

# 応用レベル:テキスト

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もありません。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もありません。

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ら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独自の決定権をもっ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もありません。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。

## 1 量子機械学習とは

- 量子機械学習: Quantum Machine Learning、QML
- 広い意味では、「機械学習と量子情報の相乗効果を用いたアプローチ全般」を指す。
- 狭義には、次のいずれかである。
  - ・ 量子コンピュータを用いた機械学習
  - ・ 量子の知見に基づいた機械学習
- 直感的には、従来の機械学習手法に対して量子の理論および特性を適用することにより、新たな優位性(例えば計算量の削減)を見出そうというものである。

機械学習

+

量子

- ・ 量子の知見
- ・ 量子コンピュータ

=

量子機械学習

## 2 機械学習における量子コンピュータの可能性

- 従来の機械学習分野における「ブレークスルー」は、計算能力の向上やデータセットの規模の拡大に基づいた成果がほとんどである。言い換えると、根本的に新しい手法が開発されたことによるものではない。
- ニューラルネットワークや深層学習などの最新の手法は基本的に1990年代以前に発明されており、新しいものではない。多くの機械学習のタスクは難しい最適化問題へと変換されてしまうため、根本的に新しい手法を見つけ出すことは困難。
- この問題を解決するためには、コンピュータは解を含む広大な領域から手当たり次第に最適な解の候補を見つけ出さなければならぬ。これは量子計算の活用を考える上で興味深いポイントであり、量子計算が機械学習手法の開発を大きく前進させる可能性を秘めている。

## 3 参考資料

[「量子コンピュータによる機械学習」\(共立出版\)](#)

[「A Few Useful Things to Know about Machine Learning」\(PDF\)](#)

## 1 4つのアプローチ

機械学習への量子適用には、従来の手法を含めて4つのアプローチが考えられる。

		データ処理デバイス	
		C	Q
データを生成する系	C	①CC	③CQ
	Q	②QC	④QQ

C: 古典  
Q: 量子

- ① CCアプローチ  
機械学習における従来のアプローチ。  
データ生成、データ処理ともに古典処理にて行う。
- ② QCアプローチ  
データ生成を量子で行う。  
機械学習が量子計算にどのように役立つか調べるもの。
- ③ CQアプローチ  
データ処理を量子で行う。  
主な課題はデータマイニングのための量子アルゴリズムの設計。
- ④ QQアプローチ  
データ生成、データ処理ともに量子で行う。  
研究成果は現在のところ**わずかしか発表されていない**。

## 2 CQアプローチにおける2つの方策

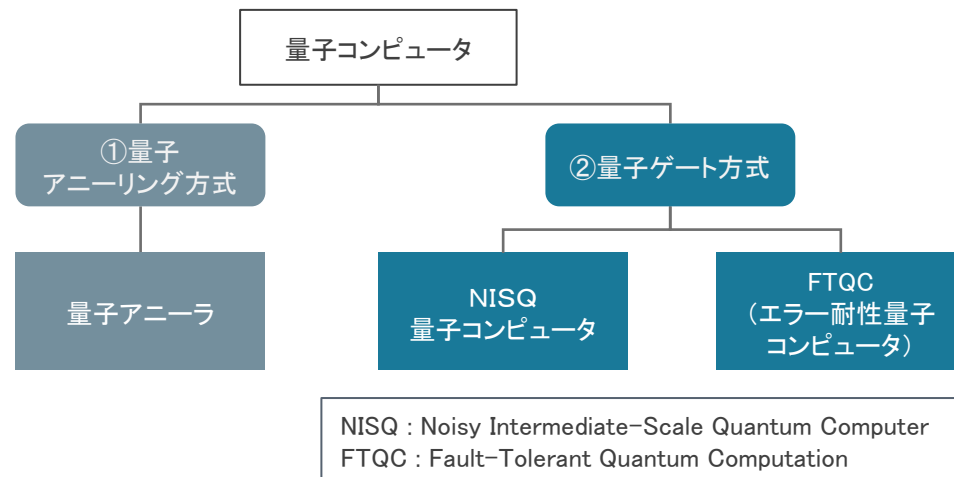
- 翻訳的アプローチ  
古典による機械学習のモデルを量子力学の言語に翻訳して**アルゴリズムの高速化**を目指すアプローチ。  
ニューラルネットワークなどのモデルの結果を再現し、計算全体もしくは一部を量子デバイスに委託する。  
量子加速性により、**既存アルゴリズムの高速化**をねらう。
- 探索的アプローチ  
量子コンピュータの利用を前提としたとき、どのタイプの機械学習モデルがその物理学的な特性や制約、形式、言語、提唱されている利点に適合するかを問うもの。量子デバイスオリジナルの学習モデルであり、古典アルゴリズムにはないものとなる。  
この方策が持つさまざまな潜在的な可能性は**未だ解明されていない**。

## 3 参考資料

[「量子コンピュータによる機械学習」\(共立出版\)](#)  
[QQアプローチの例:「Quantum-enhanced machine learning」](#)  
[JGS2020 量子コンピュータ\(PX-103\) 発表資料\(KMAP内\)](#)

## 1 量子機械学習手法のアプローチ

量子コンピュータにはいくつかの種類があり、大きくは次のように分類できる。



量子アニーリング方式〔[基礎レベルテキスト4-3](#)〕の量子コンピュータは [組合せ最適化](#)〔[基礎レベルテキスト 5-1](#)〕に特化しているため、量子機械学習は [量子ゲート方式](#)〔[基礎レベルテキスト4-2](#)〕に適用される手法となる。

量子ゲート方式は、さらにNISQとFTQC（エラー耐性量子コンピュータ）に分類することができる。量子コンピュータへの期待の中心はFTQCにあるが、ハードウェアは開発途上であり、FTQCの実現にはまだ時間がかかると考えられている。そこで現在はNISQを対象とした研究が行われており、量子機械学習アルゴリズムについても同様にNISQ向けの研究が行われている。

## 2 NISQ向けアルゴリズム

NISQ向けアルゴリズムで考慮すべきは、「計算過程で生じるエラーへの対策」である。（FTQCでは計算過程で生じるエラーが訂正されることから、エラーを気にせずアルゴリズムを構築することができるが、NISQでは同じようにはいかない。）

エラー対策手法には、例えば「計算過程で生じるエラー（ノイズ）を統計的な処理により低減させる」といったものがある。こうしたさまざまな工夫により、NISQ向けのアルゴリズムを実現している。

以下はNISQ向けアルゴリズムの例である。

- [変分量子固有値ソルバー（VQE）](#) 〈量子化学計算〉
- [量子近似最適化アルゴリズム（QAOA）](#) 〈最適化〉
- [量子回路学習（QCL）](#) 〈量子機械学習〉

FTQCが実現されるまでの間は、NISQ向けアルゴリズムを使って現実課題を解くことが量子コンピュータ普及のための一つのミッションとなっている。

## 3 参考資料

[IPA 2020年度未踏ターゲット事業「機械学習を用いたNISQアルゴリズム向けの誤り補償手法の開発」\(PDF\)](#)

- 機械学習および量子機械学習に関する手法を一覧で整理する。あわせて、量子機械学習を適用することによるメリットを整理する。

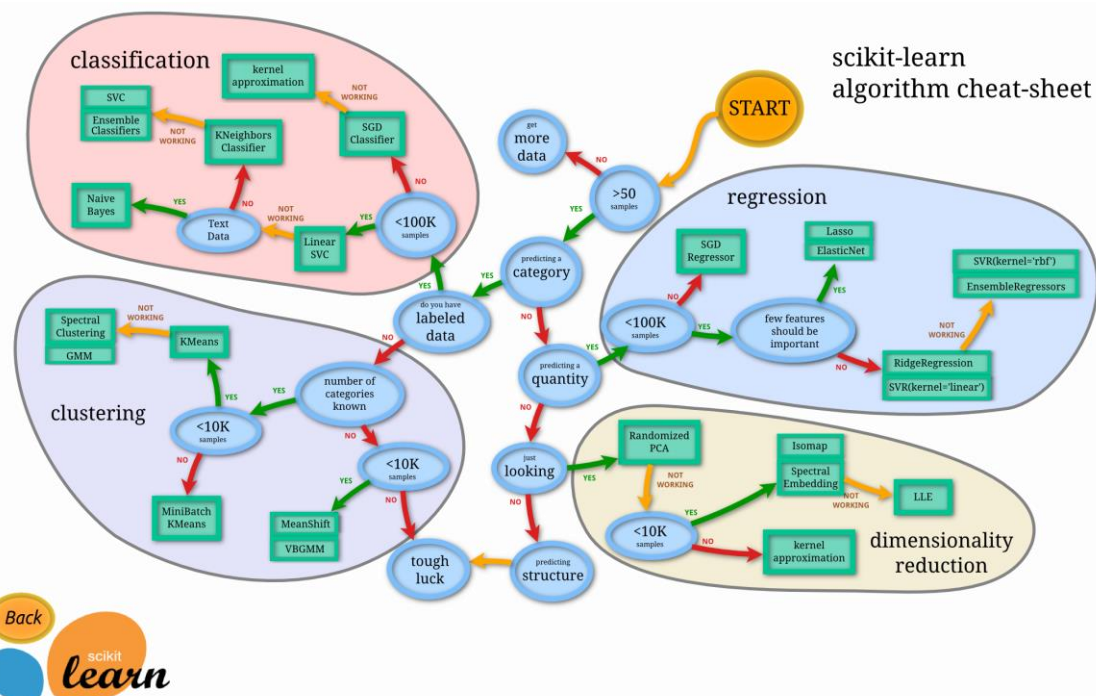
カテゴリ	古典機械学習手法	量子機械学習への応用有無	量子機械学習の適用によるメリット
教師あり学習（回帰）	線形回帰	線形回帰	
	サポートベクター回帰（SVR）	—	
教師あり学習（分類）	ロジスティック回帰	—	
	サポートベクターマシン（SVM）	QSVM	量子回路学習と同じ。
	最近傍法、k近傍法	—	
	ニューラルネットワーク	量子ニューラルネットワーク、量子回路学習	重ね合わせの原理による高次元の特徴量空間が利用可能で、古典コンピュータ上で効率的に表現しにくい非線形モデルも効率的に構築できると期待される。またユニタリー変換のみでモデルを構築することがある種の正則化として機能し、過学習が抑えられる。 <出典：嶋田義皓「量子コンピューティング」p.158>
	畳み込みニューラルネットワーク（CNN）	量子畳み込みニューラルネットワーク	量子力学の重ね合わせの原理から、指数関数的に増える多数の計算基底を使って状態を表現できることが量子コンピュータの強みです。この強みを生かすことで、データ間の複雑な相関を学習できる可能性が生まれます。そこに量子機械学習の最も大きな強みがあると考えられています。 <出典： <a href="#">東京大学素粒子物理国際研究センター「量子コンピューティング・ワークブック」</a> >
	回帰ニューラルネットワーク（RNN）	—	利点不明。学習面や予測精度の面で量子回路がCNNを上回るのは難しそう。 <出典：嶋田義皓「量子コンピューティング」p.164>
	決定木	—	
	ランダムフォレスト	—	
教師なし学習	主成分分析（PCA）	量子主成分分析	
	k-means（k平均法）	量子k-means、Q-means	出力を得るために必要とされる操作が、1回のアダマールと2回の測定のみ。これは入力ベクトルやデータセットの規模に関係なく有効。例えば、データセットがサイズ100万の訓練ベクトル10億個で構成される場合でも3つの演算にかかる実行時間が同程度である定数時間アルゴリズムが存在するだろう。 <出典：M. Schuld他「量子コンピュータによる機械学習」p.21>
	ガウス混合モデル	ガウス混合モデル	

