# Implementação do Perceptron Quântico para a Classificação Binária de imagens.

Davi Y. M. Brandão
Centro de Informática, Universidade
Federal de Pernambuco
Recife, Brasil
Email: dymb@cin.ufpe.br

ABRIL 2022

***Resumo*—Esse artigo apresenta uma implementação do Perceptron, algoritmo de classificação binária amplamente utilizado na aprendizagem de máquina, por meio de elementos da computação quântica. (*Resumo*)**

***Palavras-chave—aprendizagem de máquina, computação quântica, perceptron, spsa (****palavras-chave****)***

## I Introdução

Uma das grandes vantagens de um modelo quântico de computação em relação à sua versão clássica é a capacidade de codificar dados exponencialmente. Isto é, para um número *N* de bits clássicos, são necessários apenas *log2(N)* qubits para representar o dado em um computador quântico.

Com isso em mente, é natural imaginar que a área de *aprendizagem de máquina* (M.L) poderia prosperar na computação quântica. Afinal, algoritmos de M.L usualmente utilizam uma enorme quantidade de dados, além de necessitarem de um alto volume de vetorização.

Portanto, seguindo a modelização de um neurônio artificial quântico proposto por (MANGINI, 2020), esse artigo apresentará um modelo introdutório completo do Perceptron clássico, para o problema de classificação 0-1 em imagens.

## II Explicação do modelo

### *II.A Arquitetura Geral*

Neste artigo, avaliamos o modelo a partir de três etapas sequenciais:

- Preparação dos dados.
- Treinamento.
- Testes.

A arquitetura do modelo segue o padrão para perceptrons de uma camada, com apenas um neurônio. Implementações mais complexas, com múltiplas camadas de múltiplos neurônios, podem ser montadas a partir da nossa proposição com facilidade. Porém, devido às limitações atuais de hardware, tais implementações se tornam extremamente inviáveis.

Na figura 1, podemos observar uma abstração em fluxograma do algoritmo. O objetivo de um Perceptron classificador é obter um vetor de pesos $W = [w_1, w_2,..,w_n]$ para um input $I = [I_1, I_2,..,I_n]$ tal que A(I,W) corresponda a um valor Y binário que indica se I pertence ou não à determinada classe.

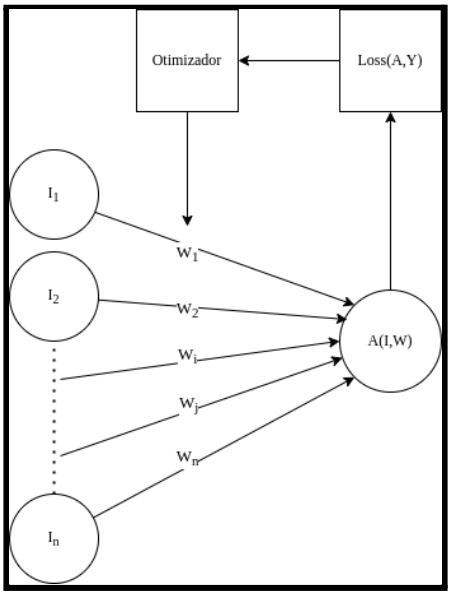


**Figura 1**: Arquitetura do modelo

Para isso, realiza-se o *Feedforward,* enviando o vetor de input I ao perceptron. Após o cálculo de A(I,W), utiliza-se de uma função de custo Loss(A,Y) para calcular a diferença estatística entre A e Y. O resultado é passado para um otimizador, que atualiza o vetor W na tentativa de minimizar Loss(A,Y), no chamado *Backpropagation.*

O algoritmo é então repetido até que uma precisão satisfatória seja alcançada (ou seja, Loss(A,Y) é minimizado suficientemente).

Em um computador quântico, os vetores I e W são equivalentes a *n* qubits em paralelo, capazes de representar $2^n$ dados. Utilizando a função de ativação sugerida por (MANGINI, 2020), temos o *Feedforward* completamente executado em circuitos quânticos.

