Vol.2022-QS-5 No.8
2022/3/24

量子特異値分解の脱量子化による エクストリーム機械学習の高速化

武田 伊織[1,a)]　高比良 宗一[1,b)]　御手洗 光祐[1,2,3,c)]　藤井 啓祐[1,2,4,d)]

概要：2016 年に Kerenidis と Prakash によって量子推薦システムが提唱され、量子計算機上で $O(\mathrm{poly}(\log n))$ で次元 n の行列の特異値分解が可能であることが示された。さらに、2018 年に Tang によって量子インスパイアアルゴリズム [1] が提唱され、適切なサンプリングを行えば古典計算機でも同様に $O(\mathrm{poly}(\log n))$ で特異値分解が計算可能であることが示された。このアルゴリズムは、入力データにセグメント木構造を持たせ、行列の行と列を乱択することで行列の次元圧縮を行い、圧縮した行列を特異値分解した後、得られた特異ベクトルなどをもちいて元の行列の特異ベクトルを復元するというものである。このように、量子計算機のアルゴリズムを古典計算機でも同様の計算量で行えるようにすることを脱量子化という。これらのアルゴリズムは、低ランク近似を行なっており、行列のランクが小さい場合、良い近似を与える。本研究では、量子インスパイア特異値分解の機械学習への応用を提案し、機械学習で用いられる標準的なデータセットにおいて低ランク近似が有効であるかどうかを数値的に検証する。

1. 序論

2016 年に Kerenidis と Prakash によって量子推薦システム [2] が提唱され、$O(\mathrm{poly}(\log n))$ で次元 n の行列の低ランク近似された特異値分解が量子計算機上で可能であることが示された。2018 年には Tang によって、古典計算機上でも行列にセグメント木構造を持たせ、サンプリングによって行列の行と列を乱択することで、同じ計算量で低ランク近似された特異値分解が計算可能であることが示された。これを量子インスパイアアルゴリズム [1] という。このアルゴリズムは 2004 年に Frieze らによって発表された、高速モンテカルロ法をもちいた行列の低ランク近似の探索 [3] に基づいており、そのことから修正 FKV アルゴリズムと言われている。その後、量子インスパイアアルゴリズムをもちいて線形回帰問題を高速化する方法 [4] が 2018 年に Gilyén らによって発表されている。このように、量子アルゴリズムを古典アルゴリズムに応用し、古典アルゴリズムを高速化することを脱量子化という。

本研究では、量子インスパイアアルゴリズムが行列の低ランク近似された特異値分解を高速に計算できることに注目し、機械学習への応用を提案する。教師あり機械学習の回帰の問題において、行列の擬似逆行列を計算する必要があり、この擬似逆行列は特異値分解を行うことで計算される。この特異値分解において、従来の古典アルゴリズムでは $O(\mathrm{poly}(n))$ の時間がかかっているため、量子インスパイアアルゴリズムによって計算の高速化が可能であるかを調査する。具体的な手法として、2004 年に Huang らによって提唱されたエクストリーム機械学習 [5] という手法をもちいる。

2. 手法

2.1 修正 FKV アルゴリズム

ランク r の行列 $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ が与えられたとき、以下のように 3 つの成分に分解することを特異値分解という。

[1] 大阪大学基礎工学研究科, 〒 560-8531 大阪府豊中市待兼山町 1–13
Graduate School of Engineering Science, Osaka University, 1–13 Toyonaka city, Osaka prefecture
[2] 大阪大学 量子情報・量子生命研究センター, 〒 560-0043 大阪府豊中市待兼山町 1–2
Center for Quantum Information and Quantum Biology, Institute for Open and Transdisciplinary Research Initiatives, Osaka University, 1–2 Toyonaka city, Osaka prefecture
[3] JST さきがけ, 〒 332-0012 埼玉県川口市本町 4–1–8
JST, PRESTO, 4–1–8 Honcho, Kawaguchi city, Saitama prefecture
[4] 理化学研究所 創発物性科学研究センター, 〒 351-0198 埼玉県和光市
Center for Emergent Matter Science, RIKEN, Wako city, Saitama prefecture
[a)] u676304i@ecs.osaka-u.ac.jp
[b)] takahira@qc.ee.es.osaka-u.ac.jp
[c)] mitarai@qc.ee.es.osaka-u.ac.jp
[d)] fujii@qc.ee.es.osaka-u.ac.jp

