Improve your learning with a quantum computer!

Daniel Copeland

Food for Thought
UCSD Math

November 11, 2016

Outline

1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

Outline

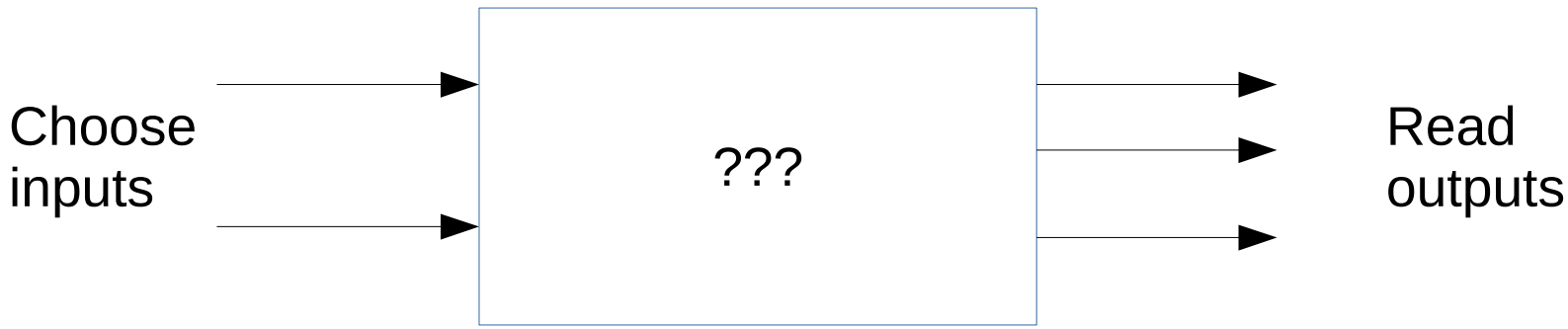
1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

Learning

Today's Answer: Learning is the process of identifying some unknown (or hidden information).

Learning

Today's Answer: Learning is the process of identifying some unknown (or hidden information).



Black boxes/oracles

The hidden information is encoded in some transformation (called the **black box** or **oracle**) which takes some input and transforms it into the output, depending on the hidden information.

Warmup for quantum mechanics

For example suppose some mystery animal is in a black box. We know it's either a cat, a rat or a bat. To find out what creature we have we're allowed to dangle some food in and see if it gets eaten.

Warmup for quantum mechanics

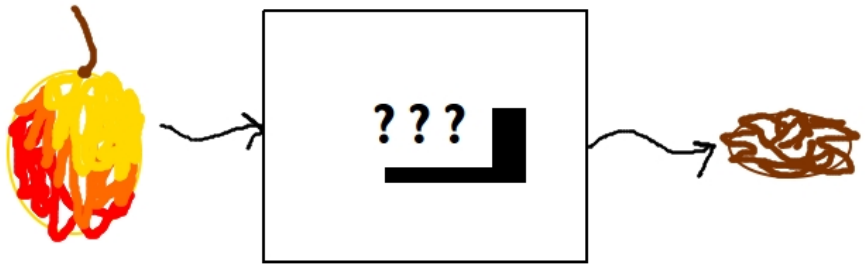
For example suppose some mystery animal is in a black box. We know it's either a cat, a rat or a bat. To find out what creature we have we're allowed to dangle some food in and see if it gets eaten.

For example we know cats and rats don't like fruit, so if we dangle some fruit and it gets eaten then we have a bat.

Warmup for quantum mechanics

For example suppose some mystery animal is in a black box. We know it's either a cat, a rat or a bat. To find out what creature we have we're allowed to dangle some food in and see if it gets eaten.

For example we know cats and rats don't like fruit, so if we dangle some fruit and it gets eaten then we have a bat.



A better example

Consider a "guess the number" game where your friend thinks of a number between 1 and N. You're allowed to guess any number between 1 and N and your friend must answer "higher", "lower" or "equal". You want to identify the number with as few questions as possible.

A better example

Consider a "guess the number" game where your friend thinks of a number between 1 and N. You're allowed to guess any number between 1 and N and your friend must answer "higher", "lower" or "equal". You want to identify the number with as few questions as possible.

Note that to formulate the question properly you should specify the distribution your friend uses to pick a number from 1 to N (we'll always assume uniform sampling!)

Query complexity

We measure the efficacy of an algorithm by the number of queries (= number of times the oracle is questioned).

Query complexity

We measure the efficacy of an algorithm by the number of queries (= number of times the oracle is questioned).

An **upper bound** for query complexity means that you have some algorithm to solve the learning problem. Eg you can think of an algorithm for "guess the number" that uses $\log_2(N)$ queries.

Query complexity

We measure the efficacy of an algorithm by the number of queries (= number of times the oracle is questioned).

An **upper bound** for query complexity means that you have some algorithm to solve the learning problem. Eg you can think of an algorithm for "guess the number" that uses $\log_2(N)$ queries.

A **lower bound** for query complexity means that you can prove some number of queries are required in order to learn the information. For instance in "guess the number" one can show that $\log_2(N)$ queries are required (in the worst case) to learn the hidden number.

Outline

1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

Computation

What is a computer?

Computation

What is a computer?

