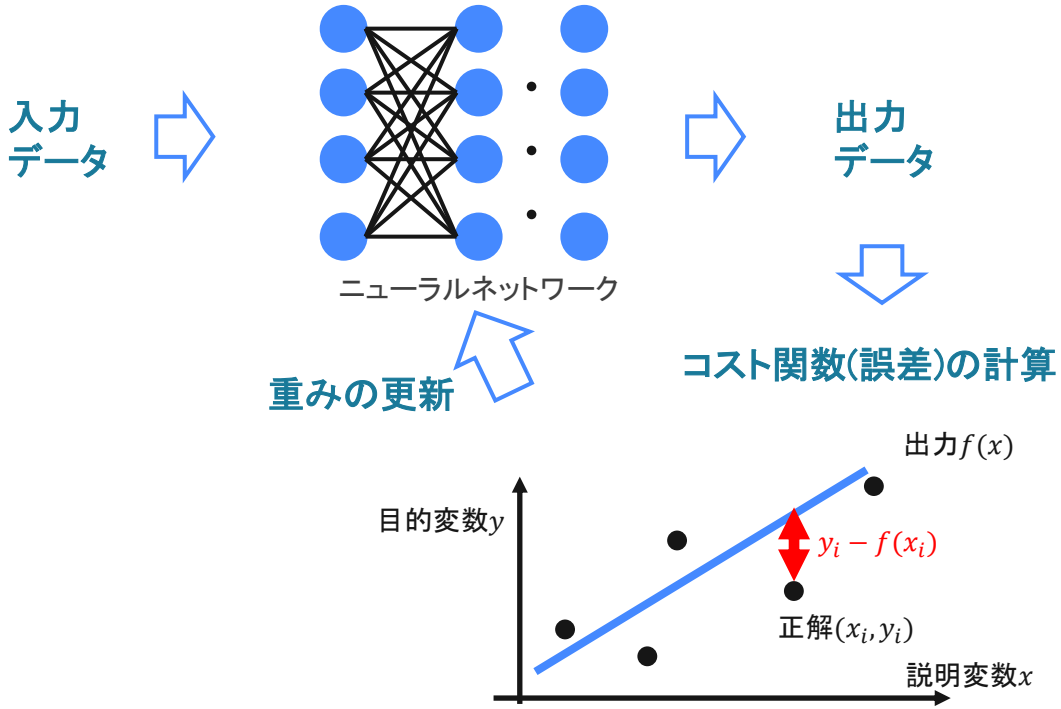
量子ニューラルネットワークとは

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ー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。

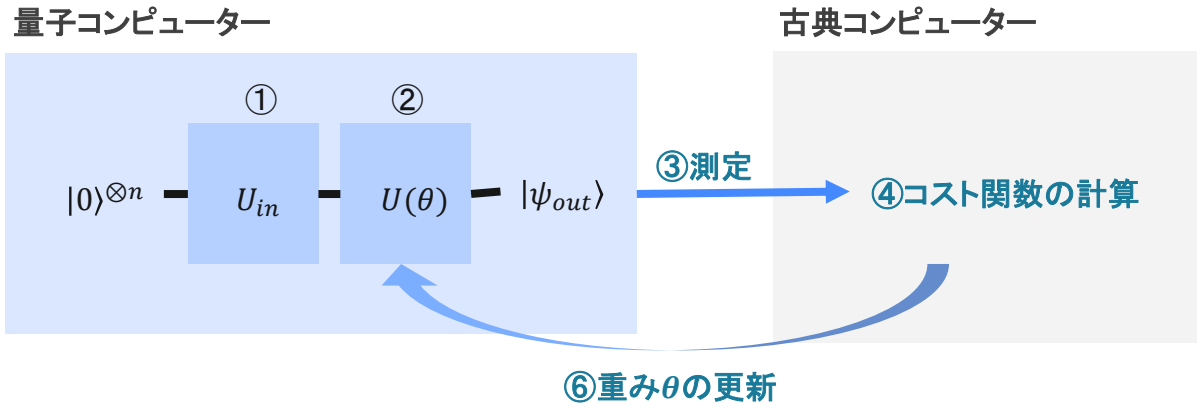
2 古典ニューラルネットワーク(NN)の学習

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。



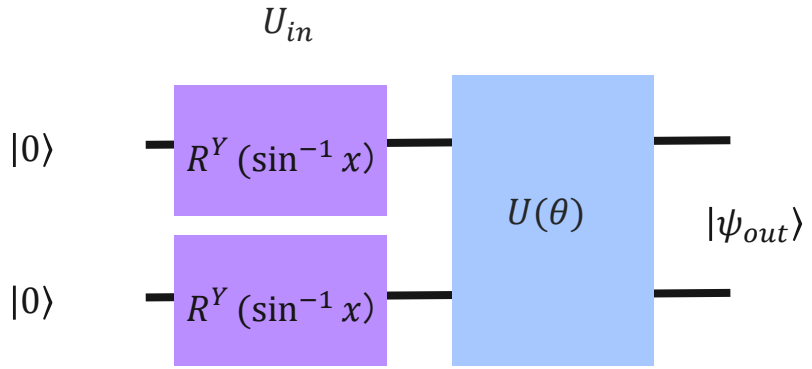
①入力値xをパラメータに持つ量子回路U_{in}を通して量子状態にエンコード
②パラメータθに持つ変分回路を通して出力状態を生成
③量子ビットを測定して出力を求める
④コスト関数を計算する
⑤コスト関数が小さくなるようにθを更新して最適化θ^*を求める

