日本語訳『Qiskit Textbook Machine Learning』勉強会

・Quantum feature maps and kernels

Daiki Murata

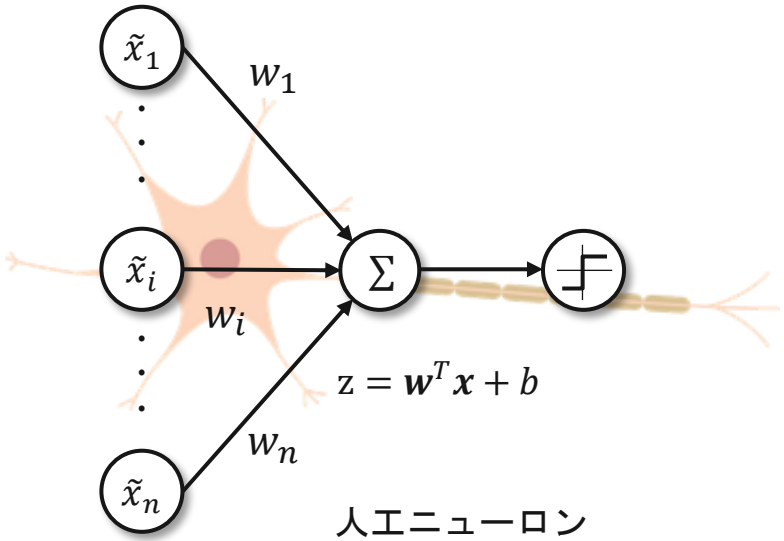
Qiskit Advocate / Senior Architect - IBM Consulting

本日のテーマ

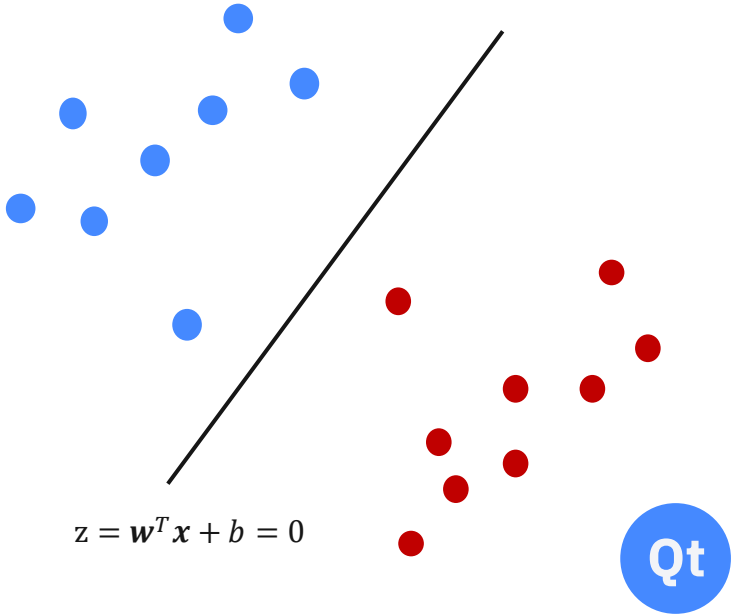
- 「カーネル」に関する考え方を少しでも理解する
- 量子計算におけるカーネルアルゴリズムの立ち位置の感覚を掴む

