

# 2024-B-06 量子コンピュータ の活用研究

IBM COMMUNITY JAPAN 2024ナレッジモール研究

# 【チーム紹介】

- ・大きく2テーマでチームを編成

テーマ	所属	氏名	備考
QMLチーム  (量子コンピュータの説明+量子機械学習)	ソフトバンク株式会社	木南 雅彦	リーダー
	株式会社エクサ	川口 凌平	
	横河デジタル株式会社	松島 輝昌	
	日本IBM Systems Engineering株式会社	河村 崇弘	
	日本IBM Systems Engineering株式会社	Tam·Ka Yeung Adrian	
	日本IBMデジタルサービス株式会社	齊部 和樹	
QCAチーム  (量子セルオートマトン+量子ウォーク)	株式会社アークシステム	遠藤 尚宏	サブリーダー
	日本IBM株式会社	町田 剛	
	日本IBM株式会社	柏 知	
研究サポート	日本IBM株式会社	沼田 祢史	アドバイザ
	日本IBM株式会社	水谷 寿介	アドバイザ

# 【目次】

- 1) 量子コンピュータとは
- 2) Qiskitとは & Qiskit1.0化
- 3) 研究テーマ
  - ① 量子機械学習
  - ② 量子セルオートマトン + 量子ウォーク

アルゴリズム 数理モデル	量子機械学習	量子セル オートマトン	量子ウォーク	その他 量子 アルゴリズム
稼働環境	Qiskit (IBM Quantum Computing) version >= 1.0			
構成要素	量子回路 (量子ビット & 量子ゲート) 重ね合わせ & 量子もつれ			
次世代 コンピュータ	量子コンピュータ IBM Quantum 実機 Qiskitシミュレータ			

# 【量子コンピュータとは】

量子が持つ「重ね合わせ」と「エンタングルメント」の性質を利用して、ユースケースによって古典（既存の）コンピュータよりも効率的に問題を解くことができると言われている次世代コンピュータです

## 重ね合わせとは

### ▶ 通常のコンピュータ

- 「ビット」というスイッチを使って「オン(1)」「オフ(0)」で情報を処理します

### ▶ 量子コンピュータ

- 「量子ビット」を使って「オン」「オフ」を同時に持つことができ、これを「重ね合わせ」と言います

0と1を同時に持つことできると効率的な計算ができます。

これは、同時にたくさんの道を探すことができる、と例えることができます。

# 【量子コンピュータとは】

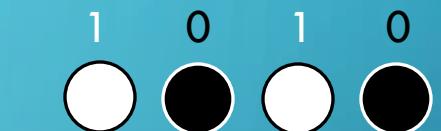
## 量子ビットとは

量子コンピュータの情報単位です

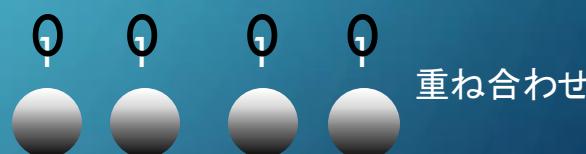
1量子ビット、2量子ビット、n量子ビットなどの表現があります

通常のビット(古典ビット)との違いとして、以下の特徴があります

- n量子ビットは $2^n$ 通りの情報を持つことができる(重ね合わせの原理より)
- 測定で得られる結果は1個のみ
- 結果は確率的に決まるため、毎回測定結果が違う



従来: 1つの状態だけ



重ね合わせ

量子:  $2^4$ 通りの状態

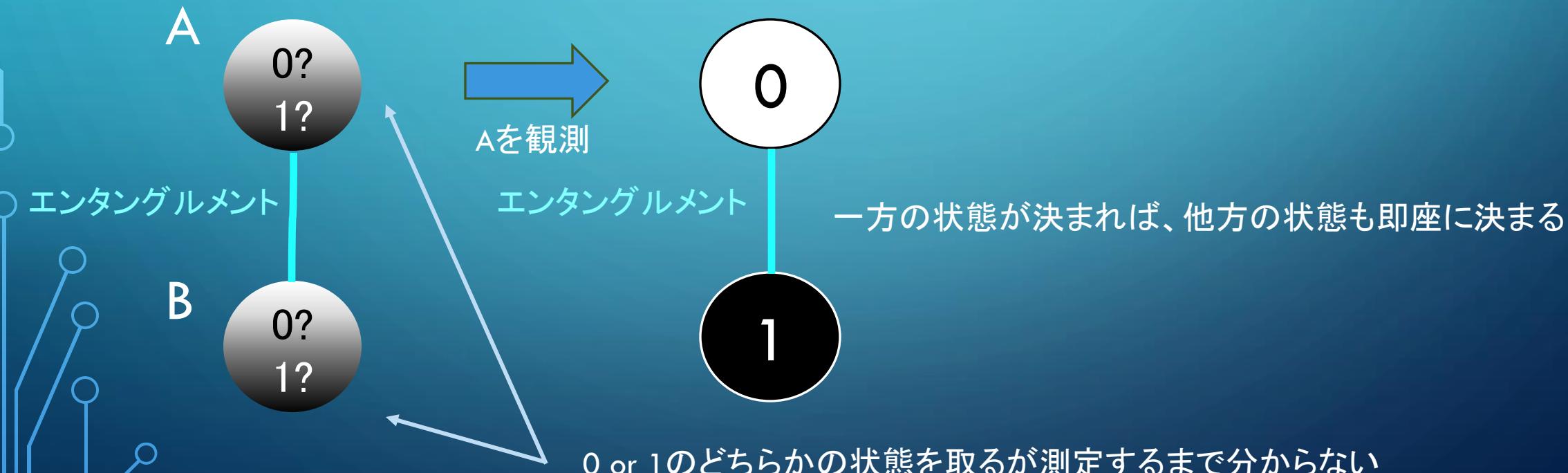
ただし量子力学的な状態を維持することは難しく、エラーの影響を受けやすいため  
エラー訂正の研究が盛んにおこなわれています

# 【量子コンピュータとは】

## 量子もつれ(エンタングルメント)とは

重ね合わせ状態にある量子が二つ以上ある場合、一方の量子の状態が、他の量子の状態に影響を与える相関関係を持つことを指します

これにより複雑な問題を効率的に解くことができます。

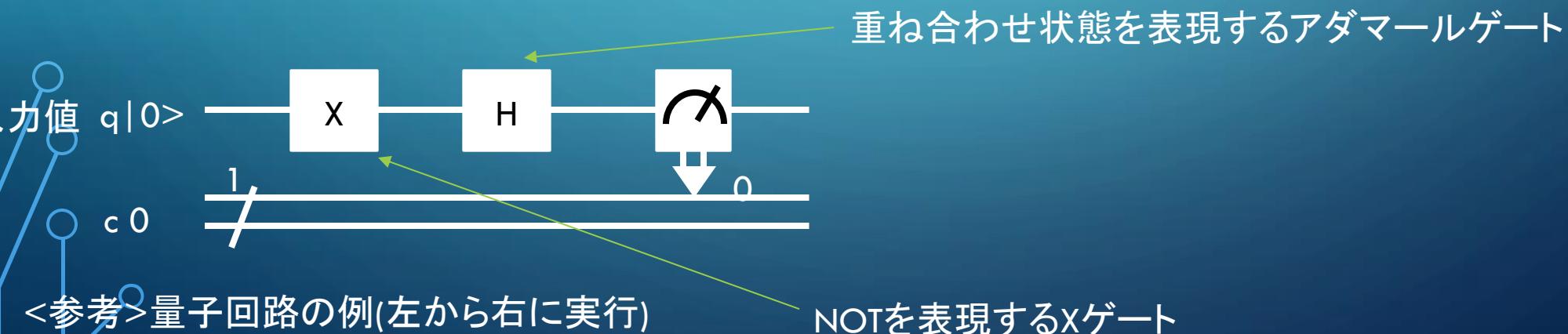


# 【量子コンピュータとは】

## 量子ゲート/量子回路とは

古典コンピュータの世界で計算について説明する際、ブール代数における AND, OR, NOT, XORなどのゲート(素子)を用いて説明ができますが、量子回路も同様です

量子コンピュータの世界では量子ゲートを組み合わせて量子回路を作成します。この量子回路に入力値として量子状態をインプットすると、計算が開始されます



# 【Qiskitとは】

IBMの量子ハードウェア・システム上で量子回路を構築・実行できるようにするオープンソース・ツールです。2017年に提供開始され、60万以上のユーザに使われています

- 2024年5月に**Qiskit1.0**がリリースされ、より高性能化しました
  - QiskitTranspilerServiceによって量子回路を最適化
  - QiskitRuntimeServiceの大幅な高速化
- **旧コードの変更が必要**

輪読図書のコードを修正することで  
qiskit1.0を学習

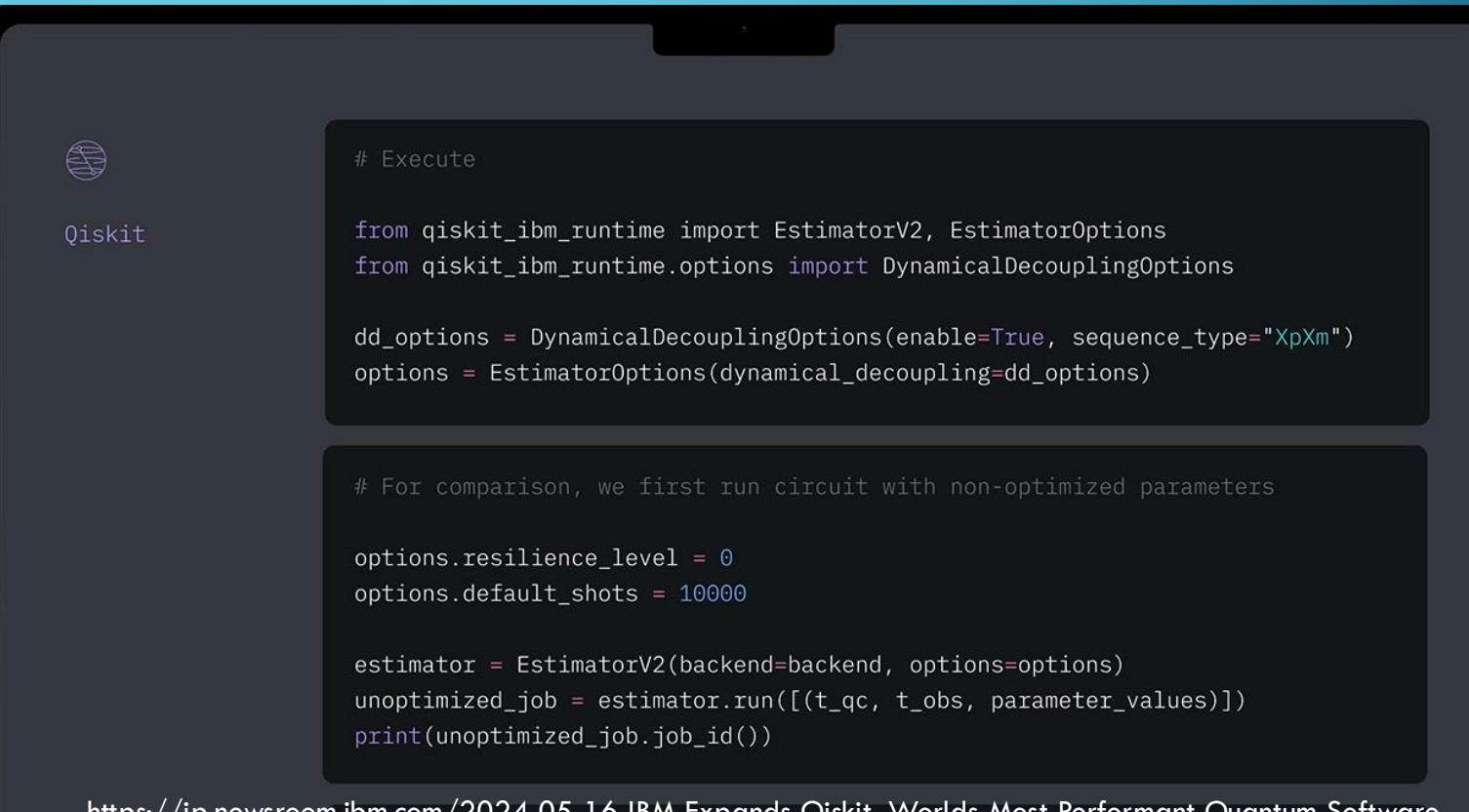
## 【輪読図書】

量子コンピュータの頭の中

—計算しながら理解する量子アルゴリズムの世界

<https://gihyo.jp/book/2023/978-4-297-13511-9>

<https://jp.newsroom.ibm.com/2024-05-16-IBM-Expands-Qiskit,-Worlds-Most-Performant-Quantum-Software>



```
# Execute

from qiskit_ibm_runtime import EstimatorV2, EstimatorOptions
from qiskit_ibm_runtime.options import DynamicalDecouplingOptions

dd_options = DynamicalDecouplingOptions(enable=True, sequence_type="XpXm")
options = EstimatorOptions(dynamical_decoupling=dd_options)

# For comparison, we first run circuit with non-optimized parameters

options.resilience_level = 0
options.default_shots = 10000

estimator = EstimatorV2(backend=backend, options=options)
unoptimized_job = estimator.run([(t_qc, t_obs, parameter_values)])
print(unoptimized_job.job_id())
```

# 【量子を活用するための研究テーマ選択】

グループでのテキスト輪読後、量子の特性を活用するため、専門性の高い

「量子機械学習」、「量子セルオートマトン」をテーマに研究しました。

これらのテーマは、古典で確立されているアルゴリズムを元に、量子の特性を取り込んだものになります。

## 古典アルゴリズム

古典機械学習

古典セルオートマトン

## 量子アルゴリズム

量子機械学習

量子セルオートマトン

# 量子機械学習の活用

QML (Quantum Machine Learning) の解説

QNN (Quantum Neural Network) の解説

VQA (Variational Quantum Algorithm) の解説と使用例

# 量子機械学習とは

## アプローチ

- ・量子データの学習に量子コンピュータを利用する
- ・古典データの学習に量子コンピュータを利用する
  - 従来の機械学習アルゴリズムの高速化

何度も複雑な演算が必要で  
現在実現されているNISQデバイスでは難しい…

- 量子機械学習アルゴリズムの調査/解析

## 古典機械学習を上回る事例は少ない

量子機械学習の精度向上のための調査研究

量子機械学習の事例の探索

## 課題

今回は量子ニューラルネットワークアルゴリズムについて調査研究を行った

# 量子ニューラルネットワークの構成要素

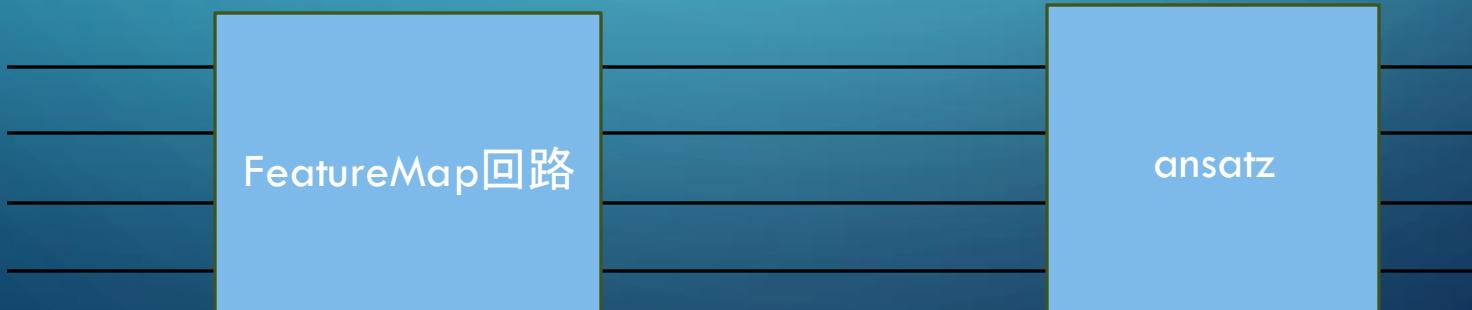
古典データ(特徴量)を  
量子状態(量子特徴量)に  
変換するFeatureMap回路

+

学習によって調整される  
パラメータをもつansatz回路

古典コンピュータでは実現で  
きない特徴量を利用可能

変分量子アルゴリズム



# 変分量子アルゴリズムとは

シュレディンガ一方程式  $H|\psi\rangle = E|\psi\rangle$  を解くと  
( $H$ : ハミルトニアン,  $E$ : エネルギー)  
 $E_0$  と  $|\psi_0\rangle$ ,  $E_1$  と  $|\psi_1\rangle$ ,  $\dots$ ,  $E_n$  と  $|\psi_n\rangle$  ( $E_0 < E_1 < \dots < E_n$ )

のようにエネルギー固有値とそれに対応する固有ベクトル(量子状態)が求まる

基本的に知りたいのはエネルギーが一番小さい状態(基底状態)である

任意の量子状態  $|\psi(\theta)\rangle$  のエネルギー固有値は基底状態のエネルギーよりも大きくなる

$$\langle \psi(\theta) | H | \psi(\theta) \rangle = E(\theta) \geq E_0$$

パラメータ  $\theta$  を変化させながら、 $\langle \psi(\theta) | H | \psi(\theta) \rangle$  を計算し  
エネルギー固有値が最低となる  $\theta$  を見つける

上記を実現する量子アルゴリズムを 変分量子アルゴリズム とよぶ

機械学習の場合

→ パラメータ  $\theta$  を変化させながら、コスト関数が最低となる  $\theta$  を探す (ansatz)

# 変分量子アルゴリズムとは

具体的には以下のようなプロセスでコスト関数を計算します

1. パラメータ化された量子回路を用意
2. 量子回路を実行しコスト関数の期待値を測定
3. 古典的な最適化オプティマイザーを使って、測定結果をパラメータに更新
4. 上記を繰り返す

コスト関数は問題に応じて適切に定義します

- 量子化学ではハミルトニアンの期待値を最小化
- 組み合わせ最適化では目的関数の期待値を最小化
- 機械学習では損失関数の期待値を最小化

この手順で量子機械学習を具体的なデータに適用しました

# 量子機械学習検証

## 検証ゴール

- 量子機械学習、Variational Quantum Classifier (VQC)、が古典機械学習、Supported Vector Machine (SVM)、の精度を比較する。

## 検証データ

- 今回の検証に使われるデータセットは異常検知のデータセットを3つ利用する[1]。
- それぞれのデータセットはテーブルデータとなっており、対象ラベルが0(正常)か1(異常)の2つがある。

## パラメーター

今回の検証において調整するパラメーター:

- SVM
  - Kernel: [rbf, linear, poly, sigmoid]
- VQC
  - Feature Map: [Z, ZZ, Pauli]
  - Feature Map Repetition: [1, 2, 3, 4, 5]
  - Entanglement: [full, circular, linear, reverse\_linear]

## 検証方法

それぞれのデータセットに対して:

- 元データから正常データを1500件、異常データを500件を、計2000件をランダムに抽出する
- 抽出したデータに対してUMAP[2]を用いて3次元まで次元圧縮を行う
- 学習データ、テストデータを8:2で分割する
- SVMモデルを構築し、評価する
- VQCモデルを構築し、評価する

## 検証指標

今回の検証で利用する評価指標:

- テストデータの全体の正解率(Accuracy)
- テストデータの再現率(Recall)

## 検証環境

今回の検証はローカルマシン上でQiskitを用いた量子コンピュータのシミュレーション上で行った。

- qiskit (1.1.1)
- qiskit-algorithms (0.3.0)
- qiskit-machine-learning (0.7.2)

1. [1911.08623] Deep Anomaly Detection with Deviation Networks (arxiv.org)

2. [1802.03426] UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction (arxiv.org)

# データ一覧

データセット名	概要	異常の定義	データ数 (全体)	異常データ数	次元数
UNSW_NB15_traintest_ba ckdoor [1]	UNSW-NB15から取得した、バックドア攻撃データ。	バックドア攻撃。	95,329	2329 (2.4%)	196
bank-additional- full_normalised [2]	ポルトガルの銀行の電話を使ったダイレクトマーケティングキャンペーンデータ。	成功した事例。	41,188	4640 (11%)	62
KDD2014_donors_10feat_ nomissing_normalised [3]	KDD Cup 2014から取得した、K-12の学校教師によって提案されたプロジェクトの関心度データ。	関心度の高いプロジェクト。	619,326	36710 (5.9%)	10

1. Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Military Communications and Information Systems Conference, 2015. 1–6.
2. <https://archive.ics.uci.edu/dataset/222/bank+marketing>
3. <https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose>

# 古典機械学習 (SVM) の精度評価

\*赤字: 一番精度の高い

	KDD2014_donors_10feat_nomissing_normalised		bank-additional-full_normalised		UNSW_NB15_traintest_backdoor	
Kernel	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
Radial Basis Function (rbf)	<b>0.9125</b>	<b>0.663</b>	<b>0.81</b>	<b>0.34</b>	<b>0.9375</b>	<b>0.825</b>
linear	0.77	0.202	0.76	0	0.8725	0.8058
Polynomial (poly)	0.88	0.558	0.79	0.23	0.8925	0.767
sigmoid	0.5	0	0.64	0.17	0.5425	0

※Recallが0の場合は異常データを全く予測できなかったことを示唆する。

# 量子機械学習 (VQC) の精度評価

\*赤字: 一番精度の高い

feature_map	feature_map_reps	entanglement	KDD2014_donors_10feat_nomissing_normalised		bank-additional-full_normalised		UNSW_NB15_traintest_backdoor	
			Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
Pauli	1	circular	0.8	0.558	0.7575	0.125	0.855	0.757
Pauli	1	full	0.8175	0.462	0.725	0.406	0.795	0.767
Pauli	1	linear	0.7525	0.663	0.7775	0.188	0.7975	0.835
Pauli	1	reverse_linear	0.765	0.635	0.775	0.073	0.7975	0.767
Z map	1	circular	0.765	0.096	0.7425	0.135	0.9125	0.757
Z map	1	full	0.835	0.481	0.7875	0.24	0.905	0.806
Z map	1	linear	0.795	0.308	0.76	0	0.9225	0.738
Z map	1	reverse_linear	0.8325	0.385	0.7875	0.125	0.7	0
ZZ	1	circular	0.8025	0.567	0.7625	0.146	0.865	0.806
ZZ	1	full	0.8325	0.567	0.7425	0.073	0.875	0.767
ZZ	1	linear	0.7775	0.798	0.775	0.365	0.86	0.806
ZZ	1	reverse_linear	0.7925	0.462	0.8	0.188	0.8375	0.767
Pauli	2	circular	0.735	0.154	0.68	0.292	0.8	0.806
Pauli	2	full	0.71	0.067	0.6825	0.552	0.795	0.806
Pauli	2	linear	0.8175	0.442	0.6925	0.333	0.8275	0.767
Pauli	2	reverse_linear	0.7625	0.327	0.69	0.354	0.7775	0.757
Z map	2	circular	0.825	0.327	0.7675	0.083	0.91	0.728
Z map	2	full	0.8	0.26	0.76	0	0.9	0.767
Z map	2	linear	0.8075	0.288	0.7525	0.177	0.9075	0.757
Z map	2	reverse_linear	0.805	0.25	0.765	0.25	0.89	0.767
ZZ	2	circular	0.6975	0.24	0.7075	0.281	0.765	0.757
ZZ	2	full	0.75	0.442	0.725	0.521	0.7875	0.767
ZZ	2	linear	0.6775	0.173	0.7225	0.344	0.7875	0.767
ZZ	2	reverse_linear	0.7775	0.269	0.7	0.344	0.8225	0.806
Pauli	3	full	0.805	0.462	0.7075	0.271	0.825	0.757
Pauli	3	linear	0.7325	0.202	0.7025	0.229	0.8375	0.796
Pauli	3	reverse_linear	0.6775	0.154	0.72	0.25	0.8625	0.767
Pauli	3	circular	0.7625	0.327	0.745	0.365	0.885	0.767
Z map	3	circular	0.74	0	0.7325	0.021	0.7375	0
Z map	3	linear	0.7125	0	0.77	0.115	0.7425	0.039
Z map	3	full	0.74	0	0.785	0.208	0.7425	0.039
Z map	3	reverse_linear	0.74	0	0.7575	0.094	0.7725	0.223
ZZ	3	reverse_linear	0.775	0.682	0.7125	0.188	0.81	0.777
ZZ	3	full	0.72	0.231	0.7	0.25	0.8275	0.757
ZZ	3	circular	0.71	0.202	0.705	0.312	0.86	0.767
ZZ	3	linear	0.745	0.202	0.73	0.25	0.8725	0.757
Pauli	4	circular	0.8275	0.538	0.6	0.323	0.81	0.777
Pauli	4	reverse_linear	0.835	0.471	0.7225	0.219	0.83	0.777
Pauli	4	linear	0.8025	0.519	0.6875	0.167	0.8375	0.786
Pauli	4	full	0.78	0.433	0.645	0.271	0.8625	0.786
Z map	4	circular	0.8325	0.385	0.765	0.385	0.86	0.825
Z map	4	full	0.84	0.413	0.74	0.01	0.87	0.767
Z map	4	linear	0.775	0.25	0.74	0.083	0.87	0.767
Z map	4	reverse_linear	0.8775	0.558	0.77	0.375	0.87	0.767
ZZ	4	reverse_linear	0.79	0.587	0.6525	0.281	0.795	0.796
ZZ	4	full	0.8725	0.587	0.6675	0.312	0.805	0.816
ZZ	4	linear	0.7475	0.548	0.655	0.281	0.8075	0.786
ZZ	4	circular	0.8125	0.635	0.6575	0.219	0.82	0.767
Pauli	5	circular	0.8275	0.538	0.6825	0.146	0.8025	0.845
Pauli	5	full	0.78	0.462	0.6525	0.177	0.765	0.777
Pauli	5	linear	0.78	0.685	0.66	0.198	0.81	0.757
Pauli	5	reverse_linear	0.8675	0.567	0.6825	0.094	0.845	0.796
Z map	5	circular	0.745	0.202	0.7675	0.25	0.82	0.864
Z map	5	full	0.855	0.471	0.7725	0.052	0.8925	0.767
Z map	5	linear	0.7	0.154	0.7575	0.073	0.9075	0.767
Z map	5	reverse_linear	0.7925	0.231	0.7675	0.177	0.8925	0.767
ZZ	5	circular	0.835	0.702	0.6725	0.115	0.79	0.777
ZZ	5	full	0.76	0.317	0.6725	0.167	0.7975	0.777
ZZ	5	linear	0.8625	0.587	0.685	0.115	0.8075	0.845
ZZ	5	reverse_linear	0.74	0.327	0.6575	0.229	0.825	0.767

# 検証結果

検証した結果の最高精度について記載する。

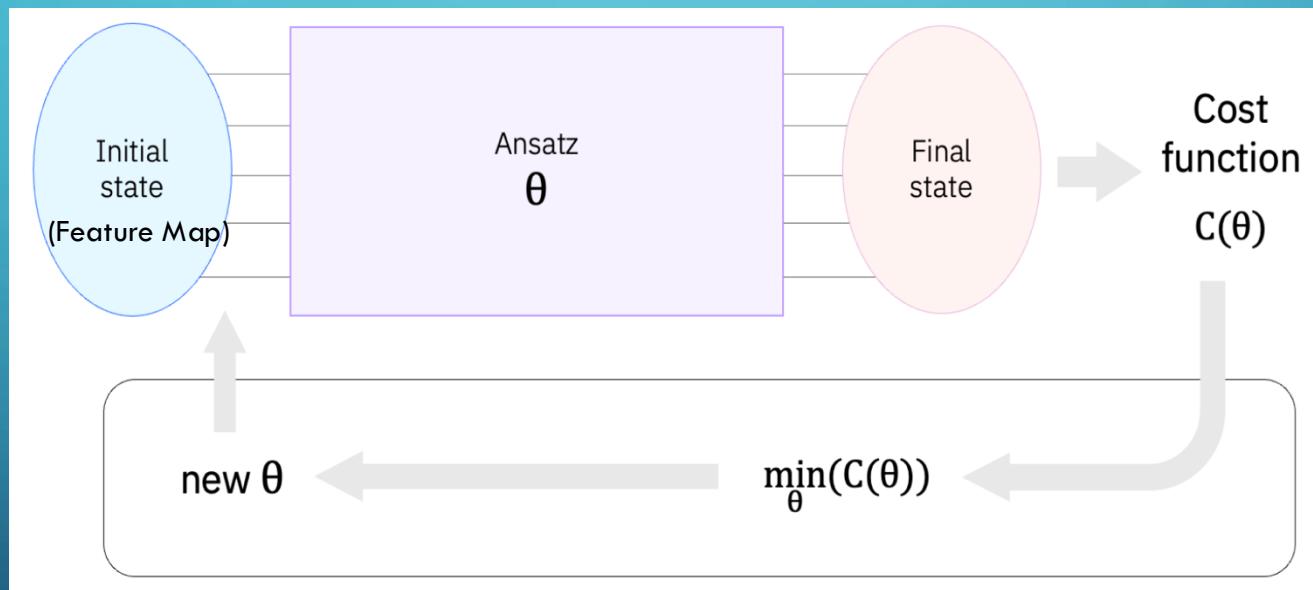
\*赤字: VQCの方が精度が高い

データセット名	古典機械学習(SVM)				量子機械学習(VQC)				比較 [量子 - 古典]	
	Accuracy	Kemel	Recall	Kemel	Accuracy	• Feature map • Repetition • Entanglement	Recall	• Feature map • Repetition • Entanglement	Acuracy	Recall
UNSW_NB15_train-test_backdoor	0.9375	rbf	0.825	rbf	0.9225	• Z • I • linear	0.864	• Z • 5 • circular	-0.015	0.039
bank-additional-full_normalised	0.81	rbf	0.34	rbf	0.8	• ZZ • I • reverse_linear	0.552	• Pauli • 2 • full	-0.01	0.212
KDD2014_donors_10feat_nomissing_normalised	0.9125	rbf	0.663	rbf	0.8775	• Z • 4 • reverse_linear	0.885	• Pauli • 5 • linear	-0.035	0.222

# 考察

1. 全体の精度(Accuracy)においては、量子機械学習(VQC)は古典機械学習(SVM)には及ばないが、精度の差は最大4%以下である。したがって、異常検知の分野においてVQCはSVMに近づく性能を持つと言える。
2. 3つのデータセットのうち、2つのデータセットでは、どのFeature Mapにおいても、Repetitionの数値が高いほど精度が下がる傾向が確認できた(最大6%の差)。Feature Mapが繰り返されるたびノイズが入ってしまうと考えられる。
3. どのデータセットにおいてもZ Feature Mapの再現率(Recall)が低いと見受けられる。一方で、PauliとZZ Feature Mapの2つの精度の差は1.2%未満である。
4. どのデータセットにおいてもEntanglementの選択による精度への影響は見られなかった。この原因について詳細な分析を今後実施する。
5. どのデータセットにおいても、VQCの方がRecallの精度が高かった。この原因について詳細な分析を今後実施する。
6. 今回選定したVQCにおけるパラメータは3つまでとなつたが、今後はより多くのパラメータを検証していきたい。

# 量子機械学習 (VQC) による乳癌データの 2値分類（癌の良・悪性分類）



This code is a part of Qiskit  
© Copyright IBM 2017, 2023.

