



Eduardo Barreto Brito

Neurônios artificiais quânticos aplicados na área de saúde



Universidade Federal de Pernambuco
graduacao@cin.ufpe.br
www.cin.ufpe.br/~graduacao

Recife
24 de agosto de 2021

Eduardo Barreto Brito

Neurônios artificiais quânticos aplicados na área de saúde

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: *Computação Quântica*

Orientador: *Fernando Maciano de Paula Neto*

Recife

24 de agosto de 2021

Dedico essa monografia primeiramente à Patrícia e Wilson, meus pais, que me deram todo o suporte necessário para que eu pudesse focar no projeto e escrita do mesmo, além do meu melhor amigo, conselheiro, revisor de texto e irmão, Hartur, que me serviu de guia, inspiração e me manteve motivado quanto à meu curso desde o início. À minha amiga Juliana, que me serviu de apoio em momentos que eu mais precisei e, por último, mas não menos importante, ao professor, orientador, e amigo, Fernando Neto, que teve paciência de me ensinar do zero as coisas e fez com que essa etapa fosse concluída da forma mais tranquila possível.

AGRADECIMENTOS

Gostaria de agradecer ao Prof. orientador e amigo Fernando Maciano de Paula Neto, que topou me ajudar com o projeto desde o início, me guiando sempre que necessário, além de ter tido paciência para me ensinar toda a carga necessária para a realização da pesquisa.

“Ohana quer dizer família, e família significa nunca abandonar ou esquecer”

–Stitch

ABSTRACT

With the advance in the quantum computing, more and more classical algorithms have been implemented or improved with quantum circuits. That said, one of the areas that are being explored is the machine learning and, more specifically, neural networks. Investigating this possibility, Tacchino et al. and Mangini et al. suggested and implemented an artificial quantum neuron based on Rosenblatt's classical perceptron model. They had an hybrid approach, using classical computation to update the input weights. Besides them, Monteiro et al. suggested an alternate way of pre processing the data in order to use it in Mangini et al.'s model. This project validates the efficiency of both models, evaluating it on health related datasets: "Breast Tissue" and "Caesarian Section Classification Dataset", both from UCI Datasets. When compared to MLP and Perceptron algorithms, the quantum neuron had good results when dealing with easily discretizable datasets, since it managed to have accuracy rate close to the classic models and F1-Scores at least as good as the classical algorithms. However, when dealing with a more irregular dataset, its results were worse than the classical algorithms, indicating a possible inefficiency in such cases.

Keywords: Artificial Neuron. Artificial Intelligence. Quantum Computing. Health.

RESUMO

Com o avanço da computação quântica, mais algoritmos da computação clássica vem sendo implementados ou aprimorados com a utilização de circuitos quânticos. Assim, uma área em exploração é a de aprendizagem de máquina e, mais especificamente, a área de redes neurais. Nesta linha, Tacchino et al. e Mangini et al. implementaram um neurônio artificial quântico baseado no modelo clássico do *perceptron* de Rosenblatt em conjunto com um algoritmo da computação clássica para atualizar os pesos das entradas. Além deles, Monteiro et al. apresentaram uma forma de pré-processamento dos dados para o modelo de Mangini et al. Este trabalho valida a eficiência dos modelos de Tacchino et al. e Mangini et al. e os avalia em banco de dados reais na área de saúde: “Breast Tissue” e “Caesarian Section Classification Dataset”, ambos disponíveis no *UCI Datasets*. Ao serem comparados com os algoritmos de MLP e *perceptron* clássicos, os modelos apresentam robustez quando lidando com problemas facilmente discretizáveis, resultando em acurácia próxima e F1-Scores tão bons quanto ou melhor que os algoritmos mais conhecidos. Já ao lidar com um banco de dados mais irregular, os modelos apresentaram resultados inferiores aos dos modelos clássicos, o que indica uma possível deficiência nesses casos.

Palavras-chave: Neurônio Artificial. Inteligência Artificial . Computação Quântica. Saúde.

LISTA DE FIGURAS

Figura 1	– Portas quânticas, suas representações nos circuitos e respectivas matrizes. Figura retirada de Yan et al. (2014)	16
Figura 2	– Portas quânticas, suas definições matriciais e suas aplicações em <i>Qubits</i> .	17
Figura 3	– Resultado da execução do Algoritmo 2.1.	18
Figura 4	– Resultado da execução do Algoritmo 2.2.	18
Figura 5	– Modelo do neurônio quântico proposto por Tacchino et al. (2019) . U_i é a parte do modelo responsável por carregar as entradas e U_w é a parte do modelo responsável por carregar os pesos. Figura retirada de Tacchino et al. (2019)	21
Figura 6	– Modelo geral do neurônio quântico utilizado por Tacchino et al. (2019) e Mangini et al. (2020) . Os vetores de entrada e de pesos são carregados seguindo um dos modelos descritos nas Seções 3.1, 3.2 e 3.3.	21
Figura 7	– Resultados obtidos ao fim do treinamento do modelo proposto por Tacchino et al. (2019) . A imagem alvo está no canto superior esquerdo. Acima de cada imagem além da imagem alvo está o grau de semelhança obtido através de álgebra linear padrão (<i>exact</i>) e o grau de semelhança apontado pelo neurônio quântico (<i>q. alg.</i>). Figura retirada de Tacchino et al. (2019)	22
Figura 8	– Resultado obtido ao fim da execução do modelo proposto por Mangini et al. (2020) . Nela, o θ_{target} é a imagem alvo, θ_{start} é a representação gráfica dos pesos iniciais e θ_{final} é a representação gráfica dos pesos ao fim do treinamento. Figura retirada de Mangini et al. (2020)	22
Figura 9	– Exemplo de um circuito quântico com entrada $v_i = [1, -1, 1, -1]$ e pesos $w_i = [1, 1, -1, 1]$, construído utilizando o modelo de Força Bruta proposto por Tacchino et al. (2019)	23
Figura 10	– Exemplo de um circuito quântico com entrada $v_i = [1, -1, 1, -1]$ e pesos $w_i = [1, 1, -1, 1]$, construído utilizando o modelo de HSGS proposto por Tacchino et al. (2019)	25
Figura 11	– Exemplo de um circuito quântico com entrada $v_i = [1.5, 1.3]$ e pesos $w_i = [0.2, 0.4]$, construído utilizando o modelo proposto por Mangini et al. (2020)	26
Figura 12	– Representação do neurônio aplicado no banco de dados “Cesarian Section Classification Dataset”.	31

Figura 13 – Representação do neurônio aplicado no banco de dados “Breast Tissue Dataset”, utilizando a abordagem de neurônio único. A ordem de cada classe é definido automaticamente durante o treinamento, dependendo da exposição de instâncias daquela classe.	32
Figura 14 – Representação do neurônio aplicado no banco de dados “Breast Tissue Dataset”, utilizando a abordagem de múltiplos neurônios.	32

LISTA DE TABELAS

Tabela 1	– Acurácia média do experimento realizado por Grant et al. no banco de dados Iris, com o <i>Tree Tensor Network</i> . Foi utilizado apenas portas simples, com parâmetros reais e a tarefa de classificação foi dividida entre 3: reconhecer entre as classes 1 ou 2, entre as classes 2 ou 3 e entre as classes 1 ou 3. Resultados retirados de Grant et al. (2018)	19
Tabela 2	– Resultado do experimento realizado por Grant et al. no banco de dados MNIST. Foram utilizados os classificadores TTN, Mera, híbrido ou logística. Em relação às portas, foram utilizadas portas simples ou genéricas, com parâmetros reais ou complexos. Nesse experimento da base, foi considerado 4 cenários de classificação: reconhecer dígitos maiores que 4 ou menores e iguais a 4, dígitos pares, 0 ou 1 e 2 ou 7. Resultados retirados de Grant et al. (2018)	20
Tabela 3	– Variação dos hiperparâmetros utilizados nos experimentos de cada modelo de neurônio quântico e abordagem, para os modelos de HSGS e <i>phase-encoding</i>	33
Tabela 4	– Variação dos hiperparâmetros utilizados nos experimentos para o modelo de neurônio quântico de Força Bruta e cada abordagem.	34
Tabela 5	– Tempo necessário para executar as simulações variando os hiperparâmetros definidos nas Tabelas 3 e 4.	34
Tabela 6	– Média dos resultados obtidos com a variação de hiperparâmetros descrita na Subseção 4.2.3 por abordagem quanto ao banco de dados “Caesarian Section Classification”.	36
Tabela 7	– Média dos resultados obtidos com a variação de hiperparâmetros descrita na Subseção 4.2.3 por abordagem quanto ao banco de dados “Breast Tissue”.	37
Tabela 8	– Métricas das configurações que resultaram em melhores F1-Scores de diferentes abordagens quanto ao banco de dados “Caesarian Section Classification”.	38
Tabela 9	– Métricas das configurações que resultaram em melhores F1-Scores de diferentes abordagens quanto ao banco de dados “Breast Tissue”.	39

LISTA DE ALGORITMOS

Algoritmo 2.1 – Algoritmo em Python utilizando Qiskit para construção e exibição de um circuito quântico.	17
Algoritmo 2.2 – Algoritmo em Python utilizando Qiskit para construção e simulação de um circuito quântico.	18
Algoritmo 3.1 – Pseudocódigo para o mapeamento do neurônio quântico utilizando força bruta.	23
Algoritmo 3.2 – Pseudocódigo para o mapeamento do neurônio quântico utilizando HSGS.	25
Algoritmo 3.3 – Pseudocódigo para o mapeamento do neurônio quântico utilizando phase encoding.	26

SUMÁRIO

1	INTRODUÇÃO	12
1.1	MOTIVAÇÃO	12
1.2	OBJETIVOS	14
1.2.1	Objetivos Gerais	14
1.2.2	Objetivos Específicos	14
2	COMPUTAÇÃO QUÂNTICA	15
2.1	CONCEITOS BÁSICOS	15
2.1.1	Características das Portas Quânticas	16
2.2	QISKIT	16
3	APRENDIZAGEM DE MÁQUINA QUÂNTICA	19
3.1	FORÇA BRUTA	22
3.2	HYPERGRAPH STATES GENERATION SUBROUTINE (HSGS)	24
3.3	NEURÔNIO DE VALORES CONTÍNUOS	26
3.4	PRÉ-PROCESSAMENTO DE DADOS	27
4	EXPERIMENTOS	29
4.1	VISÃO GERAL	29
4.2	ESPECIFICAÇÃO DO ALGORITMO	30
4.2.1	Característica dos bancos de dados	30
4.2.2	Estrutura do Algoritmo	31
4.2.3	Exploração de Hiperparâmetros	32
<i>4.2.3.1</i>	<i>Algoritmos Clássicos</i>	34
4.2.4	Bibliotecas Utilizadas	35
4.3	RESULTADOS	35
4.4	CAESARIAN DATASET	37
4.5	BREAST-TISSUE DATASET	38
5	CONCLUSÃO	40
5.1	TRABALHOS FUTUROS	40
	REFERÊNCIAS	42

1

INTRODUÇÃO

1.1 MOTIVAÇÃO

A tarefa de aprender um padrão através de dados é chamada de aprendizagem de máquina (Schuld *et al.* (2014)). Essa tarefa tem se tornado cada vez mais importante na era digital, sendo utilizado cada vez mais em atividades no dia a dia, como reconhecimento de íris em sistema de segurança, reconhecimento de e-mails que são spam, avaliação de padrão de consumo entre outros (Schuld *et al.* (2014)). Com o maior número de dados a serem processados, a busca por métodos inovadores ou mais eficientes vem sendo cada vez maior (Schuld *et al.* (2015)). Sendo assim, com a promessa de uma computação exponencialmente mais eficiente para alguns problemas, a computação quântica é estudada para otimizar os algoritmos de aprendizagem de máquina clássicos ((Schuld *et al.* (2015)).

