

Brazil Quantum Camp - Trilha Quantum Tech

Notas de Aulas do Curso de Computação Quântica

Cairo Henrique Vaz Cotrim

Período: Janeiro – Março 2026

Prefácio

Olá!

Atualmente, eu sou um estudante do Bacharelado em Física na USP. O principal objetivo desta obra é manter anotações sobre o que eu aprendi no curso Trilha Quantum Tech - Brazil Quantum Camp, realizado no início de 2026. O público-alvo consiste em: estudantes de exatas que já possuem boa base em Álgebra Linear e buscam um conhecimento introdutório e matematicamente formal das aplicações da Computação Quântica. Acredito que esta seja a melhor definição, visto que era o meu contexto enquanto eu escrevia isto.

O texto é dividido em três grandes blocos: Fundamentos (Bloco 01), Algoritmos e Otimização (Bloco 02) e Machine Learning Clássico e Quântico (Bloco 03). Nesta leitura, você vai encontrar explicações organizadas o máximo possível em linguagem matemática a nível de graduação. Além disso, sempre que consegui, eu acrescentei interpretações mais intuitivas sobre os objetos apresentados. Algumas passagens de cálculos podem parecer "jogadas". Nesses casos, o bom leitor é fortemente incentivado a "abrir as contas", ou seja, realizar tais derivações por si mesmo.

Apesar das notas serem fortemente baseadas no curso, eu escrevo algumas formalizações matemáticas a mais para sustentar melhor as ideias. Alguns podem achar que eu escrevo conteúdos extras demais aqui, podendo fugir do escopo. Por exemplo, eu apresento o Teorema da Representação de Riesz no primeiro módulo, mesmo que o teorema não seja necessário - em muitos casos - para um primeiro contato com Computação Quântica. O motivo disso é simples: eu escrevi as anotações me baseando no que eu estava aprendendo e no que eu já havia aprendido na época do curso. Portanto, fica à cargo de você, caro leitor, decidir quais conteúdos valem a pena ser ignorados com base no seu contexto pessoal.

Todo ser humano possui vieses e, portanto, todo ser humano comete erros. Mesmo prezando pela qualidade destas anotações, é possível que eu tenha deixado alguns detalhes errados sem querer. Caso você encontre algum erro, eu te encorajo fortemente a me contatar por e-mail. Se for o caso, ficarei grato por aprender junto com você.

Desejo a você uma prazerosa leitura.

Cairo Henrique Vaz Cotrim
cairo.h.vaz@usp.br | LinkedIn | GitHub | Lattes

Agradecimentos

Gostaria de agradecer, primeiramente, aos professores e tutores da Trilha Quantum Tech pelo compartilhamento de conhecimento e pela organização do Brazil Quantum Camp. Agradeço aos outros membros da CESAR School e do Instituto Eldorado por tornarem esta jornada de aprendizado possível.

Por fim, agradeço aos colegas de curso pelas discussões que enriqueceram estas notas e à minha família pelo apoio constante durante meus estudos.

Sumário

1	Repositório do Curso	8
2	Bloco 01 – Fundamentos	9
2.1	Aula Inaugural	9
2.1.1	Introdução	9
2.1.2	Calendário	9
2.1.3	Bloco 01 — Fundamentos	9
2.1.4	Bloco 02 — Algoritmos e Otimização	9
2.1.5	Bloco 03 — Machine Learning Clássico e Quântico	10
2.2	Aula 1 - Introdução à Computação Quântica	10
2.2.1	Motivação	10
2.2.2	Potenciais áreas de impacto	10
2.2.3	Sinergia com	10
2.2.4	Quantum Computing Market Map	11
2.2.5	Aplicações	11
2.2.6	Definições	11
2.2.7	O teorema fundamental da notação de Dirac	12
2.2.8	Operadores Quânticos Básicos	12
2.2.9	Circuitos	13
2.3	Aula 2 - Álgebra Linear I	14
2.3.1	Definições	14
2.3.2	Primeiros Postulados da Mecânica Quântica	19
2.3.3	Estados de Bell (emaranhados)	21
2.4	Aula 3 - Introdução à Programação	22
2.5	Aula 4 - Álgebra Linear II	22
2.5.1	Projetores	22
2.5.2	Últimos postulados da Mecânica Quântica	22
2.5.3	Esfera de Bloch	26
2.5.4	Mais portas quânticas em circuitos	27
2.5.5	Circuito da Codificação Superdensa	29
2.5.6	Circuito do Teletransporte Quântico	30

2.6	Aula 5 - Introdução à Programação II	32
2.6.1	Operador de Medida para um Estado Composto	32
2.7	Aula 6 - Álgebra Linear III	32
2.7.1	Filosofia da Mecânica	32
2.7.2	Desigualdade CHSH	33
2.7.3	Violação da Desigualdade CHSH	34
2.8	Aula 7 - Construção de Circuitos Quânticos	35
2.9	Aula 8 - Construção de Circuitos Quânticos II	35
2.9.1	Portas Clássicas	35
2.9.2	Oráculo: Como Implementar Funções em Circuitos Quânticos . . .	37
2.9.3	Deutsch: Problema da Função Constante ou Balanceada	37
2.9.4	Deutsch–Jozsa: Generalização do Problema de Deutsch	38
2.9.5	Bernstein–Vazirani: Problema da Chave Secreta	40
3	Bloco 02 – Algoritmos e Otimização	42
3.1	Aula 1 - Introdução à Teoria da Computação	42
3.1.1	Definições fundamentais	42
3.1.2	Decisão x Otimização	43
3.1.3	Complexidade de algoritmos	43
3.1.4	Principais Comportamentos Assintóticos	44
3.1.5	Classes de complexidade	45
3.2	Aula 2 - Algoritmos Quânticos I	46
3.2.1	Link de uma lista de algoritmos quânticos	46
3.2.2	Algoritmo de Simon	47
3.2.3	Algoritmo de Busca de Grover	50
3.3	Aula 3 - Algoritmos Quânticos II	54
3.3.1	Transformada de Fourier Clássica	54
3.3.2	Transformada Quântica de Fourier	55
3.3.3	Transformada Quântica de Fourier Inversa	58
3.3.4	Estimação de Fase	59
3.3.5	Criptografia RSA	60
3.3.6	Fundamentos do Algoritmo de Shor	62
3.4	Aula 4 - Introdução a Algoritmos Clássicos	63

3.4.1	Heurísticas e Metaheurísticas	63
3.4.2	Alguns problemas NP-Difíceis	63
3.4.3	Algoritmo Genético	65
3.4.4	Algoritmo Genético Inspirado em Quântica (QIGA)	66
3.4.5	QUBO	67
3.4.6	Modelo Ising	68
3.5	Aula 5 - Otimização Quântica	69
3.5.1	Evolução temporal de um qubit físico	69
3.5.2	Evolução adiabática	70
3.5.3	Computação Quântica Adiabática (CQA)	71
3.5.4	Quantum Annealing	73
3.5.5	O Hamiltoniano Ising da DWave	74
3.5.6	Comparação	77
3.6	Aula 6 - Otimização Quântica II	77
3.6.1	Algoritmo de Otimização Aproximada Quântica (QAOA)	77
3.6.2	Enunciado	78
3.6.3	Ideia do QAOA	78
3.6.4	Circuito e evolução	81
3.6.5	Relação com CQA	81
3.7	Aula 7 - Otimização Quântica III	82
3.7.1	Princípio Variacional	82
3.7.2	VQE	84
3.7.3	Ideia do VQE	84
3.7.4	Circuito e Evolução	85
3.7.5	Exemplos Ansatz	86
3.7.6	FALQON	87
3.7.7	Ideia do FALQON	87
3.7.8	Circuito e evolução	89
3.7.9	O parâmetro Δt	90
3.7.10	Qubit-Wise Commuting (QWC)	90
3.8	Aula 8 - Prática de Otimização Quântica	91
3.8.1	Link do notebook do Google Colab	91

4	Bloco 03 – Machine Learning Clássico e Quântico	92
4.1	Aula 1 - Introdução à Aprendizagem de Máquina Clássica	92
4.1.1	Aprendizagem de Máquina	92
4.1.2	k-Nearest Neighbors	93
4.1.3	Árvore de Decisão	94
4.1.4	Random Forest	95
4.1.5	Regressão Linear	96
4.1.6	Otimização no Aprendizado de Máquina	96
4.1.7	Regressão Logística	99
4.1.8	Support Vector Machine	100
4.2	Aula 2 - Introdução a Redes Neurais	102
4.2.1	Perceptron	102
4.2.2	MLP	105
4.2.3	Backpropagation	107
4.3	Aula 3 - Prática com Redes Neurais	108
4.3.1	Link do notebook Google Colab	108
4.3.2	Mais otimizadores	109
4.3.3	Redes Neurais Recorrentes	111
4.3.4	Redes Convolucionais	114
4.4	Aula 4 - Aprendizagem de Máquina Quântica	116
4.4.1	Link do notebook Google Colab	116
4.4.2	O que é?	116
4.4.3	Evolução da QML	116
4.4.4	VQA em QML	117
4.5	Aula 5 - Modelos de Aprendizagem Quântica	118
4.5.1	Métodos gerais	118
4.5.2	Quantum k-Nearest Neighbors	123
4.5.3	Quantum Support Vector Machine	124
4.5.4	Variational Quantum Classifier	126
4.5.5	Quantum k-Means	127
4.6	Aula 6 - Introdução a Redes Neurais Quânticas I	128
4.6.1	Peças chaves para a QNN	128
4.6.2	Arquiteturas	128

4.6.3	Modelo híbrido	129
4.6.4	Quantum Convolutional Neural Network	130
4.7	Aula 7 - Introdução a Redes Neurais Quânticas II	131
4.7.1	Desafios	131
4.8	Aula 8 - Apresentação de Projetos	131
4.8.1	Link do notebook Google Colab	131

Lista de Figuras

1	Esfera de Bloch. Fonte: Wikipedia.	27
2	Circuito do Teletransporte Quântico.	30
3	Circuito de Deutsch.	37
4	Circuito de Deutsch-Jozsa.	39
5	Circuito de Bernstein–Vazirani.	40
6	Circuito de Simon. Fonte: Brazil Quantum Camp.	47
7	Circuito do Algoritmo de Grover. Fonte: Xanadu Quantum.	50
8	Visualização geométrica do Algoritmo de Grover. Fonte: https://aprenda.quantumket.org/algoritmos/busca/09_2_0-grover.html	53
9	Circuito que implementa a QFT. Fonte: https://www.ictp-saifr.org/wp-content/uploads/2022/11/ICTP_SAIFR_D2.pdf	57
10	Circuito que implementa a IQFT. Fonte: https://www.ictp-saifr.org/wp-content/uploads/2022/11/ICTP_SAIFR_D2.pdf	58
11	Circuito que implementa a QPE. Fonte: Brazil Quantum Camp.	59
12	Circuito que implementa a parte quântica do Algoritmo de Shor. Fonte: Wikipedia.	62
13	Circuito que implementa a parte quântica do QAOA. Fonte: https://quantum.cloud.ibm.com/docs/en/tutorials/quantum-approximate-optimization-algor	
14	Exemplo de circuito que implementa a parte quântica do VQE. Fonte: N. Innan, M. A. Z. Khan and M. Bennai, arXiv:2305.07902 (2023).	84
15	Diagrama mostrando a implementação do FALQON. Fonte: A. B. Magann, et al. Phys. Rev. Lett. 129, 250502 (2022).	87
16	Ilustração do fenômeno barren plateaus. Fonte: Wang, S., Fontana, E., Cerezo, M. et al. Noise-induced barren plateaus in variational quantum algorithms. Nat Commun 12, 6961 (2021). https://doi.org/10.1038/s41467-021-27045-6	118
17	Diagrama do modelo híbrido de rede neural quântica. Fonte: Brazil Quantum Camp	129
18	Diagrama da rede neural convolucional quântica. Fonte: https://pennylane.ai/qml/glossary/qc	

1 Repositório do Curso

O material de programação completo do curso está disponível no repositório oficial do Brazil Quantum Camp:

`https://github.com/Brazil-Quantum-Camp/alunos-quantum-tech-01`

Este repositório reúne todos os notebooks, exercícios e projetos desenvolvidos ao longo da trilha Quantum Tech. Dessa forma, o repositório funciona como base prática, enquanto estas notas fornecem a fundamentação teórica correspondente.

2 Bloco 01 – Fundamentos

2.1 Aula Inaugural

2.1.1 Introdução

Brazil Quantum Camp é um projeto de formação e pesquisa em computação quântica realizado pela CESAR School e pelo Instituto Eldorado, em parceria com a Softex e o Ministério da Ciência e Tecnologia (MCTI).

O programa é composto por três trilhas complementares: Quantum Awareness, Quantum Tech e Quantum Immersive, que formam uma jornada completa de aprendizagem.

2.1.2 Calendário

2.1.3 Bloco 01 — Fundamentos

Data	Aula
26/01	Aula Inaugural
27/01	Aula 1 — Introdução à Computação Quântica
28/01	Aula 2 — Álgebra Linear I
29/01	Aula 3 — Introdução à Programação
30/01	Aula 4 — Álgebra Linear II
03/02	Aula 5 — Introdução à Programação II
04/02	Aula 6 — Álgebra Linear III
05/02	Aula 7 — Construção de Circuitos Quânticos
06/02	Aula 8 — Construção de Circuitos Quânticos II

2.1.4 Bloco 02 — Algoritmos e Otimização

Data	Aula
24/02	Aula 9 — Introdução à Teoria da Computação
25/02	Aula 10 — Algoritmos Quânticos I
26/02	Aula 11 — Algoritmos Quânticos II
27/02	Aula 12 — Introdução a Algoritmos Clássicos
03/03	Aula 13 — Otimização Quântica
04/03	Aula 14 — Otimização Quântica II
05/03	Aula 15 — Otimização Quântica III
06/03	Aula 16 — Prática de Otimização Quântica

2.1.5 Bloco 03 — Machine Learning Clássico e Quântico

Data	Aula
17/03	Aula 17 — Introdução à Aprendizagem de Máquina Clássica
18/03	Aula 18 — Introdução a Redes Neurais
19/03	Aula 19 — Prática com Redes Neurais
20/03	Aula 20 — Aprendizagem de Máquina Quântica
24/03	Aula 21 — Modelos de Aprendizagem Quântica
25/03	Aula 22 — Introdução Prática a Redes Neurais Quânticas
26/03	Aula 23 — Introdução Prática a Redes Neurais Quânticas
27/03	Aula 24 — Apresentação de Projetos

2.2 Aula 1 - Introdução à Computação Quântica

2.2.1 Motivação

- Área com muito potencial inovador.
- Previsão de 2 trilhões de investimento até 2035.

2.2.2 Potenciais áreas de impacto

- Fármacos;
- Engenharia de Materiais;
- Química e Biotecnologia;
- Finanças;
- Mobilidade e Logística;

2.2.3 Sinergia com

- IA;
- Robótica;
- Tecnologias de previsão climáticas;
- Cybersegurança;

2.2.4 Quantum Computing Market Map

[<https://thequantuminsider.com/2022/05/09/quantum-computing-market-map-and-data-2022/>]

2.2.5 Aplicações

Otimização Quântica

Resolução de problemas NP; aceleração exponencial de processos.

Quantum Machine Learning

Modelos mais precisos; resistência a ruído; encontrar padrões mais complexos; melhor performance de treinamento.

Simulação

Modelagem mais próxima do mundo real; simulação de moléculas e sistemas biológicos; redução do tempo de produção de fármacos e materiais.

Segurança

Uso de protocolos quânticos para proteger canais de comunicação (Criptografia quântica).

Quantum Centric Computing

Novo modelo de sistema de alta-performance; integração de quântica a modelos atuais de produção.

2.2.6 Definições

Espaços Completos

Um espaço métrico (X, d) é dito **completo** se toda sequência de Cauchy $(x_n) \subset X$ converge para um elemento x de X , isto é, existe $x \in X$ tal que

$$\lim_{n \rightarrow \infty} d(x_n, x) = 0.$$

Espaço de Hilbert

Espaço de Hilbert \mathcal{H} é um espaço vetorial munido da operação produto interno $\langle \cdot, \cdot \rangle$ que também é um espaço métrico completo em relação à métrica $d(u, v) = \|u - v\| = \sqrt{\langle u - v, u - v \rangle}$.

Espaço Dual

O **espaço dual** V^* de um espaço vetorial V sobre K é o conjunto de todos os mapeamentos lineares $f : V \rightarrow K$.

Notação de Dirac (Bra–Ket)

$|\psi\rangle$ (ket) é um elemento de \mathcal{H} .
 $\langle\psi|$ (bra) é um elemento de \mathcal{H}^* .

Qubit

Na Computação Quântica, utilizamos o espaço de Hilbert \mathbb{C}^n sobre \mathbb{C} para representar os sistemas físicos com n estados possíveis. Denotaremos esse espaço simplesmente por \mathcal{H} .

Um **qubit** é um elemento de um espaço de Hilbert \mathbb{C}^2 que representa um sistema físico com 2 estados possíveis.

Todo qubit pode ser dado como um vetor na Esfera de Bloch:

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle, \quad (1)$$

onde $\{|0\rangle, |1\rangle\}$ é uma base de \mathbb{C}^2 .

2.2.7 O teorema fundamental da notação de Dirac**Teorema de Representação de Riesz**

Seja \mathcal{H} um espaço de Hilbert complexo e seja

$$f : \mathcal{H} \rightarrow \mathbb{C}$$

um funcional linear contínuo.

Então existe um único vetor $|\phi\rangle \in \mathcal{H}$ tal que, para todo $|\psi\rangle \in \mathcal{H}$,

$$f(|\psi\rangle) = \langle\phi|\psi\rangle.$$

Ou seja, todo funcional linear contínuo pode ser representado como um produto interno com um vetor do espaço de Hilbert, justificando a identificação entre vetores e bras na notação de Dirac. Mais para frente, veremos como representar um bra na forma vetorial.

2.2.8 Operadores Quânticos Básicos

Operadores quânticos são matrizes unitárias.

Matrizes de Pauli

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (2)$$

Atua como um análogo da porta NOT. Rotação de π em torno do eixo X.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (3)$$

Troca as componentes do qubit e introduz fases opostas $\pm i$. Rotação de π em torno do eixo Y.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4)$$

Inverte o sinal da componente de $|1\rangle$. Rotação de π em torno do eixo Z.

Proposição 1 *As matrizes de Pauli*

$$X, Y, Z,$$

juntamente com a matriz identidade I , formam uma base do espaço vetorial das matrizes complexas 2×2 .

Porta Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (5)$$

Porta CNOT

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (6)$$

2.2.9 Circuitos

O Circuito de Bell, que possui 2 qubits como entrada, é dado por um Hadamard aplicado ao qubit controlador seguido por um CNOT. Representação matricial:

$$U_{Bell} = (CNOT)(H \otimes I) \quad (7)$$

Circuito de Bell Invertido:

$$U_{Bell}^{-1} = (H \otimes I)(CNOT) \quad (8)$$

2.3 Aula 2 - Álgebra Linear I

2.3.1 Definições

Operador Adjunto

Seja \mathbb{C}^n um espaço vetorial complexo. Define-se o **operador adjunto** como a aplicação

$$\dagger : \mathbb{C}^n \rightarrow \mathbb{C}^n$$

tal que, para todo vetor

$$|v\rangle = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix},$$

tem-se

$$\langle v| \equiv |v\rangle^\dagger = \begin{pmatrix} v_1^* & v_2^* & \cdots & v_n^* \end{pmatrix},$$

onde $*$ denota a conjugação complexa e \dagger representa o conjugado transposto.

Propriedades do adjunto:

$$(|u\rangle + |v\rangle)^\dagger = \langle u| + \langle v|.$$

$$(\alpha |v\rangle)^\dagger = \alpha^* \langle v|, \quad \alpha \in \mathbb{C}.$$

$$(\langle v|)^\dagger = |v\rangle.$$

$$(AB)^\dagger = B^\dagger A^\dagger.$$

$$(A^\dagger)^\dagger = A.$$

O Teorema de Representação de Riesz afirma que deve haver apenas um vetor associado ao funcional linear que faz o produto interno funcionar. Como a definição acima cumpre esse papel, o bra deve ser definido como o adjunto do ket.

Produto Interno

Sejam $|u\rangle, |v\rangle \in \mathbb{C}^n$. Define-se o **produto interno** entre $|u\rangle$ e $|v\rangle$ como

$$\langle u | v \rangle = |u\rangle^\dagger \cdot |v\rangle.$$

Explicitamente,

$$\langle u | v \rangle = \sum_{i=1}^n u_i^* v_i.$$

Propriedades:

$$\langle u | \alpha v + \beta w \rangle = \alpha \langle u | v \rangle + \beta \langle u | w \rangle, \quad \alpha, \beta \in \mathbb{C}.$$

$$\langle \alpha u + \beta w | v \rangle = \alpha^* \langle u | v \rangle + \beta^* \langle w | v \rangle, \quad \alpha, \beta \in \mathbb{C}.$$

$$\langle u | v \rangle = \langle v | u \rangle^*.$$

$$\langle v | v \rangle \geq 0.$$

$$\langle v | v \rangle = 0 \Rightarrow |v\rangle = 0.$$

Produto Exterior

Sejam $|u\rangle, |v\rangle \in \mathbb{C}^n$. Define-se o **produto exterior** entre $|u\rangle$ e $|v\rangle$ como o produto matricial:

$$|u\rangle \langle v| = |u\rangle |v\rangle^\dagger.$$

Operador Hermitiano

Um operador A é dito Hermitiano se

$$A^\dagger = A.$$

Propriedades:

- Autovalores reais
- Autovetores ortogonais
- Representam observáveis físicos

Operador Unitário

Um operador U é dito unitário se

$$U^\dagger = U^{-1}.$$

Propriedades:

- O operador unitário U preserva o produto interno.

Prova:

Seja $\langle u|v \rangle = N$: Temos

$$\langle Uu | Uv \rangle = (U|u\rangle)^\dagger (U|v\rangle).$$

Usando a propriedade do adjunto,

$$(U|u\rangle)^\dagger = \langle u| U^\dagger,$$

logo,

$$\langle Uu | Uv \rangle = \langle u| U^\dagger U |v \rangle.$$

Como U é unitário, vale $U^\dagger U = I$, onde I é o operador identidade. Portanto,

$$\langle Uu | Uv \rangle = \langle u| I |v \rangle = \langle u | v \rangle = N.$$

- O produto matricial de dois operadores unitários U_1 e U_2 também é unitário.

Prova:

Seja $U = U_1 U_2$:

Pela propriedade do adjunto do produto,

$$(U_1 U_2)^\dagger = U_2^\dagger U_1^\dagger.$$

Substituindo pelas inversas,

$$U^\dagger = U_2^{-1} U_1^{-1}.$$

Por outro lado, a inversa de U é

$$U^{-1} = (U_1 U_2)^{-1} = U_2^{-1} U_1^{-1}.$$

Logo,

$$U^\dagger = U^{-1}.$$

Produto Tensorial de Vetores

Define-se o **produto tensorial** entre $|u\rangle \in \mathbb{C}^m$ e $|v\rangle \in \mathbb{C}^n$ como o vetor

$$|u\rangle \otimes |v\rangle \in \mathbb{C}^{mn},$$

dado explicitamente por

$$|u\rangle \otimes |v\rangle = \begin{pmatrix} u_1 v_1 \\ u_1 v_2 \\ \vdots \\ u_2 v_1 \\ u_2 v_2 \\ \vdots \\ u_m v_n \end{pmatrix}.$$

Propriedades:

$$(|u\rangle_1 + |u\rangle_2) \otimes |v\rangle = |u\rangle_1 \otimes |v\rangle + |u\rangle_2 \otimes |v\rangle.$$

$$|u\rangle \otimes (|v\rangle_1 + |v\rangle_2) = |u\rangle \otimes |v\rangle_1 + |u\rangle \otimes |v\rangle_2.$$

$$(\alpha |u\rangle) \otimes |v\rangle = \alpha(|u\rangle \otimes |v\rangle), \quad |u\rangle \otimes (\beta |v\rangle) = \beta(|u\rangle \otimes |v\rangle), \quad \alpha, \beta \in \mathbb{C}.$$

$$(|u\rangle \otimes |v\rangle)^\dagger = \langle u| \otimes \langle v|.$$

$$\langle u_1 \otimes v_1 | u_2 \otimes v_2 \rangle = \langle u_1 | u_2 \rangle \langle v_1 | v_2 \rangle.$$

No contexto da Computação Quântica, o produto tensorial apenas busca listar as probabilidades de todos os eventos possíveis do sistema composto em um vetor - da mesma forma que uma simples árvore de decisão faz.

Produto Tensorial de Operadores

Define-se o **produto tensorial** entre $A \in \mathcal{L}(\mathbb{C}^m)$ e $B \in \mathcal{L}(\mathbb{C}^n)$ como o operador

$$A \otimes B \in \mathcal{L}(\mathbb{C}^{mn}),$$

definido, na base canônica, pela matriz em blocos

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mm}B \end{pmatrix},$$

onde $A = (a_{ij}) \in \mathbb{C}^{m \times m}$.

Propriedades:

$$(A_1 + A_2) \otimes B = A_1 \otimes B + A_2 \otimes B.$$

$$A \otimes (B_1 + B_2) = A \otimes B_1 + A \otimes B_2.$$

$$(\alpha A) \otimes B = \alpha(A \otimes B), \quad A \otimes (\beta B) = \beta(A \otimes B), \quad \alpha, \beta \in \mathbb{C}.$$

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD),$$

sempre que os produtos AC e BD estejam bem definidos.

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

$$I_m \otimes I_n = I_{mn}.$$

Para aplicar N operadores em N qubits, basta aplicar o produto tensorial dos operadores sobre o produto tensorial dos qubits. A definição de produto tensorial em operadores busca formalizar isso em um só operador que aceita o sistema composto como entrada.

Proposição 2 (Base dos operadores em um sistema composto) *Sejam I, X, Y, Z as matrizes de Pauli juntamente com a matriz identidade. Então o conjunto de operadores*

$$\mathcal{P}_n = \{P_1 \otimes P_2 \otimes \cdots \otimes P_n \mid P_i \in \{I, X, Y, Z\}\}$$

forma uma base do espaço vetorial das matrizes complexas $2^n \times 2^n$, isto é, do espaço de todos os operadores lineares atuando em um sistema de n qubits.

Exponencial matricial

Seja $A \in M_n(\mathbb{C})$ uma matriz quadrada. A **exponencial matricial** de A é definida pela série de potências

$$e^A := \exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!},$$

onde $A^0 = I$ (matriz identidade).

Explicitamente,

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots.$$

Propriedades

- Se H é uma matriz Hermitiana ($H^\dagger = H$), então

$$U(t) = e^{-iHt}$$

é uma matriz unitária.

Prova:

$$U(t)^\dagger U(t) = e^{iHt} e^{-iHt} = I.$$

- Se A e B são matrizes que comutam, ou seja, $AB = BA$, então vale

$$e^{A+B} = e^A e^B = e^B e^A.$$

2.3.2 Primeiros Postulados da Mecânica Quântica**Postulado I — Espaço de Estados**

O estado físico de um sistema quântico isolado é completamente descrito por um vetor unitário $|\psi\rangle$ pertencente a um espaço de Hilbert complexo $\mathcal{H} \simeq \mathbb{C}^n$, isto é,

$$|\psi\rangle \in \mathbb{C}^n, \quad \langle\psi | \psi\rangle = 1.$$

Sabe-se, por meio de vários experimentos, que a natureza quântica é probabilística. Por conta disso, não é possível representar sistemas físicos por meio das grandezas que medimos, pois estas variam aleatoriamente.

Porém, os experimentos também mostram que a distribuição de probabilidades de um sistema isolado é determinística. Se aproveitando desse fato, o Postulado I busca representar os estados por meio de vetores contendo amplitudes relacionadas às probabilidades de medição. Essa relação exata será melhor explicada pelo Postulado V.

Postulado II — Evolução Temporal

A evolução temporal de um sistema quântico isolado é descrita por um operador unitário

$$U(t) \in \mathcal{L}(\mathcal{H}), \quad U^\dagger(t)U(t) = I,$$

de tal forma que o estado no instante t é dado por

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle.$$

Alternativamente, também pode-se escrever a evolução temporal de um estado a partir da Equação de Schrödinger:

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H(t) |\psi(t)\rangle,$$

onde H é o operador Hamiltoniano hermitiano do sistema.

Proposição 3 (Solução da Equação de Schrödinger) *No caso particular de um H independente do tempo, é possível encontrar a solução da Equação de Schrödinger como:*

$$|\psi(t)\rangle = e^{-\frac{iHt}{\hbar}} |\psi(0)\rangle.$$

Assim, essa solução equivale a usar um operador unitário

$$U(t) = e^{-\frac{i}{\hbar}Ht}.$$

Tendo em mente que os sistemas físicos são representados por vetores, a escolha mais natural de representar as suas mudanças é por meio de matrizes (operadores).

A necessidade desses operadores de serem unitários é oriunda da condição de normalização unitária dos estados (Postulado I). Os operadores unitários não alteram o produto interno, garantindo que o Postulado I seja verdade em todo instante de tempo.

A condição dos estados serem unitários, por sua vez, é necessária para o Postulado V funcionar. Afinal, a definição axiomática de probabilidade define que a probabilidade de todos os resultados possíveis de um sistema deve ser igual a 1.

Postulado III — Sistemas Compostos

O espaço de estados de um sistema quântico composto por dois subsistemas A e B , descritos respectivamente pelos espaços de Hilbert \mathcal{H}_A e \mathcal{H}_B , é dado pelo produto tensorial

$$\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B.$$

Se os subsistemas estão nos estados $|\psi_A\rangle \in \mathcal{H}_A$ e $|\psi_B\rangle \in \mathcal{H}_B$, então o estado conjunto é

$$|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle.$$

Observáveis que atuam localmente sobre os subsistemas são representados por operadores do tipo

$$A \otimes I_B \quad \text{e} \quad I_A \otimes B,$$

onde $A \in \mathcal{L}(\mathcal{H}_A)$, $B \in \mathcal{L}(\mathcal{H}_B)$ e I_A, I_B são os operadores identidade.

Existem estados em \mathcal{H}_{AB} que não podem ser escritos como um produto tensorial de estados dos subsistemas; tais estados são denominados **emaranhados**.

2.3.3 Estados de Bell (emaranhados)

Os **Estados de Bell** (ou pares EPR) são os quatro estados quânticos específicos que representam o nível máximo de emaranhamento entre dois qubits. Eles constituem uma base ortonormal para o espaço de Hilbert

$$\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4.$$

Os quatro Estados de Bell são dados por:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}.$$

Dizemos que um sistema quântico está em um **Estado de Bell** quando seu estado não pode ser escrito como um produto tensorial de estados individuais, isto é,

$$|\psi\rangle \neq |\phi_1\rangle \otimes |\phi_2\rangle.$$

2.4 Aula 3 - Introdução à Programação

2.5 Aula 4 - Álgebra Linear II

2.5.1 Projetores

Projektor

Seja \mathcal{H} um espaço de Hilbert. Um operador linear $P : \mathcal{H} \rightarrow \mathcal{H}$ é chamado de **projektor** se satisfaz

$$P^2 = P.$$

Se, além disso,

$$P^\dagger = P,$$

dizemos que P é um **projektor ortogonal**.

