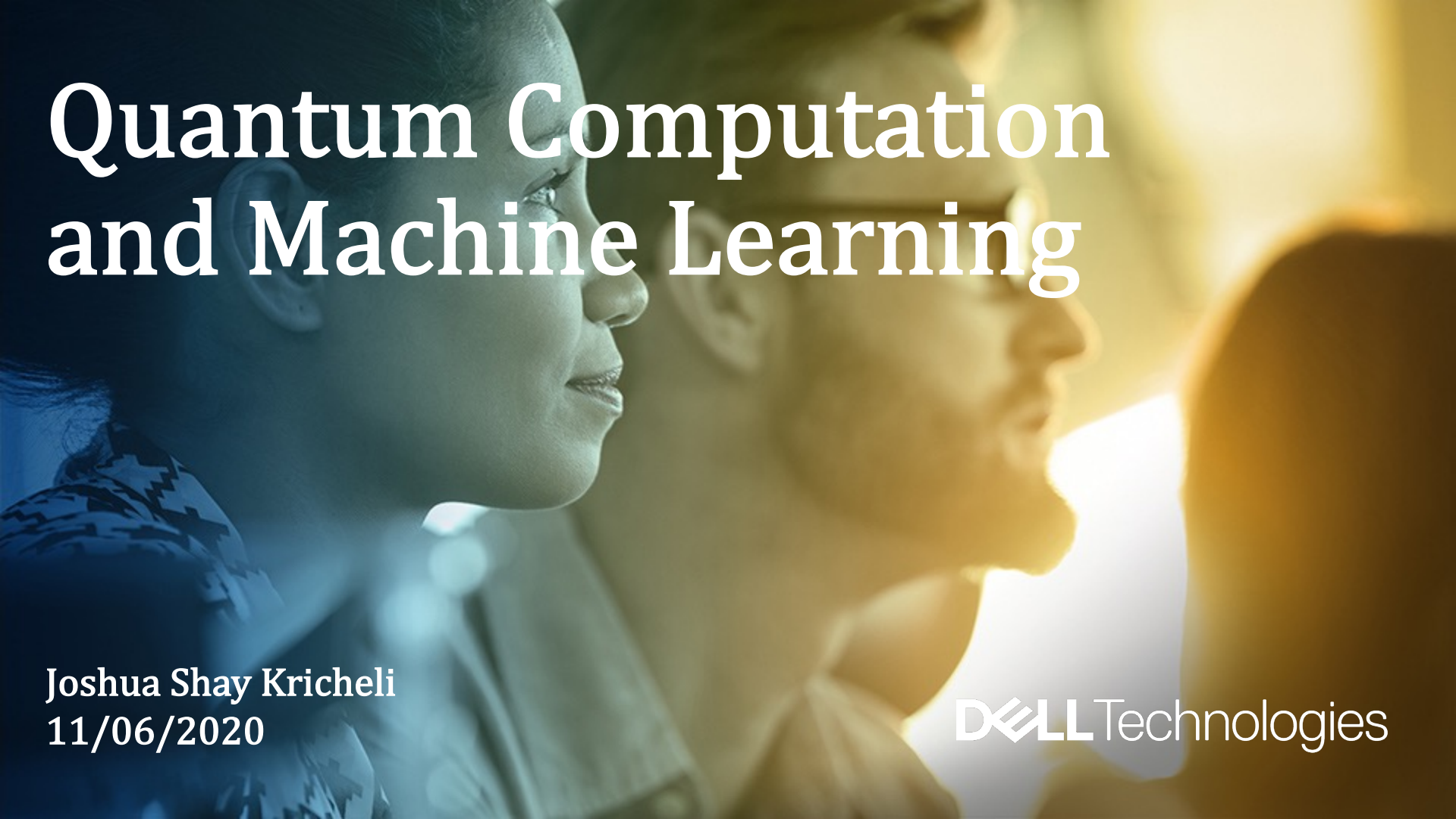
Quantum Computation and Machine Learning



Joshua Shay Kricheli
11/06/2020

DELL Technologies

Physics of Computer Science

The computers we use today are based on principals from Physics.

Principals from Electrical, Mechanical, Materials Engineering and good old-fashioned straight up Physics are involved in the effort of putting all the pieces together for a computer to work.

