LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

Classificação de Candidatas a Pulsar usando SVMs gerados de forma híbrida Clássico-Quântica

Gabriela Pinheiro
Universidade Federal Fluminense (UFF)
Niterói, RJ
gabrielapc@id.uff.br

Luis Antonio Kowada
Universidade Federal Fluminense (UFF)
Niterói, RJ
luis@ic.uff.br

RESUMO

Com o crescimento do número de candidatas a pulsares, urge a necessidade da aplicação de métodos automatizados para sua análise e classificação, como o uso de Inteligência Artificial. Máquina de Vetores de Suporte (SVM - *Support Vector Machine*) se consolidou como um dos principais algoritmos de Aprendizado de Máquina para problemas de classificação. Em sua formulação original, o treinamento de um SVM possui complexidade quadrática, o que abre margem para a busca de métodos mais eficientes de resolução. Uma forma de melhorar a eficiência é através do uso de Computação Quântica, onde é possível obter algoritmos com vantagens em relação aos computadores clássicos, como o HHL. Este artigo apresenta a classificação de candidatas a pulsares utilizando SVMs gerados através de um método automatizado híbrido clássico-quântico, onde o uso de Aprendizado de Máquina Quântico permitiu a obtenção de acurácias superiores à acurácia máxima encontrada pela versão clássica do algoritmo.

PALAVRAS CHAVE. Aprendizado de Máquina Quântico, SVM, HHL

Aprendizado de Máquina Quântico, Computação Quântica, Aprendizado de Máquina

ABSTRACT

With the growth in the number of pulsar candidates, there is an urgent need to apply automated methods for their processing and classification, such as the use of Artificial Intelligence. The Support Vector Machine (SVM) algorithm has established itself as one of the main Machine Learning algorithms for classification problems. In its original formulation, training an SVM has quadratic complexity, which opens the door to the search for more efficient resolution methods. One way to improve efficiency is through the use of Quantum Computing, where it is possible to obtain algorithms with advantages over classical computers, such as the HHL. This paper presents the classification of pulsar candidates through the use of SVMs generated via a hybrid classical-quantum automated method, where the use of Quantum Machine Learning allowed the achievement of better accuracies than the maximal accuracy obtained from the classical counterpart of the algorithm.

KEYWORDS. Quantum Machine Learning, SVM, HHL

Quantum Machine Learning, Quantum Computing, Machine Learning

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

1. Introdução

Pulsares são estrelas de nêutrons que giram em torno de seu próprio eixo e emitem pulsos de rádio em intervalos regulares. Com o avanço tecnológico no desenvolvimento de telescópios, a quantidade de dados coletados sobre possíveis candidatas a pulsares cresceu exponencialmente, tornando a análise manual impraticável [Lyon et al., 2016], gerando uma demanda por técnicas para a automação deste processo, como o uso de Aprendizado de Máquina para a classificação de candidatas.

Máquina de Vetores de Suporte (SVM - *Support Vector Machine*) é considerado um dos mais poderosos algoritmos de classificação que, devido à sua forte fundamentação teórica e generabilidade, é amplamente utilizado em diversas aplicações relevantes para o mundo real, dentre elas estão a classificação de imagens e bioinformática [Cervantes et al., 2020]. Todavia, uma das principais desvantagens deste algoritmo é o seu custo computacional, possuindo complexidade quadrática em sua formulação original.

Como forma de reduzir a complexidade do algoritmo, foi aplicada uma reformulação de quadrados mínimos na versão original do SVM, transformando-o em um sistema de equações lineares [Suykens e Vandewalle, 1999]. Esta transformação permite o uso de técnicas mais eficientes para a resoluções de sistemas de equações lineares, sendo uma possível abordagem o uso de algoritmos quânticos.

A Computação Quântica permite explorar propriedades da Mecânica Quântica de forma a obter vantagens no processamento de informações em relação aos computadores clássicos para determinados casos. Apesar de ainda não estarem disponíveis computadores quânticos de tamanho suficiente para permitir grandes computações, a área já se demonstra promissora com alguns de seus algoritmos possuindo ganhos exponenciais de performance [Gyongyosi e Imre, 2019].

O HHL [Harrow et al., 2009] é um algoritmo quântico para a resolução de sistemas lineares com o qual é possível extrair informações sobre a solução e obter um ganho exponencial em relação à algoritmos clássicos. Ele tem se popularizado na área de Aprendizado de Máquina Quântico, sendo usado em diversos tipos de algoritmos como regressão linear e SVMs [Duan et al., 2020].

Este trabalho tem como objetivo a classificação de candidatas a pulsares através da automatização do uso do HHL em um algoritmo de Aprendizado de Máquina Quântico proposto por Rebentrost et al. [2014] para geração de SVMs fixados na origem de um hiperplano.