2.5.2 Últimos postulados da Mecânica Quântica

ATENÇÃO: os postulados a seguir se referem apenas aos casos não-degenerados e discretos.

Postulado IV — Observáveis e Medidas

A toda grandeza física observável associa-se um operador hermitiano $A \in \mathcal{L}(\mathcal{H})$, isto é,

$$A^\dagger = A.$$

Os possíveis resultados de uma medida de A são seus autovalores reais $\{a_k\}$, definidos por

$$A |a_k\rangle = a_k |a_k\rangle,$$

onde $|a_k\rangle$ é o autovetor correspondente ao autovalor a_k

É fundamental entender que medições na Física necessariamente envolvem perturbações no estado de um sistema. É impossível realizar um experimento sem modificar de alguma forma o fenômeno estudado - mesmo que, muitas vezes, essa diferença seja desprezível. Por conta disso, também denotamos medições por operadores na Mecânica Quântica.

Proposição 4 (Base dos autoestados) *Todo operador hermitiano é diagonalizável e possui todos os seus autovalores a_1, a_2, \dots reais. Além disso, os seus autoestados $|a_1\rangle, |a_2\rangle, \dots$ (autovetores) formam uma base ortonormal, ou seja,*

$$\langle a_n | a_m \rangle = \delta_{nm}, \tag{9}$$

onde $\delta(m, n)$ é a função delta de Kronecker.

Proposição 5 (Estado escrito na base dos autoestados) *Como $\{|a_1\rangle, |a_2\rangle, \dots\}$ é uma base ortonormal de \mathcal{H} , pode-se escrever qualquer $|\psi\rangle \in \mathcal{H}$ como a combinação linear:*

$$|\psi\rangle = \sum_k z_k |a_k\rangle, \quad (10)$$

onde $z_k = \langle a_k | \psi \rangle$.

Note que z_k é apenas o valor da componente do vetor projetada no autovetor, pois o autovetor é unitário!

Proposição 6 (Decomposição espectral de operadores hermitianos) *Seja \hat{A} um operador hermitiano em um espaço de Hilbert \mathcal{H} , com espectro discreto. Seus autovalores $a_k \in \mathbb{R}$ e seus autovetores $\{|a_k\rangle\}$ satisfazem*

$$\hat{A}|a_k\rangle = a_k|a_k\rangle, \quad \langle a_n | a_m \rangle = \delta_{nm},$$

e formam uma base ortonormal completa de \mathcal{H} . Então,

$$\hat{A} = \sum_k a_k |a_k\rangle \langle a_k|.$$

Prova

Seja $|\psi\rangle \in \mathcal{H}$ um vetor arbitrário na base ortonormal dos autovetores do operador \hat{A} :

$$|\psi\rangle = \sum_k |a_k\rangle \langle a_k | \psi \rangle.$$

Aplicando o operador \hat{A} a ambos os lados,

$$\hat{A}|\psi\rangle = \sum_k \hat{A}|a_k\rangle \langle a_k | \psi \rangle.$$

Como $|a_k\rangle$ é autovetor de \hat{A} , segue que

$$\hat{A}|a_k\rangle = a_k|a_k\rangle,$$

logo

$$\hat{A}|\psi\rangle = \sum_k a_k |a_k\rangle \langle a_k | \psi \rangle.$$

Definindo o operador

$$\hat{B} \equiv \sum_k a_k |a_k\rangle \langle a_k|,$$

observamos que

$$\hat{A}|\psi\rangle = \hat{B}|\psi\rangle, \quad \forall |\psi\rangle \in \mathcal{H}.$$

Portanto, $\hat{A} = \hat{B}$, isto é,

$$\hat{A} = \sum_k a_k |a_k\rangle \langle a_k|. \quad \square$$

Proposição 7 (Projetor ortogonal de posto um) *Seja $|a_k\rangle$ um vetor normalizado de um espaço de Hilbert \mathcal{H} . Então o operador*

$$M_k := |a_k\rangle \langle a_k|$$

é um projetor ortogonal sobre o subespaço gerado por $|a_k\rangle$.

Note que o nome "projetor" é bem feliz, porque aplicar M_k em um vetor $|\psi\rangle$ resulta em um vetor na direção de $|a_k\rangle$ com módulo $|\langle a_k|\psi\rangle| \cdot |||a_k\rangle||$. Considerando que $|a_k\rangle$ é unitário, esta é a projeção de ψ sobre $|a_k\rangle$.

Se aplicarmos uma projeção normalizada $(\frac{M_k}{||M_k|\psi\rangle||})$ em $|\psi\rangle$, obtemos $|a_k\rangle$ com uma fase global diferente (será explicitado no Postulado VI).

Probabilidade

Seja Ω um conjunto não vazio (espaço amostral) e seja \mathcal{F} uma σ -álgebra de subconjuntos de Ω . Uma **medida de probabilidade** é uma função

$$\mathbb{P} : \mathcal{F} \longrightarrow [0, 1]$$

que satisfaz os seguintes axiomas:

1. Não-negatividade:

$$\mathbb{P}(A) \geq 0, \quad \forall A \in \mathcal{F}.$$

2. Normalização:

$$\mathbb{P}(\Omega) = 1.$$

3. σ -aditividade: Para toda família enumerável $\{A_n\}_{n \in \mathbb{N}} \subset \mathcal{F}$ de conjuntos dois a dois disjuntos,

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mathbb{P}(A_n).$$

A tripla ordenada $(\Omega, \mathcal{F}, \mathbb{P})$ é denominada **espaço de probabilidade**.

Postulado V — Probabilidade de Medição (de Born)

Se o sistema está no estado normalizado $|\psi\rangle$, a probabilidade de obter o resultado a_k (autovalor do operador observável) após uma medição é dada por

$$P(a_k) = |\langle a_k | \psi \rangle|^2.$$

Se o vetor $|\psi\rangle$ está escrito na base dos autoestados do operador de interesse, então basta calcular os quadrados dos módulos das suas coordenadas. Nesse caso, $P(a_k) = |z_k|^2$.

O Postulado V justifica o fato de observáveis serem operadores hermitianos. A principal propriedade de interesse é que esses operadores sempre possuem autovalores pertencentes a \mathbb{R} . Grandezas físicas são números reais, então é útil concentrar os esforços em operadores que permitem apenas números reais como autovalores.

Valor Esperado

Seja X uma variável aleatória discreto que assume valores $\{x_i\}$ com probabilidades $\mathbb{P}(X = x_i)$

O **valor esperado** de X é definido por

$$\mathbb{E}[X] := \sum_{i \in I} x_i \mathbb{P}(X = x_i).$$

Proposição 8 *Seja A um observável (operador auto-adjunto) com espectro discreto e seja $|\psi\rangle$ um estado normalizado. O valor esperado de A no estado $|\psi\rangle$ pode ser dado por*

$$\langle A \rangle_\psi := \sum_i a_i p(a_i) = \langle \psi | A | \psi \rangle,$$

e coincide com o valor esperado probabilístico dos resultados da medida de A .

Prova

Como A é auto-adjunto e possui espectro discreto, admite decomposição espectral

$$A = \sum_i a_i |a_i\rangle \langle a_i|,$$

onde $\{a_i\}$ são os autovalores de A e $\{|a_i\rangle\}$ formam uma base ortonormal do espaço de Hilbert.

Substituindo essa decomposição na definição de valor esperado, obtemos

$$\langle \psi | A | \psi \rangle = \langle \psi | \left(\sum_i a_i |a_i\rangle \langle a_i| \right) | \psi \rangle = \sum_i a_i \langle \psi | a_i \rangle \langle a_i | \psi \rangle.$$

Observando que

$$\langle \psi | a_i \rangle \langle a_i | \psi \rangle = |\langle a_i | \psi \rangle|^2,$$

segue que

$$\langle A \rangle_\psi = \sum_i a_i |\langle a_i | \psi \rangle|^2.$$

Pelo postulado de Born, a quantidade

$$p_i := |\langle a_i | \psi \rangle|^2$$

representa a probabilidade de obtenção do resultado a_i na medida do observável A , satisfazendo $p_i \geq 0$ e $\sum_i p_i = 1$.

Portanto, a expressão $\langle \psi | A | \psi \rangle$ coincide com o valor esperado probabilístico dos resultados da medida de A , concluindo a demonstração. \square

Postulado VI — Colapso

Se a medição do observável A sobre um sistema físico descrito pelo vetor $|\psi\rangle$ fornece o resultado a_k , o estado do sistema imediatamente após a medição é a projeção normalizada

$$|\psi'\rangle = \frac{M_k}{\|M_k|\psi\rangle\|} |\psi\rangle = e^{i\theta_k} |a_k\rangle. \quad (11)$$

Como o novo estado $|\psi'\rangle = e^{i\theta_k} |a_k\rangle$ do sistema é empiricamente idêntico ao estado $|a_k\rangle$, as medições do sistema após essa primeira medição sempre resultarão em a_k , a não ser que um novo operador unitário modifique o estado $|\psi'\rangle$.

Esse fato experimental ilustra o motivo de se utilizar autovetores nos postulados da Mecânica Quântica. Sabe-se que aplicar um operador hermitiano num autoestado resulta em um novo estado que não é alterado por esse mesmo operador (ignorando a mudança de fase). A definição de autovetor é a escolha mais natural para esse novo estado, pois o fato de ser um autovetor implica na invariância do mesmo (ignorando a mudança de fase).

2.5.3 Esfera de Bloch

Qualquer estado de um qubit pode ser representado graficamente como um vetor unitário na superfície de uma 2-esfera unitária. As coordenadas reais (x, y, z) são definidas utilizando as matrizes de Pauli:

$$x = \langle X \rangle = \langle \psi | X | \psi \rangle$$

$$y = \langle Y \rangle = \langle \psi | Y | \psi \rangle$$

$$z = \langle Z \rangle = \langle \psi | Z | \psi \rangle$$

Chamamos essa esfera de Esfera de Bloch.

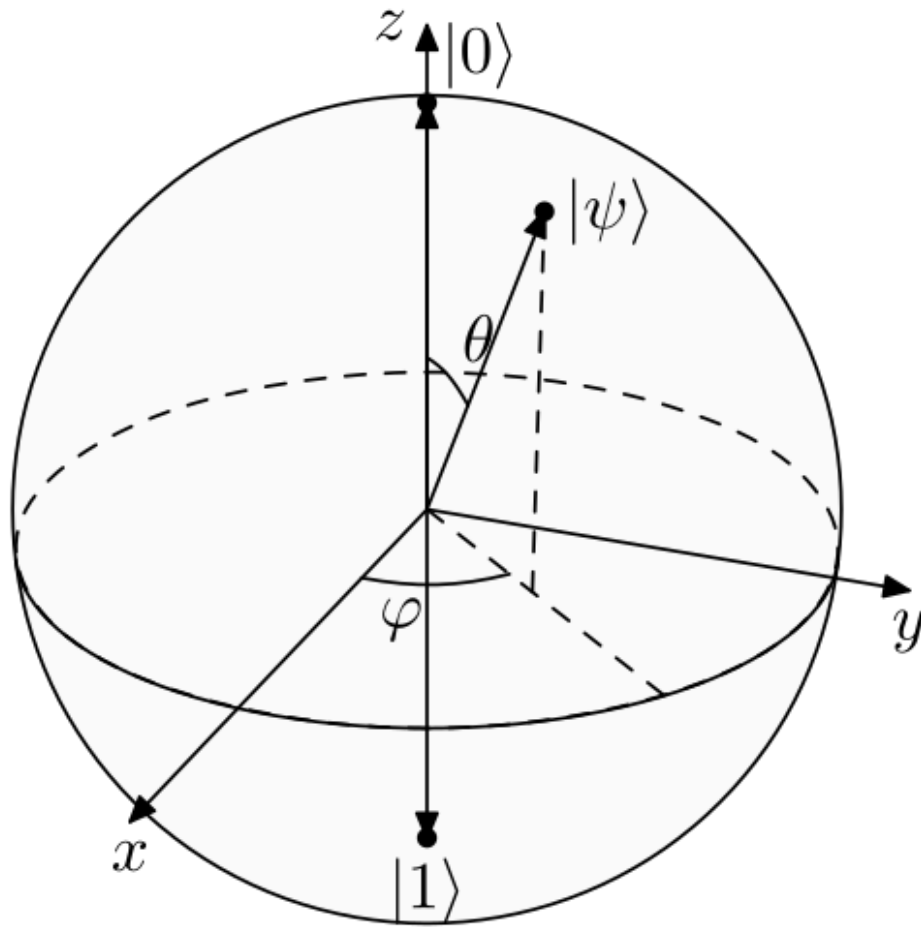


Figura 1: Esfera de Bloch. Fonte: Wikipedia.

2.5.4 Mais portas quânticas em circuitos

Porta de Fase P

A porta P é definida por

$$P := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}.$$

Ela apenas adiciona uma fase relativa θ à componente de $|1\rangle$. Não altera a probabilidade, mas muda o comportamento do qubit frente a portas.

Porta S

A porta S é definida por

$$S := \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

Ela corresponde a uma rotação de fase de $\frac{\pi}{2}$ em torno do eixo z da esfera de Bloch.

Porta T

A porta T é definida por

$$T := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

Ela corresponde a uma rotação de fase de $\frac{\pi}{4}$ em torno do eixo z da esfera de Bloch.

Porta de Rotação R_x

A porta de rotação em torno do eixo x é definida por

$$R_x(\theta) := e^{-i\theta\sigma_x/2} = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix},$$

onde σ_x é a matriz de Pauli- x .

Porta de Rotação R_y

A porta de rotação em torno do eixo y é definida por

$$R_y(\theta) := e^{-i\theta\sigma_y/2} = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix},$$

onde σ_y é a matriz de Pauli- y .

Porta de Rotação R_z

A porta de rotação em torno do eixo z é definida por

$$R_z(\theta) := e^{-i\theta\sigma_z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix},$$

onde σ_z é a matriz de Pauli- z .

Porta Multi-Controlled Z MCZ

A MCZ é uma porta que aplica uma fase -1 apenas à componente em que todos os qubits de controle estão no estado $|1\rangle$.

A matriz MCZ é diagonal, com todos os autovalores iguais a 1, exceto o elemento correspondente ao estado $|11 \cdots 1\rangle$, que é igual a -1 :

$$MCZ = \text{diag}(1, 1, \dots, 1, -1).$$

Porta Multi-Controlled X *MCX*

A MCX é uma porta que atua em n qubits e aplica X no alvo se e somente se todos os qubits de controle estão no estado $|1\rangle$.

Usando o fato de que

$$X = HZH,$$

podemos facilmente criar uma MCX a partir de uma MCZ.

Basta aplicar Hadamard, Z condicional e H no alvo de posição k . A MCX é então dada pelo produto matricial

$$MCX = H_k(MCZ)H_k,$$

onde

$$H_k = I^{\otimes k} \otimes H \otimes I^{\otimes (n-k-1)}.$$

2.5.5 Circuito da Codificação Superdensa

Alice deseja enviar dois bits clássicos de informação para Bob por meio do envio de um único qubit.

Utiliza-se o circuito de Bell para a preparação de um par de qubits emaranhados, previamente compartilhado entre Alice e Bob.

Diferentemente do teletransporte quântico, na codificação superdensa Alice não realiza medições intermediárias. Em vez disso, ela codifica dois bits clássicos aplicando operações unitárias locais sobre o seu qubit do par emaranhado, e então o envia fisicamente a Bob, que realiza uma medição conjunta no estado de Bell.

Alice e Bob compartilham inicialmente um par de qubits no estado de Bell

$$|\Phi^+\rangle_{AB} = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle), \quad (12)$$

onde o qubit A pertence a Alice e o qubit B pertence a Bob.

Alice deseja transmitir dois bits clássicos $(b_1, b_2) \in \{0, 1\}^2$. Para isso, ela aplica ao seu qubit A uma das quatro operações unitárias

$$\{I, X, Z, XZ\},$$

de acordo com a tabela de codificação:

(b_1, b_2)	Operação em A	Estado resultante
00	I	$ \Phi^+\rangle$
01	X	$ \Psi^+\rangle$
10	Z	$ \Phi^-\rangle$
11	XZ	$ \Psi^-\rangle$

Explicitamente, após a aplicação da operação unitária por Alice, o estado do sistema

torna-se

$$\begin{aligned}
 I_A|\Phi^+\rangle_{AB} &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\Phi^+\rangle, \\
 X_A|\Phi^+\rangle_{AB} &= \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) = |\Psi^+\rangle, \\
 Z_A|\Phi^+\rangle_{AB} &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\Phi^-\rangle, \\
 XZ_A|\Phi^+\rangle_{AB} &= \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) = |\Psi^-\rangle.
 \end{aligned} \tag{13}$$

Após a codificação, Alice envia fisicamente o qubit A para Bob. Bob passa então a possuir ambos os qubits e aplica o circuito inversor de Bell, constituído por uma porta CNOT seguida de uma porta Hadamard, de modo a transformar os estados de Bell na base computacional.

A ação desse circuito é dada por

$$\begin{aligned}
 |\Phi^+\rangle &\mapsto |00\rangle, \\
 |\Psi^+\rangle &\mapsto |01\rangle, \\
 |\Phi^-\rangle &\mapsto |10\rangle, \\
 |\Psi^-\rangle &\mapsto |11\rangle.
 \end{aligned}$$

Por fim, Bob mede ambos os qubits na base computacional e recupera exatamente os dois bits clássicos (b_1, b_2) enviados por Alice.

Observa-se que, graças ao emaranhamento previamente compartilhado, é possível transmitir dois bits clássicos de informação por meio do envio de apenas um qubit, sem violar os limites impostos pela causalidade, uma vez que o protocolo requer a distribuição prévia do par emaranhado.

2.5.6 Circuito do Teletransporte Quântico

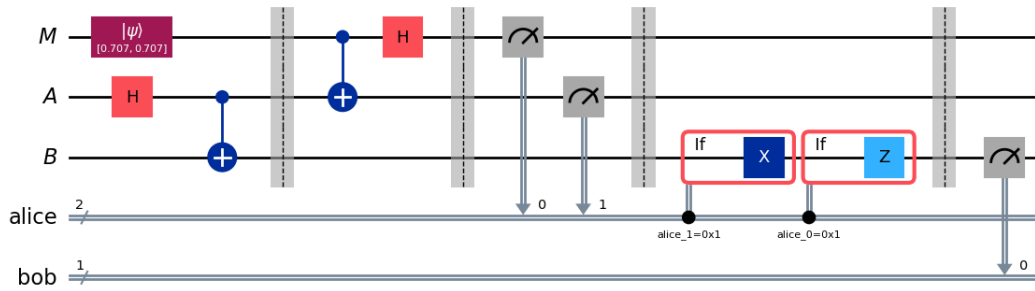


Figura 2: Circuito do Teletransporte Quântico.

Alice deseja enviar um qubit para Bob.

Utiliza-se o circuito de Bell para a preparação de um par de qubits emaranhados, compartilhado entre Alice e Bob;

Aplica-se ao sistema o circuito inversor de Bell, constituído pelas operações CNOT e Hadamard, de modo a realizar uma transformação conjunta entre o qubit a ser teletransportado e o qubit de Alice;

Por fim, após a realização das operações quânticas e da comunicação clássica necessária, Bob encontra-se em posse de um qubit cujo estado quântico é idêntico ao estado originalmente preparado.

Considere o estado arbitrário a ser teletransportado, preparado por Alice,

$$|\psi\rangle_A = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1. \quad (14)$$

Alice e Bob compartilham previamente um par de qubits emaranhados no estado de Bell

$$|\Phi^+\rangle_{BC} = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle). \quad (15)$$

O estado inicial do sistema composto pelos três qubits é, portanto,

$$|\Psi_0\rangle = |\psi\rangle_A \otimes |\Phi^+\rangle_{BC} = \frac{1}{\sqrt{2}} (\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle). \quad (16)$$

Aplica-se, em seguida, uma porta CNOT entre os qubits A (controle) e B (alvo), seguida de uma porta Hadamard no qubit A . Após essas operações, o estado do sistema pode ser reescrito como

$$|\Psi_1\rangle = \frac{1}{2} \left[|00\rangle_{AB}(\alpha|0\rangle + \beta|1\rangle)_C + |01\rangle_{AB}(\alpha|1\rangle + \beta|0\rangle)_C + |10\rangle_{AB}(\alpha|0\rangle - \beta|1\rangle)_C + |11\rangle_{AB}(\alpha|1\rangle - \beta|0\rangle)_C \right]. \quad (17)$$

Após a medição dos qubits de Alice nos estados da base computacional, o estado do qubit de Bob colapsa condicionalmente conforme o resultado obtido:

Resultado em AB	Estado de Bob (C)	Porta que Bob deve aplicar
$ 00\rangle_{AB}$	$\alpha 0\rangle + \beta 1\rangle$	I
$ 01\rangle_{AB}$	$\alpha 1\rangle + \beta 0\rangle$	X
$ 10\rangle_{AB}$	$\alpha 0\rangle - \beta 1\rangle$	Z
$ 11\rangle_{AB}$	$\alpha 1\rangle - \beta 0\rangle$	ZX

Observa-se que, após a medição dos qubits de Alice (A e B), o qubit de Bob (C) colapsa em um estado que difere de $|\psi\rangle_A$ apenas por uma transformação unitária. A partir do resultado clássico da medição, Bob pode aplicar a operação apropriada (I , X , Z ou XZ) e recuperar exatamente o estado original

$$|\psi\rangle_C = \alpha|0\rangle + \beta|1\rangle = |\psi\rangle_A. \quad (18)$$

2.6 Aula 5 - Introdução à Programação II

2.6.1 Operador de Medida para um Estado Composto

Operador de Medida

Para representar um estado de n qubits após uma medição de um qubit de posição k , é necessário aplicar o projetor (correspondente ao bit medido) no qubit k , aplicar identidade no resto e normalizar o estado. Este raciocínio é a junção do Postulado III e VI.

Se o bit medido foi 0, aplica-se

$$M_{0,k} = \frac{I^{\otimes k} \otimes |0\rangle \langle 0| \otimes I^{\otimes (n-k-1)}}{\sqrt{p(|0\rangle)}}.$$

Se o bit medido foi 1, aplica-se

$$M_{1,k} = \frac{I^{\otimes k} \otimes |1\rangle \langle 1| \otimes I^{\otimes (n-k-1)}}{\sqrt{p(|1\rangle)}}.$$

Aqui, denotamos $p(|a\rangle)$ como a probabilidade de obter o bit a após medir o qubit k :

$$p(|a\rangle) = I^{\otimes k} \otimes |a\rangle \langle a| \otimes I^{\otimes (n-k-1)}$$

2.7 Aula 6 - Álgebra Linear III

2.7.1 Filosofia da Mecânica

Na Mecânica Clássica,

- Sistemas possuem propriedades bem definidas, mesmo antes de serem medidas;
- A medida apenas revela essas propriedades para o medidor.

Em outras palavras, as propriedades de um sistema existem e são condicionadas a variáveis dele mesmo se não houver nenhuma interação desse sistema com outro (no caso, a interação é a medição).

Por conta da semelhança dessas ideias com a vertente filosófica realista, chamaremos isso de "realismo". Porém, é necessário tomar cuidado: na Filosofia, o realismo usualmente se refere à ideia de uma realidade independente de um ser epistêmico, enquanto a Física abrange essa independência a qualquer processo físico que atue como uma "medição".

O hipótese do realismo pode ser mais formalizado matematicamente no contexto quântico a partir da definição de variáveis ocultas, que condicionam totalmente o resultado da medição antes mesmo dela ocorrer.

Variáveis Ocultas (Realismo)

Seja um sistema físico isolado denotado por um estado $|\psi\rangle$ que foi medido e entregou um resultado $a \in \mathbb{R}$.

Uma teoria da Física é dita **realista** se, para qualquer medição, existe uma **variável oculta** λ e uma função f tal que

$$a = f(|\psi\rangle, \lambda).$$

Dessa forma, a aleatoriedade quântica seria apenas ignorância dos físicos sobre uma teoria mais explicativa.

Em uma tentativa de generalizar a Teoria da Relatividade, também podemos pensar na hipótese da localidade, que afirma que uma influência não pode viajar mais rápido do que a velocidade da luz.

Localidade

Sejam dois sistemas espacialmente separados A e B . Uma teoria da Física segue a **localidade** se uma medição em A não influencia uma medição imediatamente depois em B .

Variáveis Ocultas Locais (Realismo Local)

Uma teoria da Física segue o **Realismo Local** se implementa o realismo e a localidade. As variáveis ocultas são então chamadas de **variáveis ocultas locais**.

Na Mecânica Quântica, o emaranhamento traz algumas dúvidas sobre a validade do realismo local. A próxima seção apresenta uma tentativa de provar que ela não implementa.

2.7.2 Desigualdade CHSH

Considere dois sistemas físicos espacialmente separados, A (Alice) e B (Bob), preparados em um estado possivelmente emaranhado. Cada observador pode escolher entre duas possíveis medições:

- Alice escolhe entre observáveis A_0 ou A_1 ;
- Bob escolhe entre observáveis B_0 ou B_1 .

Cada medição produz resultados dicotômicos, isto é,

$$A_x, B_y \in \{-1, +1\}, \quad x, y \in \{0, 1\}.$$

Isso equivale a dizer que os operadores possuem autovalores ± 1 .

Sob a hipótese do **realismo local**, os resultados das medições são completamente determinados por uma variável oculta local λ , de modo que

$$A_x = A_x(\lambda), \quad B_y = B_y(\lambda),$$

e a distribuição de λ é dada por uma densidade de probabilidade $\rho(\lambda)$.

Define-se então a esperança dos resultados das medições A_x e B_y como

$$E(x, y) = \int d\lambda \rho(\lambda) A_x(\lambda) B_y(\lambda).$$

No nosso experimento, vamos estudar as 4 possibilidades de permutações de observáveis escolhidos por Alice e Bob e analisar uma certa combinação linear das suas esperanças.

Desigualdade CHSH

Sob as hipóteses de **realismo local**, a seguinte combinação linear de correlações satisfaz a desigualdade

$$-2 \leq S := E(0, 0) + E(0, 1) + E(1, 0) - E(1, 1) \leq 2.$$

Note que cada termo corresponde a uma escolha conjunta de observáveis feita por Alice e Bob.

Essa desigualdade é uma consequência puramente matemática da suposição de que os resultados das medições são determinados por variáveis ocultas locais.

2.7.3 Violação da Desigualdade CHSH

Considere o estado de Bell antissimétrico

$$|\Psi^-\rangle_{AB} = \frac{1}{\sqrt{2}} (|01\rangle_{AB} - |10\rangle_{AB}).$$

Escolhemos os seguintes observáveis locais, todos com autovalores ± 1 :

$$\begin{aligned} A_0 &= Z^A, & A_1 &= X^A, \\ B_0 &= -\frac{Z^B + X^B}{\sqrt{2}}, & B_1 &= \frac{-Z^B + X^B}{\sqrt{2}}. \end{aligned}$$

Para esse estado, valem as correlações quânticas conhecidas:

$$\begin{aligned} \langle Z^A Z^B \rangle &= -1, & \langle X^A X^B \rangle &= -1, \\ \langle Z^A X^B \rangle &= 0, & \langle X^A Z^B \rangle &= 0. \end{aligned}$$

A função de correlação quântica é dada por

$$E(x, y) = \langle A_x \otimes B_y \rangle.$$

Calculando cada termo da expressão CHSH, obtemos:

$$E(0, 0) = \left\langle Z^A \otimes \left(-\frac{Z^B + X^B}{\sqrt{2}} \right) \right\rangle = \frac{1}{\sqrt{2}},$$

$$E(0,1) = \left\langle Z^A \otimes \frac{Z^B - X^B}{\sqrt{2}} \right\rangle = \frac{1}{\sqrt{2}},$$

$$E(1,0) = \left\langle X^A \otimes \left(-\frac{Z^B + X^B}{\sqrt{2}} \right) \right\rangle = \frac{1}{\sqrt{2}},$$

$$E(1,1) = \left\langle X^A \otimes \frac{Z^B - X^B}{\sqrt{2}} \right\rangle = -\frac{1}{\sqrt{2}}.$$

Substituindo na combinação CHSH,

$$S = E(0,0) + E(0,1) + E(1,0) - E(1,1),$$

obtemos

$$S = 2\sqrt{2}.$$

Este é conhecido como o **limite de Tsirelson**.

Existem jeitos de realizar experimentos dessa medição, que sugerem a invalidade da Desigualdade CHSH. A violação experimental da desigualdade CHSH implica, portanto, que a Mecânica Quântica não pode ser descrita por nenhuma teoria de variáveis ocultas locais.

Existem ainda alguns físicos que contestam a validade desses experimentos, por meio de argumentos variados.

2.8 Aula 7 - Construção de Circuitos Quânticos

2.9 Aula 8 - Construção de Circuitos Quânticos II

2.9.1 Portas Clássicas

Operação XOR

Sejam $x = (x_1, \dots, x_n) \in \{0,1\}^n$ e $y = (y_1, \dots, y_n) \in \{0,1\}^n$ duas bitstrings de comprimento n . Define-se a operação *ou exclusivo* (XOR) como a aplicação

$$\oplus : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n,$$

dada componente a componente por

$$(x \oplus y)_i = x_i \oplus y_i = x_i + y_i \pmod{2}, \quad i = 1, \dots, n.$$

Equivalentemente,

$$x \oplus y = (x_1 + y_1 \pmod{2}, \dots, x_n + y_n \pmod{2}).$$

Operação AND

Sejam $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ e $y = (y_1, \dots, y_n) \in \{0, 1\}^n$ duas bitstrings de comprimento n . Define-se a operação *conjunção lógica* (AND) como a aplicação

$$\cdot : \{0, 1\}^n \times \{0, 1\}^n \longrightarrow \{0, 1\}^n,$$

dada componente a componente por

$$(x \cdot y)_i = x_i \cdot y_i = x_i y_i, \quad i = 1, \dots, n.$$

Equivalentemente,

$$x \cdot y = (x_1 y_1, \dots, x_n y_n).$$

Propriedades

- **Distributividade à esquerda:**

$$x \cdot (y \oplus z) = (x \cdot y) \oplus (x \cdot z).$$

- **Distributividade à direita:**

$$(x \oplus y) \cdot z = (x \cdot z) \oplus (y \cdot z).$$

- **Elemento neutro do produto:**

$$x \cdot 1^n = x.$$

- **Elemento absorvente do produto:**

$$x \cdot 0^n = 0^n.$$

- **Compatibilidade com a soma:**

$$x \cdot x = x, \quad x \oplus x = 0^n.$$

2.9.2 Oráculo: Como Implementar Funções em Circuitos Quânticos

Resumo (Oráculo)

Um **oráculo de XOR** em computação quântica é uma caixa preta que implementa de forma reversível uma função booleana $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ como uma operação unitária U_f sobre $n + 1$ qubits:

$$U_f : |x\rangle_{\text{entrada}} |y\rangle_{\text{alvo}} \mapsto |x\rangle_{\text{entrada}} |y \oplus f(x)\rangle_{\text{alvo}}, \quad x \in \{0, 1\}^n, y \in \{0, 1\}.$$

‘Propriedades:

- U_f é reversível (unitário) e a informação de $|x\rangle$ é preservada, pois não medimos o estado.
- Se o alvo estiver no estado $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, então a ação do oráculo reduz-se a uma *fase* na base computacional e é chamado de oráculo de fase:

$$U_f(|x\rangle |-\rangle) = (-1)^{f(x)} |x\rangle |-\rangle.$$

Essa propriedade é a chave para muitos algoritmos quânticos baseados em oráculos.