QNNの利点

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。

例



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

入力状態の密度演算子

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

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

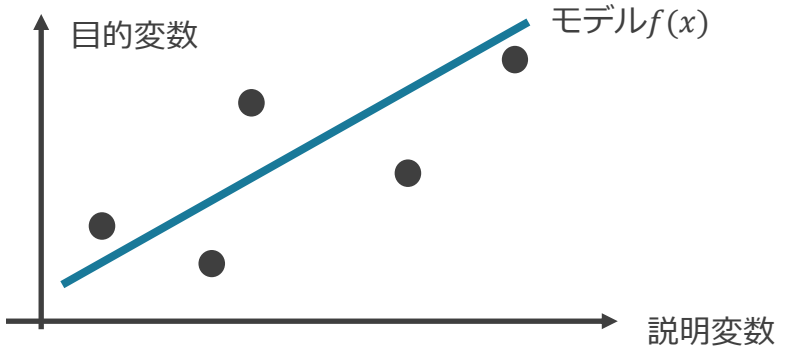
X_1X_2の期待値を計算するとx^2

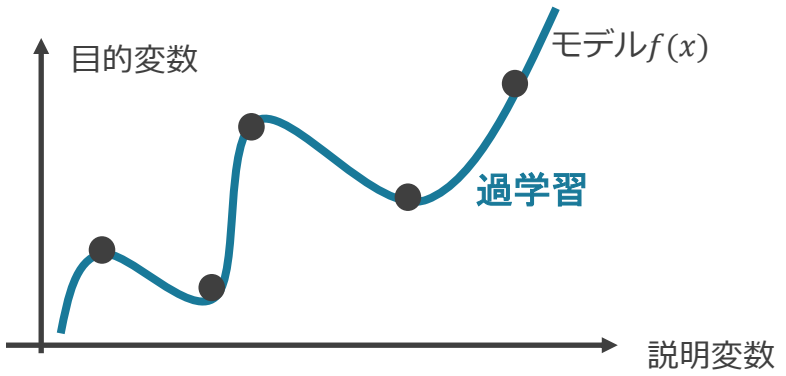
→入力データを非線形関数で変換してエンコーディングが可能

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。

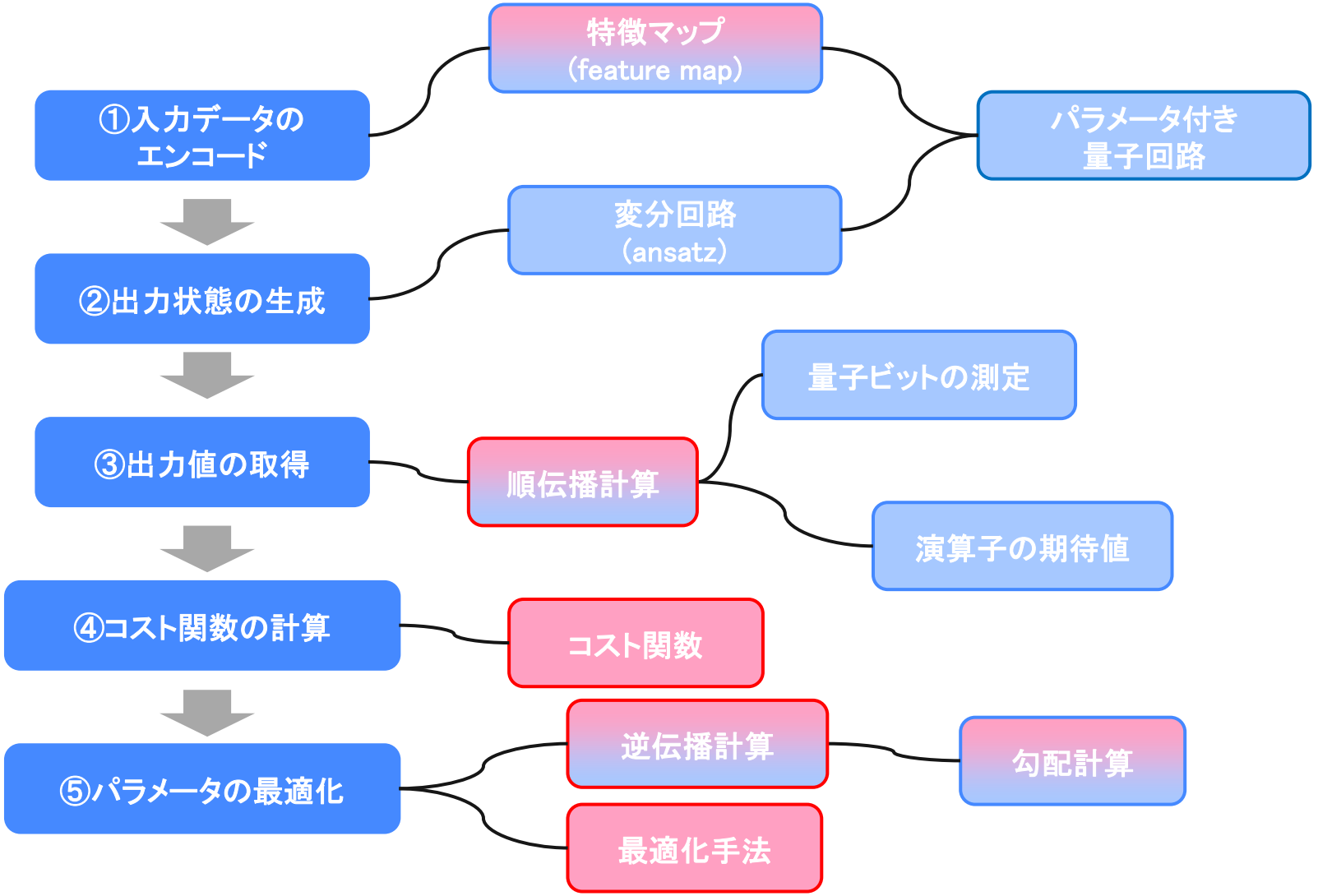
※過学習とは
構築されたモデルがデータに過剰に適合した結果、新たなデータに対する予測器として正しく機能しない現象。
古典計算ではコスト関数に制約を課す工夫(正則化)が必要 となる。





QNN逆引きマップ

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。



AI分野　量子分野

QNNとNNの順伝播計算・逆伝播計算

1 QNNとNNの順伝播計算・逆伝播計算の対応

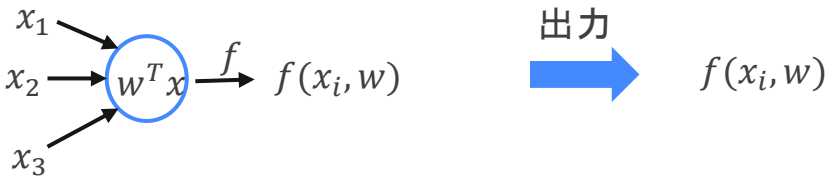
QNNとNNにおける順伝播・逆伝播計算は以下のように対応する。

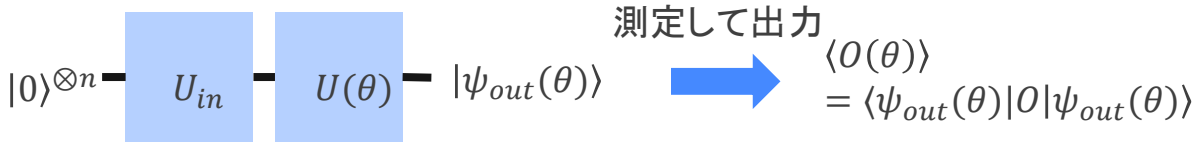
- 順伝播計算
  入力から出力を求める
- 逆伝播計算
  重みパラメータ更新のために必要なコスト関数の勾配を求める

	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

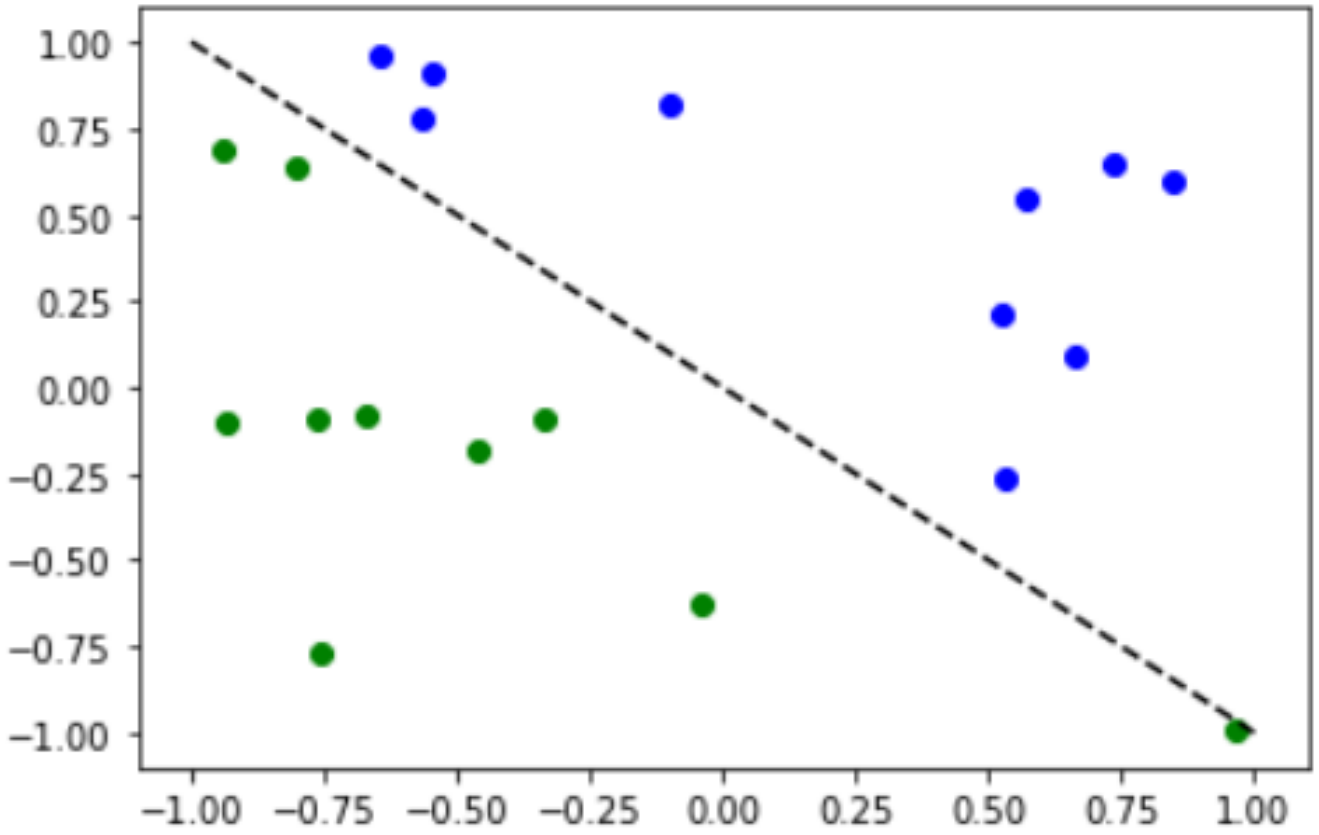
重みを更新するために、コスト関数の勾配を求める。
量子回路での勾配計算は量子の性質を利用していくつか考案されている。