Today's Answer: A computer is described by a **state** that undergoes certain **changes** until some part of that state is **read by a user**.

Computation

What is a computer?

Today's Answer: A computer is described by a **state** that undergoes certain **changes** until some part of that state is **read by a user**.

To describe a quantum computer we must specify what we mean by state of the computer, the allowable changes (or transformations), and what it means for a user to read a state. The postulates of quantum mechanics give us this.

All a mathematician needs to know about quantum computers.

Postulate 1: states

A **state** of a quantum computer is a unit vector in a finite dimensional Hilbert space.
The state space of a composite quantum system is the tensor product of the state spaces of the component systems.

Remarks

- For quantum computation we assume all state spaces are finite dimensional! So a Hilbert space $=$ $\mathbb{C}$-vector space w/Hermitian inner product.

All a mathematician needs to know about quantum computers.

Postulate 1: states

A **state** of a quantum computer is a unit vector in a finite dimensional Hilbert space.
The state space of a composite quantum system is the tensor product of the state spaces of the component systems.

Remarks

- For quantum computation we assume all state spaces are finite dimensional! So a Hilbert space $=$ $\mathbb{C}$-vector space w/Hermitian inner product.
- Dirac "bra-ket" notation: a vector in a state-space $\mathcal{H}$ is denoted $|v\rangle \in \mathcal{H}$ and called a "ket". The inner product of two states $|v\rangle$ and $|w\rangle$ is denoted $\langle v|w\rangle$. The symbol $\langle v|$ is called a "bra" and denotes the linear transformation $\mathcal{H} \to \mathbb{C}$ which sends $|w\rangle$ to $\langle v|w\rangle$.

Postulate 1: Examples

For example a unit vector in $\mathbb{C}^2$ is called a **qubit**. We usually write a standard basis of $\mathbb{C}^2$ as $|0\rangle$ and $|1\rangle$. So a qubit is a vector

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$.

A two qubit system has state space $\mathbb{C}^2 \otimes \mathbb{C}^2$.

A three k-it system has state space $\mathbb{C}^k \otimes \mathbb{C}^k \otimes \mathbb{C}^k$.

Unitary evolution

Postulate 2: Evolution of states

A quantum computer must transform a state by a unitary operator.

Unitary evolution

Postulate 2: Evolution of states

A quantum computer must transform a state by a unitary operator.

Remarks

A unitary operator is a linear transformation $U : \mathbb{C}^d \to \mathbb{C}^d$ which preserves the inner product:

$$\langle Uv|Uw\rangle = \langle u|v\rangle.$$

In other words $UU^* = U^*U = I$ (where U^* is the adjoint operator, corresponds to conjugate transpose of a matrix).
Unitaries are **reversible**!! (Compare with the classical operation OR!)

Examples

Let $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Then X operates on $\mathbb{C}^2$, ie on qubits. Recall that the standard basis is denoted $\{|0\rangle, |1\rangle\}$ so X sends 0 to 1 and 1 to 0. X is called a NOT gate.

Examples

Let $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Then X operates on $\mathbb{C}^2$, ie on qubits. Recall that the standard basis is denoted $\{|0\rangle, |1\rangle\}$ so X sends 0 to 1 and 1 to 0. X is called a NOT gate.

Let C be the unitary operator on $\mathbb{C}^2 \to \mathbb{C}^2$ defined on basis elements by

$$C|00\rangle = |00\rangle$$
$$C|01\rangle = |01\rangle$$
$$C|10\rangle = |11\rangle$$
$$C|11\rangle = |10\rangle$$

C is called a CONTROLLED-NOT gate.

Quantum circuit diagrams.

Idea: represent unitary operators by boxes, input and output states as wires.

Measurement

Postulate 3: Measurement

A **quantum measurement** of a state in $\mathbb{C}^d$ is described by a choice of orthonormal basis for $\mathbb{C}^d$, $\{|\mu_1\rangle, \ldots, |\mu_d\rangle\}$. If a quantum system is in a state $|\psi\rangle \in \mathbb{C}^d$ and a user performs the measurement described by the ONB above, then the outcome is some number $i \in \{1, \ldots, d\}$ with probability

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

Measurement

Postulate 3: Measurement

A **quantum measurement** of a state in $\mathbb{C}^d$ is described by a choice of orthonormal basis for $\mathbb{C}^d$, $\{|\mu_1\rangle, \ldots, |\mu_d\rangle\}$. If a quantum system is in a state $|\psi\rangle \in \mathbb{C}^d$ and a user performs the measurement described by the ONB above, then the outcome is some number $i \in \{1, \ldots, d\}$ with probability

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

Remarks

Note: since the measurement vectors form an ONB, we have $\sum_i P(i|\psi) = \langle \psi|\psi\rangle = 1$ (which is good).

Example

Consider state space $\mathbb{C}^2$ and a system in state $|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Use measurement basis $|0\rangle, |1\rangle$ (corresponding to outcomes 0 and 1).

$$P(0|-) = |\langle 0|-\rangle|^2 = 1/2$$

and similarly

$$P(1|-) = 1/2.$$

Outline

1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

A quantum black box (or oracle) is some unknown unitary operation, sampled uniformly from some finite set of unitary operations $U_1, \ldots, U_n$ on some state space $\mathbb{C}^d$.

