

データ工学における量子コンピューティング

株式会社KDDI総合研究所 齊藤 和広, 別府 翔平, 黒川 茂莉, 伊神 皓生, 成定 真太郎
株式会社Jij 山城悠

1. 量子コンピューティング概要（KDDI研 齊藤：15分）
2. 量子機械学習アルゴリズム（KDDI研 別府，黒川：20分）
3. 量子組合せ最適化アルゴリズム（Jij 山城，KDDI研 伊神：25分）
 - イジングマシン（量子アニーリング等）
 - 量子ゲート
4. 量子探索アルゴリズム（KDDI研 成定：20分）

1. 量子コンピューティング概要

株式会社KDDI総合研究所 齊藤 和広

「ムーアの法則」の終焉

※18ヶ月で2倍の集積率

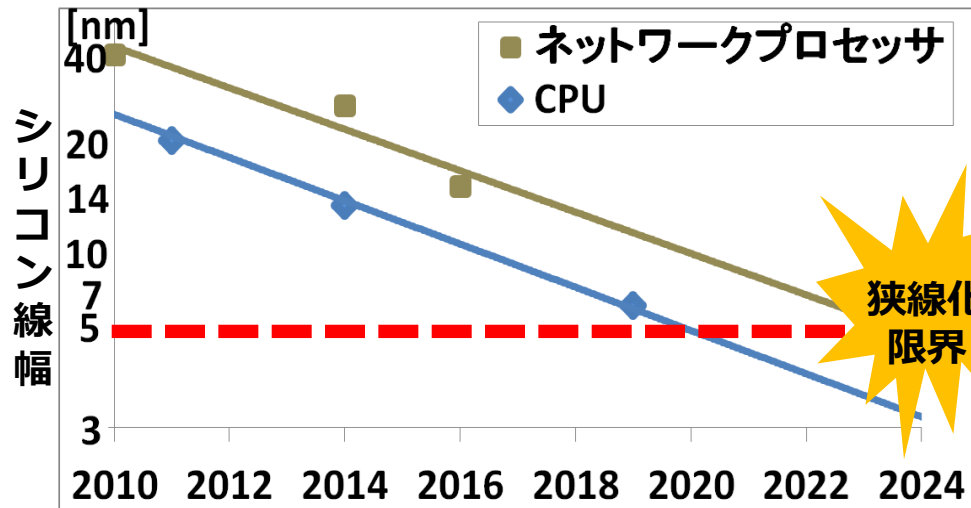


世界の取り組み

- ・用途特化のアクセラレータ
- ・従来アーキテクチャの刷新



『量子コンピュータ』が次世代コンピュータの有力候補として注目

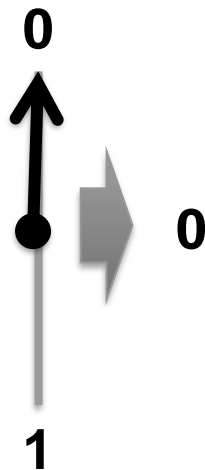


狭線化
限界

古典コンピュータ

2進数で一つの情報を保持／演算

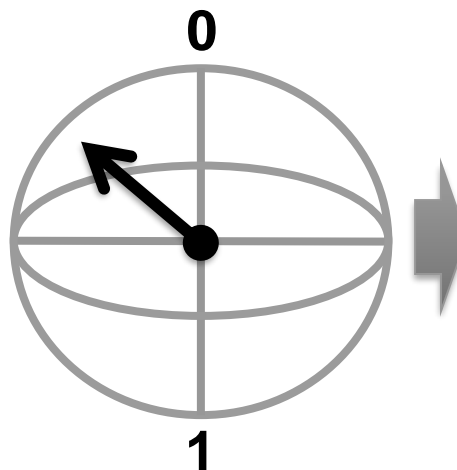
ビット bit



量子コンピュータ

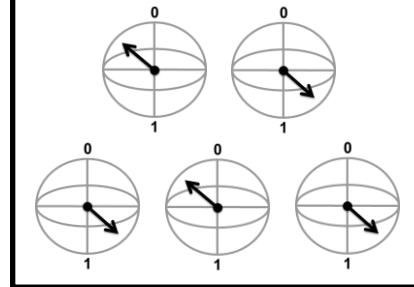
「量子重ね合わせ」により同時に複数の情報を保持／演算

量子ビット qubit



観測A
: 0
観測B
: 1

5 qubit



$2^5 = 32$ 通り
の状態を同時保持

32の並列計算が可能

超並列コンピュータ

古典コンピュータで計算に長時間を要する、または計算が困難な問題

特に量子コンピュータの活用が検討されている領域

	最適化&探索問題	シミュレーション&モデリング	AI&機械学習
概要	問題の規模に応じて 計算量が指数関数的に膨大 となる問題	物理学の基本方程式（力学／電磁気学等）のように、 微分積分などの計算量が膨大な数値計算 を含む問題	ディープラーニングや強化学習などの 計算量の多い機械学習手法
応用例	<ul style="list-style-type: none">・ 信号機制御による渋滞解消・ 資産ポートフォリオ最適化・ タンパク質設計最適化・ 暗号解読	<ul style="list-style-type: none">・ 電磁界シミュレーション・ 触媒反応シミュレーション・ 有機分子の結晶構造分析・ エンジンの熱伝達分析	<ul style="list-style-type: none">・ クラスタリングによる異常検知・ 時系列分析・ 次元圧縮・ 画像と音声の分類



データ工学で関連する用途：

- 機械学習
- 組合せ最適化
- データ探索

量子コンピュータ

量子の物理現象でビットを表現

量子ゲート

演算回路を持つ汎用型

研究段階だが

利用可能なマシンも出現

イジングマシン

組合せ最適化問題等に特化

産業界での実証が

進められているマシン

実装方式

開発組織

超電導方式

IBM, Google, Rigetti, USTC他, 等

イオントラップ方式

IonQ, Honeywell, AQT, 理研, 等

シリコン量子方式

Intel, TSMC, 等

光量子方式

Xanadu, USTC他, NTT, 等

その他の方式

Microsoft, Pasqal

ア ニ ー リ ン グ 方 式

量子アニーリング (超電導方式)

D-Wave, NEC他

デジタル回路 シミュレータ*

富士通, 日立製作所

ソフトウェア シミュレータ*

日立製作所, NEC, Jij, Fixstars, 等

分岐方式 (ソフトウェア*)

東芝

レーザネットワーク方式*

NTT他

計算方式	要件定義	設計	実装	計算
古典コンピュータ	計算内容や課題を定義	アルゴリズムの構築／ソフトウェア設計／データ構造設計	プログラム開発	プログラムに従って計算実行
イジングマシン	課題を明確化して組合せ最適化問題を定式化	イジングモデル（または QUBO）の構築	量子ビット間の結合強度と量子ビットの重みを計算	エネルギーを最小化する解を算出
量子ゲート	古典方式より高速化が期待できる計算内容や課題を定義	量子アルゴリズムの構築／量子回路の設計	量子回路を実行するプログラム開発	量子回路の量子ビット操作に従って計算実行

量子コンピュータはどのような計算でも古典コンピュータより高速なわけではない

→量子加速／超越性／優位性

【量子アルゴリズム】

量子ビットの特性を用いた計算方法。

<例>

アダマールテスト：期待値推定

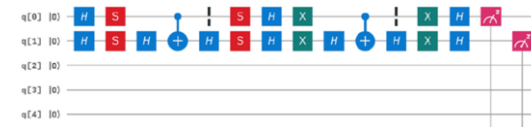
Shorアルゴリズム：素因数分解

Groverアルゴリズム：データ探索

HHL：連立 1 次方程式

【量子回路】

量子ゲートを並べて量子アルゴリズムを実現する回路。

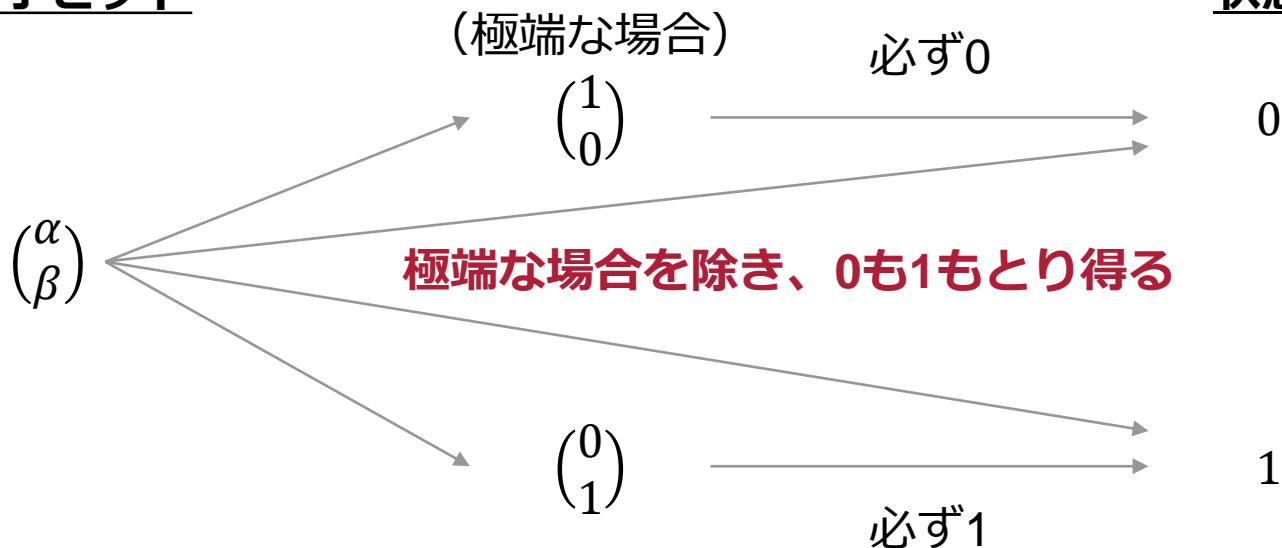


量子アルゴリズムの効果的な活用、用途に応じた量子アルゴリズムと量子回路の開発が必要

- 量子ビット：0と1の重ね合わせ状態（2行1列の複素ベクトルで表現）
- 測定してはじめて状態が定まる

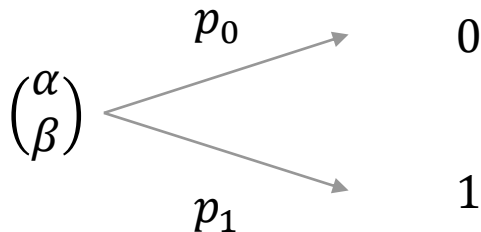
量子ビット

状態



量子ビット

状態



ブラケット記法

量子ビット ψ の状態ベクトル

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

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} : \text{古典の0}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} : \text{古典の1}$$

$$0\text{をとる確率} \quad p_0 = |\alpha|^2$$

$$1\text{をとる確率} \quad p_1 = |\beta|^2$$

$$\text{足して1(規格化条件)} \quad |\alpha|^2 + |\beta|^2 = 1$$

例：0と1の得られる確率50%

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

■ 量子ゲート：量子ビットに対する演算（量子演算）

【量子ゲートの条件】

- **線形性**：量子ビットの状態ベクトルに対する線形変換である
→ 一つの量子ビットに対する演算は2x2の行列
- **ユニタリ性**：エルミート共役をとると逆行列になる
→ 規格化条件 ($|\alpha|^2 + |\beta|^2 = 1$) を守った演算

例：Xゲート

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

$$X|0\rangle = |1\rangle, X|1\rangle = |0\rangle$$

古典のNOTゲートに対応

例：アダマールゲート（Hゲート）

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

重ね合わせ状態を作成

- 量子コンピュータはn個のビットで 2^n 個の状態を同時に計算

$$|\psi\rangle = \sum_{i_1, \dots, i_n \in \{0,1\}} c_{i_1, \dots, i_n} \underbrace{|i_1 \dots i_n\rangle}_{\text{n個の量子ビット}} = \begin{pmatrix} c_{00\dots0} \\ c_{00\dots1} \\ \vdots \\ c_{11\dots1} \end{pmatrix} \xrightarrow{\text{測定}} \text{一つの解: } |i_1 \dots i_n\rangle$$

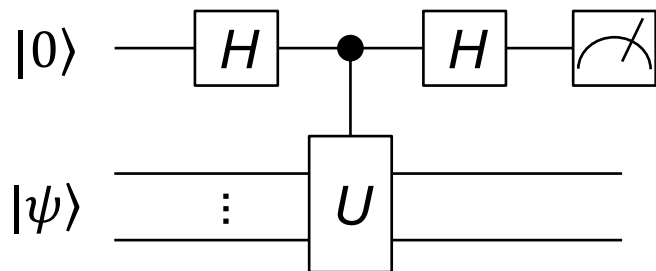
n個の量子ビット 2^n 個の状態

より高い確率でほしい解を得られる設計が必要



量子アルゴリズム

■ 量子ビット列 $|\psi\rangle$ に演算 U を作用したときの期待値を推定する量子アルゴリズム

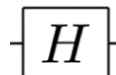



➡
複数回
測定

p_0 : 0が得られる確率
 p_1 : 1が得られる確率
期待値 = $p_0 - p_1$

※この回路は実数部の期待値のみ
虚数部の期待値も似た回路で計算可能

アダマールテストの量子回路

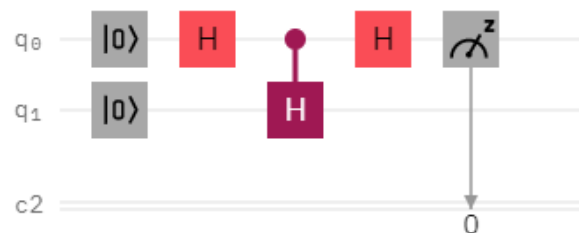
 : アダマールゲート (Hゲート)
初期状態 $|0\rangle$ から重ね合わせ状態が得られる演算

 : 制御ユニタリゲート (例：CNOTゲート)
制御量子ビット (上の●側) が $|1\rangle$ のときに演算 U をターゲット (下) に適用

■ 対象の演算：アダマールゲート

■ 本当に同じ値が得られるか確認

● まずはアダマールゲートの期待値



```
In [5]: Dagger(Qubit('0'))*H(0)*Qubit('0')
```

```
Out [5]: <0|H0|0>
```

```
In [6]: qapply(Dagger(Qubit('0'))*H(0)*Qubit('0')).evalf()
```

```
Out [6]: 0.707106781186548
```

● アダマールテストで得られる期待値 →

```
In [1]: from sympy import *
from sympy.physics.quantum import *
from sympy.physics.quantum.qubit import *
from sympy.physics.quantum.gate import *
init_printing() # ベクトルや行列を綺麗に表示するため
```

```
In [2]: # アダマールテストゲート
H(0) * CGateS(0,H(1)) * H(0) * Qubit('00')
```

```
Out [2]: H0C0(H1)H0|00>
```

```
In [3]: result = qapply(H(0) * CGateS(0,H(1)) * H(0) * Qubit('00')) # ゲート実行
simplify(result)
```

```
Out [3]: 
$$\frac{\sqrt{2}|00\rangle + 2|00\rangle - \sqrt{2}|01\rangle + 2|01\rangle + \sqrt{2}|10\rangle - \sqrt{2}|11\rangle}{4}$$