機械学習における分類問題

分類問題とはサンプルの境界を決定するタスクのことで、最も単純な問題では線形関数により学習モデルが構築されます。



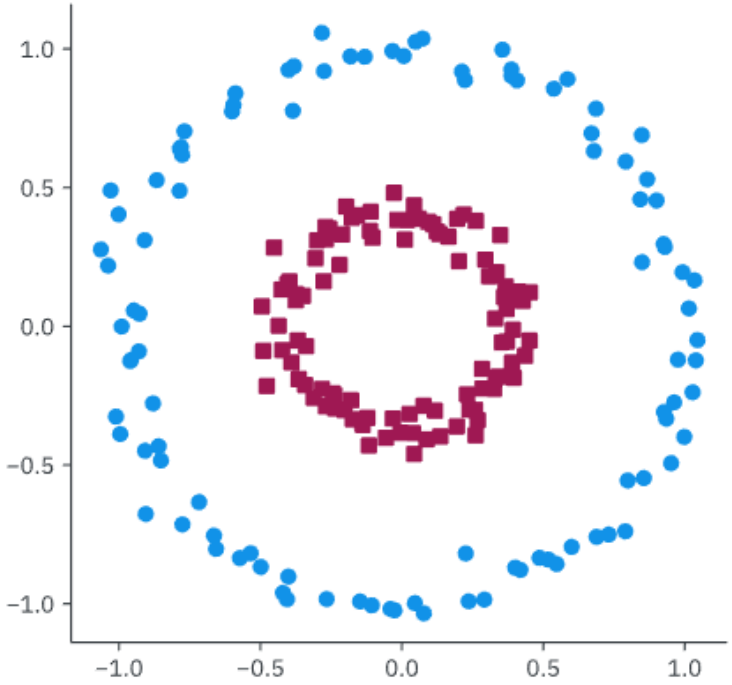
人工ニューロン
(パーセプトロン)



線形モデルの壁

しかし、全ての問題が線形に分離できるわけではありません。

なるべく簡単に境界を決定するためにはどうすればよいでしょうか。

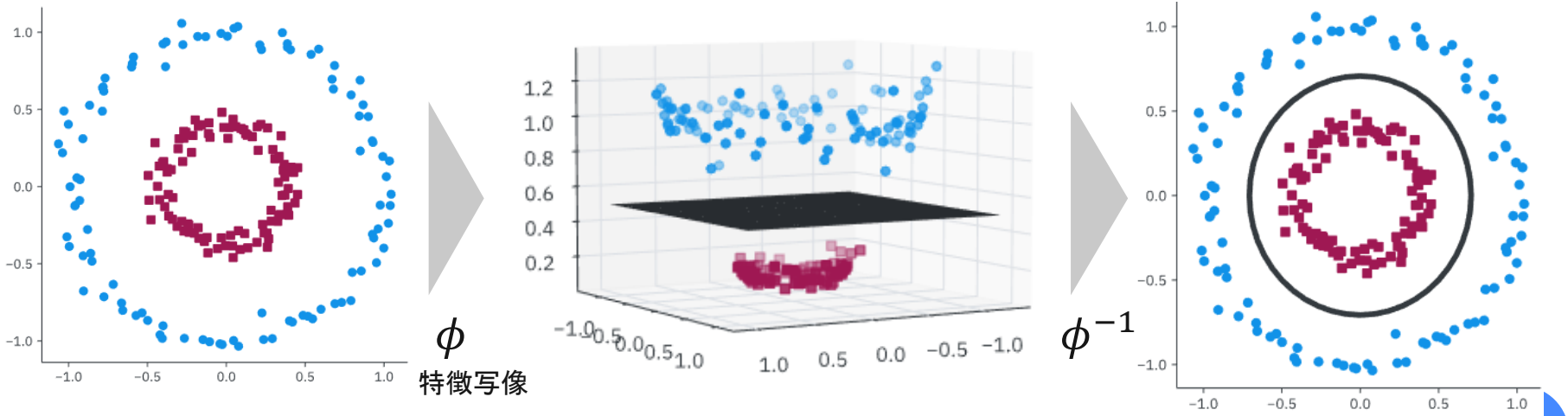


特徴写像

サンプルを高次元空間へ写像(2次元→3次元)すると、線形分離できそうです。

$$\phi(x_1, x_2) = (x_1, x_2, x^2 + y^2)$$

しかし、高次元の特徴空間への射影計算はコストが高くなってしまいます。。。



カーネル法

カーネル法とは「**計算コストを上げることなくデータを高次元の特徴空間に埋め込む**」ことで分析を容易にする手法です。

※カーネル法と調べると...

