Quantum speedup of Monte Carlo methods

Ashley ~~Montanaro~~ Montecarlo

Department of Computer Science,
University of Bristol

18 June 2015

arXiv:1504.06987

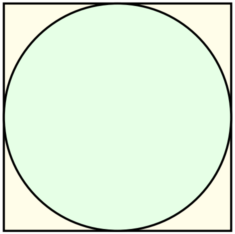
Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:

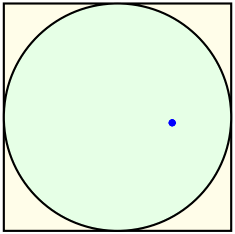


$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:



$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

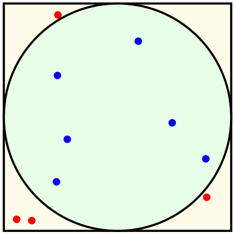
Darts landed in circle: 1/1.

Approximation to π: 4.0.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:



$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

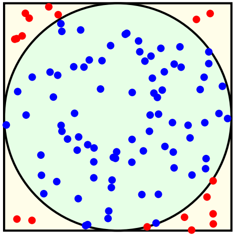
Darts landed in circle: 6/10.

Approximation to π: 2.4.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:



$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

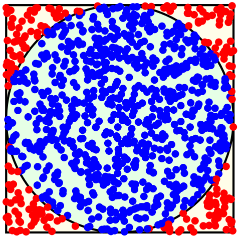
Darts landed in circle: 82/100.

Approximation to π: 3.28.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:



$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

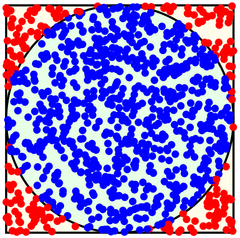
Darts landed in circle: 788/1000.

Approximation to π: 3.152.

Monte Carlo methods

Monte Carlo methods use randomness to estimate numerical properties of systems which are too large or complicated to analyse deterministically.

- For example, we can approximate π by throwing darts at a dartboard:



$\Pr[\text{dart lands in circle}] = \frac{\pi}{4}$.

Darts landed in circle: 788/1000.

Approximation to π: 3.152.

General problem

Given access to a randomised algorithm $\mathcal{A}$, estimate the expected output value μ of $\mathcal{A}$.

Classical algorithm

The following natural algorithm solves this problem for any $\mathcal{A}$:

1. Produce k samples $v_1, \ldots, v_k$, each corresponding to the output of an independent execution of $\mathcal{A}$.
2. Output the average $\widetilde{\mu} = \frac{1}{k} \sum_{i=1}^{k} v_i$ of the samples as an approximation of μ.

Classical algorithm

The following natural algorithm solves this problem for any $\mathcal{A}$:

1. Produce k samples $v_1, \dots, v_k$, each corresponding to the output of an independent execution of $\mathcal{A}$.
2. Output the average $\widetilde{\mu} = \frac{1}{k} \sum_{i=1}^{k} v_i$ of the samples as an approximation of μ.

Assuming that the variance of the output of $\mathcal{A}$ is at most σ^2,

$$\Pr[|\widetilde{\mu} - \mu| \geqslant \epsilon] \leqslant \frac{\sigma^2}{k\epsilon^2}.$$

Classical algorithm

The following natural algorithm solves this problem for any $\mathcal{A}$:

1. Produce k samples $v_1, \dots, v_k$, each corresponding to the output of an independent execution of $\mathcal{A}$.
2. Output the average $\widetilde{\mu} = \frac{1}{k} \sum_{i=1}^{k} v_i$ of the samples as an approximation of μ.

Assuming that the variance of the output of $\mathcal{A}$ is at most σ^2,

$$\Pr[|\widetilde{\mu} - \mu| \geqslant \epsilon] \leqslant \frac{\sigma^2}{k\epsilon^2}.$$

So we can take $k = O(\sigma^2/\epsilon^2)$ to estimate μ up to additive error ϵ with, say, 99% success probability.

Classical algorithm

The following natural algorithm solves this problem for any $\mathcal{A}$:

1. Produce k samples $v_1, \dots, v_k$, each corresponding to the output of an independent execution of $\mathcal{A}$.
2. Output the average $\widetilde{\mu} = \frac{1}{k} \sum_{i=1}^{k} v_i$ of the samples as an approximation of μ.

