Quantum gradient estimation and its application to quantum reinforcement learning

A. J. Cornelissen[1,2]

[1]Applied Mathematics
Delft University of Technology

[2]QuSoft
Centrum Wiskunde & Informatica

June 5th, 2019

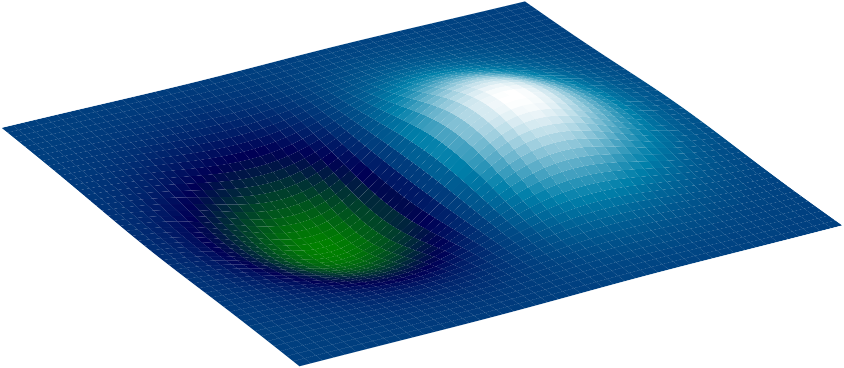
Context

Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.

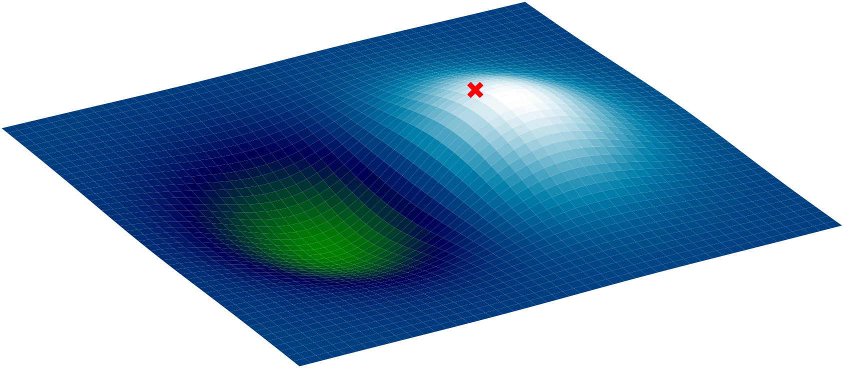
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



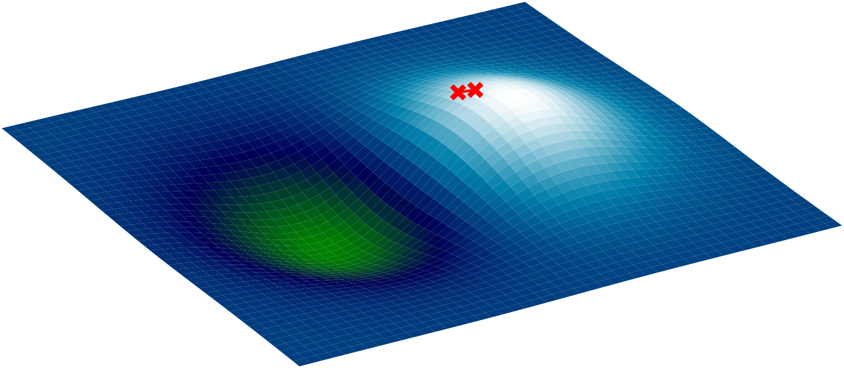
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



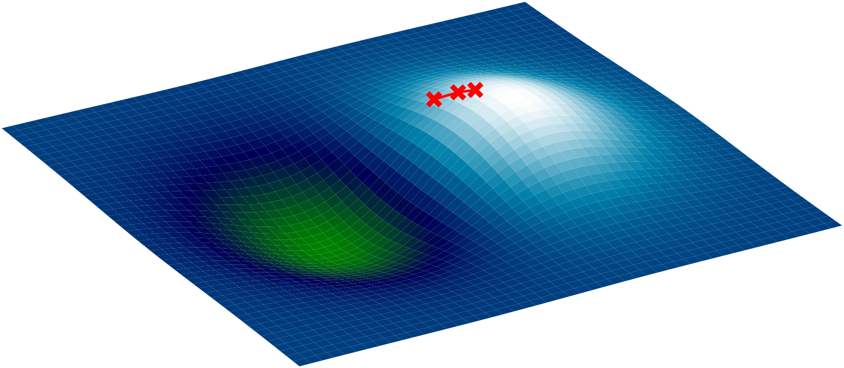
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



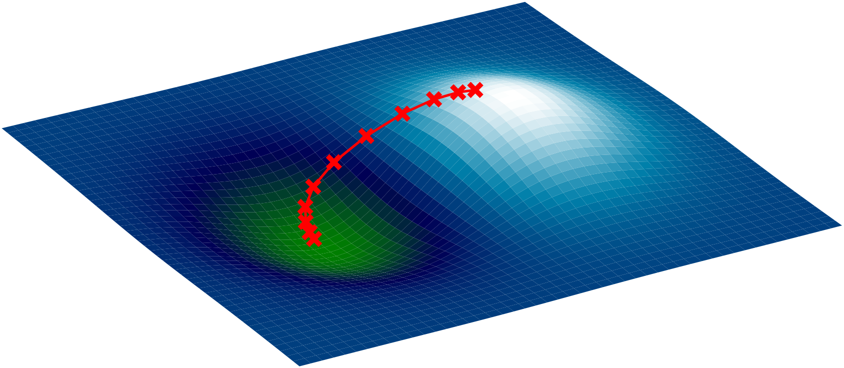
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



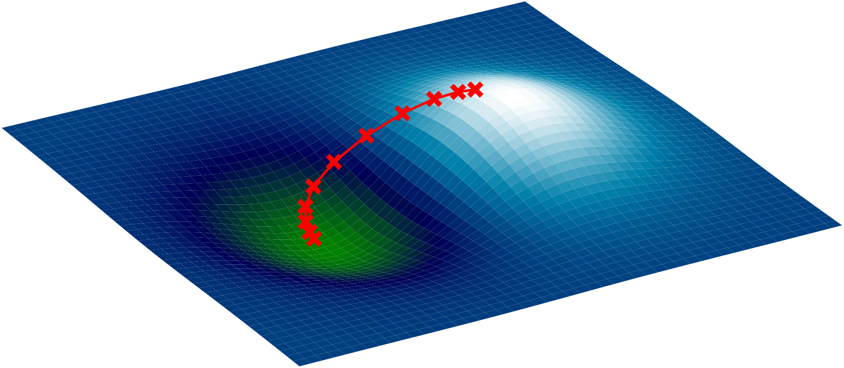
Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



Context

Problem: find the minimum of $f : \mathbb{R}^d \to \mathbb{R}$.



Can we speed up the gradient calculation step when d is large?

Classical gradient estimation

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- So, at least $d + 1$ function evaluations required classically.

Classical gradient estimation

- Easiest case: let $f : \mathbb{R}^d \to \mathbb{R}$ be linear.

$$f(\mathbf{x}) = a + g_1 x_1 + \cdots + g_d x_d, \qquad \nabla f = \begin{bmatrix} g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- Every function evaluation yields a linear constraint on the unknowns.

$$\begin{bmatrix} f(\mathbf{x}^{(1)}) \\ f(\mathbf{x}^{(2)}) \\ \vdots \\ f(\mathbf{x}^{(N)}) \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} a \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

- So, at least $d + 1$ function evaluations required classically.
- **Can we do better with a quantum computer?**

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Visualization of quantum states

Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

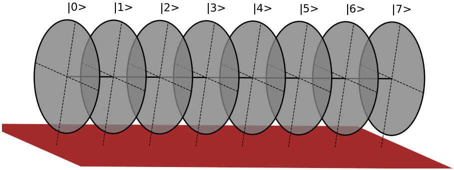
For all j: $|\alpha_j| \leq 1$.

Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

For all j: $|\alpha_j| \leq 1$.

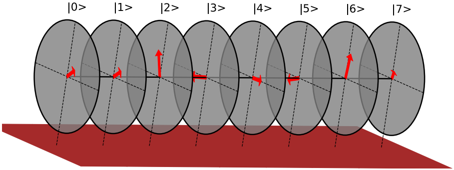


Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

For all j: $|\alpha_j| \leq 1$.



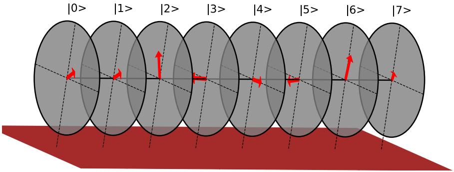
Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

For all j: $|\alpha_j| \leq 1$.

- Quantum gates move the arrows around.



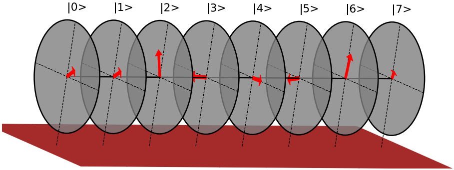
Visualization of quantum states

- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

  For all j: $|\alpha_j| \leq 1$.

- Quantum gates move the arrows around.
- The probability of getting outcome j is the length of the arrow in circle $|j\rangle$.



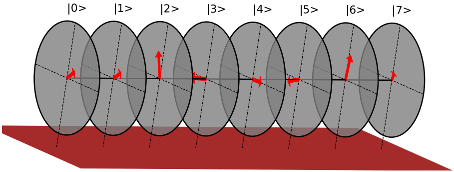
Visualization of quantum states

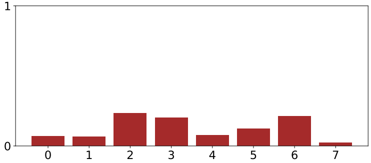
- An n-qubit state $|\psi\rangle$ is a unit vector in $\mathbb{C}^{2^n}$:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle$$

  For all j: $|\alpha_j| \leq 1$.

- Quantum gates move the arrows around.
- The probability of getting outcome j is the length of the arrow in circle $|j\rangle$.





