entrada são ponderados pelos pesos e somandos. O resultado é ainda somado a um *bias*. Por fim, ao resultado dessa operação é aplicada a função não linear, produzindo uma saída. O esquema do perceptron de Rosenblatt está representado na Figura 8. Nessa figura temos as entradas denotadas por $x_1, x_2, x_3, x_4, ..., x_n$, os pesos denotados por $w_1, w_2, w_3, w_4, ..., w_n$ e o *bias* denotado por b. Assim, a entrada da função não linear, denotada por z, é calculada como

$$z = \sum_{i=1}^{n} w_i x_i + b. \tag{14}$$

A função não linear adotada pelo modelo é a função sinal, definida como

$$\text{sinal}(x) = \begin{cases} -1, & \text{se } x \leq 0 \\ +1, & \text{se } x > 0. \end{cases} \tag{15}$$

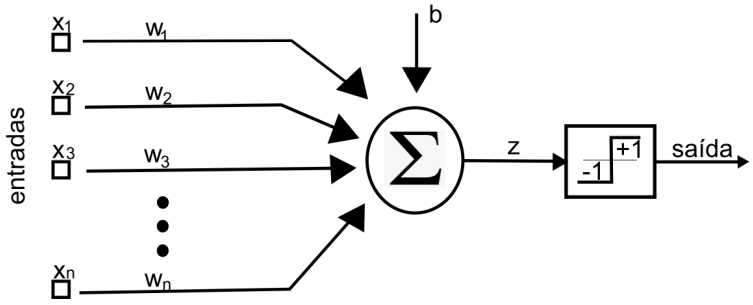


Figura 8: Modelo geral do perceptron de Rosenblatt

O objetivo do perceptron é classificar corretamente os dados de entrada em duas classes distintas, as classes C_1 e C_2. Para isso, podemos tomar a decisão com base na saída do perceptron, de tal forma que caso um ponto representado pelos dados de entrada tenha saída $+1$, ele será atribuido à classe C_1 e caso tenha saída -1, ele será atribuido à classe C_2.

A.2 Algoritmo de Aprendizado

Agrupando os dados de entrada no vetor

$$\mathbf{x}(n) = \begin{bmatrix} +1 & x_1(n) & x_2(n) & x_3(n) & \dots & x_n(n) \end{bmatrix}^{\mathrm{T}}, \tag{16}$$

assim como os pesos e o *bias* no vetor

$$\mathbf{w}(n) = \begin{bmatrix} b(n) & w_1(n) & w_2(n) & w_3(n) & \dots & w_n(n) \end{bmatrix}^{\mathrm{T}}, \tag{17}$$

podemos agora reescrever a Equação (14) na forma

$$z(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n), \tag{18}$$

e a saída do perceptron como

$$y(n) = \text{sinal}(z(n)), \tag{19}$$

em que $(\cdot)^{\mathrm{T}}$ denota a transposição.

No ajuste dos pesos, deve-se conhecer de antemão a resposta desejada, dada por

$$d(n) = \begin{cases} +1, & \text{se } x(n) \in C_1 \\ -1, & \text{se } x(n) \in C_2, \end{cases} \tag{20}$$

e então, comparar a saída $y(n)$ com a resposta desejada $d(n)$, obtendo-se o sinal de erro

$$e(n) = d(n) - y(n). \tag{21}$$

O pesos são então ajustados a fim de minimizar uma função desse erro, denominada função custo e denotada por J. A função custo mais comum é o erro quadrático dado por

$$J = e^2(n). \tag{22}$$

Utilizando o método do gradiente descendente estocástico [37], obtém-se a seguinte equação de atualização dos pesos

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \frac{\partial J}{\partial \mathbf{w}(n)}, \tag{23}$$

em que η é um passo de adaptação. O passo de adaptação é uma constante positiva, usualmente, escolhida no intervalo $0 < \eta \leq 1$, embora em algumas aplicações seja possível considerar passos maiores que 1. Um passo de adaptação muito baixo torna o aprendizado do perceptron muito lento, enquanto que um passo de adaptação muito alto provoca oscilações nos pesos e impede a convergência do processo de aprendizado. Em alguns casos, passos elevados podem inclusive levar o algoritmo à divergência, quando os pesos ultrapassam o maior valor representado no software de simulação numérica, ou seja, vão para infinito.

Cabe observar que a função sinal não é derivável em todos os pontos. Portanto, o algoritmo não leva em conta essa derivada, simplificando o gradiente pela a aproximação

$$\frac{\partial J}{\partial \mathbf{w}(n)} \approx -e(n)\mathbf{x}(n). \tag{24}$$

Finalmente, por meio do pseudocódigo representado no Algoritmo 1, podemos descrever os passos do processo iterativo que compõem o algoritmo de aprendizado. Esse processo iterativo é responsável pelo treinamento do perceptron, de tal forma que, quando ele acaba, o perceptron está devidamente treinado.

Algoritmo 1: Algoritmo de Aprendizado

```
w(0) ← 0
b(0) ← 0
para n ← 0; n < N; n ← n + 1 faça
    y(n) ← sinal(w^T(n)x(n))
    e(n) ← d(n) − y(n)
    ajuste ← η * e(n) * x(n)
    w(n + 1) ← w(n) + ajuste
fim
```

B A rede MLP (*multilayer perceptron*)

No apêndice anterior, foi analisado o Perceptron de Rosenblatt, que, basicamente, é uma rede neural de um único neurônio. Contudo, ele é limitado à classificação de padrões separáveis linearmente. Para superar esse problema estudaremos agora a rede conhecida como perceptron multicamada (MLP - *multilayer*

perceptron). A estrutura básica dessa rede consiste na presença de uma ou mais camadas ocultas, de um alto grau de conectividade e do fato de que cada neurônio possui uma função de ativação diferenciável.

Na Figura 9, está representada uma rede MLP com 3 camadas ocultas, cada uma com um número diferente de neurônios, nos quais são calculados uma combinação linear z e uma função de ativação $y = \mathrm{f}(z)$. Convém denotá-la como uma rede 3-2-4-1, já que há três neurônios na primeira camada oculta, enquanto há dois neurônios na segunda, quatro na terceira camada oculta e um neurônio na camada de saída.

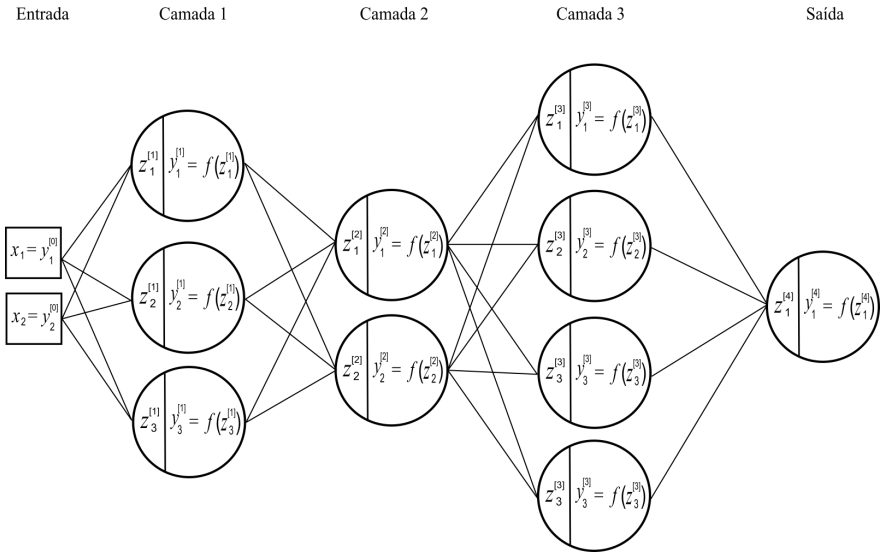


Figura 9: Rede MLP de configuração 3-2-4-1

Como agora existem múltiplas camadas com diversos neurônios, se torna conveniente definir os vetores $\mathbf{b}^{[k]}$, $\mathbf{z}^{[k]}$ e $\mathbf{y}^{[k]}$, que representam respectivamente os *bias*, o resultado das combinações lineares e as saídas das funcões de ativação dos N_k neurônios das camadas $k = 1, 2, 3, \ldots, L$ da rede, ou seja,

$$\mathbf{b}^{[k]} = \begin{bmatrix} b_1^{[k]} & b_2^{[k]} & b_3^{[k]} & \ldots & b_{N_k}^{[k]} \end{bmatrix}^{\mathrm{T}}, \tag{25}$$

$$\mathbf{z}^{[k]} = \begin{bmatrix} z_1^{[k]} & z_2^{[k]} & z_3^{[k]} & \ldots & z_{N_k}^{[k]} \end{bmatrix}^{\mathrm{T}}, \tag{26}$$

$$\mathbf{y}^{[k]} = \begin{bmatrix} y_1^{[k]} & y_2^{[k]} & y_3^{[k]} & \ldots & y_{N_k}^{[k]} \end{bmatrix}^{\mathrm{T}}. \tag{27}$$

Também é conveniente definir a matriz de pesos da camada k como

$$\mathbf{W}^{[k]} = \begin{bmatrix} w_{11}^{[k]} & w_{12}^{[k]} & w_{13}^{[k]} & \ldots & w_{1N_{k-1}}^{[k]} \\ w_{21}^{[k]} & w_{22}^{[k]} & w_{23}^{[k]} & \ldots & w_{2N_{k-1}}^{[k]} \\ w_{31}^{[k]} & w_{32}^{[k]} & w_{33}^{[k]} & \ldots & w_{3N_{k-1}}^{[k]} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{N_k1}^{[k]} & w_{N_k2}^{[k]} & w_{N_k3}^{[k]} & \ldots & w_{N_kN_{k-1}}^{[k]} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^{[k]} \\ \mathbf{w}_2^{[k]} \\ \mathbf{w}_3^{[k]} \\ \vdots \\ \mathbf{w}_{N_k}^{[k]} \end{bmatrix}. \tag{28}$$

O vetor $\mathbf{z}^{[k]}$ é obtido como

$$\mathbf{z}^{[k]} = \mathbf{W}^{[k]}\mathbf{y}^{[k-1]} + \mathbf{b}^{[k]}, \tag{29}$$

enquanto que o vetor de saída da camada k, é obtido aplicando a função de ativação elemento por elemento do vetor $\mathbf{z}^{[k]}$, ou seja,

$$\mathbf{y}^{[k]} = \mathrm{f}(\mathbf{z}^{[k]}), \tag{30}$$

em que $\mathrm{f}(\cdot)$ representa a função de ativação. Ainda vale ressaltar que $\mathbf{y}^{[0]}$ é a entrada da rede e $\mathbf{y}^{[L]}$ é a resposta da rede, vetor que contém as saídas dos neurônios da camada de saída da rede.

