Quantum-enhanced machine learning: Implementing a quantum k-nearest neighbour algorithm

Bachelor thesis defense

19. January 2017

Mark Fingerhuth

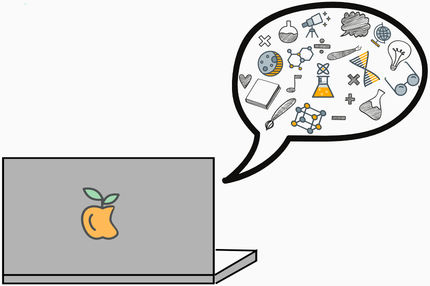
Maastricht Science Programme, Maastricht University, The Netherlands

Thesis work at the Centre for Quantum Technology, UKZN, South Africa

Table of contents

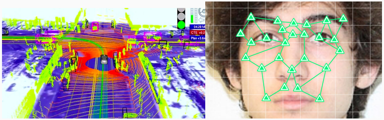
Introduction

Enhancing machine learning with quantum mechanics



Machine learning

Enable computers to learn from data

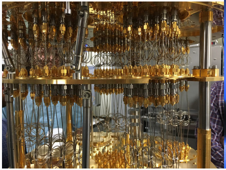


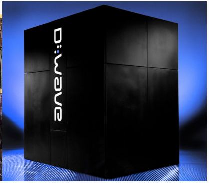
Source: IEEE Spectrum

- ML algorithms often involve[1]
  - solving large systems of linear equations
  - inverting large matrices
  - distance computations
- Performing these computations on large data sets gets increasingly difficult[2]

[1] Bishop, C. M. (2006). Pattern recognition. Machine Learning, 128 .
[2] Bekkerman, R., Bilenko, M., & Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

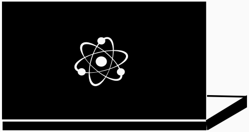
Enhancing machine learning with quantum mechanics





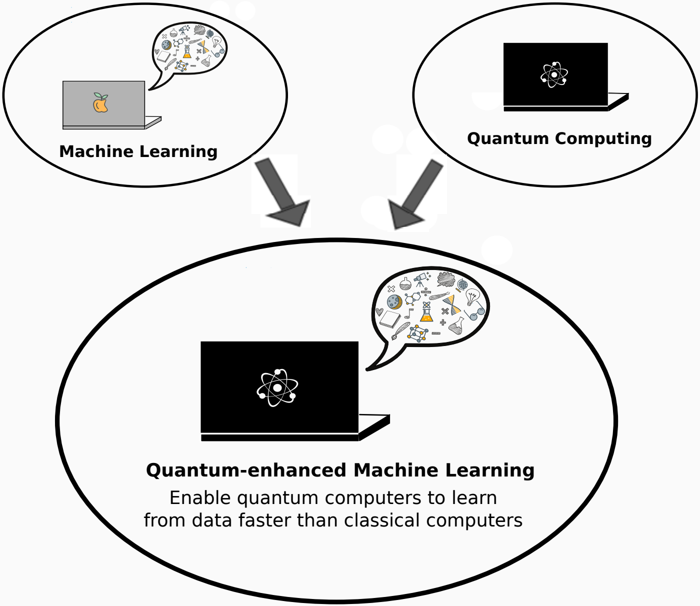
Sources: DWave, IBM

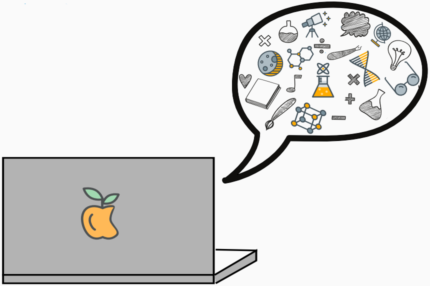
- Quantum mechanics is about vectors in complex Hilbert spaces
- Quantum computers are performing linear operations on qubits
- Many-qubit systems are described by large vectors that can be manipulated in parallel on quantum computers
- Machine learning involves manipulation of large vectors and matrices



Quantum computing
Build computer hardware based on quantum physics

Enhancing machine learning with quantum mechanics





Machine Learning

Supervised machine learning

The problem statement

Given a dataset of inputs and their corresponding outputs, predict the output of a new unknown input.

Input	Output
heartbeat	healthy or sick
last year's oil prices	tomorrow's oil price
message of a users	intention of text content
search history of a user	chance of clicking on a particular ad

1

Supervised machine learning: concrete example

ID	Colour	Class label
1		red
2		red
3		red
4		blue
5		blue
6		blue

Table 1: Example training dataset.

ID	Colour	Class label
1		?
2		?

Table 2: Example input dataset.

1

Classical k-nearest neighbour

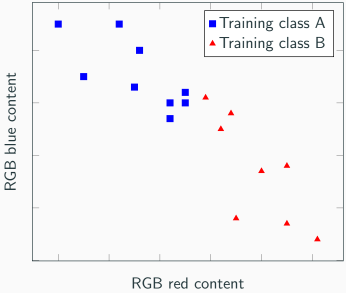
Transferring the colours into vectors

In the case of 9-bit RGB colours:

$$\color{red}{000}\ \color{green}{000}\ \color{blue}{000}$$

3 bits for red, 3 bits for green and 3 bits for blue.

$$\begin{pmatrix} \color{red}{\mathit{red}} & \color{red}{\mathit{content}} \\ \cancel{\color{green}{\mathit{green}}} & \cancel{\color{green}{\mathit{content}}} \\ \color{blue}{\mathit{blue}} & \color{blue}{\mathit{content}} \end{pmatrix} = \begin{pmatrix} 7 \\ \cancel{0} \\ 0 \end{pmatrix} \equiv \begin{pmatrix} 7 \\ 0 \end{pmatrix} \tag{1}$$



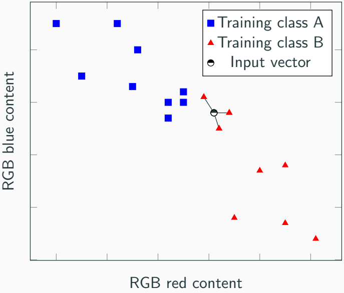
Classical k-nearest neighbour

- kNN is a non-parametric classifier
- k is a positive integer, usually chosen small

Given training dataset:
$D_T = v_0, v_1, .., v_{16}$
$v_i \in \{red, blue\}$

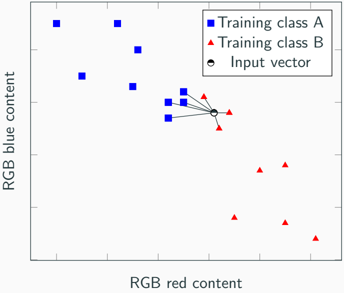
Given a new vector $\tilde{x}$ (black halfcircle):
- consider k nearest neighbours
- classify $\tilde{x}$, based on majority vote, as *red* or *blue*

$k = 3$



Classification $\rightarrow$ **RED**

$k = 7$



Classification $\rightarrow$ **BLUE**

1

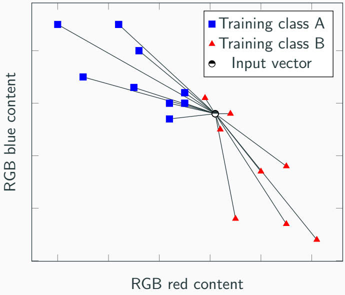
Classical k-nearest neighbour with distance-weighting

- kNN is a non-parametric classifier
- k is a positive integer, usually chosen small

Given training dataset:
$D_T = v_0, v_1, .., v_{16}$
$v_i \in \{red, blue\}$

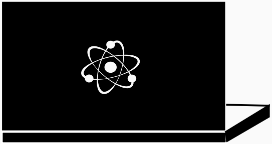
Given a new vector $\tilde{x}$ (black halfcircle):
- consider k nearest neighbours
- classify $\tilde{x}$, based on majority vote, as *red* or *blue*

$k = all$



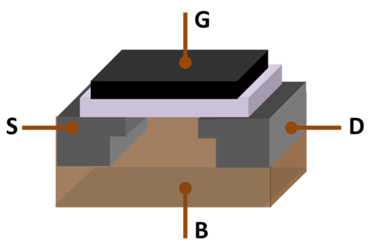
Input vector will simply be classified as the class with the most members.
Classification $\rightarrow$ **BLUE**

Assign distance-dependent weights e.g. $\frac{1}{\text{distance}}$ to increase the influence of close vectors over more distant ones!
Classification $\rightarrow$ **RED**



Quantum Computing

Classical vs. quantum bits (qubits)