```

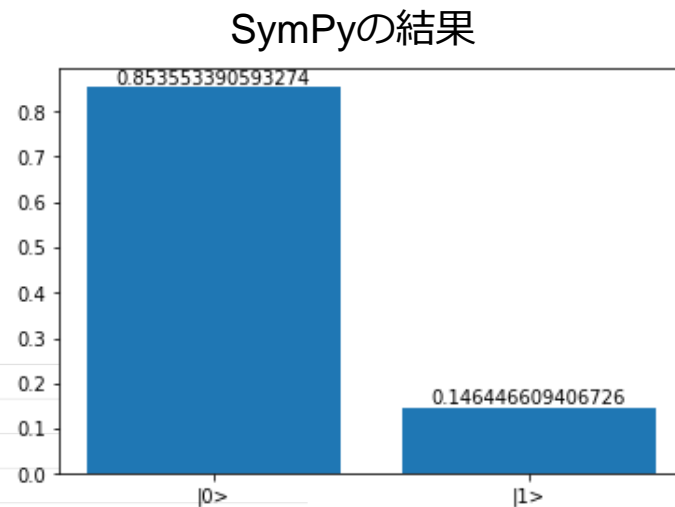
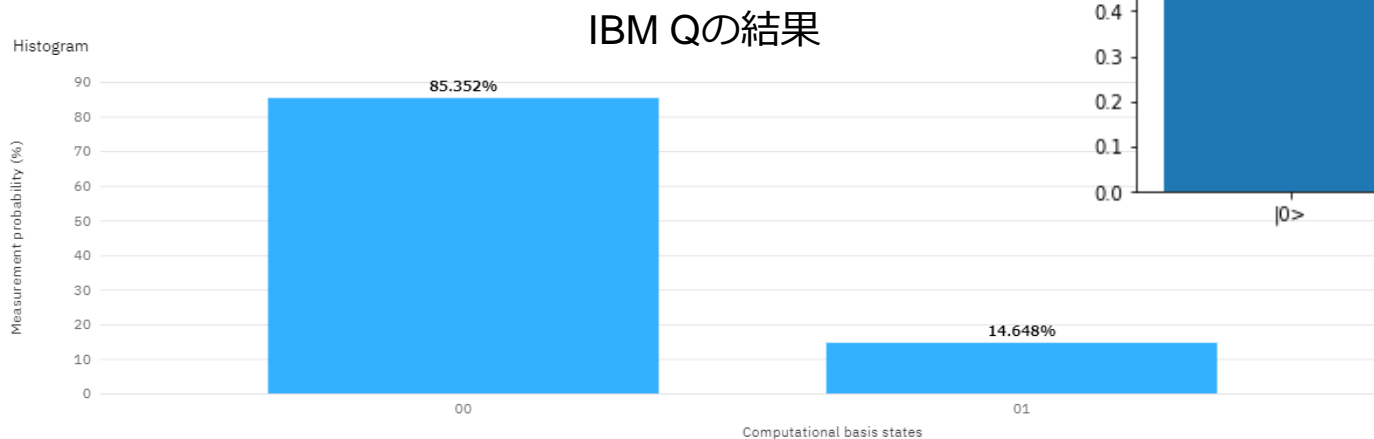
```
In [4]: q0_measured = measure_partial(result,(0,)) # 計測
p0 = q0_measured[0][1].evalf() # q0の|0>の確率
p1 = q0_measured[1][1].evalf() # q0の|1>の確率
p0-p1 # 期待値
```

```
Out [4]: 0.707106781186547
```

■ SymPyの期待値 : 0.707106781186548

■ IBM Qの期待値 : 0.70884

● 誤差0.2%程度



■ 重要な課題：量子ビットのエラー訂正

- エラー（ノイズ）とは、外的要因などによって情報が壊れること
- アプローチ：エラー訂正を備えた量子コンピュータの開発 or エラーを許容して利用

	FTQC (Fault-Tolerant Quantum Computer)	NISQデバイス (Noisy Intermediate-Scale Quantum)
概要	エラー訂正を備えた量子ゲート方式の量子コンピュータ	エラー訂正を持たない小・中規模（～数100量子ビット）な量子ゲート方式の量子コンピュータ
特徴	<ul style="list-style-type: none">● FTQC向けの量子アルゴリズムをLong-termアルゴリズムと呼ぶ● エラー訂正に多数の余剰量子ビットが必要 →実用には100万量子ビット以上が必要● エラー訂正自体が新たなエラーを生むため、量子ゲートのエラー率の低下も必要	<ul style="list-style-type: none">● NISQデバイス向けの量子アルゴリズムをNISQアルゴリズムと呼ぶ● エラーを許容しつつ、エラー削減のため短い量子回路で実行● 古典コンピュータも併用する量子古典ハイブリッドなNISQアルゴリズムが主流
実装例	<ul style="list-style-type: none">● 実装なし● 各国で2030-2035年頃の実用化に向けて開発中	<ul style="list-style-type: none">● IBM Quantum (127量子ビット, 2021)● Google Sycamore (54量子ビット, 2019)



エラー訂正手法の開発、デバイスの進化



NISQアルゴリズムの開発と実証

■ 量子コンピューティングの概要を紹介

- 現在の量子コンピュータの種類：イジングマシン, 量子ゲート
- 活用が期待できる分野：最適化&探索問題, シミュレーション&モデリング, 機械学習
- 技術概要を紹介：量子ビット, 量子ゲート, 量子アルゴリズム
- 量子アルゴリズムの種類：Long-termアルゴリズム, NISQアルゴリズム

■ 参考文献

- Quantum Native Dojo : <https://dojo.qulacs.org/ja/latest/index.html>
- 教科書（和書）：嶋田 義皓, 量子コンピューティング: 基本アルゴリズムから量子機械学習まで”
- 教科書（洋書）：M. A. Nielsen & I. L. Chuang, “Quantum Computation and Quantum Information”.
- 量子コンピュータの実装方式：T. D. Ladd et al., “Quantum computing,” Nature, 2010.
- NISQ：J. Preskill, “Quantum computing in the NISQ era and beyond,” Quantum, 2018.

2. 量子機械学習アルゴリズム

株式会社KDDI総合研究所 別府 翔平, 黒川 茂莉

■ 量子機械学習 には様々なバリエーションがあるが、代表的な2種類を紹介する

① 古典機械学習の特定の計算部分を量子コンピュータに委託し、高速に計算する

- 機械学習の精度は古典のみで行った場合と同一
- アダマールテストやGrover等の「量子加速に関して理論保証のあるアルゴリズム」を使う
→ long-term と考えられている

② 古典機械学習における表現（ニューラルネット、カーネル等）として量子系を使う

- 機械学習の精度が古典を上回る可能性がある
- たとえばノンパラメータ(SVM)、パラメータ(NN)の量子版がある
- 雑音があってもある程度動作するとされ、大規模or/and複雑な量子系の表現力に期待する → near-term と考えられている

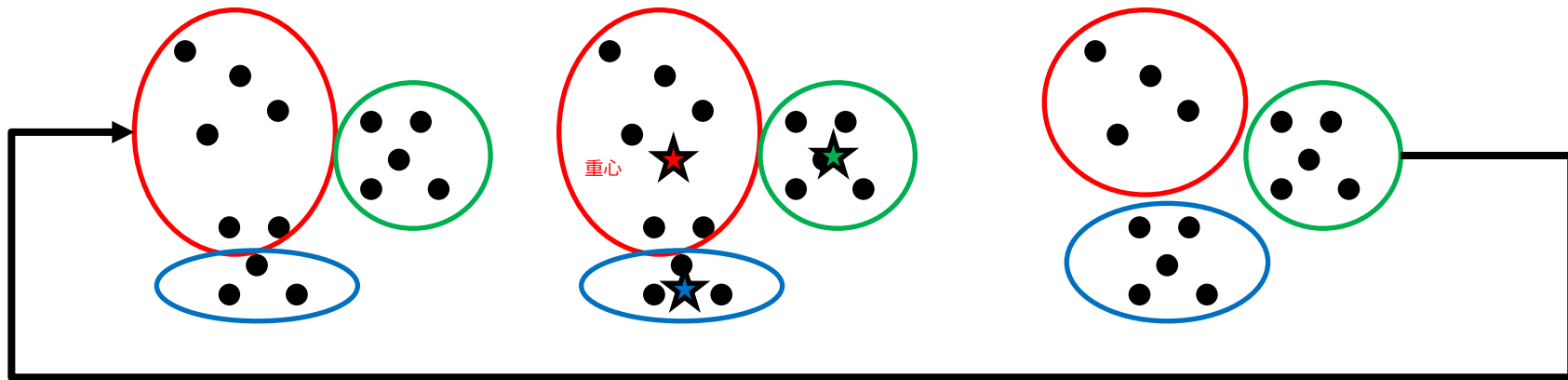
① 古典機械学習の特定の計算を量子加速するタイプ°

K-meansクラスタリングの量子加速(指数加速)

① クラスタ割り当て

② 各クラスターの重心計算

③ 重心に基づきクラスター再割り当て
(最近接の重心を持つクラスターへ割り当て)



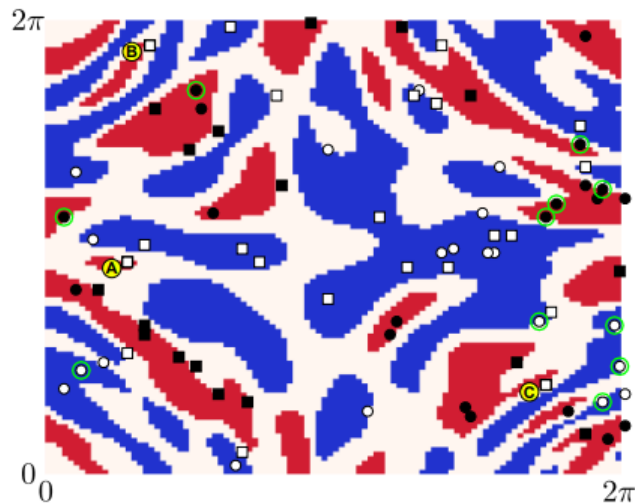
重心計算
= ユークリッド距離計算
= 内積計算
= Swap test で量子加速

$$O(\text{poly}(MN)) \rightarrow O(\log(MN))$$

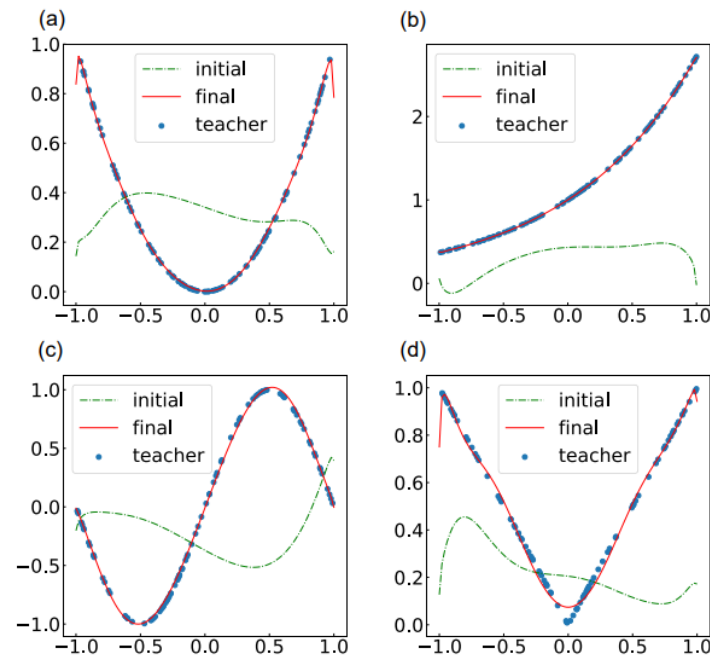
M: クラスタ数
N: データ次元

S. Lloyd et al., "Quantum algorithms for supervised and unsupervised machine learning", arXiv:1307.0411 (2013).

② 古典機械学習におけるモデルとして量子系を使うタイプ



SVMの量子版による分類[1]



NNの量子版による回帰[2]

- [1] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, pp. 209–212, 2019.
[2] K. Mitarai et al., “Quantum Circuit Learning,” Phys. Rev. A, vol. 98, no. 032309, 2018.

■ 量子機械学習 には様々なバリエーションがあるが、代表的な2種類を紹介する

① 古典機械学習の特定の計算部分を量子コンピュータに委託し、高速に計算する

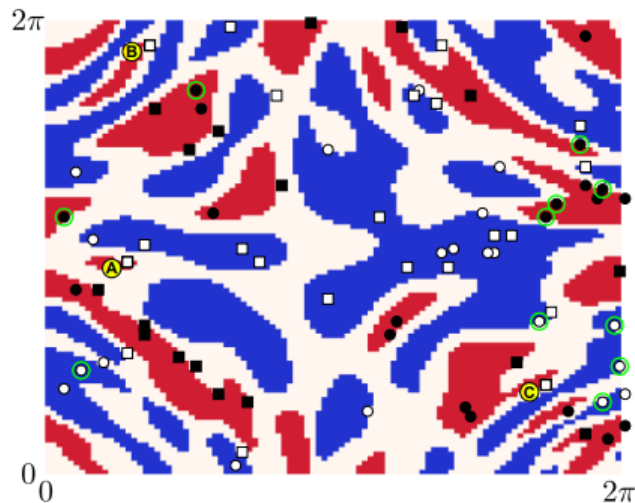
- 機械学習の精度は古典のみで行った場合と同一
- アダマールテストやGrover等の「量子加速に関して理論保証のあるアルゴリズム」を使う
→ long-term と考えられている

② 古典機械学習における表現（ニューラルネット、カーネル等）として量子系を使う

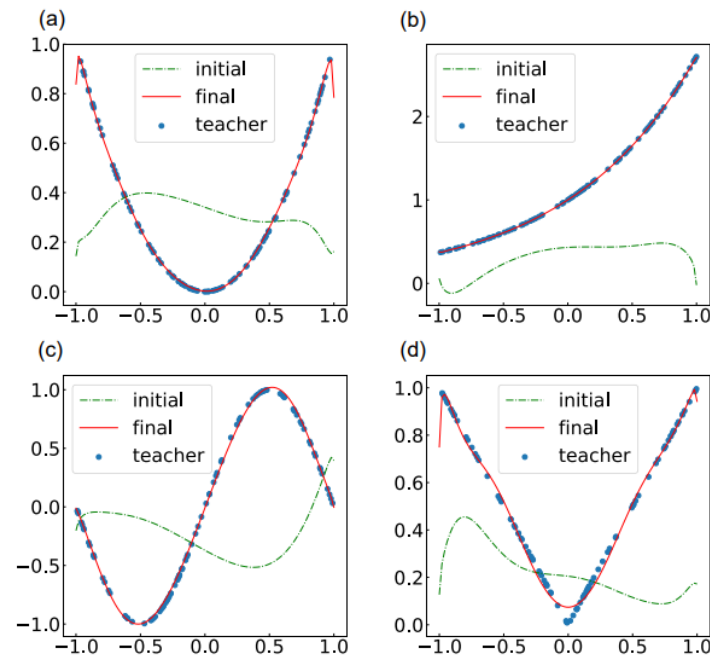
- 機械学習の精度が古典を上回る可能性がある
- たとえばノンパラメータ(SVM)、パラメータ(NN)の量子版がある
- 雑音があってもある程度動作するとされ、大規模or/and複雑な量子系の表現力に期待する → **near-term**と考えられている

Our focus

②古典機械学習におけるモデルとして量子系を使うタイプ^o



SVMの量子版による分類[1]



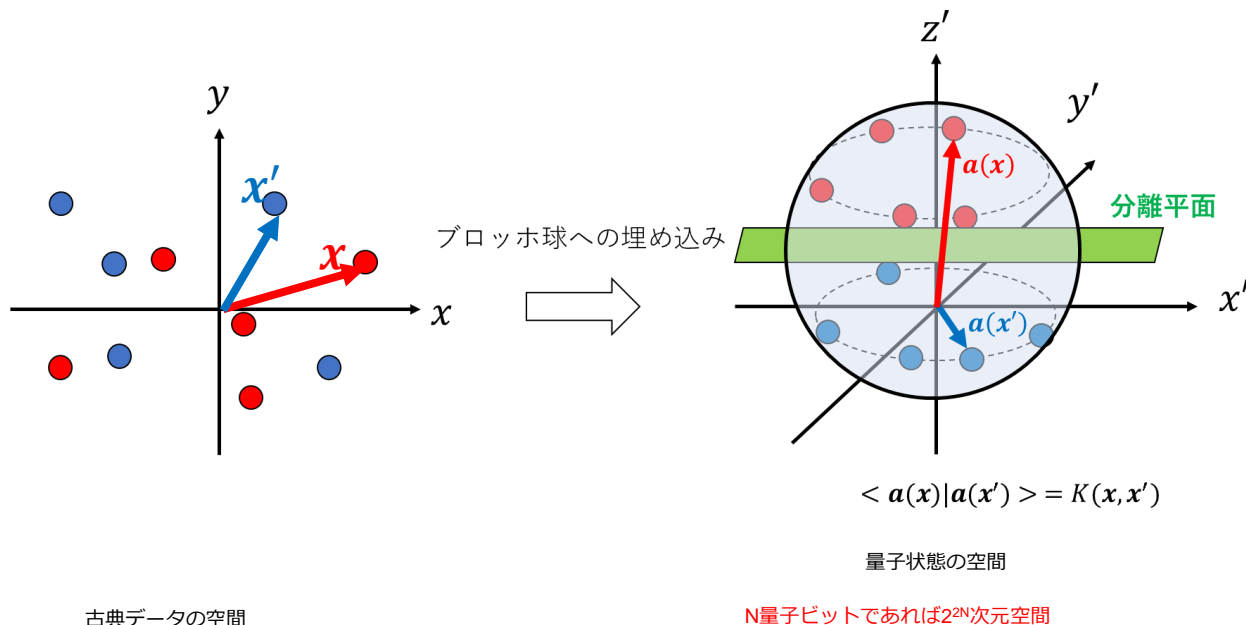
NNの量子版による回帰[2]

[1] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, pp. 209–212, 2019.

