

量子コンピュータの機械学習への応用

御手洗光祐

近年の機械学習の目覚ましい発展は、計算機の高速度が強力にサポートしてきた。そこで量子コンピュータを機械学習に利用するという方向性が近年活発に研究されている。本集中ゼミでは、代表的な量子機械学習アルゴリズムについて概観し、その可能性や課題について議論する。量子機械学習アルゴリズムは、おおまかに Noisy Intermediate Scale Quantum (NISQ) デバイスでの近未来応用を主眼に置いたものと、誤り訂正後の万能量子コンピュータを見据えたものに大別される。これら2つの方向性について代表的なアルゴリズムを紹介するために、はじめに機械学習や量子コンピュータの基礎的な事項を最初におさえる。その後前半で、NISQ デバイスに向けた量子アルゴリズムを紹介する。特にパラメータ付きの量子回路を用いた機械学習手法について概観する。その後後半では、量子サポートベクターマシンや量子線形回帰など、長期的応用を見据えたアルゴリズムについて議論する。量子機械学習自体は直接的に物性物理に関連している分野とは言えないかもしれないが、例えば量子機械学習アルゴリズムを物理系の相転移検出に応用してみるなど、物性物理学への応用のアイデアは絶えない。本集中ゼミが参加者の議論の種になれば嬉しい。

1 量子計算

はじめに量子計算に関する基礎的な事項を押さえておく。以下は最低限の概略なので、詳しくは標準的な教科書 [1] や資料 [2] を参考にしてほしい。

1.1 量子ビットとテンソル積

量子コンピュータとは、古典ビットの代わりに量子ビットを用いる計算機である。量子ビットとは量子力学的な2準位系のことで、通常量子計算の分野ではそれぞれの準位を $|0\rangle$ と $|1\rangle$ で表す。量子ビットは量子力学に従うので $|0\rangle$ と $|1\rangle$ の重ね合わせ状態を取ることができ、任意の1量子ビットの状態は複素数 α, β を使って、 $\alpha|0\rangle + \beta|1\rangle$ と書ける。 α, β は規格化条件 $|\alpha|^2 + |\beta|^2 = 1$ を満たし、この状態を測定すると確率 $|\alpha|^2, |\beta|^2$ でそれぞれ $|0\rangle, |1\rangle$ が観測される。

さらに n 量子ビットは、 $|0\rangle$ と $|1\rangle$ で張られる2次元複素ベクトル空間のテンソル積によって記述される。つまり n 量子ビットの状態ベクトルは、 $\{|0\rangle, |1\rangle\}^{\otimes n}$ によって張られる 2^n 次元複素ベクトル空間に存在する単位ベクトルである。このベクトル空間の基底の中でも $\{|0\rangle, |1\rangle\}^{\otimes n}$ は特に計算基底と呼ぶ。計算基底は $|0\rangle \otimes |1\rangle \otimes \dots \otimes |0\rangle$ のようなベクトルだが、量子計算の分野では簡単のため $|0\rangle|1\rangle\dots|0\rangle$ あるいは $|01\dots0\rangle$ のようにテンソル積記号を省略して書くことが多

い. またビット列を整数の 2 進数表現だとみなして整数を対応付け, $\{|0\rangle, |1\rangle\}^{\otimes n} = \{|k\rangle\}_{k=0}^{2^n-1}$ と表記することも多々ある. この記法を用いると, 任意の n 量子ビットの量子状態は 2^n 個の複素数 $\{c_k\}$ を用いて $\sum_{k=0}^{2^n-1} c_k |k\rangle$ と書ける. 1 量子ビットの場合と同様にこれらの係数は規格化条件 $\sum_{k=0}^{2^n-1} |c_k|^2 = 1$ を満たし, この状態に対して測定を行うと確率 $|c_k|^2$ で $|k\rangle$ という結果が得られる.