As redes MLP se valem de um conjunto de dados para que possam aprender. Esse conjunto é usualmente dividido em um conjunto de treino e um conjunto de teste. O primeiro conjunto é composto pelos dados que contém a informação sobre o que se quer classificar com seus respectivos rótulos, esse conjunto é utilizado para atualizar os pesos durante o processo de aprendizado. O segundo conjunto também é composto por dados já rotulados, porém esse conjunto é utilizado apenas para testar se, após o término do aprendizado, a rede classifica os dados de entrada corretamente.

Essa divisão dos dados facilita a identificação de alguns problemas como o *overfitting*. Esse fenômeno ocorre quando um modelo apresenta um desempenho excelente para o conjunto de treino e ao mesmo tempo um desempenho péssimo para o conjunto de teste. Isso ocorre pois o modelo aprendeu tão bem sobre conjunto de treino que perdeu a capaciade de generalização para outros dados que não os do conjunto de treino, inclusive para os dados do conjunto de teste. Ou seja, o modelo se especializou tanto que só consegue resolver o problema para os dados do conjunto de treino. É possível encontrar mais detalhes sobre *overfitting* em [38].

O processo de aprendizagem pode ser dividido em duas partes: propagação direta e retropropagação, como detalhado no Apêndice B.4. Vale ainda ressaltar que o treinamento é feito em épocas. Uma época corresponde ao ciclo em que todos os dados de treinamento são usados. Quando há poucos dados de treinamento, é comum considerar várias épocas. Antes de uma nova época ter início, os dados de treinamento são misturados para gerar uma certa diversidade nos dados. Assim, o número de épocas é o número de vezes que todo o conjunto de treino é analisado pela rede.

B.1 Inicialização dos pesos e *bias*

A velocidade do aprendizado de uma rede MLP está intimamente relacionada aos valores iniciais dos pesos e *bias*, ao valor da taxa de aprendizado, à topologia da rede e ao algoritmo de aprendizado empregado. O modo com que os pesos e *bias* são inicializados influencia em quão rápido esses parâmetros irão convergir para o valor final, além de afetar a eficiência da rede. Parâmetros mal inicializados podem aumentar consideravelmente o número épocas de treinamento, além disso, podem também, estagnar o treinamento em um mínimo local [39]. Dessa forma, determinar quais devem ser os valores iniciais para os pesos e *bias* passa a ser uma tarefa que tenta reduzir o tempo de treinamento.

Diversos métodos de inicialização dos pesos e *bias* para redes MLP foram sugeridos na literatura. Caso os parâmetros sejam inicializados com zero, eles assumem valores idênticos durante o treinamento. Então, em [40], foi sugerido que os parâmetros fossem inicializados com valores aleatórios, o que permite produzir múltiplas soluções para o mesmo problema dado a aleatoriedade dos valores iniciais. A eficiência desse método, contudo, depende principalmente da distribuição inicial dos pesos e *bias*. Para suprir essa depedência foram propostas novas formas de inicialização dos pesos e *bias*. Um estudo sobre a eficácia de algumas dessas propostas pode ser encontrado em [41].

Na Tabela 12 são apresentadas algumas formas de inicialização. Além disso, na coluna da direita estão representados os intervalos dos quais devem ser escolhidos, aleatoriamente, os valores para os parâmetros. Por fim, vale ressaltar que N_{in} representa a dimensão da entrada do neurônio, ao passo que N_{out} representa a dimensão da saída do neurônio.

B.2 Funções de Ativação

As funções de ativição são parte essencial do aprendizado de uma rede neural, são elas que permitem com que uma rede MLP consiga realizar tarefas não lineares. Entre as utilidades das funções de ativação está a possibilidade de limitar a resposta à um intervalo de valores, o que é útil quando se está trabalhando com um rede cuja resposta é a probabilidade de algo acontecer, que deve estar entre 0 e 1.

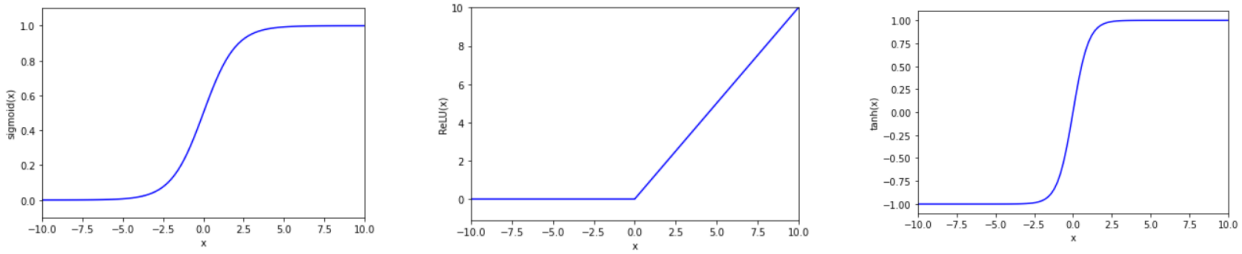
É importante lembrar que as funções de ativação devem ser diferenciáveis. Algumas funções de ativção comumente usadas são:

Método de inicialização	Possíveis valores
Zeros	Todos os parâmetros são incializados com zero
Uns	Todos os parâmetros são incializados com um
Fahlman [42]	Distribuição uniforme em intervalos variando de $[-4{,}0; 4{,}0]$ até $[-0{,}5; 0{,}5]$
Bottou [43]	$[-a/N_{in}; a/N_{in}]$, em que a é escolhido de forma que a variância dos pesos corresponde aos pontos de máxima curvatura da função de ativação
Boers e Kuiper [44]	$[-\sqrt{3/N_{in}}; \sqrt{3/N_{in}}]$
He normal [45]	média 0; desvio padrão $\sqrt{2/N_{in}}$
He uniforme [45]	$[-\sqrt{6/N_{in}}; \sqrt{6/N_{in}}]$
Xavier normal [46]	média 0; desvio padrão $\sqrt{2/(N_{in}+N_{out})}$
Xavier uniforme [46]	$[-\sqrt{6/(N_{in}+N_{out})}; \sqrt{6/(N_{in}+N_{out})}]$

Tabela 12: Exemplos de inicialização dos pesos e *bias*.

1. A função sigmoidal
$$\text{sigmoid}(x) = \frac{1}{1+\text{e}^{-x}}; \tag{31}$$
2. A função ReLU (*Rectified Linear Unit*)
$$\text{relu}(x) = \begin{cases} 0, & \text{se } x \leq 0 \\ x, & \text{se } x > 0 \end{cases}; \text{e} \tag{32}$$
3. A função tangente hiperbólica
$$\tanh(x) = \frac{\text{e}^{x} - \text{e}^{-x}}{\text{e}^{x} + \text{e}^{-x}}. \tag{33}$$

As Figuras 10a, 10b e 10c representam, respectivamente, gráficos das funções sigmoid, ReLU e tangente hiperbólica.



(a) Função sigmoid. (b) Função ReLU. (c) Função tangente hiperbólica.

Figura 10: Funções de ativação.

B.3 Função Custo

No conxteto de redes neurais, uma função custo J exerce o papel de medir quão próximo a saída da rede está da resposta desejada. Para isso, compara o vetor de saída da rede $\mathbf{y}^{[L]}$ termo a termo com o vetor das respostas desejadas $\mathbf{d}$ definido por
$$\mathbf{d} = \begin{bmatrix} d_1 & d_2 & d_3 & \dots & d_{N_{out}} \end{bmatrix}^{\text{T}}. \tag{34}$$

Essa comparação é feita por meio de uma função de perda, denotada por $\mathcal{L}(\mathbf{y}^{[k]}, \mathbf{d})$, de tal forma que a função custo é a média da função de perda, ou seja,
$$J = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} \mathcal{L}(y_i^{[L]}, d_i). \tag{35}$$

A escolha da função custo depende da aplicação da rede, para problemas de regressão é recomendado que seja utilizado o erro quadrático médio, que é definido por

$$J_{EQM} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} (d_i - y_i^{[L]})^2, \tag{36}$$

ao passo que para problemas de classificação é recomendado que se utilize a entropia cruzada, cuja definição é

$$J_{EC} = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} d_i \log(y_i^{[L]}) + (1 - d_i) \log(1 - y_i^{[L]}), \tag{37}$$

em que $\log(\cdot)$ denota o logaritmo natural.

Um dos modos de reduzir o *overfitting* é aplicar um termo de regularização à função custo da rede. A adição desse termo reduz o *overfitting* pois funciona como uma forma de controle do ajuste dos pesos, fazendo com que a rede não perca a capacidade de generalização. Uma das técnicas de regularização mais comum é a regularização L2, quando aplicada ao erro quadrático médio resulta em

$$J_{EQM_{L2}} = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} (d_i - y_i^{[L]})^2 + \frac{\lambda}{2N_{out}} \sum_{i=1}^{N_{out}} (w_i^{[L]})^2 \tag{38}$$

e quando aplicada à entropia cruzada resulta em

$$J_{EC_{L2}} = -\frac{1}{N_{out}} \sum_{i=1}^{N_{out}} d_i \log(y_i^{[L]}) + (1 - d_i) \log(1 - y_i^{[L]}) + \frac{\lambda}{2N_{out}} \sum_{i=1}^{N_{out}} (w_i^{[L]})^2. \tag{39}$$

É possível encontrar mais detalhes sobre as deduções matemáticas que explicam como o método da regularização L2 reduz o *overfitting* em [38].

B.4 O Algoritmo de Aprendizado

O algoritmo de aprendizado é chamado de algoritmo de retropropagação (do inglês *backpropagation* e é responsável pelo treinamento da rede neural. Ele pode ser interpretado como uma generalização do algoritmo do perceptron de Rosenblatt. Com esse algoritmo conseguimos atualizar os parâmetros e fazer com que a rede neural nos forneça a resposta esperada. Ele é constituído de duas partes, sendo a primeira a propagação direta enquanto que a segunda é a retropropagação.

A propagação direta é a parte do algoritmo de aprendizado responsável por calcular as saídas de cada neurônio da rede e assim obter a saída da rede como um todo. Isso é feito camada por camada. A camada k vale-se dos dados da camada $k-1$ para obter a combinação linear em cada um dos seus neurônios e então é aplicada a função de ativação. Esse procedimento, cujo funcionamento foi descrito pelas equações (29) e (30), se repete em todas as camadas até a última camada da rede, que é responsável por produzir a saída da rede.

