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

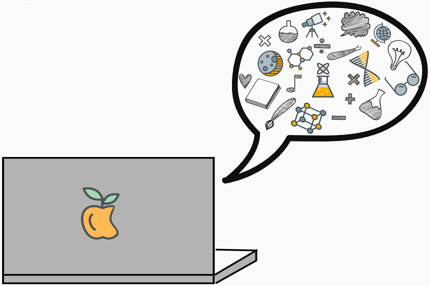
Bachelor thesis defense
19. January 2017

Mark Fingerhuth

Maastricht Science Programme, Maastricht University, The Netherlands
Thesis work at the Centre for Quantum Technology, UKZN, South Africa

Table of contents

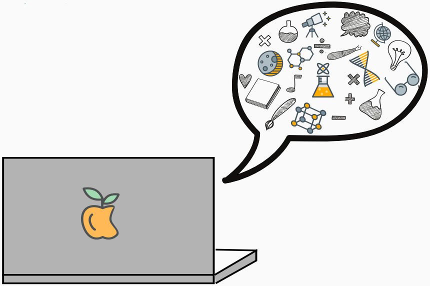
Introduction

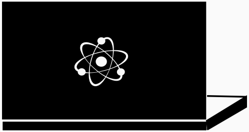


Machine learning

Enable computers to learn from data

Enhancing machine learning with quantum mechanics





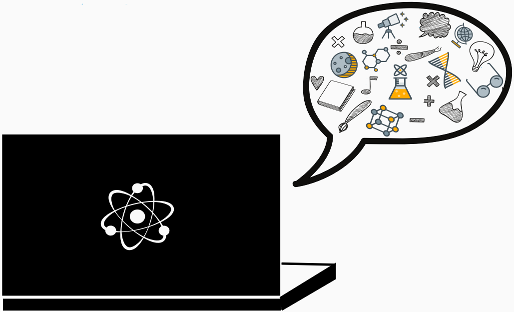
Machine learning

Enable computers to learn from data

Quantum computing

Build computer hardware based on quantum physics

Enhancing machine learning with quantum mechanics



Quantum-enhanced machine learning

Enable quantum computers to learn from data faster than classical computers

Machine Learning

Supervised machine learning

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

Table 1: Example training dataset.

$$f(x) = o\,. \tag{1}$$

$$f(\boxed{\text{colour}}) = \mathit{red}\,. \tag{2}$$

Supervised machine learning

ID	Colour	Class label
1		?
2		?

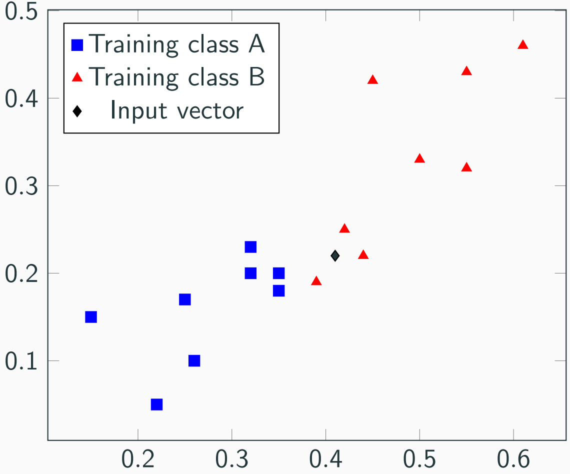
Table 2: Example input dataset.