『ベクトルの内積$x_i^T x_j$をカーネル関数に置き換えるだけでよい』

『カーネルとはサンプル間の類似性を表す関数』

『よく使われるカーネル関数は動径基底関数カーネル...』

よくわからないけど暗記という方も多いのではないでしょうか？

カーネルの定義

【定義】

$$K_{m,m'} = \kappa(x^m, x^{m'})$$

の成分を持つ行列Kが半正定値となるものをカーネルとおく。

※半正定値

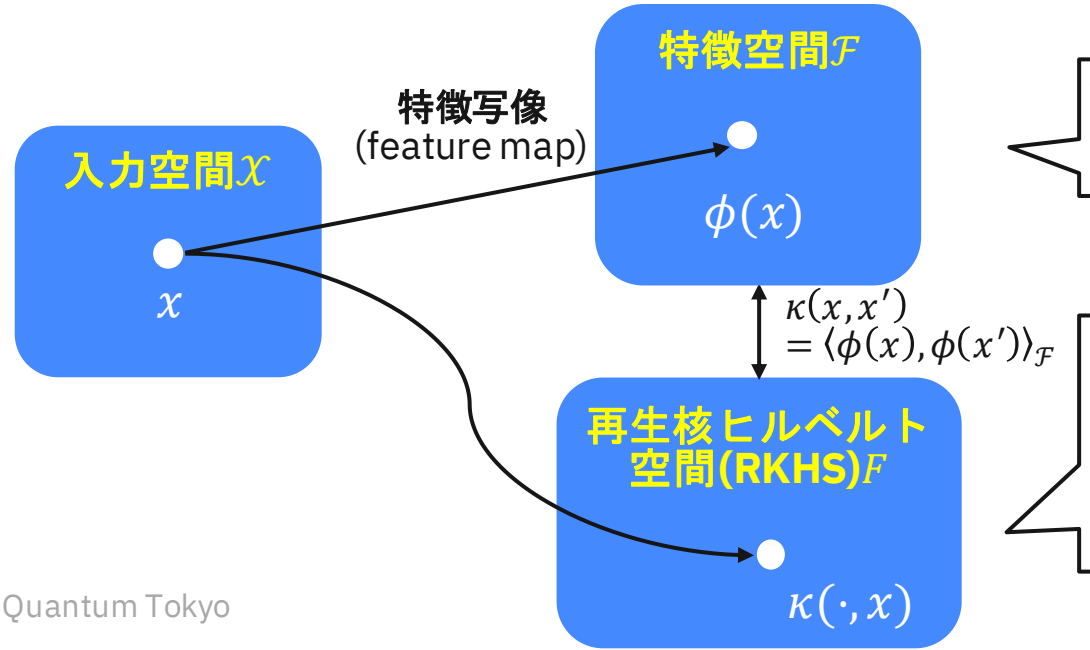
$$\kappa\left(x^m, x^{m'}\right) \geq 0$$
$$\kappa\left(x^m, x^{m'}\right) = \kappa\left(x^{m'}, x^m\right)$$

カーネルを取り巻くロジック

特徴写像を再生核ヒルベルト空間への写像とみなす($\phi(x) = \kappa(\cdot, x)$)と、

特徴空間内の内積計算を入力空間のカーネルで置き換えられる性質がわかります。

$$\kappa(x, x') = \langle \kappa(\cdot, x), \kappa(\cdot, x') \rangle = \langle \phi(x), \phi(x') \rangle$$