カテゴリ	古典機械学習手法	量子機械学習への応用有無	量子機械学習の適用によるメリット
強化学習	GAN	QGAN	古典モンテカルロ数値積分計算より、量子モンテカルロ積分は効率的。(参考: <a href="#">S. Woerner他「Quantum risk analysis」</a> ) 量子モンテカルロ計算では、確率分布を量子回路に再現するが、その際、QGANを使うと再帰的な方法よりも少数のゲートで実装可能で、計算コストが減らせる。(参考: <a href="#">中村悠馬「Systematic Preparation of Arbitrary Probability Distribution with a Quantum Computer」</a> )
	Q-Learning	—	
	Deep Q-Learning (DQN)	量子強化学習	
その他の手法	—	量子データフィッティング	
		量子勾配法	
		量子最適化近似(QAOA)	QAOAで大規模な問題を解くには多数の量子ビットが必要で、実機の量子ビットの接続トポロジーと問題のグラフは必ずしもマッチしないので、組み込みのための情調整も必要。近似解アルゴリズムとしての性能もそれほど明らかではない。QUBO形式の組み合わせ最適化問題を扱う専用マシンは多く、NISQ上のQAOAアルゴリズムがサイズや解の精度など何らかの点で優位となるような問題設定の発見がまたれる。 <出典: 嶋田義皓「量子コンピューティング」p.153>
	—	量子自然言語処理	
	—	量子Autoencoder	
※全体	—	—	量子機械学習がその恩恵を一番受けることができるのは、おそらく量子的なデータやモデルの背景に量子系があるような場面の学習問題に適用したときであると考えられる。 <出典: <a href="#">御手洗光祐他「量子コンピュータを用いた変分アルゴリズムと機械学習」</a> >



## 1 チートシートとは

- 機械学習のチートシートのひとつにscikit-learnのものがある。
- scikit-learnでは、「多くの場合、機械学習の問題を解決する上で最も難しい部分は、ジョブに適した推定器(Estimator)を見つけることである」としており、**推定器選択の一助**になるものとして下図のチートシートを公開している。



## 2 量子機械学習におけるチートシートについて

- 従来の機械学習(古典手法)と同様に、量子機械学習(量子手法)での「推定器の選択」も課題であり、適用の検討を行うこととした。
- scikit-learnのチートシートからもわかるように、適用すべき推定器は問題の種類などの条件により決定することはできるが、古典と量子のどちらの手法を選択すべきか(たとえば、古典手法のk-means、量子手法のQ-means)は**単純に決めることができない**。その理由として以下の事項があげられる。
  - 量子手法ではどの量子コンピュータの種類を対象とするかにより、**適用できるアルゴリズムが異なる**〔□□テキスト2-1〕。
  - 古典・量子手法の比較では「**統一的な指標**」や「**検証で使用可能なリファレンス実装**」などが必要だが、これらは揃っていない。
  - 量子手法の「**実機による実証**」を行える、十分な性能を持つ量子コンピュータが存在しない。
  - データを量子コンピュータで扱うための**前後処理によるオーバーヘッド**(データの量子ビットエンコーディングなど)についても考慮が必要である。
- 量子手法を含んだチートシートは量子人材にとっては有用であると思われるが、現時点では明確な条件を記述することができないため、理論・技術の両面において今後の更なる発展が期待される。

## 3 参考資料

[scikit-learn Choosing the right estimator](#)

[Key questions for the quantum machine learner to ask themselves](#)

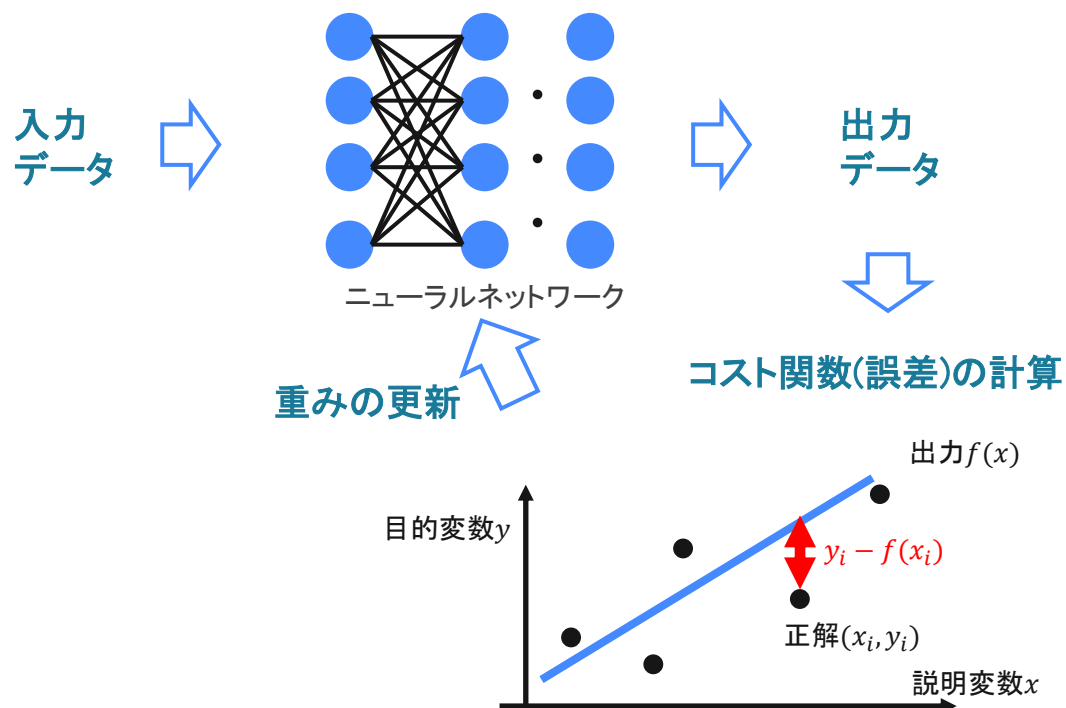


## 1 古典ニューラルネットワーク(NN)の学習

量子回路を学習モデルに見立てて機械学習タスクを行う手法。重ね合わせや量子もつれを活用して古典コンピュータで表現しにくい非線形モデルを効率的に構築することが期待されている。

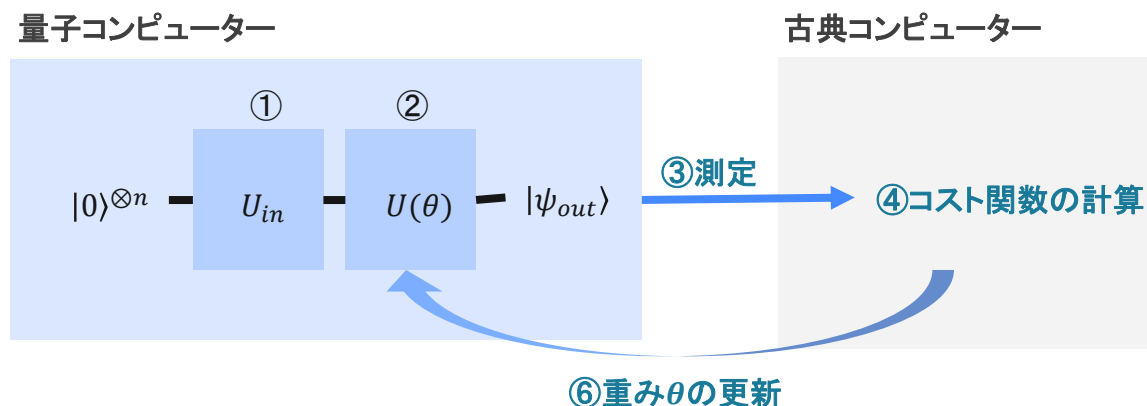
## 2 古典ニューラルネットワーク(NN)の学習

古典ニューラルネットワークでは、パラメータ付き線形変換と活性化関数(非線形)を使ってパラメータ最適化によりモデルを構築する。



## 3 量子ニューラルネットワーク(QNN)の学習

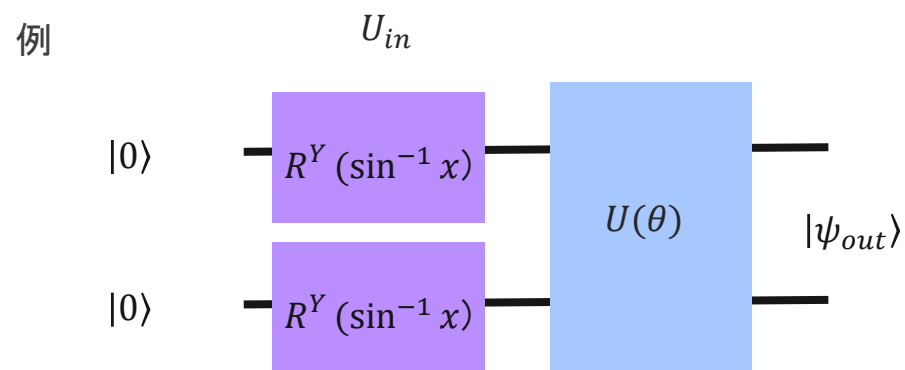
量子ニューラルネットワークでは、パラメータ付き量子回路を使ってパラメータ最適化によりモデルを構築する。



- ①入力値 $x$ をパラメータに持つ量子回路 $U_{in}$ を通して量子状態にエンコード
- ②パラメータ $\theta$ に持つ変分回路を通して出力状態を生成
- ③量子ビットを測定して出力を求める
- ④コスト関数を計算する
- ⑤コスト関数が小さくなるように $\theta$ を更新して最適化 $\theta^*$ を求める

## 1 多くの基底関数を利用可能

複数量子ビットのテンソル積構造により指数関数的に多くの基底関数を利用できるといわれている。



$$R^Y(\sin^{-1} x) = \sqrt{1 - x^2}I - ixY$$

入力状態の密度演算子

$$\rho_{in}(x) = |\psi_{in}(x)\rangle\langle\psi_{in}(x)|$$

$$= \left[ \frac{I + xX + \sqrt{1 - x^2}Z}{2} \right]^{\otimes 2}$$

$X_1 X_2$ の期待値を計算すると $x^2$

→入力データを非線形関数で変換してエンコーディングが可能

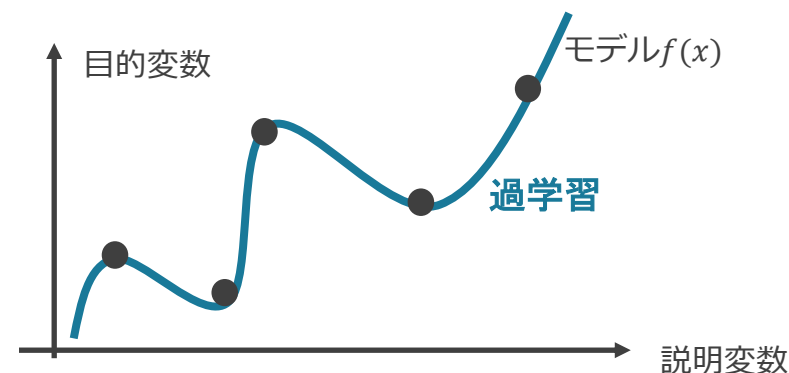
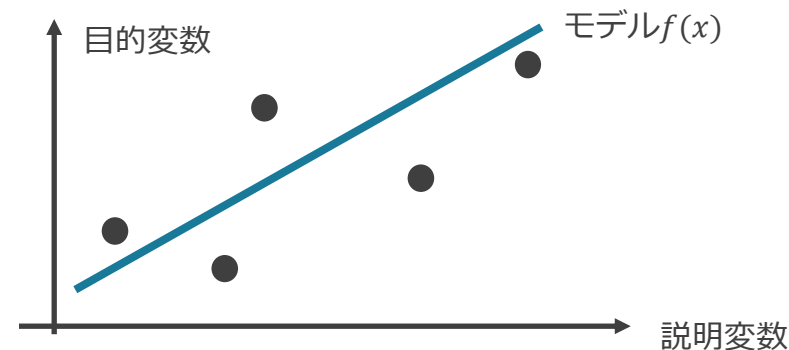
## 2 過学習の抑制