[2] K. Mitarai et al., “Quantum Circuit Learning,” Phys. Rev. A, vol. 98, no. 032309, 2018.

■ QSVMでは、データを埋め込む特徴量空間として量子状態空間を採用する。

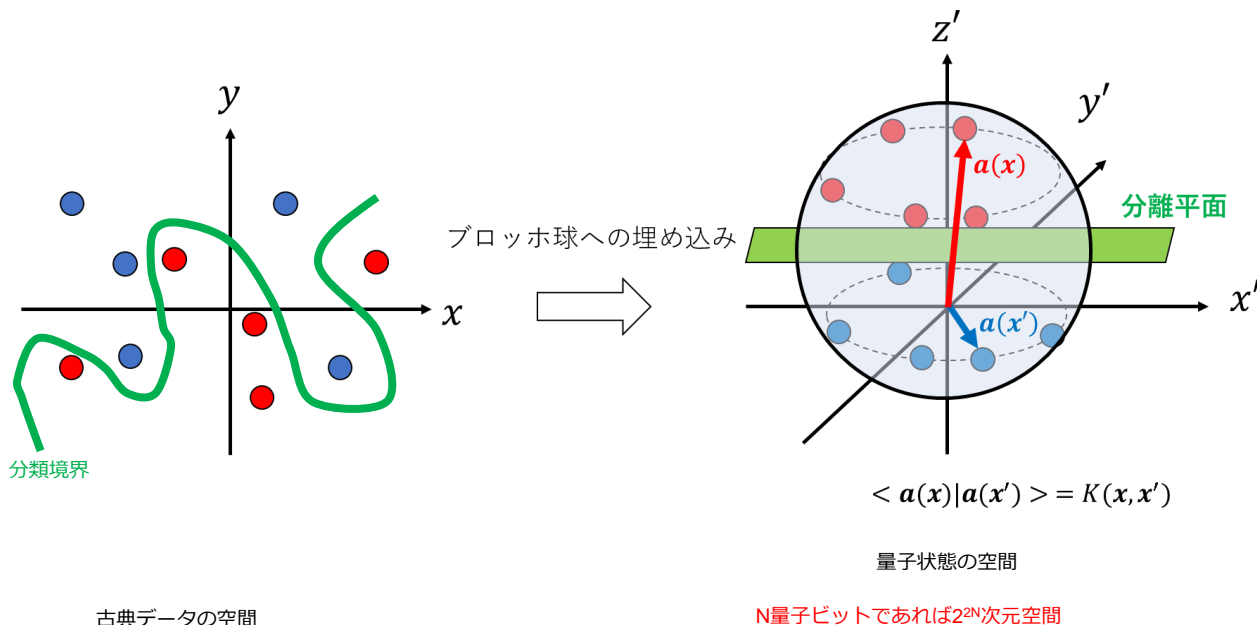
- N 量子ビットの状態空間(*)は 2^{2N} 次元であり、指数的に広い
- 複雑な分類が出来るのでは？と期待される



(*)ここでは実Blochベクトル表現を考えている

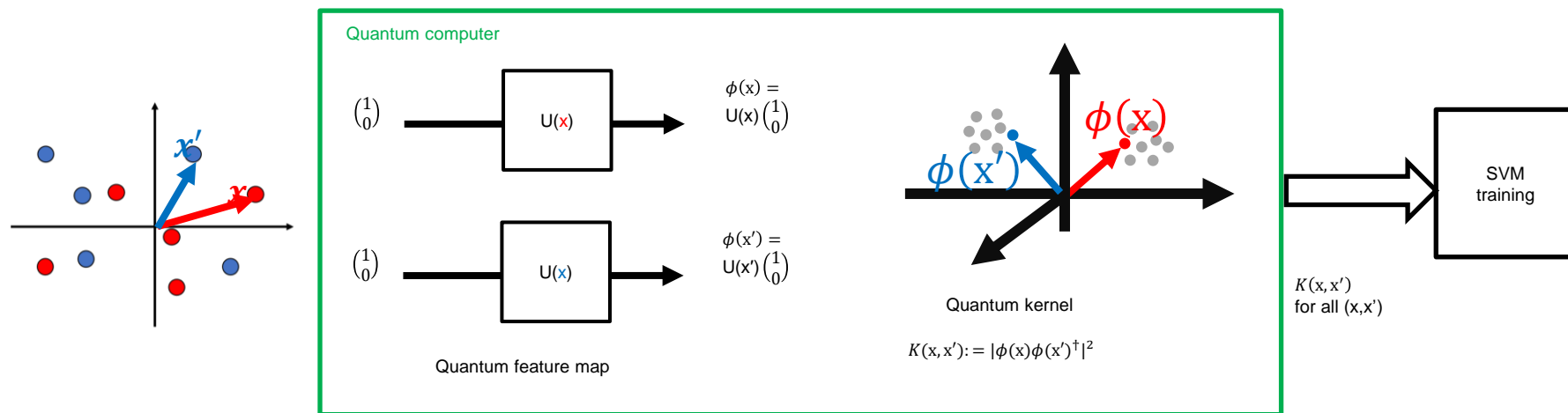
■ QSVMでは、データを埋め込む特徴量空間として量子状態空間を採用する。

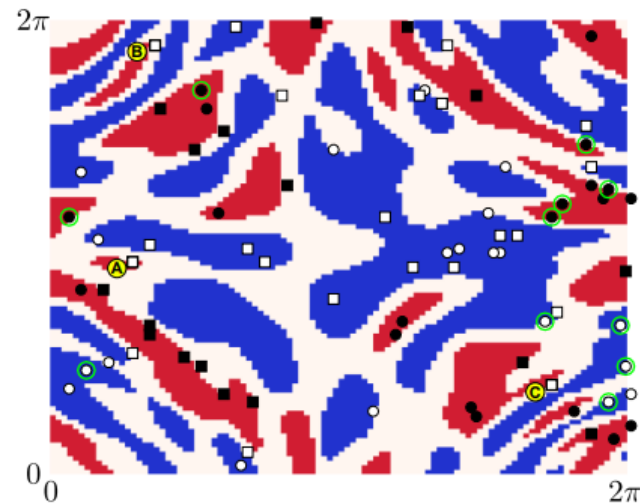
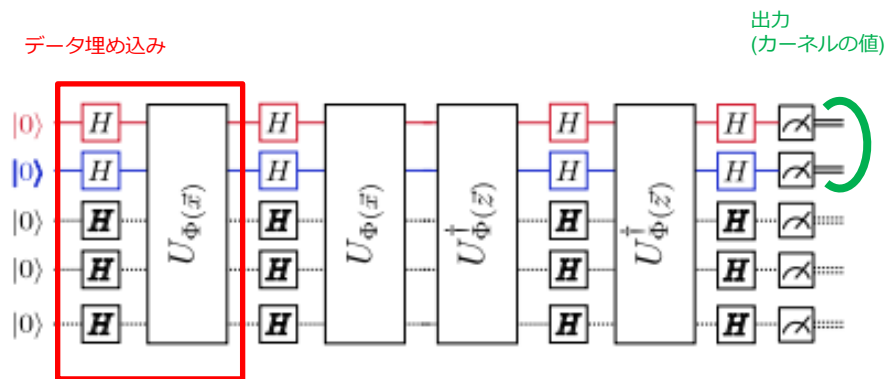
- N 量子ビットの状態空間(*)は 2^{2N} 次元であり、指数的に広い
- 複雑な分類が出来るのでは？と期待される



(*)ここでは実Blochベクトル表現を考えている

- 古典データをパラメータとする量子ゲート操作により、量子状態埋め込みを行う
 - どのような埋め込みが与えられた分類問題に適するのか？
- 量子状態ベクトル（指数的に高次元！）の内積はどうやって計算するのか
 - 内積計算も量子回路上で効率的に行うアルゴリズムが知られている

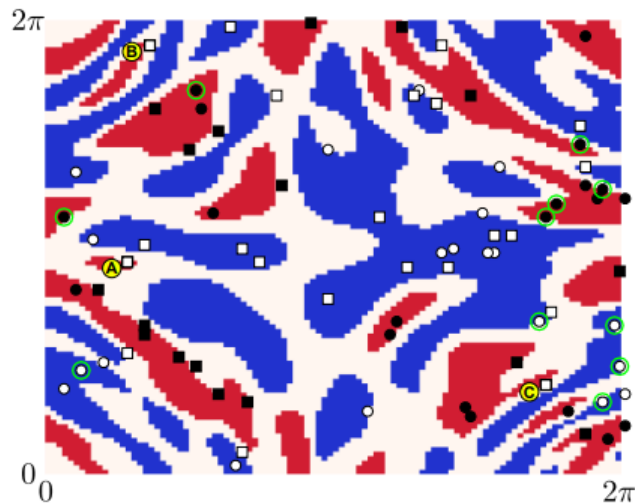




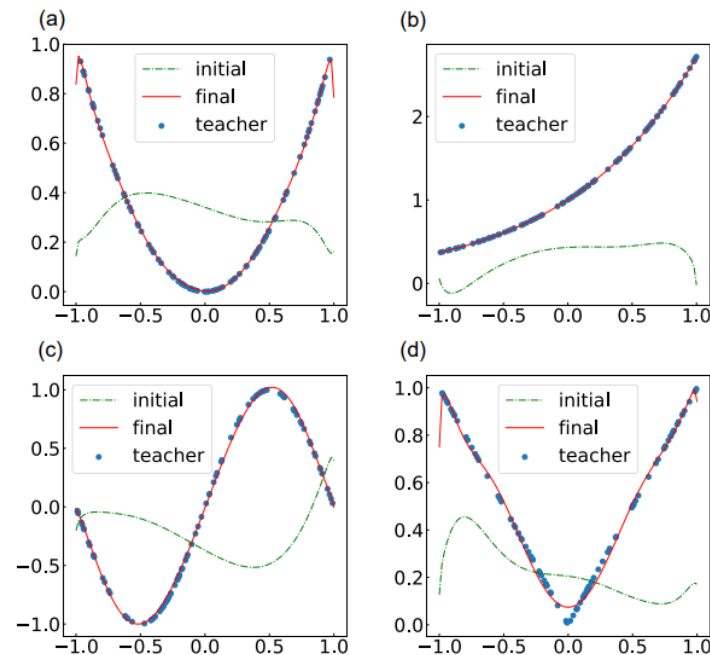
生成された分類境界の様子

V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, pp. 209–212, 2019.

②古典機械学習におけるモデルとして量子系を使うタイプ^o



SVMの量子版による分類[1]

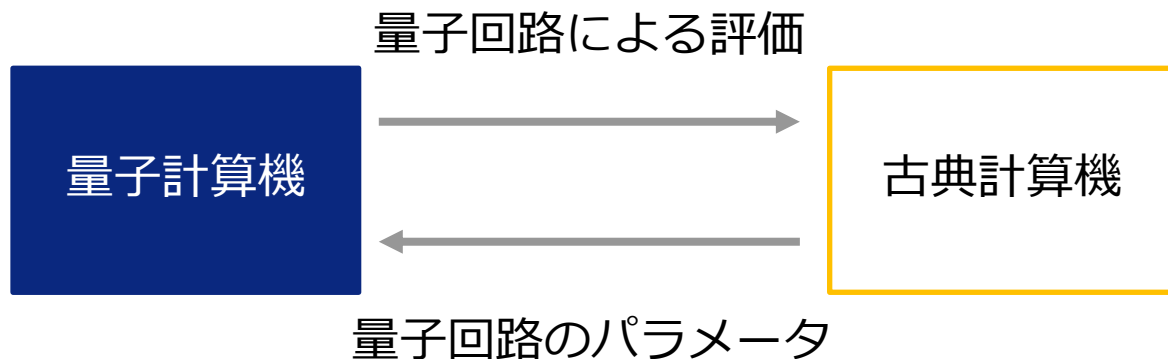


NNの量子版による回帰[2]

[1] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, pp. 209–212, 2019.

[2] K. Mitarai et al., “Quantum Circuit Learning,” Phys. Rev. A, vol. 98, no. 032309, 2018.

- 量子コンピュータを機械学習に応用するためのアルゴリズム
- 量子・古典ハイブリッドアルゴリズム (NISQで動作が期待される)
- 2019年3月にIBMの実験チームによるQCLの実機実装論文がNatureに掲載



- ポイント：指数的に高次元！の量子状態空間により表現力を向上

■ 基本的な考え方

- 入力データ x に対する出力 y を予測したい.
- QCLは x に対して y を出力するためのパラメータ θ を学習する.

■ 推論の流れ

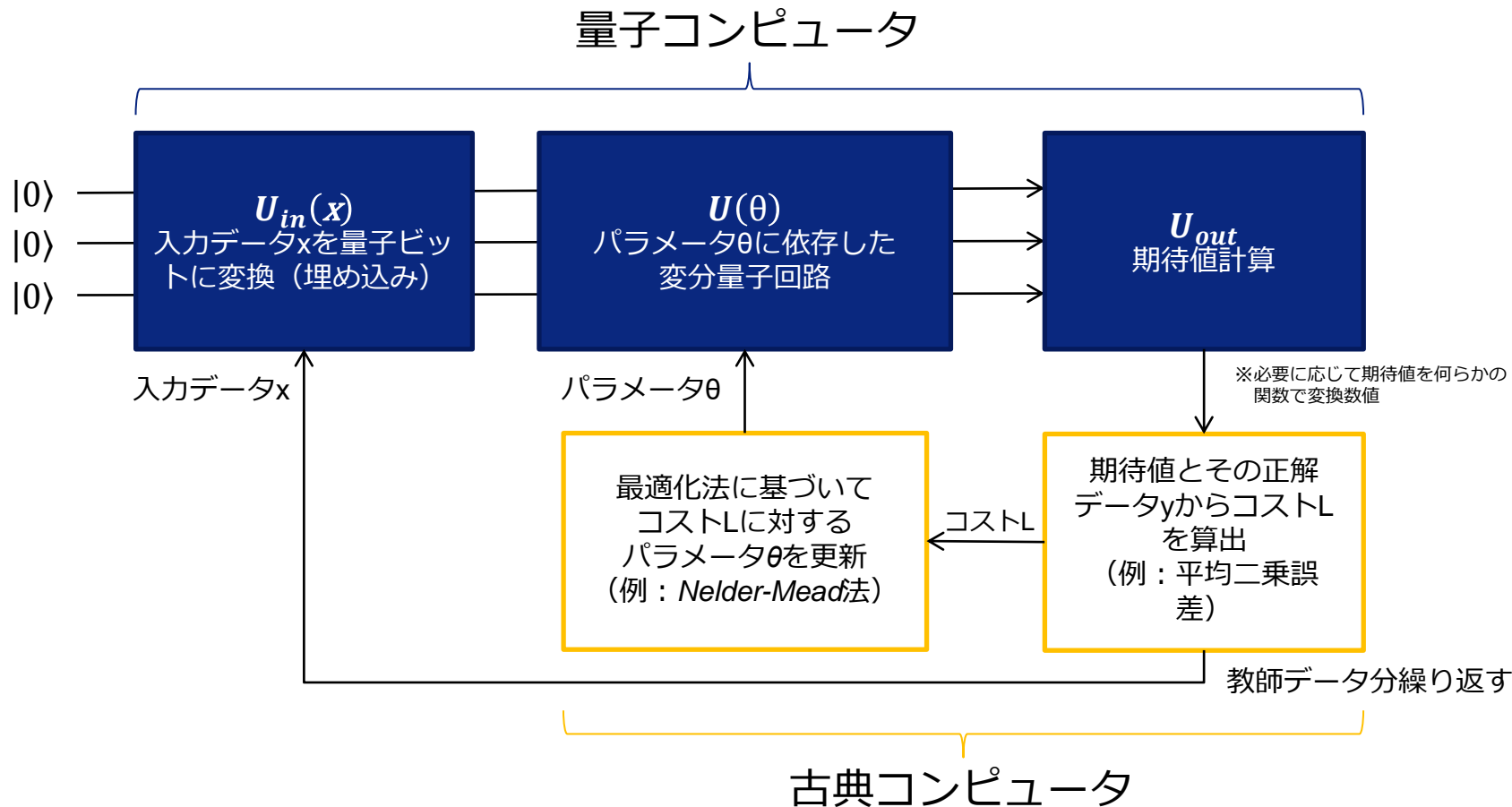
Neural NetworkのForward演算に相当

1. 入力データ x を量子ビット $|\psi\rangle$ に埋め込み
2. パラメータ θ に従って量子ビット $|\psi\rangle$ を操作し $|\psi'\rangle$ を生成 (変分量子回路)
3. 量子ビット $|\psi'\rangle$ における何らかの演算 Z の期待値 $\langle Z \rangle$ を計測 ※ここまでが量子回路
4. 期待値 $\langle Z \rangle$ を何らかの関数(sigmoidなど)で変換した結果が予測値 y

■ 学習の流れ

1. 学習用入力データ x_{train} に対して上記推論処理を実行し, 予測値 y を得る
2. 正解データ y_{train} と予測値 y のロス L を計算
3. ロス L を最小化するよう, パラメータ θ を更新
4. 全ての学習用データで1.-3.を繰り返す.