Quantum Fourier transform

Quantum Fourier transform

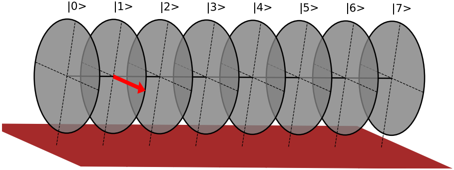
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

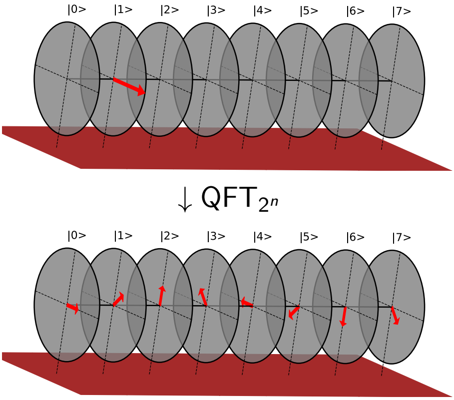
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

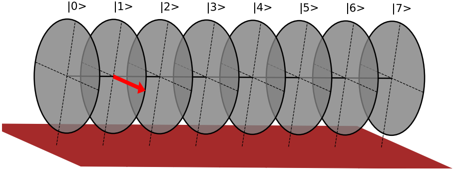
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



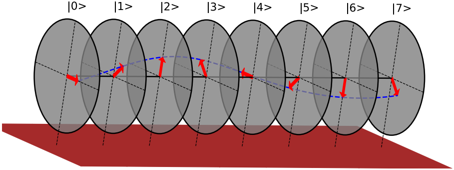
Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$



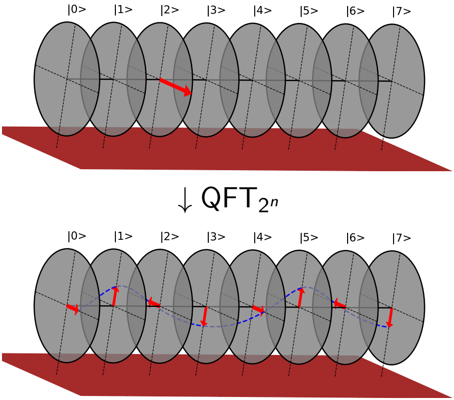
$\downarrow \mathrm{QFT}_{2^n}$



Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

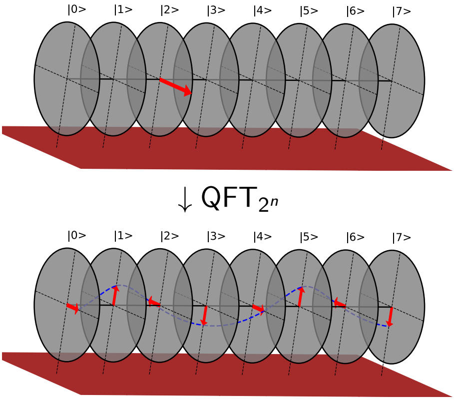


Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.



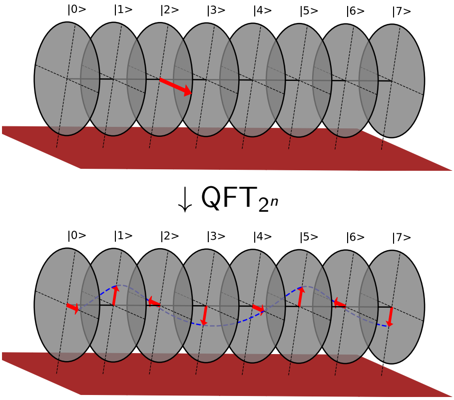
Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}^\dagger_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$



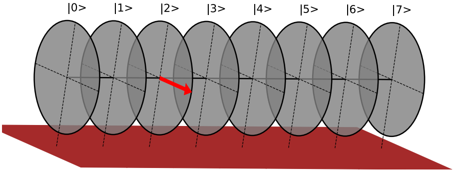
Quantum Fourier transform

- The n-qubit quantum Fourier transform is defined as:

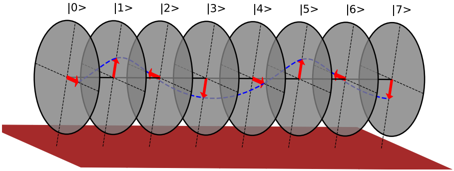
$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

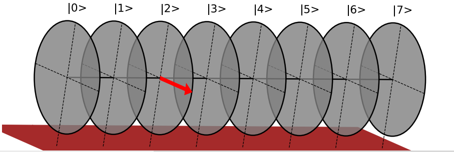
$$\mathrm{QFT}^{\dagger}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$



$\downarrow \mathrm{QFT}_{2^n}$



$\downarrow \mathrm{QFT}^{\dagger}_{2^n}$



Quantum Fourier transform

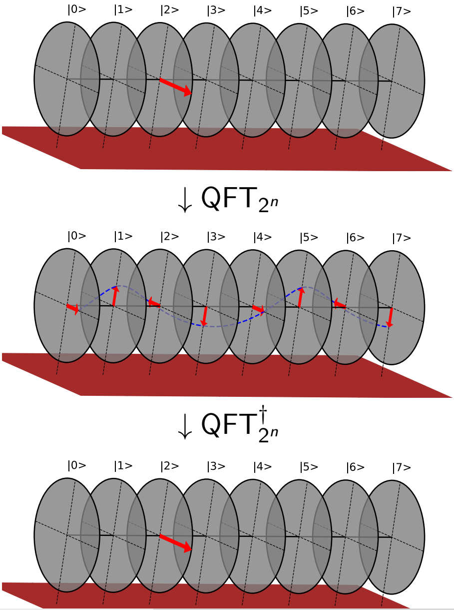
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}_{2^n}^{\dagger} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$

- Efficient implementations available.



Quantum Fourier transform

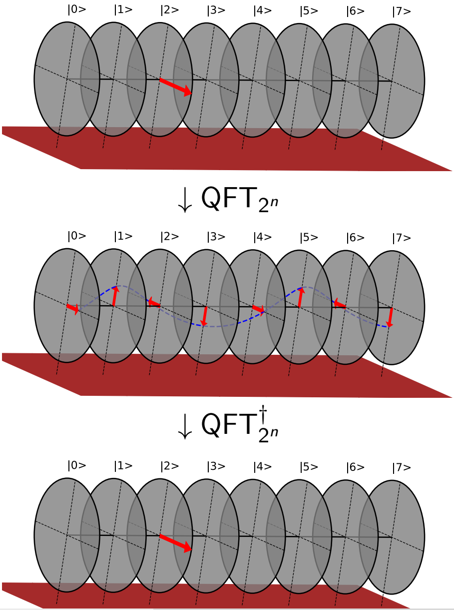
- The n-qubit quantum Fourier transform is defined as:

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi ijk}{2^n}} |k\rangle$$

- The state $\mathrm{QFT}_{2^n}|j\rangle$ can be visualized as a helix making j revolutions.
- The inverse QFT counts the number of revolutions:

$$\mathrm{QFT}_{2^n}^{\dagger} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi ijk}{2^n}} |k\rangle$$

- Efficient implementations available.
- Also works for non-integer revolutions.



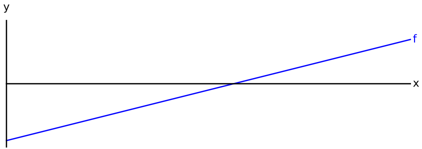
Quantum function evaluations

Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.

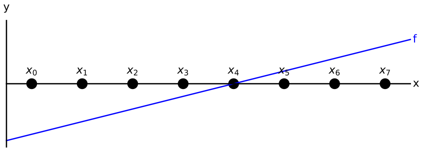
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.



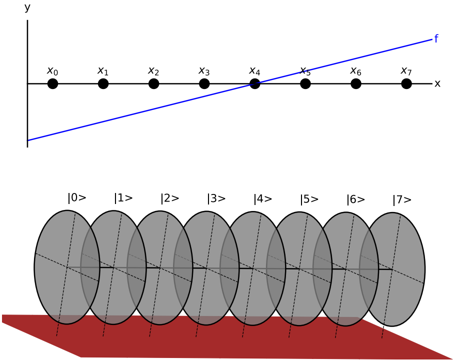
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n - 1}\} \subseteq \mathbb{R}$.



Quantum function evaluations

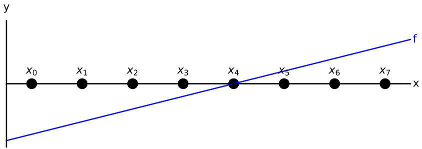
- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.

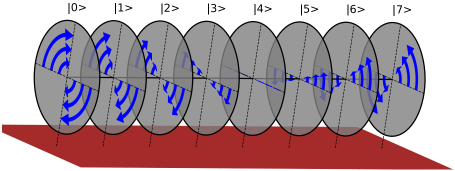


Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$



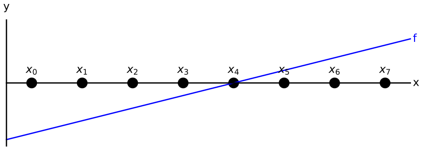


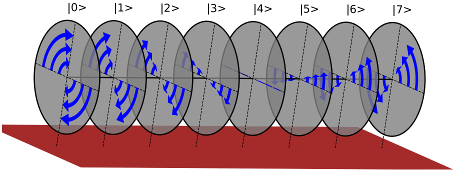
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$

- This is called the phase oracle of f on G.



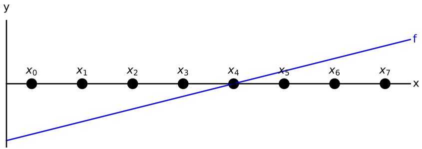


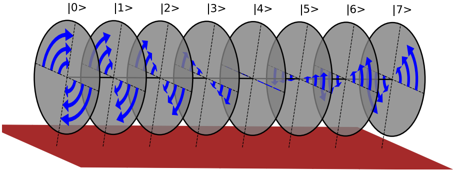
Quantum function evaluations

- Let $f : \mathbb{R} \to \mathbb{R}$.
- Let $G = \{x_0, \ldots, x_{2^n-1}\} \subseteq \mathbb{R}$.
- We associate every state $|j\rangle$ to the point x_j in the domain of f.
- We can evaluate f as follows:

$$O_f : |j\rangle \mapsto e^{if(x_j)} |j\rangle$$

- This is called the phase oracle of f on G.
- One application of this phase oracle is one *quantum function evaluation*.





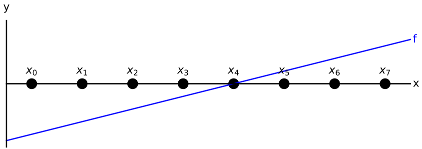
Quantum derivative estimation algorithm for linear functions

Quantum derivative estimation algorithm for linear functions

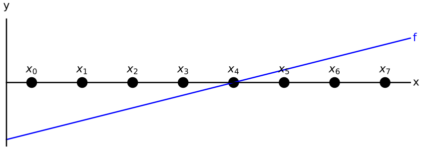
Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.



Quantum derivative estimation algorithm for linear functions



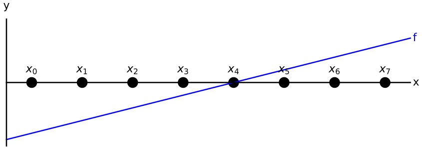
Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

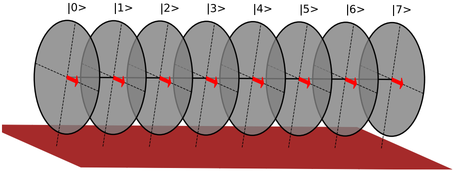
1. Create a uniform superposition over the grid.

Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.