A retropropagação é responsável por ajustar os valores dos pesos e dos *bias* de cada um dos neurônios da rede e assim aperfeiçoar o seu funcionamento, minimizando o erro. Assim como no caso do perceptron, isso é feito com o auxílio da função custo J. Novamente, os pesos são ajustados obedecendo o método do gradiente descendente estocástico, que foi apresentado na Equação (23). Contudo, nesse caso a aproximação na Equação (24) não é mais válida e as derivadas devem ser levadas em conta. Essa parte do algoritmo de aprendizado calcula o gradiente da função custo em relação aos pesos. Primeiro é calculado o gradiente na camada de saída, em seguida calcula-se o gradiente da penúltima camada utilizando a regra da cadeia em conjunto com o gradiente já calculado. Esse procedimento se repete camada por camada até chegar à camada de entrada, o que justifica o nome do algoritmo. Com os resultados dos gradientes para cada uma das camadas, basta aplicar o método do gradiente descendente estocástico para atualizar os pesos da camada k, ou seja,

$$\mathbf{W}^{[k]}(n+1) = \mathbf{W}^{[k]}(n) - \eta \frac{\partial J}{\partial \mathbf{W}^{[k]}(n)}. \tag{40}$$

Quanto ao vetor de *bias* o raciocínio é analago, de tal forma que

$$\mathbf{b}^{[k]}(n+1) = \mathbf{b}^{[k]}(n) - \eta \frac{\partial J}{\partial \mathbf{b}^{[k]}(n)}. \tag{41}$$

Por fim, vale enfatizar o caráter de retropropagação do algoritmo, uma vez que para calcular o ajuste da camada k é necessário antes ter calculado o da camada $k+1$. As equações completas do algoritmo e mais detalhes de sua derivação podem ser encontradas em [47].

B.5 Hiperparâmetros

Hiperparâmetros são os parâmetros usados para determinar a estrutura e arquitetura da rede neural. Não são treináveis pela rede. Os hiperparâmetros afetam principalmente a velocidade e qualidade do processo de aprendizado. A Tabela 13 descreve alguns hiperparâmetros e suas características. Escolher qual o melhor valor para cada hiperparâmetro é uma tarefa empírica. O processo para determinar o valor ideal requer testes com os vários valores possíveis e análise dos resultados para se chegar a um valor ótimo.

Hiperparâmetros	Características
Taxa de aprendizagem	Atribuir um valor inadequado resulta em um modelo com baixa capacidade de aprendizado devido à sua função no ajuste dos parãmetros, vide Equações (40) e (41).
Número de épocas	Número de vezes que o conjunto de treinamento é analisado pela rede.
Número de camadas escondidas	Depende da complexidade do problema a ser resolvido, um número maior que o suficente pode resultar em *overfitting*.
Número de neurônios em cada camada	Usar poucos neurônios pode não adequar o modelo de forma satisfatória, ao passo que um elevado número de neurônios pode resultar em *overfitting*, se esse número for excessivo o processo de treinamento se torna muito custoso computacionalmente.
Escolha da função de ativação	Introduzem a não linearidade à rede, permitindo extrapolar os limites do perceptron, além disso controlam quão bem o modelo irá aprender sobre o conjunto de treinamento.
Escolha da função custo	Influência no treinamento da rede, deve ser escolhida de acordo com o tipo de problema a ser resolvido, como mencionado no Apêndice B.3
Parâmetro de regularização λ	Afeta em quanto o termo de regularização alterará a função custo
Inicialização dos pesos e *bias*	Interfere na velocidade e eficiência do processo de aprendizado
Algoritmo de otimização	Controla como os pesos serão atualizados durante o processo de aprendizagem

Tabela 13: Exemplos de hiperparâmetros e suas características.

Duas técnicas comuns para encontrar os melhores hiperparâmetros são *grid search* e o *random search*. A primeira técnica irá testar todas as combinações possíveis dos hiperparâmetros, exaustivamente. Basicamente, fornece alguns valores de hiperparâmetros e testa todas as combinações plotando em um plano cartesiano, por isso o nome de *grid*. Em seguida, seleciona os hiperparâmetros que obtiveram o melhor desempenho. O *random search*, por sua vez, atribui valores aleatórios de hiperparâmetros dentro de um intervalo e testa essas combinações. Então, para escolher qual valor de hiperparâmetro que será utilizado, se analisa qual combinação produziu o melhor resultado.

Uma terceira técnica utilizada para encontrar quais são os hiperparâmetros mais adequados envolve algoritmos genéticos (AG). Esses algoritmos são capazes de simular a evolução biológica e encontrar uma solução que seja eficiente o suficente para um certo problema. Para isso, as possíveis soluções de um problema são consideradas como indivíduos, os quais são submetidos à uma simulação da seleção natural, ou seja, os indivíduos mais adaptados àquele problema são selecionados e transmitem suas características aos indivíduos da próxima geração. Esse processo se repete até que uma solução eficiente o suficente seja encontrada para resolver o problema. No contexto das redes MLP, um indivíduo pode ser considerado como uma rede e suas configurações. Assim ao término da execução do AG, os hiperparâmetros foram determinados. Mais detalhes da utilização de algoritmos genéticos para otimizar os hiperparâmetros de uma rede MLP podem ser encontrados em [48].

Um hiperparâmetro importante é o algoritmo de otimização. Esses algoritmos são métodos usados para controlar como os pesos da rede neural serão alterados, a fim de minimizar a função custo. No Apêndice A.2 foi mencionado o algoritmo do gradiente descendente estocástico. Além desse algoritmo deve-se dar atenção especial ao algoritmo Adam (*Adaptive Moment Estimation*). Adam combina as vantagens de duas outras extensões do gradiente descendente estocástico, especificamente do AdaGrad (*Adaptive Gradient*) [49] e do RMSProp (*Root Mean Square Propagation*) [50]. A ideia por trás desse algoritmo é calcular a média móvel exponencial do gradiente e do quadrado do gradiente, enquanto os parâmetros β_1 e β_2 controlam a taxa de decaimento das médias móveis. O pseudocódigo representado no Algoritmo 2 descreve o funcionamento do algoritmo Adam. As equações completas do algoritmo e mais detalhes de seu funcionamento podem ser encontradas em [51].

Algoritmo 2: Algoritmo Adam

$w(0) \leftarrow incializa$
$b(0) \leftarrow incializa$
$m(0) \leftarrow incializa$
$v(0) \leftarrow incializa$
$t \leftarrow 0$
para $n \leftarrow 0; n < N; n \leftarrow n + 1$ **faça**
  $t \leftarrow t + 1$
  $m(n+1) \leftarrow \beta_1 m(n) + (1 - \beta_1) \frac{\partial J}{\partial \mathbf{W}(n)}$
  $v(n+1) \leftarrow \beta_2 v(n) + (1 - \beta_2)(\frac{\partial J}{\partial \mathbf{W}(n)})^2$
  $m_{corrigido}(n+1) \leftarrow \frac{m(n+1)}{1-\beta_1^t}$
  $v_{corrigido}(n+1) \leftarrow \frac{v(n+1)}{1-\beta_2^t}$
  $ajuste \leftarrow \eta \frac{m_{corrigido}(n+1)}{\sqrt{v_{corrigido}(n+1)+\epsilon}}$
  $\mathbf{W}(n+1) \leftarrow \mathbf{W}(n) - ajuste$
fim

A Figura 11 mostra uma das possíveis amostragens para testar dois hiperparâmetros. Nessa figura são testadas combinações de valores para número de épocas e taxa de aprendizado. Cada ponto no gráfico possui um valor relacionado ao desempenho da rede. Esse valor pode ser atribuído com base na precisão da rede, por exemplo. Portanto, para escolher quais valores de número de épocas e taxa de aprendizado devem ser utilizados basta escolher a combinação que apresenta melhor precisão. Neste exemplo, foram escolhidos 0,005 para taxa de aprendizado e 3250 épocas, que corresponde ao menor valor da função custo atingido em regime em comparação aos demais pontos. Se os resultados ainda não foram satisfatórios, deve-se testar com novos valores de hiperparâmetros. Ademais, caso se queira testar um terceiro hiperparâmetro, basta adicionar uma dimensão ao gráfico.

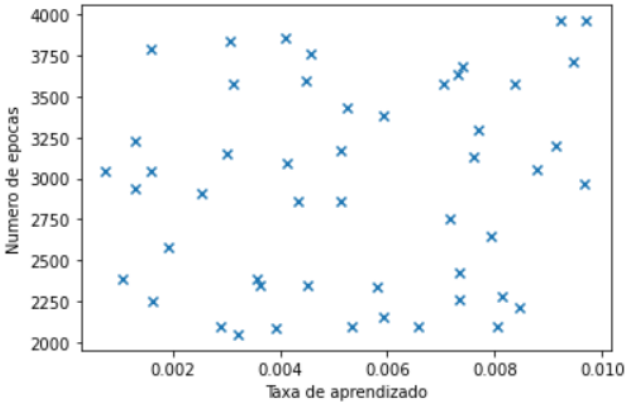


Figura 11: Amostras de hiperparâmetros de acordo com o *random search*

C Artigo de IC aceito no SBrT 2022

Treinamento distribuído de redes MLP para classificação de figuras geométricas

Lucca Gamballi, Daniel Gilio Tiglea, Renato Candido e Magno T. M. Silva

***Resumo*— Redes neurais perceptron multicamada são utilizadas para classificar figuras geométricas com uma abordagem distribuída. Os dados de treinamento são divididos entre redes que se comunicam por meio de uma determinada topologia, sem que haja compartilhamento direto desses dados entre as redes. Resultados de simulação indicam que o desempenho obtido com o treinamento distribuído e uma topologia adequada é semelhante ao observado com o treinamento clássico. Assim, garante-se a privacidade dos dados sem que ocorra perda no desempenho.**

***Palavras-Chave*— Redes neurais artificiais, perceptron multicamada, processamento distribuído, topologia, figuras geométricas.**

***Abstract*— Multilayer perceptron neural networks are used to classify geometric figures using a distributed approach. Training data are divided among neural networks that communicate through a certain topology, in which each network does not have access to the training data of the others. Simulation results indicate that the performance achieved with distributed training and a suitable topology is similar to that observed with classical training. Thus, data privacy is guaranteed without loss of performance.**

***Keywords*— Artificial neural networks, multi-layer perceptron, distributed processing, topology, geometric figures.**

I. Introdução

Na aprendizagem supervisionada, um modelo é treinado com dados de entrada que contêm rótulos conhecidos [1]. Usualmente, o processo de treinamento ocorre em apenas uma máquina. Contudo, essa abordagem centralizada pode não ser adequada em problemas de larga escala ou quando a privacidade é objeto de preocupação devido à presença de informações sensíveis. Nesses casos, o armazenamento e/ou processamento de todos os dados em uma única máquina pode ser problemático. Exemplos incluem aplicações médicas ou as que lidam com hábitos de vida dos usuários. Uma possível solução consiste em considerar uma abordagem distribuída, em que cada máquina treina um modelo local utilizando apenas uma parte dos dados [2]. Em uma segunda etapa, elas trocam informações acerca de seus modelos por meio de uma topologia pré-definida, sem compartilhar os dados de treinamento. Assim, pode-se obter um único modelo global utilizando todos os dados de treinamento sem que uma única máquina tenha acesso a todos eles. Neste trabalho, será considerado o treinamento distribuído de redes perceptron multicamada (*multilayer perceptron* – MLP) com base nos algoritmos *backpropagation* [3] e de consenso de [4]. Simulações mostram que, adotando-se uma topologia adequada, é possível preservar o desempenho da abordagem centralizada, conciliando com as vantagens do treinamento distribuído.