Source: Wikipedia

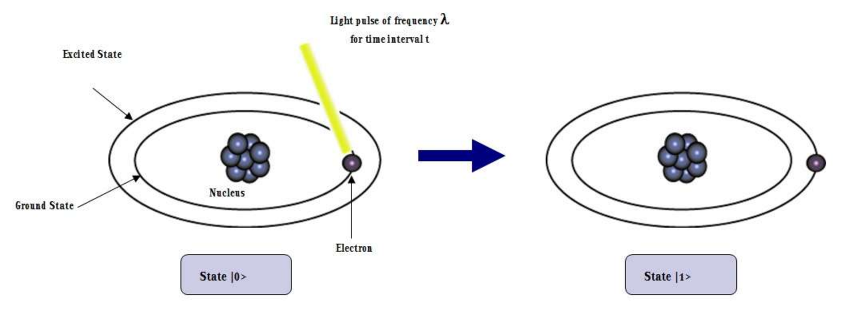
Classical bit:

- Usually implemented through MOSFETs
- 2 definite states (0,1)
- Can be either 0 OR 1

Classical vs. quantum bits (qubits)

Quantum bit (qubit):

- Can be $|0\rangle$ OR $|1\rangle$
- But it can also be $|0\rangle$ AND $|1\rangle$ $\rightarrow$ quantum superposition



Source: RF Wireless World

Qubits

Mathematically, a qubit state is expressed as:

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

where $\alpha, \beta \in \mathbb{C}$ and they are called amplitudes.

$|0\rangle$ and $|1\rangle$ can be represented as the 2-D vectors:

$$|0\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle \doteq \begin{pmatrix} 0 \\ 1 \end{pmatrix} . \tag{3}$$

Substituting into Eq. 2 yields the vector representation of $|\psi\rangle$:

$$|\psi\rangle \doteq \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} . \tag{4}$$

This object is called the **amplitude vector**.

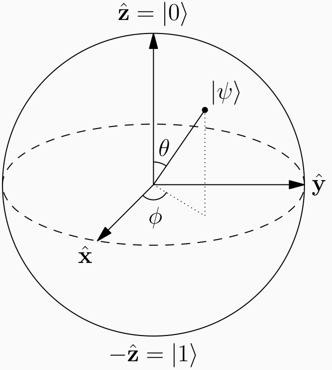


Figure 1: Arbitrary two-dimensional qubit $|\psi\rangle$ visualized on the Bloch sphere[1]

Most general form of a 2-D qubit:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \ , \qquad (5)$$

where $\alpha, \beta \in \mathbb{C}$.

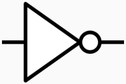
Can also be visualized in spherical polar coords on the unit or Bloch sphere as follows:

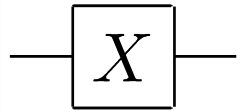
$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \ , \ (6)$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$

[1] Reprinted from Wikipedia, n.d., Retrieved September 7, 2016, from https://en.wikipedia.org/wiki/Bloch_Sphere. Copyright 2012 by Glosser.ca. Reprinted with permission.

Single-qubit quantum logic gates





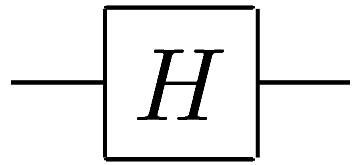
Any single-qubit quantum logic gates can be represented by a unitary 2×2 matrix whose action on a qubit is defined as:

$$U|\psi\rangle \doteq \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{pmatrix}. \tag{7}$$

- Quantum computers perform linear (unitary) operations on qubits
- A quantum computation is the manipulation of an amplitude vector with a matrix representing a quantum logic gate

1

Single-qubit quantum logic gates: Hadamard gate



A very important single-qubit quantum logic gate is the **Hadamard** gate. It is represented by the matrix:

$$H \doteq \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} . \tag{8}$$

Consider acting the H gate on the $|0\rangle$ state:

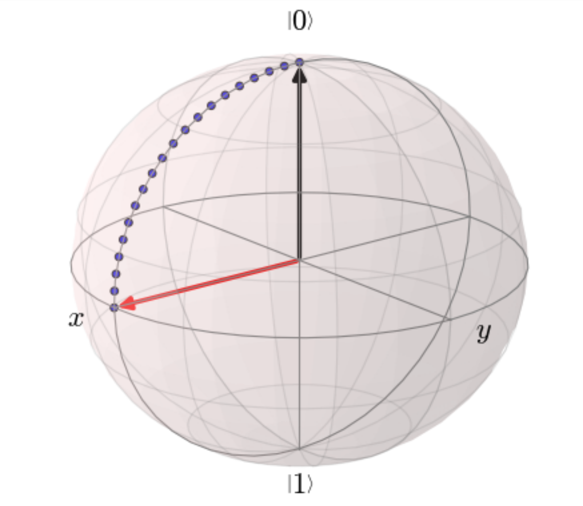
$$H|0\rangle \doteq \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \doteq \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle . \tag{9}$$

$\rightarrow$ **creates an equal superposition of $|0\rangle$ and $|1\rangle$!**

1

Single-qubit quantum logic gates: Hadamard gate

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \ . \tag{10}$$



1

Multi-qubit systems

Tensor products are required when combining several qubits.
For example, the tensor product of two $|0\rangle$ kets is defined as:

$$|0\rangle \otimes |0\rangle = |00\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{11}$$

And for three $|0\rangle$ kets:

$$|00\rangle \otimes |0\rangle = |000\rangle \doteq \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{12}$$

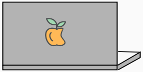
Multi-qubit systems

- It follows, that an n-qubit system has 2^n amplitudes.
  $\rightarrow$ can be used to store huge amounts of information!
- Equivalently, to simulate an n-qubit quantum computer one needs to store the value of all 2^n amplitudes:
  $\rightarrow$ requires $2^n \cdot 64$ classical bits of random-access memory (RAM).

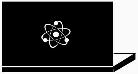
For example:

- 25 qubits can be simulated with just 2 gigabytes of RAM.
- However, 50 qubits require $\sim$8000 terabytes of RAM
- Lastly, 275 qubits have $2^{275} \approx 6 \times 10^{82}$ amplitudes which is roughly equal to the number of atoms in our universe

Three random bits vs. three qubits

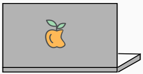


Probability	Combination
p_1	
p_2	
p_3	
.	.
.	.
.	.
p_8	

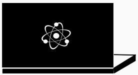


Probability	Quantum state
p_1	
p_2	
p_3	
.	.
.	.
.	.
p_8	

Three random bits vs. three qubits

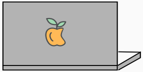


Probability	Combination
p_1	000
p_2	010
p_3	001
.	.
.	.
.	.
p_8	111

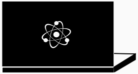


Probability	Quantum state
p_1	$\|000\rangle$
p_2	$\|010\rangle$
p_3	$\|001\rangle$
.	.
.	.
.	.
p_8	$\|111\rangle$

Three random bits vs. three qubits

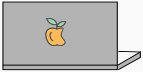


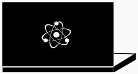
Prob.	Combination
$p_1 = \frac{1}{8}$	000
$p_2 = \frac{1}{8}$	010
$p_3 = \frac{1}{8}$	001
.	.
.	.
.	.
$p_8 = \frac{1}{8}$	111



Prob.	Amplitude	Quantum state
$p_1 = \mid a_1 \mid^2$	a_1	$\|000\rangle$
$p_2 = \mid a_2 \mid^2$	a_2	$\|010\rangle$
$p_3 = \mid a_3 \mid^2$	a_3	$\|001\rangle$
.	.	.
.	.	.
.	.	.
$p_8 = \mid a_8 \mid^2$	a_8	$\|111\rangle$

Three random bits vs. three qubits





Prob.	Combination
$p_1 = \frac{1}{8}$	000
$p_2 = \frac{1}{8}$	010
$p_3 = \frac{1}{8}$	001
.	.
.	.
.	.
$p_8 = \frac{1}{8}$	111

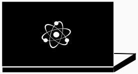
Prob.	Amplitude	Quantum state
$p_1 = \lvert \frac{1}{2\sqrt{2}} \rvert^2 = \frac{1}{8}$	$\frac{1}{2\sqrt{2}}$	$\lvert 000\rangle$
$p_2 = \frac{1}{8}$	$-\frac{1}{2\sqrt{2}}$	$\lvert 010\rangle$
$p_3 = \frac{1}{8}$	$-\frac{i}{2\sqrt{2}}$	$\lvert 001\rangle$
.	.	.
.	.	.
.	.	.
$p_8 = \frac{1}{8}$	$\frac{i}{2\sqrt{2}}$	$\lvert 111\rangle$

$\rightarrow$ In quantum mechanics amplitudes can be interferred with each other!