2. Conceitos Fundamentais

2.1. Support Vector Machines

O SVM [Cortes e Vapnik, 1995] é um algoritmo de aprendizado de máquina criado originalmente para a classificação binária, onde um conjunto de dados é divido em duas classes distintas e o algoritmo precisa aferir a qual classe um novo elemento do conjunto irá pertencer.

Seu funcionamento é feito da seguinte maneira: os dados do conjunto são mapeados não-linearmente para pontos em um hiperespaço, onde a dimensão é a quantidade de parâmetros utilizados e o objetivo do algoritmo é encontrar um hiperplano de decisão linear neste hiperespaço que separe os dados de ambas as classes, maximizando sua margem de distância para elas. Deste modo, é possível classificar um novo dado inserido no conjunto ao analisar sua posição relativa ao hiperplano.

O hiperplano a ser encontrado pode ser representado pela Equação (1), e as classes são representadas pelos rótulos 1 e -1. Um dado $\vec{\chi}$ é considerado da classe 1 se a desigualdade $\vec{w} \cdot \vec{\chi} - b \geq 1$ for verdadeira, ou é considerado da classe -1 se a desigualdade $\vec{w} \cdot \vec{\chi} - b \leq -1$ for verdadeira. A Figura 1 representa um exemplo de SVM em um hiperespaço de dimensão 2.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

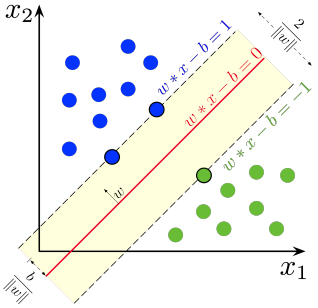


Figura 1: Exemplo de hiperplano de máxima margem em um hiperespaço de dimensão 2. Fonte: [Larhmam, 2018].

O hiperplano gerado é representado pela Equação (1), e a margem de distância entre as classes passa a ser representada por $\frac{2}{||\vec{w}||}$.

$$\vec{w} \cdot \vec{x} - b = 0 \tag{1}$$

2.2. Computação Quântica

A Computação Quântica é a área da computação que busca explorar o processamento de informações através do uso de sistemas de Mecânica Quântica [Nielsen e Chuang, 2010]. Utilizando propriedades como a superposição e o emaranhamento, é possível obter algoritmos que apresentem vantagens em relação aos computadores clássicos.

2.2.1. Qubits

Em contraste com o bit, que sempre se encontra em um único estado, sendo ele "0" ou "1", a menor unidade de informação em um computador quântico recebe o nome de qubit e é representada por um vetor de estados de módulo 1 que varia entre os estados ortonormais $\binom{1}{0}$ e $\binom{0}{1}$ da chamada base computacional, representadas como $|0\rangle$ e $|1\rangle$ utilizando a notação de Dirac [Tumulka, 2009], equivalentes aos estados "0" e "1". Devido a sua natureza vetorial, é possível formar combinações lineares de estados, gerando uma superposição representada pela Equação (2):

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2}$$

sendo α e β números complexos tal que $|\alpha|^2 + |\beta|^2 = 1$. Pelas leis da Mecânica Quântica, ao se realizar uma medição do estado $|\psi\rangle$ na base computacional, ele possui probabilidade $|\alpha|^2$ de assumir o estado "0" e probabilidade $|\beta|^2$ de assumir o estado "1".

Considerando um sistema com n qubits, o estado do sistema pode ser representado pela Equação 3:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \tag{3}$$

tal que $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$, para que o vetor tenha comprimento unitário.

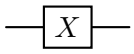
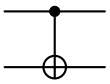
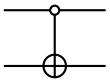
2.2.2. Operadores

Análogos aos operadores lógicos presentes na computação clássica, os circuitos quânticos são construídos a partir de operadores quânticos. Seguindo os postulados da Mecânica Quântica, o vetor que representa o estado do sistema pode ser modificado por operadores unitários, que são operadores lineares e, portanto, podem ser representados de forma matricial. Sendo assim, a aplicação de um operador em um conjunto de qubits pode ser representada pela multiplicação de sua matriz

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

com o vetor associado ao estado do sistema. Os operadores unitários não alteram o módulo do vetor, ou seja, o vetor resultante também é unitário e a inversa da matriz associada a um operador unitário é sua transposta conjugada, ou seja, $M^{-1} = M^{\dagger}$. A Tabela 1 apresenta um conjunto de operadores e símbolos básicos utilizados para representações de circuito quânticos que são utilizados neste trabalho.

