DESEMPENHO DE ALGORITMOS QUÂNTICOS E CLÁSSICOS EM TREINAMENTO DE *MACHINE LEARNING* SUPERVISIONADO

Mariana Godoy Vazquez Miano[1]
Aleccheevina Silva de Oliveira[2]

DOI: 10.47283/244670492021090281

Resumo
Este artigo aborda a temática interdisciplinar da Computação Quântica com *Machine Learning*, duas tecnologias potencialmente capazes de realizarem mudanças em como a computação é realizada, resolvendo problemas inicialmente sem solução. O foco desta pesquisa foram aplicações da Computação Quântica que resultem em ganho de desempenho computacional em tarefas específicas de *Machine Learning*. O objetivo é analisar a viabilidade do uso de algoritmos quânticos para *Machine Learning*. Mais especificamente, analisar quais algoritmos quânticos podem ser aplicados a tarefas de *Machine Learning*, comparativamente com os algoritmos clássicos, na busca por melhor desempenho. Para o desenvolvimento da pesquisa, realizou-se uma revisão bibliográfica de algoritmos quânticos e na sequência, a implementação e verificação de desempenho do algoritmo quântico QSVM e sua correspondente versão clássica SVM, na aprendizagem supervisionada com os conjuntos de dados AD HOC e IRIS.
Palavra-chave: Computação Quântica. Algoritmos Quânticos. *Machine Learning*. Métodos de Aprendizagem Supervisionada. Desempenho.

Abstract
This article addresses the interdisciplinary theme of Quantum Computing with Machine Learning, two technologies potentially capable of making changes in how computing is performed, solving initially unsolvable problems. The focus of this research was Quantum Computing applications that result in computational performance gain in specific Machine Learning tasks. The objective is to analyze the feasibility of using quantum algorithms for Machine Learning. More specifically, to analyze which quantum algorithms can be applied to Machine Learning tasks, compared to classical algorithms, in the search for better performance. For the development of the research, a bibliographic review of quantum algorithms was carried out and, subsequently, the implementation and performance verification of the quantum algorithm QSVM and its corresponding classic version SVM, in supervised learning with the AD HOC and IRIS datasets.
Keywords: Quantum Computation. Quantum Algorithms. Machine Learning. Supervised Learning Methods.Performance.

[1] Docente e pesquisadora dos Cursos de Tecnologia da Informação da Fatec Americana. E-mail: mariana.miano@fatec.sp.gov.br
[2] Aluna do Curso de Tecnologia em Segurança da Informação da Fatec Americana. E-mail: aleccheevina.oliveira@fatec.sp.gov.br;

Introdução

A área da Computação Quântica está em pleno desenvolvimento, com altos investimentos em todo o mundo, por parte de grandes empresas como IBM, Microsoft, Google, Samsung e outras empresas parceiras que vêm se aliando a essas. No início de 2020, a IBM anunciou que atingiu um novo marco de desempenho, um Volume Quântico de 32 (IBM, 2020).

Quanto maior o Volume Quântico (QV), maior a complexidade dos problemas que os computadores quânticos podem solucionar, como, por exemplo, realizar simulações químicas maiores e mais precisas. Desde 2016 a IBM tem dobrado, todos os anos, o volume quântico de seus sistemas. A IBM expandiu, ainda, o IBM Q Network, que agora inclui mais de 100 organizações, em vários setores, incluindo companhias aéreas, automotivo, bancário e financeiro, energia, seguros, eletrônicos, startups e instituições acadêmicas e governamentais. As grandes empresas parceiras são Anthem e Delta Air Lines; start-ups: AIQTECH INC, BEIT, Quantum Machines, TradeTeq e Zurich Instruments; e as instituições acadêmicas e laboratórios de pesquisa do governo: Georgia Tech e Los Alamos National Laboratory.

Como se espera que a quantidade de dados gerados em nossa sociedade cresça mais rapidamente do que o aumento dos recursos computacionais, são necessárias formas mais poderosas de processamento de informações. A computação quântica usa as propriedades fundamentais da física quântica para redefinir a maneira como os computadores criam e manipulam informações. Essas propriedades implicam uma maneira radicalmente nova de computação, usando qubits em vez de bits, e oferecem a possibilidade de obter algoritmos quânticos que podem ser substancialmente mais rápidos que os algoritmos clássicos (KERENIDIS *et al*, 2019).

Tanto a computação quântica quanto o ML são campos em ascensão cuja união trará benefícios mútuos. Os conjunto de dados tratados por ML estão cada vez mais volumosos e o processo de análise exige muito recurso computacional. A proximidade com os limites físicos de fabricação de chips juntamente com o tamanho crescente de conjuntos de dados motiva a possibilidade de usar o poder da computação quântica para acelerar algoritmos clássicos de *Machine Learning*. A característica da sobreposição na computação quântica permite trabalhar com uma grande quantidade de informação e métodos de ML podem ser usados como pré-processamento de informações reduzindo o volume de dados (CILIBERTO *et al*, 2018).

Segundo Havlicek et al (2018), *Machine Learning* (ML) e Computação Quântica são duas tecnologias, cada uma com seu potencial para mudar como a computação é realizada para que problemas anteriormente sem solução possam ser resolvidos. Essa interseção entre aprendizado de máquina e computação quântica foi denominada *Quantum Machine Learning* (QML) e atraiu considerável atenção nos últimos anos.

De acordo com Biamonte et al (2018), com o aumento da potência do computador e pelos avanços algorítmicos, as técnicas de *Machine Learnin*g tornaram-se ferramentas poderosas para encontrar padrões nos dados. Como os sistemas quânticos produzem padrões contraintuitivos que não são eficientemente produzidos pelos sistemas clássicos, acredita-se que os computadores quânticos possam superar os computadores clássicos em tarefas de aprendizado de máquina. O campo do aprendizado de máquina quântica explora como criar e implementar software quântico concreto que ofereça tais vantagens. Trabalhos recentes deixaram claro que

os desafios de hardware e software ainda são consideráveis, mas também abriram caminhos para soluções.

Nesse artigo serão apresentadas abordagens do uso de algoritmos quânticos e suas versões clássicas em tarefas de *Machine Learning,* assim como a implementação e verificação de desempenho do algoritmo quântico QSVM e sua correspondente versão clássica SVM, na aprendizagem supervisionada com os conjuntos de dados AD HOC e IRIS.