This code is licensed under the Apache License, Version 2.0. You may obtain a copy of this license in the LICENSE.txt file in the root directory of this source tree or at <http://www.apache.org/licenses/LICENSE-2.0>. Any modifications or derivative works of this code must retain this copyright notice, and modified files need to carry a notice indicating that they have been altered from the originals.

[https://github.com/qiskit-community/ibm-quantum-challenge-2024/blob/main/content/lab\\_4/lab-4-ja.ipynb](https://github.com/qiskit-community/ibm-quantum-challenge-2024/blob/main/content/lab_4/lab-4-ja.ipynb)

## 【量子の最高精度】

### 乳癌データによる癌の良性・悪性の2値分類

【データセット】

ウィスコンシン大学の乳癌データ

- ・データ数：569（正常：212、異常：357）

- ・特徴量：3

古典：

97%

量子：

vs 77%

→ 96%

# 【古典vs量子の精度比較】

古典の精度97%に対して量子の精度は、

特微量マップの工夫	量子の精度(%)	独自性	テクノロジー活用度	貢献度・有用性
①チューニング前の精度	77	△	○	×
①ZFeatureMap (チューニング手順：特微量マップ→Ansatz)	93	◎	○	◎
②ZFeatureMap (チューニング手順：Ansatz→特微量マップ)	94	◎	○	◎
③拡張版ZFeatureMap	96	○	○	○
④量子状態の角度に入力データ入力	87	○	○	△

チューニングの工夫

①～④が本研究のトピック

# 【チューニング手順】

特徴量マップとAnsatzは、それぞれにおいてチューニングファクターが複数あったため、次の2通りの方法で効率よくチューニングを行った。

## ①特徴量マップ→Ansatzの順にチューニング

1、特徴量マップをチューニング

- 1-1、特徴量マップのモデルを選択
- 1-2、「1-1、」の次元数（特徴量）を選択
- 1-3、「1-1、」のreps(繰り返し)を選択
- 1-4、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す

2、Ansatzをチューニング

- 2-1、「1、」の最も良いチューニング結果を基にAnsatzのエンタングルメントのモデルを選択
- 2-2、「2-1、」のモデルのrepsを選択
- 2-3、「2-1、」の全モデルに対して「2-1、」～「2-3、」を繰り返す

## ②Ansatz→特徴量マップの順でチューニング

1、Ansatzをチューニング

- 1-1、Ansatzのエンタングルメントのモデルを選択
- 1-1、「1-1、」のモデルのreps(繰り返し)を選択
- 1-3、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す

2、特徴量マップをチューニング

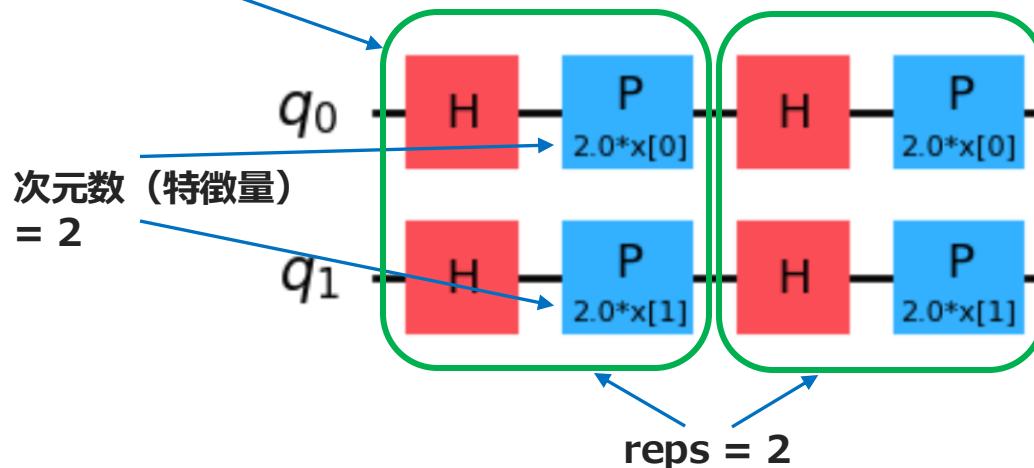
- 2-1、「1、」の最も良いチューニング結果を基に特徴量マップのモデルを選択
- 2-2、「2-1、」の次元数を選択
- 2-3、「2-1、」のrepsを選択
- 2-4、「2-1、」の全モデルに対して「2-1、」～「2-3、」を繰り返す

独自性

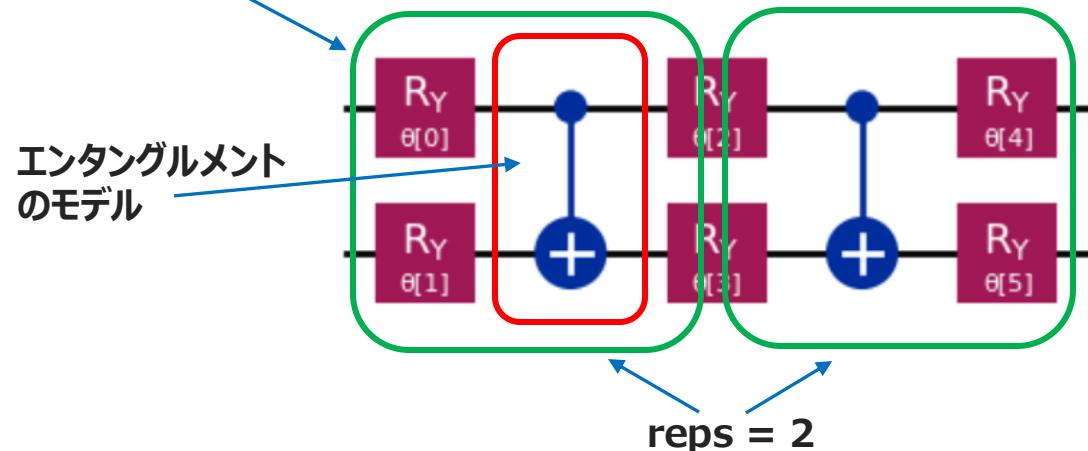
テクノロジー  
活用度

貢献度・  
有用性

### 特徴量マップ (ZFeatureMap)



### Ansatz (RealAmplitudes, entanglement = linear)



# 【チューニング後の古典vs量子の精度比較】

貢献度・有用性

- ①特徴量マップ→Ansatzの順にチューニング
- ②Ansatz→特徴量マップの順にチューニング

古典：

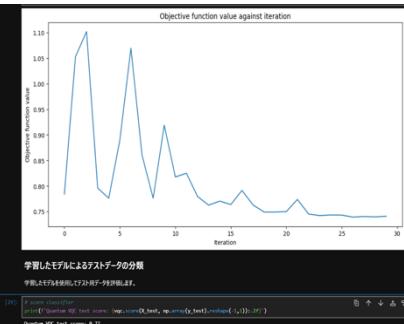
97%

量子：

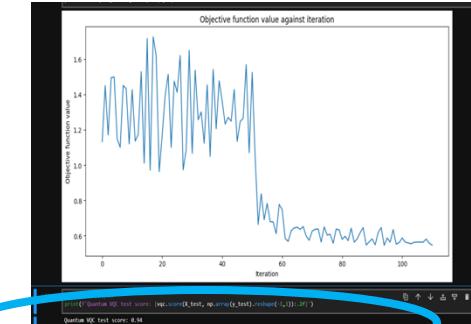
94%

vs

チューニング前



チューニング後



of kernel classification test score: 0.97  
linear kernel classification test score: 0.96  
poly kernel classification test score: 0.96  
sigmoid kernel classification test score: 0.64

量子の精度が77%→94%に向上！！

## 【チューニング結果】

貢献度・有用性

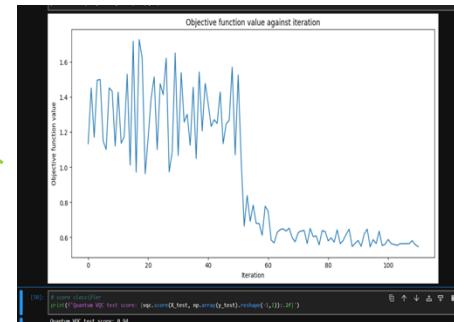
### ①特徴量マップ→Ansatzの順にチューニング

特徴量マップ	Ansatz (エンタングルメントモデル)	精度(%)
ZFeatureMap	linear	0.94
ZZFeatureMap	reverse linear	0.87
PauliFeatureMap	reverse linear	0.87

### ②Ansatz→特徴量マップの順にチューニング

特徴量マップ	Ansatz (エンタングルメントモデル)	精度(%)
ZFeatureMap	sca	0.93
ZZFeatureMap	sca	0.84
PauliFeatureMap	sca	0.84

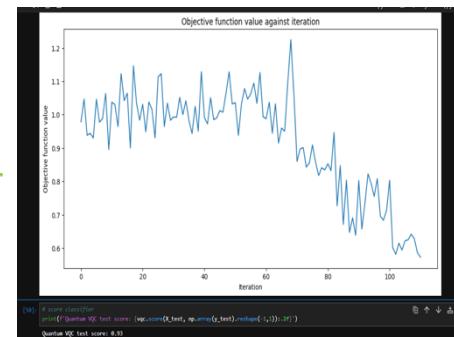
両者のチューニング結果から特徴量マップが「ZFeatureMap」の際に精度が90%台を記録した。



【チューニングモデル】

[特徴量マップ]  
モデル：ZFeatureMap  
次元数：3  
reps：4

[Ansatz]  
モデル：RealAmplitudes  
エンタングルメントのモデル：linear  
reps：3



【チューニングモデル】

[特徴量マップ]  
モデル：ZFeatureMap  
次元数：4  
reps：4

[Ansatz]  
モデル：RealAmplitudes  
エンタングルメントの作成方法：sca  
reps：3

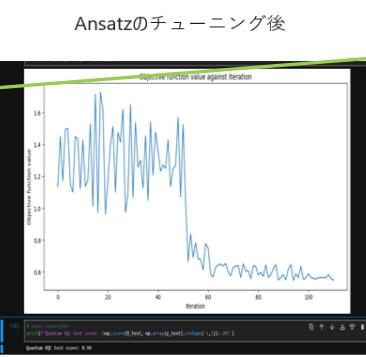
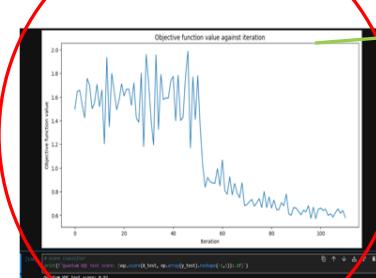
# 【精度向上の仮説】

独自性

貢献度・有用性

特微量マップ → Ansatz

特微量マップのみの  
チューニング結果

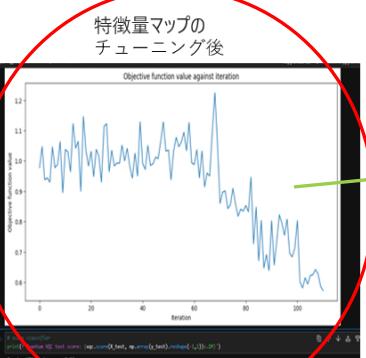
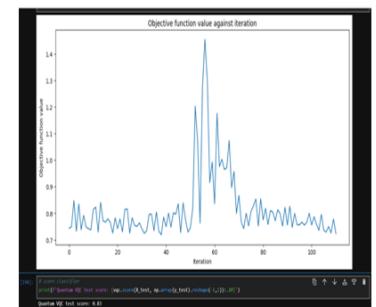


精度が0.93 → 0.94に向上

Ansatzのチューニングによって1%の精度向上しか示さなかった。

Ansatz → 特微量マップ

Ansatzのみの  
チューニング結果

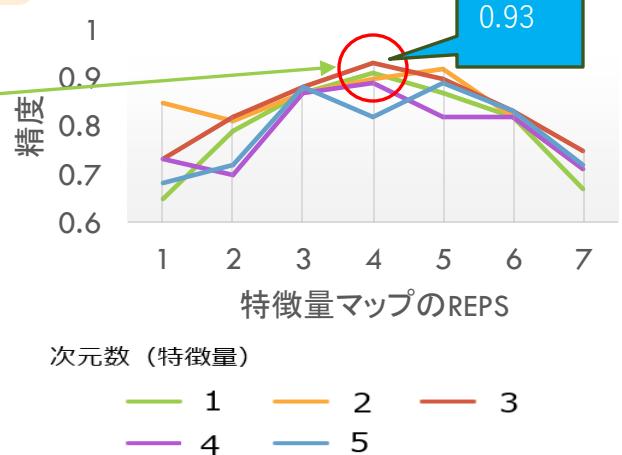


精度が0.83 → 0.93に向上

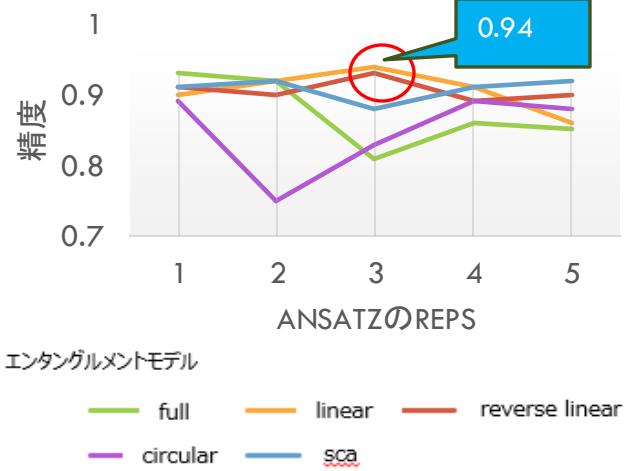
特微量マップのチューニングによって10%の精度向上が示された。

表の精度増加率より、精度向上の要因は  
「特微量マップ = ZFeatureMap」の  
チューニングに依るものと言える！！

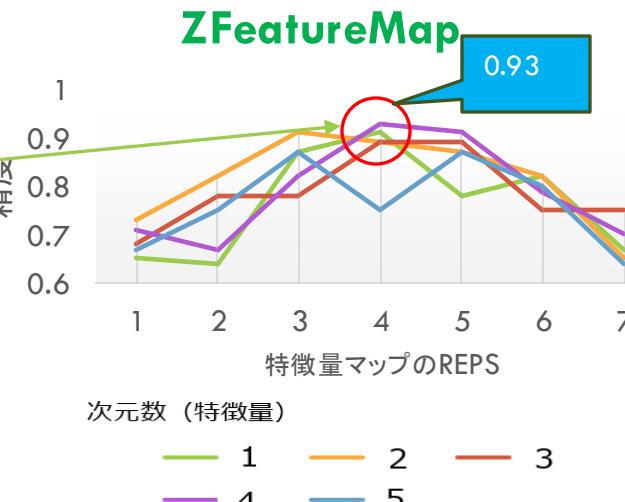
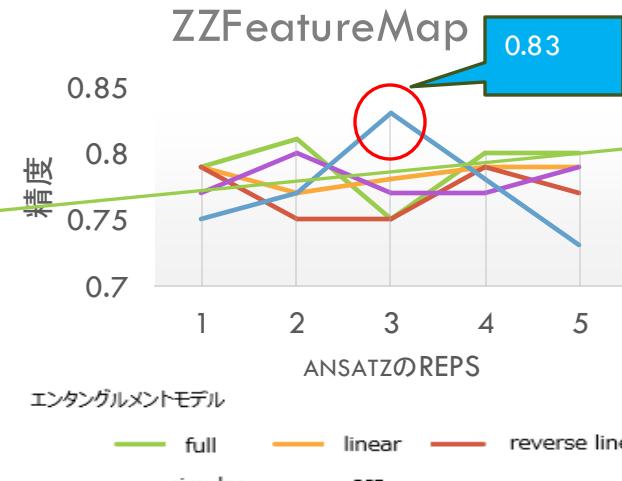
ZFeatureMap



ZFeatureMap



ZFeatureMap



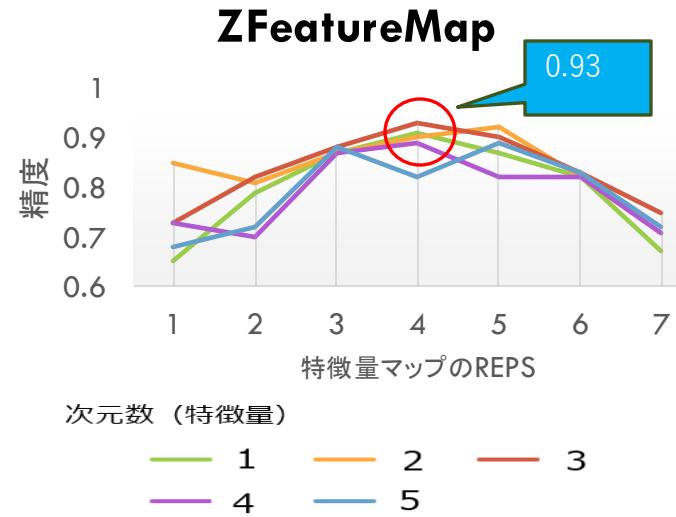
	特微量マップの最高精度の増加率(%)	Ansatzの最高精度の増加率(%)
①特微量マップ → Ansatz	20.77	1.07
②Ansatz → 特微量マップ	12.04	7.79

# 【repsと精度の関係を数式化】仮説検証中

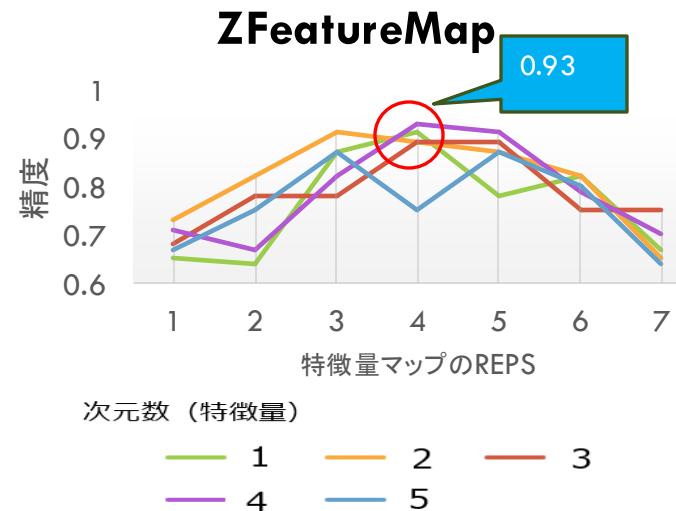
上に凸の放物線の傾向が見られるため特徴量マップ  
(ZFeatureMap)に依存する精度は放物線方程式を示す！！

独自性

①特徴量マップ → Ansatz



②Ansatz → 特徴量マップ



$$\begin{aligned}
 \text{Accuracy} &= \frac{\sum_{i=1}^n I(y_{pred,i} = y_{test,i})}{n} \\
 &= -\alpha(reps - 4)^2 + 0.93(\pm 1) \\
 &= -\alpha(reps^2 - 8reps + 16) + 0.93(\pm 1)
 \end{aligned}$$

インジケーター関数  $I$   
 真  $\Rightarrow I = 1$   
 偽  $\Rightarrow I = 0$   
 頂点の reps の位置  
 頂点  
 変化量の割合 ( $\alpha$  と仮定)  
 (平均変化率)

$$y_{pred} = \langle Z \cdots Z | U(\vec{x})U(\vec{\theta}) \rangle \cong \langle Z \cdots Z | U(\vec{x}) \rangle$$

特徴量マップ  
 Ansatz  
 ZFeatureMap  
 今回のケースでは特徴量マップのみが精度に依存しているため、Ansatzを  $U(\theta) = I$  (恒等演算子) と仮定した。

$$U(\vec{x}) = \left\{ \exp\left(\sum_{j=1}^n x_j Z_j\right) H^{\otimes n} \right\}^{d=reps} \Leftarrow Rz(x) = e^{-i\frac{x}{2}Z} = \begin{pmatrix} e^{-i\frac{x}{2}} & 0 \\ 0 & e^{i\frac{x}{2}} \end{pmatrix}$$

# 最高精度94%の更なる精度向上に挑戦！！

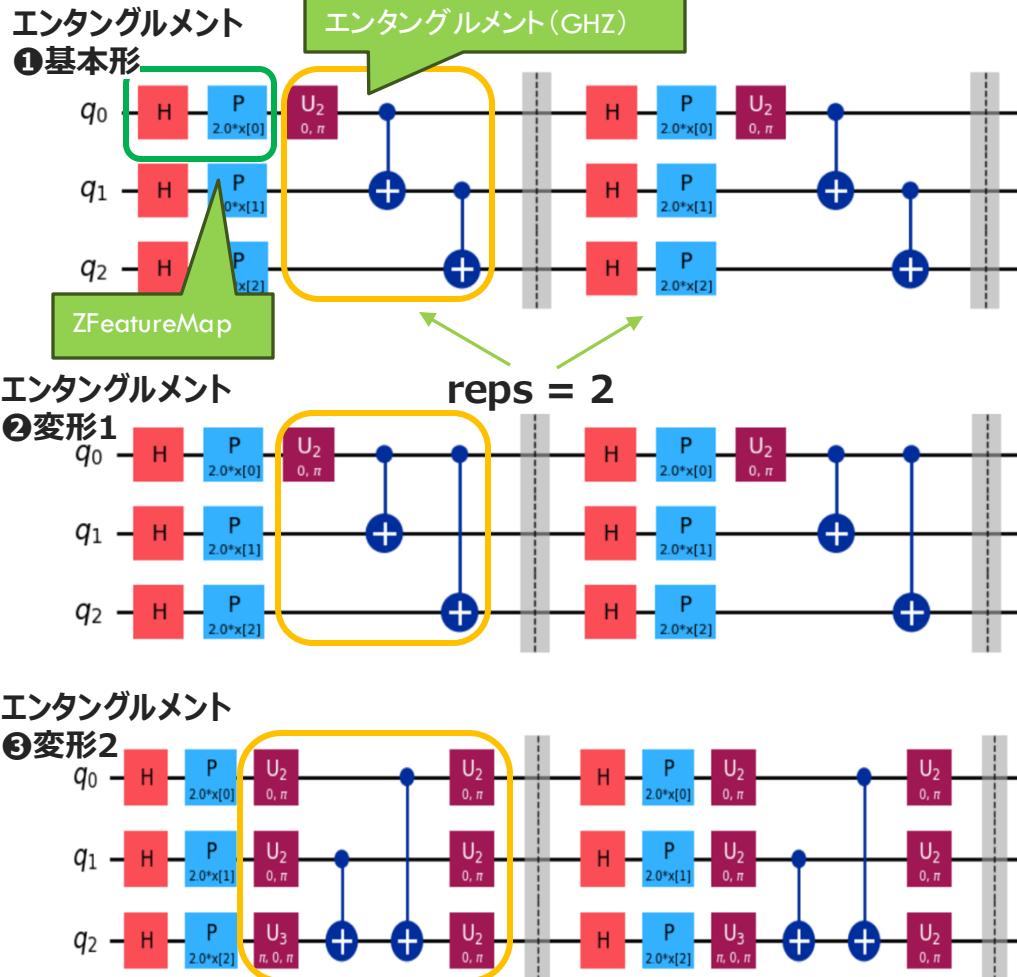
独自性

テクノロジー  
活用度

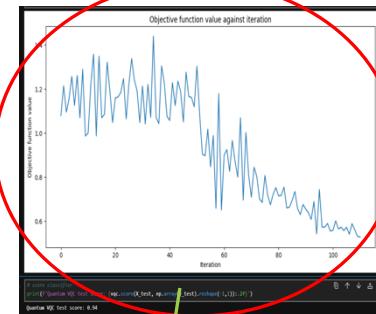
貢献度・  
有用性

## 【③拡張版ZFeatureMap】

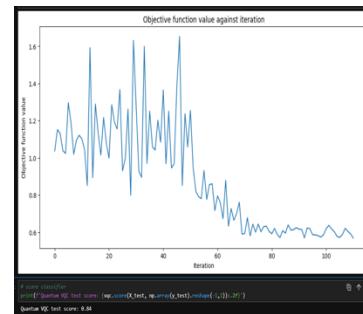
⇒ 先ほどの最高精度94%を記録した「ZFeatureMap」は、1qubitに働く特徴量マップであり、量子特有の複数qubit同士の相関を示すエンタングルメントは実現されていなかった。そのため、複数qubitのZFeatureMapに対してエンタングルメントを実装した。



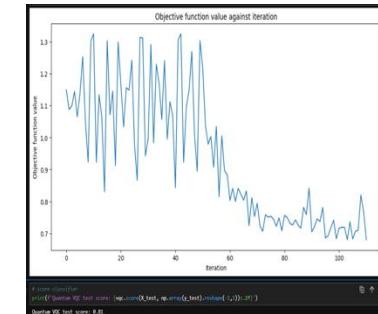
エンタングルメント  
①基本形



エンタングルメント  
②変形1



エンタングルメント  
③変形2



【拡張版ZFeatureMapのチューニングモデル】

[特徴量マップ]

モデル：ZFeatureMap + GHZ(エンタングルメント: 基本形、変形1、変形2)

次元数: 3

reps: 4

[Ansatz]

モデル：RealAmplitudes

エンタングルメントの作成方法：linear

reps: 3

エンタングルメント「①基本形」において精度94%を示し、これまでの最高精度94%と同値となった。

# 【拡張版ZFeatureMap時の古典vs量子の精度比較】

独自性

テクノロジー  
活用度

貢献度・  
有用性

先ほどの拡張版ZFeatureMapにおけるエンタングルメント

「①基本形」で更なる精度向上のためチューニング



下記チューニングモデルで最高精度94%

を上回る、精度**96%**を記録！！

古典：

97%

量子：

96%

vs

```
rbf kernel classification test score: 0.97
linear kernel classification test score: 0.96
poly kernel classification test score: 0.96
sigmoid kernel classification test score: 0.64
```



【拡張版ZFeatureMapのチューニングモデル】

[特微量マップ]

モデル：ZFeatureMap + GHZ（エンタングルメント：基本形）

次元数（特微量）：3

reps : 5

[Ansatz]

モデル：RealAmplitudes

エンタングルメントのモデル：linear, reverse\_linear, sca, circular

reps : 4, 5

## 【④量子状態（ブロッホ球）の角度θに入力データをインプット】

下記量子状態  $Ry(\theta)|\psi\rangle$  の角度θに入力データをインプットし特徴量マップとして扱い量子機械学習（VQC）を行った。  
⇒しかし、精度70~80%台で90%を超えることはなかった。

$$Ry(\theta)|\psi\rangle = e^{-i\frac{\theta}{2}y} \left( \cos \frac{\theta'}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta'}{2} |1\rangle \right) \Leftarrow 0 \leq \theta, \theta' \leq \pi, 0 \leq \varphi \leq 2\pi$$

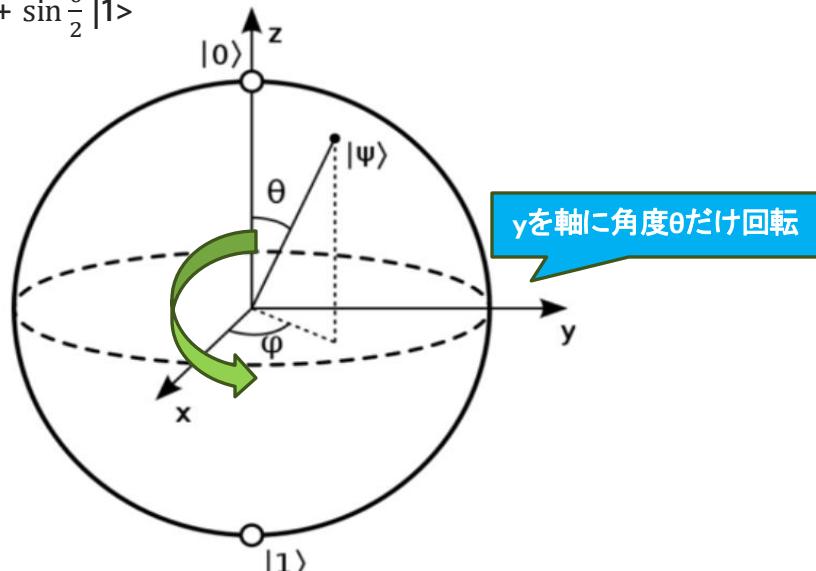
$$= \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix} \begin{pmatrix} \cos \frac{\theta'}{2} \\ e^{i\varphi} \sin \frac{\theta'}{2} \end{pmatrix}$$

$$\Leftarrow Ry(\theta) = e^{-i\frac{\theta}{2}y} = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix}$$

$$Ry(\theta)|0\rangle = \begin{pmatrix} \cos \frac{\theta}{2} \\ \sin \frac{\theta}{2} \end{pmatrix} \Leftarrow \text{qubitの初期値が}|0\rangle\text{の場合}$$

$$e^{i\varphi} = e^{i0} = 1, \theta' = 0$$

$$= \cos \frac{\theta}{2} |0\rangle + \sin \frac{\theta}{2} |1\rangle$$



### 【チューニングモデル】

#### [特徴量マップ]

モデル：・Ryの角度θに入力データをインプット

- ・Ryの角度θに入力データをインプット + GHZ(エンタングルメント: 基本形、変形1、変形2)