2.9.3 Deutsch: Problema da Função Constante ou Balanceada

Enunciado. Dada uma função $f : \{0, 1\} \rightarrow \{0, 1\}$, decidir se f é *constante* (mesmo valor para 0 e 1) ou *balanceada* (valores diferentes para 0 e 1), usando o menor número possível de consultas ao oráculo.

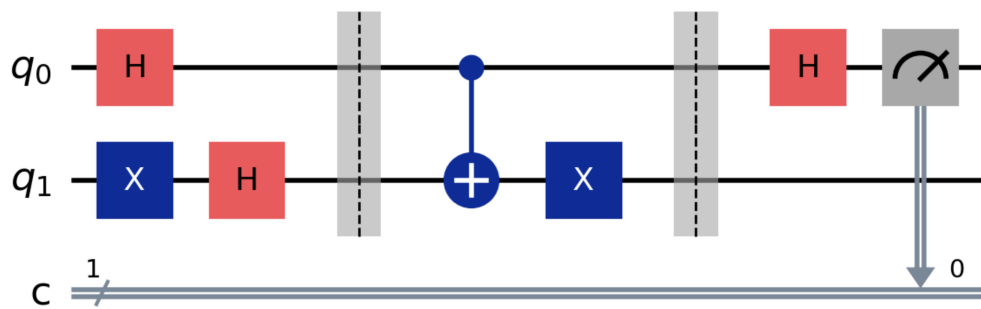


Figura 3: Circuito de Deutsch.

Circuito e evolução. Use um qubit de entrada e um qubit alvo:

$$|0\rangle |1\rangle \xrightarrow{H \otimes H} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

Aplicando o oráculo U_f com o alvo em $|-\rangle$ gera fases:

$$\frac{1}{\sqrt{2}}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle) \otimes |-\rangle.$$

Aplica-se então Hadamard ao qubit de entrada:

$$H\left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right) = \frac{1}{2}\left(((-1)^{f(0)} + (-1)^{f(1)})|0\rangle + ((-1)^{f(0)} - (-1)^{f(1)})|1\rangle\right).$$

Leitura:

- Se f é **constante**, então $(-1)^{f(0)} = (-1)^{f(1)}$, logo a amplitude de $|1\rangle$ é zero e a medição do qubit de entrada dá $|0\rangle$ com probabilidade 1.
- Se f é **balanceada**, então $(-1)^{f(0)} = -(-1)^{f(1)}$, logo a amplitude de $|0\rangle$ é zero e a medição dá $|1\rangle$ com probabilidade 1.

2.9.4 Deutsch–Jozsa: Generalização do Problema de Deutsch

Proposição 9 (Aplicação de Hadamards em $|0\rangle^{\otimes n}$) *Seja $n \in \mathbb{N}$ um número de qubits. Vale*

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle, \quad x \in \{0, 1\}^n.$$

Ou seja, aplicar hadamard em todos n qubits cria um estado com todas as combinações possíveis de bitstrings de comprimento n .

Proposição 10 (Aplicação de Hadamards em $H^{\otimes n}|0\rangle^{\otimes n}$) *Seja $n \in \mathbb{N}$ um número de qubits. Vale*

$$H^{\otimes n}(H^{\otimes n}|0\rangle^{\otimes n}) = \frac{1}{2^n} \sum_{x,y} (-1)^{x \cdot y} |y\rangle, \quad x, y \in \{0, 1\}^n.$$

Enunciado. Dada $f : \{0, 1\}^n \rightarrow \{0, 1\}$ com a garantia de que f é *ou* constante *ou* balanceada (i.e., retorna 0 para metade das entradas e 1 para a outra metade), decidir qual dos dois é com mínima quantidade de consultas.

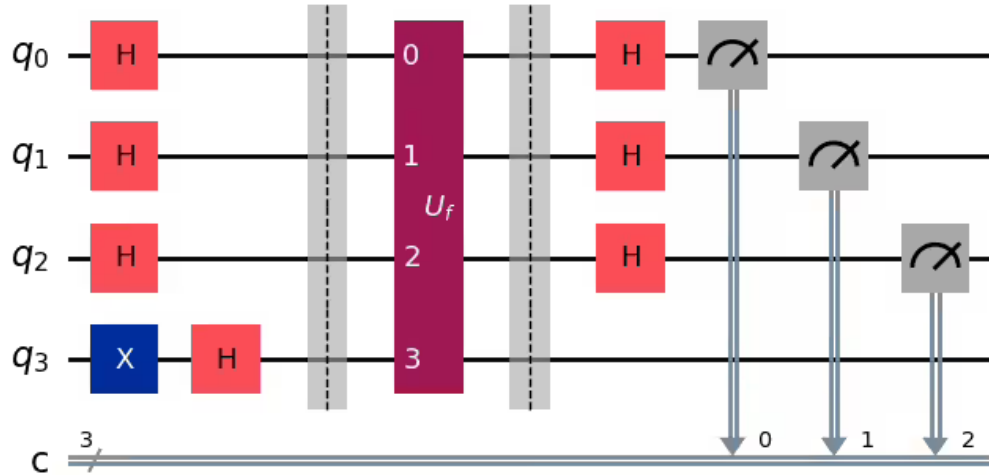


Figura 4: Circuito de Deutsch-Jozsa.

Circuito e evolução. Comece com $|0\rangle^{\otimes n} |1\rangle$. Aplique $H^{\otimes n}$ nos inputs e H no alvo:

$$\left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \right) \otimes |-\rangle.$$

Aplicando U_f (alvo em $|-\rangle$) produz fase $(-1)^{f(x)}$ em cada componente:

$$\frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle \otimes |-\rangle.$$

Aplica-se então $H^{\otimes n}$ aos n qubits de entrada:

$$\frac{1}{2^n} \sum_{x,y} (-1)^{f(x)} (-1)^{x \cdot y} |y\rangle |-\rangle.$$

Portanto, a amplitude do estado $|0\rangle^{\otimes n}$ após o segundo Hadamard é proporcional à soma

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}.$$

Leitura:

- Se f é **constante** e igual a $c \in \{0,1\}$, então $(-1)^{f(x)} = (-1)^c$ para todo x , logo

$$\frac{1}{2^n} \sum_x (-1)^{f(x)} = (-1)^c,$$

e portanto a amplitude em $|0\rangle^{\otimes n}$ tem módulo 1: a medição retorna $|0\rangle^{\otimes n}$ com probabilidade 1.

- Se f é **balanceada**, há exatamente 2^{n-1} termos $+1$ e 2^{n-1} termos -1 na soma, logo

$$\sum_x (-1)^{f(x)} = 0,$$

e a amplitude do vetor nulo é zero, então a medida nunca retorna $|0\rangle^{\otimes n}$.

A complexidade deste algoritmo é $O(1)$, pois precisamos aplicar a função apenas 1 vez (o mesmo vale para Deustch).

Já a complexidade do algoritmo clássico para uma função cujo domínio envolve n bits é $O(2^n)$. Essa complexidade vem do pior caso: o algoritmo precisa checar todas as possibilidades de bitstrings até vir alguma com um resultado diferente das anteriores; se esse resultado não vier, só resta checar se todos os resultados até a posição $\frac{2^n}{2} + 1$ são iguais.

2.9.5 Bernstein–Vazirani: Problema da Chave Secreta

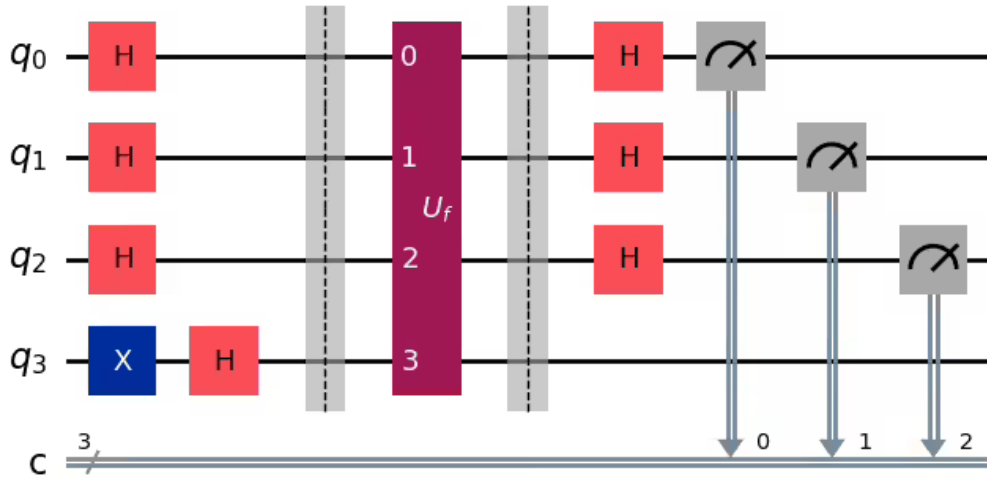


Figura 5: Circuito de Bernstein–Vazirani.

Enunciado. Existe uma chave secreta $s \in \{0,1\}^n$. O oráculo implementa um tipo de função de combinação linear:

$$f(x) = s \cdot x \pmod{2} = \bigoplus_{i=1}^n s_i x_i.$$

O objetivo é descobrir a string s com o mínimo de consultas.

Circuito e evolução. O esqueleto do circuito é basicamente o mesmo do algoritmo de Deustch-Jozsa.

Inicie em $|0\rangle^{\otimes n} |1\rangle$. Aplique $H^{\otimes n}$ aos inputs e H ao alvo para ter

$$\left(\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \right) \otimes |-\rangle.$$

O oráculo impõe fase $(-1)^{f(x)} = (-1)^{s \cdot x}$:

$$\frac{1}{\sqrt{2^n}} \sum_x (-1)^{s \cdot x} |x\rangle \otimes |-\rangle.$$

Aplicando $H^{\otimes n}$ aos inputs, usa-se a identidade

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle,$$

logo o estado dos inputs torna-se

$$\frac{1}{2^n} \sum_y \left(\sum_x (-1)^{s \cdot x} (-1)^{x \cdot y} \right) |y\rangle = \frac{1}{2^n} \sum_y \left(\sum_x (-1)^{x \cdot (s \oplus y)} \right) |y\rangle.$$

A soma interna $\sum_x (-1)^{x \cdot (s \oplus y)}$ é 2^n se $y = s$ e 0 caso contrário. Assim o estado final dos inputs é exatamente $|s\rangle$.

Leitura: Medindo os n qubits de entrada obtém-se $|s\rangle$ com probabilidade 1.

3 Bloco 02 – Algoritmos e Otimização

3.1 Aula 1 - Introdução à Teoria da Computação

3.1.1 Definições fundamentais

Problema computacional

Uma questão abstrata definida com uma relação entre dados de entrada (instância) e saída desejada (solução) que pode ser resolvida por uma sequência bem definida de passos mecânicos.

Máquina (algoritmo)

Sequência finita de passos elementares que transforma a instância de entrada em dados de saída (solução).

Complexidade de tempo

A complexidade de tempo de um algoritmo é uma função

$$T : \mathbb{N} \rightarrow \mathbb{R}^+,$$

na qual $T(n)$ denota o tempo para um algoritmo realizar um número de passos elementares ao processar uma entrada de tamanho n .

A definição de $T(n)$ depende de um modelo computacional fixado e de uma escolha explícita das operações consideradas elementares. Em geral, assume-se que cada instrução básica do modelo contribui com custo unitário, permitindo que o tempo de execução seja identificado com a contagem total dessas instruções.

Computabilidade

Todo problema para o qual existe uma máquina que o resolve é dito **computável**.

Decidibilidade

Todo problema computável para o qual a parada da máquina é garantida é dito **decidível**.

Tratabilidade

Todo problema decidível cuja complexidade de tempo é dada por um polinômio é dito **tratável**.

Determinismo

Todo problema que possui a mesma solução para uma dada instância é dito **determinístico**.

Existem problemas **não-determinísticos**, que possuem diferentes soluções para uma mesma instância, oriunda de alguma espécie de aleatoriedade no processo.

3.1.2 Decisão x Otimização**Problema de decisão**

Um problema é de **decisão** se a solução para ele é binária, ou seja, do tipo SIM, NÃO.

Problema de otimização

Um problema é de **otimização** se a sua solução consiste em determinar um vetor $x^* \in V \subseteq \mathbb{R}^n$ que minimiza (ou maximiza) uma função real

$$f : \mathbb{R}^n \rightarrow \mathbb{R},$$

denominada **função objetivo**, isto é,

$$x^* = \min_{x \in V} f(x).$$

O conjunto $V \subseteq \mathbb{R}^n$ é chamado de **conjunto viável** e contém todas as soluções que satisfazem as restrições do problema.

3.1.3 Complexidade de algoritmos

A teoria da complexidade se refere ao estudo da complexidade de tempo ou de espaço de um algoritmo. Aqui, focaremos somente na complexidade de tempo.

Essa análise pode ser feita por meio do melhor caso ($\inf T(n)$), caso médio ($\mathbb{E}(T(n))$) ou pior caso ($\sup T(n)$).

Usando $T(n)$ como uma função de complexidade de tempo, existem diversas notações que nos ajudam a padronizar a complexidade de algoritmos.

Notação Big O

Sejam $T, f : \mathbb{N} \rightarrow \mathbb{R}^+$. Diz-se que T é limitada superiormente por f , em ordem assintótica, se existem constantes $c > 0$ e $n_0 \in \mathbb{N}$ tais que

$$T(n) \leq c f(n), \quad \forall n \geq n_0.$$

Nesse caso, escreve-se $T(n) = O(f(n))$. Essa relação define uma classe de funções cujo crescimento é dominado por f para valores suficientemente grandes de n , abstraindo fatores constantes e efeitos de termos de menor ordem.

Esta é uma análise de **pior caso**.

Notação Omega

Sejam $T, f : \mathbb{N} \rightarrow \mathbb{R}^+$. Diz-se que T é limitada inferiormente por f , em ordem assintótica, se existem constantes $c > 0$ e $n_0 \in \mathbb{N}$ tais que

$$T(n) \geq c f(n), \quad \forall n \geq n_0.$$

Nesse caso, escreve-se $T(n) = \Omega(f(n))$. Essa relação define uma classe de funções cujo crescimento é, no mínimo, tão rápido quanto o de f para valores suficientemente grandes de n .

Esta é uma análise de **melhor caso**.

Notação Theta

Sejam $T, f : \mathbb{N} \rightarrow \mathbb{R}^+$. Diz-se que T é limitada superior e inferiormente por f . Nesse caso, escreve-se $T(n) = \Theta(f(n))$. Essa relação caracteriza funções que possuem a mesma ordem de crescimento assintótico, diferindo apenas por fatores constantes.

Esta é uma análise de **caso justo**.

3.1.4 Principais Comportamentos Assintóticos**Complexidade constante**

$$f(n) = O(1)$$

O tempo de execução é independente do tamanho da entrada. O algoritmo executa um número fixo de instruções, independentemente de n .

Complexidade logarítmica

$$f(n) = O(\log n)$$

Ocorre tipicamente em algoritmos que resolvem o problema reduzindo-o recursivamente a subproblemas menores, geralmente por divisão sucessiva do espaço de busca.

Complexidade linear

$$f(n) = O(n)$$

Em geral, uma quantidade constante de trabalho é realizada para cada elemento da entrada, resultando em crescimento proporcional a n .

Complexidade $n \log n$

$$f(n) = O(n \log n)$$

Complexidade característica de algoritmos que dividem o problema em subproblemas menores, resolvem cada um deles e, em seguida, combinam as soluções obtidas.

Complexidade polinomial

$$f(n) = O(n^k)$$

Para algum $k \in \mathbb{R}^+$. Muito importante para analisar classes de problemas, como veremos adiante.

Complexidade quadrática

$$f(n) = O(n^2)$$

Típica de algoritmos que processam pares de elementos da entrada, frequentemente implementados por dois laços aninhados.

Complexidade cúbica

$$f(n) = O(n^3)$$

Surge em algoritmos com três laços aninhados. Em geral, é viável apenas para instâncias pequenas devido ao rápido crescimento do custo computacional.

Complexidade exponencial

$$f(n) = O(2^n)$$

Comum em algoritmos de busca exaustiva, também conhecidos como força bruta. Tornam-se rapidamente inviáveis à medida que o tamanho da entrada cresce, sendo aplicáveis apenas a instâncias pequenas.

Complexidade fatorial

$$f(n) = O(n!)$$

Cresce mais rapidamente que a exponencial. Embora por vezes classificada como exponencial, apresenta um crescimento significativamente mais acentuado, limitando severamente sua aplicabilidade prática.

3.1.5 Classes de complexidade

Problemas P

Um problema de decisão pertence à classe P se existe um algoritmo determinístico cuja complexidade de tempo é limitada superiormente por um polinômio no tamanho da entrada.

Problemas co-P

Um problema de decisão pertence à classe $co-P$ se o seu complemento pertence à classe P . Isto é, um problema de decisão está em $co-P$ quando existe um algoritmo determinístico em tempo polinomial que decide as instâncias de forma contrária a algum problema da classe P .

Note que basta inverter a resposta final de um problema de $co-P$ para obter um problema de P correspondente. Portanto, as duas classes são iguais.

Proposição 11

$$P = co-P$$

Problemas NP

Um problema de decisão pertence à classe NP (Tempo Polinomial Não Determinístico) se existe um algoritmo que verifica a veracidade de uma resposta para ele em tempo polinomial.

Se um problema pode ser resolvido em tempo polinomial, então pode-se usar o mesmo algoritmo para checar uma dada resposta. Portanto, todo problema em P também é NP .

Proposição 12

$$P \subseteq NP$$

Classe BQP (Tempo Polinomial Quântico com Erro Limitado)

A classe BQP (*Bounded-Error Quantum Polynomial Time*) é o conjunto dos problemas de decisão que podem ser resolvidos por um circuito quântico uniforme em tempo polinomial, com probabilidade de erro limitada por uma constante menor que $\frac{1}{3}$ para todas as instâncias.

Tem-se a relação

$$P \subseteq BQP.$$

A classe BQP é frequentemente identificada como a classe de problemas computacionalmente viáveis para computadores quânticos.

3.2 Aula 2 - Algoritmos Quânticos I

3.2.1 Link de uma lista de algoritmos quânticos

<https://quantumalgorithmzoo.org/>

3.2.2 Algoritmo de Simon

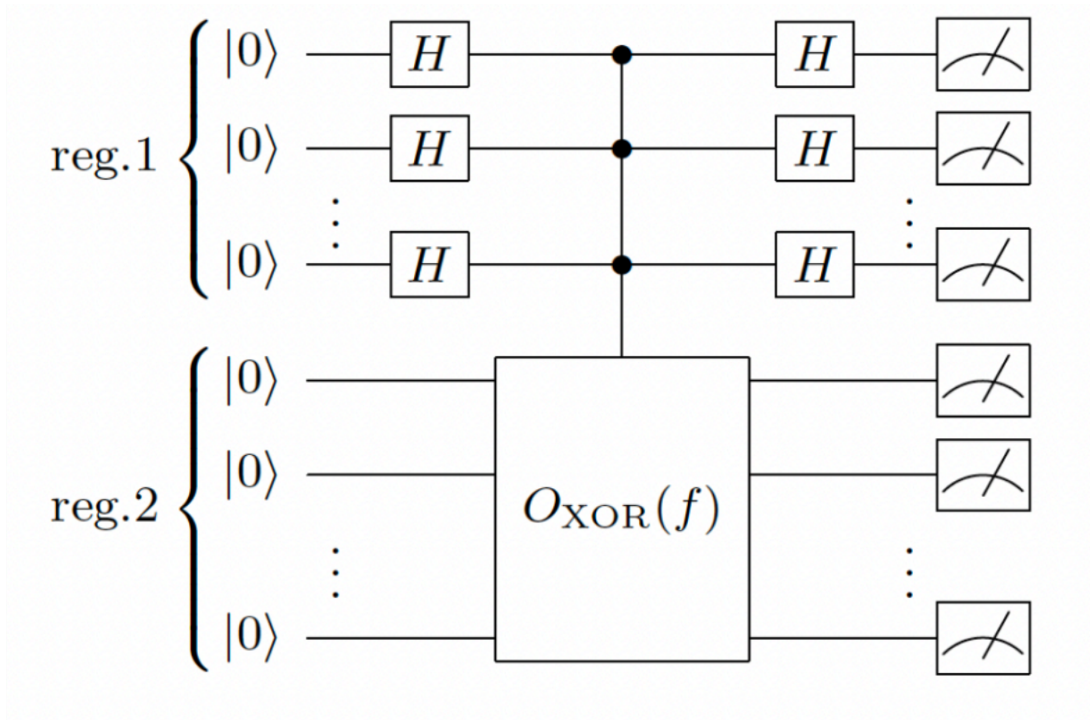


Figura 6: Circuito de Simon. Fonte: Brazil Quantum Camp.

O Algoritmo de Simon resolve um problema de identificação de periodicidade oculta em funções booleanas e constitui um dos primeiros exemplos de separação exponencial entre computação clássica e quântica no modelo de oráculo.

Este circuito usa dois registradores.

Enunciado

Considere uma função booleana periódica

$$f : \{0, 1\}^n \rightarrow \{0, 1\}^n$$

tal que existe uma string secreta $s \in \{0, 1\}^n$ com:

$$(f(x_1) = f(x_2)) \iff ((x_1 = x_2) \text{ ou } (x_2 = x_1 \oplus s)), \quad \forall x, y \in \{0, 1\}^n.$$

O objetivo é determinar s .

Circuito e evolução

1. Inicializa-se o sistema no estado

$$|0\rangle^{\otimes n} |0\rangle^{\otimes n}.$$

2. Aplica-se a transformada de Hadamard no primeiro registrador:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |x\rangle |0\rangle.$$

3. Aplica-se o oráculo U_f com alvos no segundo registrador, obtendo

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |0 \oplus f(x)\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle.$$

4. Aplica-se a transformada de Hadamard no primeiro registrador:

$$\frac{1}{2^n} \sum_x \sum_y (-1)^{x \cdot y} |y\rangle |f(x)\rangle.$$

5. Medição de todos os qubits na base computacional.
6. Repetir os passos anteriores até obter n equações linearmente independentes e resolver sistema linear para encontrar s .

Para explicar o passo 6, é necessário mais desenvolvimento algébrico.

Se o resultado da medida do passo 5 for $|y\rangle |z\rangle$, então $z = f(x)$.

Caso $s \neq 0$

Pelas propriedades da função do enunciado, para dois x_1 e x_2 distintos no somatório, vale

$$z = f(x_1) = f(x_2) \iff x_2 = x_1 \oplus c$$

Portanto, o estado $|y\rangle |z\rangle$ aparece duas vezes no somatório e podemos escrever a sua componente como

$$\begin{aligned} a_{y,z} &= \frac{1}{2^n} ((-1)^{x_1 \cdot y} + (-1)^{x_2 \cdot y}) \\ &= \frac{1}{2^n} ((-1)^{x_1 \cdot y} + (-1)^{(x_1 \oplus s) \cdot y}) \\ &= \frac{1}{2^n} ((-1)^{x_1 \cdot y} + (-1)^{(x_1 \cdot y) \oplus (s \cdot y)}) \\ &= \frac{1}{2^n} ((-1)^{x_1 \cdot y} + (-1)^{x_1 \cdot y} (-1)^{s \cdot y}) \\ &= \frac{1}{2^n} (-1)^{x_1 \cdot y} (1 + (-1)^{s \cdot y}) \\ &= \begin{cases} 0, & s \cdot y = 1, \\ \frac{2}{2^n} (-1)^{x_1 \cdot y}, & s \cdot y = 0. \end{cases} \end{aligned}$$

Observa-se que a amplitude é não-nula apenas quando

$$y \cdot s = 0.$$

Assim, cada execução do algoritmo fornece uma equação linear homogênea, que nos dá informações sobre s .

Após coletar $O(n)$ vetores y , resolve-se o sistema linear

$$\begin{cases} y_1 \cdot s = 0 \\ y_2 \cdot s = 0 \\ \dots \\ y_n \cdot s = 0 \end{cases}$$

obtendo s .

Caso $s = 0$

Nesse caso,

$$a_{y,z} = \frac{1}{2^n}((-1)^{x_1 \cdot y})$$

Note que agora qualquer vetor de bits y pode resultar da medição, e não apenas os perpendiculares a s .

Na prática, repetimos o algoritmo um certo número de vezes que permita concluir com alta probabilidade que $s = 0$ ou que $s \neq 0$. Note que, se for impossível satisfazer as equações $y_i \cdot s = 0$ no primeiro caso, então caímos no segundo caso e descobrimos que $s = 0$.

Complexidades

Consultas quânticas: $O(n)$

Consultas clássicas probabilísticas: $\Omega(2^{n/2})$.

Consultas clássicas determinísticas: $\Theta(2^n)$

3.2.3 Algoritmo de Busca de Grover

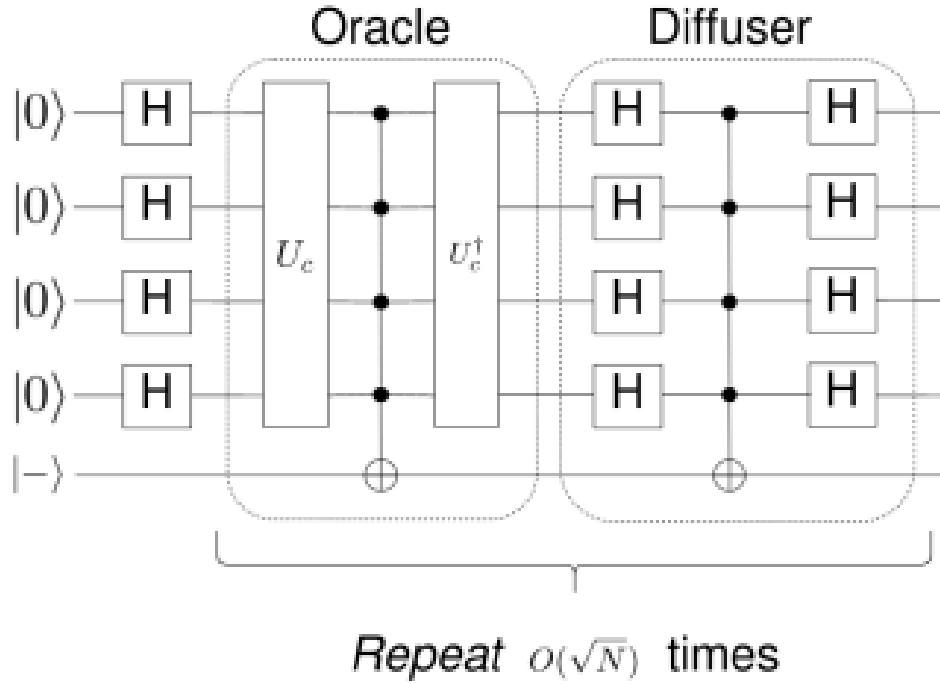


Figura 7: Circuito do Algoritmo de Grover. Fonte: Xanadu Quantum.

O Algoritmo de Busca de Grover resolve o problema de busca não estruturada em um conjunto de tamanho $N = 2^n$, fornecendo uma aceleração quadrática em relação aos algoritmos clássicos.

Este circuito utiliza um único registrador de n qubits e um oráculo quântico que identifica os estados solução.

Enunciado

Considere uma função booleana

$$f : \{0,1\}^n \rightarrow \{0,1\},$$

tal que existe pelo menos um elemento $x_0 \in \{0,1\}^n$ satisfazendo

$$f(x_0) = 1, \quad f(x) = 0 \text{ para } x \neq x_0.$$

O objetivo é encontrar o valor de x_0 .

Oráculo

O oráculo é implementado como um operador de fase

$$O_f |x\rangle = (-1)^{f(x)} |x\rangle,$$

isto é, ele inverte a fase apenas do estado solução.

Circuito e evolução

1. Inicializa-se o sistema no estado

$$|0\rangle^{\otimes n}.$$

2. Aplica-se a transformada de Hadamard em todos os qubits, obtendo

$$|\psi\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

3. Aplica-se o oráculo O_f , que produz

$$\frac{1}{\sqrt{N}} \left(\sum_{x \neq x_0} |x\rangle - |x_0\rangle \right).$$

4. Aplica-se o operador de difusão

$$D := 2|\psi\rangle\langle\psi| - I = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n},$$

que corresponde a uma reflexão em torno do eixo de $|\psi\rangle$.

5. Repete-se a aplicação do operador de Grover

$$G = DO_f$$

um certo número de vezes.

6. Mede-se o registrador na base computacional.

Análise geométrica

A evolução do algoritmo ocorre no subespaço bidimensional gerado pelos vetores

$$|x_0\rangle \quad \text{e} \quad |\alpha\rangle,$$

onde $|\alpha\rangle$ é a componente de $|\psi\rangle$ ortogonal ao estado solução.