1.2 量子ゲート・量子回路

量子ビットに対するユニタリ操作を量子ゲートと呼ぶ. 1 量子ビットゲートとして代表的なものは x, y, z -軸回転ゲートであり, 以下のように定義される.

$$R_\alpha(\theta) = \exp(-i\theta\sigma_\alpha/2), \quad (1)$$

ここで σ_α ($\alpha = x, y, z$) はパウリ行列

$$\sigma_x = |0\rangle\langle 1| + |1\rangle\langle 0|, \quad (2)$$

$$\sigma_y = -i|0\rangle\langle 1| + i|1\rangle\langle 0|, \quad (3)$$

$$\sigma_z = |0\rangle\langle 0| - |1\rangle\langle 1|, \quad (4)$$

である. これらの 1 量子ビットゲートは実験上も比較の実装しやすく, ほとんど任意の θ について精度よく実現することが可能となっている.

次に代表的な 2 量子ビットゲートもいくつか挙げておこう. 制御 NOT ゲートは, 制御量子ビットが 1 のときのみターゲット量子ビットを反転するゲートとして定義され, 次のように表される.

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \sigma_x \quad (5)$$

$$= |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11|. \quad (6)$$

ここで I は恒等演算子で $I = |0\rangle\langle 0| + |1\rangle\langle 1|$ である. また制御 Z ゲートは制御量子ビットが 1 のときのみターゲット量子ビットに σ_z を作用させるゲートであり,

$$\text{CZ} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \sigma_z \quad (7)$$

$$= |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| - |11\rangle\langle 11|, \quad (8)$$

と定義される. 最後に SWAP ゲートは 2 量子ビットの状態を交換する

$$\text{SWAP} = |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11|, \quad (9)$$

というゲートである. これらの 2 量子ビットゲートは 1 量子ビットゲートと比べると実験的な実現が難しく, 比較的精度が低くなる.

ちなみにこのうち CNOT ゲートまたは CZ ゲートと $R_\alpha(\theta)$ ゲートを使えば, 任意の量子計算が行えることが知られている [1]. つまりこれらの量子ゲートをうまく組み合わせることによって, 例えばショアの素因数分解アルゴリズムなどを構成できるのだ.

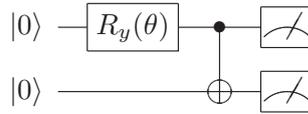


図 1 量子回路

量子計算では、量子ビットに対してこれらのゲートを順々に作用させていくことで計算を行うが、そのゲート列のことを量子回路と呼ぶ。量子回路はよく図 1 のように量子回路図によって表現される。図 1 の各水平線が 1 つの量子ビットを表し、左から順に量子ゲートがかけられていく様子を示している。最後の記号 \square は測定を表す。CNOT や CZ, SWAP はそれぞれ特別な記号が与えられていて、それぞれ

$$\text{CNOT} = \begin{array}{c} \text{control} \\ \text{target} \end{array} \begin{array}{c} \bullet \\ \oplus \end{array}, \quad \text{CZ} = \begin{array}{c} \bullet \\ \bullet \end{array}, \quad \text{SWAP} = \begin{array}{c} \times \\ \times \end{array}, \quad (10)$$

のように表される。

1.3 量子回路のシミュレーション困難性

ある一定の大きさで、かつ十分複雑な量子回路を古典コンピュータで効率的にシミュレートすることは困難であると考えられている。ナイーブには、 n 量子ビットの一般の状態 $\sum_k c_k |k\rangle$ を記述するのに必要な複素数 c_k が n に対して指数 (2^n) 個あることがその困難性の原因である。例えば単精度小数で各複素数をメモリ上に確保したとしても、 $n = 50$ の時点で PB (ペタバイト) のオーダーのメモリが必要となる。これはスーパーコンピュータ富岳の全ての RAM (ランダムアクセスメモリ) を使い切るレベルである。また、もし一般の多項式時間量子回路を効率的にシミュレートできるようなアルゴリズムがあれば、そのシミュレータによってショアの素因数分解アルゴリズムをシミュレートすることによって、素因数分解の多項式時間古典アルゴリズムが完成してしまうこともその困難性の傍証になるだろう*1。

