量子サポートベクトルマシン

量子物理学班 (小泉勇樹)

2022 年 5 月 13 日

1　はじめに

機械学習とは，コンピュータが大規模で複雑なデータセットで反復的な学習を行い，画像認識，言語処理を行う上で必要な認知パターンを再現し，そのパターンを適用することで未知の問題へ取り組むことです．その高い予測性能及び汎用性によって，近年高い注目を浴びており，多くの人々にとってテクノロジーの付き合い方を変えたといっても過言ではないでしょう．パソコン，スマートフォンなどに導入され生活に溶け込んでおり，私たちの生活の手助けをしてくれています．

その機械学習の手法の一つにサポートベクトルマシン (SVM) があり，画像，音声認識，言語など様々な分野に適用され，専門家でなくても簡単に利用できるほど有用なデータ解析ツールとなっています．この SVM の手法に，量子コンピュータによる量子計算を応用することによって，機械学習の性能がさらに向上する可能性が示唆されています．

この記事においては機械学習の手法として SVM について着目し，量子力学をある程度学習した読者を想定して量子サポートベクトルマシン (QSVM) について解説します．SVM の手法について簡単な解説を行いますが，SVM に関して基礎的な知識があれば，前半のセクションは読み飛ばしても構いません．

2　2 クラス分類

2.1　初めに

2 クラス分類問題 (binary classification) とは，与えられた入力データが二つのカテゴリーのどちらに属するかを識別する問題です．ここでこのカテゴリーを**クラス (class)** と呼び，クラス推定を行う処理を行うシステムを**分類器 (classifer)** といいます．分類器を構成する基本手法は，処理するデータがどのようなものであるかを考えることは必要なく，一つの手法によって多様なデータを処理することができます．

クラス推定を行う分類器を構成するためにはどうすれば良いでしょうか？例えば猫と犬の画像を識別する問題を考えれば，まず猫と犬の画像をある程度持っていると仮定し，これを訓練用としてコンピュータに猫と犬の分類を学習させます．コンピュータが分類規則を獲得，抽出すれば，新たな画像に対しても分類はうまくいくはずです．このように分類問題とは，**訓練データ (training data)** を使って分類に必要なパターンを獲得する問題と考えることができます．

データの数学的な表記を以下のように定義します．

Def 特徴ベクトル，入力ベクトル

入力の定義域 $\mathcal{X}$，出力の定義域 $\mathcal{Y}$ を持つ訓練入力 $\boldsymbol{x}_i$，目標出力 y_i の直積 $(\boldsymbol{x}_i, y_i)$ をデータの情報として定める．ここで $\boldsymbol{x_i}$ は入力データの特徴を表す数値であり，**特徴ベクトル (feature vector)** と呼ばれる．また，y_i は出力のクラスを表現する**ラベル (label)** と呼ばれる．

以下，断りがない限り $\mathcal{X} = \mathbb{R}^n (n \in \mathbb{N})$ とし，2 クラス分類を考える際には，$\mathcal{Y} = \{1, -1\}$ とします．また，機械学習分野でよく用いられる記法として $\{1, \cdots, n\} = [n]$ を採用します．

2.2 線型 SV 分類

n 個の元からなる訓練データ集合 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i \in [n]}$ を考えた時に，**決定函数 (decision function)** と呼ばれる函数 $f : \mathcal{X} \to \mathbb{R}$ を用いて以下のように分類器 $g(\boldsymbol{x})$ を定めます．

$$g(x) = \begin{cases} 1 & f(\boldsymbol{x}) > 0 \text{ の時} \\ -1 & f(\boldsymbol{x}) < 0 \text{ の時} \end{cases} \tag{1}$$

ここで，$f(\boldsymbol{x})$ として以下の 1 次函数を考えます（このように，線型な決定函数を利用して分類を行うことを線型 SV 分類といいます．）：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b \tag{2}$$

ただし，$\boldsymbol{w} \in \mathbb{R}^N, b \in \mathbb{R}$ は未知の変数であり，これらをどのように最適化するかが問題となります．$g(\boldsymbol{x})$ はその定義より，$f(\boldsymbol{x}) = 0$ が分類結果の変化する境界となっているため，$\mathcal{X}$ において $\boldsymbol{x}$ s.t. $f(\boldsymbol{x}) = 0$ は二つのクラスを分ける境界を形成しています．このような境界を**分類境界 (classification boundary)** といいます．

2.2.1 ハードマージン

初めに訓練集合内の任意の元を正しく分類する $\boldsymbol{w}, b$ が存在する状況，f によって**分離可能 (seperable)** と仮定します．この時，任意の $\boldsymbol{x}_i$ に関して $f(\boldsymbol{x}_i)$ の符号と y_i の符号は一致する為，$y_i f(\boldsymbol{x}_i) > 0$ となります．一般に訓練集合を分離可能とする分類境界は複数存在し，図 1 のように分類境界を挟んで，二つのクラスがどれくらい離れているかの計量を**マージン (margin)** と呼びます．分類境界のマージンを大きくすることは，SVM の汎化性能の向上につながるため，マージンの最大化を目的とします．

その境界に最も近い $\boldsymbol{x}_i$ を $\boldsymbol{x}^*$ とした時，分類境界と $\boldsymbol{x}^*$ 間の距離は，点と平面の距離の公式から以下のように表現できます：

$$\frac{\left|\boldsymbol{w}^T \boldsymbol{x}^* + b\right|}{\|\boldsymbol{w}\|} \tag{3}$$

マージン最大化は，この距離の最大化によって，実現することができます．ここで，全ての元は f によって分離可能であったことから，$y_i f(\boldsymbol{x}_i) > 0$ すなわち $\exists M > 0$ s.t. $\forall i \ y_i f(\boldsymbol{x}_i) \geq M$ が成立します．ここで $\boldsymbol{x}^*$ の定義により $y_i f(\boldsymbol{x}_i) = |f(\boldsymbol{x}_i)| \geq |f(\boldsymbol{x}^*)|$ ですから，$f(\boldsymbol{x}^*) \geq M$ なる M の最大値を考えることは，$\boldsymbol{x}^*$ と分類境界の距離の最大化に一致します．マージン最大化は，次の最適化問題で表現できます：

$$\max_{\boldsymbol{w}, b, M} \frac{M}{\|\boldsymbol{w}\|} \text{ s.t. } \forall i \ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq M \tag{4}$$

さらに，$\boldsymbol{w}, b$ を M で割ったもので置換すれば，

$$\max_{\boldsymbol{w}, b} \frac{1}{\|\boldsymbol{w}\|} \text{ s.t. } \forall i \ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 \tag{5}$$

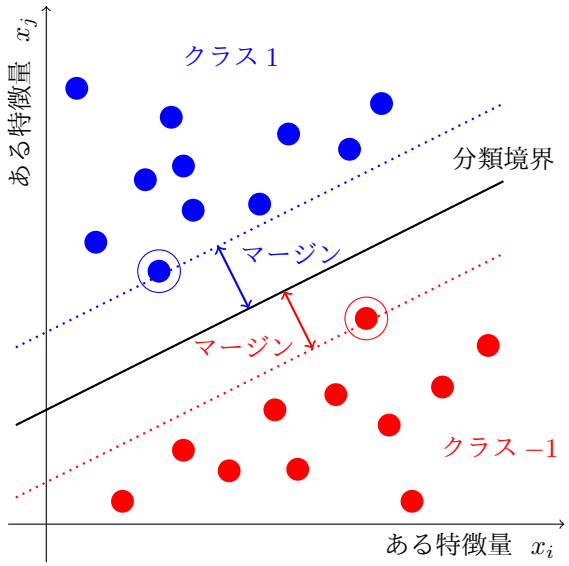

図 1　特徴量空間における分類の様子
赤丸が $y = -1$ となるクラスの特徴ベクトル，青丸が $y = 1$ となるクラスの特徴ベクトルを図示したものです．特徴ベクトルのうち，円で囲んだものが分類境界を決める特徴ベクトルであり，**サポートベクトル**と呼ばれます．