量子回路のユニタリ性により過学習を防ぐことが出来ると期待されている。

※過学習とは

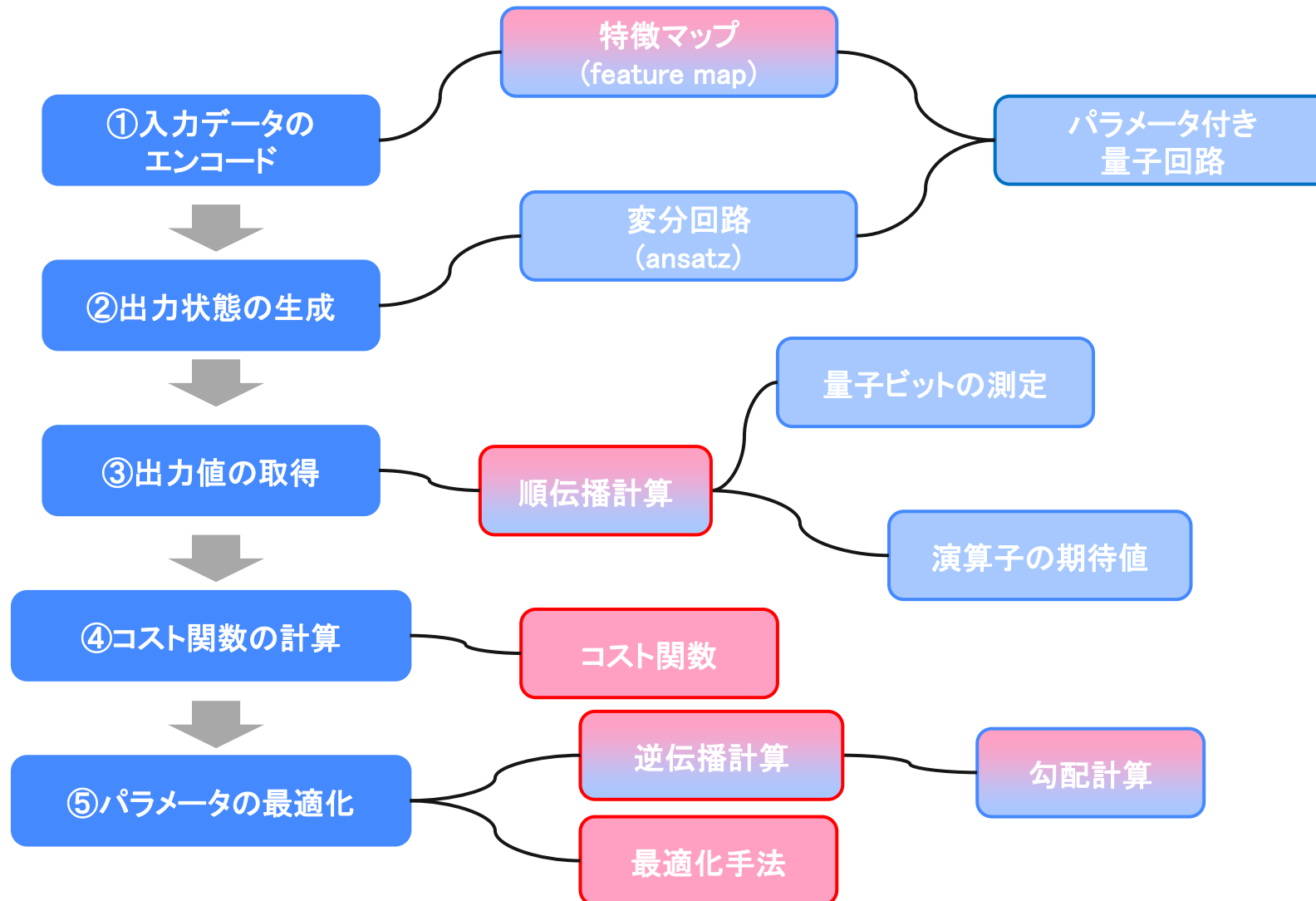
構築されたモデルがデータに過剰に適合した結果、新たなデータに対する予測器として正しく機能しない現象。

古典計算ではコスト関数に制約を課す工夫(正則化)が必要となる。



QNNのアルゴリズムを構築する各要素は以下の通り。

この中で、本書ではQNNを理解する上で重要な「順伝播計算」「逆伝播計算」について説明する。



## 1 QNNとNNの順伝播計算・逆伝播計算の対応

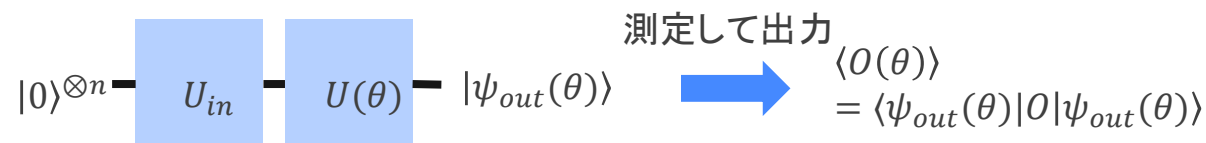
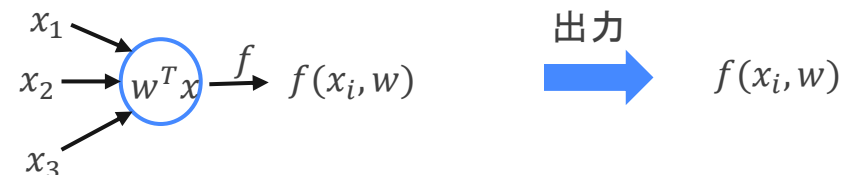
QNNとNNにおける順伝播・逆伝播計算は以下のように対応する。

- 順伝播計算  
入力から出力を求める
- 逆伝播計算  
重みパラメータ更新のために必要なコスト関数の勾配を求める

	順伝播計算	逆伝播計算
QNN	演算子測定から期待値を計算	演算子の期待値から勾配を計算
NN	関数から出力を数値計算	関数から勾配を数値計算

## 2 QNNにおける順伝播計算

入力から出力を求める。  
QNNにおける出力は、演算子 $O$ の $|\psi_{out}\rangle$ に対する期待値とみなす。



## 2 QNNにおける逆伝播計算

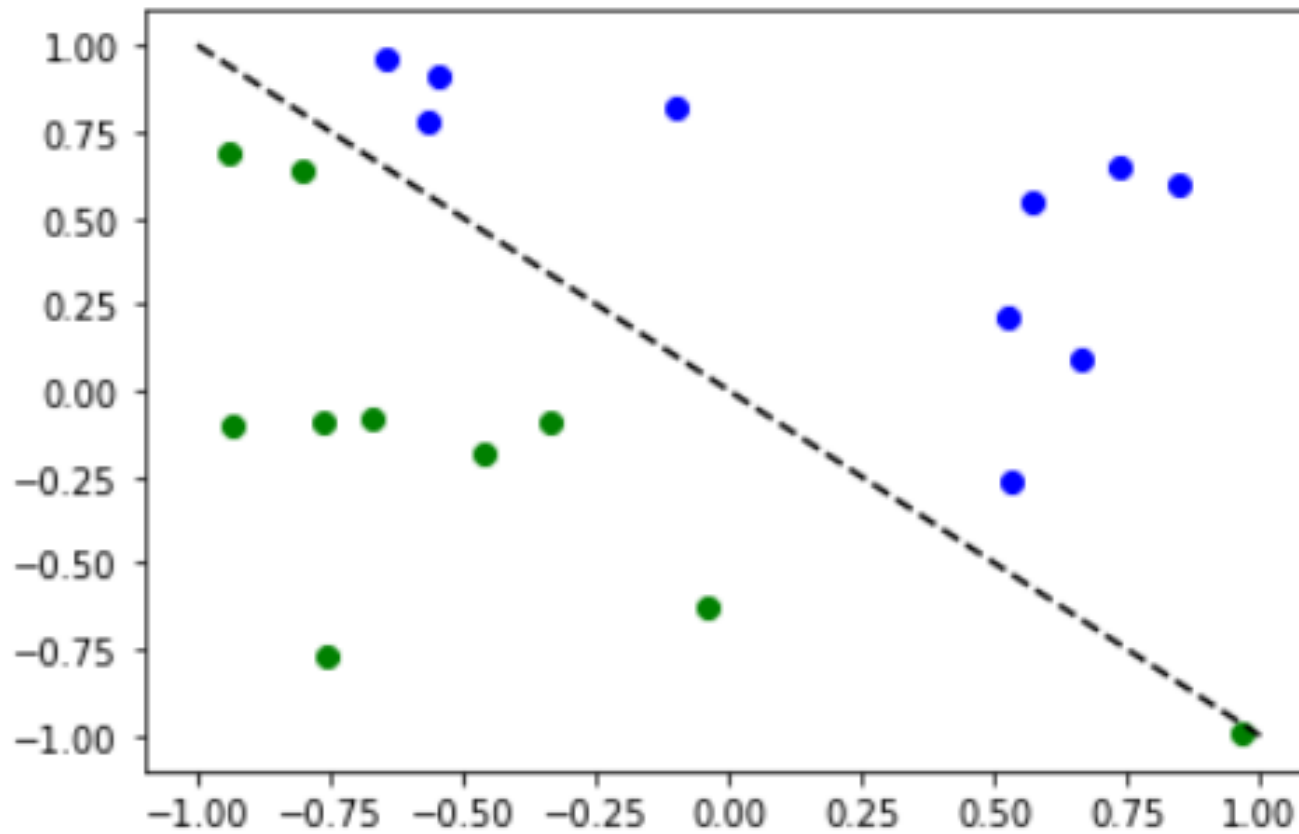
重みを更新するために、コスト関数の勾配を求める。  
量子回路での勾配計算は量子の性質を利用していくつか考案されている。

例: パラメータシフト法

$$\frac{d\langle O(\theta) \rangle}{d\theta} = \frac{1}{2} \left\{ \left\langle O \left( \theta + \frac{\pi}{2} \right) \right\rangle - \left\langle O \left( \theta - \frac{\pi}{2} \right) \right\rangle \right\}$$

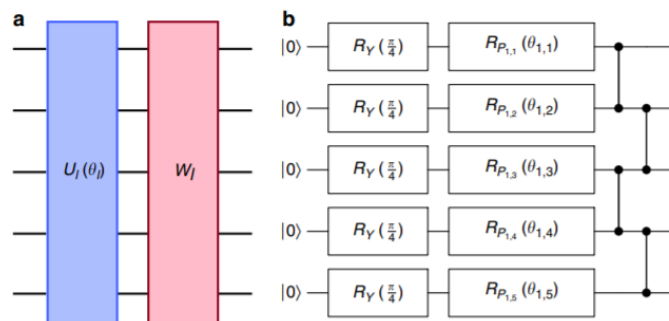
ランダムに生成した20サンプルを $y = -x$ を境界に分類する。

→ 実装は、「応用レベル:テキスト:3-2-2\_量子ニューラルネットワーク\_実装1.ipynb」(またはPDF)を参照。

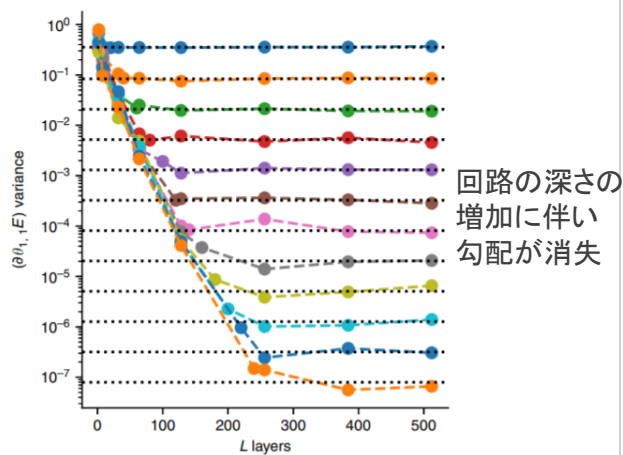
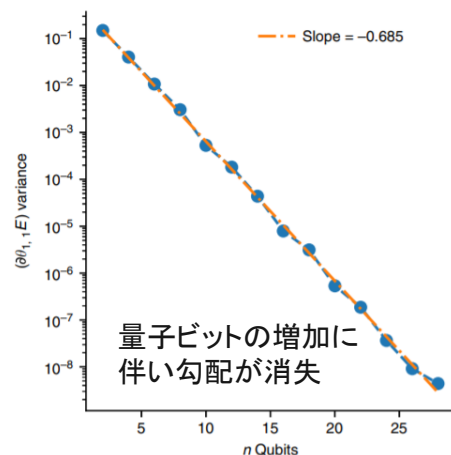