さらに、一般の多項式時間量子回路ではなく、ある限られたクラスの量子回路であっても、その古典シミュレーションが難しいことが (計算量理論で広く成り立つと信じられている仮定のもとで) 証明されている。その理論的後押しをもとに、現在ベストの古典アルゴリズムを持ってしても、実験的に実現した量子回路の古典シミュレーションが困難となるようなスキーム (量子超越) を目指したデバイス開発が進められてきた。その成果として Google のグループが 53 量子ビットを備えたデバイスを製作し、ランダム量子回路と呼ばれる量子回路を実行することで量子超越を達成したと発表した [3]。またこれに次いで、中国のグループもガウシアンボソンサンプリングと呼ばれるタスクを実験的に実現し、同じく量子超越を達成したと発表している [4]。これらの最近のデバイス

*1 逆に言えば、一般のハミルトニアンに対して、そのもとでの実時間発展が効率的に古典シミュレートできるようなアルゴリズムを考え出せれば、素因数分解ができることになり、すなわち RSA 暗号を破ることもつながる。物性分野のシミュレータを使って効率的な量子回路シミュレータを構築してみるのも面白いだろう。

は、ある特定のタスクにおいては古典コンピュータを「超越」している一方で、外界からのノイズに非常に弱いという問題を抱えている。このことを端的に示すために、最近のデバイスは noisy intermediate scale quantum デバイスを縮めて NISQ デバイスと呼ばれている。ノイズに弱いとはいえども、ある特定のタスクにおいてすでに実現している NISQ デバイスの性能が古典コンピュータの能力を凌駕していることは確かである。そこで近年これらのデバイスを何らかの形で利用するためのアルゴリズム研究が活発に行われている。

2 機械学習とは

この節も必要最低限の概略なので、詳しくは [5] などの教科書を参考にしてほしい。

2.1 教師あり機械学習

機械学習は教師なし機械学習と教師あり機械学習に大別される。ここでは教師あり機械学習について述べる。教師あり機械学習では、訓練データとして M 次元で N 個の入力データ (説明変数) $\{\mathbf{x}_i\}$ ($i = 1, \dots, N$) と、対応する N 個の教師データ $\{y_i\}$ ($i = 1, \dots, N$) が与えられる。例えば「手書き数字を分類する」というタスクでは、 \mathbf{x}_i は手書き数字の画像データであり、 y_i は各画像のラベルになる。

教師データ $\{y_i\}$ は、何らかの形で説明変数 $\{\mathbf{x}_i\}$ と関連づいていると考えられる。背後に存在するこの関係を仮に g として $g(\mathbf{x}_i)$ が教師データ y_i を与えると考えよう。機械学習の目的は、与えられた訓練データから g を近似する関数 f を構築することである。 f をデータから構築することを学習と呼び、学習によって構築される関数 f をモデルと呼ぶ。

通常モデル f には何らかの形のパラメータ付き関数を仮定し、パラメータを調整することによって学習を行う。以下に少し例を挙げよう。

- 線型回帰 $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$.
- ロジスティック回帰 $f_{\mathbf{w},b}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T \mathbf{x} + b}}$.
- ニューラルネットワーク $f_{\{W_i\}}(\mathbf{x}) = h_L(W_L h_{L-1}(\dots W_3 h_2(W_2 h_1(W_1 \mathbf{x}))))$. ここで W_i は行列であり、 h_i は何らかの非線形関数である。

これらのモデルでは、 \mathbf{w}, b や W がパラメータとなっている。

2.2 特徴量

モデルを構築する際、入力 \mathbf{x}_i をそのまま使わずに、それをなんらかの関数 $\phi(\mathbf{x})$ によって変換したのを使ったほうが学習が簡単になることがある。例として、円形データの分類問題を図 2 に示した。図 2 の左側のデータではどう頑張っても直線で分類することはできない。一方で $\phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2)$ という変換を作用させれば、各データ点は図 2 右のように変換され、平面で簡単に分離することができるようになる。このように変換された入力データ $\phi(\mathbf{x})$ のことを特徴