A quantum black box (or oracle) is some unknown unitary operation, sampled uniformly from some finite set of unitary operations $U_1, \ldots, U_n$ on some state space $\mathbb{C}^d$.

A quantum learning algorithm consists of a choice of state, a choice of unitary operators (some of which involve the black box), and a choice of measurement. The goal is to determine which hidden unitary you have access to.

A quantum black box (or oracle) is some unknown unitary operation, sampled uniformly from some finite set of unitary operations $U_1, \ldots, U_n$ on some state space $\mathbb{C}^d$.

A quantum learning algorithm consists of a choice of state, a choice of unitary operators (some of which involve the black box), and a choice of measurement. The goal is to determine which hidden unitary you have access to.

The number of times you use the unknown operation is the number of queries used by the algorithm.

Classical to quantum

Question: How do you turn a classical problem into a quantum one?

Classical to quantum

Question: How do you turn a classical problem into a quantum one?

Answer: Linearize!

Classical data becomes a labeling set for some orthonormal vectors in a vector space. Reversible classical operations are replaced by permutation matrices, which are unitary.

$$\{0, 1\} \to \{|0\rangle, |1\rangle\} \subset \mathbb{C}^2$$

Outline

1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

Grover search

Consider a variant of "guess the number": your friend picks a number between 1 and N. You're allowed to ask: "is your number equal to i?" and your friend says "0" (for no) or "1" (for yes).

Grover search

Consider a variant of "guess the number": your friend picks a number between 1 and N. You're allowed to ask: "is your number equal to i?" and your friend says "0" (for no) or "1" (for yes).

You can always find the number with $N-1$ queries, and in the worst case you also need $N-1$ queries. Asymptotically the query complexity of this problem is $\Theta(N)$.

Quantum version

To feed inputs into a quantum oracle we use a vector space $\mathbb{C}^N$ spanned by $|0\rangle, |1\rangle, \ldots |N\rangle$.

To encode outputs we need another system which holds a single bit of information, or a qubit with state space $\mathbb{C}^2$ spanned by $|0\rangle, |1\rangle$.

Using Postulate 2 our total system is the tensor product $\mathbb{C}^N \otimes \mathbb{C}^2$. The first tensor factor is called the **input register** and the second the **response register.**

The Grover oracles

For each $i \in \{1, \ldots, N\}$ there is a separate oracle. The ith one O_i acts on basis states by

$$O_i|j, b\rangle = \begin{cases} |j, b\rangle & \text{if } j \neq i \\ |j, b+1\rangle & \text{if } j = i \end{cases}$$

The Grover oracles

For each $i \in \{1, \ldots, N\}$ there is a separate oracle. The ith one O_i acts on basis states by

$$O_i|j, b\rangle = \begin{cases} |j, b\rangle & \text{if } j \neq i \\ |j, b+1\rangle & \text{if } j = i \end{cases}$$

Note: $+$ denotes addition mod 2. O_i is a unitary operator (it is a permutation matrix.)

Lov Grover, 1996

Theorem. There exists a quantum algorithm to identify the hidden oracle with success probability $> .95$ with $O(\sqrt{N})$ queries.

Lov Grover, 1996

Theorem. There exists a quantum algorithm to identify the hidden oracle with success probability $> .95$ with $O(\sqrt{N})$ queries.

More precisely we have that for N sufficiently large, there is an algorithm using $\frac{\pi}{4}\sqrt{N}$ queries which succeeds with probability $> .95$.

By repeating the algorithm you can get the error probability to decrease exponentially fast in the number of repeats (some type of Chernoff bound).

Proof.

First we we can diagonalize the oracles.

$$\begin{aligned} O_i|j,-\rangle &= O_i\left(\frac{1}{\sqrt{2}}(|j,0\rangle - |j,1\rangle\right) \\ &= (-1)^{\delta_{i,j}}|j,-\rangle. \end{aligned}$$

Proof.

First we we can diagonalize the oracles.

$$O_i|j,-\rangle = O_i\left(\frac{1}{\sqrt{2}}(|j,0\rangle - |j,1\rangle\right)$$
$$= (-1)^{\delta_{i,j}}|j,-\rangle.$$

Focus just on the subspace with a $|-\rangle$ in the response register, spanned by $|0,-\rangle,\ldots,|N,-\rangle$.

The operator O_i is orthogonal reflection in the hyperplane orthogonal to $|i,-\rangle$!

It takes $|i,-\rangle$ to $-|i,-\rangle$ and fixes everything orthogonal to $|i,-\rangle$.