O *Backpropagation,* entretanto, é efetuado utilizando um otimizador ideal para funções que sofrem com ruído estatístico relevante: o SPSA (sigla para *Simultaneous perturbation stochastic approximation)*. Isso se dá pois o cálculo de A(I,W) é realizado a partir de um número de medições do sistema quântico, que é naturalmente probabilístico.

Chamaremos então de "treinamento" a etapa de iterações consecutivas de *Feedforward* e *Backpropagation* em um input $I_{treino}$ fixo. Após a otimização de W, realizaremos o *Feedforward* em um vetor $I_{teste}$ e avaliamos a sua precisão na etapa de "teste".

Iremos definir com detalhes o funcionamento de cada uma das etapas nas seções a seguir.

### *II.B Preparação dos Dados*

O modelo trabalhará com *M = 800* amostras de imagens do banco de dados, tanto para treino quanto para teste. Esse valor pode ser aumentado arbitrariamente no código.

Um processamento das imagens é necessário, a fim de tornar as informações de cada imagem em um vetor $I_i$ de número reais. Nesse modelo, escolhemos converter as imagens para escala de cinza, e reduzir seu tamanho para a resolução *16 x 16*, sendo necessário *256* valores inteiros para representar cada imagem. Esses valores também podem ser modificados arbitrariamente no código, os valores apresentados foram escolhidos para realizar os experimentos em tempo razoável.

Enviamos ao modelo um vetor $I = [I_1, I_2,..,I_{500}]$, onde $I_i$ representa a *i-ésima* imagem do conjunto, e com um vetor de booleanos $Y = [Y_1, Y_2,..,Y_{500}]$, onde $Y_i$ indica se a imagem $I_i$ pertence ou não a classe desejada.

### *II.C Treinamento*

Para realizar o treinamento, o circuito quântico proposto em (MANGINI, 2020) é simulado *8192* vezes para cada imagem do vetor I. Como cada imagem é representada por *256* valores, o circuito necessita de *n = log2(256) = 8* qubits para executar.

Salvamos, então, o resultado de cada execução em um vetor $A = [A_1, A_2,..,A_{500}]$, onde $A_i$ indica o valor de $A(I_i, W)$. Em sequência, uma função de custo, Loss(A,Y) é executada. Neste modelo, escolhemos o método dos mínimos quadrados (MMQ):

$$L(A, Y) = \frac{1}{N}\sum_{i=1}^{N}\left(A_i - Y_i\right)^2$$

Já se sabe que existem funções de perda não lineares mais eficientes. Porém, para nossos objetivos, o MMQ é satisfatório.

Utilizamos a implementação do SPSA disponível na biblioteca *Qiskit* para a otimização, com hiperparâmetros estimados automaticamente pelo otimizador.

Para o cálculo do gradiente utilizado no SPSA, devemos representar A(I,W) numericamente como:

$$Z = I \cdot \bar{W}$$

$$A(I,W) = \frac{|Z|^2}{2^{2n}} = \frac{\left|\sum_{i=1}^{2^n} e^{i\cdot(I[i]-W[i])}\right|^2}{2^{2n}} \Rightarrow$$

$$A(I,W) = \frac{1}{2^{2n}}\cdot\left(\left(\sum_{i=1}^{2^n}\cos((I[i]-W[i]))\right)^2 + \left(\sum_{i=1}^{2^n}\sin((I[i]-W[i]))\right)^2\right) \Rightarrow$$

$$A(I,W) = \frac{1}{2^{2n}}\cdot\left((\cos(I-W)\cdot 1)^2 + (\sin(I-W)\cdot 1)^2\right)$$

Assim, o gradiente dJ / dw pode ser calculado como:

$$\frac{\partial A}{\partial w} = \frac{1}{2^{2n}}\cdot(2\times cos(I-W)\times(sen(I-W)\cdot 1) + 2\times sen(I-W)\times(-\cos(I-W)\cdot 1))$$

$$\frac{\partial A}{\partial w} = \frac{2}{2^{2n}}\cdot(cos(I-W)\times(sen(I-W)\cdot 1) - sen(I-W)\times(cos(I-W)\cdot 1))$$

$$\frac{\partial J}{\partial w} = \frac{1}{M}\sum_{i=1}^{m}\frac{\partial A_i}{\partial w}\left(A_i(I_i,W) - Y_i\right)$$

O operador (x) representa a multiplicação termo-a-termo de dois vetores, enquanto o operador (.) representa o produto interno de dois vetores.