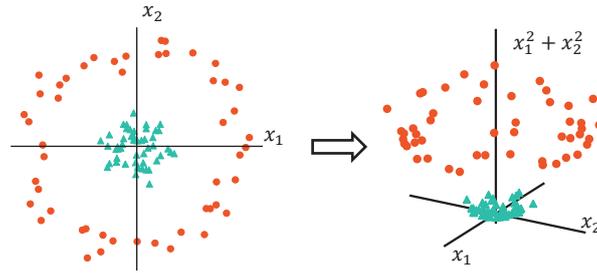


図2 円形データの分類問題

量と呼ぶ。高次元の特徴量を使うことによって、一見非線型なモデルが必要そうに見えるデータに対しても、直線や平面のような線型モデルで十分となることがある。

2.3 カーネル法

特徴量の考え方はとても強力だが、高次元の特徴量を明示的に扱っては計算量が大きくなってしまうこともある。例として線型回帰を考えよう。線型回帰では、 $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ のパラメータ \mathbf{w} を、二乗損失

$$L(\mathbf{w}) = \sum_i |f_{\mathbf{w}}(\mathbf{x}_i) - \mathbf{y}_i|^2, \tag{11}$$

が最小となるように決める。その解は、

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix}, \tag{12}$$

また $\mathbf{y} = \{y_i\}$ として $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ で与えられる。特徴量を使った線型回帰ではモデルを $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ と構成する。このとき $L(\mathbf{w}) = \sum_i |f_{\mathbf{w}}(\mathbf{x}_i) - \mathbf{y}_i|^2$ が最小となるような \mathbf{w} は上と全く同様に、

$$\Phi = \begin{pmatrix} \phi(\mathbf{x}_1^T) \\ \vdots \\ \phi(\mathbf{x}_N^T) \end{pmatrix}, \tag{13}$$

として、 $\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$ で与えられる。特徴量 ϕ の次元が非常に大きければ、行列 Φ を計算するだけでもかなりの計算量を必要とするかもしれない。

これを回避する方法としてカーネル法がある。これは特徴量空間での内積 (カーネル)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \tag{14}$$

のみを用いて、モデルを構成する手法である。カーネル法ではグラム行列 $K = \{K_{ij}\} = k(\mathbf{x}_i, \mathbf{x}_j)$ が中心的な役割を果たす。例えば線型回帰では、次のようなモデル

$$f(\mathbf{x}) = \sum_{ij} y_i (K^{-1})_{ij} k(\mathbf{x}_j, \mathbf{x}), \tag{15}$$

が $\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$ とした $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ と全く一致する. 式 (15) には明示的な特徴量 ϕ は全く現れておらず, 内積 $k(\mathbf{x}_i, \mathbf{x}_j)$ さえ効率的に計算できるような ϕ であれば ϕ の次元は全く問題にならなくなる. したがってカーネル法を使えば無限次元の特徴量空間での線型回帰すら可能である. 例えばよく使われるガウシアンカーネルは

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2), \quad (16)$$

の形で, 対応する特徴量は無限次元の

$$\phi(\mathbf{x}) = \left\{ \frac{1}{\sqrt{k!^{\frac{1}{k}}}} e^{-\frac{1}{2k}\|\mathbf{x}\|^2} \binom{k}{n_1, \dots, n_N}^{\frac{1}{2}} x_1^{n_1} \dots x_N^{n_N} \mid k \in \mathbb{N}, \sum_{j=1}^M n_j = k \right\}, \quad (17)$$

である [6].

3 NISQ デバイスによる機械学習

3.1 変分量子アルゴリズム

NISQ デバイスを応用するために考えられた手法の多くが**変分量子アルゴリズム**という枠組みに入る. 変分量子アルゴリズムと呼ばれているものの大まかなアルゴリズムは以下のようなになる.

1. 適当なパラメータ θ を持つユニタリーゲート $U(\theta)$ を初期状態 $|0\rangle$ に作用させ, $|\psi(\theta)\rangle := U(\theta)|0\rangle$ を作る.
2. $|\psi(\theta)\rangle$ について何らかのオブザーバブルを測定する.
3. 測定結果によってパラメータを更新し, θ' を使って 1, 2 を繰り返す.