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \tag{1}$$

$$= \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_r \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_r^{\mathrm{T}} \end{pmatrix}. \tag{2}$$

ただし、$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$。このとき、$\boldsymbol{X}$ の特異値のうち、σ_{k+1} 以降は非常に小さいとして、$\boldsymbol{X} \simeq \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k (\boldsymbol{U}_k \in \mathbb{R}^{m\times k}, \boldsymbol{V}_k \in \mathbb{R}^{n\times k}, \boldsymbol{\Sigma}_k \in \mathbb{R}^{k\times k})$ で以下のように近似することを、行列の低ランク近似という。

$$\boldsymbol{X} \simeq \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^{\mathrm{T}} \tag{3}$$

$$= \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_k \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_k \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_k^{\mathrm{T}} \end{pmatrix}. \tag{4}$$

また、行列 $\boldsymbol{X}$ が式 (1) のように特異値分解されたとすると、$\boldsymbol{X}$ の擬似逆行列 $\boldsymbol{X}^+$ は

$$\boldsymbol{X}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}, \tag{5}$$

と表される。

2018 年に Tang によって、次元 n の行列の低ランク近似を古典計算機上で $O(\mathrm{poly}(\log n))$ で計算できるアルゴリズム (修正 FKV アルゴリズム)[1] が発表された。そのアルゴリズムを以下に示す。このアルゴリズムの説明では、行列 $\boldsymbol{X}$ について、i 行 j 列目の成分を $\boldsymbol{X}(i,j)$、i 行目の行ベクトルを $\boldsymbol{X}(i,:)$、j 列目の列ベクトルを $\boldsymbol{X}(:,j)$ と表記する。

1. パラメータ P, K を与える。P は計算精度と計算時間を制御するパラメータであり、計算の高速化のためには $\min(m,n) > P$ であることが求められる。K は求める最小特異値のインデックスである。

2. $\boldsymbol{X}$ の行添字 i を確率 f_i で得るサンプリングを行う。修正 FKV アルゴリズムでは、式 (6) で表されるような 2-ノルムで重み付けされた確率でサンプリングされている。

$$f_i = \frac{\|\boldsymbol{X}(i,:)\|^2}{\|\boldsymbol{X}\|_F^2}. \tag{6}$$

ただし、$\|\boldsymbol{X}\|_F$ は行列 $\boldsymbol{X}$ のフロベニウスノルムである。このサンプリングを P 回行い、得られた結果を $\{i_1, i_2, \dots, i_P\}$ とする。この確率でのサンプリングを高速で行うためには、行列のデータ構造がセグメント木構造になっている必要がある。データ構造がセグメント木構造になっている場合、一回のサンプリングの計算量は $O(\log n)$ である。

3. $\boldsymbol{X}$ の列添字 j を確率 g_j で得るサンプリングを行う。修正 FKV アルゴリズムでは、式 (7) で表される確率でサンプリングを行う。

$$g_j = \frac{1}{P}\sum_{p=1}^{P} \frac{\boldsymbol{X}(i_p,j)^2}{\|\boldsymbol{X}(i_p,:)\|^2}. \tag{7}$$

このサンプリングを P 回行い、得られた結果を $\{j_1, j_2, \dots, j_P\}$ とする。

4. 正規化された部分行列 $\boldsymbol{W} \in \mathbb{R}^{P\times P}$ を以下のように定義する。

$$\boldsymbol{W}(p,q) = \frac{\boldsymbol{X}(i_p,j_q)}{P\sqrt{f_{i_p}g_{j_q}}}. \tag{8}$$