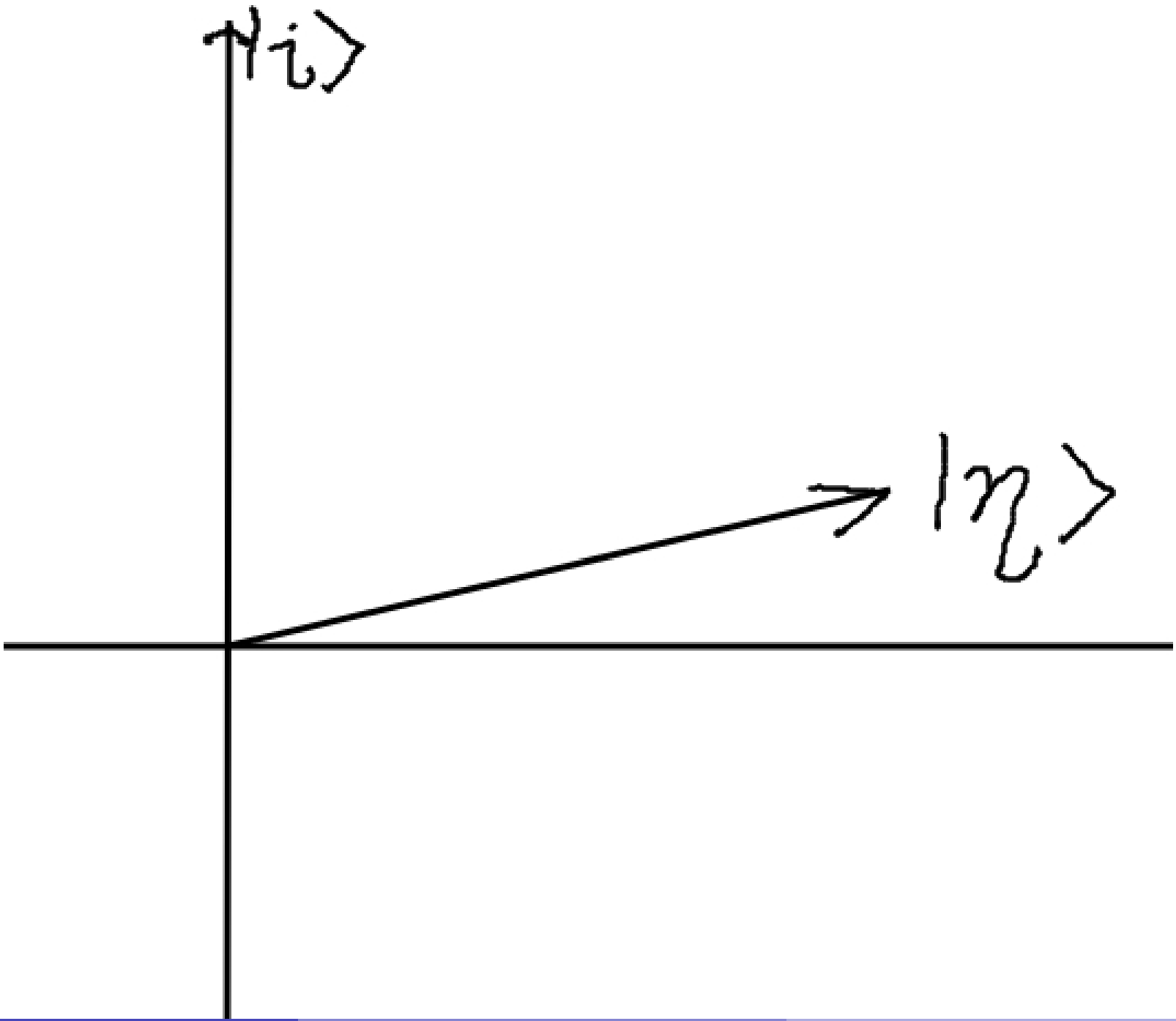
Our input state will be

$$|\eta\rangle := \frac{1}{\sqrt{N}} \sum_{j=1}^{n} |j, -\rangle.$$

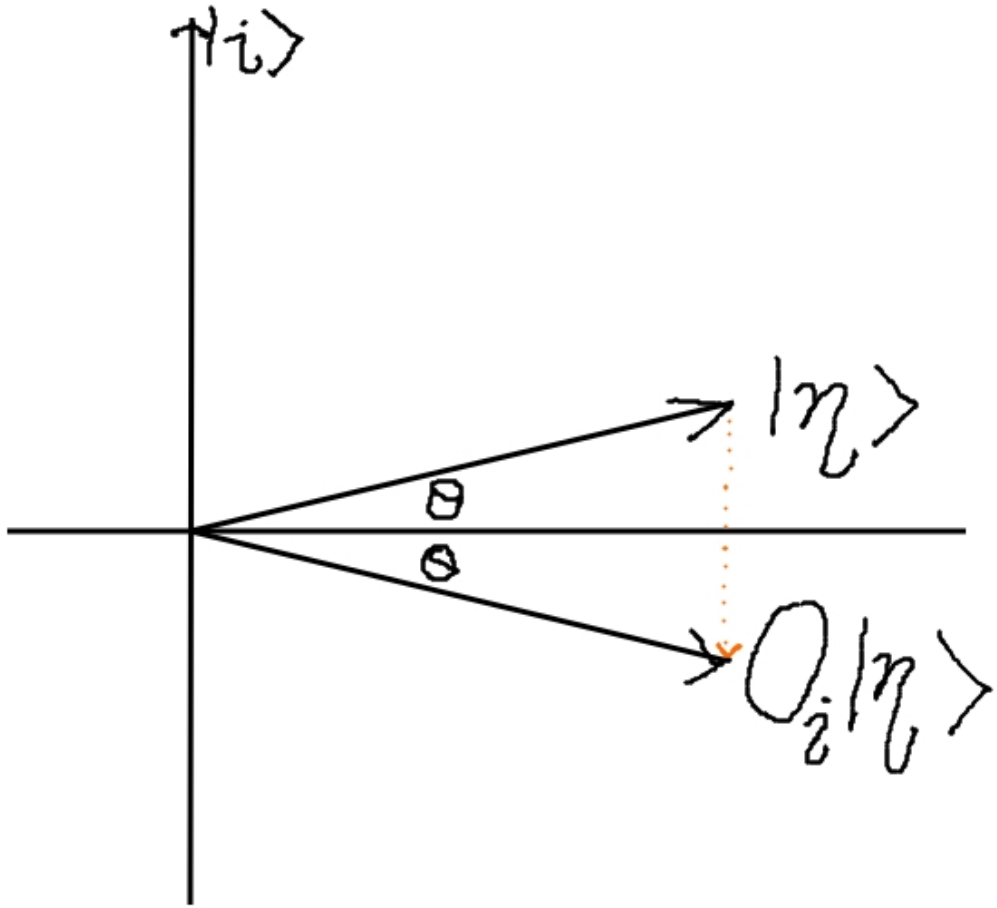
Our input state will be

$$|\eta\rangle := \frac{1}{\sqrt{N}} \sum_{j=1}^{n} |j, -\rangle.$$

Goal: Use the oracle O_i to rotate this state towards the state $|i, -\rangle$ and then use the basis vectors $|j, -\rangle$ for a measurement. The closer we get the state to $|i, -\rangle$, the better our success probability: if our final state is $|\psi\rangle$ then probability of measuring the correct output i is $|\langle\psi|j\rangle|^2$.