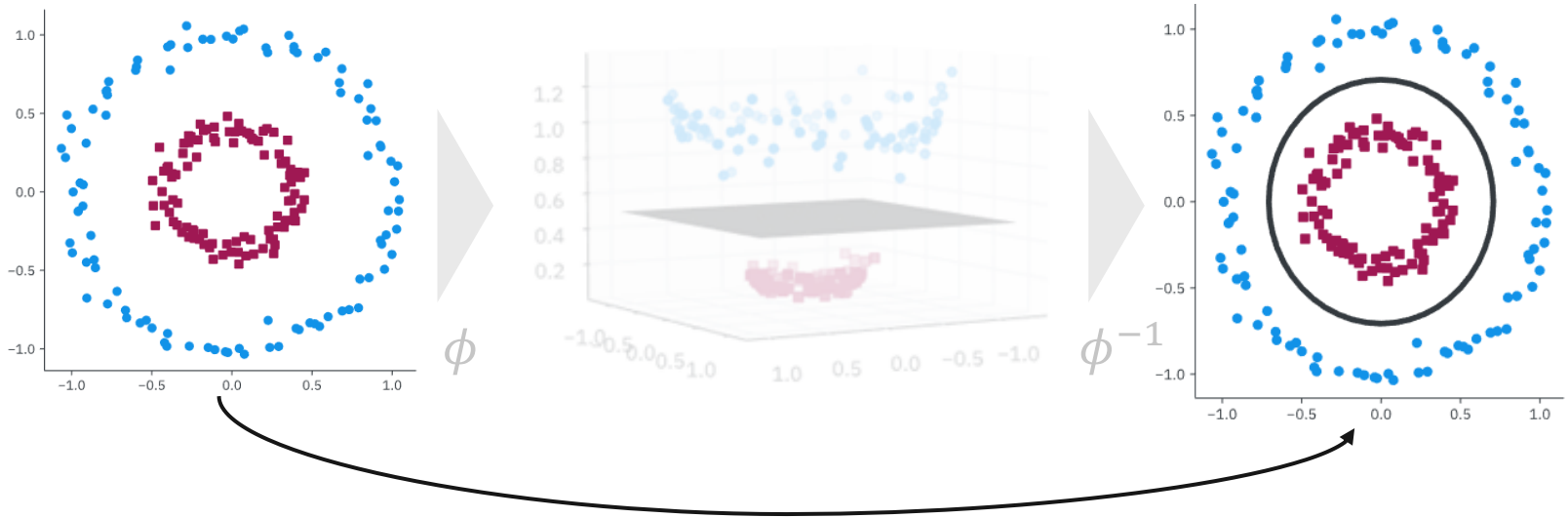


特徴写像ϕが与えられるとき、
$$\kappa(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$$
で表される関数$\kappa(x, x')$はカーネルである。

【Moore-Aronszajnの定理】
カーネルは以下を満たす入力空間上のヒルベルト空間が一意に決まる
$$f(x) = \langle f, \kappa(\cdot, x) \rangle$$
つまり、
$$\kappa(x, x') = \langle \kappa(x, \cdot), \kappa(\cdot, x') \rangle$$

カーネルトリック

高次元の特徴空間への写像を実現しつつ、高コストな写像計算$\phi(x)$を避けてモデルを構築する手法をカーネルトリックと呼びます。



表現定理

あとは最適化の結果構築される学習モデルがカーネル関数で表現出来ればOKです。

特定の形式を持つ教師あり学習タスク(経験損失最小化問題)では、学習モデルはカーネル関数で表されることが数学的に知られています。

$$f_{opt}(x) = \sum_{m=1}^{M} \alpha_m \kappa(x^m, x)$$

振り返ってみると...

- 『ベクトルの内積$x_i^T x_j$をカーネル関数に置き換えるだけでよい』

特徴空間への写像を考えると、$\langle x_i, x_j \rangle \rightarrow \langle \phi(x_i), \phi(x_j) \rangle$なのでカーネル関数で計算可能(カーネルトリック)

- 『カーネルとはサンプル間の類似性を表す関数』

特徴空間における2つのサンプルの「距離」を表現する指標ととらえることが出来ますね。

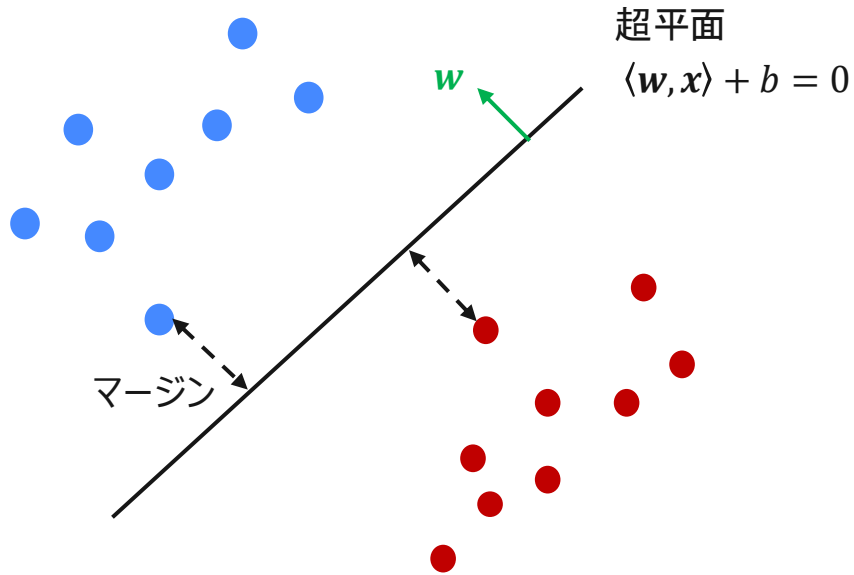
- 『よく使われるカーネル関数は動径基底関数カーネル...』

解くべき問題に応じてよく機能する代表的なカーネル関数がいくつか知られています。

SVM(サポートベクターマシン)