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, e-mails: lucca.gamballi@usp.br, {dtiglea, renatocan}@lps.usp.br e magno.silva@usp.br. Este trabalho foi financiado pelo CNPq (139071/2021-0), CAPES (88887.512247/2020-00 e Código de Financiamento 001) e FAPESP (2021/02063-6).

II. Treinamento distribuído de redes MLP

No treinamento distribuído de redes MLP, os N exemplos de treinamento são divididos entre $\mathcal{V}$ máquinas que se comunicam segundo uma determinada topologia. Cada máquina $i \in \{1, 2, \ldots, \mathcal{V}\}$ treina uma MLP com o conjunto de dados $\mathcal{D}_i$ de cardinalidade N_i, tal que $\sum_{i=1}^{\mathcal{V}} N_i = N$. Considerando um problema de classificação com C classes, é comum adotar a entropia cruzada categórica ponderada como função custo. Assim, a MLP da i-ésima máquina minimiza [5]

$$J_{\mathrm{ECCP}_i} = -\sum_{n_i=1}^{N_i}\sum_{\ell=1}^{C} p_{i,\ell}\; d_{i,\ell}(n_i)\;\ln[y_{i,\ell}(n_i)],$$

em que $y_{i,\ell}(n_i)$ é a ℓ-ésima saída da rede para o n_i-ésimo dado de treinamento com rótulo $d_{i,\ell}(n_i)$, $\ell = 1, 2, \ldots, C$ e $n_i = 1, 2, \ldots, N_i$. Classes minoritárias devem ser ponderadas com pesos $p_{i,\ell}$ maiores, o que é adequado para lidar com conjuntos de dados desbalanceados [5]. Inspirando-se na heurística de [6], considerou-se neste trabalho o peso $p_{i,\ell} = \lceil 10N_i/(CB_{i,\ell})\rceil$, em que $B_{i,\ell}$ é o número de amostras da classe ℓ em $\mathcal{D}_i$.

Utilizando-se o algoritmo de retropropagação do erro (*backpropagation*) para o treinamento das redes MLP [3], [7] e uma única iteração do algoritmo de consenso para a comunicação entre as máquinas a cada iteração m do *backpropagation*, obtém-se a seguinte regra de atualização da matriz de pesos da k-ésima camada da MLP da máquina i (os *biases* estão incluídos na matriz):

$$\mathbf{W}_i^{[k]}(m+1) = \sum_{j=1}^{\mathcal{V}} a_{ij}\left(\mathbf{W}_j^{[k]}(m) - \eta\frac{\partial J_{\mathrm{ECCP}_j}}{\partial \mathbf{W}_j^{[k]}(m)}\right),$$

em que η é o passo de adaptação e $a_{ij} \geq 0$ são pesos de combinação, tal que $a_{ij} = 0$ se as máquinas i e j não estão diretamente conectadas e $\sum_{j=1}^{\mathcal{V}} a_{ij} = 1$ para todo i, j. Existem diversas regras possíveis para a escolha desses pesos. Neste trabalho, adota-se a regra Metropolis [4], [8]. O caso particular em que $a_{ij} = 1$ se $i = j$ e $a_{ij} = 0$ caso contrário corresponde a um cenário de treinamento individual sem comunicação entre as máquinas. O fato da comunicação entre as máquinas ocorrer apenas quando os pesos da rede são atualizados é fundamental para se ter um custo computacional reduzido.

III. Banco de dados

O conjunto de dados utilizado neste artigo é constituído de imagens em preto e branco com dimensões 30×30 criadas aleatoriamente com auxílio da biblioteca PILLOW [9]. Cada imagem apresenta uma de três possíveis figuras geométricas: quadrados (Q), retângulos (R) ou circunferências (C).

A espessura do traço pode ser fina ou grossa, determinada aleatoriamente. Foram considerados dois cenários: um com classes balanceadas e outro com classes desbalanceadas, como mostrado na Tabela I.

TABELA I: Número de amostras para cada figura.

		Quadrado	Retângulo	Circunferência
Balanceado	Treino	5000	5000	5000
	Teste	5000	5000	5000
Desbalanceado	Treino	5940	689	2819
	Teste	6347	1390	2371

IV. Resultados de simulação

Foram considerados três tipos de treinamento: o centralizado, o individual e o distribuído. Nestes dois últimos, foram consideradas as topologias ilustradas na Fig. 1 [4]. Cada nó do grafo representa uma máquina onde é treinada uma MLP. As redes MLP foram implementadas usando a biblioteca Tensorflow [10] e são compostas por duas camadas ocultas com 32 e 16 neurônios cada e função de ativação ReLU. A camada de saída é composta de três neurônios com a função *Softmax*. Por fim, foi considerado o algoritmo de otimização Adam [11] com parâmetros $\beta_1 = 0,9$, $\beta_2 = 0,99$ e $\epsilon = 10^{-7}$. Os pesos foram inicializados de acordo com a inicialização Glorot [12] e foi considerado o *backpropagation* com passo de adaptação $\eta = 0,001$, tamanho de mini-batch $k = 2048$ e 1600 épocas de treinamento.

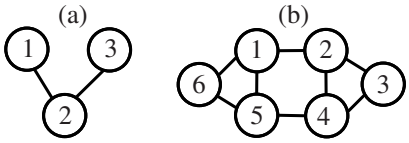


Fig. 1: Topologias com (a) 3 e (b) 6 máquinas.

Nas Tabelas II e III, são mostrados a sensibilidade (Se), precisão (P), F1-*Score* (F1) e coeficiente de correlação de Matthews (MCC) alcançados com cada abordagem para os dados de teste no cenário balanceado e desbalanceado, respectivamente. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, sendo que nos casos distribuído e individual, considerou-se a média das métricas das máquinas.

Analisando-se a Tabela II, nota-se que o treinamento centralizado apresenta as melhores métricas para todas as classes. O treinamento distribuído com a topologia da Fig. 1(a) apresenta métricas comparáveis às do caso centralizado para a classe C e um pouco inferiores para as classes Q e R. Comparando com o caso individual, o treinamento distribuído alcançou resultados superiores para a maior parte das métricas (22 de 24 métricas) considerando a mesma topologia. Por fim, os resultados obtidos com o treinamento distribuído para a topologia da Fig. 1(a) foram superiores aos da Fig. 1(b) para as classes Q e R e próximos para a classe C.

Para o caso desbalanceado (Tabela III) e topologia da Fig. 1(a), o treinamento individual forneceu os melhores resultados, com exceção das métricas da classe C e a precisão da classe R. Nota-se também neste caso que o treinamento distribuído com a topologia da Fig. 1(a) superou as métricas do centralizado, com exceção da sensibilidade para a classe Q e precisão para a classe R. O mesmo não é válido para a topologia da Fig. 1(b). Por fim, observa-se que o treinamento individual para a topologia da Fig. 1(b) obteve resultados piores que os da topologia da Fig. 1(a).

TABELA II: Métricas de desempenho para dados balanceados.

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**96,02**	89,40	81,85	88,34	82,45
	P(%)	99,36	89,30	90,81	**99,48**	95,31
	F1(%)	**97,66**	89,12	86,05	93,58	88,37
	MCC	**0,96**	0,84	0,79	0,90	0,83
R	Se(%)	99,34	90,44	89,26	**99,41**	94,56
	P(%)	**95,78**	89,90	79,69	86,82	79,74
	F1(%)	**97,3**	89,97	84,14	92,68	86,35
	MCC	**0,96**	0,85	0,77	0,90	0,81
C	Se(%)	99,90	99,92	99,52	99,94	**99,98**
	P(%)	**100,0**	99,96	99,12	**100,0**	**100,0**
	F1(%)	99,95	99,94	99,31	99,97	**99,99**
	MCC	**1,00**	**1,00**	0,99	**1,00**	**1,00**

TABELA III: Métricas de desempenho para dados desbalanceados.

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**93,77**	93,54	89,03	93,27	90,45
	P(%)	91,57	**99,80**	98,69	93,48	74,13
	F1(%)	92,66	**96,53**	93,60	93,36	77,39
	MCC	0,80	**0,90**	0,82	0,82	0,63
R	Se(%)	65,06	**98,02**	75,15	71,25	57,64
	P(%)	**71,65**	67,52	43,93	68,75	60,65
	F1(%)	68,20	**78,64**	57,72	69,63	47,27
	MCC	0,63	**0,78**	0,58	0,65	0,44
C	Se(%)	99,63	99,26	98,83	99,72	**99,95**
	P(%)	**100,0**	99,68	98,84	**100,0**	99,42
	F1(%)	99,81	99,47	98,84	**99,86**	99,69
	MCC	**1,00**	0,99	0,99	**1,00**	**1,00**

Apesar da abordagem distribuída apresentar um custo computacional total maior que o da centralizada devido ao algoritmo de consenso, esse custo é distribuído entre as máquinas. Assim, os requisitos quanto à capacidade computacional dos processadores das máquinas são menores no caso distribuído.

V. Conclusões

O treinamento distribuído de redes neurais é relativamente recente e tem grande potencial, pois visa resolver problemas de privacidade. Neste trabalho, constatou-se que o treinamento distribuído pode alcançar um desempenho comparável ao do caso centralizado desde que se considere uma topologia adequada. Deve-se tomar cuidado com topologias que contêm muitas máquinas, pois isso pode resultar em conjuntos de treinamento pequenos para cada rede neural, o que pode prejudicar o desempenho. O desempenho superior do treinamento individual em relação ao distribuído e ao centralizado, observado em alguns casos, será investigado posteriormente.