The machine learning algorithm should approximate the function f such that it can predict the output o for a new unknown input $\tilde{x}$:

$$f(\tilde{x}) = ?\,. \tag{3}$$

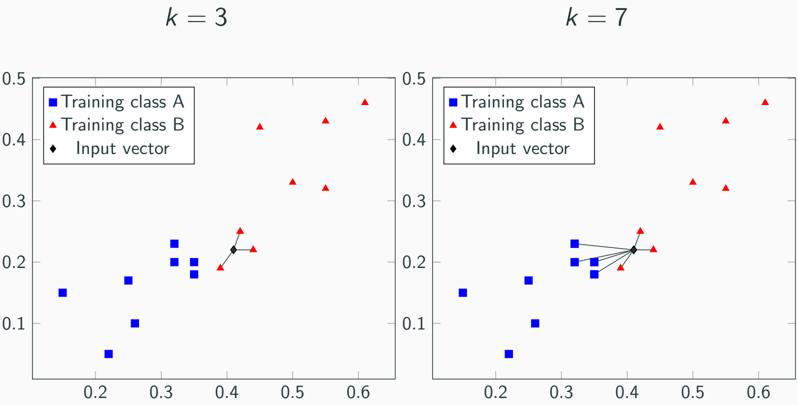
1

Classical k-nearest neighbour



1

Classical k-nearest neighbour



1

Classical k-nearest neighbour

Explain the kNN!

1

Quantum Computing

Classical vs. quantum bits (qubits)

PICTURE OF MOSFET

Classical bit:

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

1

Classical vs. quantum bits (qubits)

Quantum bit (qubit):

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

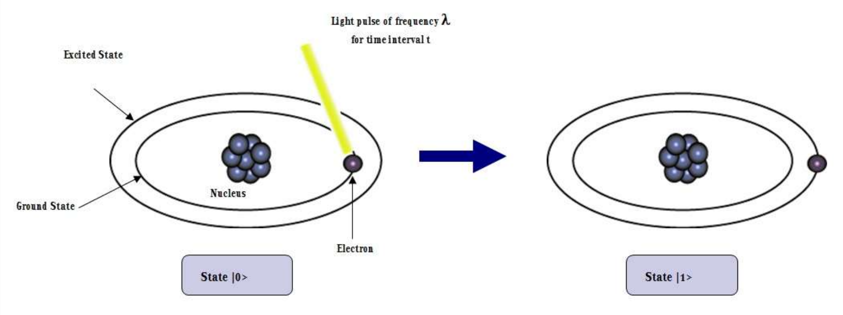


Figure 1: Example of a physical qubit.[1]

[1] Reprinted from RF Wireless World, n.d., Retrieved December 23, 2016, from http://www.rfwireless-world.com/Terminology/Difference-between-Bit-and-Qubit.html. Copyright 2012 by RF Wireless World.

Qubits

Mathematically, a qubit state is expressed as:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \ , \tag{4}$$

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

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

$$|0\rangle \doteq \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle \doteq \begin{pmatrix} 0 \\ 1 \end{pmatrix} . \tag{5}$$

Substituting into Eq. 4 yields the vector representation of $|\psi\rangle$:

$$|\psi\rangle \doteq \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} . \tag{6}$$

The Bloch sphere

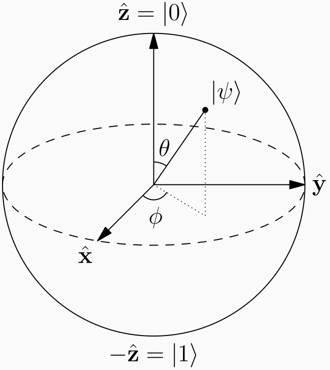


Figure 2: Arbitrary two-dimensional qubit $|\psi\rangle$ visualized on the Bloch sphere[1]

Most general form of a 2-D qubit:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \ , \qquad (7)$$

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

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

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \ , \ (8)$$

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.

Quantum gates

$$U|\psi\rangle \doteq \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{pmatrix} . \tag{9}$$

show matrix representation show matrix vector multiplication?? Show two simple examples (Hadamard and X gate?)