$\rightarrow$ This is impossible to do on a classical computer!

Three random bits vs. three qubits

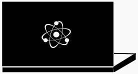
Applying an H gate to the first qubit leads to quantum interference such that:



Prob.	Amplitude	Quantum state
$p_1 = \mid a_1 + a_5 \mid^2$	$a_1 + a_5$	$\vert 000\rangle$
$p_2 = \mid a_2 + a_6 \mid^2$	$a_2 + a_6$	$\vert 010\rangle$
$p_3 = \mid a_3 + a_7 \mid^2$	$a_3 + a_7$	$\vert 001\rangle$
$p_4 = \mid a_4 + a_8 \mid^2$	$a_4 + a_8$	$\vert 011\rangle$
$p_5 = \mid a_1 - a_5 \mid^2$	$a_1 - a_5$	$\vert 100\rangle$
$p_6 = \mid a_2 - a_6 \mid^2$	$a_2 - a_6$	$\vert 110\rangle$
$p_7 = \mid a_3 - a_7 \mid^2$	$a_3 - a_7$	$\vert 101\rangle$
$p_8 = \mid a_4 - a_8 \mid^2$	$a_4 - a_8$	$\vert 111\rangle$

Three random bits vs. three qubits

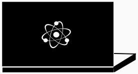
For example, substituting the values for $a_1 = \frac{1}{2\sqrt{2}}$ and $a_5 = \frac{1}{2\sqrt{2}}$ yields:



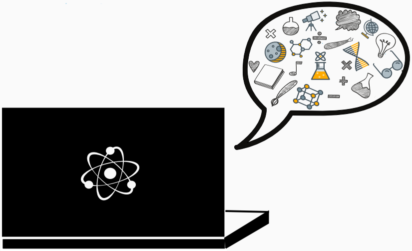
Prob.	Amplitude	Quantum state
$p_1 = \mid a_1 + a_5 \mid^2$	$\frac{1}{\sqrt{2}}(\frac{1}{2\sqrt{2}} + \frac{1}{2\sqrt{2}})$	$\|000\rangle$
$p_2 = \mid a_2 + a_6 \mid^2$	$a_2 + a_6$	$\|010\rangle$
$p_3 = \mid a_3 + a_7 \mid^2$	$a_3 + a_7$	$\|001\rangle$
$p_4 = \mid a_4 + a_8 \mid^2$	$a_4 + a_8$	$\|011\rangle$
$p_5 = \mid a_1 - a_5 \mid^2$	$\frac{1}{\sqrt{2}}(\frac{1}{2\sqrt{2}} - \frac{1}{2\sqrt{2}})$	$\|100\rangle$
$p_6 = \mid a_2 - a_6 \mid^2$	$a_2 - a_6$	$\|110\rangle$
$p_7 = \mid a_3 - a_7 \mid^2$	$a_3 - a_7$	$\|101\rangle$
$p_8 = \mid a_4 - a_8 \mid^2$	$a_4 - a_8$	$\|111\rangle$

Three random bits vs. three qubits

For example, substituting the values for $a_1 = \frac{1}{2\sqrt{2}}$ and $a_5 = \frac{1}{2\sqrt{2}}$ yields:



Prob.	Amplitude	Quantum state	
$p_1 = \mid \frac{1}{2} \mid^2 = \frac{1}{4}$	$\frac{1}{2}$	$\lvert 000 \rangle$	$\rightarrow$ constructive interference
$p_2 = \mid a_2 + a_6 \mid^2$	$a_2 + a_6$	$\lvert 010 \rangle$	"
$p_3 = \mid a_3 + a_7 \mid^2$	$a_3 + a_7$	$\lvert 001 \rangle$	"
$p_4 = \mid a_4 + a_8 \mid^2$	$a_4 + a_8$	$\lvert 011 \rangle$	"
$p_5 = \mid 0 \mid^2 = 0$	0	$\lvert 100 \rangle$	$\rightarrow$ destructive interference
$p_6 = \mid a_2 - a_6 \mid^2$	$a_2 - a_6$	$\lvert 110 \rangle$	"
$p_7 = \mid a_3 - a_7 \mid^2$	$a_3 - a_7$	$\lvert 101 \rangle$	"
$p_8 = \mid a_4 - a_8 \mid^2$	$a_4 - a_8$	$\lvert 111 \rangle$	"



Quantum-enhanced Machine Learning

Encoding classical data into qubits

There are two fundamentally different ways for state preparation:

Data encoded into qubits

k-dimensional probability vector requires $4k$ classical bits which are encoded one-to-one into $4k$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} * 10 \rightarrow \begin{pmatrix} 6 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 0110 \\ 0100 \end{pmatrix} \rightarrow n = 01100100 \rightarrow |n\rangle = |01100100\rangle$$

Data encoded into amplitudes

k-dimensional probability vector is encoded into $log_2(k)$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle$$

1

Encoding classical data into amplitudes

There are two fundamentally different ways for state preparation:

Data encoded into qubits

k-dimensional probability vector requires $4k$ classical bits which are encoded one-to-one into $4k$ qubits, e.g.

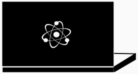
$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} * 10 \rightarrow \begin{pmatrix} 6 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 0110 \\ 0100 \end{pmatrix} \rightarrow n = 01100100 \rightarrow |n\rangle = |01100100\rangle$$

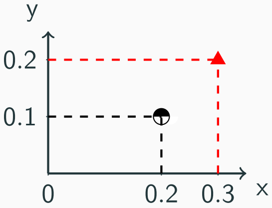
Data encoded into amplitudes

k-dimensional probability vector is encoded into $log_2(k)$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle$$

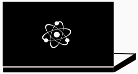
Calculating distances with interference

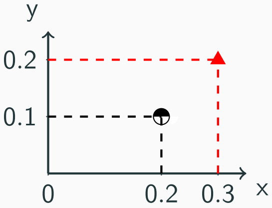




Prob.	Amplitude	Quantum state
$p_1 = 0.22$	$\frac{0.2}{\sqrt{0.18}}$	$\|000\rangle$
$p_2 = 0.055$	$\frac{0.1}{\sqrt{0.18}}$	$\|010\rangle$
$p_3 = 0$	0	$\|001\rangle$
$p_4 = 0$	0	$\|011\rangle$
$p_5 = 0.5$	$\frac{0.3}{\sqrt{0.18}}$	$\|100\rangle$
$p_6 = 0.22$	$\frac{0.2}{\sqrt{0.18}}$	$\|110\rangle$
$p_7 = 0$	0	$\|101\rangle$
$p_8 = 0$	0	$\|111\rangle$

Calculating distances with interference





Prob.	Amplitude	Quantum state
$p_1 = \frac{(0.2+0.3)^2}{0.18}$	$\frac{0.2+0.3}{\sqrt{0.18}}$	$\|000\rangle$
$p_2 = \frac{(0.1+0.2)^2}{0.18}$	$\frac{0.1+0.2}{\sqrt{0.18}}$	$\|010\rangle$
$p_3 = 0$	0	$\|001\rangle$
$p_4 = 0$	0	$\|011\rangle$
$p_5 = \frac{(0.2-0.3)^2}{0.18}$	$\frac{0.2-0.3}{\sqrt{0.18}}$	$\|100\rangle$
$p_6 = \frac{(0.1-0.2)^2}{0.18}$	$\frac{0.1-0.2}{\sqrt{0.18}}$	$\|110\rangle$
$p_7 = 0$	0	$\|101\rangle$
$p_8 = 0$	0	$\|111\rangle$

1

The amplitude-based kNN algorithm

1. $|\psi_0\rangle = \frac{1}{\sqrt{2M}} \sum_{m=1}^{M} (|0\rangle |\Psi_{\tilde{x}(\star)}\rangle + |1\rangle |\Psi_{x^m}\rangle) |y^m\rangle |m\rangle$
   [Initial quantum state]

2. $|\psi_1\rangle = \frac{1}{2\sqrt{M}} \sum_{m=1}^{M} (|0\rangle [|\Psi_{\tilde{x}}\rangle + |\Psi_{x^m}\rangle] + |1\rangle [|\Psi_{\tilde{x}}\rangle - |\Psi_{x^m}\rangle]) |y^m\rangle |m\rangle$
   [Distance computations with quantum interference]

3. $|\psi_2\rangle = \frac{1}{2\sqrt{M}} \sum_{m=1}^{M} \sum_{i=1}^{N} (\tilde{x}_i + x_i^m) |0\rangle |i\rangle |y^m\rangle |m\rangle$
   [Conditional measurement]

4. $\mathrm{Prob}(|y^m\rangle = |1\rangle) = \sum_{m|y^m=1} 1 - \frac{1}{4M} \mid \tilde{x} - x^m \mid^2$
   [Probability to measure a certain class]

5. $y = \begin{cases} 0, & \text{if } \mathrm{Prob}(|y^0\rangle) > \mathrm{Prob}(|y^1\rangle) \\ 1, & \text{if } \mathrm{Prob}(|y^1\rangle) > \mathrm{Prob}(|y^0\rangle) \\ -, & \text{otherwise} \end{cases}$ [Classification]

Schuld, Maria (2016) unpublished.