A computação quântica se baseia no comportamento dos átomos para realizar cálculos (Portugal & Marquezino (2019)) e traz vantagens em relação à computação clássica quanto a alguns problemas. Uma das vantagens é retratada por Shor (1994), que propõe um algoritmo para resolver a fatoração de números inteiros em tempo polinomial em relação ao tamanho da entrada. Esse algoritmo, conhecido como “Algoritmo de Shor”, apresenta uma vantagem em relação a computação clássica pelo fato de que não se é conhecido um algoritmo que resolve o problema de fatoração de números inteiros em tempo mais rápido que exponencial em relação ao tamanho da entrada até o momento. Além de Shor, Arute *et al.* (2019) também estudam essa vantagem simulando instâncias aleatórias de circuitos quânticos 1 milhão de vezes - tarefa que Arute *et al.* estimam demorar cerca de 10000 anos em um supercomputador do estado da arte da época - demorando cerca de 200 segundos no processador quântico *Sycamore*.

Uma das diferenças que a computação quântica traz em relação a computação clássica está na unidade básica de processamento: na computação clássica, essa unidade é o *bit* e tem seu valor 0 ou 1 (Portugal & Marquezino (2019)). Já na computação quântica, a unidade é o *Qubit*, que é representado por um vetor bidimensional de norma 1, permitindo que essa unidade admita a coexistência entre 0 e 1 (Portugal & Marquezino (2019)). Em relação aos algoritmos de aprendizagem de máquina clássica, Schuld *et al.* explicam que essa possibilidade de estar em um estado intermediário permitiria um possível aumento de velocidade em termos de complexidade,

uma vez que as operações poderiam ser feitas em mais de um estado da base ao mesmo tempo (Schuld *et al.* (2015)). Por outro lado, a computação quântica tem a limitação de que seus operadores devem ser unitários, uma vez que a soma das probabilidades do sistema ser 0 ou 1 deve resultar em 1 (Portugal & Marquezino (2019)).

Redes neurais é um algoritmo de aprendizagem de máquina que já foi bastante estudado e melhorado, sendo provado a sua eficiência em problemas reais (de Pádua Braga *et al.* (2007)). A adaptação das redes neurais para a computação quântica é tratada por Schuld *et al.* (2014), que trazem a dificuldade de exploração da não linearidade na adaptação dos algoritmos de aprendizagem de máquina clássicos para a computação quântica devido à limitação dos operadores unitários. Entretanto, Schuld *et al.* explicam uma possível potência na exploração de redes neurais quânticas no fato de que esse modelo utilizaria as vantagens tanto do estado de superposição quanto do processamento paralelo.

Em busca das possíveis vantagens na utilização da computação quântica em redes neurais, Tacchino *et al.* (2019) e Mangini *et al.* (2020) propuseram, implementaram e avaliaram um neurônio quântico, se baseando no *perceptron* clássico de Rosenblatt (Rosenblatt (1958)) e no modelo de neurônio de McCulloch-Pitts (McCulloch & Pitts (1943)). Ambos os autores implementaram seus neurônios artificiais quânticos seguindo um modelo semelhante, diferindo-se apenas ao conjunto de portas quânticas utilizadas, o que resultou em limitações diferentes para o carregamento do vetor de entradas e de peso. Para os modelos de Tacchino *et al.*, os pesos e as entradas são limitadas binariamente, sendo -1 ou 1. Já para Mangini *et al.*, os pesos e entradas são limitadas entre $[0, \pi]$, uma vez que baseiam-se no deslocamento de fase. Esses modelos e suas limitações são melhores apresentados no Capítulo 3.

Os neurônios quânticos propostos por Tacchino *et al.* (2019) e Mangini *et al.* (2020) utilizam um algoritmo de computação clássico para realizar o treinamento dos seus pesos. Esse método de treinamento híbrido, chamado de “treinamento clássico-quântico”, atualiza os pesos baseado na saída do circuito quântico. Com o algoritmo de treinamento híbrido, Tacchino *et al.* e Mangini *et al.* validaram seus modelos em um problema artificial, que consiste em tentar deduzir o grau de semelhança à uma imagem alvo, utilizando apenas as imagens e seus graus de semelhança como treino. Para Tacchino *et al.*, as imagens poderiam ser apenas preto ou branco, já que essa é a limitação de seus modelos. Por outro lado, para Mangini *et al.*, as imagens seriam uma variação de escalas de cinza, por poderem assumir valores reais. Além de avaliar neste problema artificial, Mangini *et al.* avaliaram o seu modelo no banco de dados MNIST, que é um banco de dados com imagens em tons de cinza de números escritos à mão. Mangini *et al.* se limitaram a classificar as imagens dos números 0 e 1, e atingiu uma acurácia de 98%.

Assim, motivado pelos resultados obtidos por Tacchino *et al.* e Mangini *et al.*, espera-se que este modelo quântico de neurônio performe de forma semelhante ou até mais eficiente que os neurônios clássicos em problemas reais quanto as métricas de acurácia, F1-Score, precisão e revocação, como os problemas da área de saúde.

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

Avaliar a eficácia quanto à acurácia, F1-Score, precisão e revocação dos modelos de neurônios quânticos propostos por [Tacchino *et al.* \(2019\)](#) e [Mangini *et al.* \(2020\)](#), em um treinamento híbrido (clássico-quântico), no banco de dados “Breast Tissue” e no banco de dados “Caesarian Section Classification Dataset”, ambos disponíveis no *UCI Datasets* ([Asuncion & Newman \(2007\)](#)), simulando os circuitos quânticos utilizando a biblioteca *Qiskit*.

1.2.2 Objetivos Específicos

- Implementar o neurônio artificial quântico utilizando diferentes modelos: força bruta e “*hypergraph states generation subroutine*” (HSGS), propostos por [Tacchino *et al.*](#) e o modelo de valores contínuos proposto [Mangini *et al.*](#), definidos no Capítulo 4;
- Combinar os neurônios artificiais quânticos com um algoritmo clássico de regra delta simplificada para atualização de pesos, semelhante ao aplicado por [Tacchino *et al.*](#) e [Mangini *et al.*](#);
- Aplicar o algoritmo híbrido aos banco de dados especificados na Subseção 1.2.1;
- Aplicar um *perceptron* e *multilayer perceptrons* clássicos aos banco de dados especificados na Subseção 1.2.1;
- Comparar os resultados obtidos pelos modelos quânticos e pelos modelos clássicos de neurônios.

2

COMPUTAÇÃO QUÂNTICA

2.1 CONCEITOS BÁSICOS

Uma das diferenças mais importantes entre a computação quântica e computação clássica se dá no fato que, enquanto a unidade básica da computação clássica (os *bits*) tem seus valores definidos em zero ou um, a unidade básica na computação quântica (os *Qubits*) tem seu valor definido por matrizes, tendo sua base canônica definidos na Equação 2.1. Dessa forma, um *Qubit* genérico poder ser representado pela Equação 2.2, onde α e β são as amplitudes do sistema.

Como os *Qubits* deixam de ser valores escalares e passam a ser vetores (ou matrizes de coluna única), eles são representados utilizando a notação de Dirac ($|0\rangle$ e $|1\rangle$). Assim, pode-se observar no *Qubit* um estado intermediário entre 0 e 1, fazendo com que o mesmo possa ser 0 e 1 ao mesmo tempo. Essa propriedade dos *Qubits* é chamada de superposição de estados, e, segundo [Portugal & Marquezino \(2019\)](#), só pode ser alcançada quando o sistema quântico estiver isolado do sistema exterior, alterado apenas pelas portas quânticas, sendo afetada quando alguém afere (observa) o sistema, onde o *Qubit* é lido como 0 ou 1 com uma probabilidade, que é proporcional à amplitude do sistema.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (2.1)$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (2.2)$$

Assim como na computação clássica, que a manipulação de *bits* é feita nos circuitos clássicos utilizando portas lógicas, a manipulação dos *Qubits* é feita através da construção de um circuito quântico, que possui diversas portas quânticas. A diferença é que, enquanto as portas lógicas agem sobre um valor específico (0 AND 1 \Rightarrow 0, por exemplo), as portas quânticas vão agir como multiplicação de matrizes. As portas utilizadas neste projeto são definidas na Subseção 2.1.1.

2.1.1 Características das Portas Quânticas

As portas quânticas mais básicas são as portas *NOT*, *Pauli-Z* e *Hadamard* (Figura 1), que possuem e tem características próprias, além de serem comumente utilizadas nos circuitos de várias aplicações devido à essas características: a porta *Pauli-X*, ou *NOT*, tem função análoga à porta *NOT* clássica: inverter o valor do *Qubit*. Já a *Pauli-Z*, ou simplesmente *Z*, inverte o **sinal** da amplitude do sistema caso o *Qubit* seja $|1\rangle$. A Hadamard (H) é uma das mais importantes por ter a característica de levar um *Qubit* à um estado de superposição de forma reversível. Ou seja, quando aplicado à duas portas Hadamard consecutivas, o *Qubit* voltará ao seu estado original. Além disso, o estado de superposição construído pela porta H aplicada a um estado da base computacional possui a mesma probabilidade do *Qubit* ser lido como 0 ou 1.

Além das portas definidas anteriormente, existem variações delas, que consistem em sua forma controlada. Por exemplo, a CNOT (*controlled NOT*), também definida na Figura 1, aplica a porta NOT à um *Qubit* apenas se o *Qubit* de controle for $|1\rangle$. De forma análoga ao CNOT, temos a *controlled Z*, ou CZ, e as suas versões *multiple controlled* (MCNOT, MCZ), que são controladas por mais de um *Qubit*. Por fim, outra variação da porta Z é a P(λ), onde, ao invés de inverter o sinal do sistema, a multiplica por uma fase recebida como parâmetro (λ). A porta P (*phase*) também possui sua variação controlada, a CP (*controlled phase*) e MCP (*multiple controlled phase*). As definições de todas as portas mencionadas são exibidas na Figura 2.





Gate	Notation	Matrix
NOT (Pauli-X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CNOT (Controlled NOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Figura 1: Portas quânticas, suas representações nos circuitos e respectivas matrizes.

Figura retirada de [Yan et al. \(2014\)](#).

