Tutorial on Quantum Machine Learning

Ronald de Wolf

CWI

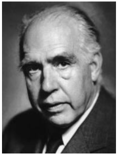
QuSoft

Universiteit van Amsterdam

Quantum computers

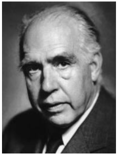
Quantum computers

Quantum mechanics:
developed from 1900

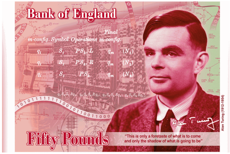


Quantum computers

Quantum mechanics:
developed from 1900

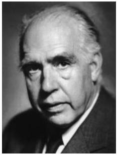


Computer science:
developed from 1930s

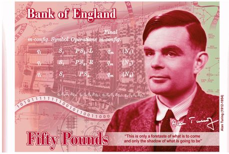


Quantum computers

Quantum mechanics:
developed from 1900



Computer science:
developed from 1930s



Richard Feynman, David Deutsch

in early 1980s:

Harness those quantum effects for useful computations!

The math of quantum computing on one slide

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- *n*-qubit system: superposition of all *n*-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x|x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes.

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x\in\{0,1\}^n} \alpha_x|x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x\in\{0,1\}^n} \alpha_x|x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x\in\{0,1\}^n} \alpha_x|x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

or on 2 qubits, CNOT: $|a, b\rangle \mapsto |a, a \oplus b\rangle$

The math of quantum computing on one slide

- Qubit is superposition of 0 and 1: $\alpha_0|0\rangle + \alpha_1|1\rangle \in \mathbb{C}^2$
- n-qubit system: superposition of all n-bit strings:

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \in \mathbb{C}^{2^n}$$

- Measurement: see outcome $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$
- Unitary transformation: matrix that preserves the length of the vector of amplitudes. Gates: unitaries on 1 qubit

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

or on 2 qubits, CNOT: $|a, b\rangle \mapsto |a, a \oplus b\rangle$

- Combine simultaneous gates via tensor product, combine sequential gates via matrix product

Quantum algorithms

Quantum algorithms

Q algorithms work by interplay of superposition and interference

Quantum algorithms

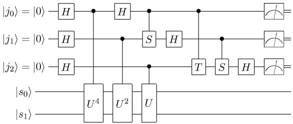
Q algorithms work by interplay of superposition and interference:

1. Start with qubits in some simple state (e.g. all $|0\rangle$)

Quantum algorithms

Q algorithms work by interplay of superposition and interference:

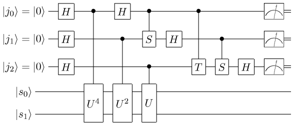
1. Start with qubits in some simple state (e.g. all $|0\rangle$)
2. Run circuit of gates to create the the right interference, so final state has most of its weight on solutions to your computational problem



Quantum algorithms

Q algorithms work by interplay of superposition and interference:

1. Start with qubits in some simple state (e.g. all $|0\rangle$)
2. Run circuit of gates to create the the right interference, so final state has most of its weight on solutions to your computational problem

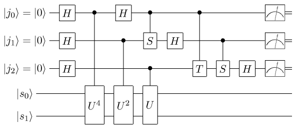


3. Measuring final state then gives solution to your problem

Quantum algorithms

Q algorithms work by interplay of superposition and interference:

1. Start with qubits in some simple state (e.g. all $|0\rangle$)
2. Run circuit of gates to create the the right interference, so final state has most of its weight on solutions to your computational problem



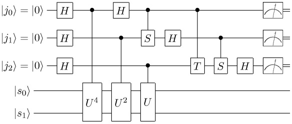
3. Measuring final state then gives solution to your problem

Two important questions:

Quantum algorithms

Q algorithms work by interplay of superposition and interference:

1. Start with qubits in some simple state (e.g. all $|0\rangle$)
2. Run circuit of gates to create the the right interference, so final state has most of its weight on solutions to your computational problem



3. Measuring final state then gives solution to your problem

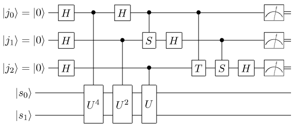
Two important questions:

- Can we build such a computer?

Quantum algorithms

Q algorithms work by interplay of superposition and interference:

1. Start with qubits in some simple state (e.g. all $|0\rangle$)
2. Run circuit of gates to create the the right interference, so final state has most of its weight on solutions to your computational problem



3. Measuring final state then gives solution to your problem

Two important questions:

- Can we build such a computer?
- What can it do?