O estado inicial pode ser escrito como

$$|\psi\rangle = \sin(\theta/2) |x_0\rangle + \cos(\theta/2) |\alpha\rangle, \quad \text{com} \quad \sin(\theta/2) = \frac{1}{\sqrt{N}}.$$

Proposição 13 (Geometria do Algoritmo de Grover) *Cada aplicação do operador de Grover G corresponde a uma rotação anti-horária de ângulo θ nesse subespaço.*

Prova

Considere o subespaço bidimensional

$$\mathcal{H}_2 = \text{span}\{|x_0\rangle, |\alpha\rangle\},$$

onde $|\alpha\rangle$ é normalizado e satisfaz

$$\langle x_0 | \alpha \rangle = 0.$$

O estado inicial do algoritmo é

$$|\psi\rangle = \sin(\theta/2) |x_0\rangle + \cos(\theta/2) |\alpha\rangle, \quad \sin(\theta/2) = \frac{1}{\sqrt{N}}.$$

Aplicando o oráculo ao estado $|\psi\rangle$, obtemos

$$O_f |\psi\rangle = -\sin(\theta/2) |x_0\rangle + \cos(\theta/2) |\alpha\rangle.$$

Calculamos inicialmente

$$\langle\psi||O_f\psi\rangle = -\sin^2(\theta/2) + \cos^2(\theta/2) = \cos(\theta).$$

Assim,

$$\begin{aligned} DO_f |\psi\rangle &= (2 |\psi\rangle \langle\psi| - I) O_f |\psi\rangle \\ &= 2 \cos(\theta) |\psi\rangle - (-\sin(\theta/2) |x_0\rangle + \cos(\theta/2) |\alpha\rangle). \end{aligned}$$

Substituindo a expressão de $|\psi\rangle$, temos

$$\begin{aligned} G |\psi\rangle &= 2 \cos(\theta) (\sin(\theta/2) |x_0\rangle + \cos(\theta/2) |\alpha\rangle) + \sin(\theta/2) |x_0\rangle - \cos(\theta/2) |\alpha\rangle \\ &= (2 \cos(\theta) \sin(\theta/2) + \sin(\theta/2)) |x_0\rangle + (2 \cos(\theta) \cos(\theta/2) - \cos(\theta/2)) |\alpha\rangle. \end{aligned}$$

Usando identidades trigonométricas, obtemos

$$\begin{aligned} 2 \cos(\theta) \sin(\theta/2) + \sin(\theta/2) &= \sin(3\theta/2), \\ 2 \cos(\theta) \cos(\theta/2) - \cos(\theta/2) &= \cos(3\theta/2). \end{aligned}$$

Logo,

$$G |\psi\rangle = \sin(3\theta/2) |x_0\rangle + \cos(3\theta/2) |\alpha\rangle.$$

Portanto, cada aplicação do operador de Grover G corresponde a uma rotação anti-horária de ângulo θ nesse subespaço bidimensional. \square

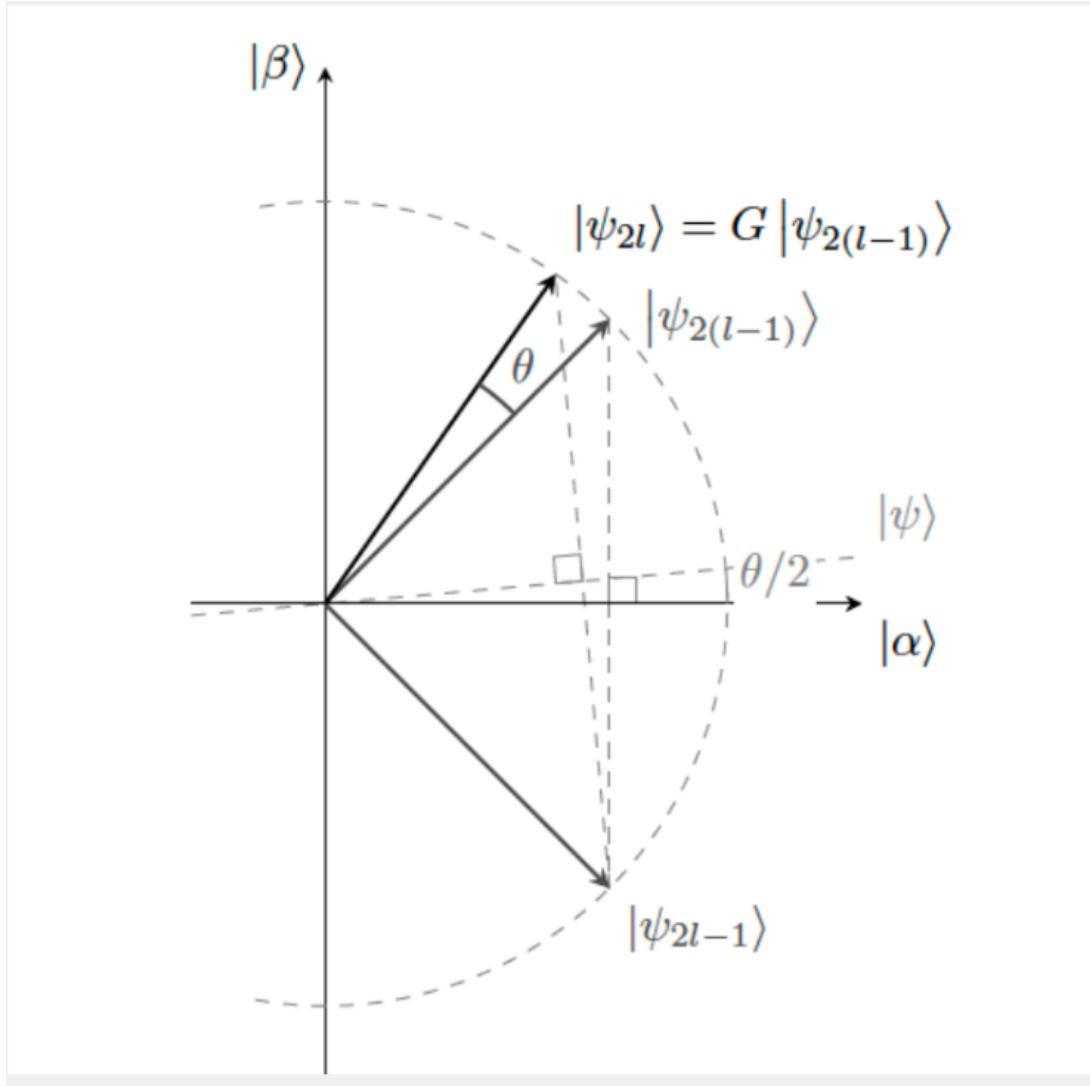


Figura 8: Visualização geométrica do Algoritmo de Grover. Fonte: https://aprenda.quantumket.org/algoritmos/busca/09_2_0-grover.html.

Após k iterações, o vetor rotaciona $k\theta$ a partir do ângulo inicial $\theta/2$, então o estado do sistema é

$$\sin((k + 1/2)\theta) |x_0\rangle + \cos((k + 1/2)\theta) |\alpha\rangle.$$

A probabilidade de medir o estado solução é maximizada quando o vetor se torna aproximadamente paralelo ao estado $|x_0\rangle$, alcançando um ângulo aproximadamente igual a

$$(k + 1/2)\theta \approx \frac{\pi}{2},$$

o que ocorre para

$$k \approx \frac{\pi}{4} \sqrt{N}.$$

Após esse número de iterações, a medição do registrador retorna x_0 com alta probabilidade, próxima de 1.

Complexidades

Algoritmo quântico:

$$O(\sqrt{N})$$

Algoritmo clássico:

$$O(N)$$

O algoritmo de Grover é ótimo no modelo de consulta, pois nenhum algoritmo quântico pode resolver o problema de busca não estruturada com menos de $\Omega(\sqrt{N})$ consultas ao oráculo.

3.3 Aula 3 - Algoritmos Quânticos II

3.3.1 Transformada de Fourier Clássica

A Transformada de Fourier Clássica é uma ferramenta matemática fundamental que permite representar uma função em termos de suas componentes de frequência. Ela estabelece uma mudança de base, trocando a descrição no *domínio do tempo* (ou do espaço discreto) pela descrição no *domínio das frequências*.

Motivação

Muitos fenômenos físicos, sinais e funções apresentam estruturas que não são facilmente identificáveis em sua representação original, mas tornam-se simples quando expressos como superposição de oscilações periódicas. A Transformada de Fourier fornece exatamente essa decomposição.

Caso contínuo

Dada uma função integrável

$$f : \mathbb{R} \rightarrow \mathbb{C},$$

define-se a Transformada de Fourier de f como

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt.$$

A quantidade $\hat{f}(\omega)$ mede a contribuição da frequência ω na função original. Quanto maior o seu valor, mais aquela frequência é relevante na superposição.

Caso discreto

Transformada Discreta de Fourier (DFT)

Para funções definidas em um conjunto finito, considera-se a *Transformada Discreta de Fourier (DFT)*. Dado um vetor

$$x = (x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N,$$

define-se sua transformada como

$$X := DFT(x) = (x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N,$$

tal que os seus componentes são

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{-2\pi i kn/N}, \quad k = 0, 1, \dots, N-1.$$

A DFT pode ser escrita como uma multiplicação matricial:

$$X = DFT_N x,$$

onde DFT_N é a matriz de Fourier de ordem N , cujos elementos são

$$(DFT_N)_{k,n} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{2\pi i/N} & (e^{2\pi i/N})^2 & \dots & (e^{2\pi i/N})^{N-1} \\ 1 & (e^{2\pi i/N})^2 & (e^{2\pi i/N})^4 & \dots & (e^{2\pi i/N})^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (e^{2\pi i/N})^{N-1} & \dots & \dots & (e^{2\pi i/N})^{(N-1)^2} \end{pmatrix}.$$

Complexidade computacional

O cálculo direto da DFT exige

$$O(N^2)$$

operações. Algoritmos clássicos como a *Transformada Rápida de Fourier (FFT)* reduzem essa complexidade para

$$O(N \log N).$$

3.3.2 Transformada Quântica de Fourier

Na definição abaixo, usaremos uma notação para estados compostos que considera uma equivalência nas representações de números em base 10 e em base 2.

Exemplo:

$$|3\rangle = |011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle.$$

Transformada Quântica de Fourier (QFT)

No contexto da Computação Quântica, considera-se a *Transformada Quântica de Fourier (QFT)*, que é a versão unitária da Transformada Discreta de Fourier atuando sobre estados quânticos.

Seja $N = 2^n$ e considere um estado arbitrário de n qubits,

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle, \quad \alpha_x \in \mathbb{C}.$$

Define-se a ação da QFT sobre a base computacional por

$$\text{QFT}_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i x k / N} |k\rangle, \quad x = 0, 1, \dots, N-1.$$

Por linearidade, a QFT aplicada a um estado arbitrário resulta em

$$\text{QFT}_N |\psi\rangle = \sum_{k=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \alpha_x e^{2\pi i x k / N} \right) |k\rangle.$$

A QFT pode ser vista como um operador unitário

$$\text{QFT}_N : \mathcal{H}_N \rightarrow \mathcal{H}_N,$$

cuja matriz na base computacional coincide com a matriz de Fourier normalizada, com elementos

$$(\text{QFT}_N)_{k,x} = \frac{1}{\sqrt{N}} e^{2\pi i k x / N}, \quad k, x = 0, 1, \dots, N-1.$$

Porta de Fase R_r

Considere uma porta quântica de um único qubit que realiza uma rotação em torno do eixo z da esfera de Bloch por um ângulo

$$\theta = \frac{2\pi}{2^r}.$$

Define-se a *porta de fase* R_r como o operador unitário

$$R_r = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^r} \end{pmatrix}.$$

É possível provar que uma certa associação de portas H e R_r geram o operador QFT .

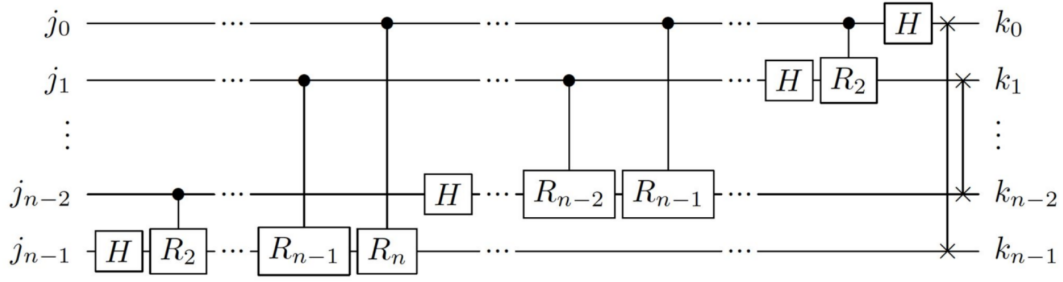


Figura 9: Circuito que implementa a QFT. Fonte: https://www.ictp-saifr.org/wp-content/uploads/2022/11/ICTP_SAIFR_D2.pdf.

As últimas portas do circuito da Figura 9 servem apenas para trocar de lugar os seus dois qubits marcados. Isto é necessário, pois, sem elas, a bitstring obtida na medição desses qubits viria invertida de trás para frente.

A QFT nada mais é que uma série de rotações de fase controladas cada vez menores. Ela transforma a informação codificada na amplitude (base computacional) em fase (base de Fourier). Assim, ao invés de representar números base 2 como 1s e 0s, os representamos como qubits em superposição.

Complexidade computacional

Para um registrador de n qubits ($N = 2^n$), a Transformada Quântica de Fourier QFT_N pode ser implementada por um circuito quântico composto por portas de Hadamard e portas de fase controladas R_r .

O número total de portas elementares necessárias é da ordem de

$$\sum_{k=1}^n k = \frac{n(n+1)}{2},$$

isto é, cresce quadraticamente com o número de qubits.

Portanto, a complexidade do circuito quântico que implementa a QFT é

$$O(n^2),$$

ou, equivalentemente, em função do tamanho do espaço de Hilbert $N = 2^n$,

$$O((\log N)^2).$$

3.3.3 Transformada Quântica de Fourier Inversa

Transformada Quântica de Fourier Inversa (IQFT)

A Transformada Quântica de Fourier Inversa (IQFT) é definida como o operador adjunto (conjugado transposto) da QFT. Por se tratar de um operador unitário, a IQFT satisfaz

$$\text{IQFT}_N = \text{QFT}_N^\dagger, \quad \text{IQFT}_N \text{QFT}_N = I.$$

Seja $N = 2^n$ e considere a base computacional. A ação da IQFT sobre essa base é dada por

$$\text{IQFT}_N |k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{-2\pi i x k / N} |x\rangle, \quad k = 0, 1, \dots, N-1.$$

Por linearidade, para um estado arbitrário

$$|\phi\rangle = \sum_{k=0}^{N-1} \beta_k |k\rangle, \quad \beta_k \in \mathbb{C},$$

tem-se

$$\text{IQFT}_N |\phi\rangle = \sum_{x=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \beta_k e^{-2\pi i x k / N} \right) |x\rangle.$$

A IQFT pode ser vista como um operador unitário

$$\text{IQFT}_N : \mathcal{H}_N \rightarrow \mathcal{H}_N,$$

cuja matriz na base computacional possui elementos

$$(\text{IQFT}_N)_{x,k} = \frac{1}{\sqrt{N}} e^{-2\pi i x k / N}, \quad x, k = 0, 1, \dots, N-1.$$

Operacionalmente, a IQFT é implementada por um circuito quântico obtido ao inverter a ordem das portas da QFT e substituir cada porta de fase R_r por sua adjunta R_r^\dagger (note que $H = H^\dagger$, mantendo a mesma complexidade computacional $O(n^2)$).

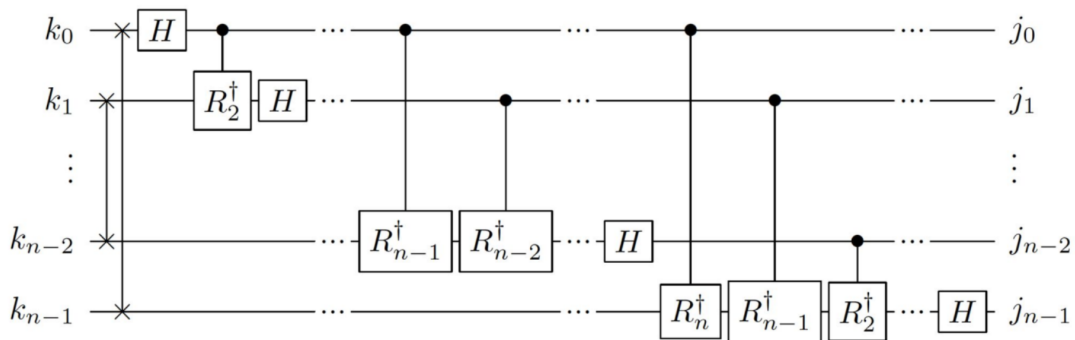


Figura 10: Circuito que implementa a IQFT. Fonte: https://www.ictp-saifr.org/wp-content/uploads/2022/11/ICTP_SAIIR_D2.pdf.

3.3.4 Estimação de Fase

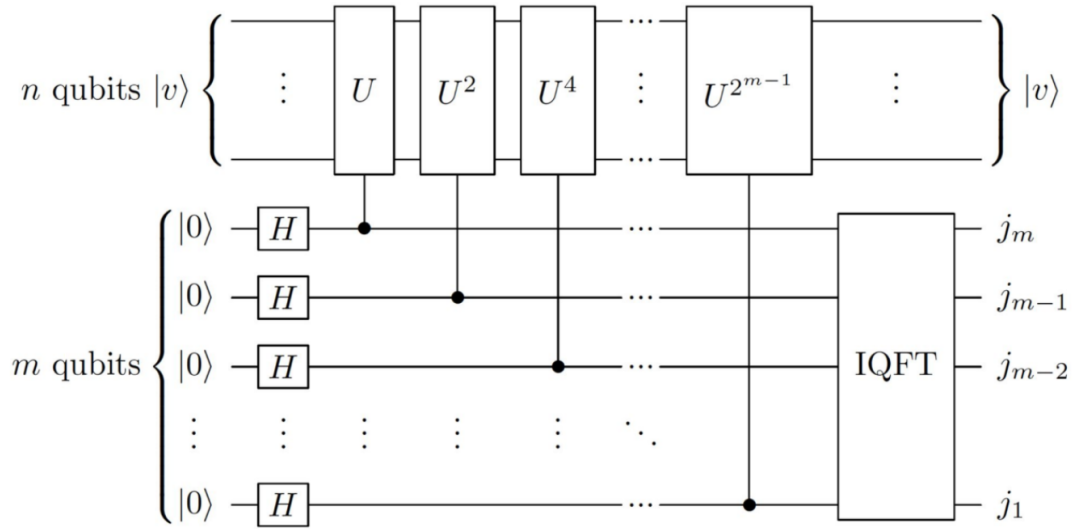


Figura 11: Circuito que implementa a QPE. Fonte: Brazil Quantum Camp.

A Estimação de Fase Quântica (Quantum Phase Estimation, QPE) é um algoritmo quântico fundamental cujo objetivo é estimar a fase associada a um autovalor de um operador unitário. Trata-se de uma sub-rotina central em vários algoritmos quânticos, como o algoritmo de Shor e simulações quânticas de sistemas físicos.

Enunciado

Seja U um operador unitário e seja $|u\rangle$ um autovetor de U tal que

$$U |u\rangle = e^{2\pi i \phi} |u\rangle, \quad \phi \in [0, 1).$$

O problema da estimação de fase consiste em determinar o valor de ϕ com alta precisão.

Descrição do algoritmo

O circuito de estimação de fase utiliza dois registradores:

- um registrador de controle com n qubits, inicialmente no estado $|0\rangle^{\otimes n}$;
- um registrador alvo preparado no autovetor $|u\rangle$.

Inicialmente, aplica-se a porta de Hadamard em todos os qubits do registrador de controle, produzindo a superposição

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle.$$

Em seguida, são aplicadas portas controladas

$$CU^{2^j},$$

onde o j -ésimo qubit de controle aplica a potência U^{2^j} ao registrador alvo. Como $|u\rangle$ é autovetor de U , o estado global passa a ser

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i k \phi} |k\rangle |u\rangle.$$

Após a aplicação das portas controladas, aplica-se a Transformada Quântica de Fourier Inversa (IQFT) ao registrador de controle. Essa operação converte a informação de fase, codificada nas amplitudes complexas, em informação clássica nos estados computacionais.

Ao medir o registrador de controle, obtém-se uma aproximação binária de ϕ , com erro que decresce exponencialmente com o número de qubits n .

Leitura

Se ϕ possui uma expansão binária finita com n bits,

$$\phi = 0.\phi_1\phi_2\ldots\phi_n,$$

então o algoritmo retorna exatamente essa sequência de bits com probabilidade unitária. Caso contrário, o valor medido corresponde à melhor aproximação de ϕ com n bits, com alta probabilidade.

3.3.5 Criptografia RSA

O RSA é um esquema de criptografia assimétrica baseado em propriedades da aritmética modular e na dificuldade computacional da fatoração de inteiros grandes. Ele utiliza um par de chaves: uma *chave pública*, usada para criptografar mensagens, e uma *chave privada*, usada para descriptografá-las.

Geração das chaves

1. Escolhem-se dois números primos grandes e distintos,

$$p \text{ e } q.$$

Em aplicações práticas como o RSA-2048, esses primos possuem cerca de 300 dígitos decimais cada.

2. Calcula-se o número

$$N = pq,$$

que será utilizado tanto na chave pública quanto na chave privada.

3. Calcula-se a função totiente de Euler de N :

$$\Phi(N) = (p-1)(q-1).$$

4. Escolhe-se um inteiro e tal que

$$1 < e < \Phi(N) \text{ e } \gcd(e, \Phi(N)) = 1.$$

Na prática, escolhe-se um e pequeno e ímpar para tornar a criptografia mais eficiente.

5. Determina-se d como o inverso multiplicativo de e módulo $\Phi(N)$, isto é,

$$ed \equiv 1 \pmod{\Phi(N)}.$$

Chaves

- A *chave pública* do RSA é o par

$$P = (e, N),$$

que pode ser divulgada livremente.

- A *chave privada* do RSA é o par

$$S = (d, N),$$

que deve ser mantido em sigilo.

Criptografia e descriptografia

Seja m uma mensagem representada como um inteiro tal que $0 \leq m < N$.

- A criptografia é feita utilizando a chave pública:

$$c \equiv m^e \pmod{N},$$

onde c é o texto cifrado.

- A descriptografia é feita utilizando a chave privada:

$$m \equiv c^d \pmod{N}.$$

A correção do método decorre do teorema de Euler, que garante que

$$m^{ed} \equiv m \pmod{N},$$

desde que m seja coprimo com N (ou, mais geralmente, pelo teorema chinês do resto).

Por que ela é tão segura?

A segurança do RSA baseia-se no fato de que, conhecendo apenas a chave pública (e, N) , é computacionalmente inviável calcular a chave privada d sem fatorar N em seus fatores primos p e q . Para tamanhos de chave adequados, não se conhecem algoritmos clássicos eficientes que resolvam esse problema em tempo viável.

3.3.6 Fundamentos do Algoritmo de Shor

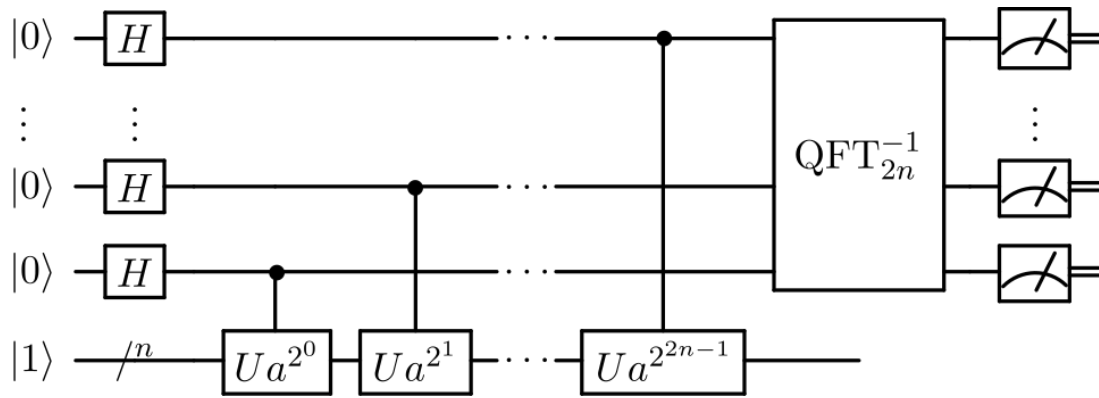


Figura 12: Circuito que implementa a parte quântica do Algoritmo de Shor. Fonte: Wikipedia.

O Algoritmo de Shor é um algoritmo quântico que resolve, em tempo polinomial, os problemas da fatoração de inteiros por meio do cálculo do período de funções aritméticas.

Shor demonstrou, pela primeira vez, que computadores quânticos podem resolver certos problemas de interesse prático de forma exponencialmente mais rápida do que computadores clássicos. Por esse motivo, ele é um dos principais motivadores do desenvolvimento da computação quântica e da pesquisa em criptografia pós-quântica.

Enunciado

Dado um inteiro composto

$$N = pq,$$

onde p e q são primos grandes, o objetivo é determinar seus fatores para quebrar a criptografia RSA.

Linha geral do algoritmo

O algoritmo de Shor baseia-se na observação de que a fatoração de N pode ser reduzida ao problema de encontrar o período de uma função. Escolhe-se um inteiro a tal que

$$\gcd(a, N) = 1,$$

e define-se a função

$$f(x) = a^x \bmod N.$$

Essa função é periódica, isto é, existe um menor inteiro $r > 0$ tal que

$$a^r \equiv 1 \pmod{N}.$$

Esse inteiro r é chamado de *período* da função.

Uma vez conhecido r , se r for par e

$$a^{r/2} \not\equiv -1 \pmod{N},$$

então os fatores de N podem ser obtidos por

$$\gcd(a^{r/2} \pm 1, N),$$

com alta probabilidade.

Complexidade

O algoritmo de Shor fatoriza um inteiro N em tempo

$$O((\log N)^3),$$

considerando a implementação eficiente das operações aritméticas modulares. Isso representa uma separação exponencial entre a complexidade clássica conhecida e a complexidade quântica do problema.

3.4 Aula 4 - Introdução a Algoritmos Clássicos

3.4.1 Heurísticas e Metaheurísticas

Muitas vezes, é difícil procurar diretamente a solução ótima de um problema de otimização.

Heurísticas e Metaheurísticas

Heurística é tipo de algoritmo que busca uma solução para o problema de otimização, que não necessariamente é a melhor.

Metaheurística é um tipo de algoritmo que busca melhorar várias heurísticas de forma a se aproximar mais da solução ótima.

Exemplos de algoritmos de otimização que usam essas estratégias se encontram no notebook do Google Colab.

3.4.2 Alguns problemas NP-Difíceis

Problema do Caixeiro Viajante (PCV)

Considere um conjunto de n cidades, rotuladas por $[n] = \{1, 2, \dots, n\}$, e uma matriz de distâncias

$$d_{ij} \geq 0, \quad \forall i, j \in [n],$$

onde d_{ij} representa o custo (ou distância) para viajar da cidade i à cidade j .

Uma **rota** é uma permutação $\pi \in S_n$, que define um ciclo Hamiltoniano (visitando cada cidade exatamente uma vez e retornando à cidade de origem).

O **Problema do Caixeiro Viajante** consiste em encontrar a permutação $\pi^* \in S_n$ que minimiza o custo total do ciclo:

$$\pi^* = \arg \min_{\pi \in S_n} \sum_{i=1}^n d_{\pi_i, \pi_{i+1}},$$

onde se impõe a condição $\pi_{n+1} = \pi_1$ para garantir o retorno ao ponto inicial.

Problema do Corte Máximo (MaxCut)

Seja $G = (V, E)$ um grafo não direcionado, com conjunto de vértices $V = \{1, 2, \dots, n\}$ e conjunto de arestas $E \subseteq V \times V$. A cada aresta $(i, j) \in E$ associa-se um peso $w_{ij} \geq 0$.

Um **corte** do grafo é uma partição dos vértices em dois subconjuntos disjuntos $S \subset V$ e $V \setminus S$. Dizemos que uma aresta (i, j) é *cortada* se $i \in S$ e $j \in V \setminus S$ (ou vice-versa).

O **Problema do Corte Máximo (MaxCut)** consiste em encontrar um conjunto $S^* \subset V$ que maximiza o peso total das arestas cortadas, isto é,

$$S^* = \arg \max_{S \subset V} \sum_{(i,j) \in E} w_{ij} \mathbf{1}[(i, j) \text{ é cortada}].$$

Equivalentemente, introduzindo variáveis binárias $x_i \in \{-1, +1\}$ associadas a cada vértice, o problema pode ser escrito como

$$\max_{x \in \{-1, +1\}^n} \frac{1}{2} \sum_{(i,j) \in E} w_{ij} (1 - x_i x_j).$$

Problema 3-SAT

Problema de satisfação booleana com 3 variáveis por cláusula. O objetivo é encontrar os valores binários $\{x_1, x_2, \dots, x_n\}$ que satisfazem uma forma normal conjuntiva $C \in \{0, 1\}$ da forma

$$C = \bigwedge_{j=1}^m (\ell_{j1} \vee \ell_{j2} \vee \ell_{j3}),$$

onde cada ℓ_{jk} é um literal, isto é, $\ell_{jk} \in \{x_i, \neg x_i\}$ para algum $i \in \{1, \dots, n\}$.

Problema da Mochila (Knapsack)

Considere um conjunto de n itens, rotulados por $[n] = \{1, 2, \dots, n\}$. Cada item $i \in [n]$ possui um **valor** $v_i \geq 0$ e um **peso** $w_i \geq 0$.

Seja W a capacidade máxima da mochila.

Uma solução é representada por um vetor binário

$$x = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n,$$

onde $x_i = 1$ indica que o item i é colocado na mochila e $x_i = 0$ indica que ele não é selecionado.

O **Problema da Mochila** consiste em determinar um vetor $x^* \in \{0, 1\}^n$ que maximize o valor total dos itens selecionados, respeitando a restrição de capacidade:

$$x^* = \arg \max_{x \in \{0, 1\}^n} \sum_{i=1}^n v_i x_i,$$

sujeito à restrição

$$\sum_{i=1}^n w_i x_i \leq W.$$

3.4.3 Algoritmo Genético

Um algoritmo genético (GA) é um método de otimização populacional inspirado em mecanismos de seleção natural. Ele mantém uma população de soluções candidatas que evolui ao longo de gerações por meio de seleção, recombinação e mutação.

Considere um problema de otimização definido por uma função de aptidão $f : \{0, 1\}^n \rightarrow \mathbb{R}$. Uma instância básica de algoritmo genético é descrita pelas seguintes etapas:

1. **Representação.** Cada indivíduo (ou solução candidata) é representado por uma cadeia de bits

$$s = (s_1, s_2, \dots, s_n) \in \{0, 1\}^n,$$

chamada de *genótipo*.

2. **População inicial.** Inicializa-se uma população $T^{(0)}$ contendo μ indivíduos, gerados de forma aleatória e independente.
3. **Seleção.** Selecionam-se $|P| = \mu$ pares de indivíduos da população atual. Cada indivíduo s é escolhido com probabilidade proporcional à sua aptidão $f(s)$, favorecendo soluções de maior qualidade.
4. **Recombinação (crossover).** Para cada par de pais selecionado, aplica-se um operador de recombinação em um ponto, produzindo duas novas soluções (descendentes) que combinam partes dos genótipos dos pais. Essa parte é inspirada no crossover do DNA, que acontece durante a meiose de seres vivos.

5. **Mutação.** Cada bit de cada nova solução gerada é invertido com uma probabilidade fixa p , tipicamente escolhida como $p = \frac{1}{n}$. A mutação introduz diversidade genética na população.
6. **Atualização da população.** O conjunto de descendentes gerado possui tamanho λ . A nova população $T^{(t+1)}$ é formada selecionando-se os μ melhores indivíduos entre os descendentes, caracterizando uma estratégia de seleção (μ, λ) .