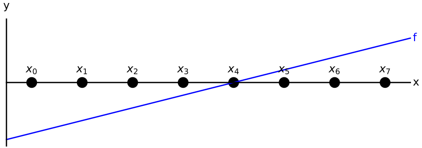


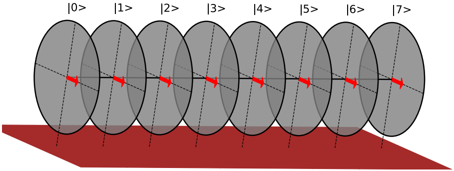


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

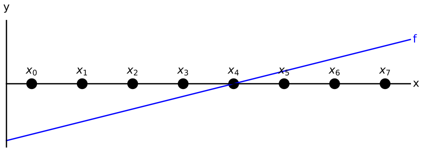


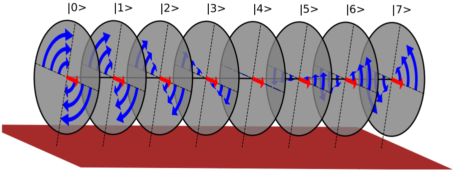


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

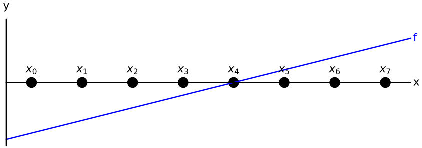


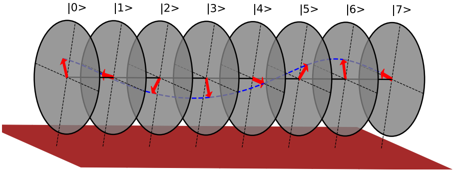


Quantum derivative estimation algorithm for linear functions

Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.

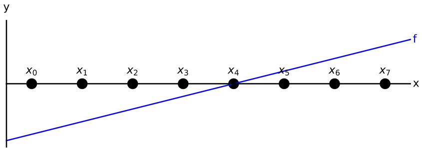


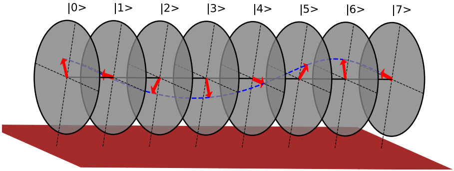


Quantum derivative estimation algorithm for linear functions

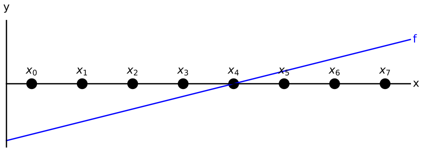
Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.



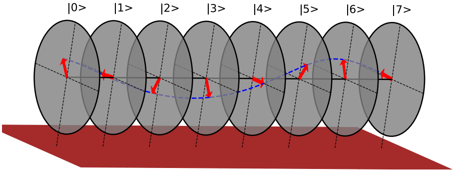


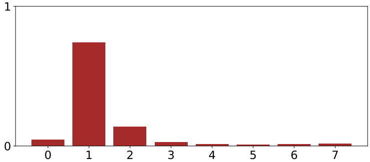
Quantum derivative estimation algorithm for linear functions



Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.



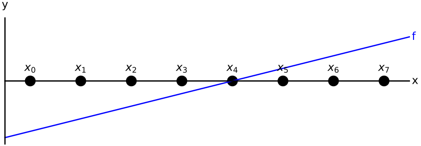


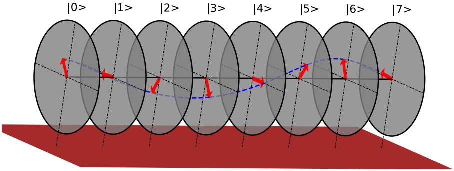
Quantum derivative estimation algorithm for linear functions

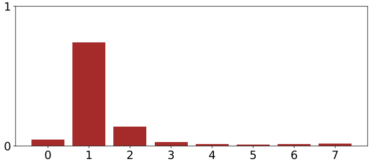
Let $f : \mathbb{R} \to \mathbb{R}$ linear with $|f'| \leq C$.

1. Create a uniform superposition over the grid.
2. Apply the phase oracle O_f.
3. Apply the inverse QFT.
4. Measure.

Generalizes to $f : \mathbb{R}^d \to \mathbb{R}$.
(Jordan, 2004)







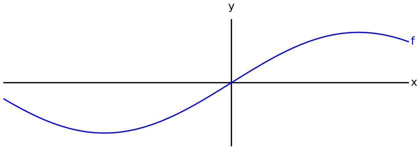
Modifications for non-linear functions

Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.

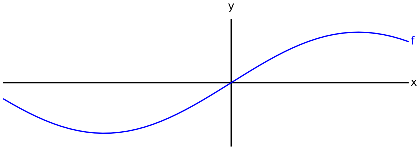
Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.



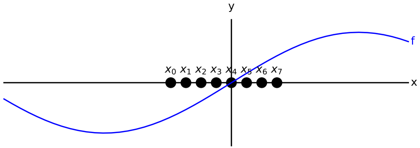
Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \dots, x_{2^n-1}\}$ tight around the origin.



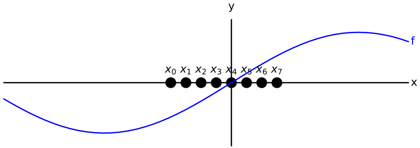
Modifications for non-linear functions

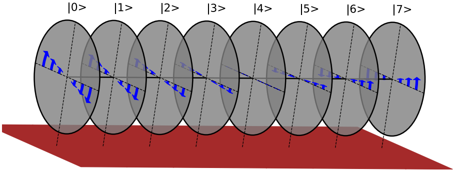
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.



Modifications for non-linear functions

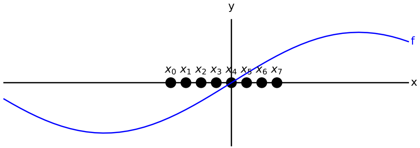
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.

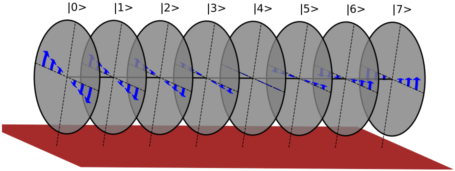




Modifications for non-linear functions

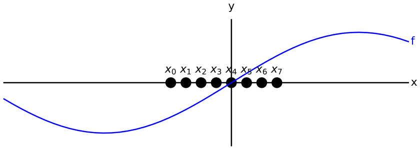
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:

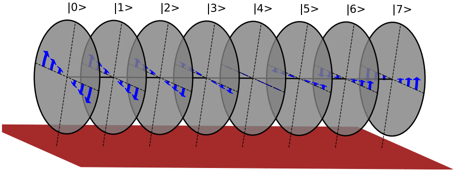




Modifications for non-linear functions

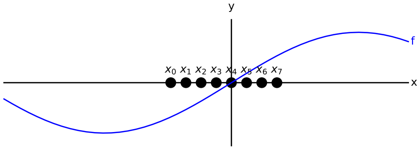
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.

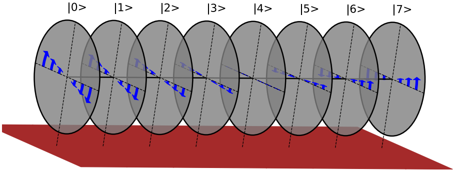




Modifications for non-linear functions

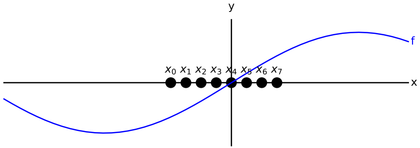
- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.
  - Function evaluations must be very precise.

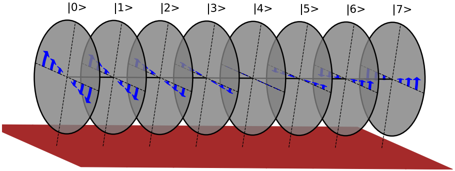




Modifications for non-linear functions

- Let $f : \mathbb{R} \to \mathbb{R}$, want to find $f'(0)$.
- Naive approach: $\{x_0, \ldots, x_{2^n-1}\}$ tight around the origin.
- Problems:
  - Rotations become very small.
  - Function evaluations must be very precise.
- Key idea: central difference method to extend region of approximate linearity.





Central difference method

Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell\mathbf{x})$$

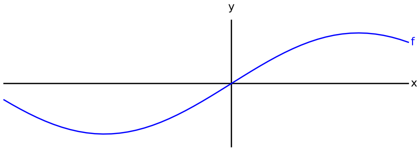
- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$

Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$



- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$

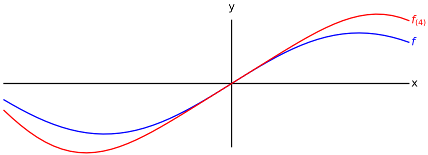
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



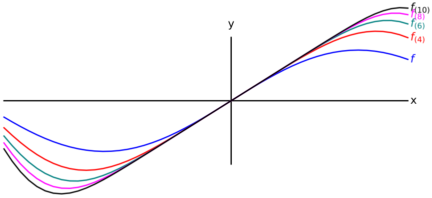
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell\mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



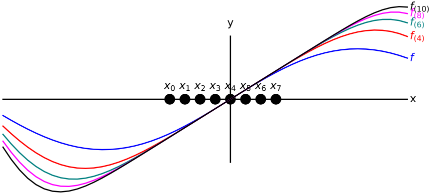
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



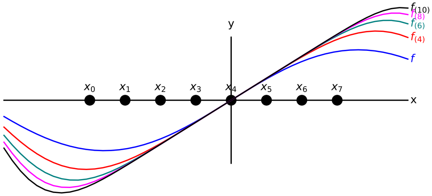
Central difference method

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell\mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0})\cdot\mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right)$$



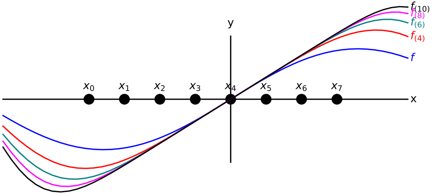
Central difference method

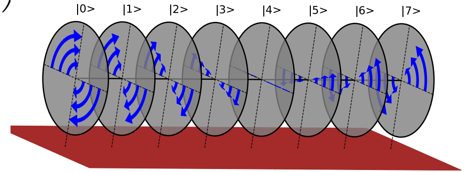
- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right).$$





Central difference method

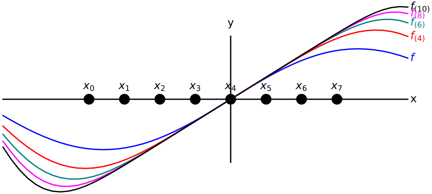
- Let $f : \mathbb{R}^d \to \mathbb{R}$ and $m > 0$. We define:

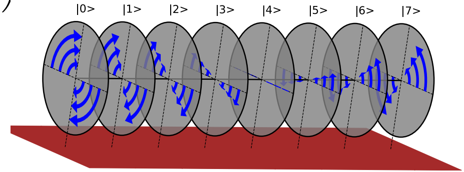
$$\widetilde{f}_{(2m)}(\mathbf{x}) = \sum_{\ell=-m}^{m} a_\ell^{(2m)} f(\ell \mathbf{x})$$

- such that:

$$\widetilde{f}_{(2m)}(\mathbf{x}) = \nabla f(\mathbf{0}) \cdot \mathbf{x} + \mathcal{O}\left(\|\mathbf{x}\|^{2m+1}\right).$$

- One can implement $O_{\widetilde{f}_{(2m)}}$ using $\widetilde{\mathcal{O}}(m)$ queries to O_f. (Gilyén, Arunachalam, Wiebe, 2018)





Smoothness conditions (Gilyén et al. 2018)

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f_{(k)}}$ is linear.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Smoothness condition	Polynomial degree k			
Best known algorithm	$\widetilde{\mathcal{O}}(k)$			
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 2: Gevrey

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Smoothness condition	Polynomial degree k			
Best known algorithm	$\widetilde{\mathcal{O}}(k)$			
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

Smoothness condition	Polynomial degree k			
Best known algorithm	$\widetilde{\mathcal{O}}(k)$			
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k			
Best known algorithm	$\widetilde{\mathcal{O}}(k)$			
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	Gevrey $\sigma = \frac{1}{2}$	Gevrey $\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$			
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f:\mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f:\mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^\sigma)$
Best known lower bound	$\Omega(1)$			

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	Gevrey $\sigma = \frac{1}{2}$	Gevrey $\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$		$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	Gevrey $\sigma = \frac{1}{2}$	Gevrey $\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

Smoothness conditions (Gilyén et al. 2018)

Case 1: Polynomial

- Let $f : \mathbb{R}^d \to \mathbb{R}$ be a multivariate polynomial of total degree k.
- Then $\widetilde{f}_{(k)}$ is linear.
- So gradient estimation takes $\widetilde{\mathcal{O}}(k)$ queries.