2.2 QISKIT

Uma das bibliotecas que facilitam a implementação dos circuitos quânticos é a *Qiskit*, disponível para a linguagem *Python*. O *Qiskit* permite a implementação de circuitos quânticos de forma simples utilizando a computação clássica, além de permitir a simulação dos mesmos

$$\begin{aligned}
\mathbf{I} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \mathbf{I}|0\rangle &= |0\rangle \\
& & \mathbf{I}|1\rangle &= |1\rangle \\
\mathbf{H} &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \mathbf{H}|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
& & \mathbf{H}|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
\mathbf{X} &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & \mathbf{X}|0\rangle &= |1\rangle \\
& & \mathbf{X}|1\rangle &= |0\rangle \\
\mathbf{CX/CNOT} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \mathbf{CX}|00\rangle &= |00\rangle \\
& & \mathbf{CX}|01\rangle &= |01\rangle \\
& & \mathbf{CX}|10\rangle &= |11\rangle \\
& & \mathbf{CX}|11\rangle &= |10\rangle \\
\mathbf{Z} &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & \mathbf{Z}|0\rangle &= |0\rangle \\
& & \mathbf{Z}|1\rangle &= -|1\rangle \\
\mathbf{CZ} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} & \mathbf{CZ}|00\rangle &= |00\rangle \\
& & \mathbf{CZ}|01\rangle &= |01\rangle \\
& & \mathbf{CZ}|10\rangle &= |10\rangle \\
& & \mathbf{CZ}|11\rangle &= -|11\rangle \\
\mathbf{P}(\lambda) &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix} & \mathbf{P}(\lambda)|0\rangle &= |1\rangle \\
& & \mathbf{P}(\lambda)|1\rangle &= \lambda|0\rangle \\
\mathbf{CP}(\lambda) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix} & \mathbf{CP}(\lambda)|00\rangle &= |00\rangle \\
& & \mathbf{CP}(\lambda)|01\rangle &= |01\rangle \\
& & \mathbf{CP}(\lambda)|10\rangle &= |10\rangle \\
& & \mathbf{CP}(\lambda)|11\rangle &= \lambda|11\rangle
\end{aligned}$$

Figura 2: Portas quânticas, suas definições matriciais e suas aplicações em *Qubits*.

em diversos tipos de simuladores, gerando visualização dos circuitos, histogramas, retornando amplitudes e outras informações, fazendo com que seja possível validar o circuito implementado antes de executá-lo em um computador quântico real. Além disso, a biblioteca possui integração com a ferramenta da empresa IBM¹, que disponibiliza computadores quânticos para rodar testes gratuitamente. Neste projeto, todos os resultados foram retirados de simulações de execução de circuitos quânticos construídos com o auxílio do *Qiskit*.

Um exemplo de uso da biblioteca está disponível nos Algoritmos 2.1 e 2.2, com saídas exibidas nas Figuras 3 e 4. Percebe-se que, devido às portas de Hadamard, todos os valores possíveis de 2 *Qubits* apareceram com uma probabilidade próxima de 0.25 na Figura 4.

```

1 import matplotlib
2 from qiskit import *
3
4 quantum_registers = QuantumRegister(2, 'q')
5 classical_registers = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(quantum_registers, classical_registers)
7
8 circuit.h(quantum_registers)
9 circuit.cx(quantum_registers[0], quantum_registers[1])
10 circuit.barrier(quantum_registers)
11
12 circuit.measure(quantum_registers, classical_registers)
13 circuit.draw(output='mpl', initial_state=True)

```

Algoritmo 2.1: Algoritmo em Python utilizando Qiskit para construção e exibição de um circuito quântico.

¹<https://quantum-computing.ibm.com/>

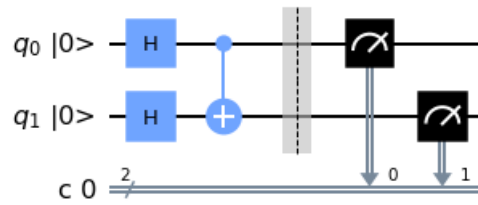


Figura 3: Resultado da execução do Algoritmo 2.1.

```

1 import matplotlib
2 from qiskit import *
3 from qiskit.tools.visualization import plot_histogram
4
5 quantum_registers = QuantumRegister(2, 'q')
6 classical_registers = ClassicalRegister(2, 'c')
7 circuit = QuantumCircuit(quantum_registers, classical_registers)
8
9 circuit.h(quantum_registers)
10 circuit.cx(quantum_registers[0], quantum_registers[1])
11 circuit.barrier(quantum_registers)
12
13 circuit.measure(quantum_registers, classical_registers)
14 simulator = Aer.get_backend('qasm_simulator')
15 result = execute(circuit, backend = simulator).result()
16 plot_histogram(result.get_counts(circuit))

```

Algoritmo 2.2: Algoritmo em Python utilizando Qiskit para construção e simulação de um circuito quântico.

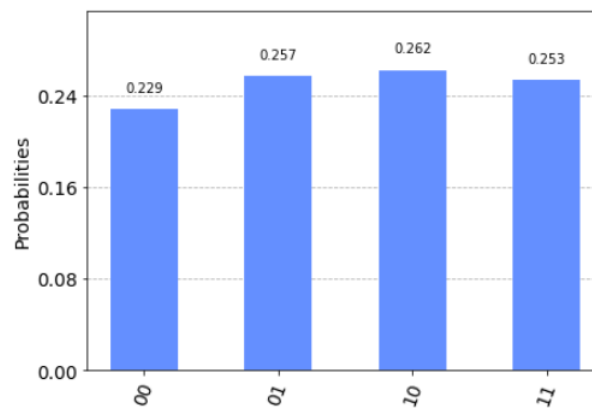


Figura 4: Resultado da execução do Algoritmo 2.2.

3

APRENDIZAGEM DE MÁQUINA QUÂNTICA

A ideia de aliar aprendizagem de máquina com computação quântica é retratada por vários autores, que estudam sua eficiência com diversos experimentos. [Biamonte et al. \(2017\)](#) publicaram uma revisão dos principais trabalhos em aprendizagem de máquina quântica, como os modelos *Quantum Principal Component Analysis* (QPCA), *Quantum Support Vector Machines* (QSVM), e *Quantum Deep Learning*. Já [Grant et al. \(2018\)](#), em busca de investigar os benefícios dessa aliança, avaliam um modelo de circuito quântico para o classificador *Tree Tensor Network* (TTN), uma vez que a estrutura de redes de tensores hierárquicas possui semelhanças com a estrutura de uma rede neural. Ainda na mesma publicação, [Grant et al.](#) avaliam um outro modelo, chamado de MERAs (*Multi-Scale Entanglement Renormalization Ansatz*). Os autores simulam os modelos 1D de ambos os algoritmos, enquanto simulam apenas o modelo 2D do TTN, uma vez que o modelo 2D do MERA é muito extenso, fazendo com que a simulação tomasse muito tempo. O experimento foi feito com os banco de dados Iris ([Fisher \(1936\)](#)) e MNIST ([LeCun \(1998\)](#)). Para o Iris, o circuito de TTN obteve uma acurácia maior que 97% (classificação binária entre classes, Tabela 1). Para o banco de dados MNIST, [Grant et al. \(2018\)](#) realizaram uma classificação binária, usando os classificadores TTN, MERA, híbrido e logística. O classificador híbrido realiza um pré-treinamento completamente clássico para inicializar os parâmetros das portas quânticas, já o logística é a função de regressão logística. Em relação às portas, [Grant et al.](#) varia entre portas simples (único *Qubit*) e portas genéricas (mais de um *Qubit*), considerando os valores de parâmetros como reais e complexos. Nesse experimento da base, foram considerados 4 cenários de classificação: reconhecer dígitos maiores que 4 ou menores e iguais a 4, dígitos pares, 0 ou 1 e 2 ou 7 e os resultados estão dispostos na Tabela 2.

Classificador	Portas	Tipo de porta	1 ou 2	2 ou 3	1 ou 3
TTN	Simple	Reais	100.00 ± 0.00	96.77 ± 0.00	100.00 ± 0.00

Tabela 1

Acurácia média do experimento realizado por [Grant et al.](#) no banco de dados Iris, com o *Tree Tensor Network*. Foi utilizado apenas portas simples, com parâmetros reais e a tarefa de classificação foi dividida entre 3: reconhecer entre as classes 1 ou 2, entre as classes 2 ou 3 e entre as classes 1 ou 3. Resultados retirados de [Grant et al. \(2018\)](#).

Classificador	Portas	Tipo de porta	>4	É Par	0 ou 1	2 ou 7
TTN	Simple	Real	65.59 ± 0.57	72.17 ± 0.89	92.12 ± 2.17	68.07 ± 2.42
TTN	Geral	Real	74.89 ± 0.95	83.13 ± 1.08	99.79 ± 0.02	97.64 ± 1.60
MERA	Geral	Real	75.20 ± 1.51	82.83 ± 1.19	99.84 ± 0.06	98.02 ± 1.40
Híbrido	Geral	Real	76.30 ± 1.04	83.53 ± 0.21	99.87 ± 0.02	98.07 ± 1.46
TTN	Simple	Complexo	70.90 ± 0.73	80.12 ± 0.64	99.37 ± 0.12	94.09 ± 3.37
TTN	Geral	Complexo	77.56 ± 0.45	83.53 ± 0.69	99.77 ± 0.02	97.63 ± 1.48
MERA	Geral	Complexo	79.10 ± 0.90	84.85 ± 0.20	99.74 ± 0.02	98.86 ± 0.07
Híbrido	Geral	Complexo	78.36 ± 0.45	84.38 ± 0.28	99.78 ± 0.02	98.46 ± 0.19
Logística	N/A	N/A	70.70 ± 0.01	81.72 ± 0.01	99.53 ± 0.01	96.17 ± 0.01

Tabela 2

Resultado do experimento realizado por [Grant et al.](#) no banco de dados MNIST. Foram utilizados os classificadores TTN, Mera, híbrido ou logística. Em relação às portas, foram utilizadas portas simples ou genéricas, com parâmetros reais ou complexos. Nesse experimento da base, foi considerado 4 cenários de classificação: reconhecer dígitos maiores que 4 ou menores e iguais a 4, dígitos pares, 0 ou 1 e 2 ou 7. Resultados retirados de [Grant et al. \(2018\)](#).

Partindo para a área de redes neurais, [Schuld et al. \(2014\)](#) tratam, em sua publicação, de possíveis benefícios do uso de computação quântica em conjunto com redes neurais, como o uso do estado de superposição para operações em mais de um estado simultaneamente, mas traz a dificuldade de exploração da não linearidade que é proposta pelo algoritmo clássico no neurônio quântico, devido a necessidade dos operadores quânticos serem unitários. Tentando fazer uso das vantagens da computação quântica e lidando com o desafio de trazer uma não linearidade para o modelo, [Tacchino et al. \(2019\)](#) e [Mangini et al. \(2020\)](#) propuseram modelos de neurônios quânticos utilizando portas H, Z, CZ, MCZ, X, CX, MCX ([Tacchino et al. \(2019\)](#)), P, CP e MCP ([Mangini et al. \(2020\)](#)) para mapear as entradas e os pesos, de forma que o circuito quântico realizasse o produto vetorial entre entrada e pesos, seguindo o esquema da Figura 5. Dessa forma, o neurônio quântico seria capaz de aproveitar do paralelismo quântico para realizar a multiplicação e gerar funções não lineares, uma vez que a ativação do mesmo é baseado em probabilidade, que é uma função quadrática (amplitude do sistema ao quadrado) ([Tacchino et al. \(2019\)](#)).

Esse modelo simplificado garante que o neurônio quântico carregue os vetores de entrada e de pesos seguindo um algoritmo (explicado nas Seções 3.1, 3.2 e 3.3), precedidos de portas Hadamard (para assegurar a sobreposição) e seguidos de portas Hadamard, para desfazer a sobreposição, e uma porta X, para fazer com que o resultado da multiplicação seja proporcional à amplitude do sistema, que, por sua vez, é proporcional à ativação do neurônio. Por fim, os *Qubits* são conectados à um *Qubit* auxiliar chamado de *Ancilla* com uma porta MCX, que é medido após a execução. Esse modelo é apresentado em mais detalhes na Figura 6.