Neural Networkとの違い: 誤差逆伝搬せず、Forward演算を何度も実行して評価

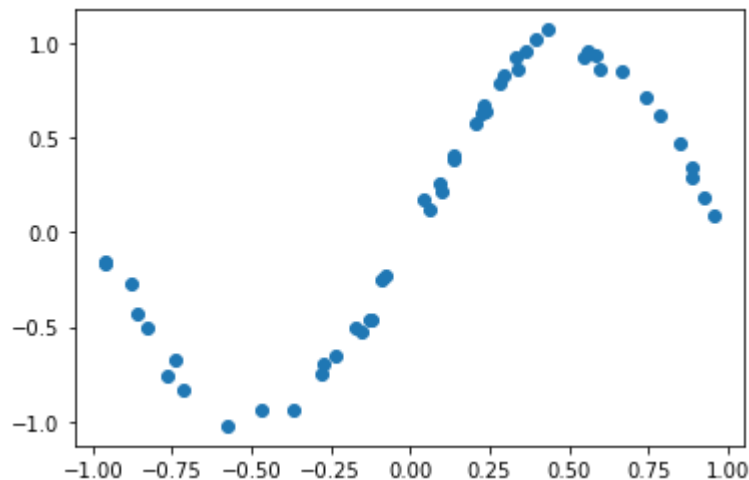


■ 問題 : Curve fitting

- 対象の関数 : $y = \sin \pi x$

■ まずは学習用データを作成

- x を $[-1,1]$ でランダム生成し, 対象の関数に少し誤差を与える
- 学習データ数 : 50個

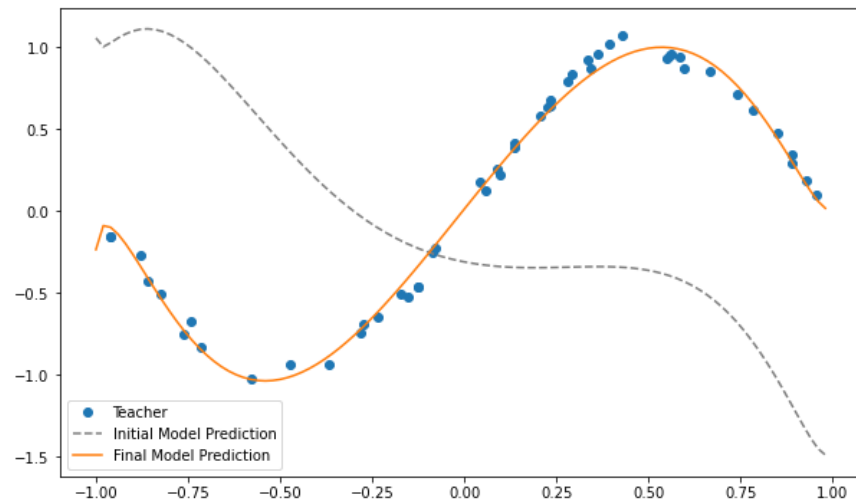


■ QCLの学習を実行 → 学習時間 43秒

- 量子ビット数 : 3
- 層の深さ : 3
- パラメータ θ の数 : 27個
 - ・ 量子ビット数 * 層の深さ * 変分用ゲート数
- パラメータ初期値 : $[0, 2\pi]$ のランダム生成
- ロス関数 : 平均二乗誤差 (MSE)
- 最適化法 : Nelder-Mead法

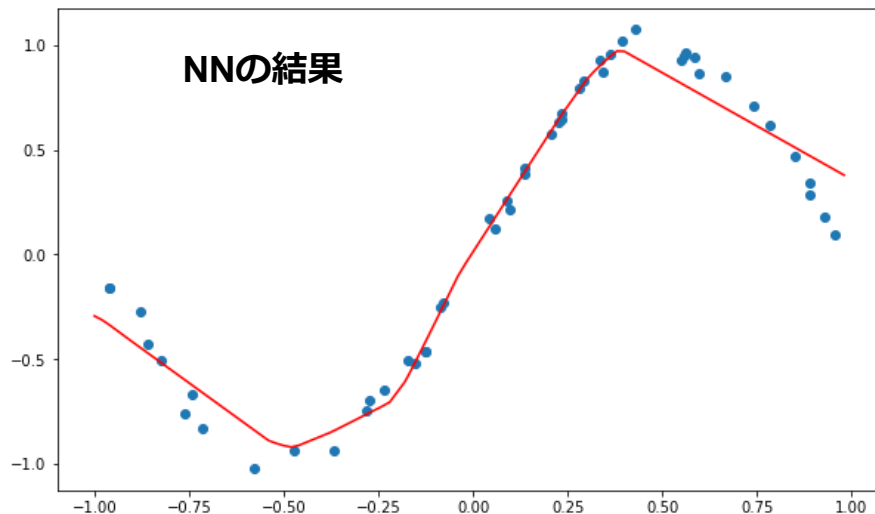
■ 学習結果の確認

- $[-1, 1]$ の連続値を学習済みQCLに入力
- その出力結果をプロット

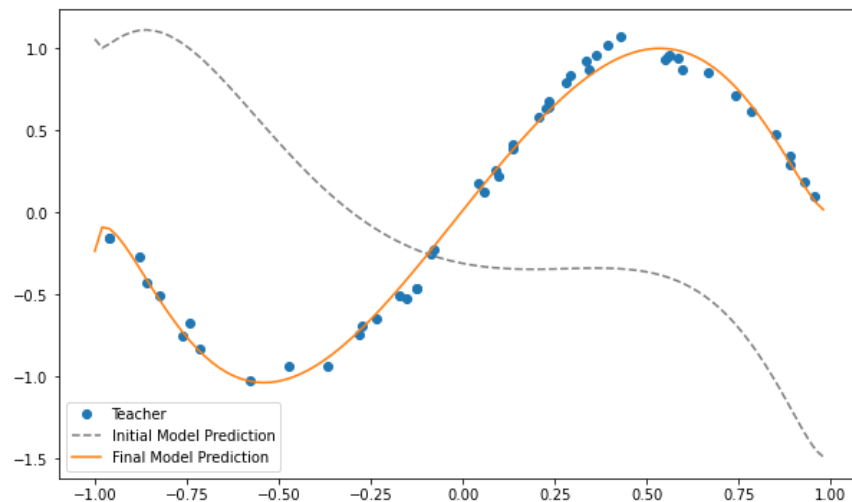


■ Kerasで実行 → 学習時間 6秒

- データは一緒
- 中間層も3層で、それぞれ9ユニット、活性化関数はReLU、ロス関数はMSE
 - ・ パラメータ数は210個
 - ・ エポックは300
- 最適化はAdam



QCLの結果（再掲）

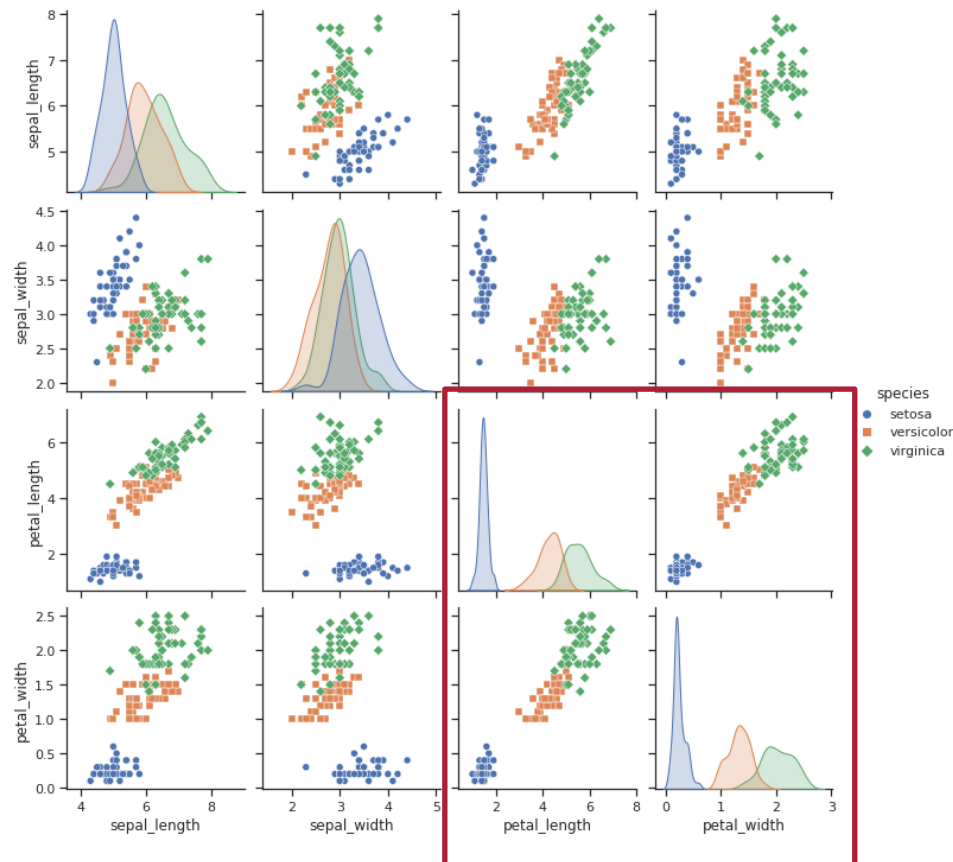


■ データ : scikit-learnのIris

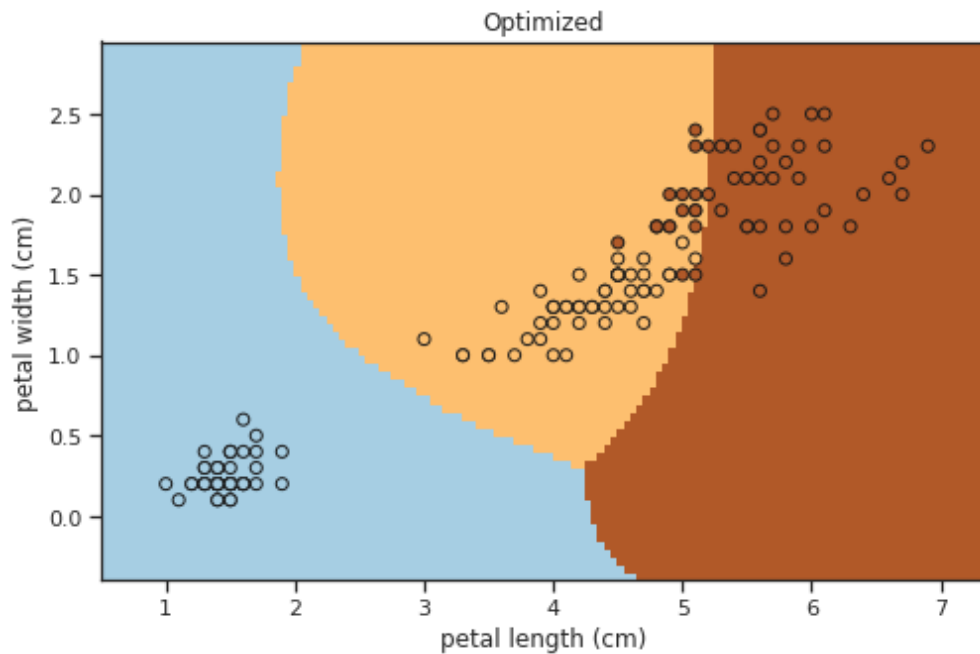
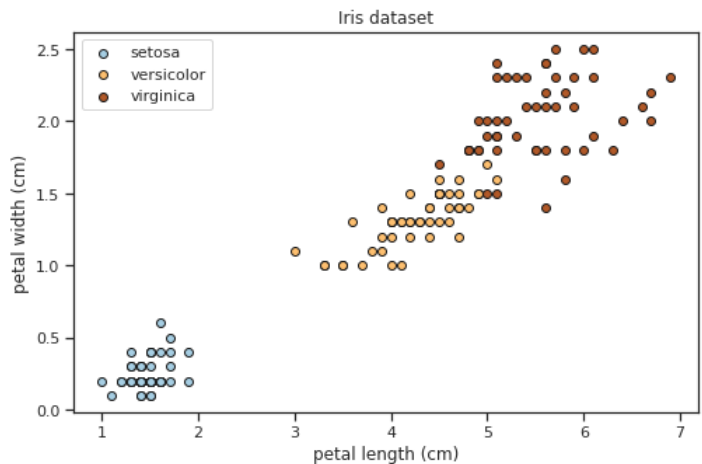
- アヤメの品種データ
- Setosa, Versicolor, Virginicaの3品種のデータが150件
- 特徴量 : がく片の長さ／幅, 花びらの長さ／幅
 - それぞれの相関関係 →

■ 花びらの長さと幅で分類

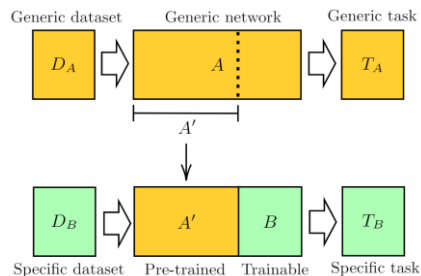
- 量子ビット数 : 3
- 層の深さ : 2
- パラメータ θ の数 : 18個
- ロス関数 : 平均二乗誤差 (MSE)
- 最適化法 : BFGS法



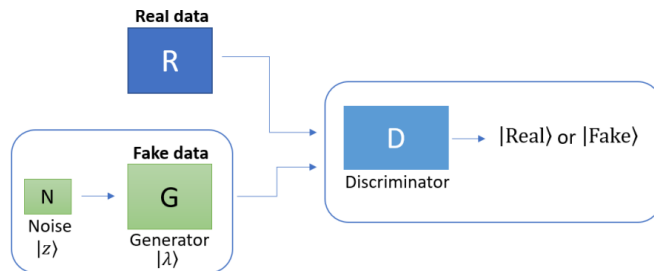
■ 分類結果 → 実行時間 43秒



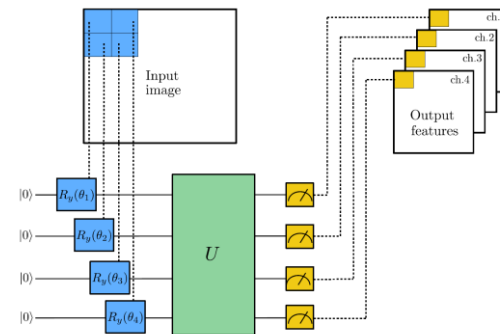
■ いろんな古典機械学習手法の量子版が提案されている



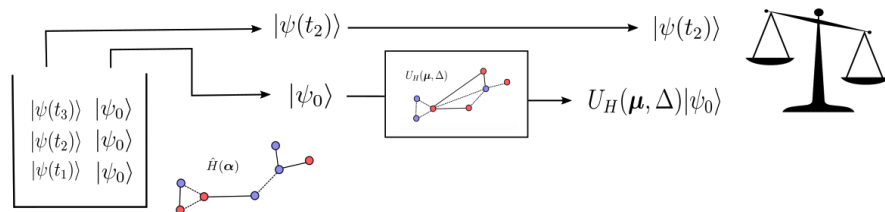
量子転移学習



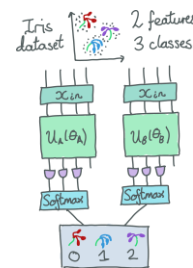
量子GAN



量子CNN



量子グラフNN



量子アンサンブル学習

- データの埋め込み・後処理がボトルネックとなる
- 大規模化すると、雑音やデコヒーレンスの影響が無視できない
- パラメータ最適化が大変（勾配の計算コストが高い、勾配消失が起こる）
- 変分量子回路の設計指針が確立されていない
- 量子アドバンテージがなかなか得られない
- etc.