É possível calcular o gradiente utilizando apenas circuitos quânticos, mas para fins de eficiência decidimos calculá-lo apenas numericamente.

### *II.D Testes*

Os testes do modelo são realizados utilizando de outras *M = 200* amostras do banco de dados, suas respectivas classificações e o vetor W otimizado pelo SPSA. A partir de um *threshold* t, definimos a classe predita pelo classificador com:

- 1 (Pertence à classe) se A(I,W) > t
- 0 (Não pertence a classe) se A(I,W) <= t

O *threshold* pode ser escolhido arbitrariamente e pode ser otimizado para melhor eficiência. Nos experimentos com o modelo, decidimos utilizar t = 0.5.

## III EXPLICAÇÃO DA BASE

A base de dados utilizada neste artigo foi a *Open Image Dataset V6* do *Google*. A grande vantagem do uso dessa base no experimento se dá graças à biblioteca FiftyOne, que possibilita integração total com o banco de imagens, sem a necessidade de separar manualmente as classes desejadas e os conjuntos necessários.

Por isso, é possível selecionar com uma flexibilidade considerável quais dados serão processados pelo Perceptron.

Além disso, há uma quantidade gigantesca de dados disponíveis na base, cortesia do *Google Images.*

## IV EXPLICAÇÃO DOS EXPERIMENTOS

Infelizmente, como computadores quânticos reais não são acessíveis e nem estão em estados de fácil manipulação, é necessário simular o resultado de um circuito quântico por em um computador clássico e avaliar os resultados teóricos do experimento com essa limitação.

Por isso, alguns experimentos serão propostos para assegurar a consistência do modelo:

1. Teste da Função de Ativação.
2. Teste de um desempenho teórico com o simulador.
3. Teste de um desempenho teórico numericamente.
4. Comparação com modelos conhecidos.

Os códigos referenciados por todo o arquivo podem ser adquiridos via contato por e-mail.

Cada um dos experimentos será detalhado a seguir.

### *IV.A Teste da Função de Ativação.*

Neste experimento, comparamos o valor da função de ativação calculada com o simulador pelo valor numérico que demonstramos previamente. Assim, podemos garantir que o circuito está funcionando da maneira esperada.

A Figura 2 demonstra o resultado obtido. Calculamos a diferença entre ambas a ativação e, como é possível notar, obtivemos um gráfico praticamente constante em torno de 0, com pequenas variações na ordem de 1e-2.

Esse resultado é extremamente positivo, e indica que todos os cálculos foram realizados da maneira esperada e os circuitos funcionam devidamente.