この計算には $O(P^2)$ の時間がかかる。

5. $\boldsymbol{W}$ に特異値分解を行い、左特異ベクトル $\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_P$、右特異ベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_P$ と特異値 $\sigma_1, \sigma_2, \dots, \sigma_P$ を得る。その中からさらに特異値の大きい順に K 個の特異値と特異ベクトルをもちいて、行列 $\boldsymbol{U}' \in \mathbb{R}^{P\times K}, \Sigma_K \in \mathbb{R}^{K\times K}, \boldsymbol{V}' \in \mathbb{R}^{P\times K}$ を以下のように定める。この計算には $O(P^3)$ の時間がかかる。

$$\boldsymbol{U}' = (\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_K) \tag{9}$$

$$\boldsymbol{V}' = (\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_K) \tag{10}$$

$$\boldsymbol{\Sigma}_K = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_K). \tag{11}$$

6. 行列 $\boldsymbol{S} \in \mathbb{R}^{P\times n}$ を以下のように定義し、右特異ベクトル $\boldsymbol{V}_K$ と左特異ベクトル $\boldsymbol{U}_K$ を復元する。

$$\boldsymbol{S}(p,:) = \frac{\boldsymbol{X}(i_p,:)}{\sqrt{Pf_p}}. \tag{12}$$

右特異ベクトルと左特異ベクトルは以下のように復元できる。

$$\boldsymbol{V}_K = \boldsymbol{S}^{\mathrm{T}}\boldsymbol{U}'{\boldsymbol{\Sigma}_K}^{-1} \tag{13}$$

$$\boldsymbol{U}_K = \boldsymbol{X}\boldsymbol{V}_K\boldsymbol{\Sigma_K^{-1}}. \tag{14}$$

$\boldsymbol{V}_K$ の復元は $O(nK^2)$、$\boldsymbol{U}_K$ の復元には $O(nmK)$ の時間がかかる。

2.2 エクストリーム機械学習

D 個の組からなるデータセット $\{x_i, y_i\}_{i=1}^D$ が与えられたとする。x_i が画像などのデータ、y_i がラベル (画像に何が映っているかの答えなど) である。このとき、線形基底関数モデルを考える。これは、複数の非線形関数の線形結合により未知の関数 $f(x)$ を近似するものである。まず、基底関数と呼ばれる $\phi_1(x), \dots, \phi_M(x)$ の M 個の非線形関数を用意する。基底関数を重み w_j によって線形結合し、未知の関数 $f(x)$ を表現することを試みる。

$$y = f(x) = \sum_{j=1}^{M} w_j\phi_j(x) + w_0. \tag{15}$$

w_0 はバイアスである。ここで、$\phi_0(x), \ldots, \phi_M(x)$ による $f(x)$ の表現を、データ x を $M+1$ 次元の特徴量空間へと移し、その空間上で線形回帰をしていると考えても良い。エクストリーム機械学習では、乱数により生成したベクトル $\boldsymbol{a}_j, \boldsymbol{b}_j (j = 1, 2, 3, \ldots, M)$ と ReLu などの非線形関数 $g(x)$ をもちいて、基底関数を $\phi_j(\boldsymbol{x}) = g(\boldsymbol{a}_j \cdot \boldsymbol{x} + b_j)$ と表現し、$\boldsymbol{w}$ は擬似逆行列をもちいて決定する [5]。エクストリーム機械学習のアルゴリズムを以下に示す。

0. 学習にもちいる入力データを $\boldsymbol{x}_i \in \mathbb{R}^n$、ラベル $\boldsymbol{y}_i \in \mathbb{R}^m$ を全て集めた行列を $\boldsymbol{Y} \in \mathbb{R}^{D \times m}$ とする。

1. 乱数によって行列 $\boldsymbol{A} \in \mathbb{R}^{n \times M}$ を生成する。$\boldsymbol{a}_i \in \mathbb{R}^n$ を $\boldsymbol{A}$ の第 i 列目とする。

2. 乱数によってバイアス $\boldsymbol{b} \in \mathbb{R}^M$ を生成する。

3. 非線形関数 $g(\boldsymbol{x})$ をもちいて $h_{ij} = g(\boldsymbol{a}_i \cdot \boldsymbol{x}_j + b_i)$ $(i = 1, 2, 3, \ldots, M,\ j = 1, 2, 3, \ldots, D)$ を計算する。$\boldsymbol{H} = \{h_{ij}\} (\in \mathbb{R}^{D \times M})$ が隠れ層となる。