となります．そして，$1/\|\boldsymbol{w}\|$ の最大化が，扱いやすい $\|\boldsymbol{w}\|^2$ の最小化に対応することを考えれば，結局次の最適化問題に帰着できます．

$$\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2 \text{ s.t. } \forall i \; y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \tag{6}$$

このような分離可能性を仮定した SV 分類を**ハードマージン (hard margin)** と呼びます．

最適化問題 (6) の最適解を求めると，$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + \boldsymbol{b}) = 1$ なる $\boldsymbol{x_i}$ がいくつか現れます．これが，分類境界に最も近い $\boldsymbol{x}_i$ に対応し，これを分類境界を支えているとして，**サポートベクトル (support vector)** と呼びます．

2.2.2　ソフトマージン

ハードマージンでは分離可能性を仮定しましたが，現実問題では分離可能でない問題がほとんどです．この場合，**ソフトマージン (soft margin)** と呼ばれる拡張を考えます．ここで非負の実数 $\xi_i \geq 0$ を導入することで，制約条件 $\boldsymbol{w}^T\boldsymbol{x}_i + b \geq 1$ を次のように条件を書き換えます：

$$\boldsymbol{w}^T\boldsymbol{x}_i + b \geq 1 - \xi_i \tag{7}$$

ここで，誤分類を抑制するためには $\sum_{1\leq i\leq n} \xi_i$ をなるべく小さく保つことが必要ですから，最適化問題を改良して，以下のように定義します[*1]：

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in[n]} \xi_i \text{ s.t. } \forall i \; y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \tag{8}$$

ただし，$\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)$ とし，C を**正則化係数 (regularization parameter)** とします．C の役割を簡単に説明すれば，抑制の度合いの調整のためのパラメータです．C を大きくすればハードマージンに近づくこと

[*1] $\frac{1}{2}$ を最初の項にかけているのは，後の計算を簡便にするためです．元の問題と等価であることは簡単に確かめられると思います．

を述べておきます．どのように C を設定すべきかはデータに依存するため，交差検証法を用いて設定すべきでありますが，ここでは詳しく触れないこととします．

2.3 双対表現

ここまで定式化してきた最適化問題に対して，同等の主張を与える**双対問題 (dual problem)** を導くことで SVM の場合，比較的解きやすい分類器を得ることができます．詳しい詳細は付録の方に回すことにして，ここでは結果のみ与えておきます．得られた最適化問題 (8) に対して，双対問題を考えることによって以下の同等な問題に帰着することができます．ここで特徴的なのは変数が $\boldsymbol{\alpha}$ という一つの変数のみで書き表すことができる点です：

$$\max_{\boldsymbol{\alpha}} \ -\frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\alpha_i\alpha_j y_i y_j \boldsymbol{x_i}^T\boldsymbol{x_j} + \sum_{i\in[n]}\alpha_i \text{ s.t. } \sum_{i\in[n]}\alpha_i y_i = 0 \text{ and } \forall i, 0 \le \alpha_i \le C \tag{9}$$

$$f(\boldsymbol{x}) = \sum_{i\in[n]}\alpha_i y_i \boldsymbol{x_i}^T\boldsymbol{x_j} + b \tag{10}$$

$$b = y_i - \sum_{j\in[n]}\alpha_j y_j \boldsymbol{x_i}^T\boldsymbol{x_j} \quad i \in \{i \in \{1,\cdots,n\} \mid 0 < \alpha_i < C\} \tag{11}$$

2.4 kernel 函数

入力 $\boldsymbol{x}$ を特徴量空間 $\mathcal{F}$ に移す写像 $\phi : \mathcal{X} \to \mathcal{F}$ を考え，$\phi(\boldsymbol{x})$ を新たなベクトルと見なすことによって，決定函数は次のように書くことができます．

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b \tag{12}$$

この場合，$\boldsymbol{w} \in \mathcal{F}$ と変更を受けます．写像 ϕ が非線型であれば，$f(\boldsymbol{x}) = 0$ による分類境界は，元の $\mathcal{X}$ において非線型な分類境界となりえます．そのため，式 (12) に関して，これまで行った議論を適用することで，次のような双対問題の最適化に帰着できます．

$$\max_{\boldsymbol{\alpha}} \ -\frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\alpha_i\alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) + \sum_{i\in[n]}\alpha_i \text{ s.t. } \sum_{i\in[n]}\alpha_i y_i = 0 \text{ and } \forall i, 0 \le \alpha_i \le C \tag{13}$$

ここで上式は，$\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ という内積の形でしか ϕ が現れていないことに注目します．これが双対問題をわざわざ考えた意義です．すなわち，非線型な分類問題を考えた際に ϕ を直接計算する必要なく，内積さえ計算することができれば，最適化問題を解く条件は揃うことになります．

Def kernel 函数

$$\mathcal{K}(\boldsymbol{x_j}, \boldsymbol{x_i}) = \phi(\boldsymbol{x}_j)^T\phi(\boldsymbol{x}_i) \tag{14}$$

この kernel 函数を利用することによって，最適化問題及び決定函数は以下のように書くことができます．

$$\max_{\boldsymbol{\alpha}} \ -\frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\alpha_i\alpha_j y_i y_j \mathcal{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) + \sum_{i\in[n]}\alpha_i \text{ s.t. } \sum_{i\in[n]}\alpha_i y_i = 0 \text{ and } \forall i, 0 \le \alpha_i \le C \tag{15}$$

$$f(\boldsymbol{x}) = \sum_{i\in[n]} \alpha_i y_i \mathcal{K}(\boldsymbol{x_i}, \boldsymbol{x}) + b \tag{16}$$

$$b = y_i - \sum_{j\in[n]} \alpha_j y_j \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i) \quad i \in \{i \in \{1, \cdots, n\} \mid 0 < \alpha_i < C\} \tag{17}$$

このようにして，特定の問題が特徴ベクトルの内積のみで表現されている場合に，内積を kernel 函数で置き換えて線型分類問題に帰着する方法を**カーネルトリック (kernel trick)** と呼び，主成分分析などでも同様の手法が利用されています[*2]．

3 量子計算

QSVM を理解する上での SVM の基礎知識を前節において紹介しました．この節においては，量子計算ないし量子情報の入門知識に関して解説します．前節では統計分野でよく利用されるベクトルの内積の記法の表現として $\boldsymbol{x}^T\boldsymbol{x}$ などを利用しましたが，本節では量子力学の慣習に倣って Braket 記法を採用しています．量子力学の原理をもとに解説をしているため，量子力学の基礎概念を理解した上で読むことを想定しています．量子力学を学習したことのない読者でも，同じ Physics lab. 2022 内で量子計算を行う上で必要な諸概念に関して解説した記事がありますので，この節の理解の助けとなると思われるので参考にしてほしいです．

3.1 量子力学の原理と量子コンピュータ

量子力学は以下の 5 つの基本法則によって支配されています．[*3]本記事ではこの基本法則の厳密な正当性については議論しません．

Axiom 量子力学の基本法則

I. 系の状態は完備な状態空間 $\mathcal{H}$ のノンゼロベクトルで表現されます．ここで $\mathcal{H}$ はエルミート内積つき複素線型空間 (Hilbert 空間) だとします[a]．

II. 系の観測量は $\mathcal{H}$ に作用するエルミート演算子で表現可能とします．

III. 観測量の固有ベクトルを $|i\rangle, A|i\rangle = a_i|i\rangle$ とすると，状態 $|\psi\rangle$ において物理量 A を観測した際の結果は a_i のうちどれかであり，a_i が得られる確率は $|c_i|^2$ に比例します．ただし，$c_i \in \mathbb{C}$ は，$|\psi\rangle = \sum_i c_i|i\rangle$ と展開した時の係数とし，$\sum_i |c_i|^2 = 1$ を満たします．

IV. 全系が部分系 A と部分系 B からなり，部分系がそれぞれ状態空間 $\mathcal{H}_\mathrm{A}, \mathcal{H}_\mathrm{B}$ を有する場合，全系の状態はそのテンソル積 $\mathcal{H}_\mathrm{A} \otimes \mathcal{H}_\mathrm{B}$ で表されます．