Referências

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, 2015.

[2] M. Fahimi and A. Ghasemi, "A distributed learning automata scheme for spectrum management in self-organized cognitive radio network," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 1490–1501, 2016.

[3] S. Haykin, *Neural netw. and learning machines*, Pearson, 3rd ed., 2009.

[4] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, pp. 1771–1778, 2019.

[5] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[6] G. King and L. Zeng, "Logistic regression in rare events data," *Political Analysis*, vol. 9, pp. 137–163, 2001.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[8] V. Schwarz, G. Hannak, and G. Matz, "On the convergence of average consensus with generalized metropolis-hasting weights," in *Proc. ICASSP*, 2014, pp. 5442–5446.

[9] A. Clark *et al.*, "Pillow (PIL fork) documentation," Disponível em `https://pillow.readthedocs.io/`, 2015.

[10] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. ICAIS*, PMLR, vol. 9, pp.249-256, 2010.

D Artigo completo aceito no SBrT 2022

Redes MLP distribuídas para classificação de arritmias cardíacas

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva

***Resumo*— Neste artigo, redes neurais perceptron multicamada são utilizadas para classificar arritmias cardíacas com uma abordagem distribuída. Os dados de treinamento são divididos entre redes que se comunicam por meio de uma determinada topologia, sem que uma rede tenha acesso aos dados de treinamento das outras. Essa abordagem é empregada para garantir a privacidade dos dados dos pacientes. Para obter um resultado clinicamente realista, dados de um mesmo paciente não são considerados concomitantemente nos conjuntos de treinamento e teste. Simulações indicam que o desempenho obtido com o treinamento distribuído e uma topologia adequada é semelhante ao observado com o treinamento clássico.**

***Palavras-Chave*— Redes neurais artificiais, perceptron multicamada, processamento distribuído, topologia, arritmias cardíacas.**

***Abstract*— In this paper, multilayer perceptron neural networks are used to classify cardiac arrhythmias with a distributed approach. The training data is split between networks that communicate through a certain topology, in which each network does not have access to the training data of the others. This approach is employed to ensure patient data privacy. To obtain a clinically realistic result, data from the same patient are not considered concurrently in the training and test sets. Simulations indicate that the performance obtained with distributed training and a suitable topology is similar to the one observed with the classical training.**

***Keywords*— Artificial neural networks, multi-layer perceptron, distributed processing, topology, cardiac arrhythmia.**

I. Introdução

A detecção e a classificação de arritmias cardíacas dependem de uma análise precisa da atividade elétrica do coração, registrada no sinal de eletrocardiograma (ECG) [1]. O sinal de ECG contém informações clínicas que permitem que essa tarefa seja realizada por especialistas, mas pode demandar muito tempo e é inviável quando se necessita de resultados imediatos. Diante disso, soluções baseadas em técnicas de aprendizado de máquina para a classificação automática de arritmias cardíacas têm sido amplamente propostas na literatura [1]–[6].

A principal abordagem em aprendizado de máquina é o treinamento supervisionado, em que um modelo é ajustado para mapear exemplos de entrada a rótulos pertencentes a um banco de dados de treinamento. Em geral, o treinamento é realizado em uma única máquina, o que pode não ser adequado quando se trabalha com uma grande quantidade de dados ou quando a privacidade é importante [7], [8]. Atualmente, a aplicação de técnicas de aprendizado de máquina na medicina, por exemplo, é dificultada pela disponibilidade limitada de conjuntos de dados para treinamento e teste dos algoritmos. Isso ocorre devido à ausência de registros médicos eletrônicos padronizados e devido aos requisitos legais e éticos estritos para proteger a privacidade do paciente. Para garantir a privacidade do paciente e, ao mesmo tempo, promover pesquisas científicas com grandes conjuntos de dados, a busca por soluções que atendem simultaneamente às demandas de proteção e utilização dos dados tem crescido [7]–[10].

Nesse contexto, o treinamento de redes neurais de forma distribuída tem despertado interesse recentemente na literatura [7], [8]. Na abordagem distribuída, cada máquina treina um modelo local utilizando apenas uma parte dos dados e troca informações acerca de seu modelo com máquinas vizinhas, sem compartilhar os dados de treinamento. Para comunicação entre as máquinas considera-se uma topologia pré-definida. Assim, pode-se obter um único modelo global utilizando todos os dados de treinamento sem que uma única máquina tenha acesso a todos eles [8].

Neste artigo, propõe-se o uso de redes neurais perceptron multicamada (*multilayer perceptron* – MLP) treinadas com uma abordagem distribuída para o diagnóstico automático de arritmias cardíacas. Foram considerados sinais de ECG do banco de dados *MIT-BIH Arrhythmia Database* (MITDB) [11], [12], levando-se em conta a separação dos dados dos pacientes nas fases de treinamento e teste. Conforme recomendações da *Association for the Advancement of Medical Instrumentation* (AAMI) [13], os conjuntos de dados de treinamento e de teste dos classificadores de arritmia não devem conter dados dos mesmos pacientes, já que na prática o sistema será utilizado em pacientes cujos dados não foram usados no treinamento.

O artigo está organizado da seguinte forma. Na Seção II, aborda-se o treinamento distribuído de redes MLP. Na Seção III, descrevem-se o banco de dados e a extração de características utilizados e as classes de arritmia consideradas. As Seções IV e V contêm respectivamente os resultados de simulação e as principais conclusões do trabalho.

II. Treinamento distribuído de redes MLP

Consideremos um problema de aprendizagem supervisionada com um conjunto de treinamento $\mathcal{D}$ de cardinalidade N. A ideia do treinamento distribuído de redes MLP consiste em dividir $\mathcal{D}$ em V subconjuntos disjuntos de cardinalidade N_i, denotados por $\mathcal{D}_i$, $i = 1,2,\ldots,V$, tal que $\sum_{i=1}^{V} N_i = N$. Os exemplos do subconjunto $\mathcal{D}_i$ são utilizados no treinamento da MLP da máquina i. As V máquinas consideradas se comunicam por meio de uma determinada topologia, como

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, e-mails: lucca.gamballi@usp.br, {dtiglea, renatocan}@lps.usp.br e magno.silva@usp.br. Este trabalho foi financiado pelo CNPq (139071/2021-0), CAPES (88887.512247/2020-00 e Código de Financiamento 001) e FAPESP (2021/02063-6).

a da Fig. 1. Nessa figura, cada máquina corresponde a um nó do grafo e as arestas indicam a existência de uma conexão entre determinados pares de máquinas. A *vizinhança* do nó i, denotada por $\mathcal{V}_i$, é o conjunto de nós com os quais ele consegue se comunicar diretamente (incluindo ele próprio).

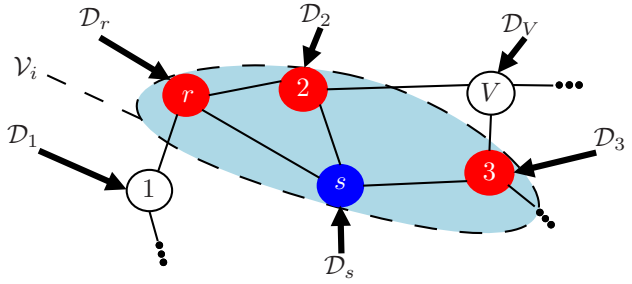


Fig. 1: Exemplo de uma rede de difusão, em que cada nó $i \in \{1,2,\ldots,V\}$ treina uma rede MLP com base no subconjunto de treinamento $\mathcal{D}_i$. Neste exemplo, a vizinhança do nó s é $\mathcal{V}_s = \{2,3,r,s\}$.

O objetivo de cada nó i é treinar uma rede MLP utilizando o subconjunto $\mathcal{D}_i$ a fim de minimizar uma função custo, cuja forma é comum a todos os nós do grafo. Considerando um problema de classificação com C classes, é comum adotar como função custo a entropia cruzada categórica ponderada. Assim, a MLP da i-ésima máquina minimiza [14]

$$J_{\text{ECCP}_i} = -\sum_{n_i=1}^{N_i}\sum_{\ell=1}^{C} p_{i,\ell}\; d_{i,\ell}(n_i)\;\ln[y_{i,\ell}(n_i)], \qquad (1)$$

em que $y_{i,\ell}(n_i)$ é a ℓ-ésima saída da rede para o n_i-ésimo dado de treinamento com rótulo $d_{i,\ell}(n_i)$, $\ell = 1,2,\ldots,C$ e $n_i = 1,2,\ldots,N_i$. Por fim, os pesos $p_{i,\ell}$ são utilizados em (1) para conferir uma ponderação maior a classes minoritárias [14]. Inspirando-se na heurística de [15], considerou-se neste trabalho o peso $p_{i,\ell} = \lceil 10N_i/(CB_{i,\ell})\rceil$, em que $B_{i,\ell}$ é o número de amostras da classe ℓ em $\mathcal{D}_i$.

A matriz de pesos (incluindo os *biases*) da k-ésima camada da MLP do nó i, denotada por $\mathbf{W}_i^{[k]}$, pode ser atualizada com o algoritmo de retropropagação do erro (*backpropagation*) [16], [17]. Para garantir a privacidade dos dados, cada nó i pode trocar informações com a sua vizinhança acerca de seu próprio modelo, mas não sobre seus dados de treinamento. Assim, a matriz $\mathbf{W}_i^{[k]}$ é compartilhada com os nós vizinhos após cada atualização. A ordem em que essas operações são realizadas dá origem a dois tipos de configuração: adaptação seguida da combinação (ATC – *adapt-then-combine*) e combinação seguida da adaptação (CTA – *combine-then-adapt*) [8]. Por simplicidade, neste artigo será considerada apenas a configuração ATC, embora resultados semelhantes possam ser obtidos com a configuração CTA. Assim, a regra para atualização de $\mathbf{W}_i^{[k]}$ pode ser descrita por [8]

$$\begin{cases} \boldsymbol{\Psi}_i^{[k]}(m+1) = \mathbf{W}_i^{[k]}(m) - \eta\dfrac{\partial J_{\text{ECCP}_i}}{\partial \mathbf{W}_i^{[k]}(m)} & \text{(2a)} \\ \mathbf{W}_i^{[k]}(m+1) = \displaystyle\sum_{j\in\mathcal{V}_i} a_{ij}\,\boldsymbol{\Psi}_i^{[k]}(m+1), & \text{(2b)} \end{cases}$$