4. $\boldsymbol{H}$ の擬似逆行列 $\boldsymbol{H}^+$ を計算し、$\boldsymbol{w} = \boldsymbol{H}^+ \boldsymbol{Y}$ により重みを決定する。

5. データ $\boldsymbol{x}$ に対して $\boldsymbol{h}' = g(\boldsymbol{x}\boldsymbol{A} + \boldsymbol{b})$ を計算し、推論解 $\tilde{\boldsymbol{y}} = \boldsymbol{h}'\boldsymbol{w}$ を計算する。

本研究では、隠れ層の行列 $\boldsymbol{H}$ に低ランク性を持たせるために、ラベル $\boldsymbol{y}$ と推論解 $\tilde{\boldsymbol{y}}$ の差が小さくなるように、行列 $\boldsymbol{A}$ とバイアス $\boldsymbol{b}$ を最適化する。最適化することによって、隠れ層の行列 $\boldsymbol{H}$ の特異値が大きい値を持つものと小さい値を持つものにわかれ、隠れ層の行列 $\boldsymbol{H}$ が低ランクになることが期待される。アルゴリズムを以下に示す。

1. 上記のエクストリーム機械学習の手順 **1** から **4** を実行する。

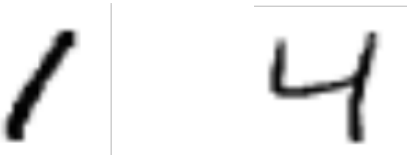
2. 教師データを入力したときの推論解 $\tilde{\boldsymbol{y}}$ と、ラベル $\boldsymbol{y}$ との差 E を式 (16) によって計算する。

$$E = \sum_{i=1}^{D} (y_i - \tilde{y}_i)^2 \tag{16}$$

3. E が最小となるように $\boldsymbol{A}$ と $\boldsymbol{b}$ を最適化する。

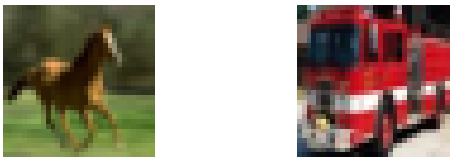
4. 再度、疑似逆行列 $\boldsymbol{H}^+$ を計算する。

行列 $\boldsymbol{H}$ の擬似逆行列を求めるには、特異値分解を行う必要があるが、これは修正 FKV アルゴリズムで高速に計算することができる。しかし、修正 FKV アルゴリズムの誤差がエクストリーム機械学習で許容できるものかはわかっていない。また、このアルゴリズムは行列 $\boldsymbol{H}$ が低ランクである必要があるが、実データをもちいた学習においてどのようなパラメータ $\boldsymbol{A}, \boldsymbol{b}$ を与えれば $\boldsymbol{H}$ が十分に低ランクになるかどうかはよくわかっていない。そして、低ランク近似を行う際にサンプリングをして次元圧縮を行うことで学習にどのような影響が出るのか、修正 FKV アルゴリズムは実計算時間でも従来の特異値分解アルゴリズムに対して優位性があるのかを調べる必要がある。一方で、修正 FKV アルゴリズムは小さい値の特異値を間引いているが、これが過学習を回避するための正則化になっているのではないかという期待がある。また、もしエクストリーム機械学習における行列 $\boldsymbol{H}$ が十分に低ランクでない場合、修正 FKV アルゴリズムの手順 **2** と **3** において、一様な確率でサンプリングを行ってもよい近似になるのではないかという疑問がある。一様な確率でのサンプリングでよいのであれば、$\boldsymbol{H}$ のセグメント木構造を用意する必要がなくなり、修正 FKV アルゴリズムの工夫が有効とはいえなくなるのではないかと考えられる。



(a) 1　　(b) 4

図 1　MNIST の画像データの例 [6]



(a) 馬の画像　　(b) トラックの画像

図 2　CIFAR-10 の画像データ例 [7], [8]