Case 2: Gevrey

- Let $f : \mathbb{R}^d \to \mathbb{R}$ have a convergent Taylor series:

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \sum_{\alpha \in [d]^k} \frac{\partial_\alpha f(\mathbf{0})}{k!} \mathbf{x}^\alpha$$

- Let $\sigma \in [\frac{1}{2}, 1]$:

$$|\partial_\alpha f(\mathbf{0})| \leq (k!)^\sigma$$

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	Gevrey $\sigma = \frac{1}{2}$	Gevrey $\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

Analog arithmetics (Gilyén et al., 2018)

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?

Binary oracle
$B_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle |f(\mathbf{x})\rangle$

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?

Binary oracle
$B_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle |f(\mathbf{x})\rangle$

Phase oracle
$O_f : |\mathbf{x}\rangle \mapsto e^{if(\mathbf{x})} |\mathbf{x}\rangle$

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?

Binary oracle
$B_f : |\mathbf{x}\rangle\,|0\rangle \mapsto |\mathbf{x}\rangle\,|f(\mathbf{x})\rangle$

$\xrightarrow[\text{1 query}]{\textbf{Phase kickback}}$

Phase oracle
$O_f : |\mathbf{x}\rangle \mapsto e^{if(\mathbf{x})}\,|\mathbf{x}\rangle$

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?

Binary oracle
$B_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle |f(\mathbf{x})\rangle$

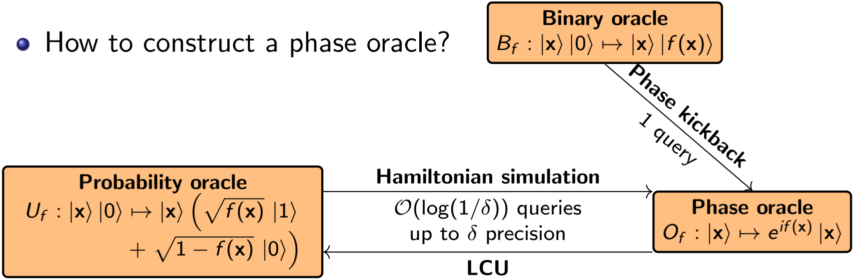
$$\text{Binary oracle} \xrightarrow[\text{1 query}]{\textbf{Phase kickback}} \text{Phase oracle}$$

Probability oracle
$$U_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle \left(\sqrt{f(\mathbf{x})}\,|1\rangle + \sqrt{1 - f(\mathbf{x})}\,|0\rangle\right)$$

Phase oracle
$O_f : |\mathbf{x}\rangle \mapsto e^{if(\mathbf{x})} |\mathbf{x}\rangle$

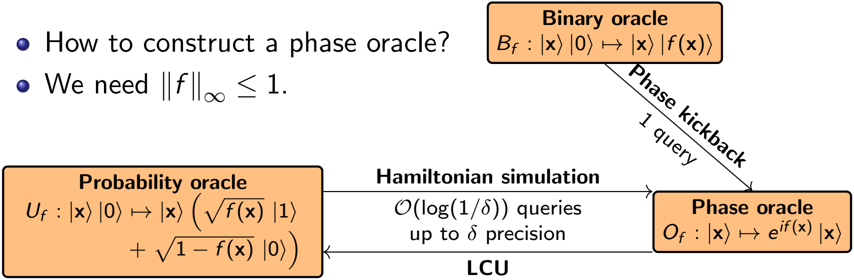
Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?



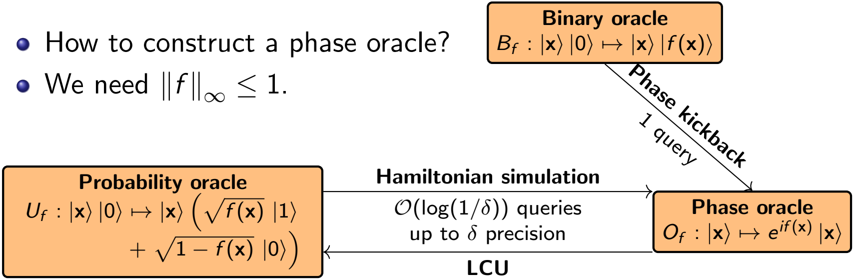
Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?
- We need $\|f\|_\infty \leq 1$.



Analog arithmetics (Gilyén et al., 2018)

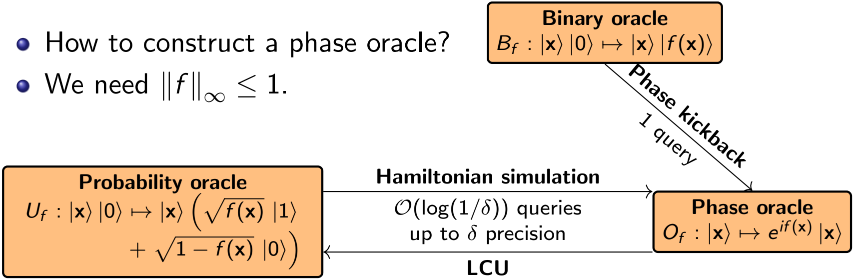
- How to construct a phase oracle?
- We need $\|f\|_\infty \leq 1$.



- Analog arithmetical operations:

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?
- We need $\|f\|_\infty \leq 1$.



- Analog arithmetical operations:
  - **Addition:** consecutive applications of phase oracles.

$$O_f O_g : |\mathbf{x}\rangle \mapsto e^{i(f(\mathbf{x})+g(\mathbf{x}))} |\mathbf{x}\rangle$$

Analog arithmetics (Gilyén et al., 2018)

- How to construct a phase oracle?
- We need $\|f\|_\infty \leq 1$.

Binary oracle
$B_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle |f(\mathbf{x})\rangle$

Phase kickback
1 query

Probability oracle
$U_f : |\mathbf{x}\rangle |0\rangle \mapsto |\mathbf{x}\rangle \left(\sqrt{f(\mathbf{x})}\, |1\rangle + \sqrt{1 - f(\mathbf{x})}\, |0\rangle\right)$

Hamiltonian simulation
$\mathcal{O}(\log(1/\delta))$ queries
up to δ precision
LCU

Phase oracle
$O_f : |\mathbf{x}\rangle \mapsto e^{if(\mathbf{x})} |\mathbf{x}\rangle$

- Analog arithmetical operations:
  - **Addition:** consecutive applications of phase oracles.

$$O_f O_g : |\mathbf{x}\rangle \mapsto e^{i(f(\mathbf{x})+g(\mathbf{x}))} |\mathbf{x}\rangle$$

  - **Multiplication:** consecutive applications of probabilty oracles.

$$(U_f)_1 (U_g)_2 : |\mathbf{x}\rangle |00\rangle \mapsto \sqrt{f(\mathbf{x}) g(\mathbf{x})}\, |\mathbf{x}\rangle |11\rangle + |\perp\rangle$$

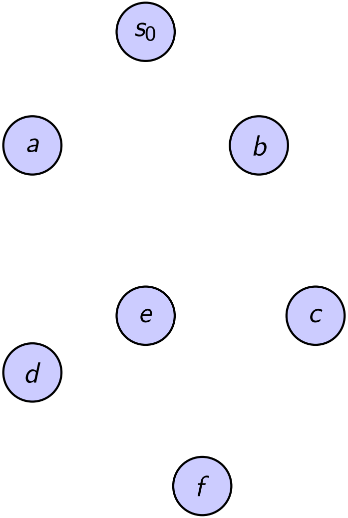
Markov reward processes

Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.

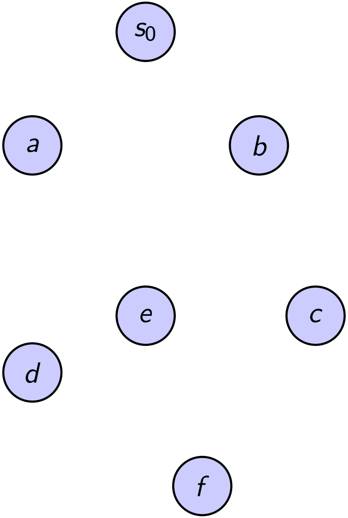
Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.



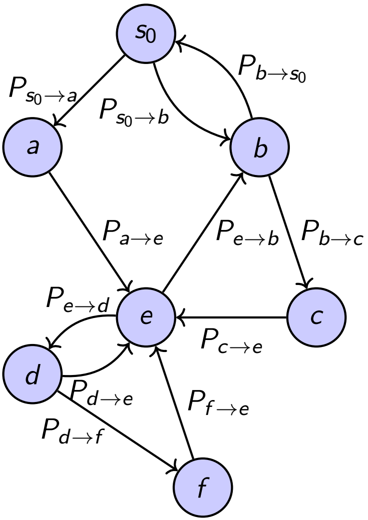
Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.



Markov reward processes

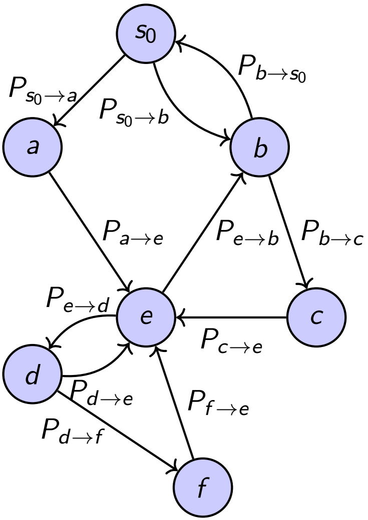
- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.



Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

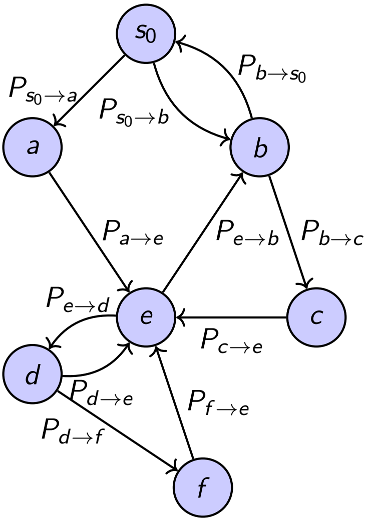


Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

- Let $R(s)$ be the reward that you obtain at state s.

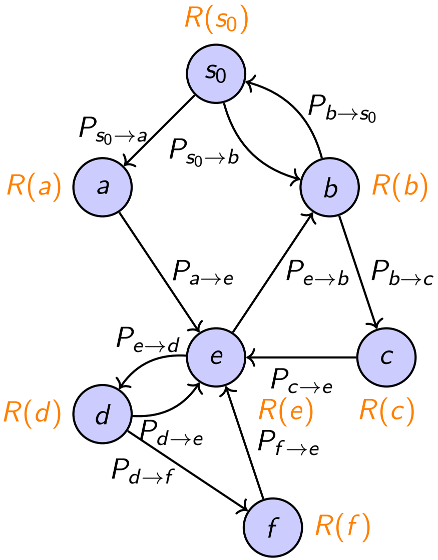


Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

- Let $R(s)$ be the reward that you obtain at state s.



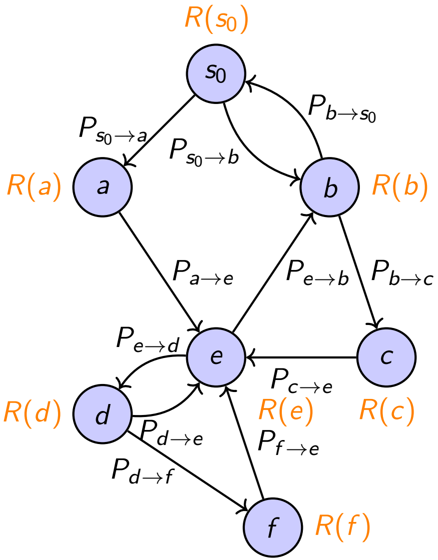
Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

- Let $R(s)$ be the reward that you obtain at state s.

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$



Markov reward processes

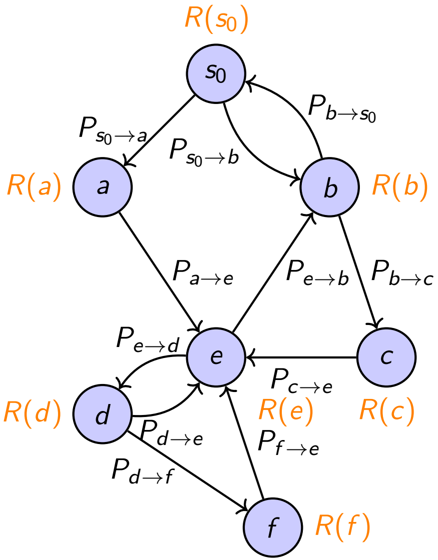
- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

- Let $R(s)$ be the reward that you obtain at state s.

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

- Let $0 < \gamma < 1$.



Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

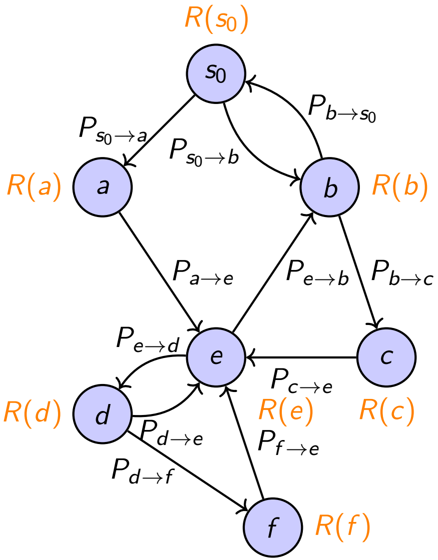
$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

- Let $R(s)$ be the reward that you obtain at state s.

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

- Let $0 < \gamma < 1$.
- Problem: evaluate the value function:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$



Markov reward processes

- Let S be a state space and $s_0 \in S$ some initial state.
- Let $P_{s \to s'}$ be the transition probability function.

$$\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

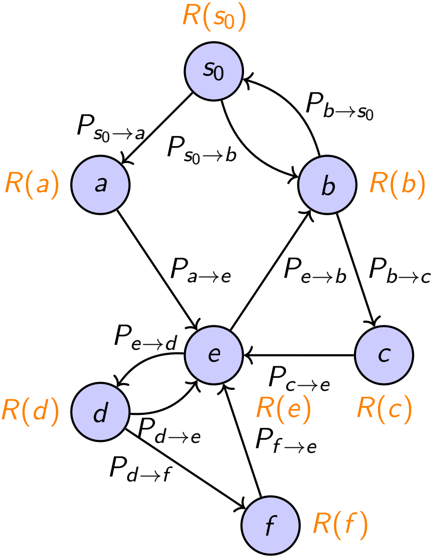
- Let $R(s)$ be the reward that you obtain at state s.

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

- Let $0 < \gamma < 1$.
- Problem: evaluate the value function:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

- Quantum value estimation



Interpretation of the value function

Interpretation of the value function

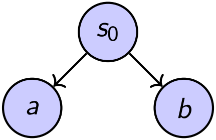
- Let's consider the tree of possible paths.

Interpretation of the value function

- Let's consider the tree of possible paths.

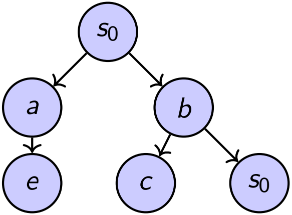
Interpretation of the value function

- Let's consider the tree of possible paths.



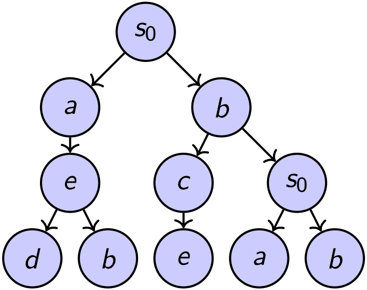
Interpretation of the value function

- Let's consider the tree of possible paths.



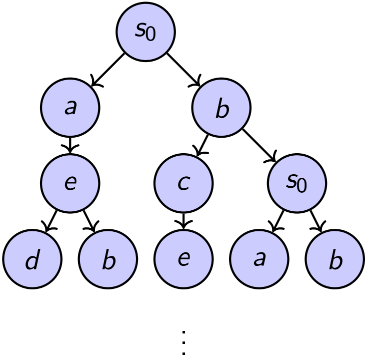
Interpretation of the value function

- Let's consider the tree of possible paths.



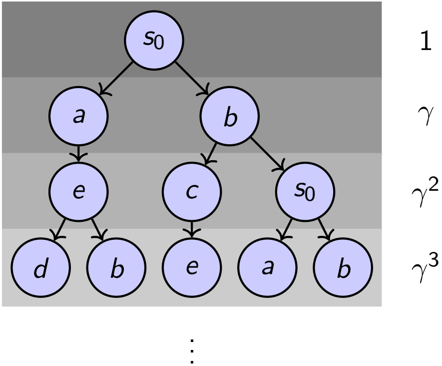
Interpretation of the value function

- Let's consider the tree of possible paths.



Interpretation of the value function

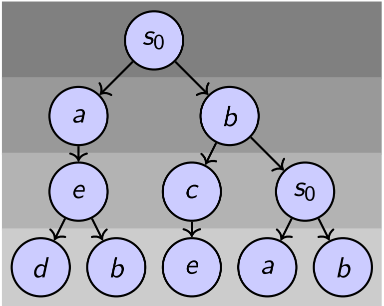
- Let's consider the tree of possible paths.



Interpretation of the value function

- Let's consider the tree of possible paths.
- Cutoff at:

$$T = \Theta\left(\frac{1}{1-\gamma}\log\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)}\right)\right)$$



1

γ

γ^2

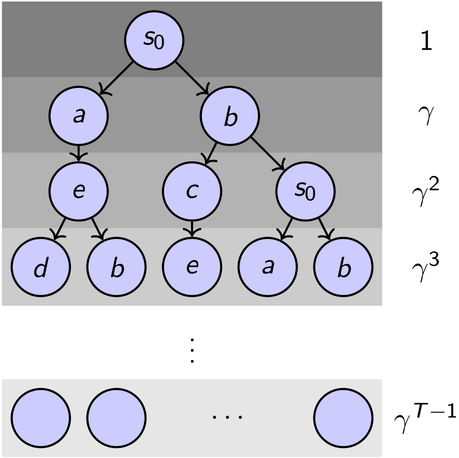
γ^3

$\vdots$

Interpretation of the value function

- Let's consider the tree of possible paths.
- Cutoff at:

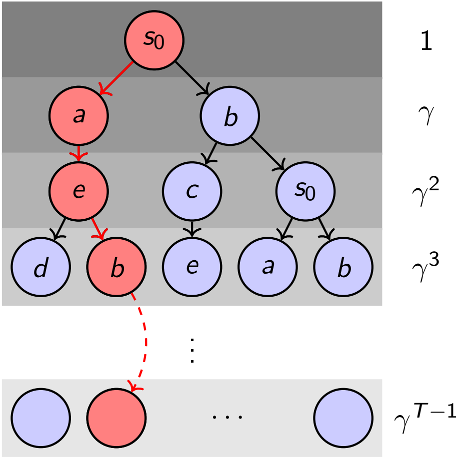
$$T = \Theta\left(\frac{1}{1-\gamma}\log\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)}\right)\right)$$



Interpretation of the value function

- Let's consider the tree of possible paths.
- Cutoff at:

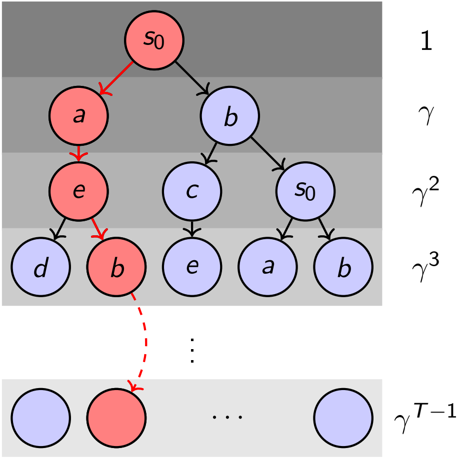
$$T = \Theta\left(\frac{1}{1-\gamma}\log\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)}\right)\right)$$



Interpretation of the value function

- Let's consider the tree of possible paths.
- Cutoff at:

$$T = \Theta\left(\frac{1}{1-\gamma}\log\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)}\right)\right)$$



$$\mathbb{P}(\mathbf{s}) = P_{s_0\to a}\cdot P_{a\to e}\cdot P_{e\to b}\cdots$$
$$R(\mathbf{s}) = R(s_0) + \gamma R(a) + \gamma^2 R(e) + \cdots$$