V. 系に対する観測以外の操作（時間発展，回転など）はユニタリ演算子で表現できます．

[a] 混合状態については扱わないので，本記事で扱う状態は全て純粋状態であるとします．

法則 I に関しては，基本的に本記事内において，回路型量子コンピュータ[*4]における量子系として $\mathcal{H} \simeq \mathbb{C}^2$ を基本的な状態空間として扱っていきます．この量子系を**量子ビット (Quantum bit)** 以下 **qubit** と呼ぶことにする．回路型量子コンピュータは，状態の時間変化を正確に制御できる，n qubits の物理的な実装のこ

[*2] ここでは，特徴量の内積をもとに kernel 函数を構成しており，特徴量空間の内積であれば kernel 函数として利用できます．加えて，特徴量写像が不明であっても，Mercer の定理を満たす性質のいい函数ならば kernel 函数として利用できます．

[*4] ここで回路型と接頭語をつけているのは，観測型，アニーリング型などの本記事で解説する量子コンピュータとは異なる計算手法があるからです．基本的に解説しているのは回路型ですので，あまり気に留める必要はありません．

とをいいます．それに付随する概念である量子アルゴリズムは，系から情報を取得するための測定を伴う，量子系に対する操作全体を指します．

法則 II,III と関連して，$\mathcal{H}$ の基底を次のように取ることにし，**計算基底 (computational basis)** と呼ぶことにします．

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{18}$$

このように取った時，系の観測量について次のような Pauli の Z 演算子 σ_z と表現するものを考えます．ここで後述するゲート操作のために Pauli 演算子の行列表現を示しておきます：

$$\sigma_x = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}, \quad \sigma_y = \begin{bmatrix}0 & -i\\i & 0\end{bmatrix}, \quad \sigma_z = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \tag{19}$$

行列表現から明らかなように $|0\rangle, |1\rangle$ は σ_z の固有ベクトルとなっており，その固有値はそれぞれ $1, -1$ となっている．ここで，少し紛らわしいのですが古典ビット列との対応のために，量子系の物理量 σ_z の観測値として 1 が得られた場合，$|0\rangle$ が得られたとして 0 の状態が得られた，（或いは 0 の測定値が得られた）と表現し，それに対応して σ_z の観測値として -1 が得られた場合には，1 の測定値が得られたと表現します．[*5]また，本記事で扱う観測量は σ_z のみとします[*6]．このように定めることで，古典ビットから qubit への単純な量子化が定められます：

$$0 \longrightarrow |0\rangle \quad 1 \longrightarrow |1\rangle$$

3.2 1 qubit の幾何的表現

後述する量子ゲート操作の理解を助けるために 1 qubit 系については幾何学的な状態表現である **Bloch 球 (Bloch sphere)** モデルについて解説していきます．これまでの議論と，法則 III より，1 qubit の任意の状態 $|\psi\rangle$ は $\alpha, \beta \in \mathbb{C}$ を使って次のように書くことができます：

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \text{with} \quad |\alpha|^2 + |\beta|^2 = 1 \tag{20}$$

ここで α, β は複素数であり，$\langle\psi|\psi\rangle = 1$ という条件より 3 つのパラメータ γ, θ, φ を利用して，少々恣意的ですが次のように表現できます：

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \tag{21}$$

但し，γ, θ, φ はいずれも実数であり，$0 \le \theta \le \pi, 0 \le \varphi < 2\pi$ を満たすとします．ここで $e^{i\gamma}$ の因子に関して，物理量 A を観測した時に得られる際の結果を法則 III から考察してみます．もし $e^{i\gamma} = 1$ の場合，a_i であり，$e^{i\gamma} \neq 1$ の場合，$e^{i\gamma}a_i$ となりますが，全ての a_i に関して位相因子がかかっているため，得られる観測量に対して影響は与えません．そのため，この位相因子は大域的であるとして $e^{i\gamma} = 1$ として無視することができる[*7]．このようにして式 (21) で表した状態を二次元球面上のベクトル $(\cos\varphi\sin\theta, \sin\varphi\sin\theta, \cos\theta) \in S^2$ として表現すると図 2 のようになります．これが Bloch 球モデルです．

[*5] どうしても $|i\rangle$ と a_i について $i = a_i$ にしたければ，$\frac{1-\sigma_z}{2}$ という物理量を考えても良い．

[*6] ここで別の観測量として σ_x, σ_y などもエルミート演算子ですから，この射影測定を行うことや，POVM 測定に関して許されてはいるものの，量子回路において測定できる物理量は基本 σ_z のみであり，$\sigma_x(\sigma_y)$ を観測したい場合は後述するゲート操作を系に与えてから，σ_z を観測することによって $\sigma_x(\sigma_y)$ の観測結果に対応する結果を間接的に得るという操作を考えます．POVM 測定に関しては環境系の役割を補助 qubit に与えて，観測を行うことを考えれば良いです．

[*7] 余談でありますが，このような因子を **global factor** として，式 (21) 中の $e^{i\varphi}$ として表現されている **relative factor** とは区別します．このように大域的な位相因子を無視する操作，すなわち写像 $S^3 \to S^2$ を構成することを Hopf ファイブレーションといいます．

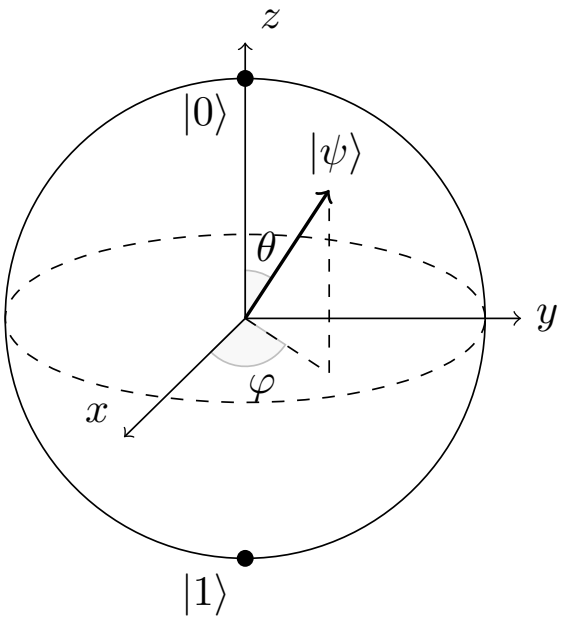

図 2　Bloch 球

視覚的に位相因子がどのようにかかっているかがわかりやすく，ベクトルの各軸への射影が各 Pauli 演算子に対応する物理量の期待値となっているため，量子系の物理状態に関して直感的な理解のしやすいモデルとなっています．

3.3　合成系

法則 IV に関して，対象の量子系 $\mathcal{H}$ が二つの量子系 $\mathcal{H}_{\mathrm{A}}, \mathcal{H}_{\mathrm{B}}$ で構成されていると仮定します．このテンソル積をとった合成 Hilbert 空間における状態は，各正規直交基底をそれぞれ，$\{|e_j\rangle_{\mathrm{A}}\}, \{|e_{j'}\rangle_{\mathrm{B}}\}(1 \leq j \leq \dim \mathcal{H}_{\mathrm{A}}, 1 \leq j' \leq \dim \mathcal{H}_{\mathrm{B}})$ とすれば，次のように表現できます：

$$\sum_{j=1}^{\dim \mathcal{H}_{\mathrm{A}}} \sum_{j'=1}^{\dim \mathcal{H}_{\mathrm{B}}} c_{jj'} \, |e_j\rangle_{\mathrm{A}} \otimes \left|e'_j\right\rangle_{\mathrm{B}} \tag{22}$$