## 1 barren plateaus問題

現在の研究ではQNNはパラメータがランダムに初期化された場合、量子ビットや量子回路の深さの増加に伴い勾配が消失してしまい、学習が停滞してしまうことが報告されている。



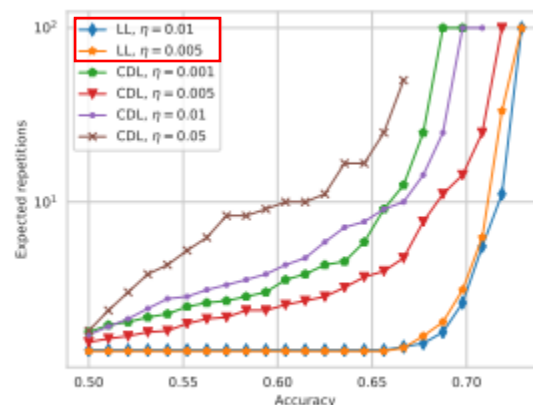
量子ビット増  
回路の深さ増



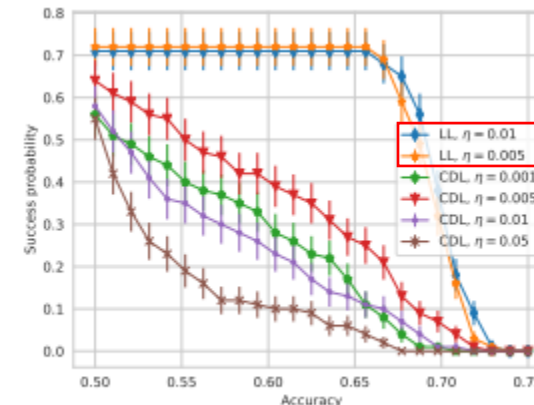
McClean, J.R., Boixo, S., Smelyanskiy, V.N. et al.  
Nat Commun 9, 4812 (2018).

## 2 最新の動向

最新の研究では浅い学習層に区切って繰り返し学習させる  
Layerwise Learningを始めとするいくつかの手法が考案されていて日々進化している。



(a) expected number of repetitions



(b) success probability

Skolik, A., McClean, J.R., Mohseni, M. et al.  
Quantum Mach. Intell. 3, 5 (2021).

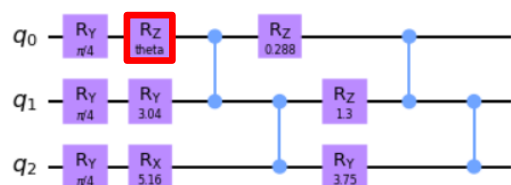


## 1 論文の実装を再現する

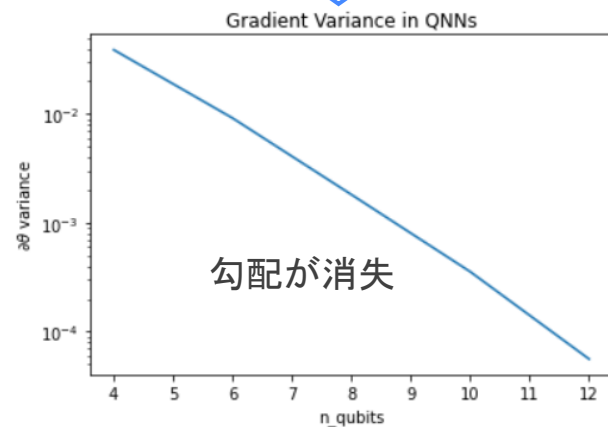
2018年に報告されたbarren plateaus問題を量子SDKであるQiskitを利用して、量子シミュレータ上において論文の報告内容の再現を試みた。

報告された回路において量子ビット数の増加に伴い、勾配が称していることを再現することに成功した。

3量子ビットの例(パラメーター一つ)



量子ビット増



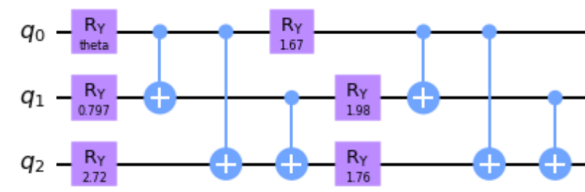
## 2 別の回路でも再現することを検証する

この事象が他の量子回路においても再現することの検証を行った。

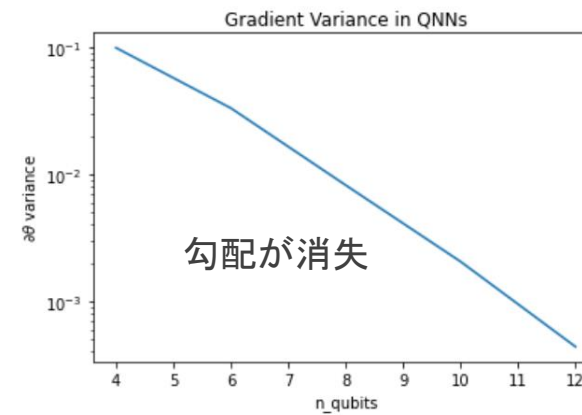
量子回路には量子ニューラルネットワークによく利用されるReal Amplitude回路を使用し、左と同条件で勾配を計算した。

結果、他の回路においても同様の事象が生じることの検証に成功した。

Real Amplitude回路

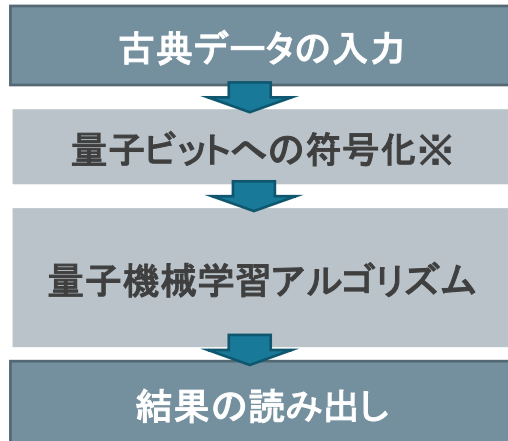


量子ビット増



## 1 量子コンピュータを使ったクラスタリング

- **クラスタリングとは**  
データを類似したグループに分類するアルゴリズムであり、教師なし機械学習のひとつである。  
クラスタリングアルゴリズムとしてK平均法などがある。
- **量子コンピュータを用いたクラスタリング**  
クラスタリングの量子コンピュータ応用例としてQ-MEANSが考えられている。  
類似性の評価に量子コンピュータの特性を活かし、アルゴリズムを効率化する方法である。現時点では、量子状態の記憶デバイスがなく、古典表現と組み合わせたものとなっている。
- **量子機械学習の実装**

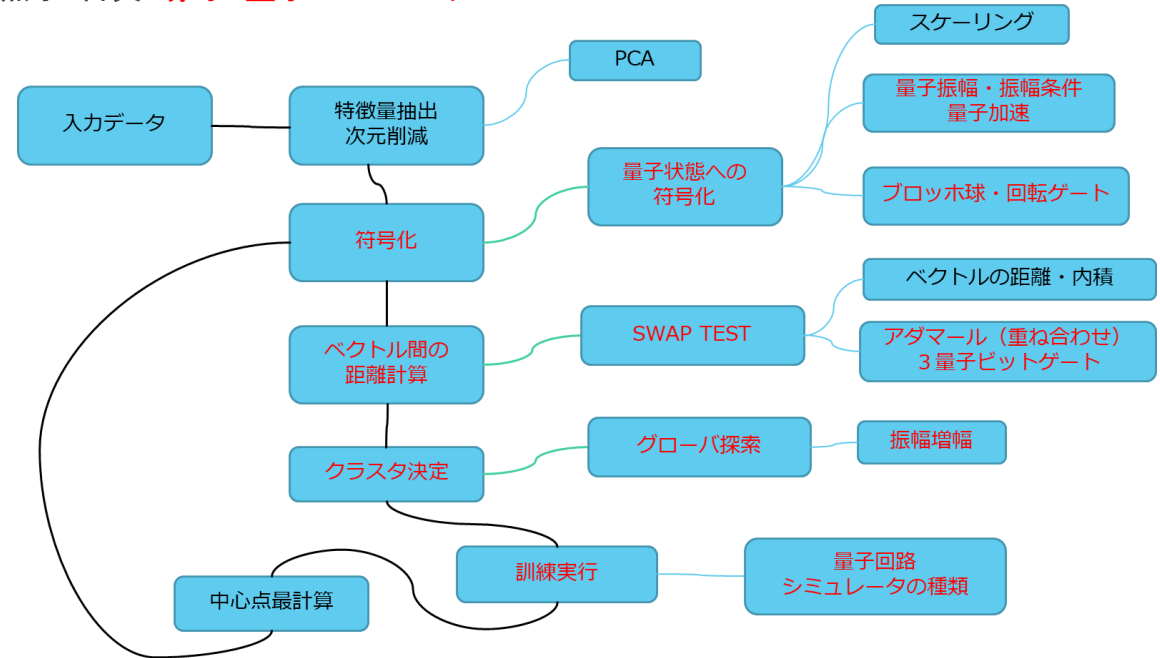


※現在の量子コンピュータには量子状態でデータを保存しておく記憶デバイスがない。  
古典データの inputs は、量子ビットに符号化するステップが必要とされる。

## 2 量子コンピュータアルゴリズム例: Q-MEANS

- 関連する量子コンピュータの要素を下図に示す。

黒字: 古典 赤字: 量子コンピュータ



## 3 参考資料

「量子コンピュータによる機械学習」(共立出版)