1 PERSPECTIVA CLÁSSICA DE MACHINE LEARNING QUÂNTICO

Os algoritmos de aprendizagem de máquina estão chegando ao limite computacional devido ao aumento da quantidade de dados. Assim, a computação quântica, apresenta uma solução para esse problema ao usar os fenômenos quânticos, tais como a interferência e o emaranhamento, alinhados à Aprendizagem de Máquina. A esses algoritmos dá-se o nome de *Quantum Machine Learning* (QML). Até o momento (2018) não é possível afirmar um impacto significativo da aplicação dos algoritmos quânticos em *Machine Learning* (ML) *(*CILIBERTO et al**,** 2018).

A computação quântica é o estudo de processamento e armazenamento de informação quânticas cuja unidade fundamental, chamada de qubit, é representada por $\psi = \alpha_0 e_0 + \alpha_1 e_1$, em que $\alpha_0, \alpha_1 \in \mathbb{C}, |\alpha_0|^2 + |\alpha_1|^2 = 1$ e a probabilidade de valer 0 ou 1 é dado por $\{|\alpha_0|^2, |\alpha_1|^2\}$.

A complexidade computacional de uma tarefa é dada pelo número de etapas elementares, necessária para cumpri-la em função da entrada de dados. Normalmente, é considerado uma etapa elementar aquela em que o algoritmo precisa consultar um oraculo. Para denotar a eficiência do algoritmo usa-se a notação $O(f(n)$, e um algoritmo eficiente consulta esse oraculo um número polinomial de vezes. A eficiência de um programa QML ocorre porque o oraculo do algoritmo trata os dados em sobreposição.

Machine Learning pode prever o comportamento de um processo a partir de um conjunto de dados de entrada. Nos modelos Aprendizado Aproximadamente Correto (PAC) e a Teoria da Aprendizagem Estatística, a eficiência do sistema está relacionada à complexidade do tempo e da amostra. A complexidade do tempo se refere ao tempo de execução, já a complexidade da amostra se refere a quantidade mínima de exemplos o modelo de ML precisa para aprender a função. Em geral a eficiência de um algoritmo de aprendizado clássico de máquina é dada por $O(N^3)$.

O algoritmo de ML ideal é aquele que se ajusta bem aos dados de entrada (complexidade de amostra) e tenha um bom desempenho na previsão do comportamento (complexidade de tempo). Para isso, existem diversas abordagens, na computação clássica, para garantir que o algoritmo seja certeiro e eficiente: Parada Antecipada (executa um número limitado de etapas), Dividir e Conquistar (divide os dados de treinamento em diferentes computadores combinando os preditores gerados por eles) e uma abordagem mais recente que seleciona uma amostragem aleatória do preditores candidatos, reduzindo a quantidade de dados. Com essas abordagens é possível melhorar a eficiência dos algoritmos de ML, resultando em uma taxa de $O(N^2)$.

Por se tratar de análise de grandes quantidades de dados, a computação quântica pode ser útil ao *Machine Learning* devido a capacidade de sobreposição de dados. Ainda que não seja comprovado que a implementação de ML em computador quântico reduza a quantidade de dados necessário (complexidade da amostra), ao considerar um que um *learner* tenha acesso aos

dados em sobreposição e um oraculo, a melhoria da eficiência se dá pelo número de consultas ao oráculo (complexidade do tempo).

Um fator importante em QML é a transformação de dados clássicos em quânticos: de maneira geral a RAM Quântica pode carregar dados de maneira eficientes, mas os recursos necessários da implementação real não trazem uma vantagem. Ainda assim, alguns algoritmos de ML classicamente difíceis podem se beneficiar da QRAM além disso, existem outros protocolos para transformação de dados quânticos que podem ser usados.

Muitos métodos de QML são baseados em algoritmos quânticos de álgebra linear, como resolver equações lineares através de inversões de matrizes. O melhor algoritmo para resolução de sistemas de equações, tem uma eficiência de $O(N^{2.373})$, enquanto o Algoritmo do Sistema Linear Quântico resolve o problema com eficácia $\widetilde{O}(\log(N)k^2s^2/\epsilon)$.

Os métodos de otimização são fundamentais para algoritmos ML, a computação quântica fornece ferramentas tanto para a programação sem definida, tanto para a satisfação de restrição. Um outro campo de aplicação para a computação quântica, é os estudos de redes neurais artificiais, ainda que a linearidade da mecânica quântica e o desenvolvimento rápido das redes neurais clássicas impeça a maturidade de estudo na área.

Para concluir um ponto importante para o estudo QML é análise de ruídos: como o ruído pode aliviar problemas de ajuste de modelo em *Machine Learning* em computação clássica, espera-se que possa aproveitar os ruídos na computação quântica com a mesma finalidade.

1. 1 Machine Learning Quântico

A aceleração quântica é definida pela complexidade de consulta e de porta. A complexidade de consulta se trata de quantas vezes o algoritmo precisa consultar a fonte de informação, já a complexidade de porta se trata de quantas operações elementares ou portas lógicas são necessários para executar o programa. O primeiro passo para descobrir se um algoritmo quântico é eficiente identificar esses dois parâmetros (BIAMONTE et al, 2018).

Para aplicar esse conceito em *Machine Learning*, é preciso saber que existe uma variedade de algoritmos de ML com requisitos diferentes. Isto é, podem ser supervisionados, quando a coleção de dados é rotulada, não-supervisionado, quando os dados não são identificados e, pode ser uma combinação entre os dois.

Além disso, boa parte dos sistemas de ML utilizam operações em matrizes em espaços vetoriais de alta dimensões, base da qual parte a computação quântica. Isso porque os próprios qubits são representados em um vetor de duas dimensões e as operações são feitas através de matrizes de dimensão 2^n. Devido a essa propriedade os computadores quânticos são capazes de resolver equações lineares em tempo polinomial.

Como exemplo, tem-se o PCA (*Principal Component Analysis*), em que os dados são apresentados em um espaço vetorial v_j e a correlação entre os diferentes tipos de dados são dados pela matriz de covariância $C = \sum_j \vec{v}_j \vec{v}_j^T$. Os algoritmos PCA tem uma eficiência de $O(d^2)$, enquanto sua versão quântica tem a eficiência dada por $O((\log\ d)^2)$.

As máquinas de vetor suporte (SVM – *Support Vector Machine*) e Perceptrons são algoritmos quânticos supervisionados que separam dados em classes, as quais são separadas em hiperplanos. Nesse caso a coleção de dados disponibilizada para o computador quântico é processada com estimativa de fase e inversão de matriz.

Como *Machine Learning* reconhece os mesmos padrões estatístico em dados de entrada e nos resultados que produz, é de se esperar que computadores quânticos, por produzir dados que são difíceis de serem reconhecidos por computadores clássicos, também possam reconhecer dados difíceis de serem reconhecidos por programas de *Machine Learning* clássico.