Quantum-enhanced Machine Learning

Encoding classical data into qubits

Play/Pause

Encoding classical data into amplitudes

Amplitude interference

1

Results: Qubit-based kNN algorithm

Results: Amplitude-based kNN algorithm

efesafefesf

Quantum Computing & Qubits

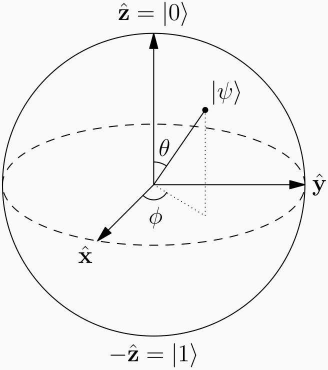


Figure 3: Arbitrary two-dimensional qubit $|\psi\rangle$ visualized on the Bloch sphere[1]

Most general form of a 2-D qubit:

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad (10)$$

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

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

$$|q\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \quad (11)$$

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 $\rightarrow$ 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.

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

Data encoded into amplitudes

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

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

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

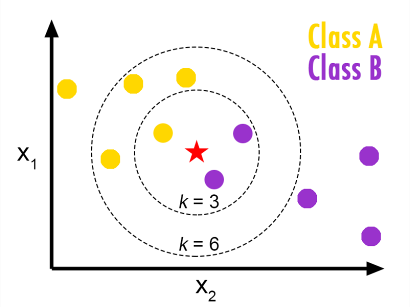


Figure 4: Visualization of a kNN classifier[1]

[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.

$$\frac{1}{\sqrt{2M}}\sum_{m=1}^{M}(|0\rangle\,|\Psi_{\tilde{x}}(\star)\rangle + |1\rangle\,|\Psi_{x^m}\rangle)\,|y^m(A\ \mathit{or}\ B)\rangle\,|m\rangle \tag{12}$$

where

$$|\Psi_{\tilde{x}}(\star)\rangle = \sum_{i=1}^{N}\tilde{x}_i\,|i\rangle \qquad |\Psi_{x^m}\rangle = \sum_{i=1}^{N}x_i^m\,|i\rangle \tag{13}$$

$$e.g. \quad \begin{pmatrix}0.6\\0.4\end{pmatrix} \quad \rightarrow \quad |n\rangle = \sqrt{0.6}\,|0\rangle + \sqrt{0.4}\,|1\rangle \tag{14}$$

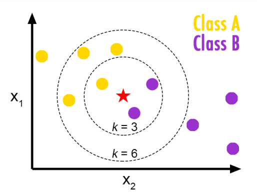


Figure 5: Visualization of a kNN classifier[1]

Maria Schuld (2016), unpublished

The algorithm

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

$$\frac{1}{2\sqrt{M}}\sum_{m=1}^{M}(|0\rangle[|\Psi_{\tilde{x}}\rangle+|\Psi_{x^m}\rangle]+|1\rangle[|\Psi_{\tilde{x}}\rangle-|\Psi_{x^m}\rangle])\,|y^m(A \text{ or } B)\rangle\,|m\rangle \quad (15)$$

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

The algorithm

After successful conditional measurement, the state is proportional to

$$\frac{1}{2\sqrt{M}} \sum_{m=1}^{M} \sum_{i=1}^{N} (\tilde{x}_i + x_i^m) |0\rangle |i\rangle |y^m(A \text{ or } B)\rangle |m\rangle \tag{16}$$

Probability to measure class B:

$$p(|y^m\rangle = |1(B)\rangle) = \sum_{m|y^m=1(B)} 1 - \frac{1}{4M} |\ \tilde{x} - x^m\ |^2 \tag{17}$$

Overall algorithmic complexity

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

Simple binary classification case

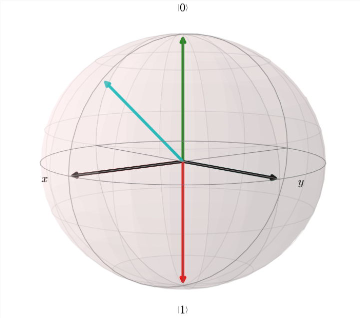


Figure 6: Simple binary classification problem of a quantum state

$$\frac{1}{\sqrt{2M}}\sum_{m=1}^{M}(|0\rangle|\Psi_{\tilde{x}}(\star)\rangle + |1\rangle|\Psi_{x^m}\rangle)|y^m(A \text{ or } B)\rangle|m\rangle \tag{18}$$