O processo é repetido por um número fixo de gerações ou até que um critério de parada seja satisfeito.

3.4.4 Algoritmo Genético Inspirado em Quântica (QIGA)

Um Algoritmo Genético Inspirado em Quântica (QIGA) é um método de otimização populacional que combina princípios de algoritmos genéticos com conceitos da mecânica quântica, como superposição e observação. Apesar do nome, o QIGA é executado em computadores clássicos.

Considere um problema de otimização definido por uma função de aptidão $f : \{0, 1\}^n \rightarrow \mathbb{R}$. Uma instância básica de QIGA é descrita pelas seguintes etapas:

1. **Representação.** Cada indivíduo é representado por uma sequência de *bits quânticos* (q-bits),

$$q = (q_1, q_2, \dots, q_n), \quad q_i = \begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} \in \mathbb{R}^2,$$

onde $|\alpha_i|^2 + |\beta_i|^2 = 1$. Os coeficientes α_i e β_i representam as amplitudes de probabilidade dos estados clássicos 0 e 1, respectivamente.

2. **População inicial.** Inicializa-se uma população quântica $Q^{(0)}$ contendo μ indivíduos, tipicamente com amplitudes uniformes,

$$\alpha_i = \beta_i = \frac{1}{\sqrt{2}},$$

representando uma superposição equiprovável de todas as soluções clássicas.

3. **Observação (medição).** Cada indivíduo quântico é observado, gerando uma população clássica $T^{(t)} \subset \{0, 1\}^n$. O bit observado s_i assume valor 1 com probabilidade $|\beta_i|^2$ e valor 0 com probabilidade $|\alpha_i|^2$.
4. **Avaliação e seleção.** As soluções clássicas observadas são avaliadas pela função de aptidão f . Indivíduos de maior aptidão são utilizados como referência para guiar a evolução da população quântica.
5. **Atualização quântica.** Os q-bits são atualizados por meio do operador de rotação, ajustando as amplitudes (α_i, β_i) de modo a aumentar a probabilidade de observar bits consistentes com as melhores soluções encontradas. Essa etapa substitui os operadores clássicos de crossover e mutação.
6. **Atualização da população.** A população quântica atualizada define a próxima geração $Q^{(t+1)}$. O processo de observação, avaliação e atualização é repetido até que um critério de parada seja satisfeito.

3.4.5 QUBO

Quadratic Unconstrained Binary Optimization (QUBO)

Um problema de **Otimização Binária Quadrática sem Restrições (QUBO)** consiste em determinar um vetor binário

$$x = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$$

que minimiza uma função objetivo quadrática $f : \{0, 1\}^n \rightarrow \mathbb{R}$ da forma

$$f(x) = x^\top Q x = \sum_{i=1}^n \sum_{j=1}^n (Q_{ij} x_i x_j),$$

onde $Q \in \mathbb{R}^{n \times n}$ é uma matriz real (tipicamente simétrica), cujos elementos definem os coeficientes lineares e quadráticos do problema.

Para separar visualmente os termos lineares e quadráticos, pode-se dividir o somatório em dois somatórios menores. A função objetivo pode ser então escrita como

$$f(x) = \sum_{i=1}^n Q_{ii} x_i + \sum_{1 \leq i \neq j \leq n} Q_{ij} x_i x_j,$$

pois $x_i^2 = x_i$.

A QUBO é escrita desse jeito pois é o jeito mais fácil de escrever uma equação de grau 2 com todas as possibilidades de combinações entre as variáveis x_i (par ou isolada). Note que os termos constantes não interessam e podem ser ignorados, pois o objetivo é minimizar a função.

Para entender a QUBO mais intuitivamente, vamos considerar que o vetor binário x representa uma lista de lâmpadas que podemos ligar ou desligar. Nosso objetivo é descobrir a configuração que minimiza o custo (função objetivo).

Assim, a matriz Q é apenas uma lista de todos os custos associados a ligar quaisquer pares de lâmpadas. Os termos na diagonal se referem aos custos de ligar uma só lâmpada (termos lineares), enquanto os termos fora dela se referem aos custos de ligar pares de lâmpadas (termos quadráticos). O termo quadrático é nulo quando qualquer uma das lâmpadas do par correspondente está apagada.

Esse exemplo também ilustra o motivo da matriz Q ser simétrica em muitos problemas. Isso ocorre quando a ordem dos pares não influencia no custo, ou seja, em problemas comutativos. Nesses casos, o custo de ij é igual ao custo de ji ($Q_{ij} = Q_{ji}$).

Apesar do nome, pode-se adicionar restrições ao QUBO, desde que elas sejam incorporadas como termos quadráticos ou lineares na função objetivo. Isso torna desfavoráveis todas as soluções que violam as restrições.

Para adicionar uma restrição $g(x) = 0$, o seguinte termo deve ser somado a $f(x)$:

$$P \cdot g(x)^2,$$

onde $P \in \mathbb{R}^+$ é um parâmetro que demonstra a importância da restrição. Quanto maior P , mais as soluções que violam $g(x)$ serão punidas. Isso faz sentido porque, para zerar o termo $P \cdot g(x)^2$ com um P grande, é necessário diminuir muito $g(x)$, de forma a aproximá-lo de 0 (que é o objetivo).

3.4.6 Modelo Ising

O Modelo de Ising foi originalmente introduzido na Mecânica Estatística como um modelo simplificado para descrever fenômenos de magnetismo, no qual partículas interagem por meio de spins binários.

Ele tornou-se útil em otimização combinatória. Isso ocorre porque a energia do sistema é uma função quadrática de variáveis binárias, permitindo que problemas QUBO sejam mapeados para a minimização de um hamiltoniano de Ising.

Aqui, ele será útil posteriormente para fundamentar algoritmos quânticos de otimização (Quantum Annealing e QAOA).

Modelo de Ising

Um problema de otimização no **Modelo de Ising** consiste em determinar um vetor de spins

$$z = (z_1, z_2, \dots, z_n) \in \{-1, +1\}^n$$

que minimiza uma função de energia denominada **Hamiltoniano de Ising**

$$H : \{-1, +1\}^n \rightarrow \mathbb{R}$$

da forma

$$H(z) = -z^\top J z - h^\top z = -\sum_{i=1}^n \sum_{j=1}^n J_{ij} z_i z_j - \sum_{i=1}^n h_i z_i,$$

onde $J \in \mathbb{R}^{n \times n}$ é uma matriz real (tipicamente simétrica), que codifica os acoplamentos entre os spins, e $h \in \mathbb{R}^n$ representa o campo magnético externo aplicado a cada spin.

Proposição 14 (Equivalência QUBO-Ising) *Todo modelo QUBO é equivalente a um modelo de Ising e vice-versa.*

Prova

A conversão entre os modelos é estabelecida através da transformação linear $z_i = 1 - 2x_i$, que mapeia variáveis binárias $x_i \in \{0, 1\}$ do modelo QUBO para variáveis de spin $z_i \in \{-1, 1\}$ do modelo de Ising.

Substituindo a forma vetorial da transformação, $z = \vec{1} - 2x$, na função Hamiltoniana (ou de energia) do modelo de Ising com matriz de acoplamento J e campo magnético h , desenvolvemos a equivalência da seguinte forma:

$$\begin{aligned}
H(x) &= -(\vec{1} - 2x)^T J (\vec{1} - 2x) - h^T (\vec{1} - 2x) \\
&= -(\vec{1}^T J \vec{1} - 2\vec{1}^T J x - 2x^T J \vec{1} + 4x^T J x) - h^T \vec{1} + 2h^T x \\
&= -\vec{1}^T J \vec{1} + 4\vec{1}^T J x - 4x^T J x - h^T \vec{1} + 2h^T x \\
&\equiv (2\vec{1}^T J + h^T)x - 2x^T J x \\
&= x^T \text{diag}(2J\vec{1} + h)x - x^T (2J)x \\
&= x^T [\text{diag}(2J\vec{1} + h) - 2J]x \\
&= x^T Qx
\end{aligned}$$

Na terceira linha, agrupa-se os termos mistos explorando a simetria da matriz J .

Na quarta linha, o símbolo de equivalência (\equiv) indica o descarte dos termos constantes isolados ($-\vec{1}^T J \vec{1} - h^T \vec{1}$), pois adicionar uma constante a uma função objetivo não altera o ponto ótimo da otimização.

Na quinta linha, usa-se a propriedade $x_i^2 = x_i$ para reescrever o termo linear como uma forma quadrática com matriz diagonal.

Portanto, a energia do modelo de Ising pode ser perfeitamente mapeada na forma de um modelo QUBO $x^T Qx$, onde a matriz Q é definida por:

$$Q = \text{diag}(2J\vec{1} + h) - 2J$$

□

3.5 Aula 5 - Otimização Quântica

3.5.1 Evolução temporal de um qubit físico

Hamiltoniano de um sistema de dois níveis (qubit)

Considere um sistema quântico de dois níveis energéticos (spin 1/2, 2 níveis atômicos, etc) cujo estado fundamental $|1\rangle$ possui energia $-\frac{\hbar\omega}{2}$ e estado excitado $|0\rangle$ possui energia $\frac{\hbar\omega}{2}$. O hamiltoniano desse sistema é

$$H = \begin{pmatrix} \frac{\hbar\omega}{2} & 0 \\ 0 & -\frac{\hbar\omega}{2} \end{pmatrix} = \frac{\hbar\omega}{2} Z,$$

Note que os autovalores da hamiltoniana do qubit são

$$E_0 = \frac{\hbar\omega}{2}, \quad E_1 = -\frac{\hbar\omega}{2}.$$

E que os autoestados são a base computacional. Isso é esperado, pois são essas as energias que serão medidas de cada estado.

Recorde que, se o hamiltoniano é independente do tempo, a solução do sistema é

$$|\psi(t)\rangle = e^{-iHt/\hbar} |\psi(0)\rangle.$$

Além disso, diagonalizando o hamiltoniano,

$$H |n\rangle = E_n |n\rangle,$$

podemos escrever

$$|\psi(t)\rangle = \sum_n c_n(0) e^{-iE_n t/\hbar} |n\rangle,$$

onde $c_n(0)$ são constantes que dependem de qual é o estado inicial do qubit. Se o estado inicial for

$$|\psi(0)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

então

$$|\psi(t)\rangle = e^{-i\omega t/2} \frac{1}{\sqrt{2}} (|0\rangle + e^{i\omega t/2} |1\rangle).$$

A fase global (termo exponencial em evidência) é fisicamente irrelevante, mas a fase relativa determina a dinâmica observável. Portanto, pode-se escrever o estado como

$$|\psi(t)\rangle \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ e^{i\omega t} \end{pmatrix}$$

Essa evolução corresponde a uma rotação em torno do eixo z na esfera de Bloch:

$$R_z(\theta) = e^{-i\theta Z/2},$$

então as probabilidades do qubit nesse regime não mudam com o tempo.

3.5.2 Evolução adiabática

Se o hamiltoniano depende do tempo, $H = H(t)$, a solução não é simplesmente $e^{-iHt/\hbar}$.

Estado Fundamental

Seja H um Hamiltoniano atuando em um espaço de Hilbert \mathcal{H} , com espectro de autovalores

$$E_0 \leq E_1 \leq E_2 \leq \dots$$

O **estado fundamental** de H é qualquer autovetor $|\psi_0\rangle$ associado ao menor autovalor E_0 , isto é,

$$H |\psi_0\rangle = E_0 |\psi_0\rangle.$$

O valor E_0 é chamado de **energia fundamental** do sistema.

Se o menor autovalor possui mais de um autovetor linearmente independente, dizemos que o estado fundamental é **degenerado**.

Evolução Adiabática

Em uma **evolução adiabática**, um sistema quântico que começa no estado fundamental não degenerado de um hamiltoniano $H(t)$ transicionará para o estado fundamental do hamiltoniano $H(t + dt)$ após um intervalo temporal pequeno dt . Para isso acontecer, é preciso que o hamiltoniano mude suficientemente devagar, ou seja, que satisfaça a seguinte igualdade:

$$\frac{\max_{0 \leq t \leq T} \left| c_n(0) \hbar \left\langle k(t) \left| \frac{dH(t)}{dt} \right| n(t) \right\rangle \right|}{\min_{0 \leq t \leq T} |E_n(t) - E_k(t)|^2} \ll 1, \quad (n \neq k),$$

onde

- n (estado fundamental) e k são diferentes níveis energéticos do qubit;
- $H(t)$ é o hamiltoniano dependente do tempo;
- T é o tempo total de evolução;
- $|n(t)\rangle$ e $|k(t)\rangle$ são autovetores instantâneos de $H(t)$;
- $E_n(t)$ e $E_k(t)$ são os respectivos autovalores instantâneos.

A condição expressa que a taxa de variação do hamiltoniano deve ser pequena comparada ao quadrado do gap espectral mínimo entre os níveis n e k .

3.5.3 Computação Quântica Adiabática (CQA)

A Computação Quântica Adiabática traduz uma função de custo de um problema de otimização para o formalismo hamiltoniano e utiliza a evolução adiabática para minimizar essa função.

Neste modelo, resolvemos qualquer tipo de problema, porque é uma computação universal. Muitos problemas computacionais podem ser formulados como a minimização de alguma função de custo, descrita pelo hamiltoniano.

No CQA, varia-se o hamiltoniano adiabaticamente de forma que o sistema não transicione para estados não-fundamentais, embora o estado em que o próprio sistema se encontre seja alterado com o passar do tempo. Como o sistema sempre vai ficar próximo ao estado fundamental durante esse tempo, é vantajoso que a Hamiltoniana final do sistema seja a Hamiltoniana do problema a ser resolvido (H_P), pois assim o autovalor medido (que é o de menor energia) será igual ao valor mínimo da função de custo. Assim, é possível codificar a solução de um problema de otimização no estado final da evolução temporal.

Na CQA, resolvemos um problema interpolando entre dois hamiltonianos H_I e H_P . Note que o hamiltoniano total desse sistema deve satisfazer $H(0) = H_I$ e, quando o algoritmo acha a solução após um tempo T , tem-se $H(T) = H_P$. Uma maneira de alcançar isso é definir H a partir de uma função de scheduling s :

$$H(t) = (1 - s(t))H_I + s(t)H_P,$$

com

- $s(0) = 0$, $s(T) = 1$;
- H_I possui estado fundamental de fácil preparação;
- H_P codifica a solução do problema em seu estado fundamental.

Como a solução do problema foi codificada no autoestado de menor energia de H_P , o que temos que fazer é garantir que ao final da evolução o sistema esteja neste estado.

Uma maneira de alcançar esse objetivo é impondo que a evolução entre o estado inicial e final seja adiabática. Assim, se partirmos do estado de menor energia no início, garantiremos que ao final o sistema estará no estado de menor energia.

Usualmente, o hamiltoniano inicial é

$$H_I = - \sum_i X_i,$$

onde X_i representa a porta X aplicada ao qubit i . Já o estado inicial típico é o estado fundamental do H_I acima, ou seja,

$$|\psi(0)\rangle = |+\rangle^{\otimes n}.$$

É possível mudar o processo de transição do hamiltoniano inicial pro final mudando a função $s(t)$. Tipicamente,

$$s(t) = \frac{t}{T},$$

mas é muitas vezes vantajoso utilizar outras funções s . Por exemplo, para implementar o algoritmo de Grover em CQA, é necessário definir uma s bem extensa para garantir a complexidade $O(\sqrt{N})$.

3.5.4 Quantum Annealing

O quantum annealing (QA) é um processo de otimização para encontrar o mínimo global de uma determinada função objetivo em um dado conjunto de estados (soluções) candidatos, por meio de um processo que utiliza flutuações quânticas para atravessar barreiras de potencial e sair de mínimos locais. No modelo estudado aqui, que é o da empresa D-Wave, o QA é um modelo computacional dedicado, ou seja, não é universal.

Em geral, o QA é recomendado para aplicações em que o espaço de busca é discreto, como em problemas de otimização combinatória.

Este método é fortemente inspirado em um método clássico de otimização chamado Simulated Annealing. Neste caso, são utilizadas flutuações térmicas. Isso significa que a probabilidade da solução sair de um mínimo local flutua.

No QA, a função objetivo está codificada no hamiltoniano-problema H_P , cujo estado fundamental contém a solução para um problema de otimização e o sistema inicia em um estado arbitrário. A transição dos hamiltonianos ocorre por meio de um campo magnético externo e transversal ao spin dos qubits.

O hamiltoniano é escrito como

$$H(t) = H_P + \Gamma(t)H_D,$$

onde:

- H_P é o hamiltoniano problema;
- H_D é o hamiltoniano de campo transversal;
- $\Gamma(t)$ é o coeficiente de campo transversal, que decresce lentamente até zero.

Valores iniciais altos de $\Gamma(t)$ permitem mais tunelamento quântico e, conseqüentemente, mais escape de mínimos locais. Isso significa que a escolha da função Γ deve ser estratégica, para garantir que o sistema explore vários mínimos locais, mas também se estabilize num bom mínimo local. Esse tradeoff é interessante.

Note que o hamiltoniano do campo da QA é similar ao hamiltoniano inicial da CQA. Por conta disso, esses dois modelos são similares, mas é importante destacar que não são iguais.

Após a evolução do sistema por um tempo T , mede-se os qubits na base computacional, retornando uma solução na forma de uma bitstring, e escolhe-se a configuração de menor energia. Várias execuções são feitas, pois o algoritmo possui aleatoriedade.

Na prática, as flutuações quânticas sofrem interferência de flutuações térmicas, que podem afetar os resultados. Um annealer ideal só funcionaria a uma temperatura de 0 K. Essa temperatura é impossível de ser obtida em laboratório, por conta da Terceira Lei da Termodinâmica.

Para mitigar esses efeitos de decoerência, os qubits são resfriados a aproximadamente 15 mK, que é uma temperatura baixíssima. Por conta disso, a Criogenia é uma área de estudo importante para a Computação Quântica.

3.5.5 O Hamiltoniano Ising da DWave

Hamiltoniano de Ising (matricial)

Um Hamiltoniano de Ising matricial consiste em implementar o Hamiltoniano de Ising numérico a partir dos autovalores de um operador. Esse Hamiltoniano é

$$H_P = \sum_i h_i Z_i + \sum_{i,j;i \neq j} J_{ij} Z_i Z_j.$$

onde $J \in \mathbb{R}^{n \times n}$ é uma matriz real (tipicamente simétrica), que codifica os acoplamentos entre os spins, $h \in \mathbb{R}^n$ representa o campo magnético externo aplicado a cada spin e Z_i corresponde à porta Z sendo aplicada ao i -ésimo qubit.

Conforme já provado na Proposição 14, definindo a transformação

$$x_i = \frac{1 + z_i}{2}, \quad z_i \in \{-1, 1\},$$

podemos mapear o problema QUBO no modelo de Ising na computação clássica.

Na formulação quântica do problema, as variáveis clássicas deixam de ser números e passam a ser representadas por operadores quânticos. A ideia é que os valores possíveis da variável binária sejam obtidos como autovalores desses operadores quando medidos.

O operador Z possui autovalores $\{+1, -1\}$, que correspondem aos valores possíveis das variáveis de spin do modelo de Ising $z_i \in \{-1, 1\}$.

Queremos, porém, representar variáveis binárias do tipo $x_i \in \{0, 1\}$.

Para isso, construímos um operador cujo conjunto de autovalores seja $\{0, 1\}$. Esse operador é definido por

$$\hat{x}_i = \frac{I - Z_i}{2},$$

onde I é o operador identidade.

De fato, aplicando esse operador nos estados da base computacional, obtemos

$$Z_i |0\rangle = +|0\rangle, \quad Z_i |1\rangle = -|1\rangle.$$

Assim,

$$\hat{x}_i |0\rangle = \frac{1 - 1}{2} |0\rangle = 0 |0\rangle,$$

e

$$\hat{x}_i |1\rangle = \frac{1 - (-1)}{2} |1\rangle = 1 |1\rangle.$$

Portanto, os autovalores do operador \hat{x}_i são exatamente

$$\{0, 1\},$$

reproduzindo os valores possíveis da variável binária clássica.

Dessa forma, ao substituir cada variável x_i pelo operador \hat{x}_i , a função custo clássica pode ser promovida a um hamiltoniano quântico. O estado fundamental desse hamiltoniano corresponde então à solução ótima do problema de otimização.

Assim, problemas de otimização combinatória podem ser implementados em hardware de QA. Nos annealers da D-Wave, o hamiltoniano final é do tipo Ising:

$$H_P = \sum_i h_i Z_i + \sum_{i,j;i \neq j} J_{ij} Z_i Z_j.$$

Este é o motivo dos annealers da DWave serem um modelo computacional não-universal: o seu hamiltoniano é restrito para a forma Ising.

Proposição 15 (Autoestados do Hamiltoniano de Ising) *Considere o Hamiltoniano de Ising atuando em n qubits,*

$$H_P = \sum_i h_i Z_i + \sum_{i < j} J_{ij} Z_i Z_j,$$

onde $h_i, J_{ij} \in \mathbb{R}$ e Z_i é o operador de Pauli-Z atuando no qubit i .

Então os estados da base computacional

$$|x\rangle = |x_1, x_2, \dots, x_n\rangle, \quad x_i \in \{0, 1\},$$

são autovetores de H_P . O autovalor correspondente é

$$E(x) = \sum_i h_i z_i + \sum_{i < j} J_{ij} z_i z_j,$$

onde

$$z_i = \begin{cases} +1 & \text{se } x_i = 0, \\ -1 & \text{se } x_i = 1. \end{cases}$$

Assim,

$$H_P |x\rangle = E(x) |x\rangle.$$

Prova

Sabemos que os autovalores do operador de Pauli-Z são

$$Z|0\rangle = +|0\rangle, \quad Z|1\rangle = -|1\rangle.$$

Portanto, para um estado da base computacional

$$|x\rangle = |x_1, x_2, \dots, x_n\rangle,$$

temos

$$Z_i|x\rangle = z_i|x\rangle,$$

onde $z_i \in \{-1, 1\}$ é definido por

$$z_i = \begin{cases} +1 & \text{se } x_i = 0, \\ -1 & \text{se } x_i = 1. \end{cases}$$

Como os operadores Z_i e Z_j atuam em qubits distintos, segue que

$$Z_i Z_j |x\rangle = z_i z_j |x\rangle.$$

Aplicando o Hamiltoniano ao estado $|x\rangle$, obtemos

$$H_P|x\rangle = \left(\sum_i h_i Z_i + \sum_{i<j} J_{ij} Z_i Z_j \right) |x\rangle.$$

Substituindo os autovalores dos operadores de Pauli,

$$H_P|x\rangle = \left(\sum_i h_i z_i + \sum_{i<j} J_{ij} z_i z_j \right) |x\rangle.$$

Definindo

$$E(x) = \sum_i h_i z_i + \sum_{i<j} J_{ij} z_i z_j,$$

segue que

$$H_P|x\rangle = E(x)|x\rangle.$$

Logo, os estados da base computacional são autovetores de H_P , com autovalores dados pela função de energia clássica do modelo de Ising.

A proposição acima prova que, ao fazer um sistema de qubits evoluir para o Hamiltoniano de Ising, a medição desse sistema retorna o valor da função de custo correspondente à bitstring medida.

3.5.6 Comparação

	Computação Quântica Circuital	Computação Quântica Adiabática	Quantum Annealing
Tipo de evolução temporal	Discreta	Contínua, ou contínua na maior parte do tempo	Contínua, ou contínua na maior parte do tempo
Universalidade	Modelo universal	Modelo universal	Modelo dedicado
Medida de complexidade computacional	Número de portas lógicas e profundidade do circuito	Depende do gap de energia, mas em geral não sabemos calculá-lo	Depende do gap de energia, mas em geral não sabemos calculá-lo

3.6 Aula 6 - Otimização Quântica II

3.6.1 Algoritmo de Otimização Aproximada Quântica (QAOA)

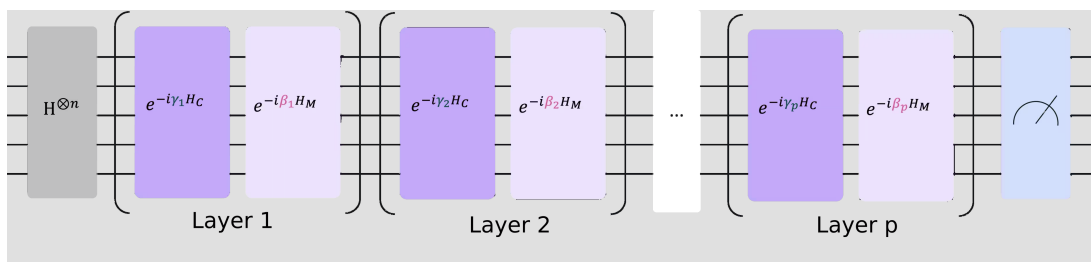


Figura 13: Circuito que implementa a parte quântica do QAOA. Fonte: <https://quantum.cloud.ibm.com/docs/en/tutorials/quantum-approximate-optimization-algorithm>.

O Quantum Approximate Optimization Algorithm (QAOA) é um algoritmo variacional projetado para resolver problemas de otimização combinatória. A ideia central é utilizar um circuito quântico parametrizado para aproximar o estado que minimiza uma função custo associada ao problema.

O QAOA consiste em:

1. Codificar o problema em um Hamiltoniano de custo H_C .

2. Preparar um Hamiltoniano mixer H_M
3. Preparar um estado inicial uniforme.
4. Aplicar camadas parametrizadas de evolução sob H_C e H_M .
5. Medir o valor esperado da função custo.
6. Usar um otimizador clássico para atualizar os parâmetros.]
7. Repetir até achar uma solução suficientemente boa.

O resultado final é um estado quântico cuja medição produz com alta probabilidade uma boa solução para o problema de otimização.

3.6.2 Enunciado

Considere um problema de otimização combinatória onde desejamos minimizar uma função custo

$$C(x_1, x_2, \dots, x_n),$$

onde cada variável $x_i \in \{0, 1\}$.

O objetivo é encontrar a bitstring x que minimiza o valor da função custo.

Esses problemas podem ser frequentemente escritos na forma de um Hamiltoniano de custo H_C - como o Ising - tal que o valor da função custo corresponde ao valor esperado desse Hamiltoniano em um estado computacional.

Assim,

$$H_C |x\rangle = C(x) |x\rangle.$$

Logo, o problema de otimização se torna equivalente a encontrar o estado fundamental do Hamiltoniano de custo.

3.6.3 Ideia do QAOA

Recorde a Proposição 2. Segundo ela, qualquer Hamiltoniano pode ser decomposto em uma soma de produtos tensoriais das matrizes:

$$H = \sum_k c_k P_k = \sum_k H_k, \tag{19}$$

onde

- $c_k \in \mathbb{R}$ é o k -ésimo coeficiente;

- P_k é uma Pauli String, ou seja, um produto tensorial das matrizes do conjunto $\{I, X, Y, Z\}$.

Isso permite a criação de circuitos quânticos que implementam o hamiltoniano. Como fazer isso?

Para isso, implementamos o operador de evolução temporal

$$U(\theta) = e^{-i\theta H}. \quad (20)$$

A forma de construir o circuito quântico depende de uma propriedade fundamental dos termos do Hamiltoniano: se eles comutam ou não entre si.

Caso 1: termos que comutam

Se todos os termos do Hamiltoniano comutam entre si,

$$[H_k, H_j] = 0 \quad \text{para todo } k, j,$$

então vale a propriedade da exponencial matricial

$$e^{A+B} = e^A e^B.$$

Logo,

$$e^{-i\theta H} = e^{-i\theta \sum_k H_k} = \prod_k e^{-i\theta H_k}. \quad (21)$$

Nesse caso, a evolução total pode ser implementada aplicando sequencialmente circuitos que realizam cada operador

$$e^{-i\theta H_k}.$$

Como cada H_k é uma Pauli string, esses operadores correspondem a rotações controladas na base de Pauli, que podem ser implementadas com portas elementares (rotações e CNOTs).

Caso 2: termos que não comutam

Se os termos do Hamiltoniano não comutam,

$$[H_k, H_j] \neq 0,$$

então, em geral,

$$e^{A+B} \neq e^A e^B.$$

Nesse caso, a evolução temporal não pode ser decomposta exatamente como um produto de exponenciais simples. Uma estratégia comum é usar a **aproximação de Trotter**, que aproxima a evolução como

$$e^{-i\theta H} = e^{-i\theta \sum_k H_k} \approx \left(\prod_k e^{-i\theta H_k/r} \right)^r, \quad (22)$$

onde r é o número de passos de Trotter.

Quanto maior for r , melhor será a aproximação da evolução ideal. Assim, mesmo quando os termos do Hamiltoniano não comutam, é possível construir circuitos quânticos que aproximam a evolução temporal do sistema.

Com isso, a ideia do QAOA é usar dois operadores unitários com um hamiltoniano cada. Precisamos de um operador que muda as fases dos autoestados x_i com base nos valores $C(x_i)$. Além disso, precisamos de um operador que "mistura" as probabilidades dos autoestados, de forma a explorar mais o espaço de soluções.

O QAOA constrói um estado quântico parametrizado com duas variáveis aplicando alternadamente esses dois tipos de evolução unitária p vezes:

$$U_C(\gamma) = e^{-i\gamma H_C}$$

e

$$U_M(\beta) = e^{-i\beta H_M}.$$

Aqui:

- H_C é o **Hamiltoniano de custo**, que codifica o problema;
- H_M é o **Hamiltoniano misturador**;
- γ e β são parâmetros que podem ser variados pelo otimizador clássico após o circuito quântico;
- p é o número de camadas contendo o operador $U_M U_C$.

O Hamiltoniano misturador mais comum é

$$H_M = \sum_{i=1}^n X_i,$$

onde X_i é o operador de Pauli-X atuando no qubit i .

Esse Hamiltoniano promove transições entre estados da base computacional, permitindo que o algoritmo explore diferentes configurações.

O operador unitário referente a H_M é

$$U_M(\beta) = e^{-i\beta H_M} = \prod_k (R_x(2\beta))_k.$$

Dessa forma, é aplicada a porta $R_X(2\beta)$ em cada qubit.

3.6.4 Circuito e evolução

O algoritmo começa preparando o estado uniforme

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle = H^{\otimes n} |0\rangle^{\otimes n}.$$

Esse estado representa uma superposição de todas as soluções possíveis do problema.

O estado final do QAOA com profundidade p é

$$|\psi(\gamma, \beta)\rangle = U_M(\beta_p)U_C(\gamma_p) \dots U_M(\beta_1)U_C(\gamma_1) |s\rangle.$$

Os parâmetros são

$$\gamma = (\gamma_1, \dots, \gamma_p), \quad \beta = (\beta_1, \dots, \beta_p).$$

Cada par (γ_k, β_k) constitui uma camada do circuito.

Note que o estado final é totalmente dependente dessas duas variáveis vetoriais. O pulo do gato é este: podemos otimizar classicamente e encontrar (γ, β) que entregam um estado ψ onde a probabilidade de medição de boas bitstrings é alta!