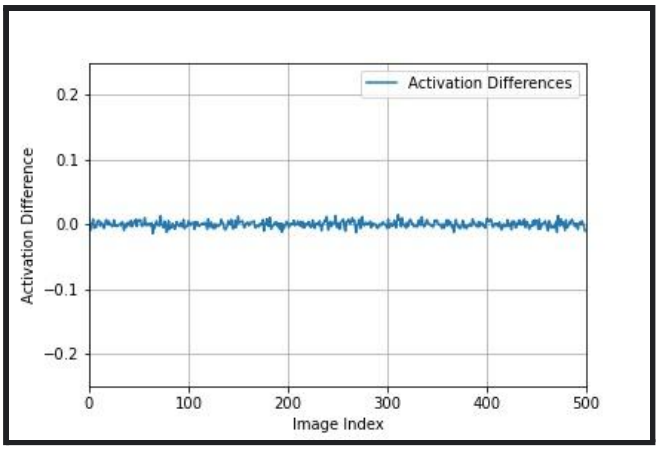


**Figura 2**: Diferença entre a ativação numérica e a ativação quântica para 500 imagens.

### *IV.B Teste de um desempenho teórico com o simulador.*

Neste experimento, iremos executar *200* Iterações do otimizador, com 800 imagens de treino, cada uma *16 x 16*. Normalmente, um número consideravelmente maior em todos os parâmetros deveria ser escolhido para um treinamento razoável do modelo. Porém, a simulação de um circuito quântico de 8 qubits, com aproximadamente *510* portas quânticas (*255* para I, *255* para W) já se mostra muito custosa, na casa de *0.5*s por simulação, em média. Considerando que o otimizador realiza *200* iterações, ao simular o circuito 800 vezes, teremos uma duração média para a finalização do treinamento de *5.5* horas!

Mesmo fazendo uso da GPU para simulação dos circuitos quânticos, e aproveitando do paralelismo suportado pela biblioteca *numpy*, o teste foi custoso: aproximadamente 7 horas foram necessárias para executar a otimização completamente. Em comparação, o teste IV.C, que utiliza parâmetros mais ambiciosos, demorou cerca de *5* minutos.

É necessário pontuar que, em um computador quântico ideal, seria possível paralelizar até mesmo o conjunto I, utilizando-se de *500 * 8 = 2000* qubits em um circuito. Um grande salto de desempenho seria possibilitado, porém no momento presente computadores com essa magnitude de processamento ainda não existem. Por esse e outros motivos, não foi possível convergir a função de custo com o simulador. Portanto, uma precisão de apenas 40% foi atingida, um desempenho baixo porém justificado pelos parâmetros.

Como demonstrado em IV.A, a ativação quântica se comporta de forma extremamente similar à ativação numérica, indicando que o comportamento quântico com os parâmetros do teste IV.4 deverá ser parecido com o comportamento numérico.

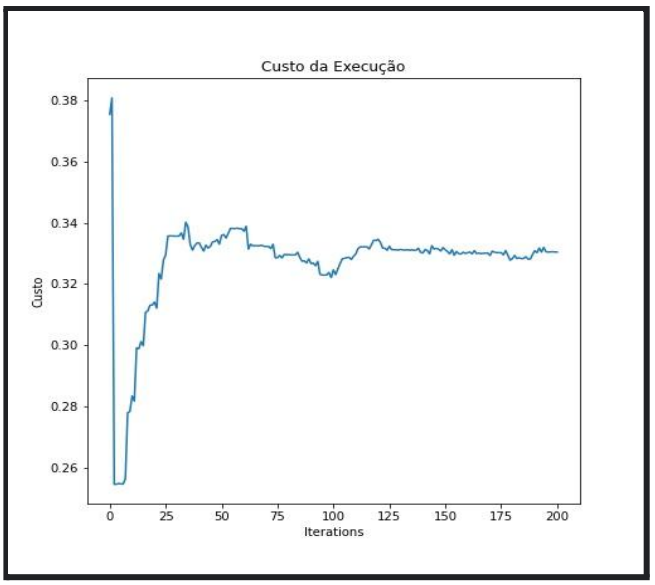


**Figura 3**: Função de Custo da Ativação Quântica.

### *IV.C Teste de um desempenho teórico numericamente.*

Neste experimento, executamos o modelo apresentado pela figura 1, porém o cálculo da função de ativação se deu pela fórmula numérica demonstrada em II.C. Assim, podemos estimar o potencial do modelo quando as limitações de hardware já explicadas deixarem de ser um empecilho.

Para isso, utilizaremos as mesmas 800 imagens, todas 96 x 96, e realizaremos 800 iterações do otimizador.

Como demonstrado na Figura 4, o modelo numérico consegue convergir sua função de custo, e atinge uma precisão de aproximadamente 61% nas imagens de teste. Esse resultado demonstra que é possível atingir uma precisão comparável à de modelos convencionais ao alimentar o modelo com mais dados. Esperamos que futuramente seja possível realizar treinamentos nessa escala em um computador quântico real.