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

$$|\Psi_0\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle|0\rangle + |1\rangle|0\rangle)|y^m\rangle|m\rangle \tag{19}$$

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

$${}^1_0CR_y(\frac{\pi}{4})|\Psi_0\rangle = |\Psi_1\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle|0\rangle+|1\rangle|\Psi_{\tilde{x}}\rangle)|y^m\rangle|m\rangle \tag{20}$$

Flip the ancilla qubit in the first register

$$(X\otimes\mathbb{1}\otimes\mathbb{1}\otimes\mathbb{1})|\Psi_1\rangle = |\Psi_2\rangle = \frac{1}{2}\sum_{m=1}^{2}(|0\rangle|\Psi_{\tilde{x}}\rangle+|1\rangle|0\rangle)|y^m\rangle|m\rangle \tag{21}$$

Implementation with IBM's quantum computer

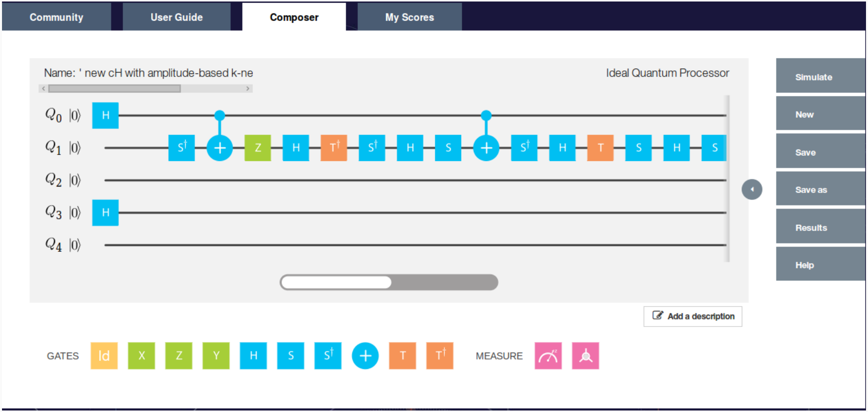


Figure 7: IBM's quantum composer

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

IBM's universal gate set

Figure 8: IBM's universal gate set

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

Controlled U gate

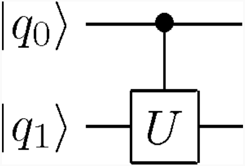


Figure 9: Controlled U-gate

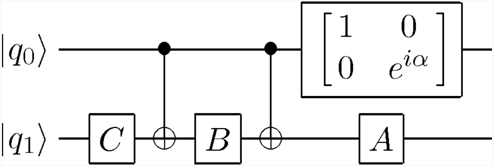


Figure 10: Decomposition of a controlled U-gate[1]

Choose A,B,C and α s.t.

$$e^{i\alpha} * A * X * B * X * C = U \quad and \quad A * B * C = \mathbb{1} \tag{22}$$

Need to solve the following equation[1]

$$U = \begin{pmatrix} e^{i(\alpha - \frac{\beta}{2} - \frac{\delta}{2})} \cos \frac{\gamma}{2} & -e^{i(\alpha - \frac{\beta}{2} + \frac{\delta}{2})} \sin \frac{\gamma}{2} \\ e^{i(\alpha + \frac{\beta}{2} - \frac{\delta}{2})} \sin \frac{\gamma}{2} & e^{i(\alpha + \frac{\beta}{2} + \frac{\delta}{2})} \cos \frac{\gamma}{2} \end{pmatrix} \tag{23}$$

