Universidade Federal Rural de Pernambuco
Programa De Pós-Graduação Em Informática Aplicada

Daivid Vasconcelos Leal

Comitê de Redes Neurais Quânticas

Recife, 2022

Daivid Vasconcelos Leal

Comitê de Redes Neurais Quânticas

Orientador: Adenilton J. da Silva

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade Federal Rural de Pernambuco como requisito parcial para conclusão do Mestrado.

Recife, 2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

L435c Leal, Daivid Vasconcelos
Comitê de Redes Neurais Quânticas / Daivid Vasconcelos Leal. - 2022.
56 f. : il.

Orientador: Adenilton Jose da Silva.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2022.

1. Computação Quântica. 2. Aprendizado de Máquina. 3. Rede Neurais Artificiais. I. Silva, Adenilton Jose da, orient. II. Título

CDD 004

À minha família, meus amigos e meus professores.

Às memórias de todas as pessoas mortas pela pandemia como uma forma de revindicar os direitos de todos os brasileiros à vida, à liberdade, à igualdade, à segurança, à saúde, à educação e à propriedade.

Agradecimentos

Aos meus pais, Paulo Sérgio Leal Silva e Iria de Fátima da Silva Vasconcelos, pois foi com o ensinamento e o exemplo deles que eu cheguei até aqui.

Às minhas avós, Sueli Leal e Rose Mare Vasconcelos, que foram essenciais na minha vida e no meu desenvolvimento pessoal.

À Dielder Leal, Flávia da Silva e Elisa Leal, meu irmão e minhas irmãs que me fazem ter força para seguir em frente e servir de exemplo todos os dias.

À Sayonara Lucena, que, como minha companheira, me ajudou a pensar positivo, acreditar que eu conseguiria terminar e ter paciência comigo.

À Juliana Lins, Matheus Uehara, Delando Júnior, Bernardo Júnior, Victor Pina, Nayrene Oliveira, Flávio de Freitas e Lucas de Oliveira pois sem eles eu não teria fôlego para continuar produzindo.

Ao professor Adenilton José da Silva, pois, em meio a pandemia e diversas turbulências que aconteceram durante o tempo de mestrado, acreditou em mim e na minha capacidade de finalizar esse trabalho.

À Israel de Araújo e Tiago de Lima que me ajudaram a produzir artigos e a codificar o necessário para ter os resultados aqui obtidos.

E, por fim, aos professores Tiago Ferreira e Leon Silva por participarem da minha banca avaliadora.

"If there is a God, he's a great mathematician."

(Paul Dirac)

Resumo

Redes Neurais Profundas abrem várias possibilidades para resolver problemas difíceis. O avanço no uso da computação quântica nos permitiu usar recursos quânticos que não têm contrapartida clássica, e nos trouxe muitos algoritmos e técnicas no campo do aprendizado de máquina quântica (AMQ). Uma das propostas do ramo do AMQ era uma rede neural quântica binária (RNQB) que usa um método de amplificação baseado no conhecido algoritmo de busca de Grover. Além de ser um método puramente quântico, utilizou-se um mecanismo de seleção de arquitetura como uma abordagem significativa. Apesar dos aprimoramentos, uma das principais desvantagens é o consumo de muitos recursos computacionais quânticos, que não disponibilizamos nos dias atuais. Portanto, apresentamos uma série de melhorias na proposta, desde carregar os dados usando uma sobreposição em vez de uma codificação de base original, até a utilização de menos Qubits e menos profundidade do circuito quântico proposto. Além disso, também mudamos o processo de treinamento, substituindo a custosa pesquisa de Grover por otimização de gradiente descendente, fazendo com que possamos treinar não somente uma RNBQ mas sim um Comitê de Classificadores dentro de um sistema Quântico, assim, diminuindo o número de operações no Computador Quântico. Finalmente, mostramos que é possível obter um modelo geral melhor do que utilizar o algoritmo de Grover.

Palavras-chave: Computação Quântica, Aprendizado de Máquina, Rede Neurais Artificiais.

Abstract

Deep Neural Networks open several possibilities to solve hard problems. The advancement in the use of quantum computation has allowed us to use quantum features which have no classical counterpart. It has brought forth many algorithms and techniques in the field of quantum machine learning (QML). One of the proposals was a binary quantum neural network (QBNN) which used an amplification method based on the well-known Grover's algorithm search. Besides the fact of being a purely quantum method, it used an architecture selection mechanism as a meaningful approach. Despite its enhancements, one of the main disadvantages is the consumption of a lot of quantum computational resources. Therefore, We present a series of improvements from loading the data using a superposition instead of an original base encoding. Further, we also change the training process, replacing the costly Grover's search with gradient descent optimization. It reduces the quantum computational loss, shrinking the number of operations and qubits. Moreover, we applied the concept of ensemble classification, instead of using a single specific quantum binary weight. Finally we show that it is possible to achieve a general model better using the Grover algorithm. **Keywords**: Quantum Computing, Machine Learning, Artificial Neural Networks.

Sumário

Lista de tabelas

Lista de ilustrações

Lista de abreviaturas e siglas

RNAs	Redes Neurais Artificiais
RBNQ	Rede Neural Binária Quântica
CRBNQ	Comitê de Redes Neurais Binárias Quântica
NISQs	Dispositivos Ruidosos de Escala Intermediária
AM	Aprendizagem de Máquina
RNB	Rede Neural Binária
AMQ	Aprendizagem de Máquina Quântica
DL	Deep Learning

1 Introdução

O desenvolvimento de Redes Neurais Artificiais (RNAs) requer experimentação custosa e que gasta muito tempo computacional para ajustar a arquitetura de redes neurais para um conjunto de dados específico (1). Com o intuito de melhorar o aprendizado de máquina, a busca de arquiteturas em RNAs tenta otimizar sua arquitetura algoritmicamente (2). Nos últimos 20 anos (3) a comunidade científica vem fazendo uso de meta-heurísticas para selecionar o resultado de arquiteturas de RNAs. Mais recentemente, várias estratégias foram propostas para otimizar modelos de Aprendizagem Profunda (DL, do inglês Deep Learning) (4).

A topologia das RNAs é definida pela maneira que os neurônios se conectam e se comunicam (2). Os grafos que representam RNAs tradicionais são geralmente representados em camadas e não têm conexões que se comunicam com camadas anteriores. Alguns problemas são possíveis de ser solucionados com estratégias de DL que possuem conexões não diretas entre as camadas. Assim, como existe a possibilidade de aprender partes do problema separadamente, ao coligar essas partes solucionadas, é possível resolver o problema como um todo. Porém, consumindo muitos recursos computacionais e por vezes levando muito tempo para chegar a uma solução aceitável. Como exemplos desses problemas podemos citar o reconhecimento de imagens (5, 6, 7), a detecção de doenças causadas por insetos vetores (8, 9, 10), a resolução de tráfegos em redes (11, 12, 13) entre outros.

Ainda não existe um algoritmo com custo satisfatório que consiga aprimorar o aprendizado e meta-aprendizado de RNAs de alta complexidade. Outro problema que encontramos nas estratégias de busca são as sementes de aleatoriedade utilizadas na busca de hiperparâmetros, que geram um impacto direto na busca dessas arquiteturas (14). Visto que o treinamento de uma RNA tradicional já é uma tarefa difícil, treinar um Comitê de Classificadores se torna um problema mais complexo e mais custoso dado que a quantidade de otimizadores aprimorados é maior por ser um conjunto capaz de aprender partes específicas de um problema (15).

A Computação Quântica (16, 17) é um campo de pesquisa crescente (18), e Algoritmos Quânticos (AQ) podem resolver certos problemas de forma mais eficiente do que algoritmos clássicos bem conhecidos. Como exemplo disso temos o algorítmo de Grover que é um algoritmo quântico de busca que encontra com alta probabilidade uma entrada exclusiva para uma função

caixa preta. O algoritmo de Grover produz um valor de saída específico, usando apenas avaliações na ordem de $\mathcal{O}(\sqrt{N})$, onde N é o tamanho do domínio da função (19). O problema análogo a esse na computação clássica não pode ser resolvido com custo menor que $\mathcal{O}(N)$, pois, no pior dos casos, o padrão procurado pode estar na última posição pesquisada (20).

1.1 Problema de Pesquisa

Neste trabalho, foi proposta uma estratégia para realizar uma busca de arquitetura em um comitê de classificadores utilizando redes neurais clássicas em um dispositivo quântico baseando o ajuste dos pesos das redes através da técnica gradiente descendente, sem dependência da inicialização aleatória dos pesos ou de avaliação prévia da arquitetura. O algoritmo usa circuitos quânticos variacionais (21) para pesquisar um conjunto de redes neurais clássicas com diferentes arquiteturas. A estratégia híbrida utilizada no circuito quântico variacional reduz a profundidade do circuito quântico e aumenta a possibilidade de implementação em dispositivos quânticos ruidosos de escala intermediária (NISQs, Noisy intermediate-scale quantum) no futuro (18).

Foi verificado que também é possível realizar a seleção da arquitetura com dados de entrada em superposição. Os experimentos mostram a vantagem de tempo do método proposto sobre o uso de uma busca quântica. Os computadores quânticos ainda não são uma realidade e a simulação de circuitos quânticos tem um custo exponencial (22). Tendo esse problema em vista, os experimentos foram realizados com dois problemas de baixa complexidade devido ao esforço exigido para a codificação de um bit em um qubit.

1.2 Justificativa

O paralelismo quântico é inerente aos dispositivos quânticos e nos permite calcular a saída de uma função para diferentes valores em superposição com uma única chamada da função. Em (23), o paralelismo quântico é teoricamente usado para avaliar arquiteturas de RNAs sendo utilizado com pesos em uma superposição quântica. O algoritmo de aprendizagem não linear proposto em (24) é usado em (23) e é uma limitação desses trabalhos porque não sabemos se operadores quânticos não lineares são fisicamente possíveis.

Um experimento usando o algoritmo de busca de Grover (25) mostra que é possível

usar a superposição para selecionar redes neurais em um computador quântico. Por outro lado, o algoritmo de Grover (19) nos permite realizar uma pesquisa em um conjunto de dados não estruturados com uma aceleração quadrática (19). Porém, a busca quântica ainda possui um custo exponencial relacionado ao número de qubits do espaço de busca. O procedimento de aprendizagem proposto por Fawaz et al. não escala eficientemente com o número de parâmetros livres. Uma segunda limitação no uso de uma pesquisa quântica é a profundidade do circuito quântico necessário. A busca quântica realizada por Fawaz et al. requer um circuito com alta profundidade e não é adequada para computadores quânticos ruidosos de escala intermediária (NISQ). E sabemos que a busca de arquitetura para RNAs ainda é um problema sem uma heurística eficiente em um dispositivo clássico ou quântico.

1.3 Objetivos

Este trabalho tem como objetivo desenvolver um algoritmo de treinamento para um comitê de redes neurais clássicas executado em um computador quântico que tenha o mesmo custo que o treinamento de um circuito variacional quântico.

Para atingir esse objetivo, objetivos específicos foram definidos, e são:

- Replicar experimento e resultados obtidos em Fawaz et al.;
- Aprimorar o algoritmo desenvolvido por Fawaz et al.;
- Modificar o treinamento baseado em Fawaz et al. utilizando o gradiente descendente ao invés do algoritmo de Grover;
- Possibilitar a utilização de redes neurais quânticas em superposição com o intuito de simular um comitê de redes neurais clássicas;

1.4 Organização dos capítulos

Esta dissertação está organizada da seguinte maneira, o Capítulo 2 apresenta uma breve visão geral dos conceitos necessários para entender este trabalho. O Capítulo 3 demonstra a criação de um rede neural em um circuito quântico passando pelas particularidades que essa rede apresenta e mostrando as possibilidades e melhorias que foram incluídas no trabalho de Fawaz et al.. O Capítulo 4 expõe as modificações necessárias para a criação de um comitê de

redes neurais quânticas, sendo essa a maior contribuição desse trabalho. O Capítulo 5 disserta sobre os resultados alcançados com o método proposto no Capítulo 4 e faz uma comparação com o método apresentado por Fawaz et al.. Por fim, o Capítulo 6 relata os problemas que foram resolvidos neste trabalho e finaliza apresentando possíveis trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo serão definidos os conceitos necessários para compreensão deste trabalho.

2.1 Qubits e Superposição

Bits quânticos ou qubits são representados como $a|0\rangle + b|1\rangle$, onde a e b são números complexos, além disso, $|0\rangle$ e $|1\rangle$ são vetores:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.1}$$

Temos a e b como amplitudes de probabilidade associadas ao qubit. Quando as amplitudes não são nulas dizemos que o qubit está em *superposição*, isso significa que ambos os estados $|0\rangle$ e $|1\rangle$ são representados simultaneamente. A superposição de n qubits também é possível (26, 27).

2.2 Quibit encoding

Neste trabalho foi utilizado o conceito de codificação na base para carregar os dados, então cada exemplo de entrada será representado pelo vetor $\boldsymbol{x}_j$, podendo ser mapeado para um estado quântico $|\boldsymbol{x}_j\rangle$, onde $\boldsymbol{x}_j = (b_1, \ldots, b_N)$ com $b \in \{0, 1\}$ e j é o índice de exemplo.

Um conjunto de exemplos de dados $M = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ pode ser codificado em $\lceil \log_2(M) \rceil + N$ qubits como

$$|\psi_1\rangle = \frac{1}{\sqrt{M}} \sum_{j=1}^{M} |\boldsymbol{x}_j\rangle \otimes |j\rangle \,. \tag{2.2}$$

Cada subespaço tem amplitude de probabilidade igual $a_j = 1/M \forall j$, então podemos reescrever Eq. (2.2) de uma maneira mais conveniente como

$$|\psi_1\rangle = \sum_{j=1}^{M} \sqrt{a_j} \, |\boldsymbol{x}_j\rangle \otimes |j\rangle \,. \tag{2.3}$$

2.3 Medições de Qubits

Tendo dois bits, na abordagem clássica, somos capazes de criar quatro estados, $00, 01, 10, 11$. Da mesma forma, dois qubits têm quatro estados possíveis, $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Esses dois qubits também podem existir como uma combinação de todos os estados, que chamamos de superposição. Para definir este estado, temos a seguinte equação:

$$|x\rangle \ = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle. \tag{2.4}$$

Nesse caso, o resultado da medição para o estado $|x\rangle$ pode ser $00, 01, 10$ ou 11 com probabilidade de $|\alpha_x|^2$.