Assuming that the variance of the output of $\mathcal{A}$ is at most σ^2,

$$\Pr[|\widetilde{\mu} - \mu| \geqslant \epsilon] \leqslant \frac{\sigma^2}{k\epsilon^2}.$$

So we can take $k = O(\sigma^2/\epsilon^2)$ to estimate μ up to additive error ϵ with, say, 99% success probability.

This scaling is optimal for classical algorithms [Dagum et al. '00].

Classical algorithm

The following natural algorithm solves this problem for any $\mathcal{A}$:

1. Produce k samples $v_1, \dots, v_k$, each corresponding to the output of an independent execution of $\mathcal{A}$.
2. Output the average $\widetilde{\mu} = \frac{1}{k} \sum_{i=1}^{k} v_i$ of the samples as an approximation of μ.

Assuming that the variance of the output of $\mathcal{A}$ is at most σ^2,

$$\Pr[|\widetilde{\mu} - \mu| \geqslant \epsilon] \leqslant \frac{\sigma^2}{k\epsilon^2}.$$

So we can take $k = O(\sigma^2/\epsilon^2)$ to estimate μ up to additive error ϵ with, say, 99% success probability.

This scaling is optimal for classical algorithms [Dagum et al. '00].

- To estimate π up to 4 decimal places with success probability 0.5, we would need $> 10^9$ darts!

Quantum speedup

With a quantum computer, we can do better:

Theorem [AM '15]

There is a quantum algorithm which estimates μ up to additive error ϵ with 99% success probability and

$$\widetilde{O}(\sigma/\epsilon)$$

uses of $\mathcal{A}$.

Quantum speedup

With a quantum computer, we can do better:

Theorem **[AM '15]**

There is a quantum algorithm which estimates μ up to additive error ϵ with 99% success probability and

$$\widetilde{O}(\sigma/\epsilon)$$

uses of $\mathcal{A}$.

- The $\widetilde{O}$ notation hides polylog factors: more precisely, the complexity is $O((\sigma/\epsilon)\log^{3/2}(\sigma/\epsilon)\log\log(\sigma/\epsilon))$.

Quantum speedup

With a quantum computer, we can do better:

Theorem **[AM '15]**

There is a quantum algorithm which estimates μ up to additive error ϵ with 99% success probability and

$$\widetilde{O}(\sigma/\epsilon)$$

uses of $\mathcal{A}$.

- The $\widetilde{O}$ notation hides polylog factors: more precisely, the complexity is $O((\sigma/\epsilon)\log^{3/2}(\sigma/\epsilon)\log\log(\sigma/\epsilon))$.
- This complexity is optimal up to these polylog factors [Nayak and Wu '98].

Quantum speedup

With a quantum computer, we can do better:

Theorem [AM '15]

There is a quantum algorithm which estimates μ up to additive error ϵ with 99% success probability and

$$\widetilde{O}(\sigma/\epsilon)$$

uses of $\mathcal{A}$.

- The $\widetilde{O}$ notation hides polylog factors: more precisely, the complexity is $O((\sigma/\epsilon)\log^{3/2}(\sigma/\epsilon)\log\log(\sigma/\epsilon))$.
- This complexity is optimal up to these polylog factors [Nayak and Wu '98].

The underlying algorithm $\mathcal{A}$ can now be quantum itself.

Related work

This problem connects to several previous works, e.g.:

- Approximating the mean of an arbitrary bounded function (with range $[0, 1]$), with respect to the uniform distribution. Quantum complexity: $O(1/\epsilon)$ [Heinrich '01], [Brassard et al. '11].

Related work

This problem connects to several previous works, e.g.:

- Approximating the mean of an arbitrary bounded function (with range $[0,1]$), with respect to the uniform distribution. Quantum complexity: $O(1/\epsilon)$ [Heinrich '01], [Brassard et al. '11].
- Estimating the expected value $\mathrm{tr}(A\rho)$ of certain observables A which are bounded [Wocjan et al. '09], or whose tails decay quickly [Knill et al. '07].