No entanto, mesmo que no quesito de processamento a aceleração quântica seja perceptível, existem os chamados problema de entrada e problema de saída, que trazem uma desaceleração significativa devido à dificuldade de colocar dados clássicos em estados quânticos e em ler os dados em computadores clássicos depois de passarem por processadores quânticos. Como *Machine Learning* envolve uma coleção de dados grandes, o tempo para carregar esses dados em computadores quânticos é exponencial, e a alternativa de usar QRAM tem um custo muito alto. Assim, apesar do potencial da aceleração quântica, é preciso investir em um trabalho de otimização na entrada e saída de dados.

Enquanto esse problema persiste, é possível aplicar de maneira eficiente métodos de análise de dados em computadores quânticos para uma variedade de tarefas, como processar dados já em estado quânticos (estudo de propriedades da luz e da matéria) e reconstruir dinâmicas e estados em estudos com uso de simuladores quânticos.

Um outro desafio se trata da correção de erros quânticos, pois é preciso fazer ajustes nas portas para que a interferência seja minimizada e a eficiência otimizada. Algumas resoluções incluem usar supercondutores com átomos vizinhos mais próximos, construção de portas Toffoli de disparo único e o uso de algoritmos para simular a porta CNOT com qubits auxiliares e portas imperfeitas para suprimir erros digitais.

1.2 Demonstração da vantagem quântica em Machine Learning

A maior parte dos algoritmos quânticos envolve o uso de um oráculo, ou seja, uma espécie de caixa preta com a solução de um problema difícil de resolver para um computador clássico. A eficiência de um programa em termos de complexidade de consulta, trata-se de quantas vezes o algoritmo precisa consultar esse oráculo (RISTE et. al., 2017)

A tabela 1 apresenta a diferença de eficiência entre algoritmos clássicos e quânticos de ML. Nota-se que a versão quântica é exponencialmente mais eficiente do que a versão clássica. Isso porque como os dados estão em sobreposição, o número de passos chaves (denominado oráculo O) durante o treinamento em computador quântico é bem menor do que em um computador clássico. Somado a isso, muitos métodos de *Quantum Machine Learning* (QML) são baseados em algoritmos quânticos de álgebra linear como o HHL, com eficácia $\widetilde{O}(\log(N)k^2s^2/\epsilon)$.

Tabela 1 – Comparação em termos de consulta ao oráculo entre métodos de ML quânticos e clássicos

Algoritmo	**Versão Clássica**	**Versão Quântica**
K-médias	$O(N^3)$	$O(ndtk)$
SVM	$O(N^3)$	$\widetilde{O}(\log(nm)k^2/\epsilon)$
HHL	$O(N^{2.373})$	$\widetilde{O}(\log(N)k^2s^2/\epsilon)$

Fonte: Elaborada pelas autoras a partir de Duan (2018) e Ciliberto *et a*l (2018)

A eficiência dos métodos de QML é proporcional ao tamanho do conjunto de dados (n) e ao número de agrupamentos ou classes (k). No caso do SVM, o cálculo da eficiência leva em consideração a dimensão (m) do conjunto de dados (n) e a taxa de erro tolerada (∈) (CILIBERTO et al, 2018).

2 Algoritmos quânticos para machine learning

Boa parte dos algoritmos quânticos são baseados em três técnicas: estimativa de fase, estimativa de amplitude e simulação hamiltoniana (*PRAKASH, 2014, p.1).* Além disso, de acordo com Nielsen, existem duas classes de algoritmos que possibilitam a solução de problemas que exigem muito recursos computacional:

> "A primeira classe de algoritmos é baseada na transformada quântica de Fourier de *Shor* e inclui algoritmos notáveis para resolver os problemas de fatoração e logaritmo discreto, fornecendo uma aceleração exponencial impressionante sobre os algoritmos clássicos mais conhecidos. A segunda classe de algoritmos é baseada no algoritmo de *Grover* para realizar a pesquisa quântica. Eles fornecem uma aceleração quadrática menos impressionante, mas ainda notável, sobre os melhores algoritmos clássicos possíveis" (NIELSEN, CHUANG, p.172, 2010, *tradução nossa*)

A transformada quântica de Fourier mapeia um estado quântico, na base ortonomal Z (|0⟩ e |1⟩) para a base de Fourier X (|+⟩ e |-⟩) de acordo com a equação 1 a seguir:

$$y_k = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\Pi i \frac{jk}{N}} |\mathbf{y}\rangle \textit{(1)}$$

A Transformada de Fourier é importante para processamento de dados, por exemplo pode ser usada para transformar o som digital para reconhecimento de fala, no entanto, sua versão quântica tem limitações por não ser possível medir os estados na base de Fourier (NIELSEN & CHUANG, p. 217 – 220, 2010).

Uma das grandes contribuições da QFT é a sub-rotina de estimativa de fase, usada na maioria dos algoritmos quânticos uma vez que permite encontrar autovalores de operações unitárias, ou seja, permite estimar φ numa operação unitária $U|u\rangle = e^{2\pi i\varphi}|u\rangle$, na qual $e^{2\pi i\varphi}$ é um autovalor e |u⟩ é um autovetor. De maneira geral a estimativa de fase é obtida em três etapas, primeiro a fase de U é escrito em sobreposição na base de Fourier, depois são feitas várias rotações controladas nos *qubtis* em U até obter um número correspondente a φ que depois de ser passado para a base Z que pode ser medido (NIELSEN & CHUANG, p. 221-222, 2010).

O algoritmo de *Grover* encontrar um elemento em uma lista de dados não estruturada, ou seja, uma coleção de dados cuja estrutura e organização é desconhecido. Na computação clássica sua eficiência é em média $\frac{N}{2}$ e na pior das hipóteses N, em que N é o número de dados presente na lista de consulta. Já sua versão quântica tem complexidade de $\sqrt{N}$ (Nielsen, Chung, p. 248 - 249, 2010). O algoritmo de *Grover* é baseado em uma implementação em oraculo, em que traduz o problema de encontrar um elemento em encontrar seu índice dentro da lista. Dado uma lista de N elementos, o número de índice será representado por $N = 2^n$ e o número de soluções será M, tal que $1 \leq M \leq N$. O oráculo implementa uma função que retorna $f(x) = 1$ se x é a solução

do problema e $f(x) = 0$ se x não for a solução. Esse oraculo é dado por$|x> |q> \rightarrow |x> |q \otimes f(x)>$.

O algoritmo de *Grover* é extremamente útil para *Machine Learning*, uma vez que permite encontrar o mínimo ou o máximo em uma matriz desordenada (WITTEK, 2014).