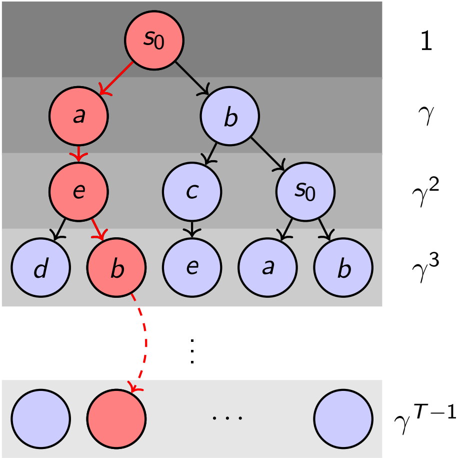
Interpretation of the value function

- Let's consider the tree of possible paths.
- Cutoff at:

$$T = \Theta\left(\frac{1}{1-\gamma}\log\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)}\right)\right)$$

- Value function approximately equal to:

$$V(s_0) = \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s}) + \mathcal{O}(\varepsilon)$$



$$\mathbb{P}(\mathbf{s}) = P_{s_0\to a}\cdot P_{a\to e}\cdot P_{e\to b}\cdots$$

$$R(\mathbf{s}) = R(s_0) + \gamma R(a) + \gamma^2 R(e) + \cdots$$

QVE step 1: Setting up the tree

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$$|s_0\rangle$$

$$|0\rangle$$

$$|0\rangle$$

$$|0\rangle$$

$$\vdots$$

$$|0\rangle$$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle\,|0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}}\,|s'\rangle$

$$\begin{array}{c} |s_0\rangle \\ |0\rangle \\ |0\rangle \\ |0\rangle \\ \vdots \\ |0\rangle \end{array}$$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$|s_0\rangle$

$|0\rangle$

$|0\rangle$

$|0\rangle$

$\vdots$

$|0\rangle$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$$\sum_{s_1 \in S} \sqrt{P_{s_0 \to s_1}} |s_0\rangle |s_1\rangle$$

$$|0\rangle$$

$$|0\rangle$$

$$\vdots$$

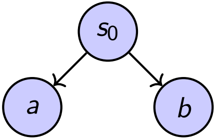
$$|0\rangle$$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$$\sum_{s_1 \in S} \sqrt{P_{s_0 \to s_1}} |s_0\rangle |s_1\rangle$$



$$|0\rangle$$

$$|0\rangle$$

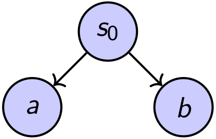
$$\vdots$$

$$|0\rangle$$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$ $\sum_{s_1 \in S} \sqrt{P_{s_0 \to s_1}} |s_0\rangle |s_1\rangle$



$\mathcal{P}$

$|0\rangle$

$|0\rangle$

$\vdots$

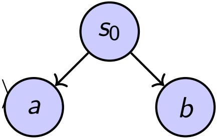
$|0\rangle$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$$\sum_{s_1, s_2 \in S} \sqrt{P_{s_0 \to s_1} P_{s_1 \to s_2}} |s_0\rangle |s_1\rangle |s_2\rangle$$



$\mathcal{P}$

$|0\rangle$

$\vdots$

$|0\rangle$

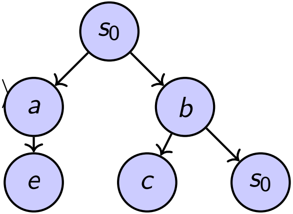
QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle|0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$$\sum_{s_1, s_2 \in S} \sqrt{P_{s_0 \to s_1} P_{s_1 \to s_2}} |s_0\rangle |s_1\rangle |s_2\rangle$$

$\mathcal{P}$



$|0\rangle$

$\vdots$

$|0\rangle$

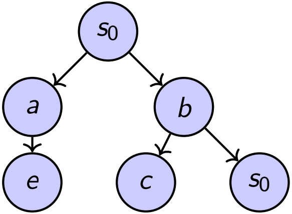
QVE step 1: Setting up the tree

$$\text{We have access to: } \mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$$

$\mathcal{P}$

$$\sum_{\mathbf{s} \in S^2} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle$$

$\mathcal{P}$



$|0\rangle$

$\vdots$

$|0\rangle$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$\mathcal{P}$

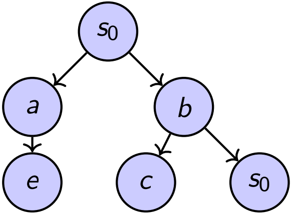
$\mathcal{P}$

$$\sum_{\mathbf{s} \in S^2} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle$$

$$|0\rangle$$

$$\vdots$$

$$|0\rangle$$



QVE step 1: Setting up the tree

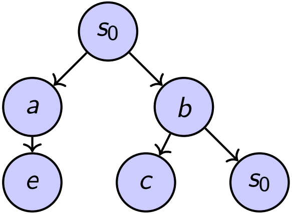
We have access to: $\mathcal{P} : |s\rangle\,|0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}}\,|s'\rangle$

$\mathcal{P}$

$\mathcal{P}$

$\mathcal{P}$

$$\sum_{\mathbf{s} \in S^3} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle$$



$$\vdots$$

$$|0\rangle$$

QVE step 1: Setting up the tree

We have access to: $\mathcal{P} : |s\rangle|0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}}|s'\rangle$

$\mathcal{P}$

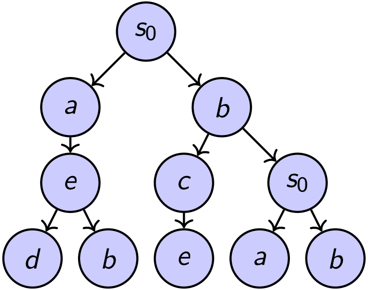
$\mathcal{P}$

$\mathcal{P}$

$$\sum_{\mathbf{s} \in S^3} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle$$

$\vdots$

$|0\rangle$



QVE step 1: Setting up the tree

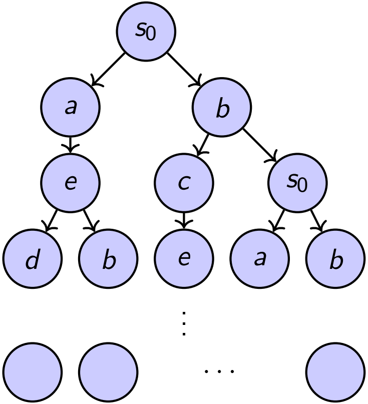
We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$\mathcal{P}$

$\mathcal{P}$

$$\sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle$$



QVE step 1: Setting up the tree

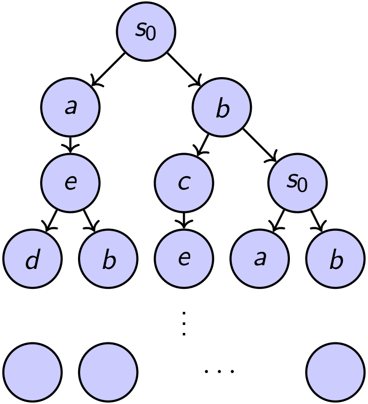
We have access to: $\mathcal{P} : |s\rangle |0\rangle \mapsto |s\rangle \sum_{s' \in S} \sqrt{P_{s \to s'}} |s'\rangle$

$\mathcal{P}$

$\mathcal{P}$

$\mathcal{P}$

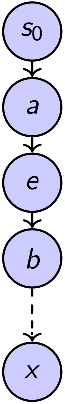
$$\sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle$$



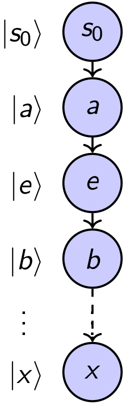
$\overline{\mathcal{P}} : |s_0\rangle |0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle$ with $T-1$ queries to $\mathcal{P}$

QVE step 2: Calculating the reward for a path

QVE step 2: Calculating the reward for a path



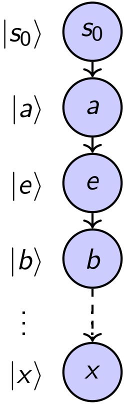
QVE step 2: Calculating the reward for a path



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$



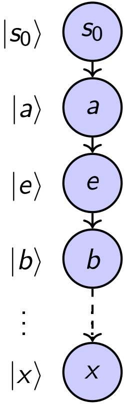
QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$



QVE step 2: Calculating the reward for a path

We have access to:

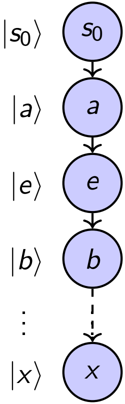
$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$|s_0\rangle$

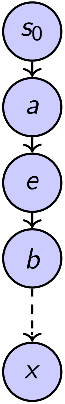
$|a\rangle$

$|e\rangle$

$|b\rangle$

$\vdots$

$|x\rangle$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

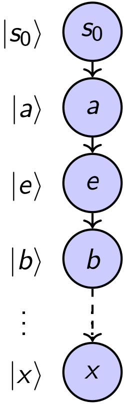
Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

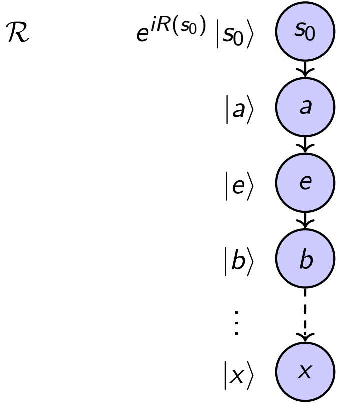
$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$ $\quad e^{iR(s_0)} |s_0\rangle$

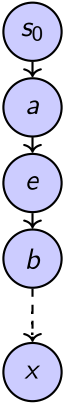
$\mathcal{R}^\gamma$ $\quad |a\rangle$

$|e\rangle$

$|b\rangle$

$\vdots$

$|x\rangle$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)}\, |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)}\, |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$ $\qquad e^{iR(s_0)} |s_0\rangle$

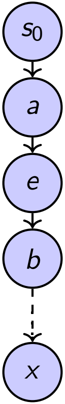
$\mathcal{R}^\gamma$ $\qquad e^{i\gamma R(a)} |a\rangle$

$|e\rangle$

$|b\rangle$

$\vdots$

$|x\rangle$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

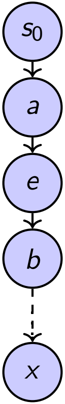
Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$	$e^{iR(s_0)} \|s_0\rangle$	
$\mathcal{R}^\gamma$	$e^{i\gamma R(a)} \|a\rangle$	
$\mathcal{R}^{\gamma^2}$	$e^{i\gamma^2 R(e)} \|e\rangle$	
	$\|b\rangle$	
	$\vdots$	
	$\|x\rangle$	



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)}\,|s\rangle$$

Convert:

$$|s\rangle\,|0\rangle \mapsto \sqrt{R(s)}\,|s\rangle\,|1\rangle + |\perp\rangle$$

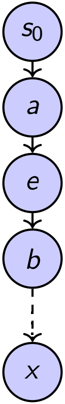
Now multiply by c:

$$|s\rangle\,|00\rangle \mapsto \sqrt{cR(s)}\,|s\rangle\,|11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)}\,|s\rangle$$