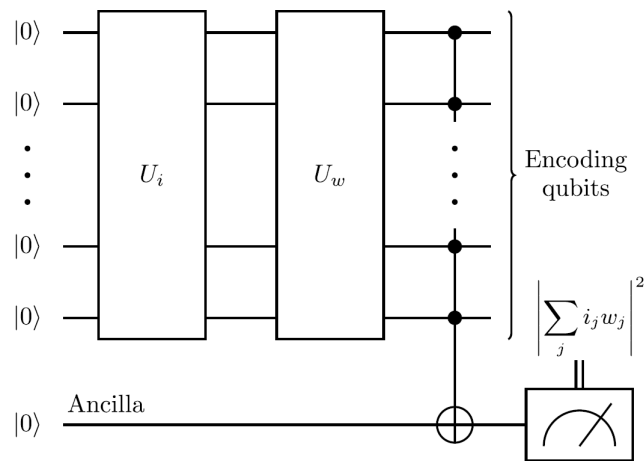


Figura 5: Modelo do neurônio quântico proposto por [Tacchino et al. \(2019\)](#). U_i é a parte do modelo responsável por carregar as entradas e U_w é a parte do modelo responsável por carregar os pesos. Figura retirada de [Tacchino et al. \(2019\)](#)

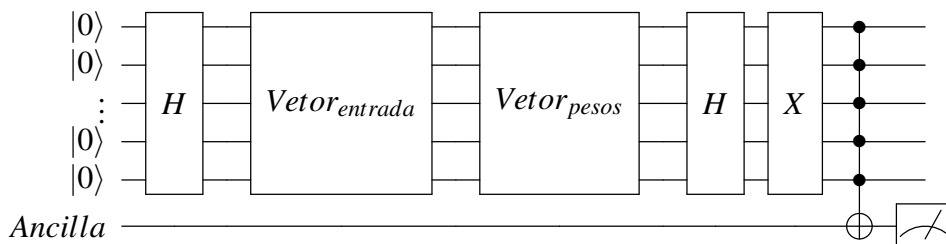


Figura 6: Modelo geral do neurônio quântico utilizado por [Tacchino et al. \(2019\)](#) e [Mangini et al. \(2020\)](#). Os vetores de entrada e de pesos são carregados seguindo um dos modelos descritos nas Seções 3.1, 3.2 e 3.3.

Os modelos propostos por [Tacchino et al. \(2019\)](#) e [Mangini et al. \(2020\)](#) são validados nos experimentos descritos na Seção 1.1, obtendo resultados expressivos (Figuras 7 e 8), e são explicados nas Seções 3.1, 3.2 e 3.3. Em suas publicações, Tacchino et. al e Mangini et. al não validam os modelos em bancos de dados reais completos, uma vez que um dos experimentos é um problema artificial, e o problema do MNIST foi limitado a identificar entre os dígitos 0 ou 1. Além disso, os modelos não foram aplicados em problemas com mais de duas possíveis classificações.

Em 2020, [de Paula Neto et al. \(2020\)](#) buscam atacar o problema de um possível *overfitting* em neurônios quânticos, utilizando como base os modelos de [Tacchino et al.](#) e [Mangini et al.](#) A proposta de [de Paula Neto et al.](#) é utilizar mais portas CNOTs, de forma que amplitudes de mais estados quânticos sejam considerados no resultado final, armazenando-os no *ancilla*. [de Paula Neto et al.](#) utilizam um algoritmo genético simples para definir o limiar do circuito (CT), que é, em outras palavras, o quanto o circuito é permissivo em relação à ativação do *Qubit*, e do neurônio (NT), que é o limiar onde considera-se se o neurônio foi ativado ou não. Esses dois valores controlam a quantidade de CNOTs indiretamente (mais informações disponíveis em

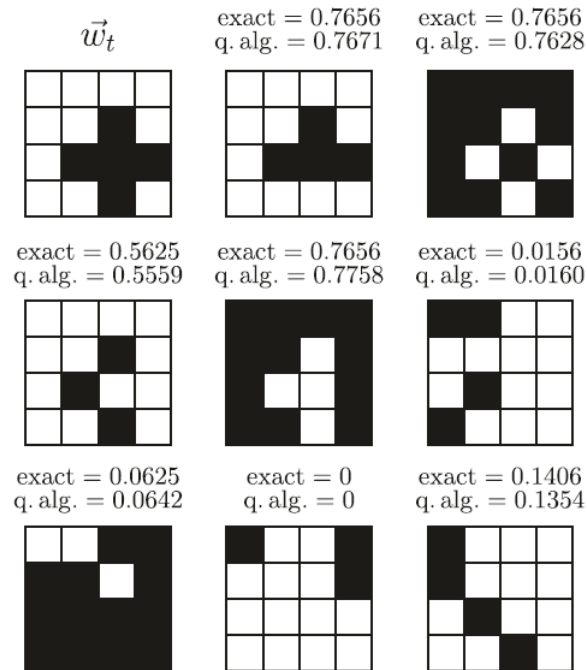


Figura 7: Resultados obtidos ao fim do treinamento do modelo proposto por [Tacchino et al. \(2019\)](#). A imagem alvo está no canto superior esquerdo. Acima de cada imagem além da imagem alvo está o grau de semelhança obtido através de álgebra linear padrão (*exact*) e o grau de semelhança apontado pelo neurônio quântico (*q. alg.*). Figura retirada de [Tacchino et al. \(2019\)](#).

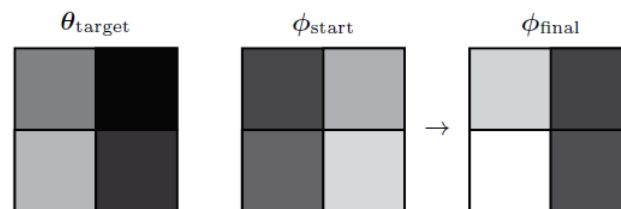


Figura 8: Resultado obtido ao fim da execução do modelo proposto por [Mangini et al. \(2020\)](#). Nela, o θ_{target} é a imagem alvo, θ_{start} é a representação gráfica dos pesos iniciais e θ_{final} é a representação gráfica dos pesos ao fim do treinamento. Figura retirada de [Mangini et al. \(2020\)](#).

de Paula Neto et al. (2020)).

Este trabalho, entretanto, está focado apenas nos modelos de neurônios quânticos de [Tacchino et al. \(2019\)](#) e [Mangini et al. \(2020\)](#), e avalia ainda as estratégias de pré-processamento propostas por [Monteiro \(2021\)](#) (explicada na Seção 3.4).

3.1 FORÇA BRUTA

A primeira proposta de [Tacchino et al. \(2019\)](#) foi de implementar o neurônio quântico utilizando a força bruta para mapear os vetores de entrada e pesos: iterar uma a uma posição no vetor e utilizar portas MCZ, CZ, Z e X para inverter ou não o sinal da amplitude do sistema de

acordo com o valor da posição. Devido a forma que o modelo é proposto, os valores possíveis para os vetores de entrada e de pesos devem ser -1 ou 1, para que seja possível reproduzi-los no circuito quântico. Assim, em um problema real, os valores de cada instância devem ser mapeados para um vetor binário, onde uma conversão deve ser definida (e.g. *float* para binário, separando entre inteiros e decimais).

Um exemplo do modelo de neurônio quântico construído a partir das amplitudes de entrada $[1, -1, 1, -1]$ e pesos $[1, 1, -1, 1]$ utilizando a construção através de força bruta é descrito pela Figura 9. Cada entrada em que o sinal do sistema é negativo é separado por uma barreira, e o pseudocódigo para a construção do modelo está exibido no Algoritmo 3.1. O Algoritmo 3.1 faz com que o circuito quântico cresça em comprimento proporcionalmente à quantidade de amplitudes negativas da binarização dos vetores de entrada e peso, e tem complexidade $O(n \log(n))$, uma vez que itera sobre o vetor (complexidade $O(n)$) e, para cada elemento do vetor, percorre os *Qubits* correspondentes à amplitude daquele elemento (complexidade $O(\log(n))$).

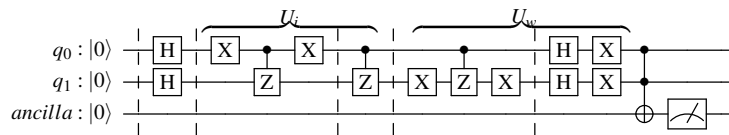


Figura 9: Exemplo de um circuito quântico com entrada $v_i = [1, -1, 1, -1]$ e pesos $w_i = [1, 1, -1, 1]$, construído utilizando o modelo de Força Bruta proposto por [Tacchino et al. \(2019\)](#).

```

1 estados_base = ['00...0', '00...1', ..., '11...1']
2
3 def adiciona_not_em_estado_base_0(estado_base):
4     # estado_base aqui seria '010' por exemplo
5     for qubit in estado_base:
6         if qubit == 0:
7             circuito.x(h)
8
9 def forca_bruta(elementos):
10    for indice in range(elementos):
11        elemento = elementos[indice]
12        estado_base = estados_base[indice]
13        if elemento == -1:
14            adiciona_not_em_estado_base_0(estado_base)
15            circuito.mcz(estado_base)
16            adiciona_not_em_estado_base_0(estado_base)
17
18 def main():
19     @global circuito
20     for qubit in estado_base:
21         circuito.hadamard(qubit)
22
23     forca_bruta(entrada)
24     forca_bruta(peso)
25
26     for qubit in estado_base:
27         circuito.hadamard(qubit)

```



```

28     circuito.x(qubit)
29
30     circuit.mcx(estado_base, ancilla)

```

Algoritmo 3.1: Pseudocódigo para o mapeamento do neurônio quântico utilizando força bruta.

3.2 HYPERGRAPH STATES GENERATION SUBROUTINE (HSGS)

Ainda seguindo a proposta de utilizar entradas e pesos binários (-1 ou 1), [Tacchino et al. \(2019\)](#) propuseram um outro modelo, chamado de *Hypergraph States Generation Subroutine*, ou HSGS, que, em pior caso, montaria um circuito tão extenso quanto o de força bruta, mas que o circuito montado seria menor na maioria dos casos. O modelo consiste em iterar sobre os *Qubits* com a mesma quantidade de 1s: primeiro todos os *Qubits* com um 1 (001, 010, 100...), depois todos com dois 1s (011, 101, 110...), depois com três 1s (1110, 0111...) etc., e, a cada iteração, adiciona uma porta Z ou CZ e atualiza o sinal de todos os *Qubits* afetados. Caso o estado atual seja o desejado, a iteração é interrompida. Como esse modelo utiliza apenas portas Z ou CZ, o algoritmo não lida com vetores onde o primeiro elemento da entrada ou dos pesos é negativo, uma vez que as portas Z ou CZ necessitam de ao menos um *Qubit* igual a 1 para ser ativado e o primeiro elemento do vetor de entrada/pesos equivale ao *Qubit* |00...00). Sendo assim, um tratamento adicional para estes casos deve ser considerado (e.g. inverter todos os sinais daquele vetor).

Um exemplo de neurônio construído utilizando o modelo HSGS, com entrada [1, -1, 1, -1] e peso [1, 1, -1, 1] é exibido na Figura 10. Comparando as Figuras 9 e 10, percebe-se que a última utiliza menos portas para a mesma combinação de entrada e pesos. O pseudocódigo para a construção do modelo está exibido no Algoritmo 3.2. Uma otimização desse pseudo-código poderia ser feita ao ordenar o vetor de possíveis *Qubits*, uma vez que as posições de *Qubits* com mesma quantidade de 1s seriam adjacentes.