|i>
|η>
|η>

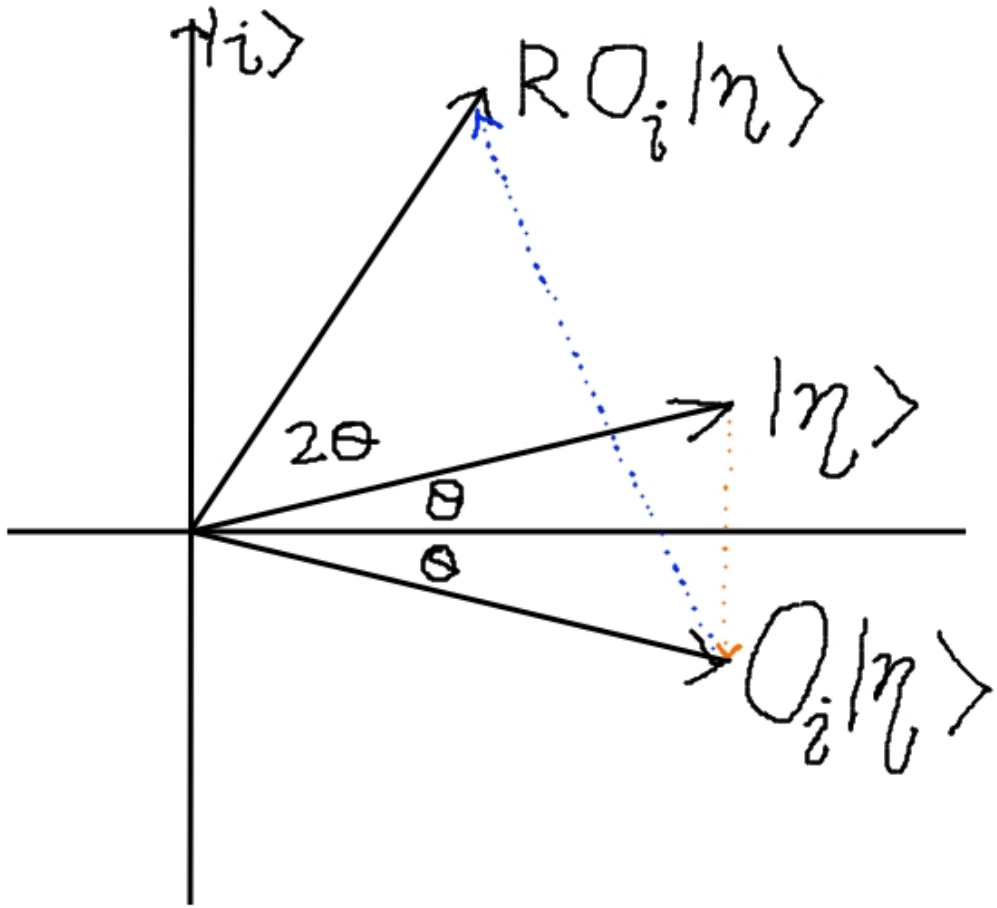
After one oracle call



Note $\sin\Theta = \frac{1}{\sqrt{N}}$. We need to get this closer to $|i\rangle$!

How to do this? Use the unitary operator
$R := -$(reflection about hyperplane orthogonal to $|\eta\rangle$). In our 2d-plane this acts by reflection about $|\eta\rangle$!

How to do this? Use the unitary operator
$R := -$(reflection about hyperplane orthogonal to $|\eta\rangle$). In our 2d-plane this acts by reflection about $|\eta\rangle$!