2.1 K-medias e K-medianas

O k-media, assim como a maioria de algoritmos de agrupamento quânticos são baseados na pesquisa de Grover. O k-media, por exemplo, utiliza a pesquisa de Grover para calcular os centroides mais próximos. No caso do k-medianas o algoritmo de Grover é chamado para calcular a mediana de cada agrupamento e redistribuir todos os pontos de dados, para garantir que todos tenham pelo menos um elemento (WITTEK, 2014).

2.2 Máquina de Vetor Suporte e Regressão Linear

O método de aprendizagem máquina de vetor suporte (SVM - *Support Vector Machine*) é um exemplo de como os algoritmos de *Grover* e HLL podem oferecer aceleração quântica para *Machine Learning*. Isso porque o algoritmo de Grover pode ser usado para busca de mínimos quando as funções de custo são discretizadas no SVM. Além disso a eficiência desse método depende da rapidez em que é realizada a inversão de matriz quântica, da eficiência da simulação de matrizes esparsas e da rapidez que as matrizes não esparsas revelam sua auto estrutura.

De maneira geral, é um algoritmo classificador supervisionado que divide os pontos de dados em classes diferentes. Dado um conjunto de treinamento $D = \{(x_1, y_1), \dots, (x_m, y_m)\}$, com $\vec{x}_j \in \mathbb{R}^d$ e os rótulos $y_j \in \{+1, -1\}$ o SVM deve fazer uma separação otimizada no hiperplano $\vec{w}.\vec{x} + b$ entre duas classes de dados. Assim o treinamento deve resolver a equação abaixo:

$$argmax\ \alpha L(a) = \sum_{j=1}^{m} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{m} \alpha_j k_{jk} \alpha_k$$

em que $\sum_{j=1}^{m} \alpha_j = 0$, $y_j \alpha_j \geq 0$ e $K_{jk} = k(x_j x_k) = x_j \cdot x_k$ é função linear de kernel $k(x, x')$. A classificação dos dados é dada por $y(x) = sgn\left(\sum_{j=1}^{m} \alpha_j k(x_j, x) + b\right)$.

Uma vez que a classificação pode ser descrita em termos de programação linear, pode ser resolvida com inversão de matriz e nesse ponto HLL pode ser usado. Assim o SVM quântico pode ser treinado com inversão de matriz a seguir, dada pela equação:

$$F\begin{pmatrix} b \\ \alpha \end{pmatrix} \equiv \begin{pmatrix} 0 & 1^T \\ 1 & K + y^{-1}I \end{pmatrix}\begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}$$

Regressão linear trata-se de outro método de *Machine Learning*. Dado uma coleção de pontos de dados $(x_i y_i)_{i=1}^{N}$ onde $(x_{i1}, \dots., x_{im}) \in \mathbb{R}^m$ são vetores de dimensão M com entradas $y_i \in \mathbb{R}$ o algoritmo de regressão linear assume que existe uma função que relaciona $f(x) = w^T x$ caracterizada pelos parâmetros $w = (w_1, \dots, w_M)$. Existe uma versão desse algoritmo chamada Regressão Linear Ordinária Quântica (QOLR - *Quantum Ordinary Linear Regression*) com o parâmetro w dado por mínimos quadrados e inversão de matriz assim como o SVM.

2.2.1 Máquinas de Vetor Suporte (SVM)

Máquinas de Vetor Suporte é um método de *Machine Learning* fundamentado na Teoria da Aprendizagem Estatística, originalmente desenvolvida para classificação binária através da construção de hiperplano de decisão.

> "A Teoria do Aprendizado Estatístico visa estabelecer condições matemáticas que permitem escolher um classificador, com bom desempenho, para o conjunto de dados disponíveis para treinamento e teste. Em outras palavras esta teoria busca encontrar um bom classificador levando em consideração todo o conjunto de dados, porém se abstendo de casos particulares" (GONÇALVES, 2010, p.3).

Os algoritmos de SVM buscam construir um classificador f que produza a menor quantidade de predições erradas possíveis durante o treinamento. Desse modo é definida uma taxa de erro mínimo (erro empírico) $R_{emp}f(x)$, ou seja, a diferença entre a resposta correta e a resposta obtida. Essa taxa é chamada de erro empírico e é dada pela equação a seguir:

$$R_{emp}f(x) = \frac{1}{N}\sum_{i=1}^{n} c(f(x_i), y_i)$$

Em que $c(\,)$ é a função de custo da previsão de $f(x_i)$ em relação ao resultado y_i . E processo de minimização do custo é dado pelo sistema linear:

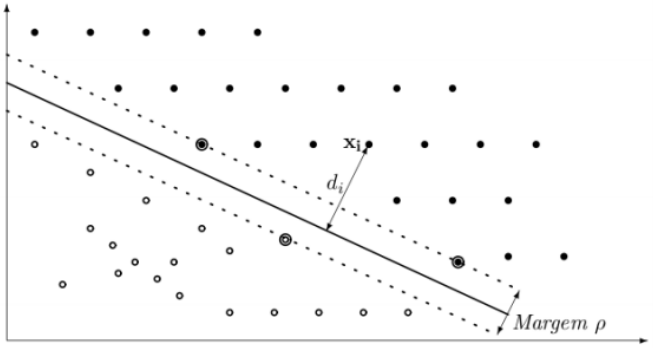
$$c(f(x_i), y_i) = \begin{cases} 1, & se\ y_i f(x) < 0 \\ 0, & se\ y_i f(x) \geq 0 \end{cases} (11)$$

Dado um hiperplano, $f(x) = (\ w\ \cdot x) + b$, define-se como margem a menor distância entre vetores de treinamento e o hiperplano, além disso, a esses dados mais próximos de $f(x)$, dá -se o nome de vetores suportes. A margem ótima é dada pela equação a seguir:

$$p = \min y_i\, f(x_i) (12)$$

E é representada pela figura 1 abaixo, em que os vetores de suporte, ou seja, os dados críticos que criam sozinhos um hiperplano ótimo está em cima da linha pontilhada. Os vetores x_i são dados não críticos e que podem ser descartados do conjunto de treinamento, sem afetar os resultados.