SVMはグループ間の分類境界となる超平面を決定するアルゴリズムです。

マージンを最大化するように最適化することで汎化性能を高めることが目的です。



カーネル法を用いたSVM

SVMは線形分離出来ないデータセットをそのまま学習出来ないので、特徴写像により高次元空間で決定境界となる超平面を探索します。

特徴空間において解となる超平面は、表現定理を適用するとカーネルで表されます。

$$\sum_i \alpha_i \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x}) \rangle + b = \sum_i \alpha_i \kappa(\boldsymbol{x_i}, \boldsymbol{x}) + b$$

例) $\phi: (x_1, x_2)^T \to \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)^T$

$$\kappa(\boldsymbol{x}, \boldsymbol{x'}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x'}) \rangle = x_1^2 x'^2_1 + 2x_1x_2x'_1x'_2 + x_2^2x'^2_2 = \langle \boldsymbol{x}, \boldsymbol{x'} \rangle$$

特徴写像を計算せずとも線形カーネルを用いて決定境界が計算可能です。

量子特徴マップの定義

量子特徴マップは文脈によって定義の仕方が異なる場合があります。

よく使われる定義は以下の2つです。

①Diracベクトルを特徴ベクトルとして定義

$$\phi_1 : x \rightarrow |\phi(x)\rangle$$

②密度行列を特徴ベクトルとして定義

$$\phi_2 : x \rightarrow \rho(x)\ (= |\phi(x)\rangle\langle\phi(x)|)$$

①と区別するためにData-encoding feature mapと呼ばれることもあります

量子特徴量マップ

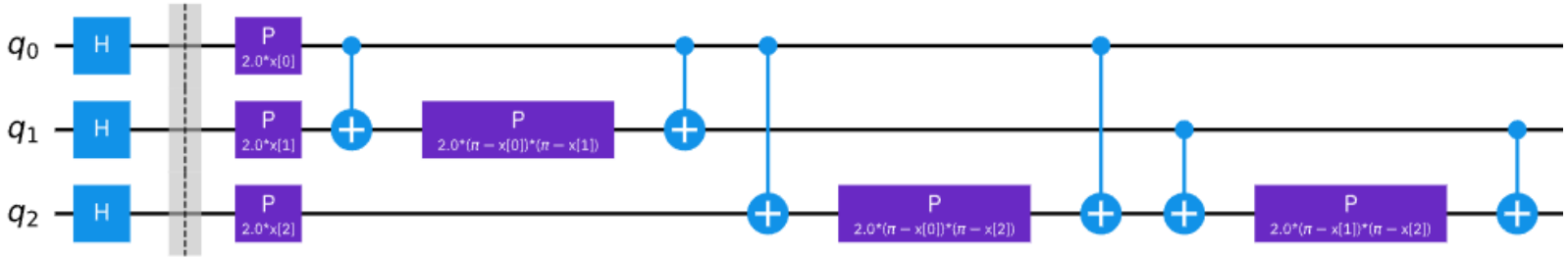
パラメータ化された量子回路では古典データxを量子状態$|\phi(x)\rangle$への変換を通して、