Composition of these two reflections yields rotation by 2Θ towards $|i\rangle$!

How many such rotations should we perform?

How many such rotations should we perform?

For N large, $\Theta \approx \sin\Theta = \frac{1}{\sqrt{N}}$ and each time we rotate by 2Θ so it takes $\frac{\pi/2}{2/\sqrt{N}} = \frac{\pi}{4}\sqrt{N}$ queries to get within an angle of Θ with $|i\rangle$ (again letting N grow makes this angle shrink).

How many such rotations should we perform?

For N large, $\Theta \approx \sin\Theta = \frac{1}{\sqrt{N}}$ and each time we rotate by 2Θ so it takes $\frac{\pi/2}{2/\sqrt{N}} = \frac{\pi}{4}\sqrt{N}$ queries to get within an angle of Θ with $|i\rangle$ (again letting N grow makes this angle shrink).

Once our state $|\psi\rangle$ is within an angle of Θ with $|i\rangle$ we measure using the vectors $|0\rangle, \ldots, |N-1\rangle$ and the output will be the correct outcome i with probability

$$|\langle i|\psi\rangle|^2 \geq \cos^2(\Theta).$$

Thus the success probability gets arbitrarily close to 1 as N grows. $\square$

AMAZING!

AMAZING!

But get ready for a (possibly) more amazing speedup!

The Bernstein-Vazirani problem

For this problem suppose your friend picks a secret binary string $a \in \mathbb{Z}_2^n$. You are allowed to pick another binary string $x \in \mathbb{Z}_2^n$ and ask "How many 1s do the strings a and x have in common mod 2?".

The Bernstein-Vazirani problem

For this problem suppose your friend picks a secret binary string $a \in \mathbb{Z}_2^n$. You are allowed to pick another binary string $x \in \mathbb{Z}_2^n$ and ask "How many 1s do the strings a and x have in common mod 2?".

Example

- Say $n = 3$ and your friend picks 110. If you guess 101 then they return 1. If you guess 110 then they return 0.
- Classical query complexity? $\Theta(n)$. Hint for upper bound: use the inputs $\{10\ldots0, 01\ldots0, \ldots, 00\ldots1\}$.

Quantum setup

Given two strings $a, x \in \mathbb{Z}_2^n$ let $(a, x) := \sum_i a_i x_i \mod 2$ (which is equal to the number of 1s the strings a and x have in common).

Quantum setup

Given two strings $a, x \in \mathbb{Z}_2^n$ let $(a, x) := \sum_i a_i x_i \mod 2$ (which is equal to the number of 1s the strings a and x have in common).

Linearize the classical problem: we have an input register $\mathbb{C}[\mathbb{Z}_2^n]$ with orthonormal basis $\{|x\rangle : x \in \mathbb{Z}_2^n\}$. (This Hilbert space is $\cong (\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$). We use a one qubit response register $\mathbb{C}^2$ spanned by $|0\rangle, |1, \rangle$.

For each $a \in \mathbb{Z}_2^n$ we have a unitary oracle acting on $\mathbb{C}[\mathbb{Z}_2^n] \otimes \mathbb{C}^2$ by

$$O_a|x, b\rangle = |x, b + (a, x)\rangle.$$

Ethan Bernstein and Umesh Vazirani, 1997

Theorem

There exists a quantum algorithm which identifies the unknown oracle O_a using 1 query and achieves 100% success probability.

Proof

Diagonalize the oracles:

$$O_a|x,-\rangle = (-1)^{(a,x)}|x,-\rangle.$$

Proof

Diagonalize the oracles:

$$O_a|x,-\rangle = (-1)^{(a,x)}|x,-\rangle.$$

Our input state is

$$|\psi\rangle := \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} |x,-\rangle.$$

Proof

Diagonalize the oracles:

$$O_a|x,-\rangle = (-1)^{(a,x)}|x,-\rangle.$$

Our input state is

$$|\psi\rangle := \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{Z}_2^n} |x,-\rangle.$$

Let $|\psi_a\rangle$ be the result we would get from one oracle call if the mystery oracle was O_a:

$$|\psi_a\rangle = O_a|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(a,x)}|x,-\rangle.$$

The point!

The vectors $\{|\psi_a\rangle \mid a \in \mathbb{Z}_2^n\}$ form an orthonormal set!

$$\langle \psi_a | \psi_b \rangle = \frac{1}{2^n} \sum_{x,y \in \mathbb{Z}_2^n} (-1)^{(a,x)} (-1)^{(b,y)} \langle x, - | y, - \rangle$$

The point!

The vectors $\{|\psi_a\rangle \mid a \in \mathbb{Z}_2^n\}$ form an orthonormal set!

$$\langle \psi_a | \psi_b \rangle = \frac{1}{2^n} \sum_{x,y \in \mathbb{Z}_2^n} (-1)^{(a,x)} (-1)^{(b,y)} \langle x, - | y, - \rangle$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(a,x)+(b,x)}$$

The point!