例：パラメータシフト法

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

【実装例】QNNを使った分類タスク

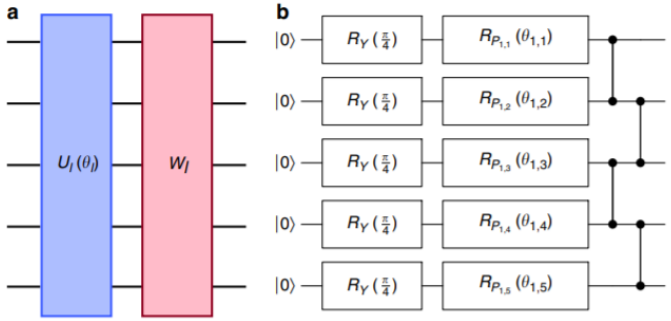
ランダムに生成した20サンプルを$y = -x$を境界に分類する。
→　実装は、「応用レベル：テキスト：3-2-2_量子ニューラルネットワーク_実装1.ipynb」（またはPDF）を参照。



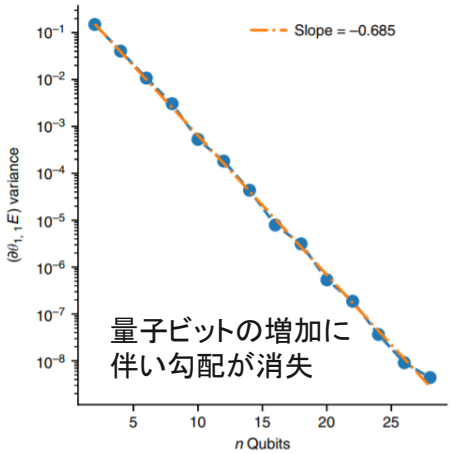
QNNの課題

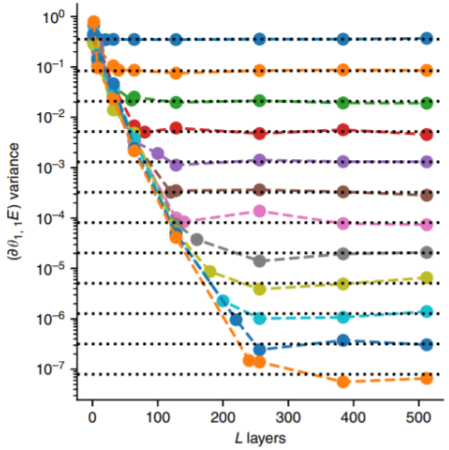
1 barren plateaus問題

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



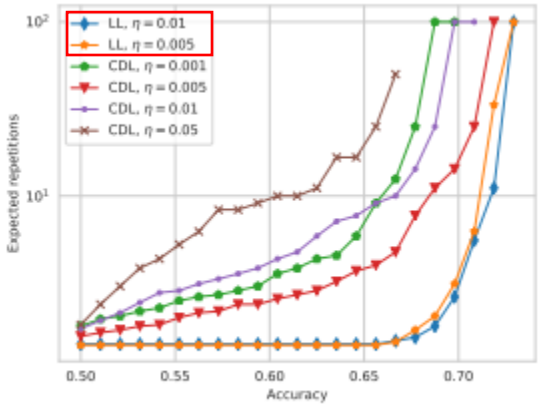


回路の深さの
増加に伴い
勾配が消失

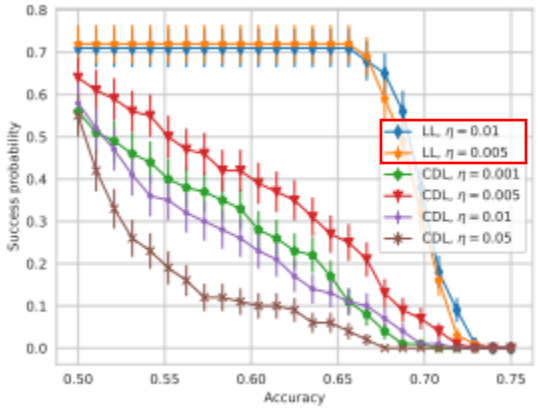
McClean, J.R., Boixo, S., Smelyanskiy, V.N. *et al.* *Nat Commun* **9,** 4812 (2018).

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).

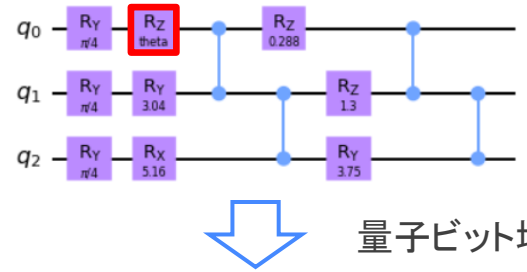
量子機械学習 barren plateaus問題の深堀り

1 論文の実装を再現する

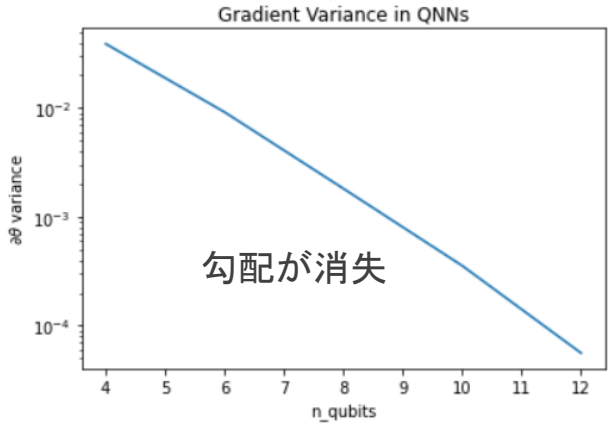
2018年に報告されたbarren plateaus問題を量子SDKであるQiskitを利用して、量子シミュレータ上において論文の報告内容の再現を試みた。

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



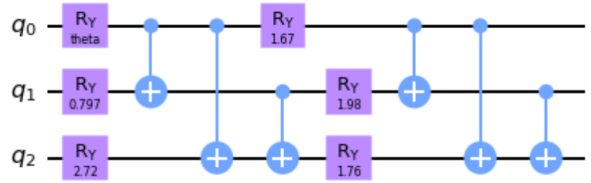
2 別の回路でも再現することを検証する

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。

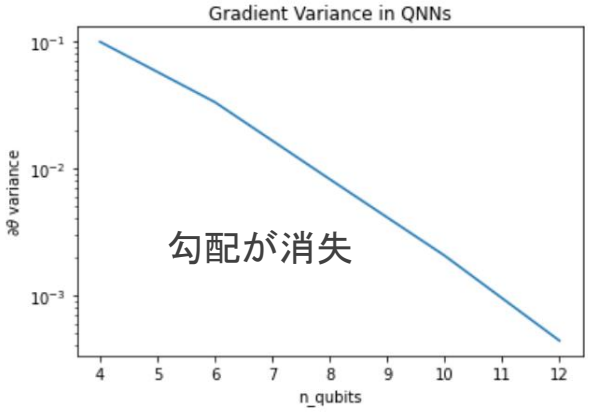
量子回路には量子ニューラルネットワークによく利用されるReal Amplitude回路を使用し、左と同条件で勾配を計算した。