高次元のヒルベルト空間にマッピングする特徴量マップとみなせます。

いかに古典シミュレーションが困難な量子回路を設計できるかが研究の中心です。

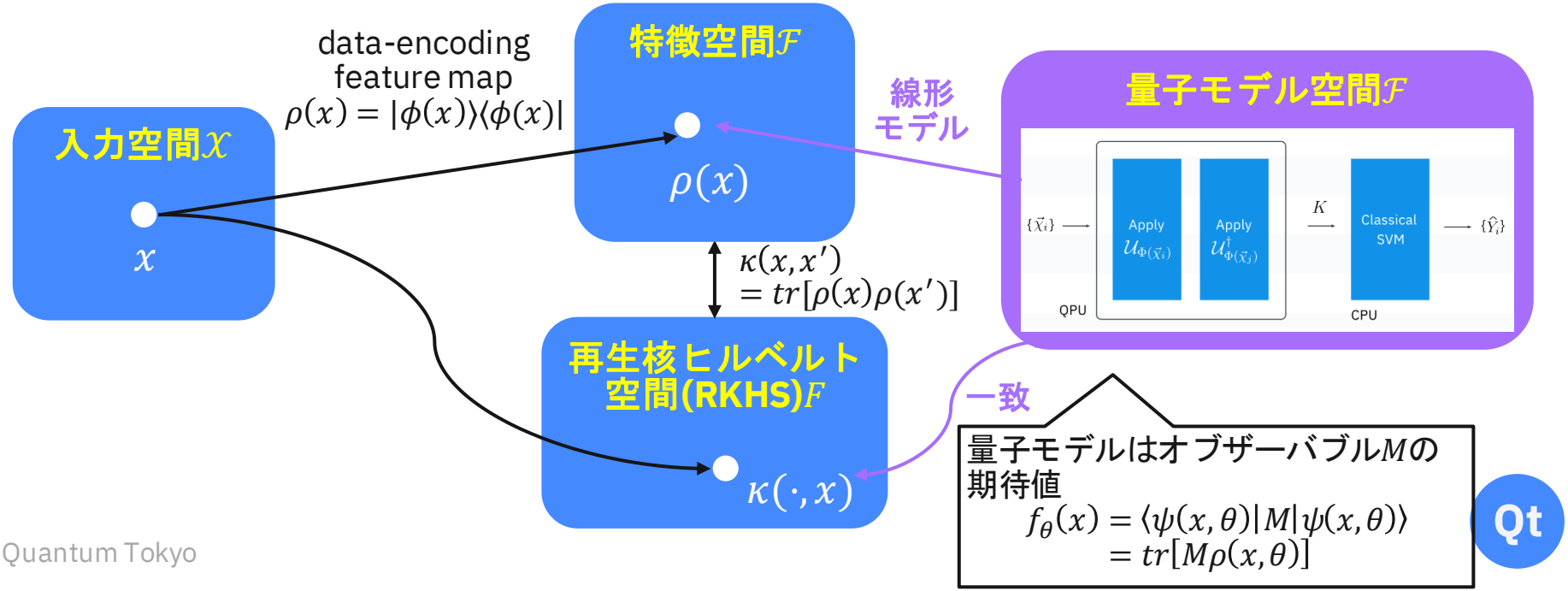
例: ZZFeaturemap

$$\mathcal{U}_{\Phi(x)} = \prod_d U_{\Phi(x)} H^{\otimes n}, U_{\Phi(x)} = \exp(i \sum_{S \subseteq [n]} \phi_S(x) \prod_{k \in S} P_i)$$