Related work

This problem connects to several previous works, e.g.:

- Approximating the mean of an arbitrary bounded function (with range $[0, 1]$), with respect to the uniform distribution. Quantum complexity: $O(1/\epsilon)$ [Heinrich '01], [Brassard et al. '11].

- Estimating the expected value $\mathrm{tr}(A\rho)$ of certain observables A which are bounded [Wocjan et al. '09], or whose tails decay quickly [Knill et al. '07].

- Approximating the mean, with respect to the uniform distribution, of functions with bounded L^2 norm [Heinrich '01]

Related work

This problem connects to several previous works, e.g.:

- Approximating the mean of an arbitrary bounded function (with range $[0, 1]$), with respect to the uniform distribution. Quantum complexity: $O(1/\epsilon)$ [Heinrich '01], [Brassard et al. '11].

- Estimating the expected value $\mathrm{tr}(A\rho)$ of certain observables A which are bounded [Wocjan et al. '09], or whose tails decay quickly [Knill et al. '07].

- Approximating the mean, with respect to the uniform distribution, of functions with bounded L^2 norm [Heinrich '01]

Here we generalise these by approximating the mean output value of arbitrary quantum algorithms, given only a bound on the variance.

Ideas behind the algorithm

The algorithm combines and extends ideas of [Heinrich '01], [Brassard et al. '11], [Wocjan et al. '09].

Ideas behind the algorithm

The algorithm combines and extends ideas of [Heinrich '01], [Brassard et al. '11], [Wocjan et al. '09].

Let $v(\mathcal{A})$ be the random variable corresponding to the output of $\mathcal{A}$.

Ideas behind the algorithm

The algorithm combines and extends ideas of [Heinrich '01], [Brassard et al. '11], [Wocjan et al. '09].

Let $v(\mathcal{A})$ be the random variable corresponding to the output of $\mathcal{A}$.

First, in the special case where $v(\mathcal{A}) \in [0, 1]$:

- We can write down a quantum algorithm which outputs 1 bit, and whose expected output value is μ.
- We then use amplitude estimation to approximate μ up to additive error ϵ.
- The algorithm uses $\mathcal{A}$ $O(1/\epsilon)$ times.

Ideas behind the algorithm

Now consider the more general case where $v(\mathcal{A}) \geqslant 0$, $\mathbb{E}[v(\mathcal{A})^2] = O(1)$.

Ideas behind the algorithm

Now consider the more general case where $v(\mathcal{A}) \geqslant 0$, $\mathbb{E}[v(\mathcal{A})^2] = O(1)$.

In this case (based on ideas of [Heinrich '01]):

- Divide up the output values of $\mathcal{A}$ into blocks, such that in the t'th block $2^{t-1} \leqslant v(\mathcal{A}) \leqslant 2^t$.
- Use $\widetilde{O}(1/\epsilon)$ iterations of the previous algorithm to estimate the average values of each of the first $O(\log 1/\epsilon)$ blocks, each divided by 2^t.
- Sum up the results (after rescaling again).

Ideas behind the algorithm

Now consider the more general case where $v(\mathcal{A}) \geqslant 0$, $\mathbb{E}[v(\mathcal{A})^2] = O(1)$.

In this case (based on ideas of [Heinrich '01]):

- Divide up the output values of $\mathcal{A}$ into blocks, such that in the t'th block $2^{t-1} \leqslant v(\mathcal{A}) \leqslant 2^t$.
- Use $\widetilde{O}(1/\epsilon)$ iterations of the previous algorithm to estimate the average values of each of the first $O(\log 1/\epsilon)$ blocks, each divided by 2^t.
- Sum up the results (after rescaling again).

The constraint that $\mathbb{E}[v(\mathcal{A})^2] = O(1)$ implies that the overall error is at most ϵ.

Ideas behind the algorithm

The final step is to change the dependence on $\mathbb{E}[v(\mathcal{A})^2]$ to a dependence on

$$\mathsf{Var}(v(\mathcal{A})) = \mathbb{E}[(v(\mathcal{A}) - \mu)^2] = \sigma^2.$$

Ideas behind the algorithm

The final step is to change the dependence on $\mathbb{E}[v(\mathcal{A})^2]$ to a dependence on

$$\mathsf{Var}(v(\mathcal{A})) = \mathbb{E}[(v(\mathcal{A}) - \mu)^2] = \sigma^2.$$

- Run $\mathcal{A}$ once and use the output $\widetilde{m}$ as a guess for μ. $|\widetilde{m} - \mu| = O(\sigma)$ with high probability.