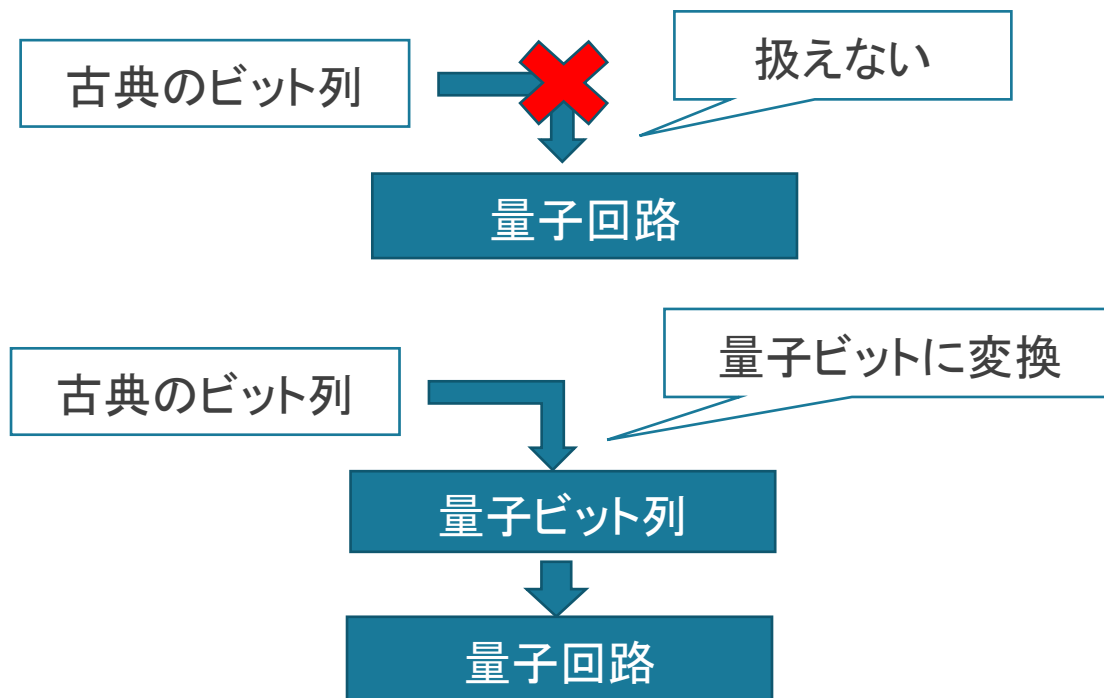
<https://qiskit.org/documentation/>

<https://github.com/Morcu/q-means>

## 1 データを量子状態に符号化するとは？

現在の量子コンピュータには、量子状態でデータを保存する記憶デバイスがない。そのため、古典で保存された入力データを量子ビットでの表現に変換する必要がある。

量子ビットには、古典のビットにはない重ね合わせの特性がある。この特性を活かし、古典のビットより多くのデータ表現をすることができる。



- 将来、量子状態を記憶するデバイスが開発されると、古典と同様にデータ入力・出力が考えられている。

## 2 符号化方式と特徴

### ● 計算基底符号化(入力特徴量=2値)

古典のビット配列を量子ビットの計算基底状態に紐付ける方法。  
古典のビット数と同じ量子ビットが必要となる。

例) 実数3を計算基底符号化する。

古典ビット2進数表現 (0011)  $\Rightarrow$  量子ビット(|0011>)

### ● 振幅符号化(入力特徴量=連続値)

古典の実数ベクトルを量子ビットの振幅値とする。  
1量子ビットで2次元ベクトルを表現できる。

例) 2次元の実数ベクトル( $x_1, x_2$ )

$\Rightarrow$  量子状態  $|\psi\rangle = a|0\rangle + b|1\rangle$  の振幅 ( $a, b$ ) に変換

※ 振幅条件  $|a|^2 + |b|^2 = 1$  を満たす

### ● その他の符号化

量子状態の角度( $\theta, \phi$ )に変換する方法や確率分布、ハミルトニアンとして表現する方法などが考えられている。

## 3 参考資料

「量子コンピュータによる機械学習」(共立出版)

<https://qiskit.org/documentation/>

<https://github.com/Morcu/q-means>

## 1 ベクトルの距離の算出

## ● データの類似性を評価する

機械学習においては、データの類似性を評価するためにベクトル間の距離が手法としてよく使われる。

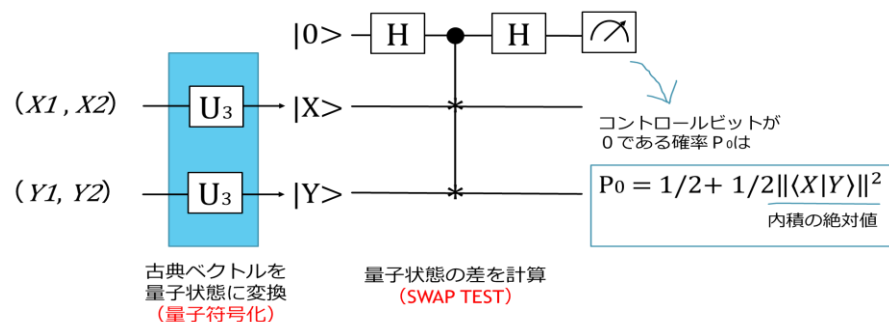
ここでは量子ビットに符号化した入力ベクトルの距離を算出する方法を紹介する。

## ● SWAP TESTによる内積算出

SWAP TESTは、コントロールビットと2つの量子ビットを干渉させコントロールビットが0である確率から内積を算出する。

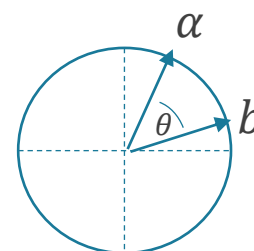
## ➤ SWAP TESTの手順

- ・ 1つのコントロールビットと2つのターゲットビットを用意する。
- ・ コントロールビットにアダマールゲートを作用させ、2つの入力ベクトルを符号化したターゲットビットを「1」のブランチでのみ交換する。再度アダマールゲートをかけ、測定する。
- ・ この回路でコントロールビットを測定した結果に、2つのベクトルの内積が現れる。(下記の回路図)



## 2 ベクトルの距離と符号化の制約

## ● ベクトルの距離と内積



$$\cos \theta = \frac{\langle a|b \rangle}{\|a\| \|b\|}$$

$\langle a|b \rangle$  は  $a, b$  の内積

$\|a\|$  はベクトル  $a$  の大きさ

ベクトルの大きさを1とした場合、 $\cos \theta$  は内積と等しく、内積値により2つのベクトルがどの程度同じ方向をさしているかがわかる。

## ● 符号化の制約

古典データを量子ビットに符号化し、量子回路で内積を算出することができる。しかし、符号化の前提条件に適するデータしか扱えないことを考慮する必要がある。

## ➤ 振幅符号化を利用する場合の条件

- ・ 量子振幅条件  $|a|^2 + |b|^2 = 1$  に規格する
- ・ 線形性であること、非線形性を持つデータには向かない

## 3 参考資料

「量子コンピュータによる機械学習」(共立出版)

<https://qiskit.org/documentation/>

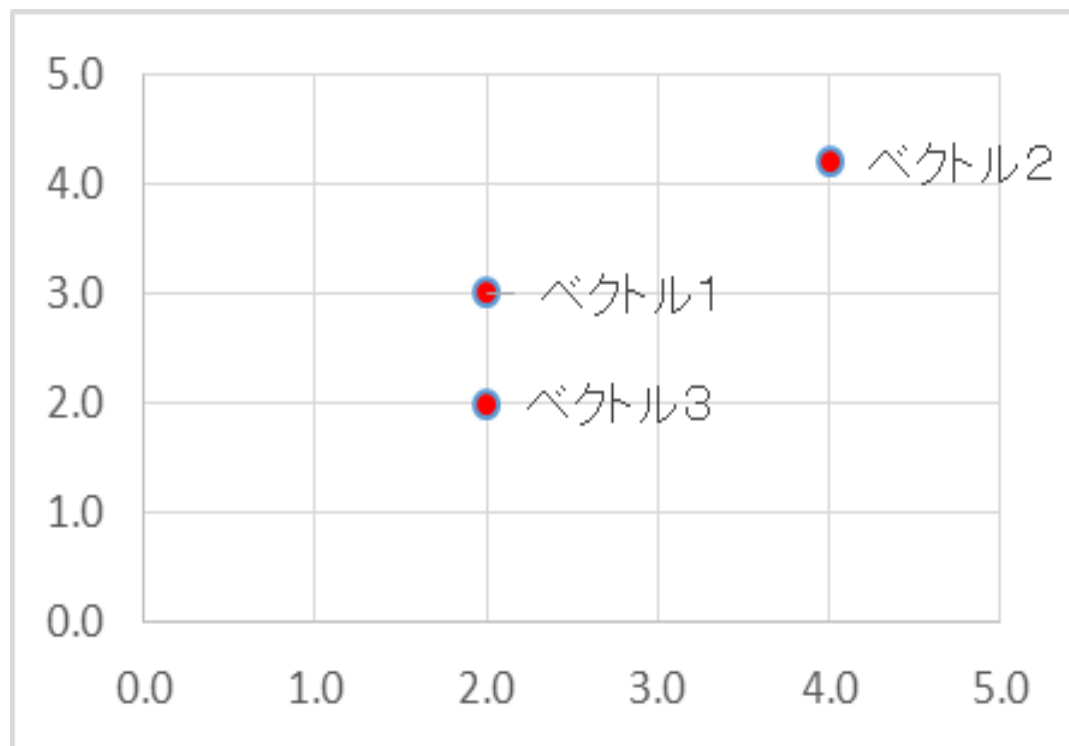
<https://github.com/Morcu/q-means>

例) 入力値として、下記の3つの2次元ベクトルを量子ビットに符号化、ベクトル1、2其々とベクトル3の内積計算を行う。

ベクトル1 = [2.0, 3.0]

ベクトル2 = [4.0, 4.2]

ベクトル3 = [2.0, 2.0]



- 古典のクラスタリングで使われるユークリッド距離では、ベクトル3とベクトル1に近い

$$\begin{aligned} D(\text{ベクトル1} \cdot \text{ベクトル3}) \\ = \sqrt{(2-2)^2 + (3-2)^2} = \sqrt{1} \end{aligned}$$

$$\begin{aligned} D(\text{ベクトル2} \cdot \text{ベクトル3}) \\ = \sqrt{(4-2)^2 + (4.2-2)^2} = \sqrt{8.84} \end{aligned}$$

- ベクトルの大きさは関係なく方向で評価する  
コサイン距離では、ベクトル3とベクトル2が近くなる

```
x = [[2.0, 3.0],[4.0, 4.2],[2.0, 2.0]]
x_n = np.array(x)
```

```
n_size,n_features = x_n.shape
x_n_scale = np.zeros((n_size,n_features),dtype=np.float64)
x_n_scale[:,0] = min_max(x_n[:,0])
x_n_scale[:,1] = min_max(x_n[:,1])
```

0～1にスケールリング

```
x_n_ang = np.zeros((n_size,n_features),dtype=np.float64)
x_n_ang[:,0] = (x_n_scale[:,0] / sum(x_n_scale[:,0])) * math.pi
x_n_ang[:,1] = (x_n_scale[:,1] / sum(x_n_scale[:,1])) * (math.pi * 2)
print(x_n_ang)
```

角度に変換  
 $(0 < \theta < \pi)$   
 $(0 < \phi < 2\pi)$

```
for i in range(2):
    q = QuantumRegister(3)
    c = ClassicalRegister(1)
    qc = QuantumCircuit(q,c)
    qc.h(q[0])
    qc.h(q[1])
    qc.h(q[2])
    qc.u3(x_n_ang[i,0],x_n_ang[i,1],0,q[1])
    qc.u3(x_n_ang[2,0],x_n_ang[2,1],0,q[2])
    qc.cswap(q[0],q[1],q[2])
    qc.h(q[0])
    qc.measure(q[0],c[0])
    # 回路実行
    r = execute(qc,Aer.get_backend('qasm_simulator'),shots=1000).result()
    rc = r.get_counts()
    print('rc=',rc)
```

量子ビットを用意(3量子ビット)