ここで $c_{jj'}$ は $|e_j\rangle_{\mathrm{A}} \otimes \left|e'_j\right\rangle_{\mathrm{B}}$ にかかる係数であるとします．正規直交基底のテンソル積は，合成空間での正規直交基底となっており，$\dim \mathcal{H} = \dim \mathcal{H}_{\mathrm{A}} \times \dim \mathcal{H}_{\mathrm{B}}$ が成り立ちます．以下，n qubit を考える際には次のような，一番右の ket を 0 番目の qubit と定義し，左に行くにつれラベル番号が 1 ずつ増加していき，一番左の qubit を $n-1$ 番目の qubit と定義するラベル付を採用します[*8]．また，テンソル積の記号 $\otimes$ は断りがなければ基本以下のように省略するものとします．

$$|0\rangle \otimes |0\rangle = |00\rangle \, , |1\rangle \otimes |1\rangle \otimes |0\rangle = |110\rangle \tag{23}$$

また，行列表記で合成系を表現する時には Kronecker の表記を採用します．具体的な例を示せば，$|\psi\rangle =$

[*8] このラベル付の差異によって，行列表記が異なるので注意してください．逆のラベル付の方が一般的ですが，2 進数表記の bit 列を表すのに整合的なのはこちらのラベル付なので筆者はこちらの記法の方が好みです．

$\alpha|0\rangle + \beta|1\rangle, |\phi\rangle = \gamma|0\rangle + \delta|1\rangle$ を考えた場合には，

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \otimes |\phi\rangle = \begin{bmatrix}\alpha|\phi\rangle\\ \beta|\phi\rangle\end{bmatrix} = \begin{bmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{bmatrix} \tag{24}$$

また，合成系の内積などを取る操作は各量子系の内積を取った後，それの積を取れば良いです：

$$(\langle\psi'| \otimes \langle\phi'|)|\psi\rangle \otimes |\phi\rangle = \langle\psi'|\psi\rangle \cdot \langle\phi'|\phi\rangle \tag{25}$$

3.4 ゲート操作

3.4.1 1-qubit ゲート

法則 V に関連して，系に対する観測以外の操作は，ユニタリ演算子で表現できるため，逆に回路型量子コンピュータ内において適切な回路を設計することによって任意のユニタリ演算子で表現される操作を行うことができます．ここで $\mathbb{C}^2$ におけるユニタリ演算子とは $UU^\dagger = U^\dagger U = 1$ であり，端的に言えば複素ベクトルの内積を不変に保つ性質があったため，二次元球面 S^2 上のベクトルである Bloch 球モデルでの任意の軸周りの回転操作に対応しています．このように qubit の状態変化，回転操作を引き起こす物理操作のことを古典ビットの状態を変更する素子である論理ゲートに対応して，**量子論理ゲート (Quantum logic gate)** 単にゲート操作と呼び，殊に 1-qubit のみに行うゲート操作を 1-qubit ゲートと呼びます．

量子回路においてよく利用される 1-qubit ゲートを紹介していきます．まず，Pauli 演算子はエルミートでありユニタリーでもありますから，ゲート操作の表現としても許されます．以後，物理量を表す Pauli 演算子について言及する際には σ_z などと書き，ゲート操作を指す場合には単に Z ゲートなどといい，数式では Z と書くことにします[*9]．回路記号に関しては，図 3 のように表現します．

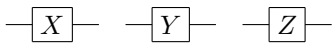

図 3　Pauli ゲート

また，Pauli 演算子以外で量子情報において頻繁に利用される演算子として次の **Hadamard 演算子** があります．

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{26}$$

これは，Bloch 球上の x 軸と z 軸のちょうど中間に位置する軸周りの 180 度回転となっており，計算基底の各基底から qubit の重ね合わせ状態を作ることのできるゲート操作になっています．また，特定軸周り 180 度の回転ゲートについて言及しましたが，各 Bloch 球の x, y, z 軸周りの角度 θ の回転ゲート操作を

[*9] 数式上では等価なものであるため，混同を防ぐためにこの記法を採用しています．

$R_x(\theta), R_y(\theta), R_z(\theta)$ と表現し，行列表現は次のようになります．

$$R_x(\theta) = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{27}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{28}$$

$$R_z(\theta) = \begin{bmatrix} e^{-\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \tag{29}$$

また，回路図においてはそれぞれ図 4 のように表現します．

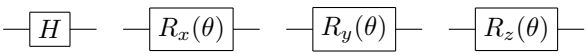

図 4　Hadamard ゲートおよび各軸の θ 回転ゲート

3.4.2　2-qubit ゲート

前節において，単一 qubit に対するゲート操作を解説しましたが，合成系に対して 1-qubit 量子ゲートを作用させる場合には，演算子のテンソル積を考える必要があります．例えば，3 qubits を考え，初期状態を $|000\rangle$ とします．0 番目の qubit にのみ，X ゲートを作用させることを考えると，これは次のように表現されます：

$$(I \otimes I \otimes X_0)\,|000\rangle = I\,|0\rangle \otimes I\,|0\rangle \otimes X_0\,|0\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle = |001\rangle \tag{30}$$

ここで I は恒等演算子，X_0 の添字 0 は 0 番目の qubit に作用することを強調する意味でつけました．例から推測して，1-qubit ゲートのみを多 qubits に作用する場合は，全体としてのゲート操作を考えると，1-qubit ゲートのテンソル積で表現し尽くせることがわかると思います．

これまでの議論は，本質的には 1-qubit ゲートを考えるのと同等のことであり，合成系をわざわざ考慮する必要はありません．そこで，次のような 1-qubit ゲート U を用いて次のように表現される制御ゲート G を考えることにします．

$$G\,|00\rangle = |00\rangle, G\,|10\rangle = |10\rangle, G\,|01\rangle = U_1\,|01\rangle, G\,|11\rangle = U_1\,|11\rangle \tag{31}$$

すなわち，0 番目の qubit の状態に対応して，1 番目の qubit に U の作用するか否かを決定するゲート操作です．これは明らかに 0 番目の qubit の状態に依存するため，1-qubit ゲートとして表現できないことがわかります．典型的な 2-qubit ゲートの例として $U = Z$ とした，CNOT ゲートあります．これはすなわち，次のような変換を与えるゲート操作となっています．

$$|00\rangle \to |00\rangle, |10\rangle \to |10\rangle, |01\rangle \to |11\rangle, |11\rangle \to |01\rangle \tag{32}$$

ちなみに回路図で CNOT ゲートは図 5 のように表現されます．

図 5　CNOT ゲート

また，以下では数式において CNOT ゲートを表現する場合には $\mathrm{CNOT}(c,t)$ などと表現し t は X ゲートを作用させる対象となるビットのラベルを表し，c は制御ビットのラベルとします．例えば，$\mathrm{CNOT}(0,1)$ は以下のようになります．

$$\mathrm{CNOT}(0,1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{33}$$

3.5 測定

前述した通り，測定する物理量は σ_z でしたから，$\mathcal{H}^n$ 上の状態ベクトルで表現されている，展開係数のノルムの 2 乗に比例した確率で，測定値が得られます．得られた値は古典ビット列としてレジスタに保存されます．例として，$|\psi\rangle = \sqrt{0.34}\,|00\rangle + i\sqrt{0.16}\,|10\rangle - \sqrt{0.20}\,|01\rangle + \sqrt{0.30}\,|11\rangle$ という状態があれば，$o_z^{\otimes 2}$ の測定結果として，16% の確率で古典ビット列 10 という測定値が得られ，その情報がレジスタに保存されるということです．このような測定操作を回路図では図 6 のように表現することにします．

図 6　測定

これを利用することによって未知の純粋状態 $|\psi\rangle$ の計算基底における確率振幅のノルムに関しては多数の測定から，推定することができます．例えば，$|\psi\rangle$ のうち $|0\rangle$ の確率振幅の推定を考えた時，これは σ_z の測定を行い，測定値として 0 が得られた確率が確率振幅のノルムの 2 乗が近似値となります[*10]．

4　量子サポートベクトルマシン (QSVM) の理論

今回扱う量子機械学習は，量子コンピュータの知識と SVM の知識を融合させた計算手法である，**量子サポートベクトルマシン (QSVM)** と呼ばれるものです．本記事の QSVM では，量子系を利用することによって量子特徴写像, 量子 kernel を構成することが目的です．量子系による特徴写像は，複雑な分非常に強力な表現能力を有しているが，ノイズの影響を受けやすいという問題があります．誤り訂正を有さない NISQ[*11]で扱う際には，特にそのバランスを考慮して構成を考える必要があります．