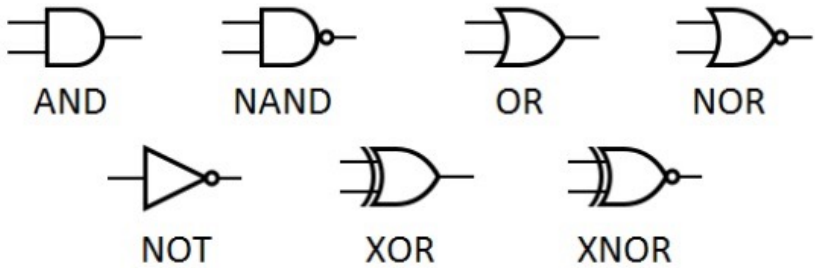
After all this effort – the reward we get is that computers are able to perform calculations faster and more accurately than us humans can.

But still, even computers have their limit.

What are those limits?

Our computers as we know them today are based on principals of classical Physics – as computation is achieved by performing arithmetic operations using various combinations of simple logic gates.

These gates receive classical bits as inputs, perform a certain calculation upon them and return several bits as output. Related fields in computer science have set limits on how fast a calculation can occur using only classical gates.



What is a Logic Gate: A Beginner's Guide, https://www.edrawmax.com/logic-gate/

Computation Theory

Computation theory is a branch of mathematics and computer science that deals with some of these limits and with that – provides insights on what can be calculated and just how fast can a computation work with regards to its input.

Using tools from computation theory, one can claim and prove lower and upper bounds on running times of algorithms with respect to their corresponding input.

Run-Time Complexity

We use the big-$\boldsymbol{O}$ and big-$\boldsymbol{\Omega}$ notations to denote run-time complexity upper and lower bounds respectively. Roughly speaking:

Let us consider an algorithm $\mathcal{A}$. For an input of length n, we say that $\mathcal{A}$ has a run-time complexity of $O(f(n))$ if $\mathcal{A}$ perfromes a number of calculations which is **at most** as the order of the value $f(n)$, and we say that $\mathcal{A}$ has a run-time complexity of $\Omega(g(n))$ if $\mathcal{A}$ perfromes a number of calculations which is **at least** as the order of the value $g(n)$.

$$O$$

$$\Omega$$

Classical Complexity of algorithms

Let us observe the complexity of several commonly used algorithms (assuming input of length n):

- **Unsorted Array Search** - $\Omega(n)$ – linear complexity
- **Array Sort** (without any assumptions), **Fast Fourier Transform**, - $\Omega(n\log n)$ – linearithmic
- **Matrix Gaussian Elimination, Multiplication, Inversion, Determinant calculation, Solving a Linear System, Finding Eigenvalues, Least Squares** - $O(n^3)$ – cubic
- **Prime Number Factorization** - $O((1+\varepsilon)^{\log n})$ – sub-exponential

Some of these have more efficient quantum counterparts

Now, What does Quantum mean?

In physics, a ***quantum*** (plural *quanta*) is the minimum amount of any physical entity (physical property) involved in an interaction. This means that the magnitude of the physical property can take on only discrete values consisting of integer multiples of one quantum.

For example, a photon is a single quantum of light (or of any other form of electromagnetic radiation). Similarly, the energy of an electron bound within an atom is quantized and can exist only in certain discrete values.

And what about Quantum Mechanics?

Quantum mechanics describes physical properties of nature on an atomic scale.

Classical physics, the description of physics that existed before the theory of relativity and quantum mechanics, describes many aspects of nature at an ordinary (macroscopic) scale, while quantum mechanics explains the aspects of nature at small (atomic and subatomic) scales, for which classical mechanics is insufficient, as it cannot encapsulate the phenomena occurring at such small scale.

How Does It All Work?

Delving a bit deeper, let us try to explain the quantum basic unit of computation – the qubit: The **qubit** (short for 'quantum bit') is the fundamental information carrying unit used in quantum computers. It can be seen as the quantum mechanical generalization of a bit used in classical computers. More precisely, a qubit is a two dimensional quantum system. The state of a qubit can be expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

Here α and β are complex numbers such that $|\alpha|^2 + |\beta^2| = 1$. In the **ket-notation** or the **Dirac** notation:

$$|0\rangle = \begin{pmatrix}1\\ 0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\ 1\end{pmatrix}$$

are shorthands for the vectors encoding the two basis states of a two-dimensional vector space.