次元数: 3

reps: 4

#### [Ansatz]

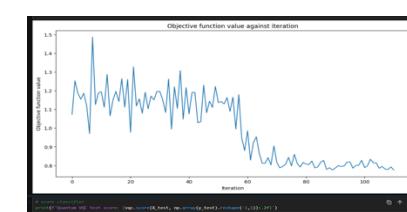
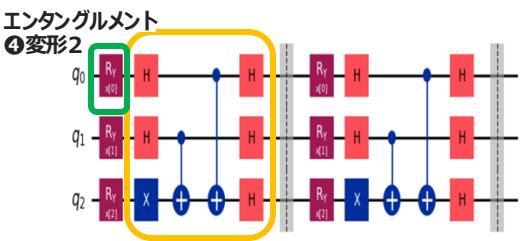
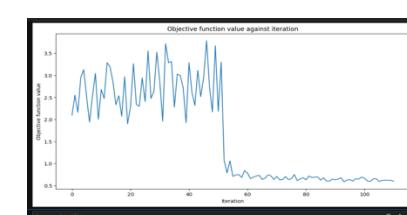
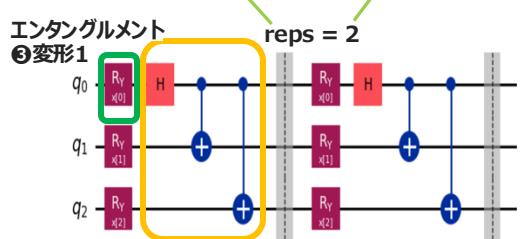
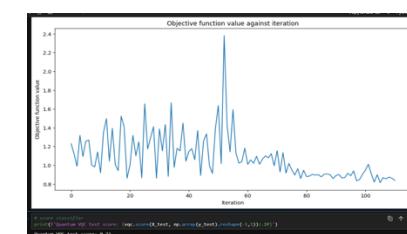
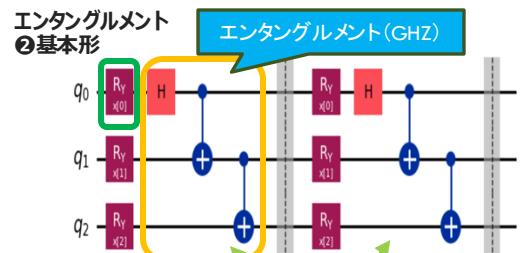
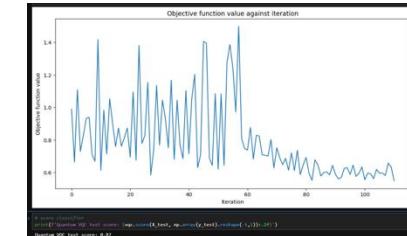
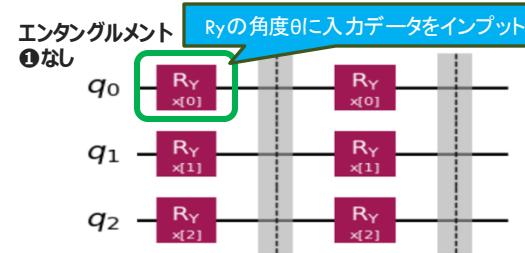
モデル: RealAmplitudes

エンタングルメントの作成方法: linear

reps: 3

独自性

テクノロジー  
活用度



# 量子セルオートマトン / 量子ウォーク

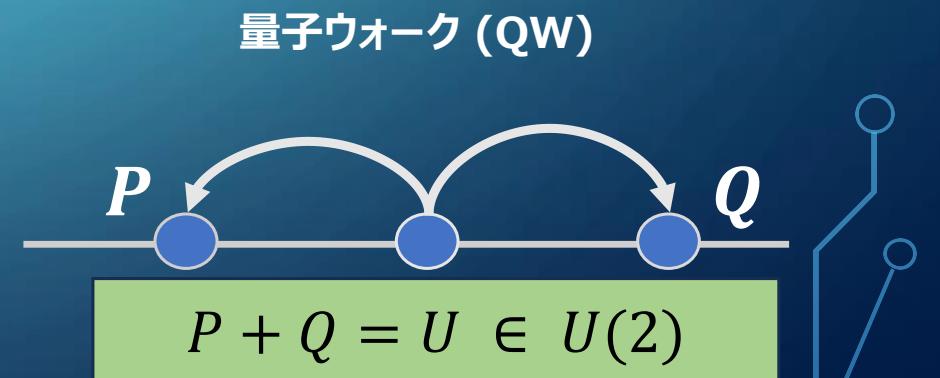
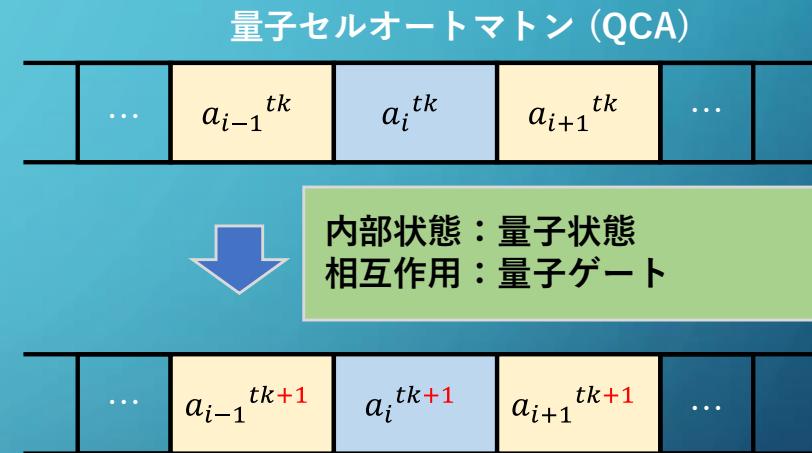
- 量子セルオートマトン(QCA) と 量子ウォーク(QW)
- 量子セルオートマトン(QCA)
- 量子ウォーク(QW)
- 数理モデル間の対応関係
- 実機/シミュレータでの実行結果比較
- APPENDIX (参考文献)

# 【量子セルオートマトンと量子ウォーク】①研究対象

「量子セルオートマトン(QCA)」、および「量子ウォーク(QW)」という2つの数理モデルを研究しました。

「空間が時間発展する」ポイントが共通しています。

アルゴリズム 数理モデル	量子機械学習	量子セル オートマトン	量子ウォーク	その他 量子 アルゴリズム
稼働環境	Qiskit (IBM Quantum Computing) version >= 1.0			
構成要素	量子回路 (量子ビット & 量子ゲート) 重ね合わせ & 量子もつれ			
次世代 コンピュータ	量子コンピュータ IBM Quantum 実機 Qiskitシミュレータ			



# 【量子セルオートマトンと量子ウォーク】②研究動向

## ★(1) 概要

どちらの数理モデルも量子コンピュータ研究の中ではメインストリームの研究ではなく、

専門的に特化した先進的な領域に位置付けられています。

このため、初学者が参入するための間口が狭い状態と言えます。

## ★(2) 量子セルオートマトン

理論研究が中心。大規模な量子ビット数が必要となるため、実装するための技術が確立していません。

量子を効果的に利用するためのアルゴリズムについて、理論研究の段階で開発中となっています。

## ★(3) 量子ウォーク

2000年頃より本格的に研究され始めた新しい数理モデル。

量子コンピュータのアルゴリズム開発において重要な位置を占めているものの、

量子コンピュータの領域の中であっても、モデルを理解するのが難しく、一般的には普及していません。 [1]

# 【量子セルオートマトンと量子ウォーク】③研究成果 - 1

## ★(1)量子セルオートマトン

(1-1) Qiskit1.0で初めて「1次元量子セルオートマトン」、「2次元量子セルオートマトン」を実装。

確率振幅の伝播をわかりやすくグラフィカルに表示。

独自性

テクノロジー活用度

貢献度・有用性

(1-2) 行列積(matrix product state)を使用して、量子ビット数を拡張

独自性

テクノロジー活用度

貢献度・有用性

(1-3) 2次元量子セルオートマトンでは、2次元量子ライフゲームを実装。

・先行事例[2][3][4]では計算の途中に古典計算を挟んでいるが、本研究ではこれを改善し、

量子の性質を最後の測定の直前まで維持した「2次元量子ライフゲーム」を実装。

・量子ライフゲーム特有の12パターンをアニメーションで表示 [5]

独自性

テクノロジー活用度

貢献度・有用性

# 【量子セルオートマトンと量子ウォーク】③研究成果 - 2

## ★(2)量子ウォーク

(2-1) Qiskit1.0で「1次元2状態アダマールウォーク」を実装。

貢献度・有用性

(2-2) 「1次元2状態アダマールウォークでは」、確率振幅の伝わり方をわかりやすく表示。

先行事例<sup>[6]</sup>では1～2ステップの実行しかないが、本研究では130ステップ程度の出力を行ったり。

Qiskitのビットオーダーを変換することにより、量子ウォークの挙動の全容を一目でわかるようにした。

貢献度・有用性

(2-3) 発展課題として、量子検索用の教材のコードを修正してQiskit1.0で実装。

貢献度・有用性

## ★(3) 量子セルオートマトン/古典セルオートマトン/量子ウォーク/ランダムウォーク

(3-1) 各モデル間の対応関係をまとめた。

独自性

貢献度・有用性

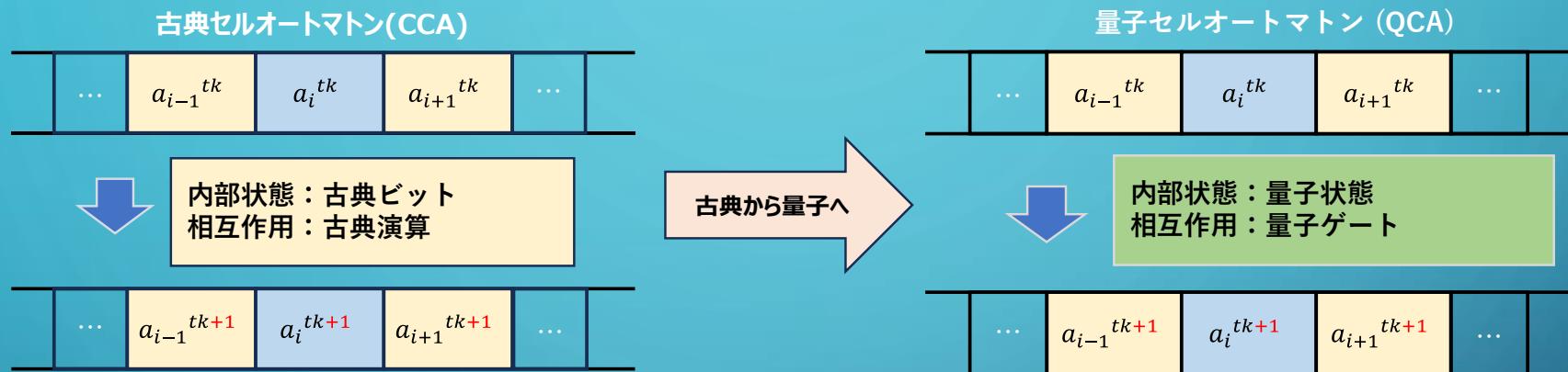
(3-2) 実機/シミュレータの実行比較を行った。

貢献度・有用性

# 【量子セルオートマトン】①量子セルオートマトンとは

(1) 「古典セルオートマトン(CCA)」という数理モデルをベースにして

量子の性質を加えて拡張したモデルが「量子セルオートマトン(QCA)」です。[18]



(2) 量子セルオートマトン(QCA)の有用性および将来性

- ・量子誤り訂正

「原理的には大規模量子システムに対しても適用可能」なため、量子誤り訂正の手法として可能性が追及されています。[20]

- ・量子格子ガスセルオートマトン (Quantum Lattice Gas Cellular Automata)

量子力学系や流体现象をシミュレートするための有望な手段として、量子格子ガスセルオートマトンが基礎段階から研究されています。[19] [21]

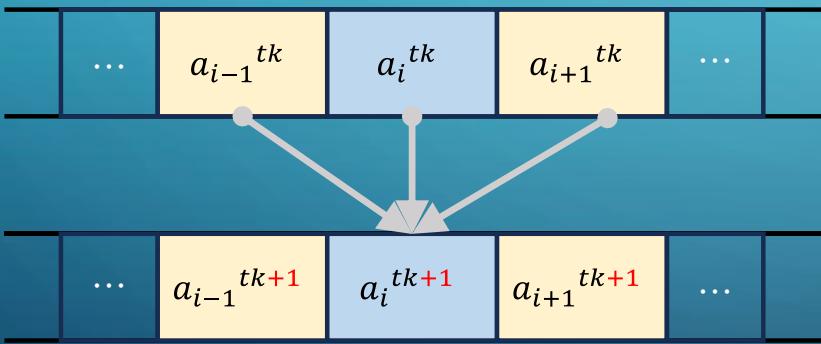
## 【量子セルオートマトン】②セルオートマトンの仕組み

### (1) セルオートマトンの仕組み(古典/量子共通)

- (a) 空間を“セル”という単位で分割します。
- (b) セルはそれぞれ内部状態を持っています。(例えば、0や1、ONやOFFなどを使用します。)
- (c) 他のセルとの相互作用によって、時間的に内部状態が変化します。[\[7\]](#) [\[8\]](#) [\[9\]](#) [\[10\]](#)

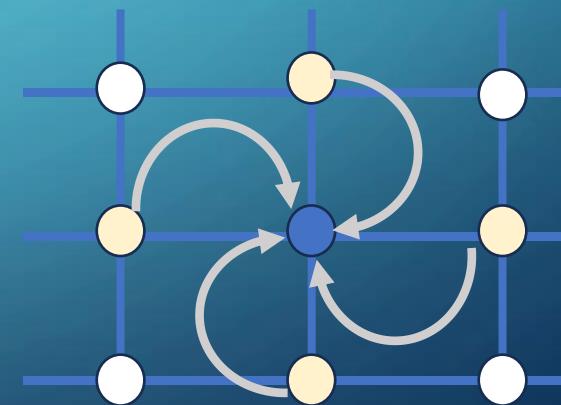
「前世代セルの内部状態」と「ルール」により、「次世代セルの内部状態」が決定されます。

1次元セルオートマトンの例



1次元の場合、前世代の3セルが“ON,ON,OFF”的場合、次世代の中央セルを“ON”にするなどのルールがあります。

2次元セルオートマトンの例



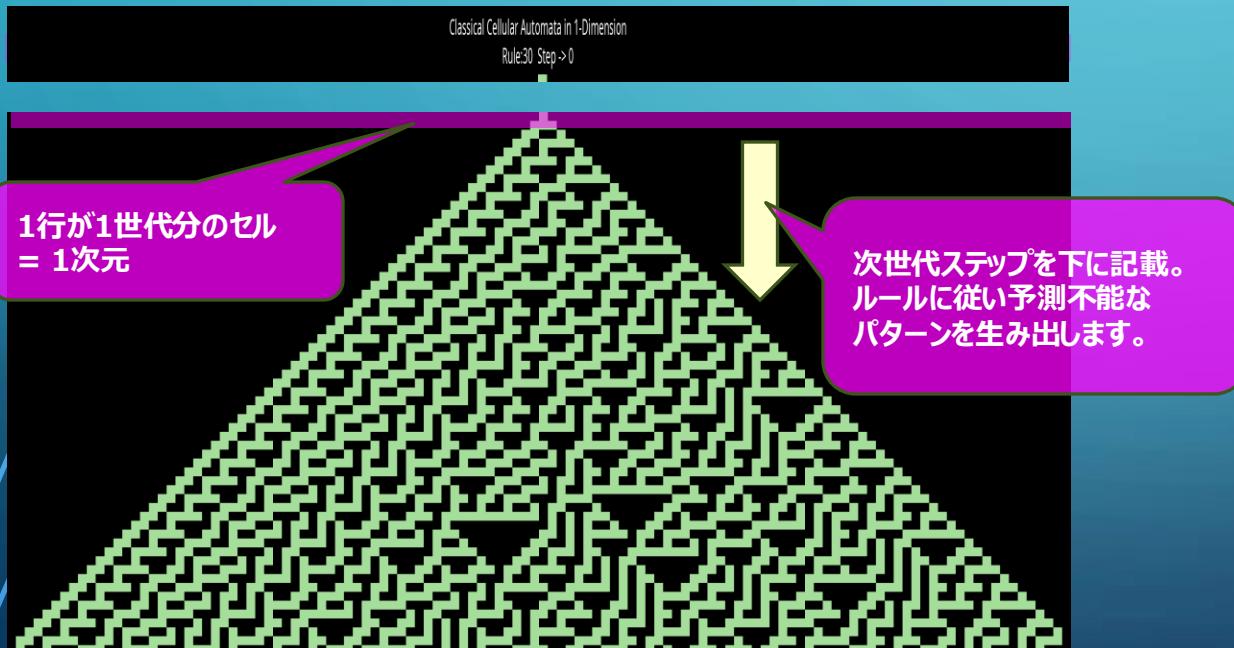
2次元の場合、前世代周囲のセルに“ON”が3つあった場合、次世代の中央セルを“ON”にするなどのルールがあります。

# 【量子セルオートマトン】③古典セルオートマトン

## (1) 古典セルオートマトンの特徴

- ・セルの内部状態と相互作用が古典状態。(重ね合わせ状態やもつれ状態は無し。)
- ・シンプルなルールと、シンプルな初期状態の組み合わせにもかかわらず、予測不能なパターンを生み出します。
- ・自然現象や都市計画、生物学でのシミュレーションで活用されています。
- ・古典セルオートマトンの一部は、「チューリング完全」の条件を満たしており、古典コンピュータと同等の計算力を持っています。 [32] [33]

### 1次元古典セルオートマトン（ルール30の場合）



ルール30：前世代の3つのセルの並びが“100”, “001”的な場合のみ、次世代の中央セルが“ON”となる。  
その他の場合は、次世代の中央セルが“OFF”となる。

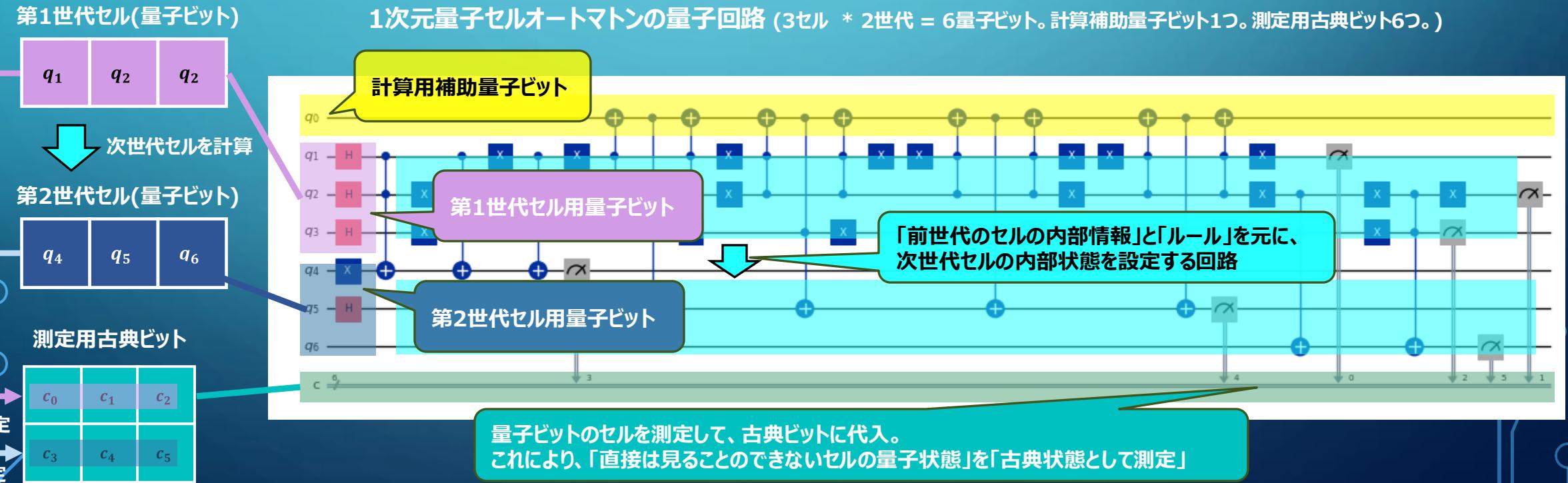
### 2次元古典セルオートマトン（ライフゲームの場合）



# 【量子セルオートマトン】④1次元量子セルオートマトン - 1

## (1) 1次元量子セルオートマトンをQiskit1.0で実装

- (a) セルの内部状態を“量子ビット”で表現します。セルが量子状態を保持することができます。
- (b) セル間の相互作用で“ユニタリゲート”を使用します。計算前後で量子状態を保持することができます。
- (c) 下記の図は「1次元量子セルオートマトン」の量子回路です。[11]

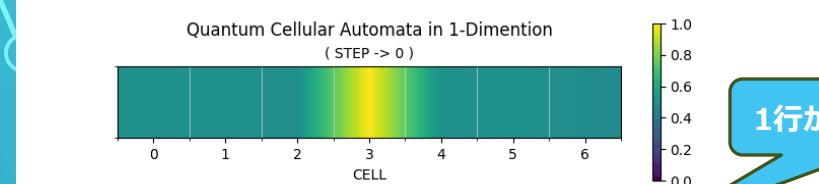


# 【量子セルオートマトン】④1次元量子セルオートマトン - 2

使用量子ビット数 : 57  
量子回路の深さ : 667

同じルール122を使用しているが、出力されるパターンが異なる。  
ルール122：前世代の3つセルの並びが“110”, “101”, “100”, “011”, “001”的場合のみ、  
次世代の中央セルが“ON”となり。これ以外の場合は、次世代の中央セルが“OFF”となる。

## 1次元量子セルオートマトン (ルール122の場合)



1行が1世代分のセル

1行が1世代分のセル

初期状態に重ね合わせを使用

★古典と同一の「ルール」を使用。

★古典状態では市松模様にしかなりませんが、  
量子ビットに重ね合わせをセットすると、  
複数の可能性が同時に存在していることにより、  
干渉パターンのような濃淡のあるパターンが  
出現します。

★確率振幅の波が各セルに伝わっている様子が  
わかります。

## 1次元古典セルオートマトン (ルール122の場合)



・古典セルオートマトンでは、セルの内部状態が「緑: 1(ON)」  
または「黒: 0(OFF)」の2通りからなる  
規則的な市松模様が形成されます。

# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 1

## (1) 2次元量子セルオートマトンによる量子ライフゲームをQiskit1.0で実装

(a) ライフゲーム(Conway's Game of Life)は1970年より研究されてきたセルオートマトンの一種です。

生命の誕生や進化、淘汰などのプロセスを連想させる複雑なパターンを生み出すことが出来ます。 [8] [12] [13]

(b) 本研究では、下記のルールを用いて量子ライフゲームを実装しました。

(b-1) 3セル \* 3セル平方の周囲セルの内部状態により、次世代セルの内部状態をセット

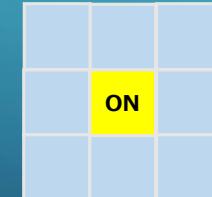
(b-2) 8個の周囲のセルで、ON状態のセルの数により、次世代セルの内部状態をセット

(c) これにより、古典には見られない量子ライフゲーム特有のパターンを観測しました。※パターンについては先行研究あり [5]

(b-1)周囲の8セルの状態により次世代の中央の状態が決まる。

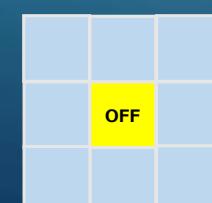
ON		ON
	OFF	
	ON	

周囲にONのセルが3つ  
-> 次世代の中央セルがON



ON		ON
ON	ON	ON
ON	ON	

周囲にONのセルが6つ  
-> 次世代の中央セルがOFF



(b-2) 周囲のセルでON状態のセルの数と対応するルール

セルの現在の状態	8個の周囲のセルでON状態のセルの数								次世代セルの状態
	2	3	4	5	6	7	8		
ON : ○			2	3					ON ○
	0	1		4	5	6	7	8	OFF ×
OFF : ×			3						ON ○
	0	1	2		4	5	6	7	OFF ×

# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 2

## (2) 量子ライフゲーム特有のパターンを紹介

(a) 量子ライフゲームで複数ステップに渡り安定して存在し続けるパターンを紹介します。

それぞれパターンごとに、「セルの初期状態(1世代目セルの内部状態)」、「遷移アニメーション」を表示します。

(b) セルの初期状態を下記のように表現します。

白セル :  $|0\rangle$  (OFF)をセット

青セル : 量子ゲートXをかけることで $|1\rangle$  (ON)をセット

黄セル : 量子ゲートHをかけることで、

$|0\rangle$  (OFF)と $|1\rangle$  (ON)の重ね合わせ状態をセット

★quTabやquBoatがパターン名

★パターンの特徴

- Oscillator : 周期的にON<->OFFを振動するパターン

- Still Life : ONのまま変化がないパターン

(b-1) セル内部状態の凡例

$$|\varphi\rangle = |0\rangle$$

$|1\rangle$

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

$|+\rangle$

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

(b-2) セル初期状態(1世代目セル)の凡例

quTub : Oscillator

パターン名

パターンの特徴

$ +\rangle$	$ 1\rangle$	$ +\rangle$
$ 1\rangle$		$ 1\rangle$
$ +\rangle$	$ 1\rangle$	$ +\rangle$

セルの初期状態

使用量子ビット数 : 58  
量子回路の深さ : 39696

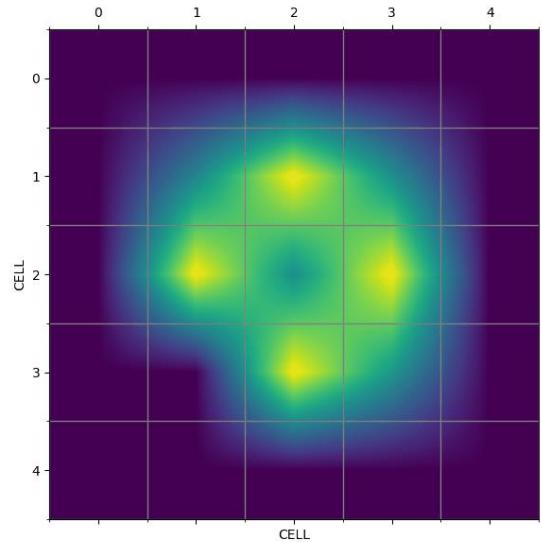
# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 3

quTub : Oscillator

	+>	1>	+>	
	1>		1>	
	+>	1>	+>	



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

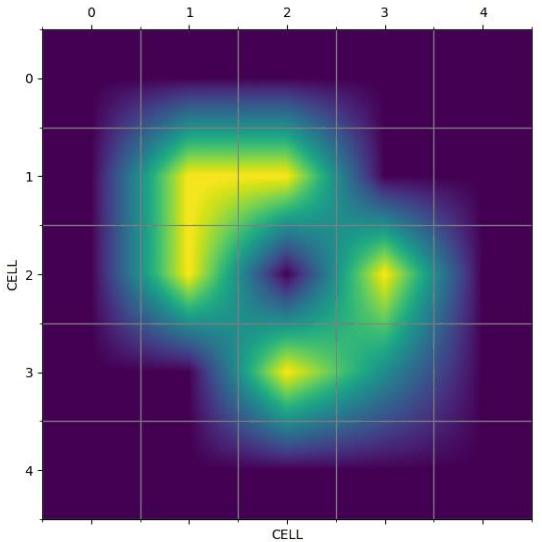


quBoat : Still Life

		1>	1>	
		1>		1>
			1>	+>



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

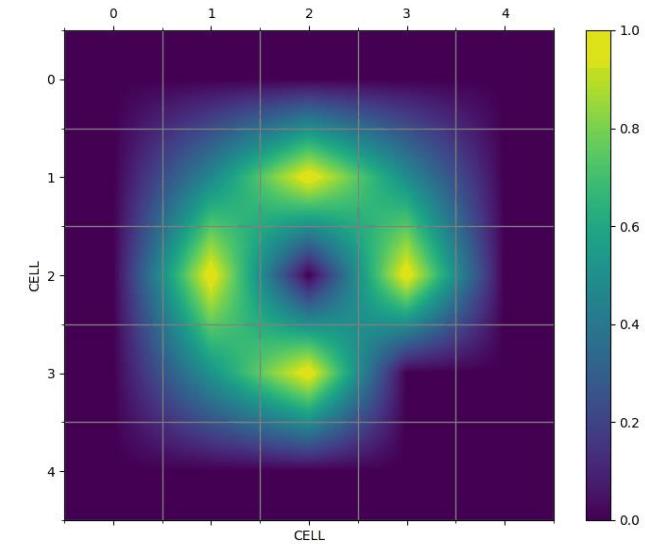


quAtamaran : Oscillator

		+>	1>	+>
		1>		1>
		+>	1>	



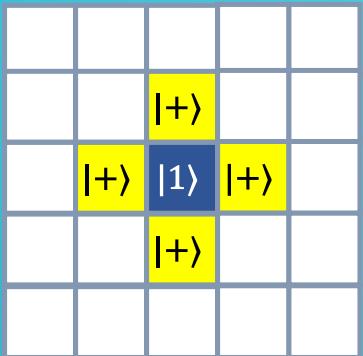
Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



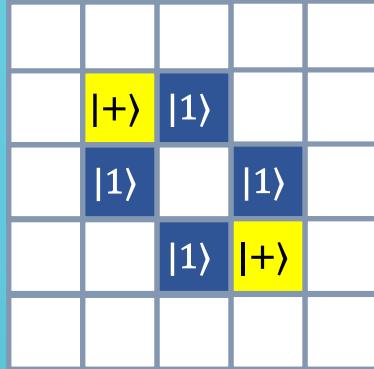
# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 4

使用量子ビット数 : 58  
量子回路の深さ : 39696

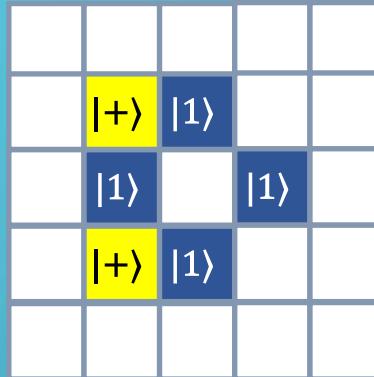
quDot : Oscillator



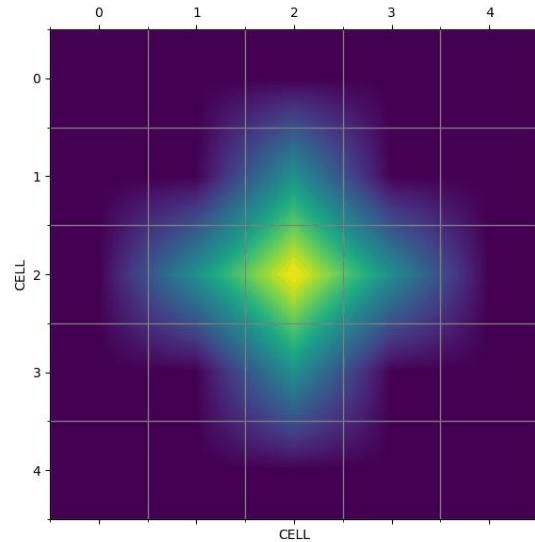
quRacle : Still Life



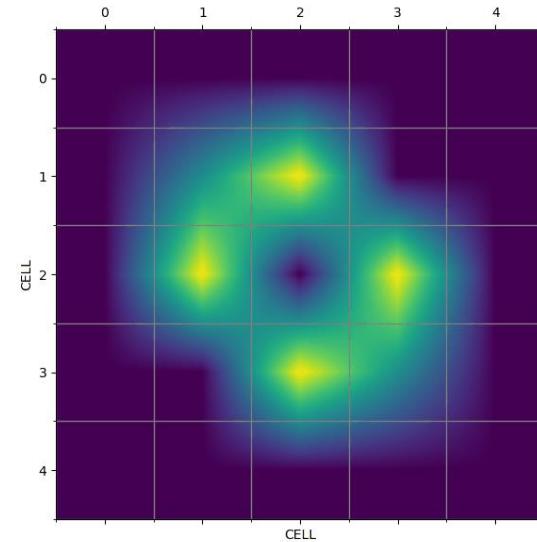
quTug : Oscillator



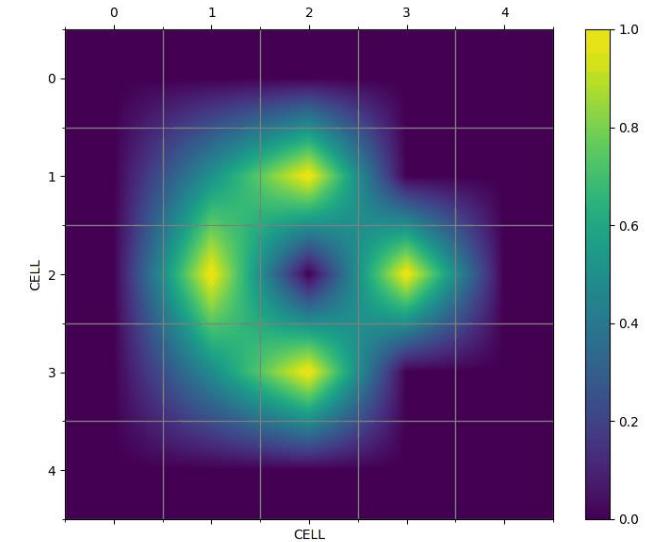
Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