4.1 量子特徴写像

NISQ を利用して，入力データ $\boldsymbol{x}_i \in \mathcal{X} = [0, 2\pi)^N$[*12] を非線型的に量子状態へと移す写像 $\Phi : \boldsymbol{x} \in \mathcal{X} \mapsto |\Phi(\boldsymbol{x})\rangle \in \mathcal{F}_\Phi$[*13]を構成することを考えます．そのような写像を構成するに当たって，Havlíček ら [5] は次の

[*10] 相対位相まで，推定することを考える時には量子トモグラフィという手順を踏む必要があります．

[*11] NISQ とは，Noisy Intermediate-Scale Quantum device の略で，数十から数百量子ビット程度を扱う，誤り訂正能力を持たない小規模の量子コンピュータのことです．誤り訂正を行うことができない為，量子ビットにノイズの影響が大きく生じてしまい，ノイズの影響を無視できるほど簡単な計算しかできないものとなっています．その為，計算途中のノイズによる誤り訂正を完全に達成することができ，計算結果が完全に正しく出力される量子コンピュータこそ，真の計算機であるといえます．

[*12] ここでは，$[0, 2\pi)^N$ を $[0, 2\pi)$ の N 個の直積であるとします．特徴データの元の空間が $\mathbb{R}^N$ ならば，この空間に収まるように正規化をすれば良いです．

[*13] あえて，2,3 節でそれぞれ用いたベクトルの記法を同時に使います．機械学習と量子計算の領域が混ざっていることが意図されています．

ゲート操作を提案しました：

$$\mathcal{U}_{\Phi(\boldsymbol{x})} = \prod_d U_{\Phi(\boldsymbol{x})} H^{\otimes N}, U_{\Phi(\boldsymbol{x})} = \exp\left(-i \sum_{S\subset[n]} \phi_S(\boldsymbol{x}) \bigotimes_{i\in S} Z_i \right) \tag{34}$$

ここで，$\phi_S(\boldsymbol{x})$ は $\boldsymbol{x}$ を実数空間へと移す写像とし，d は回路の深さを表す正整数です．ここでは，簡単のために $|S| \leq 2$ となるような集合のみを考えるものとします[*14]．$\phi_S(\boldsymbol{x})$ に関して，$|S| \leq 2$ としたので，$k, l \in [n], k \neq l$ として以下のように定めます：

$$\phi_S(\boldsymbol{x}) = x_k \quad S = \{k\} \tag{35}$$

$$\phi_S(\boldsymbol{x}) = (\pi - x_k)(\pi - x_l) \quad S = \{k, l\} \tag{36}$$

このように定めた時に，以下のやや天下り的な計算：

$$\begin{aligned}
\exp(-i\phi Z) &= \sum_{m=0}^{\infty} \frac{(-i\phi)^m}{m!} Z^m \\
&= \sum_{m=0}^{\infty} \frac{(-1)^m \phi^{2m}}{2m!} I - i \sum_{m=0}^{\infty} \frac{(-1)^m \phi^{2m+1}}{(2m+1)!} Z \\
&= \begin{bmatrix} \cos\phi - i\sin\phi & 0 \\ 0 & \cos\phi + i\sin\phi \end{bmatrix} = R_z(2\phi)
\end{aligned} \tag{37}$$

$$\begin{aligned}
\exp(-i\phi Z_1 \otimes Z_0) &= \sum_{m=0}^{\infty} \frac{(-1)^m \phi^{2m}}{2m!} (I_1 \otimes I_0) - i \sum_{m=0}^{\infty} \frac{(-1)^m \phi^{2m+1}}{(2m+1)!} (Z_1 \otimes Z_0) \\
&= \begin{bmatrix} e^{-i\phi} & 0 & 0 & 0 \\ 0 & e^{i\phi} & 0 & 0 \\ 0 & 0 & e^{i\phi} & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} e^{-i\phi} & 0 & 0 & 0 \\ 0 & e^{-i\phi} & 0 & 0 \\ 0 & 0 & e^{i\phi} & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \mathrm{CNOT}(0,1)[R_z(2\phi)_1 \otimes I_0]\mathrm{CNOT}(0,1)
\end{aligned} \tag{38}$$

を考えることにより，$U_{\Phi(\boldsymbol{x})}$ の構成要素及び $\mathcal{U}_{\Phi(\boldsymbol{x})}$ は回路図において以下のように表現できます．

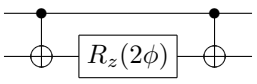

図 7　$\exp(-i\phi Z_1 Z_0)$ の表現

ここで $U_{\Phi(\boldsymbol{x})} H^{\otimes N}$ の繰り返しの数は d です．ここで $|S| = 1$ とした場合，$U_{\Phi(\boldsymbol{x})}$ は 1-qubit ゲートのみで構成することができ，量子縺れの要素を含むことはありません．そのため，古典的なシミュレート，すなわち古

[*14] 量子縺れの性能を鑑みても，CNOT ゲートを利用するだけで済むため現段階では $|S| \leq 2$ で行うのが良いと思います．

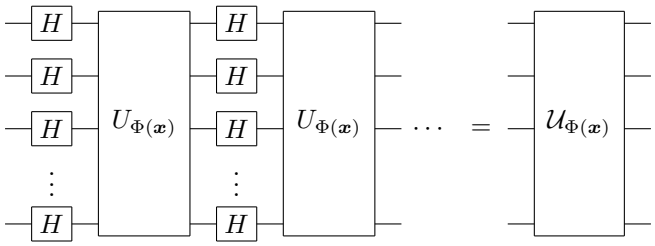

図 8　$\mathcal{U}_{\Phi(\boldsymbol{x})}$ の表現

典コンピュータ[*15]で容易にシュミレートすることができ，量子的な優位性[*16]は得られることはありません．簡単な例としては，計算量の優位性などを思い浮かべれば良いと思います．1-qubit ゲートのみで構成される行列計算は，各 qubit に対してゲートを作用させていくことを考えます．この操作は，テンソル積を考えればゲートの深さを d，量子ビットの数を N とすれば，$\mathbb{C}^2$ 上の行列を 1 回掛けるだけでは大した計算量ではないため，計算量は $O(dN+N^2)$ で済みます．しかしながら，2-qubit ゲートが含まれる場合，行列要素は容易に分解することはできないため，上手い計算手法を考えなければ $N\times N$ の行列の積を d 回行う必要があり，$O(N^{2d})$ の計算を行う必要があります．

以上より，式 (38) で表現されるゲートを用いれば，古典的なシミュレートの難しい特徴量写像を得ることができます．

4.2　量子 kernel

式 (14) で考えたように，SVM を利用した最適化問題を解くに当たって，特徴写像の値は必要なくその内積さえ得ることができれば良かったのを思い出しましょう．そのため，kernel 函数として

$$\begin{aligned}\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) &= |\langle\Phi(\boldsymbol{x}_j)|\Phi(\boldsymbol{x}_i)\rangle|^2 \\ &= \left|\langle 0|^{\otimes N}\,\mathcal{U}^\dagger_{\Phi(\boldsymbol{x}_i)}\mathcal{U}_{\Phi(\boldsymbol{x}_j)}\,|0\rangle^{\otimes N}\right|^2 \end{aligned} \tag{39}$$

