Simon のアルゴリズム解説

@0917datw

はじめに

本スライドは，下記の論文解説を行なうものである.

Simon, Daniel R. ”On the power of quantum computation.” SIAM journal on computing 26.5 (1997): 1474-1483.

目次

1. Simon のアルゴリズムの概要

1. Simon のアルゴリズムの概要
2. Simon のアルゴリズムの詳細
3. Simon のアルゴリズムの実装

Simon のアルゴリズムの概要

Simon のアルゴリズムは次の問題を多項式時間で求めるアルゴリズムである.

Input : $n, m \in \mathbb{N}$ s.t. $n \leq m$
Output : one-to-one or two-to-one

$f : \{0,1\}^n \to \{0,1\}^m$ について,

$$x, x' \in \{0,1\}^n,\ f(x) = f(x') \Leftrightarrow f : \text{one-to-one or two-to-one}$$

が成り立つとき, f が one-to-one **か** two-to-one **か判定せよ.**
ただし,

$$f : \text{one-to-one} \Leftrightarrow \forall b \in \{0,1\}^m, \exists a \in \{0,1\}^n \text{ s.t. } f(a) = b$$
$$f : \text{two-to-one} \Leftrightarrow \forall b \in \{0,1\}^m, \exists a, c \in \{0,1\}^n \text{ s.t. } f(a) = f(c) = b,\ a \neq c$$

Simon のアルゴリズムの補足

一般に Simon のアルゴリズムは次のような形で書かれる.

Input : $n, m \in \mathbb{N}$ s.t. $n \leq m$
Output : $s \in \{0,1\}^n$

$f : \{0,1\}^n \to \{0,1\}^m$ **について,**

$$x, x' \in \{0,1\}^n,\ \exists s \in \{0,1\}^n \text{ s.t. } f(x) = f(x') \Leftrightarrow x' \in \{x, x \oplus s\}$$

が成り立つような $s \in \{0,1\}^n$ **を特定せよ.**

$s = 0^n$ **のとき,** $f(x) = f(x') \Rightarrow x = x'$ **は単射の定義そのもの.**
$s \neq 0^n$ **のとき,** $x \neq x \oplus s$ **だが,** $f(x) = f(x \oplus s)$ **となっている.**

Simon のアルゴリズムの補足

感覚的には，$2^{n-1}+1$ 個の x について $f(x)$ の値を調べれば判定できる．つまり，そのときの計算量は $O(2^n)$ である．
しかし，このアルゴリズムでは，$O(2^{\frac{n}{2}})$ の計算量で判定できる．

2. Simon のアルゴリズムの詳細

1. Simon のアルゴリズムの概要
2. Simon のアルゴリズムの詳細
3. Simon のアルゴリズムの実装

Simon のアルゴリズムの詳細

初めに以下の 4 ステップを十分な回数（後に定量化）繰り返す.

i) Hadamard 変換を作用させて $x \in 0,1^n$ に対して, $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle$ なる状態を作る

ii) *i*) を使った合成 $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle \otimes |0\rangle$ なる状態を量子オラクルへ与えることで $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle \otimes |x \oplus s\rangle$ なる状態を得る

iii) *ii*) の前半 n ビットにもう一度 Hadamard 変換を作用させることで, $\frac{1}{2^n}\sum_y\sum_x (-1)^{xy}|x\rangle \otimes |y \oplus s\rangle$ なる状態を得る

iv) *iii*) を測定する

上記の測定値が生成するベクトル空間 V の次元を ℓ とするとき,

$\ell = n \Rightarrow s = 0^n$

$\ell < n \Rightarrow s =$ (V の任意の元と直交するベクトル)

Simon のアルゴリズムの詳細

以下を考察する.

- 測定値で得られたベクトルは s と直交する
- ループ回数の定量化

測定値で得られたベクトルは s と直交する

iii) に対して，$(|z\rangle)_{z\in\{0,1\}^n}$ に関する測定を行う．$s = 0^n$ のときは，z が等確率で出現し，このとき，s は $\{0,1\}^n$ の任意の元と直交する．

$s \neq 0^n$ のときは，$(|y\rangle)$ に関する測定を行うと，$|y\rangle = \frac{1}{2}(|x\rangle + |x \oplus s\rangle)$ と表せるから，*iii*) は $\frac{1}{2^{(n+1)/2}} \sum_y (-1)^{xy}|x\rangle + |x \oplus s\rangle$ となる．