Tabela 1: Operadores e símbolos básicos presentes em computadores quânticos que serão utilizados neste trabalho. Fonte:[Nielsen e Chuang, 2010]

Operador	Símbolo	Forma matricial ou Descrição
Hadamard		$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
Pauli-X (Not)		$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$
X-Controlado (CNot)		$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$
X-Controlado Aberto		O operador X será aplicado no alvo quando o controle estiver no estado $\|0\rangle$
Medição		Projeta o qubit para $\|0\rangle$ ou $\|1\rangle$
Qubit		Representação de um qubit no circuito

2.2.3. Era NISQ

A Computação Quântica se encontra atualmente na era NISQ (*Noisy Intermediate-Scale Quantum*), referente ao tipo de computadores disponíveis, chamados computadores quânticos ruidosos de escala intermediária [Preskill, 2018]. Estes computadores ainda possuem capacidade de processamento limitada, onde já não é possível simular seu comportamento com os supercomputadores clássicos existentes [Pednault et al., 2017], mas ainda não é possível implementar circuitos muito expressivos. Outro fator limitador é a alta taxa de erros de medição, o ruído.

O ruído ocorre em decorrência de diversos fatores, como erros na execução dos operadores, principalmente em operadores de múltiplos qubits, e decoerência, onde o qubit acaba interagindo com o ambiente e perdendo seu estado [Saki et al., 2021]. A quantidade de ruído está diretamente ligada ao tamanho do circuito, onde quanto maior o número de qubits e operadores, maior o ruído na medição. Sendo possível obter resultados completamente corrompidos pelas alterações geradas.

2.3. HHL

Nomeado em homenagem aos seus criadores, A. Harrow, A. Hassidim e S. Lloyd, o HHL é um algoritmo quântico para a resolução de sistemas lineares. Considerando o sistema $A\vec{x} = \vec{b}$, dados o vetor $\vec{b}$ e a matriz A, ele consegue extrair informações sobre a solução $\vec{x}$ de forma eficiente, chegando a ser, em certas circunstâncias, exponencialmente mais rápido que qualquer algoritmo clássico [Harrow et al., 2009]. Por não apresentar a solução exata, ele é adequado para aplicações em que se obter algum conjunto de propriedades já seja o suficiente. No caso da geração de SVMs apresentado neste trabalho, basta a razão entre as componentes da solução.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

2.3.1. Reformulação de quadrados mínimos

A formulação original do SVM tem como objetivo maximizar a margem de distância $\frac{2}{||\vec{w}||}$ entre as classes, referente à Equação (1), o que é equivalente a minimizar $\frac{\|\vec{w}\|^2}{2}$ seguindo a restrição $y_j(\vec{w} \cdot \vec{x_j} + b) \geq 1$ pertencente ao conjunto de dados.

Em [Rebentrost et al., 2014], foi proposta uma reformulação do problema através de uma aproximação de quadrados mínimos, transformando o SVM na resolução do sistema linear (4).

$$F \begin{pmatrix} b \\ \vec{a} \end{pmatrix} \equiv \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \gamma^{-1} I \end{pmatrix} \begin{pmatrix} b \\ \vec{a} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \tag{4}$$

Sendo K a matriz de Kernel, uma matriz $M \times M$ construída a partir de M vetores de treinamento pela função $K_{ij} = k(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$, sendo $\vec{y} = (y_0, y_1, ..., y_{M-1})$ e $\vec{1} = (1, ..., 1)$. I representa a matriz identidade e γ é um valor definido pelo usuário para representar o peso relativo do erro de treinamento em relação ao objetivo. Resultando em uma matriz $F_{(M+1)\times(M+1)}$ na qual os parâmetros do SVM passam a ser representados em função de b e $\vec{a}$.

A partir dessa reformulação, [Rebentrost et al., 2014] propôs a utilização do HHL para a resolução desse sistema linear, gerando um algoritmo de Aprendizado de Máquina Quântico. A classificação de cada um dos M dados $\vec{x_i}$ passa a ser obtida de acordo com as condições em (5).

$$y_i = \begin{cases} +1 \text{ se } \sum_{j=0}^{M-1} a_i k(\vec{x_i}, \vec{x_j}) + b \geq 0 \\ -1 \text{ se } \sum_{j=0}^{M-1} a_i k(\vec{x_i}, \vec{x_j}) + b < 0 \end{cases} \tag{5}$$

Para reduzir o número de operadores, a quantidade de qubits necessários para a implementação do circuito e, consequentemente, sua complexidade, [Yang et al., 2019] gerou um SVM que apresenta bons resultados fixando o parâmetro b em 0, implicando que o hiperplano gerado passe necessariamente pela origem do hiperespaço. Permitindo que o sistema possa ser simplificado com a remoção tanto da primeira linha quanto da primeira coluna de F, pois seus valores se tornariam zero pela multiplicação. A forma simplificada resultante pode ser vista na Equação (6).