3. 量子組合せ最適化アルゴリズム

株式会社Jij 山城 悠

株式会社KDDI総合研究所 伊神 皓生

量子組合せ最適化アルゴリズム (量子アニーリング)

株式会社 Jij

山城 悠

目次

- 数理最適化
- 局所最適化と大域最適化
- 量子アニーリング
- イジングモデル・QUBOを使った最適化計算の実装
- 国内外の研究開発
- まとめ
- 参考文献

数理最適化とは？

数理最適化とは？

解きたい課題

- ・配送計画問題
- ・基地局の周波数割り当て問題
- ・シフト作成問題



計画

例:

- 配送計画(ルート最適化)
- シフト作成(配置最適化)
- 材料の構造(構造最適化)

...

数理モデル

$$\max \sum_{\text{item}} R_{\text{item}}$$

目的関数

例: - 売上の最大化
- 費用の最小化

$$\sum_{\text{item}} f_{\text{item}} \leq C$$

制約条件

例: 予算の上限
運用ルール

...

アルゴリズム

- 汎用Solver
- 問題ごとにアルゴリズム実装
ヒューリスティクス



様々な数理モデル

連続最適化問題

- ・線形計画問題 (Linear programming problem: LP)
- ・凸2次計画問題
- ・非線形計画問題

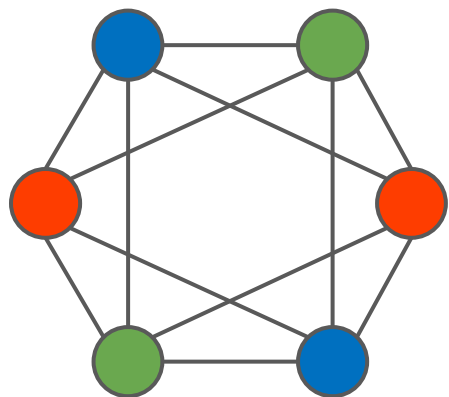
- 変数は離散か連続か？
- 関数は線形か非線形か？
- 制約条件ありか？無しか？

離散最適化問題 (組合せ最適化問題)

- ・典型問題
資源配分問題・ナップサック問題
最短経路問題
- ・混合整数計画問題 (Mixed Integer programming problem : MIP)
- ・二次制約無し二値最適化問題
(Quadratic Unconstrained Binary Optimization Problem: QUBO)
(Unconstrained Binary Quadratic optimization Problem: UBQP)

組合せ最適化問題：グラフ彩色問題

グラフ彩色問題：辺で接続されているノードを違う色で塗りたい



$$\min \sum_{u,v \in E} \sum_{c=0}^{n-1} x_{u,c} x_{v,c}$$

目的関数

$$\text{s.t.} \quad \sum_{c=0}^{n-1} x_{v,c} = 1, \quad \forall v \in V$$

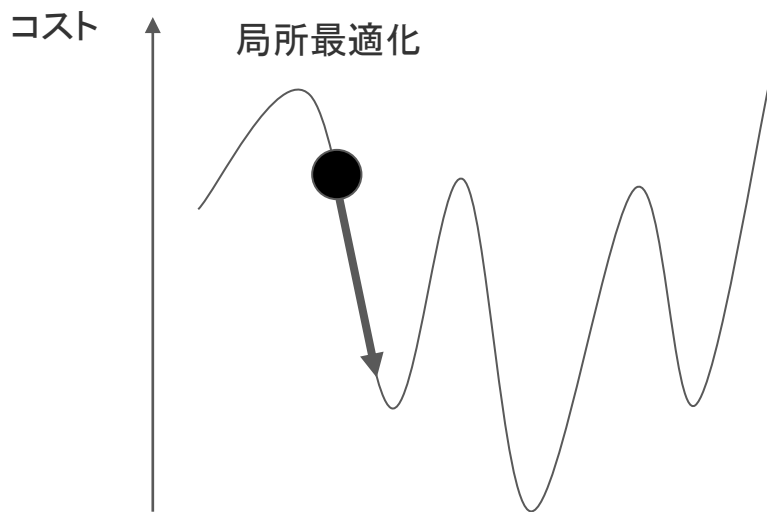
制約条件

$$x_{v,c} \in \{0, 1\}$$

決定変数

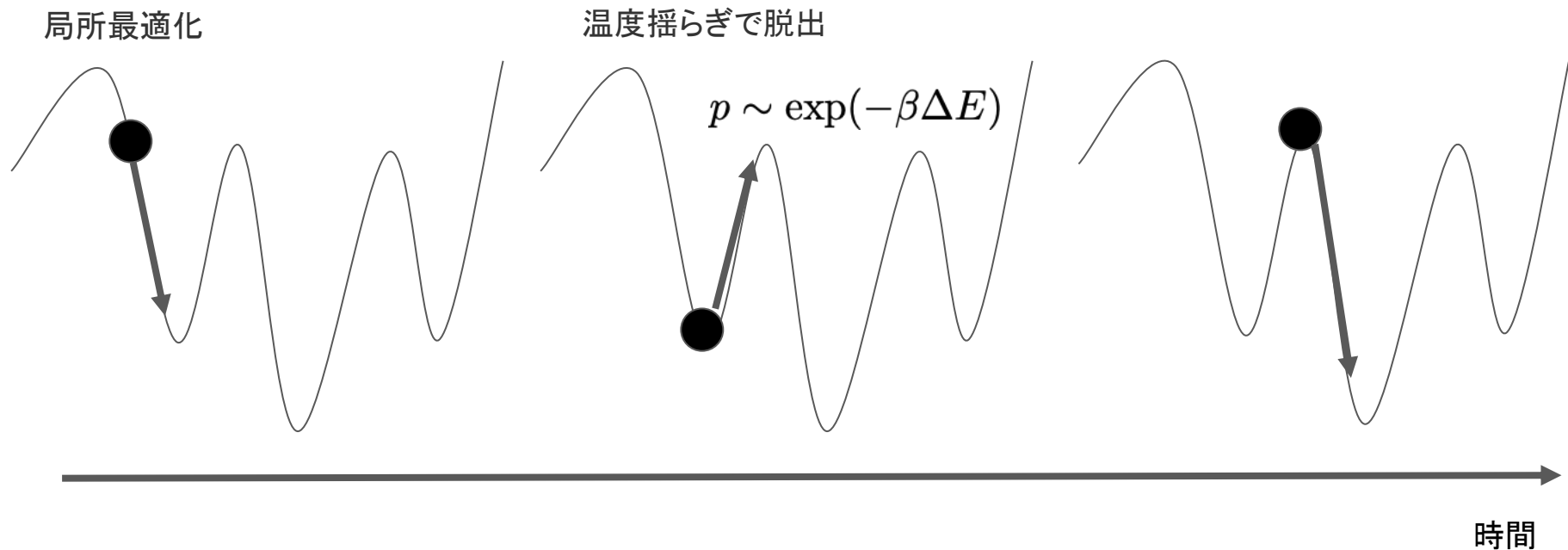
局所最適化と大域最適化

愚直に目的関数が下がる方向に動くと局所最適解にはまってしまう



局所最適化と大域最適化

シミュレーテッドアニーリング(焼きなまし法)



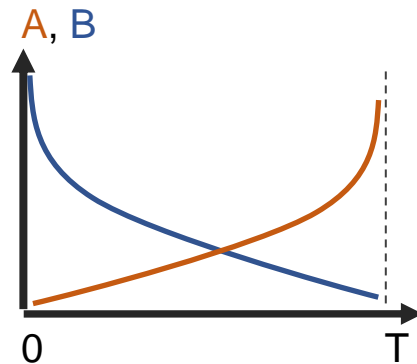
量子アニーリング

量子アニーリング

$$\hat{H}(t) = B(t)\hat{H}_r + A(t)\hat{H}_c$$

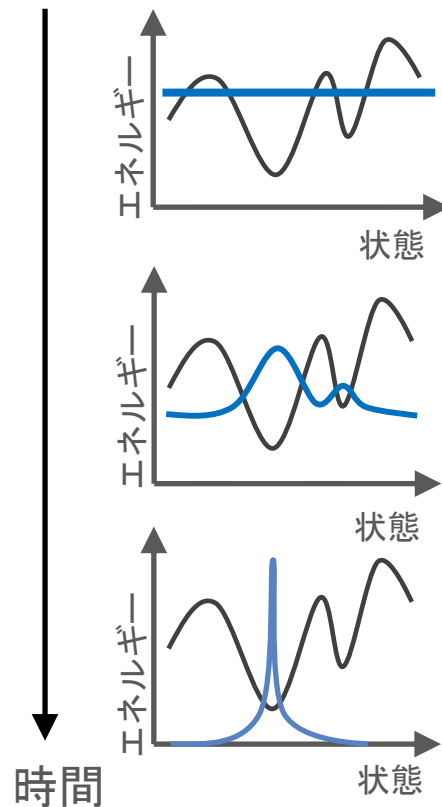
量子揺らぎ

解きたいコスト関数



$$A(t) : 0 \rightarrow 1$$

$$B(t) : 1 \rightarrow 0$$

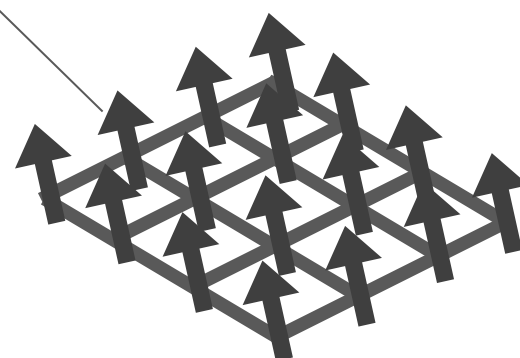


イジングモデル

$$\sum_{i < j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_i h_i \hat{\sigma}_i^z$$

$\hat{\sigma}_i^z \in \{-1, 1\}$

スピン

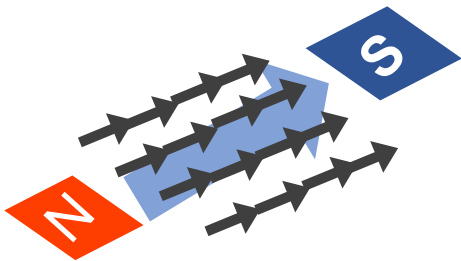


- ・磁石を表現するための物理モデル
- ・イジングモデルの基底状態(上記のエネルギー関数の最小エネルギー解)と求めたい解が対応

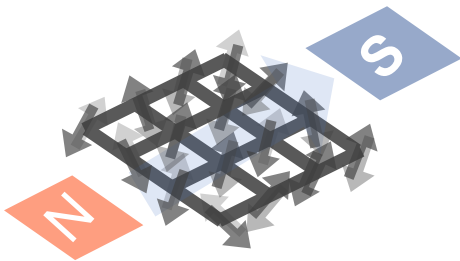
横磁場量子アニーリング

[3] T. Kadowaki, H. Nishimori, 1998

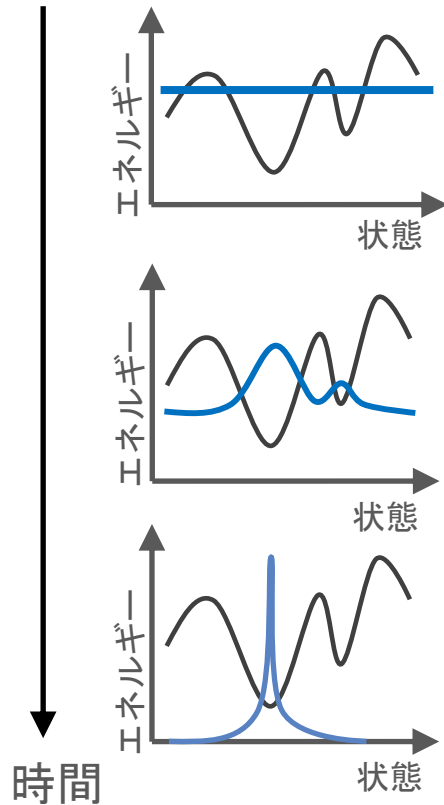
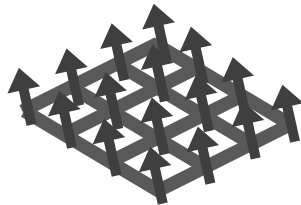
1. 横磁場（量子効果）
を強くかけ、
自明な状態にセット



2. 横磁場を徐々に
消していく



3. 横磁場を消して知りたかった
基底状態を得る



断熱量子計算 (Adiabatic Quantum Computing)

量子断熱定理

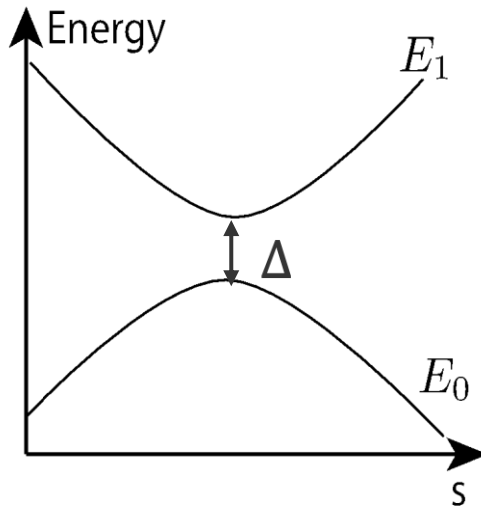
(計算時間とエネルギーギャップの関係)

$$T \propto 1/\text{Poly}(\Delta)$$

AQCでの量子プログラミング

=> ハミルトニアンの設計

(自由に量子ハミルトニアンを設計できることが前提)



量子揺らぎの設計と計算量の関係性は重要なテーマ

AQCについてまとまっているレビュー論文

[4] Tameem Albash and Daniel A. Lidar, “Adiabatic quantum computation” [Rev. Mod. Phys. 90, 015002 \(2018\)](#)

量子アニーリングマシン : D-Wave

横磁場量子アニーリング

$$\hat{H} = A(t) \left(\sum_{i < j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_i h_i \hat{\sigma}_i^z \right) + B(t) \left(\sum_i \hat{\sigma}_i^x \right)$$

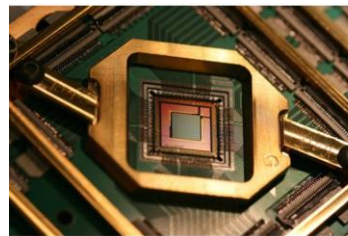
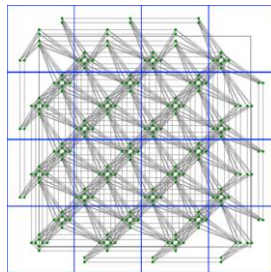
扱えるグラフ構造

配線などの物理的な理由により制限されたグラフの
イジングモデルしか扱えない

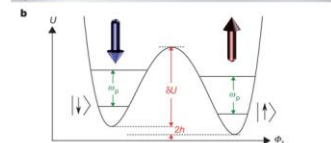
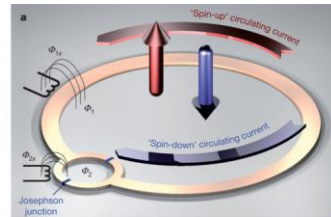
ペガサスグラフ