Quantum algorithms: main examples that we know

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)
- Grover's algorithm'96: search through an unstructured search space of size N in time $\sqrt{N}$

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)
- Grover's algorithm'96: search through an unstructured search space of size N in time $\sqrt{N}$
- Quantum walks'00ff: for more structured search problems on graphs, typically quadratic quantum speed-up or less

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)
- Grover's algorithm'96: search through an unstructured search space of size N in time $\sqrt{N}$
- Quantum walks'00ff: for more structured search problems on graphs, typically quadratic quantum speed-up or less
- HHL algorithm'09: can solve a sparse, well-conditioned linear system $Ax = b$ very efficiently, but provides the answer as a quantum state $\sum_i x_i|i\rangle$

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)
- Grover's algorithm'96: search through an unstructured search space of size N in time $\sqrt{N}$
- Quantum walks'00ff: for more structured search problems on graphs, typically quadratic quantum speed-up or less
- HHL algorithm'09: can solve a sparse, well-conditioned linear system $Ax = b$ very efficiently, but provides the answer as a quantum state $\sum_i x_i|i\rangle$ (when is this useful?)

Quantum algorithms: main examples that we know

- Shor's algorithm'94: can factor large integers and find discrete logarithms efficiently (runtime quadratic in number input bits)
- Grover's algorithm'96: search through an unstructured search space of size N in time $\sqrt{N}$
- Quantum walks'00ff: for more structured search problems on graphs, typically quadratic quantum speed-up or less
- HHL algorithm'09: can solve a sparse, well-conditioned linear system $Ax = b$ very efficiently, but provides the answer as a quantum state $\sum_i x_i |i\rangle$ (when is this useful?)
- Hamiltonian simulation'96ff: given classical description of a local Hamiltonian $H = \sum_j H_j$, implement the unitary evolution e^{-iHt} as a small circuit of gates

Quantum machine learning

- Machine learning: huge success since $\pm$ 2012

Quantum machine learning

- Machine learning: huge success since $\pm$ 2012
- Quantum machine learning: huge hype since $\pm$ 2015

Quantum machine learning

- Machine learning: huge success since $\pm$ 2012
- Quantum machine learning: huge hype since $\pm$ 2015
- Often mentioned by startups and newspaper articles as an obvious area where quantum computers are great

Quantum machine learning

- Machine learning: huge success since ± 2012
- Quantum machine learning: huge hype since ± 2015
- Often mentioned by startups and newspaper articles as an obvious area where quantum computers are great
- What do we actually have?

Quantum machine learning

- Machine learning: huge success since $\pm$ 2012
- Quantum machine learning: huge hype since $\pm$ 2015
- Often mentioned by startups and newspaper articles as an obvious area where quantum computers are great
- What do we actually have?
  - Hard-to-assess claims about speedups for natural problems using variational circuits ("quantum neural networks")

Quantum machine learning

- Machine learning: huge success since ± 2012
- Quantum machine learning: huge hype since ± 2015
- Often mentioned by startups and newspaper articles as an obvious area where quantum computers are great
- What do we actually have?
  - Hard-to-assess claims about speedups for natural problems using variational circuits ("quantum neural networks")
  - Proven claims about quantum improvements in time/sample complexity for problems with quantum data

Quantum machine learning

- Machine learning: huge success since $\pm$ 2012
- Quantum machine learning: huge hype since $\pm$ 2015
- Often mentioned by startups and newspaper articles as an obvious area where quantum computers are great
- What do we actually have?
  - Hard-to-assess claims about speedups for natural problems using variational circuits ("quantum neural networks")
  - Proven claims about quantum improvements in time/sample complexity for problems with quantum data
  - Proven but subsequently dequantized quantum ML algorithms (Kerenidis-Prakash recommendation system by Ewin Tang)

This talk: theoretical aspects of quantum ML

This talk: theoretical aspects of quantum ML

- ML = data + optimization

This talk: theoretical aspects of quantum ML

- ML = data + optimization

	Classical learner	*Quantum learner*
Classical data	Classical ML	**This talk**
Quantum data	?	**This talk**

This talk: theoretical aspects of quantum ML

- ML = data + optimization

	Classical learner	*Quantum learner*
Classical data	Classical ML	**This talk**
Quantum data	?	**This talk**

- Subareas of ML:

  1. Supervised learning: from labeled data
     PAC learning from quantum data, positive & negative results

This talk: theoretical aspects of quantum ML

- ML = data + optimization

	Classical learner	*Quantum learner*
Classical data	Classical ML	**This talk**
Quantum data	?	**This talk**

- Subareas of ML:
  1. Supervised learning: from labeled data
     PAC learning from quantum data, positive & negative results
  2. Unsupervised learning: from unlabeled data
     Quantum linear algebra, e.g. Principal Component Analysis