2.4 Portas Lógicas

Em um sistema quântico sem influência de ruídos, portas quânticas são representadas por matrizes unitárias. Uma matriz U é unitária, se e somente se, $U \cdot U^\dagger = I$ onde $U^\dagger$ é a combinação da operação de transposição e conjugação da matriz U. Algumas operações requerem apenas um qubit (*operação de qubit único*) como o NOT (Pauli-X) e Hadamard (H) (26).

A porta X transforma o qubit de $|0\rangle$ para $|1\rangle$ e vice-versa. Outro exemplo do mesmo tipo de operação é a porta H, que pode transformar um qubit em uma superposição de estados diferentes. Esta operação transforma o estado $|0\rangle$ em $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ e o estado $|1\rangle$ em $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Também é importante notar que uma operação Hadamard pode ser aplicada a mais de um estado ao mesmo tempo e a operação é representado por $H^{\otimes n}$.

$$Hadamard = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Devemos também citar operações que recebem o nome de operação parametrizada e como exemplo temos $U_1(\lambda)$ (28, 29, 16).

$$U_1(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}$$

Dentre as operações parametrizadas, é necessário citar a operação $R_y(\theta)$ que é uma operação que rotaciona o qubit em um ângulo θ qualquer.

$$R_y(\theta) \equiv e^{-i\theta\frac{Y}{2}} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

Além destes exemplos, outros tipos de operação requerem dois ou mais qubits como a *operação controlada*. Um exemplo de operação controlada temos o $CNOT$ que tendo o estado $|xy\rangle$ o segundo estado $|y\rangle$ (destino) é alterado se o primeiro qubit $|x\rangle$ (controle) for $|1\rangle$ (26).

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.5 Algoritmo de Grover

Para explicar o algoritmo de Grover, considere o problema de buscar um conjunto de índices específicos em um array de banco de dados não ordenado. Uma solução natural é iterar em cada índice até encontrar o item procurado (16). Consequentemente, o custo envolvido em tal operação é $\mathcal{O}(N)$, onde N é o número de índices no array. Este problema de busca inspirou a criação do algoritmo de Grover (16). Ele nos permite encontrar um conjunto de soluções M em nosso banco de dados com custo na ordem de $\mathcal{O}(\sqrt{N/M})$ (16). A ideia por trás da busca de Grover é usar os recursos da Computação Quântica (16) para otimizar o custo dessa busca.

2.5.1 Oráculo

Para entender o algoritmo de Grover, considere um oráculo quântico capaz de reconhecer um padrão que buscamos e marcá-lo para que possamos identificá-lo posteriormente. Ao contrário de saber a resposta antes de inicializar a busca, ele só pode identificar a solução quando passa por ela. Constantemente as operações de oráculo são definidas usando a função clássica dada pela equação 2.5, onde f recebe uma entrada n de bits e retorna uma saída m de bits.

$$f : \{0,1\}^n \to \{0,1\}^m \tag{2.5}$$

A operação quântica para realizar essa tarefa usa um qubit auxiliar. Este qubit auxiliar será utilizado pelo oráculo para marcar a solução que procuramos no banco de dados. O oráculo

inverterá o estado quântico auxiliar marcando a solução. O estado inicial auxiliar é dado por $\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Logo temos:

$$|x\rangle|a\rangle \xrightarrow{O} |x\rangle|a\rangle \oplus |f(x)\rangle \tag{2.6}$$

$$|x\rangle\frac{|0\rangle-|1\rangle}{\sqrt{a}} \xrightarrow{O} (-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \tag{2.7}$$

Como discutimos antes, a função $f(x)$ gera 1, se encontrarmos a solução e 0 caso contrário. Portanto, quando a combinação da busca é encontrada, o estado $|x\rangle$ mudará e marcará a combinação que pretende-se encontrar. Sendo assim conseguimos verificar se o registro se encontra ou não em nossa base de dados.

2.5.2 Procedimento de Grover

Suponha que estejamos procurando m índices específicos em um banco de dados de n índices no total. Os índices são conjuntos binários de uma quantidade x de bits. O algoritmo de Grover é uma sequência de operações que nos permite encontrar o índice de dados que está sendo buscado dentro de uma base de dados qualquer. Logo, para exemplificar, considere também que a nossa base de dados está compreendida da seguinte maneira:

Índice	*Data*
$\|000\ldots0\rangle$	$data_0$
$\|000\ldots1\rangle$	$data_1$
$\vdots$	$\vdots$
$\|111\ldots1\rangle$	$data_n$

Tabela 1 – Esta tabela descreve o array de dados no qual realizaremos a busca dos índices m.

Por fim, seguimos os seguintes passos.

- **Etapa 1:** O primeiro passo é criar uma superposição dos estados iniciais. Considerando $N = 2^n$, obteremos todos os índices possíveis de nosso banco de dados de array. Essa operação é possível usando uma porta Hadamard para n qubits (ver subseção 2.4), onde n é o número de qubits (16). Além disso, inicializamos o qubits auxiliar no estado $|1\rangle$ e, portanto, aplicamos um Hadamard (veja a Figura 1).

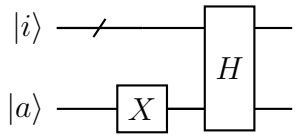


Figura 1 – Preparação dos estados para a amplificação do Grover.

- **Etapa 2:** Em seguida, aplicamos a função oráculo que inverterá o sinal do índice dos dados que estão sendo buscados (veja a Subseção 2.5.1).

- **Etapa 3:** Por fim, o passo seguinte consiste em realizar uma inversão sobre a média. Veja a Figura 2 para entender o procedimento de inversão sobre a média. Finalmente, temos uma operação Hadamard no final do nosso procedimento (16).

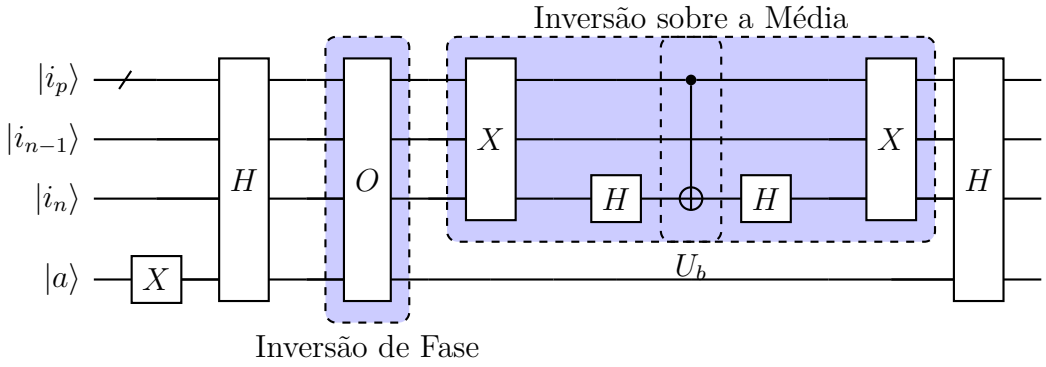


Figura 2 – Aplicação da Inversão de fase e da Inversão sobre a média para resultar na amplificação dos qubits da busca. p representa os índices dos qubits, onde $p = \{0, 1, 2 \ldots, n-2\}$. O é definido como uma operação que demarca o padrão que está sendo buscado na amplificação. U_b é definido como uma operação multicontrolada.

- **Etapa 4:** Repetir a inversão de fase e a inversão sobre a média N vezes, onde o N_{ideal} é dado pela Eq. 2.8.

$$N_{ideal} = \lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} - \frac{1}{2} \rfloor \tag{2.8}$$

2.6 Gradiente Descendente

O objetivo de treinar uma Rede Neural Artificial (RNA) é adaptar seus pesos para produzir a resposta correta. Uma das muitas técnicas existentes consiste no uso de gradiente descendente (30). O objetivo principal é minimizar a função de perda, que nos diz a que distância estamos de nosso objetivo final. Essa proposta deu origem a muitos algoritmos que prometem fornecer uma otimização para a função de perda de qualquer RNA (30). Um deles

é a estimativa de momento adaptativo (Adam). É um otimizador de gradiente descendente estocástico eficiente (31) que requer menos uso de memória. Além disso, ele só precisa de um gradiente de primeira ordem e também é adequado para dados de gradiente esparsos. Tem sido eficiente em ajudar as redes neurais binárias a resolver problemas (30). Outro otimizador que trouxe melhorias para a otimização de dados esparsos foi o otimizador Adagrad (32). O Adam e o Adagrad superam os modelos que não utilizam técnicas adaptativas e também se mostraram úteis para resolver o problema de encontrar valores de rotações para servir de entrada para o circuito variacional.

2.7 Circuitos quânticos variacionais

Um circuito quântico é uma função onde um conjunto de operações quânticas irá gerar uma saída esperada (33). Em um caso particular, ele também pode ser alimentado com parâmetros de variáveis contínuas (33, 34). Essa variáveis determinarão como seus operadores agirão no estado quântico de entrada. Consequentemente, conforme mudamos os valores dos parâmetros, a saída esperada também varia. Os circuitos quânticos que possuem tais características são conhecidos como circuitos quânticos variacionais que mapeiam $f : R^m \rightarrow R^n$ (34). Além disso, $\vec{\theta}$ é um conjunto de parâmetros e $\hat{B}$ é o operador observável que gera o rendimento do valor esperado pelos circuitos após sua operação. Finalmente, $U(\vec{\theta})$ é o conjunto de operadores alimentados com seu parâmetro.

$$f(\theta) := \langle \hat{B} \rangle = \langle 0|U^{\dagger}(\theta)\hat{B}U(\theta)|0\rangle \tag{2.9}$$

É possível encontrar pesquisas (35, 36) que têm trabalhado na aplicação dessa abordagem no desenvolvimento de redes neurais quânticas. O circuito quântico variacional considera os parâmetros como os pesos e então os otimiza para produzir a saída esperada (36). Este trabalho descreve como essa otimização é realizada com mais detalhes no Capítulo 4, enquanto descreve como o Comitê de Redes Neurais Binárias funciona.

3 Redes Neurais Binárias Quânticas

Uma Rede Neural Binária (RNB) é equivalente a um RNA, mas usa números binários em vez de números reais como pesos, entradas e saídas. Esse modelo pode ser usado para treinar Redes Neurais Profundas com recursos computacionais limitados (37). Também pode ser usado para classificação com menos recursos do que RNAs (38). Nesta seção é descrita uma Rede Neural Binária Quântica (RNBQ) (25), como realizar o aprendizado e como fazer a busca dos hiper-parâmetros que devem ser definidos na arquitetura dessa rede.

3.1 Neurônio Quântico

Pesos, Entradas e Saídas em uma RNBQ são codificados em registradores quânticos $|w\rangle$, $|i\rangle$ e $|o\rangle$ respectivamente. A Figura 3 apresenta um neurônio quântico proposto por Fawaz et al., onde U_a representa a função de ativação Maioria que foi utilzada para gerar a saída da RBNQ (39).

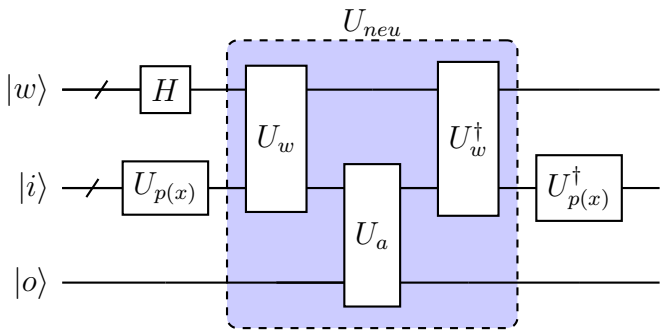


Figura 3 – Representação de um neurônio quântico U_{neu}. Onde, $U_{p(x)}$ carrega os dados clássicos das entradas, sendo p uma entrada do conjunto de dados e x o índice desse conjunto. U_w é a multiplicação dos pesos pelas entradas, que pode ser descrito como a combinação de portas lógicas X e uma operação controlada com o qubit de controle no estado $|0\rangle$ e o qubit objetivo inicializado também no estado $|0\rangle$. Em seguida aplicamos novamente X no qubit de controle para reverter a modificação. U_a é uma operação que utiliza as entradas como qubits de controle e a saída como qubit alvo para mudar o estado do qubit de saída caso a função Maioria seja ativada. Em seguida, $U_{p(x)}^\dagger$ reverte a operação de carregamento para que os qubits possam ser utilizados na próxima execução.

A função Maioria pode ser descrita como uma combinação de operações de controle que transformam o qubit objetivo em $|1\rangle$ quando a maioria dos qubits de controle são $|1\rangle$ ou não muda o estado quando a maioria dos qubits de controle são $|0\rangle$ (veja a Figura. 5).

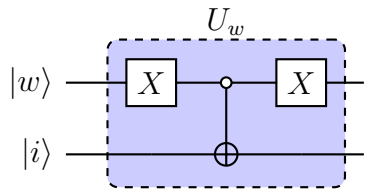


Figura 4 – Representação de U_w para um único qubit.