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



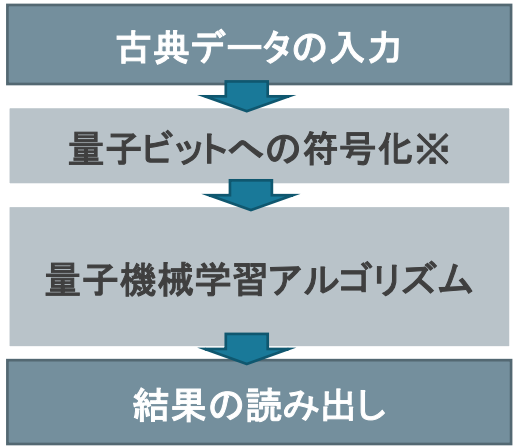
量子機械学習 クラスタリングへの応用

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**

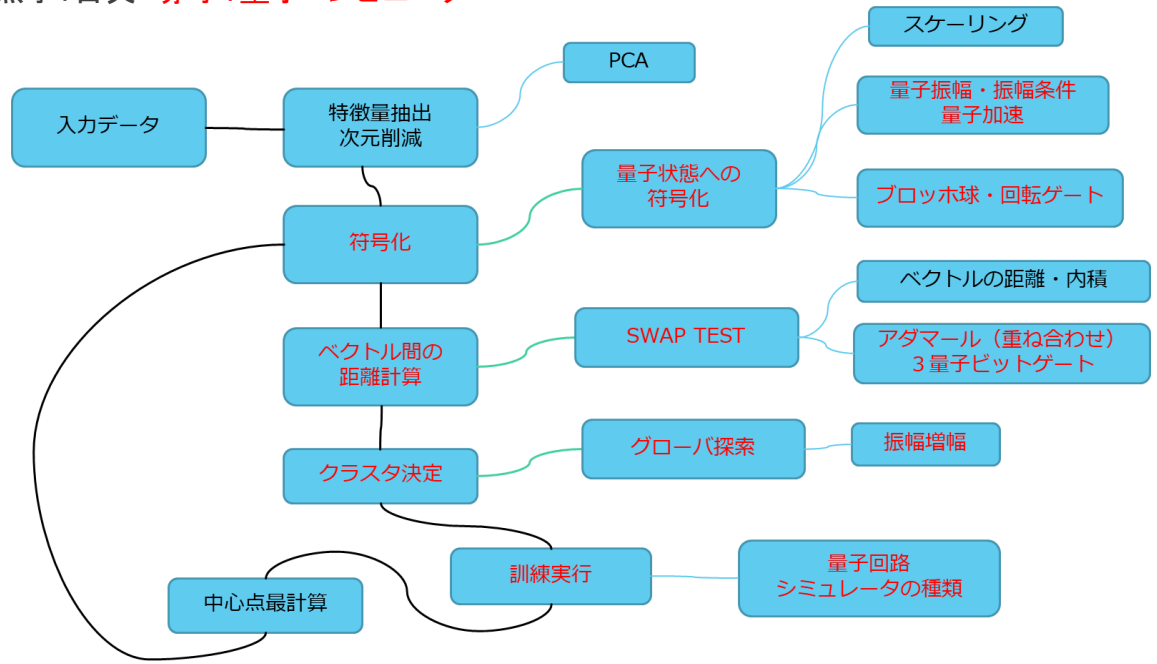


※現在の量子コンピュータには量子状態でデータを保存しておく記憶デバイスがない。
古典データの入力は、量子ビットに符号化するステップが必要とされる。

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



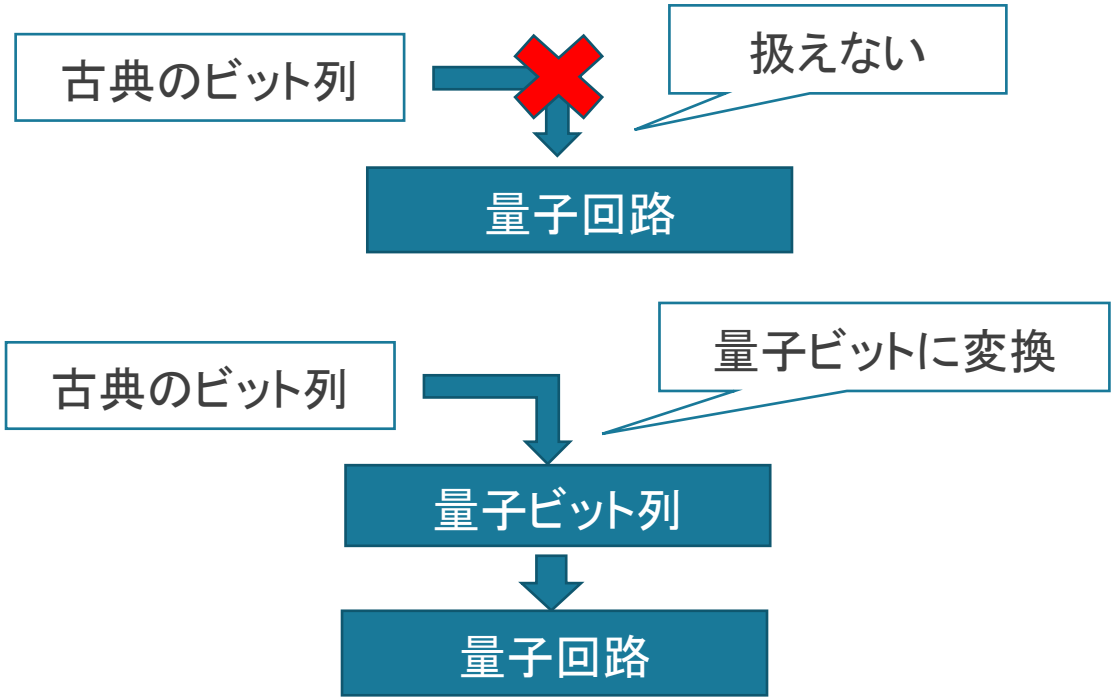
3 参考資料

「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

古典データを量子ビットに符号化する

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・出力ができると考えられている。

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） ⇒ 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）
  ⇒ 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（θ 、ϕ）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

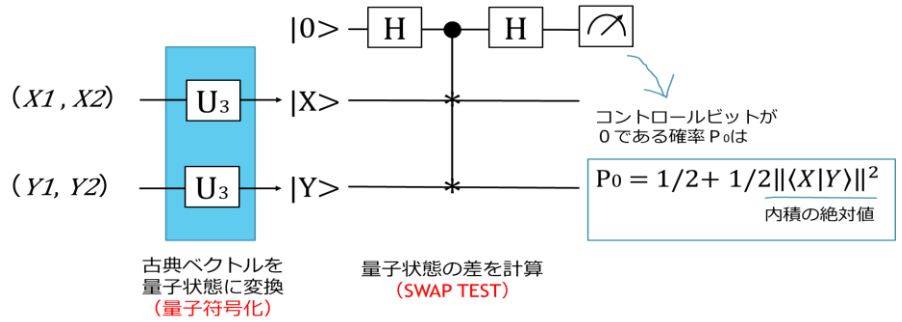
ベクトルの距離計算を量子回路で行うには

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。

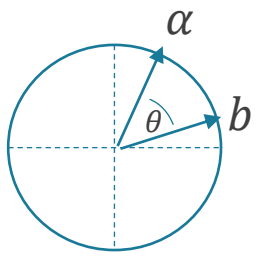
➢ **SWAP TESTの手順**

- 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 θ 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