Figura 1: Margem do Hiperplano de Separação



Fonte: GOLÇALVES, 2010, p. 6

A distância euclidiana entre os vetores suportes e o hiperplano é dado por d_ para os vetores negativos e d+ para os vetores positivos, sendo a margem p dada pela equação abaixo:

$$p = (d_- + d_+)$$

Considerando essas definições e o hiperplano ótimo de uma classe linearmente separável dada pela equação abaixo:

$$< w \cdot x > \ + b = 0$$

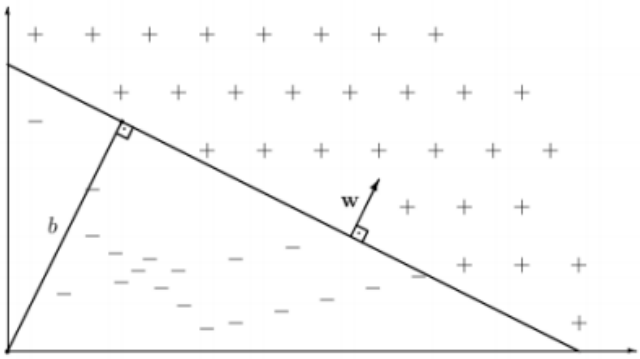
em que w é um vetor peso e x o vetor bias. A margem ideal é representada por:

$$y_i(< w \cdot x > \ + b) \geq \ +1, i = \{1,2, \dots, n\}$$

E a distância x_i ao hiperplano (w, b) como

$$d_i(w, b; x_i)$$

Figura 2: Vetores bias e peso no hiperplano de separação



Fonte: GOLÇALVES, 2010, p. 7

Note que o vetor peso (w) é responsável pela direção perpendicular ao hiperplano, enquanto o vetor bias (erro sistemático, ou seja, a diferença permitida entre os dados reais e os dados medidos) é responsável pelo movimento paralelo do hiperplano sobre si mesmo.

A equação a seguir define a otimização do hiperplano:

$$d_i(w, b; x_i) = \frac{|< w \cdot x > + b|}{\|w\|} = \frac{y_i(< w \cdot x > + b)}{\|w\|}$$

Que dada a restrição $y_i(< w \cdot x > \ + b) \geq \ +1$, pode ser reescrita na seguinte equação:

$$d_i(w, b; x_i) \geq \frac{1}{\|w\|}$$

Isto é, para cada lado do hiperplano. Considerando a margem total como a soma das distâncias Euclidianas, obtém -se a equação:

$$d_- = \ d_+ = \frac{1}{\|w\|} \text{ e } p = (d_- + \ d_+) = \frac{2}{\|w\|}$$

Na qual o hiperplano ótimo pode ser obtido através da normalização de w.

A vantagem quântica para esse método trata-se da otimização poder ser feita com utilizando inversão de matrizes com o algoritmo quântico HHL (DUAN et al, 2020, p. 24). Além disso, a aceleração quântica mapeando o espaço de características da máquina de vetor suporte para um computador quântico (KERENIDIS, 2018, p.22).

2.3 Análise de Componentes Principais - PCA

Análise de Componentes Principais é um método de machine learning usado para redução em massa de dados, sem perda de informações. Essa técnica é usada para agrupamento de indivíduos e geração de índices (VARELLA, 2008, p. 3 – 7).

> "A análise de componentes principais é uma técnica da estatística multivariada que consiste em transformar um conjunto de variáveis originais em outro conjunto de variáveis de mesma dimensão denominadas de componentes principais. Os componentes principais apresentam propriedades importantes: cada componente principal é uma combinação linear de todas as variáveis originais, são independentes entre si e estimados com o propósito de reter, em ordem de estimação, o máximo de informação, em termos da variação total contida nos dados" (VARELLA, 2008, p.3)

Dado um conjunto de características p de n indivíduos de uma população π, as características observadas são denominadas X. Assim tem se a matriz de dados X de ordem $n \times p$. O objetivo é passar o conjunto de variáveis X para o conjunto para o conjunto de variáveis Y não correlacionadas e com variâncias ordenadas.

Figura 3: Matriz de dados X de ordem n × p.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2p} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x & x_{n3} & \cdots & x_{np} \end{bmatrix}$$

Fonte: VARELLA, 2008, p. 4

A partir da matriz de dados X é feita a matriz de covariância $\sum$ da população π.

Figura 4: Matriz de covariância S de ordem p.

$$S = \begin{bmatrix} \hat{\mathrm{V}}\mathrm{ar}(x_1) & \hat{\mathrm{C}}\mathrm{ov}(x_1x_2) & \hat{\mathrm{C}}\mathrm{ov}(x_1x_3) & \cdots & \hat{\mathrm{C}}\mathrm{ov}(x_1x_p) \\ \hat{\mathrm{C}}\mathrm{ov}(x_2x_1) & \hat{\mathrm{V}}\mathrm{ar}(x_2) & \hat{\mathrm{C}}\mathrm{ov}(x_2x_3) & \cdots & \hat{\mathrm{C}}\mathrm{ov}(x_2x_p) \\ \hat{\mathrm{C}}\mathrm{ov}(x_3x_1) & \hat{\mathrm{C}}\mathrm{ov}(x_3x_2) & \hat{\mathrm{V}}\mathrm{ar}(x_3) & \cdots & \hat{\mathrm{C}}\mathrm{ov}(x_3x_p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\mathrm{C}}\mathrm{ov}(x_px_1) & \hat{\mathrm{C}}\mathrm{ov}(x_px_2) & \hat{\mathrm{C}}\mathrm{ov}(x_px_3) & \cdots & \hat{\mathrm{V}}\mathrm{ar}(x_p) \end{bmatrix}$$

Fonte: VARELLA, 2008, p. 4

Quando as unidades de medidas das variáveis X_j são diferentes entre si é preciso padronizar esses dados, com média zero e variância 1 ou média 1 e variância qualquer. Após essa etapa é gerada a matriz de dados padronizados Z.

Figura 5 : Matriz de dados padronizados Z de ordem n × p.

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} & \cdots & z_{1p} \\ z_{21} & z_{22} & z_{23} & \cdots & z_{2p} \\ z_{31} & z_{32} & z_{33} & \cdots & z_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & z_{n3} & \cdots & z_{np} \end{bmatrix}$$

Fonte: VARELLA, 2008, p. 5

Os componentes principais são obtidos a partir da matriz de correlação R, isto é, resolvendo a equação característica $\det[R - \lambda I] = 0$ ou $|R - \lambda I| = 0$.