System of Qubits

The mathematical structure of a qubit generalizes to higher dimensional quantum systems as well. The state of any quantum system is a normalized vector (a vector of norm one) in a complex vector space. The normalization is necessary to ensure that the total probability of all the outcomes of a measurement sum to one.

A quantum computer contains many number of qubits. So it is necessary to know how to construct the combined state of a system of qubits given the states of the individual qubits.

The joint state of a system of qubits is described using an operation known as the tensor product. Given $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$:

$$|\psi\phi\rangle = |\psi\rangle \otimes |\phi\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix}$$

Qubit Measurement

Unlike a classical bit the state of a qubit cannot be measured without changing it. **Measuring** a qubit will yield the classical value of either zero ($|0\rangle$) with probability $|\alpha|^2$ or one ($|1\rangle$) with probability $|\beta^2|$.

Measurement corresponds to transforming the quantum information (stored in a quantum system) into classical information. For example, measuring a qubit typically corresponds to reading out a classical bit, i.e., whether the qubit is 0 or 1.

A central principle of quantum mechanics is that measurement outcomes are **probabilistic**.

Qubit Transformations

A qubit or a system of qubits changes its state by going through a series of **unitary transformations**. A unitary transformation is described by a matrix U with complex entries. The matrix U is called unitary if:

$$UU^{\dagger} = U^{\dagger}U = I$$

Where $U^{\dagger}$ is the transposed, complex conjugate of U (called its Hermitian conjugate) and I is the identity matrix. Operators acting on different qubits can be combined using the Kronecker product. For an n qubit system the set of physically allowed transformations, excluding measurements , consists of all $2^n \times 2^n$ unitary matrices. In practice a transformation on n qubits is effected by using a combination of unitary transformations that act only on one or two qubits at a time. By analogy to classical logic gates like NOT and AND, such basic unitary transformations, which are used to build up more complicated n qubit transformations, are called **quantum gates**.

Moving on to Quantum Computation

Quantum computers use effects such as quantum coherence and entanglement to process information in ways that classical computers can not. Using quantum gates, quantum algorithms can in principal outperform the best known classical algorithms when solving certain problems. This is known as a **quantum speedup.**

Quantum Speedup

Quantum Speedup examples:

- **Unsorted Array Search: Grover's Algorithm** - $O(\sqrt{n})$ – square root speedup

- **Fast Fourier Transform: QFFT** - $O(\log^2 n)$ – exponential speedup

- **Prime Number Factorization** - $O(\log^3 n)$ – sub-exponential speedup

- **Least Squares, Matrix Operations, Finding Eigenvalues, Solving Linear System** - exponential speedup

Quantum Speedup of ML subroutines

Method	Speedup
Bayesian Inference [107, 108]	$O(\sqrt{N})$
Online Perceptron [109]	$O(\sqrt{N})$
Least squares fitting [9]	$O(\log N^{(*)})$
Classical BM [20]	$O(\sqrt{N})$
Quantum BM [22, 62]	$O(\log N^{(*)})$
Quantum PCA [11]	$O(\log N^{(*)})$
Quantum SVM [13]	$O(\log N^{(*)})$
Quantum reinforcement learning [30]	$O(\sqrt{N})$

In the above table, speedups are taken with respect to their classical counterpart(s) — hence, O(√N) means quadratic speedup and O(log(N)) means exponential relative to their classical counterpart.
(∗) denotes important caveats that can limit applicability of method

References

1. Quantum Machine Learning - Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, Seth Lloyd
2. Obtaining the Quantum Fourier Transform from the classical FFT with QR decomposition - F.L.Marquezino, R.Portugal, F.D.Sasse
3. A quantum algorithm providing exponential speedincrease for finding eigenvalues and eigenvectors - Daniel S. Abrams, Seth Lloyd
4. Quantum Data-Fitting, Nathan Wiebe, Daniel Braun and Seth Lloyd
5. Quantum Algorithm Implementations for Beginners, arXiv:1804.03719v2
6. Class Lectures by Or Sattath, Quantum Computing, BGU
7. Quantum, Quantum Mecahnics, Wikipedia