使用量子ビット数 : 58  
量子回路の深さ : 39696

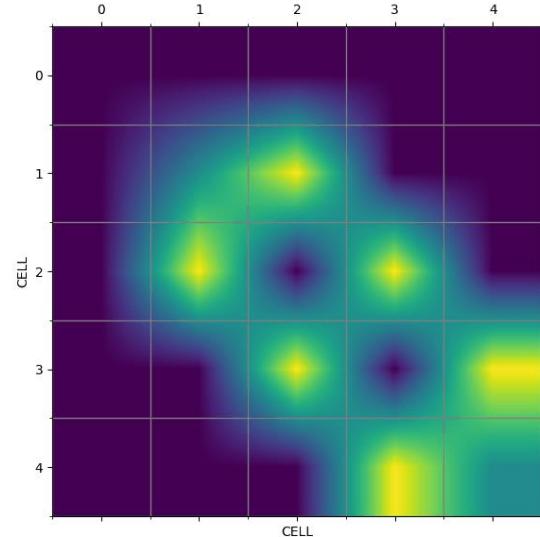
# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 5

quBarge : Still Life

	+>	1>		
1>			1>	
	1>			1>
		1>	1>	+>



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

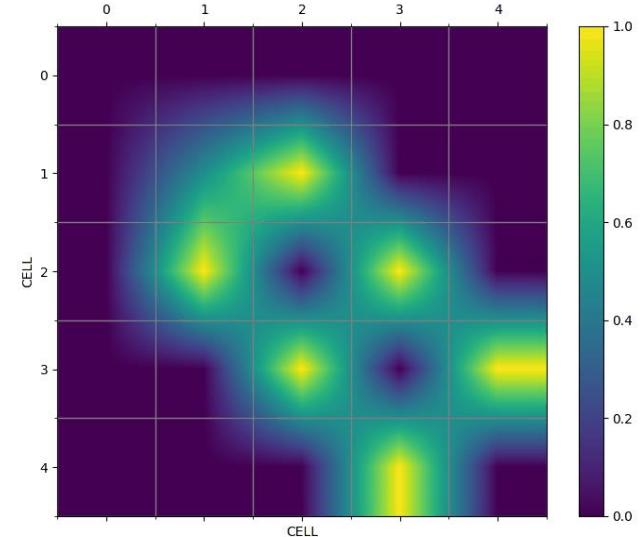


quRuiser : Still Life

	+>	1>		
1>			1>	
	1>			1>
		1>	1>	1>



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

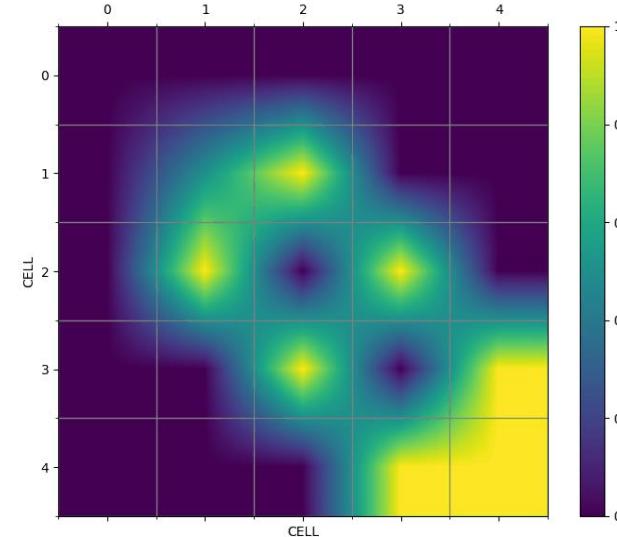


quLongBoat : Still Life

	+>	1>		
1>			1>	
	1>			1>
		1>	1>	1>



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



# 【量子セルオートマトン】⑤2次元量子セルオートマトン - 6

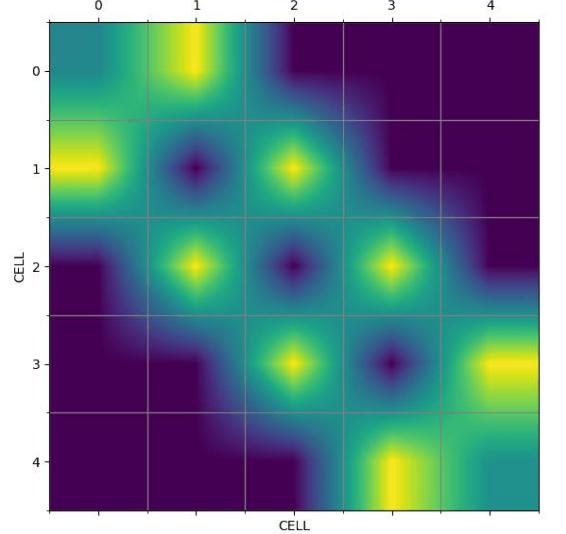
使用量子ビット数 : 58  
量子回路の深さ : 39696

quNoe : Oscillator

$ +\rangle$	$ 1\rangle$			
$ 1\rangle$		$ 1\rangle$		
	$ 1\rangle$		$ 1\rangle$	
		$ 1\rangle$		$ 1\rangle$
			$ 1\rangle$	$ +\rangle$



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

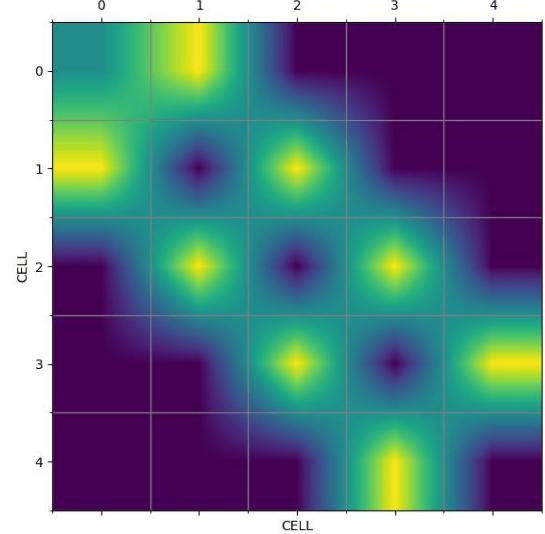


quYak : Oscillator

$ +\rangle$	$ 1\rangle$			
$ 1\rangle$		$ 1\rangle$		
	$ 1\rangle$		$ 1\rangle$	
		$ 1\rangle$		$ 1\rangle$
			$ 1\rangle$	$ +\rangle$



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2

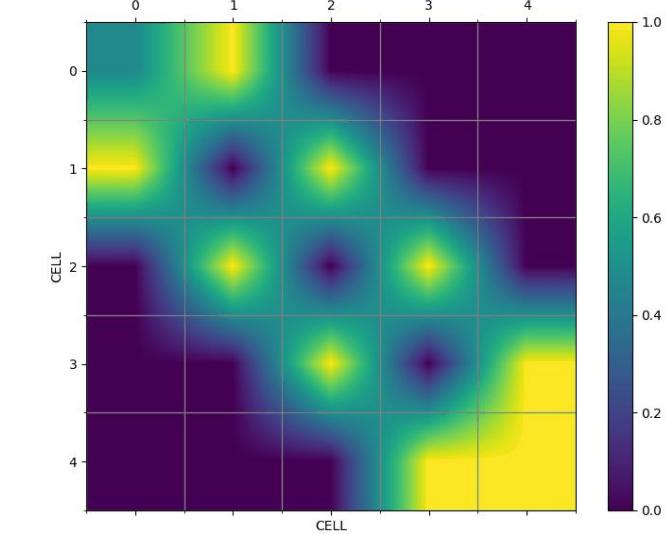


quVeryLongBoat : Oscillator

$ +\rangle$	$ 1\rangle$			
$ 1\rangle$		$ 1\rangle$		
	$ 1\rangle$		$ 1\rangle$	
		$ 1\rangle$		$ 1\rangle$
			$ 1\rangle$	$ 1\rangle$



Quantum Cellular Automata in 2-Dimension  
<Quantum Game of LIFE> Step -> 1 / 2



# 【量子セルオートマトン】⑥量子セルオートマトン実装の課題 – 1

## (1) 量子セルオートマトン実装の課題 (最大量子ビット数) - 1

(a) 量子セルオートマトンは、膨大な量の量子ビットを必要とする一方で、

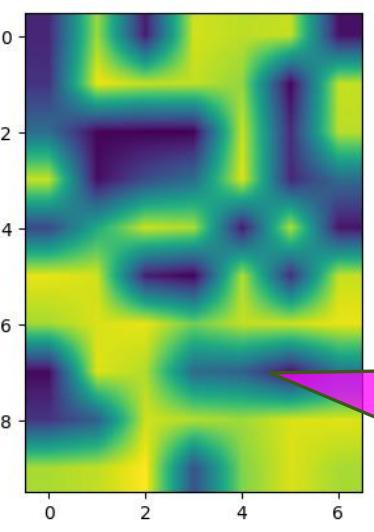
現在の量子コンピュータには、実機/シミュレータに関わらず量子ビット数の制限があります。

(b) 7セル10世代の1次元量子セルオートマトンの場合、下記の通り最低でも71量子ビットが必要となります。

$$(1\text{世代}7\text{セル} * 10\text{世代}) + \text{補助量子ビット } 1 = 71\text{量子ビット}$$

(c) Qiskitのシミュレータや実機では下記の通り、使用できる量子ビット数に制限があります。

使用量子ビット数 : 71  
量子回路の深さ : 32435



★basic\_simulator -> 最大 24量子ビット (量子ビット数が足りない)

★fake provider (ibm\_Kyoto) -> 最大124量子ビット (実行から7時間後にpython kernelがクラッシュ)

★qasm\_simulator -> PCのメモリ量に応じて量子ビットを増やせるが、71量子ビットを使用する場合、

$36,028,797,018,963,968\text{ MB} = 32\text{ ZB (32 ゼタバイト !)}$  必要。

★IBM Quantum 実機 (ibm\_Kyoto) -> 最大127量子ビット

IBM Quantum 実機での実行例  
1次元セルオートマトン (ルール30 : 1世代7セル 10世代)

実機処理時間 7sec

シミュレータでは実行できない量子ビット数でも、軽々と実行可能！

量子セルオートマトンの回路はステップ数が多く、

現在の実機では精度をもった正しい結果とならない。

# 【量子セルオートマトン】⑥量子セルオートマトン実装の課題 – 1

## (1) 量子セルオートマトン実装の課題 (最大量子ビット数) - 2

### (d) 行列積状態 : Matrix Product State (MPS)

近年、ニーズの高まっている技術 「行列積状態 (MPS)」を使用することで、量子ビットを63量子ビットまで拡張しました。

これにより、5\*5セル平方の量子ライフゲームを2ステップ実行できるようになり、量子ライフゲーム特有のパターンを観測しました。

※ 2ステップ分のセルを交互に使用することで、ステップ数を拡張することが出来るかどうか確認予定。

IBMの量子コンピュータ実機は、量子ビット数を大規模に拡張している最中です。本研究では、今後、量子ビット数が増大していく実機の計算の正しさを検証するための準備として、MPSを使用したシミュレーションモデルを実装しました。

### (e) 行列積状態:Matrix Product Stateとは

古典コンピュータで量子計算を行う場合、通常は量子ビットをState Vector(ベクトル)として表現します。

State Vectorでは、全ての量子ビットが取りうるパターンの確率振幅を記録しているため、

使用する量子ビット量に応じて、メモリで管理するベクトルのサイズが指数的に増大します。

Matrix Product Stateではテンソルネットワークを使い、量子状態やゲートの情報を行列の積で表現します。

全量子ビットのパターンそのものではなく、量子状態を間接的に表現することで必要メモリ数を減らし、

使用できる量子ビット数を増やします。 [14] [15] [16] [22]

※MPSは量子もつれの少ないモデルであれば有効で、量子もつれの多いモデルの場合は効率が下がります。

## 【量子セルオートマトン】⑥量子セルオートマトン実装の課題 - 2

### (2) 量子ライフゲーム実装の課題 (古典計算の除去) - 1

#### (a) 古典計算の除去

量子ライフゲームの先行事例を調査したところ、計算の途中で古典計算を挟んでおり、

量子の特性を完全には生かせていないモデルばかりでした。 [2][3][4]

このため、本研究では計算途中での古典計算を除去し、

より多く、量子の特性を組み込んだ量子ライフゲームを実装しました。

## 【量子セルオートマトン】⑥量子セルオートマトン実装の課題 - 2

### (2) 量子ライフゲーム実装の課題 (古典計算の除去) - 2

(b) 古典計算を挟んだ半量子ライフゲームは下記の論文で初出 [2]

SQGoL : Semi-Quantum Game of Life

"Towards a Quantum Game of Life" by Adrian P. Flitney & Derek Abbott (2008)

(c) 実装ごとに違いはあるものの、古典計算は下記の用途で使用されています。 [2] [3] [4]

(i) 周囲のON状態のセルをカウントする足し算で古典計算を使用。

→ 量子計算ではユニタリ発展を行うユニタリゲートを使う必要があります。

(ii) カウント結果をもとに、相互作用として次世代セルにセットする値が古典の固定値

→ 量子状態を正しく扱うためには、複素行列で扱う必要があります。

# 【量子セルオートマトン】⑥量子セルオートマトン実装の課題 - 2

独自性

## (2) 量子ライフゲーム実装の課題 (古典計算の除去) - 3

テクノロジー活用度

貢献度・有用性

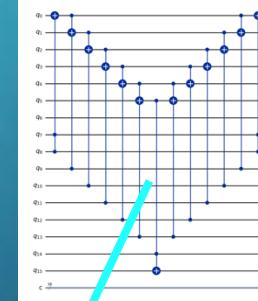
### (d) 全ルール書き下しによる古典計算の除去

周囲のセルを足し算でカウントする代わりに、全てのルールを書き下し、

ルールを1ずつユニタリ演算を使用して処理することで量子状態を保存したまま

次世代セルをセットできるようになりました。

c6xトヨリゲート (ccccccx)



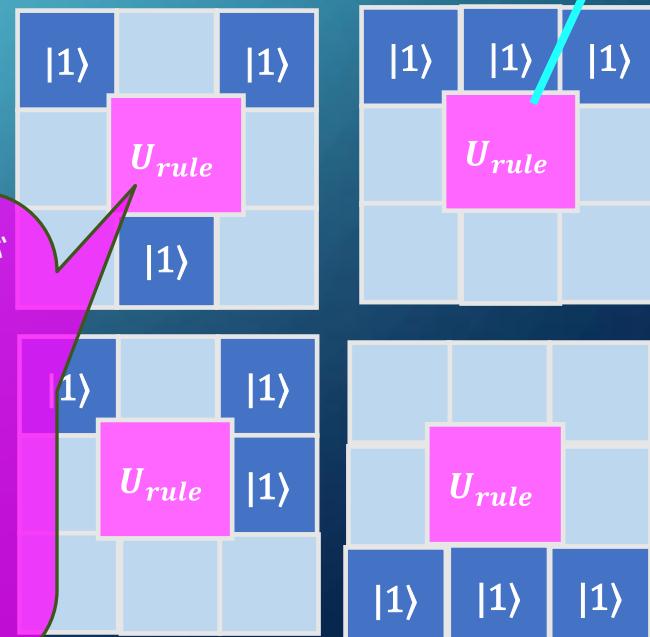
SQGoL (古典計算を使用)



- ★周囲のセルの状態を足し算してON (1)の状態のセル数合計が3と算出。
- ★足し算を使用すると、可逆性を失います。  
逆向きの計算を行って次世代から前世代の状態に戻すことが出来ません。

★周囲のON (1)の状態のセル数合計が3となるルールをすべて列挙し、ルールごとにユニタリ演算で処理します。各ルールによる演算結果が量子状態を保ったまま次世代セルに蓄積されます。  
★ユニタリ性を保っているので、逆向きの計算により、次世代から前世代の状態に戻すことが出来ます。  
[16] [17]

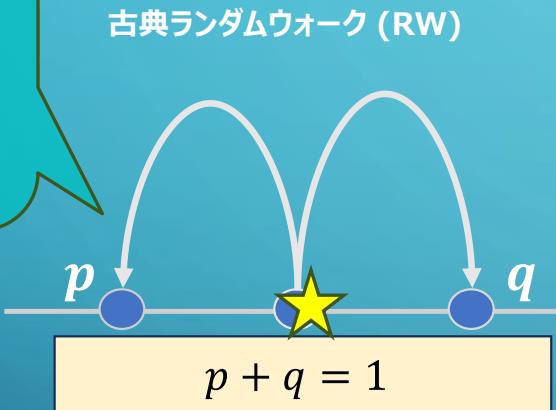
本研究のモデル (全ルール書き下し)



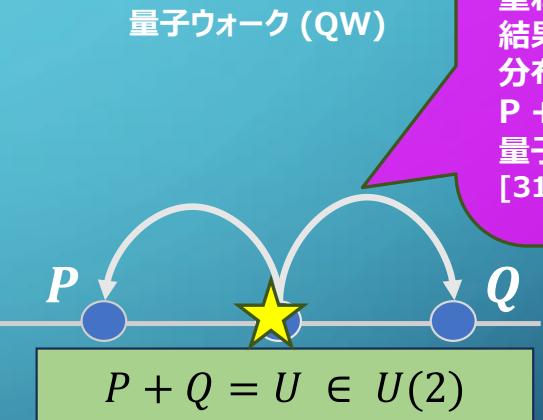
# 【量子ウォーク】①量子ウォークとは

- (1) 「古典ランダムウォーク(RW)」をベースにして、量子的な性質を持たせたモデルが「量子ウォーク(QW)」です。
- 両方ともに、空間を時間発展で移動するウォーカーの動きに焦点を当てたモデルです。

ウォーカー(★)が確率に従ってランダムに移動。  
左に移動する確率はp  
右に移動する確率はq  
 $p + q = 1$   
[1]



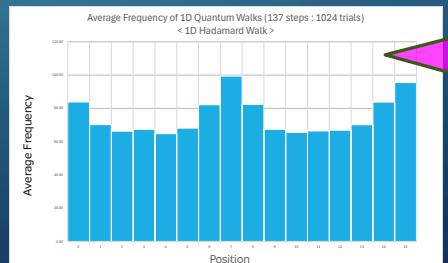
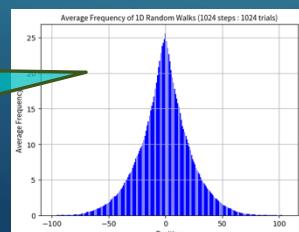
古典から量子へ



量子ウォーカー(★)は量子の性質を持つため、左向きの量子ウォーカーPと右向きの量子ウォーカーQが重ね合わさって存在します。結果として、複数の点に「確率振幅」として分布していきます。  
 $P + Q = U$ のUはユニタリ行列を表し、量子状態を保つために必要となります。  
[31]

## (2) 古典ランダムウォーク(RW)と、量子ウォーク(QW)のダイナミクス

古典ランダムウォークの頻度分布  
スタート地点の頻度が一番多く、  
ピークとなり、局在化しています。  
= 空間に広がりにくい。



量子ウォーク(アダマールウォーク)の頻度分布。  
謙虚なピークは見られず、局在化していません。  
= 空間に広がりやすい。  
= 空間内の検索に応用できる。

# 【量子ウォーク】②1次元量子ウォークの実装 - 1

## (1) 1次元2状態アダマールウォークをQiskit1.0で実装

(a) サイクル16の1次元2状態アダマールウォークを実装。

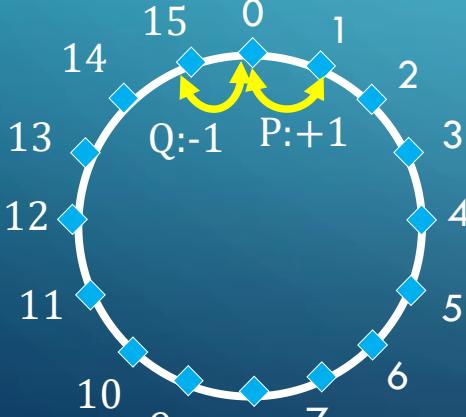
1次元の場所が16か所存在し、サイクルとして端がつながってるモデルです。左、および右への移動の2状態があります。

(b) 量子回路は①初期状態の設定、②コイン作用素、③シフト作用素から構成されます。[6] [25] [26] [31]

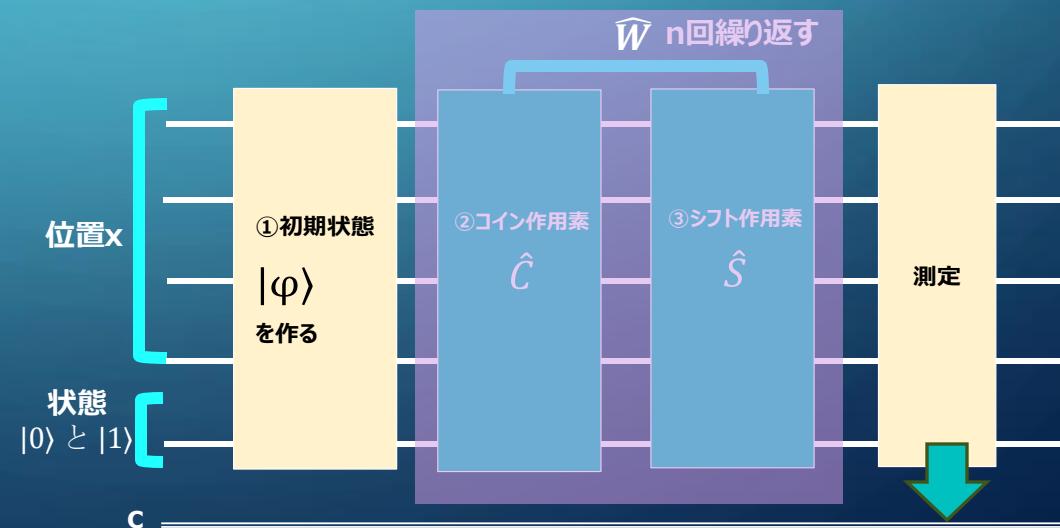
コイン作用素は、量子ウォーカーの動きを決めています。

シフト作用素は、左向き量子ウォーカーを左に移動させ、右向きの量子ウォーカーを右に移動させています。

(a) サイクル16の無向グラフ



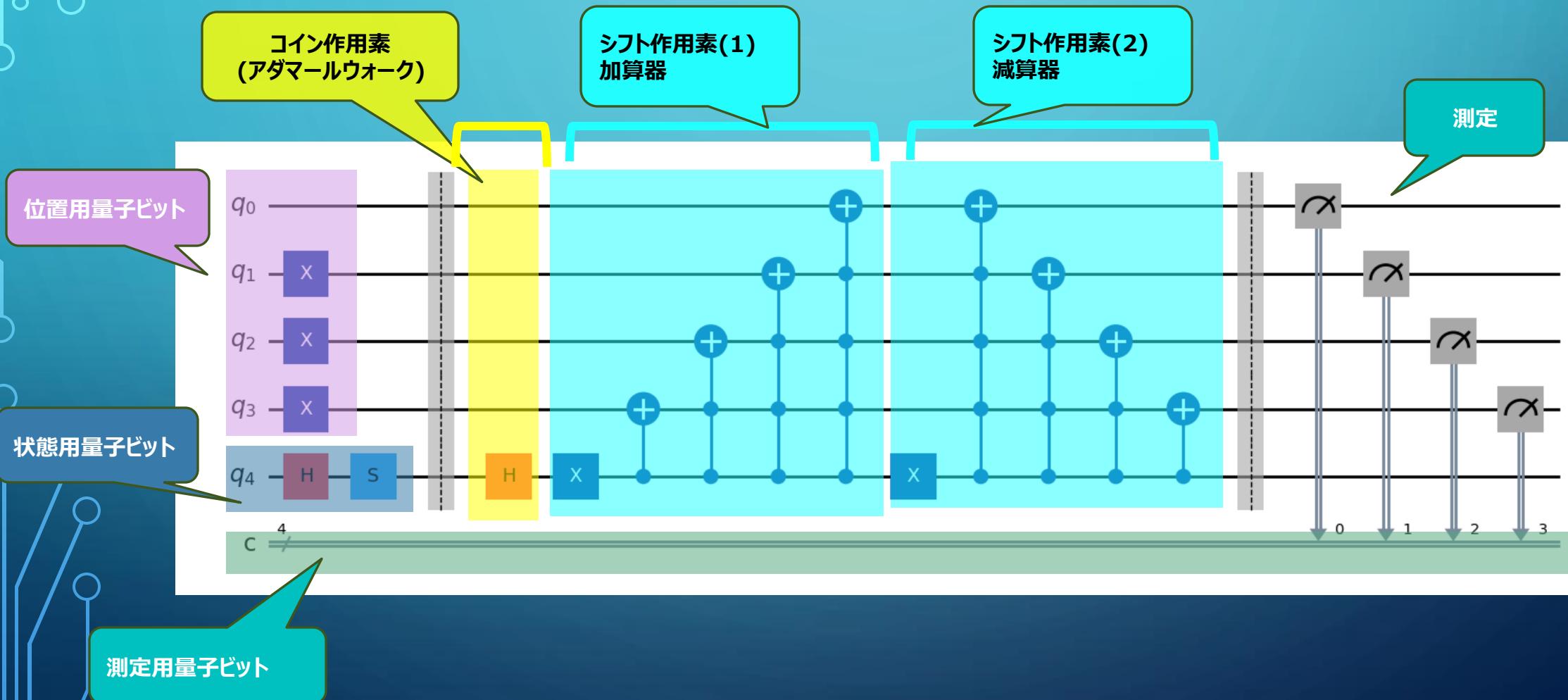
(b) 量子回路概要



# 【量子ウォーク】②1次元量子ウォークの実装 - 2

## (1) 1次元2状態アダマールウォークをQiskit1.0で実装

(c) 量子回路詳細



## (e) 頻度分布表 (0~137ステップ)

使用量子ビット数 : 5  
量子回路の深さ : 1499

bin(qiskit)	0000	1000	0100	1100	0001	1010	0110	1110	0000	1001	0101	1101	0011	1011	0111	1111
0 dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 step:0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0
2 step:1	0	0	0	0	0	0	0	0	622	0	578	0	0	0	0	0
3 step:2	0	0	0	0	0	0	0	0	311	0	588	0	303	0	0	0
4 step:3	0	0	0	0	0	0	0	0	143	0	447	0	431	0	179	0
5 step:4	0	0	0	0	0	0	0	0	450	0	168	0	450	0	51	0
6 step:5	0	0	0	0	0	0	0	0	407	0	164	0	195	0	383	0
7 step:6	0	0	0	0	0	0	0	0	13	0	388	0	156	0	20	0
8 step:7	0	0	0	0	0	0	0	0	293	0	208	0	110	0	122	0
9 step:8	0	0	0	0	0	0	0	0	195	0	253	0	99	0	02	0
10 step:9	0	0	0	0	0	0	0	0	302	0	324	0	76	0	85	0
11 step:10	0	0	0	0	0	0	0	0	66	0	103	0	79	0	95	0
12 step:11	0	0	0	0	0	0	0	0	91	0	89	0	81	0	87	0
13 step:12	0	0	0	0	0	0	0	0	123	0	48	0	151	0	393	0
14 step:13	0	0	0	0	0	0	0	0	388	0	156	0	137	0	158	0
15 step:14	0	0	0	0	0	0	0	0	143	0	447	0	431	0	179	0
16 step:15	0	0	0	0	0	0	0	0	79	0	450	0	168	0	452	0
17 step:16	0	0	0	0	0	0	0	0	407	0	164	0	151	0	393	0
18 step:17	0	0	0	0	0	0	0	0	13	0	388	0	156	0	20	0
19 step:18	0	0	0	0	0	0	0	0	208	0	110	0	122	0	250	0
20 step:19	0	0	0	0	0	0	0	0	195	0	253	0	99	0	303	0
21 step:20	0	0	0	0	0	0	0	0	302	0	324	0	76	0	85	0
22 step:21	0	0	0	0	0	0	0	0	66	0	103	0	79	0	95	0
23 step:22	0	0	0	0	0	0	0	0	89	0	81	0	87	0	102	0
24 step:23	0	0	0	0	0	0	0	0	123	0	48	0	151	0	393	0
25 step:24	0	0	0	0	0	0	0	0	388	0	156	0	137	0	158	0
26 step:25	0	0	0	0	0	0	0	0	143	0	447	0	431	0	179	0
27 step:26	0	0	0	0	0	0	0	0	79	0	450	0	168	0	452	0
28 step:27	0	0	0	0	0	0	0	0	407	0	164	0	151	0	393	0
29 step:28	0	0	0	0	0	0	0	0	13	0	388	0	156	0	20	0
30 step:29	0	0	0	0	0	0	0	0	205	0	148	0	45	0	49	0
31 step:30	0	0	0	0	0	0	0	0	121	0	304	0	208	0	216	0
32 step:31	0	0	0	0	0	0	0	0	169	0	14	0	237	0	177	0
33 step:32	0	0	0	0	0	0	0	0	0	...	...	...	...	...	...	...
34 step:33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35 step:34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36 step:35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37 step:36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38 step:37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39 step:38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40 step:39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41 step:40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42 step:41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43 step:42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44 step:43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45 step:44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46 step:45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47 step:46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48 step:47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49 step:48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50 step:49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
51 step:50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
52 step:51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
53 step:52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
54 step:53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55 step:54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56 step:55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57 step:56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
58 step:57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
59 step:58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60 step:59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
61 step:60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
62 step:61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
63 step:62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64 step:63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
65 step:64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
66 step:65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
67 step:66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
68 step:67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