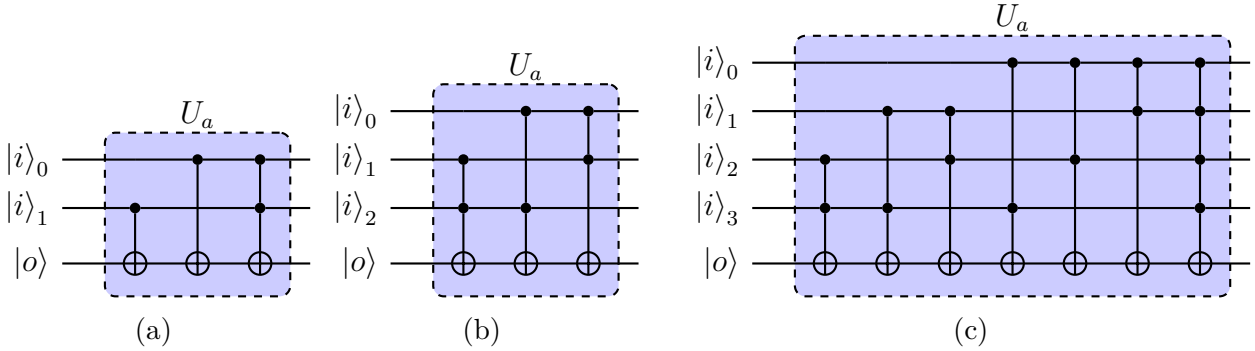


Figura 5 – Representação de U_a aplicado em (a) dois, (b) três, e (c) quatro qubits de entrada.

3.2 Meta-learning

Para selecionar a arquitetura da rede neural o meta aprendizado foi incluído como a adição de M registradores $|m\rangle$ em cada neurônio (veja a Figura. 6). Os registradores $|m\rangle$ podem assumir uma superposisão de 2^M estados possíveis. Cada estado indica uma arquitetura diferente assumida por cada neurônio (com cada caso particular de pesos) no qual será ativado quando os qubits de controle do meta aprendizado estiverem no estado $|1\rangle$ (40).

Desta forma, se $M = 2$ e os registradores do meta aprendizado estiverem no estado $|00\rangle$ será ativado o primeiro estado da arquitetura do circuito criado, assim como se o circuito estiver no estado $|01\rangle$, $|10\rangle$, ou $|11\rangle$ será ativado o segundo, terceiro ou quarto estado da arquitetura respectivamente.

A técnica utilizada torna possível ter todas as arquiteturas em superposição (veja Figura 6). O conjunto de 2^M neurônios de meta aprendizado representados por $U_{neu_1}, \ldots, U_{neu_{2^M}}$ podem ser codificados em $M + N$ qubits como:

$$|\Psi\rangle = \frac{1}{\sqrt{2^M}} \sum_{m=0}^{2^M-1} |U_{neu_m}\rangle |m\rangle , \tag{3.1}$$

onde N é o número de arquitetura possíveis dos neurônios.

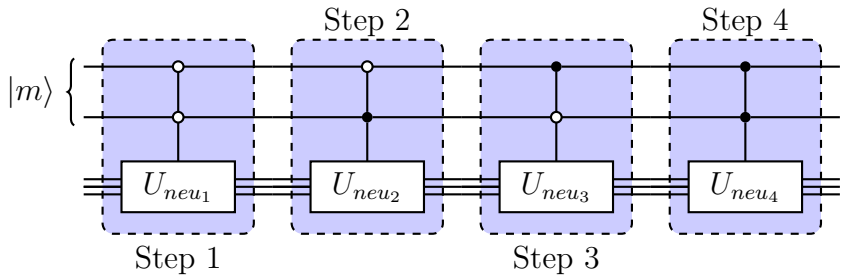


Figura 6 – Superposição dos quatro possíveis estados do neurônio que contém 2 qubits de entrada. U_{neu_1}, U_{neu_2}, U_{neu_3} e U_{neu_4} se diferenciam entre si pelo operador de mutiplicação U_w.

3.3 Circuito que representa uma RNBQ

Usando a definição da representação de um neurônio quântico binário descrito na Seção 3.1, é possível construir um circuito que descreve uma Rede Neural Binária Quântica (RNBQ).

Equivalente a uma Rede Neural Binária (RNB), uma RNBQ pode ter várias camadas, com um número finito de neurônios por camada e cada saída de cada neurônio é propagada adiante para a próxima camada (15). Além disso, uma RNBQ terá quantas saídas forem necessárias para resolver o problema proposto. O caso demonstrado nessa seção tem apenas uma saída, que pode ser 0 ou 1. Além disso, a estrutura criada é maleável e pode ser modificada para conectar ou desconectar cada ligação entre os neurônios e a próxima camada.

Para explicar o processo da RNBQ, é necessário introduzir os três operadores descritos abaixo.

$$
\begin{aligned}
U_f &= \prod_{n=1}^{N_1} U_{neu(1,n)} \left(|w\rangle_{1,n} |i\rangle |o\rangle_{1,n} \right) \\
U_h &= \prod_{l=2}^{L-1} \prod_{n=1}^{N_l} U_{neu(l,n)} \left(|w\rangle_{l,n} \left(\otimes_{m=1}^{N_{l-1}} |o\rangle_{l-1,m} \right) |o\rangle_{l,n} \right) \\
U_o &= U_{neu(L,1)} \left(|w\rangle_{L,1} \left(\otimes_{m=1}^{N_{L-1}} |o\rangle_{L-1,m} \right) |o\rangle_{L,1} \right)
\end{aligned}
$$

onde L é o número de camadas, N_j é o número de neurônios em cada camada. O par de registradores de índices (l, n) são respectivamente a camadas e o neurônio associado ao registrador. Usando esses operadores, a porta U_{net} representa um passo feed-forward (15) do processo da RNBQ definida na Eq. 3.2.

$$U_{net} = U_f^{-1} U_h^{-1} U_o U_h U_f \tag{3.2}$$

U_f representa a camada de entrada da rede, U_h representa as camadas entre a de entrada e a de

saída, e U_o a última camada com um único neurônio. U_f e U_h são invertidos para que possâmos reutilizar os qubits.

A Figura 7 apresenta uma RNBQ com 3 neurônios divididos em 2 camadas. O operador U_{net} define como um passo do feed-forward é feito. A Figura 9 representa uma época da RNBQ onde U_f é o passo feed-forward para todo o conjunto de dados utilizando meta aprendizado. Para gerar o circuito com meta aprendizado foi necessário inserir os qubits do meta-aprendizado em superposição utilizando uma porta H junto com os qubits de pesos, sendo assim foi preciso modificar o operador U_w para aceitar os novos qubits na operação de multiplicação. Dessa forma, o circuito está preparado para treinar todas as possíveis arquiteturas desta RNB.

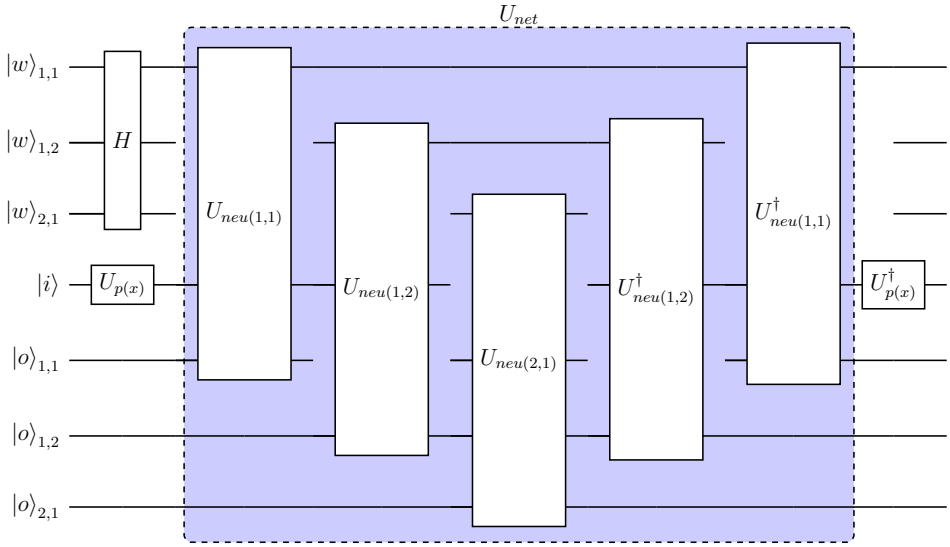


Figura 7 – Representação de como a rede gera uma saída. Neste caso, U_{net} representa o passo feed-forward para uma entrada específica.

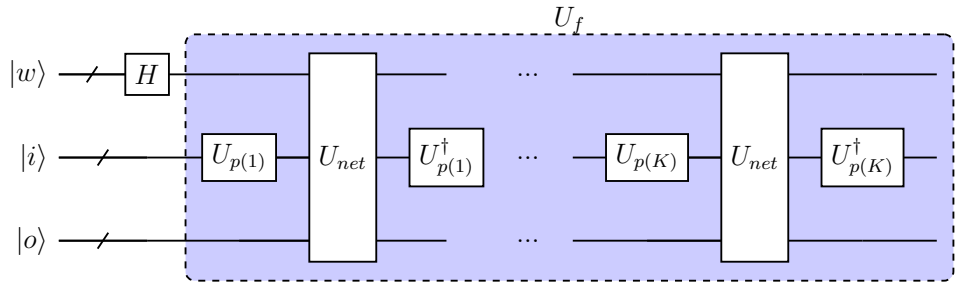


Figura 8 – Representação de um passo de feed-forward U_f para todas as amostras K, ou seja, uma época da Rede Quântica.

3.3.1 Treinamento da RNBQ

Após uma época da RNBQ, a precisão total da rede pode ser calculada com base no valor de saída. Um operador U_o é definido como um oráculo para marcar os qubits de

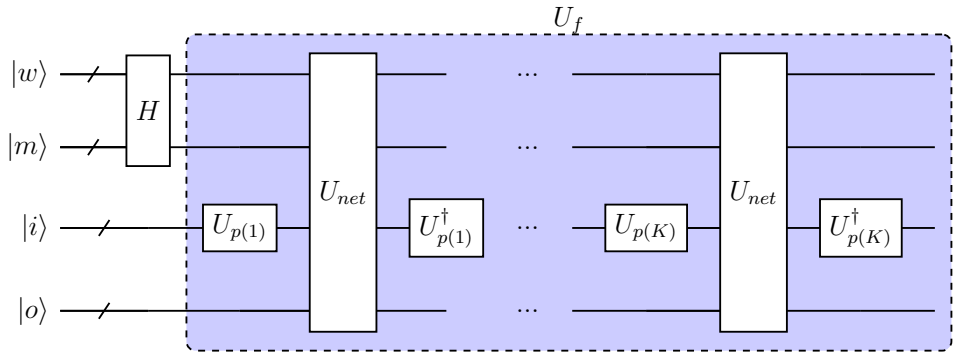


Figura 9 – Representação de uma etapa feed-forward U_f com meta-aprendizagem para todas as amostras K, ou seja, uma época da Rede Quântica.

saída com uma porta X onde seus valores esperados são 0, em seguida fazemos uma operação multicontrolada para verificar se alcançamos 100% de respostas corretas. Além disso, aplicamos novamente uma porta X para reverter a modificação onde os valores de saída deveriam ser 0, concluindo a inversão de fase desta amplificação. Por fim, fazemos uma inversão sobre a média definida por U_m para buscar a melhor combinação.

Como dito antes, o processo de avaliação usa o algoritmo de amplificação de Grover que amplifica a melhor combinação de qubits para os pesos e qubits que representam o meta-aprendizagem da RNBQ. Este processo aumenta a probabilidade de boas combinações de pesos e ligações de meta-aprendizagem permanecerem, enquanto diminui a probabilidade de amplitude de combinações que não retornam as saídas corretamente. A Figura 10 mostra o circuito criado para treinar um QBNN.

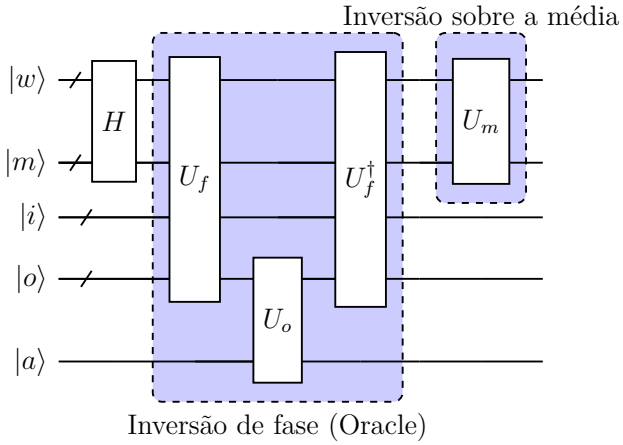


Figura 10 – Representação das etapas para treinar e meta-treinar a RNBQ, usando abordagem de amplificação. A Inversão de Fase e Inversão Sobre a Média devem ser aplicadas até atingirmos a precisão que nos propomos alcançar. O registro $|a\rangle$ armazena a precisão total da RNBQ.

3.3.2 Rede Neural Binária Quântica

Para auxiliar no entendimento dessa proposta de melhoria, é necessário recriar o exemplo de Fawaz et al.. Os problemas propostos por Fawaz et al. consistem nas funções $Sinal(b_1 * b_3 + b_2)$ e $Sinal(b_1 + b_2 + b_3)$, dada uma entrada binária de comprimento 3 (b_1, b_2, b_3) com $b \in \{1, -1\}$. Ambas as funções retornam 1 se o resultado for igual ou maior que 0 e -1 caso contrário. Os valores de entrada clássicos 1 e -1 são mapeados para os estados quânticos $|0\rangle$ e $|1\rangle$, respectivamente.

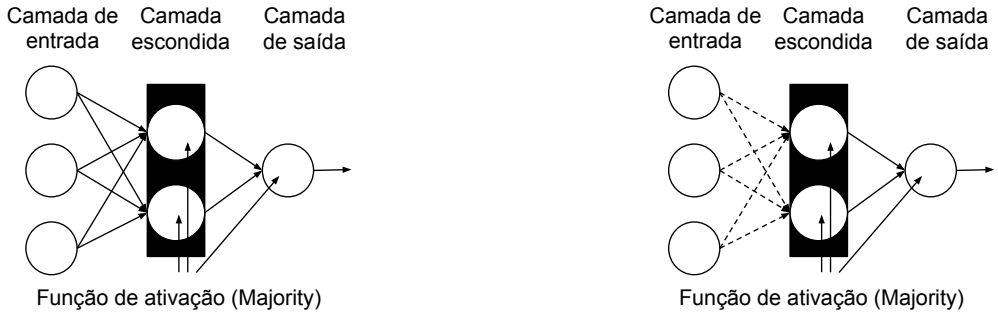
A RNBQ proposta por Fawaz et al. contém três camadas de neurônios quânticos. Portanto, usamos a seguinte quantidade de qubits para construir a RNBQ:

1. Seis qubits para entradas, três para cada neurônio da primeira camada (Fawaz et al. não reutiliza os qubits de entrada);
2. Oito qubits para pesos, sendo 6 para a primeira camada e dois para a segunda;
3. Dois qubits para armazenar as saídas da camada oculta;
4. Oito qubits para salvar a saída de cada passo de feed-foward da RNBQ;
5. Um qubit para armazenar a precisão total da rede.