Após preparar o estado $|\psi\rangle$, calculamos o valor esperado do Hamiltoniano de custo:

$$F(\psi) = \langle \psi | H_C | \psi \rangle.$$

Esse valor pode ser estimado através de medições no computador quântico.

O objetivo do algoritmo de otimização clássica é resolver

$$\min_{\gamma, \beta} F(\psi(\gamma, \beta)).$$

.

Esse esquema permite explorar o espaço de soluções usando um circuito quântico enquanto a busca de parâmetros é feita por um algoritmo clássico.

3.6.5 Relação com CQA

O QAOA pode ser interpretado como uma versão discretizada da CQA.

Na computação adiabática, o sistema evolui lentamente de um Hamiltoniano simples para o Hamiltoniano do problema H_C .

No QAOA, essa evolução é aproximada por uma sequência alternada de operadores:

$$e^{-i\gamma H_C} \quad e^{-i\beta H_M}.$$

À medida que o número de camadas p aumenta, o algoritmo pode aproximar cada vez melhor a evolução adiabática ideal.

3.7 Aula 7 - Otimização Quântica III

3.7.1 Princípio Variacional

Teorema do Princípio Variacional

Seja H um hamiltoniano que atua em um espaço de Hilbert e seja E_0 o menor autovalor de H , associado ao estado fundamental $|\psi_0\rangle$.

Então, para qualquer estado normalizado $|\psi\rangle$ não-degenerado, vale

$$E_0 \leq \langle \psi | H | \psi \rangle.$$

Em outras palavras, o valor esperado da energia calculado em qualquer estado de teste fornece um limite superior para a energia do estado fundamental. A igualdade ocorre se, e somente se, $|\psi\rangle$ coincide com o estado fundamental $|\psi_0\rangle$ (até uma fase global).

Prova

Seja $\{|\psi_n\rangle\}_{n=0}^{\infty}$ uma base ortonormal de autovetores do hamiltoniano H , tal que

$$H|\psi_n\rangle = E_n|\psi_n\rangle,$$

com os autovalores ordenados como

$$E_0 \leq E_1 \leq E_2 \leq \dots$$

Considere um estado arbitrário normalizado $|\psi\rangle$. Como os autovetores formam uma base completa, podemos escrever

$$|\psi\rangle = \sum_n c_n |\psi_n\rangle,$$

onde $c_n \in \mathbb{C}$ e, pela normalização,

$$\sum_n |c_n|^2 = 1.$$

O valor esperado da energia nesse estado é

$$\langle \psi | H | \psi \rangle = \left(\sum_m c_m^* \langle \psi_m | \right) H \left(\sum_n c_n | \psi_n \rangle \right).$$

Usando a relação de autovalor $H | \psi_n \rangle = E_n | \psi_n \rangle$, obtemos

$$\langle \psi | H | \psi \rangle = \sum_{m,n} c_m^* c_n E_n \langle \psi_m | \psi_n \rangle.$$

Como os autovetores são ortonormais, $\langle \psi_m | \psi_n \rangle = \delta_{mn}$, então só sobram os termos onde $m = n$:

$$\langle \psi | H | \psi \rangle = \sum_n |c_n|^2 E_n.$$

Como E_0 é o menor autovalor, temos $E_n \geq E_0$ para todo n . Logo,

$$\langle \psi | H | \psi \rangle = \sum_n |c_n|^2 E_n \geq \sum_n |c_n|^2 E_0.$$

Fatorando E_0 ,

$$\langle \psi | H | \psi \rangle \geq E_0 \sum_n |c_n|^2.$$

Pela normalização do estado, $\sum_n |c_n|^2 = 1$, portanto

$$E_0 \leq \langle \psi | H | \psi \rangle.$$

Isso prova que o valor esperado da energia em qualquer estado normalizado fornece um limite superior para a energia do estado fundamental.

A igualdade ocorre somente quando $c_n = 0$ para todo $n \neq 0$, isto é, quando $|\psi\rangle$ coincide com o estado fundamental $|\psi_0\rangle$ (até uma fase global).

□

3.7.2 VQE

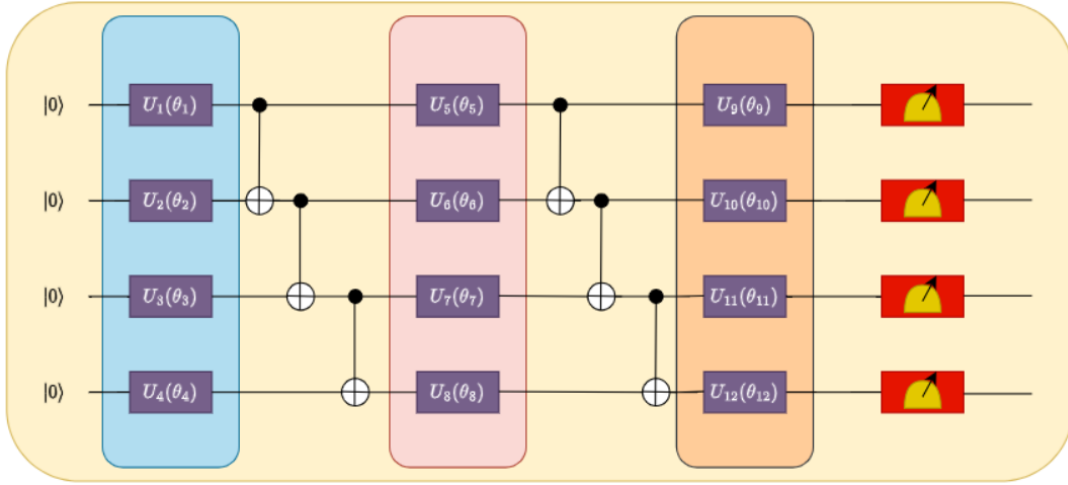


Figura 14: Exemplo de circuito que implementa a parte quântica do VQE. Fonte: N. Innan, M. A. Z. Khan and M. Bennai, arXiv:2305.07902 (2023).

O Variational Quantum Eigensolver (VQE) é um algoritmo híbrido quântico-clássico projetado para aproximar o autovalor mínimo (energia fundamental) de um Hamiltoniano. O método é particularmente adequado para dispositivos quânticos atuais, pois utiliza circuitos relativamente curtos e transfere parte significativa do esforço computacional para um otimizador clássico.

O VQE foi inicialmente proposto para problemas de Química Quântica, como a estimativa de energias de moléculas.

3.7.3 Ideia do VQE

O VQE baseia-se no princípio variacional da mecânica quântica:

$$E_0 \leq \langle \psi | H | \psi \rangle,$$

onde H é o Hamiltoniano do sistema e E_0 é sua energia fundamental.

Portanto, se conseguirmos preparar estados quânticos parametrizados $|\psi(\boldsymbol{\theta})\rangle$ e minimizar o valor esperado

$$E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle,$$

então o menor valor obtido fornece uma aproximação para a energia fundamental do sistema.

O VQE utiliza um ciclo híbrido composto por três etapas principais.

1. Preparação do estado parametrizado

Escolhe-se um circuito quântico parametrizado (ansatz)

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle^{\otimes n},$$

onde $U(\boldsymbol{\theta})$ é um circuito dependente de parâmetros clássicos $\boldsymbol{\theta} = (\theta_1, \dots, \theta_m)$.

Esse circuito define a família de estados que o algoritmo pode explorar.

2. Medição do valor esperado do Hamiltoniano

O Hamiltoniano do problema é decomposto como soma de operadores de Pauli

$$H = \sum_i c_i P_i,$$

onde P_i são Pauli Strings e c_i são coeficientes reais.

Assim,

$$E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle = \sum_i c_i \langle \psi(\boldsymbol{\theta}) | P_i | \psi(\boldsymbol{\theta}) \rangle = \sum_i c_i \langle P_i \rangle_{\boldsymbol{\theta}}.$$

Cada termo $\langle P_i \rangle$ é estimado experimentalmente através da de média aritmética das medições no circuito quântico.

3. Otimização clássica

Um algoritmo clássico, como o COBYLA, ajusta os parâmetros $\boldsymbol{\theta}$ para minimizar $E(\boldsymbol{\theta})$.

Após atualizar os parâmetros, o circuito quântico é executado novamente até a média da energia (estimação do valor esperado) ser pequena o suficiente, formando um ciclo iterativo. Pelo princípio variacional, quanto menor esse valor esperado for, mais próximo da energia fundamental será.

3.7.4 Circuito e Evolução

O circuito do VQE consiste em um ansatz parametrizado, ou seja, uma evolução unitária da forma

$$|\psi(\boldsymbol{\theta})\rangle = U_L(\theta_L) \cdots U_2(\theta_2) U_1(\theta_1) |0\rangle^{\otimes n},$$

onde cada operador $U_k(\theta_k)$ representa uma porta quântica dependente de um parâmetro clássico.

Na prática, essas portas costumam ser rotações em torno dos eixos X , Y ou Z , como

$$R_X(\theta), \quad R_Y(\theta), \quad R_Z(\theta),$$

combinadas com portas de dois qubits que geram entrelaçamento, como a CNOT.

O VQE pode ser interpretado como uma busca variacional dentro de uma família de estados gerada pelo ansatz. O algoritmo tenta encontrar parâmetros que aproximem o estado fundamental do Hamiltoniano.

Se o ansatz for suficientemente expressivo, existe um conjunto de parâmetros para o qual

$$|\psi(\boldsymbol{\theta})\rangle \approx |\psi_0\rangle,$$

onde $|\psi_0\rangle$ é o estado fundamental.

3.7.5 Exemplos Ansatz

No algoritmo VQE, a escolha do ansatz é crucial, pois determina o espaço de estados explorado durante a otimização.

Alguns ansatz comuns são:

Hardware-Efficient Ansatz (HEA)

Consiste em camadas alternadas de portas de rotações e CNOT:

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta}) |0\rangle^{\otimes n},$$

onde

$$U(\boldsymbol{\theta}) = \prod_{l=1}^L U_{cnot}^{(l)} U_{rot}^{(l)}(\theta_i).$$

É amplamente utilizado por ser fácil de implementar em dispositivos reais, embora nem sempre incorpore a estrutura física do problema.

Dicke State Ansatz

Esse ansatz utiliza os chamados estados de Dicke, que são estados quânticos simétricos com um número fixo de excitações. Para n qubits e k excitações, o estado de Dicke é definido como

$$|D_n^{(k)}\rangle = \frac{1}{\sqrt{\binom{n}{k}}} \sum_{\text{permutações}} |1^k 0^{n-k}\rangle,$$

ou seja, uma superposição uniforme de todos os estados computacionais com exatamente k qubits no estado $|1\rangle$.

Com isso, o estado final é

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta}) |D_n^{(k)}\rangle.$$

Esse tipo de ansatz é útil quando o problema possui simetrias ou restrições que preservam o número de excitações.

QAOA

O QAOA também pode ser interpretado como um ansatz variacional. Nesse caso, o estado é preparado por uma sequência alternada de evoluções unitárias geradas por dois Hamiltonianos:

$$|\psi(\gamma, \beta)\rangle = \prod_{l=1}^p e^{-i\beta_l H_M} e^{-i\gamma_l H_C} |+\rangle^{\otimes n},$$

onde H_C é o Hamiltoniano de custo do problema, H_M é um Hamiltoniano misturador e (γ_l, β_l) são parâmetros variacionais.

3.7.6 FALQON

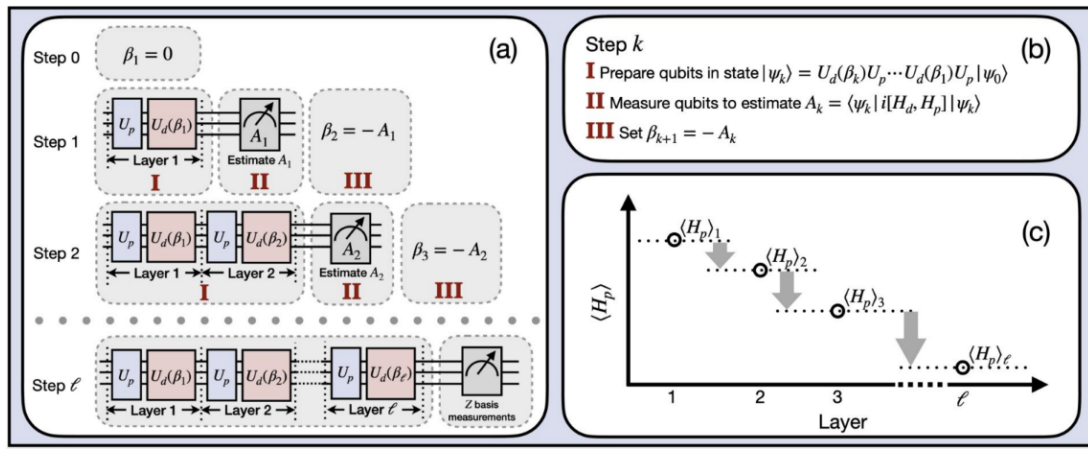


Figura 15: Diagrama mostrando a implementação do FALQON. Fonte: A. B. Magann, et al. Phys. Rev. Lett. 129, 250502 (2022).

O Feedback-based Algorithm for Quantum Optimization (FALQON) é um algoritmo variacional inspirado na estrutura do QAOA, mas que elimina a necessidade de um otimizador clássico externo. Em vez disso, os parâmetros do circuito são determinados iterativamente por uma regra de *feedback* baseada em medições feitas no próprio circuito quântico.

3.7.7 Ideia do FALQON

Seja uma Hamiltoniana

$$H(t) = H_P + \beta(t)H_D,$$

onde H_P é a hamiltoniana de custo do problema e H_D é chamada Hamiltoniana driver, cuja contribuição é modulada pela variável $\beta(t)$. O objetivo é encontrar $\beta(t)$ que minimiza a energia de um sistema com estado $|\psi(t)\rangle$ após um tempo T .

Para isso, definimos a função

$$J(t) := \langle \psi(t) | H_P | \psi(t) \rangle,$$

que representa o valor esperado do Hamiltoniano de custo H_p no estado do sistema no instante t . Como o objetivo do problema de otimização é minimizar a energia associada a H_p , minimizar $J(t)$ equivale a aproximar o estado fundamental desse Hamiltoniano.

Proposição 16 (Caso constante do Teorema de Ehrenfest) *Seja $|\psi(t)\rangle$ um estado quântico que evolui segundo a equação de Schrödinger*

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle,$$

e seja O um operador que não depende explicitamente do tempo. Então a derivada temporal do valor esperado de O é dada por

$$\frac{d}{dt}\langle O \rangle = i\langle [H(t), O] \rangle,$$

onde $[H(t), O] = H(t)O - OH(t)$ é o comutador entre $H(t)$ e O .

Para estudar como $J(t)$ evolui no tempo, calculamos sua derivada temporal usando a Proposição 16:

$$\frac{d}{dt}J(t) = \langle \psi(t) | i[H(t), H_p] | \psi(t) \rangle.$$

Substituindo a expressão de $H(t)$ no comutador, temos

$$[H(t), H_p] = [H_p + \beta(t)H_d, H_p].$$

Usando a linearidade do comutador, segue que

$$[H_p + \beta(t)H_d, H_p] = [H_p, H_p] + \beta(t)[H_d, H_p].$$

Como qualquer operador comuta consigo mesmo, $[H_p, H_p] = 0$. Portanto,

$$[H(t), H_p] = \beta(t)[H_d, H_p].$$

Substituindo isso na expressão da derivada, obtemos

$$\frac{d}{dt}J(t) = \langle \psi(t) | i[H_d, H_p] | \psi(t) \rangle \beta(t).$$

Definindo o termo

$$A(t) := \langle \psi(t) | i[H_d, H_p] | \psi(t) \rangle,$$

tem-se

$$\frac{d}{dt}J(t) = A(t)\beta(t).$$

Escolhendo uma lei para β que depende de um parâmetro $\eta \in \mathbb{R}^+$ (denotando a intensidade de atualização de $J(t)$),

$$\beta(t) = -\eta A(t)$$

também tem-se

$$\frac{d}{dt}J(t) = -\eta(A(t))^2 < 0.$$

Portanto, garantimos que $J(t)$ decresce monotonicamente ao longo do tempo.

O procedimento completo do FALQON pode ser resumido da seguinte forma:

1. Inicializar o sistema no estado $|\psi_0\rangle = |+\rangle^{\otimes n}$.
2. Aplicar a evolução $e^{-i\Delta t H_C}$;
3. Medir o valor esperado $\langle i[H_C, H_M] \rangle$;
4. Calcular o parâmetro β usando a regra de feedback;
5. Aplicar a evolução $e^{-i\beta H_M}$;
6. Repetir o processo para construir novas camadas do circuito;
7. Parar quando o valor esperado A estiver pequeno o suficiente.

3.7.8 Circuito e evolução

Na prática, a evolução contínua descrita pela equação de Schrödinger é aproximada por um circuito quântico variacional composto por p camadas, de forma análoga ao QAOA. No entanto, enquanto no QAOA os ângulos são otimizados globalmente, no FALQON eles são determinados camada por camada.

A evolução do estado quântico ocorre por uma sequência de operadores unitários da forma

$$|\psi_p\rangle = \prod_{k=1}^p e^{-i\beta_k H_d} e^{-i\Delta t H_P} |\psi_0\rangle,$$

onde $|\psi_0\rangle$ é um estado inicial simples (geralmente $|+\rangle^{\otimes n}$), Δt é um pequeno passo de evolução associado ao Hamiltoniano de custo e β_k é um parâmetro que depende de β_{k-1} .

Seja o estado na camada k dado por:

$$|\psi_k\rangle = e^{-i\Delta t \beta_k H_d} e^{-i\Delta t H_P} |\psi_{k-1}\rangle,$$

onde Δt é um passo de tempo pequeno e fixo.

Após preparar o estado $|\psi_k\rangle$, mede-se o valor esperado experimentalmente

$$A_k = \langle \psi_k | i[H_C, H_M] | \psi_k \rangle.$$

O próximo parâmetro β do circuito é, então, definido pela regra iterativa

$$\beta_{k+1} = -\eta A_k,$$

Assim, o algoritmo produz uma sequência de estados com energia cada vez menor, aproximando progressivamente o estado fundamental do Hamiltoniano de custo.

A principal vantagem do FALQON é eliminar a etapa de otimização clássica, que frequentemente constitui o principal gargalo em algoritmos variacionais. Como os parâmetros são determinados diretamente a partir das medições quânticas, o algoritmo pode apresentar maior estabilidade e escalabilidade em comparação com métodos variacionais tradicionais.

Por outro lado, o desempenho do FALQON depende da escolha do passo Δt , do parâmetro de feedback η e da estrutura dos Hamiltonianos envolvidos.

3.7.9 O parâmetro Δt

No algoritmo FALQON, o parâmetro Δt controla o passo efetivo da evolução gerada pelo Hamiltoniano de custo. Em outras palavras, ele determina o quanto o estado quântico evolui em cada iteração do circuito.

Existe um limite superior para esse parâmetro que garante a estabilidade do algoritmo:

$$\Delta t \leq \frac{1}{4\|H_p\| \|H_d\|^2},$$

onde $\|H_p\|$ e $\|H_d\|$ denotam as normas dos Hamiltonianos numéricos de custo e driver, respectivamente.

Essa condição estabelece uma escala máxima para o passo temporal do algoritmo. Intuitivamente, se Δt for escolhido muito grande, a evolução aplicada pelo circuito pode ser excessiva, fazendo com que o sistema ultrapasse a região onde a aproximação utilizada na dedução do método é válida.

Quando essa condição não é respeitada, o comportamento do algoritmo pode se tornar instável. Em vez de convergir gradualmente para estados com menor energia, o valor da função objetivo $J(t)$ pode passar a oscilar ao longo das iterações. Como consequência, o algoritmo perde sua propriedade de convergência monotônica e pode deixar de aproximar corretamente o estado fundamental do Hamiltoniano de custo.

3.7.10 Qubit-Wise Commuting (QWC)

No algoritmos VQE e FALQON, é necessário medir os valores esperados de Pauli Strings $\langle P_i \rangle$. Entretanto, como diferentes operadores de Pauli podem requerer diferentes bases

de medição, medir cada termo separadamente pode demandar um grande número de execuções do circuito quântico.

Uma técnica importante para reduzir esse custo experimental é o Qubit-Wise Commuting (QWC).

Qubit-Wise Commuting (QWC)

Dizemos que dois operadores de Pauli P_i e P_j são **qubit-wise commuting** se, para cada qubit k , os operadores locais correspondentes comutam. Em outras palavras, escrevendo

$$P_i = \bigotimes_{k=1}^n \sigma_k^{(i)}, \quad P_j = \bigotimes_{k=1}^n \sigma_k^{(j)},$$

com $\sigma_k^{(i)}, \sigma_k^{(j)} \in \{I, X, Y, Z\}$, dizemos que P_i e P_j são QWC se

$$[\sigma_k^{(i)}, \sigma_k^{(j)}] = 0 \quad \forall k.$$

Quando dois operadores satisfazem essa propriedade, eles podem ser medidos simultaneamente na mesma base local. Assim, é possível agrupar vários termos do Hamiltoniano em um mesmo conjunto de medições.

Por exemplo, considere os operadores

$$P_1 = Z_1 Z_2 = Z \otimes Z, \quad P_2 = Z_1 I_2 = Z \otimes I.$$

Esses operadores são QWC, pois em cada qubit os operadores locais comutam. Portanto, ambos podem ser medidos utilizando a mesma base Z .

Por outro lado,

$$P_1 = Z_1 Z_2 = Z \otimes Z, \quad P_3 = X_1 Z_2 = X \otimes Z$$

não são QWC, pois Z e X não comutam no primeiro qubit. Nesse caso, medições separadas são necessárias.

Na prática, algoritmos de agrupamento são utilizados para particionar os termos do Hamiltoniano em conjuntos QWC. Cada conjunto pode ser medido com a mesma configuração de base, reduzindo significativamente o número total de execuções do circuito necessárias para estimar a energia do problema de otimização.

3.8 Aula 8 - Prática de Otimização Quântica

3.8.1 Link do notebook do Google Colab

<https://colab.research.google.com/drive/1TfWNUmV9PmLRdxW1yiNwMZL-taK2H6Wx?usp=sharing>

4 Bloco 03 – Machine Learning Clássico e Quântico

4.1 Aula 1 - Introdução à Aprendizagem de Máquina Clássica

4.1.1 Aprendizagem de Máquina

Inteligência Artificial (IA) é o ramo da Ciência da Computação que estuda técnicas computacionais que imitam a inteligência humana para resolver problemas.

Em especial, estudaremos a IA conexionista, que consiste em usar a capacidade de processamento de um computador para simular estruturas básicas, cuja ação conjunta resulta em um comportamento inteligente.

Aprendizado de Máquina

Diz-se que um programa de computador aprende a partir de uma **experiência** E com respeito a uma classe de **tarefas** T e uma medida de **desempenho** P , se o seu desempenho em tarefas de T , medido por P , melhora com a experiência E . (Tom Mitchell, 1997)

Regressão

Regressão é o problema de aprendizado supervisionado no qual o objetivo é aprender uma função $f : \mathcal{X} \rightarrow \mathbb{R}$ que mapeia entradas em valores reais contínuos, a partir de um conjunto de dados rotulados (x_i, y_i) , onde $y_i \in \mathbb{R}$.

Classificação Supervisionada

Classificação supervisionada é o problema de aprendizado supervisionado no qual o objetivo é aprender uma função $f : \mathcal{X} \rightarrow \mathcal{Y}$ que associa cada entrada a uma classe discreta, a partir de um conjunto de dados rotulados (x_i, y_i) , onde $y_i \in \mathcal{Y}$ e \mathcal{Y} é um conjunto finito de rótulos.

Classificação Não-Supervisionada

Classificação não-supervisionada (ou **agrupamento**) é o problema de aprendizado no qual, dado um conjunto de dados não rotulados $\{x_i\}$, busca-se identificar estruturas, padrões ou agrupamentos intrínsecos nos dados, particionando-os em subconjuntos de acordo com alguma noção de similaridade.

4.1.2 k-Nearest Neighbors

k-Nearest Neighbors (KNN)

Seja $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ um conjunto de dados rotulados, com $x_i \in \mathcal{X}$ e $y_i \in \mathcal{Y}$, e seja $d: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ uma função de distância.

Dado um novo ponto $x \in \mathcal{X}$, o método dos k **vizinhos mais próximos** consiste em:

- Determinar o conjunto $N_k(x) \subset \mathcal{D}$ formado pelos k pontos de \mathcal{D} mais próximos de x , segundo a métrica d ;
- No caso de **classificação**, prever o rótulo por maioria:

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \sum_{(x_i, y_i) \in N_k(x)} \mathbf{1}(y_i = y);$$

- No caso de **regressão**, prever o valor médio:

$$\hat{y} = \frac{1}{k} \sum_{(x_i, y_i) \in N_k(x)} y_i.$$

Métricas de Distância

Tipo	Equação
Distância de Manhattan	$d(x, y) = \sum_{i=1}^n x_i - y_i $
Distância de Minkowski	$d(x, y) = (\sum_{i=1}^n x_i - y_i ^p)^{1/p}$
Distância Euclidiana ($p = 2$)	$d(x, y) = (\sum_{i=1}^n x_i - y_i ^2)^{1/2}$

4.1.3 Árvore de Decisão

Árvore de Decisão

Uma **árvore de decisão** é um modelo de aprendizado supervisionado que representa uma função $f : \mathcal{X} \rightarrow \mathcal{Y}$ por meio de uma estrutura de árvore direcionada, onde:

- Cada **nó interno** está associado a um teste sobre uma característica da entrada $x \in \mathcal{X}$ (por exemplo, $x_j \leq t$);
- Cada **aresta** corresponde ao resultado possível desse teste;
- Cada **folha** está associada a uma predição $\hat{y} \in \mathcal{Y}$ (no caso de classificação) ou $\hat{y} \in \mathbb{R}$ (no caso de regressão).

A predição para uma entrada x é obtida percorrendo a árvore desde a raiz até uma folha, seguindo os testes definidos nos nós internos.

A árvore particiona o espaço de entrada \mathcal{X} em regiões disjuntas $\{R_m\}_{m=1}^M$, tais que

$$\mathcal{X} = \bigcup_{m=1}^M R_m, \quad R_i \cap R_j = \emptyset \text{ para } i \neq j,$$

e define uma função por partes

$$f(x) = c_m \quad \text{se } x \in R_m,$$

onde c_m é uma constante associada à região R_m .

A construção de uma árvore de decisão é realizada de forma recursiva, particionando o espaço de entrada \mathcal{X} com base em critérios que maximizam a “pureza” dos subconjuntos gerados.

Seja \mathcal{D} o conjunto de dados em um nó. Para cada possível divisão (split) s , particiona-se \mathcal{D} em dois subconjuntos:

$$\mathcal{D}_{\text{esq}}(s) \quad \text{e} \quad \mathcal{D}_{\text{dir}}(s).$$

Define-se uma **medida de impureza** $I(\mathcal{D})$, que quantifica o quão heterogêneo é o conjunto de dados no nó. O objetivo é escolher o split s^* que minimiza a impureza ponderada dos subconjuntos:

$$s^* = \arg \min_s \left[\frac{|\mathcal{D}_{\text{esq}}(s)|}{|\mathcal{D}|} I(\mathcal{D}_{\text{esq}}(s)) + \frac{|\mathcal{D}_{\text{dir}}(s)|}{|\mathcal{D}|} I(\mathcal{D}_{\text{dir}}(s)) \right].$$

Equivalentemente, pode-se maximizar o **ganho de informação**:

$$\Delta I(s) = I(\mathcal{D}) - \left[\frac{|\mathcal{D}_{\text{esq}}(s)|}{|\mathcal{D}|} I(\mathcal{D}_{\text{esq}}(s)) + \frac{|\mathcal{D}_{\text{dir}}(s)|}{|\mathcal{D}|} I(\mathcal{D}_{\text{dir}}(s)) \right].$$

O processo é repetido recursivamente em cada nó até que um critério de parada seja satisfeito.

Exemplos de medidas de impureza

Para problemas de classificação, são comuns:

- **Impureza de Gini:**

$$I(\mathcal{D}) = 1 - \sum_k p_k^2;$$

- **Entropia:**

$$I(\mathcal{D}) = - \sum_k p_k \log p_k;$$

onde p_k é a proporção de elementos da classe k no conjunto \mathcal{D} .

Para regressão, utiliza-se tipicamente a variância:

$$I(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} (y_i - \bar{y})^2.$$

4.1.4 Random Forest

Random Forest

Uma **Random Forest** é um modelo de aprendizado supervisionado do tipo ensemble (união de modelos), que consiste em um conjunto finito de árvores de decisão independentes $\{f_b\}_{b=1}^B$, construídas a partir de amostras aleatórias do conjunto de dados e de subconjuntos aleatórios de características.

Seja $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ um conjunto de treinamento. Para cada $b = 1, \dots, B$:

- Constrói-se um conjunto \mathcal{D}_b por **amostragem com reposição** de \mathcal{D} ;
- Treina-se uma árvore de decisão f_b sobre \mathcal{D}_b , onde, a cada divisão, considera-se apenas um subconjunto aleatório das variáveis de entrada.

A predição final é obtida pela agregação das predições individuais:

- **Classificação:**

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \sum_{b=1}^B \mathbf{1}(f_b(x) = y);$$

- **Regressão:**

$$\hat{y} = \frac{1}{B} \sum_{b=1}^B f_b(x).$$

4.1.5 Regressão Linear

Regressão Linear

A **regressão linear** é um modelo de aprendizado supervisionado que busca aproximar uma função $f : \mathcal{X} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$ por uma combinação linear das variáveis de entrada.

No caso geral (regressão linear múltipla), o modelo é dado por:

$$f(x) = b + \sum_{j=1}^d w_j x_j = b + \mathbf{w}^\top x,$$

onde $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, $\mathbf{w} = (w_1, \dots, w_d)$ são os pesos do modelo e b é o termo de bias.

Dado um conjunto de dados $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, os parâmetros são estimados tipicamente pela minimização do erro quadrático médio:

$$\min_{b, \mathbf{w}} \frac{1}{n} \sum_{i=1}^n (y_i - (b + \mathbf{w}^\top x_i))^2.$$

Caso simples: quando $d = 1$, obtém-se a regressão linear simples:

$$f(x) = b + wx,$$

que corresponde ao ajuste de uma reta aos dados no plano.

4.1.6 Otimização no Aprendizado de Máquina

Função de Perda (Loss)

Seja $(x, y) \in \mathcal{X} \times \mathcal{Y}$ um par entrada-saída, e seja $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ um modelo parametrizado por θ . Uma **função de perda** é uma aplicação

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$$

tal que $\ell(f_\theta(x), y)$ mede o custo associado a prever $f_\theta(x)$ quando o valor verdadeiro é y .