古典ランダムウォークのように原点に局在化しません。

最初はそろっていた確率振幅の波が広がり、干渉のようなパターンが広がっていきます。

bin(qiskit)	0000	1000	0100	1100	0001	1010	0110	1110	0000	1001	0101	1101	0011	1011	0111	1111
69 step:68	0	25	0	48	0	396	0	134	0	419	0	37	0	24	0	117
70 step:69	0	6	0	64	0	55	0	411	0	431	0	49	0	44	0	80
71 step:70	0	69	0	18	0	85	0	721	0	108	0	25	0	69	0	105
72 step:71	0	87	0	15	0	13	0	481	0	477	0	17	0	11	0	119
73 step:72	0	64	0	5	0	59	0	621	0	80	0	12	0	61	0	118
74 step:73	0	106	0	17	0	22	0	470	0	429	0	24	0	18	0	104
75 step:74	0	10	0	22	0	322	0	298	0	314	0	25	0	13	0	216
76 step:75	0	0	17	0	219	0	247	0	271	0	213	0	24	0	107	0
77 step:76	0	73	0	152	0	238	0	208	0	236	0	144	0	65	0	84
78 step:77	0	93	0	115	0	290	0	107	0	118	0	124	0	61	0	101
79 step:78	0	159	0	271	0	122	0	153	0	113	0	217	0	162	0	94
80 step:79	0	81	0	221	0	148	0	100	0	124	0	121	0	157	0	103
81 step:80	0	177	0	210	0	187	0	137	0	118	0	125	0	149	0	102
82 step:81	0	97	0	197	0	140	0	121	0	130	0	198	0	120	0	99
83 step:82	0	90	0	145	0	155	0	133	0	133	0	133	0	133	0	99
84 step:83	0	34	0	230	0	38</td										

# 【量子ウォーク】④量子検索と量子優位性

## (1) 量子検索

グローバーウォークというタイプの量子ウォークが量子検索問題に応用されています。

## (2) 量子優位性

古典アルゴリズムよりも量子アルゴリズムの方が大幅に計算量が少ないアルゴリズムは、**量子優位性**があると呼びます。

※ 量子超越性 (Quantum supremacy) 古典コンピュータにとって現実的な時間内では解けない問題を、量子コンピュータが解ける場合。

※ 量子優位性 (Quantum advantage) 古典コンピュータが解くよりも、高速または少ないリソースで量子コンピュータが解ける場合。

古典アルゴリズムによる検索アルゴリズムの場合、 $O(N)$  回の計算量が必要ですが、

グローバーウォークを利用した検索アルゴリズムでは  $O(\sqrt{N})$  回の計算量を行うと

$1 - (1/N)$  以上の確率で正しい結果を求めることができます。

これにより、グローバーウォークの量子優位性は数学的にも証明されています。[27] [28] [30]

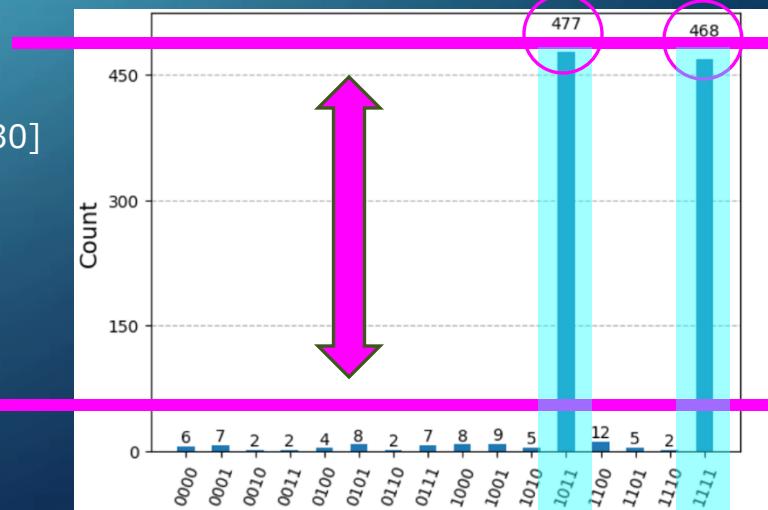
(3) 本研究ではQiskit notebook教材の「量子ウォークによる探索アルゴリズム」を  
Qiskit 1.0で稼働するように改修しました。[26]

図は「4次元超立方体上の量子ウォークによる検索」の実行結果。

検索対象のビットのみ、確率振幅が大幅に増幅していることが分かります。

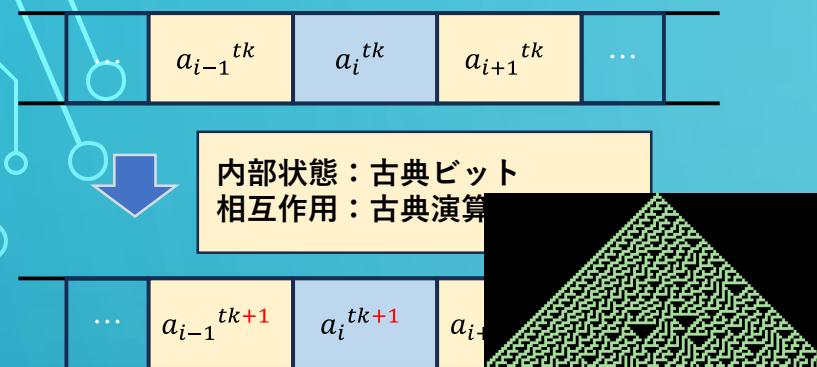
使用量子ビット数 : 11  
量子回路の深さ : 9561

4次元超立方体上の量子ウォークによる検索結果



## 【数理モデル間の対応関係】

## (A) 古典セルオートマトン (CCA)



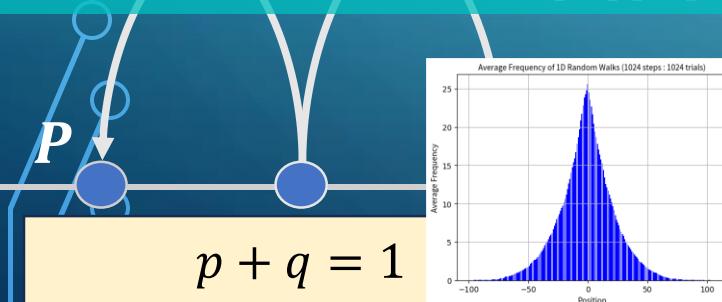
## 【共通点】

## 空間が時間発展します。

シンプルなルールから複雑なパターンが生み出されます。

## 【異なる点】

- ・古典セルオートマトンは決定論的なプロセスのため、完全に同じ条件/同じで実行する限り、出力は常に同一となります。
  - ・古典ランダムウォークは、確率的なプロセスのため、各ステップがランダムに定まります。[23] [24]

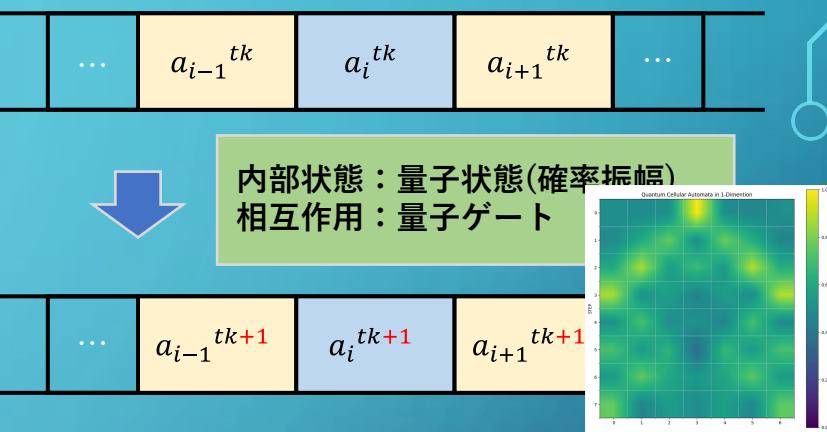


### (C) 古典ランダムウォーク (RW)

## 【異なる点】

量子セルオートマトンでは、初期状態に量子状態をセットすることで、同じルールでも時間発展が異なる。

### (B) 量子セルオートマトン (QCA)



## 【共通点】

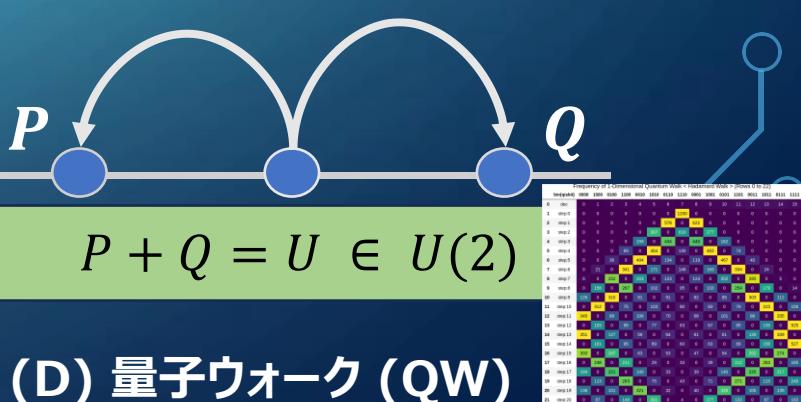
定義だけ見ると異なるように見えますが、一般的な設定では量子セルオートマトンと量子ウォークは各要素で対応する同等の数理モデルとなります。[29]

【共通点】

- ・時間的に独立にランダムなユニタリ行列によって時間発展する量子ウォークに対して、期待値をとると古典ランダムウォークとなります。[29]

【異なる点】

- ・量子ウォークは、純粹にランダム性に依存しないプロセスのため、ランダムの名称が外されています。
  - ・量子ウォークは、古典ランダムウォークよりも効率的に検索することが出来るアルゴリズムもある。[29]



# 【実機/シミュレータ】① 1次元量子セルオートマトン

IBM Quantum 実機 / シミュレータ比較 : 1次元量子セルオートマトン (ルール : 122)

実機とシミュレータは同じコードを使用しています。  
エラー訂正、エラー緩和の処理は入っていません。

セルオートマトンの「初期設定」および「ルール」は同一。

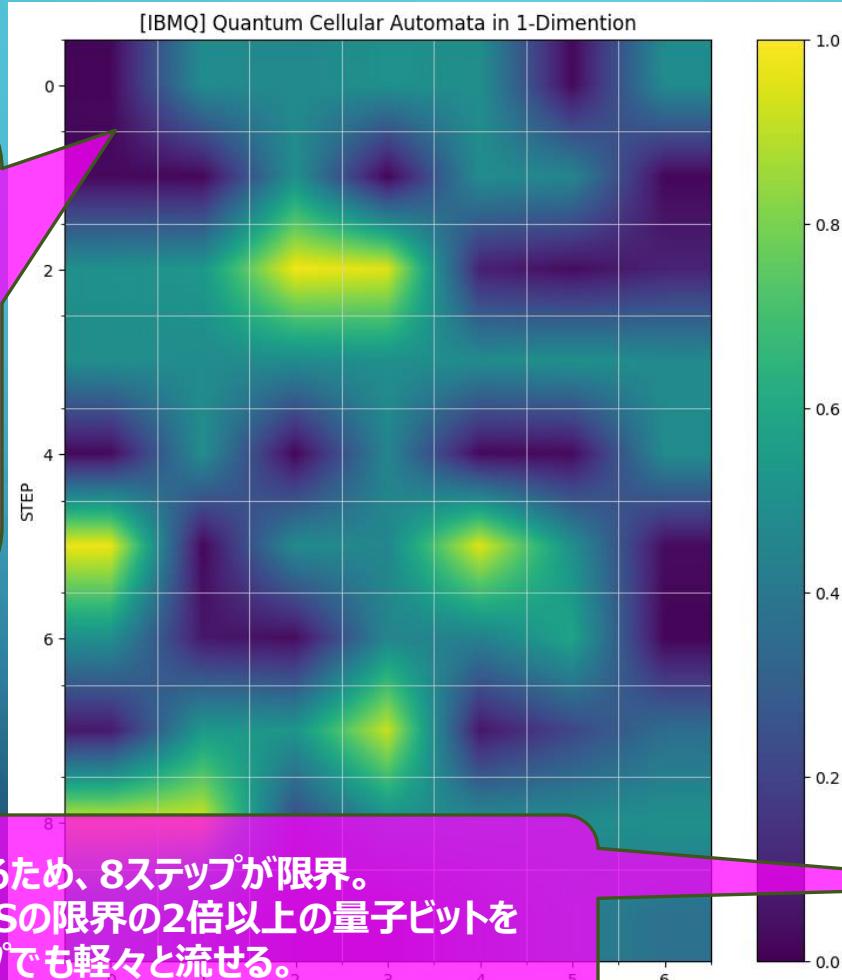
実機では、シミュレータとは異なるパターンが出現する。

これから、量子ビット数が増大していく実機の出力を検証するため、シミュレータにおいても、扱える量子ビット数の増大が望まれる。

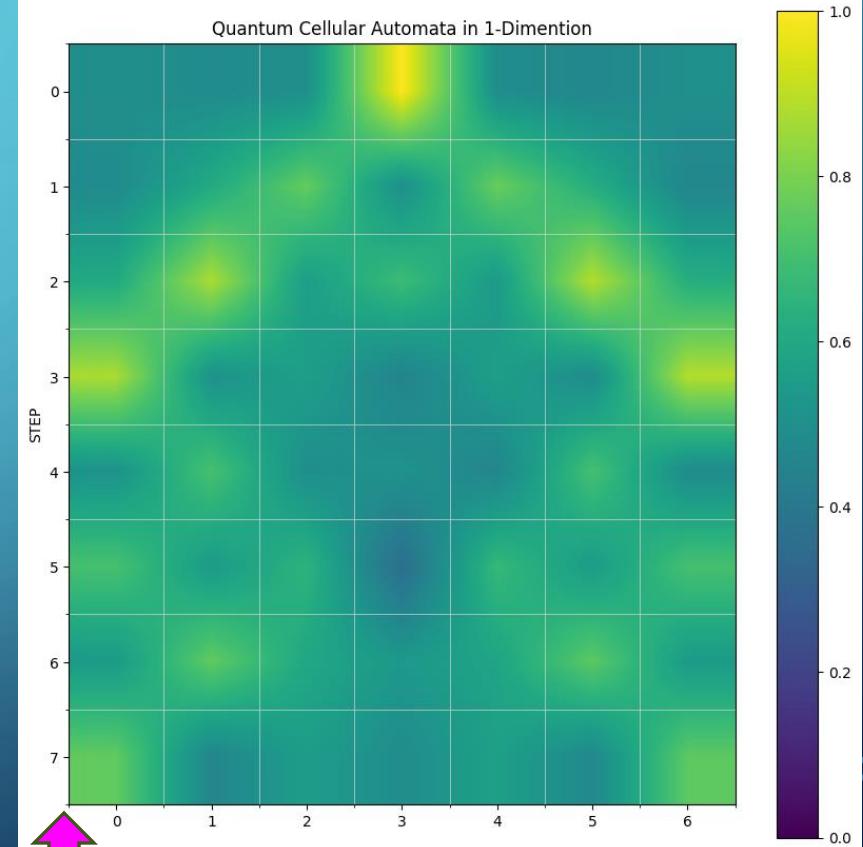
MPSでは、最大63量子ビット制限があるため、8ステップが限界。  
現在のIBM Quantum実機では、MPSの限界の2倍以上の量子ビットを使用できるため、+2ステップの10ステップでも軽々と流せる。

IBM Quantum 実機  
(ibm\_sherbrooke)

使用量子ビット数 : 71  
量子回路の深さ : 32435



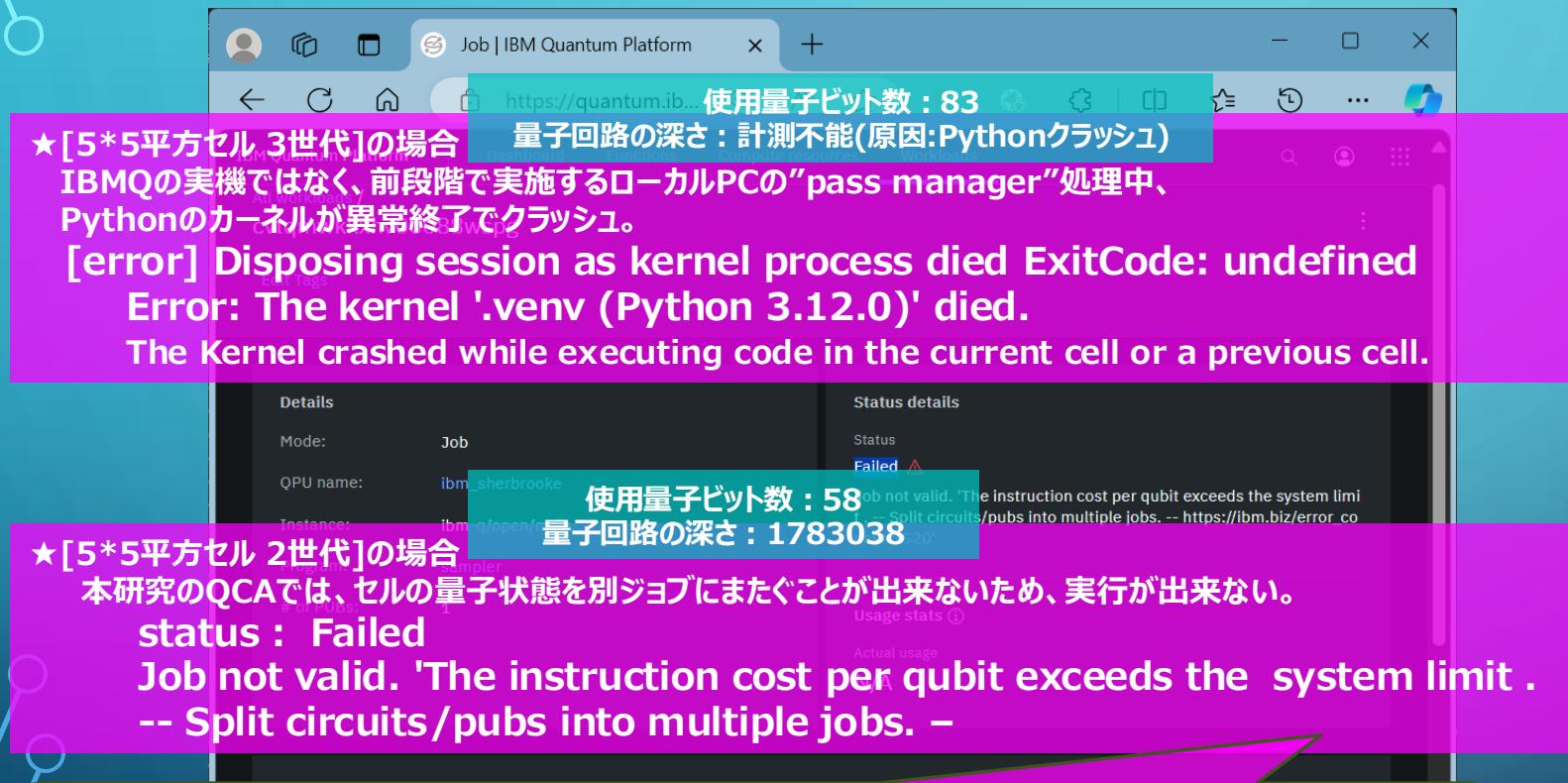
Qiskit シミュレータ  
(qiskit\_aermatrix\_product\_state)



# 【実機/シミュレータ】② 2次元量子セルオートマトン - 1

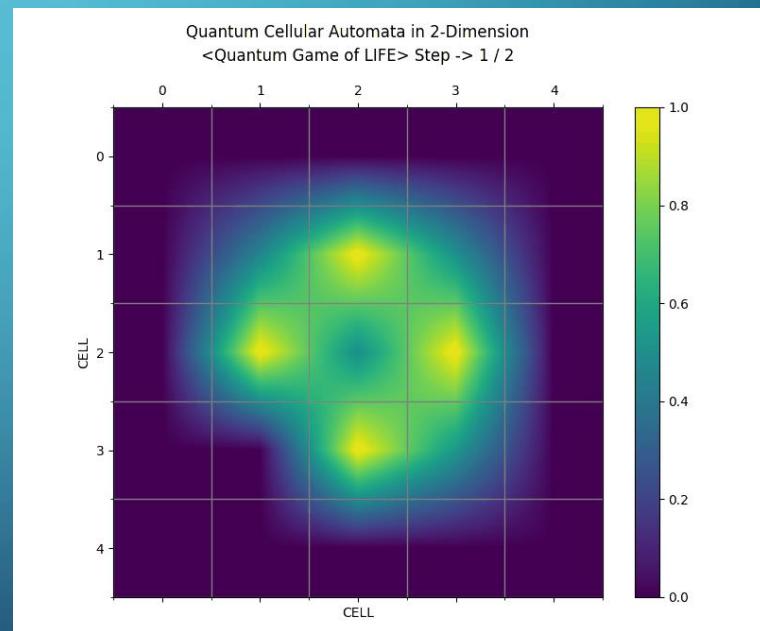
IBM Quantum 実機 / シミュレータ比較 : 2次元量子セルオートマトン 5平方セル

IBM Quantum 実機  
(ibm\_sherbrooke)



最新のQiskitRuntimeServiceライブラリでは  
量子回路の深さ制限にかかり、実行できないことが分かりました。  
これらの事象については、コミュニティーで新たなイシューを発行することで、貢献していきたいと思  
います。

Qiskit シミュレータ  
(qiskit\_aer / matrix\_product\_state)

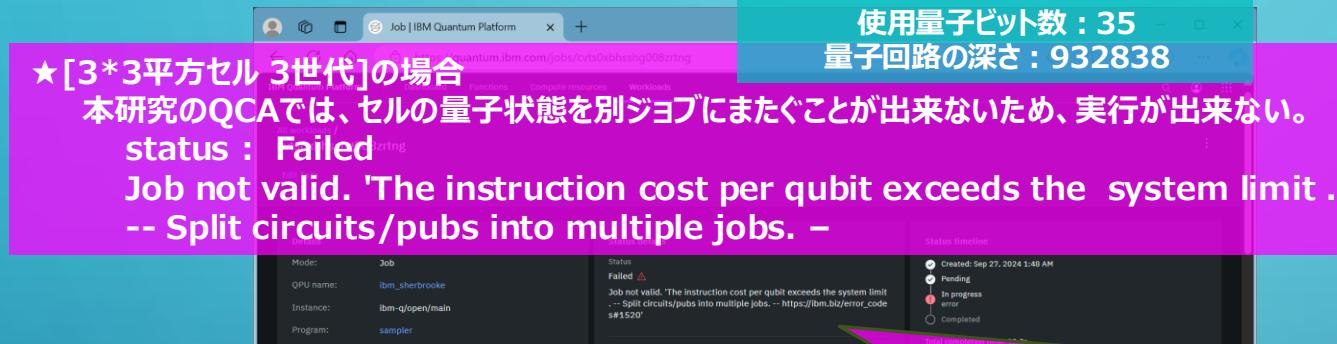


使用量子ビット数 : 58  
量子回路の深さ : 39696

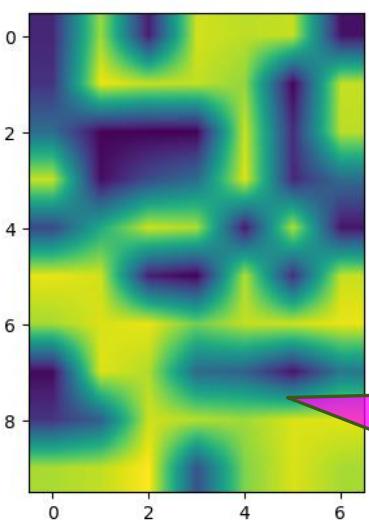
# 【実機/シミュレータ】② 2次元量子セルオートマトン - 2

## IBM Quantum 実機 / シミュレータ比較 2次元量子セルオートマトン（ライフゲーム）3平方セル 3世代の場合

IBM Quantum 実機  
(ibm\_sherbrooke)



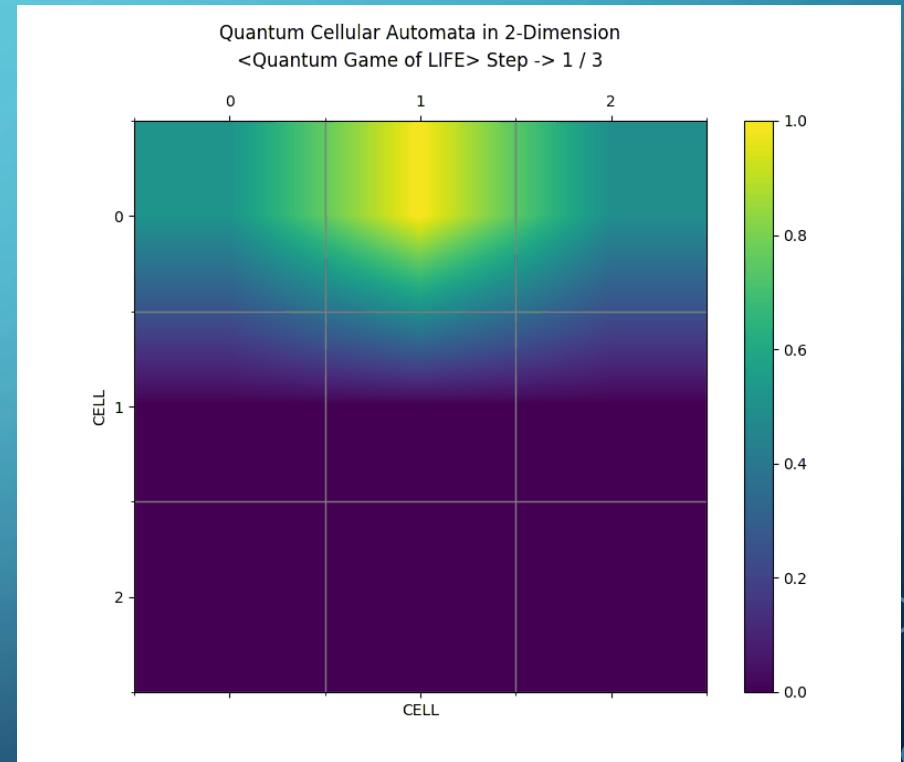
使用量子ビット数 : 71  
量子回路の深さ : 32435



最新のQiskitRuntimeServiceライブラリでは  
量子回路の深さ制限にかかり、実行できないことが分かりました。  
これらの事象について、コミュニティで新たなイシューを発行して、貢献  
していきたいと思います。

qiskit\_ibm\_providerライブラリを使用した場合、  
実機では 3平方セル \* 3世代であっても18秒で軽く実行可能。

Qiskit シミュレータ  
(qiskit\_aer / matrix\_product\_state)



使用量子ビット数 : 35  
量子回路の深さ : 20655

# 【実機/シミュレータ】② 2次元量子セルオートマトン - 3

実機とシミュレータは同じコードを使用しています。  
エラー訂正、エラー緩和の処理は入っていません。

廃止予定となっている  
旧 ibm\_provider  
ライブラリでは実機において、  
最新ibm\_runtimeの  
制限を超えて  
4ステップ実行可能でした。

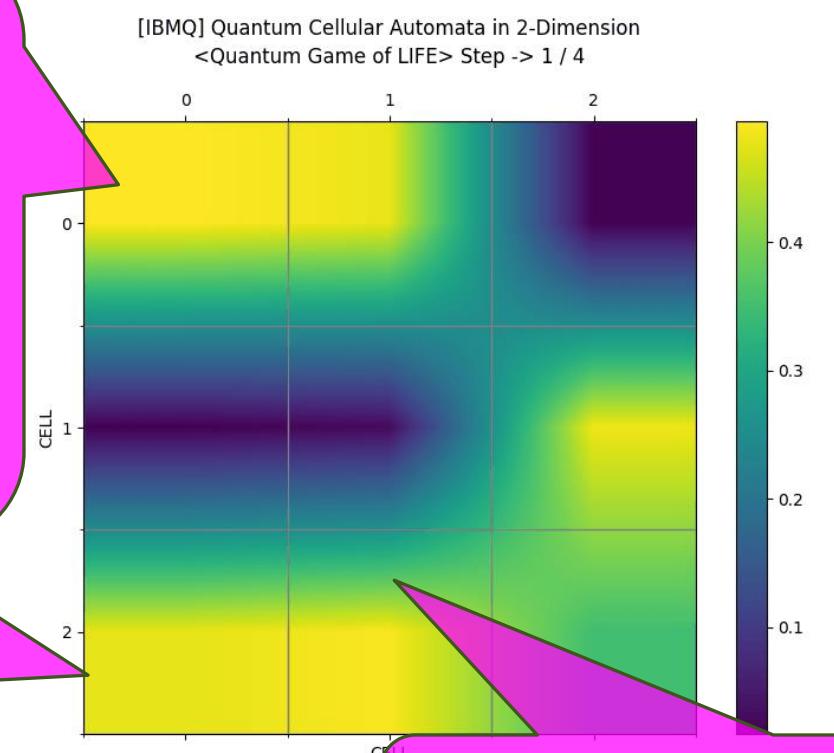
※ibm\_providerであっても、  
5平方セル3世代では  
ローカルPCのtranspile中に  
Pythonがクラッシュするため  
実行できない。

初期設定は同じだが  
1ステップ目から  
MPSシミュレータとは  
出力が異なる。

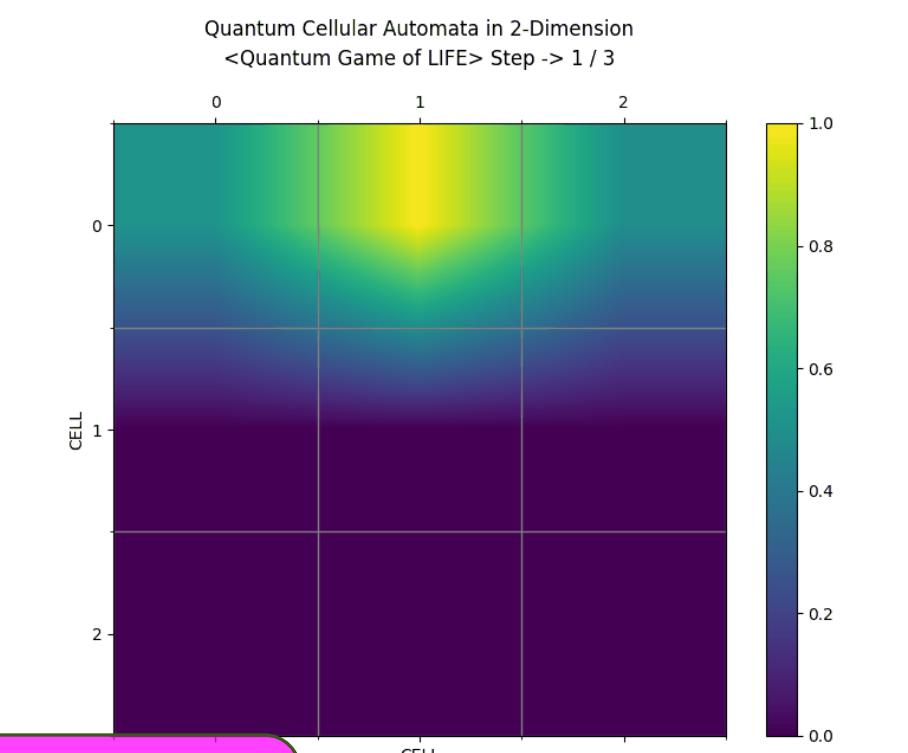
使用量子ビット数 : 35  
量子回路の深さ : 1432973

IBM Quantum 実機 / シミュレータ比較  
2次元量子セルオートマトン (ライフゲーム) : 3平方セル の場合

IBM Quantum 実機  
(ibm\_sherbrooke)