角度に変換したベクトルをU3回転ゲートのパラメータとして作用させる

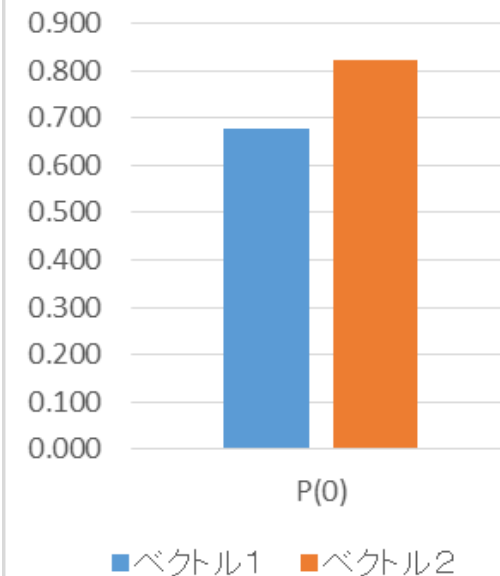
制御スワップゲートで特定のブランチでターゲットを交換する

## ● 角度による符号化

入力ベクトルを $\theta$ 、 $\phi$ の角度に変換し  
 U3回転ゲートを作用させて量子ビットを  
 初期化する。

- ・ベクトル1、2とベクトル3のSwap testの結果、 $P_0$ の確率から、ベクトル2の方が近い算出になる。
- ・この実装例では、**ベクトルの向き** による判定となる。

角度符号化





```
x = [[2.0, 3.0],[4.0, 4.2],[2.0, 2.0]]
x_n = np.array(x)

n_size,n_features = x_n.shape
x_n_ampenc = np.zeros((n_size,n_features),dtype=np.float64)
for i in range(n_size):
    x_n_ampenc[i,:] = x_n[i,:] / math.sqrt(np.sum(x_n[i,:] ** 2))
print(x_n_ampenc)
```

量子振幅値への変換

```
initial_state_q2 = x_n[2,:]
for i in range(2):
    initial_state_q1 = x_n[i,:]
    q = QuantumRegister(3)
    c = ClassicalRegister(1)
    qc = QuantumCircuit(q,c)
    qc.h(q[0])
    qc.initialize(initial_state_q1, 1)
    qc.initialize(initial_state_q2, 2)
    qc.cswap(q[0],q[1],q[2])
    qc.h(q[0])
    qc.measure(q[0],c[0])

# 回路実行
r = execute(qc,Aer.get_backend('qasm_simulator'),shots=1000).result()
rc = r.get_counts()
print('rc=',rc)
```

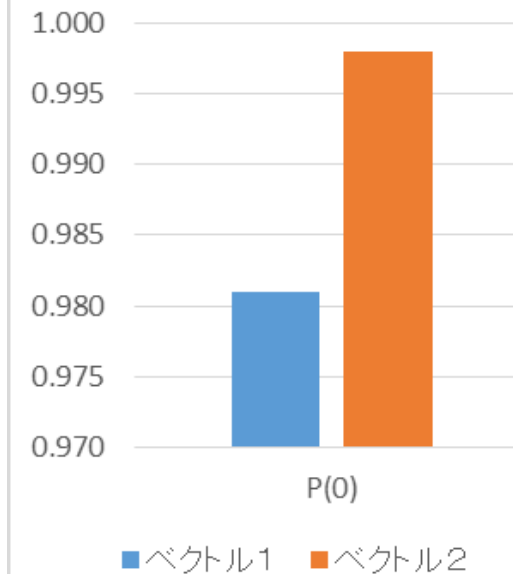
量子ビットを用意(3量子ビット)

Qiskit Initializeで振幅値を与えて初期化する

● 振幅値による符号化  
入力ベクトルを量子振幅値に変換し、量子状態の初期値として与える。

・ベクトル1、2とベクトル3のSwap testの結果、 $P_0$ の確率から、ベクトル2の方が近い算出になる。  
・角度符号化と同じく、**ベクトルの向き** による判定となる。

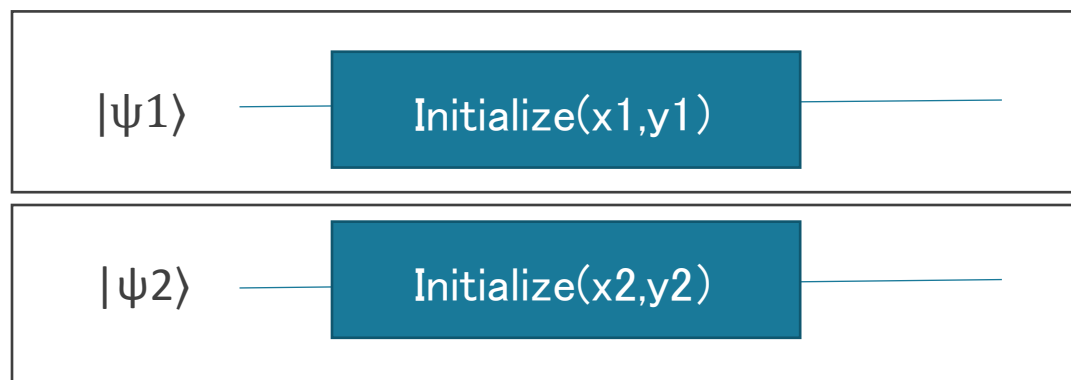
振幅符号化



## ● 量子状態のFidelity(一致度)の評価

- 2つの量子状態が同じかどうかを確認する
- 同じであれば、Fidelityは1に等しくなる

$$F(|\psi_1\rangle, |\psi_2\rangle) = |\langle\psi_1|\psi_2\rangle|^2$$

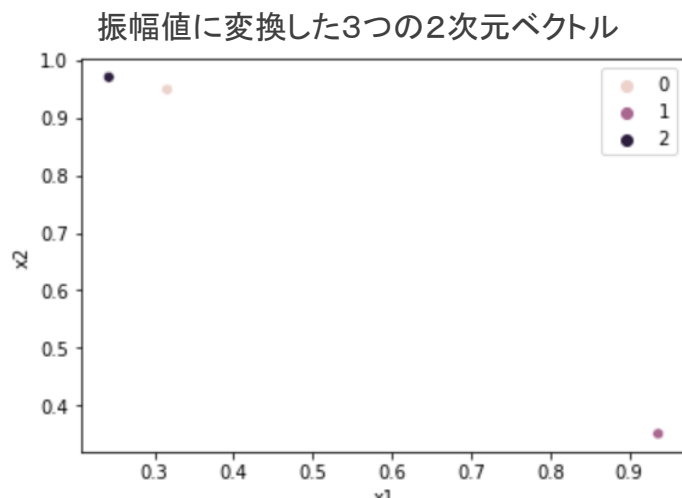


回路実行し、量子状態を得る

```
backend = BasicAer.get_backend('statevector_simulator')
job = backend.run(transpile(qc, backend))
qc_state = job.result().get_statevector(qc)
```

2つの量子状態のFidelityを得る

```
f = state_fidelity(qc_state1, qc_state2)
```



・ベクトル0、1、2 をサンプルに、ベクトル0,1 とベクトル2 の距離をFidelityで表現する

ベクトルの量子状態

Vector0 =[0.31622777+0j 0.9486833 +0j]

Vector1 =[0.93632918+0j 0.35112344+0j]

Vector2 =[0.24253563+0j 0.9701425 +0j]

Vector0とVector2のfidelity = 0.994

Vector1とVector2のfidelity = 0.322

ベクトル0,1とベクトル2のfidelityでも、ベクトル0と2の一致度が、0と1よりも高い判定 ⇒ クラスタリングの評価に使える。

## 1 クラスタリングへの応用(現状)

### ● 現状と課題

- ・ 量子ビットに符号化することで、古典ビットより少ないビット数で多くのデータを表現でき、理論的には量子加速が期待される。次元数が多いデータを次元削減なく高速にクラスタリングできるのではないかと（今回未検証）
- ・ 現状では、クラスタリングの一部（内積計算）などの部分を量子回路で実現できることを確認できているところまでである。
- ・ 特徴量が表現しきれないケースもあり、期待されたクラスタリングができない。
- ・ クラスタ判別を古典データとして処理しており、量子コンピュータでの計算より結果的に時間がかかる。

### ➤ 結論

- ・ 現時点では、クラスタリングは量子回路でもできることを確認している段階。
- ・ 量子加速の可能性を感じるが、量子状態の記憶デバイスがないと全体としての高速化が難しい。
- ・ 特徴量の表現方法など、更なる方法論の研究が必要と思われる。

## 2 今後の取り組み・期待されること

- 今回、4次元配列まで検証したが、もっと多次元でのクラスタリングについては未検証。今後追加で検証。
- 量子記憶デバイス、または、代替方法の開発について、今後の技術の進展をウォッチする。
- 量子機械学習のアルゴリズムについて、他の手法の検証はできていないため、今後検証を試みる。

## 3 参考資料

「量子コンピュータによる機械学習」(共立出版)

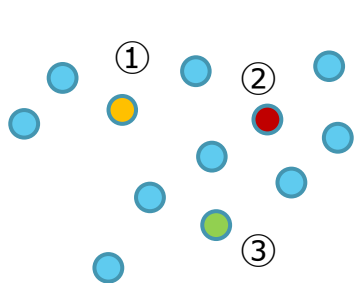
<https://qiskit.org/documentation/>

<https://github.com/Morcu/q-means>

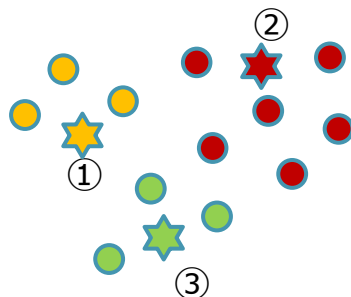
例) K平均法でのクラスタリングを量子回路で実行してみる。

➤ K平均法によるクラスタリング

- ・データから各クラスターの重心点の距離を計算し、近い距離のクラスターに分類する。
- ・最初はランダムに抽出した重心でクラスター分類し、クラスターの平均を新しく重心とする。
- ・重心に変動がなくなった時点で分類終了。  
(あるいは、分類の繰り返し回数で終了)