Exemplos comuns incluem:

- Erro quadrático: $\ell(\hat{y}, y) = (\hat{y} - y)^2$
- Entropia cruzada (classificação): $\ell(\hat{y}, y) = - \sum_i y_i \log \hat{y}_i$

Risco e Risco Empírico

Seja (X, Y) uma variável aleatória com distribuição desconhecida \mathbb{P} . O **risco** (ou risco esperado) de um modelo f_θ é definido como:

$$\mathcal{R}(\theta) = \mathbb{E}_{(X,Y) \sim \mathbb{P}} [\ell(f_\theta(X), Y)]$$

Como a distribuição \mathbb{P} é desconhecida, utilizamos um conjunto de dados amostrados $\{(x_i, y_i)\}_{i=1}^n$ para definir o **risco empírico**:

$$\hat{\mathcal{R}}_n(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x_i), y_i)$$

O objetivo do aprendizado de máquina é encontrar:

$$\theta^* = \arg \min_{\theta} \hat{\mathcal{R}}_n(\theta)$$

esperando que $\hat{\mathcal{R}}_n(\theta)$ seja uma boa aproximação de $\mathcal{R}(\theta)$.

Gradiente Descendente

O **gradiente descendente** é um método iterativo de otimização para minimizar uma função diferenciável $J : \mathbb{R}^d \rightarrow \mathbb{R}$.

Dado um ponto inicial $\theta^{(0)} \in \mathbb{R}^d$, o método gera uma sequência $\{\theta^{(t)}\}_{t \geq 0}$ definida por:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J(\theta^{(t)}),$$

onde $\eta > 0$ é a **taxa de aprendizado** e $\nabla J(\theta^{(t)})$ é o gradiente de J no ponto $\theta^{(t)}$. Sob condições adequadas (como suavidade de J e escolha apropriada de η), a sequência $\{\theta^{(t)}\}$ converge para um ponto crítico de J , isto é, um ponto θ^* tal que

$$\nabla J(\theta^*) = 0.$$

Early Stopping

Early stopping é uma técnica de regularização utilizada durante o treinamento de modelos de aprendizado de máquina, que consiste em interromper o processo de otimização antes da convergência completa, com base no desempenho em um conjunto de validação. O conjunto de validação é um subconjunto dos dados, que é disjunto aos dados de treino e de teste.

Sejam $\{\theta^{(t)}\}$ os parâmetros ao longo das iterações e $J_{\text{val}}(\theta^{(t)})$ a função de custo avaliada no conjunto de validação. O treinamento é interrompido no instante

$$t^* = \arg \min_t J_{\text{val}}(\theta^{(t)}),$$

ou quando J_{val} deixa de diminuir por um número pré-definido de iterações.

Regularização L_1 e L_2

A **regularização** consiste em modificar a função de custo original adicionando um termo de penalização sobre os parâmetros do modelo, com o objetivo de controlar sua complexidade.

Dada uma função de custo $J(b, \mathbf{w})$, define-se:

- **Regularização L_2 (Ridge):**

$$J_\lambda(b, \mathbf{w}) = J(b, \mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 = J(b, \mathbf{w}) + \lambda \sum_{j=1}^d w_j^2;$$

- **Regularização L_1 (Lasso):**

$$J_\lambda(b, \mathbf{w}) = J(b, \mathbf{w}) + \lambda \|\mathbf{w}\|_1 = J(b, \mathbf{w}) + \lambda \sum_{j=1}^d |w_j|.$$

Aqui, $\lambda \geq 0$ é o parâmetro de regularização que controla o grau de penalização.

A regularização L_2 tende a produzir soluções com pesos pequenos e distribuídos, enquanto a regularização L_1 é ainda mais forte, levando alguns coeficientes a serem nulos.

4.1.7 Regressão Logística

Regressão Logística

A **regressão logística** é um modelo de aprendizado supervisionado para classificação binária, que modela a probabilidade condicional de uma classe $y \in \{0, 1\}$ dado $x \in \mathbb{R}^d$ por meio da função logística.

O modelo é definido por:

$$P(y = 1 \mid x) = \sigma(b + \mathbf{w}^\top x),$$

onde $\mathbf{w} \in \mathbb{R}^d$ são os pesos, $b \in \mathbb{R}$ é o bias e $\sigma : \mathbb{R} \rightarrow (0, 1)$ é a função sigmoide:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

A predição é obtida por:

$$\hat{y} = \begin{cases} 1, & \text{se } P(y = 1 \mid x) \geq \frac{1}{2}, \\ 0, & \text{caso contrário.} \end{cases}$$

Dado um conjunto de dados $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, os parâmetros (b, \mathbf{w}) são estimados pela maximização da verossimilhança, o que é equivalente à minimização da **perda logística** (ou entropia cruzada):

$$\min_{b, \mathbf{w}} -\frac{1}{n} \sum_{i=1}^n [y_i \log \sigma(b + \mathbf{w}^\top x_i) + (1 - y_i) \log (1 - \sigma(b + \mathbf{w}^\top x_i))].$$

4.1.8 Support Vector Machine

Support Vector Machine (SVM)

Uma **Support Vector Machine (SVM)** é um modelo de aprendizado supervisionado para classificação (e regressão) que busca encontrar um hiperplano que separa os dados com a maior margem possível.

No caso linear e separável, dado $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, com $x_i \in \mathbb{R}^d$ e $y_i \in \{-1, +1\}$, o problema é:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{sujeito a} \quad y_i(\mathbf{w}^\top x_i + b) \geq 1, \quad \forall i.$$

Para dados não separáveis, introduzem-se variáveis de folga $\xi_i \geq 0$ e resolve-se:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i \quad \text{sujeito a} \quad y_i(\mathbf{w}^\top x_i + b) \geq 1 - \xi_i, \quad \forall i,$$

onde $C > 0$ controla o compromisso entre maximizar a margem e minimizar os erros.

A predição é dada por:

$$\hat{y} = \text{sign}(\mathbf{w}^\top x + b).$$

Em sua forma dual, o modelo depende apenas de produtos internos entre dados, permitindo o uso de **kernels** $K(x_i, x_j)$ para obter fronteiras de decisão não lineares.

Teorema de Mercer

Seja $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ uma função contínua, simétrica, definida em um conjunto compacto $\mathcal{X} \subset \mathbb{R}^d$. Então, K é um **kernel positivo definido** se, e somente se, para toda função $g \in L^2(\mathcal{X})$, vale:

$$\int_{\mathcal{X}} \int_{\mathcal{X}} g(x) K(x, y) g(y) dx dy \geq 0.$$

Equivalentemente, existe uma base ortonormal $\{\phi_k\}_{k=1}^\infty$ de $L^2(\mathcal{X})$ e uma sequência de autovalores não negativos $\{\lambda_k\}_{k=1}^\infty$ tal que:

$$K(x, y) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(y), \quad \text{com } \lambda_k \geq 0.$$

Em particular, isso implica que existe uma aplicação $\phi : \mathcal{X} \rightarrow \mathcal{H}$ para um espaço de Hilbert \mathcal{H} tal que:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle,$$

justificando o uso de kernels na construção de modelos como as Support Vector Machines.

Na formulação dual da SVM, a solução ótima pode ser escrita como uma combinação

linear dos dados de treinamento:

$$f(x) = \sum_{i=1}^n \alpha_i y_i \langle x_i, x \rangle + b,$$

onde $\alpha_i \geq 0$ são coeficientes obtidos pela otimização. Note que os dados aparecem apenas por meio de produtos internos $\langle x_i, x \rangle$.

A ideia central do caso não-linear é substituir o produto interno por uma função kernel

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle,$$

onde $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ é uma aplicação (possivelmente de alta ou infinita dimensão) para um espaço de características \mathcal{H} .

Assim, a função de decisão torna-se:

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x_i, x) + b,$$

sem que seja necessário computar explicitamente $\phi(x)$ — técnica conhecida como kernel trick.

Geometricamente, isso equivale a mapear os dados para um espaço onde eles se tornam linearmente separáveis e, então, encontrar um hiperplano nesse espaço. No espaço original, isso corresponde a uma fronteira de decisão não linear.

Exemplos comuns de kernels incluem:

- **Kernel linear:** $K(x_i, x_j) = x_i^\top x_j$;
- **Kernel polinomial:** $K(x_i, x_j) = (x_i^\top x_j + c)^p$;
- **Kernel Gaussiano (RBF):** $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$.

Para que K seja válido, ele deve ser simétrico e definido positivo, garantindo que exista um espaço \mathcal{H} e uma aplicação ϕ associados (teorema de Mercer).

Os pontos com $\alpha_i > 0$ são chamados de **vetores de suporte**, pois são os únicos que influenciam diretamente a fronteira de decisão.

4.2 Aula 2 - Introdução a Redes Neurais

4.2.1 Perceptron

Perceptron

Um **perceptron** é uma função parametrizada

$$f_{\theta} : \mathbb{R}^n \rightarrow \mathbb{R},$$

definida por

$$f_{\theta}(\mathbf{x}) = \phi(\mathbf{w} \cdot \mathbf{x} + b),$$

onde:

- $\mathbf{x} \in \mathbb{R}^n$ é o vetor de entrada;
- $\mathbf{w} \in \mathbb{R}^n$ é o vetor de pesos;
- $b \in \mathbb{R}$ é o viés (bias);
- $\phi : \mathbb{R} \rightarrow \mathbb{R}$ é a função de ativação.

No caso clássico de classificação binária, utiliza-se a função degrau:

$$\phi(z) = \begin{cases} 1, & \text{se } z \geq 0, \\ 0, & \text{se } z < 0, \end{cases}$$

de modo que o perceptron define uma regra de decisão linear dada por um hiperplano

$$\mathbf{w} \cdot \mathbf{x} + b = 0.$$

Veremos que o Perceptron é um neurônio de uma rede neural.

Funções de Ativação

- **Função Degrau (Heaviside):**

$$\phi(z) = \begin{cases} 1, & \text{se } z \geq 0, \\ 0, & \text{se } z < 0, \end{cases}$$

Utilizada no perceptron clássico; não é diferenciável.

- **Sigmoide (Logística):**

$$\phi(z) = \frac{1}{1 + e^{-z}},$$

Suave e diferenciável; muito usada historicamente em redes neurais.

- **Tangente Hiperbólica (tanh):**

$$\phi(z) = \tanh(z),$$

Similar à sigmoide, mas centrada em zero: $\phi(z) \in (-1, 1)$.

- **ReLU (Rectified Linear Unit):**

$$\phi(z) = \max(0, z),$$

Amplamente utilizada em redes profundas devido à sua simplicidade e eficiência computacional.

- **Leaky ReLU:**

$$\phi(z) = \begin{cases} z, & z \geq 0, \\ \alpha z, & z < 0, \end{cases} \quad \text{com } 0 < \alpha \ll 1,$$

Variante da ReLU que evita o problema de neurônios mortos.

- **Softmax:**

$$\phi_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_j e^{z_j}},$$

Usada na camada de saída para classificação multiclasse, produzindo uma distribuição de probabilidades.

Otimização do Perceptron

Considere um conjunto de dados rotulados

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad \mathbf{x}_i \in \mathbb{R}^n, \quad y_i \in \{-1, +1\}.$$

O perceptron define uma regra de decisão dada por

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b).$$

O objetivo da otimização é encontrar parâmetros (\mathbf{w}, b) tais que os dados sejam corretamente classificados, isto é,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0 \quad \text{para todo } i.$$

O algoritmo do perceptron realiza uma otimização iterativa baseada em atualizações apenas quando ocorre erro de classificação:

$$\text{Se } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \leq 0, \quad \text{então}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i, \quad b \leftarrow b + \eta y_i,$$

onde $\eta > 0$ é a taxa de aprendizado.

Equivalentemente, essa atualização pode ser interpretada como uma descida de gradiente sobre a seguinte loss:

$$\mathcal{L}(\mathbf{w}, b) = \sum_{i=1}^N \max(0, -y_i(\mathbf{w} \cdot \mathbf{x}_i + b)).$$

Teorema de Convergência do Perceptron

Seja $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ um conjunto de dados com $\mathbf{x}_i \in \mathbb{R}^n$ e $y_i \in \{-1, +1\}$. Suponha que o conjunto seja **linearmente separável**, isto é, existem $\mathbf{w}^* \in \mathbb{R}^n$ e $b^* \in \mathbb{R}$ tais que

$$y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) > 0, \quad \forall i.$$

Defina

$$R = \max_i \|\mathbf{x}_i\|, \quad \gamma = \min_i \frac{y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*)}{\|\mathbf{w}^*\|}.$$

Então, o algoritmo do perceptron realiza um número finito de atualizações e encontra um classificador linear que separa corretamente os dados. Mais precisamente, o número de atualizações T satisfaz

$$T \leq \left(\frac{R}{\gamma}\right)^2.$$

4.2.2 MLP

Perceptron Multicamada

Uma rede neural do tipo **Perceptron Multicamada (MLP)** é uma função parametrizada

$$f_{\theta} : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_L},$$

definida pela composição de transformações afins e funções de ativação não lineares organizadas em camadas.

Formalmente, para uma rede com L camadas, a transformação é dada por

$$\mathbf{x}^{(0)} = \mathbf{x},$$

$$\mathbf{x}^{(\ell)} = \phi^{(\ell)}(W^{(\ell)}\mathbf{x}^{(\ell-1)} + \mathbf{b}^{(\ell)}), \quad \ell = 1, 2, \dots, L,$$

onde:

- $\mathbf{x} \in \mathbb{R}^{n_0}$ é o vetor de entrada;
- $\mathbf{x}^{(\ell)} \in \mathbb{R}^{n_\ell}$ é o vetor de ativações da ℓ -ésima camada;
- $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ é a matriz de pesos da camada ℓ ;
- $\mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}$ é o vetor de vieses (bias);
- $\phi^{(\ell)} : \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_\ell}$ é a função de ativação aplicada componente a componente.

O conjunto de parâmetros da rede é

$$\theta = \{W^{(1)}, \mathbf{b}^{(1)}, \dots, W^{(L)}, \mathbf{b}^{(L)}\}.$$

O objetivo do treinamento consiste em ajustar θ de modo a minimizar uma função de custo definida sobre um conjunto de dados.

Teorema da Aproximação Universal

Seja $\phi : \mathbb{R} \rightarrow \mathbb{R}$ uma função de ativação não constante, limitada e contínua. Considere uma rede neural do tipo **Multi-Layer Perceptron** com uma única camada oculta composta por um número finito $m \in \mathbb{N}$ de neurônios e função de ativação ϕ . Então, para qualquer função contínua

$$f : K \rightarrow \mathbb{R},$$

definida em um conjunto compacto $K \subset \mathbb{R}^n$, e para todo $\varepsilon > 0$, existe uma rede neural da forma

$$F(\mathbf{x}) = \sum_{i=1}^m \alpha_i \phi(\mathbf{w}_i \cdot \mathbf{x} + b_i)$$

tal que

$$\sup_{\mathbf{x} \in K} |f(\mathbf{x}) - F(\mathbf{x})| < \varepsilon.$$

Ou seja, uma rede neural MLP com uma única camada oculta, número suficiente de neurônios e função de ativação correta pode aproximar arbitrariamente bem qualquer função contínua definida em um conjunto compacto.

Note que, classicamente, o Perceptron pode resolver apenas problemas lineares (regressão linear na regressão ou dados linearmente separados na classificação). O SVM resolve esse problema aplicando kernels antes de traçar um hiperplano em um novo espaço.

Em Redes Neurais, resolvemos problemas não-lineares de outra forma. Uma boa estratégia seria utilizar o gráfico de uma função que é equivalente a uma junção de funções lineares, uma em cada intervalo.

Por exemplo, para aproximar a função $F(x) = x^2$, que é não-linear é possível utilizar a função

$$f(x) = \begin{cases} -x, & x < 0 \\ x, & x > 0 \end{cases} \quad (23)$$

Note que f é apenas uma junção de duas funções lineares. Poderíamos melhorar essa aproximação adicionando mais funções lineares e diminuindo o intervalo de cada uma delas.

Isso pode ser alcançado se utilizarmos vários perceptrons em sequência, com uma função de ativação ReLu entre eles. Enquanto $w \cdot x + b$ define rotações e translações da reta, a função ReLu permite zerar essa reta fora de um intervalo.

Assim, um MLP com ReLu apenas encadeia várias funções lineares uma após a outra, de forma a se aproximar aos dados de interesse.

Padrões mais suaves podem ser alcançados com as outras funções de ativação. No geral, o MLP combina várias versões deformadas (esticadas, deslocadas e somadas) da função de ativação à escolha. Isso demonstra intuitivamente a sua eficácia.

4.2.3 Backpropagation

Proposição 17 (Regra da Cadeia) *Sejam $f : \mathbb{R}^m \rightarrow \mathbb{R}$ e $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ funções diferenciáveis, e considere a função composta $h = f \circ g : \mathbb{R}^n \rightarrow \mathbb{R}$. Então, para todo $x \in \mathbb{R}^n$, vale:*

$$\frac{\partial h}{\partial x_i}(x) = \sum_{j=1}^m \frac{\partial f}{\partial y_j}(g(x)) \cdot \frac{\partial g_j}{\partial x_i}(x), \quad i = 1, \dots, n,$$

onde $g(x) = (y_1, \dots, y_m)$.

Backpropagation

Considere uma rede neural feedforward com L camadas. Para cada camada $l = 1, \dots, L$, e neurônios indexados por i , definimos:

$$z_i^{(l)} = \sum_j W_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}, \quad a_i^{(l)} = f^{(l)}(z_i^{(l)})$$

onde:

- $a^{(0)} = x$ é a entrada,
- $W_{ij}^{(l)}$ conecta o neurônio j da camada $l - 1$ ao neurônio i da camada l .

Seja $\mathcal{L}(a^{(L)}, y)$ a função de perda. Definimos:

$$\delta_i^{(l)} := \frac{\partial \mathcal{L}}{\partial z_i^{(l)}}$$

Então:

$$\delta_i^{(L)} = \frac{\partial \mathcal{L}}{\partial a_i^{(L)}} \cdot f'^{(L)}(z_i^{(L)})$$

Para $l = L - 1, \dots, 1$:

$$\delta_i^{(l)} = \left(\sum_k W_{ki}^{(l+1)} \delta_k^{(l+1)} \right) \cdot f'^{(l)}(z_i^{(l)})$$

Os gradientes são dados por:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)}, \quad \frac{\partial \mathcal{L}}{\partial b_i^{(l)}} = \delta_i^{(l)}$$

O algoritmo de backpropagation é uma aplicação sistemática da Regra da Cadeia (Proposição 17) para calcular como a função de perda \mathcal{L} depende de cada parâmetro da rede. Começamos com a propagação direta (forward pass): dada uma entrada $x = a^{(0)}$, calculamos sucessivamente, para cada camada l ,

$$z_i^{(l)} = \sum_j W_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}, \quad a_i^{(l)} = f^{(l)}(z_i^{(l)}),$$

até obter a saída final $a^{(L)}$ e então o valor da perda $\mathcal{L}(a^{(L)}, y)$.

O objetivo é determinar como \mathcal{L} varia em relação aos parâmetros $W_{ij}^{(l)}$ e $b_i^{(l)}$. Para isso, definimos

$$\delta_i^{(l)} := \frac{\partial \mathcal{L}}{\partial z_i^{(l)}},$$

que mede a sensibilidade da perda em relação à entrada do neurônio i na camada l .

Começamos pela última camada. Como \mathcal{L} depende de $z_i^{(L)}$ apenas através de $a_i^{(L)} = f^{(L)}(z_i^{(L)})$, pela regra da cadeia temos

$$\delta_i^{(L)} = \frac{\partial \mathcal{L}}{\partial a_i^{(L)}} \cdot \frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} = \frac{\partial \mathcal{L}}{\partial a_i^{(L)}} \cdot f'^{(L)}(z_i^{(L)}).$$

Para as camadas internas, a dependência é indireta. A variável $z_i^{(l)}$ influencia todos os neurônios da camada seguinte. Aplicando a regra da cadeia,

$$\delta_i^{(l)} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}}.$$

Como

$$z_k^{(l+1)} = \sum_j W_{kj}^{(l+1)} a_j^{(l)} + b_k^{(l+1)}, \quad a_j^{(l)} = f^{(l)}(z_j^{(l)}),$$

temos

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = W_{ki}^{(l+1)} \cdot f'^{(l)}(z_i^{(l)}).$$

Substituindo, obtemos

$$\delta_i^{(l)} = \left(\sum_k W_{ki}^{(l+1)} \delta_k^{(l+1)} \right) \cdot f'^{(l)}(z_i^{(l)}),$$

o que fornece a regra de propagação do erro de trás para frente.

Uma vez conhecidos os $\delta_i^{(l)}$, podemos calcular os gradientes. Para os pesos,

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} \cdot a_j^{(l-1)},$$

pois $z_i^{(l)}$ depende linearmente de $W_{ij}^{(l)}$. Para os vieses,

$$\frac{\partial \mathcal{L}}{\partial b_i^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial b_i^{(l)}} = \delta_i^{(l)}.$$

Assim, o backpropagation consiste em três etapas encadeadas: calcular as ativações na passagem direta, determinar os erros $\delta_i^{(l)}$ começando da saída e propagando-os para trás, e finalmente obter os gradientes dos parâmetros.

4.3 Aula 3 - Prática com Redes Neurais

4.3.1 Link do notebook Google Colab

https://colab.research.google.com/drive/1nAQcYkBAMq1LJ-lpkF76tXQ9V_lkyMhC?usp=sharing

4.3.2 Mais otimizadores

Stochastic Gradient Descent (SGD)

Considere uma função de perda $\mathcal{L}(\boldsymbol{\theta})$. O método de **SGD** atualiza os parâmetros $\boldsymbol{\theta}$ de forma iterativa por

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_t),$$

onde:

- $\eta > 0$ é a taxa de aprendizado;
- \mathcal{L}_i é a perda avaliada em uma amostra (ou minibatch);
- $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i$ é o gradiente estocástico.

Muitas vezes, o GD clássico é muito lento quando há muitos dados de treinamento. Nesses casos, é útil usar o SGD, que utiliza um subconjunto dos dados de treinamento (minibatch) em cada época do treinamento.

SGD com Momentum

O método de **SGD com momentum** introduz uma variável de velocidade \mathbf{v}_t , dada por

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_t),$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{v}_{t+1},$$

onde $\beta \in [0, 1)$ controla a contribuição das atualizações passadas.

Note que cada componente da variável velocidade aumenta conforme a derivada naquele ponto for maior.

Interpretando a curva loss como função de energia potencial gravitacional, é como se SGD com momentum adicionasse mais velocidade conforme o modelo desce uma curva muito forte e perdesse velocidade conforme sobe. Essa velocidade permite escapar de mínimos locais e, quando o modelo chega em um mínimo global, ele dificilmente consegue sair.

RMSProp (Root Mean Square Propagation)

O método **RMSProp** adapta a taxa de aprendizado para cada parâmetro com base em uma média móvel dos quadrados dos gradientes.

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1 - \beta) (\nabla_{\theta} \mathcal{L}_i(\theta_t))^2,$$

$$\theta_{t+1} = \theta_t - \eta \frac{\nabla_{\theta} \mathcal{L}_i(\theta_t)}{\sqrt{\mathbf{v}_{t+1} + \varepsilon}},$$

onde:

- $\beta \in [0, 1)$ controla a média móvel dos gradientes ao quadrado;
- $\varepsilon > 0$ evita divisão por zero.

Em alguns casos, utilizar a mesma taxa de aprendizado η para todos os pesos não é uma boa ideia. Faz mais sentido aumentar a taxa de aprendizado conforme uma média móvel que diz quanto o gradiente está baixo. Assim, se para um peso, a curva de loss está muito plana, a sua taxa de aprendizado pessoal é aumentada, para acelerar a busca por mínimos. Se, para outro peso, a curva de loss está muito inclinada, a sua taxa de aprendizado pessoal deve ser diminuída, para evitar pular um mínimo.

ADAM (Adaptive Moment Estimation)

O algoritmo **ADAM** combina estimativas de primeira e segunda ordem do gradiente.

$$\mathbf{m}_{t+1} = \beta_1 \mathbf{m}_t + (1 - \beta_1) \nabla_{\theta} \mathcal{L}_i(\theta_t),$$

$$\mathbf{v}_{t+1} = \beta_2 \mathbf{v}_t + (1 - \beta_2) (\nabla_{\theta} \mathcal{L}_i(\theta_t))^2,$$

As estimativas corrigidas são

$$\hat{\mathbf{m}}_{t+1} = \frac{\mathbf{m}_{t+1}}{1 - \beta_1^{t+1}}, \quad \hat{\mathbf{v}}_{t+1} = \frac{\mathbf{v}_{t+1}}{1 - \beta_2^{t+1}},$$

e a atualização dos parâmetros é

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{\mathbf{m}}_{t+1}}{\sqrt{\hat{\mathbf{v}}_{t+1} + \varepsilon}},$$

onde $\varepsilon > 0$ evita divisão por zero.

O ADAM combina as ideias do momentum e do RMSProp. Por um lado, utiliza uma média móvel do gradiente (primeiro momento), que funciona como uma velocidade acumulada e suaviza a direção da descida. Por outro, utiliza uma média móvel dos quadrados do gradiente (segundo momento), que adapta a taxa de aprendizado para cada parâmetro individualmente.

Na prática, isso torna o ADAM mais robusto e eficiente do que SGD puro, sendo uma das escolhas padrão no treinamento de redes neurais profundas.

ADAMW

O **ADAMW** é uma variante do ADAM que desacopla a regularização ℓ_2 (weight decay) da atualização do gradiente.

Mantêm-se as mesmas definições de \mathbf{m}_t e \mathbf{v}_t do ADAM. A atualização dos parâmetros é dada por

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left(\frac{\hat{\mathbf{m}}_{t+1}}{\sqrt{\hat{\mathbf{v}}_{t+1} + \varepsilon}} + \lambda \boldsymbol{\theta}_t \right),$$

onde $\lambda > 0$ é o coeficiente de decaimento de pesos.

4.3.3 Redes Neurais Recorrentes**Rede Neural Recorrente (RNN)**

Uma **Rede Neural Recorrente (RNN)** é uma função parametrizada que processa sequências de dados introduzindo uma dependência temporal por meio de um estado oculto.

Considere uma sequência de entrada

$$(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T), \quad \mathbf{x}_t \in \mathbb{R}^n.$$

A RNN define, para cada instante $t = 1, \dots, T$, um estado oculto $\mathbf{h}_t \in \mathbb{R}^m$ e uma saída \mathbf{y}_t , dados por

$$\mathbf{h}_t = \phi(W_{xh}\mathbf{x}_t + W_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h),$$

$$\mathbf{y}_t = \psi(W_{hy}\mathbf{h}_t + \mathbf{b}_y),$$

onde:

- \mathbf{h}_0 é o estado inicial (tipicamente $\mathbf{0}$);
- $W_{xh} \in \mathbb{R}^{m \times n}$ conecta entrada para estado oculto;
- $W_{hh} \in \mathbb{R}^{m \times m}$ modela a recorrência temporal;
- $W_{hy} \in \mathbb{R}^{k \times m}$ conecta estado oculto à saída;
- $\mathbf{b}_h \in \mathbb{R}^m$ e $\mathbf{b}_y \in \mathbb{R}^k$ são vieses;
- ϕ e ψ são funções de ativação.

O conjunto de parâmetros é

$$\boldsymbol{\theta} = \{W_{xh}, W_{hh}, W_{hy}, \mathbf{b}_h, \mathbf{b}_y\}.$$

RNNs são úteis para trabalhar com conjuntos de dados cujo número de features é variável. Ao invés de treinar uma rede neural nova com t neurônios de input para cada número de

features t , usa-se um loop dentro da rede neural para simular cópias da rede. Note que os pesos são os mesmos para cada cópia. Exemplos de aplicações de RNNs são predição de séries temporais e predição de textos.

Long Short-Term Memory (LSTM)

Uma **LSTM** é uma variante de Rede Neural Recorrente que introduz um estado de memória \mathbf{c}_t e mecanismos de *gates* para controlar o fluxo de informação ao longo do tempo.

Dada uma sequência de entrada

$$(\mathbf{x}_1, \dots, \mathbf{x}_T), \quad \mathbf{x}_t \in \mathbb{R}^n,$$

a LSTM define, para cada $t = 1, \dots, T$, um estado oculto $\mathbf{h}_t \in \mathbb{R}^m$ e um estado de célula $\mathbf{c}_t \in \mathbb{R}^m$ por

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i),$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o),$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

onde:

- \mathbf{f}_t é o **forget gate**;
- \mathbf{i}_t é o **input gate**;
- \mathbf{o}_t é o **output gate**;
- $\tilde{\mathbf{c}}_t$ é o estado candidato da célula;
- \odot denota o produto elemento a elemento;
- σ é a função sigmoide;
- \tanh é a tangente hiperbólica;
- W_*, U_* são matrizes de pesos e \mathbf{b}_* são vieses.

O conjunto de parâmetros é

$$\boldsymbol{\theta} = \{W_f, U_f, \mathbf{b}_f, W_i, U_i, \mathbf{b}_i, W_c, U_c, \mathbf{b}_c, W_o, U_o, \mathbf{b}_o\}.$$

Gated Recurrent Unit (GRU)

Uma **GRU** é uma variante de Rede Neural Recorrente que utiliza mecanismos de *gates* para controlar o fluxo de informação, sem introduzir um estado de célula separado (como na LSTM).

Dada uma sequência de entrada

$$(\mathbf{x}_1, \dots, \mathbf{x}_T), \quad \mathbf{x}_t \in \mathbb{R}^n,$$

a GRU define, para cada $t = 1, \dots, T$, um estado oculto $\mathbf{h}_t \in \mathbb{R}^m$ por

$$\mathbf{z}_t = \sigma(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} + \mathbf{b}_z),$$

$$\mathbf{r}_t = \sigma(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} + \mathbf{b}_r),$$

$$\tilde{\mathbf{h}}_t = \tanh(W_h \mathbf{x}_t + U_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h),$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t,$$

onde:

- \mathbf{z}_t é o **update gate**;
- \mathbf{r}_t é o **reset gate**;
- $\tilde{\mathbf{h}}_t$ é o estado oculto candidato;
- \odot denota o produto elemento a elemento;
- σ é a função sigmoide;
- \tanh é a tangente hiperbólica;
- W_*, U_* são matrizes de pesos e \mathbf{b}_* são vieses.

O conjunto de parâmetros é

$$\boldsymbol{\theta} = \{W_z, U_z, \mathbf{b}_z, W_r, U_r, \mathbf{b}_r, W_h, U_h, \mathbf{b}_h\}.$$

4.3.4 Redes Convolucionais

Convolução Discreta (em CNNs)

Seja $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ um tensor de entrada e $W \in \mathbb{R}^{P \times Q \times C \times K}$ um conjunto de filtros. A **convolução discreta** (mais precisamente, correlação cruzada, como usada em CNNs) produz uma saída

$$\mathbf{Y} \in \mathbb{R}^{H' \times W' \times K},$$

definida por

$$\mathbf{Y}_{i,j,k} = \sum_{c=1}^C \sum_{u=1}^P \sum_{v=1}^Q W_{u,v,c,k} \mathbf{X}_{i+u,j+v,c}.$$

Opcionalmente, incluindo viés e função de ativação:

$$\mathbf{Y}_{i,j,k} = \phi \left(\sum_{c=1}^C \sum_{u=1}^P \sum_{v=1}^Q W_{u,v,c,k} \mathbf{X}_{i+u,j+v,c} + b_k \right).$$

Os índices (i, j) percorrem as posições espaciais da saída, e k indexa os diferentes filtros aplicados.