を採用することができます．ここで，量子コンピュータを利用すれば，$|0\rangle^{\otimes N}$ に対して，パラメータ付き量子回路 $U_{\Phi(\boldsymbol{x}_j)}, U^\dagger_{\Phi(\boldsymbol{x}_i)}$ を順に作用させ，測定により $|0\rangle^{\otimes N}$ の状態が得られた回数を計数することができます．ここで法則 III を思い出せば，$|0\rangle^{\otimes N}$ の状態の得られる確率が $\mathcal{K}(\boldsymbol{x}_j,\boldsymbol{x}_i)$ に一致するため，量子 kernel 函数の値を推定することができます．このようにして構成した量子 kernel を古典的に推定するのは難しいため，古典的な手法に対して優位性を持つ可能性があります[*17]．注意点としては，あくまでも量子 kernel の古典的な推定の困難さは量子的な優越性のための必要条件ではあるものの十分条件ではないことに注意してください．しかしながら，Liu ら [6] によって，全ての古典的な学習器に対して量子優位性をもつような問題の存在が示されています．

[*15] 念の為かいておくと，古典コンピュータは，古典的なビットを利用したブール計算で動く計算機のことを言います．量子コンピュータとの比較のため，接頭語を附属させているだけなので，私たちが普段使っているコンピュータのことだと考えれば良いです．

[*16] 古典コンピュータによる演算の精度を上回る演算が量子コンピュータによって達成できることを量子的な優越性という．

[*17] 特徴データの組みに対して，$\mathcal{U}_{\Phi(\boldsymbol{x})}$ の作用にかかる計算量を M として，推定値を得るために多数の演算の回数を L とします．かなり雑な仮定ではありますが，計算量の推定として，計算量は $O(dML)$ として，$O(N^{2d})$ よりも計算量を小さくできると考えることができます．

4.3 QSVM アルゴリズム

以上の議論により，QSVM について 2 クラス分類問題の評価は次のようなプロセスで構成されています．

QSVM アルゴリズム

1. 訓練データの組み $(\boldsymbol{x}_j, \boldsymbol{x}_i)$ に対して，量子コンピュータを利用して $\mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_i)$ を推定する．
2. 訓練データ $\boldsymbol{x}$ およびテストデータ $\tilde{\boldsymbol{x}}$ に対しても，$\mathcal{K}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ を推定する．
3. プロセス (1) から得られたカーネル函数の推定値を使って，古典コンピュータによる SVM アルゴリズムを活用することで，最適化問題 (15) を解く．
4. 得られた決定函数にテストデータから得られた kernel 函数を代入して，決定函数の評価を行う．

実際に次節以降で Qiskit を利用して，QSVM を古典コンピュータでシミュレートして分類問題を解いてみよう．

5 QSVM による 2 クラス分類

本記事で使用する package を記しておきます．この問題は，IBM Quantum Challenge 2021 Fall の 3-1 に基づいています[*18]．

```
# General imports
import os
import gzip
import numpy as np
from math import pi
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# scikit-learn imports
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Qiskit imports
from qiskit import Aer, execute
from qiskit.circuit import QuantumCircuit
from qiskit.circuit.library import PauliFeatureMap
from qiskit_machine_learning.kernels import QuantumKernel
```

MNIST とは，機械学習分野入門においてよく利用される，手書き数字画像のデータセットです．ここでは，

[*18] Github の https://github.com/qiskit-community/ibm-quantum-challenge-fall-2021 において，MNIST 画像のデータなどやガイドとなるコードがありますので，qiskit を実際に動かしてみたいという読者は参考にすると良いです．

そのデータセットのうち 4, 9 に関して着目しその分類を考えることにします．

Listing 1　MNIST 画像の取得

```
# Load MNIST dataset
DATA_PATH = './resources/ch3_part1.npz'
data = np.load(DATA_PATH)

sample_train = data['sample_train']
labels_train = data['labels_train']
sample_test = data['sample_test']

# Split train data
sample_train, sample_val, labels_train, labels_val = train_test_split(
    sample_train, labels_train, test_size=0.2, random_state=29)

# Visualize samples
fig = plt.figure()

LABELS = [4, 9]
num_labels = len(LABELS)
for i in range(num_labels):
    ax = fig.add_subplot(1, num_labels, i+1)
    img = sample_train[labels_train==LABELS[i]][0].reshape((28, 28))
    ax.imshow(img, cmap="Greys")
```

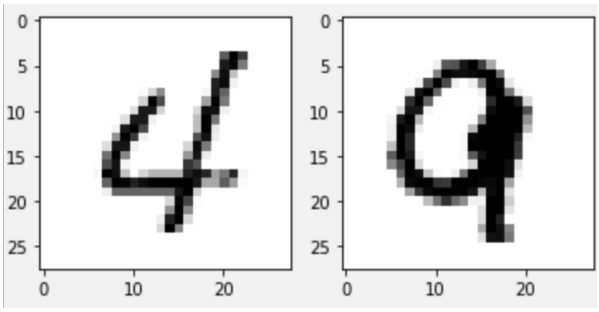

図 9　MNIST 画像の一例

上の実行結果からわかるように，各データは，28×28 次元の手書き数字データ画像を配列データにしており，画像のある部分における濃さを 0(白) から 255(黒) として並べたものとなっています．また，このデータセットは，80 個の訓練データと 20 個のテストデータから構成されています．

ここで入力データの定義域は $[0, 2\pi)$ に収める必要があるため，正規化を行った後，この定義域内に収まるように処理をします．ここでは，のちに説明する ZZFeatureMap の引数の性質上 $[0, 6.28]$ とします[*19]．それ

[*19] 余談ではありますが，この Challenge 内で指定されていた定義域は $[-1, 1]$ でありますが，これでは $\phi_S(\boldsymbol{x})$ の効果が全く別のものになってしまうため，訂正しました．しかしながら，$[-1, 1]$ でも分類できてしまいました．あくまでも量子 kernel の中身は BlackBox でしかないということを忘れてはならないことでしょうか．(もし，定義域が $[-1, 1]$ でも問題ない，元の論文で想定していたであろう効果が望めると考える読者がいれば意見を送ってほしいです．)

に加え，このデータの次元は 28×28 であり，NISQ である程度正確に扱える量子ビットの次元を大幅に超えているため，次元圧縮を行い 5 次元データとします．

Listing 2　正規化，次元圧縮

```
# Standardize
ss = StandardScaler()
sample_train = ss.fit_transform(sample_train)
sample_val = ss.transform(sample_val)
sample_test = ss.transform(sample_test)

# Reduce dimensions
N_DIM = 5
pca = PCA(n_components=N_DIM)
sample_train = pca.fit_transform(sample_train)
sample_val = pca.transform(sample_val)
sample_test = pca.transform(sample_test)

# Normalize
mms = MinMaxScaler((0,6.28))
sample_train = mms.fit_transform(sample_train)
sample_val = mms.transform(sample_val)
sample_test = mms.transform(sample_test)
```

以上の操作により，データ処理は行うことができたため量子特徴写像を構成するための量子回路を構成します．qiskit.circuit.library 内の PauliFeatureMap および QuantumKernel を使用することによって，図 8 で示された回路の一つを構成し，全てのデータの組みに関する kernel 函数の出力を集めた kernel 行列の生成することができます．

Listing 3　量子写像，量子 kernel 生成

```
# 5 features, depth 1, full entanglement
pauli_map = PauliFeatureMap(feature_dimension=N_DIM, reps=1, entanglement ='full' ,
    paulis = ['Z', 'ZZ'])
pauli_kernel = QuantumKernel(feature_map=pauli_map, quantum_instance=Aer.get_backend('
    statevector_simulator'))
```

ここで PauliFeatureMap の各引数に関して，feature-dimension は量子ビットの次元，reps は回路の深さ d，entanglment は量子縺れ生成回路の構造の指定，そして paulis は用いる回転ゲートの種類（順序込み）[*20]を表している．また，QuantumKernel の各引数については quantum-instance は利用するシュミレータの指定を表しています．

実際に 0 と 1 番目の訓練データ間の Kernel 函数推定のための回路は図 11 のようになっています[*21]．

このように回路を各訓練データないしテストデータに対して構成し，kernel 行列を計算，表示するコードは次のようなものです．また，出力結果を図 10 に示しました．

[*20] 元の論文では $R_z(\theta)$ ゲートに限った回路を想定していますが，$R_x(\theta)$ ゲートや $R_y(\theta)$ ゲートを利用した回路も設計することができます．