O Algoritmo 3.2 faz com que o circuito quântico cresça em comprimento proporcionalmente à quantidade de amplitudes negativas da binarização dos vetores de entrada e peso, mas pode conseguir resolver mais de uma amplitude com apenas uma porta, dependendo da posição. Ele tem complexidade $O(n^2 \log(n))$ no pior caso do carregamento, que acontece quando o algoritmo analisa todos os estados base que possuem ao menos um 1. O laço da linha 94 itera entre $\log(n)$ valores, uma vez que é a quantidade máxima de 1s. Para cada iteração desse laço, todos os estados base possíveis são analisados, o que significa que são n iterações. Já a função `adiciona_portas_z_a_qubits_1` itera sobre um estado base, tendo complexidade $\log(n)$ e a função `atualiza_estado_atual` itera sobre o vetor de amplitudes, tendo complexidade n . Dessa forma, a complexidade desse algoritmo de carregamento como $\log(n) * (n)(\log(n) + n) = O(n^2 \log(n))$. Entretanto, na maioria das vezes, o algoritmo terminará antes, uma vez que para ao atingir o estado de amplitudes desejado.

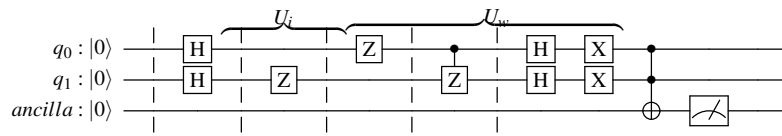


Figura 10: Exemplo de um circuito quântico com entrada $v_i = [1, -1, 1, -1]$ e pesos $w_i = [1, 1, -1, 1]$, construído utilizando o modelo de HSGS proposto por [Tacchino et al. \(2019\)](#).

```

1 estados_base = ['00...0', '00...1', ..., '11...1']
2
3 def pega_indices_qubits_1(estado_base):
4     indices_estado_base_1 = []
5     for indice in range(len(estado_base)):
6         qubit = estado_base[indice]
7         if qubit == 1:
8             indices_estado_base_1.append(indice)
9     return indices_estado_base_1
10
11 def adiciona_portas_z_a_qubits_1(estado_base, qtde_1s):
12     indices_estado_base_1 = pega_indices_qubits_1(estado_base)
13     if qtde_1s > 1:
14         circuito.mcz(indices_estado_base_1)
15     else:
16         circuito.z(indices_estado_base_1)
17
18 def atualiza_estado_atual(estado_atual, qubits):
19     indices_qubits_1 = pega_indices_qubits_1(qubits)
20     for indice in range(len(estados_base)):
21         estado_base = estados_base[indice]
22         indices_qubits_1_atual = pega_indices_qubits_1(estado_base)
23
24         if indices_qubits_1 in indices_qubits_1_atual:
25             estado_atual[indice] *= -1
26
27     return estado_atual
28
29 def hsgs(elementos):
30     num_qubits = len(estados_base[0])
31     estado_atual = [1, 1, ..., 1]
32
33     for qtde_1s in range(1, num_qubits):
34         for estado_base in estados_base:
35             if count('1', estado_base) == qtde_1s and estado_atual != elementos:
36                 adiciona_portas_z_a_qubits_1(estado_base, qtde_1s)
37                 estado_atual = atualiza_estado_atual(estado_atual, estado_base)
38             if estado_atual == elementos:
39                 return
40
41 def main():
42     @global circuito
43     for qubit in qubits:
44         circuito.hadamard(qubit)
45
46     hsgs(entrada)
47     hsgs(peso)

```

```

48
49     for qubit in qubits:
50         circuito.hadamard(qubit)
51         circuito.x(qubit)
52
53     circuit.mcx(qubits, ancilla)

```

Algoritmo 3.2: Pseudocódigo para o mapeamento do neurônio quântico utilizando HSGS.

3.3 NEURÔNIO DE VALORES CONTÍNUOS

Baseando-se nos modelos propostos por [Tacchino *et al.* \(2019\)](#), [Mangini *et al.* \(2020\)](#) propuseram uma alteração nos modelos para que as entradas pudessem agora variar entre 0 e π , pois utilizariam o deslocamento de fase de cada amplitude. A alteração proposta por [Mangini *et al.*](#) consiste em alterar as portas Z/CZ/MCZ do modelo de força bruta (Seção 3.1) por uma porta P, CP ou MCP. Devido ao fato do modelo se basear no deslocamento de fases da entrada ($e^{\lambda i}$), deve-se ter todas as entradas positivas, uma vez que o intervalo é entre 0 e π . Caso a entrada seja negativa, o sinal é considerado em seus pesos, pois o modelo realiza o produto vetorial entre entrada e (-peso). Além disso, a atualização dos pesos também deve ser feita baseada nos expoentes de cada peso, uma vez que esse é o parâmetro que vai ser variado.

Um exemplo de neurônio construído utilizando o modelo de [Mangini *et al.*](#), aqui chamado de *phase encoding*, com entradas [1.5, 1.3] e pesos [0.2, 0.4] é exibido na Figura 11, e o pseudocódigo para a construção do modelo está exibido no Algoritmo 3.3. Em relação ao tamanho do circuito, diferentemente do força bruta, o *phase encoding* cresce diretamente proporcional ao tamanho da entrada, uma vez que é necessário uma porta P/CP/MCP para cada amplitude. Por outro lado, a quantidade de elementos no vetor de entrada e no vetor de peso diminui, uma vez que não é necessária a binarização dos mesmos. Já tratando sobre a complexidade do algoritmo de carregamento, ele possui a mesma complexidade que o algoritmo de força bruta: $O(n \log(n))$.

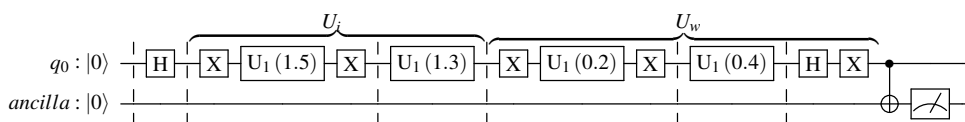


Figura 11: Exemplo de um circuito quântico com entrada $v_i = [1.5, 1.3]$ e pesos $w_i = [0.2, 0.4]$, construído utilizando o modelo proposto por [Mangini *et al.* \(2020\)](#)

```

1 estados_base = ['00...0', '00...1', ..., '11...1']
2 pi = 3.1415...
3
4 def adiciona_not_em_qubits_0(estado_base):
5     # estado_base aqui seria '010' por exemplo
6     for qubit in estado_base:
7         if qubit == 0:

```

```

8         circuito.x(h)
9
10 def phase_encoding(elementos):
11     for indice in range(elementos):
12         elemento = input_vector[indice]
13         estado_base = estados_base[indice]
14
15         adiciona_not_em_qubits_0(estado_base)
16         circuito.mcp(estado_base, elemento)
17         adiciona_not_em_qubits_0(estado_base)
18
19 def normaliza_vetor_ate_pi(elementos):
20     vetor_normalizado = []
21     fator_normalizacao = vetor_input.max()/pi
22
23     for elemento in elementos:
24         vetor_normalizado.append(elemento/fator_normalizacao)
25
26     return vetor_normalizado
27
28 def main():
29     @global circuit
30     for qubit in estado_base:
31         circuito.hadamard(qubit)
32
33     entrada = normaliza_vetor_ate_pi(entrada)
34     peso = normaliza_vetor_ate_pi(peso)
35
36     phase_encoding(entrada)
37     phase_encoding(peso)
38
39     for qubit in qubits:
40         circuito.hadamard(qubit)
41         circuito.x(qubit)
42
43     circuito.mcx(qubits, ancilla)

```

Algoritmo 3.3: Pseudocódigo para o mapeamento do neurônio quântico utilizando phase encoding.

3.4 PRÉ-PROCESSAMENTO DE DADOS

Continuando o trabalho de [Mangini et al. \(2020\)](#), [Monteiro \(2021\)](#) propõe uma forma de pré-processamento das entradas, baseando-se na representação gráfica em (N)D dos mesmos, onde N é o tamanho do vetor de entrada. Para um vetor de entrada V de tamanho n igual a $V = [v_0, v_1, \dots, v_{n-1}]$, ao invés de utilizar o deslocamento de fase diretamente, [Monteiro](#) utiliza as seguintes características: (1) ângulo de cada entrada, descrito na Equação 3.1, (2) o raio de cada entrada (Equação 3.2) e (3) uma combinação entre o raio e o ângulo de cada entrada, onde é utilizado cada ângulo (Equação 3.1) na transformação do raio (Equação 3.2). Além disso, o vetor gerado por esse pré-processamento também é concatenado com os resultados das funções descritas nas Equações 3.3 e 3.4.

$$\arctan \frac{v_k}{v_{k+1}}, k = 0, 2, 4, \dots, n-1 \quad (3.1)$$

$$\sqrt{v_k^2 + v_{k+1}^2}, k = 0, 2, 4, \dots, n-1 \quad (3.2)$$

$$\sqrt{\sum_{j=0}^{n-1} v_j^2} \quad (3.3)$$

$$\arcsin \frac{v_{n-1}}{\sqrt{\sum_{j=0}^{n-1} v_j^2}} \quad (3.4)$$

Além do pré processamento, [Monteiro](#) também propõe uma alteração no modelo HSGS, onde torna possível o uso de entradas ou pesos reais. Essa alteração é feita de forma similar ao feito por [Mangini et al. \(2020\)](#) e consegue manter a proporcionalidade entre a probabilidade de ativação do *ancilla* e a multiplicação dos vetores de peso e entrada. Mais informações em [Monteiro \(2021\)](#).

4

EXPERIMENTOS

4.1 VISÃO GERAL

Neste capítulo, apresentaremos os experimentos realizados usando os modelos de neurônios propostos por [Tacchino *et al.* \(2019\)](#) e [Mangini *et al.* \(2020\)](#), com e sem as estratégias de pré processamento de entradas e pesos propostas por [Monteiro \(2021\)](#), aplicados a duas bases de dados da área de saúde. O algoritmo de treinamento utilizado é uma combinação entre computação clássica e quântica, uma vez que o pré processamento dos dados e atualização dos pesos são feitos com um algoritmo clássico, utilizando a regra delta de atualização, e o neurônio que realiza a multiplicação entre entrada e pesos, retornando a ativação, é um neurônio quântico. Sendo assim, os modelos de construção do neurônio seguem os modelos descritos no Capítulo 4, onde variou-se os parâmetros taxa de aprendizagem, limiar, presença ou não de *bias* e estratégia de pré-processamento.

Para cada combinação de hiperparâmetro, avaliam-se a eficácia dos neurônios utilizando as métricas de acurácia média, o F1-Score médio, a precisão média e a revocação média entre as classes. O código da aplicação está disponível no GitHub, disponível sob requisição.

As bases de dados também foram avaliadas nos neurônios clássicos e na MLP, com o objetivo a performance dos modelos. Foram utilizadas a mesma semente aleatória para a divisão entre os conjuntos de dados de treino e teste, garantindo que os mesmos conjuntos de dados sejam apresentados a todos os modelos. A divisão entre os banco de dados foi feita de forma estratificada, a fim de que ambos possuíssem instâncias de todas as classes. O código da execução dos algoritmos clássicos estão disponíveis no GitHub¹. Neste repositório, além dos algoritmos clássicos, há códigos simplificados para geração dos neurônios quânticos seguindo os modelos descritos no Capítulo 4.