$U(\theta)$ は NISQ デバイスのノイズ下でも実行できるような回路深さでなくてはならない. 例えば Google のデバイス [3] では各量子ビットに 20 回程度 2 量子ビットゲートをかけるのが限界である. 変分量子アルゴリズムは, 回路形 $U(\theta)$ を先に実験で実現できるようなものに固定してからパラメータだけを調整するという手続きをとるので, 比較的 NISQ に適したアルゴリズムであると考えられている.

3.2 量子回路学習

量子回路学習 (quantum circuit learning, QCL) [7] は NISQ 上で教師ありの機械学習を行うための変分量子アルゴリズムである. アルゴリズムを以下に示す. 図 3 はその概要である.

1. それぞれの \mathbf{x}_i について, 適当な量子ゲート $V(\mathbf{x}_i)$ を使って, データをエンコードした状態 $|\varphi(\mathbf{x}_i)\rangle := V(\mathbf{x}_i)|0\rangle^{\otimes n}$ を作る.
2. パラメータ付きの量子回路 $U(\theta)$ を使って $|\psi(\mathbf{x}_i, \theta)\rangle := U(\theta)|\varphi(\mathbf{x}_i)\rangle$ を作る.
3. $|\psi(\mathbf{x}_i, \theta)\rangle$ に対して適当なオブザーバブルを観測しその期待値を得る. その値を \hat{y}_i とする. 例えば 1 番目の量子ビットの σ_z 期待値などを取る.

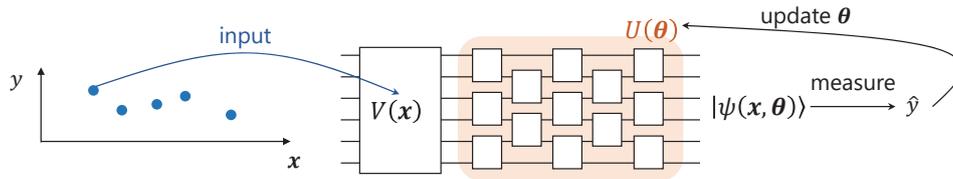


図3 量子回路学習の概念図.

4. \hat{y}_i と教師データ y_i の差が小さくなるようにパラメータ θ を調節し, 1 に戻る.

このアルゴリズムが構築するモデルは, 例えば 1 番目の量子ビットの σ_z 期待値を出力として測定するとき,

$$f_{\theta}(\mathbf{x}) = \langle \psi(\mathbf{x}, \theta) | \sigma_z \otimes I^{\otimes n-1} | \psi(\mathbf{x}, \theta) \rangle, \quad (18)$$

である. 簡単な例として \mathbf{x} の次元 M が量子ビット数 n に等しいとき, このモデル $f_{\theta}(\mathbf{x})$ はどのような構造を持つか調べよう. 入力ゲート $V(\mathbf{x})$ としては, 量子回路をできるだけ浅くするため 1 量子ビットの回転ゲート

$$V(x_i) |0\rangle = x_i |0\rangle + \sqrt{1 - x_i^2} |1\rangle, \quad (19)$$

をそれぞれ独立に i 番目の量子ビットに作用させることにする. このとき $|\varphi(\mathbf{x})\rangle$ は

$$|\varphi(\mathbf{x})\rangle = \bigotimes_{i=1}^n \left(x_i |0\rangle + \sqrt{1 - x_i^2} |1\rangle \right), \quad (20)$$

となる. ここで重要なのは, テンソル積構造によって元のデータ \mathbf{x} に対する非線形性が生まれることである. 例えば $|0\rangle^{\otimes n}$ の係数は $x_1 x_2 \cdots x_n$ になる. 量子ビット数に対して指数的な種類の非線形項が生成されることがわかるだろう. このように生まれた非線形項が, $U(\theta)$ によって組み合わせられてモデル $f_{\theta}(\mathbf{x})$ に現れるのだから, 量子デバイスを使うことで指数関数的に多数の基底関数によって $f_{\theta}(\mathbf{x})$ を構成できることになる.