3. 数値実験

修正 FKV アルゴリズムをエクストリーム機械学習における擬似逆行列計算にもちいた場合の優位性を数値実験により確かめる。そのために、擬似逆行列の計算に修正 FKV アルゴリズムをもちい、従来の擬似逆行列を求める手法をもちいた場合と比較した。また、パラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化し、隠れ層の行列 $\boldsymbol{H}$ の特異値がどのように変化するのか、ノルムによるサンプリングが学習の精度の面で優位に働くかを調べた。使用した計算機の構成は、CPU が Intel Xeon E5-2687W が 2 つ、RAM 容量が 125GB、OS は Cent OS 7 である。数値実験にもちいたプログラムは Python で実装した。データセットには、図 1 のような手書き数字のデータセットである MNIST[6] と、図 2 のような動物や乗り物のカラー画像のデータセットである CIFAR-10[7] をもちいた。

3.1 数値実験手法

学習を行いやすくするため、scikit-learn[9] に含まれている Onehotencoder クラスをもちいてラベル y をワンホット表示にした。また、画像データ $\boldsymbol{x}_i$ を scikit-learn に含まれている MinMaxScaler クラスをもちいて、最小値 0、最大値 1 になるように正規化した。まず、教師データをエクストリーム機械学習に入力し、非線形関数に ReLU、パラメータ $\boldsymbol{A}, \boldsymbol{b}$ には、$[0, 1]$ を一様な確率でとる乱数によって

決定したものと、最適化したものの両方をもちいて学習を行った。パラメータ $\boldsymbol{A}, \boldsymbol{b}$ の最適化には、機械学習ライブラリ Pytorch に実装されている Adam[10] をもちい、学習率は 0.1、反復回数は、MNIST データセットをもちいる場合は 10^3 回、CIFAR-10 データセットをもちいる場合は 2×10^4 回とした。次に、教師データと学習で得られたパラメータをもちいて推論解を計算した。推論解は最も値の大きい要素の番号を答えとし、推論解をラベルと比較した。そして、テストデータ数に対して正解した数の割合を正解率とし、学習精度の指標とした。また、擬似逆行列の計算にかかった時間を計測し、どの程度計算が高速化できているのかの指標とした。従来の手法として、Python の数値計算ライブラリである NumPy[11] に含まれている lstsq 関数をもちいた。

数値実験では、まず lstsq 関数の機能をもちい、エクストリーム機械学習における隠れ層 $\boldsymbol{H}$ の特異値の分布が、乱数で初期化したままの場合と、パラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化した場合でどのように変化するのかを調べた。厳密に擬似逆行列を計算した場合、ノード数 M を変化させると正解率はどのように変化するのか、低ランク近似をすると正解率はどのくらい下がるのかを調べた。そして、従来の手法では計算時間はどのくらいかかるのかを調べた。次に、修正 FKV アルゴリズムをもちいて擬似逆行列計算を行い、サンプル数 P と採用する特異値の数である K を変化させたときの正解率と計算時間を調べ、従来の手法と同じランク近似で同程度の正解率を達成するためにはどの程度にサンプル数を設定する必要があるのか、同じサンプル数で特異値の数を変化させると正解率はどのように変化するのかを調べた。また、ノルムによるサンプリングが有効であるほど $\boldsymbol{H}$ が低ランクであるかを確かめるため、一様な確率でのサンプリングで修正 FKV アルゴリズムを実行し、正解率の比較を行った。パラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化した場合や別の乱数の分布で設定した場合、$\boldsymbol{H}$ のランクがどのように変化し、正解率にどのような影響を与えるのかを調べた。

3.2 特異値の分布

まず、修正 FKV アルゴリズムが行列の特異値分解によい近似を与えるための条件である、行列の低ランク性を確かめるため、lstsq 関数から出力される特異値をデータセット MNIST と CIFAR-10 のそれぞれをもちいた場合の隠れ層 $\boldsymbol{H}$ とそれぞれパラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化した場合について調べた。エクストリーム機械学習における隠れ層の特異値の分布を図 3 に示す。全ての特異値を最大特異値で正規化してある。パラメータを最適化しなかった場合、ノード数に関係なくデータセットが MNIST の場合はインデックスが 700 程度、CIFAR-10 の場合は 3000 程度から特異値が急激に小さくなっていることがわかる。また、それ以外の場合は特異値のインデックス i に対して多項式的に特異値が小さくなっていることがわかる。10 番目の特異値で既に最大特異値の 1/100 以下となっているため、低ランク近似が良い近似になると期待される。パラメータを最適化した場合、インデックスが小さい特異値は最適化前より大きくなり、そこから最適化前より急激に小さくなっている。これは、上位のいくつかの特異値が隠れ層 $\boldsymbol{H}$ の主要な成分となっており、低ランク近似がさらに有効であることが期待される。