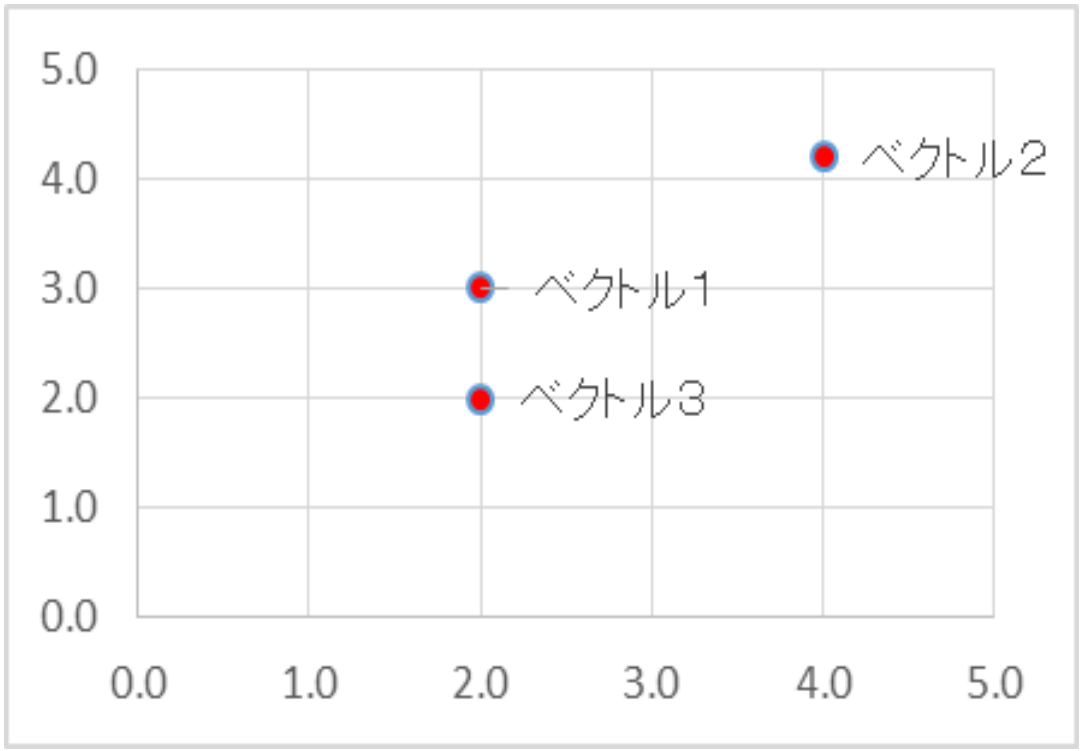
3　参考資料

「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

量子回路での内積計算−サンプルコード

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が近い

D（ベクトル1・ベクトル3）
＝ $\sqrt{(2-2)^2}+(3-2)^2=\sqrt{1}$

D（ベクトル2・ベクトル3）
＝ $\sqrt{(4-2)^2}+(4.2-2)^2=\sqrt{8.84}$

- 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

量子回路での内積計算−サンプルコード①


```
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])
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)
```

0～1にスケーリング

角度に変換
（0＜θ＜π）
（0＜φ＜2π）

```
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）

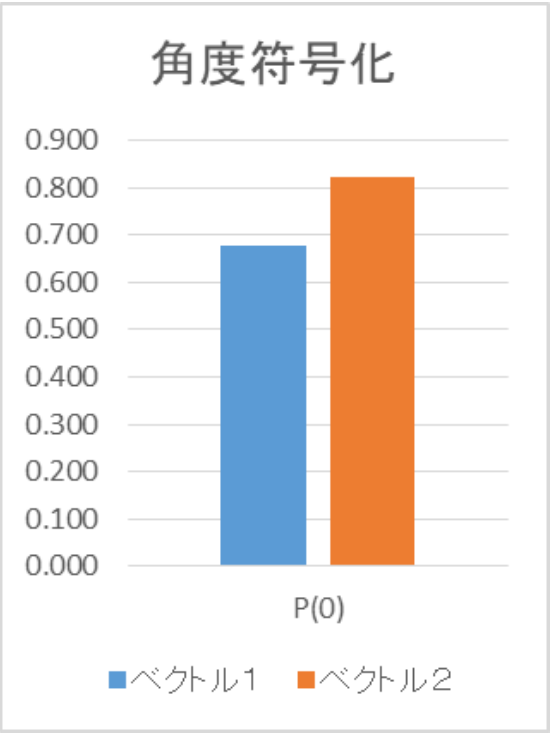
角度に変換したベクトルをU3回転ゲートのパラメータとして作用させる

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θ、φ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。



量子回路での内積計算−サンプルコード②


```
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_nl[2,:]
for i in range(2):
  initial_state_q1 = x_nl[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)
```

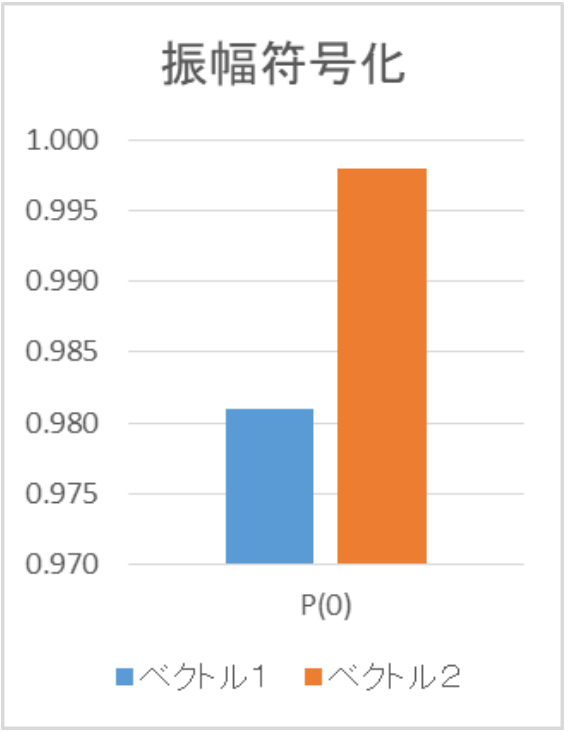
量子ビットを用意（3量子ビット）

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。

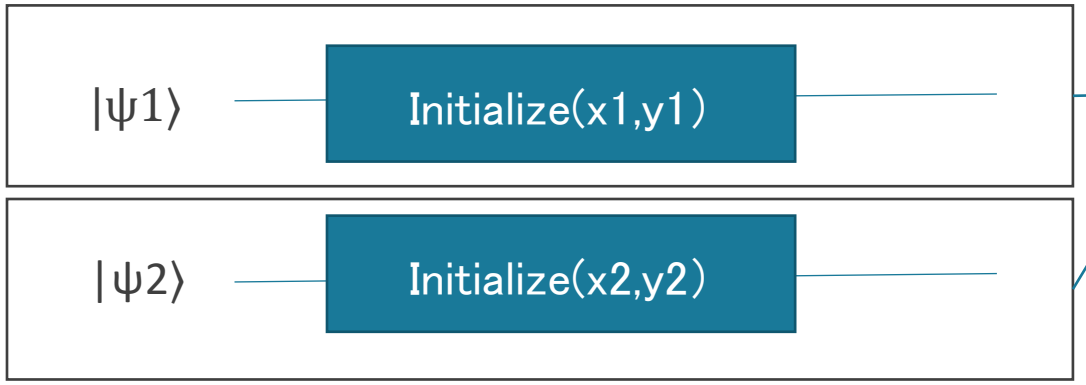
・ベクトル1、2とベクトル3のSwap testの結果、P_0の確率から、ベクトル2の方が近い算出になる。
・角度符号化と同じく、**ベクトルの向き**　による判定となる。



量子状態の類似性の評価：State_Fidelityを使う

- 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

$$F(|\psi 1\rangle,|\psi 2\rangle)=|\langle\psi 1|\psi 2\rangle|^2$$



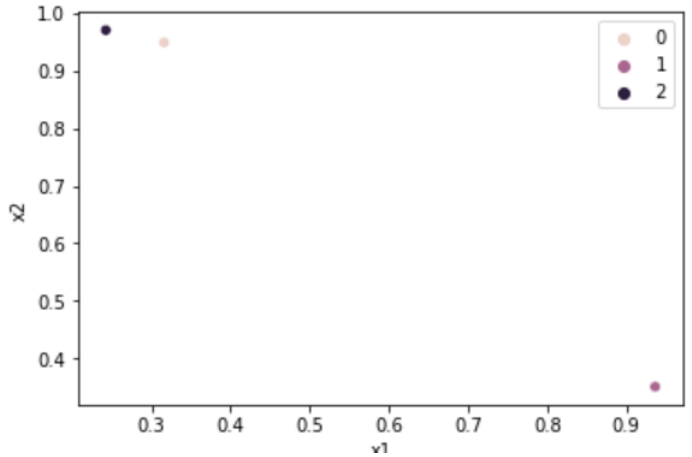
回路実行し、量子状態を得る