QCL は上のような原理によって, 古典コンピュータにはできない機械学習を可能にするかもしれない. 一方で古典コンピュータでもガウシアンカーネルなどを使うことによって無限次元の特徴量空間を利用することが可能なので, QCL は特に恩恵をもたらさないかもしれない. QCL の優位性についてはまだまだ議論の余地があり, 今後もっと具体的な機械学習タスクに適用して, その性能を調べていく必要がある.

3.3 量子カーネル法

量子カーネル法のアイデアは, 適当な量子回路 $U(\mathbf{x})$ で作った量子状態 $|\psi(\mathbf{x})\rangle = U(\mathbf{x}) |0\rangle^{\otimes n}$ をそのまま特徴量ベクトルとして用いてカーネル法を展開するというものである [8, 9]. このときカーネル $k(\mathbf{x}_i, \mathbf{x}_j)$ は量子状態間の内積

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i) | \psi(\mathbf{x}_j) \rangle, \quad (21)$$

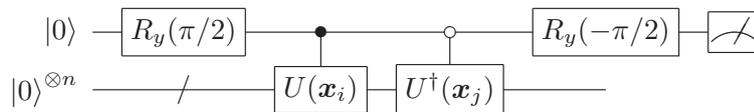


図4 $\langle \psi(\mathbf{x}_i) | \psi(\mathbf{x}_j) \rangle$ の実部を測定する回路. 虚部を測定するには $R_y(-\pi/2)$ の前に $R_z(\pi/2)$ を挿入する.

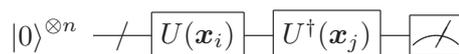


図5 $\text{Tr}(\rho(\mathbf{x}_i)\rho(\mathbf{x}_j))$ を測定する回路.

ととればよい. しかしここで問題になるのは, 位相を含めた $\langle \psi(\mathbf{x}_i) | \psi(\mathbf{x}_j) \rangle$ の測定は図4のように少し複雑な回路が必要となることである. しかし少し見方を変えて, 量子状態 $|\psi(\mathbf{x})\rangle$ に対応する密度演算子 $\rho(\mathbf{x}) = |\psi(\mathbf{x})\rangle \langle \psi(\mathbf{x})|$ を特微量ベクトルとみなすと回路が簡単化される. 密度演算子の内積は $k(\mathbf{x}_i, \mathbf{x}_j) = \text{Tr}(\rho(\mathbf{x}_i)\rho(\mathbf{x}_j)) = |\langle \psi(\mathbf{x}_i) | \psi(\mathbf{x}_j) \rangle|^2$ と定義できるので, 図5のような回路で測定できるようになるのだ. 制御 U ゲートを含まないこの形であれば実験的にも実現しやすい. 筆者もこの方向性での研究を行っており, 量子カーネル法を用いた機械学習の実験的デモンストレーションを [10] で報告した. [10] では, $U(\mathbf{x})$ として入力データ \mathbf{x} に依存したハミルトニアン $H(\mathbf{x})$ による時間発展 $e^{-iH(\mathbf{x})t}$ を使い, 実験的にカーネル $k(\mathbf{x}_i, \mathbf{x}_j)$ を測定して学習を行った.

量子回路を使って内積を計算するという性質上, 少なくとも量子カーネル法と全く同じことを古典コンピュータで行うことは困難であるが, QCL と同じように実用的なタスクに対してどのくらいの性能が出るかはまだまだ不透明な状況だ. さらなる研究が望まれている.

4 誤り耐性を前提とした量子機械学習アルゴリズム

このセクションでは長期的な誤り耐性量子コンピュータを前提とした量子機械学習アルゴリズムを紹介する. アルゴリズムの詳細はかなり込み入った議論が必要なので, ここではその概要を述べるに留める.