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 ${}^1_0CR_y(\frac{\pi}{4})$:

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

$$\alpha = \pi; \quad \beta = 2\pi; \quad \delta = \frac{7}{8}\pi; \quad \gamma = 0 \tag{24}$$

Then,

$$A = R_z(\beta)R_y(\frac{\gamma}{2}) = R_z(2\pi) = \mathbb{1} \tag{25}$$

$$B = R_y(-\frac{\gamma}{2})R_z(-\frac{\delta+\beta}{2}) = R_z(-\frac{23}{16}\pi) = ??? \tag{26}$$

$$C = R_z(\frac{\delta-\beta}{2}) = R_z(-\frac{9}{16}\pi) = ??? \tag{27}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = Z \tag{28}$$

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

The Solovay-Kitaev theorem

$$
\begin{aligned}
B &= R_z(-\frac{23}{16}\pi) &= \textcolor{red}{???} \qquad (29)\\
C &= R_z(-\frac{9}{16}\pi) &= \textcolor{red}{???} \qquad (30)
\end{aligned}
$$

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.**

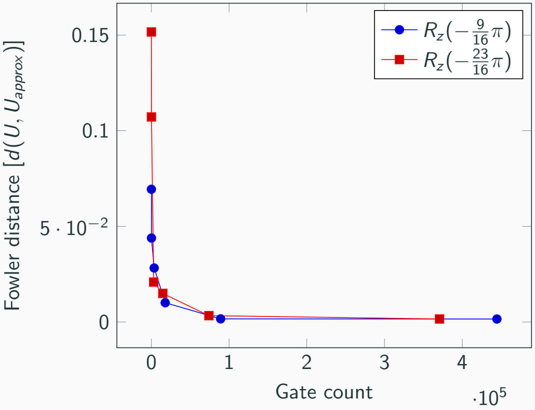
→ **But needs to be computed classically!**

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

The Solovay-Kitaev algorithm

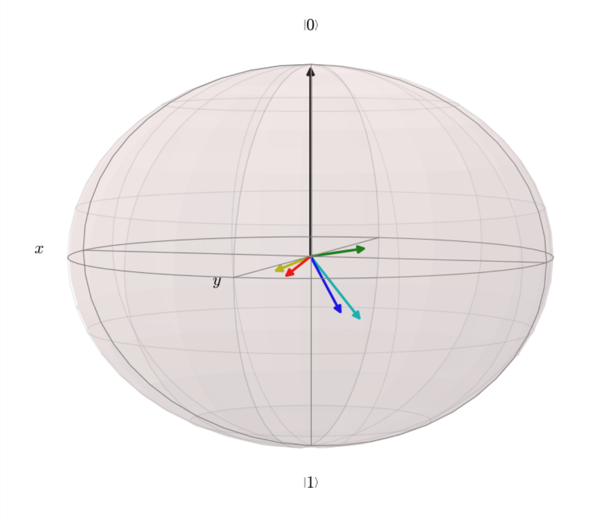
Fowler distance[1]:

$$dist(U, U_{approx}) = \sqrt{\frac{2- \mid tr(U \cdot U_{approx}^{\dagger}) \mid}{2}} \tag{31}$$



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

The Solovay-Kitaev algorithm



$$d = 0.22739 \quad (32)$$

$$d = 0.15165 \quad (33)$$

$$d = 0.10722 \quad (34)$$

$$d = 0.02086 \quad (35)$$

$$d = 0.00156 \quad (36)$$

Figure 11: 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 μs
	0.10722	109	$\sim$14 μs
	0.02086	2997	$\sim$390 μs
	0.01494	14721	$\sim$1914 μs
	0.003327	74009	$\sim$9621 μs
	0.001578	370813	$\sim$48 206 μs