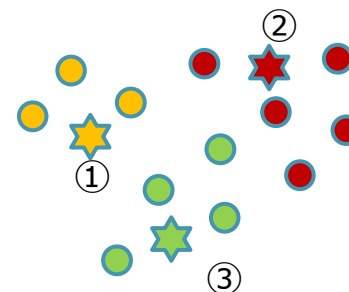
- ①クラスター数を決める  
②初期の中心点を決める



- ③一番近い距離の中心点に所属するクラスターを決める  
④クラスターの平均値を再計算し、新たな中心点にする。

③④を繰り返し、中心点の変動がなくなったら終了

クラスタリング終了



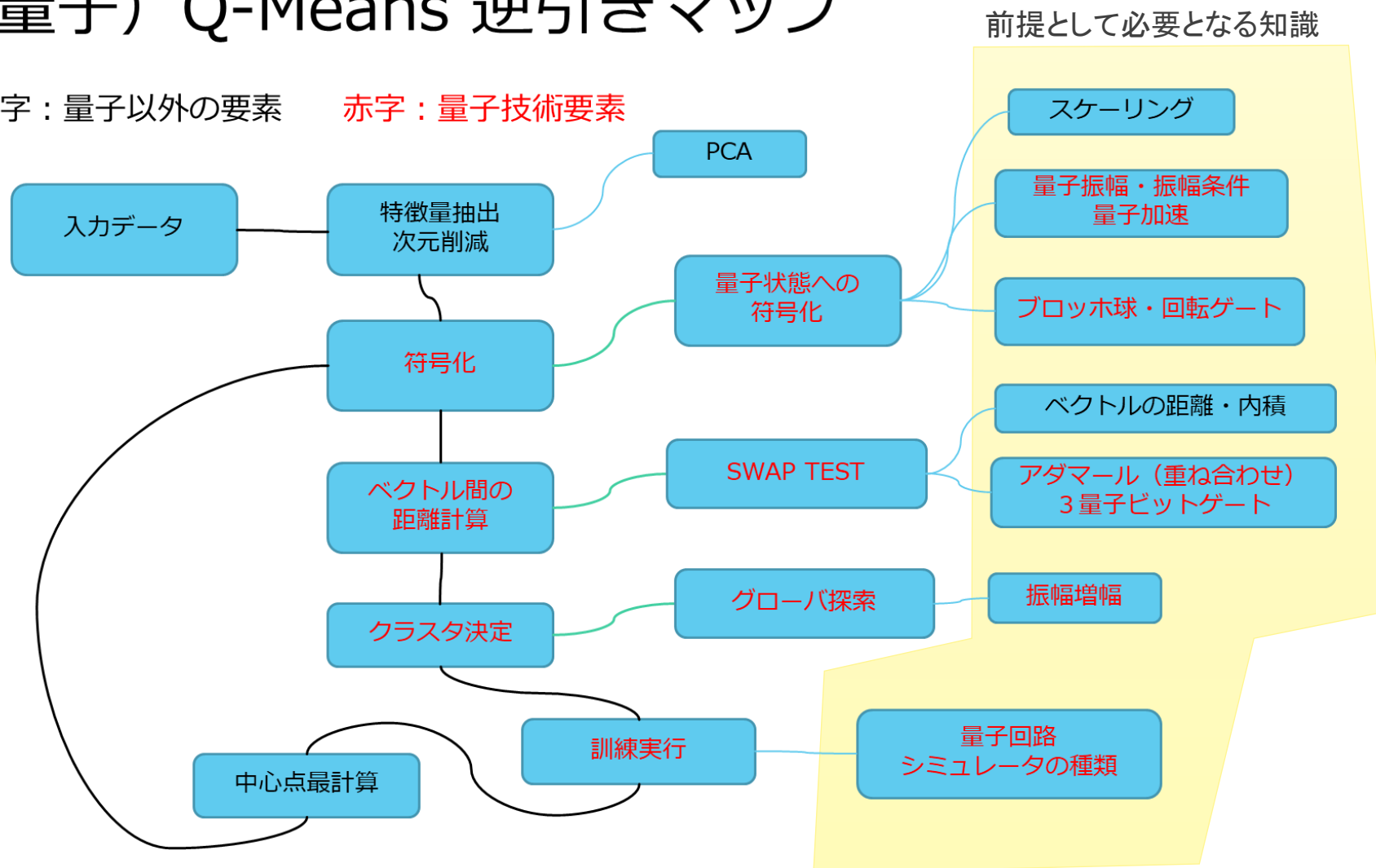
- ⑤ ③④を繰り返し、中心点の変動がなくなったら終了

- 量子回路を使ったクラスタリング(Q-Means)のアルゴリズムと前提となる量子コンピュータ学習要素を逆引きマップとした。  
前提となる要素知識については基礎編で確認してもらいたい。

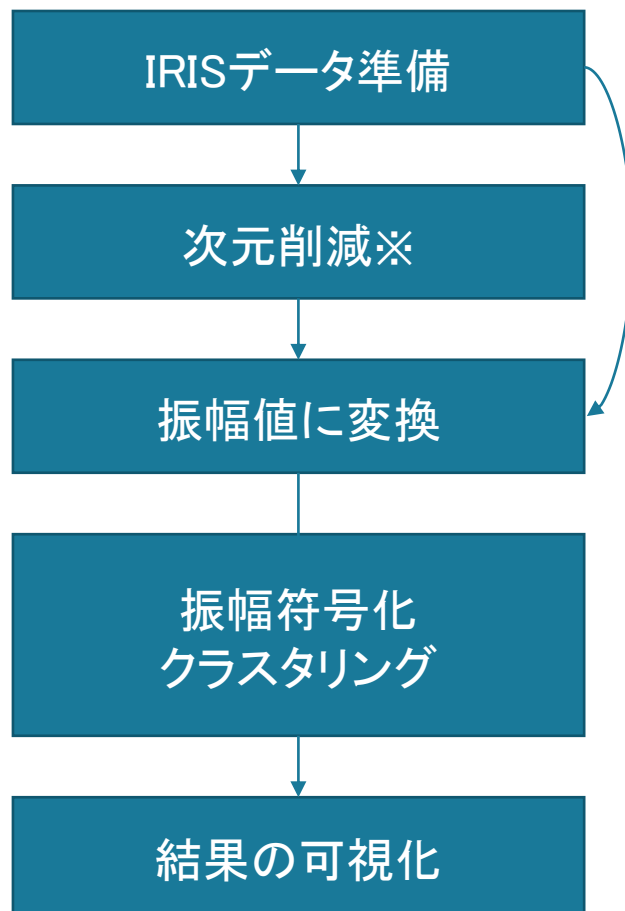
## (量子) Q-Means 逆引きマップ

黒字：量子以外の要素

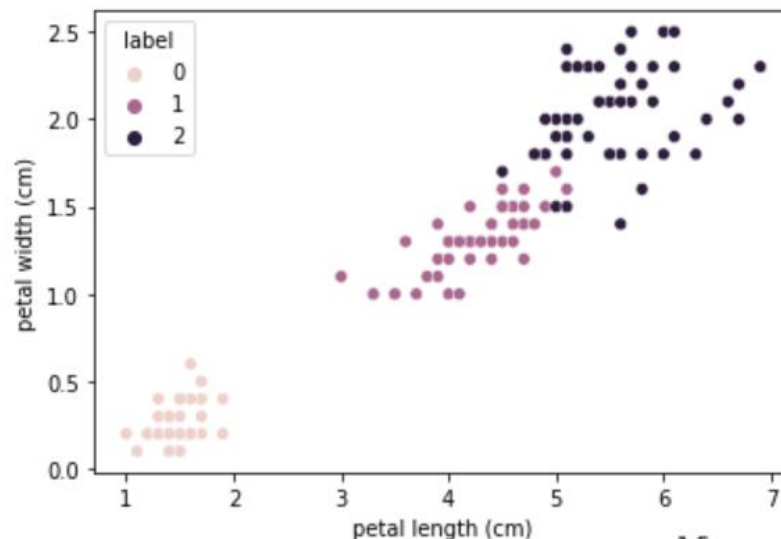
赤字：量子技術要素



IRIS (あやめ) サンプルデータを使ってクラスタリングを試みる。

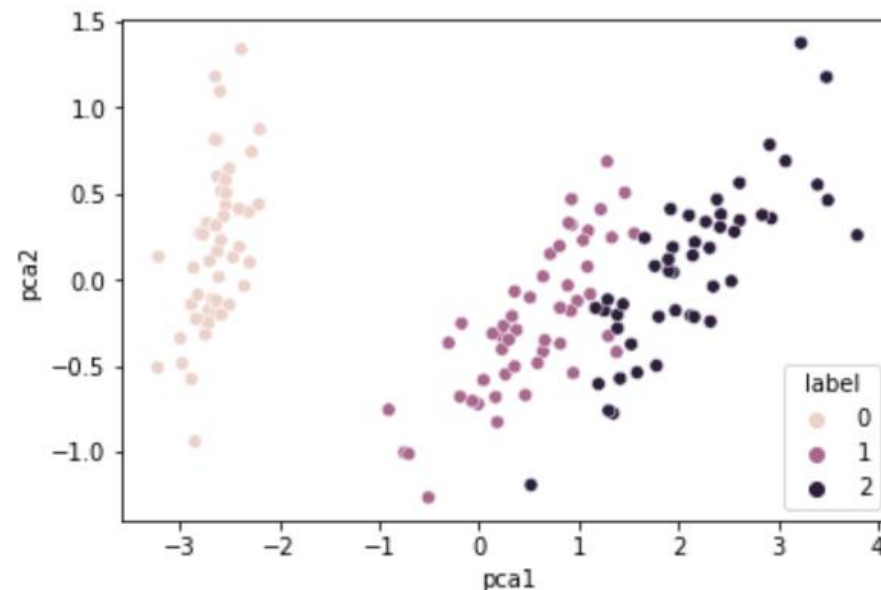


※多次元でも表現できるため、次元削減は必須ではない



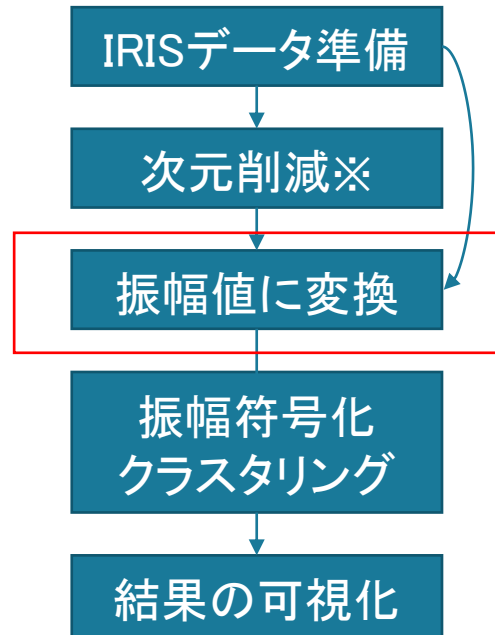
2項目を使って、クラス分けを可視化した。  
(正しい分類)

IRISデータは4項目あるので  
2つの主成分に次元削減





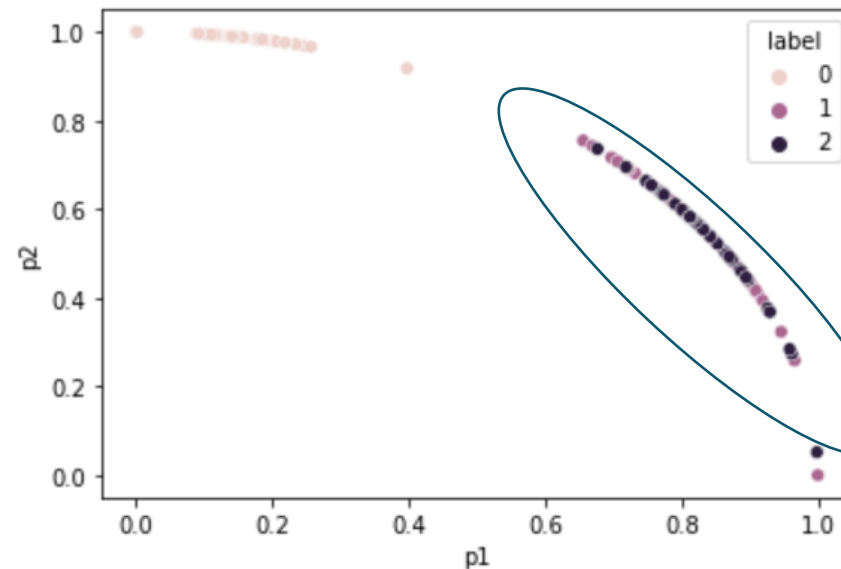
2つの主成分(2次元)のデータを振幅値変換する。



入力の配列を振幅値に変換するサンプルコード

```
def amp_def(X):  
    X_size,X_features = X.shape  
    # 出力用配列の初期化  
    amp_X = np.zeros((X_size, X_features),dtype=np.float64)  
    # 入力配列を振幅値に変換する  
    for i in range(X_size):  
        if np.sum(X[i,:]) == 0:  
            X[i,:] += 1  
        amp_X[i,:] = X[i,:] / math.sqrt(np.sum(X[i,:] ** 2))  
    return amp_X
```