Qiskit シミュレータ  
(qiskit\_aer / matrix\_product\_state)



旧 ibm\_providerで実施可能な回路が  
最新のQiskitRuntimeServiceライブラリでは実行できない事象  
については、コミュニティで新たなイシューを発行して貢献していく  
思います。

使用量子ビット数 : 35  
量子回路の深さ : 20655

# 【実機/シミュレータ】③ 量子ウォーク <1次元アダマールウォーク>

貢献度・有用性

実機とシミュレータは同じコードを使用しています。  
エラー訂正、エラー緩和の処理は入っていません。

IBM Quantum 実機  
(ibm\_Sherbrooke 4ステップのみ実行)

実機の場合は、0ステップ目の初期設定の時点で誤差が出ており、“1110”以外の量子ビットにも|1>がセットされている。

bin(qiskit)	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
0 dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 step:0	0	1	1	62	0	18	14	1099	0	0	0	0	0	0	0	5
2 step:1	84	101	86	88	76	98	85	92	70	52	58	62	68	62	48	66
3 step:2	83	105	91	95	75	90	88	81	56	61	58	63	64	63	64	63
4 step:3	98	109	91	97	87	110	91	87	63	60	50	41	56	50	50	50

実機の場合は、量子ウォーカーの確率振幅が2方向ではなく、1ステップ目から空間全体に広がってしまう。

STEP	Qubit	Depth
1	5	4
2	5	806
3	5	2145
4	5	2413

IBM Quantum 実機 / シミュレータ比較  
量子ウォーク (アダマールウォーク)

Qiskit シミュレータ  
(qiskit.primitives / StatevectorSampler)

bin(qiskit)	0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
0 dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 step:0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	0	0
2 step:1	0	0	0	0	0	0	0	622	0	578	0	0	0	0	0	0
3 step:2	0	0	0	0	0	0	0	311	0	588	0	301	0	0	0	0
4 step:3	0	0	0	0	0	0	0	143	0	447	0	431	0	179	0	0
5 step:4	0	0	0	0	0	0	0	79	0	450	0	168	0	452	0	51
6 step:5	0	0	0	0	0	0	0	34	0	407	0	164	0	151	0	393
7 step:6	13	0	388	0	156	0	137	0	158	0	328	0	20	0	0	0
8 step:7	11	0	251	0	208	0	110	0	122	0	250	0	235	0	13	0
9 step:8	0	190	0	263	0	99	0	82	0	111	0	263	0	184	0	8
10 step:9	132	0	324	0	76	0	85	0	86	0	79	0	300	0	118	0
11 step:10	0	302	0	66	0	103	0	79	0	95	0	75	0	304	0	176
12 step:11	320	0	91	0	89	0	81	0	87	0	102	0	106	0	324	0
13 step:12	0	133	0	119	0	59	0	55	0	70	0	91	0	146	0	527
14 step:13	348	0	123	0	48	0	59	0	59	0	49	0	121	0	393	0
15 step:14	0	155	0	73	0	64	0	64	0	61	0	85	0	162	0	536
16 step:15	301	0	204	0	63	0	45	0	57	0	59	0	178	0	293	0
17 step:16	0	207	0	217	0	40	0	48	0	40	0	209	0	241	0	198
18 step:17	216	0	205	0	148	0	45	0	49	0	141	0	180	0	216	0
19 step:18	0	86	0	257	0	86	0	51	0	89	0	284	0	115	0	232
20 step:19	138	0	111	0	310	0	36	0	318	0	98	0	150	0	0	0
21 step:20	0	75	0	141	0	29	0	3	0	282	0	134	0	88	0	183
22 step:21	169	0	14	0	237	0	177	0	172	0	143	0	12	0	176	0

シミュレータでは、「右向き」、「左向き」量子ウォーカーの確率振幅が2方向に広がっていく様子が分かる。

使用量子ビット数 : 5  
量子回路の深さ : 1499

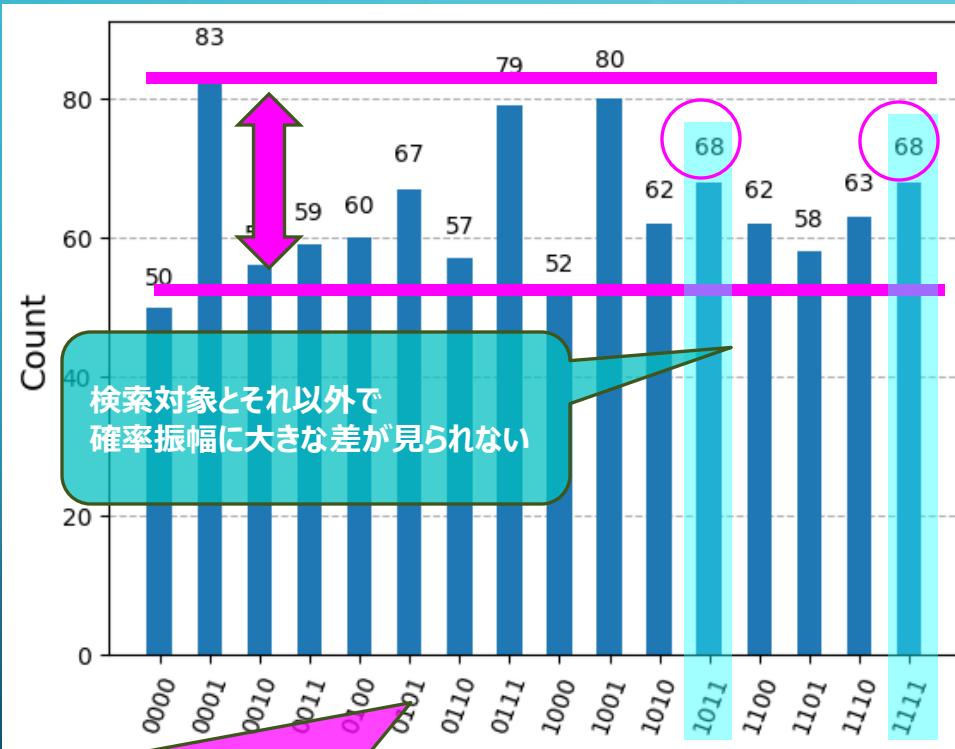
# 【実機/シミュレータ】④ 量子ウォーク <量子検索>

実機とシミュレータは同じコードを使用しています。  
エラー訂正、エラー緩和の処理は入っていません。

IBM Quantum 実機 / シミュレータ比較  
量子ウォーク (4次元超立方体上の量子ウォークによる探索)  
グローバーコインを使用したコイン量子ウォーク

IBM Quantum 実機  
(ibm\_sherbrooke)

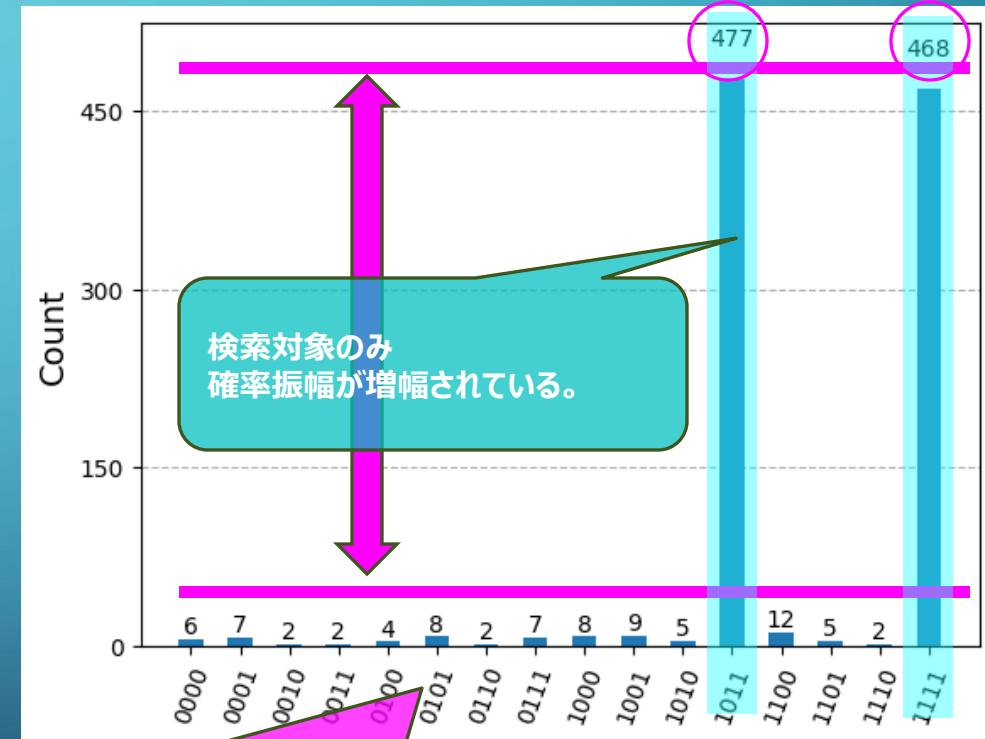
使用量子ビット数 : 11  
量子回路の深さ : 98286



実機の場合は、位相推定が正しく機能しておらず、  
事前に位相オラクルでマークした“1011”、“1111”的確率振幅と、それ以外の確率振幅に差が見られない。  
※シミュレータ/実機ともに誤り訂正是行っていません。

Qiskit シミュレータ  
(Qiskit 1.0 / BasicSimulator)

使用量子ビット数 : 11  
量子回路の深さ : 9561



グローバーウォークによる位相推定が正しく機能し、  
事前に位相オラクルでマークした“1011”、“1111”的確率振幅が増幅されている。

# 【量子セルオートマトンと量子ウォークまとめ】

## ★本研究の内容

量子セルオートマトンと量子ウォークをQiskit 1.0で実装することにより、これらの挙動を確認することができました。

## ★課題と今後の展望

- ・量子ビット数制限

さらに大きな盤面、さらに多いステップ数を目的として、現在の量子ビット数制限を超えていくことで、さらなる拡張をしていきたいです。

- ・実機での量子計算の信頼性向上

シミュレータとの比較検討により、シミュレータのコードそのまま実機で実行するだけでは出力にノイズが多く出てしまうことが分かりました。これから「量子エラー緩和」、「量子エラー訂正」を勉強して実装することで、実機における出力を改善していきたいです。

# 【研究成果まとめ】

独自性

最新のqiskit1.0以上で作成したコードをgithubにまとめました

[GitHub - wg-quantum/2024-b-06](#)

<https://github.com/wg-quantum/2024-b-06/>

テクノロジ  
活用度

貢献度  
有用性

量子機械学習はさまざまなパラメータを調整することによって古典機械学習にせまる精度を出すことができることが分かりました

一部のデータについては**古典よりも高い精度**が出たが、結果に対する考察は十分な時間がなく、今後の課題となりました

独自性

最新のQiskit1.0以上で量子セルオートマトンおよび量子ウォークを実装しました

先行事例のモデルを改善し、古典計算を除去した**2次元量子ライフゲーム**を実装しました

シミュレータにおいて、**Matrix Product State (MPS)**により、**63量子ビット**まで拡張した

量子セルオートマトンを実装することで、**実機とシミュレーションの実行結果を検証しました**

テクノロジ  
活用度

貢献度  
有用性

本資料の著作権は、日本アイ・ビー・エム株式会社(IBM Corporationを含み、以下、IBMといいます。)に帰属します。ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものではありません。本資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMまたはセッション発表者は責任を負わないものとします。本資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものではありません。

本資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれが使用可能であることを暗示するものではありません。本資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもつていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものではありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

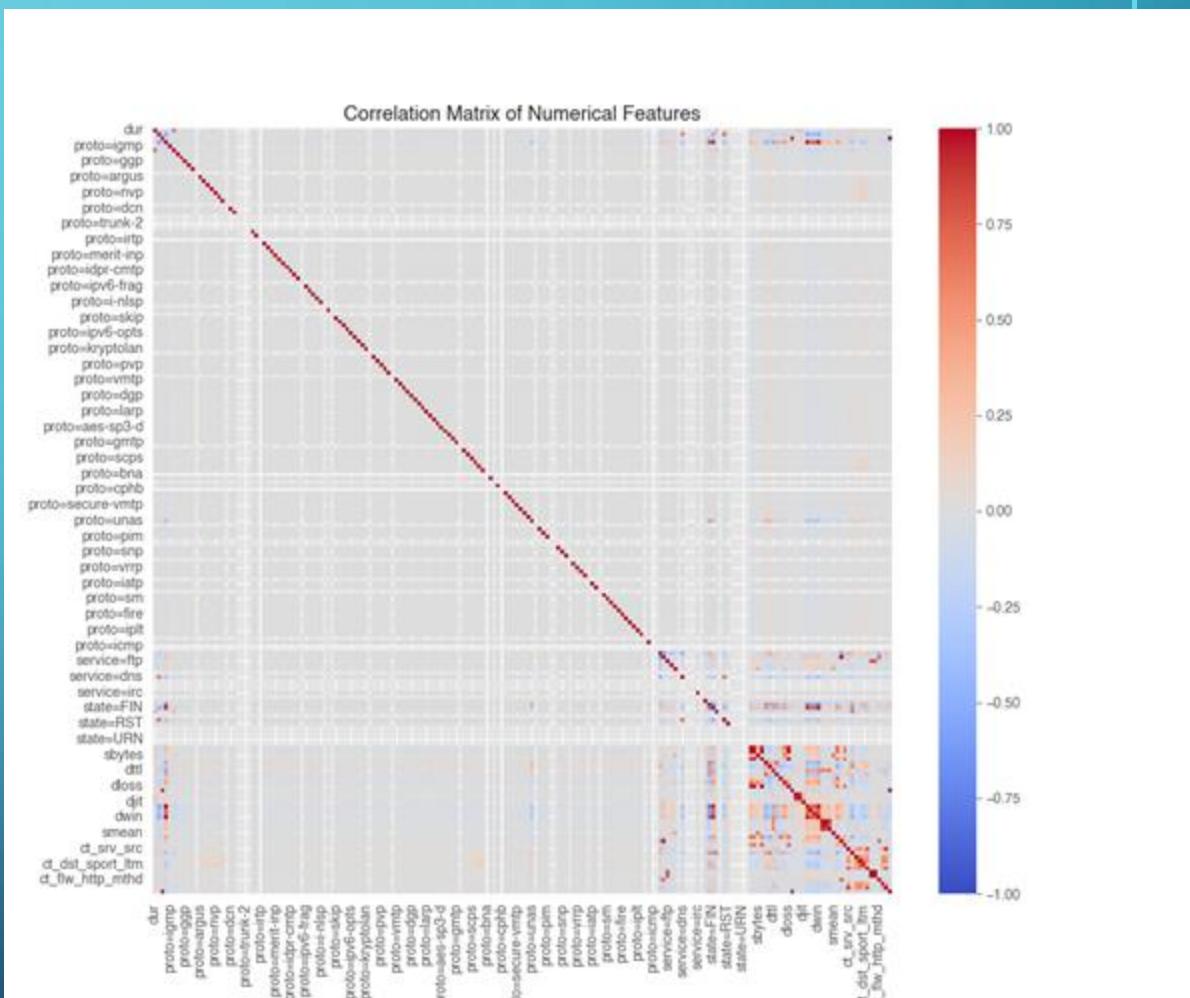
記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴは、米国やその他の国におけるInternational Business Machines Corporationの商標または登録商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点での IBM の商標リストについては、[ibm.com/trademark](http://ibm.com/trademark)をご覧ください。

# APPENDIX

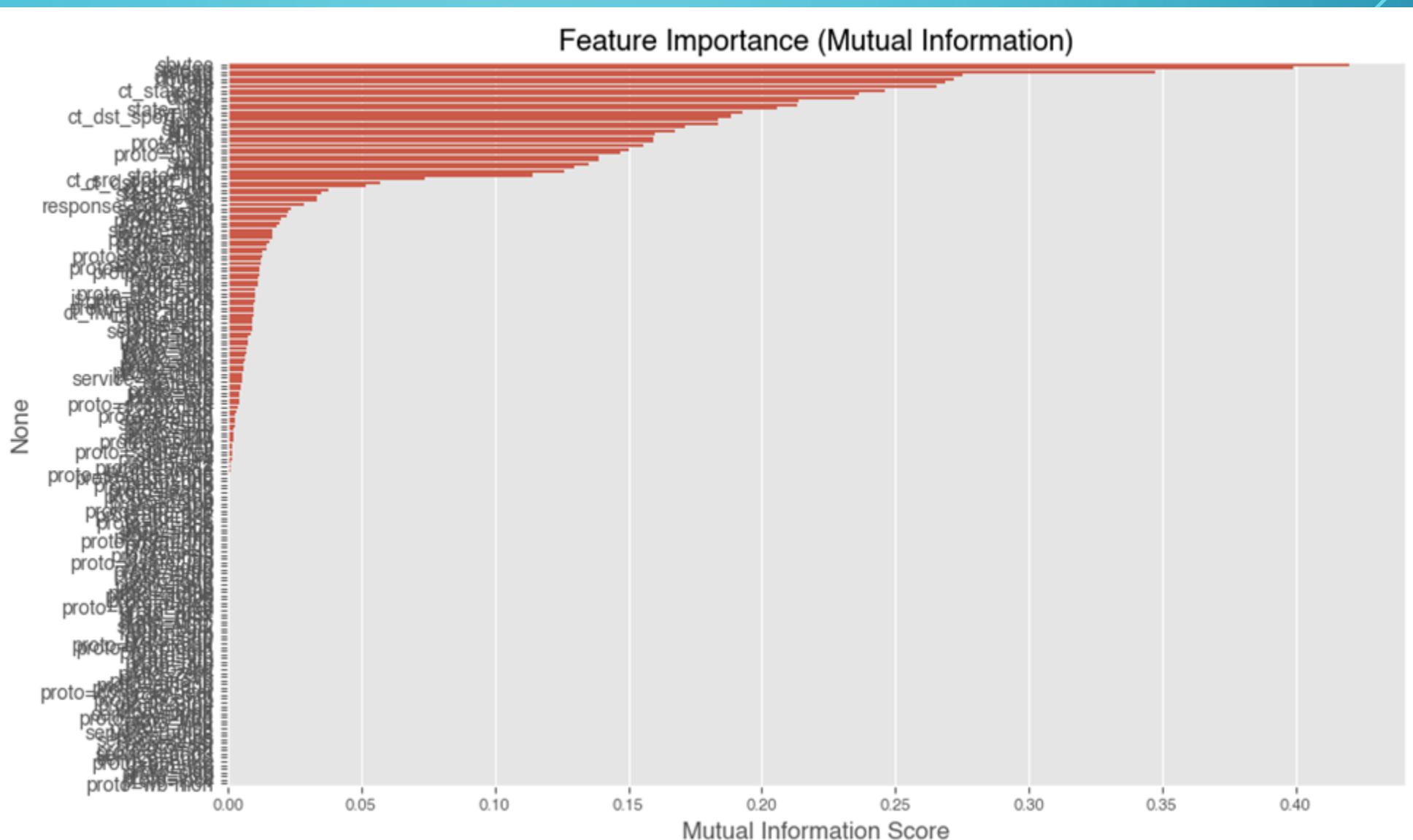
# データの詳細 - UNSW\_NB15\_TRAINTEST\_BACKDOOR

## Correlation Plot



# データの詳細 - UNSW\_NB15\_TRAINTEST\_BACKDOOR

# Feature Importance



# データの詳細 - BACKDOOR

UMAPデータ

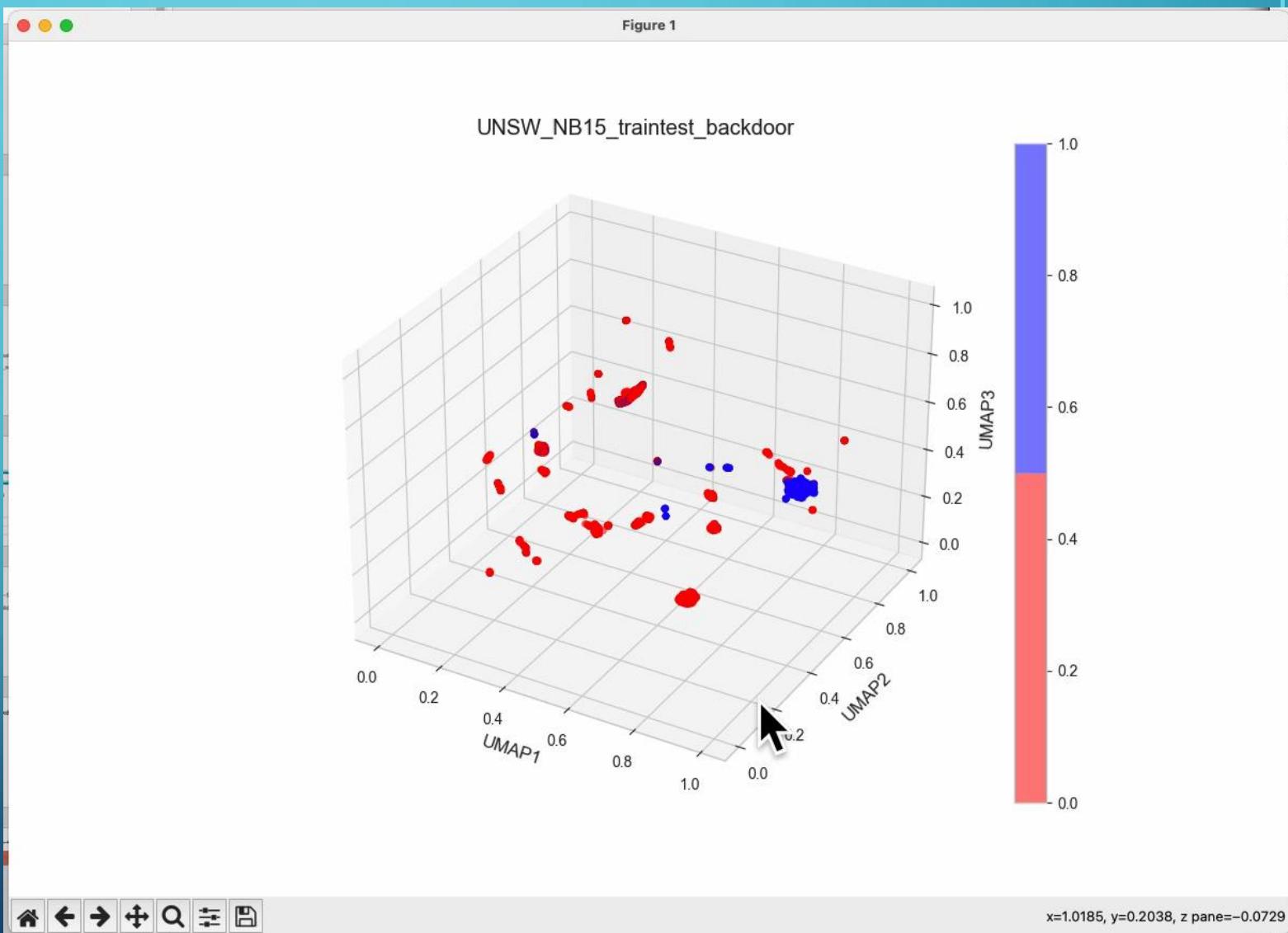
	pca_0	pca_1	pca_2	class
count	2000.000000	2000.000000	2000.000000	2000.000000
mean	0.459645	0.661148	0.317765	0.250000
std	0.285032	0.211464	0.192630	0.433121
min	0.000000	0.000000	0.000000	0.000000
25%	0.231782	0.559585	0.121226	0.000000
50%	0.357060	0.718988	0.363488	0.000000
75%	0.730834	0.792113	0.411498	0.250000
max	1.000000	1.000000	1.000000	1.000000