$$F\left(\vec{a}\right) \equiv \left(K + \gamma^{-1} I\right)\left(\vec{a}\right) = \left(\vec{y}\right) \tag{6}$$

Deste modo, a matriz F passa a ser de tamanho $M \times M$ e é necessário encontrar apenas $\vec{a}$ onde, por ele ser um vetor, a razão entre suas componentes já possui informação suficiente para aferir sua inclinação e traçar o hiperplano de decisão do SVM. Isto resulta em um cenário ideal para a aplicação do HHL, pois ele consegue encontrar essa razão de forma eficiente. Caso b não fosse fixado, somente as razões entre ele e os componentes de $\vec{a}$ não seriam suficientes para a geração do SVM, o que resultaria na necessidade de um pós-processamento extra dos resultados para encontrar seu valor exato.

2.4. Automação do circuito

Por ser uma área emergente, ainda não estão disponíveis muitas ferramentas automatizadas para a implementação de algoritmos de Aprendizado de Máquina Quântico, o que exige que a geração dos circuitos seja feita manualmente para a maioria dos casos de uso. A geração manual dos circuitos tem como consequência a exigência de conhecimento prévio do usuário sobre Computação Quântica, o que tem mantido as aplicações de Aprendizado de Máquina Quântico restritas à pesquisa dentro da área. A automação da geração dos circuitos não apenas facilita a diversificação de

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

casos testados como também permite a utilização do algoritmo para diferentes aplicações sem a exigência de conhecimento prévio sobre Computação Quântica.

Para automação foi feita uma generalização para matrizes de dimensão $2^n \times 2^n, n \in \mathbb{N}$ do método apresentado em [Morrell Jr et al., 2021]. Este artigo apresenta um modelo de construção de circuito generalizável para a implementação do HHL, em contrapartida, a quantidade de operadores tende a crescer conforme a precisão necessária para representar a matriz K.

A automação dos circuitos para SVMs em hiperplanos de dimensão 2^n foi feita utilizando a linguagem de programação Python junto com a biblioteca Qiskit [Cross, 2018] seguindo o método proposto. Os circuitos foram gerados para solucionar o sistema apresentado na Equação (6). A seguir é descrito este processo de implementação.

3. Implementação

3.1. Pré-processamento da matriz de Kernel

O circuito gerado possui três registradores (sequências de qubits) $|y\rangle_{n_y}|k\rangle_{n_k}|anc\rangle$, onde os primeiros n_y qubits são usados para representar $\vec{y}$, os n_k qubits seguintes são usados para representar a matriz K e o último qubit anc é um qubit auxiliar nomeado de ancilla. A codificação de $\vec{y}$ é feita ao rotacionar as amplitudes dos n_y bits para corresponder aos componentes de $\vec{y}$, permitindo que ele seja codificado por $\log_2 M$ bits, onde $M \times M$ é a dimensão da matriz F. Nesta automação o valor de γ foi fixado em 0, fazendo com que $F = K$.

A matriz K é representada no sistema através de seus autovalores pela estimativa de fase quântica, seu funcionamento é abordado na subseção 3.3. Os autovalores são mapeados no circuito em função da razão entre eles, a forma adotada para se fazer o mapeamento neste trabalho é a divisão de todos os autovalores pelo menor autovalor, como mostrado na Equação (7). Desta forma, o menor autovalor se torna 1 e os autovalores restantes se tornam a aproximação inteira da razão. Sendo n_k a quantidade de bits necessária para representar binariamente o maior destes valores e $N_k = 2^{n_k}$.

$$\lambda'_j = \frac{\lambda_j}{min(\lambda)} \tag{7}$$

Portanto, o circuito gerado possui $n_k + n_y + 1$ qubits.

3.2. Mapeamento de $\vec{y}$

Como cada linha da matriz de Kernel representa um elemento de cada classe, é possível fixar a ordem das classes dos mesmos e, consequentemente, o valor do vetor $\vec{y}$, apresentado em (8).

$$\vec{y} = \begin{pmatrix} +1 \\ -1 \\ \vdots \\ +1 \\ -1 \end{pmatrix} \tag{8}$$

Ao normalizar $\vec{y}$ se obtém a Equação (9), que pode ser mapeado para uma sequência do estado conhecido na literatura como $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$, como pode ser visto na Equação (10):

$$\vec{y'} = \frac{1}{\sqrt{2^{n_y}}} \begin{pmatrix} +1 \\ -1 \\ \vdots \\ +1 \\ -1 \end{pmatrix} \tag{9}$$

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

$$
\begin{aligned}
|y'\rangle &= H^{n_y} X^{n_y} \otimes |0\rangle^{n_y} \\
|y'\rangle &= H^{n_y} \otimes |1\rangle^{n_y} \\
|y'\rangle &= \frac{(|0\rangle - |1\rangle)^{n_y}}{\sqrt{2^{n_y}}}
\end{aligned}
\tag{10}
$$

3.3. Estimativa de Fase

A estimativa de fase quântica (QPE) é originalmente um algoritmo de estimativa de autovalores utilizado para a codificação da matriz K. Ele é constituído por três partes, O emaranhamento dos qubits de n_k obtido com o uso de operadores H, seguida de uma rotação controlada e a Transformada de Fourier Quântica Inversa(IQFT).