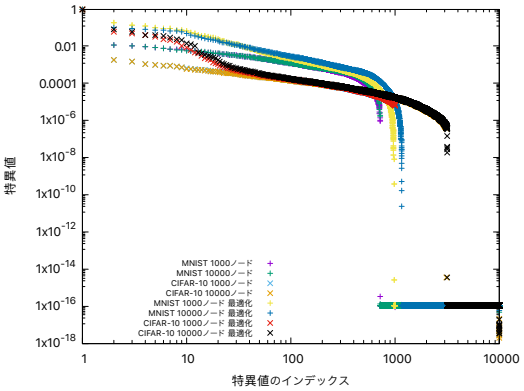


図 3 エクストリーム機械学習における隠れ層の特異値の分布

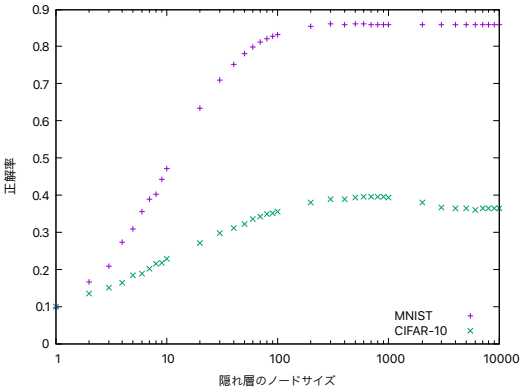


図 4 隠れ層ノード数と正解率の関係

3.3 ノード数の影響

隠れ層のノード数 M を変化させたときの正解率と計算時間を調べた。まず、ノード数を変化させたときの正解率の変化を図 4 に示す。データセットが MNIST 場合、200 ノードで 85%程度の正解率が得られており、学習としては十分であるといえる。CIFAR-10 はデータの次元が大きいため、10^4 ノードの場合でも満足な精度を得られていない。ランダムに推論を行った場合、正解率はラベルの種類の数 m に対して $1/m$ となる、したがって、この場合はノード数を 10^4 程度確保すれば、正解率が 40%程度になるため、ある程度学習は行えていると考えられる。

次に、ノード数を変化させたときの計算時間の変化を図 5 に示す。ノード数が 20 のところからノード数 M に対して M^α (α は定数) の傾きで計算時間が増加しており、ノー