Isso resulta em vinte e cinco qubits para o treinamento da RNBQ. Para um treinamento da RNBQ com meta-aprendizagem, devemos incluir seis qubits extras para os registradores de meta-aprendizagem controlar os pesos entre as camadas de entrada e a camada oculta, resumindo trinta e um qubits. A representação da arquitetura de meta-aprendizagem está na Figura 11b.

A primeira camada carrega os dados de entrada usando três qubits. A função U_p é representada na Figura 3. A segunda camada contém dois neurônios quânticos, cada um com sua função de ativação Maioria. A função de ativação na Figura 5 armazena a saída em um qubit específico para cada neurônio quântico (veja a Figura 7) para ter uma visão completa desta etapa. Então, a terceira camada tem um neurônio quântico que produz a saída do processo de feed-forward da rede.

Para verificar os resultados, invertemos o registrador quântico de saída usando uma porta NOT aplicada onde o resultado esperado é 0. Assim, obtendo um array com oito qubits



(a) (b)

Figura 11 – (a) Esta figura representa a arquitetura da Rede Neural Binária Quântica que será discutida ao longo deste artigo sem meta-aprendizagem. (b) Esta figura representa a arquitetura da Rede Neural Binária Quântica que será discutida ao longo deste artigo com meta-aprendizagem. Os controles de meta-aprendizagem estarão atuando nas linhas tracejadas, representando a possibilidade de ativar ou não a conexão.

marcados como $|1\rangle$ para 100% de precisão[1]. Este processo termina revertendo os qubits de saída usando outras portas NOTs. O operador U_o representa este processo completo (veja a Figura 10).

Por fim, no procedimento de amplificação, já com os qubits de pesos e meta-aprendizagem em sobreposição, aplicamos a inversão sobre a média U_m descrita na Figura 10. Esta amplificação aumenta a probabilidade de uma arquitetura adequada ao problema, habilitando e desabilitando a conexão entre os neurônios quânticos conforme necessário. Dessa forma, amplificamos a amplitude de probabilidade dos estados associados ao conjunto de parâmetros que dão um resultado com maior acurácia para a saída do RNBQ.

A RNBQ proposta por Fawaz et al. pode ser modificada para usar menos recursos de Computação Quântica (qubits e profundidade do circuito). O principal problema desta proposta é que precisamos saber com antecedência a precisão máxima que o conjunto de dados pode alcançar. Além disso, assumimos que podemos resolver o problema com uma RNBQ. Na Seção 3.4 mostramos como reduzir os recursos computacionais quânticos necessários. No Capítulo 4 apresentamos uma abordagem que elimina a necessidade de conhecer a precisão máxima de antemão.

[1] Podemos realizar esta tarefa usando uma operação multi-controlada, veja (29).

3.4 Aprimoramento da RNBQ

Para minimizar os recursos computacionais quânticos usados pela proposta de Fawaz et al., foi possível remover algumas etapas. Em primeiro lugar, o método de carregamento de dados foi alterado. Ao invés de carregar a entrada e os pesos separadamente, a entrada foi carregada diretamente nos qubits do registrador de pesos (veja a Figura 12). Consequentemente, a função Maioria foi aplicada usando os pesos qubits como os de controle. Observe que o grupo chamado U_{neu} foi modificado e agora consiste apenas no operador U_a. Dessa forma, foi necessário apenas aplicar a função de ativação para obter a saída para uma entrada. A partir daqui, o operador U_a é idêntico ao operador U_{neu}, mas entendendo que U_{neu} tem apenas uma operação.

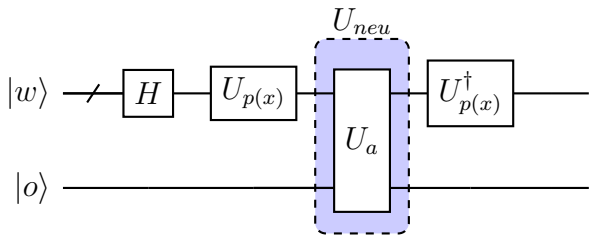


Figura 12 – Representação da forma reduzida do Neurônio. Observe a ausência de U_w e $U_w^\dagger$. Os dados de entrada são carregados no mesmo registrador dos pesos, realizando a multiplicação.

Além disso, a maneira como é carregada a saída na próxima camada foi adaptada. Em vez de armazenar os valores de saída em um registrador dedicado, eles são carregados diretamente no registrador de pesos do último neurônio usando uma operação multi-controlada, diminuindo o uso de qubits (veja a Figura 13) . Consequentemente, é necessário reverter as modificações dos pesos antes do procedimento de amplificação, preservando apenas a saída final do passo feed-forward (Eq. (3.3)).

$$U_h = \prod_{l=1}^{L-1} \prod_{n=1}^{N_l} U_{neu(l,n)} \left(|w\rangle_{l,n} \left(\otimes_{m=1}^{N_{l+1}} |w\rangle_{l+1,m} \right) \right)$$
$$U_o = U_{neu(L,1)} \left(|w\rangle_{L,1} |o\rangle_{L,1} \right)$$

$$U_{net} = U_h^{-1} U_o U_h \tag{3.3}$$

Foi possível diminuir mais um qubit (o qubit de precisão) e medir todos os qubits de saída para acompanhar como a precisão evolui em cada etapa de amplificação. Usar um qubit para verificar a precisão total do RNBQ é melhor para reduzir o erro de medição, mas não seria possível acompanhar a evolução da precisão sem medir os qubits de saída da rede.

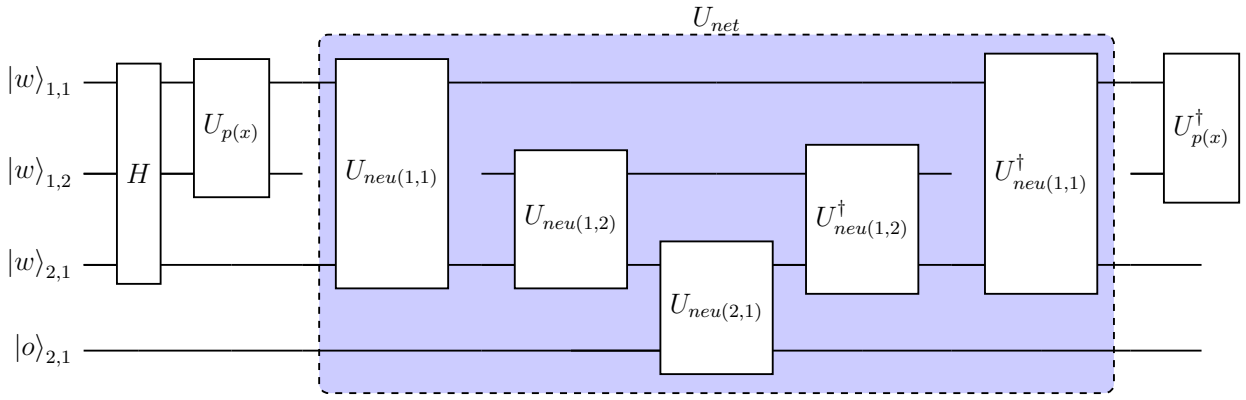


Figura 13 – Representação de como obter uma saída da rede reduzida.

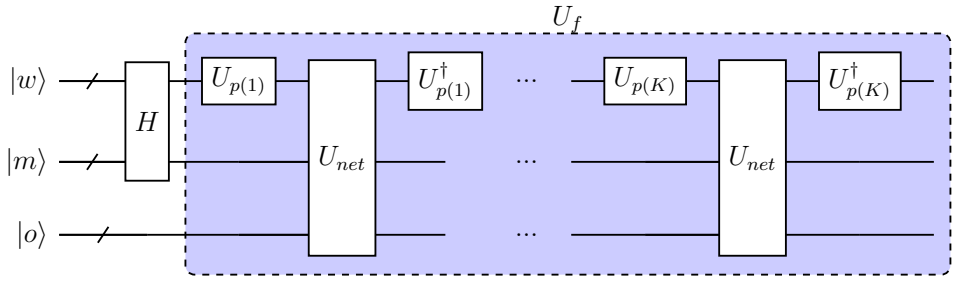


Figura 14 – Representação de uma época da RNBQ Reduzida. Aqui, U_f realiza o feed-forward para todas as entradas da rede e depois desfaz a época da RNBQ.

O circuito apresentado na Figura 14 é uma versão melhorada do circuito apresentado na Figura 9, ambos produzem o mesmo resultado para uma entrada específica. A Figura 15 mostra o exemplo ilustrado na Figura 10, porém com as melhorias.

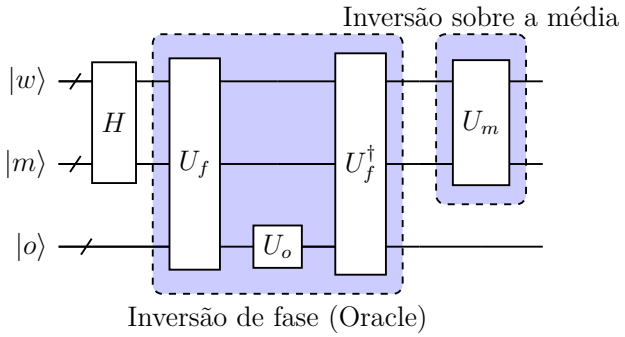


Figura 15 – Representação das etapas para treinar e meta-treinar a RNBQ reduzida, usando a mesma abordagem de amplificação. A "Inversão de Fase" e a "Inversão Sobre a Média" devem ser aplicadas até atingirmos uma precisão aceitável. Neste caso, podemos ver a precisão evoluindo ao medir todas as saídas.

Como prova de conceito, foram utilizados os mesmos dois problemas descritos no paper (25) implementando o código usando Qiskit (29). Após rodar o circuito com e sem meta-aprendizagem, os mesmos resultados foram encontrados em (25). Os problemas propostos um e dois atingem a precisão máxima após seis e cinco amplificações, respectivamente. Percebeu-

se que cada vez que medições são realizadas e não é possível atingir a precisão esperada, é necessário executar todo o algoritmo de treinamento a partir da primeira iteração com mais uma etapa de amplificação de Grover para obter o próximo resultado devido ao problema inerente ao procedimento da RNBQ (25). O problema de não saber a precisão máxima da RNBQ ainda não foi resolvido com essas melhorias. No Capítulo 4 mostramos como resolvemos este problema. Veja os resultados da RNBQ reduzida e treinada usando a técnica de meta-aprendizagem para o primeiro problema na Figura 16 e para o segundo problema na Figura 17.

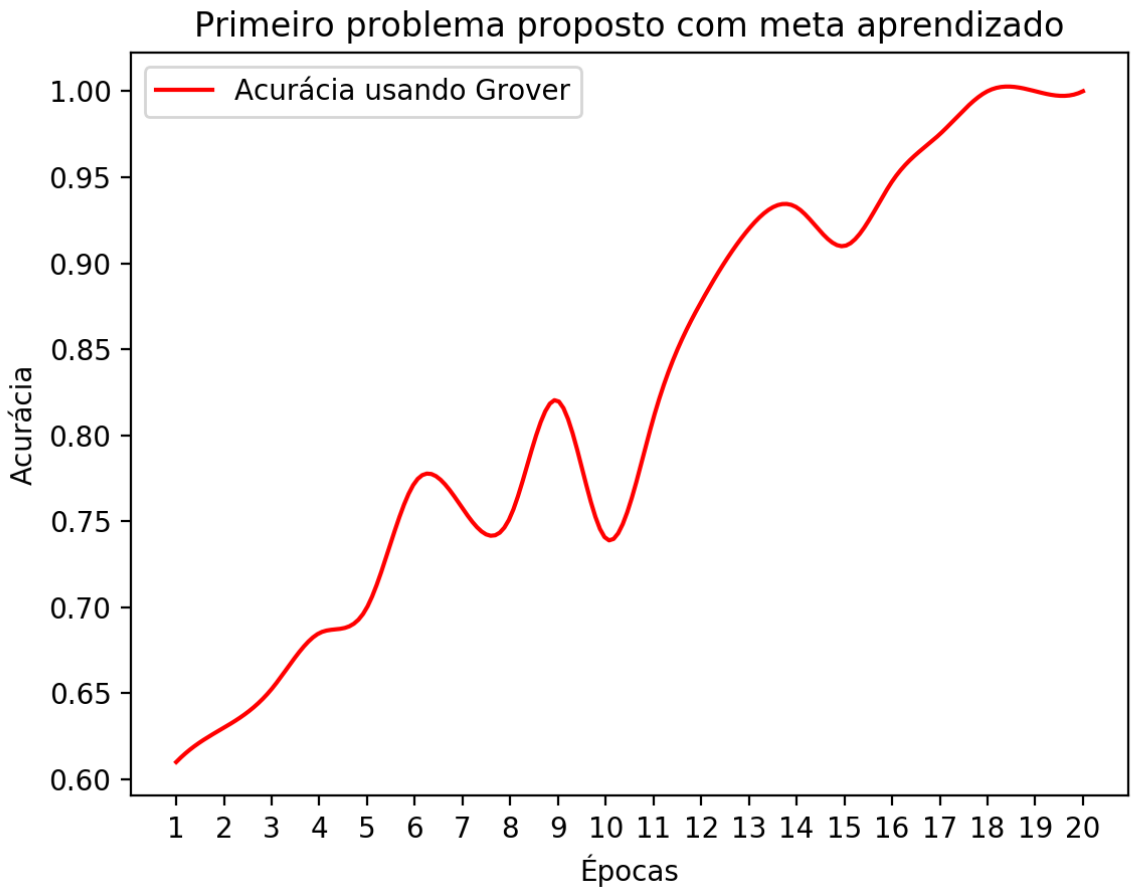


Figura 16 – Acurácia utilizando o método de amplificação baseado no Grover para o primeiro problema proposto.