This talk: theoretical aspects of quantum ML

- ML = data + optimization

	Classical learner	*Quantum learner*
Classical data	Classical ML	**This talk**
Quantum data	?	**This talk**

- Subareas of ML:

1. Supervised learning: from labeled data
   PAC learning from quantum data, positive & negative results
2. Unsupervised learning: from unlabeled data
   Quantum linear algebra, e.g. Principal Component Analysis
3. Reinforcement learning: from interaction with the environment
   Very interesting, but won't cover it here

A mathematical model for supervised learning: PAC

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \rightarrow \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0,1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

, $+$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 $, +$  $, -$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples:
  $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 $, +$  $, -$  $, +$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0,1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 $, +$  $, -$  $, +$  $, -$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0, 1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,. . .
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 $, +$  $, -$  $, +$  $, -$

- Goal: using some i.i.d. examples, learner for $\mathcal{C}$ should output hypothesis h that is probably approximately correct (PAC).

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0,1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 , $+$  , $-$  , $+$  , $-$

- Goal: using some i.i.d. examples, learner for $\mathcal{C}$ should output hypothesis h that is probably approximately correct (PAC).

  Error of h w.r.t. target f: $\text{err}_{\mathcal{D}}(f, h) = \Pr_{x \sim \mathcal{D}}[f(x) \neq h(x)]$

A mathematical model for supervised learning: PAC

- Concept: some function $f : \mathcal{X} \to \{-1, 1\}$ (think $\mathcal{X} = \{0,1\}^n$)
  Concept class $\mathcal{C}$: set of concepts, e.g. small circuits, DNFs,...
- Want to learn unknown target concept $f \in \mathcal{C}$ from examples: $(x, f(x))$, where $x \sim$ unknown distribution $\mathcal{D}$ on $\mathcal{X}$

 $, +$  $, -$  $, +$  $, -$

- Goal: using some i.i.d. examples, learner for $\mathcal{C}$ should output hypothesis h that is probably approximately correct (PAC).

  Error of h w.r.t. target f: $\text{err}_{\mathcal{D}}(f, h) = \Pr_{x \sim \mathcal{D}}[f(x) \neq h(x)]$
- An algorithm (ε, δ)-PAC-learns $\mathcal{C}$ if:

$$\forall f \in \mathcal{C} \;\; \forall \mathcal{D} : \quad \Pr[\underbrace{\text{err}_{\mathcal{D}}(f, h) \leq \varepsilon}_{h \text{ is approximately correct}}] \geq 1 - \delta$$

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition.

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition. But this is expensive...

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition. But this is expensive...
- Let's try to circumvent the problem of putting classical data in superposition, by assuming we start from quantum data

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition. But this is expensive...
- Let's try to circumvent the problem of putting classical data in superposition, by assuming we start from quantum data
- Bshouty-Jackson'95: suppose example is a superposition

$$\sum_{x \in \mathcal{X}} \sqrt{\mathcal{D}(x)} \, |x, f(x)\rangle$$

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition. But this is expensive. . .
- Let's try to circumvent the problem of putting classical data in superposition, by assuming we start from quantum data
- Bshouty-Jackson'95: suppose example is a superposition

$$\sum_{x \in \mathcal{X}} \sqrt{\mathcal{D}(x)}\, |x, f(x)\rangle$$

Measuring this quantum state gives classical example $\sim \mathcal{D}$ so quantum examples are at least as powerful as classical

PAC learning from **quantum** examples

- Much interesting quantum ML assumes classical data can be turned into quantum superposition. But this is expensive...
- Let's try to circumvent the problem of putting classical data in superposition, by assuming we start from quantum data
- Bshouty-Jackson'95: suppose example is a superposition

$$\sum_{x \in \mathcal{X}} \sqrt{\mathcal{D}(x)}\,|x, f(x)\rangle$$

Measuring this quantum state gives classical example $\sim \mathcal{D}$ so quantum examples are at least as powerful as classical

- Next slides: some cases where quantum examples are more powerful than classical for a fixed distribution $\mathcal{D}$

Uniform quantum examples can help sometimes

Uniform quantum examples can help sometimes

- Quantum example for target concept f under uniform $\mathcal{D}$:

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

Uniform quantum examples can help sometimes

- Quantum example for target concept f under uniform $\mathcal{D}$:

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

- Key subroutine: Fourier sampling (Bernstein-Vazirani'93)

Uniform quantum examples can help sometimes

- Quantum example for target concept f under uniform $\mathcal{D}$:

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

- Key subroutine: Fourier sampling (Bernstein-Vazirani'93):
Can convert (with probability $1/2$) quantum example to

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

Uniform quantum examples can help sometimes

- Quantum example for target concept f under uniform $\mathcal{D}$:

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

- Key subroutine: Fourier sampling (Bernstein-Vazirani'93): Can convert (with probability $1/2$) quantum example to

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

Hadamard transform turns this into $\displaystyle\sum_{s \in \{0,1\}^n} \widehat{f}(s)|s\rangle$,

$\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x}$ are the Fourier coefficients of f

Uniform quantum examples can help sometimes

- Quantum example for target concept f under uniform $\mathcal{D}$:

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

- Key subroutine: Fourier sampling (Bernstein-Vazirani'93): Can convert (with probability $1/2$) quantum example to

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

  Hadamard transform turns this into $\displaystyle\sum_{s \in \{0,1\}^n} \widehat{f}(s)|s\rangle$,

  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x}$ are the Fourier coefficients of f

- This allows us to sample s from distribution $\widehat{f}(s)^2$

Two cases where Fourier sampling helps learning

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2): $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:

  $\widehat{f}(s)$

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:

  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x}$

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x}$

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!
- Bshouty-Jackson'95: learn Disjunctive Normal Form (DNF) formulas in poly-time under uniform $\mathcal{D}$:

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!
- Bshouty-Jackson'95: learn Disjunctive Normal Form (DNF) formulas in poly-time under uniform $\mathcal{D}$:
  Fourier sampling gives a parity-function that's weakly correlated with target DNF function f,

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n}\sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n}\sum_x (-1)^{(a \oplus s)\cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!
- Bshouty-Jackson'95: learn Disjunctive Normal Form (DNF) formulas in poly-time under uniform $\mathcal{D}$:
  Fourier sampling gives a parity-function that's weakly correlated with target DNF function f, can combine this with classical "boosting" to find good hypothesis h.

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2):
  $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!
- Bshouty-Jackson'95: learn Disjunctive Normal Form (DNF) formulas in poly-time under uniform $\mathcal{D}$:
  Fourier sampling gives a parity-function that's weakly correlated with target DNF function f, can combine this with classical "boosting" to find good hypothesis h.
  Best known classical learner takes time $n^{O(\log n)}$

Two cases where Fourier sampling helps learning

- Concept class $\mathcal{C}$ of linear functions (mod 2): $f(x) = (-1)^{a \cdot x}$ for fixed $a \in \{0,1\}^n$.
  Linear functions have very simple Fourier coefficients:
  $\widehat{f}(s) = \frac{1}{2^n} \sum_x f(x)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_x (-1)^{(a \oplus s) \cdot x} = \begin{cases} 1 & \text{if } s = a \\ 0 & \text{otherwise} \end{cases}$
  We can learn a (and hence f) from one Fourier sample!
- Bshouty-Jackson'95: learn Disjunctive Normal Form (DNF) formulas in poly-time under uniform $\mathcal{D}$:
  Fourier sampling gives a parity-function that's weakly correlated with target DNF function f, can combine this with classical "boosting" to find good hypothesis h.
  Best known classical learner takes time $n^{O(\log n)}$
- But what about learners that work for all $\mathcal{D}$?

VC-dimension determines sample complexity in PAC model

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

  $\text{VC-dim}(\mathcal{C}) = \max\{d : \exists S \subseteq \mathcal{X} \text{ of size } d \text{ shattered by } \mathcal{C}\}$

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

  VC-dim($\mathcal{C}$) $= \max\{d : \exists S \subseteq \mathcal{X}$ of size d shattered by $\mathcal{C}\}$

  Set $S = \{s_1, \ldots, s_d\} \subseteq \mathcal{X}$ is shattered by $\mathcal{C}$ if
  for all $\ell \in \{0, 1\}^d$, there is an $f \in \mathcal{C}$ s.t. $\forall i \in [d] : f(s_i) = \ell_i$

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

  VC-dim$(\mathcal{C}) = \max\{d : \exists S \subseteq \mathcal{X}$ of size d shattered by $\mathcal{C}\}$

  Set $S = \{s_1, \ldots, s_d\} \subseteq \mathcal{X}$ is shattered by $\mathcal{C}$ if
  for all $\ell \in \{0,1\}^d$, there is an $f \in \mathcal{C}$ s.t. $\forall i \in [d] : f(s_i) = \ell_i$

- Classical sample complexity of (ε, δ)-PAC-learner for $\mathcal{C}$:

$$\Theta\left(\frac{d}{\varepsilon} + \frac{\log(1/\delta)}{\varepsilon}\right) \text{ examples}$$