# データの詳細 - UNSW\_NB15\_TRAINTEST\_BACKDOOR

## UMAPの分布

- $n_{neighbors}=15$
- $\text{min\_dist}=0.1$

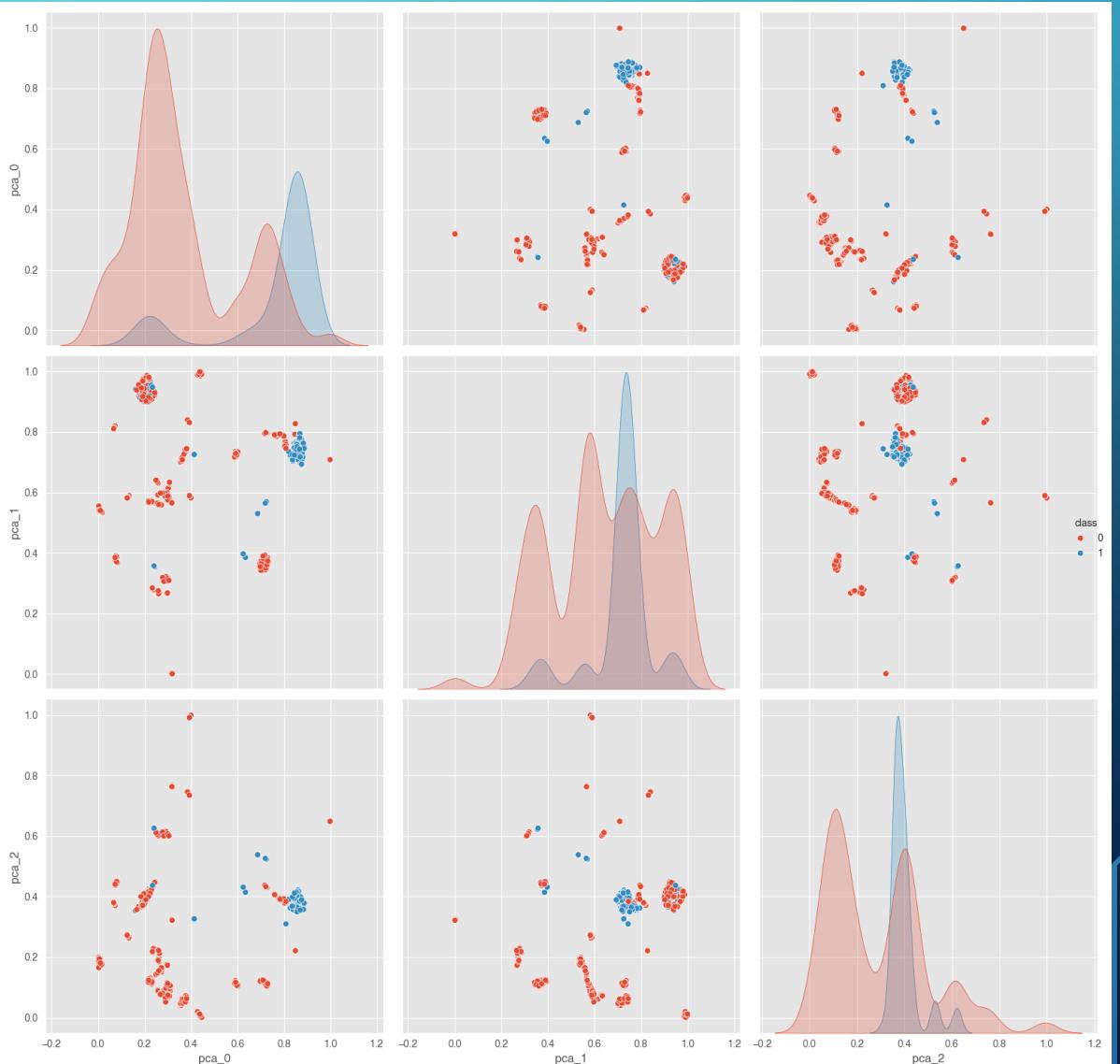
[凡例]  
0: 正常  
1: 異常



# データの詳細 - UNSW\_NB15\_TRAINTEST\_BACKDOOR

UMAPの分布

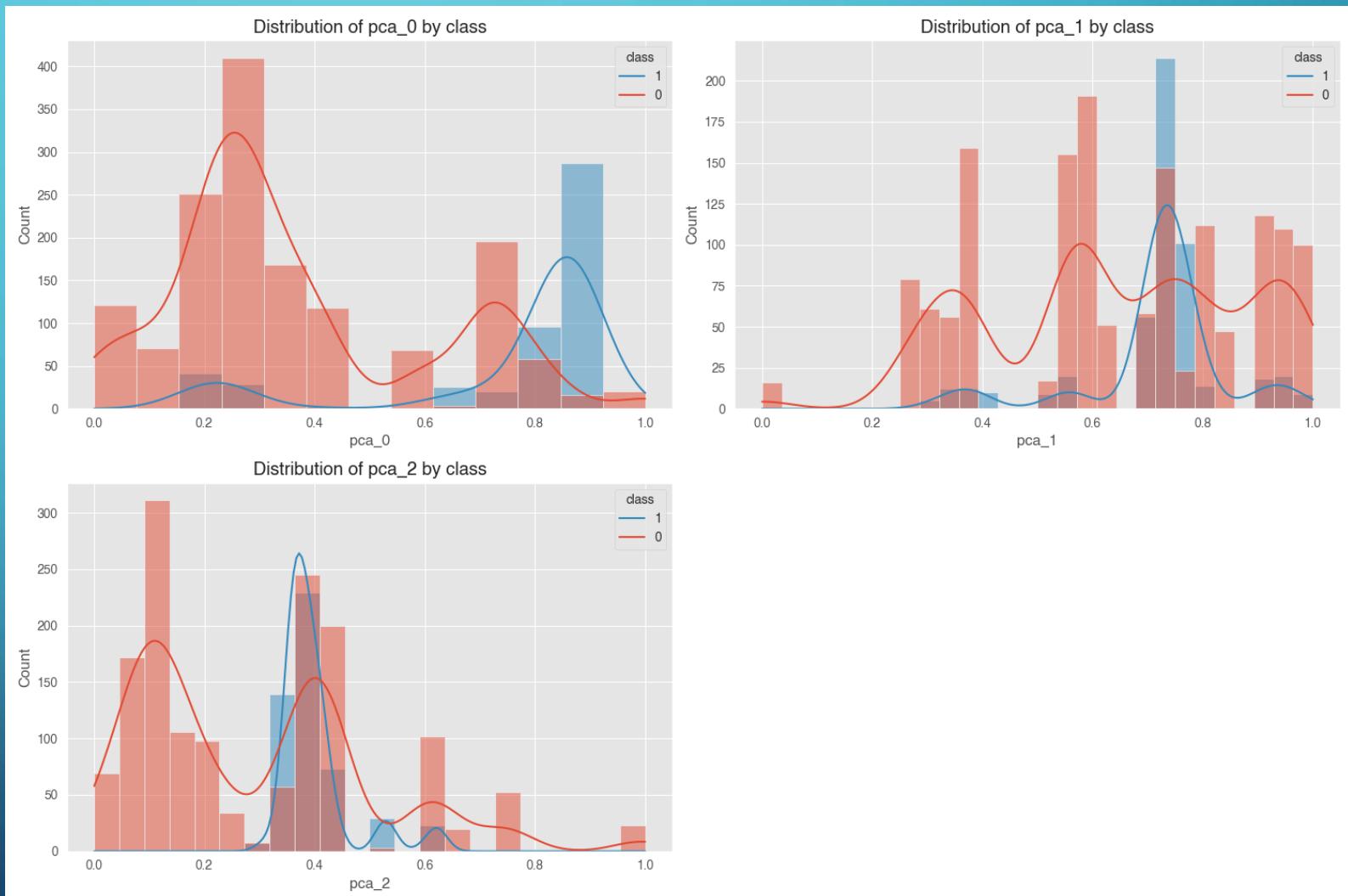
[凡例]  
0: 正常  
1: 異常



# データの詳細 - UNSW\_NB15\_TRAINTEST\_BACKDOOR

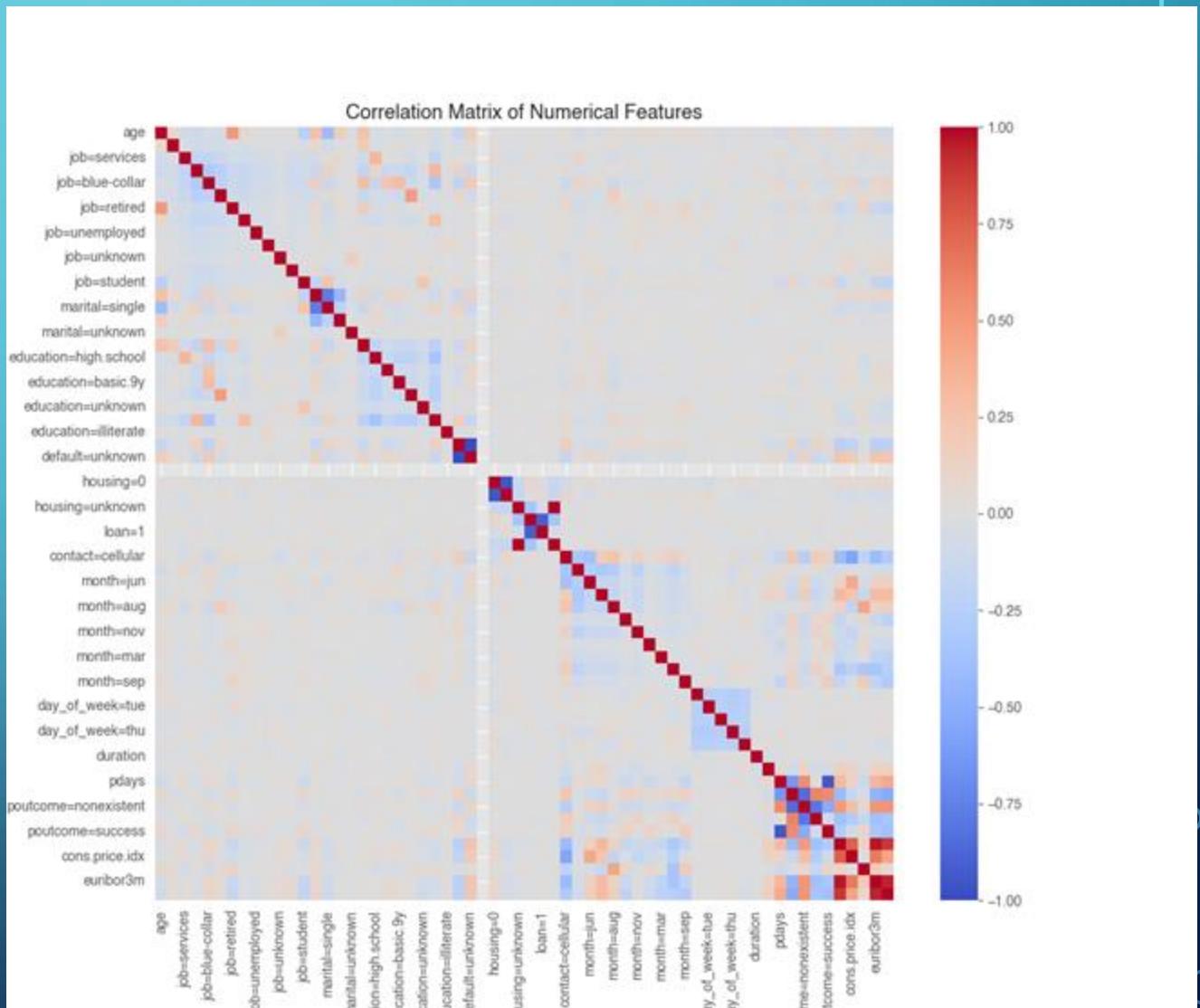
UMAPの分布

[凡例]  
0: 正常  
1: 異常



# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

## Correlation Plot



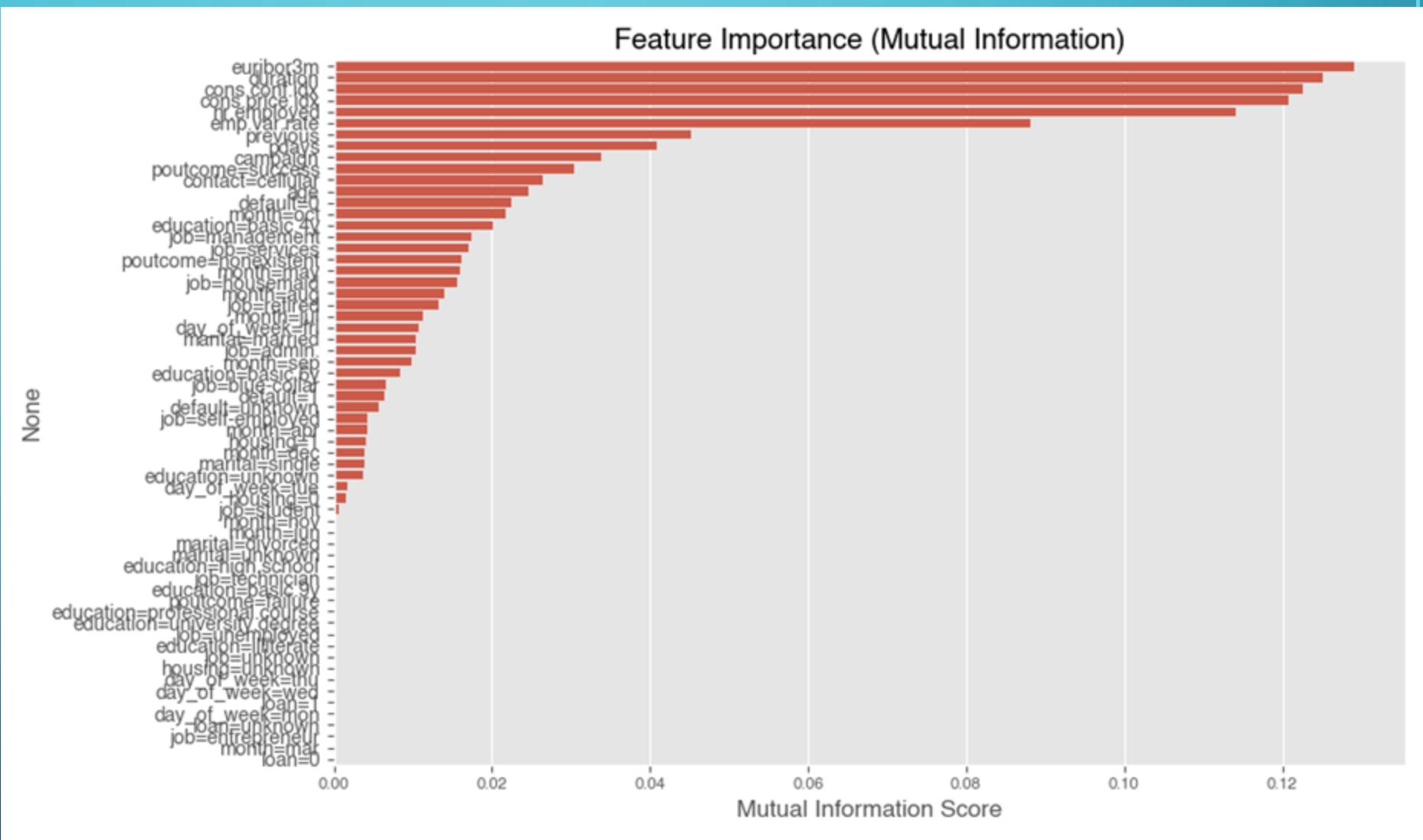
# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

UMAPデータ

	pca_0	pca_1	pca_2	class
count	2000.000000	2000.000000	2000.000000	2000.000000
mean	0.455378	0.541093	0.561940	0.250000
std	0.274075	0.287266	0.227873	0.433121
min	0.000000	0.000000	0.000000	0.000000
25%	0.241423	0.270677	0.385495	0.000000
50%	0.377932	0.548639	0.608431	0.000000
75%	0.717619	0.828222	0.727489	0.250000
max	1.000000	1.000000	1.000000	1.000000

# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

## Feature Importance

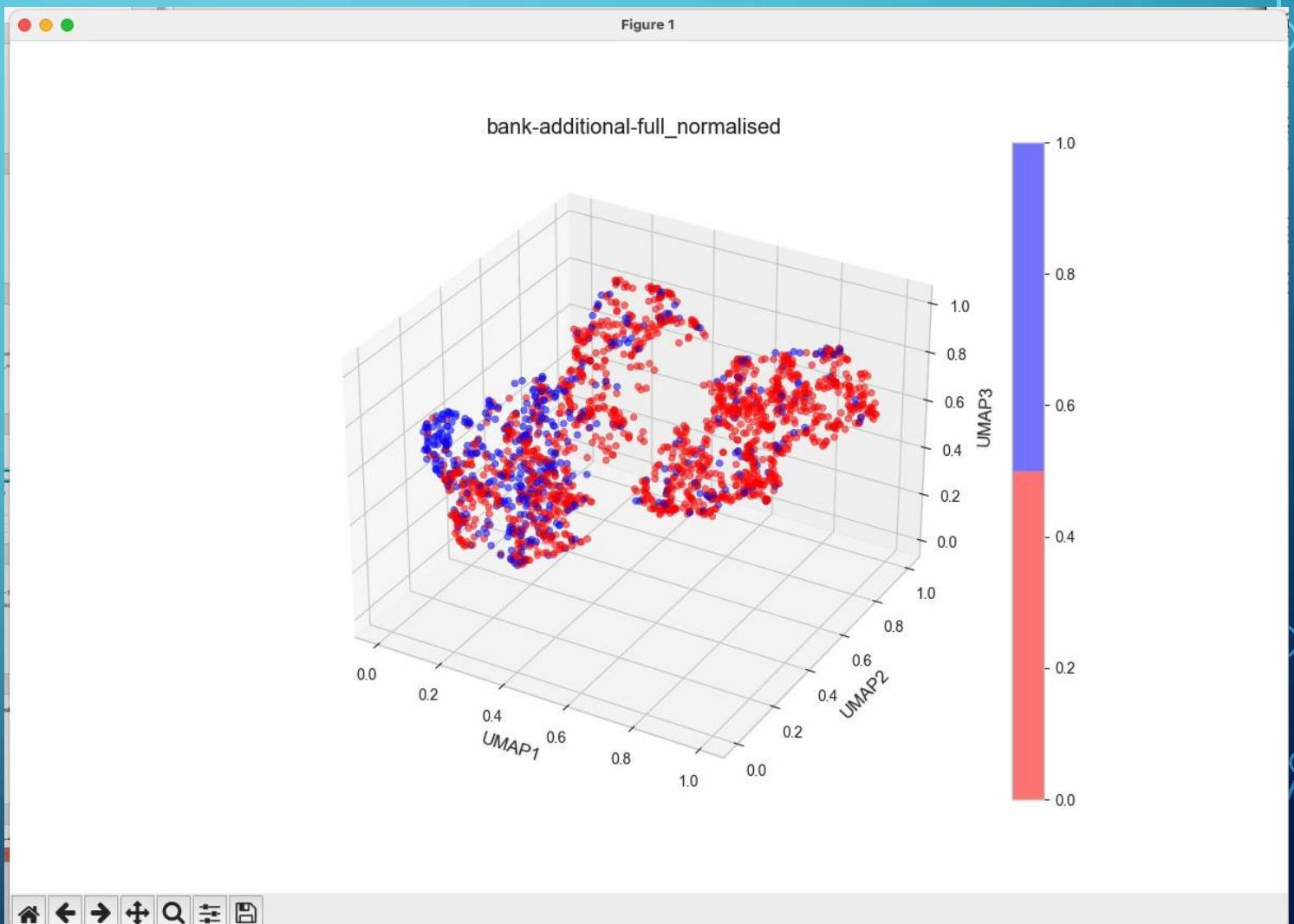


# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

## UMAPの分布

- n\_neighbors=15
- min\_dist=0.1

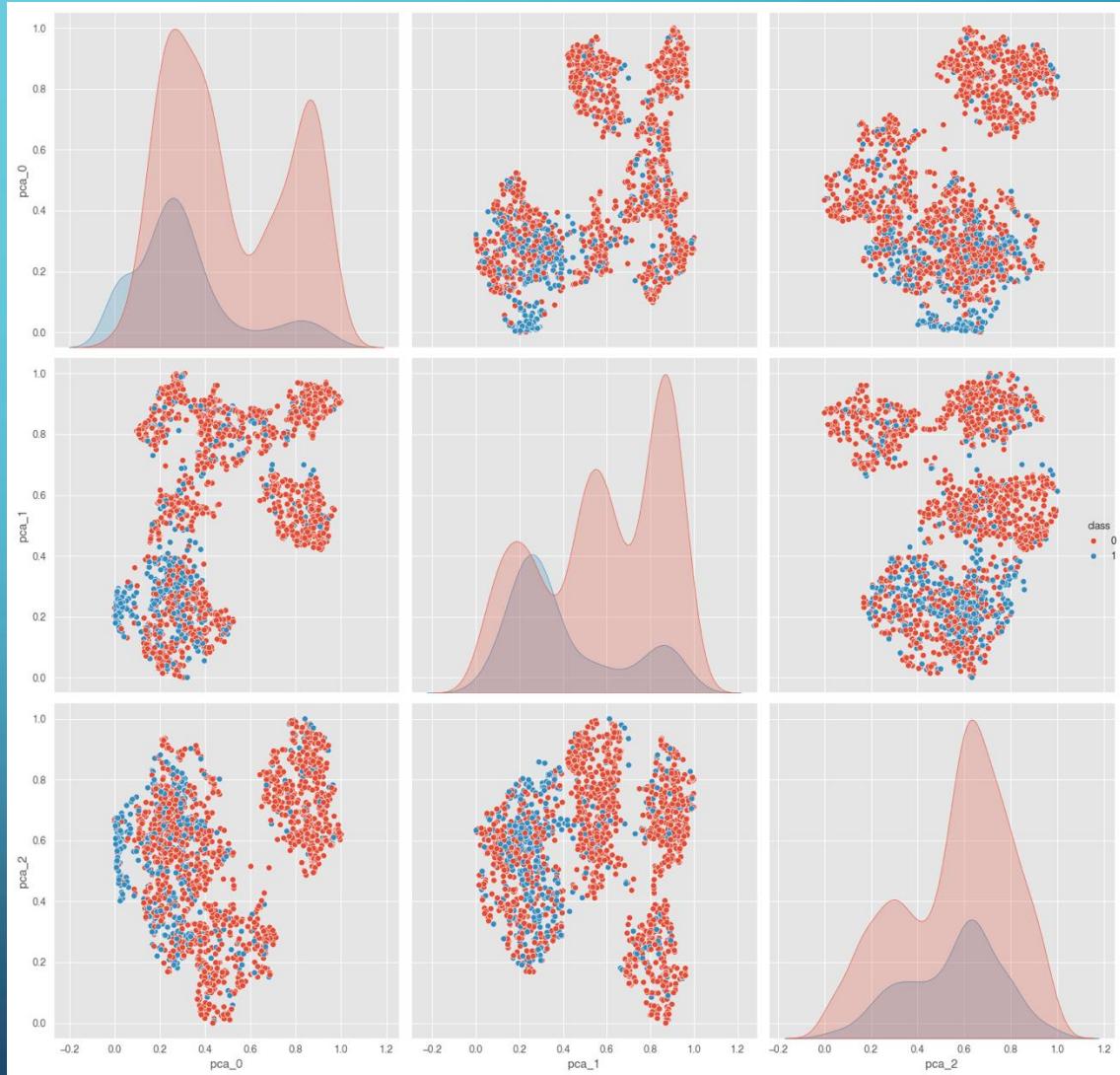
[凡例]  
0: 正常  
1: 異常



# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

UMAPの分布

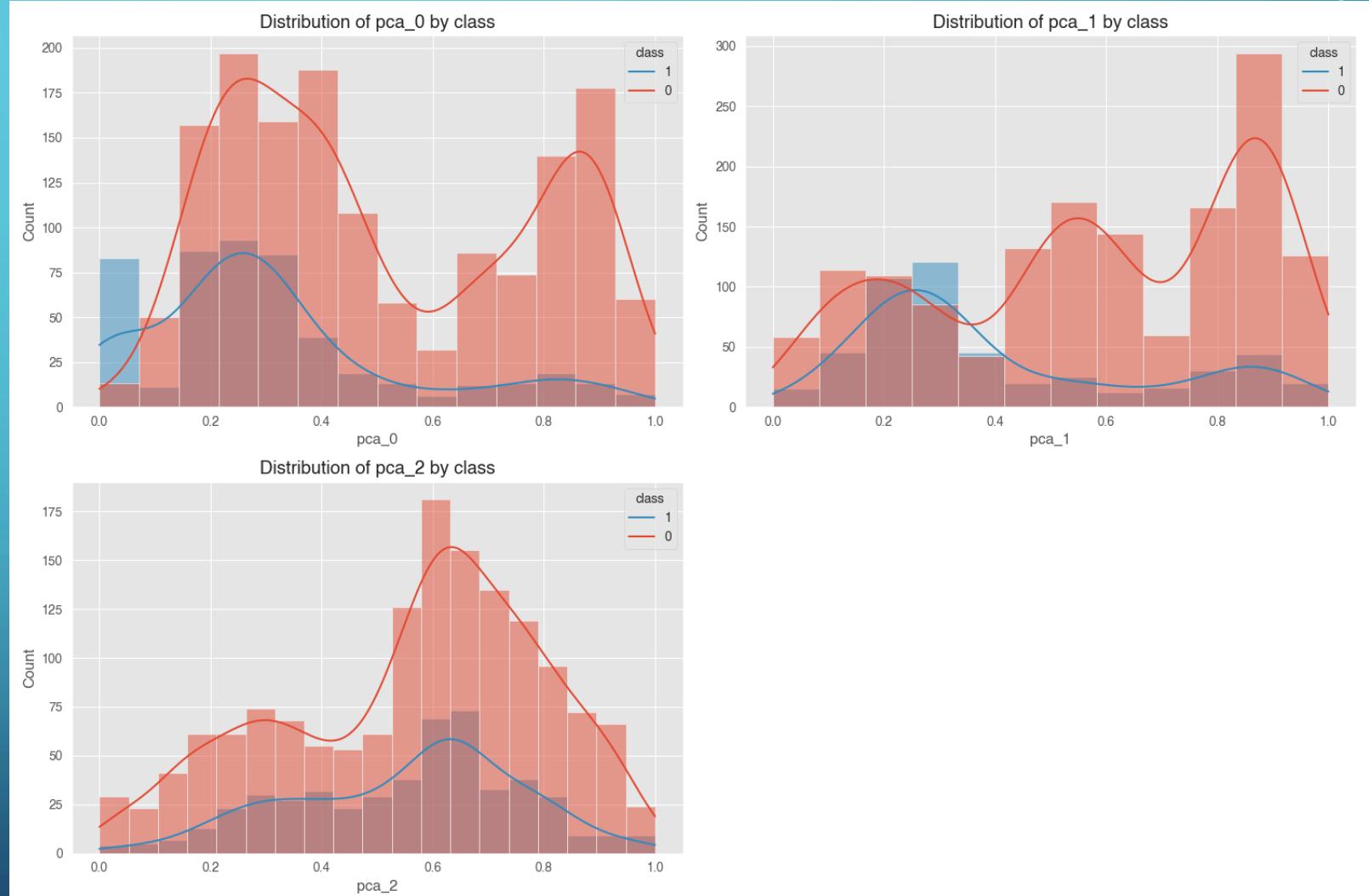
[凡例]  
0: 正常  
1: 異常



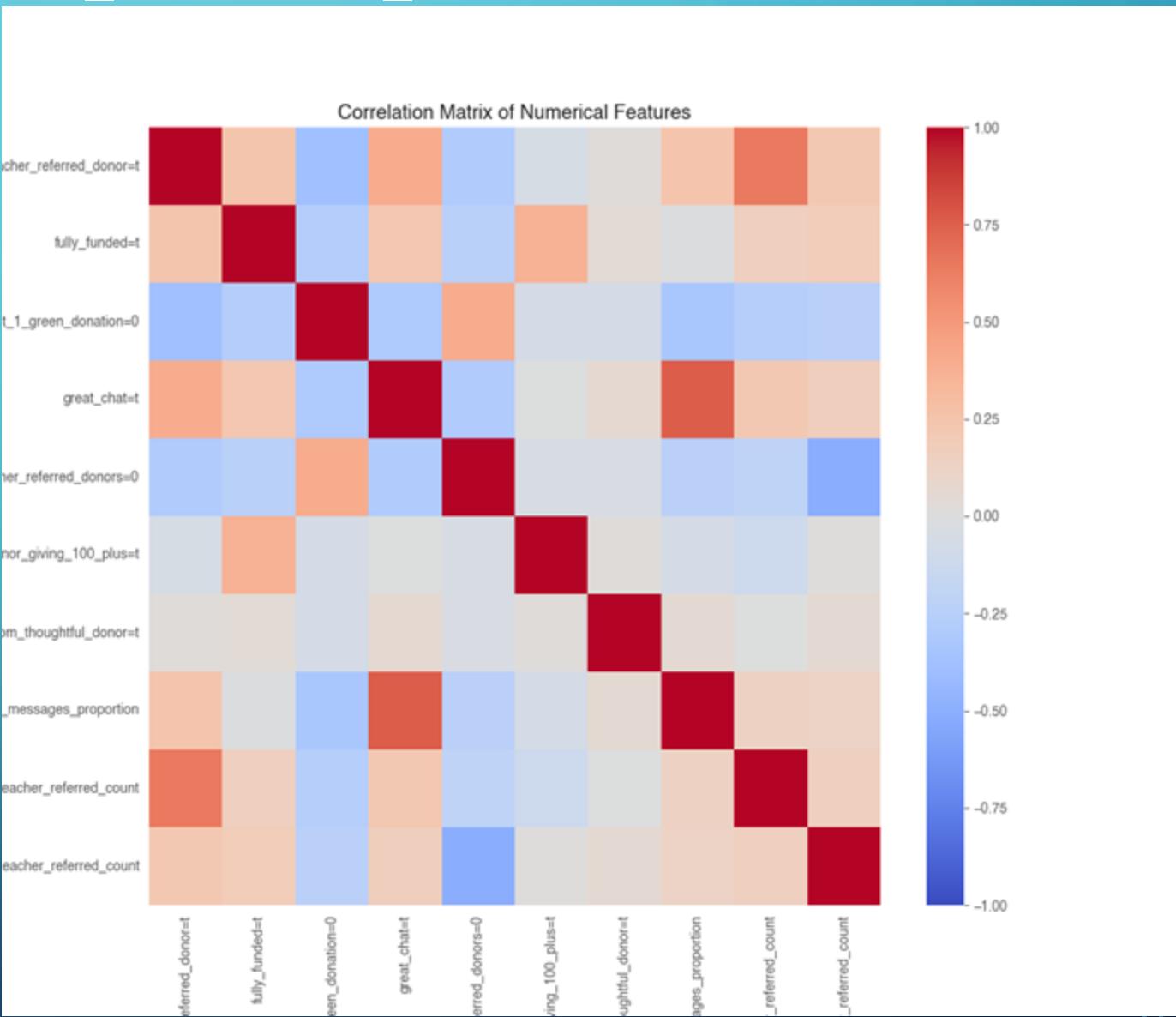
# データの詳細 - BANK-ADDITIONAL-FULL\_NORMALISED

UMAPの分布

[凡例]  
0: 正常  
1: 異常



# Correlation Plot



データの詳細 -

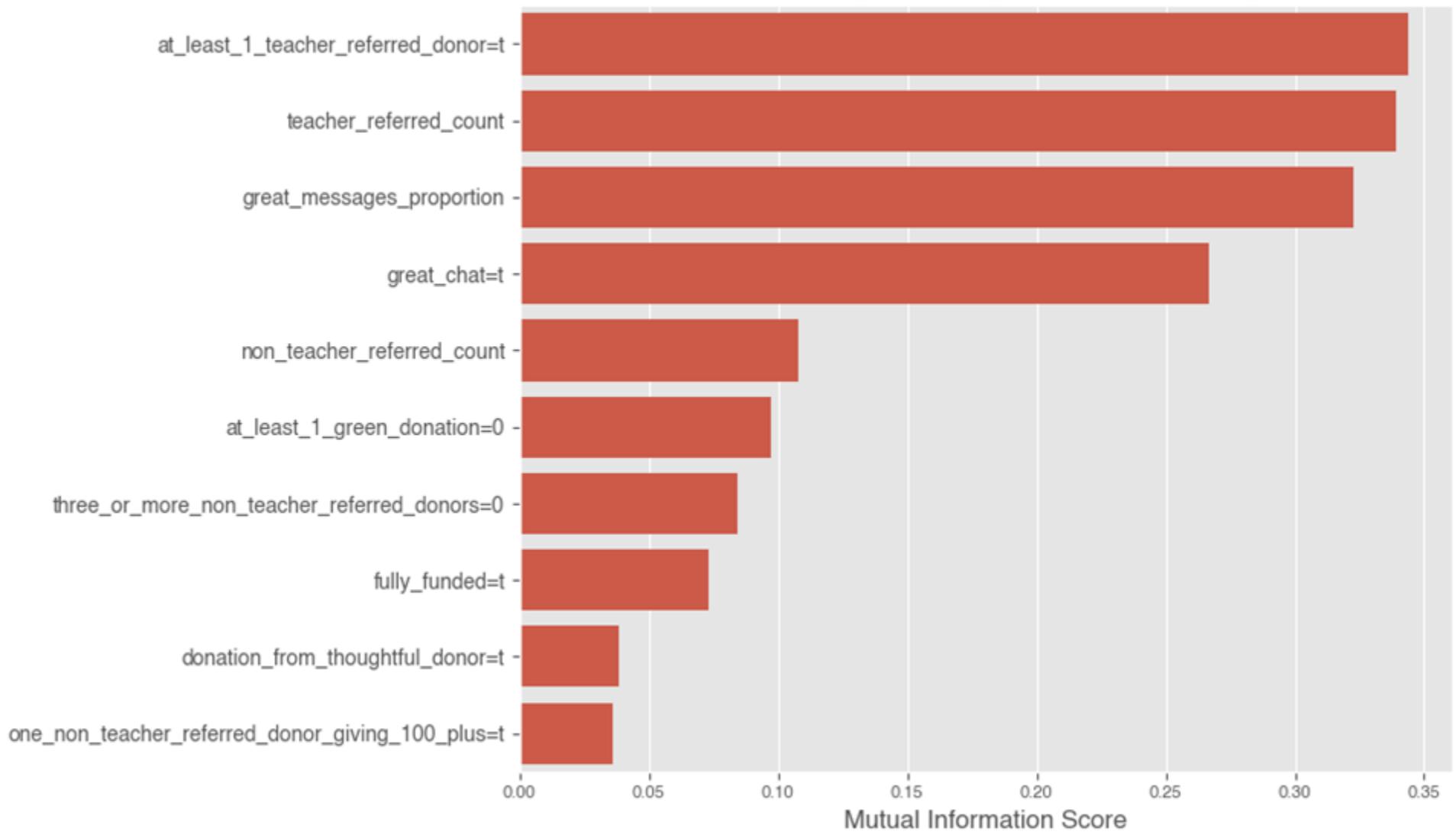
## KDD2014\_DONORS\_10FEAT\_NOMISSING\_NORMALISED

UMAPデータ

	pca_0	pca_1	pca_2	class
<b>count</b>	2000.000000	2000.000000	2000.000000	2000.000000
<b>mean</b>	0.530726	0.460741	0.480301	0.250000
<b>std</b>	0.264526	0.217918	0.237898	0.433121
<b>min</b>	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	0.347337	0.296187	0.353500	0.000000
<b>50%</b>	0.528842	0.529524	0.478351	0.000000
<b>75%</b>	0.756402	0.569472	0.685564	0.250000
<b>max</b>	1.000000	1.000000	1.000000	1.000000

## データの詳細 -

Feature Importance (Mutual Information)