O problema inerente de não saber a precisão máxima que pode ser alcançada pela RNBQ ainda não foi resolvido neste Capítulo com essas melhorias. No Capítulo 4 mostramos como este problema será resolvido.

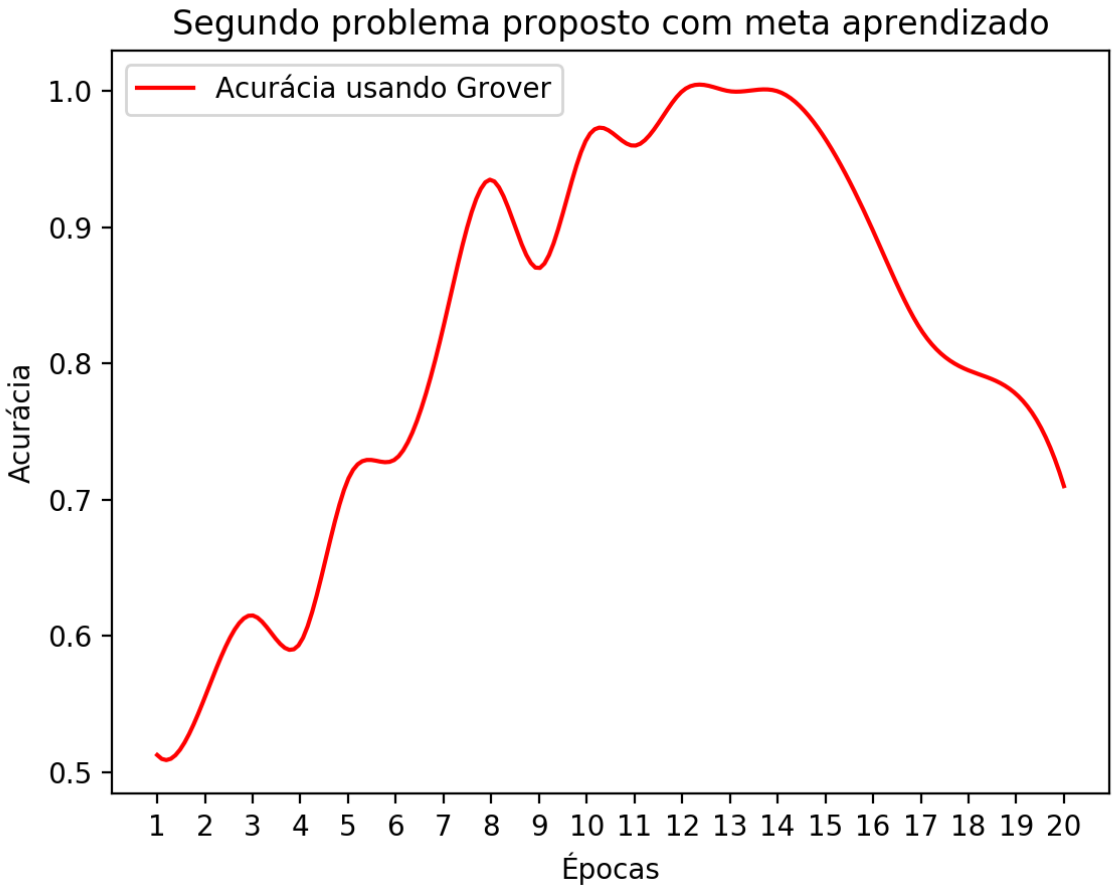


Figura 17 – Acurácia utilizando o método de amplificação baseado no Grover para o segundo problema proposto.

4 Comitê de Redes Neurais Binárias Quânticas

Um Comitê de Classificadores (15) pode produzir melhores resultados do que técnicas de inteligência artificial que utilizam somente um modelo de classificação (41, 42, 43). Quando combinamos classificadores, estes podem aprender diferentes áreas do conjunto de dados, permitindo que eles alcancem um desempenho mais satisfatório em relação à técnicas tradicionais (43). Neste Capítulo, trazemos a mesma ideia para a Computação Quântica. Consideramos algumas modificações da rede binária quântica descrita em (25), introduzindo duas novas abordagens. Ao invés de fixar um único conjunto binário de pesos a uma RNBQ, atribuímos uma superposição não uniforme de strings binárias por meio de rotações de qubit parametrizadas, resultando, após o treinamento, em uma variedade de arquiteturas RNBQs viáveis. Essa abordagem nos permite criar um Comitê de Redes Neurais Binárias Quânticas (CRNBQ) em computadores quânticos usando uma estrutura de neurônios semelhante à RNBQ descrita no Capítulo 3 com as melhorias da Seção 3.4. A principal diferença em relação ao trabalho criado por Fawaz et al. é que em cada iteração criamos um conjunto diferente de RNBQs, executamos todas as RNBQs de uma vez, medimos o qubit de saída para comparar com o resultado esperado e atualizamos os parâmetros de rotação por meio de técnicas conhecidas de otimização de parâmetros.

O principal problema encontrado no artigo de Fawaz et al. é a necessidade de conhecer antecipadamente a precisão máxima que a RNBQ pode atingir dado um conjunto de dados. Este problema foi resolvido usando o método CRNBQ, pois a acurácia do processo de treinamento pode ser verificada em cada etapa do algoritmo de aprendizado, sendo assim é possível verificar se o valor de acurácia desejado foi alcançado.

4.1 Neurônio Quântico no CRNBQ

O método desenvolvido nesta seção é uma extensão do proposto na Seção 3.4. Ao invés de usar portas Hadamard para criar uma superposição uniforme em qubits de pesos e de meta-aprendizagem, portas R_y são aplicadas para criar uma superposição controlada, produzindo o conjunto de Redes Neurais Binárias (veja as Figuras 18 e 19). Com esta modificação no procedimento da criação dos neurônios, podemos criar um CRNBQ escolhendo o valor de $\vec{\theta}$ para gerar uma superposição configurável de neurônios.

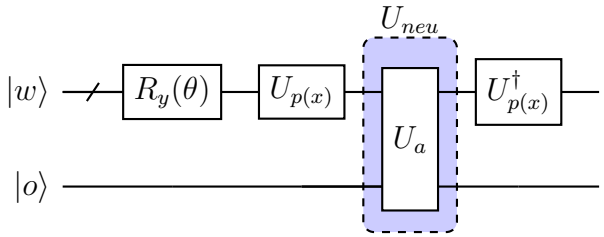


Figura 18 – Representação do neurônio do CRNBQ. Nesta figura, $R_y(\theta)$ representa as rotações feitas para criar a superposição controlada de neurônios.

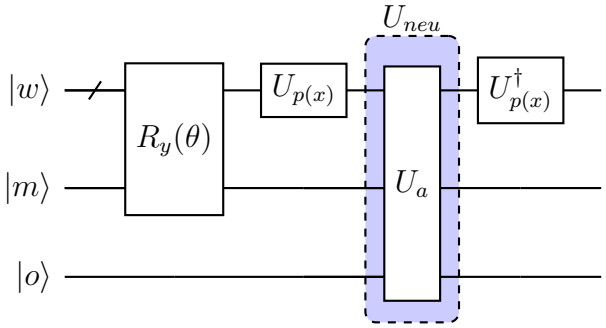


Figura 19 – Representação do neurônio com meta-aprendizado do CRNBQ. Nesta figura, $R_y(\theta)$ representa as rotações feitas para criar a superposição controlada de neurônios.

4.1.1 Carregando as entradas da CRNBQ em superposição

Uma outra abordagem criada, foi a possibilidade de carregar as entradas em superposição, veja as Figuras 20 e 21.

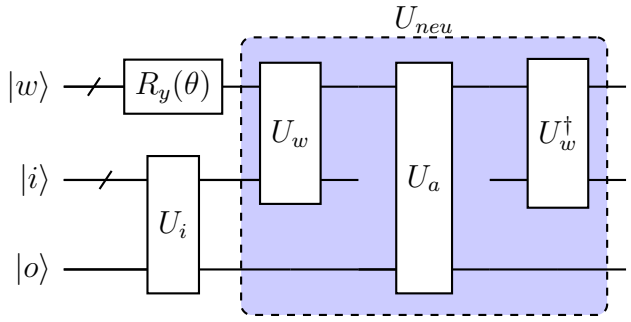


Figura 20 – Representação de neurônio quântico criado para uso no CRNBQ com entradas em superposição. As operações são as mesmas descritas na Figura 3.

Se as entradas estiverem em sobreposição, não é possível usar a melhoria proposta na Seção 3.4. Carregar dados clássicos em uma sobreposição quântica não uniforme é uma tarefa complexa que requer recursos exponenciais, e pode dominar o custo computacional da aplicação quântica (44, 45, 46). Por isso, neste trabalho é utilizada uma superposição uniformemente distribuída de todos os estados da base, sendo facilmente preparada através de operações de Hadamard. Na prática, é preciso criar uma superposição representando os padrões de entrada usando algoritmos de preparação de estados quânticos em (40, 47, 48).

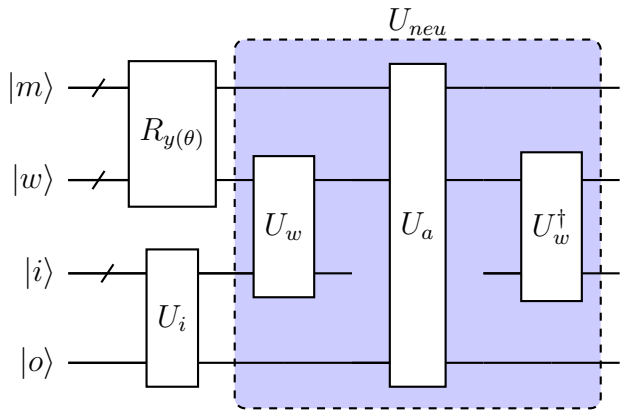


Figura 21 – Representação de neurônio quântico com meta-aprendizado criado para uso no CRNBQ com entradas em superposição. As operações são as mesmas descritas na Figura 3.

As entradas são representadas de forma binária, mapeando os estados $|0\rangle$ e $|1\rangle$ para as entradas 1 e -1, respectivamente. Carregamos cada entrada emaranhada com seu respectivo rótulo de saída usando uma operação CNOT controlada pela entrada.

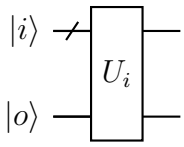


Figura 22 – Representação da inicialização de entrada. Observe que nesta figura, o operador U_i carrega uma superposição de entradas emaranhadas com a saída.

Seguindo a ideia da Seção 4.1, portas $R_y(\theta)$ são usadas para criar uma superposição parametrizada de neurônios. A superposição de neurônios nos permite avaliar todas as respostas possíveis para uma combinação de pesos binários de uma só vez. Para fazer a multiplicação, já que estamos usando as entradas como controles e os pesos como alvo, precisamos usar uma porta NOT para realizar a multiplicação como proposta por Fawaz et al.. Veja as Figuras 20 e 21 para uma representação de um circuito para um neurônio com superposição.

Estas Figuras 18, 19, 20 e 21 representam apenas um neurônio, mas fica claro que combinado com outros neurônios criamos uma camada do CRNBQ. Além disso, reforçamos que a saída de uma camada pode ser direcionada para outros neurônios que representam a próxima camada e fazemos isso para criar uma conexão entre essas duas camadas, até que tenhamos criado nosso circuito que representa o passo feed-forward de uma CRNBQ, como descrito como função U_f nas Figuras 8, 9, 13 e 14.

4.2 Feed-forward do CRNBQ

Nesta seção, descrevemos como criar um CRNBQ. Inicialmente foi criada uma superposição por meio de rotações $R_y(\theta)$ aplicadas aos qubits que representam dados de peso e meta-aprendizagem. Uma matriz binária é carregada no registro de entrada usando codificação básica e um conjunto de rotações R_y cria uma superposição não uniforme de pesos binários controlados por uma matriz de parâmetros de valor real. Finalmente, uma função de ativação, que é definida como a função Maioria (25) (Figuras 5 e 5b), gera a saída para a próxima camada. Para uma visão geral do processo quântico, veja Figura 23 e Figura 24.

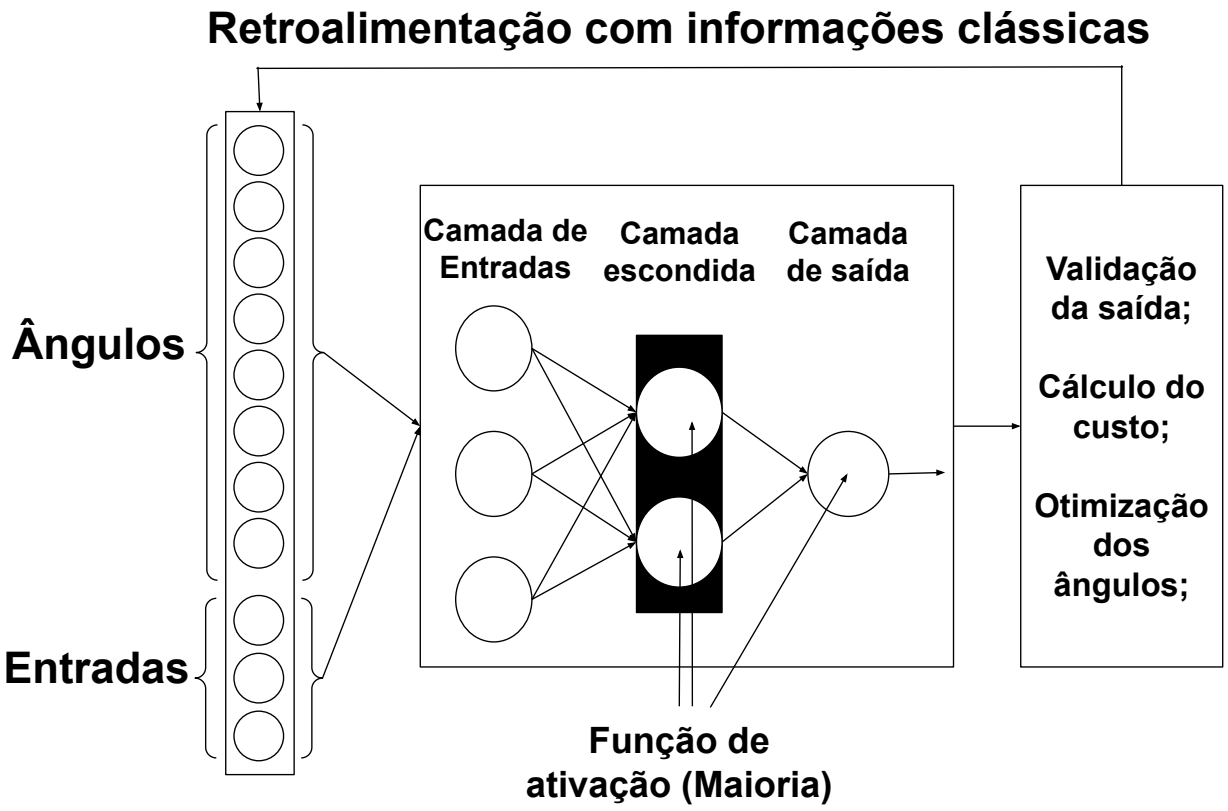


Figura 23 – Arquitetura do sistema sem Superposição das entradas.