ノイズ

温度やノイズにより
理想的な量子アニーリング
を実行するのが難しい



www.dwavesys.com



Johnson et al. (2011)



[5] D-Wave Advantage

イジングモデルを使った最適化

QUBO (Quadratic Unconstrained Binary Optimization)

イジングモデル

$$\sum_{i < j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_i h_i \hat{\sigma}_i^z$$
$$\hat{\sigma}_i^z \in \{-1, 1\}$$

QUBO

$$\min_x \sum_{i,j} Q_{ij} x_i x_j$$
$$x_i \in \{0, 1\}$$



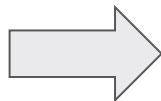
等価

制約付き最適化問題のQUBO表現

制約付き最適化問題

$$\min_x \sum_{i,j} Q_{ij} x_i x_j$$

$$\text{s.t. } \sum_i b_{ki} x_i = C_k, \forall k$$



ペナルティ法(罰金法)

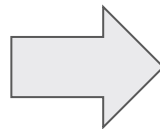
$$\min_x \sum_{i,j} Q_{ij} x_i x_j + A \sum_k \left(\sum_i b_{ki} x_i - C_k \right)^2$$

制約付き最適化問題のQUBO表現：グラフ彩色問題

制約付き最適化問題

$$\sum_{u,v \in E} \sum_{c=0}^{n-1} x_{u,c} x_{v,c}$$

$$\sum_{c=0}^{n-1} x_{v,c} = 1, \quad \forall v \in V$$



ペナルティ法(罰金法)

$$\sum_{u,v \in E} \sum_{c=0}^{n-1} x_{u,c} x_{v,c}$$

$$+ A \sum_{v \in V} \left(\sum_{c=0}^{n-1} x_{v,c} - 1 \right)^2$$

PyQUBO_[8, 9]を使った実装 : 数理モデリング

ペナルティ法 (罰金法)

$$\sum_{u,v \in E} \sum_{c=0}^{n-1} x_{u,c} x_{v,c} + A \sum_{v \in V} \left(\sum_{c=0}^{n-1} x_{v,c} - 1 \right)^2$$

\$ pip install pyqubo

```
import pyqubo as pyq
import numpy as np

V, n = 10, 3
E = [[0, 1], [1, 2], [3, 1], [2, 3]]

# 決定変数の作成
x = pyq.Array.create("x", shape=(V, n), vartype="BINARY")

# 目的関数項
H = sum(x[u, c]*x[v, c] for c in range(n) for u, v in E)
# 未定乗数となるプレースホルダーを作成
A = pyq.Placeholder("A")
# ペナルティ項の追加
H += A*pyq.Constraint(sum((np.sum(x[v, :]) - 1)**2 for v in range(V)), "onehot")
```

OpenJij_[10]を使ってQUBOを解く



イジングモデル・QUBO を解くためのOSS

D-Waveを使う時と同じインターフェース

Simulated Annealing (焼き鈍し法)
Simulated Quantum Annealing
(量子アニーリングの
シミュレーションアルゴリズム)

```
$ pip install openjij
```

```
import openjij as oj
```

```
# QUBOコンパイル
```

```
model = H.compile()
```

```
qubo, constant = model.to_qubo(feed_dict={"A": 1})
```

```
# OpenJij の Simulated Quantum annealingで解く
```

```
sampler = oj.SQASampler()
```

```
response = sampler.sample_qubo(qubo)
```

```
print(response)
```

```
x[0][0] x[0][1] x[0][2] x[1][0] x[1][1] x[1][2] ... x[9][2] energy num_oc.  
0 0 0 1 1 0 0 ... 0 -8.0 1  
['BINARY', 1 rows, 1 samples, 30 variables]
```

国内外での研究開発

国内外で研究開発・実証実験が進む

ハードウェアの開発

- D-Wave (カナダ) : 量子アニーリングマシンのクラウドサービスの提供
- 富士通 : 独自開発イジングマシン「デジタルアニーラ」の提供
- Microsoft (アメリカ) : Microsoft QIO (FPGA実装・複数のアニーリングアルゴリズム) の提供
- NTT, 日立, 東芝 : LASOLV, CMOSアニーリングマシン, SBM など独自のハードウェア・アルゴリズムを開発
- アメリカ・ヨーロッパでの国家プロジェクト



ONISQ プロジェクト (米)



AVaQus プロジェクト (欧)

応用プロジェクト

- Volkswagen : 交通流最適化 (リスボン市でのバス路線最適化)
- DENSO : 倉庫内 AGVの最適化
- OTI Lunimonics: 量子化学計算
- etc ...

まとめ

- 組合せ最適化を解く量子アルゴリズムとしての量子アニーリング
 - 現在の量子アニーリングハードウェアの課題
 - ・ 限られた量子ハミルトニアン（量子揺らぎ）
 - ・ 扱えるグラフ構造の制約
 - ・ ノイズ
 - イジングモデル・QUBOを使った最適化計算の実装のやり方 (PyQUBO, OpenJij)
 - 量子アニーリングはアルゴリズムなので量子コンピュータ上でも実現可能 (QAOA)
- 限られた量子ハミルトニアンなどの課題は、量子コンピュータ上での実装で解決されるかもしれない。

参考文献

数理最適化を初めて学ぶための本

[1] 穴井 宏和, 斉藤 努, “今日から使える! 組合せ最適化 離散問題ガイドブック”, 講談社, 2015.

[2] 梅谷 俊治, “しっかり学ぶ数理最適化 モデルからアルゴリズムまで”, 講談社, 2020.

量子アニーリングの提案論文

[3] T. Kadowaki, H. Nishimori, “Quantum Annealing in the Transverse Ising Model”, Phys. Rev. E 58, 5355, 1998.

AQCの理論研究についてまとまっているレビュー論文

[4] Tameem Albash and Daniel A. Lidar, “Adiabatic quantum computation” Rev. Mod. Phys. 90, 015002, 2018.

D-Wave Advantage についてのホワイトペーパー

[5] D-Wave Systems Inc., “Advantage Processor Overview”, D-Wave Technical Report Series, 2022.

イジング最適化の定式化集、これまでの産業界での応用例のレビュー

[6] A. Lucas, “Ising formulations of many NP problems”, Frontiers in Physics 2, 5, 2014.

[7] S. Yarkoni, “Quantum Annealing for Industry Applications: Introduction and Review”, arXiv:2112.07491, 2021.

OSSのQUBOモデリングツール

[8] K. Tanahashi, *et al.*, Application of Ising Machines and a Software Development for Ising Machines, J. Phys. Soc. Jpn. 88, 061010, 2019.

[9] PyQUBO (GitHub): <https://github.com/recruit-communications/pyqubo>

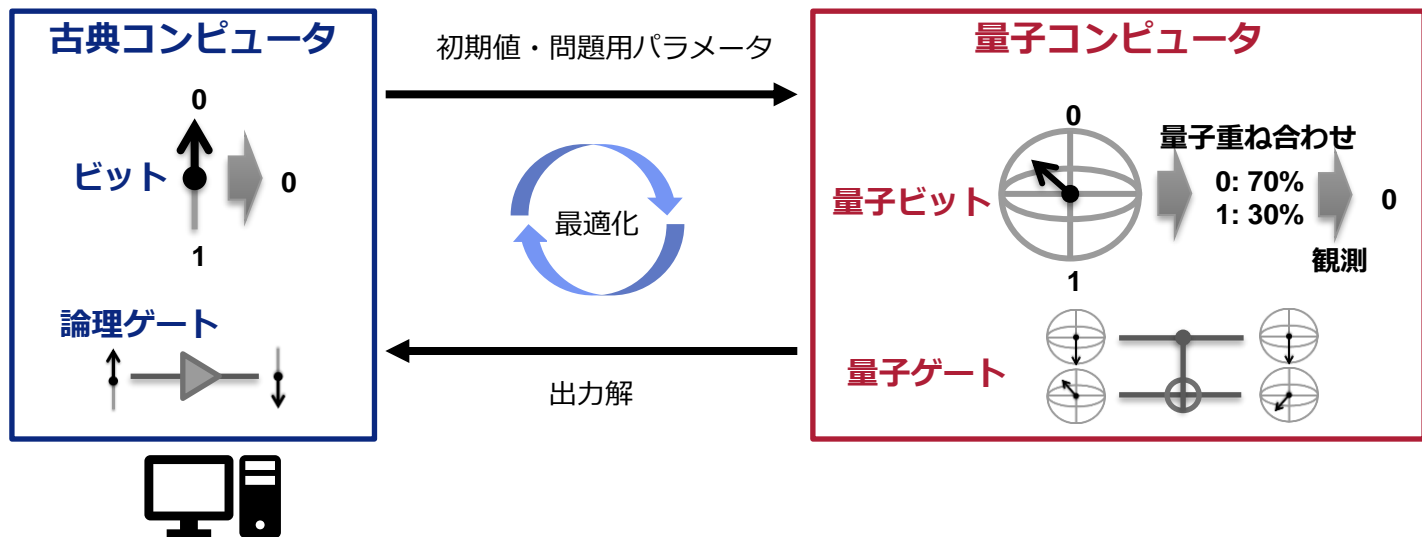
OSSのイジング最適化ツール

[10] OpenJij (GitHub): <https://github.com/OpenJij/OpenJij>

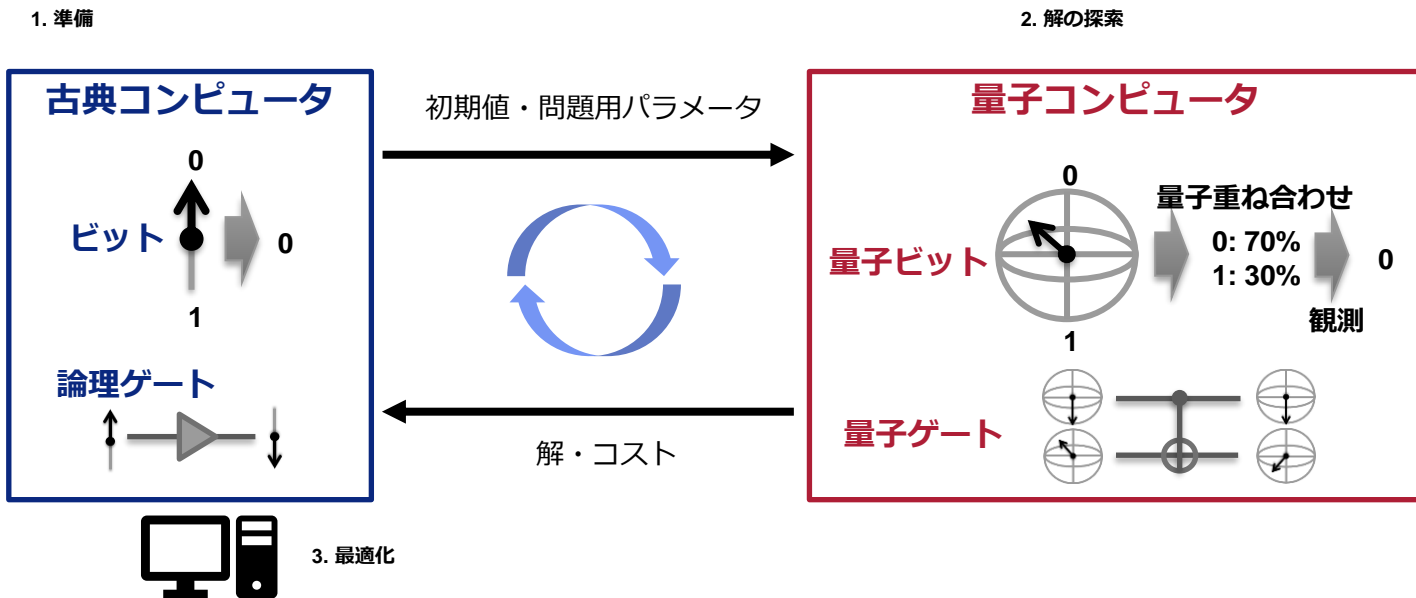
QAOA(Quantum Approximate Optimization Algorithm)

株式会社KDDI総合研究所 伊神 皓生

- NISQアルゴリズムの一つ **Quantum Approximate Optimization Algorithm (QAOA; 量子近似最適化アルゴリズム)**を学ぶ。
- QAOAは、量子アニーリングと同様に、**組み合わせ最適化問題の解を求めるためのアルゴリズム**である
- 古典コンピュータと量子コンピュータを用いる**ハイブリットアルゴリズム**



1. 準備：量子コンピュータで扱えるよう、組み合わせ最適化問題を**変換**
2. 探索：量子コンピュータに初期値とパラメータを入力し、**解を探索**
3. チューニング：量子コンピュータからの出力から探索パラメータを**古典コンピュータで最適化**



■ 構成要素

- 決定変数 : $x \rightarrow$ 量子状態 $|q\rangle$ (ベクトル) で表現
- 目的関数 : $f(x)$, 制約 : $y(x) \leq 0 \rightarrow$ ハミルトニアン H (行列) で表現
- コスト関数 $C = \langle q | H | q \rangle$ を最小化する問題に変換
- ハミルトニアン H は問題毎に生成し、以下のイジングモデルで記載する

$$H = \sum_{i=1}^n \sum_{j=1}^i J_{ij} Z_i Z_j + \sum_{i=1}^n h_i Z_i$$

$$Z = Z_1 Z_2 \dots Z_n (Z_i = 0, 1)$$

J_{ij} : Z_i, Z_j の相互作用
 h_i : Z_i の係数

■ 量子断熱計算

- 求めたい解を基底状態に持つコスト関数ハミルトニアン $H_c(t = t_f)$
- 自明な基底状態を持つリファレンスハミルトニアン(初期値) $H_r(t = 0)$
- 量子断熱定理と呼ばれる量子系の性質により、時間を $t=0 \rightarrow t = t_f$ と少しずつ変化させることで、 H_c を得ることができる

$$H(t) = \left(1 - \frac{t}{t_f}\right) H_r + \frac{t}{t_f} H_c$$

- つまり、量子状態の時間発展を量子コンピュータで模擬できれば良い
 - ・ 汎用な量子コンピュータ(量子ゲート)で模擬→QAOA
 - ・ 専用ハードウェア(D-WAVE)で模擬→量子アニーリング

- 量子状態の時間発展は、ユニタリ変換 $U(t) = e^{-iHt}$ で表現できる
- $U(t) = e^{-iHt}$ を Trotter 分解で近似すると、2量子ビットゲートの多項式個の作用の積で表現できる
 →汎用の量子コンピュータで量子断熱定理の時間発展を近似演算することができる！
 →初期値 $|q_{init}\rangle$ から $|q\rangle = U(\beta, \gamma) |q_{init}\rangle$ を演算し、近似的に最小エネルギーとなるコスト関数 $C = \langle q | Hc | q \rangle$ を計算可能

QAA代入

$$H(t) = \left(1 - \frac{t}{t_f}\right) H_r + \frac{t}{t_f} H_c$$

$$U(t) = e^{-iH(t)t} \approx \left(e^{-i\frac{1}{p}(1-\frac{t}{t_f})H_r t} e^{-i\frac{1}{p}\frac{t}{t_f}H_c t} \right)^p$$

$$= e^{-i\beta_p H_r} e^{-i\gamma_p H_c} e^{-i\beta_{p-1} H_r} e^{-i\gamma_{p-1} H_c} \dots e^{-i\beta_0 H_r} e^{-i\gamma_0 H_c}$$

Trotter 分解で p 分割

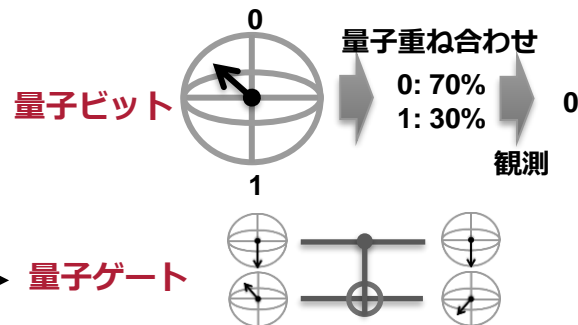
$$\beta = 1/p(1 - t/t_f)t, \quad \gamma = 1/p(t/t_f)t$$