Table 3: 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

$$\mathcal{O}(\frac{1}{p_{acc}}) \tag{37}$$

suddenly to

$$\mathcal{O}(m * log^{2.71}(\frac{m}{\epsilon})) \tag{38}$$

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:

$$e^{t/T_i} \tag{39}$$

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

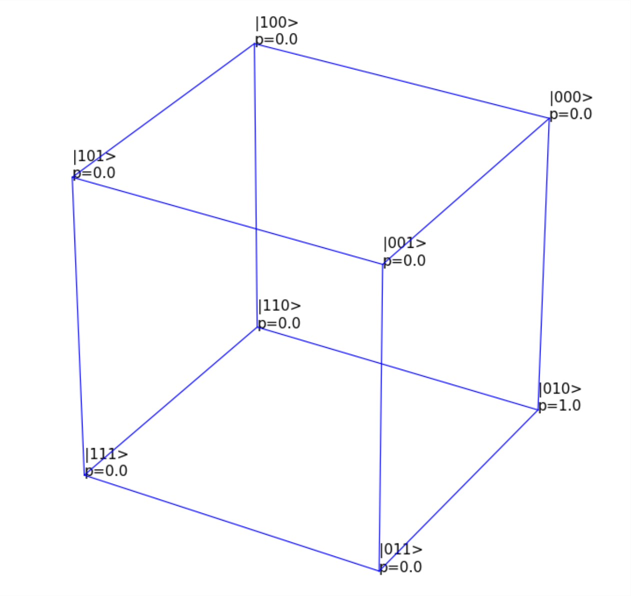


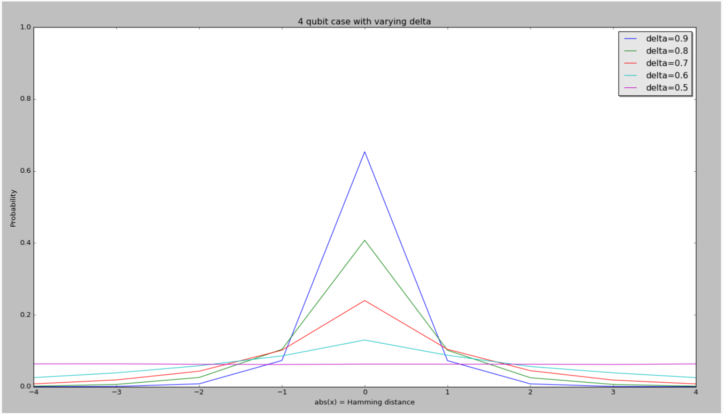
Figure 12: Representation of hamming distance on 3D cube

Liqui|⟩ simulations: Taking it further

Applying the following matrix

$$\begin{pmatrix} \sqrt{\delta} & 1-\sqrt{\delta} \\ 1-\sqrt{\delta} & -\sqrt{\delta} \end{pmatrix} \tag{40}$$