Ideas behind the algorithm

The final step is to change the dependence on $\mathbb{E}[v(\mathcal{A})^2]$ to a dependence on

$$\mathsf{Var}(v(\mathcal{A})) = \mathbb{E}[(v(\mathcal{A}) - \mu)^2] = \sigma^2.$$

- Run $\mathcal{A}$ once and use the output $\widetilde{m}$ as a guess for μ.
  $|\widetilde{m} - \mu| = O(\sigma)$ with high probability.
- Apply the previous algorithm to the subroutine produced by subtracting $\widetilde{m}$ and dividing by σ, with accuracy $O(\epsilon/\sigma)$.
- Estimate the positive and negative parts separately.

Ideas behind the algorithm

The final step is to change the dependence on $\mathbb{E}[v(\mathcal{A})^2]$ to a dependence on

$$\mathsf{Var}(v(\mathcal{A})) = \mathbb{E}[(v(\mathcal{A}) - \mu)^2] = \sigma^2.$$

- Run $\mathcal{A}$ once and use the output $\widetilde{m}$ as a guess for μ. $|\widetilde{m} - \mu| = O(\sigma)$ with high probability.
- Apply the previous algorithm to the subroutine produced by subtracting $\widetilde{m}$ and dividing by σ, with accuracy $O(\epsilon/\sigma)$.
- Estimate the positive and negative parts separately.

A similar idea works to estimate μ up to relative error ϵ: if $\sigma^2/\mu^2 \leqslant B$, we can estimate μ up to additive error $\epsilon\, \mathbb{E}[v(\mathcal{A})]$ with $\widetilde{O}(B/\epsilon)$ uses of $\mathcal{A}$.

Application: partition functions

Consider a (classical) physical system which has state space Ω, and a Hamiltonian $H : \Omega \to \mathbb{R}$ specifying the energy of each configuration $x \in \Omega$. Assume that H takes integer values in the set $\{0, \ldots, n\}$.

Application: partition functions

Consider a (classical) physical system which has state space Ω, and a Hamiltonian $H : \Omega \to \mathbb{R}$ specifying the energy of each configuration $x \in \Omega$. Assume that H takes integer values in the set $\{0, \dots, n\}$.

We want to compute the partition function

$$Z(\beta) = \sum_{x \in \Omega} e^{-\beta H(x)}$$

for some inverse temperature β.

Application: partition functions

Consider a (classical) physical system which has state space Ω, and a Hamiltonian $H : \Omega \to \mathbb{R}$ specifying the energy of each configuration $x \in \Omega$. Assume that H takes integer values in the set $\{0, \dots, n\}$.

We want to compute the partition function

$$Z(\beta) = \sum_{x \in \Omega} e^{-\beta H(x)}$$

for some inverse temperature β.

Encapsulates some interesting problems:

- Physics: The Ising and Potts models
- Computer science: counting k-colourings of graphs, counting matchings (monomer-dimer coverings), ...

Application: partition functions

Goal: estimate $Z(\beta)$ up to relative error ϵ, i.e. find $\widetilde{Z}$ such that

$$|\widetilde{Z} - Z(\beta)| \leqslant \epsilon\, Z(\beta).$$

Application: partition functions

Goal: estimate $Z(\beta)$ up to relative error ϵ, i.e. find $\widetilde{Z}$ such that

$$|\widetilde{Z} - Z(\beta)| \leqslant \epsilon \, Z(\beta).$$

Standard classical approach (e.g. [Stefankovič et al. '09]):