$(-1)^{xy}|x\rangle + |x \oplus s\rangle = (-1)^{xy}|x\rangle + (-1)^{(x\oplus s)y}|x\rangle$
$= (-1)^{xy}(1 + (-1)^{sy})|x\rangle = (-1)^{xy}|x\rangle$ if $sy = 0$

ゆえ，$ys = 0$ i.e. y と s が直交する結果のみを得る．

ループ回数の定量化

$s \neq 0$ のとき，ℓ 回測定を行なって，生成するベクトル空間 V_ℓ の次元を d_ℓ と表す．すると，d_ℓ の取りうる値は高々 $n-1$ である．
$d_\ell = n-1$ になるときの ℓ を評価すれば良い．

簡単のため、$\ell+1$ 回目の測定値が V_k に属さない（$= V_{k+1}$ の次元が d_k+1 になる）確率を ℓ によらず $\frac{1}{2}$ とする．すると，このことは，成功する確率が $\frac{1}{2}$ の一様分布に従う事象において，$\ell-1$ 回目まで $n-2$ 回成功し，ℓ 回目も成功する場合を考えることと同値．

ループ回数の定量化

下記の不等式で，$N = \ell,\ p = \frac{1}{2},\ t = \frac{1}{2} - \frac{n-2}{\ell}$ とした場合，右辺は $2\exp\left(-\frac{(\ell - 2(n-2))^2}{2\ell}\right)$ ゆえ，$\ell \geq 3n$ とすると，右辺を $2\exp\left(-\frac{n}{6}\right)$ で抑えることができる.

以上より，測定を，つまり，ループを $3n$ 回以上行えば，失敗する確率は $2\exp(-\frac{n}{6})$ 以下になることが示された.

Thm

$1 \leq i \leq N$ で確率変数 X_i が i.i.d. のとき，$0 \leq p \leq 1,\ \Pr[X_i = 1] = p,\ \Pr[X_i = 0] = 1 - p,\ t > 0$ とするとき，次の不等式が成り立つ

$$\Pr\left[\left|\frac{X_1 + \cdots + X_N}{N} - p\right| \geq t\right] \leq 2\exp(-2Nt^2)$$

3. Simon のアルゴリズムの実装

1. Simon のアルゴリズムの概要
2. Simon のアルゴリズムの詳細
3. Simon のアルゴリズムの実装

Simon のアルゴリズムの実装

初めに以下をインポートする.

```
from qiskit import IBMQ, BasicAer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, execute
```

Simon のアルゴリズムの実装

全てまとめたスクリプト，今回は $s = 11000$ で実装している

```
def main():
    b = "11000"
    n = len(b)
    simon_circuit = QuantumCircuit(n*2, n)

    # オラクルに入力する前にアダマールゲートを適用する
    simon_circuit.h(range(n))
    # 可視性の向上のため、境界を挿入する
    simon_circuit.barrier()

    simon_circuit = quantum_simon_oracle(b, simon_circuit)
    # 可視性の向上のため、境界を挿入する
    simon_circuit.barrier()
    # 入力レジスターにアダマールゲートを適用する
    simon_circuit.h(range(n))

    # 量子ビットを測定する
    simon_circuit.measure(range(n), range(n))
    # ローカルシミュレーターを利用する
    backend = BasicAer.get_backend('qasm_simulator')
    shots = 1024
    results = execute(simon_circuit, backend=backend, shots=shots).result()
    counts = results.get_counts()

    for z in counts:
        print( '{}.{} = {} (mod 2)'.format(b, z, bdotz(b,z)) )
```

量子オラクルの実装

今回の量子オラクルをスクラッチ実装したもの

```
def quantum_simon_oracle(b: str, circuit: list) -> list:
    """ビット列
    n x に対して, |x>|0> を入力として |x> |x \oplus b> を出力する関数ただし,
     x \oplus b で x と b の排他的論理和を表す
    |x>|0> → |x>|x> と，初めに1つめのレジスタの内容をつめにコピーする2次に，後半
      n ビットと b との排他的論理和で与える
    """
    n = len(b)

    # |x>|0> → |x>|x>
    for index in range(n):
        circuit.cx(index, index+ n)

    # |x>|x> → |x>|x \oplus b>
    for index in range(n):
        if b[n - 1 - index] == '1':
            target_index = index
            break
    for index in range(n):
        if b[index] == '1':
            circuit.cx(target_index, 2 * n - 1 - index)

    return circuit
```

参考文献

[1] M. A. Nielsen and I.L. Chuang:
Quantum Computation and Quantum Information, Cambridge University Press, 2000

[2] **縫田 光司,『耐量子計算機暗号』, 森北出版,** 2020 **年**