The vectors $\{|\psi_a\rangle \mid a \in \mathbb{Z}_2^n\}$ form an orthonormal set!

$$\langle \psi_a | \psi_b \rangle = \frac{1}{2^n} \sum_{x,y \in \mathbb{Z}_2^n} (-1)^{(a,x)} (-1)^{(b,y)} \langle x, - | y, - \rangle$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(a,x)+(b,x)}$$

$$= \frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(a+b,x)}$$

The last equality can be seen using the formula $(a, x) = \sum_i a_i x_i \mod 2$.

Let $c = a + b$. Then compute

$$\frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(c,x)}$$

.

Let $c = a + b$. Then compute

$$\frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(c,x)}$$

.

Alternating sum: if $c = 000 \ldots 0$ then $(c, x) = 0$ so the sum comes out to 1.

Let $c = a + b$. Then compute

$$\frac{1}{2^n} \sum_{x \in \mathbb{Z}_2^n} (-1)^{(c,x)}$$

.

Alternating sum: if $c = 000\ldots0$ then $(c, x) = 0$ so the sum comes out to 1.

Otherwise we claim the sum is zero. Indeed, if a 1 appears anywhere in c, say in the ith position, then the map

$$x \mapsto x + 00\ldots1\ldots0$$

(with a 1 in the ith position) is an involution of $\mathbb{Z}_2^n$ that reverses the sign in our sum.

Therefore the states $|\psi_a\rangle = O_a|\psi\rangle$ are mutually orthogonal! That means we can use them as measurement vectors, with the vector $|\psi_a\rangle$ corresponding to outcome $a \in \mathbb{Z}_2^n$.

Therefore the states $|\psi_a\rangle = O_a|\psi\rangle$ are mutually orthogonal! That means we can use them as measurement vectors, with the vector $|\psi_a\rangle$ corresponding to outcome $a \in \mathbb{Z}_2^n$.

The probability of correctly measuring outcome a given that the hidden oracle was O_a is therefore

$$P(a|\psi_a) = |\langle\psi_a|\psi_a\rangle|^2 = 1. \qquad \square$$

Fancier proof.

The oracles O_a in the BV problem form a **representation** of the group $\mathbb{Z}_2^n$.
The assignment $a \mapsto O_a$ is a group homomorphism

$$\mathbb{Z}_2^N \mapsto U(\mathbb{C}^{2^n} \otimes \mathbb{C}_2).$$

Fancier proof.

The oracles O_a in the BV problem form a **representation** of the group $\mathbb{Z}_2^n$. The assignment $a \mapsto O_a$ is a group homomorphism

$$\mathbb{Z}_2^N \mapsto U(\mathbb{C}^{2^n} \otimes \mathbb{C}_2).$$

Let V denote this representation. If we decompose V into irreducible subspaces $|x, -\rangle$ we see that 2^n distinct characters appear. Therefore the representation contains a copy of the regular representation, so there is an (inner-product preserving!) injection

$$\mathbb{C}[\mathbb{Z}_2^n] \hookrightarrow V$$

of $\mathbb{C}[\mathbb{Z}_2^n]$-modules.

Fancier proof.

The oracles O_a in the BV problem form a **representation** of the group $\mathbb{Z}_2^n$. The assignment $a \mapsto O_a$ is a group homomorphism

$$\mathbb{Z}_2^N \mapsto U(\mathbb{C}^{2^n} \otimes \mathbb{C}_2).$$

Let V denote this representation. If we decompose V into irreducible subspaces $|x, -\rangle$ we see that 2^n distinct characters appear. Therefore the representation contains a copy of the regular representation, so there is an (inner-product preserving!) injection

$$\mathbb{C}[\mathbb{Z}_2^n] \hookrightarrow V$$

of $\mathbb{C}[\mathbb{Z}_2^n]$-modules.

Let $|\psi\rangle$ image of the identity $e \in \mathbb{Z}_2^n$ under this map. Then the states $O_a|\psi\rangle$ form an orthonormal set (because their preimages do in $\mathbb{C}[\mathbb{Z}_2^n]$). Using these vectors as measurement vectors will determine the value a with probability 1. $\square$

Outline

1. What is learning?
2. What is a quantum computer?
3. What is a quantum learning algorithm?
4. Examples
5. Guess the Permutation!

Some nonabelian problems.

Suppose G is a group of permutations acting on a set Ω. Consider the following learning problem:

Some nonabelian problems.

Suppose G is a group of permutations acting on a set Ω. Consider the following learning problem: :

Your friend picks a mystery group element $g \in G$. You may choose an element $\omega \in \Omega$ for input, to which your friend responds with the element $g \cdot \omega$.

How many guesses do you need to identify g?

Classical query complexity

The number of classical queries required is equal to the smallest tuple $(\omega_1, \ldots, \omega_t)$ such that the response tuple $(g \cdot \omega_1, \ldots, g \cdot \omega_t)$ determines g completely.

Classical query complexity

The number of classical queries required is equal to the smallest tuple $(\omega_1, \ldots, \omega_t)$ such that the response tuple $(g \cdot \omega_1, \ldots, g \cdot \omega_t)$ determines g completely.

This is the same as finding a t-tuple such that if g fixes pointwise every element in the tuple, then g is the identity.

The minimum possible t is called the **minimal base size** of G and has been studied since the 1800s!