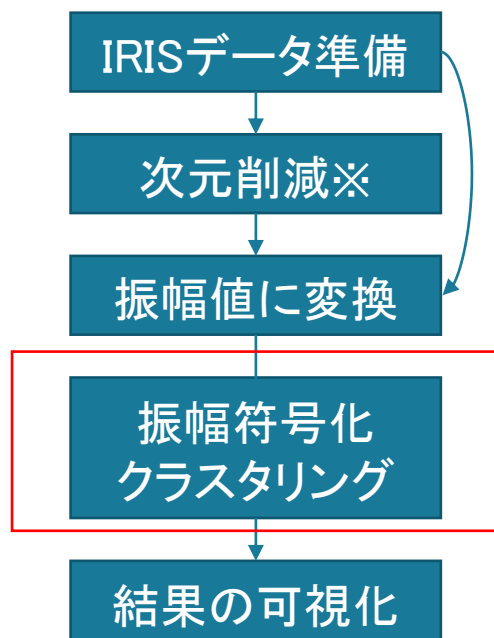
振幅値の条件  
配列の二乗和=1を満たすように  
正規化する。



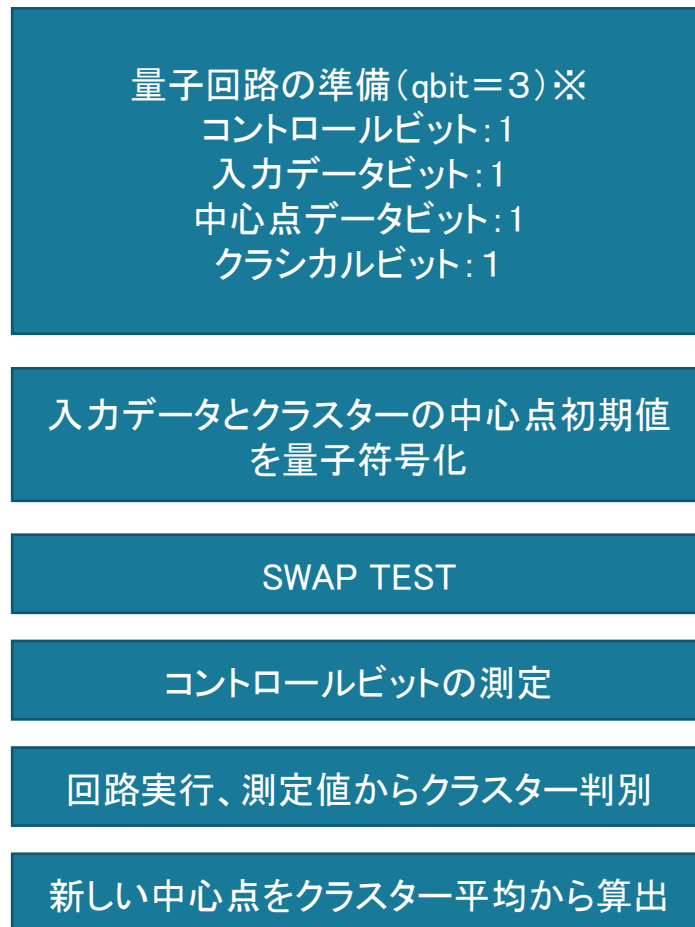
主成分の振幅値変換後を可視化

Label 1と2のデータはクラスタリングが難しそうである。

振幅値変換した主成分データをインプットとして、クラスタリングする。  
判別方法は、SWAP TESTとState Fidelityの2通りを試してみる。



## SWAP TESTによる判別



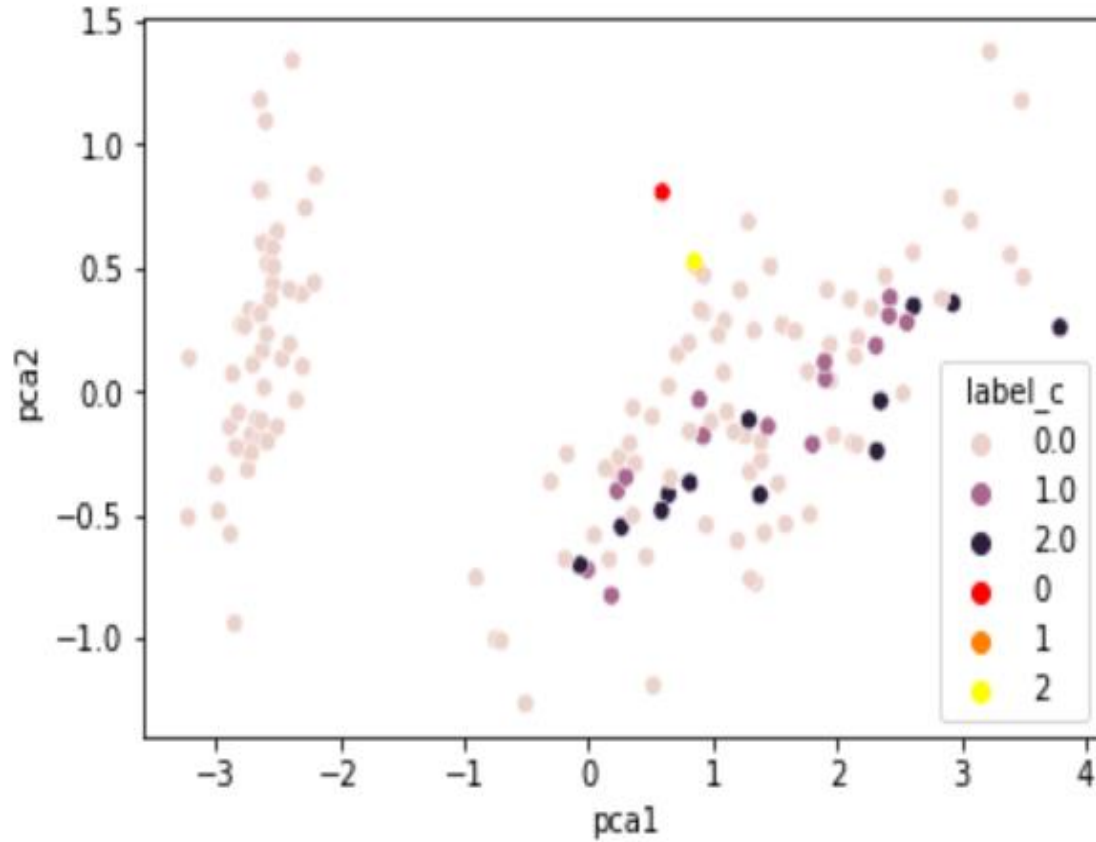
※2次元のデータを前提としたqbit数

## State\_Fidelity による判別

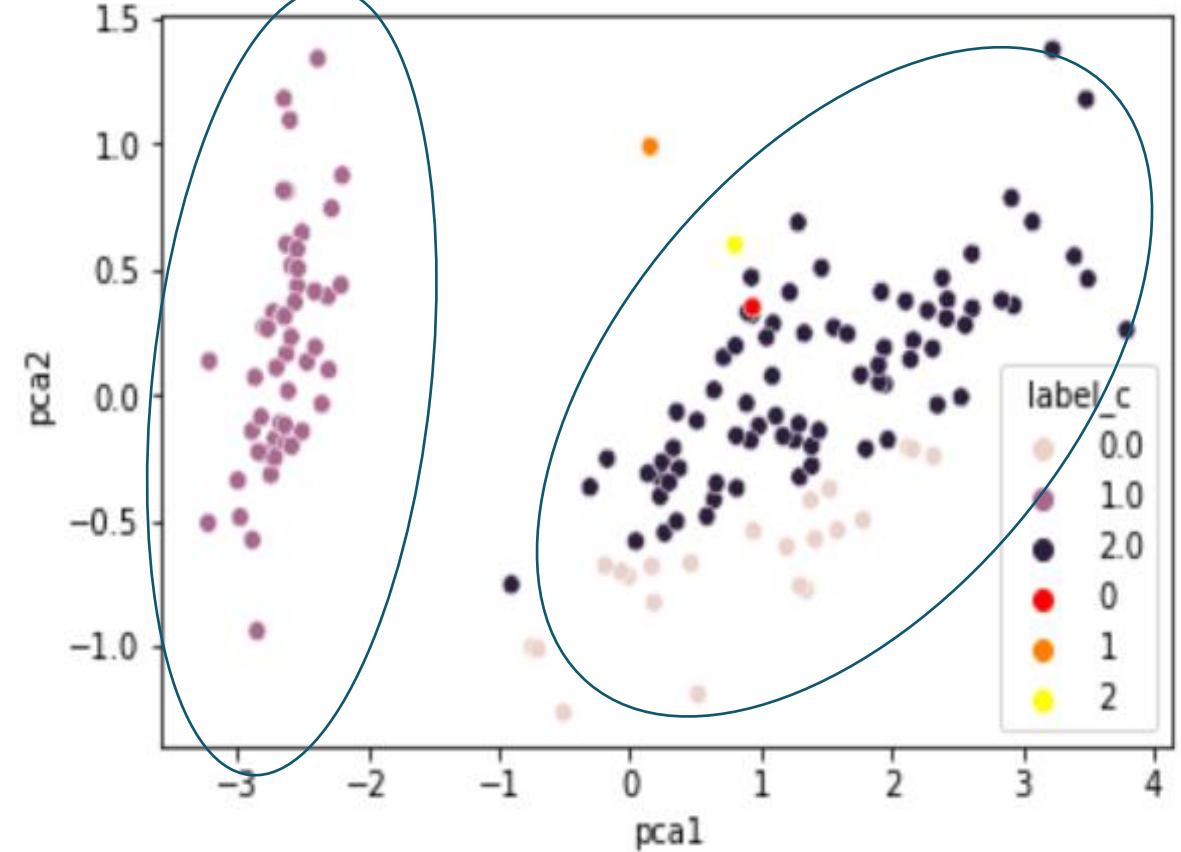


- SWAP TEST、state\_Fidelity の判別では、state\_Fidelityの方が精度が高かった
- Label 1と2のデータは、振幅値変換で特徴量が表現しきれないため、正解と異なる結果になった。

SWAP TESTの結果



State\_Fidelityの結果



参考) 多次元のクラスタリング: IRISデータ4項目を使ってみる。

基本は、2次元と同じだが、次元数に応じて量子ビットを用意する。(量子ビット数の二乗=表現できる次元数)

4項目の表現には、2の二乗=4 2量子ビットで表現できる。

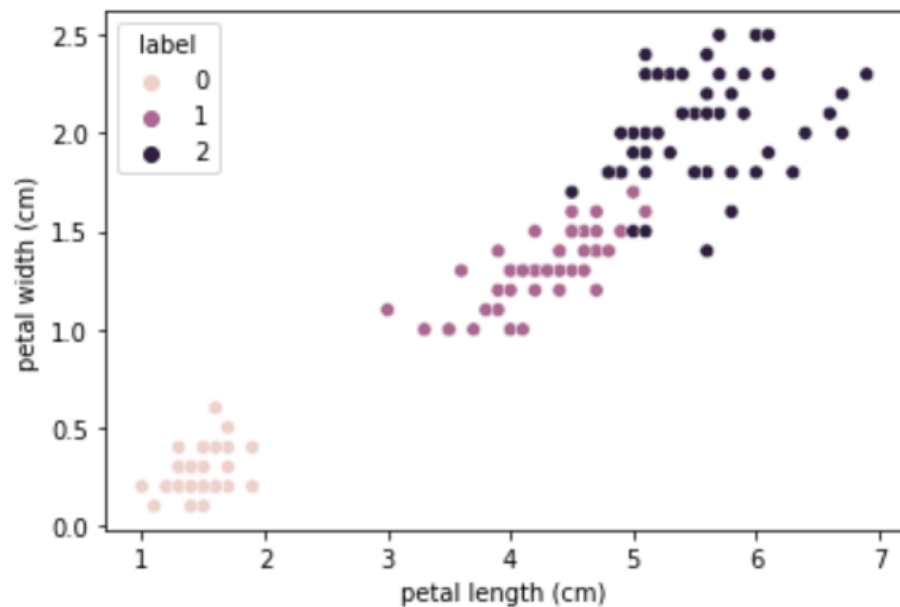
( 3量子ビット=8項目、4量子ビット=16項目……)

振幅値 = [ X1 , X2 ,X3 ,X4 ] 4項目の配列

=> X1 |00> X2 |01> X3 |10> X4 |11> のように振幅値として割り当てる

判別手法は、state\_Fidelityを使用。

正解ラベルでの可視化



4項目を使ってクラスタリングした結果を可視化