A superposição de peso e meta-controladores é criada através de portas $R_y(\theta)$ aplicadas em seus respectivos qubits e controlada pelos parâmetros de ângulo. Para atualizar esses parâmetros, empregamos uma estratégia estocástica de gradiente descendente (SGD) híbrida. As Figuras 23 e 24 ilustram uma iteração de treinamento para uma entrada e uma iteração de treinamento para muitas entradas em sobreposição. Usando todo o conjunto de dados, o método sem superposição otimiza os pesos 2^n vezes para strings binárias de comprimento n. No entanto, a estratégia de superposição otimiza os pesos apenas uma vez porque os dados são carregados

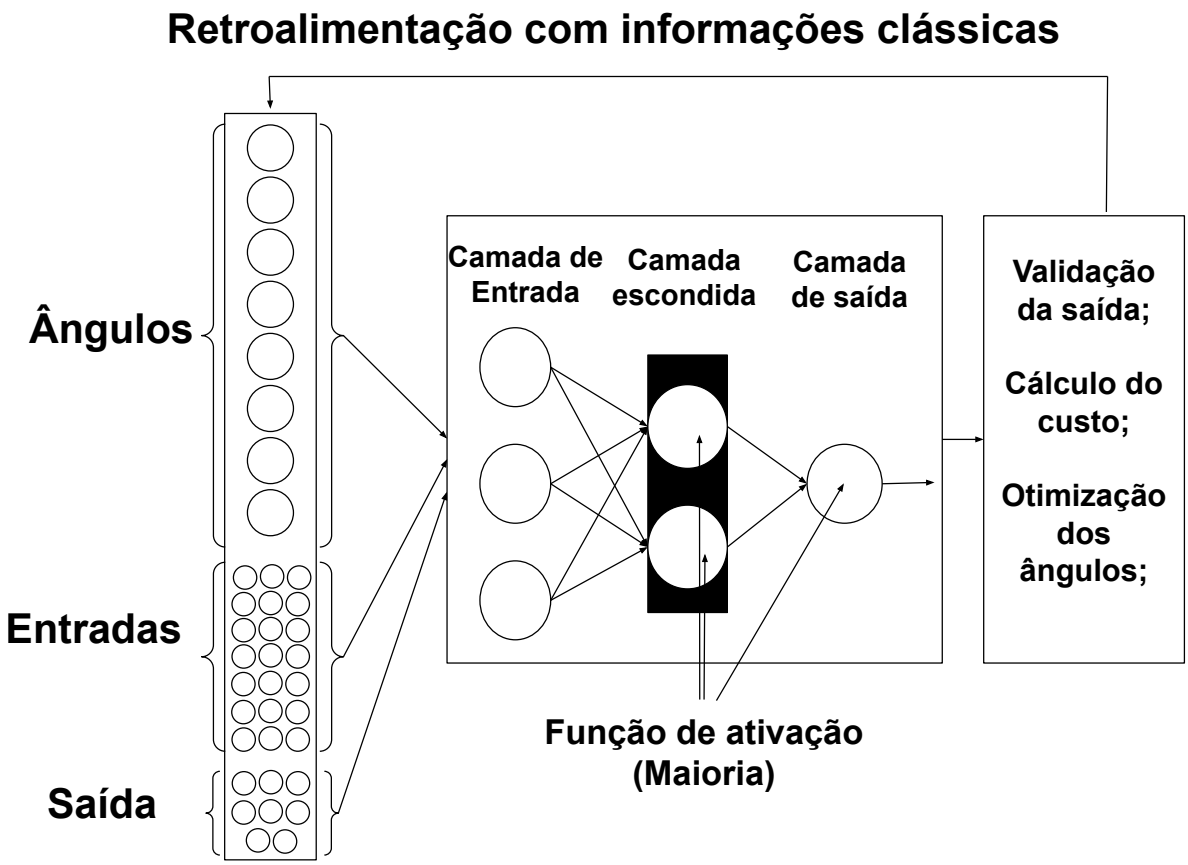


Figura 24 – Arquitetura do sistema com Superposição das entradas.

de uma só vez. Dessa forma, para comparar os dois métodos, devemos executar a abordagem de superposição 2^n vezes. Executamos experimentos que são apresentados no Capítulo 5 usando essas duas abordagens e comparamos o desempenho com o método que utiliza uma modificação da amplificação de Grover.

4.3 Treinamento Híbrido do CRNBQ

Esta etapa considera que o qubit de saída armazena o resultado de precisão da classificação do CRNBQ. No final do feed-forward, o valor esperado da medição do qubit de saída na base Z é comparado com o rótulo esperado usando uma função de perda para calcular o erro, seguido por uma rotina de otimização para atualizar os ângulos das rotações R_y. Nosso objetivo é encontrar um conjunto de ângulos para criar um conjunto capaz de classificar o conjunto de dados corretamente. Realizamos esta atualização usando uma abordagem híbrida combinando a técnica numérica de rede neural clássica SGD (15) com a regra de mudança de parâmetro para avaliar gradientes conforme descrito na Subseção 2.7. O processo é repetido até

que um critério de parada seja alcançado.

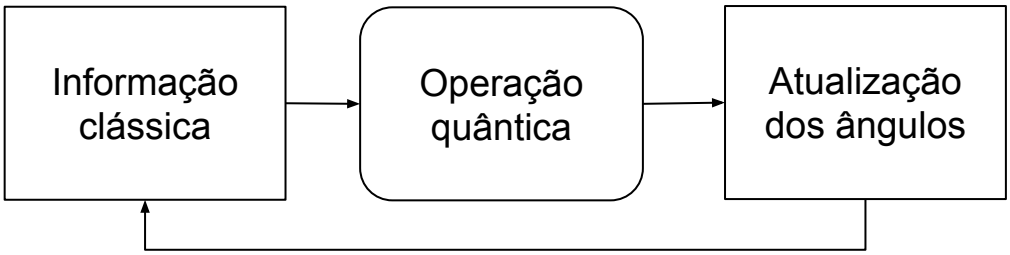


Figura 25 – Descrição do fluxo do treinamento Híbrido do CRNBQ.

Os parâmetros treináveis $\vec{\theta}$ são ângulos de rotação aplicados nos qubits de pesos e de meta-aprendizagem através dos operadores $R_y(\theta)$. Almeja-se alcançar um conjunto de rotações capaz de gerar um resultado para nosso CRNBQ que alcance o mesmo valor ou um valor superior de acurácia alcançado usando a amplificação de Grover. O ajuste desses parâmetros permite a alteração da saída final e minimiza uma função de perda.

A função de perda é definida da seguinte forma: 0 representará a menor perda que podemos obter e números maiores que 0 representarão mais erros. No método sem sobreposição, podemos fazer a comparação diretamente com as saídas definidas. Para o método usando superposição, calculamos o custo utilizando uma função bem conhecida chamada Negative Log Likelihood Loss (NLLLoss) descrita em (49) que obtém a probabilidade da saída estar no estado $|1\rangle$ em vez de um vetor de rótulos, pois estamos carregando os dados em superposição.

O objetivo é otimizar a NLLloss para diminuir o erro entre os ângulos esperados e os ângulos utilizados. Iremos atualizá-los utilizando a técnica gradiente descendente, com os otimizadores Adam (31) e Adagrad (50). Dessa forma, reduzimos a quantidade de recursos quânticos necessários, visto que todo o processo de atualização dos pesos está sendo feito classicamente.

Sabendo que o principal problema encontrado em (25) é que temos que saber o número de operação para amplificar o melhor conjunto de parâmetros, trazemos a solução utilizando o CRNBQ aqui proposto. Com o CRNBQ não precisamos saber quantas iterações temos que executar para obter um resultado satisfatório, seja um limite de operações ou então uma acurácia definida, pois podemos ver o erro associado ao ângulo ao fazer o processo de treinamento usando uma abordagem clássica. Finalmente, é possível descrever esse método como uma técnica híbrida

capaz de treinar e meta-treinar um conjunto de RNBs em computadores quânticos.

5 Experimentos

Tendo em consideração todas as declarações colocadas antes para treinar e meta-treinar o método proposto neste artigo, usamos os mesmos problemas propostos por Fawaz et al.. Eles consistem em dois problemas que dadas entradas (x_1, x_2, x_3) temos as funções definidas abaixo respectivamente do primeiro problema e do segundo problema:

$$y(x_1, x_2, x_3) = sinal(x_3 * x_1 + x_2) \tag{5.1}$$

$$y(x_1, x_2, x_3) = sinal(x_1 + x_2 + x_3) \tag{5.2}$$

Ambas as equações têm a função *sinal* definida como:

$$sinal(x) = \begin{cases} 1, & \text{if x} \geq 0. \\ -1, & \text{caso contrário.} \end{cases} \tag{5.3}$$

O código para os experimentos é implementado usando o framework Pytorch e Qiskit (51). O framework Skorch (52) é usado para uma pesquisa aleatória no intervalo de parâmetros necessários para construir a parte clássica do algoritmo. Esta busca segue a distribuição de parâmetros apresentada na Tabela 2. Esses parâmetros foram escolhidos empiricamente para minimizar a função de perda e aumentar a precisão.

Distribuição de parâmetros	
Taxa de Aprendizado	Distribuição Uniforme entre [0, 0.5]
Épocas	Distribuição Uniforme entre [1, 150]
Otimizadores	[Adam, Adagrad]

Tabela 2 – Distribuição de parâmetro

Primeiro, os dados e pesos são inicializados fazendo uma rotação de $\frac{\pi}{2}$ nos pesos como substituição das portas H e X como proposto por Fawaz et al. (ver Capítulo 4). Em seguida, carregamos os parâmetros θ no circuito quântico usando o operador $R_y(\theta)$. Optamos por iniciar todos os oito pesos com uma rotação de $\frac{\pi}{2}$ para treinar com igualdade os dois modelos junto com a abordagem da amplificação baseada no algoritmo de Grover. Observe que a perda obtida de cada problema não representa diretamente cada acurácia.

Para a análise dos resultados descrevemos aqui como custo o erro associado a resolução dos problemas propostos. Para os experimentos com entradas em superposição, todas as entradas são inicializadas em superposição usando as operações de Hadamard em seguida o feed-forward é executado para todas as entradas. Dessa forma, para comparar os resultados dessa abordagem, é necessário executar a atualização dos ângulos 2^n vezes para um array de entrada com n elementos.

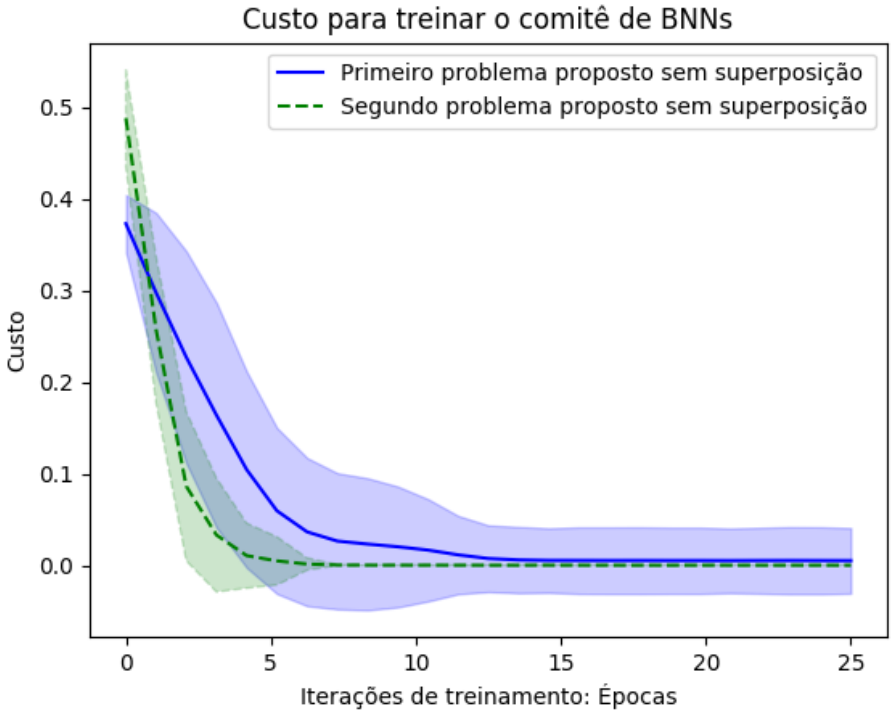


Figura 26 – Otimização da função de custo sem meta-aprendizagem.