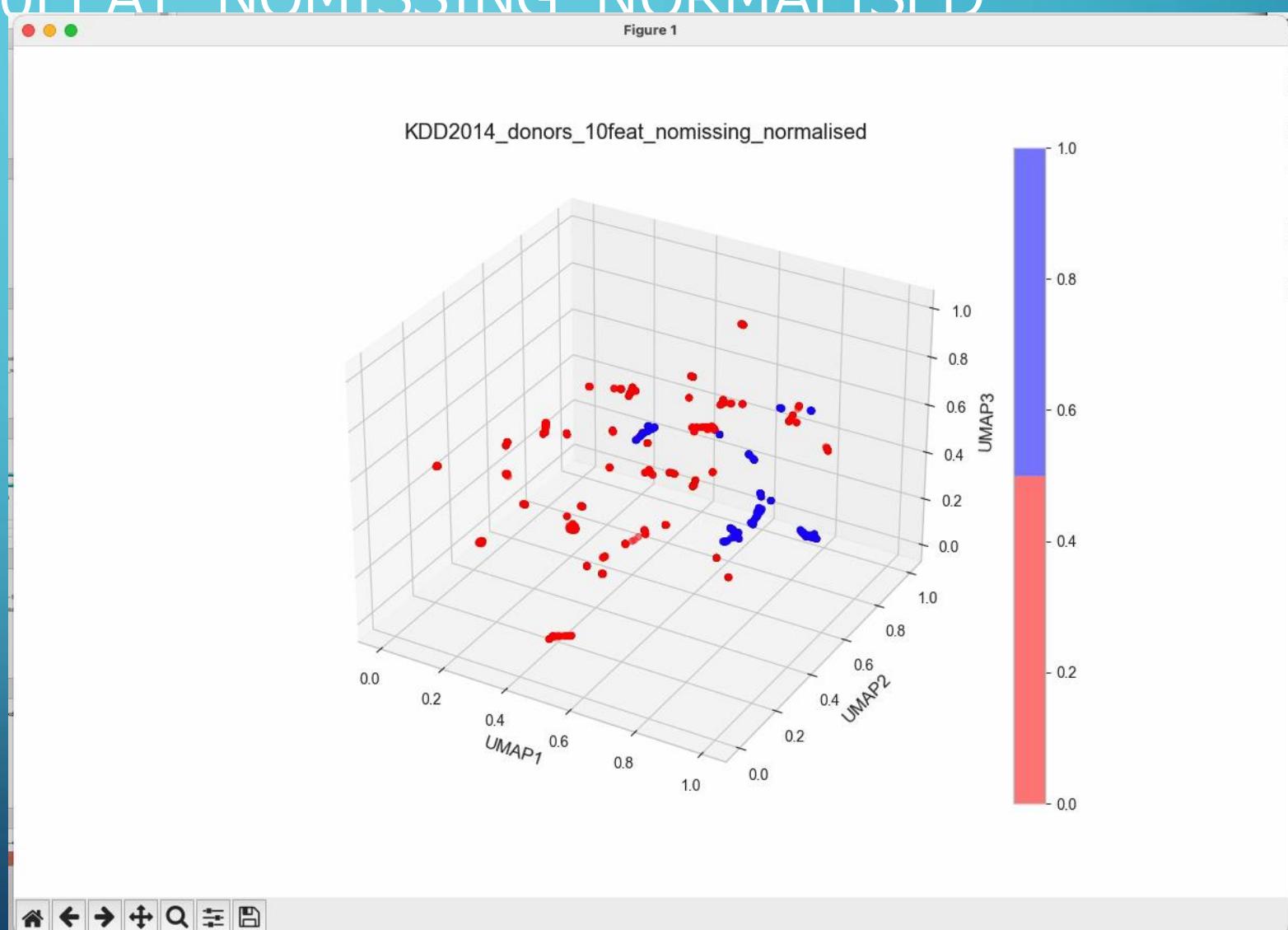


# データの詳細 -

KDD2014\_DONORS\_10FFAT NOMISSSTNG NORMAL TSFD

# UMAPの分布

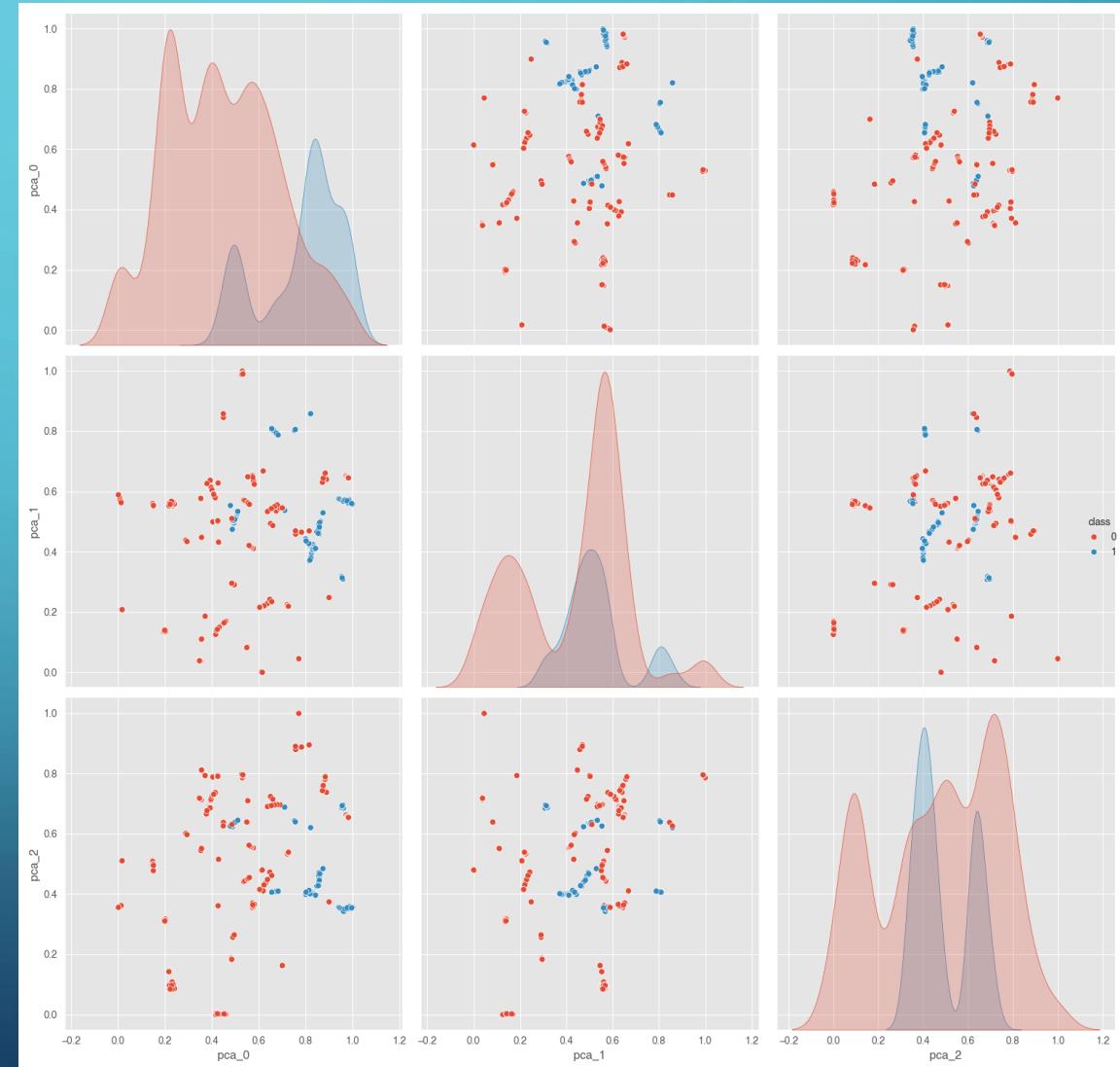
- n\_neighbors=15
  - min\_dist=0.1



データの詳細 -

# KDD2014\_DONORS\_10FEAT\_NOMISSING\_NORMALISED

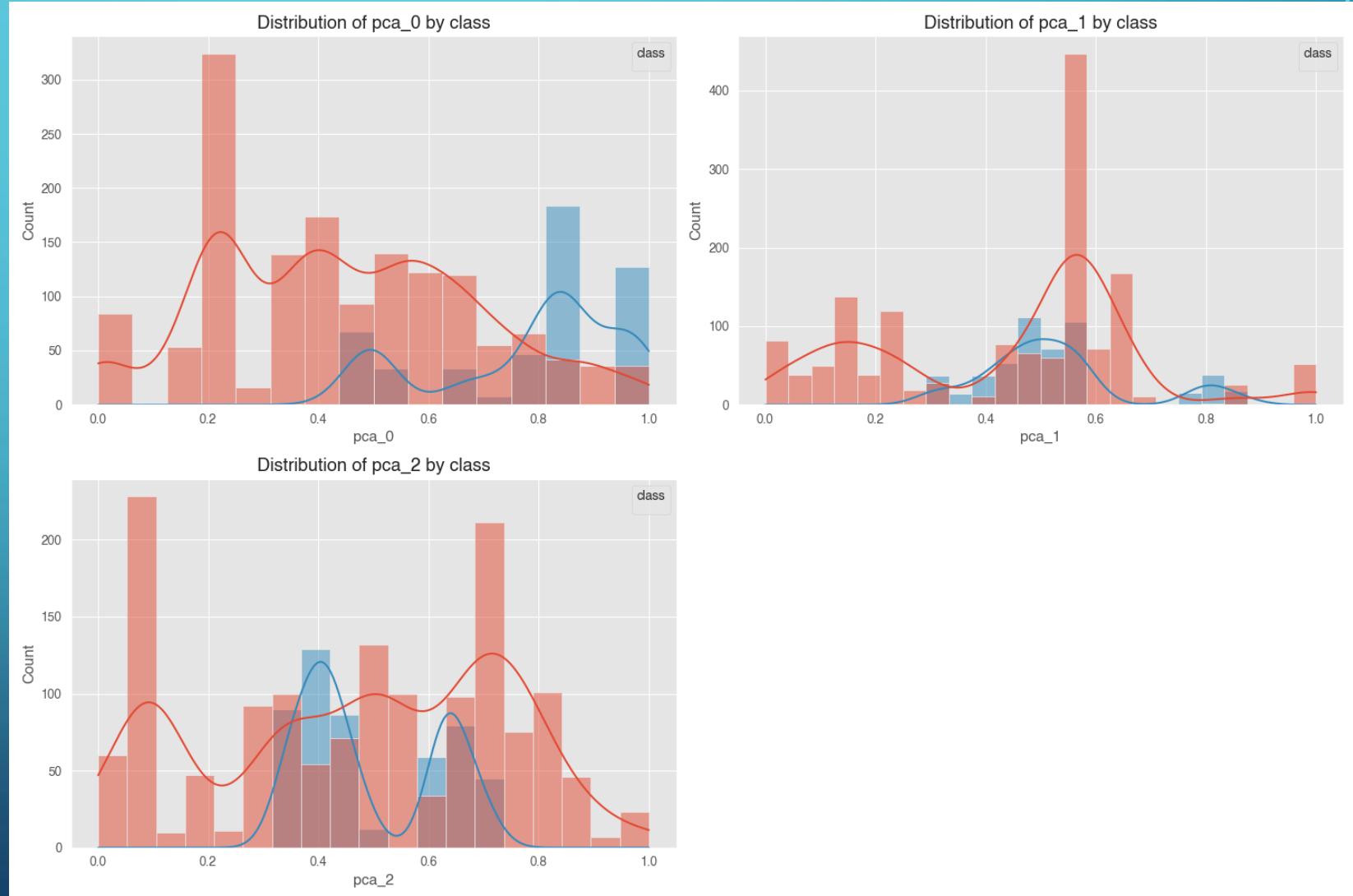
UMAPの分布



データの詳細 -

# KDD2014\_DONORS\_10FEAT\_NOMISSING\_NORMALISED

UMAPの分布



# 【特徴量マップにフォーカスしたチューニング】

## ①特徴量マップのみのチューニング（最高精度93%）

### 【チューニング方法・手順】

1、特徴量マップをチューニング（チューニング対象：特徴量マップのモデルと次元数、reps）

1-1、特徴量マップのモデルを選択

1-2、「1-1、」の次元数を選択

1-3、「1-1、」のreps(繰り返し)を選択

1-4、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す

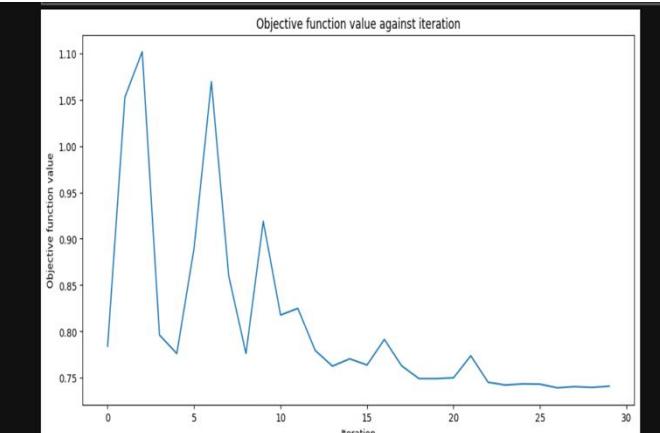
### 結果：

特徴量マップ→Ansatzのみのチューニングより、  
・精度が0.77→0.93に向上了（しかし、古典の0.97には及ばない）  
・特徴量マップ部分のみのチューニングによって0.90を超える精度向上が見られた

[特徴量マップ]  
モデル : ZFeatureMap  
次元数 : 3  
reps : 4

[Ansatz] ← デフォルト値  
モデル : RealAmplitudes  
エンタングルメントの作成方法 : reverse\_linear  
reps : 3

### チューニング前



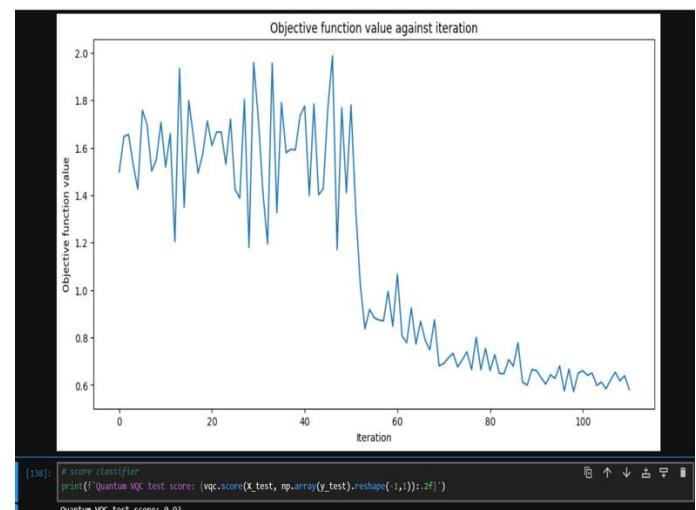
学習したモデルによるテストデータの分類  
学習したモデルを使用してテスト用データを評価します。

```
[29]: # score classifier
print("Quantum VQC test score: (%.2f)" % vqc.score(X_test, np.array(y_test).reshape(-1,1)))
Quantum VQC test score: 0.77
```

古典の精度 : 0.97

```
rbf kernel classification test score: 0.97
linear kernel classification test score: 0.96
poly kernel classification test score: 0.96
sigmoid kernel classification test score: 0.64
```

### 特徴量マップのチューニング後



精度が0.77 → 0.93に向上了

## ②特徴量マップ→Ansatzの順にチューニング（最高精度94%）

### 【チューニング方法・手順】

1、特徴量マップをチューニング（チューニング対象：特徴量マップのモデルと次元数、reps）

1-1、特徴量マップのモデルを選択

1-2、「1-1、」の次元数を選択

1-3、「1-1、」のreps(繰り返し)を選択

1-4、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す

2、Ansatzをチューニング（チューニング対象：Ansatzのモデルとreps）

2-1、「1、」の最も良いチューニング結果を基にAnsatzのエンタングルメントのモデルを選択

2-1、「2-1、」のモデルのrepsを選択

2-3、「2-1、」の全モデルに対して「2-1、」～「2-3、」を繰り返す

### 結果：

特徴量マップ→Ansatzの順にチューニングを行い、最終的な両者のチューニング値により、

・精度が0.77→0.94に向上（しかし、古典の0.97には及ばない）

・特徴量マップ部分のチューニングによる精度向上は見られたが、Ansatz部分のチューニングによる精度向上は見られなかった。

[特徴量マップ]

モデル：ZFeatureMap

次元数：3

reps：4

[Ansatz]

モデル：RealAmplitudes

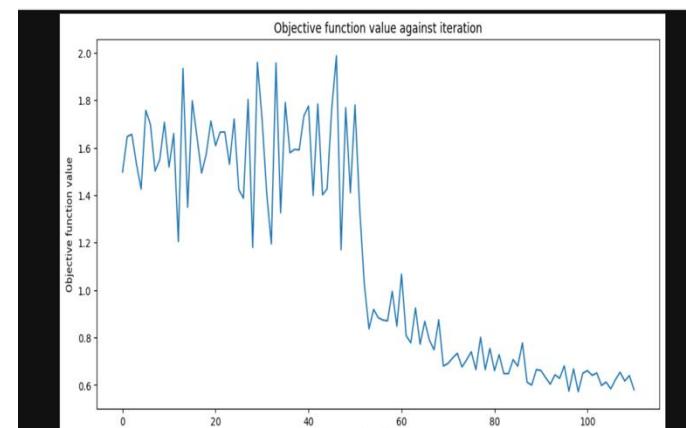
エンタングルメントの作成方法：linear

reps：3

古典の精度：0.97

```
rbf kernel classification test score: 0.97  
linear kernel classification test score: 0.96  
poly kernel classification test score: 0.96  
sigmoid kernel classification test score: 0.64
```

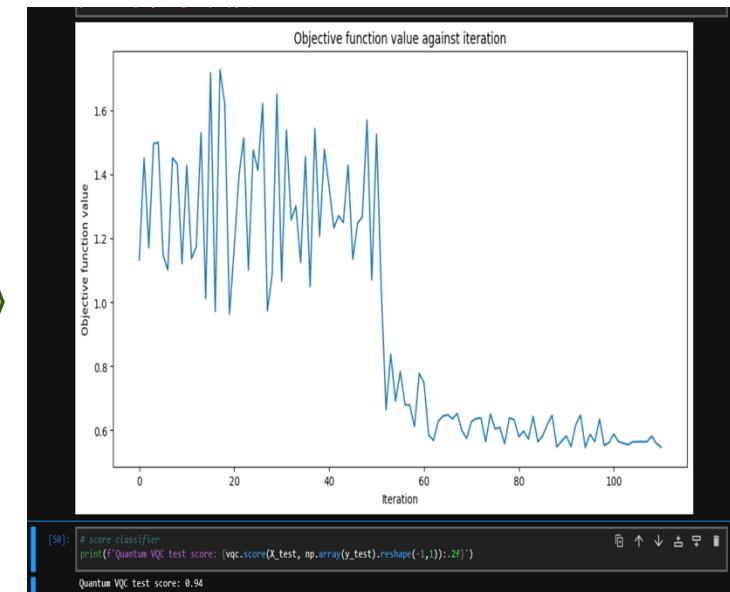
特徴量マップのみの  
チューニング結果



精度が0.93 → 0.94に向上

Ansatzのチューニングによって1%の精度向上しか示さなかった。

Ansatzのチューニング後



# 【Ansatzにフォーカスしたチューニング】

## ①Ansatzのみのチューニング（最高精度83%）

### 【チューニング方法・手順】

1. Ansatzをチューニング（チューニング対象：Ansatzのモデルとreps）

1-1. Ansatzのエンタングルメントのモデルを選択

1-1、「1-1、」のモデルのreps（繰り返し）を選択

1-3、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す

### 結果：

Ansatzのみのチューニングより、  
・精度が0.77→0.83に向上（しかし、古典の0.96には及ばない）  
・Ansatz部分のみのチューニングによって0.90を超える精度向上は見られなかった

[Ansatz]  
モデル：RealAmplitudes  
エンタングルメントの作成方法：sca  
reps: 3

[特徴量マップ] ← デフォルト値  
モデル：ZZFeatureMap

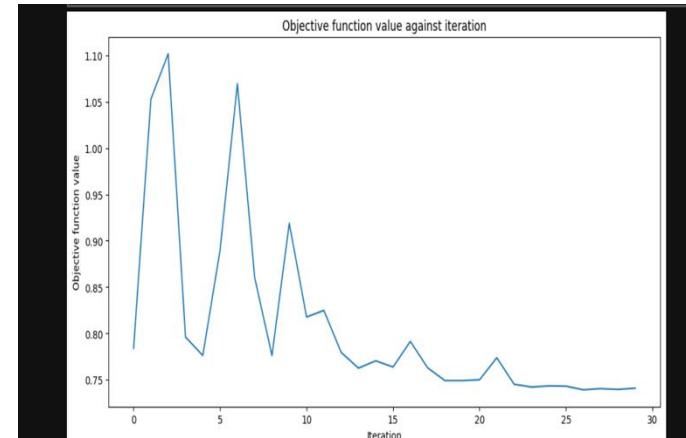
次元数: 2

reps: 1

古典の精度：0.96

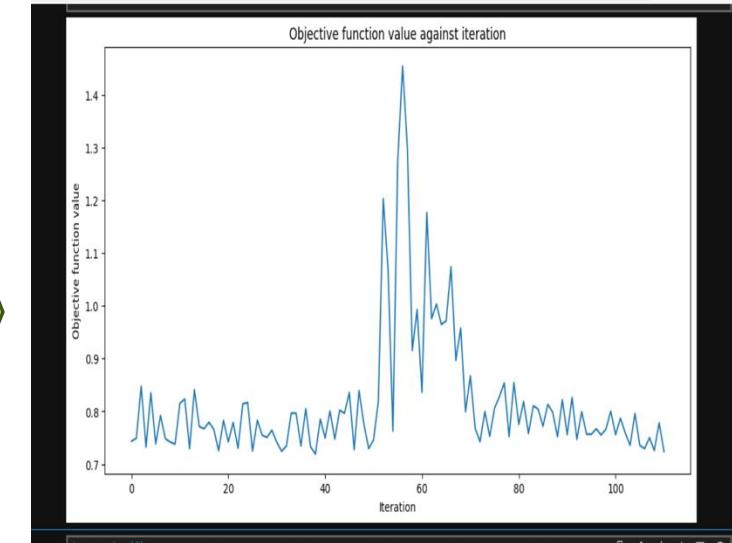
```
rbf kernel classification test score: 0.96
linear kernel classification test score: 0.93
poly kernel classification test score: 0.96
sigmoid kernel classification test score: 0.54
```

チューニング前



```
[29]: # score classifier
print(f'Quantum VQC test score: {vqc.score(X_test, np.array(y_test).reshape(-1,1));2f}')
Quantum VQC test score: 0.77
```

Ansatzのチューニング後



```
[190]: # score classifier
print(f'Quantum VQC test score: {vqc.score(X_test, np.array(y_test).reshape(-1,1));2f}')
Quantum VQC test score: 0.83
```

精度が0.77 → 0.83に向上

## ②Ansatz→特微量マップの順でチューニング（最高精度93%）

### 【チューニング方法・手順】

- 1、Ansatzをチューニング（チューニング対象：Ansatzのモデルとreps）
  - 1-1、Ansatzのエンタングルメントのモデルを選択
  - 1-1、「1-1、」のモデルのreps（繰り返し）を選択
  - 1-3、「1-1、」の全モデルに対して「1-1、」～「1-3、」を繰り返す
- 2、特微量マップをチューニング（チューニング対象：特微量マップのモデルと次元数、reps）
  - 2-1、「1、」の最も良いチューニング結果を基に特微量マップのモデルを選択
  - 2-2、「2-1、」の次元数を選択
  - 2-3、「2-1、」のrepsを選択
  - 2-4、「2-1、」の全モデルに対して「2-1、」～「2-3、」を繰り返す

### 結果：

Ansatz → 特微量マップの順にチューニングを行い、最終的な両者のチューニング値により、  
・精度が0.77→0.93に向上了（しかし、古典の0.97には及ばない）  
・Ansatz部分のチューニングによる精度向上は見られなかったが、特微量マップ部分のチューニングによる精度向上は見られた。

#### [Ansatz]

モデル：RealAmplitudes  
エンタングルメントの作成方法：sca  
reps : 3

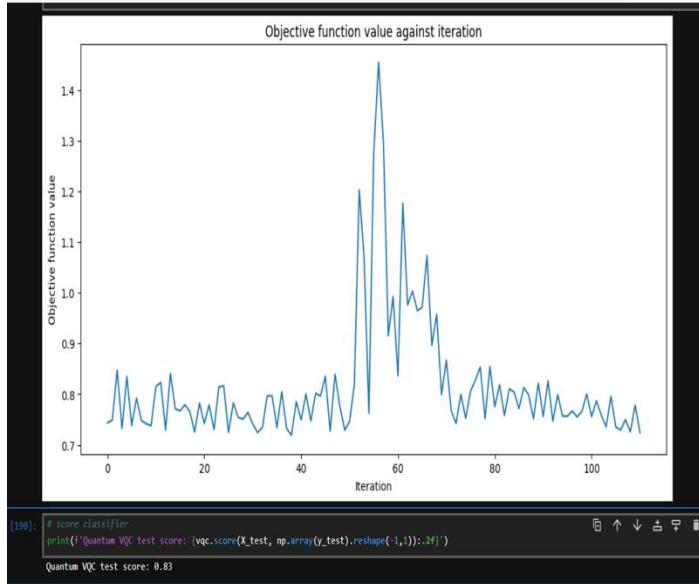
#### [特微量マップ]

モデル：ZFeatureMap  
次元数：4  
reps : 4

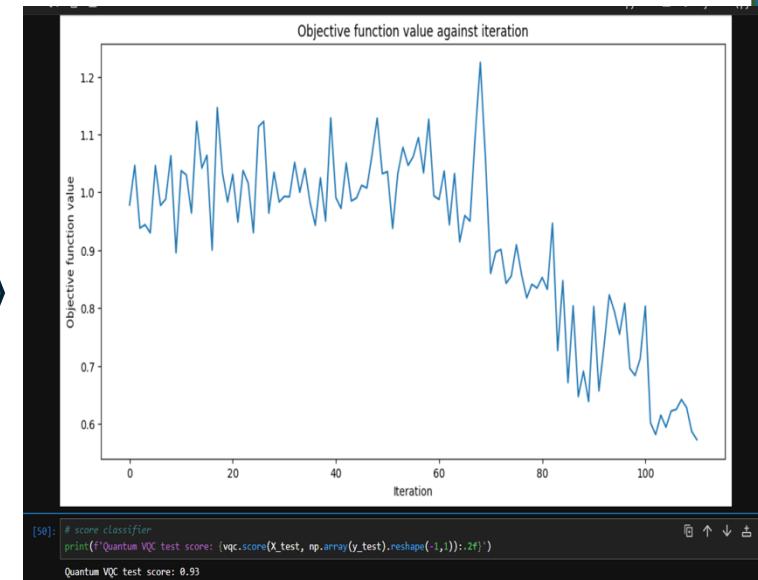
古典の精度：0.97

```
rbf kernel classification test score: 0.97
linear kernel classification test score: 0.96
poly kernel classification test score: 0.97
sigmoid kernel classification test score: 0.64
```

Ansatzのみの  
チューニング結果



特微量マップの  
チューニング後



精度が0.83 → 0.93に向上

特微量マップのチューニングによって10%の精度向上が示された。

# 【量子セルオートマトンと量子ウォーク】参考文献 - 1

## 【量子セルオートマトン/量子ウォーク】参考文献

- [1] 図で解る 量子ウォーク入門 - 町田拓也 / 森北出版
- [2] "Towards a Quantum Game of Life" Adrian P.Flitney & Derek Abbott  
[https://link.springer.com/chapter/10.1007/978-1-84996-217-9\\_23](https://link.springer.com/chapter/10.1007/978-1-84996-217-9_23)
- [3] Qiskit Camp - Hackaton Madrid 2019 - Quantum Game of Life  
<https://github.com/qonwaygameoflife/qonwaygameoflife>
- [4] HackNC 2023 Submission: Quame of Life  
<https://github.com/HatsuHodowa/HackNC-2023-Life-Game>
- [5] "The semi-quantum game of life" - D.A.Faux,1 M.Shah, and C.Knapp  
<https://arxiv.org/pdf/1902.07835>
- [6] 周期グラフ上の離散時間量子ウォーク(Qiskit)  
<https://qiita.com/notori48/items/00022f6c66c189a24836>
- [7] "Cellular Automata:Analysis and Applications" - Karl-Peter Hadeler Johannes Muller / Springer
- [8] セルオートマトン - Joel L.Schiff 共立出版
- [9] セルオートマトン法 - 加藤恭義・光成友孝・葉山洋 / 森北出版
- [10] Pythonによる数値計算とシミュレーション- 小高知宏 / Ohmsha
- [11] "Quantum Cellular Automata: Implementing an Elementary Cellular Automata on a QC" - Richard Beattie  
<https://medium.com/mit-6-s089-intro-to-quantum-computing/quantum-cellular-automata-implementing-an-elementary-cellular-automata-on-a-qc-736ea944042>
- [12] ライフゲームの宇宙- ウィリアム・パウンドストーン / 日本評論社
- [13] ガイドツアー 複雑系の世界 - メラニー・ミッケル / 紀伊国屋書

# 【量子セルオートマトンと量子ウォーク】 参考文献 - 2

【量子セルオートマトン/量子ウォーク】参考文献

[14] Matrix Product State (MPS)とはなんぞや？

<https://hackmd.io/@GM3D/SJefWhpKj>

[15] IBM-Q の53qubit量子コンピュータのシミュレータを作ってみた

<https://qiita.com/j-i-k-o/items/6b7244896791b3fc6655>

[16] ISC 2024に参加・発表しました #3 ~ここまで来た！ 量子技術最前線～

<https://blog.fltech.dev/entry/2024/06/21/isc2024-3-ja>

[17] 量子セルオートマトンに基づく画像圧縮のための画像変換アルゴリズム

<https://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1744-26.pdf>

[18] 応用オートマトン工学 - 西野哲朗 若月光男 後藤隆彰 / コロナ社

[19] "A review of Quantum Cellular Automata" - Terry Farrelly

<https://arxiv.org/abs/1904.13318>

[20] "Quantum cellular automata for quantum error correction and density classification"

T.L.M.Guedes, D.Winter, M.Muller

<https://arxiv.org/pdf/2309.03608>

[21] "FROM QUANTUM CELLULAR AUTOMATA TO QUANTUM LATTICE GASES" - David A.Meyer

<https://arxiv.org/abs/quant-ph/9604003>

[22] テンソルネットワーク入門 - 西野友年 / 講談社

[23] ランダム・ウォーク - 津野義道 / 牧野書店

[24] ランダムウォークはじめの一歩 - J.Klafter I.M.Sokolov

# 【量子セルオートマトンと量子ウォーク】 参考文献 - 3

【量子セルオートマトン/量子ウォーク】参考文献

[25] Implementation of Quantum Walks on Cycle Graph

[https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/terra/qis\\_adv/quantum\\_walk.ipynb](https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/terra/qis_adv/quantum_walk.ipynb)

[26] 量子ウォークによる探索アルゴリズム

[https://github.com/wg-quantum/2024-b-06/blob/main/QW/quantum-walk-search-algorithm\\_qiskit10\\_basicSimulator.ipynb](https://github.com/wg-quantum/2024-b-06/blob/main/QW/quantum-walk-search-algorithm_qiskit10_basicSimulator.ipynb)

[27] "Quantum Walks on Graphs" - Aharonov, Ambainis, Kempe, and Vazirani

<https://arxiv.org/abs/quant-ph/0012090>

[28] "Spatial Search by Quantum Walks" - Childs and Goldstone

<https://arxiv.org/abs/quant-ph/0306054>

[29] 量子ウォークの数理 - 今野紀雄 産業図書

[30] 量子探索 - 今野紀雄 / 近代科学社

[31] 量子ウォーク - 今野紀雄 / 森北出版

[32] セルオートマトン入門 - 磯川貞二郎, 今井勝信, 梅尾浩, ペパー・フェルディナンド

[https://www.jstage.jst.go.jp/article/isciesci/63/7/63\\_264/\\_pdf/-char/ja](https://www.jstage.jst.go.jp/article/isciesci/63/7/63_264/_pdf/-char/ja)

[33] Universality in Elementary Cellular Automata - Matthew Cook

[https://www.complex-systems.com/abstracts/v15\\_i01\\_a01/](https://www.complex-systems.com/abstracts/v15_i01_a01/)

EoF