A rotação controlada ocorre com a aplicação da matriz de rotação unitária $U = e^{iKt}$ ao registrador $|y\rangle_{n_y}$. Sendo representada no circuito por uma sequência de n_k operadores U^{2^n} com controles em $|k\rangle_{n_k}$ correspondentes à n, logo, o operador U^{2^0} terá o seu controle no qubit 0 de $|k\rangle_{n_k}$.

O operador U^{2^n} é criado com o uso do autovetores $\vec{u_j}$ de K e dos valores λ'_j calculados em (7). Os autovetores são representados em função da matriz de transformação V, apresentada em (11):

$$
V = (\vec{u_0}, \ldots, \vec{u_{2^{n_y}}}) \tag{11}
$$

os operadores U^{2^n} são obtidos de forma iterativa a partir de suas versões diagonalizadas, como apresentado em (12):

$$
\begin{aligned}
U_{diag} &= \begin{pmatrix} e^{\frac{i2\pi\lambda'_0}{2^{n_k}}} & \ldots & 0 \\ & \ddots & \\ 0 & \ldots & e^{\frac{i2\pi\lambda'_{2^{n_y}}}{2^{n_k}}} \end{pmatrix} \\
U^{2^n}_{diag} &= U^{2^{n-1}}_{diag} U^{2^{n-1}}_{diag}
\end{aligned}
\tag{12}
$$

a partir das versões diagonalizadas, U^{2^n} são obtidos com o uso da matriz V, como visto em (13):

$$
\begin{aligned}
U &= V U_{diag} V^\dagger \\
U^{2^n} &= V U^{2^n}_{diag} V^\dagger
\end{aligned}
\tag{13}
$$

Após a rotação controlada, é aplicada a IQFT, Inversa da Transformada de Fourier Quântica [Weinstein et al., 2001], que é a versão quântica da Inversa da Transformada Discreta de Fourier, representada pela Equação (14). Por ser um algoritmo amplamente usado na área da Computação Quântica, existem funções de geração automática de circuitos que implementam os operadores IQFT e QFT, utilizados na implementação deste trabalho.

$$
QFT_{n_k}|x\rangle = \frac{1}{\sqrt{N_k}} \sum_{x'=0}^{N_k-1} e^{\frac{2\pi i x x'}{N_k}} |x'\rangle \tag{14}
$$

3.4. Rotação controlada na Ancilla

Após a aplicação da QPE é necessário rotacionar o qubit de Ancilla de acordo com os autovalores codificados em $|k\rangle_{n_k}$. Essa rotação é feita através do operador RY, descrito pela Equação (15), com controles em todos os qubits de $|k\rangle_{n_k}$.

$$
RY(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -sen(\frac{\theta}{2}) \\ sen(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{15}
$$

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

Sendo $\theta = 2arcsen(\frac{1}{\lambda'_j})$ e os controles são inseridos de forma a codificar a representação binária de λ'_j, para todo λ'_j de K. Deste modo, por exemplo, para $\lambda' = 2$, tem-se $\theta = \frac{\pi}{3}$ e os controles são codificados como "10", adicionando $n_k - 2$ "0" à esquerda.

3.5. Estimativa de Fase Inversa (Descomputação)

A última parte do algoritmo consiste na inversão da QPE aplicada em 3.3, ela ocorre com o intuito de remover o emaranhamento entre os qubits de $|y\rangle_{n_y}$ e $|k\rangle_{n_k}$ e obter o valor de $\vec{a}$ armazenado em $|y\rangle_{n_y}$. Como o nome sugere, ela consiste em aplicar a QPE de forma invertida iniciando com a Transformada de Fourier Quântica, seguida da rotação controlada invertida e o desemaranhamento dos qubits de n_k através de operadores H.