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

  VC-dim($\mathcal{C}$) $= \max\{d : \exists S \subseteq \mathcal{X}$ of size d shattered by $\mathcal{C}\}$

  Set $S = \{s_1, \ldots, s_d\} \subseteq \mathcal{X}$ is shattered by $\mathcal{C}$ if
  for all $\ell \in \{0,1\}^d$, there is an $f \in \mathcal{C}$ s.t. $\forall i \in [d] : f(s_i) = \ell_i$

- Classical sample complexity of (ε, δ)-PAC-learner for $\mathcal{C}$:

$$\Theta\left(\frac{d}{\varepsilon} + \frac{\log(1/\delta)}{\varepsilon}\right) \text{ examples}$$

- Arunachalam & dW'17: same bound for quantum sample complexity!

VC-dimension determines sample complexity in PAC model

- Cornerstone of classical sample complexity: VC-dimension

  VC-dim($\mathcal{C}$) $= \max\{d : \exists S \subseteq \mathcal{X}$ of size d shattered by $\mathcal{C}\}$

  Set $S = \{s_1, \ldots, s_d\} \subseteq \mathcal{X}$ is shattered by $\mathcal{C}$ if
  for all $\ell \in \{0,1\}^d$, there is an $f \in \mathcal{C}$ s.t. $\forall i \in [d] : f(s_i) = \ell_i$

- Classical sample complexity of (ε, δ)-PAC-learner for $\mathcal{C}$:

$$\Theta\left(\frac{d}{\varepsilon} + \frac{\log(1/\delta)}{\varepsilon}\right) \text{ examples}$$

- Arunachalam & dW'17: same bound for quantum sample complexity! Hence in distribution-independent PAC learning quantum examples are not significantly better than classical

Quantum linear algebra

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow$ $\log(d)$ qubits), manipulate with unitaries

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix}$$

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\to \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

- Singular-value transformation (Gilyén, Su ao): can efficiently apply low-degree polynomial to A.

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

- Singular-value transformation (Gilyén, Su ao): can efficiently apply low-degree polynomial to A. Can recover most known quantum algorithms this way, and design new algorithms

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

- Singular-value transformation (Gilyén, Su ao): can efficiently apply low-degree polynomial to A. Can recover most known quantum algorithms this way, and design new algorithms
- Problems: (1) usually assumes quantum input

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

- Singular-value transformation (Gilyén, Su ao): can efficiently apply low-degree polynomial to A. Can recover most known quantum algorithms this way, and design new algorithms
- Problems: (1) usually assumes quantum input, (2) usually produces quantum output

Quantum linear algebra

- View data-vector as amplitudes of quantum state (d dimensions $\rightarrow \log(d)$ qubits), manipulate with unitaries
- Early example: HHL algorithm to solve linear system $Ax = b$: given ability to prepare $|b\rangle$ and implement e^{iA}, we can efficiently compute solution-vector as quantum state $|x\rangle$
- Modern approach: block-encoding of a matrix A into a unitary

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} \qquad U|0\rangle|\psi\rangle = |0\rangle A|\psi\rangle + |1\rangle|?\rangle$$

- Singular-value transformation (Gilyén, Su ao): can efficiently apply low-degree polynomial to A. Can recover most known quantum algorithms this way, and design new algorithms
- Problems: (1) usually assumes quantum input, (2) usually produces quantum output, (3) sometimes "dequantizable"...

Unsupervised learning: quantum PCA (LMR'14)

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i|$$

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m} A$$

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m} A$$

  This quantum state has the same eigenvectors as A

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m} A$$

  This quantum state has the same eigenvectors as A

- Quantum PCA: extract top-k eigenvectors as quantum states via "phase estimation" on a copy of ρ.

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of
$$A = \sum_{i=1}^{m} v_i v_i^T$$
- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state
$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m} A$$
This quantum state has the same eigenvectors as A
- Quantum PCA: extract top-k eigenvectors as quantum states via "phase estimation" on a copy of ρ. For that we want to implement (powers of) the unitary $e^{i\rho}$.

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of
$$A = \sum_{i=1}^{m} v_i v_i^T$$
- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state
$$\rho = \frac{1}{m}\sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m}A$$
This quantum state has the same eigenvectors as A
- Quantum PCA: extract top-k eigenvectors as quantum states via "phase estimation" on a copy of ρ. For that we want to implement (powers of) the unitary $e^{i\rho}$. We can implement $e^{i\rho\delta}$ with error $O(\delta^2)$ using one copy of ρ.