[*21] P ゲートは $R_z(\theta)$ と等価なゲートと考えれば良く，興味があれば調べてみると良いと思います．Global 位相と関連があります．

Listing 4　kernel 行列の計算

```
matrix_train = pauli_kernel.evaluate(x_vec=sample_train)
matrix_val = pauli_kernel.evaluate(x_vec=sample_val, y_vec=sample_train)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(np.asmatrix(matrix_train),
                interpolation='nearest', origin='upper', cmap='Blues')
axs[0].set_title("training kernel matrix")
axs[1].imshow(np.asmatrix(matrix_val),
                interpolation='nearest', origin='upper', cmap='Reds')
axs[1].set_title("validation kernel matrix")
plt.show()
```

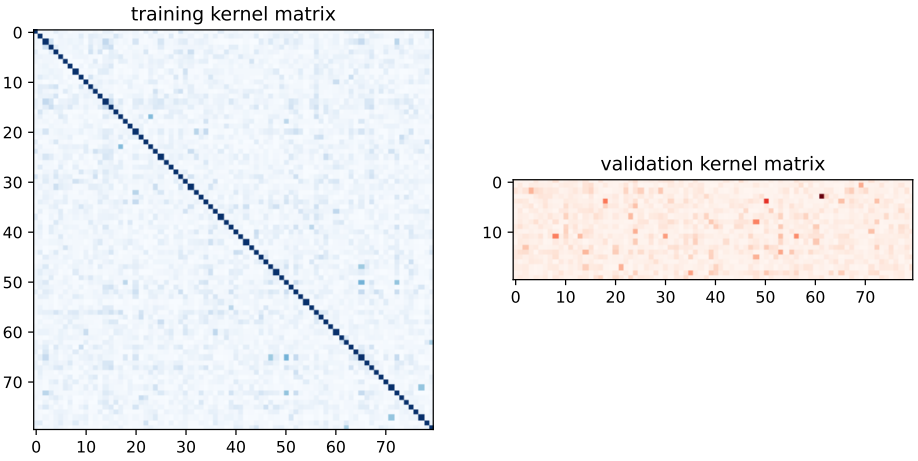

図 10　量子 kernel 行列

最後に得られた kernel 行列から SVM アルゴリズムを利用して最適化問題を解き，決定函数を評価します．

Listing 5　SVC アルゴリズム

```
pauli_svc = SVC(kernel='precomputed')
pauli_svc.fit(matrix_train, labels_train)
pauli_score = pauli_svc.score(matrix_val, labels_val)

print(f'Precomputed kernel classification test score: {pauli_score*100}%')
```

出力結果は”Precomputed kernel classification test score: 70.0%”でした．以上より，量子回路を利用して

2 クラス分類問題の決定函数を得ることができました[*22][*23].

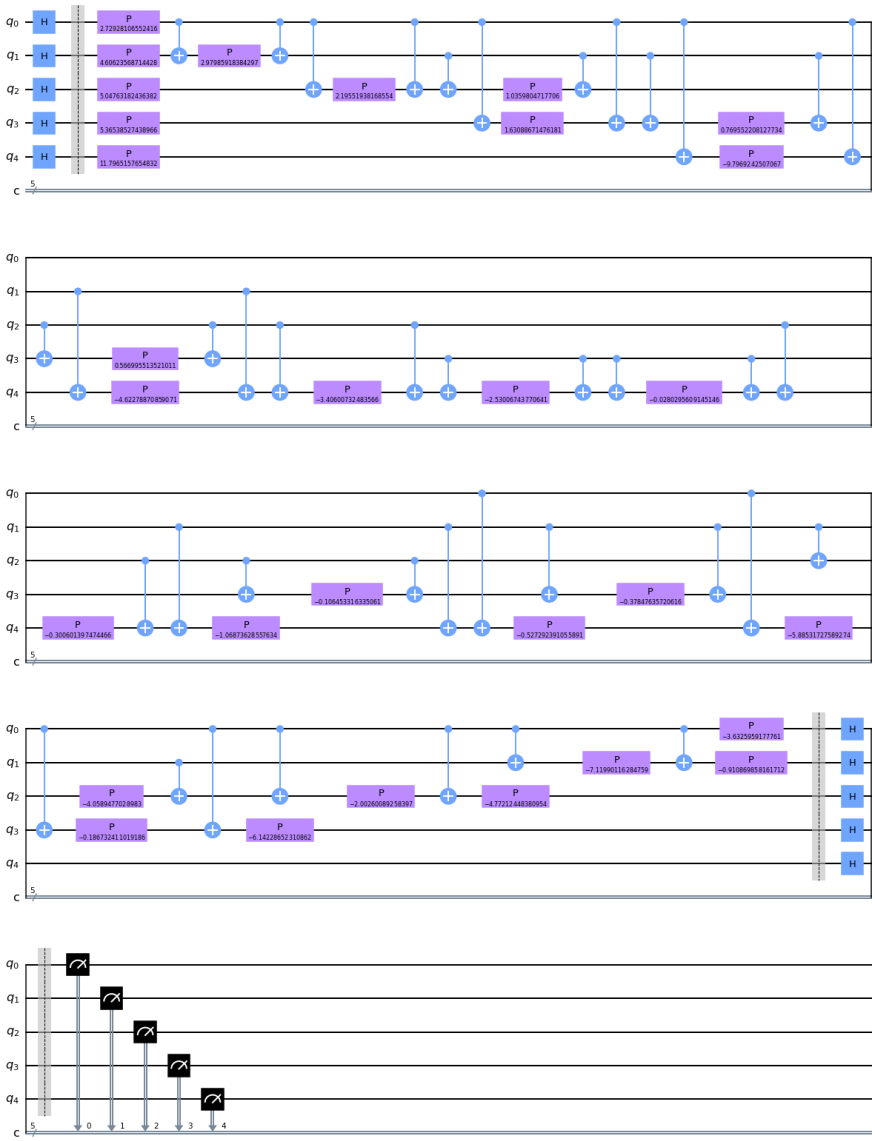

図 11　量子 kernel 推定のための回路

[*22] 余談でありますが，内積の値を推定ではなく計算によって完全に求め出すことができているシミュレータ上でもこの程度の精度です．実際 NISQ で動かし，kernel 函数の値を推定することを考えるとノイズの効果も考慮する必要があるため，精度はさらに低下すると考えられます．あくまでも量子 kernel の計算に関して量子的な優越性は示されているものの，その高次元の特徴量空間が高精度な分類を達成するかどうかは不明であることに留意する必要があります．

[*23] さらに余談ですが，同じ SVM の手法を用いて MNIST 画像分類 (4 と 9 のみに限らず，0 から 9 までの 10 クラス分類) の正答率は，95% を超えます [7]．あくまでも，量子 karnel 函数を用いた機械学習が将来的に実現しうるという可能性を秘めているだけで，実用化にはまだまだほど遠い手法であることを留意しておく必要があります．

付録 A　点と平面の距離

$\mathbb{R}^N$ において，いわゆる平面を表す式が $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$ とします．この時，$\boldsymbol{y} \in \mathbb{R}^N$ を考え，これと平面の距離を考えましょう．ここで，$\boldsymbol{w}$ がこの平面の法線ベクトルですから，平面内の点 $\boldsymbol{x}$ で，$\boldsymbol{y} - \boldsymbol{x} \parallel \boldsymbol{w}$ を満たす点を探し出せば $\|\boldsymbol{y} - \boldsymbol{x}\|$ が求める距離であることがわかります．この時，$\boldsymbol{x} = \boldsymbol{y} - a\boldsymbol{w}$ とかけ，

$$f(\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{y} - a\boldsymbol{w}) + b = 0$$

であるから，a についてとけば，

$$a = \frac{\boldsymbol{w}^T\boldsymbol{y} + b}{\|\boldsymbol{w}\|^2}$$

これより，

$$\|\boldsymbol{y} - \boldsymbol{x}\| = \|a\boldsymbol{w}\| = \frac{\left|\boldsymbol{w}^T\boldsymbol{y} + b\right|}{\|\boldsymbol{w}\|} \tag{40}$$