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



Liqui|⟩ simulations: Taking it further

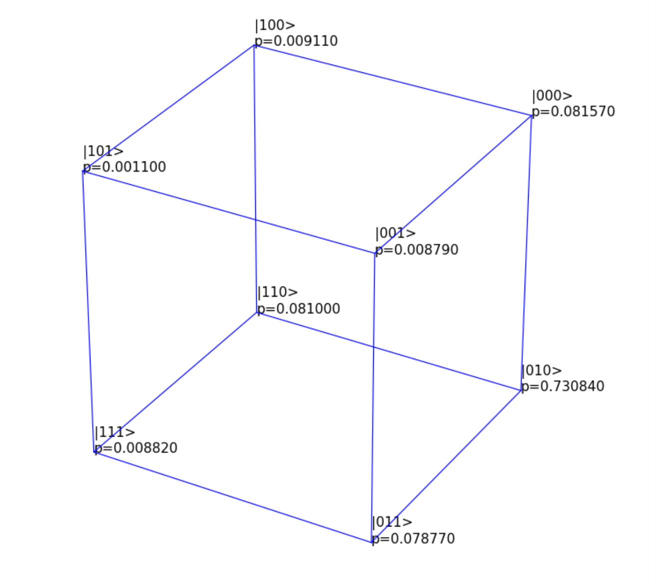


Figure 14: Representation of gaussian diffusion on 3D cube

Liqui|⟩ simulations: Taking it further

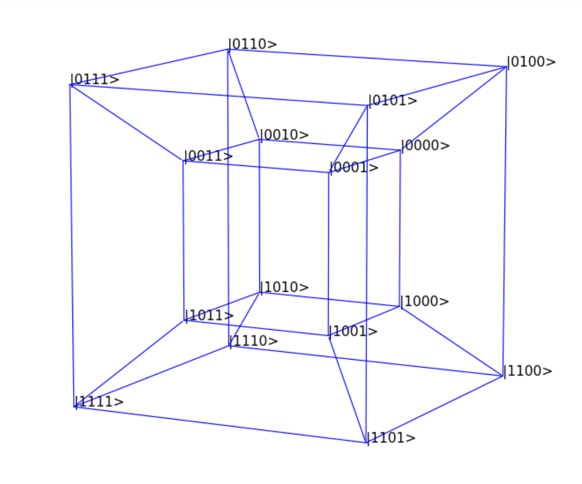


Figure 15: Representation of gaussian diffusion on 3D cube

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

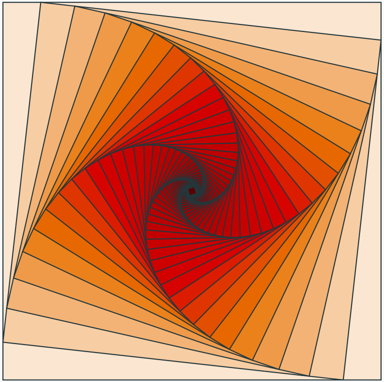


Figure 16: Rotated square from texample.net.

Tables

Table 4: 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.

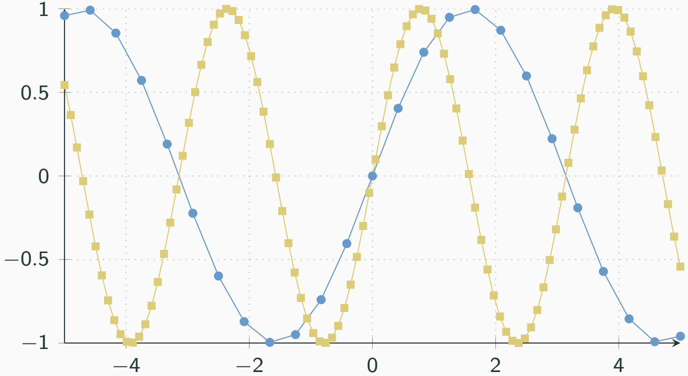
Example

Block content.

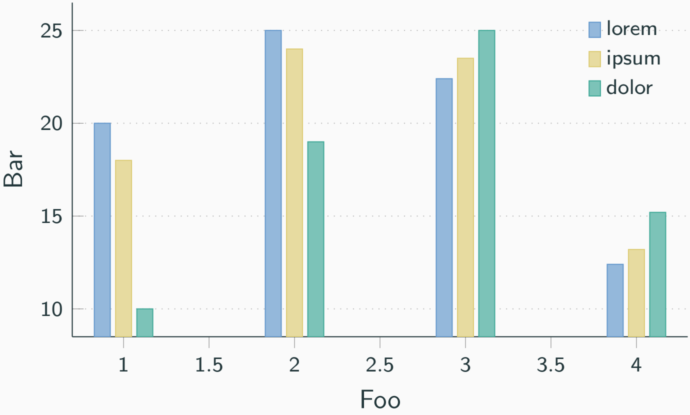
Math

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

Line plots



Bar charts



Quotes

Veni, Vidi, Vici