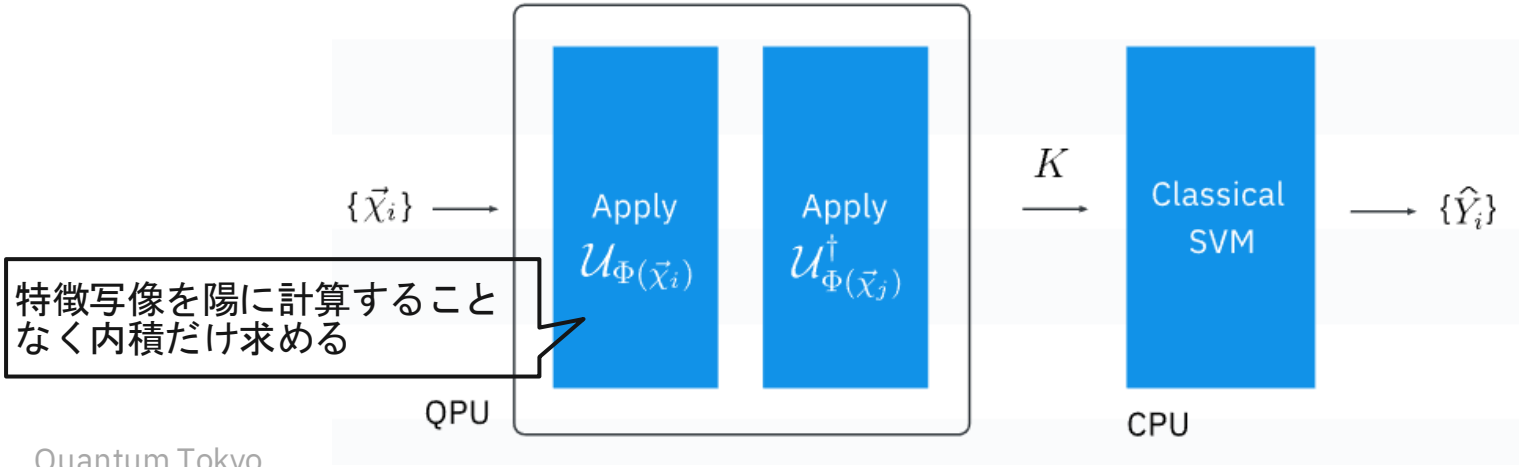
量子カーネル

量子回路のユニタリ変換を特徴量マップとみなすと、カーネルは自然と拡張されます。
量子カーネルが張る空間(RKHS)内の関数は、特徴空間$\mathcal{F}$における線形モデルとなります。



量子カーネルSVM

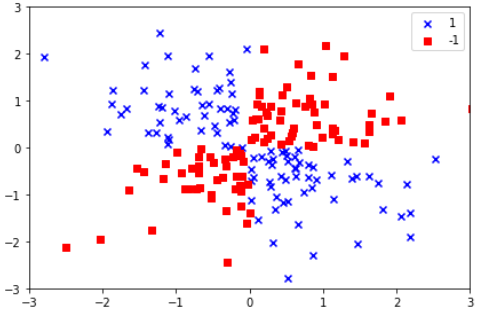
1. 量子カーネルの計算
    1. 訓練データに対して特徴量マップを適用して遷移確率を計算
    2. 訓練データとテストデータに対して特徴量マップを適用して遷移確率を計算
2. 訓練データとテストデータから計算した量子カーネルを古典SVMに適用

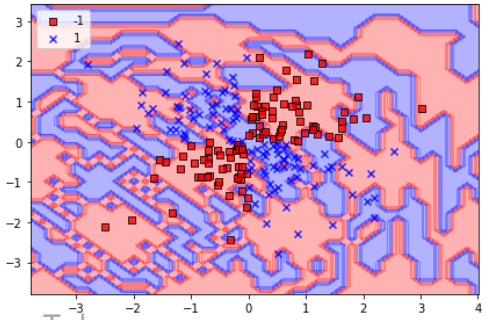


Qiskitでの実装

線形分離不可能な論理XORゲート形式なデータセットを用意して、

QiskitのQuantumKernelを利用して量子カーネルによる学習を実行してみます。






```
## カーネルの定義(手組)
from qiskit import BasicAer
from qiskit_machine_learning.kernels import QuantumKernel
from sklearn.svm import SVC

# Create the quantum feature map
map_zz = ZZFeatureMap(feature_dimension=2,
                      reps=2,
                      entanglement='linear')
# Create the quantum kernel
adhoc_kernel = QuantumKernel(feature_map=map_zz,
                             quantum_instance=BasicAer.get_backend('statevector_simulator'))

# Set the SVC algorithm to use our custom kernel
model = SVC(kernel=adhoc_kernel.evaluate)
model.fit(X_xor, y_xor)
```

※Quantum Challenge 2021 Fall ex3もよいサンプルになると思います

量子カーネルアルゴリズムの立ち位置

①古典機械学習のように必ずしもカーネル法はニューラルネットワークに劣らない

量子ニューラルネットワークの勾配計算はハードウェアの都合で逆伝播計算ではなくパラメータシフトが使われているためモデルによってはカーネル法が優位です。

	カーネルベースの手法		ニューラルネットワーク
古典計算	$\mathcal{O}(M^2)$	$>$	$\mathcal{O}(M)$
量子計算	$\mathcal{O}(M^2)$	$\geq$	$\mathcal{O}(\lvert\theta\rvert M)$

パラメータ数$|\theta|$がデータサイズMに対してどれだけスケールするかでカーネルベースのモデルが優位になり得る

②カーネル計算から最適化まで扱う量子アルゴリズムが原理的には可能なはず