- Write $Z(\beta)$ as a product $\mathbb{E}[Y_0] \dots \mathbb{E}[Y_{\ell-1}]$ for random variables Y_i such that

  $$Y_i(x) = e^{-(\beta_{i+1} - \beta_i) H(x)},$$

  where $0 = \beta_0 < \beta_1 < \cdots < \beta_\ell = \beta$, and x is picked from the Gibbs distribution

  $$\pi_i(x) = \frac{1}{Z(\beta_i)} e^{-\beta_i H(x)}.$$

Application: partition functions

Goal: estimate $Z(\beta)$ up to relative error ϵ, i.e. find $\widetilde{Z}$ such that

$$|\widetilde{Z} - Z(\beta)| \leqslant \epsilon\, Z(\beta).$$

Standard classical approach (e.g. [Stefankovič et al. '09]):

- Write $Z(\beta)$ as a product $\mathbb{E}[Y_0] \dots \mathbb{E}[Y_{\ell-1}]$ for random variables Y_i such that

  $$Y_i(x) = e^{-(\beta_{i+1} - \beta_i)H(x)},$$

  where $0 = \beta_0 < \beta_1 < \cdots < \beta_\ell = \beta$, and x is picked from the Gibbs distribution

  $$\pi_i(x) = \frac{1}{Z(\beta_i)} e^{-\beta_i H(x)}.$$

- Then sample from the π_i distributions to estimate $\mathbb{E}[Y_i]$.

Markov chains and rapid mixing

This procedure will be efficient if $\mathbb{E}[Y_i^2]/\mathbb{E}[Y_i]^2 = O(1)$.

Markov chains and rapid mixing

This procedure will be efficient if $\mathbb{E}[Y_i^2]/\mathbb{E}[Y_i]^2 = O(1)$.

- For any partition function problem such that $|\Omega| = A$, there is a Chebyshev cooling schedule (sequence of β_i's) that achieves this with $\ell = \widetilde{O}(\sqrt{\log A})$ [Stefankovič et al. '09].

Markov chains and rapid mixing

This procedure will be efficient if $\mathbb{E}[Y_i^2]/\mathbb{E}[Y_i]^2 = O(1)$.

- For any partition function problem such that $|\Omega| = A$, there is a Chebyshev cooling schedule (sequence of β_i's) that achieves this with $\ell = \widetilde{O}(\sqrt{\log A})$ [Stefankovič et al. '09].
- Implies a classical algorithm using $\widetilde{O}((\log A)/\epsilon^2)$ samples.

Markov chains and rapid mixing

This procedure will be efficient if $\mathbb{E}[Y_i^2]/\mathbb{E}[Y_i]^2 = O(1)$.

- For any partition function problem such that $|\Omega| = A$, there is a Chebyshev cooling schedule (sequence of β_i's) that achieves this with $\ell = \widetilde{O}(\sqrt{\log A})$ [Stefankovič et al. '09].
- Implies a classical algorithm using $\widetilde{O}((\log A)/\epsilon^2)$ samples.

But how do we sample from the π_i distributions?

- Classically, we can use rapidly mixing Markov chains.

Markov chains and rapid mixing

This procedure will be efficient if $\mathbb{E}[Y_i^2]/\mathbb{E}[Y_i]^2 = O(1)$.

- For any partition function problem such that $|\Omega| = A$, there is a Chebyshev cooling schedule (sequence of β_i's) that achieves this with $\ell = \widetilde{O}(\sqrt{\log A})$ [Stefankovič et al. '09].
- Implies a classical algorithm using $\widetilde{O}((\log A)/\epsilon^2)$ samples.

But how do we sample from the π_i distributions?

- Classically, we can use rapidly mixing Markov chains.
- If the Markov chains have relaxation time τ, we get an overall classical algorithm using $\widetilde{O}((\log A)\tau/\epsilon^2)$ steps of the Markov chains [Stefankovič et al. '09].

Rapid mixing via quantum walks

- It turns out that the Chebyshev cooling schedule condition implies that quantum walks can be used to mix rapidly (mixing time improves from $O(\tau)$ to $O(\sqrt{\tau})$), based on techniques of [Wocjan and Abeyesinghe '08].

Rapid mixing via quantum walks

- It turns out that the Chebyshev cooling schedule condition implies that quantum walks can be used to mix rapidly (mixing time improves from $O(\tau)$ to $O(\sqrt{\tau})$), based on techniques of [Wocjan and Abeyesinghe '08].

- We can then apply quantum mean estimation to also improve the dependence on ϵ.