em que $\boldsymbol{\Psi}_i^{[k]}$ é uma estimativa intermediária de mesma dimensão que $\mathbf{W}_i^{[k]}$, η é o passo de adaptação, m é a iteração do algoritmo e $a_{ij} \geq 0$ são pesos de combinação, tal que $a_{ij} = 0$ se as máquinas i e j não estão diretamente conectadas e $\sum_{j=1}^{V} a_{ij} = 1$ para todo i, j. Existem diversas regras possíveis para a escolha desses pesos. Neste trabalho, adota-se a regra Metropolis, dada por [7], [18]

$$a_{ij} = \begin{cases} \dfrac{1}{\max\{|\mathcal{V}_i|,|\mathcal{V}_j|\}}, & \text{se } j\in\mathcal{V}_i \text{ e } j\neq i \\ 1-\sum_{q\in\mathcal{V}_i} a_{iq}, & \text{se } j = i \\ 0, & \text{caso contrário} \end{cases}, \qquad (3)$$

em que $|\cdot|$ denota cardinalidade. O caso particular em que $a_{ij} = 1$ se $i = j$ e $a_{ij} = 0$ caso contrário corresponde a um cenário de treinamento individual, ou não-cooperativo, em que cada nó treina sua MLP com base em $\mathcal{D}_i$ sem interagir com os demais. Pode-se demonstrar que, adotando-se a abordagem cooperativa e inicializando-se as redes MLP distribuídas da mesma maneira, o treinamento distribuído leva a um único modelo global. Nesse caso, o desempenho como um todo da redes MLP distribuídas é equivalente ao obtido com uma única rede MLP treinada com o conjunto $\mathcal{D}$, sem que nenhuma máquina tenha acesso a todos os elementos desse conjunto simultaneamente [8].

III. Banco de dados e entrada da MLP

O banco de dados de arritmia do MIT contém 48 gravações de 30 min. de dois sinais de derivação de ECG, denotados como derivação A e derivação B [11]. Na maioria das gravações, a derivação A é uma derivação II modificada e a derivação B é a derivação precordial V1. Os dados são filtrados com um filtro passa-faixa 0,1–100 Hz e amostrados em 360 Hz. Seguindo o padrão da AAMI, as quatro gravações de pacientes com marcapasso foram retiradas da análise e as gravações restantes foram divididas em dois conjuntos de dados, denotados como DS1 e DS2 de acordo com a Tab. I. Cada conjunto de dados contém aproximadamente 5×10^4 batimentos cardíacos e tem uma mistura de arritmias de rotina ("100 series") e complexas ("200 series") [12]. Neste trabalho, o DS1 foi utilizado no treinamento e o DS2 no teste dos classificadores, conforme feito em [6], [19].

TABELA I: Esquema de divisão interpaciente de [6], [19].

DS1	101 106 108 109 112 114 115 116 118 119 122 124 201 203 205 207 208 209 215 220 223 230
DS2	100 103 105 111 113 117 121 123 200 202 210 212 213 214 219 221 222 228 231 232 233 234

Cada batimento cardíaco do MITDB contém anotações feitas por especialistas com as informações de diagnóstico. No total, existem 15 anotações de diagnóstico diferentes. A AAMI recomenda agrupar essas 15 anotações em cinco classes: batimentos do nó sinoatrial (N), batimentos ectópicos supraventriculares (S), batimentos ectópicos ventriculares (V), fusão de batimentos ectópicos normais e ventriculares (F) e batimentos desconhecidos ou com marcapasso (Q). Devido à quantidade reduzida de dados das classes Q e F, elas não foram consideradas neste trabalho. Portanto, o problema se reduz à classificação em três classes: N, S e V. Clinicamente, as classes S e V contêm batimentos cardíacos das arritmias mais graves. Portanto, muitos trabalhos têm focado na obtenção de classificadores capazes de distinguir essas classes [20]–[22].

O diagnóstico automático de arritmias cardíacas com redes neurais pode ser dividido em duas etapas: segmentação dos batimentos cardíacos e classificação. Em alguns trabalhos, uma etapa de pré-processamento também é incluída para reduzir ruídos e artefatos [1]. A etapa de segmentação fornece o delineamento do sinal de ECG com a determinação dos picos e limites das ondas QRS, P e T [23]. Já o classificador decide a classe do batimento com base em suas características. Para a etapa de segmentação, foi utilizado o algoritmo de Pan-Tompkins [24]. Na etapa de classificação, foram utilizadas redes MLP, implementadas com as bibliotecas Tensorflow e Keras [25]. Seguindo [6], [19], a etapa de pré-processamento para reduzir ruídos e artefatos não foi considerada aqui.

Para determinar a entrada da MLP, cada batimento cardíaco foi considerado como contendo 300 amostras do sinal de ECG, o que representa aproximadamente o número médio de amostras dos batimentos cardíacos contidos no MITDB. Este trecho é então centralizado utilizando a posição do pico R. Utilizando diferentes números de batimentos cardíacos na classificação, observou-se em [6] que as informações dos batimentos anteriores e posteriores contribuem para aumentar as métricas de classificação para a classe S. Além disso, as gravações da derivação A apresentam as métricas de classificação mais altas na maioria dos casos. Portanto, baseando-se nos resultados de [6], como entrada das redes MLP, foram considerados trechos de 900 amostras do sinal de ECG da derivação A, centralizado em torno do pico R do batimento cardíaco cuja classificação se deseja obter. Na Fig. 2, é mostrado um exemplo de entrada da rede MLP.

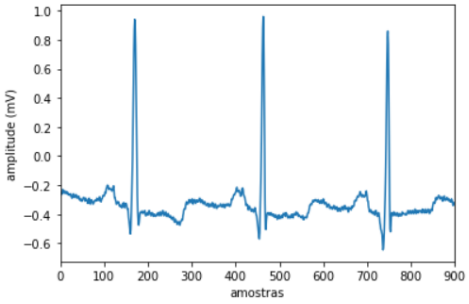


Fig. 2: Exemplo de entrada da rede MLP.

IV. Resultados de Simulação

Cada classe do MITDB possui um número diferente de exemplos, o que torna o uso desse conjunto de dados desafiador. Para lidar com esse desbalanceamento, considerou-se a entropia cruzada categórica ponderada como detalhado na Seção II. Além disso, foram consideradas duas situações: a situação S_1 em que apenas a classe N foi subamostrada e a a situação S_2 em que todas as classes foram subamostradas. Na Tab. II, são mostradas a quantidade de dados disponíveis no MITDB e a quantidade de dados usados neste trabalho em cada situação [6].

TABELA II: Total de batimentos do MITDB e quantidade de batimentos utilizados para os conjuntos de treinamento e teste.

Classe	Total de batimentos		Quantidade de batimentos utilizados			
			S_1		S_2	
	DS1	DS2	DS1	DS2	DS1	DS2
N	45784	44173	9448	10108	5940	6347
S	940	1837	940	1837	689	1390
V	3783	3218	3783	3218	2819	2371

Foram considerados três tipos de treinamento: o centralizado, o individual e o distribuído, sendo que para os dois últimos foram consideradas as topologias ilustradas na Fig. 3 [7]. Na Fig. 4, é apresentado um esquema de cada tipo de treinamento para a topologia da Fig. 3(a). Cada nó do grafo representa uma máquina onde é treinada uma MLP. As redes MLP consideradas são compostas por duas camadas ocultas com 32 e 16 neurônios cada e função de ativação ReLU. A camada de saída é composta de três neurônios com a função *Softmax*. Por fim, foi considerado o algoritmo de otimização Adam [26] com parâmetros $\beta_1 = 0{,}9$, $\beta_2 = 0{,}99$ e $\epsilon = 10^{-7}$. Os pesos foram inicializados de acordo com a inicialização Glorot [27] e foi considerado o *backpropagation* com passo de adaptação $\eta = 0{,}001$, tamanho de mini-batch $k = 2048$ e 1600 épocas de treinamento. Cabe observar que para comparar o desempenho dos treinamentos centralizado, individual e distribuído, a arquitetura da rede e os parâmetros do algoritmo de treinamento não influenciam, desde que sejam mantidos iguais em cada abordagem. Diante disso, foram escolhidos os parâmetros e as arquiteturas descritas anteriormente por apresentarem bons desempenhos de classificação [6].

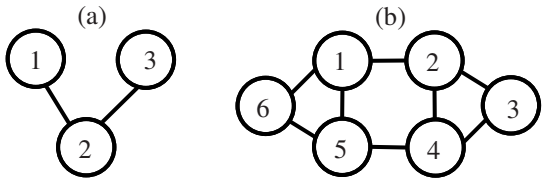


Fig. 3: Topologias com (a) 3 e (b) 6 máquinas.

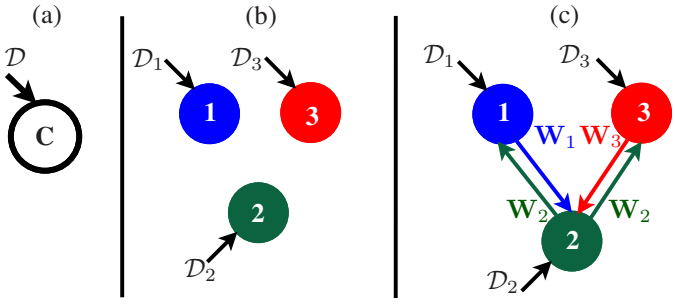


Fig. 4: Tipos de treinamento: (a) Centralizado, (b) Individual e (c) Distribuído, considerando a topologia da Fig. 3(a).

Na Tab. III, são mostrados para o cenário S_1 as seguintes métricas:

sensibilidade

$$\text{Se} = VP/(VP + FN),$$

precisão

$$\text{P} = VP/(VP + FP),$$

F1-*score*

$$\text{F1} = 2 \times \text{P} \times \text{Se}/(\text{P} + \text{Se}) \quad \text{e}$$

coeficiente de correlação de Matthews

$$\text{MCC} = \frac{VP \times VN - FP \times FN}{\sqrt{(VP+FP)(VP+FN)(VN+FP)(VN+FN)}}$$

alcançados com cada abordagem para os dados de teste, sendo VP, VN, FP e FN as quantidades de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, respectivamente. Cabe observar que $-1 \leq \mathrm{MCC} \leq 1$, sendo que são desejáveis resultados próximos de 1, em que a classificação é perfeita. Na Tab. IV, as mesmas medidas são mostradas para o cenário S_2. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, sendo que no caso individual, considerou-se a média das máquinas.