量子ゲートで記載できる

パラメータ $(\beta, \gamma)_p$

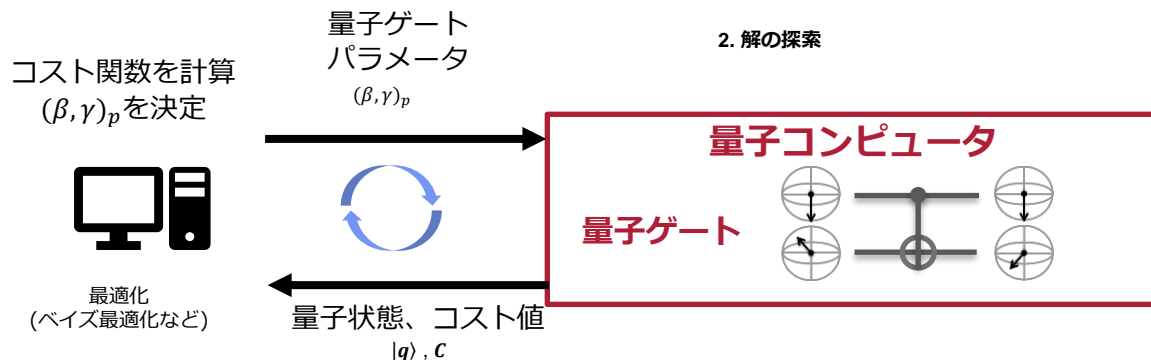
2. 解の探索

量子コンピュータ



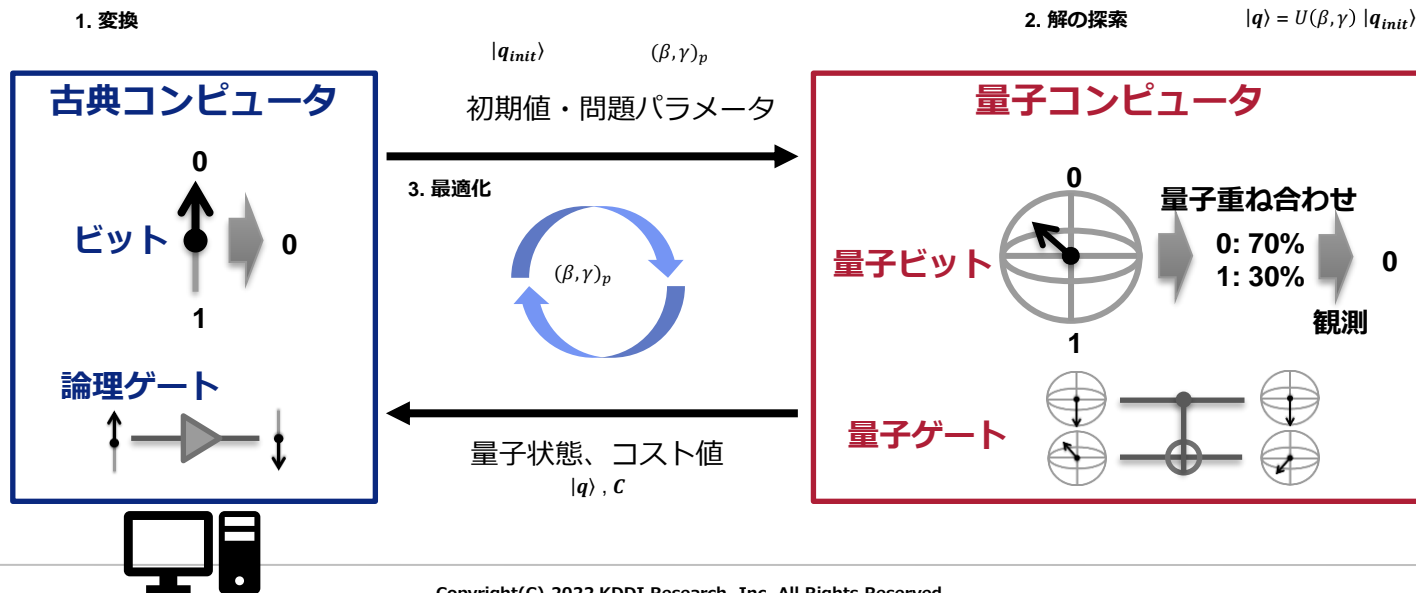
■ 量子コンピュータの動作パラメータ(近似パラメータ)を最適化する

1. 古典コンピュータを用いてコスト関数 $C = \langle q | H | q \rangle$ がより小さくなるよう、量子ゲートのパラメータ $(\beta, \gamma)_p = \beta_1, \gamma_1, \dots, \beta_p, \gamma_p$ をアップデートする
→これは、量子ゲートを最適化問題の性質に応じて調整することにあたる
2. パラメータが決まったら、量子状態 $|q\rangle$ を観測し最適化問題の解とする。



- QAOAは、量子ゲートのパラメータを古典コンピュータで最適化する
- 問題の特性にあった量子ゲートパラメータ $(\beta, \gamma)_p$ を上手く見つけさえすれば、少ない試行回数で近似解 $|q\rangle$ を観測することができる
→ 量子アドバンテージを得られる最適化問題の特徴を見極める必要
- 課題として、 Trotter分解の p を大きく (=ゲート数大) し、近似精度を高めようとする と量子雑音が増加するトレードオフが存在
→ 直近の実用化に向け、 $p=1$ など、ゲート数を絞った場合の検討が
→ 長期的には、ゲート数が増え量子雑音がないHWが開発され量子ゲートの表現力が向上し、より広範囲な問題への適応ができるかも？

コスト関数 $C = \langle q | H | q \rangle$



4. 量子探索アルゴリズム

株式会社KDDI総合研究所 成定 真太郎

- 量子探索アルゴリズムの1つ
- 現在は分かりやすさの観点から、量子振幅増幅(QAA: Quantum Amplitude Amplification)の一部として論文上で紹介されることが多い
- n ビットの探索問題を $O(2^{n/2})$ で求解する
 - 解が1つの場合
 - 全探索の計算量: $O(2^n)$
- 様々な探索問題に応用できる
(SAT問題、部分和問題、積分、暗号解読、etc.)

$n = 4$

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

解

Groverのアルゴリズム

入力： n ビットの全状態の重ね合わせ $|s\rangle = \underbrace{\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle}_{2^n \text{個}} \quad (n = 2)$
出力：解 w

■ アルゴリズム

- ① n ビットの全状態の重ね合わせ $|s\rangle$ を入力
- ② オラクル U_w を $|s\rangle$ に作用させる： $|s'\rangle = U_w|s\rangle$
- ③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$
- ④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

※各量子ビットのブラケット表記
(中身はただのベクトルです)

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

※ $n = 2$ 、解 $w = 11$ を考える

Groverのアルゴリズム

入力： n ビットの全状態の重ね合わせ $|s\rangle = \underbrace{\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle}_{2^n \text{個}} \quad (n = 2)$
出力：解 w

■ アルゴリズム

- ① n ビットの全状態の重ね合わせ $|s\rangle$ を入力
- ② オラクル U_w を $|s\rangle$ に作用させる： $|s'\rangle = U_w|s\rangle$
- ③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$
- ④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

※各量子ビットのブラケット表記
(中身はただのベクトルです)

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

※ $n = 2$ 、解 $w = 11$ を考える

① n ビットの全状態の重ね合わせ $|s\rangle$ を入力

- 初期状態 $|00\rangle$ に対してアダマール行列 H を適用する：

$$(H \otimes H)|00\rangle = |s\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

係数の二乗が各量子ビットの存在確率（それぞれ25%）

Groverのアルゴリズム

入力： n ビットの全状態の重ね合わせ $|s\rangle = \underbrace{\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle}_{2^n \text{個}} \quad (n = 2)$
出力：解 w

■ アルゴリズム

- ① n ビットの全状態の重ね合わせ $|s\rangle$ を入力
- ② オラクル U_w を $|s\rangle$ に作用させる： $|s'\rangle = U_w|s\rangle$
- ③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$
- ④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

※ $n = 2$ 、解 $w = 11$ を考える

- 入力ビット列 x に対して、 x が問題の解かどうか2択で教えてくれる関数

$$f(x) = \begin{cases} 1 & (x \text{ が解}) \\ 0 & (x \text{ が解ではない}) \end{cases}$$

- オラクルの構築方法：①解そのものを使う ②解の判定条件を使う

- 例) 素因数分解：素数 x は N の素因数？ (p, q を N の素因数とする)

①

$$f(x) = \begin{cases} 1 & (x = p \text{ or } x = q) \\ 0 & (\text{otherwise}) \end{cases}$$

②

$$f(x) = \begin{cases} 1 & (N \bmod x = 0) \\ 0 & (\text{otherwise}) \end{cases}$$

- 量子ゲートで①もしくは②のオラクル $f(x)$ を構築することを考える

- 現実の問題を解くにはもちろん②でなければならないが、以降は簡単のため①を考える

② オラクル U_w を $|s\rangle$ に作用させる : $|s'\rangle = U_w|s\rangle$

- オラクル $f(x)$ の代わりに次のようなオラクル行列 U_w を定義 :

$$f(x) = \begin{cases} 1 & (x = 11) \\ 0 & (\text{otherwise}) \end{cases}$$

$$U_w|x\rangle = \begin{cases} -|x\rangle & (x = 11) \\ |x\rangle & (\text{otherwise}) \end{cases}$$

$|x\rangle$ が解の時
符号を反転

- 解が判明している場合、 U_w は次のように構築できる :

$$U_w = I - 2|w\rangle\langle w| = I - 2|\mathbf{11}\rangle\langle\mathbf{11}| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

- $|s'\rangle = U_w|s\rangle$ は次のようになる :

$$|s'\rangle = U_w|s\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|\mathbf{11}\rangle$$

- 解（に対応する量子ビット）のマーキング操作ととらえることもできる

Groverのアルゴリズム

入力： n ビットの全状態の重ね合わせ $|s\rangle = \underbrace{\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle}_{2^n \text{個}} \quad (n = 2)$
出力：解 w

■ アルゴリズム

- ① n ビットの全状態の重ね合わせ $|s\rangle$ を入力
- ② オラクル U_w を $|s\rangle$ に作用させる： $|s'\rangle = U_w|s\rangle$
- ③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$
- ④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

※ $n = 2$ 、解 $w = 11$ を考える

③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$

■ ②でマーキングした解に対応する量子の存在確率を**増幅**させ、それ以外の量子の存在確率を**減衰**

■ 振幅増幅行列 U_s ：

$$U_s = 2|s\rangle\langle s| - I$$

※ s は重ね合わせ状態

■ 2量子ビットの場合、 $U_s = \frac{1}{2} \begin{bmatrix} -1 & +1 & +1 & +1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & +1 \\ +1 & +1 & +1 & -1 \end{bmatrix}$

■ $|s''\rangle$ は次のようになる：

$$|s''\rangle = U_s U_w |s\rangle = 0 \cdot |00\rangle + 0 \cdot |01\rangle + 0 \cdot |10\rangle + \mathbf{1 \cdot |11\rangle}$$

Groverのアルゴリズム

入力： n ビットの全状態の重ね合わせ $|s\rangle = \underbrace{\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle}_{2^n \text{個}} \quad (n = 2)$
出力：解 w

■ アルゴリズム

- ① n ビットの全状態の重ね合わせ $|s\rangle$ を入力
- ② オラクル U_w を $|s\rangle$ に作用させる： $|s'\rangle = U_w|s\rangle$
- ③ 振幅増幅操作 U_s を $|s'\rangle$ に作用させる： $|s''\rangle = U_s|s'\rangle$
- ④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

※ $n = 2$ 、解 $w = 11$ を考える

④ ②－③を $2^{n/2}$ 回行った後に $|s''\rangle$ を測定

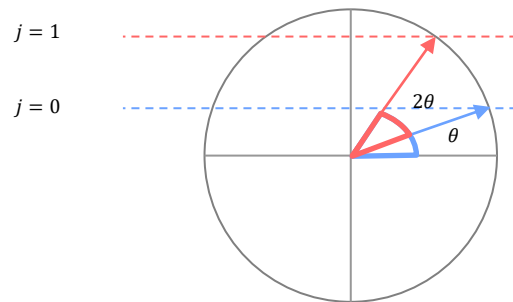
- $|s''\rangle = U_s U_w |s\rangle$ に対して、 $|s''\rangle$ を解 ($|w\rangle$) とそれ以外 ($|\hat{s}\rangle$) に分けて考える
- $\sin \theta = \frac{1}{\sqrt{2^n}}$ に対して、②－③を j 回行った後の $|s''\rangle$ は次のように書ける：

$$|s''\rangle = (U_s U_w)^j |s\rangle = \sin((2j+1)\theta) |w\rangle + \cos((2j+1)\theta) |\hat{s}\rangle$$

- 1イテレーション毎に解の量子ビット $|w\rangle$ の存在確率が**増幅**、それ以外が**減衰**
- $|w\rangle$ の存在確率が最大化するのは、 $\sin((2j+1)\theta) = 1 \Leftrightarrow (2j+1)\theta = \frac{\pi}{2}$ 、 $\sin \theta \leq \theta$ より：

$$j \leq \frac{\pi}{4} \sqrt{2^n} - \frac{1}{2} \leq 2^{n/2}$$

- $j = 2^{n/2}$ 回後に測定すると、 $|w\rangle$ が高確率で出力される



Groverを使った量子探索の応用

■ 様々な研究領域においてGroverのアルゴリズムが活用されています

■ 一例：

計算複雑性

衝突問題 彩色問題
SAT 部分和問題

データベース

クエリ (SELECT句など)
集合演算 関係演算

セキュリティ

暗号解読 (共通鍵暗号、
公開鍵暗号、ハッシュ関数)

通信

信号検出
ルーティング

機械学習

SVM 最近傍法
Neural Network クラスタリング
強化学習 k-means

数学

多変数多項式
積分

文字列学

文字列照合
共通文字列

■ 様々な研究領域においてGroverのアルゴリズムが活用されています

■ 一例：

計算複雑性

衝突問題 彩色問題
SAT **部分和问题**

データベース

クエリ (SELECT句など)
集合演算 関係演算

セキュリティ

暗号解読 (共通鍵暗号、
公開鍵暗号、ハッシュ関数)

通信

信号検出
ルーティング

機械学習

SVM 最近傍法
Neural Network クラスタリング
強化学習 k-means

数学

多変数多項式
積分

文字列学

文字列照合
共通文字列

- 参考：Quantum algorithm and experimental demonstration for the subset sum problem, 2022

部分和問題

入力： n 個の自然数からなる集合 $S = \{s_1, s_2, \dots, s_n\}$ 、ターゲット自然数 t

出力： $\sum_{i=1}^I s_i = t$ を満たす整数集合の部分集合 $I \in \{1, 2, \dots, n\}$

■ 例：

i	1	2	3	4
s_i	11	20	36	52

$t = 67$

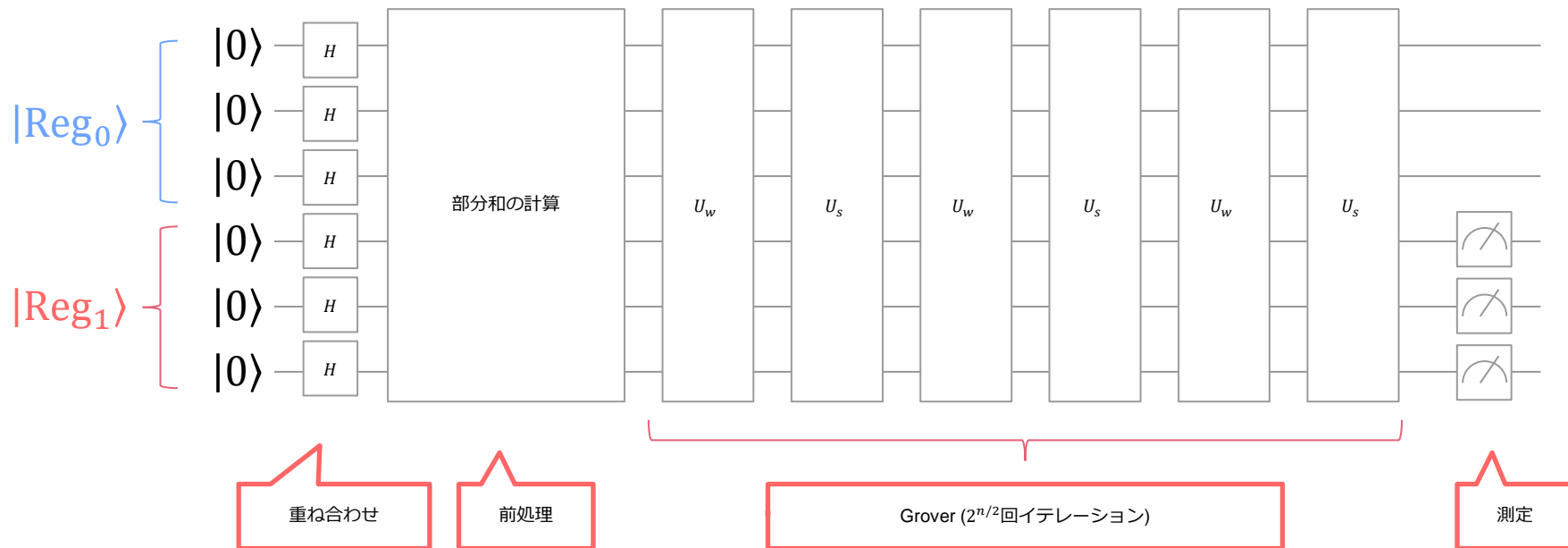
解：{1,2,3} ($11 + 20 + 36 = 67$)