Rapid mixing via quantum walks

- It turns out that the Chebyshev cooling schedule condition implies that quantum walks can be used to mix rapidly (mixing time improves from $O(\tau)$ to $O(\sqrt{\tau})$), based on techniques of [Wocjan and Abeyesinghe '08].

- We can then apply quantum mean estimation to also improve the dependence on ϵ.

- The final quantum complexity is $\widetilde{O}((\log A)(\sqrt{\tau}/\epsilon + \tau))$.

Rapid mixing via quantum walks

- It turns out that the Chebyshev cooling schedule condition implies that quantum walks can be used to mix rapidly (mixing time improves from $O(\tau)$ to $O(\sqrt{\tau})$), based on techniques of [Wocjan and Abeyesinghe '08].
- We can then apply quantum mean estimation to also improve the dependence on ϵ.
- The final quantum complexity is $\widetilde{O}((\log A)(\sqrt{\tau}/\epsilon + \tau))$.

Note 1: A similar idea was proposed by [Wocjan et al. '09]. However, that work needed $Z(\beta_{i+1})/Z(\beta_i) = \Omega(1)$, which would require $\ell = \Omega(\log A)$.

Rapid mixing via quantum walks

- It turns out that the Chebyshev cooling schedule condition implies that quantum walks can be used to mix rapidly (mixing time improves from $O(\tau)$ to $O(\sqrt{\tau})$), based on techniques of [Wocjan and Abeyesinghe '08].
- We can then apply quantum mean estimation to also improve the dependence on ϵ.
- The final quantum complexity is $\widetilde{O}((\log A)(\sqrt{\tau}/\epsilon + \tau))$.

Note 1: A similar idea was proposed by [Wocjan et al. '09]. However, that work needed $Z(\beta_{i+1})/Z(\beta_i) = \Omega(1)$, which would require $\ell = \Omega(\log A)$.

Note 2: The $\widetilde{O}((\log A)\tau)$ part of the bound is the complexity of computing the Chebyshev cooling schedule itself.

Example: The ferromagnetic Ising model

We are given as input a graph $G = (V, E)$ with n vertices.

- We consider the Ising Hamiltonian ($z \in \{\pm 1\}^n$)

$$H(z) = -\sum_{(u,v) \in E} z_u z_v .$$

- We want to approximate

$$Z(\beta) = \sum_{z \in \{\pm 1\}^n} e^{-\beta\, H(z)} .$$

Example: The ferromagnetic Ising model

We are given as input a graph $G = (V, E)$ with n vertices.

- We consider the Ising Hamiltonian ($z \in \{\pm 1\}^n$)

$$H(z) = -\sum_{(u,v)\in E} z_u z_v.$$

- We want to approximate

$$Z(\beta) = \sum_{z\in\{\pm 1\}^n} e^{-\beta\, H(z)}.$$

- Best classical runtime known: $\widetilde{O}(n^2/\epsilon^2)$ [Stefankovič '09] (if β is small enough)
- Quantum runtime: $\widetilde{O}(n^{3/2}/\epsilon + n^2)$.

Example: The ferromagnetic Ising model

We are given as input a graph $G = (V, E)$ with n vertices.

- We consider the Ising Hamiltonian ($z \in \{\pm 1\}^n$)

$$H(z) = -\sum_{(u,v)\in E} z_u z_v.$$

- We want to approximate

$$Z(\beta) = \sum_{z\in\{\pm 1\}^n} e^{-\beta H(z)}.$$

- Best classical runtime known: $\widetilde{O}(n^2/\epsilon^2)$ [Stefankovič '09] (if β is small enough)
- Quantum runtime: $\widetilde{O}(n^{3/2}/\epsilon + n^2)$.

Other applications from computer science: counting matchings (monomer-dimer coverings) and k-colourings.

Summary

- There is a quantum algorithm which estimates μ up to additive error ϵ with $\widetilde{O}(\sigma/\epsilon)$ uses of $\mathcal{A}$.

Summary

- There is a quantum algorithm which estimates μ up to additive error ϵ with $\widetilde{O}(\sigma/\epsilon)$ uses of $\mathcal{A}$.
- We can use this to approximate partition functions more quickly than the best classical algorithms known.