As Figuras 26, 27, 28 e 29 representam a função de custo do primeiro problema e do segundo problema (Eqs. 5.1 e 5.2) com e sem meta-aprendizagem. O algoritmo de aprendizado pode otimizar a função de custo para ambos os casos e é capaz de realizar uma busca diferenciável de Arquitetura de Redes Neurais usando gradiente descendente. Ao contrário do algoritmo proposto por Fawaz et al., o CRNBQ permite obter informações intermediárias durante o processo de treinamento, sendo assim, não é mais necessário saber antecipadamente a precisão máxima que o algoritmo consegue alcançar dado o conjunto de dados e tornando possível estabelecer o critério de parada, visto que podemos obter informações intermediárias do processo de treinamento.

As Figuras 30, 31, 32 e 33 apresentam a precisão dos problemas definidos pelas eqs. 5.1 e 5.2 usando a abordagem proposta (com e sem meta-aprendizagem) e o algoritmo de Grover

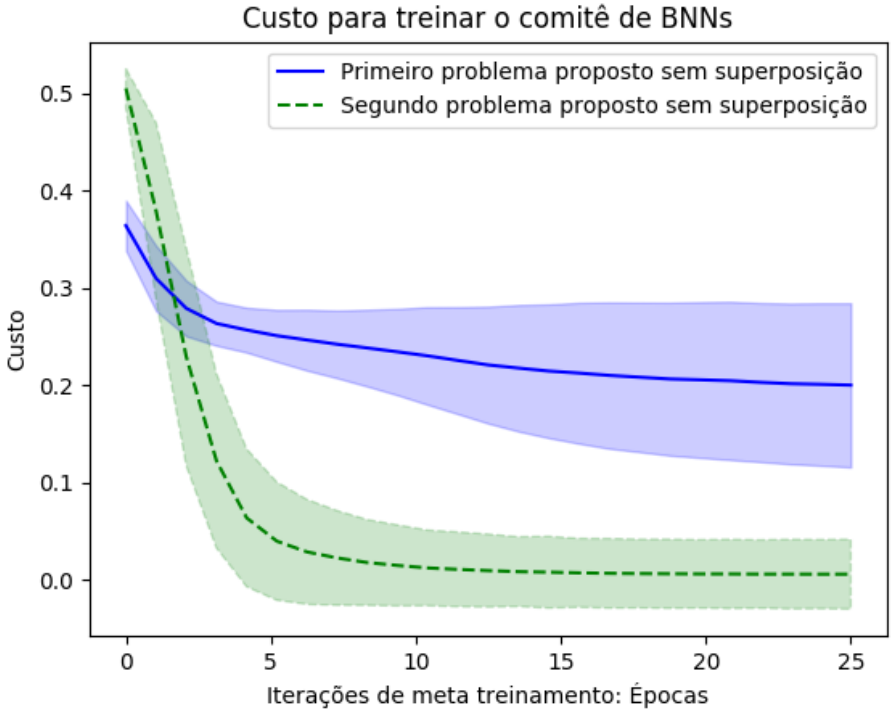


Figura 27 – Otimização da função de custo com meta-aprendizagem.

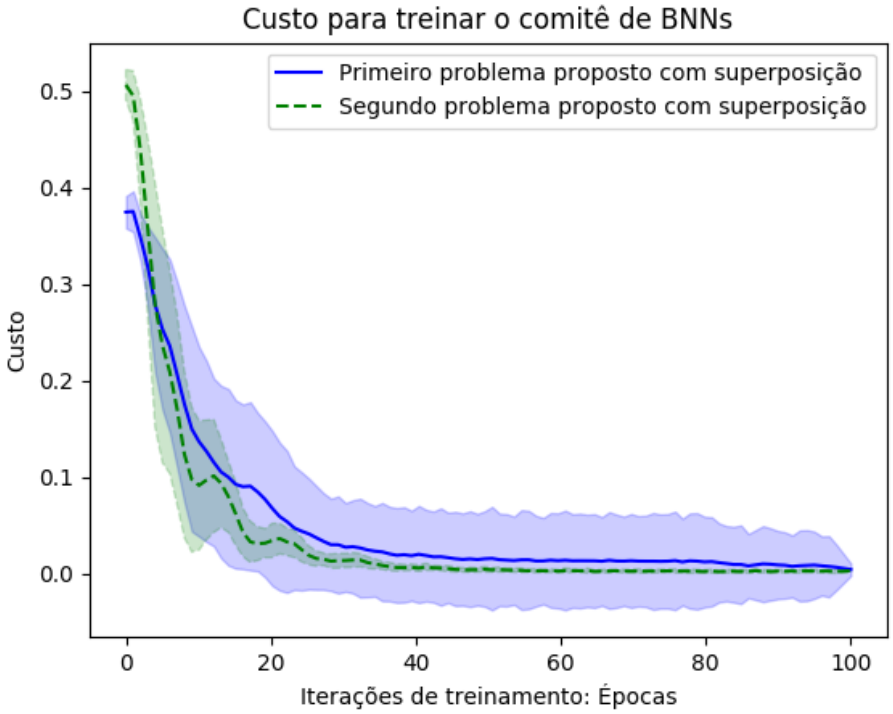


Figura 28 – Otimização da função de custo sem meta-aprendizagem.

(com os resultados do experimento realizado por Fawaz et al.). Cada ponto representa uma simulação de 50 execuções dos algoritmos.

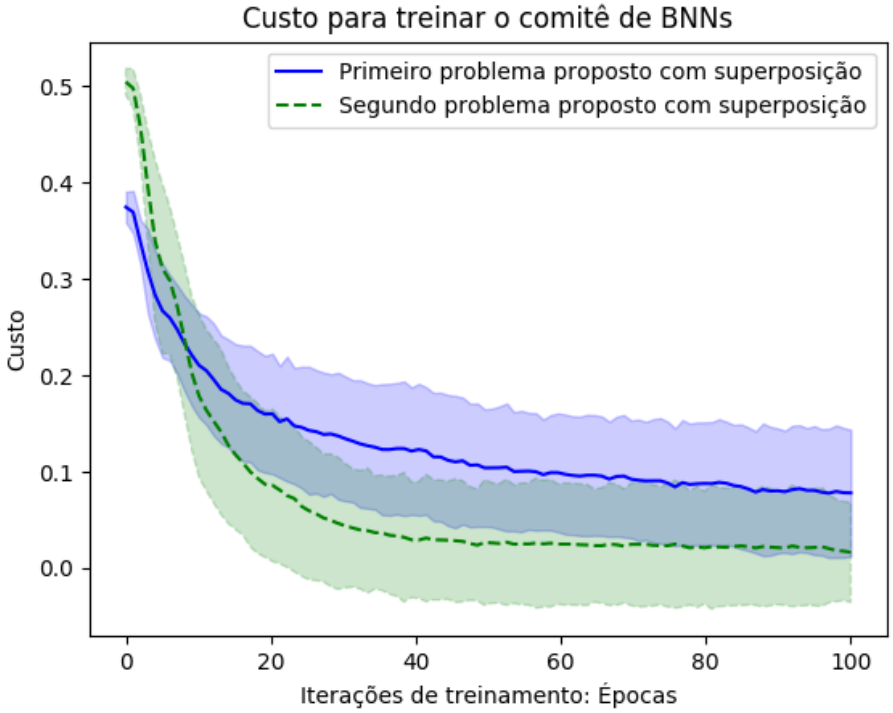


Figura 29 – Otimização da função de custo com meta-aprendizagem.

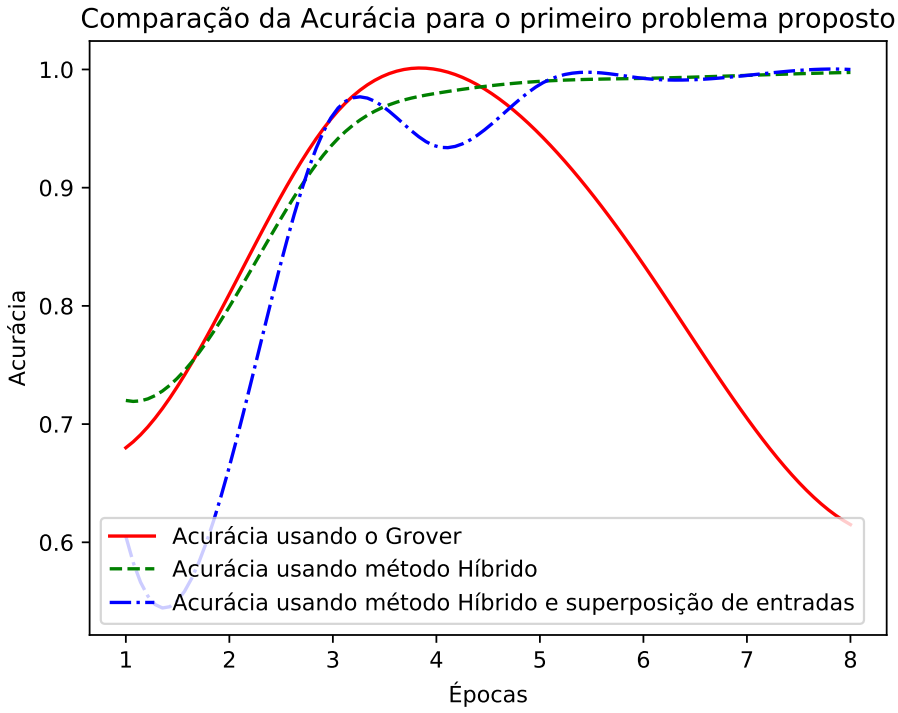


Figura 30 – Primeiro problema sem meta-aprendizagem e sem superposição.

A taxa de convergência da amplificação de amplitude (algoritmo de Grover) sem meta-aprendizagem é competitiva para o algoritmo de aprendizagem CRNBQ, devido ao baixo número de qubits. No experimento de meta-aprendizagem, o método proposto tem uma convergência

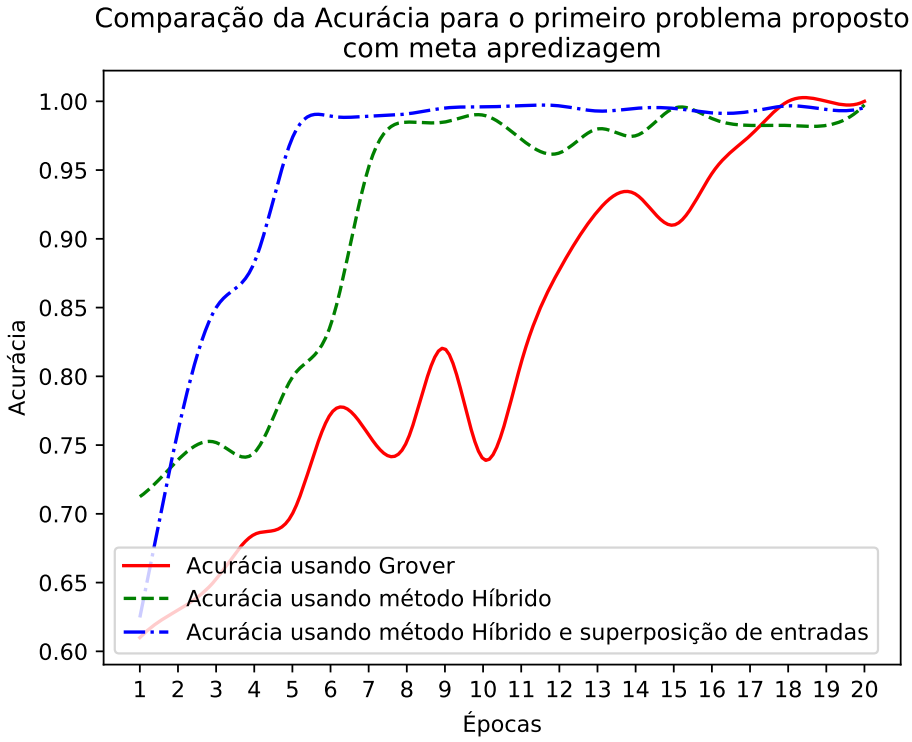


Figura 31 – Primeiro problema com meta-aprendizagem e sem superposição.

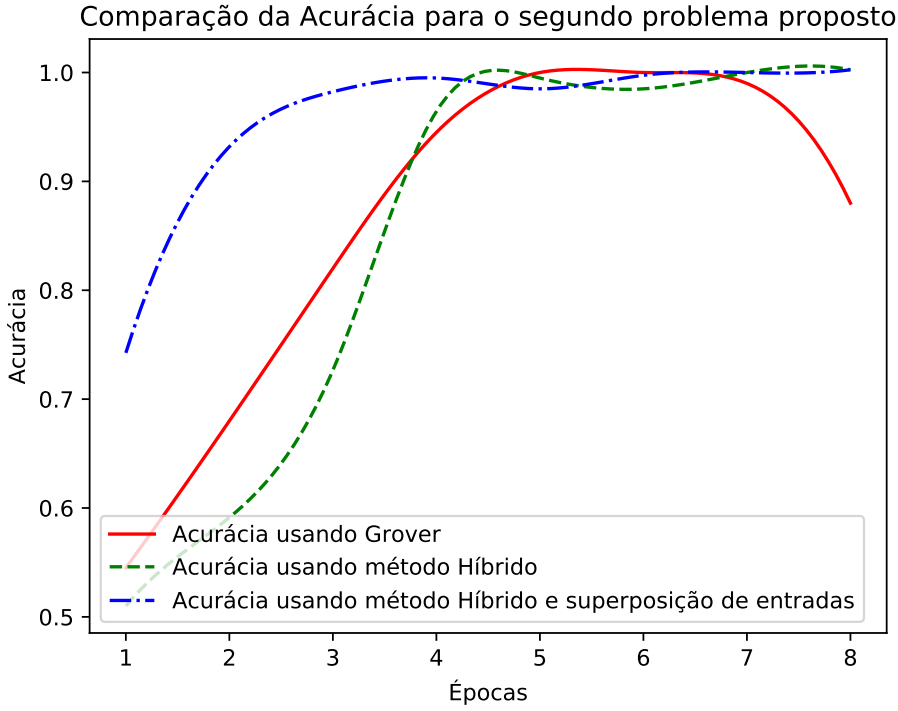


Figura 32 – Segundo problema sem meta-aprendizagem e com superposição.

mais rápida, pois a busca quântica tem um custo exponencial em relação ao número de qubits no espaço de busca. Não está claro quando devemos usar dados de treinamento em superposição e em dispositivos quânticos reais, pois o uso de superposição pode aumentar o ruído em experimentos.