Unsupervised learning: quantum PCA (LMR'14)

- Principal Component Analysis: given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, reduce dimension to k by projecting on top-k eigenvectors of

$$A = \sum_{i=1}^{m} v_i v_i^T$$

- Suppose we can efficiently prepare $\log(d)$-qubit state $|v_i\rangle$. Doing this for a random i gives "mixed" quantum state

$$\rho = \frac{1}{m} \sum_{i=1}^{m} |v_i\rangle\langle v_i| = \frac{1}{m} A$$

  This quantum state has the same eigenvectors as A

- Quantum PCA: extract top-k eigenvectors as quantum states via "phase estimation" on a copy of ρ. For that we want to implement (powers of) the unitary $e^{i\rho}$. We can implement $e^{i\rho\delta}$ with error $O(\delta^2)$ using one copy of ρ. Doing this $O(t/\delta)$ times with $\delta = \varepsilon/t$ implements $e^{i\rho t}$ with error ε.

Quantum speedups for optimization problems

Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization

Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization
- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming.

Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization

- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming. Often uses amplitude amplification/estimation

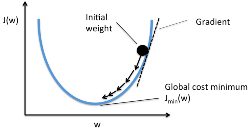
Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization
- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming. Often uses amplitude amplification/estimation
- **Continuous optimization**: for linear programs, semidefinite programs, matrix scaling and balancing, linear regression...

Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization
- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming. Often uses amplitude amplification/estimation
- **Continuous optimization**: for linear programs, semidefinite programs, matrix scaling and balancing, linear regression...

  Gradient descent: common iterative method to find local minimum of $f : \mathbb{R}^n \to \mathbb{R}$



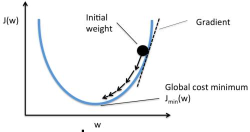
Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization

- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming. Often uses amplitude amplification/estimation

- **Continuous optimization**: for linear programs, semidefinite programs, matrix scaling and balancing, linear regression. . .

  Gradient descent: common iterative method to find local minimum of $f : \mathbb{R}^n \to \mathbb{R}$

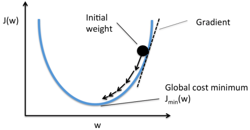
  

  Move current point along the direction of steepest descent ($=-$gradient of f at current point).

Quantum speedups for optimization problems

- ML = data + optimization.
  If data is classical, we can still try to speed up optimization
- **Discrete optimization**: for graph problems (shortest paths, sparsification), string problems, backtracking, dynamic programming. Often uses amplitude amplification/estimation
- **Continuous optimization**: for linear programs, semidefinite programs, matrix scaling and balancing, linear regression...

  Gradient descent: common iterative method to find local minimum of $f : \mathbb{R}^n \to \mathbb{R}$

  Move current point along the direction of steepest descent ($=-$gradient of f at current point).

  Jordan's algorithm can compute gradient more efficiently

One example of a quantum optimization algorithm for ML

One example of a quantum optimization algorithm for ML

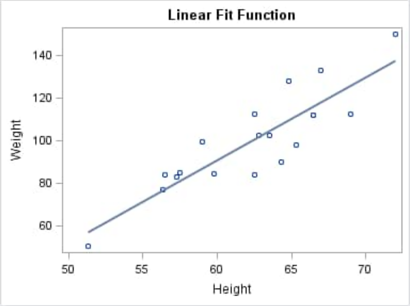
- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, fit line through them

One example of a quantum optimization algorithm for ML

- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T\theta$ is a good predictor of y-variable

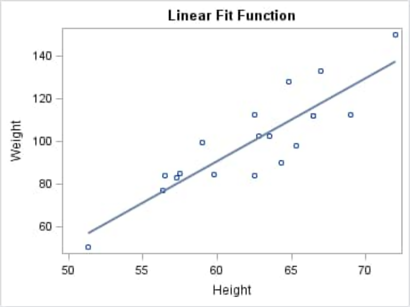
One example of a quantum optimization algorithm for ML

- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T\theta$ is a good predictor of y-variable



One example of a quantum optimization algorithm for ML

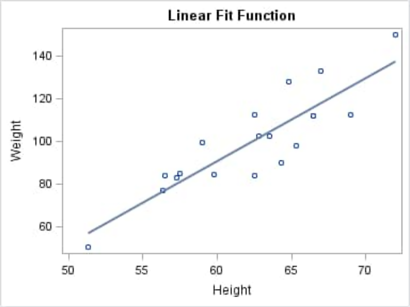
- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T \theta$ is a good predictor of y-variable



- Find θ to minmze least-squares loss $L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(x_i^T\theta - y_i)^2$

One example of a quantum optimization algorithm for ML

- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T\theta$ is a good predictor of y-variable