と求めることができます．

付録 B　双対表現

B.1　双対問題

得られた最適化問題 (8) に Lagrange の未定乗数法を利用することで双対問題を導出できます．具体的には，次のような Lagrange 函数 L:

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in[n]}\xi_i - \sum_{i\in[n]}\alpha_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i] - \sum_{i\in[n]}\mu_i\xi_i \tag{41}$$

この函数 L に関して $\boldsymbol{w}, b, \boldsymbol{\xi}$ は**主変数 (main variable)**，$\boldsymbol{\alpha}, \boldsymbol{\mu}$ を**双対変数 (dual variable)** と呼びます．ここで，次のような双対変数について最大化した函数 $P(\boldsymbol{w}, b, \boldsymbol{\xi})$:

$$P(\boldsymbol{w}, b, \boldsymbol{\xi}) := \max_{\boldsymbol{\alpha}\geq\boldsymbol{0}, \boldsymbol{\mu}\geq\boldsymbol{0}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{42}$$

を考えた時に，次の最適化問題

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} P(\boldsymbol{w}, b, \boldsymbol{\xi}) = \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \max_{\boldsymbol{\alpha}\geq\boldsymbol{0}, \boldsymbol{\mu}\geq\boldsymbol{0}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{43}$$

を考えます．この問題に関して，制約条件を満たす，すなわち**実行可能 (feasible)** である時を考えます．つまり，$\forall i, y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geq 0 \quad \text{and} \quad \xi_i \geq 0$ が成立しますので，明らか $\max_{\boldsymbol{\alpha}\geq\boldsymbol{0}, \boldsymbol{\mu}\geq\boldsymbol{0}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$ となる $\boldsymbol{\alpha}, \boldsymbol{\mu}$ はいずれも $\boldsymbol{0}$ となる時であります．また，実行可能でない場合はいくらでも α_i, μ_i を大きくすることができてしまい，L の値を大きくできるので上限が存在しないため，式 (43) を定義することはできません．以上より，式 (43) は結局，

$$= \begin{cases} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \dfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in[n]}\xi_i & \text{(実行可能である時)} \\ \\ \text{定義なし} & \text{(実行可能でない時)} \end{cases} \tag{44}$$

と書けますから，式 (43) は元の最適化問題と等価であることがわかります．ここで，主変数と双対変数の最大，最小を取る順を入れ替えた問題，すなわち**双対問題 (dual problem)** について考えてみましょう．すなわち $L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$ を主変数に関して最小化した函数を $D(\boldsymbol{\alpha}, \boldsymbol{\mu})$ とおけば，

$$\max_{\boldsymbol{\alpha}\geq\boldsymbol{0},\boldsymbol{\mu}\geq\boldsymbol{0}} D(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \max_{\boldsymbol{\alpha}\geq\boldsymbol{0},\boldsymbol{\mu}\geq\boldsymbol{0}} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{45}$$

と表される問題です．これに関して主変数による L の最小化は各変数の偏微分に関する条件：

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i = \boldsymbol{0} \tag{46}$$

$$\frac{\partial L}{\partial b} = -\sum_{i\in[n]} \alpha_i y_i = 0 \tag{47}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \tag{48}$$

を利用することによって，$\boldsymbol{\mu}$ を消去することができて，

$$D(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_{i\in[n]} \alpha_i \tag{49}$$

でありますから，結局偏微分の式による制約条件などを考えることによって，SVM の双対問題は

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & -\frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_{i\in[n]} \alpha_i \\ & \text{s.t.} \sum_{i\in[n]} \alpha_i y_i = 0 \text{ and } \forall i, 0 \leq \alpha_i \leq C \end{aligned} \tag{50}$$

と書き表せます．ここで，簡単のために最適な主変数，双対変数をそれぞれ $\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\mu}^*$ が存在すると仮定したときに，主問題と双対問題に対して以下の**強双対性 (strong duality)** が成立します[*24]．

$$D(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) = P(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*) \tag{51}$$

B.2 最適性条件

前節の議論より，双対問題の解は主問題の最適解に一致します．双対問題の解 $\boldsymbol{\alpha}$ を利用することによって，L の最小化を考えた際に $\boldsymbol{w}$ の偏微分が 0 となる制約条件より，

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i = \boldsymbol{0} \tag{52}$$

でありますから，決定函数 $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ のうち $\boldsymbol{w}$ を $\boldsymbol{\alpha}$ で消去することができて，

$$f(\boldsymbol{x}) = \sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x}_i + b \tag{53}$$

ここで b に関して，後ろで証明する以下の**相補性条件 (complementary condition)** という新たな条件[*25]

$$\forall i, \alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x} + b) - 1 - \xi_i) = 0 \tag{54}$$

$$\forall i, \mu_i\xi_i = 0 \tag{55}$$

[*24] 厳密な証明は省略します

[*25] Lagrange 函数に関する主変数による偏微分，主問題の制約条件，双対変数の非負性，相補性条件をまとめて，**Karush-Kuhn-Tucker condition(以下 KKT 条件)** と呼び，SV 分類の解の最適性の必要十分条件となっています．

を考えることによって $\boldsymbol{\alpha}$ で消去することができます．その結果，

$$b = y_i - \sum_{j\in[n]} \alpha_j y_j \boldsymbol{x}_j^T \boldsymbol{x}_i \quad i \in \{i \in \{1, \cdots, n\} \mid 0 < \alpha_i < C\} \tag{56}$$

これは，条件を満たす任意の i に関して成立しますが，実際の実装では数値計算の安定性を考慮して条件を満たす全ての i に関して計算を行った b の平均を取ることもあります．以上より，双対問題の解を求めれば決定函数を求めることができることが示せました．

B.3　相補性条件

相補性条件は，強双対性から導かれる性質で次のように求めることができます：

$$\begin{aligned}
P(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*) &= D(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) \\
&= \min_{\boldsymbol{w},b,\boldsymbol{\xi}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) \\
&= \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in[n]} \xi_i - \sum_{i\in[n]} \alpha_i^*[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i] - \sum_{i\in[n]} \mu_i^* \xi_i \\
&= \frac{1}{2}\|\boldsymbol{w}^*\|^2 + C\sum_{i\in[n]} \xi_i^* - \sum_{i\in[n]} \alpha_i^*[y_i(\boldsymbol{w}^{*T}\boldsymbol{x}_i + b^*) - 1 + \xi_i^*] - \sum_{i\in[n]} \mu_i^* \xi_i^*
\end{aligned} \tag{57}$$

ここで，実行可能である時，$\forall i, y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geq 0 \quad \text{and} \quad \xi_i \geq 0$ が成立しますので

$$\leq \frac{1}{2}\|\boldsymbol{w}^*\|^2 + C\sum_{i\in[n]} \xi_i^* = P(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*) \tag{58}$$

従って，強双対性の条件から最後の不等号は等号となり，式 (57) の第三，四項は 0 となる必要があるため，

$$\forall i, \alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x} + b) - 1 - \xi_i) = 0 \tag{59}$$

$$\forall i, \mu_i \xi_i = 0 \tag{60}$$

が成立します．

参考文献

[1] 竹内一郎，鳥山昌幸 (2015).『サポートベクトルマシン』講談社

[2] Maria Schuld，Francesco Petruccione 著，大関真之 監訳 (2020).『量子コンピュータによる機械学習』共立出版社

[3] 立川裕二 (2020).『駒場現代物理学 (2020) 第 6 回講義ノート』

[4] Nielsen. M. A，Chuang, I. L. (2000). Quantum Computation and Quantum Information. Cambridge University Press.

[5] Havlíček, V., Córcoles, A.D., Temme, K. et al. Supervised learning with quantum-enhanced feature spaces. Nature 567, 209–212 (2019).

[6] Yunchao Liu, Srinivasan Arunachalam, Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning Nat. Phys. 17, 1013–1017 (2021).

[7] Decoste, D., Schölkopf, B. Training Invariant Support Vector Machines. Machine Learning 46, 161–190 (2002).