Figura 6: Matriz de correlação R de ordem p

$$R = \begin{bmatrix} 1 & r(x_1x_2) & r(x_1x_3) & \cdots & r(x_1x_p) \\ r(x_2x_1) & 1 & r(x_2x_3) & \cdots & r(x_2x_p) \\ r(x_3x_1) & r(x_3x_2) & 1 & \cdots & r(x_3x_p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r(x_px_1) & r(x_px_2) & r(x_px_3) & \cdots & 1 \end{bmatrix}$$

Fonte: VARELLA, 2008, p. 6

Se a matriz R não tiver nenhuma coluna que seja combinação linear de outras, então a equação $R - \lambda I| = 0$ terá p raízes representadas por λ_i chamadas de autovalores de R. Para cada autovalor existe um autovetor $\tilde{a}$ normalizado e ortogonal. E assim o i-*ésimo* componente principal é dado pela equação 20:

$$Y_i = a_{i1} X_1 + a_{i2} X_2 + \ldots + a_{ip} X_p \quad (20)$$

É ressaltar a importância do algoritmo PCA para tornar a entrada de dados mais eficiente uma vez que reduz o conjunto de dados sem diminuir a informação, podendo ser usado para otimizar outros algoritmos para machine learning. Além disso, esse método pode ser usado para contornar o problema de entrada e saída de dados em computadores quânticos, uma vez que uma menor quantidade de dados exige menos QRAM. Isto foi mostrado na construção e teste do algoritmo *Q-means*, no qual o conjunto de teste passou por um pré-processamento com redução de dimensão usando PCA (KERENIDIS, 2018, p.22).

Ainda em relação a computação quântica, percebe-se que o PCA por sua característica matricial se beneficia, por essa ter suas operações também baseadas em matrizes.

> "Uma ampla variedade de análises de dados e protocolos de aprendizado de máquina operam realizando operações de matriz em vetores em um espaço vetorial de alta dimensão. Mas a mecânica quântica tem tudo a ver com operações de matriz em vetores em espaços vetoriais de alta

> dimensão. O ingrediente chave por trás desses métodos é que o estado quântico de n bits ou qubits quânticos é um vetor em um espaço vetorial complexo 2^n - dimensional; operações de lógica quântica ou medições realizadas em qubits multiplica o vetor de estado correspondente por matrizes $2^n \times 2^n$ (BIAMONTE, 2018, p. 5, tradução nossa).''

Os algoritmos PCA tem uma eficiência de $O(d^2)$, já a sua versão quântica eficiência dada por $O((\log\ d)^2)$. Isto é, mapeando o vetor de dados v_j em um espaço vetorial quântico usando QRAM para gerar uma matriz de densidade relativa à matriz de covariância para dados clássicos, é possível determinar os autovalores e autovetores para descobrir os componentes principais combinando a exponenciação da matriz de densidade com o algoritmo de estimativa de fase. (BIAMONTE, 2018, p. 6).

3 Dados

É preciso ressaltar que a eficiência de algoritmos de *Machine Learning Quântico* (QML) dependem também da velocidade em que são carregados no computador quântico. Há diferenças nas situações em que os dados já chegam em um estado quântico ou se será necessário convertê-los. Atualmente, são três alternativas para o problema da entrada de dados (carregar dados clássicos em um computador quântico).

A primeira opção, chamada de codificação digital ou codificação de base, trata-se de converter os dados de sua forma binária para qubits com o uso do operador NOT; no entanto, o número de qubits é diretamente proporcional ao volume de dados. A segunda opção é conhecida como codificação analógica e os vetores de dados clássicos são codificados em amplitudes dos qubits. Nessa técnica, o número necessário é logarítmico polinomial ao tamanho de dados. A terceira técnica trata-se de codificar os dados clássicos em um estado hamiltoniano, ajustando os parâmetros físicos de alguns qubits (mas esse método é usado geralmente em métodos de amostragem ou otimização) (DUAN et al, 2020, p.2 - 5).

Em 2008 foi criado uma arquitetura para memória RAM quântica (QRAM - *Quantum Randon Memory Access*) que permite a consulta de endereços em sobreposição e retorna o dado correspondente na enésima célula de memória. De maneira geral, a QRAM é usada na codificação de dados digital, no entanto, é possível usar a codificação analógica carregando dados através de estruturas de árvores binárias.

4 Implementação

4.1. Classificadores

O *SciKit Learn* foi usado para gerar o classificador na versão clássica da máquina de vetor suporte e para treinamento do classificador tanto clássico quanto quântico (QISKIT MACHINE LEARNING TUTORIAL).

O método para gerar o classificador no *scikit learn* é o SVC abreviação de *Suport Vector Classifier* ou em português Classificador de Vetor Suporte, esse método pertence a biblioteca *Python* SVM. O trecho de código abaixo exemplifica a construção do classificador clássico:

```
classificador_classico = SVC()
```

Para criar o classificador usando recursos quânticos é preciso de duas etapas, primeiro é criado o mapa de recursos quântico com o método *ZZFeatureMap* e depois é gerado o kernel quântico com o método *QuantumKernel*.

mapa_recursos = ZZFeatureMap()
kernel_quantico = QuantumKernel()

Depois dessa etapa o kernel gerado pode ser passado para o método QSVC que irá criar o classificador usando os recursos do computador de quântico.

classificador_quantico = QSVC()

É importante ressaltar que o kernel quântico também pode ser usado com o classificador clássico no método SVC.

Após gerar os classificadores, é usado o método *fit* do *scikit learn* para treiná-los (com o conjunto de treinamento) e na sequência, o método *predict* com o conjunto de teste (para verificar a eficiência do classificador).

4.2 Treinamento de Dados

Neste tópico serão apresentados os processos de treinamento supervisionado com os Conjuntos de Dados AD HOC e DIGITS.

4.2.1 Treinamento do Conjunto de dados AD HOC

O passo a passo para o treinamento com o conjunto AD HOC foi baseado no exemplo disponível pela própria ferramenta Qiskit e pelo artigo correspondente *"Supervised learning with quantum enhanced feature spaces"* (HAVCELIK et al, 2018).Inicialmente, realizou-se a importação do conjunto de dados disponível na biblioteca *Qiskit_Machine_Learning* como mostrado na figura 7 e 8. Com a importação do conjunto de dados no Qiskit, já é gerado o conjunto de treinamento, os rótulos de treinamento, o conjunto de teste e o rótulo de teste (QISKIT Documentation).

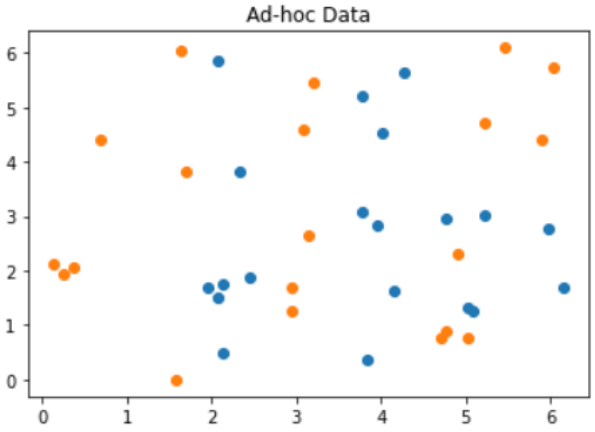
Figura 7: Conjunto de dados Ad Hoc

```
from qiskit_machine_learning.datasets import ad_hoc_data
adhoc_dimension = 2
train_features, train_labels, test_features, test_labels, adhoc_total = ad_hoc_data(
    training_size=20,
    test_size=5,
    n=adhoc_dimension,
    gap=0.3,
    plot_data=True, one_hot=False, include_sample_total=True
)
```

Fonte: próprio autor.