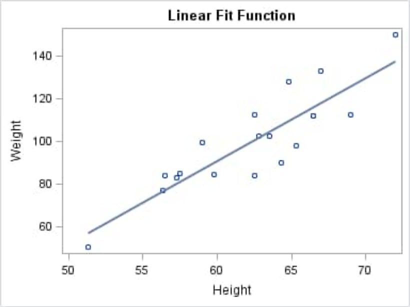


- Find θ to minmze least-squares loss $L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(x_i^T\theta - y_i)^2$

  Closed-form solution for the minimizer: $\theta^* = (X^TX)^+X^Ty$

One example of a quantum optimization algorithm for ML

- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T\theta$ is a good predictor of y-variable



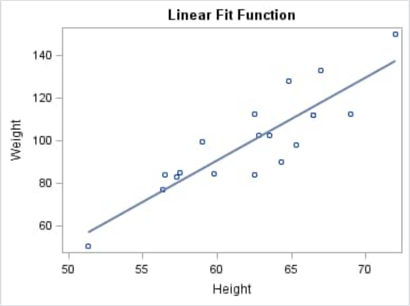
- Find θ to minmze least-squares loss $L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(x_i^T\theta - y_i)^2$

  Closed-form solution for the minimizer: $\theta^* = (X^T X)^+ X^T y$

- Problems: this tends to overfit and yield very dense θ-vectors

One example of a quantum optimization algorithm for ML

- Given m points $(x_1, y_1), \ldots, (x_m, y_m)$ with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, fit line through them: find coefficient-vector $\theta \in \mathbb{R}^d$ s.t. linear function $x_i^T\theta$ is a good predictor of y-variable



- Find θ to minmze least-squares loss $L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(x_i^T\theta - y_i)^2$

  Closed-form solution for the minimizer: $\theta^* = (X^TX)^+X^Ty$

- Problems: this tends to overfit and yield very dense θ-vectors

- Lasso adds "ℓ_1-regularizer": $\min L(\theta)$ subject to $\sum_{j=1}^{d}|\theta_j| \leq 1$

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$
- Finding the exact minimizer is a hard problem, so we typically try to find a vector θ whose loss is not much worse:

$$L(\theta) \leq L_{\min} + \varepsilon \quad \text{subject to } \sum_{j=1}^{d} |\theta_j| \leq 1$$

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$
- Finding the exact minimizer is a hard problem, so we typically try to find a vector θ whose loss is not much worse:

$$L(\theta) \leq L_{\min} + \varepsilon \quad \text{subject to} \sum_{j=1}^{d} |\theta_j| \leq 1$$

- Best classical algorithm runs in time $\tilde{O}(d/\varepsilon^2)$

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$
- Finding the exact minimizer is a hard problem, so we typically try to find a vector θ whose loss is not much worse:

$$L(\theta) \leq L_{\min} + \varepsilon \quad \text{subject to} \ \sum_{j=1}^{d} |\theta_j| \leq 1$$

- Best classical algorithm runs in time $\tilde{O}(d/\varepsilon^2)$
- Chen & dW'21: quantum algorithm that in time $\tilde{O}\left(\sqrt{d}/\varepsilon^2\right)$

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$
- Finding the exact minimizer is a hard problem, so we typically try to find a vector θ whose loss is not much worse:

$$L(\theta) \leq L_{\min} + \varepsilon \quad \text{subject to} \quad \sum_{j=1}^{d} |\theta_j| \leq 1$$

- Best classical algorithm runs in time $\tilde{O}(d/\varepsilon^2)$
- Chen & dW'21: quantum algorithm that in time $\tilde{O}\left(\sqrt{d}/\varepsilon^2\right)$ by speeding up Frank-Wolfe algorithm using various quantum tricks (min-finding, amplitude estimation, data structures)

Quantum algorithm for Lasso

- Lasso: minimize least-squares $L(\theta)$ subject to $\sum_{j=1}^{d} |\theta_j| \leq 1$
- Finding the exact minimizer is a hard problem, so we typically try to find a vector θ whose loss is not much worse:

$$L(\theta) \leq L_{\min} + \varepsilon \quad \text{subject to} \sum_{j=1}^{d} |\theta_j| \leq 1$$

- Best classical algorithm runs in time $\tilde{O}(d/\varepsilon^2)$
- Chen & dW'21: quantum algorithm that in time $\tilde{O}\left(\sqrt{d}/\varepsilon^2\right)$ by speeding up Frank-Wolfe algorithm using various quantum tricks (min-finding, amplitude estimation, data structures)
- Also proved $\sqrt{d}/\varepsilon^{1.5}$ lower bound for all quantum algorithms. The true bound is still unknown!