A rotação controlada invertida é feita com a aplicação em ordem decrescente dos operadores U'^{2^n}, que consistem nas matrizes inversas dos operadores U^{2^n}, conforme (16).

$$U'^{2^n}_{diag} = (U^{2^n}_{diag})^{-1} \tag{16}$$

3.6. Medição

Ao final do circuito, a medição é feita somente no qubit de Ancilla e no registrador $|y\rangle_{n_y}$, onde as medições em que o Ancilla apresentar o valor "0" são descartadas e somente as medições contendo o valor "1" são consideradas, a razão entre os elementos de $\vec{a}$ é obtida pela razão entre os valores medidos em $|y\rangle_{n_y}$. Considerando que o Ancilla é medido como o bit menos significativo e que $prob_{01}$ representa a probabilidade do estado "01" ser medido e $prob_{11}$ a probabilidade do estado "11" ser medido, o vetor $\vec{a'}$ é encontrado a partir dessas probabilidades, como apresentado em (17):

$$\vec{a'} = \begin{pmatrix} prob_{01} \\ \vdots \\ prob_{2^{n_y}} \end{pmatrix} \tag{17}$$

A Figura 2 contém o circuito completo obtido a partir do método de construção apresentado.

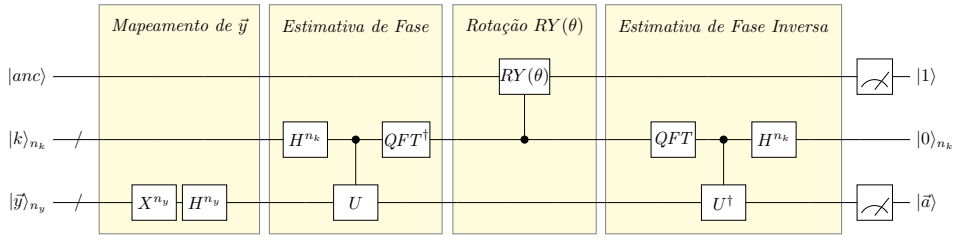


Figura 2: Circuito do algoritmo HHL para a geração de SVMs.

3.7. Pós-processamento de $\vec{a'}$

Como os valores obtidos são probabilísticos, o vetor $\vec{a'}$ contém somente o módulo dos valores buscados, sendo necessário um pós-processamento para obter seus valores reais. Este processamento ocorre com a redução para o problema da soma de subconjuntos(SSP).

3.7.1. Problema da Soma de Subconjuntos

O SSP é um problema clássico de computação onde, dado um conjunto de valores inteiros S, o objetivo do algoritmo é descobrir se existe algum subconjunto de S cuja soma se iguale a um valor inteiro t. Com o uso de programação dinâmica é possível resolvê-lo com complexidade da ordem de $|S|t$ e adaptá-lo de para retornar um conjunto que apresente essa soma sem aumentá-la.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

3.7.2. Reduzindo à SSP

Como o vetor $\vec{a'}$ contém a razão entre os elementos de $\vec{a}$, ele pode ser reescrito como $|\vec{a}| = k\vec{a'}$, sendo k um valor real desconhecido. Considerando K_{ij} o elemento na linha i e coluna j de K e a'_i o elemento na linha i de $\vec{a'}$, as duas primeiras linhas da solução do sistema (6) podem ser escritas como na equação (18):

$$\begin{aligned} K_{00}ka'_0 + K_{01}ka'_1 + \cdots + K_{0M}ka'_M &= 1 \\ K_{10}ka'_0 + K_{11}ka'_1 + \cdots + K_{1M}ka'_M &= -1 \end{aligned} \tag{18}$$

Ao somar ambas as equações será gerada a equação (19) que, ao ser dividida por k, se transforma na equação (20) contendo somente variáveis conhecidas.

$$k[(K_{00} + K_{10})a'_0 + (K_{01} + K_{11})a'_1 + \cdots + (K_{0M} + K_{1M})a'_M] = 0 \tag{19}$$

$$(K_{00} + K_{10})a'_0 + (K_{01} + K_{11})a'_1 + \cdots + (K_{0M} + K_{1M})a'_M = 0 \tag{20}$$

Considerando que $K, \vec{a'} \neq 0$, os elementos negativos de $\vec{a'}$ são encontrados ao se mapear a equação (20) para um SSP com t sendo a metade da soma dos valores absolutos dos seus elementos. Como K e $\vec{a'}$ são vetores reais, é necessário arrendondar os valores da equação (20), o que pode influenciar negativamente a eficácia do algoritmo.

3.7.3. Algoritmo para o SSP com programação dinâmica

A implementação do Algoritmo 1 é feita a partir de uma adaptação do algoritmo para resolução de SSPs com programação dinâmica proposto por [Pisinger, 1999], onde são retornados os índices dos elementos negativos de $\vec{a'}$. É possível que o algoritmo retorne o conjunto inverso dos índices buscados, sendo necessário utilizar o complemento do conjunto retornado caso a acurácia encontrada esteja abaixo de 50%.

4. Resultados

O dataset escolhido para a geração dos SVMs foi o HTRU2 [Lyon et al., 2016; Lyon, 2017], gerado a partir dos dados do *High Time Resolution Universe Survey* (HTRU) e sendo composto por 17898 elementos representados por 8 features distintas. O conjunto de testes é composto por 30% dos dados escolhidos aleatoriamente e o conjunto de treinamento pelo restante.