```
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)
```

振幅値に変換した3つの2次元ベクトル



・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+0.j 0.9486833 +0.j]
Vector1 =[0.93632918+0.j 0.35112344+0.j]
Vector2 =[0.24253563+0.j 0.9701425 +0.j]

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そう。

量子機械学習 クラスタリングへの応用 – 課題

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？ （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。

・クラスター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、今後検証をしてみる。**

3 参考資料

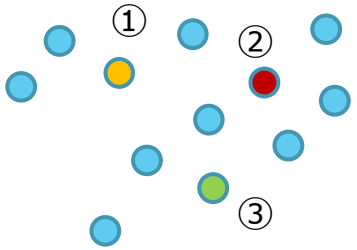
「量子コンピュータによる機械学習」（共立出版）
https://qiskit.org/documentation/
https://github.com/Morcu/q-means

量子回路を使ったクラスタリング・サンプルコード

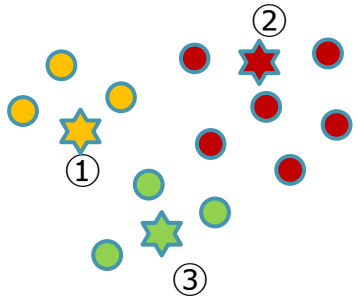
例）K平均法でのクラスタリングを量子回路で実行してみる。

➢ 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の上限で終了）