¹<https://github.com/EduardoBrito97/QuantumArtificialNeuron>

4.2 ESPECIFICAÇÃO DO ALGORITMO

4.2.1 Característica dos bancos de dados

O banco de dados “Caesarian Section Classification” consiste em classificar casos de partos onde foi necessária uma cirurgia de cesária ou não, sendo um problema de classificação binária, e possuindo um total de 80 instâncias. Além disso, os seus dados são, em sua maioria, categóricas (são números inteiros e de pouca variação). São elas, em tradução livre:

- idade, variando entre 17 e 40;
- quantidade de filhos, variando entre 1 e 4;
- tempo de gestação, variando entre 0 e 2, sendo 0 no tempo esperado, 1 prematuro e 2 pós-termo;
- pressão sanguínea, variando entre 0 e 2, sendo 0 baixa, 1 normal e 2 alta; e
- problema do coração, variando entre 0 e 1, sendo 0 sem problemas e 1 com problemas.

Já o banco de dados “Breast Tissue” consiste em classificar tecidos da mama entre 6 possíveis classes: ‘car’ (carcinoma), ‘fad’ (fibro-adenoma), ‘mas’ (mastopatia), ‘gla’ (glandular), ‘con’ (conjuntivo) ou ‘adi’ (adiposo), e possui 106 instâncias. Seus dados são todos contínuos, possuindo infinitos valores, incluindo frações. Suas colunas trazem, em tradução livre:

- impeditividade na frequência zero (em Ohm);
- ângulo de fase a 500KHz;
- inclinação da fase do ângulo em alta frequência;
- distância de impedância entre o fim dos espectros;
- área abaixo do espectro;
- área normalizada pelo DA;
- máximo do espectro;
- distância entre IO e a parte real do ponto de frequência máxima; e
- comprimento da curva espectral.

Todas essas medidas possuem um grau 4 de precisão decimal e não seguem nenhum padrão que facilite a binarização.

Para executar os experimentos, ambos os bancos de dados são normalizados e adicionados colunas constantes a fim de que o número de colunas independentes sejam potências de 2. Assim, são adicionadas 3 colunas constantes ao banco de dados “Caesarian Section Classification”, totalizando 8 variáveis independentes, e 7 colunas constantes ao banco de dados “Breast Tissue”, totalizando 16 variáveis independentes. Todas as colunas constantes possuem valor igual a 1 e também são utilizadas pelos algoritmos clássicos.

4.2.2 Estrutura do Algoritmo

O algoritmo utiliza a regra delta simples para a atualização dos pesos do neurônio. A regra é definida na Equação 4.1, onde ‘ w_i ’ é o peso da posição i , ‘ lr ’ é a taxa de aprendizado, ‘ y_{real} ’ é a resposta correta da instância, ‘ y_{pred} ’ é a resposta dada pelo modelo e ‘ i_i ’ é o elemento do vetor de entrada. Essa é uma simplificação do gradiente descendente e foi utilizada de forma não ideal para o modelo de *phase encoding*, uma vez que a atualização para esse modelo deve ser feita nas fases dos pesos, e a Equação 4.1 age baseada na ativação do modelo. Além disso, a conversão de *float* para binário, utilizada nos modelos de HSGS e força bruta, foi feita de forma tradicional, separando entre inteiros e decimais, e utilizando 4 *bits* de precisão.

$$w_i = w_i - (lr \cdot (y_{real} - y_{pred}) \cdot i_i) \quad (4.1)$$

Devido a sua natureza binária, um neurônio quântico com apenas 1 limiar é suficiente para resolver o problema exposto no banco de dados “Caesarian Section Classification”. A representação do algoritmo utilizado para resolver o problema deste banco de dados é ilustrado na Figura 12.

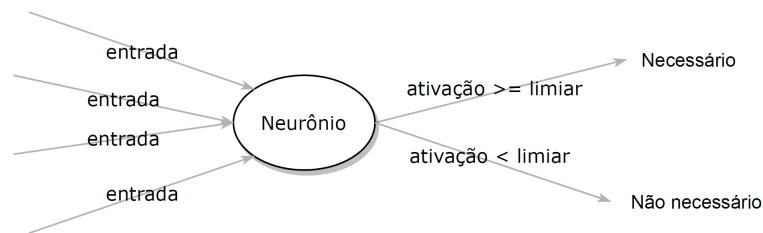


Figura 12: Representação do neurônio aplicado no banco de dados “Caesarian Section Classification Dataset”.

Por outro lado, o problema multi classe do banco de dados “Breast Tissue” faz com que um neurônio quântico com apenas um limiar não seja suficiente para o resolver, e duas abordagens são propostas neste projeto. A primeira abordagem consiste em um único neurônio com 5 limiares, onde cada intervalo entre limiares corresponderia à uma classe específica. A forma de decidir esses 5 limiares é variada entre dividir 1 (100%) igualmente entre 6 parcelas, uma vez que precisamos de 6 intervalos, e dividir 1 de forma ponderada, fazendo com que a classe com maior quantidade de instâncias possuísse um intervalo maior de ativação, e garantindo que cada classe possuísse um intervalo mínimo, a fim de evitar classes sem intervalo. A representação desta abordagem é ilustrado na Figura 13, utilizando os limiares (0.165, 0.33, 0.495, 0.66, 0.825).

A segunda abordagem ao problema de classificação de tecidos consiste em dividir o problema em 6: cada neurônio é treinado para distinguir apenas se é ou não um determinado tecido, e o tipo de tecido o qual o neurônio que possuir a maior ativação define qual é a classe daquela instância. A representação desta abordagem é ilustrado na Figura 14.

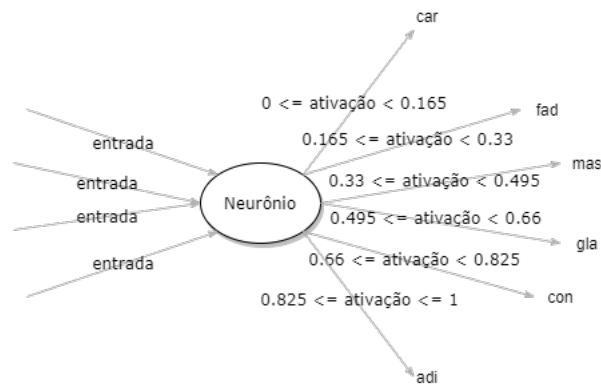


Figura 13: Representação do neurônio aplicado no banco de dados “Breast Tissue Dataset”, utilizando a abordagem de neurônio único. A ordem de cada classe é definido automaticamente durante o treinamento, dependendo da exposição de instâncias daquela classe.

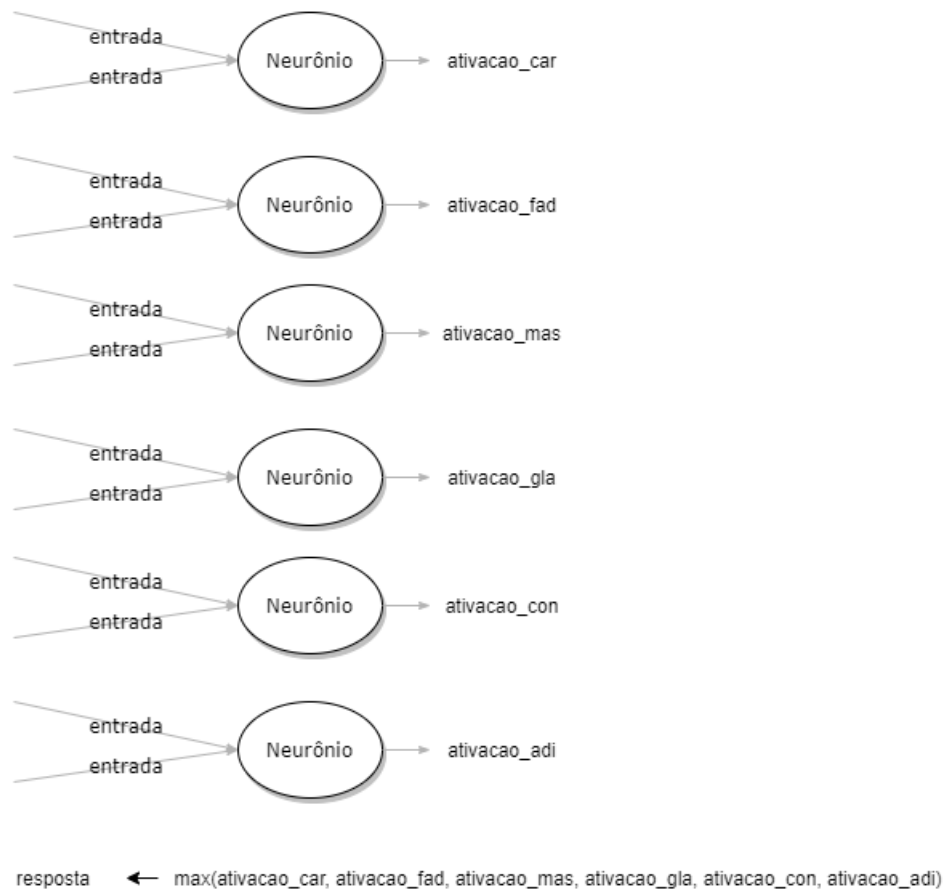


Figura 14: Representação do neurônio aplicado no banco de dados “Breast Tissue Dataset”, utilizando a abordagem de múltiplos neurônios.

4.2.3 Exploração de Hiperparâmetros

Como o problema proposto pelo banco de dados “Caesarian Section Classification” é um problema mais simples, ele consegue ser abordado utilizando uma quantidade menor de *Qubits* e as simulações para esse experimento são mais rápidas, permitindo uma maior variação dos

parâmetros. Por outro lado, além de possuir mais instâncias e mais variáveis independentes, o banco de dados “Breast Tissue” tem dados com muita variação em casas decimais, passando a requerer uma quantidade maior de *Qubits*, o que é afetado ainda mais pela binarização dos dados nos modelos de HSGS e força bruta. Por causa disso, o tempo de simulação para a abordagem de força bruta é bem maior que o HSGS, principalmente em relação ao banco de dados “Breast Tissue”. Dessa forma, há uma variação maior de hiperparâmetros para o banco de dados “Caesarian Section Classification”, principalmente para o modelo de HSGS. Todas as combinações são exibidas na Tabela 3.

Outra redução de execução de configurações em simulações é feita nos experimentos de força bruta. Devido ao comprimento de seu circuito quântico ser essencialmente maior ou igual ao do modelo HSGS e ter os princípios de funcionamento iguais, as configurações usando este modelo demoram mais a serem simuladas e obtêm resultados próximos ou iguais ao do HSGS. A título de exemplo, utilizando uma instância de treinamento do banco de dados “Breast Tissue”, o modelo de força bruta utiliza 4523 portas, enquanto o modelo HSGS utiliza 926. Já para o banco de dados “Caesarian Section Classification”, para uma instância de treinamento, o modelo de força bruta utiliza 1177 portas, enquanto o modelo HSGS utiliza 303. Essa diferença de tempo de execução é exibida na Tabela 5. Assim, para as configurações de força bruta, variou-se o limiar apenas entre [0.3, 0.5, 0.7] para o banco de dados “Caesarian Section Classification”. Para o banco de dados “Breast Tissue”, quanto a abordagem de um único neurônio, optou-se por não utilizar o *bias* e variar a taxa de aprendizagem entre [0.5, 0.7]. A variação dos hiperparâmetros para a abordagem de força bruta está definida na Tabela 4.