Base size

Some amazing things are known about the (minimal) base size $b(G)$ of a permutation group G acting on n elements.

- $2^{b(G)} \leq |G| \leq n^{b(G)}$ and so

Base size

Some amazing things are known about the (minimal) base size $b(G)$ of a permutation group G acting on n elements.

- $2^{b(G)} \leq |G| \leq n^{b(G)}$ and so
- $\frac{\log |G|}{\log n} \leq b(G) \leq \log |G|$. (Historically first use of $b(G)$).

Base size

Some amazing things are known about the (minimal) base size $b(G)$ of a permutation group G acting on n elements.

- $2^{b(G)} \leq |G| \leq n^{b(G)}$ and so
- $\frac{\log |G|}{\log n} \leq b(G) \leq \log |G|$. (Historically first use of $b(G)$).
- Kenneth Blaha 1992: the decision problem "is the base size $\leq k$?" is NP-hard!

Base size

Some amazing things are known about the (minimal) base size $b(G)$ of a permutation group G acting on n elements.

- $2^{b(G)} \leq |G| \leq n^{b(G)}$ and so
- $\frac{\log |G|}{\log n} \leq b(G) \leq \log |G|$. (Historically first use of $b(G)$).
- Kenneth Blaha 1992: the decision problem "is the base size $\leq k$?" is *NP*-hard!
- Ákos Seress 1996: if G is primitive and solvable then $b(G) \leq 4$!

The quantum problem

Linearize: let $\mathbb{C}\Omega$ be the vector spaces spanned by orthonormal basis $\{|\omega\rangle \mid \omega \in \Omega\}$.

The quantum problem

Linearize: let $\mathbb{C}\Omega$ be the vector spaces spanned by orthonormal basis $\{|\omega\rangle \mid \omega \in \Omega\}$.

The oracle corresponding to g is O_g and acts by the permutation matrix defined via

$$O_g|\omega\rangle = |g \cdot \omega\rangle.$$

This defines a representation of $V = \mathbb{C}\Omega$ of G called the **permutation module** associated to the action of G on Ω.

The quantum solution

Define

$$t := \min_i \{\text{every irreducible character of } G \text{ appears in } V^{\otimes i}\}.$$

This exists by a theorem of Brauer and Burnside, using the fact that V is a faithful representation.

The quantum solution

Define

$$t := \min_i \{\text{every irreducible character of } G \text{ appears in } V^{\otimes i}\}.$$

This exists by a theorem of Brauer and Burnside, using the fact that V is a faithful representation.

Theorem (with Bucicovschi, Meyer and Pommersheim)

There exists a quantum algorithm to identify the hidden permutation $g \in G$ with success probability 100% using t queries. Furthermore, t queries are required by any quantum algorithm to identify the permutation with certainty.

The quantum solution

Define

$$t := \min_i \{\text{every irreducible character of } G \text{ appears in } V^{\otimes i}\}.$$

This exists by a theorem of Brauer and Burnside, using the fact that V is a faithful representation.

Theorem (with Bucicovschi, Meyer and Pommersheim)

There exists a quantum algorithm to identify the hidden permutation $g \in G$ with success probability 100% using t queries. Furthermore, t queries are required by any quantum algorithm to identify the permutation with certainty.

In fact we also give the upper and lower bound for the **bounded error query complexity**, which asks for the number of queries required to identify the hidden $g \in G$ with probability $\geq 2/3$.

Invariant of finite group representations

Given a faithful rep'n V of a finite group G we're led to study

$$\gamma_V := \min_K \{\text{every irreducible character of } G \text{ appears in } \oplus_{i=0}^{K} V^{\otimes i}\}.$$

(This is equal to t above for permutation representations).

Invariant of finite group representations

Given a faithful rep'n V of a finite group G we're led to study

$$\gamma_V := \min_K \{\text{every irreducible character of } G \text{ appears in } \oplus_{i=0}^{K} V^{\otimes i}\}.$$

(This is equal to t above for permutation representations).

Very little is known about this invariant! Of course $\gamma_V \leq b(G)$ when V is a permutation module (since quantum query complexity is always $\leq$ classical query complexity). However γ_V can be efficiently calculated if you know the character table of G and the character of V (take inner products).

Invariant of finite group representations

Given a faithful rep'n V of a finite group G we're led to study

$$\gamma_V := \min_K \{\text{every irreducible character of } G \text{ appears in } \oplus_{i=0}^{K} V^{\otimes i}\}.$$

(This is equal to t above for permutation representations).

Very little is known about this invariant! Of course $\gamma_V \leq b(G)$ when V is a permutation module (since quantum query complexity is always $\leq$ classical query complexity). However γ_V can be efficiently calculated if you know the character table of G and the character of V (take inner products).

Hanspeter Kraft: using GAP computed γ_V for every faithful irrep of a bunch of groups, eg every simple group of order ≤ 100000.

You can do it too!

You can do it too!

Any comparison of $b(G)$ vs. γ_V immediately yields a comparison between classical and quantum query complexity of some learning problem. What's the best speedup we can find?

You can do it too!

Any comparison of $b(G)$ vs. γ_V immediately yields a comparison between classical and quantum query complexity of some learning problem. What's the best speedup we can find?

Thank you!