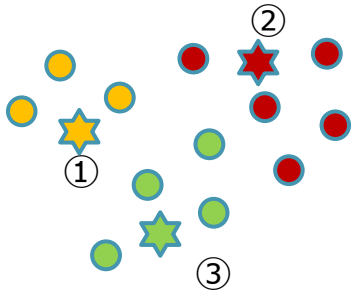
①クラスター数を決める
②初期の中心点を決める



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

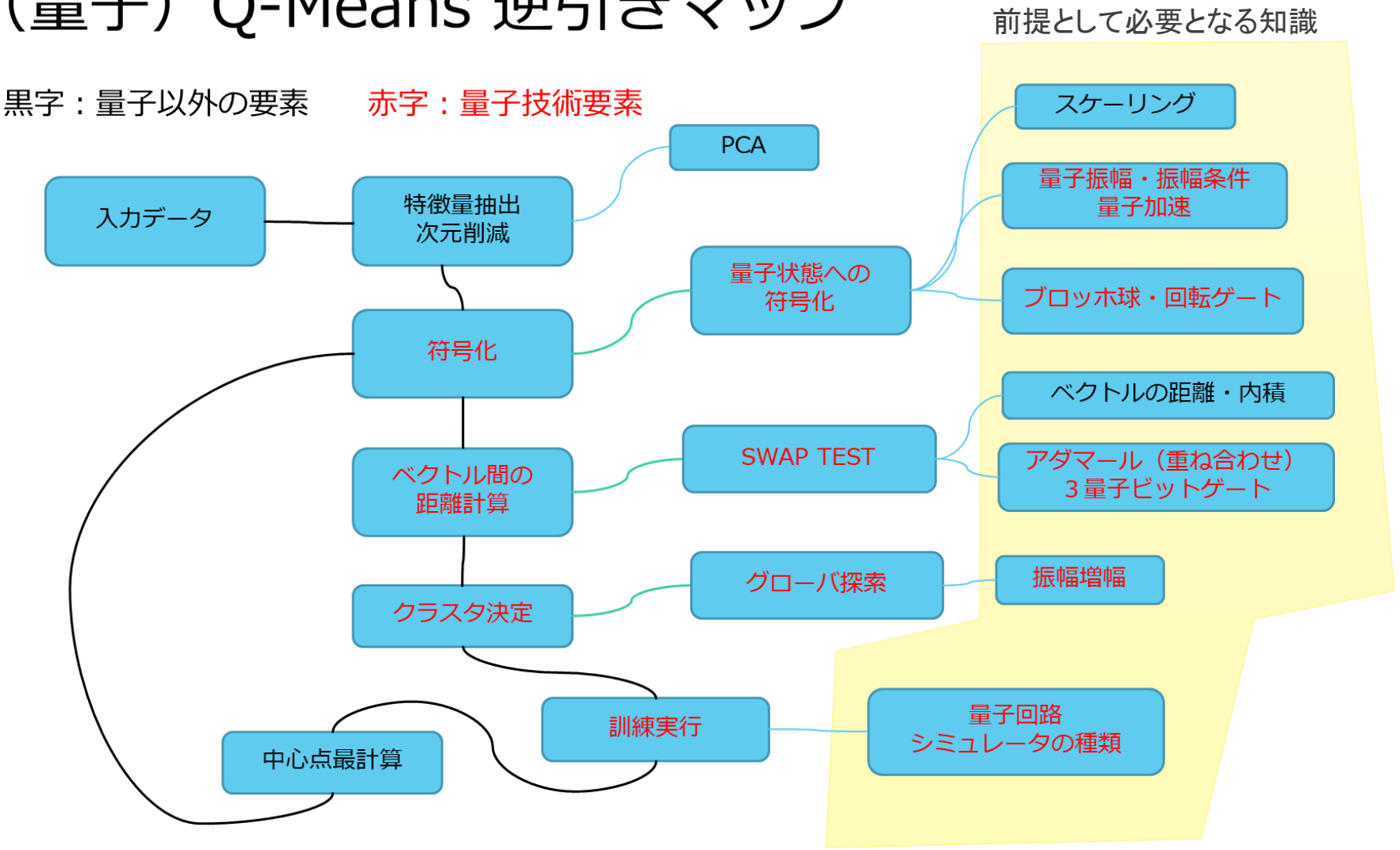


⑤　③④を繰り返し、中心点の変動がなくなったら終了

量子回路を使ったクラスタリング・サンプルコード

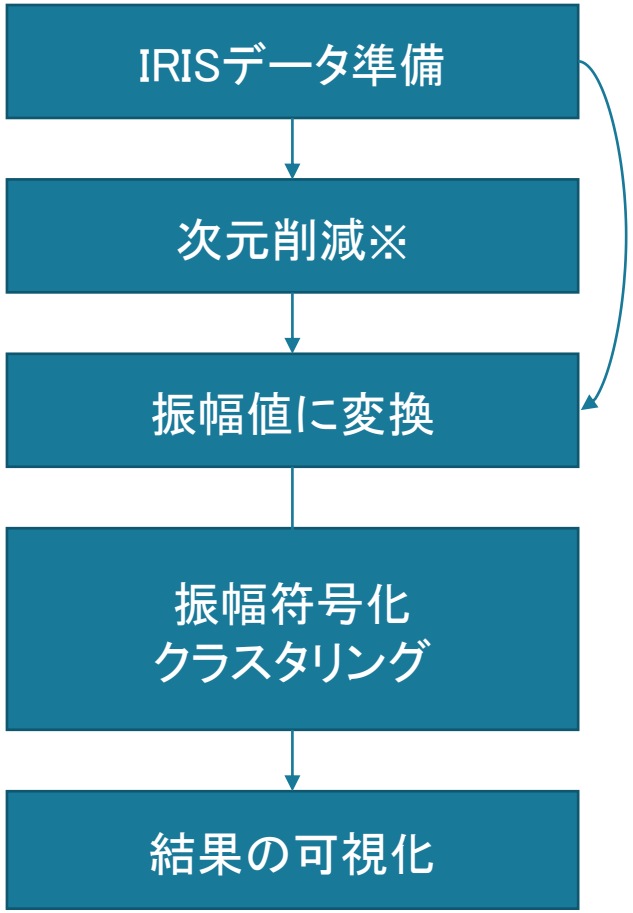
- 量子回路を使ったクラスタリング（Q-Means）のアルゴリズムと前提となる量子コンピュータ学習要素を逆引きマップとした。
  前提となる要素知識については基礎編で確認してもらいたい。

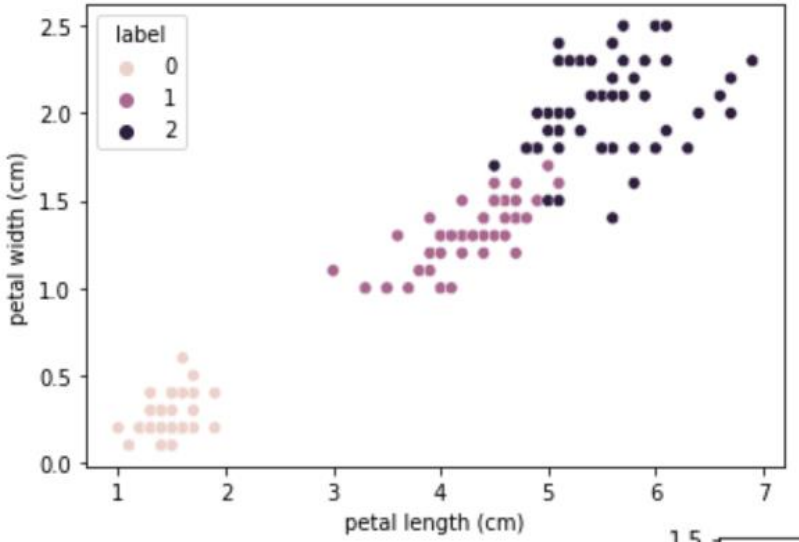
（量子）Q-Means 逆引きマップ



量子回路を使ったクラスタリング・サンプルコード

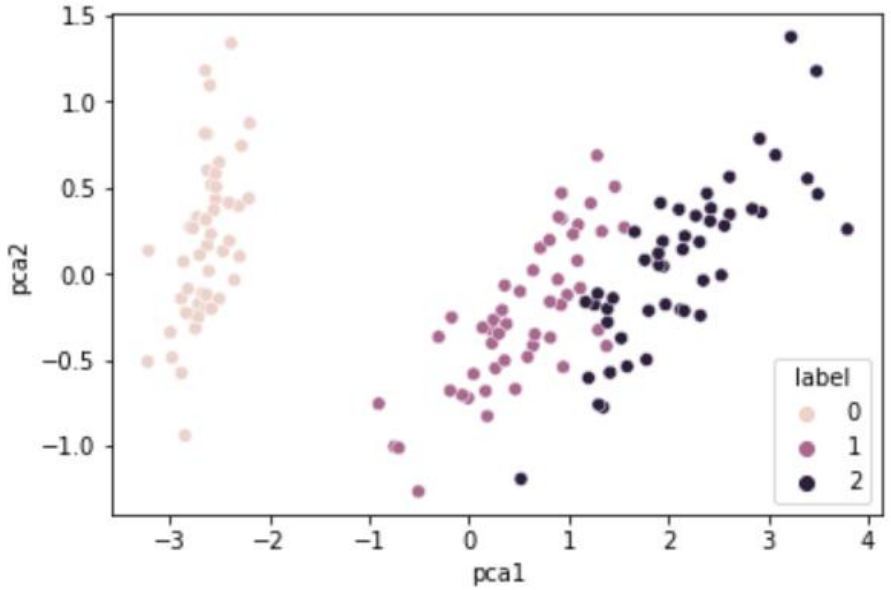
IRIS（あやめ）サンプルデータを使ってクラスタリングをしてみる。





2項目を使って、クラス分けを可視化した。
（正しい分類）

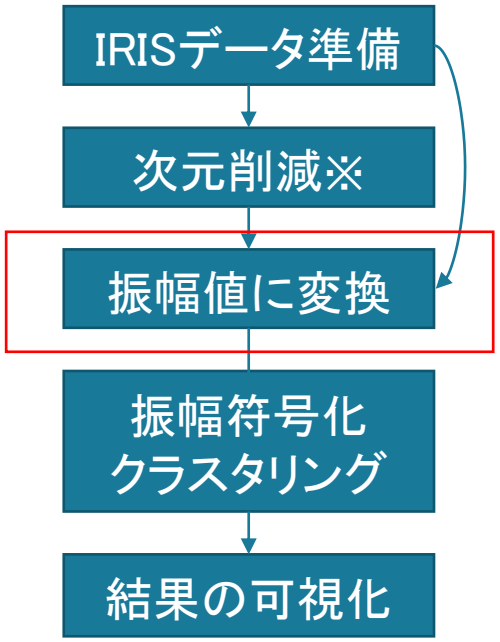
IRISデータは4項目あるので
2つの主成分に次元削減



※多次元でも表現できるため、次元削減は必須ではない

量子回路を使ったクラスタリング・サンプルコード

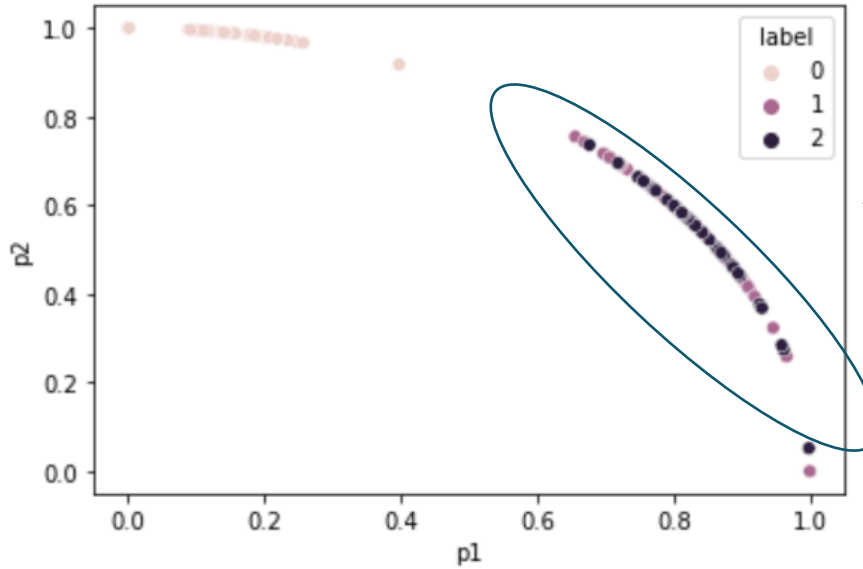
2つの主成分（2次元）のデータを振幅値変換する。



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を満たすように正則化する。