Analisando-se as Tabs. III e IV, nota-se que o treinamento individual resultou nas piores métricas tanto no caso S_1 como no caso S_2. Apesar disso, os resultados do treinamento individual são comparáveis aos dos treinamentos centralizado e distribuído no caso S_1 para a sensibilidade da classe N e para precisão, F1-*score* e MCC da classe V. Já no cenário S_2, o treinamento individual produziu desempenhos semelhantes aos observados para os treinamentos centralizado e distribuído apenas para a sensibilidade das classes N e V. Ainda assim, a sensibilidade para a classe N só é comparável aos demais tipos de treinamento considerando-se o treinamento individual com a topologia da Fig. 3(b). Em contrapartida, para a sensibilidade da classe V, apenas a topologia da Fig. 3(a) leva a um desempenho por parte do treinamento individual comparável aos treinamentos centralizado e distribuído. Cabe ressaltar que em ambos os cenários as métricas obtidas com o treinamento individual para a classe S foram significativamente inferiores àquelas alcançadas com os demais tipos de treinamento, qualquer que seja a topologia considerada.

Ainda com relação à Tab. III, nota-se que a diferença entre as métricas alcançadas com a média das máquinas no treinamento distribuído e o treinamento centralizado no cenário S_1 é inferior em módulo a 1,5% para a sensibilidade, precisão e F1-*score* de todas as classes, exceto para a precisão e o F1-*score* da classe V. Nesses últimos casos, as métricas obtidas com o treinamento centralizado foram superiores, sendo que o treinamento distribuído considerando-se a topologia da Fig. 3(a) apresentou desempenho mais próximo do treinamento centralizado do o caso em que se considera a topologia da Fig. 3(b). Analogamente, a diferença entre o MCC alcançado com o treinamento distribuído e aquele atingido com o treinamento centralizado foi de no máximo $0{,}02$ em módulo, exceto para a classe V considerando-se o treinamento distribuído com a topologia da Fig. 3(b). Nesse caso, a diferença foi de $0{,}04$, com o treinamento distribuído apresentando o melhor desempenho.

Realizando-se a mesma análise para o cenário S_2, nota-se que o treinamento distribuído com a topologia da Fig. 3(a) levou a resultados superiores aos do treinamento centralizado para todas as métricas, exceto para a precisão da classe N e para a sensibilidade das classes S e V. Por outro lado, nota-se que a adoção da topologia da Fig. 3(b) em vez daquela da Fig. 3(a) leva a um desempenho consideravelmente inferior para o treinamento distribuído, exceto para a sensibilidade da classe N e para as métricas da classe V. Em particular, nota-se uma degradação significativa para as métricas da classe S, com diferenças superiores a 30% para a sensibilidade, precisão e F1-*score* entre os treinamentos distribuídos considerando-se a topologia da Fig. 3(a) e aquela da Fig. 3(b). Em comparação com o treinamento centralizado, o treinamento distribuído com a topologia da Fig. 3(b) levou a desempenhos melhores apenas para a sensibilidade da classe N e para as métricas da classe V. Uma possível interpretação para isso reside no fato de que quando a rede de computadores apresenta muitas máquinas, a divisão do conjunto de treinamento $\mathcal{D}$ pode levar a subconjuntos $\mathcal{D}_i$ com cardinalidades muito pequenas. Nesse caso, a quantidade de exemplos a que cada máquina tem acesso pode ser insuficiente para a obtenção de um modelo adequado, o que não é mitigado pela etapa de combinação representada pela Eq. (2b). Cabe notar que a topologia da Fig. 3(b) apresenta o dobro do número de máquinas da topologia da Fig. 3(a). Além disso, a comparação entre os cenários S_1 e S_2 sugere que esse efeito é agravado pela subamostragem de todas as classes, o que implica ainda menos exemplos disponíveis para o treinamento de cada máquina quando se considera a topologia da Fig. 3(b) para o cenário S_2.

V. Conclusões

A classificação automática de arritmias utilizando métodos computadorizados é uma área com muito espaço para desenvolvimento. Por sua vez, o treinamento distribuído de redes neurais tem grande potencial para aplicações práticas, uma vez que visa resolver problemas de privacidade que constituem uma preocupação real em diversos tipos de situações. Neste trabalho, avaliaram-se os efeitos da adoção de treinamentos distribuídos para redes perceptron multicamadas. Constatou-se que com esse tipo de treinamento pode-se alcançar um desempenho comparável com a abordagem centralizada, em que se utiliza uma única máquina para treinar a rede neural com todos os dados do conjunto de treinamento. Nesse caso, pode-se conciliar o desempenho com a proteção à privacidade dos dados. Os resultados mostram que, para isso, é necessário adotar uma topologia adequada. Em particular, deve-se garantir que o número de máquinas não seja excessivamente elevado para que a divisão do conjunto de dados entre as máquinas não leve a subconjuntos de treinamento muito pequenos.

Referências

[1] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ECG analysis," *Biomedical Signal Processing and Control*, vol. 43, pp. 216–235, 2018.

[2] R. Banerjee, A. Ghose, and S. Khandelwal, "A novel recurrent neural network architecture for classification of atrial fibrillation using single-lead ECG," in *Proc. of 27th European Signal Processing Conference (EUSIPCO)*, Sep. 2019, pp. 1–5.

[3] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, pp. 65–69, Jan. 2019.

[4] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network," *Information Sciences*, vol. 405, pp. 81–90, Sep. 2017.

[5] U. R. Acharya, H. Fujita, O. S. Lih, M. Adam, J. H. Tan, and C. K. Chua, "Automated detection of coronary artery disease using different durations of ECG segments with convolutional neural network," *Knowledge-Based Systems*, vol. 132, pp. 62–71, Sep. 2017.

[6] N. Nagata, R. Candido, and M. T. M. Silva, "Combinações de redes neurais e discriminantes lineares para classificação de arritmias cardíacas," in *Anais do Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'21)*, Fortaleza, CE, 2021.

[7] B. Liu and Z. Ding, "Distributed heuristic adaptive neural networks with variance reduction in switching graphs," *IEEE Transactions on Cybernetics*, vol. 51, pp. 3836–3844, 2019.

TABELA III: Métricas de desempenho para o caso S_1.

		Central	Individual		Distribuído										
			Topologia (a)	Topologia (b)	Topologia (a)				Topologia (b)						
					Nó 1	Nó 2	Nó 3	Média	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Média
N	Se(%)	85,68	88,22	87,67	86,62	83,97	87,08	85,89	83,59	**91,95**	83,52	83,13	83,68	83,06	84,82
	P(%)	90,99	81,43	80,96	90,45	91,18	91,42	91,02	91,94	90,53	92,03	91,20	**92,30**	90,81	91,47
	F1(%)	88,26	84,68	84,18	86,90	87,43	**89,20**	87,84	87,57	86,02	87,56	86,98	87,78	86,76	87,11
	MCC	0,67	0,51	0,49	0,65	0,65	**0,69**	0,66	0,67	0,64	0,65	0,65	0,65	0,66	0,65
S	Se(%)	66,63	9,05	5,67	66,72	65,61	67,13	66,49	71,58	72,62	67,66	71,08	**73,84**	70,20	71,16
	P(%)	64,69	29,63	22,55	64,86	63,58	**69,89**	66,11	61,11	64,89	58,91	64,71	63,00	66,67	63,22
	F1(%)	65,65	13,82	9,01	65,78	64,58	68,48	66,28	65,93	**68,54**	62,98	67,74	67,99	68,39	66,93
	MCC	0,61	0,10	0,05	0,61	0,59	**0,64**	0,61	0,61	**0,64**	0,58	0,63	**0,64**	**0,64**	0,62
V	Se(%)	90,18	86,46	86,62	**90,87**	90,32	89,62	90,27	90,37	88,10	89,44	88,63	88,13	89,38	89,01
	P(%)	**77,34**	76,72	75,21	75,24	72,23	76,61	74,76	76,79	72,66	72,12	72,33	72,13	73,73	73,29
	F1(%)	83,27	81,42	80,49	**85,32**	80,27	82,72	81,77	83,02	79,64	79,85	79,65	79,33	80,80	80,38
	MCC	**0,79**	0,76	0,75	0,77	0,75	0,78	0,77	0,78	0,74	0,75	0,74	0,74	0,75	0,75

TABELA IV: Métricas de desempenho para o caso S_2.

		Central	Individual		Distribuído										
			Topologia (a)	Topologia (b)	Topologia (a)				Topologia (b)						
					Nó 1	Nó 2	Nó 3	Média	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Média
N	Se(%)	84,04	83,67	85,56	88,45	88,94	**89,20**	88,86	82,93	84,01	87,96	82,51	86,44	86,75	85,10
	P(%)	**90,01**	78,39	77,72	89,67	88,77	89,80	89,41	83,09	82,99	83,08	82,12	82,78	83,79	82,97
	F1(%)	86,92	80,93	81,43	89,05	88,85	**89,50**	89,13	83,01	83,5	85,45	82,31	84,57	85,25	84,02
	MCC	0,67	0,46	0,46	0,71	0,70	**0,72**	0,71	0,53	0,55	0,59	0,53	0,57	0,60	0,56
S	Se(%)	**67,27**	6,84	6,28	63,56	65,01	67,17	65,25	33,91	33,33	32,20	33,04	31,69	33,18	32,89
	P(%)	67,46	22,80	21,06	72,89	74,14	**75,79**	74,27	49,69	52,78	58,46	44,91	55,00	51,39	52,04
	F1(%)	67,36	5,25	9,38	67,90	69,28	**71,22**	69,47	40,31	40,86	41,53	38,07	40,21	40,33	40,22
	MCC	0,62	0,05	0,04	0,63	0,65	**0,67**	0,65	0,34	0,35	0,37	0,31	0,35	0,34	0,34
V	Se(%)	90,30	89,34	86,02	91,03	88,25	89,51	89,60	**91,58**	89,25	88,58	89,41	90,24	90,05	89,85
	P(%)	76,57	72,76	74,95	81,80	**82,77**	82,52	82,36	76,32	75,80	79,68	78,74	78,62	81,72	78,48
	F1(%)	82,87	80,14	80,07	**86,17**	85,42	85,87	85,82	83,25	81,97	83,89	83,74	84,03	85,68	83,76
	MCC	0,78	0,74	0,73	**0,82**	0,81	0,81	0,81	0,78	0,76	0,79	0,78	0,79	0,81	0,79

[8] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Transations on Neural Networks and Learning Systems*, vol. 31, pp. 1771–1778, May 2020.

[9] G.A. Kaissis, M.R. Makowski, D. Rückert, and R.F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nat Mach Intell*, vol. 2, pp. 305–311, 2020.