Calculating distances with interference

1

Methods

Methods: IBM Quantum Experience

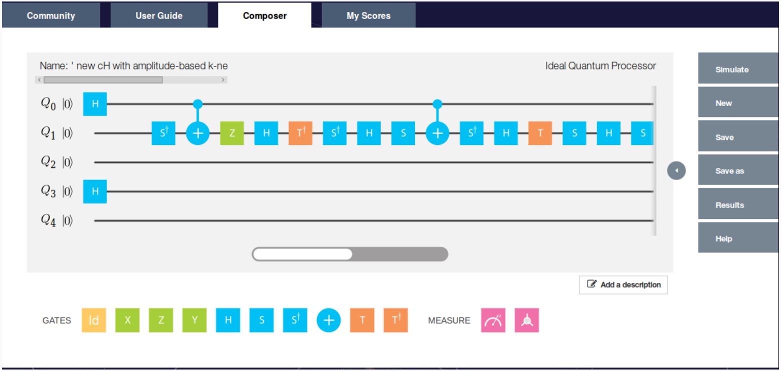


Figure 2: IBM's quantum composer[1]

- Accessible to the public
- Allows for ideal + real simulations
- 5 superconducting qubits
- 40 gates (39 gates + 1 measurement)

[1] Screenshot taken from `https://quantumexperience.ng.bluemix.net/qstage/#/editor`

Methods: Liqui$|\rangle$

Remember: Small quantum computers can still be simulated!

Liqui$|\rangle$...

- stands for Language-Integrated Quantum Operations.
- is a quantum simulation toolsuite written in F# .
- was developed by Microsoft Research.
- allows for simulations of up to 30 qubits with 16GB RAM.

Results: Amplitude-based kNN algorithm

Illustrate the problems with IBMQE?

Show Bloch vector classification simulation results here

1

Simple binary classification case

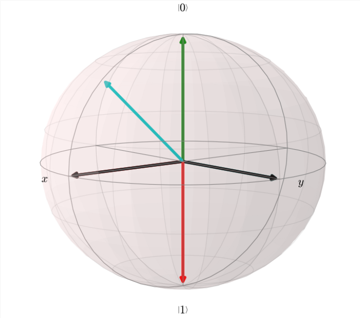


Figure 3: Simple binary classification problem of a quantum state

$$\frac{1}{\sqrt{2M}}\sum_{m=1}^{M}(|0\rangle\,|\Psi_{\tilde{x}}(\star)\rangle + |1\rangle\,|\Psi_{x^m}\rangle)\,|y^m(A \text{ or } B)\rangle\,|m\rangle \tag{13}$$

Procedure to load the input vector $\tilde{x}$:

$$|\Psi_0\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle\,|0\rangle + |1\rangle\,|0\rangle)\,|y^m\rangle\,|m\rangle \tag{14}$$

Apply controlled rotation ${}_0^1CR_y(\frac{\pi}{4})$ s.t.

$${}_0^1CR_y(\frac{\pi}{4})\,|\Psi_0\rangle = |\Psi_1\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle\,|0\rangle+|1\rangle\,|\Psi_{\tilde{x}}\rangle)\,|y^m\rangle\,|m\rangle \tag{15}$$

Flip the ancilla qubit in the first register

$$(X\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1})\,|\Psi_1\rangle = |\Psi_2\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle\,|\Psi_{\tilde{x}}\rangle+|1\rangle\,|0\rangle)\,|y^m\rangle\,|m\rangle \tag{16}$$

IBM's universal gate set

Figure 4: IBM's universal gate set

How can we implement the $^{1}_{0}CR_y(\frac{\pi}{4})$ gate?

Controlled U gate

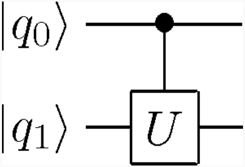


Figure 5: Controlled U-gate

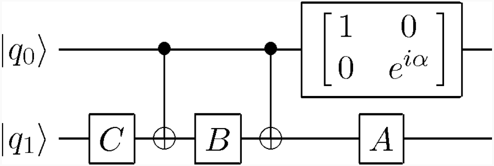


Figure 6: Decomposition of a controlled U-gate[1]

Choose A,B,C and α s.t.

$$e^{i\alpha} * A * X * B * X * C = U \quad and \quad A * B * C = \mathbb{1} \tag{17}$$

Need to solve the following equation[1]

$$U = \begin{pmatrix} e^{i(\alpha - \frac{\beta}{2} - \frac{\delta}{2})} \cos\frac{\gamma}{2} & -e^{i(\alpha - \frac{\beta}{2} + \frac{\delta}{2})} \sin\frac{\gamma}{2} \\ e^{i(\alpha + \frac{\beta}{2} - \frac{\delta}{2})} \sin\frac{\gamma}{2} & e^{i(\alpha + \frac{\beta}{2} + \frac{\delta}{2})} \cos\frac{\gamma}{2} \end{pmatrix} \tag{18}$$

[1] Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.
[2] Jat, R. N., & Ruhela, D. S. (2011). Comparative study of complexity of algorithms for iterative solution of non-linear equations. Journal of International Academy Of Physical Sciences, 15(4).

Problems with universal gate sets

In our case we need to find A, B, C and α for ${}^1_0CR_y(\frac{\pi}{4})$:

Using a root finding algorithm for non-linear equations we find:

$$\alpha = \pi; \quad \beta = 2\pi; \quad \delta = \frac{7}{8}\pi; \quad \gamma = 0 \tag{19}$$

Then,

$$A = R_z(\beta)R_y(\frac{\gamma}{2}) = R_z(2\pi) = \mathbb{1} \tag{20}$$

$$B = R_y(-\frac{\gamma}{2})R_z(-\frac{\delta+\beta}{2}) = R_z(-\frac{23}{16}\pi) = ??? \tag{21}$$

$$C = R_z(\frac{\delta-\beta}{2}) = R_z(-\frac{9}{16}\pi) = ??? \tag{22}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = Z \tag{23}$$

[1] Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

The Solovay-Kitaev theorem

$$B = R_z(-\frac{23}{16}\pi) = ??? \tag{24}$$

$$C = R_z(-\frac{9}{16}\pi) = ??? \tag{25}$$

The Solovay-Kitaev theorem guarantees that given a set of single-qubit quantum gates which generates a dense subset of $SU(2)$, then that set is guaranteed to fill $SU(2)$ quickly.[1]

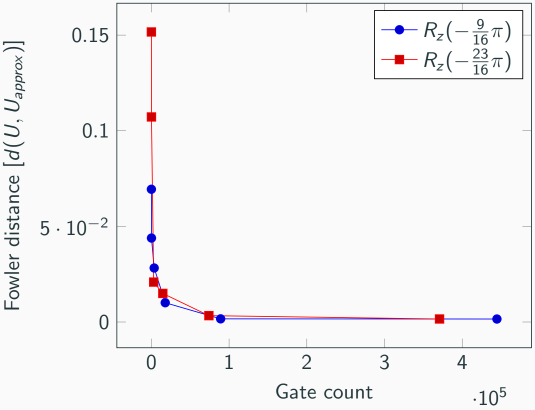
→ **Hence, given any universal gate set it is possible to obtain good approximations to any desired gate.**