4.1 データの取り扱い

いわゆる「ビッグデータ」に対する機械学習を高速化したいという要求に答えるためには, データ数 N に対する計算の高速化を要求することが自然だろう. 例えば線型回帰のような機械学習アルゴリズムは $\text{poly}(N)$ 時間の計算量が必要である. そこでこれを量子コンピュータによって $\text{poly} \log(N)$ 時間へ加速するための理論提案が行われてきた. しかしよく考えてみると, ナイーブにはデータを読み込むだけでも $O(N)$ 時間が必要となりそうである. なぜそんな加速が現実的になるのだろうか.

必要となるのは、 N 個のデータを $\text{poly log}(N)$ 時間で読み込めるアーキテクチャである。そのようなアーキテクチャを量子ランダムアクセスメモリ (quantum random access memory, QRAM) と呼んでいる。QRAM とは、あるバイナリデータ x_i のアドレス i を記述する量子ビット列 $|i\rangle$ が与えられたとき、そのデータ x_i を量子ビット列 $|x_i\rangle$ として取り出す機能のことである。具体的には、次のような機能を $\text{poly log}(N)$ 時間で実装するものを QRAM と呼ぶ。

$$|i\rangle |0\rangle \xrightarrow{\text{QRAM}} |i\rangle |x_i\rangle. \quad (22)$$

古典コンピュータの RAM も同じ機能を持ち、一回のクエリをデータ数 N に対して $\text{poly log } N$ 時間で処理できるが、QRAM は上記の変換をユニタリーなプロセスとして実現するところが異なる。つまり QRAM は以下のように重ね合わせ状態を入力すれば、同時並列的に N 個のデータ全てを量子状態にエンコードできるのだ。

$$\frac{1}{\sqrt{N}} \sum_{i=1}^N |i\rangle |0\rangle \xrightarrow{\text{QRAM}} \frac{1}{\sqrt{N}} \sum_i |i\rangle |x_i\rangle. \quad (23)$$

さらに、QRAM に特定のデータ構造をもたせれば、振幅にデータ x_i をエンコードした状態 $\sum_i x_i |i\rangle$ を作れることが知られている [11].

注意しておかなければならないのは、QRAM が実現すればデータ読み込みの問題が解決されるが、物理的に QRAM を実現可能であるかどうかはまだわからないということだ。具体的な実装方法の提案 [12] はあるものの、QRAM の誤り訂正が効率的に可能かどうかには議論の余地がある [13].

4.2 データ数について加速する量子機械学習アルゴリズム

先に述べたように、線型回帰の解は $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$ によって与えられる。この解 \mathbf{w} を使って未知の入力データ \mathbf{x} に対して $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ によって予測を行うのだった。したがってもし $|\mathbf{w}\rangle = \sum_i w_i |i\rangle$ という状態を作り出すことができれば、この状態と未知の入力データをエンコードした状態 $|\mathbf{x}\rangle = \sum_i x_i |i\rangle$ との内積を測定することによって、線型回帰による予測が行える。

このようなタスクに使える有用なアルゴリズムとして、Harrow, Hassidim, Lloyd によって提案された HHL アルゴリズムがある [14]. HHL アルゴリズムは $N \times N$ の行列 A についての線型連立方程式 $A\mathbf{v} = \mathbf{b}$ の解を、 $|\mathbf{v}\rangle = \sum_i (A^{-1}\mathbf{b})_i |i\rangle$ という量子状態の形で出力するアルゴリズムで、適当な仮定のもとで $\text{poly log}(N)$ 時間で動作する。^{*2} このアルゴリズムに対して少しの変更を加えることで、 $|\mathbf{w}\rangle$ を高速に生成するアルゴリズムが [15] で提案された。もちろんこのアルゴリズムには QRAM が必須である。

他にも HHL アルゴリズムを少し変更することで、推薦システムの量子加速 [11] や、分類タスクに使われるサポートベクトルマシンの量子加速 [16] などが達成されている。これらは指数的な

^{*2} ただし $|\mathbf{v}\rangle$ が得られるだけであって、 \mathbf{v} の全ての成分 v_i を得るためには何度も測定を繰り返す必要がある。そしてそのためには通常 $\mathcal{O}(N)$ 回の測定が必要であり、 v_i を直接得ようとするとは指数加速がキャンセルされてしまうことには注意が必要である。