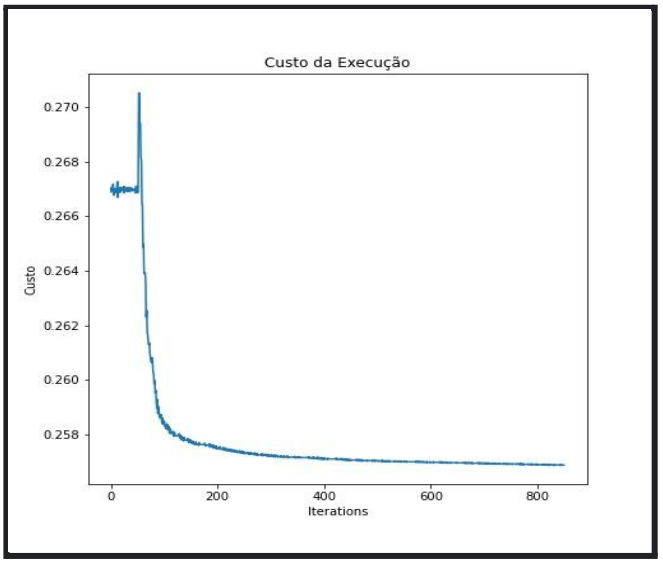


**Figura 4**: Valor da função de custo após execução do otimizador

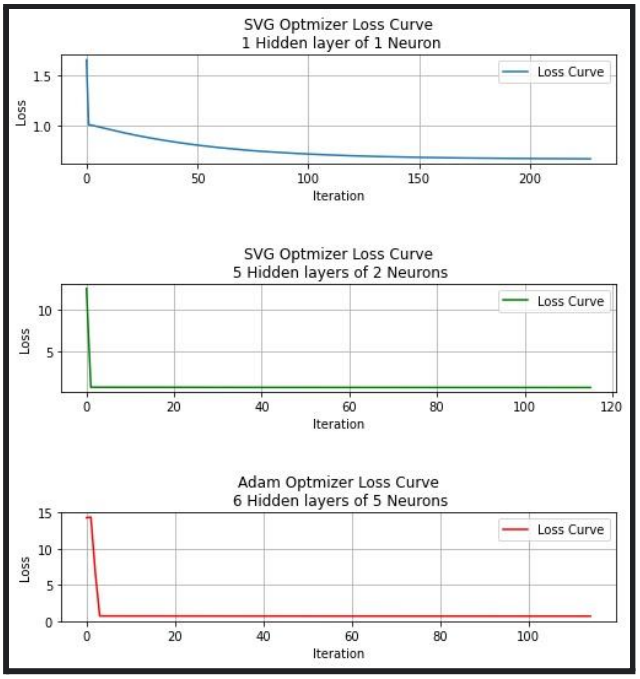


**Figura 5**: Curva de custo para cada um dos modelos

### *IV.D Comparação com modelos conhecidos.*

Neste experimento, comparamos os resultados adquiridos em IV.B e IV.C com modelos já estabelecidos de machine learning.

A base de dados enviada para os 3 modelos foi exatamente a mesma do teste IV.B.

Os resultados dos 3 modelos, implementados utilizando a biblioteca *scikit learn,* demonstram que o desempenho do nosso classificador quântico se aproxima eficientemente do desempenho esperado de uma rede neural convencional.

## V Conclusões

Neste artigo demonstramos que é possível construir um Perceptron Linear utilizando circuitos quânticos para o cálculo da função de ativação.

Demonstramos ainda que a eficiência de tal Perceptron é comparável à eficiência de modelos já estabelecidos no ramo, o que indica um enorme potencial na união da computação quântica com o aprendizado de máquina. Esperamos que em um futuro breve o mesmo modelo possa ser executado em um computador quântico real, e que resultados parecidos com o que nossos experimentos apontam possam ser observados.

## Referências

**[1] Stefano Mangini** et al 2020 Mach. Learn.: Sci. Technol. 1 0450