$\mathcal{R}$	$e^{iR(s_0)}\,|s_0\rangle$	s_0
$\mathcal{R}^{\gamma}$	$e^{i\gamma R(a)}\,|a\rangle$	a
$\mathcal{R}^{\gamma^2}$	$e^{i\gamma^2 R(e)}\,|e\rangle$	e
$\mathcal{R}^{\gamma^3}$	$e^{i\gamma^3 R(b)}\,|b\rangle$	b
	$\vdots$	
	$|x\rangle$	x



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)}\, |s\rangle |1\rangle + |\perp\rangle$$

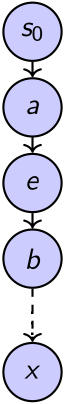
Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)}\, |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$	$e^{iR(s_0)} \|s_0\rangle$
$\mathcal{R}^{\gamma}$	$e^{i\gamma R(a)} \|a\rangle$
$\mathcal{R}^{\gamma^2}$	$e^{i\gamma^2 R(e)} \|e\rangle$
$\mathcal{R}^{\gamma^3}$	$e^{i\gamma^3 R(b)} \|b\rangle$
$\vdots$	$\vdots$
$\mathcal{R}^{\gamma^{T-1}}$	$e^{i\gamma^{T-1} R(x)} \|x\rangle$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

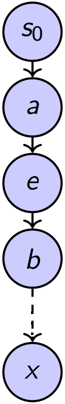
Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$		$\|s_0\rangle$
$\mathcal{R}^{\gamma}$		$\|a\rangle$
$\mathcal{R}^{\gamma^2}$	$e^{iR(\mathbf{s})}$	$\|e\rangle$
$\mathcal{R}^{\gamma^3}$		$\|b\rangle$
$\vdots$		$\vdots$
$\mathcal{R}^{\gamma^{T-1}}$		$\|x\rangle$



QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

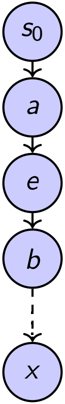
Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$		$|s_0\rangle$
$\mathcal{R}^{\gamma}$		$|a\rangle$
$\mathcal{R}^{\gamma^2}$	$e^{iR(\mathbf{s})}$	$|e\rangle$
$\mathcal{R}^{\gamma^3}$		$|b\rangle$
$\vdots$		$\vdots$
$\mathcal{R}^{\gamma^{T-1}}$		$|x\rangle$



$$|\mathbf{s}\rangle \mapsto e^{iR(\mathbf{s})} |\mathbf{s}\rangle \qquad \text{with } \widetilde{\mathcal{O}}(T) \text{ queries to } \mathcal{R}$$

QVE step 2: Calculating the reward for a path

We have access to:

$$\mathcal{R} : |s\rangle \mapsto e^{iR(s)} |s\rangle$$

Convert:

$$|s\rangle |0\rangle \mapsto \sqrt{R(s)} |s\rangle |1\rangle + |\perp\rangle$$

Now multiply by c:

$$|s\rangle |00\rangle \mapsto \sqrt{cR(s)} |s\rangle |11\rangle + |\perp\rangle$$

Convert back:

$$\mathcal{R}^c : |s\rangle \mapsto e^{icR(s)} |s\rangle$$

$\mathcal{R}$ $\quad |s_0\rangle$

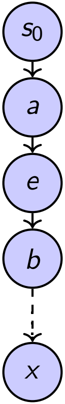
$\mathcal{R}^{\gamma}$ $\quad |a\rangle$

$\mathcal{R}^{\gamma^2}$ $\quad e^{iR(\mathbf{s})}$ $\quad |e\rangle$

$\mathcal{R}^{\gamma^3}$ $\quad |b\rangle$

$\vdots$ $\quad \vdots$

$\mathcal{R}^{\gamma^{T-1}}$ $\quad |x\rangle$



$$|\mathbf{s}\rangle \mapsto e^{iR(\mathbf{s})} |\mathbf{s}\rangle \qquad \text{with } \widetilde{\mathcal{O}}(T) \text{ queries to } \mathcal{R}$$

$$\overline{\mathcal{R}} : |\mathbf{s}\rangle |0\rangle \mapsto \sqrt{R(\mathbf{s})} |\mathbf{s}\rangle |1\rangle + |\perp\rangle \qquad \text{with } \widetilde{\mathcal{O}}(T) \text{ queries to } \mathcal{R}$$

Quantum value estimation algorithm

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle |0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle |0\rangle \mapsto \sqrt{R(\mathbf{s})} |\mathbf{s}\rangle |1\rangle + |\perp\rangle$$

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle\,|0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle\,|0\rangle \mapsto \sqrt{R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\bot\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s})$$

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle\,|0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle\,|0\rangle \mapsto \sqrt{R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s})$$

- Composing $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$ yields:

$$|s_0\rangle\,|0\rangle^{\otimes(T-1)}\,|0\rangle$$

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle\,|0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle\,|0\rangle \mapsto \sqrt{R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s})$$

- Composing $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$ yields:

$$|s_0\rangle\,|0\rangle^{\otimes(T-1)}\,|0\rangle \overset{\overline{\mathcal{P}}}{\mapsto} \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle\,|0\rangle$$

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle |0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle |0\rangle \mapsto \sqrt{R(\mathbf{s})} |\mathbf{s}\rangle |1\rangle + |\perp\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s} \in S^{T-1}} \mathbb{P}(\mathbf{s}) R(\mathbf{s})$$

- Composing $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$ yields:

$$|s_0\rangle |0\rangle^{\otimes(T-1)} |0\rangle \overset{\overline{\mathcal{P}}}{\mapsto} \sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})} |\mathbf{s}\rangle |0\rangle \overset{\overline{\mathcal{R}}}{\mapsto} \sum_{\mathbf{s} \in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s}) R(\mathbf{s})} |\mathbf{s}\rangle |1\rangle + |\perp\rangle$$

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle\,|0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle\,|0\rangle \mapsto \sqrt{R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s})$$

- Composing $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$ yields:

$$|s_0\rangle\,|0\rangle^{\otimes(T-1)}\,|0\rangle \overset{\overline{\mathcal{P}}}{\mapsto} \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle\,|0\rangle \overset{\overline{\mathcal{R}}}{\mapsto} \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- One can obtain the value function with amplitude estimation up to precision ε with

$$\widetilde{\mathcal{O}}\left(\frac{T|R|_{\max}}{\varepsilon(1-\gamma)}\right) = \widetilde{\mathcal{O}}\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)^2}\right)$$

queries to $\mathcal{P}$ and $\mathcal{R}$, quadratically faster than classical algorithms.

Quantum value estimation algorithm

- We have constructed the following operations with $\widetilde{\mathcal{O}}(T)$ queries to $\mathcal{P}$ and $\mathcal{R}$:

$$\overline{\mathcal{P}} : |s_0\rangle\,|0\rangle^{\otimes(T-1)} \mapsto \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle \qquad \text{and} \qquad \overline{\mathcal{R}} : |\mathbf{s}\rangle\,|0\rangle \mapsto \sqrt{R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- We want to calculate:

$$V(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \approx \sum_{\mathbf{s}\in S^{T-1}} \mathbb{P}(\mathbf{s})R(\mathbf{s})$$

- Composing $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$ yields:

$$|s_0\rangle\,|0\rangle^{\otimes(T-1)}\,|0\rangle \overset{\overline{\mathcal{P}}}{\mapsto} \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})}\,|\mathbf{s}\rangle\,|0\rangle \overset{\overline{\mathcal{R}}}{\mapsto} \sum_{\mathbf{s}\in S^{T-1}} \sqrt{\mathbb{P}(\mathbf{s})R(\mathbf{s})}\,|\mathbf{s}\rangle\,|1\rangle + |\perp\rangle$$

- One can obtain the value function with amplitude estimation up to precision ε with

$$\widetilde{\mathcal{O}}\left(\frac{T|R|_{\max}}{\varepsilon(1-\gamma)}\right) = \widetilde{\mathcal{O}}\left(\frac{|R|_{\max}}{\varepsilon(1-\gamma)^2}\right)$$

  queries to $\mathcal{P}$ and $\mathcal{R}$, quadratically faster than classical algorithms.
- This is essentially optimal for $\varepsilon \downarrow 0$, $|R|_{\max} \to \infty$, $\gamma \uparrow 1$.

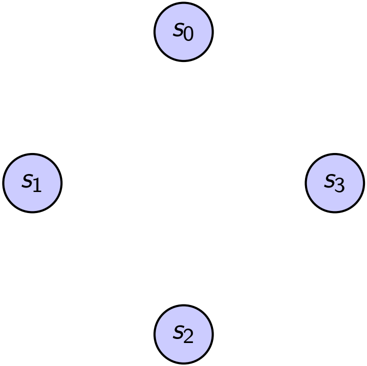
Markov decision processes

Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.

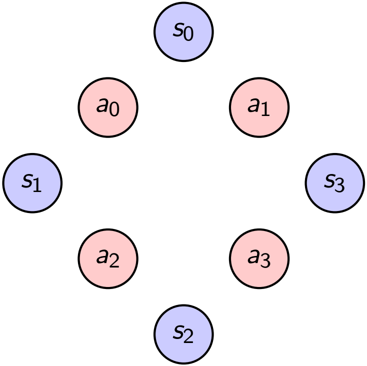
Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.



Markov decision processes

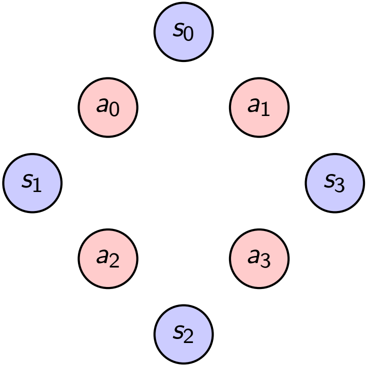
- Let S be a set of states, A be a set of actions, and s_0 the initial state.



Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s \to a}$ be a policy.