[10] R. Torkzadehmahani, R. Nasirigerdeh, D. B. Blumenthal, T. Kacprowski, M. List, J. Matschinske, J. Spaeth, N. K. Wenke, and J. Baumbach, "Privacy-preserving artificial intelligence techniques in biomedicine," *Methods of Information in Medicine*, 2022.

[11] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[12] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[13] Association for the Advancement of Medical Instrumentation et al., "ANSI/AAMI EC57:1998/(R)2008 - Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," *American National Standards Institute, Arlington, VA, USA*, 2008, Association for the Advancement of Medical Instrumentation (AAMI), ANSI/AAMI/ISO EC57,1998-(R)2008, 2008.

[14] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[15] G. King and L. Zeng, "Logistic regression in rare events data," *Political Analysis*, vol. 9, pp. 137–163, 2001.

[16] S. Haykin, *Neural networks and learning machines*, Pearson Education, Upper Saddle River, 3 edition, 2009.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[18] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[19] P. de Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[20] E. J. S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey," *Computer methods and programs in biomedicine*, vol. 127, pp. 144–164, 2016.

[21] Z. Luo, L. Liu, J. Yin, Y. Li, and Z. Wu, "Deep learning of graphs with ngram convolutional neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2125–2139, 2017.

[22] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ECG classification by 1-D convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2015.

[23] J. P. Martínez, R. Almeida, S. Olmos, A. P. Rocha, and P. Laguna, "A wavelet based ECG delineator: evaluation on standard databases," vol. 51, no. 4, pp. 570–581, 2004.

[24] P. S. Hamilton and W. J. Tompkins, "Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database," *IEEE Transactions on Biomedical Engineering*, vol. BME-33, no. 12, pp. 1157–1165, Dec 1986.

[25] M. Abadi et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Yee Whye Teh and Mike Titterington, Eds., Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010, vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256, PMLR.

Referências

[1] Hannun, Awni Y, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia e Andrew Y Ng: *Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.* Nature medicine, 25(1):65–69, 2019.

[2] Guglin, Maya E e Deepak Thatai: *Common errors in computer electrocardiogram interpretation.* International journal of cardiology, 106(2):232–237, 2006.

[3] Shah, Atman P e Stanley A Rubin: *Errors in the computerized electrocardiogram interpretation of cardiac rhythm.* Journal of electrocardiology, 40(5):385–390, 2007.

[4] Liu, Bo e Zhengtao Ding: *Distributed heuristic adaptive neural networks with variance reduction in switching graphs.* IEEE Transactions on Cybernetics, 51(7):3836–3844, 2019.

[5] Liu, B., Z. Ding e C. Lv: *Distributed Training for Multi-Layer Neural Networks by Consensus.* IEEE Transations on Neural Networks and Learning Systems, 31(5):1771–1778, May 2020.

[6] Kaissis, G.A., M.R. Makowski, D. Rückert e R.F. Braren: *Secure, privacy-preserving and federated machine learning in medical imaging.* Nat Mach Intell, 2:305–311, 2020.

[7] Torkzadehmahani, Reihaneh, Reza Nasirigerdeh, David B. Blumenthal, Tim Kacprowski, Markus List, Julian Matschinske, Julian Spaeth, Nina Kerstin Wenke e Jan Baumbach: *Privacy-Preserving Artificial Intelligence Techniques in Biomedicine.* Methods of Information in Medicine, 2022.

[8] Berkaya, Selcan Kaplan, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal e M Bilginer Gulmezoglu: *A survey on ECG analysis.* Biomedical Signal Processing and Control, 43:216–235, 2018.

[9] Regis, Carlos Danilo Miranda, Luiz Guedes Caldeira e Edmar Candeia Gurjão: *Avaliação da amostragem compressiva em sinais de ECG e imagens digitais.* Revista Principia-Divulgação Científica e Tecnológica do IFPB, 1:95.

[10] Sayed, Ali H *et al.*: *Adaptation, learning, and optimization over networks.* Foundations and Trends® in Machine Learning, 7(4-5):311–801, 2014.

[11] Cattivelli, Federico S. e Ali H. Sayed: *Diffusion LMS Strategies for Distributed Estimation.* IEEE Transactions on Signal Processing, 58(3):1035–1048, 2010.

[12] Fernandez-Bes, Jesus, Jeronimo Arenas-Garcia, Magno TM Silva e Luis A Azpicueta-Ruiz: *Adaptive diffusion schemes for heterogeneous networks.* IEEE Transactions on Signal Processing, 65(21):5661–5674, 2017.

[13] Li, Mu, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita e Bor Yiing Su: *Scaling distributed machine learning with the parameter server.* Em *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, páginas 583–598, 2014.

[14] Suresh, Ananda Theertha, X Yu Felix, Sanjiv Kumar e H Brendan McMahan: *Distributed mean estimation with limited communication.* Em *International Conference on Machine Learning*, páginas 3329–3337. PMLR, 2017.

[15] Olfati-Saber, Reza e Richard M Murray: *Consensus problems in networks of agents with switching topology and time-delays.* IEEE Transactions on automatic control, 49(9):1520–1533, 2004.

[16] Blondel, Vincent D, Julien M Hendrickx, Alex Olshevsky e John N Tsitsiklis: *Convergence in multiagent coordination, consensus, and flocking.* Em *Proceedings of the 44th IEEE Conference on Decision and Control*, páginas 2996–3000. IEEE, 2005.

[17] Xiao, Lin e Stephen Boyd: *Fast linear iterations for distributed averaging.* Systems & Control Letters, 53(1):65–78, 2004.

[18] Xiao, Lin e Stephen Boyd: *Fast linear iterations for distributed averaging.* Systems & Control Letters, 53(1):65–78, 2004, ISSN 0167-6911. `https://www.sciencedirect.com/science/article/pii/S0167691104000398`.

[19] Takahashi, Noriyuki, Isao Yamada e Ali H Sayed: *Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis.* IEEE Transactions on Signal Processing, 58(9):4795–4810, 2010.

[20] Tu, Sheng Yuan e Ali H Sayed: *Optimal combination rules for adaptation and learning over networks.* Em *2011 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, páginas 317–320. IEEE, 2011.

[21] Ho, Y. e S. Wookey: *The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling.* IEEE Access, 8:4806–4813, 2020, ISSN 2169-3536. `http://dx.doi.org/10.1109/ACCESS.2019.2962617`.

[22] King, Gary e Langche Zeng: *Logistic regression in rare events data.* Political analysis, 9(2):137–163, 2001.

[23] Haykin, Simon: *Neural networks and learning machines.* Pearson Education, Upper Saddle River, 3ª edição, 2009.

[24] Goodfellow, I., Y. Bengio e A. Courville: *Deep Learning.* MIT Press, 2016.

[25] Clark, Alex: *Pillow (PIL Fork) Documentation*, 2015.

[26] Moody, G. B. e R. G. Mark: *The impact of the MIT-BIH arrhythmia database.* IEEE Engineering in Medicine and Biology Magazine, 20(3):45–50, 2001.

[27] Goldberger, Ary L, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung Kang Peng e H Eugene Stanley: *PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.* circulation, 101(23):e215–e220, 2000.

[28] Niwas, S. I., R. S. S. Kumari e V. Sadasivam: *Artificial neural network based automatic cardiac abnormalities classification.* Em *Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05)*, páginas 41–46. IEEE, 2005.

[29] Ozbay, Yuksel e Bekir Karlik: *A recognition of ECG arrhytihemias using artificial neural networks.* Em *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, páginas 1680–1683. IEEE, 2001.

[30] Raut, Ranjana D e Sanjay V Dudul: *Arrhythmias classification with MLP neural network and statistical analysis.* Em *2008 First International Conference on Emerging Trends in Engineering and Technology*, páginas 553–558. IEEE, 2008.

[31] Geselowitz, David B: *On the theory of the electrocardiogram.* Proceedings of the IEEE, 77(6):857–876, 1989.

[32] Hamilton, Patrick S e Willis J Tompkins: *Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database.* IEEE transactions on biomedical engineering, (12):1157–1165, 1986.

[33] De Chazal, Philip, Maria O'Dwyer e Richard B Reilly: *Automatic classification of heartbeats using ECG morphology and heartbeat interval features.* IEEE transactions on biomedical engineering, 51(7):1196–1206, 2004.

[34] Nagata, N, R. Candido e M. T. M. Silva: *Combinações de redes neurais e discriminantes lineares para classificação de arritmias cardíacas*. Em *Anais do Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT'21)*, Fortaleza, CE, 2021.

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

[36] Rosenblatt, Frank: *Perceptron simulation experiments*. Proceedings of the IRE, 48(3):301–309, 1960.

[37] Bottou, Léon: *Large-scale machine learning with stochastic gradient descent*. Em *Proceedings of COMPSTAT'2010*, páginas 177–186. Springer, 2010.

[38] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning*. MIT Press, 2016.

[39] Bengio, Yoshua: *Practical recommendations for gradient-based training of deep architectures*. Em *Neural networks: Tricks of the trade*, páginas 437–478. Springer, 2012.

[40] McClelland, James L e David E Rumelhart: *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. MIT press, 1989.

[41] Thimm, G. e E. Fiesler: *High-order and multilayer perceptron initialization*. IEEE Transactions on Neural Networks, 8(2):349–359, 1997.

[42] Fahlman, Scott E *et al.*: *An empirical study of learning speed in back-propagation networks*. Carnegie Mellon University, Computer Science Department Pittsburgh, PA, USA, 1988.

[43] LeCun, Yann, Léon Bottou, Yoshua Bengio e Patrick Haffner: *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[44] Boers, Egbert JW e Herman Kuiper: *Biological metaphors and the design of modular artificial neural networks*. 1992.

[45] He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. Em *Proceedings of the IEEE international conference on computer vision*, páginas 1026–1034, 2015.

[46] Glorot, Xavier e Yoshua Bengio: *Understanding the difficulty of training deep feedforward neural networks*. Em Teh, Yee Whye e Mike Titterington (editores): *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 de *Proceedings of Machine Learning Research*, páginas 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[47] Haykin, Simon S.: *Neural networks and learning machines*, capítulo 4.4. Pearson Education, terceira edição, 2009.

[48] Itano, Fernando, Miguel Angelo de Abreu de Sousa e Emilio Del-Moral-Hernandez: *Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm*. Em *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–8, 2018.

[49] Duchi, John, Elad Hazan e Yoram Singer: *Adaptive subgradient methods for online learning and stochastic optimization.* Journal of machine learning research, 12(7), 2011.

[50] Tieleman, Tijmen, Geoffrey Hinton *et al.*: *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural networks for machine learning, 4(2):26–31, 2012.

[51] Kingma, Diederik P e Jimmy Ba: *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014.