Ao importar o conjunto de dados é possível gerar o gráfico para facilitar a visualização do conjunto de dados e suas classes (através do parâmetro plot_data mostrado na figura 7). A figura 8 apresenta o gráfico de dispersão do conjunto Ad Hoc.

Figura 8: Gráfico do Conjunto Ad Hoc



Fonte: próprio autor.

Após essa etapa, realizou-se a construção do classificador com o método SVM e depois o treinamento do classificador com o metodo *fit*. Para mostrar os dados e os rótulos previsto durante o treinamento, foi usado o método *predict* como mostra a figura 9. Todos esses métodos pertecem à biblioteca Scykit Learn.

Figura 9: Treinamento do classificador clássico

```
from sklearn import svm
classficador_sklearn_svm = svm.SVC(C=1.0)
classficador_sklearn_svm.fit(train_features,train_labels)
classficador_sklearn_svm.predict(test_features)
previsao = classficador_sklearn_svm.predict(test_features)
print(previsao)
```

```
[0 0 0 0 0 0 1 0 1 0]
```

Fonte: próprio autor.

A figura 10 mostra a construção do kernel quântico. Primeiramente, é preciso criar o mapa de recursos, ou seja, os dados e seus rótulos que serão para treinamento com o método *ZZfeatureMap* e depois é gerado o kernel com o método *QuantumKernel*.

Figura10: 7 Espaço de Recursos Quântico

```
adhoc_feature_map = ZZFeatureMap(feature_dimension=adhoc_dimension,
                                 reps=2, entanglement='linear')

adhoc_backend = QuantumInstance(BasicAer.get_backend('qasm_simulator'), shots=1024,
                                seed_simulator=seed, seed_transpiler=seed)

adhoc_kernel = QuantumKernel(feature_map=adhoc_feature_map, quantum_instance=adhoc_backend)
```

Fonte: próprio autor.

Também foi testado o metodo clássico SVC no kernel quântico criado, como mostra a figura 11:

Figura 11: Treinamento do kernel quântico com métodos clássicos

```
adhoc_svc = SVC(kernel=adhoc_kernel.evaluate)
adhoc_svc.fit(train_features, train_labels)
adhoc_svc.predict(test_features)
previsao_qkernel = adhoc_svc.predict(test_features)
print(previsao_qkernel)
```

```
[0 0 0 0 0 1 1 1 1 1]
```

Fonte: próprio autor.

E por último realizou-se o treinamento construindo-se um classificador com a função quântica *qsvc*(QISKIT), como mostra a figura 12:

Figura 12: Classificador Quântico

```
qsvc = QSVC(quantum_kernel=adhoc_kernel)
qsvc.fit(train_features, train_labels)
qsvc.predict(test_features)
previsao_qsvc = qsvc.predict(test_features)
print(previsao_qsvc)
```

```
[0 0 0 0 0 1 1 1 1 1]
```

Fonte: próprio autor.

Para medir os resultados, foi utilizado o módulo métrica do *sklearn*. Verificou-se que a classificação é mais eficiente em termo de acuracidade quando usado o kernel criado no computador quântico com a função *ZZfeatureMap*. Esse resultado é mostrado nas figuras 13, 14, 15 e 16:

Figura 13: Métricas de Resultados

```
from sklearn.metrics import classification_report
target_names = ['class 0', 'class 1']
```

Fonte: próprio autor.

Figura 14: Resultados do Treinamento Classificador Clássico

```
print(classification_report(test_labels, previsao, target_names = target_names))
```

	precision	recall	f1-score	support
class 0	0.62	1.00	0.77	5
class 1	1.00	0.40	0.57	5
accuracy			0.70	10
macro avg	0.81	0.70	0.67	10
weighted avg	0.81	0.70	0.67	10

Fonte: próprio autor.

Figura 15: Resultados do treinamento do classificador quântico

```
print(classification_report(test_labels, previsao_qkernel, target_names = target_names))
```

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	5
class 1	1.00	1.00	1.00	5
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10

Fonte: próprio autor.

Figura 16: Resultados do treinamento com classificador clássico e kernel quântico

```
print(classification_report(test_labels, previsao_qsvc, target_names = target_names))
```

	precision	recall	f1-score	support
class 0	1.00	1.00	1.00	5
class 1	1.00	1.00	1.00	5
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10

Fonte: próprio autor.

Para o conjunto de dados AD HOC, obteve-se a acuracidade (taxa de acerto) de 100 % para a computação quântica e 70% para a computação clássica.

4.2.2 Treinamento do Conjunto de dados IRIS

Para treinar o conjunto de dados Iris, utilizou-se o mesmo modelo do conjunto de dados *AD HOC*. No entanto, foi preciso usar a versão mais antiga da importação do conjunto de dados Iris para fazer duas alterações. A primeira alteração diz respeito ao formato dos dados dos rótulos, pois na versão atual do Qiskit os rótulos são dados em lista dentro do *array*, mas para passar para o método SVM do *scikit learn* era preciso que esses rótulos fossem do tipo inteiro. Na versão antiga desse conjunto de dados, foi possível realizar essa alteração ao marcar o

método *one hot* como *false*, uma vez que esse método retorna o conjunto de rótulo como uma matriz quando marcado como *true* (Github QISKIT).

A segunda alteração diz respeito à própria classificação, uma vez que na computação quântica, dados os recursos atuais, a classificação binária é mais eficiente. Assim, ao invés de usar os três conjuntos de espécie da planta Iris, usamos somente 2 espécies do conjunto. Para que a comparação fosse justa, usamos o conjunto modificado para treinamento tanto no método clássico quanto no método quântico (DUAN, B. et. al, 2018). A importação do conjunto de dados e os resultados do treinamento podem ser visualizados em partes do código implementado (figuras 17, 18 e 19).