→ **But needs to be computed classically!**

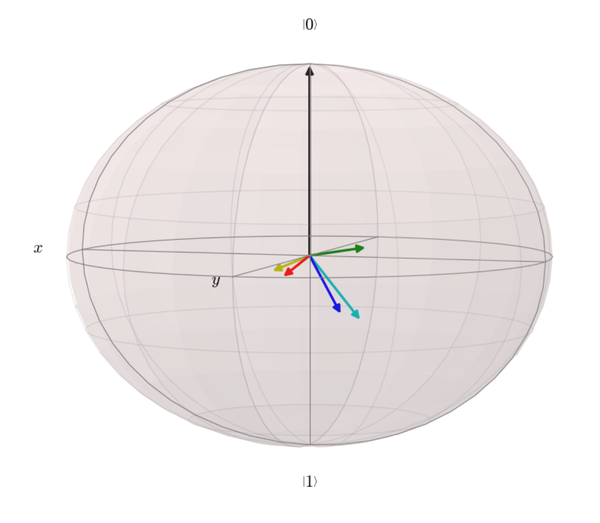
[1] Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

Fowler distance[1]:

$$dist(U, U_{approx}) = \sqrt{\frac{2- \mid tr(U \cdot U_{approx}^{\dagger}) \mid}{2}} \tag{26}$$



[1] Booth Jr, J. (2012). Quantum compiler optimizations. arXiv preprint arXiv:1206.3348.



$$d = 0.22739 \quad (27)$$

$$d = 0.15165 \quad (28)$$

$$d = 0.10722 \quad (29)$$

$$d = 0.02086 \quad (30)$$

$$d = 0.00156 \quad (31)$$

Figure 7: Various Fowler distances visualized on Bloch sphere

The Solovay-Kitaev algorithm

IBM's quantum computer needs **130ns for single-qubit gates** and **500ns for CNOT gates.**

IBM qubit decoherence times:

$49.5\,\mu s \leq T_1 \leq 85.3\,\mu s$ "amplitude damping"
$56.0\,\mu s \leq T_2 \leq 139.7\,\mu s$ "phase damping"

Approx. Gate	Distance	Gate count	Execution time
$R_z(-\frac{23}{16}\pi)$	0.15165	25	$\sim 3\,\mu s$
	0.10722	109	$\sim 14\,\mu s$
	0.02086	2997	$\sim 390\,\mu s$
	0.01494	14721	$\sim 1914\,\mu s$
	0.003327	74009	$\sim 9621\,\mu s$
	0.001578	370813	$\sim 48\,206\,\mu s$

Table 3: SK algorithm results

Liqui$|\rangle$ simulations

Currently impossible to implement the quantum algorithm on IBM's quantum computer! $\rightarrow$ can only simulate the algorithm with Liqui$|\rangle$

In Liqui$|\rangle$, one can directly implement the controlled R_y rotation!

Vector representation	Prob(CM)	Prob ($\|c\rangle = \|0\rangle$)	Prob ($\|c\rangle = \|1\rangle$)	Expected class	A
$e^{-i\frac{\pi}{8}}\begin{pmatrix}0.92388\\0.38268\end{pmatrix}$	$\frac{0.8266^*}{0.8050}$	$\frac{0.5818^*}{0.5863}$	$\frac{0.4182^*}{0.4137}$	$\|0\rangle$	
$e^{-i\frac{\pi}{16}}\begin{pmatrix}0.98079\\0.19509\end{pmatrix}$	$\frac{0.7940^*}{0.7710}$	$\frac{0.6237^*}{0.6420}$	$\frac{0.3763^*}{0.3580}$	$\|0\rangle$	

Conclusion

Summary

- Initial state preparation is non-trivial! (even for very simple examples)
- In this case, state preparation dominates the overall algorithmic complexity
- Solovay-Kitaev yields long gate sequences for good approximations
- Some universal gate sets are only useful when combined with long qubit lifetimes
- **Need for better quantum compiling and more general state preparation algorithms!**

Taking it further

- Complexity analysis of the qubit-based kNN algorithm
- Classification of gaussian probability distributions
- Implementing more general state preparation algorithms
- Waiting for IBM QASM 2.0 ...

References

Bekkerman, R., Bilenko, M., & Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

Booth Jr, J. (2012). Quantum compiler optimizations. arXiv preprint arXiv:1206.3348.

Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

IBM. (2016). What is big data? https://www-01.ibm.com/software/data/bigdata/what-is-big -data.html. (Accessed: 2016-09-08)

Jat, R. N., & Ruhela, D. S. (2011). Comparative study of complexity of algorithms for iterative solution of non-linear equations. Journal of International Academy Of Physical Sciences, 15(4).

Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

Questions?

Results: Qubit-based kNN algorithm

1

Quantum Computing & Qubits

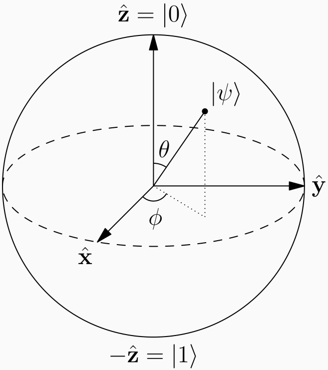


Figure 8: Arbitrary two-dimensional qubit $|\psi\rangle$ visualized on the Bloch sphere[1]

Most general form of a 2-D qubit:

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad (32)$$

where $\alpha, \beta \in \mathbb{C}$.

Can also be visualized in spherical polar coords on the unit or Bloch sphere as follows:

$$|q\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \qquad (33)$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$

[1] Reprinted from Wikipedia, n.d., Retrieved September 7, 2016, from https://en.wikipedia.org/wiki/Bloch_Sphere. Copyright 2012 by Glosser.ca. Reprinted with permission.

Machine Learning

- Approximately 2.5 quintillion (10^{18}) bytes of digital data are created every day[1]
- Need for advanced algorithms that can make sense of data content, retrieve patterns and reveal correlations → Machine learning (ML)
- ML algorithms often involve
  - solving large systems of linear equations
  - inverting large matrices
  - distance computations
- Performing these computations on large data sets gets increasingly difficult[2]

[1] IBM. (2016). What is big data? https://www-01.ibm.com/software/data/bigdata/what-is-big -data.html. (Accessed: 2016-09-08)
[2] Bekkerman, R., Bilenko, M., & Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

Quantum Machine Learning

1. ML involves manipulation of large vectors and matrices
2. Quantum mechanics is about vectors $\in$ complex Hilbert spaces
3. Quantum computers are performing linear operations on qubits

$\rightarrow$ Hence, we can manipulate large vectors in parallel on quantum computers

So can we use QC to improve classical ML algorithms??

- Classical ML is a very practical topic
- BUT, QML has been of almost entirely theoretical nature

Quantum data encoding

There are two fundamentally different ways for state preparation:

Data encoded into qubits

k-dimensional probability vector requires $4k$ classical bits which are encoded one-to-one into $4k$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} * 10 \rightarrow \begin{pmatrix} 6 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 0110 \\ 0100 \end{pmatrix} \rightarrow n = 01100100 \rightarrow |n\rangle = |01100100\rangle$$

Data encoded into amplitudes

k-dimensional probability vector is encoded into $log_2(k)$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle$$

Quantum data encoding

There are two fundamentally different ways for state preparation:

Data encoded into qubits

k-dimensional probability vector requires $4k$ classical bits which are encoded one-to-one into $4k$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} * 10 \rightarrow \begin{pmatrix} 6 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 0110 \\ 0100 \end{pmatrix} \rightarrow n = 01100100 \rightarrow |n\rangle = |01100100\rangle$$

Data encoded into amplitudes

k-dimensional probability vector is encoded into $log_2(k)$ qubits, e.g.

$$\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle$$

Classical k-nearest neighbour

- kNN is a non-parametric classifier
- k is a positive integer, usually chosen small

Given training data set:
$D_T = v_0, v_1, .., v_{10}$
$v_i \in \{A, B\}$

Given a new vector $\tilde{x}$ (red star):
- consider k nearest neighbours
- classify $\tilde{x}$, based on majority vote, as A or B

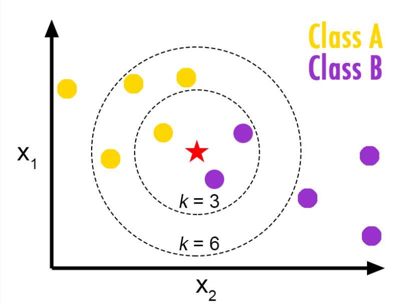


Figure 9: Visualization of a kNN classifier[1]

[1] Reprinted from GitHub, Burton de Wilde, Retrieved September 13, 2016, from http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/. Copyright 2012 by Burton de Wilde. Reprinted with permission.