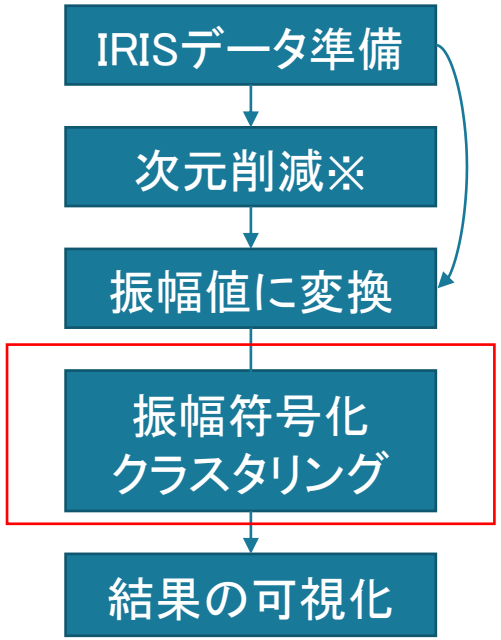


主成分の振幅値変換後を可視化

Label 1と2のデータはクラスタリングが難しそうである。

量子回路を使ったクラスタリング・サンプルコード

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

- 量子回路の準備（qbit＝3）※
  コントロールビット：1
  入力データビット：1
  中心点データビット：1
  クラシカルビット：1
- 入力データとクラスターの中心点初期値を量子符号化
- SWAP TEST
- コントロールビットの測定
- 回路実行、測定値からクラスター判別
- 新しい中心点をクラスター平均から算出

※2次元のデータを前提としたqbit数

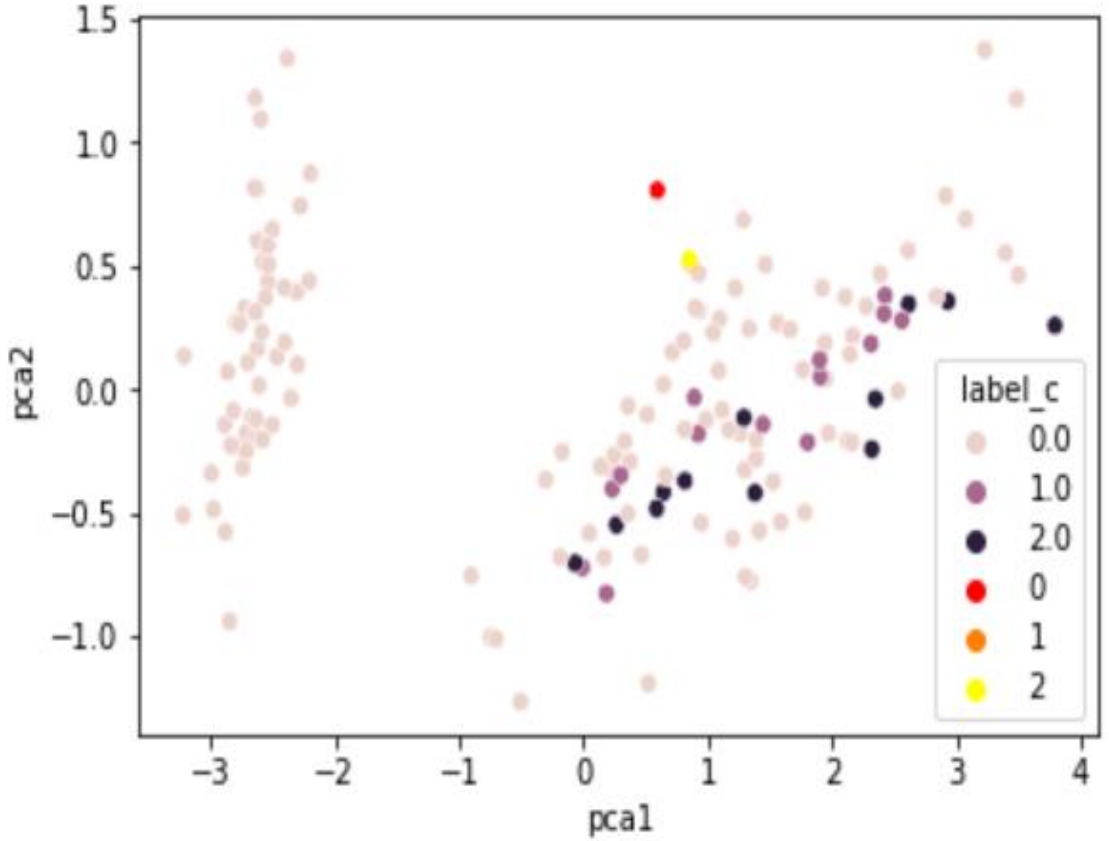
State_Fidelity による判別

- 量子回路の準備（qbit＝1）※
  入力データビット：1
- 入力データとクラスターの中心点初期値を量子符号化
- 回路実行
  入力データとクラスターの中心点初期値の量子状態の取得
- 入力データと各中心点の量子状態の一致性を評価（state_Fidelity）
- Fodelityでクラスターを判別
- 新しい中心点をクラスター平均から算出

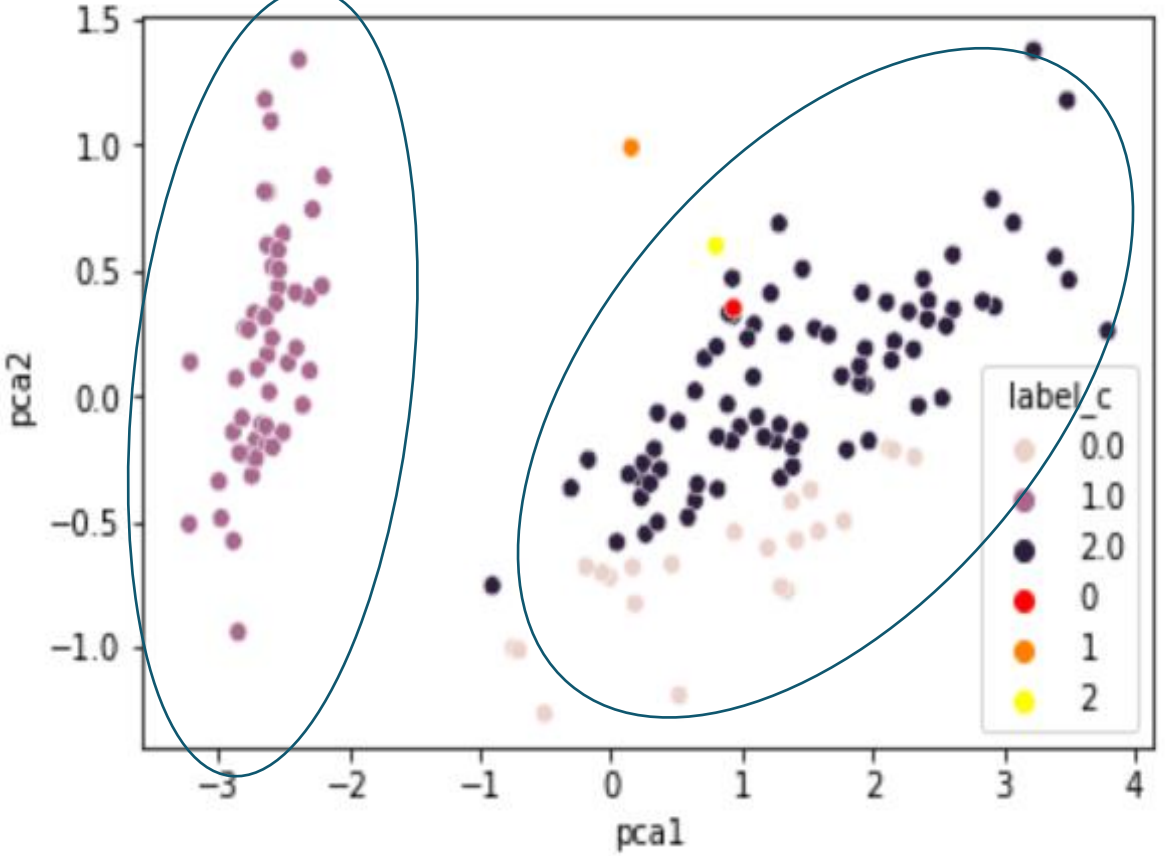
量子回路を使ったクラスタリング・サンプルコード

- 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

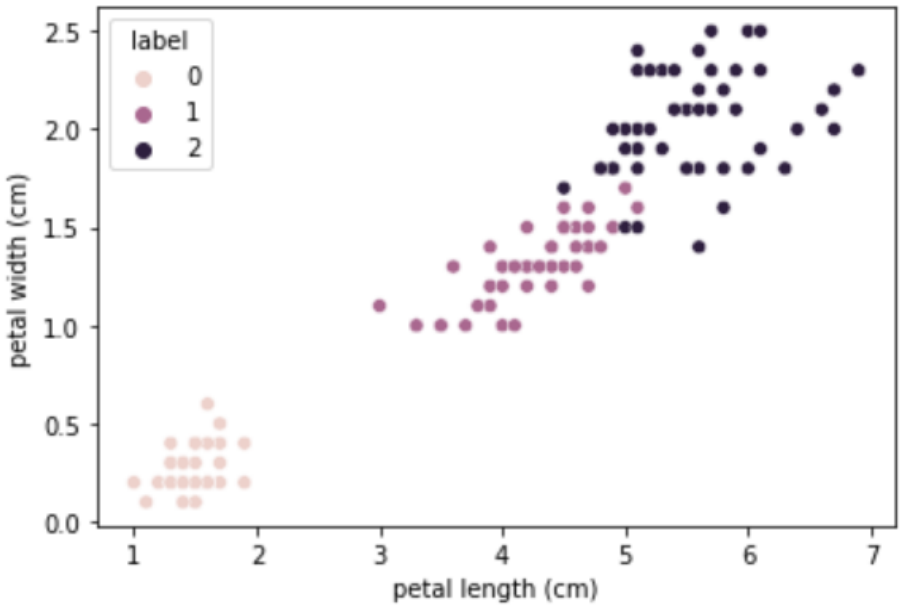


量子回路を使ったクラスタリング・サンプルコード

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