4.1. Geração da Matriz de Kernel

Para a geração da matriz K, os elementos de cada classe do conjunto de treinamento são divididos em $\frac{M}{2}$ subconjuntos, onde a média dos valores normalizados do conjunto de features escolhidas de seus elementos são utilizados como os elementos geradores de K.

4.2. Seleção de Features

Devido às limitações em relação ao tamanho do circuito, foi necessário fixar o número de features usadas em, no máximo, 4. Por ser um valor que gera sistemas de rápida execução em computadores clássicos, os conjuntos de features foram escolhidos pela técnica de "força bruta", onde todos os conjuntos possíveis de 2 e 4 features foram testados classicamente e os que apresentaram os melhores resultados foram selecionados para a geração dos circuitos quânticos. A título de curiosidade, o SVM contendo todas as 8 features do dataset foi gerado classicamente e apresentou uma acurácia de apenas $75,43\%$.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

Algoritmo 1 Algoritmo adaptado para a resolução do SSP

função $buscaElementosNegativos(K, a')$
  para $i \leftarrow 0$ até $length(a')$ **faça**
    $S[i] \leftarrow round((K_{0i} + K_{0i}) \times a'i)$
  fim para
  $metade \leftarrow sum(S)$ // 2
  para $i \leftarrow 0$ até $length(S) + 1$ **faça**
    para $j \leftarrow 0$ até $metade + 1$ **faça**
      $dp[i][j] \leftarrow False$
    fim para
  fim para
  $dp[0][0] \leftarrow True$
  para $i \leftarrow 1$ até $length(S) + 1$ **faça**
    para $j \leftarrow 0$ até $metade + 1$ **faça**
      $dp[i][j] \leftarrow dp[i-1][j]$
      se $j \geq S[i-1]$ **então**
        $dp[i][j] \leftarrow dp[i][j]$ **or** $S[i-1][j]$
      fim se
    fim para
  fim para
  para $i \leftarrow metade$ até $0,\ i \leftarrow i-1$ **faça**
    se $dp[-1][i]$ **então**
      $somaSubconjunto \leftarrow i$
      break
    fim se
  fim para
  $indices \leftarrow []$
  para $i \leftarrow length(S)$ até $0,\ i \leftarrow i-1$ **faça**
    se $somaSubconjunto \geq S[i-1]$ **e** $dp[i-1][somaSubconjunto - S[i-1]]$ **então**
      $indices.append(i-1)$
      $somaSubconjunto \leftarrow somaSubconjunto - S[i-1]$
    fim se
  fim para
  retorna $indices$
fim função

4.3. Experimentos

Foram gerados SVMs a partir de seis conjuntos de features, com três contendo duas e três contendo quatro features distintas. Para cada conjunto foram gerados três SVMs: um a partir da resolução clássica do sistema linear para servir como base de comparação, um a partir do resultado de simulações do comportamento do circuito em computadores clássicos e um a partir de execuções em computadores quânticos reais. As simulações foram executadas sem ruído, representando um cenário ideal, e as execuções foram realizadas no *ibm_sherbrooke*, um computador quântico com 128 qubits disponibilizado gratuitamente pela IBM, sendo feitas 10.000 medições por execução em ambos os casos.

4.3.1. Resultados

Os experimentos foram avaliados em função da acurácia obtida pelos SVMs gerados, calculada por $\frac{\text{classificações corretas}}{\text{total de classificações}}$. A Tabela 2 apresenta os melhores resultados obtidos em cada um dos casos.

Ao se analisar individualmente cada linha da Tabela 2, nota-se que os SVMs gerados via simulação se mostraram superiores a suas versões clássicas em quatro dos seis possíveis casos, indicando uma tendência a apresentar resultados melhores do que o esperado em um cenário de execução próximo do ideal, enquanto os SVMs gerados por execuções em computadores quânticos reais se demonstraram superiores em dois casos, indicando que, apesar da influência do ruido, já é possível obter resultados melhores do que o esperado com o uso do HHL para a geração dos SVMs.

Observando as acurácias apresentadas na Tabela 2 de forma global, é possível perceber que a geração dos SVMs tanto através de simulação quanto execução em computadores quânticos

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

Tabela 2: Acurácias medidas pelos SVMs gerados classicamente, através de simulação e execução em computadores quânticos reais dos circuitos usando diferentes conjuntos de features.