The algorithm

$$\frac{1}{\sqrt{2M}}\sum_{m=1}^{M}(|0\rangle\,|\Psi_{\tilde{x}}(\star)\rangle + |1\rangle\,|\Psi_{x^m}\rangle)\,|y^m(A \text{ or } B)\rangle\,|m\rangle \tag{34}$$

where

$$|\Psi_{\tilde{x}}(\star)\rangle = \sum_{i=1}^{N} \tilde{x}_i\,|i\rangle \qquad |\Psi_{x^m}\rangle = \sum_{i=1}^{N} x_i^m\,|i\rangle \tag{35}$$

$$e.g. \quad \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle \tag{36}$$

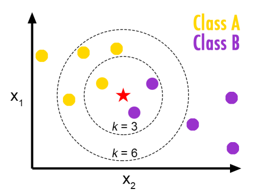


Figure 10: Visualization of a kNN classifier[1]

Maria Schuld (2016), unpublished

The algorithm

Applying the **Hadamard gate** interferes the input and the training vectors:

$$\frac{1}{2\sqrt{M}}\sum_{m=1}^{M}\left(|0\rangle\left[|\Psi_{\tilde{x}}\rangle + |\Psi_{x^m}\rangle\right] + |1\rangle\left[|\Psi_{\tilde{x}}\rangle - |\Psi_{x^m}\rangle\right]\right)|y^m(A \text{ or } B)\rangle\,|m\rangle \quad (37)$$

$\rightarrow$ Perform **conditional measurement** on ancilla qubit.
Successful if $|0\rangle$ state is measured.

Maria Schuld (2016), unpublished

The algorithm

After successful conditional measurement, the state is proportional to

$$\frac{1}{2\sqrt{M}} \sum_{m=1}^{M} \sum_{i=1}^{N} (\tilde{x}_i + x_i^m) |0\rangle |i\rangle |y^m(A \text{ or } B)\rangle |m\rangle \tag{38}$$

Probability to measure class B:

$$p(|y^m\rangle = |1(B)\rangle) = \sum_{m|y^m=1(B)} 1 - \frac{1}{4M} \mid \tilde{x} - x^m \mid^2 \tag{39}$$

Overall algorithmic complexity

$\mathcal{O}(\frac{1}{p_{acc}})$ where p_{acc} is the probability of measuring ancilla in the $|0\rangle$ state

Maria Schuld (2016), unpublished

Simple binary classification case

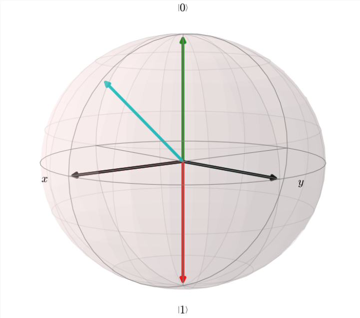


Figure 11: Simple binary classification problem of a quantum state

$$\frac{1}{\sqrt{2M}} \sum_{m=1}^{M} (|0\rangle |\Psi_{\tilde{x}}(\star)\rangle + |1\rangle |\Psi_{x^m}\rangle) |y^m(A \text{ or } B)\rangle |m\rangle \tag{40}$$

Procedure to load the input vector $\tilde{x}$:

$$|\Psi_0\rangle = \frac{1}{2} \sum_{m=1}^{2} (|0\rangle |0\rangle + |1\rangle |0\rangle) |y^m\rangle |m\rangle \tag{41}$$

Apply controlled rotation ${}^1_0CR_y(\frac{\pi}{4})$ s.t.

$${}^1_0CR_y(\frac{\pi}{4}) |\Psi_0\rangle = |\Psi_1\rangle = \frac{1}{2} \sum_{m=1}^{2} (|0\rangle |0\rangle + |1\rangle |\Psi_{\tilde{x}}\rangle) |y^m\rangle |m\rangle \tag{42}$$

Flip the ancilla qubit in the first register

$$(X \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}) |\Psi_1\rangle = |\Psi_2\rangle = \frac{1}{2} \sum_{m=1}^{2} (|0\rangle |\Psi_{\tilde{x}}\rangle + |1\rangle |0\rangle) |y^m\rangle |m\rangle \tag{43}$$

Implementation with IBM's quantum computer

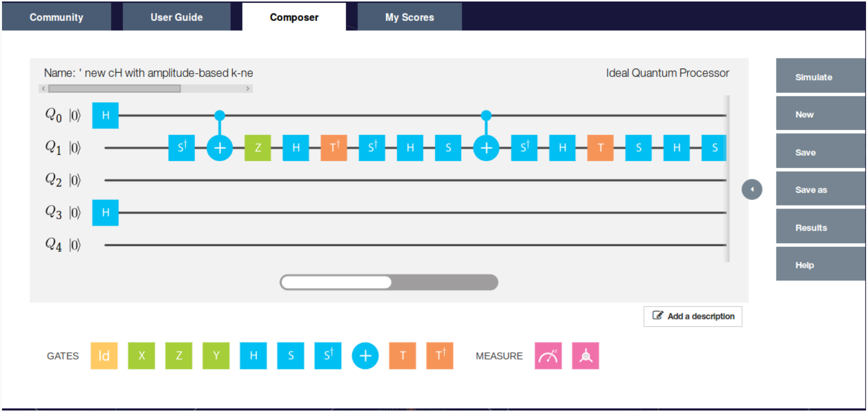


Figure 12: IBM's quantum composer

- Accessible to the public
- Allows for ideal + real simulations
- 5 superconducting qubits
- 40 gates (39 gates + 1 measurement)

IBM's universal gate set

Figure 13: IBM's universal gate set

How can we implement the ${}^{1}_{0}CR_y(\frac{\pi}{4})$ gate?

Controlled U gate

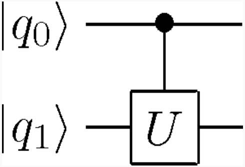


Figure 14: Controlled U-gate

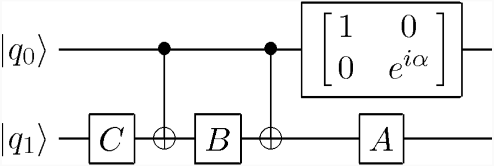


Figure 15: Decomposition of a controlled U-gate[1]

Choose A,B,C and α s.t.

$$e^{i\alpha} * A * X * B * X * C = U \quad and \quad A * B * C = \mathbb{1} \tag{44}$$

Need to solve the following equation[1]

$$U = \begin{pmatrix} e^{i(\alpha - \frac{\beta}{2} - \frac{\delta}{2})} \cos \frac{\gamma}{2} & -e^{i(\alpha - \frac{\beta}{2} + \frac{\delta}{2})} \sin \frac{\gamma}{2} \\ e^{i(\alpha + \frac{\beta}{2} - \frac{\delta}{2})} \sin \frac{\gamma}{2} & e^{i(\alpha + \frac{\beta}{2} + \frac{\delta}{2})} \cos \frac{\gamma}{2} \end{pmatrix} \tag{45}$$

Overall algorithmic complexity

$\mathcal{O}(\frac{1}{p_{acc}}) + \mathcal{O}(k)$ where k is number of root finding iterations[2]

[1] Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

[2] Jat, R. N., & Ruhela, D. S. (2011). Comparative study of complexity of algorithms for iterative solution of non-linear equations. Journal of International Academy Of Physical Sciences, 15(4).

Problems with universal gate sets

In our case we need to find A, B, C and α for ${}^{1}_{0}CR_y(\frac{\pi}{4})$:

Using a root finding algorithm for non-linear equations we find:

$$\alpha = \pi; \quad \beta = 2\pi; \quad \delta = \frac{7}{8}\pi; \quad \gamma = 0 \tag{46}$$

Then,

$$A = R_z(\beta)R_y(\frac{\gamma}{2}) = R_z(2\pi) = \mathbb{1} \tag{47}$$

$$B = R_y(-\frac{\gamma}{2})R_z(-\frac{\delta+\beta}{2}) = R_z(-\frac{23}{16}\pi) = ??? \tag{48}$$

$$C = R_z(\frac{\delta-\beta}{2}) = R_z(-\frac{9}{16}\pi) = ??? \tag{49}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = Z \tag{50}$$

[1] Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

The Solovay-Kitaev theorem

$$B = R_z(-\frac{23}{16}\pi) = ??? \tag{51}$$

$$C = R_z(-\frac{9}{16}\pi) = ??? \tag{52}$$

The Solovay-Kitaev theorem guarantees that given a set of single-qubit quantum gates which generates a dense subset of $SU(2)$, then that set is guaranteed to fill $SU(2)$ quickly.[1]

→ **Hence, given any universal gate set it is possible to obtain good approximations to any desired gate.**

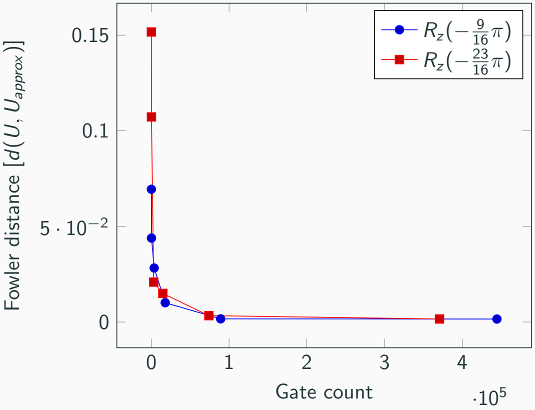
→ **But needs to be computed classically!**

[1] Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

The Solovay-Kitaev algorithm

Fowler distance[1]:

$$dist(U, U_{approx}) = \sqrt{\frac{2 - \mid tr(U \cdot U_{approx}^{\dagger}) \mid}{2}} \tag{53}$$



[1] Booth Jr, J. (2012). Quantum compiler optimizations. arXiv preprint arXiv:1206.3348.