解をビット列で表現 (S の各要素を「選ぶ」「選ばない」)

解：1110

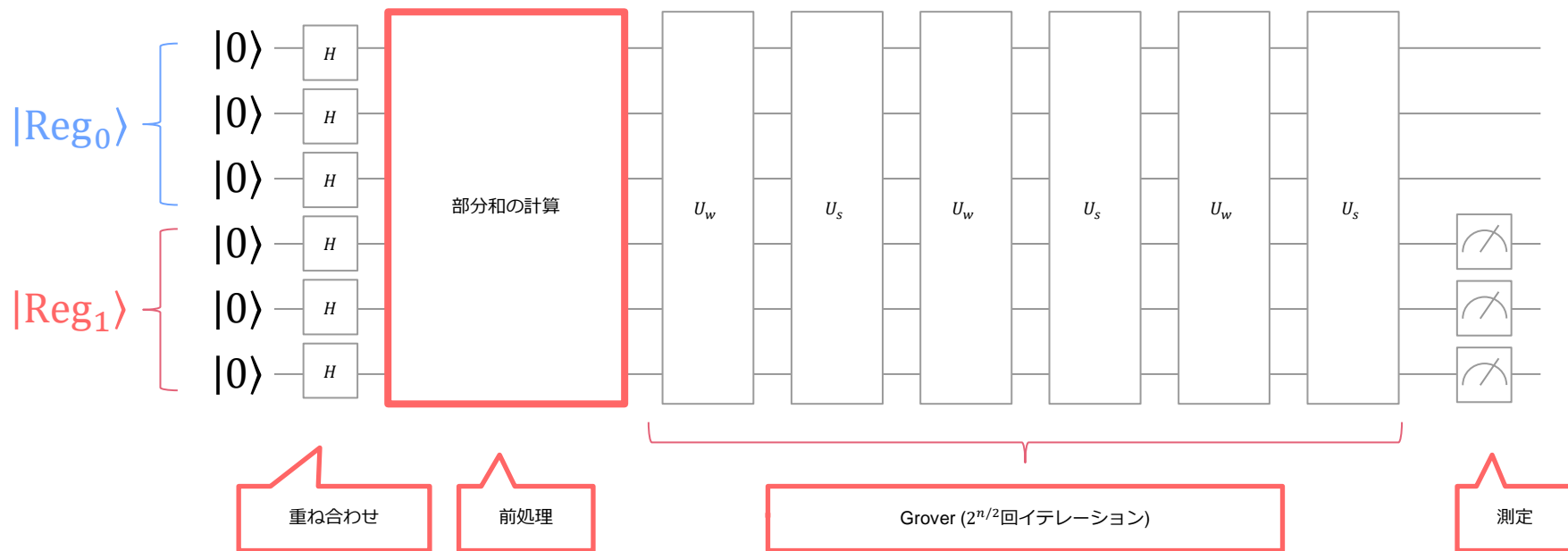
2^n 通りの組み合わせ



■ $|\text{Reg}_0\rangle$: $|\text{Reg}_1\rangle$ に対応する**部分和とターゲット t との差**を格納する n 量子ビットのレジスタ

■ $|\text{Reg}_1\rangle$: 整数集合 S の取り方の**組み合わせ**を格納する n 量子ビットのレジスタ

※実際は量子位相推定(QPE)というものを使っているので、厳密な回路の形は異なります



- $|\text{Reg}_0\rangle$: $|\text{Reg}_1\rangle$ に対応する**部分和とターゲット t との差**を格納する n 量子ビットのレジスタ
- $|\text{Reg}_1\rangle$: 整数集合 S の取り方の**組み合わせ**を格納する n 量子ビットのレジスタ

※実際は量子位相推定(QPE)というものを使っているので、厳密な回路の形は異なります

- $|\text{Reg}_0\rangle$ には組合せの**部分和とターゲット t の差**が格納
- 部分和の計算・ $|\text{Reg}_1\rangle$ と $|\text{Reg}_0\rangle$ の対応付けには量子位相推定 (QPE) が使用される
 - S の各要素は**角度情報(位相)**として R ゲートに格納
 - 部分和は**位相の和**としてエンコードされる

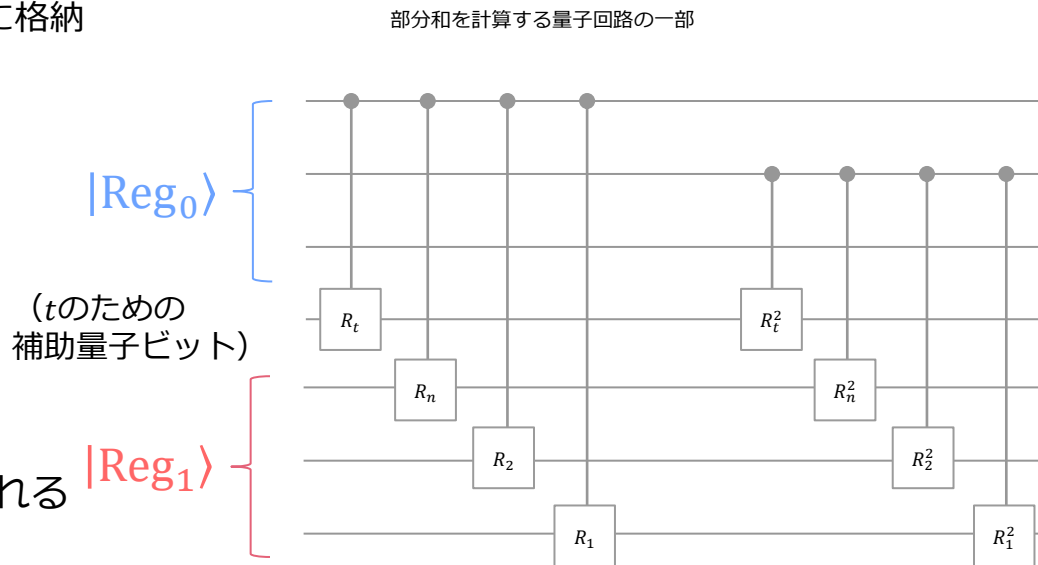
※実際は部分和は $[0,1]$ に正規化され、 0 から 2π までの位相として表現されています。
位相は逆FFTを使うと $|\text{Reg}_0\rangle$ に取り出せます

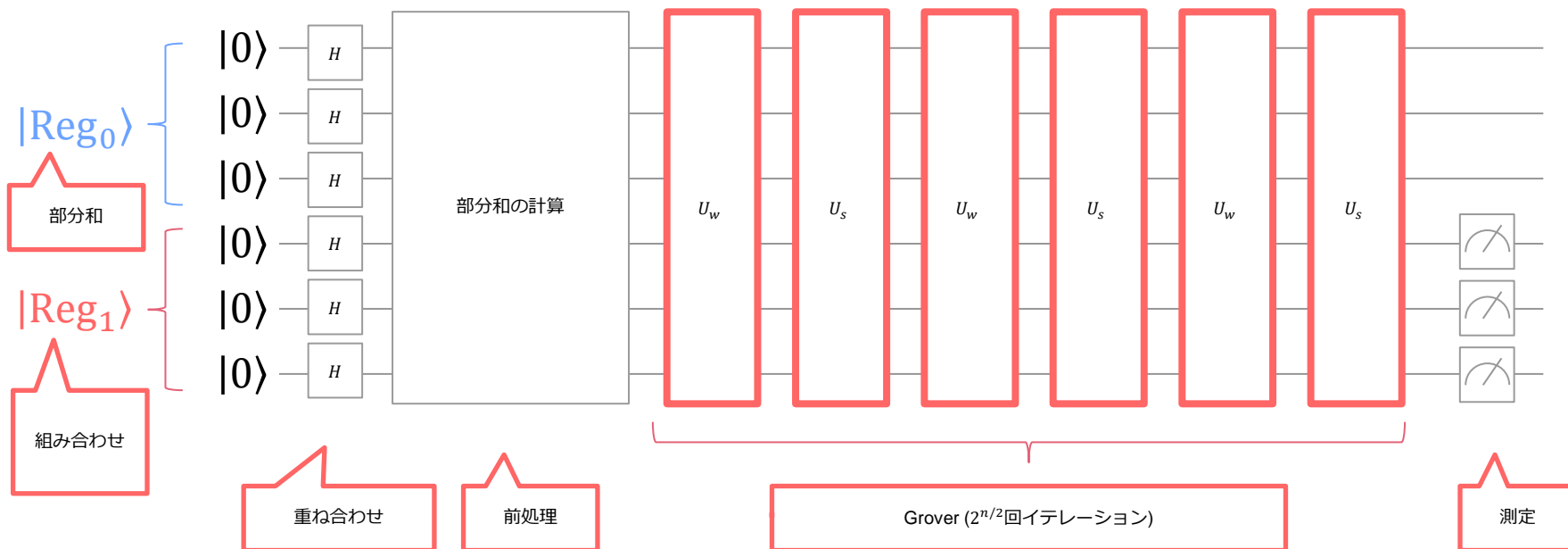
例) $S = \{2,3,5,7\}$, $t = 12$

$|\text{Reg}_1\rangle = |0011\rangle$ のとき、
 $|\text{Reg}_0\rangle = |12 - (5 + 7)\rangle = |0\rangle$ が対応付けられる

→ $|\text{Reg}_0\rangle = |0\rangle$ が解に対応

→ オラクル U_w は、 $|\text{Reg}_0\rangle = |0\rangle$ に対応する $|\text{Reg}_1\rangle$ をマーキング（反転）すればいい





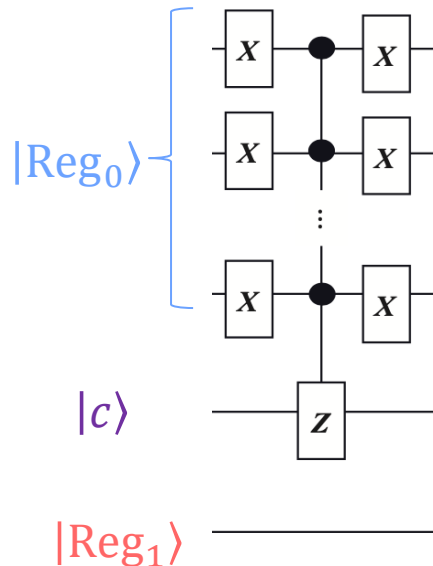
- $|\text{Reg}_0\rangle$: $|\text{Reg}_1\rangle$ に対応する**部分和とターゲット t との差**を格納する n 量子ビットのレジスタ
- $|\text{Reg}_1\rangle$: 整数集合 S の取り方の**組み合わせ**を格納する n 量子ビットのレジスタ

- $|\text{Reg}_0\rangle = |0\rangle$ に対応する量子ビット $|\text{Reg}_1\rangle$ をマーキング（位相を反転）するオラクル U_w
- 実際は1補助量子ビット $|c\rangle$ （ $|\text{Reg}_1\rangle|\text{Reg}_0\rangle$ の符号に相当）の符号を反転する
- $|\text{Reg}_1\rangle|1\rangle|0\rangle \rightarrow -|\text{Reg}_1\rangle|1\rangle|0\rangle$ とする

- 量子回路上はControlled-Zを使ってこんな感じになります→

- $|\text{Reg}_1\rangle|1\rangle|0\rangle$ を増幅させる行列 U_s
 - 中身は省略しますが、結果として振幅は以下のように増幅します
（ $|w\rangle$ ：解となるビット列、 $|\hat{s}\rangle$ ：それ以外、 $|\phi\rangle$ ：0以外の角度）

$$\sin((2j+1)\theta) |\text{w}\rangle|1\rangle|0\rangle + \cos((2j+1)\theta) |\hat{s}\rangle|1\rangle|\phi\rangle$$



- 量子探索アルゴリズムであるGroverのアルゴリズムを紹介しました
- Groverのアルゴリズムを使うと n ビットの探索を $O(2^{n/2})$ で求解できます
- 様々な研究領域でGroverのアルゴリズムが活用されています
- 応用例として、部分和問題（暗号解読等に関連が深い）を紹介しました

- 衝突問題 : Quantum algorithms for element distinctness
- 彩色問題 : Improved Complexity of Quantum Oracles for Ternary Grover Algorithm for Graph Coloring
- SAT : An Introduction to Quantum Computing, Without the Physics
- 部分和問題 : Quantum algorithms for subset finding
- クエリ : Quantum databases: Trends and challenges
- 集合演算 : Quantum Search Algorithm for Set Operation
- 関係演算 : Quantum Computation and Its Effects in Database Systems
- SVM : Quantum optimization for training support vector machines
- Neural Network : Quantum artificial neural network architectures and components
- クラスタリング : Quantum clustering algorithms
- 強化学習 : Quantum reinforcement learning
- 最近傍法 : Quantum algorithms for nearest neighbor methods for supervised and unsupervised learning
- k-means : Quantum algorithms for supervised and unsupervised machine learning
- 信号検出 : A Quantum Search Based Signal Detection for MIMO-OFDM Systems
- ルーティング : A modified ant colony optimization algorithm (mACO) for energy efficient wireless sensor networks
- 共通鍵暗号 : Introduction to Post-Quantum Cryptography
- 公開鍵暗号 (符号暗号) : A Complete Quantum Circuit to Solve the Information Set Decoding Problem
- ハッシュ関数 : Quantum Cryptanalysis of Hash and Claw-Free Functions
- 多変数多項式 : Solving Binary MQ with Grover's Algorithm
- 積分 : Practical numerical integration on NISQ devices
- 文字列照合 : String matching in $O(n+m)$ quantum time
- 共通文字列 : Near-Optimal Quantum Algorithms for String Problems

■ データ工学における量子コンピューティングを紹介

- 量子コンピューティング概要
- 量子機械学習アルゴリズム：QSVM、QCL
- 量子組合せ最適化アルゴリズム：量子アニーリング等イジングマシン、QAOA
- 量子探索アルゴリズム：Groverアルゴリズム、部分和問題

■ 量子コンピュータのユースケース

	最適化&探索問題	シミュレーション&モデリング	AI&機械学習
概要	問題の規模に応じて 計算量が指数関数的に膨大 となる問題	物理学の基本方程式（力学／電磁気学等）のように、 微分積分などの計算量が膨大な数値計算 を含む問題	ディープラーニングや強化学習などの 計算量の多い機械学習手法
応用例	<ul style="list-style-type: none">・ 信号機制御による渋滞解消・ 資産ポートフォリオ最適化・ タンパク質設計最適化・ 暗号解読	<ul style="list-style-type: none">・ 電磁界シミュレーション・ 触媒反応シミュレーション・ 有機分子の結晶構造分析・ エンジンの熱伝達分析	<ul style="list-style-type: none">・ クラスタリングによる異常検知・ 時系列分析・ 次元圧縮・ 画像と音声の分類

まだ見ぬ効果的なユースケースがデータ工学分野にもあるはず！