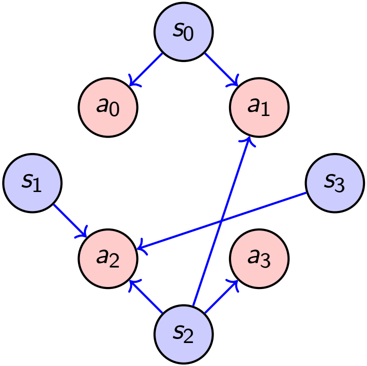
$$\Pi : |s\rangle |0\rangle \mapsto |s\rangle \sum_{a \in A} \sqrt{\pi_{s \to a}} |a\rangle$$



Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.

$$\Pi : |s\rangle|0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}}\,|a\rangle$$



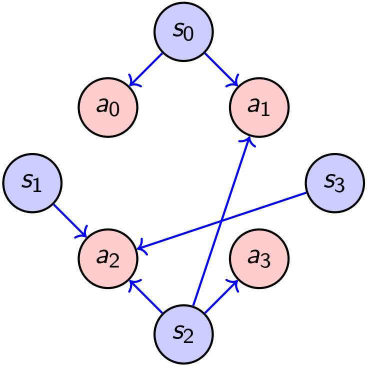
Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.

$$\Pi : |s\rangle |0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}}\, |a\rangle$$

- Let $P_{s,a\to s'}$ be the transition function.

$$\mathcal{P} : |s\rangle |a\rangle |0\rangle \mapsto |s\rangle |a\rangle \sum_{s'\in S} \sqrt{P_{s,a\to s'}}\, |s'\rangle$$



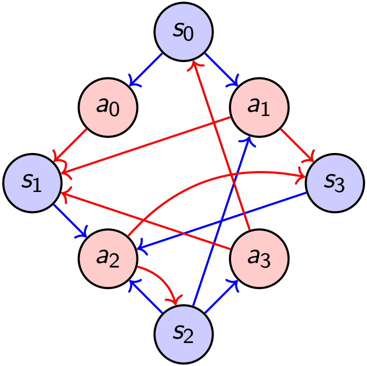
Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.

$$\Pi : |s\rangle |0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}} |a\rangle$$

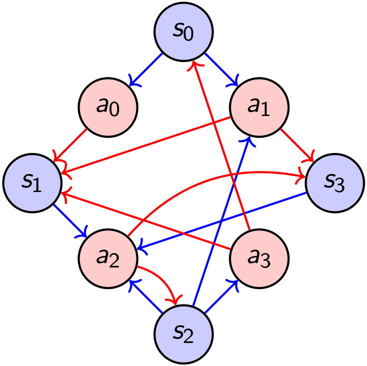
- Let $P_{s,a\to s'}$ be the transition function.

$$\mathcal{P} : |s\rangle |a\rangle |0\rangle \mapsto |s\rangle |a\rangle \sum_{s'\in S} \sqrt{P_{s,a\to s'}} |s'\rangle$$



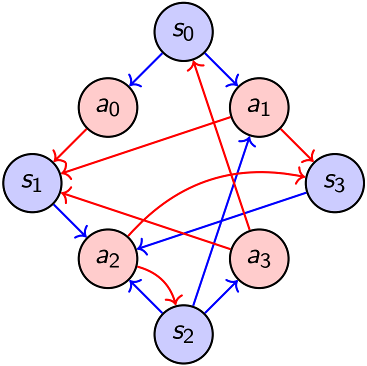
Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.
$$\Pi : |s\rangle |0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}} |a\rangle$$
- Let $P_{s,a\to s'}$ be the transition function.
$$\mathcal{P} : |s\rangle |a\rangle |0\rangle \mapsto |s\rangle |a\rangle \sum_{s'\in S} \sqrt{P_{s,a\to s'}} |s'\rangle$$
- Let $R(s,a)$ be the reward function.
$$\mathcal{R} : |s\rangle |a\rangle \mapsto e^{iR(s,a)} |s\rangle |a\rangle$$



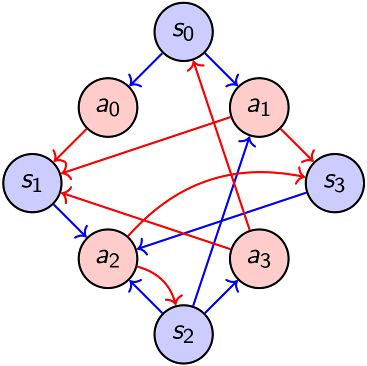
Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.
$$\Pi : |s\rangle|0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}}\,|a\rangle$$
- Let $P_{s,a\to s'}$ be the transition function.
$$\mathcal{P} : |s\rangle|a\rangle|0\rangle \mapsto |s\rangle|a\rangle \sum_{s'\in S} \sqrt{P_{s,a\to s'}}\,|s'\rangle$$
- Let $R(s,a)$ be the reward function.
$$\mathcal{R} : |s\rangle|a\rangle \mapsto e^{iR(s,a)}|s\rangle|a\rangle$$
- Let $0 < \gamma < 1$.



Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s \to a}$ be a policy.
$$\Pi : |s\rangle |0\rangle \mapsto |s\rangle \sum_{a \in A} \sqrt{\pi_{s \to a}} |a\rangle$$
- Let $P_{s,a \to s'}$ be the transition function.
$$\mathcal{P} : |s\rangle |a\rangle |0\rangle \mapsto |s\rangle |a\rangle \sum_{s' \in S} \sqrt{P_{s,a \to s'}} |s'\rangle$$
- Let $R(s,a)$ be the reward function.
$$\mathcal{R} : |s\rangle |a\rangle \mapsto e^{iR(s,a)} |s\rangle |a\rangle$$
- Let $0 < \gamma < 1$.
- Goal: find the policy π such that:
$$V(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right]$$
is maximized.



Markov decision processes

- Let S be a set of states, A be a set of actions, and s_0 the initial state.
- Let $\pi_{s\to a}$ be a policy.

$$\Pi : |s\rangle|0\rangle \mapsto |s\rangle \sum_{a\in A} \sqrt{\pi_{s\to a}}\,|a\rangle$$

- Let $P_{s,a\to s'}$ be the transition function.

$$\mathcal{P} : |s\rangle|a\rangle|0\rangle \mapsto |s\rangle|a\rangle \sum_{s'\in S} \sqrt{P_{s,a\to s'}}\,|s'\rangle$$

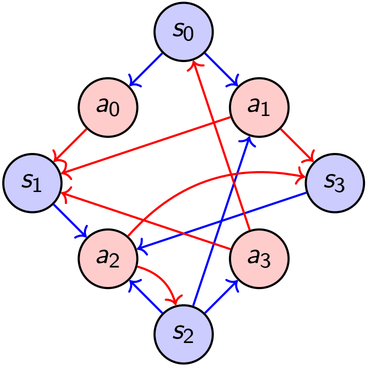
- Let $R(s,a)$ be the reward function.

$$\mathcal{R} : |s\rangle|a\rangle \mapsto e^{iR(s,a)}|s\rangle|a\rangle$$

- Let $0 < \gamma < 1$.
- Goal: find the policy π such that:

$$V(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)\right]$$

  is maximized.
- Quantum policy optimization



Quantum policy optimization – ideas

Quantum policy optimization – ideas

- *Classical* idea: define some parameter space $\mathbb{R}^d$, and embed it into the space of policies.

$$\kappa : \theta \mapsto \pi^{(\theta)}$$

Quantum policy optimization – ideas

- *Classical* idea: define some parameter space $\mathbb{R}^d$, and embed it into the space of policies.
$$\kappa : \theta \mapsto \pi^{(\theta)}$$
- Now, we define the following function $f : \mathbb{R}^d \to \mathbb{R}$:
$$f : \theta \mapsto V(\kappa(\theta))$$

Quantum policy optimization – ideas

- *Classical* idea: define some parameter space $\mathbb{R}^d$, and embed it into the space of policies.
$$\kappa : \theta \mapsto \pi^{(\theta)}$$
- Now, we define the following function $f : \mathbb{R}^d \to \mathbb{R}$:
$$f : \theta \mapsto V(\kappa(\theta))$$
- We wish to find the maximum of this function $\Rightarrow$ gradient ascent!

Quantum policy optimization – ideas

- *Classical* idea: define some parameter space $\mathbb{R}^d$, and embed it into the space of policies.
$$\kappa : \theta \mapsto \pi^{(\theta)}$$
- Now, we define the following function $f : \mathbb{R}^d \to \mathbb{R}$:
$$f : \theta \mapsto V(\kappa(\theta))$$
- We wish to find the maximum of this function $\Rightarrow$ gradient ascent!
- Construction of a phase oracle of f can be done by modifying the quantum value estimation algorithm.

Quantum policy optimization – ideas

- *Classical* idea: define some parameter space $\mathbb{R}^d$, and embed it into the space of policies.
$$\kappa : \theta \mapsto \pi^{(\theta)}$$
- Now, we define the following function $f : \mathbb{R}^d \to \mathbb{R}$:
$$f : \theta \mapsto V(\kappa(\theta))$$
- We wish to find the maximum of this function $\Rightarrow$ gradient ascent!
- Construction of a phase oracle of f can be done by modifying the quantum value estimation algorithm.
- How well does it work?

Performance analysis (work in progress)

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.
  - Smoothness of V is in general of Gevrey type with $\sigma = 1$.

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.
  - Smoothness of V is in general of Gevrey type with $\sigma = 1$.
  - Smoothness of compositions of two Gevrey functions is in general Gevrey with $\sigma = 1$.

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.
  - Smoothness of V is in general of Gevrey type with $\sigma = 1$.
  - Smoothness of compositions of two Gevrey functions is in general Gevrey with $\sigma = 1$.
  - What happens when we use polynomial approximations to V?

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.
  - Smoothness of V is in general of Gevrey type with $\sigma = 1$.
  - Smoothness of compositions of two Gevrey functions is in general Gevrey with $\sigma = 1$.
  - What happens when we use polynomial approximations to V?
  - What happens if we restrict to policies that are sufficiently non-deterministic?

Performance analysis (work in progress)

- Performance depends on the smoothness of the objective function.

$$f(\theta) = V(\kappa(\theta))$$

  - Smoothness of κ can be freely chosen.
  - Smoothness of V is in general of Gevrey type with $\sigma = 1$.
  - Smoothness of compositions of two Gevrey functions is in general Gevrey with $\sigma = 1$.
  - What happens when we use polynomial approximations to V?
  - What happens if we restrict to policies that are sufficiently non-deterministic?
- Even classical gradient ascent with quantum value evaluation as subroutine provides speed-up!

Summary & outlook

Summary & outlook

1. Quantum gradient estimation:

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	Gevrey $\sigma = \frac{1}{2}$	Gevrey $\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

- Speed-up for polynomials and $\sigma < 1$.

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

- Speed-up for polynomials and $\sigma < 1$.
- Open problem: Close the gap for $\sigma = 1$.

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

- Speed-up for polynomials and $\sigma < 1$.
- Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

- Speed-up for polynomials and $\sigma < 1$.
- Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:
   - Optimal algorithm with analog computation.

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

   - Speed-up for polynomials and $\sigma < 1$.
   - Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:
   - Optimal algorithm with analog computation.
3. Quantum policy optimization:

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

   - Speed-up for polynomials and $\sigma < 1$.
   - Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:
   - Optimal algorithm with analog computation.
3. Quantum policy optimization:
   - Speed-up using QVE.

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

   - Speed-up for polynomials and $\sigma < 1$.
   - Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:
   - Optimal algorithm with analog computation.

3. Quantum policy optimization:
   - Speed-up using QVE.
   - Open problem: approximating V for more smoothness?

Summary & outlook

1. Quantum gradient estimation:

Smoothness condition	Polynomial degree k	Gevrey $\sigma \in [0, \frac{1}{2})$	$\sigma = \frac{1}{2}$	$\sigma \in (\frac{1}{2}, 1]$
Best known algorithm	$\widetilde{\mathcal{O}}(k)$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\frac{1}{2}})$	$\widetilde{\mathcal{O}}(d^{\sigma})$
Best known lower bound	$\Omega(1)$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$	$\Omega(d^{\frac{1}{2}})$

From ℓ^∞ to ℓ^p approximations: multiply upper and lower bounds by $\Theta(d^{\frac{1}{p}})$.

   - Speed-up for polynomials and $\sigma < 1$.
   - Open problem: Close the gap for $\sigma = 1$.

2. Quantum value estimation:
   - Optimal algorithm with analog computation.
3. Quantum policy optimization:
   - Speed-up using QVE.
   - Open problem: approximating V for more smoothness?

Thanks for your attention!

`arjan@cwi.nl`