The Solovay-Kitaev algorithm

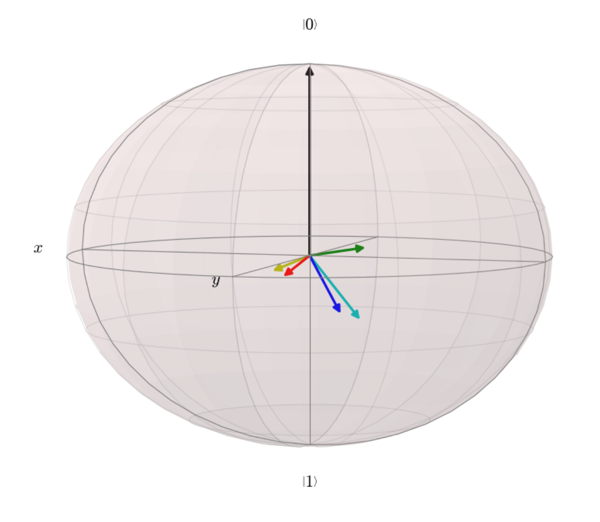


Figure 16: Various Fowler distances visualized on Bloch sphere

$$d = 0.22739 \quad (54)$$

$$d = 0.15165 \quad (55)$$

$$d = 0.10722 \quad (56)$$

$$d = 0.02086 \quad (57)$$

$$d = 0.00156 \quad (58)$$

The Solovay-Kitaev algorithm

IBM's quantum computer needs **130ns for single-qubit gates** and **500ns for CNOT gates.**

IBM qubit decoherence times:

$49.5\,\mu s \leq T_1 \leq 85.3\,\mu s$ "amplitude damping"
$56.0\,\mu s \leq T_2 \leq 139.7\,\mu s$ "phase damping"

Approx. Gate	Distance	Gate count	Execution time
$R_z(-\frac{23}{16}\pi)$	0.15165	25	$\sim 3\,\mu s$
	0.10722	109	$\sim 14\,\mu s$
	0.02086	2997	$\sim 390\,\mu s$
	0.01494	14721	$\sim 1914\,\mu s$
	0.003327	74009	$\sim 9621\,\mu s$
	0.001578	370813	$\sim 48\,206\,\mu s$

Table 4: SK algorithm results

Adding complexities

Executing the SK algorithm adds to our overall algorithmic complexity:

Overall algorithmic complexity

$\mathcal{O}(\frac{1}{p_{acc}}) + \mathcal{O}(k) + \mathcal{O}(m * log^{2.71}(\frac{m}{\epsilon}))$ for ϵ-approximations of m gates[1]

Due to state preparation we went from

$$\mathcal{O}(\frac{1}{p_{acc}}) \tag{59}$$

suddenly to

$$\mathcal{O}(m * log^{2.71}(\frac{m}{\epsilon})) \tag{60}$$

where m is the number of gates that need approximation to ϵ-accuracy

[1] Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

Liqui$|\rangle$ simulations

Currently impossible to implement the quantum algorithm on IBM's quantum computer! $\rightarrow$ can only simulate it with i.e. Liqui$|\rangle$

In Liqui$|\rangle$ we can directly implement the controlled R_y rotation!

Conclusion

Summary

- Initial state preparation is non-trivial! (even for very simple examples)
- In this case, state preparation dominates the overall algorithmic complexity
- Solovay-Kitaev yields long gate sequences for good approximations
- Some universal gate sets are only useful when combined with long qubit lifetimes
- **Need for better quantum compiling and more general state preparation algorithms!**

Taking it further

- Complexity analysis of the qubit-based kNN algorithm
- Classification of gaussian probability distributions
- Implementing more general state preparation algorithms
- Waiting for IBM QASM 2.0 ...

References

Bekkerman, R., Bilenko, M., & Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

Booth Jr, J. (2012). Quantum compiler optimizations. arXiv preprint arXiv:1206.3348.

Dawson, C. M., & Nielsen, M. A. (2005). The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030.

IBM. (2016). What is big data? https://www-01.ibm.com/software/data/bigdata/what-is-big -data.html. (Accessed: 2016-09-08)

Jat, R. N., & Ruhela, D. S. (2011). Comparative study of complexity of algorithms for iterative solution of non-linear equations. Journal of International Academy Of Physical Sciences, 15(4).

Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.

Questions?

Backup slide: Qubit decoherence times

We expect exponential decay:

$$e^{t/T_i} \tag{61}$$

T_1: **Longitudinal coherence time (amplitude damping)**

- Prepare $|0\rangle$ state
- Apply the X (NOT) gate s.t. qubit is in $|1\rangle$ state
- Wait for time t
- Measure the probability of being in $|1\rangle$ state

T_2: **Transversal coherence time (phase damping)**

- Prepare $|0\rangle$ state
- Apply Hadamard $\rightarrow \frac{|0\rangle+|1\rangle}{\sqrt{2}}$
- Wait for time t
- Apply Hadamard again
- Measure the probability of being in $|0\rangle$ state

We expect this probability to go to 0.5 $\rightarrow$ qubit lost quantum behaviour

Chuang I. (2003) MIT Lecture 19: How to Build Your Own Quantum Computer

Backup Slide II: Experimental realizations

Only few experimental verifications of QML algorithms:

- Li, Liu, Xu, and Du (2015) successfully distinguished a handwritten six from a nine using a quantum support vector machine on a four-qubit nuclear magnetic resonance test bench[1]

- Cai et al. (2015) were first to experimentally demonstrate quantum machine learning on a photonic QC and showed that the distance between two vectors and their inner product can indeed be computed quantum mechanically[2]

- Rist et al. (2015) solved a learning parity problem with five superconducting qubits and found that a quantum advantage can already be observed in non error-corrected systems[3]

[1] Li, Z., Liu, X., Xu, N., & Du, J. (2015). Experimental realization of a quantum support vector machine. Physical Review Letters, 114 (14), 15. doi: 10.1103/PhysRevLett.114.140504

[2] Cai, X. D., Wu, D., Su, Z. E., Chen, M. C., Wang, X. L., Li, L., . . . Pan, J. W. (2015). Entanglement- based machine learning on a quantum computer.

Machine Learning

Machine learning can be subdivided into three major fields.

Supervised ML

- Based on *input* and *output* data

"I know how to classify this data but I need the algorithm to do the computations for me."

Unsupervised ML

- Based on *input* data only

"I have no clue how to classify this data, can the algorithm create a classifier for me?"

Reinforcement learning

- Based on *input* data only

"I have no clue how to classify this data, can the algorithm classify this data and I'll give it a reward if it's correct or I'll punish it if it's not."

Machine Learning

Machine learning can be subdivided into three major fields.

Supervised ML

- Based on *input* and *output* data

"I know how to classify this data but I need the algorithm to do the computations for me."

Unsupervised ML

- Based on *input* data only

"I have no clue how to classify this data, can the algorithm create a classifier for me?"

Reinforcement learning

- Based on *input* data only

"I have no clue how to classify this data, can the algorithm classify this data and I'll give it a reward if it's correct or I'll punish it if it's not."