Por fim, as quantidade máxima de épocas de treinamento dos neurônios quânticos foi fixada em 20 para as abordagens que utilizam um neurônio único e em 10 para a abordagem que usa mais de um neurônio.

Abordagem	Taxa de aprendizagem	Limiar	Bias	Estratégia (válido apenas para Phase-Encoding)	Num. Épocas
Neurônio único <i>Caesarian</i>	[0.02, 0.1]	[0.3, 0.5, 0.7, 0.9]	[Presente, Ausente]	[original, ângulo, raio, ângulo/raio]	20
Neurônio único <i>Breast Tissue</i>	[0.02, 0.1]	[(0.165, 0.33, 0.495, 0.66, 0.825), limiares através da média ponderada]	[Presente, Ausente]	[original, ângulo, raio, ângulo/raio]	20
Múltiplos Neurônios <i>Breast Tissue</i>	[0.1]	[0.5, 0.7]	[Ausente]	[original]	10

Tabela 3

Variação dos hiperparâmetros utilizados nos experimentos de cada modelo de neurônio quântico e abordagem, para os modelos de HSGS e *phase-encoding*.

Abordagem	Taxa de Aprendizagem	Limiar	Bias	Num. Épocas
Neurônio único <i>Caesarian</i>	[0.02, 0.1]	[0.3, 0.5, 0.7]	[Presente, Ausente]	20
Neurônio único <i>Breast Tissue</i>	[0.02, 0.1]	[(0.165, 0.33, 0.495, 0.66, 0.825), limiars através da média ponderada]	[Ausente]	20
Múltiplos Neurônios <i>Breast Tissue</i>	[0.1]	[0.5, 0.7]	[Ausente]	10

Tabela 4

Variação dos hiperparâmetros utilizados nos experimentos para o modelo de neurônio quântico de Força Bruta e cada abordagem.

Abordagem	Breast Tissue	Breast Tissue Tempo/Config	Caesarian	Caesarian Tempo/Config
Neurônio Quântico Força Bruta	Um neurônio: 2d 3h 22min 6s Multi-Neurônios: 4d 17h 32min 16s	Um neurônio: 12h 50min 31.5s Multi-Neurônios: 2d 8h 46min 8s	2d 18h 42min 11s	5h 33min 30.92s
Neurônio Quântico HSGS	Um neurônio: 1d 9h 28min 35s Multi-Neurônios: 12h 23min 45s	Um neurônio: 4h 11min 4.38s Multi-Neurônios: 6h 11min 25s	13h 53min 26s	52min 5.38s
Neurônio Quântico Phase-Encoding	Um neurônio: 4d 15h 34min 37s Multi-Neurônios: 5d 9h 21min 34s	Um neurônio: 3h 29min 12.41s Multi-Neurônios: 2d 16h 40min 47s	2d 1h 35min 1s	46min 29.08s

Tabela 5

Tempo necessário para executar as simulações variando os hiperparâmetros definidos nas Tabelas 3 e 4.

4.2.3.1 Algoritmos Clássicos

Em relação aos hiperparâmetros dos algoritmos clássicos, como sua execução é mais rápida, a exploração pôde ser mais abrangente.

Entretanto, a biblioteca utilizada (*sklearn*) não permite variar os hiperparâmetros de limiar e presença ou não de *bias* nas chamadas de suas funções. Dessa forma, os seguintes hiperparâmetros foram variados nos experimentos: limiar e presença ou não de *bias*. Sendo assim, para o MLP varia-se os parâmetros:

- taxa de aprendizagem, entre [0.02, 0.1, 0.001, 0.9];
- otimizador, entre ['adam'², 'sgd'³, 'lbfgs'⁴];
- camadas ocultas, 16x32x6, para o banco de dados “Breast Tissue”, e 8x16x2 para o banco de dados “Caesarian Section Classification”.

Já para o perceptron clássico, varia-se os parâmetros:

- taxa de aprendizagem, entre [0.02, 0.1, 0.001, 0.9];
- penalidade, entre ['l1'⁵, 'l2'⁶, 'elasticnet'⁷];

Além disso, ao ser confrontado com um problema multi classe, o *Perceptron* da biblioteca *sklearn* utiliza a abordagem de multi-neurônios para o resolver, sendo semelhante a segunda abordagem apresentada para o banco de dados “Breast Tissue” na Subseção 4.2.2.

²Adam Optimization Algorithm.

³Stochastic Gradient Descent.

⁴Otimizador pertencente na família dos métodos *quasi-Newton*.

⁵L1-Norm.

⁶L2-Norm.

⁷Combinação linear entre L1-Norm e L2-Norm.

4.2.4 Bibliotecas Utilizadas

Para a criação e simulação dos neurônios quânticos, foi utilizada a biblioteca *Qiskit*, em sua versão 0.28.0. Apesar da integração com a API da IBM, os neurônios não foram executados em computadores quânticos reais. Além disso, os algoritmos híbridos que utilizam o modelo de força bruta (Seção 3.1) foram executados com uma variação menor de hiperparâmetros, uma vez que os resultados são próximos ao do HSGS (Seção 3.2), já que usam o mesmo princípio, e o tempo de simulação é maior, uma vez que o circuito quântico é mais extenso.

Com o intuito de simplificar a divisão entre conjuntos de dados de treino e teste, foi utilizada a biblioteca *scikit-learn*, em sua versão 0.24.1. A mesma biblioteca também gera um relatório de diferentes métricas (*classification_report*) e a matriz de confusão automaticamente. Por fim, para manipular os dados, foi utilizado a biblioteca '*pandas*', em sua versão 1.2.3. Todos os *scripts* foram escritos na linguagem *Python*, na versão 3.8.8. Os experimentos foram realizados no sistema operacional Windows 10 Education, versão 20H2, compilação 19042.1110, com processador Intel(R) Core(TM) I5-8250U e 8GB de Memória Ram.

4.3 RESULTADOS

De maneira geral, os resultados médios dos neurônios quânticos no banco de dados Caesarian foram melhores que os algoritmos clássicos ao comparar o F1-Score médio, precisão e a revocação dos modelos HSGS e força bruta em relação aos clássicos, chegando o último a superar os modelos nessas 3 métricas (Tabela 6). Ainda na Tabela 6, observa-se que, mesmo na acurácia média - onde os algoritmos clássicos se sobressaíram - a maior a diferença está entre o HSGS e o *MLP*, chegando a 0.075. Esses resultados levam a crer que, caso houvesse uma maior exploração de hiperparâmetros, os modelos HSGS e força bruta seriam tão bom quanto ou levemente melhor que os algoritmos clássicos, mesmo na acurácia média. Por outro lado, o modelo de *phase-encoding* apresentou resultados aquém do esperado, uma vez que o mesmo usa os valores de peso como pontos flutuantes ao invés de realizar a conversão para binário. Um dos principais motivos para esse comportamento é a utilização da regra delta de atualização dos pesos não ideal, como mencionado na Subseção 4.2.2, que usa a ativação do neurônio diretamente para atualizar os pesos, pesos estes que são utilizados como expoentes do número de Euler.

Em relação ao banco de dados "Breast Tissue", na abordagem de único neurônio, os modelos de neurônio quântico apresentam resultados bastante inferiores aos algoritmos clássicos do *Perceptron* e *MLP*, quando observado a acurácia média e revocação média (Tabela 7). Para os modelos de HSGS e Força Bruta, dois motivos que potencializam essa ineficiência é a irregularidade do banco de dados, apontada na Subseção 4.2.1, o que torna a binarização mais complicada, e o fato de ter usado apenas 4 bits de precisão para as entradas, que provavelmente é uma precisão insuficiente. Já o modelo *phase-encoding* compartilha do mesmo problema citado anteriormente: uma regra delta de atualização não ideal. Ainda assim, quando comparado os

F1-Scores e precisão, essa disparidade cai, chegando o HSGS a superar os modelos clássicos. Esse resultado demonstra que, apesar de não atingir a acurácia próxima dos modelos clássicos, ambas as abordagens (híbrida ou apenas clássica) acabam enviesando suas predições de formas semelhantes, tendo dificuldade de distinguir algumas classes específicas, o que indica uma possível falta de relação entre as variáveis independentes e a dependente do banco de dados ou que, possivelmente, seja preciso otimizar a arquitetura da rede neural artificial e seu modelo de treinamento, a fim de atingir melhores resultados.

Já a abordagem de múltiplos neurônios quânticos para o banco de dados “Breast Tissue” se mostrou aquém do esperado, apresentando resultados bem abaixo tanto dos modelos clássicos quanto da abordagem de neurônio quântico único. Devido aos baixos valores de F1-Score, percebe-se que essa abordagem acaba sempre tendo uma classe predileta, e que, no caso do HSGS e do Força Bruta, provavelmente aconteceu dessa classe escolhida ter sido a mais presente nos bancos de dados de teste, gerando uma acurácia maior.

No banco de dados “Breast Tissue”, os valores de precisão média e F1-Score médios entre as abordagens de força bruta e HSGS apresentam-se um pouco distantes, mesmo com a semelhança entre o mapeamento de pesos e entrada entre os modelos. Essa diferença chega a 0.149 entre as precisões médias, e pode ser justificada pela diferente variação entre os hiperparâmetros apontada na Subseção 4.2.3 - enquanto 8 configurações são exploradas no HSGS, apenas 4 são no força bruta. Por outro lado, os resultados para as abordagens de HSGS e força bruta foram bem próximos para o banco de dados “Caesarian”, tendo a maior diferença entre as médias de precisão, chegando em 0.026. Esses resultados são esperados e comprovam que, apesar da diferença em relação às portas usadas, a similaridade entre as abordagens para o mapeamento de entrada e peso faz com que os resultados sejam semelhantes, dando a vantagem ao HSGS pela maior rapidez de execução (Tabela 5).

Abordagem	Acurácia	F1-Score	Precisão	Revocação
Neurônio Quântico Força Bruta	0.499 ± 0.094	0.438 ± 0.094	0.457 ± 0.23	0.539 ± 0.087
Neurônio Quântico HSGS	0.496 ± 0.071	0.414 ± 0.071	0.431 ± 0.228	0.541 ± 0.06
Neurônio Quântico Phase-Encoding Original	0.466 ± 0.073	0.356 ± 0.073	0.322 ± 0.191	0.512 ± 0.051
Neurônio Quântico Phase-Encoding Ângulo	0.477 ± 0.072	0.366 ± 0.072	0.337 ± 0.212	0.529 ± 0.055
Neurônio Quântico Phase-Encoding Raio	0.47 ± 0.093	0.363 ± 0.093	0.309 ± 0.182	0.516 ± 0.087
Neurônio Quântico Phase-Encoding Ângulo/Raio	0.42 ± 0.06	0.311 ± 0.06	0.274 ± 0.139	0.471 ± 0.074
MLP	0.571 ± 0.069	0.425 ± 0.069	0.37 ± 0.181	0.538 ± 0.083
Perceptron Clássico	0.535 ± 0.065	0.393 ± 0.065	0.367 ± 0.197	0.53 ± 0.059

Tabela 6

Média dos resultados obtidos com a variação de hiperparâmetros descrita na Subseção 4.2.3 por abordagem quanto ao banco de dados “Caesarian Section Classification”.