Summary

- There is a quantum algorithm which estimates μ up to additive error ϵ with $\widetilde{O}(\sigma/\epsilon)$ uses of $\mathcal{A}$.
- We can use this to approximate partition functions more quickly than the best classical algorithms known.
- Open problem: Is there a more efficient quantum algorithm for computing a Chebyshev cooling schedule?

Summary

- There is a quantum algorithm which estimates μ up to additive error ϵ with $\widetilde{O}(\sigma/\epsilon)$ uses of $\mathcal{A}$.
- We can use this to approximate partition functions more quickly than the best classical algorithms known.
- Open problem: Is there a more efficient quantum algorithm for computing a Chebyshev cooling schedule?

Thanks!

Bonus application: the distance between probability distributions

- Imagine we can sample from probability distributions p and q on n elements.
- We would like to estimate the total variation distance

$$\|p - q\| = \frac{1}{2} \sum_{x} |p(x) - q(x)|$$

  up to additive error ϵ.

Bonus application: the distance between probability distributions

- Imagine we can sample from probability distributions p and q on n elements.
- We would like to estimate the total variation distance

$$\|p - q\| = \frac{1}{2} \sum_x |p(x) - q(x)|$$

  up to additive error ϵ.
- Classically, this needs about $\Omega(n)$ samples [Valiant '11].
- Quantumly, we can do it using $O(\sqrt{n}/\epsilon^8)$ samples [Bravyi, Harrow and Hassidim '11].

Bonus application: the distance between probability distributions

- Imagine we can sample from probability distributions p and q on n elements.
- We would like to estimate the total variation distance

$$\|p-q\| = \frac{1}{2}\sum_x |p(x)-q(x)|$$

  up to additive error ϵ.
- Classically, this needs about $\Omega(n)$ samples [Valiant '11].
- Quantumly, we can do it using $O(\sqrt{n}/\epsilon^8)$ samples [Bravyi, Harrow and Hassidim '11].
- Using quantum mean estimation we improve this to $\widetilde{O}(\sqrt{n}/\epsilon^{3/2})$.

Bonus application: the distance between probability distributions

- We can write $\|p-q\| = \mathbb{E}_x[R(x)]$, where

$$R(x) = \frac{|p(x)-q(x)|}{p(x)+q(x)},$$

  and x is drawn from the distribution $r = (p+q)/2$.

Bonus application: the distance between probability distributions

- We can write $\|p-q\| = \mathbb{E}_x[R(x)]$, where
$$R(x) = \frac{|p(x)-q(x)|}{p(x)+q(x)},$$
and x is drawn from the distribution $r = (p+q)/2$.

- For each x, $R(x)$ can be computed up to accuracy ϵ using $\widetilde{O}(\sqrt{n/\epsilon})$ iterations of amplitude estimation.

Bonus application: the distance between probability distributions

- We can write $\|p - q\| = \mathbb{E}_x[R(x)]$, where

$$R(x) = \frac{|p(x) - q(x)|}{p(x) + q(x)},$$

  and x is drawn from the distribution $r = (p + q)/2$.

- For each x, $R(x)$ can be computed up to accuracy ϵ using $\widetilde{O}(\sqrt{n/\epsilon})$ iterations of amplitude estimation.

- Wrapping this within $O(1/\epsilon)$ iterations of the mean-estimation algorithm, we obtain an overall algorithm running in time $\widetilde{O}(\sqrt{n}/\epsilon^{3/2})$.

Applications

Some partition function applications:

- The ferromagnetic Ising model at high enough temperature. Quantum runtime: $\widetilde{O}(n^{3/2}/\epsilon + n^2)$ steps (compare classical: $\widetilde{O}(n^2/\epsilon^2)$ steps).

- Counting valid k-colourings of a degree $d < k/2$ graph on n vertices. Quantum runtime: $\widetilde{O}(n^{3/2}/\epsilon + n^2)$ (classical: $\widetilde{O}(n^2/\epsilon^2)$)

- Counting matchings (monomer-dimer coverings) of a graph with n vertices and m edges. Quantum runtime: $\widetilde{O}(n^{3/2}m^{1/2}/\epsilon + n^2 m)$ (classical: $\widetilde{O}(n^2 m/\epsilon^2)$)