加速を達成するものであり、量子コンピュータの実応用先として有望視されていたが、ごく最近になってこれらの問題に対して同様に指数加速を達成する古典アルゴリズムが発表された [17, 18]. 特筆すべきは、この古典アルゴリズムが量子アルゴリズムに触発されたもの (quantum-inspired algorithms) であったことである. これは入力データ \boldsymbol{x} について $\sum_i x_i |i\rangle$ という状態を作るためのデータ構造 [11] が、古典コンピュータにとっても有用であることが見いだされたためである. 現在、この古典アルゴリズムの応用先の模索とともに、他の量子アルゴリズムでもこのような脱量子化 (dequantization) を目指す研究が精力的に進められている.

5 おわりに

量子コンピュータ、機械学習の基礎からはじめ、現在の量子機械学習の概略を解説した. 量子コンピュータと機械学習という2つの「パスワード」をかけ合わせたような分野で注目を集めてはいるものの、NISQ 向けのアルゴリズム、誤り耐性量子コンピュータ向けのアルゴリズムのどちらにしても、まだまだ解決すべき問題が山積みであることがわかっていただけたかと思う*3. すでに非常に高いレベルに発展している古典コンピュータ上の機械学習アルゴリズムとまともに戦うためには、さらなる研究が必須だ.

一つの方向性として、何らかの形で量子的なデータの機械学習を行えると面白いと思う. 例えばあるハミルトニアン H の何らかの性質を学習するために、そのハミルトニアンによる時間発展演算子 e^{-iHt} を量子カーネル法の $U(\boldsymbol{x})$ として使ってみる、などというアイデアが考えられる. e^{-iHt} のような演算子は古典コンピュータで効率的に扱うことが難しいので、このセットアップであれば量子コンピュータを使うご利益があるかもしれない. このような方向性で物性物理分野の問題に量子機械学習を適用してみることは大いに意義があるだろう. 本集中ゼミの参加者が、何らかの形で量子機械学習分野に参入していただけるようなことがあれば大変うれしく思う.

参考文献

- [1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2011).
- [2] Quantum native dojo, <https://dojo.qulacs.org/>.
- [3] F. Arute *et al.*, *Nature* **574**, 505 (2019).
- [4] H.-S. Zhong *et al.*, *Science* **370**, 1460 (2020).
- [5] C. M. Bishop, *Pattern Recognition and Machine Learning* (, 2006).
- [6] A. Shashua, Introduction to machine learning: Class notes 67577, 2009, arXiv:0904.3664.
- [7] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, *Phys. Rev. A* **98**, 032309 (2018).
- [8] V. Havlíček *et al.*, *Nature* **567**, 209 (2019).

*3 逆に言えばいくらでも研究できるということ.

- [9] M. Schuld and N. Killoran, *Phys. Rev. Lett.* **122**, 040504 (2019).
- [10] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, Experimental quantum kernel machine learning with nuclear spins in a solid, 2019, arXiv:1911.12021.
- [11] A. Prakash, *Quantum Algorithms for Linear Algebra and Machine Learning.*, PhD thesis, EECS Department, University of California, Berkeley, 2014.
- [12] V. Giovannetti, S. Lloyd, and L. Maccone, *Phys. Rev. A* **78**, 052310 (2008).
- [13] S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, and P. V. Srinivasan, *New Journal of Physics* **17**, 123010 (2015).
- [14] A. W. Harrow, A. Hassidim, and S. Lloyd, *Phys. Rev. Lett.* **103**, 150502 (2009).
- [15] M. Schuld, I. Sinayskiy, and F. Petruccione, *Phys. Rev. A* **94**, 022342 (2016).
- [16] P. Rebentrost, M. Mohseni, and S. Lloyd, *Phys. Rev. Lett.* **113**, 130503 (2014).
- [17] E. Tang, A quantum-inspired classical algorithm for recommendation systems, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 217–228, 2019.
- [18] N.-H. Chia *et al.*, Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning, in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 387–400, 2020.