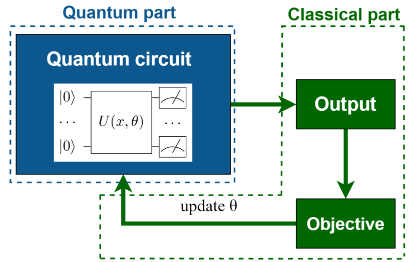
Heuristic methods

Heuristic methods

- Variational methods:
  use classical methods
  to optimize over some
  parametrized circuits

Heuristic methods

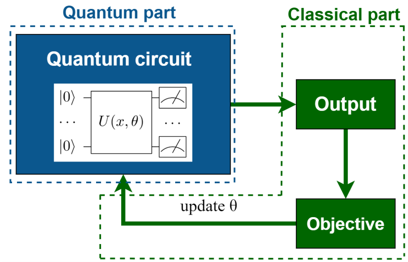
- Variational methods: use classical methods to optimize over some parametrized circuits



https://dkopczyk.quantee.co.uk/wp-content/uploads/2019/05/vc4.png

Heuristic methods

- Variational methods: use classical methods to optimize over some parametrized circuits

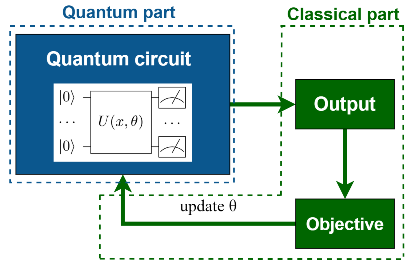


https://dkopczyk.quantee.co.uk/wp-content/uploads/2019/05/vc4.png

- For instance angles in a fixed circuit, or “classical shadows”

Heuristic methods

- Variational methods:
  use classical methods
  to optimize over some
  parametrized circuits

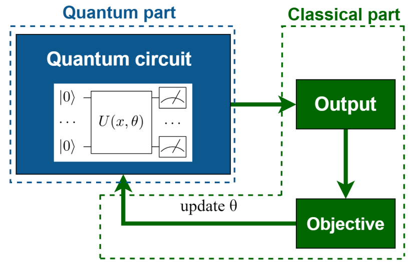


https://dkopczyk.quantee.co.uk/wp-content/uploads/2019/05/vc4.png

- For instance angles in a fixed circuit, or “classical shadows”
- This is similar to neural networks:
  you have some parametrized model where you optimize the
  parameters (the weights of the NN) in some feedback loop

Heuristic methods

- Variational methods:
  use classical methods
  to optimize over some
  parametrized circuits

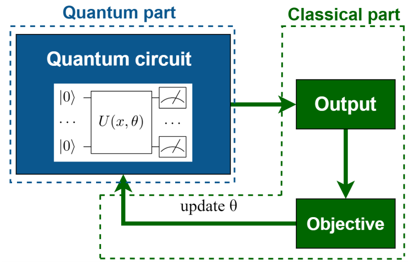


https://dkopczyk.quantee.co.uk/wp-content/uploads/2019/05/vc4.png

- For instance angles in a fixed circuit, or "classical shadows"
- This is similar to neural networks:
  you have some parametrized model where you optimize the
  parameters (the weights of the NN) in some feedback loop
- Like with NN, it's hard to prove things about such methods

Heuristic methods

- Variational methods:
  use classical methods to optimize over some parametrized circuits



https://dkopczyk.quantee.co.uk/wp-content/uploads/2019/05/vc4.png

- For instance angles in a fixed circuit, or “classical shadows”
- This is similar to neural networks:
  you have some parametrized model where you optimize the parameters (the weights of the NN) in some feedback loop
- Like with NN, it's hard to prove things about such methods
- Worse, unlike classical NN we can't run big experiments yet

Summary

- Machine learning = data + optimization

Summary

- Machine learning = data + optimization

- Quantum data (superposition of classical data) can sometimes be useful, but not in distribution-independent PAC learning

Summary

- Machine learning = data + optimization
- Quantum data (superposition of classical data) can sometimes be useful, but not in distribution-independent PAC learning
- "Quantum linear algebra" can be useful to efficiently extract properties of data *as quantum states*

Summary

- Machine learning = data + optimization
- Quantum data (superposition of classical data) can sometimes be useful, but not in distribution-independent PAC learning
- "Quantum linear algebra" can be useful to efficiently extract properties of data *as quantum states*
- There's a growing body of quantum speedups for optimization problems, some rigorous and some heuristic.
  Much of this could be applied to ML problems