Figura 17: Importação do conjunto de treinamento Iris

```
#alterei o nome da função para my_iris e one_hot para False
def my_iris(training_size, test_size, n, plot_data=False, one_hot=False):
    """returns iris dataset"""
    class_labels = [r"A", r"B"]
    data, target = datasets.load_iris(return_X_y=True)
    sample_train, sample_test, label_train, label_test = train_test_split(
        data, target, test_size=test_size, random_state=42
    )
```

Fonte: próprio autor.

Figura 18: Resultado do treinamento com classificador quântico

```
In [59]: print(classification_report(q_class_labels, y_pred, target_names = target_names))
```

	precision	recall	f1-score	support
Class A	1.00	0.58	0.73	19
Class B	0.62	1.00	0.76	13
accuracy			0.75	32
macro avg	0.81	0.79	0.75	32
weighted avg	0.85	0.75	0.75	32

Fonte: próprio autor.

Figura 19: Resultado do treinamento com classificador quântico

```
print(classification_report(q_class_labels, y_pred_svc, target_names = target_names))
```

	precision	recall	f1-score	support
Class A	1.00	1.00	1.00	19
Class B	1.00	1.00	1.00	13
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

Fonte: próprio autor.

Para o conjunto de dados IRIS, obteve-se a acuracidade (taxa de acerto) de 75 % para a computação quântica e 100% para a computação clássica.

Conclusão

Com base nos resultados observados, a partir da implementação dos algoritmos QSVM (quântico) usando o kernel quântico para ML disponível no *IBM Quantum Experience* e o framework *Qiskit* para computação quântica na linguagem de programação *Python*, e da implementação do SVM (clássico), conclui-se que a aceleração quântica no campo de ML otimizará o processamento de dados, com uma escolha "acertada" do conjunto de dados para teste.

Utilizando- se o conjunto de Dados AD HOC, observa-se claramente a vantagem quântica nos testes de acuracidade (taxa de acerto) em relação ao *Machine Learning* supervisionado: 100 % para a computação quântica e 70% para a computação clássica. Vale ressaltar que esse conjunto foi utilizado integralmente (sem nenhum tipo de alteração).

Em contrapartida, utilizando-se o conjunto de Dados IRIS, a computação clássica se mostrou mais vantajosa (100% de acuracidade), em relação à computação quântica (70%). Acredita-se que esse resultado ocorreu devido à necessidade das duas alterações realizadas no conjunto IRIS : primeiramente, em relação ao formato dos dados dos rótulos, pois na versão atual do Qiskit os rótulos são dados em lista dentro do *array*, mas para passar para o método SVM do *scikit learn* era preciso que esses rótulos fossem do tipo inteiro; e a segunda alteração, relacionada à própria classificação, uma vez que na computação quântica, dados os recursos atuais, a classificação binária é mais eficiente. Assim, ao invés de usar os três conjuntos de espécie da planta Iris, utilizou-se somente duas espécies do conjunto.

Houve a tentativa do uso de um terceiro conjunto de dados (Digits), seguindo o mesmo modelo das implementações com o AD HOC e IRIS, no entanto, não foi possível gerar o kernel em computador quântico devido ao tamanho do conjunto de dados que possui imagens. Ao tentar gerar esse kernel direto pelo *IBM-Q*, extrapolou-se o limite de 8GB de memória disponibilizado pela ferramenta. Já ao tentar simular localmente no computador pessoal com a biblioteca Qiskit instalada, o programa trava na criação do kernel e não consegue concluir.

Como indicação de pesquisa futuras, acredita-se que a escolha de novos conjuntos de dados e a disponibilização de simuladores quânticos com mais memória (para geração do kernel) mostrará com mais realidade a "supremacia quântica" em termos de desempenho em *Machine Learning*, como demonstrado na teoria da computação quântica.

Referências

BIAMONTE, J.; WITTEK, P.; PANCOTTI, N.; el al. ***Quantum Machine Learning*.** Nature 549, 195–202 (2017). https://doi.org/10.1038/nature23474

CILIBERTO, C.; HERBSTER, M.; IALONGO, A.D.; PONTIL. M.; ROCCHETTO, A.; SEVERINI, S.; WOSSNIG, L. ***Quantum machine learning: a classical perspective.*** *Proceedings Royal Society* A 474: 20170551.2018. https://doi.org/10.1098/rspa.2017.0551

DUAN, B. **A survey on HHL algorithm: From theory to application in quantum machine learning**. In. Elservier, v. 384, 2018. Disponível em:

https://www.sciencedirect.com/science/article/pii/S037596012030462X. Acesso em 29 mar 2021.

GONÇALVES, A. R. **Máquinas de Vetores Suporte**. 2010, p. 18. Disponível em: https://andreric.github.io/files/pdfs/svm.pdf. Acesso em 30 mai 2021.

HAVLICEK, V.; CORCOLES, A. D.; TEMME, K. et al. ***Supervised learning with quantum enhanced feature spaces.*** *Nature* **567,** 209–212 (2019). https://doi.org/10.1038/s41586-019-0980-2

IBM. **Do laboratório para a indústria: IBM anuncia últimos avanços em computação quântica.** IBM Inovação, 2020. Disponível em: https://www.ibm.com/blogs/ibm-comunica/do-laboratorio-para-a-industria-ibm-anuncia-ultimos-avancos-em-computacao-quantica

KERENIDIS, I.; LANDMAN, J.; LUONGO, A.; PRAKASH, A. *q-means: A quantum algorithm for unsupervised machine learning. In: 33rd CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NEURIPS 2019), Vancouver, Canada*. 2019.

NIELSEN, M. A; CHUANG, I. L. **Quantum Computation and Quantum Information**. Cambridge: Cambridge University Press, 2010. 698 p.

PRAKASH**,** A**. Quantum Algorithms for Linear Algebra and Machine Learning**. 2014, p.90. Dissertação (Doutorado em Filosofia em Ciência da Computação) - Ciência da Computação, Universidade da California. Berkeley, p.90, 2014.

QISKIT DOCUMENTATION. **Qiskit Machine Learning API Reference**. Disponível em: https://qiskit.org/documentation/machine-learning/tutorials/03_quantum_kernel.html Acesso em: 07 de nov. de 2021.

RISTÈ, D.; da SILVA, M. P.; RYAN, C. A. et al***. Demonstration of quantum advantage in machine learning***. *npj Quantum Information* **3,** 16 (2017). https://doi.org/10.1038/s41534-017-0017-3

VARELLA, C. A. A. **Análise de Componentes Principais: Análise Multivariada Aplicada às Ciências Agrárias**. Pós-graduação em agronomia, Universidade Federal Rural do Rio de Janeiro. Seropédica, p. 12, 2008.

WITTEK, P. **Quantum Machine Learning. What Quantum Computing Means to Data Mining**. Academic Express. P. 196, 2014.