Abordagem	Acurácia	F1-Score	Precisão	Revocação
Único Neurônio Quântico Força Bruta (Sem uso de Biais)	0.25 ± 0.023	0.225 ± 0.023	0.295 ± 0.03	0.24 ± 0.023
Único Neurônio Quântico HSGS (Com e sem uso de Biais)	0.234 ± 0.028	0.151 ± 0.028	0.146 ± 0.102	0.224 ± 0.026
Único Neurônio Quântico Phase-Encoding Original	0.165 ± 0.052	0.056 ± 0.052	0.035 ± 0.021	0.163 ± 0.055
Único Neurônio Quântico Phase-Encoding Ângulo	0.169 ± 0.032	0.059 ± 0.032	0.037 ± 0.018	0.165 ± 0.023
Único Neurônio Quântico Phase-Encoding Raio	0.181 ± 0.065	0.075 ± 0.065	0.048 ± 0.029	0.189 ± 0.07
Único Neurônio Quântico Phase-Encoding Ângulo/Raio	0.11 ± 0.076	0.051 ± 0.076	0.04 ± 0.045	0.119 ± 0.084
Múltiplos Neurônios Quântico Força Bruta	0.23 ± 0.0	0.06 ± 0.0	0.04 ± 0.0	0.17 ± 0.0
Múltiplos Neurônios Quântico HSGS	0.23 ± 0.0	0.06 ± 0.0	0.04 ± 0.0	0.17 ± 0.0
Múltiplos Neurônios Quântico Phase-Encoding Original	0.14 ± 0.0	0.04 ± 0.0	0.02 ± 0.0	0.17 ± 0.0
MLP	0.321 ± 0.13	0.187 ± 0.13	0.177 ± 0.146	0.297 ± 0.127
Perceptron Clássico	0.318 ± 0.097	0.17 ± 0.097	0.178 ± 0.179	0.273 ± 0.112

Tabela 7

Média dos resultados obtidos com a variação de hiperparâmetros descrita na Subseção 4.2.3 por abordagem quanto ao banco de dados “Breast Tissue”.

4.4 CAESARIAN DATASET

As melhores versões dos neurônios quânticos apresentaram resultados quase tão bons quanto ou superiores ao *perceptron* clássico no banco de dados “Caesarian” (Tabela 8), seguindo o padrão dos resultados médios expostos na Tabela 6. Esses resultados fortalecem a hipótese da média dos neurônios quânticos estarem um pouco inferiores aos do *perceptron* clássico devido a pequena exploração de hiperparâmetros.

Já ao comparar os neurônios quânticos com o MLP, ainda que apresentando F1-Score, precisão e recall médios superiores, a melhor configuração do modelo clássico do MLP se apresentou superior em todas as métricas. Apesar disso, a abordagem de *phase-encoding* chegou próximo aos resultados do MLP, tendo a maior diferença entre as revocações, chegando a 0.09. Esse resultado leva a crer que um modelo utilizando a abordagem *phase-encoding* com uma atualização via regra delta de forma ideal possa chegar ainda mais próximo ou superar o modelo clássico.

Por fim, devido a similaridade dos modelos, as melhores configurações das abordagens de HSGS e força bruta apresentam resultados idênticos.

Abordagem	Acurácia	F1-Score	Precisão	Revocação	Configuração
Neurônio Quântico Força Bruta	0.62	0.63	0.63	0.63	Taxa de aprendizagem = 0.02 Limiar = 0.5 Bias ausente
Neurônio Quântico HSGS	0.62	0.63	0.63	0.63	Taxa de aprendizagem = 0.02 Limiar = 0.5 Bias ausente
Neurônio Quântico Phase-Encoding	0.69	0.68	0.68	0.67	Taxa de aprendizagem = 0.02 Limiar = 0.5 Bias presente Estratégia de uso de raio
MLP	0.75	0.75	0.76	0.76	Taxa de aprendizagem = 0.02 Camadas ocultas = 8x16x2 Ativação = relu Otimizador = lbfgs
Perceptron Clássico	0.62	0.62	0.67	0.65	Taxa de aprendizagem = 0.001 L1-Norm como penalidade

Tabela 8

Métricas das configurações que resultaram em melhores F1-Scores de diferentes abordagens quanto ao banco de dados “Caesarian Section Classification”.

4.5 BREAST-TISSUE DATASET

Ao analisar a Tabela 9, percebe-se que, apesar de ter seus F1-Scores médios próximos aos resultados dos modelos clássicos, as melhores configurações dos neurônios quânticos em relação ao F1-Score apresentaram resultados bastante aquém das melhores configurações dos algoritmos clássicos, tendo aproximadamente metade do rendimento. Um possível fator causador disto é o hiperparâmetro de taxa de aprendizagem ter sido alto para essa aplicação, causando uma alteração muito grande na atualização dos pesos - hipótese sustentada pelo fato da melhor configuração do *perceptron* clássico - que utiliza uma abordagem semelhante à abordagem de múltiplos neurônios para o banco de dados - ter usado uma taxa de aprendizagem 100x menor que o utilizado pela abordagem de múltiplos neurônios quânticos.

Além disso, assim como no banco de dados ‘Caesarian’, observa-se que os modelos de força bruta e HSGS apresentam resultados próximos. O comportamento é esperado devido ao fato das abordagens dos modelos quanto aos vetores de entrada e de pesos serem semelhantes.

Abordagem	Acurácia	F1-Score	Precisão	Revocação	Configuração
Neurônio Único Força Bruta	0.27	0.24	0.25	0.26	Taxa de aprendizagem = 0.02 Limiares = Intervalos Ponderados Bias ausente
Neurônio Único HSGS	0.27	0.25	0.30	0.26	Taxa de aprendizagem = 0.1 Limiares = Intervalos iguais Bias ausente
Neurônio Único Phase-Encoding	0.32	0.16	0.11	0.33	Taxa de aprendizagem = 0.1 Limiares = Intervalos iguais Bias ausente Estratégia de utilizar o raio dos pesos
Multi Neurônios Força Bruta	0.23	0.06	0.04	0.17	Taxa de aprendizagem = 0.1 Limiar = 0.5 Bias ausente
Multi Neurônios HSGS	0.23	0.06	0.04	0.17	Taxa de aprendizagem = 0.1 Limiar = 0.5 Bias ausente
Multi Neurônios Phase-Encoding	0.18	0.05	0.03	0.17	Taxa de aprendizagem = 0.1 Limiar = 0.5 Bias ausente
MLP	0.55	0.52	0.58	0.50	Taxa de aprendizagem = 0.1 Camadas ocultas = 16x32x6 Ativação = relu Otimizador = lbfgs
Perceptron Clássico	0.45	0.36	0.46	0.42	Taxa de aprendizagem = 0.001 L1-Norm como penalidade

Tabela 9

Métricas das configurações que resultaram em melhores F1-Scores de diferentes abordagens quanto ao banco de dados “Breast Tissue”.

5

CONCLUSÃO

Neste trabalho, avaliaram-se a performance dos modelos de neurônios quânticos propostos por [Tacchino et al. \(2019\)](#) e [Mangini et al. \(2020\)](#), que são inspirados no modelo clássico do *Perceptron* ([Rosenblatt \(1958\)](#)), além de uma abordagem de pré-processamento utilizada no trabalho de [Monteiro \(2021\)](#). Eles foram treinados usando um algoritmo clássico com uma regra delta simplificada, e executados nas bases de dados “Caesarian Section Classification” e “Breast Tissue”, ambos disponíveis no *UCI Datasets* ([Asuncion & Newman \(2007\)](#)).

Através dos experimentos realizados, pode-se concluir que os modelos de neurônios quânticos propostos por [Tacchino et al. \(2019\)](#) aparentam ser uma alternativa para situações onde as variáveis independentes são categóricas, ou seja, possuem valores definidos e limitados, uma vez que obtém resultados próximos de algoritmos já consolidados na literatura, levando a crer que uma maior exploração nessa área pode se tornar promissora. Na mesma linha de pensamento, os modelos propostos por [Mangini et al. \(2020\)](#), mesmo com a deficiência de uma regra de atualização própria para o modelo, também apresentaram resultados satisfatórios, uma vez que se saiu tão bom quanto ou melhor que os modelos clássicos no caso do banco de dados “Caesarian Section Classification”. O experimento também reitera a eficiência da estratégia de pré-processamento utilizada por [Monteiro \(2021\)](#), já que foi uma das estratégias que se sobressaiu entre os modelos de neurônio quânticos no banco de dados “Breast Tissue”.

5.1 TRABALHOS FUTUROS

A utilização da regra delta simplificada para o modelo de *phase-encoding* é inadequada, logo um experimento com os mesmos banco de dados utilizando uma regra delta coerente com o formato de pesos do modelo é uma sugestão de trabalho futuro, especialmente na base “Breast Tissue”, que apresentou resultados insatisfatórios quando comparado ao *perceptron* clássico. O uso de treinamento dos modelos usando estratégia de circuito variacional também é promissor e tem apresentado bons resultados ([Chen et al. \(2020\)](#)). Outra sugestão é simular o banco de dados “Breast Tissue” com mais configurações da abordagem com múltiplos neurônios, principalmente com taxa de aprendizagem menores, para verificar a hipótese da causa do baixo desempenho da mesma. Além disso, verificar a performance dos neurônios quânticos reais é uma

importante etapa de validação, uma vez que os modelos passam a serem avaliados na prática. Fazer uma análise assintótica dos modelos - sem limitar as épocas de treinamento - também é uma importante análise da eficácia dos mesmos. Por fim, rodar os mesmo experimentos em problemas com múltiplas classe, porém com variáveis independentes mais facilmente discretizadas ajudará a comprovar ou combater os resultados aqui explicitados para os modelos de força bruta e HSGS, já que acredita-se que a limitação da conversão entre decimal e binário foi um dos principais causadores da queda de desempenho quanto ao banco de dados “Breast Tissue”.

Referências

- Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G., Buell, D. A., *et al.* (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510.
- Asuncion, A. & Newman, D. (2007). Uci machine learning repository.
- Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671):195–202.
- Chen, S. Y.-C., Huang, C.-M., Hsing, C.-W., & Kao, Y.-J. (2020). Hybrid quantum-classical classifier based on tensor network and variational quantum circuit. *arXiv preprint arXiv:2011.14651*.
- de Pádua Braga, A., de Leon Ferreira, A. C. P., & Ludermir, T. B. (2007). *Redes neurais artificiais: teoria e aplicações*. LTC Editora Rio de Janeiro, Brazil:.
- de Paula Neto, F. M., Gustavo Filho, I., & Monteiro, C. A. (2020). Approaches to avoid overfitting in a quantum perceptron. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.
- Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., Green, A. G., & Severini, S. (2018). Hierarchical quantum classifiers. *npj Quantum Information*, 4(1).
- LeCun, Y. (1998). The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- Mangini, S., Tacchino, F., Gerace, D., Macchiavello, C., & Bajoni, D. (2020). Quantum computing model of an artificial neuron with continuously valued input data. *Machine Learning: Science and Technology*, 1(4):045008.
- McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- Monteiro, C. A. (2021). Quantum neurons with real weights for diabetes prediction. Master's thesis, Universidade Federal de Pernambuco, Brasil.
- Portugal, R. & Marquezino, F. (2019). Introdução à programação de computadores quânticos. *Sociedade Brasileira de Computação*.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- Schuld, M., Sinayskiy, I., & Petruccione, F. (2014). The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586.
- Schuld, M., Sinayskiy, I., & Petruccione, F. (2015). An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185.
- Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, 124–134.

Tacchino, F., Macchiavello, C., Gerace, D., & Bajoni, D. (2019). An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26.

Yan, F., Ilyasu, A. M., & Jiang, Z. (2014). Quantum computation-based image representation, processing operations and their applications. *Entropy*, 16(10):5290–5338.