Liqui|⟩ simulations: Taking it further

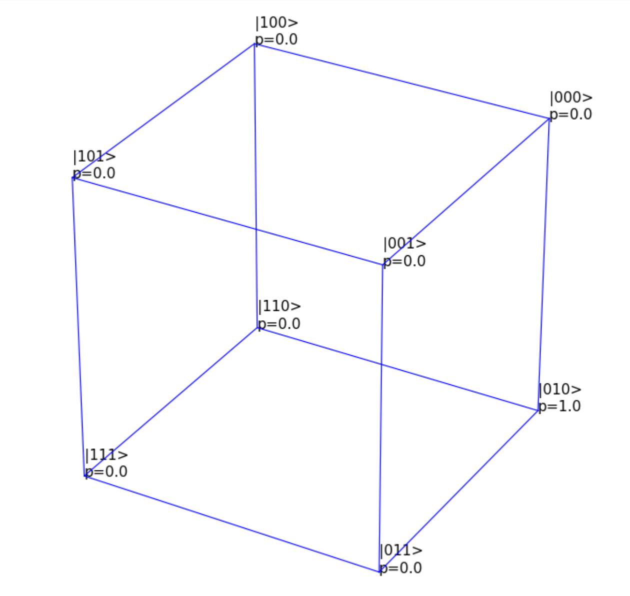


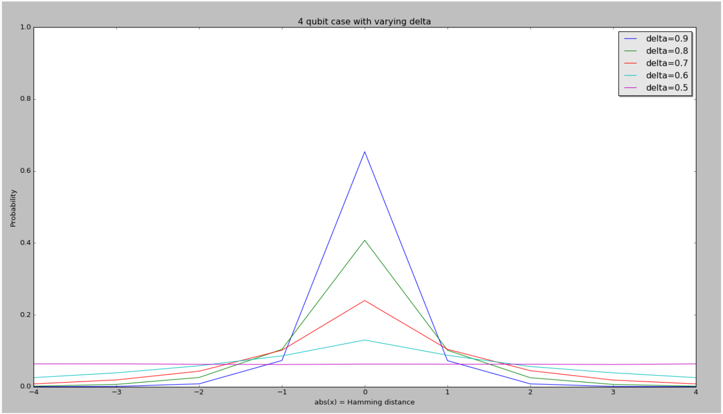
Figure 17: Representation of hamming distance on 3D cube

Liqui|⟩ simulations: Taking it further

Applying the following matrix

$$\begin{pmatrix} \sqrt{\delta} & 1-\sqrt{\delta} \\ 1-\sqrt{\delta} & -\sqrt{\delta} \end{pmatrix} \tag{62}$$

to all qubits in the data register leads to a gaussian distribution over the "Hamming distance" cube:



Liqui|⟩ simulations: Taking it further

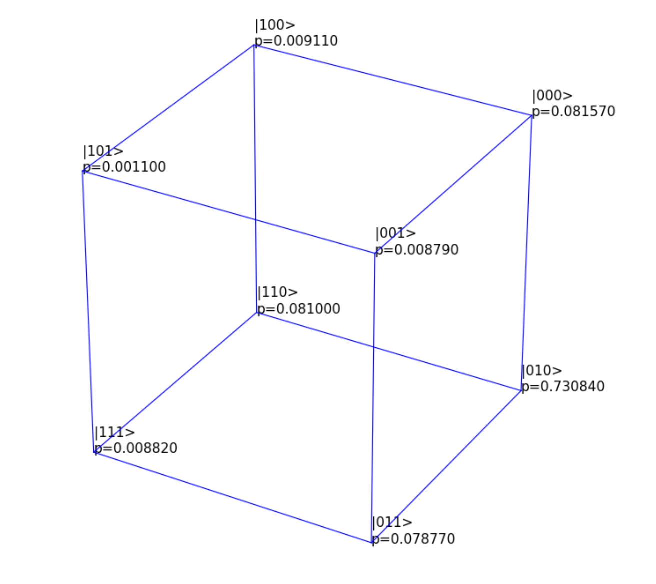


Figure 19: Representation of gaussian diffusion on 3D cube

Liqui|⟩ simulations: Taking it further

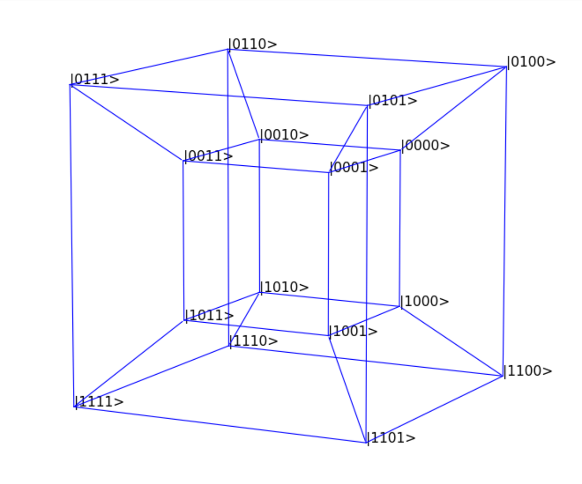


Figure 20: Representation of gaussian diffusion on 3D cube

Backup slide II I

IBM.

What is big data?

https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html, 2016.

Accessed: 2016-09-08.

Qubit-based kNN quantum algorithm

Typography

```
The theme provides sensible defaults to
\emph{emphasize} text, \alert{accent} parts
or show \textbf{bold} results.
```

becomes

The theme provides sensible defaults to *emphasize* text, accent parts or show **bold** results.

Font feature test

- Regular
- *Italic*
- SmallCaps
- **Bold**
- ***Bold Italic***
- **Bold SmallCaps**
- `Monospace`
- *`Monospace Italic`*
- **`Monospace Bold`**
- ***`Monospace Bold Italic`***

References go here

Lists

Items

- Milk
- Eggs
- Potatos

Enumerations

1. First,
2. Second and
3. Last.

Descriptions

PowerPoint Meeh.

Beamer Yeeeha.

Animation

- This is important

Animation

- This is important
- Now this

Animation

- This is important
- Now this
- And now this

Animation

- This is really important
- Now this
- And now this

Figures

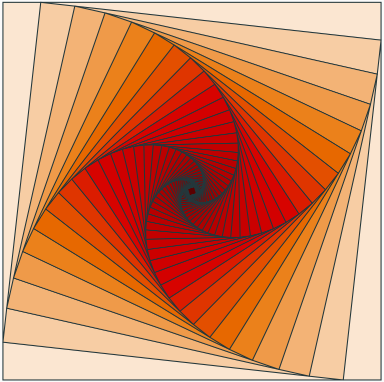


Figure 21: Rotated square from texample.net.

Tables

Table 5: Largest cities in the world (source: Wikipedia)

City	Population
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

Blocks

Three different block environments are pre-defined and may be styled with an optional background color.

Default

Block content.

Alert

Block content.

Example

Block content.

Default

Block content.

Alert

Block content.

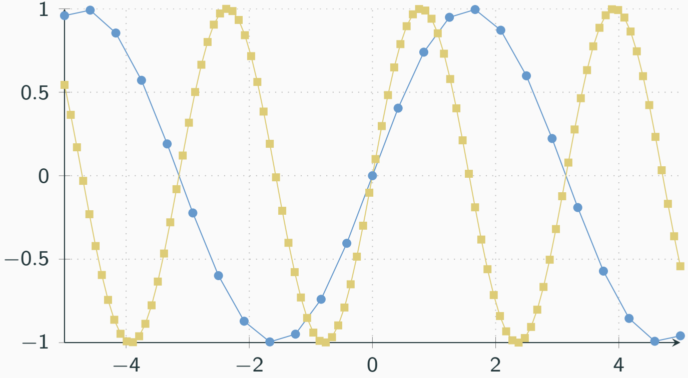
Example

Block content.

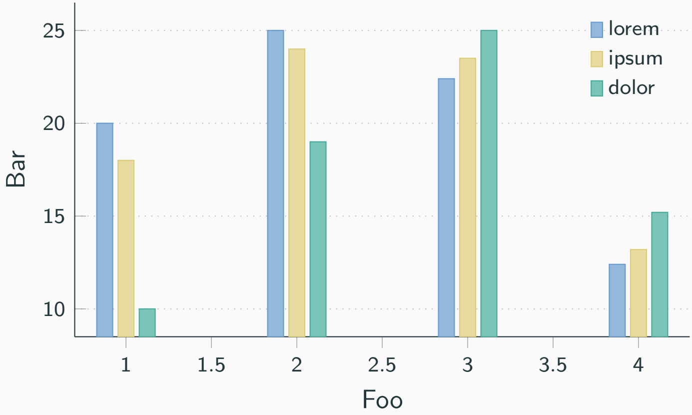
Math

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

Line plots



Bar charts



Quotes

Veni, Vidi, Vici