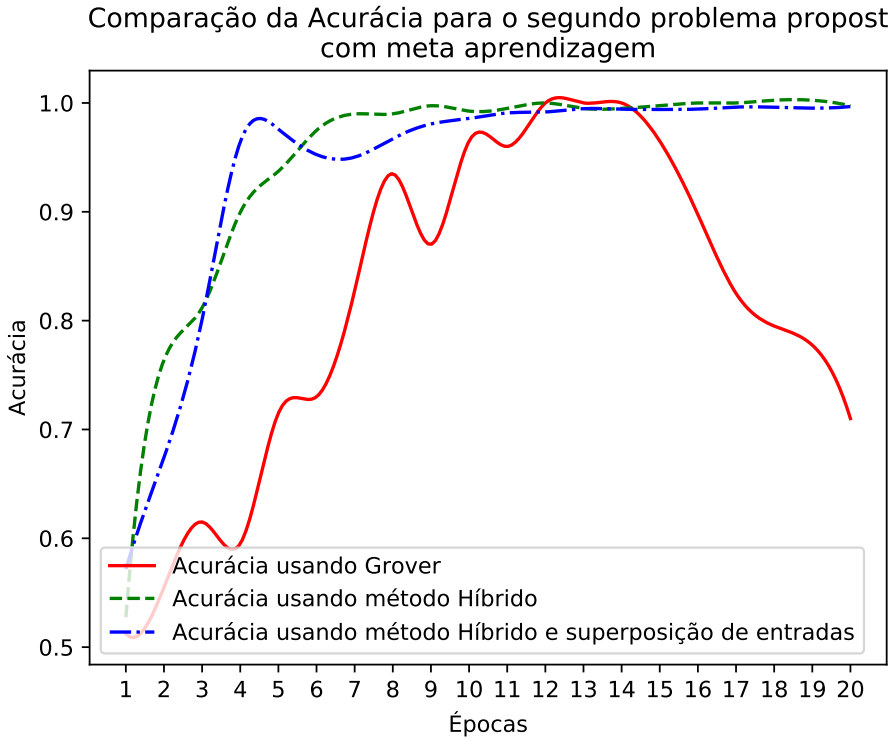


Figura 33 – Segundo problema com meta-aprendizagem e com superposição.

Após fazer uma Busca Aleatória sobre o conjunto de parâmetros descritos na Tabela 2 foi possível verificar que os parâmetros descritos na Tabela 3 para executar o CRNBQ trazem um resultado de acurácia que se equipara aos resultados obtidos por Fawaz et al. e em alguns casos são, inclusive, melhores. Executamos os experimentos salvando os resultados de precisão de cada problema para poder comparar com a abordagem de Grover. Cada ponto representa uma simulação de 50 execuções separadas do algoritmo e a probabilidade associada a cada resultado de qubit de precisão. Além disso, o conjunto de combinação de parâmetros foi adequado para os problemas com diferentes abordagens, tornando-nos capazes de convergir para 100% de precisão, veja as Figuras 30, 31, 32 e 33.

Taxa de Aprendizado	0.3888
Épocas	11
Otimizador	Adagrad

Tabela 3 – Distribuição de parâmetros selecionados usada para pesquisar a matriz de pesos e rotações de meta-aprendizagem para o primeiro e segundo problema proposto sem superposição nas entradas.

Taxa de Aprendizado	0.2917
Épocas	27
Otimizador	Adagrad

Tabela 4 – Distribuição de parâmetros selecionados usada para pesquisar a matriz de pesos e rotações de meta-aprendizagem para o segundo problema proposto utilizando superposição nos parâmetros de entrada da rede.

6 Considerações Finais

Aumentar a capacidade de aprendizado de uma RNA é uma tarefa difícil quando temos uma grande quantidade de dados e grandes redes neurais para treinar (1). A seleção da arquitetura é outro problema que muitos trabalhos procuram resolver na última década. Foi possível mostrar como treinar um Comitê de classificadores com um número exponencial de Redes Neurais em dispositivos quânticos com o custo computacional de treinar uma única Rede Neural. O algoritmo de aprendizado proposto realiza uma busca de arquitetura através de um meta-treinamento e é capaz de encontrar um comitê de classificadores composto por redes neurais com diferentes arquiteturas que seja capaz de classificar corretamente o conjunto de dados aqui proposto. O método criado é baseado em circuitos quânticos variacionais e na codificação de redes neurais clássicas em dispositivos quânticos.

6.1 Conclusão do Trabalho

Neste trabalho, mostramos como treinar e meta treinar um conjunto de RNB aproveitando dispositivos quânticos. A proposta de modificação RNBQ consome menos qubits conforme a abordagem de Grover. Também apresentamos um novo método de treinamento que propõe treinar um conjunto de pesos quânticos. Isso não limita o QBNN a ter um peso binário específico, mas, em vez disso, permite que ele aproveite a superposição quântica sobre os pesos criando um conjunto de RNB. Portanto, aumentamos o poder preditivo de nosso RNB usando uma técnica clássica, mas dentro de dispositivos quânticos. Além disso, carregamos as entradas em sobreposição conforme proposto anteriormente em (28, 53).

6.2 Trabalhos Futuros

Com parte dos resultados obtidos nesse trabalho foi possível realizar uma publicação (54) e a produção de um novo artigo está em andamento para que seja possível realizar uma nova publicação visando disseminar os demais resultados obtidos nessa pesquisa.

Tendo em vista os avanços alcançados neste trabalho, um novo método de otimização pode surgir, uma vez que a profundidade do circuito é menor do que a abordagem de amplificação

de Grover. Além disso, decidimos otimizar o circuito usando um otimizador convencional e bem conhecido (Adam e Adagrad) que mostrou que é possível encontrar os pesos mais adequados.

Além disso, podemos abordar a questão sobre o uso de QBNN em dispositivos quânticos reais e sob influência de ruído. Também podemos treinar a QBNN usando outro otimizador como o Otimizador Natural Quantum por meio de um relatório de caso de uso. Carregar um conjunto de dados maior e mais difícil é uma das questões que também poderia ser abordada em outros trabalhos. Por fim, também pudemos testar outros métodos de otimização para verificar se a seleção dos pesos e da arquitetura é mais adequada com métodos heurísticos. Encontrar maneiras de carregar entradas diferentes das binárias deve ser um grande avanço para este trabalho.

Com este trabalho pudemos ver o aprimoramento criado para selecionar pesos e arquitetura do RNB utilizando comitê de classificadores em dispositivos quânticos. Também sabemos que isso abre as portas para muitas possibilidades para outras técnicas híbridas. E alguns trabalhos futuros serão abordados e guiados pela abordagem híbrida apresentada neste trabalho.

Referências

1 KOVÁCS, Z. L. *Redes neurais artificiais*. [S.l.]: Editora Livraria da Fisica, 2002.

2 ELSKEN, T.; METZEN, J. H.; HUTTER, F. Neural architecture search: A survey. *The Journal of Machine Learning Research*, v. 20, 2019.

3 OJHA, V. K.; ABRAHAM, A.; SNÁŠEL, V. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, v. 60, p. 97–116, 2017.

4 LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

5 BENTON, A. Facial recognition 1990. *Cortex*, Elsevier, v. 26, n. 4, p. 491–499, 1990.

6 PAK, M.; KIM, S. A review of deep learning in image recognition. In: IEEE. *2017 4th international conference on computer applications and information processing technology (CAIPT)*. [S.l.], 2017. p. 1–3.

7 ALEKSANDER, I.; THOMAS, W.; BOWDEN, P. Wisard· a radical step forward in image recognition. *Sensor review*, MCB UP Ltd, 1984.

8 JACOBS-LORENA, M.; LEMOS, F. Immunological strategies for control of insect disease vectors: a critical assessment. *Parasitology Today*, Elsevier, v. 11, n. 4, p. 144–147, 1995.

9 SOUZA, E. P. d. et al. Aplicações do deep learning para diagnóstico de doenças e identificação de insetos vetores. *Saúde em Debate*, SciELO Public Health, v. 43, p. 147–154, 2020.

10 CAREY, A. F.; CARLSON, J. R. Insect olfaction from model systems to disease control. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 108, n. 32, p. 12987–12995, 2011.

11 KWON, D. et al. A survey of deep learning-based network anomaly detection. *Cluster Computing*, Springer, v. 22, n. 1, p. 949–961, 2019.

12 GARTNER, N. H.; STAMATIADIS, C.; TARNOFF, P. J. Development of advanced traffic signal control strategies for intelligent transportation systems: Multilevel design. *Transportation Research Record*, SAGE Publishing, v. 1494, p. 98–105, 1995.

13 GOSLING, G. D. Identification of artificial intelligence applications in air traffic control. *Transportation Research Part A: General*, Elsevier, v. 21, n. 1, p. 27–38, 1987.

14 YANG, A.; ESPERANÇA, P. M.; CARLUCCI, F. M. NAS evaluation is frustratingly hard. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2019.

15 BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Pearson, 2005.

16 NIELSEN, M. A.; CHUANG, I. *Quantum computation and quantum information*. [S.l.]: Cambridge University Press, 2002.

17 FEYNMAN, R. P. et al. Simulating physics with computers. *Int. j. Theor. phys*, v. 21, n. 6/7, 1982.

18 PRESKILL, J. Quantum computing in the NISQ era and beyond. *Quantum*, v. 2, p. 79, 2018.

19 GROVER, L. K. A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1996. p. 212–219.

20 MONTANARO, A. Quantum algorithms: an overview. *npj Quantum Information*, Nature Publishing Group, v. 2, n. 1, p. 1–8, 2016.

21 KHAIRY, S. et al. Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems. *arXiv preprint arXiv:1911.04574*, 2019.

22 LIMA, J. F.; SEMENTE, R. S. Simulação da transformada quântica de fourier com o simulador zeno. *Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)*, n. 2, 2018.

23 SILVA, A. J. da; LUDERMIR, T. B.; OLIVEIRA, W. R. de. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, v. 76, p. 55–64, 2016.

24 PANELLA, M.; MARTINELLI, G. Neural networks with quantum architecture and quantum learning. *International Journal of Circuit Theory and Applications*, v. 39, p. 61–77, 2011.

25 FAWAZ, A. et al. Training and meta-training binary neural networks with quantum computing. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2019. p. 1674–1681.

26 YANOFSKY, N. S.; MANNUCCI, M. A. *Quantum Computing for Computer Scientists*. [S.l.]: Cambridge University Press, 2008.

27 RIEFFEL, E.; POLAK, W. *Quantum Computing: A Gentle Introduction*. 1st. ed. [S.l.]: The MIT Press, 2011. ISBN 9780262015066.

28 FARHI, E.; NEVEN, H. Classification with quantum neural networks on near term processors (2018). *arXiv preprint arXiv:1802.06002*, 1802.

29 DAVILA, A. R.; AL, A. J. C. et. *Qiskit: the Quantum Information Science Kit*. 2019.

30 QIN, H. et al. Binary neural networks: A survey. *Pattern Recognition*, Elsevier, v. 105, p. 107281, 2020.

31 KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2014.

32 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. 61, p. 2121–2159, 2011. Disponível em: <http://jmlr.org/papers/v12/duchi11a.html>.

33 BERGHOLM, V. et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

34 SCHULD, M. et al. Evaluating analytic gradients on quantum hardware. *Physical Review A*, APS, v. 99, n. 3, p. 032331, 2019.

35 GRIOL-BARRES, I. et al. Variational quantum circuits for machine learning. an application for the detection of weak signals. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 11, n. 14, p. 6427, 2021.

36 CHEN, S. Y.-C. et al. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, IEEE, v. 8, p. 141007–141024, 2020.

37 COURBARIAUX, M.; BENGIO, Y.; DAVID, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, v. 28, 2015.

38 RASTEGARI, M. et al. Xnor-net: Imagenet classification using binary convolutional neural networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 525–542.

39 AL., A. J. da Silva et. *qclib: Quantum Computing Library*. 2021.

40 MOTTONEN, M. et al. Transformation of Quantum States Using Uniformly Controlled Rotations. *Quantum Info. Comput.*, p. 467–473, 2005.

41 DIETTERICH, T. G. Ensemble methods in machine learning. In: *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. p. 1–15.

42 BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

43 SCHULD, M.; PETRUCCIONE, F. Quantum ensembles of quantum classifiers. *Scientific reports*, Nature Publishing Group, v. 8, n. 1, p. 1–12, 2018.

44 BIAMONTE, J. et al. Quantum machine learning. *Nature*, p. 195–202, 2017.

45 HARROW, A. W.; HASSIDIM, A.; LLOYD, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, p. 150502, 2009.

46 AARONSON, S. Read the fine print. *Nature Physics*, p. 291–293, 2015.

47 SHENDE, V. V.; BULLOCK, S. S.; MARKOV, I. L. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, p. 1000–1010, 2006.

48 ARAUJO, I. F. et al. A divide-and-conquer algorithm for quantum state preparation. *Scientific Reports*, p. 6329.

49 YAO, H. et al. Negative log likelihood ratio loss for deep neural network classification. In: SPRINGER. *Proceedings of the Future Technologies Conference*. [S.l.], 2019. p. 276–282.

50 DéFOSSEZ, A. et al. On the convergence of adam and adagrad. *CoRR*, 2020.

51 PASZKE, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates, Inc., 2019. 8024–8035 p. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

52 TIETZ, M. et al. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*. [S.l.], 2017. Disponível em: <https://skorch.readthedocs.io/en/stable/>.

53 NGOC, V. P.; WIKLICKY, H. Tunable quantum neural networks for boolean functions. *arXiv preprint arXiv:2003.14122*, 2020.

54 LEAL, D.; LIMA, T. D.; SILVA, A. J. D. Training ensembles of quantum binary neural networks. In: IEEE. *2021 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2021. p. 1–6.