Features	**Acurácia**		
	Clássico	**Simulação**	**Execução**
Profile_stdev e *Profile_skewness*	**96,56%**	**97,32%**	$90,81\%$
Profile_stdev e *Profile_kurtosis*	$96,15\%$	$96,62\%$	**96,58%**
Profile_mean e *Profile_stdev*	$95,61\%$	$95,5\%$	$93,47\%$
Profile_skewness, *Profile_kurtosis*, *DM_skewness* e *DM_kurtosis*	$94,28\%$	$94,8\%$	$90,22\%$
Profile_mean , *Profile_stdev*, *Profile_kurtosis* e *DM_kurtosis*	$89,13\%$	$95,67\%$	$95,43\%$
Profile_skewness, *Profile_kurtosis*, *DM_mean* e *DM_kurtosis*	$95,71\%$	$90,81\%$	$90,81\%$

reais conseguiram apresentar resultados que superam o melhor SVM gerado classicamente, demonstrando a eficiência do método proposto. Outro fator observado é que, mesmo com o uso de menos informação, os SVMs gerados através de pares de features apresentaram melhores resultados em todos os cenários.

5. Conclusão

Os resultados superiores encontrado pelos circuitos, tanto em simulações quanto execuções em computadores quânticos, em relação à versão clássica apontam uma possível vantagem do uso de computadores quânticos para a geração desses SVMs. Além das acurácias acima de $96,5\%$ encontradas indicarem a viabilidade do método para a classificação de candidatas a pulsares.

Uma vantagem do uso de SVMs é o seu baixo custo de inferência, sendo somente necessário aferir o valor da equação (5), resultando em condições ideais para utilização em cenários onde se precise se processar uma grande quantidade de dados com poder de processamento limitado.

Em conclusão, este artigo apresentou a classificação de candidatas a pulsares através do uso de SVMs gerados através de um método automatizado híbrido clássico-quântico.

6. Trabalhos futuros

Para trabalhos futuros, pretende-se reduzir a quantidade de operadores e qubits necessários para a geração dos circuitos através da busca por métodos mais otimizados de implementação dos componentes do algoritmo HHL. Além de se buscar métodos de mapeamento da matriz de Kernel para o circuito que não exijam o conhecimento prévio de seus autovalores.

7. Agradecimentos

Este trabalho foi desenvolvido com o apoio da Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ (260003/015313/2021) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES.

Referências

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., e Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215.

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

Cross, A. (2018). The ibm q experience and qiskit open-source quantum computing software. In *APS March meeting abstracts*, volume 2018, p. L58–003.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br

Duan, B., Yuan, J., Yu, C.-H., Huang, J., e Hsieh, C.-Y. (2020). A survey on hhl algorithm: From theory to application in quantum machine learning. *Physics Letters A*, 384(24):126595.

Gyongyosi, L. e Imre, S. (2019). A survey on quantum computing technology. *Computer Science Review*, 31:51–71.

Harrow, A. W., Hassidim, A., e Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502.

Larhmam, C. B.-S. . v. W. C. (2018). Maximum-margin hyperplane and margin for an svm trained on two classes. samples on margins are called support vectors. URL `https://commons.wikimedia.org/wiki/File:SVM_margin.png`. File: SVM_margin.png.

Lyon, R. (2017). HTRU2. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DK6R.

Lyon, R. J., Stappers, B., Cooper, S., Brooke, J. M., e Knowles, J. D. (2016). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123.

Morrell Jr, H. J., Zaman, A., e Wong, H. Y. (2021). Step-by-step hhl algorithm walkthrough to enhance the understanding of critical quantum computing concepts. *arXiv preprint arXiv:2108.09004*.

Nielsen, M. A. e Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

Pednault, E., Gunnels, J. A., Nannicini, G., Horesh, L., Magerlein, T., Solomonik, E., Draeger, E. W., Holland, E. T., e Wisnieff, R. (2017). Pareto-efficient quantum circuit simulation using tensor contraction deferral. *arXiv preprint arXiv:1710.05867*.

Pisinger, D. (1999). Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 33(1):1–14.

Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, 2:79.

Rebentrost, P., Mohseni, M., e Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503.

Saki, A. A., Alam, M., e Ghosh, S. (2021). Impact of noise on the resilience and the security of quantum computing. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, p. 186–191. IEEE.

Suykens, J. A. e Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9:293–300.

Tumulka, R. (2009). Dirac notation. In *Compendium of Quantum Physics*, p. 172–174. Springer.

Weinstein, Y. S., Pravia, M., Fortunato, E., Lloyd, S., e Cory, D. G. (2001). Implementation of the quantum fourier transform. *Physical review letters*, 86(9):1889.

Yang, J., Awan, A. J., e Vall-Llosera, G. (2019). Support vector machines on noisy intermediate scale quantum computers. *arXiv preprint arXiv:1909.11988*.

DOI: 10.59254/sbpo-2024-193639
https://proceedings.science/p/193639?lang=pt-br