Rede Neural Convolutacional (CNN)

Uma **Rede Neural Convolutacional (CNN)** é uma função parametrizada que utiliza a operação de convolução para extrair características locais de dados estruturados (tipicamente imagens).

Considere uma entrada

$$\mathbf{X} \in \mathbb{R}^{H \times W \times C},$$

onde H , W e C representam altura, largura e número de canais.

Uma camada convolutacional produz uma saída

$$\mathbf{Y} \in \mathbb{R}^{H' \times W' \times K},$$

dada pela convolução discreta

$$\mathbf{Y}_{i,j,k} = \phi \left(\sum_{c=1}^C \sum_{u=1}^P \sum_{v=1}^Q W_{u,v,c,k} \mathbf{X}_{i+u,j+v,c} + b_k \right),$$

onde:

- $W \in \mathbb{R}^{P \times Q \times C \times K}$ é o conjunto de filtros (kernels);
- $P \times Q$ é o tamanho espacial do kernel;
- K é o número de filtros;
- $b_k \in \mathbb{R}$ é o viés associado ao filtro k ;
- ϕ é a função de ativação.

Em geral, uma CNN é composta por uma sequência de camadas convolucionais, possivelmente intercaladas com operações de pooling, seguidas por camadas totalmente conectadas.

O conjunto de parâmetros é formado pelos filtros e vieses de todas as camadas.

Em muitos casos, na prática, MLPs não são o suficiente para classificar imagens. Para treinar uma MLP com um conjunto de dados de imagens (matrizes), é necessário transformar as matrizes em vetores (flattening). Esse input é bem difícil para o MLP entender, pois os pixels da imagem perdem a estrutura espacial quando são planificados.

A ideia da CNN é aplicar algum tipo de operação sobre essas matrizes que geram vetores mais característicos do que simplesmente um vetor de pixels.

As CNNs exploram a estrutura espacial dos dados, diferentemente das MLPs. Elas conseguem aplicar filtros locais que percorrem a entrada, detectando padrões como bordas, texturas e formas.

Operações de pooling são frequentemente utilizadas para reduzir a dimensionalidade e tornar as representações mais robustas a pequenas variações da entrada.

Assim, a cada camada convolutacional, a rede extrai e filtra os aspectos mais importantes das features. Com um bom número de camadas, ela prepara bem os dados para passar por uma rede MLP, que produzirá a saída final de interesse.

4.4 Aula 4 - Aprendizagem de Máquina Quântica

4.4.1 Link do notebook Google Colab

https://colab.research.google.com/drive/1nAQcYkBAMq1LJ-lpkF76tXQ9V_lkyMhC?usp=sharing

4.4.2 O que é?

Quantum Machine Learning (QML) é a interseção entre Computação Quântica e Aprendizado de Máquina.

Podemos classificar os métodos de Machine Learning no geral de acordo com a natureza dos dados e do processamento:

- **CC (Classical \rightarrow Classical):** Algoritmos clássicos de *Machine Learning* executados em computadores clássicos para analisar dados clássicos.
- **QC (Quantum \rightarrow Classical):** Algoritmos clássicos de aprendizado de máquina aplicados a dados provenientes de sistemas quânticos (por exemplo, medições experimentais).
- **CQ (Classical \rightarrow Quantum):** Algoritmos quânticos de aprendizado de máquina que processam dados clássicos, utilizando computadores quânticos com o objetivo de acelerar ou melhorar a análise.
- **QQ (Quantum \rightarrow Quantum):** Algoritmos quânticos que processam dados quânticos, com potencial de reduzir o custo computacional na análise de sistemas quânticos complexos.

Aqui, chamaremos de QML tudo aquilo que envolver CQ e QQ.

4.4.3 Evolução da QML

As primeiras ideias de Quantum Machine Learning surgiram ainda na década de 1990, com propostas como o Perceptron Quântico (1994), que sugeria modelos com capacidade expressiva potencialmente ilimitada.

Entre os anos de 2009 e 2016, a área ganhou grande destaque devido à promessa de speedups exponenciais em relação a algoritmos clássicos. Nesse período, foram propostos diversos algoritmos quânticos para tarefas de aprendizado de máquina, como:

- HHL (para sistemas lineares);
- qPCA (análise de componentes principais quântica);
- QSVM (máquinas de vetor de suporte quânticas);
- Sistemas de recomendação quânticos.

A principal hipótese por trás desses avanços era a existência de um acesso eficiente a uma QRAM (Quantum Random Access Memory), capaz de armazenar e fornecer dados de forma eficiente para algoritmos quânticos.

No entanto, em 2018, a cientista computacional Ewin Tang demonstrou que vários desses algoritmos podiam ser simulados eficientemente de forma clássica, desde que certas estruturas de dados fossem utilizadas. Esse resultado teve um impacto profundo na área: o foco deixou de ser algoritmos puramente quânticos.

4.4.4 VQA em QML

Após os resultados de Ewin Tang, algoritmos variacionais emergem como a principal abordagem para QML em dispositivos NISQ, substituindo algoritmos que dependiam de QRAM e QPE.

Esses algoritmos seguem uma estrutura híbrida (quântica-clássica):

- Codifica-se o dado \mathbf{x} em um estado quântico por meio de um *feature map* $V(\mathbf{x})$:

$$|\psi_{\mathbf{x}}\rangle = V(\mathbf{x}) |0\rangle.$$

- Aplica-se um circuito quântico parametrizado $U(\boldsymbol{\theta})$:

$$|\psi_{\mathbf{x},\boldsymbol{\theta}}\rangle = U(\boldsymbol{\theta}) |\psi_{\mathbf{x}}\rangle.$$

- Mede-se um observável M (por exemplo, Z_0). A predição do modelo é:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \langle \psi_{\mathbf{x},\boldsymbol{\theta}} | M | \psi_{\mathbf{x},\boldsymbol{\theta}} \rangle.$$

- A função de perda é avaliada classicamente:

$$\mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), y).$$

- Um otimizador clássico ajusta os parâmetros $\boldsymbol{\theta}$ para minimizar a perda média sobre o conjunto de dados.

Esse pipeline já foi visto no Bloco 2. De fato, algoritmos como VQE e QAOA podem ser adaptados para QML.

Um problema que muitas vezes ocorre em VQA quando há muitos qubits no circuito é o **barren plateaus**. Esse fenômeno ocorre quando gradientes de cada componente se tornarem quase nulos, devido à distribuição do custo sobre as muitas componentes. Geometricamente, isso quer dizer que o modelo ficou preso em uma região plana da superfície da função de custo.

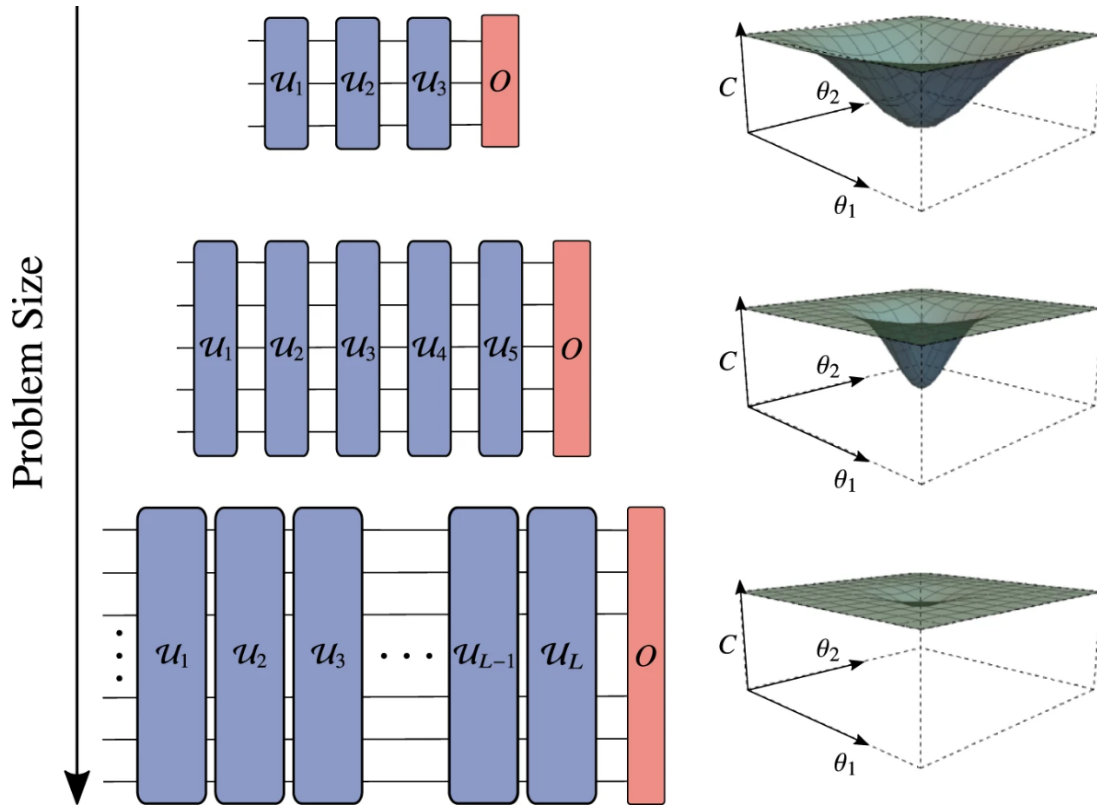


Figura 16: Ilustração do fenômeno barren plateaus. Fonte: Wang, S., Fontana, E., Cerezo, M. et al. Noise-induced barren plateaus in variational quantum algorithms. Nat Commun 12, 6961 (2021). <https://doi.org/10.1038/s41467-021-27045-6>

4.5 Aula 5 - Modelos de Aprendizagem Quântica

4.5.1 Métodos gerais

Feature Map

Um **feature map** é uma função $f : \mathbb{R}^n \rightarrow \mathcal{H}$ que busca representar um vetor de dados clássicos na forma de um estado quântico. Assim,

$$x \mapsto |\psi(x)\rangle = f(x).$$

Exemplos:

- **Angle Encoding**

Os dados clássicos são codificados como ângulos de rotações em qubits. Para um vetor de entrada $x = (x_1, x_2, \dots, x_n)$, um exemplo comum é:

$$|\psi(x)\rangle = \bigotimes_{i=1}^n R_y(x_i) |0\rangle$$

Esse método é simples e eficiente, sendo amplamente utilizado em Variational Quantum Circuits.

- **Amplitude Encoding**

Os dados clássicos são codificados diretamente nas amplitudes do estado quântico:

$$x = (x_0, x_1, \dots, x_{2^n-1})$$

$$|\psi(x)\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} x_i |i\rangle$$

Esse método é muito eficiente em termos de número de qubits, pois 2^n dados são codificados em apenas n qubits.

- **Basis Encoding**

Os dados clássicos são codificados diretamente nos estados base computacionais:

$$x = (x_1, x_2, \dots, x_n), \quad x_i \in \{0, 1\}$$

$$|\psi(x)\rangle = |x_1 x_2 \dots x_n\rangle$$

Por exemplo,

$$x = (1, 0, 1) \quad \mapsto \quad |\psi(x)\rangle = |101\rangle$$

Esse método é simples, mas limitado a dados binários.

- **IQP Feature Map**

O *Instantaneous Quantum Polynomial (IQP)* feature map utiliza portas diagonais e emaranhamento:

$$|\psi(x)\rangle = U_{\text{IQP}}(x) H^{\otimes n} |0\rangle$$

onde

$$U_{\text{IQP}}(x) = \exp \left(i \sum_j x_j Z_j + i \sum_{j,k} x_j x_k Z_j Z_k \right)$$

Esse tipo de feature map é utilizado em **Quantum Kernel Methods**.

- **Data Re-uploading**

Os dados são inseridos múltiplas vezes ao longo do circuito:

$$|\psi(x)\rangle = U(\theta_L, x) \dots U(\theta_2, x) U(\theta_1, x) |0\rangle$$

onde cada bloco é um feature map seguido de uma porta parametrizada:

$$U(\theta_i, x) = U(\theta_i) f(x)$$

Essa estratégia aumenta a expressividade do circuito quântico, usado principalmente em redes neurais quânticas para que elas se tornem mais próximas da universalidade e para evitar barren plateaus.

- **Hamiltonian-based Feature Maps**

Os dados são codificados por evolução temporal de um Hamiltoniano dependente dos dados:

$$|\psi(x)\rangle = e^{-iH(x)t}|0\rangle$$

onde o Hamiltoniano depende do vetor de entrada:

$$H(x) = \sum_i x_i H_i$$

Utilizamos feature maps em QML porque a maioria dos dados são clássicos, de tal forma que precisamos representá-los de outra forma para que o circuito quântico possa trabalhar com eles.

Proposição 18 (Regra do Parameter-Shift) *Considere um circuito quântico parametrizado por uma rotação*

$$U(\theta) = e^{-i\theta H}$$

onde o gerador H possui apenas dois autovalores $\pm \frac{1}{2}$. Seja

$$f(\theta) = \langle O \rangle_\theta = \langle 0|U^\dagger(\theta) O U(\theta)|0\rangle$$

o valor esperado de um observável O . Então,

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{1}{2} \left[f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) \right]$$

Prova:

Considere

$$f(\theta) = \langle 0|U^\dagger(\theta) O U(\theta)|0\rangle$$

Derivando em relação a θ :

$$\frac{\partial f}{\partial \theta} = \langle 0|\frac{\partial U^\dagger}{\partial \theta} O U|0\rangle + \langle 0|U^\dagger O \frac{\partial U}{\partial \theta}|0\rangle$$

Como

$$U(\theta) = e^{-i\theta H}$$

temos

$$\frac{\partial U}{\partial \theta} = -iHU(\theta)$$

e

$$\frac{\partial U^\dagger}{\partial \theta} = iU^\dagger(\theta)H$$

Substituindo:

$$\begin{aligned}\frac{\partial f}{\partial \theta} &= i\langle OH \rangle_\theta - i\langle HO \rangle_\theta \\ &= i\langle [O, H] \rangle_\theta\end{aligned}$$

Agora usamos o fato de que H possui autovalores $\pm \frac{1}{2}$. Nesse caso, podemos escrever

$$H = \frac{1}{2}P$$

onde P é um operador de Pauli ($P^2 = I$).

Assim,

$$U(\theta) = e^{-i\theta P/2}$$

Usando a identidade exponencial para operadores de Pauli:

$$e^{-i\theta P/2} = \cos\left(\frac{\theta}{2}\right)I - i\sin\left(\frac{\theta}{2}\right)P$$

Portanto, o valor esperado $f(\theta)$ torna-se uma combinação de senos e cossenos:

$$f(\theta) = A\cos(\theta) + B\sin(\theta) + C$$

Derivando:

$$\frac{\partial f}{\partial \theta} = -A\sin(\theta) + B\cos(\theta)$$

Agora calculamos:

$$f\left(\theta + \frac{\pi}{2}\right) = -A \sin(\theta) + B \cos(\theta) + C$$

$$f\left(\theta - \frac{\pi}{2}\right) = A \sin(\theta) - B \cos(\theta) + C$$

Subtraindo:

$$f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) = 2[-A \sin(\theta) + B \cos(\theta)]$$

Logo,

$$\frac{\partial f}{\partial \theta} = \frac{1}{2} \left[f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right) \right]$$

o que conclui a demonstração.

□

O método do parameter-shift nos permite calcular gradientes de circuitos quânticos variacionais sem backpropagation, usando apenas medições.

4.5.2 Quantum k-Nearest Neighbors

Quantum k-Nearest Neighbors

O **Quantum k-Nearest Neighbors (QKNN)** é a versão quântica do algoritmo *k-Nearest Neighbors* (kNN), na qual os dados clássicos são codificados em estados quânticos e a similaridade entre amostras é calculada no espaço de Hilbert. Considere um conjunto de treinamento rotulado

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

onde:

- $x_i \in \mathbb{R}^d$ são os dados de entrada
- $y_i \in \{1, \dots, C\}$ são os rótulos

Cada dado é codificado em um estado quântico através de um **feature map quântico**

$$x \mapsto |\phi(x)\rangle = S(x)|0\rangle$$

onde $S(x)$ é um circuito quântico dependente dos dados.

Dado um ponto de teste x^* , o QKNN calcula a similaridade entre $|\phi(x^*)\rangle$ e cada estado do conjunto de treinamento:

$$\text{sim}(x^*, x_i) = |\langle \phi(x^*) | \phi(x_i) \rangle|^2$$

Essa quantidade corresponde ao overlap quântico entre os estados e pode ser estimada experimentalmente por medições (por exemplo, *swap test* ou circuitos equivalentes).

Em seguida:

1. Calculam-se as similaridades entre o ponto de teste e todos os dados
2. Selecionam-se os k maiores valores de similaridade
3. Obtêm-se os rótulos correspondentes
4. O rótulo final é determinado por votação majoritária ou média ponderada

4.5.3 Quantum Support Vector Machine

Quantum Support Vector Machine (QSVM)

O **Quantum Support Vector Machine (QSVM)** é uma versão híbrida do *Support Vector Machine* (SVM), na qual o kernel é calculado utilizando um circuito quântico, enquanto o treinamento e a predição permanecem clássicos. Considere um conjunto de treinamento rotulado

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

onde

- $x_i \in \mathbb{R}^d$ são os vetores de entrada
- $y_i \in \{-1, +1\}$ são os rótulos

No QSVM, cada dado é codificado em um estado quântico através de um feature map:

$$x \mapsto |\phi(x)\rangle = S(x)|0\rangle$$

onde $S(x)$ é um circuito quântico dependente dos dados.

A similaridade entre dois dados é definida por um quantum kernel

$$K(x_i, x_j) = |\langle \phi(x_i) | \phi(x_j) \rangle|^2$$

Esse kernel é estimado experimentalmente por medições em circuitos quânticos (por exemplo, *swap test* ou circuitos equivalentes).

Após a construção da matriz de kernel

$$K_{ij} = K(x_i, x_j)$$

o treinamento do SVM é realizado resolvendo o problema dual:

$$\max_{\alpha} \left[\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j K_{ij} \right]$$

sujeito às restrições

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^N \alpha_i y_i = 0$$

Após o treinamento, a função de decisão é dada por

$$f(x) = \sum_{i=1}^N \alpha_i y_i K(x_i, x) + b$$

e a predição final é

$$\hat{y}(x) = \text{sign}(f(x))$$

4.5.4 Variational Quantum Classifier

Variational Quantum Classifier (VQC)

O **Variational Quantum Classifier (VQC)** é um modelo de classificação quântica baseado em VQA, no qual os dados clássicos são codificados em estados quânticos e a classificação é obtida por meio da otimização de parâmetros de um circuito quântico.

Considere um conjunto de treinamento rotulado

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

onde

- $x_i \in \mathbb{R}^d$ são os vetores de entrada
- $y_i \in \{1, \dots, C\}$ são os rótulos

Cada dado é codificado em um estado quântico por meio de um feature map. Em seguida, aplica-se um circuito variacional parametrizado

$$|\psi(x, \theta)\rangle = U(\theta) |\phi(x)\rangle$$

onde

- $U(\theta)$ é um circuito quântico parametrizado
- θ representa os parâmetros treináveis

A classificação é obtida medindo um observável M :

$$f(x, \theta) = \langle \psi(x, \theta) | M | \psi(x, \theta) \rangle$$

Para classificação binária, a predição é dada por

$$\hat{y}(x) = \text{sign}(f(x, \theta))$$

Para classificação multiclasse, utiliza-se um conjunto de observáveis $\{M_k\}$ ou múltiplas medições:

$$f_k(x, \theta) = \langle \psi(x, \theta) | M_k | \psi(x, \theta) \rangle$$

e a classe prevista é

$$\hat{y}(x) = \arg \max_k f_k(x, \theta)$$

Os parâmetros θ são otimizados classicamente minimizando uma função de custo sobre o conjunto de treinamento:

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(x_i, \theta), y_i)$$

onde \mathcal{L} é uma função loss.

O VQC, por ser uma função composta que pode ser universal, é frequentemente denominado como o exemplo mais simples de uma rede neural quântica (QNN).

4.5.5 Quantum k-Means

Também existe um modelo de QML para o problema de classificação não-supervisionada.

Quantum k-Means

O **Quantum k-Means** é a versão quântica do algoritmo clássico *k-Means*, na qual as distâncias entre os dados e os centróides são calculadas utilizando estados quânticos e medições no espaço de Hilbert.

Considere um conjunto de dados não rotulados

$$\mathcal{D} = \{x_i\}_{i=1}^N, \quad x_i \in \mathbb{R}^d$$

O objetivo do algoritmo é particionar os dados em k clusters $\{C_1, \dots, C_k\}$, minimizando a função custo

$$J = \sum_{j=1}^k \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

onde μ_j representa o centróide do cluster C_j .

No Quantum k-Means, cada dado é codificado em um estado quântico através de um feature map quântico.

A distância entre dois pontos é estimada utilizando o overlap quântico:

$$\text{sim}(x_i, x_j) = |\langle \phi(x_i) | \phi(x_j) \rangle|^2$$

A distância quântica pode então ser definida como

$$d_Q(x_i, x_j) = 1 - |\langle \phi(x_i) | \phi(x_j) \rangle|^2$$

O algoritmo segue os seguintes passos:

1. Inicializar k centróides $\{\mu_1, \dots, \mu_k\}$ (os primeiros são aleatórios)
2. Codificar os dados e centróides em estados quânticos

$$|\phi(x_i)\rangle, \quad |\phi(\mu_j)\rangle$$

3. Atribuir cada ponto ao cluster mais próximo:

$$C_j = \left\{ x_i \mid j = \arg \min_l d_Q(x_i, \mu_l) \right\}$$

4. Atualizar os centróides:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

5. Repetir até convergência

4.6 Aula 6 - Introdução a Redes Neurais Quânticas I

4.6.1 Peças chaves para a QNN

Uma rede neural quântica é um modelo de aproximação de funções que utiliza circuitos quânticos parametrizados (variacionais) para processar os dados e retornar um valor por meio do valor esperado de um operador de medição.

O campo de estudos das QNNs ainda é recente. Até agora, muitas das conclusões são computacionalmente experimentais, carecendo de fundamento teórico robusto. Ainda não foi descoberto se existe um Teorema da Aproximação Universal de QNN, mas as expectativas sobre esse modelo são altas.

O feature map geralmente usado em QNN é o **data re-uploading**, principalmente para evitar o barren plateaus em problemas que precisam de muitos qubits ou de um ansatz profundo.

Após o feature map, precisamos de um circuito! Pense que precisamos de algo que simule arestas de interação entre os qubits e superposição. Há várias formas de fazer esses circuitos, sendo elas bons temas de artigos científicos.

4.6.2 Arquiteturas

Diversas arquiteturas de circuitos variacionais são utilizadas em QNNs, diferenciando-se principalmente pela estrutura de emaranhamento entre os qubits. A escolha da arquitetura afeta diretamente a expressividade e o custo computacional do modelo.

- **Brick Wall**

Na arquitetura *Brick Wall*, os qubits são conectados em camadas alternadas de portas de dois qubits, formando um padrão semelhante a tijolos em uma parede. Tipicamente, aplica-se:

- rotações locais em todos os qubits
- portas de emaranhamento entre pares vizinhos alternados

- **Chain**

Na arquitetura *Chain*, os qubits são conectados sequencialmente em uma cadeia linear:

$$q_1 \leftrightarrow q_2 \leftrightarrow q_3 \leftrightarrow \cdots \leftrightarrow q_n$$

- **Lambda**

Na arquitetura *Lambda*, os qubits são conectados a partir de um qubit central, formando uma estrutura semelhante à letra λ . Um qubit atua como um "hub" de emaranhamento:

$$q_1 \leftrightarrow q_c \leftrightarrow q_2$$

$$q_3 \leftrightarrow q_c \leftrightarrow q_4$$

- **Hyperbolic**

Na arquitetura *Hyperbolic*, os qubits são conectados em uma estrutura hierárquica inspirada em grafos hiperbólicos ou árvores.

- **Supercube**

A arquitetura *Supercube* conecta qubits de forma não-local, inspirada em hipercubos de alta dimensão.

4.6.3 Modelo híbrido

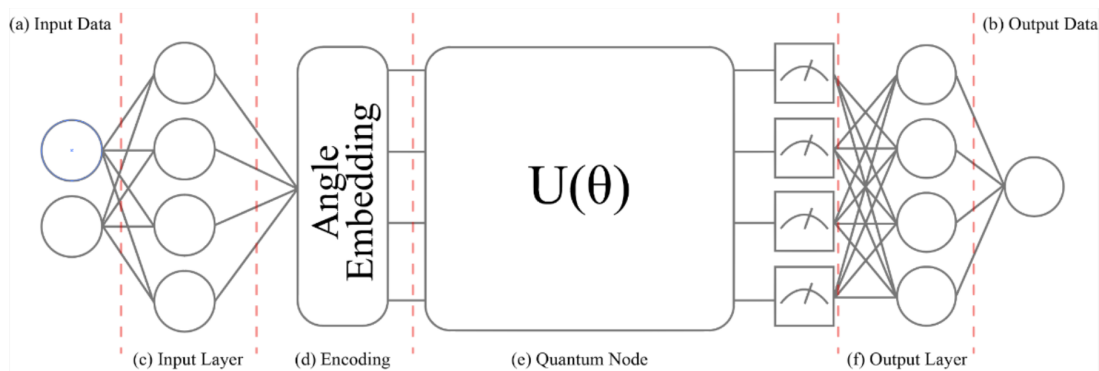


Figura 17: Diagrama do modelo híbrido de rede neural quântica. Fonte: Brazil Quantum Camp

Em aplicações, um dos modelos mais usados é o híbrido, que consiste em usar uma QNN dentro de uma rede neural clássica.

A ideia geral é aproveitar o melhor dos dois mundos: a estabilidade da clássica e a expressividade instável da quântica. É como se a primeira rede clássica desse um input melhor para a rede quântica, e a quântica devolvesse com um input ainda melhor para a segunda rede clássica. Isso ajuda a capturar detalhes do conjunto de dados.

Outra vantagem do modelo híbrido é que, em alguns cenários, a parte quântica pode atingir desempenho semelhante (ou melhor) com menos parâmetros treináveis do que uma rede singular.

4.6.4 Quantum Convolutional Neural Network

Quantum Convolutional Neural Network (QCNN)

A **Quantum Convolutional Neural Network (QCNN)** é uma arquitetura de rede neural quântica inspirada nas *Convolutional Neural Networks* (CNNs) clássicas, composta por camadas locais de convolução quântica, operações de pooling quântico e uma camada final de classificação.

Considere um estado de entrada codificado em n qubits $|\psi_0(x)\rangle$.

A QCNN consiste em uma sequência de camadas do tipo

$$|\psi_L(x, \theta)\rangle = U_L(\theta_L) \cdots U_2(\theta_2) U_1(\theta_1) |\psi_0(x)\rangle$$

onde cada camada $U_l(\theta_l)$ é composta por:

- **Camada de Convolução Quântica**

Unitárias parametrizadas aplicadas localmente em qubits vizinhos:

$$U_{\text{conv}}(\theta) = \prod_{(i,j) \in \mathcal{N}} U_{ij}(\theta)$$

onde \mathcal{N} representa pares de qubits vizinhos e $U_{ij}(\theta)$ são portas de dois qubits parametrizadas.

- **Pooling Quântico**

Redução do número de qubits através de medições ou operações controladas, removendo graus de liberdade:

$$|\psi\rangle \longrightarrow |\psi'\rangle$$

com menor número de qubits.

Após várias camadas de convolução e pooling, aplica-se uma **camada final variacional** (simulando a camada fully connected):

$$|\psi_{\text{final}}(x, \theta)\rangle = U_{\text{final}}(\theta) |\psi_L(x, \theta)\rangle$$

A classificação é obtida medindo um observável

$$f(x, \theta) = \langle \psi_{\text{final}}(x, \theta) | M | \psi_{\text{final}}(x, \theta) \rangle,$$

que será então passado por alguma função de ativação caso o problema seja de classificação.

Os parâmetros θ são aprendidos minimizando uma função de custo.

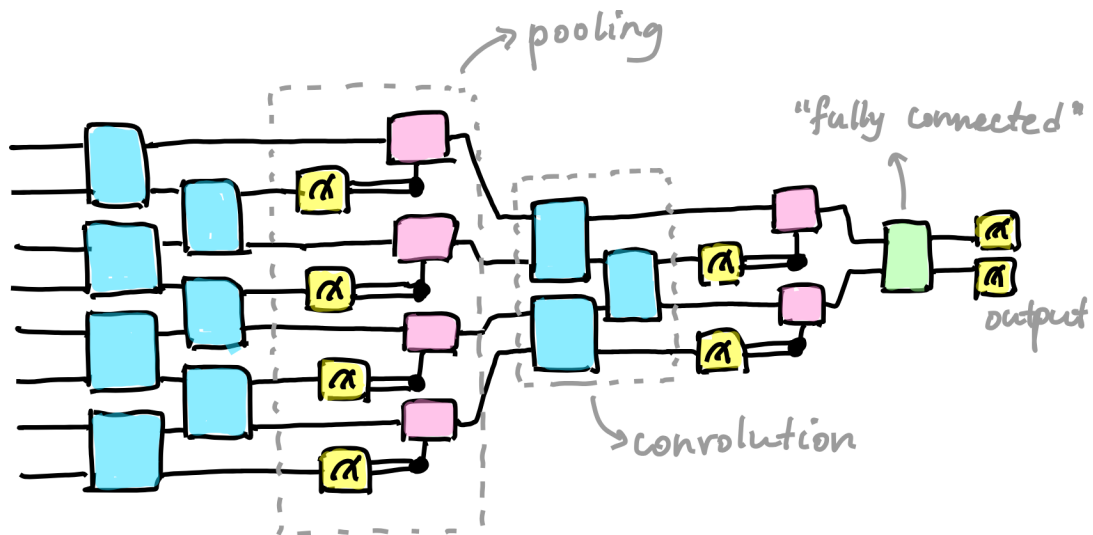


Figura 18: Diagrama da rede neural convolucional quântica. Fonte: <https://pennylane.ai/qml/glossary/qcnn>

4.7 Aula 7 - Introdução a Redes Neurais Quânticas II

4.7.1 Desafios

4.8 Aula 8 - Apresentação de Projetos

4.8.1 Link do notebook Google Colab

https://colab.research.google.com/drive/1GBNZtN6EwR6E9bhF3oCt3D58e_DGiwPh?usp=sharing