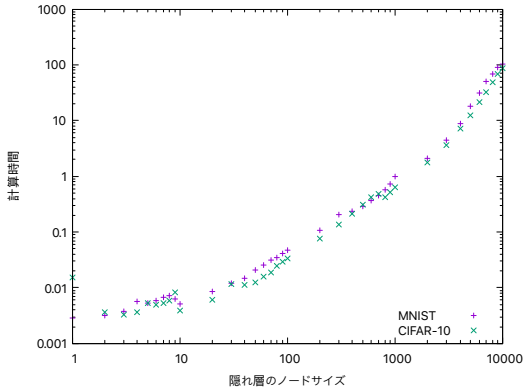


図 5 隠れ層ノード数と計算時間の関係

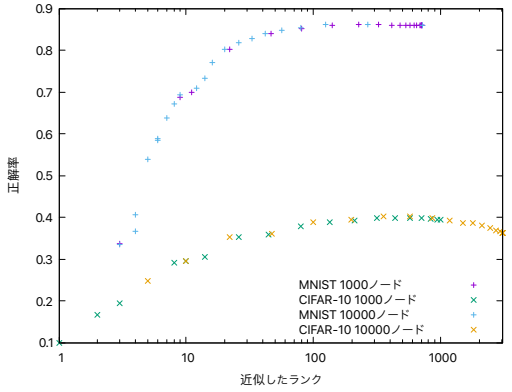


図 6 採用した特異値の数と正解率の関係

ド数に対して計算時間が急激に増加していることがわかる。

3.4 打ち切り特異値分解の影響

NumPy の lstsq 関数を用い、特異値を途中で打ち切って擬似逆行列を計算したときの学習への影響を 10^3 ノードと 10^4 ノードの場合で調べた。採用する特異値の個数を変えたときの正解率の変化を図 6 に示す。ノード数が違っていても近似したランクの数が同じであれば正解率がほぼ同じになることがわかった。これは、10^3 個程度の基底を用意すれば、低ランク近似には十分な種類の基底が用意されていることを表している。また、ノード数を変化させた場合と異なり、20 程度まで特異値を使わなかった場合でも、正解率は大きく落ちなかった。例えば、データセットに MNIST をもちい、隠れ層のノード数を 10 にして学習を行った場合、正解率は図 4 より 0.4 程度であるが、ノード数を 10^3 にして 10 個の特異値をもちいた場合、正解率は図 6 より 0.7 程度である。これは、同じ数の基底 (隠れ層) で学習する場合でも、単純にその数の基底を用意するのではなく、最初に多く基底を用意しておいて特異値の大きなものを厳選してもちいた方が精度の高い学習が行えることを示している。つまり、高次元の特徴量空間を用意し、それをサンプリングして次元を落とした方が精度が高いということであり、修正 FKV アルゴリズムでの高速化が期待

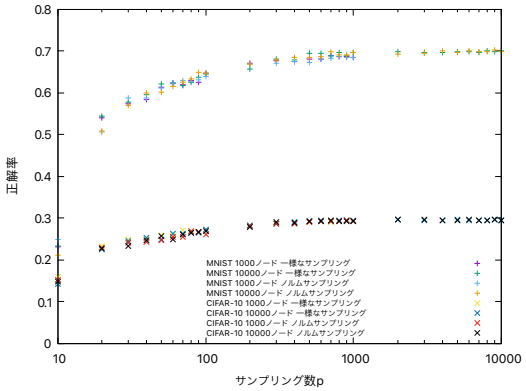


図 7 特異値を 10 個採用で固定し、サンプル数を変化させたときの正解率

できる。

3.5 修正 FKV アルゴリズム

修正 FKV アルゴリズムを実装し、エクストリーム機械学習に組み込んで実際に学習を行わせた。また、修正 FKV アルゴリズムがどの程度計算が速いのかを調べるために、計算時間の比較を行った。まず、データセットとして MNIST をもちい、採用する特異値を 10 個で固定してサンプル数を変化させた。サンプル数と正解率の関係を図 7 に示す。サンプリング数を大きくすると正解率が向上していることがわかる。これは、サンプリングをもちいて次元圧縮を行っているため、サンプル数を小さくしすぎると厳密に上位 10 個の特異値を持ってくることができないからである。しかし、サンプリングで 10^2 次元の部分行列を生成すれば、厳密に上位 10 個の特異値を採用したときと同程度の精度で学習を行うには十分であることがわかる。また、隠れ層 $\boldsymbol{H}$ のノルムをもちいてサンプリングを行った場合と、一様な確率でサンプリングを行った場合で正解率に違いは見られなかった。これは、図 3 のような特異値の分布では、ノルムのサンプリングが有効ではないことを表している。次に、サンプリング数を固定し、部分行列を特異値分解したあとに採用する特異値の数を変化させた。採用した特異値の数と正解率の関係を図 8 に示す。特異値の数を多くすると正解率が良くなる傾向にあるが、サンプル数 P に対して $0.5P$ のあたりで正解率が減少している。これは、ノルムによるサンプリングで生成した部分行列を特異値分解したとき、特異値を全て含めて特異ベクトルの復元をしてしまうと、非常に小さい特異値によって $\boldsymbol{H}_K$ の近似精度が悪くなってしまうことを表している。よって、特異値を途中で打ち切って小さい特異値の影響を排除するということが重要であることがわかる。また、サンプリング数を変化させた場合と同様に、ノルムをもちいてサンプリングした場合と、一様な確率でサンプリングした場合とで正解率には違いが見られなかった。

サンプル数や採用する特異値の数と計算時間の関係を図

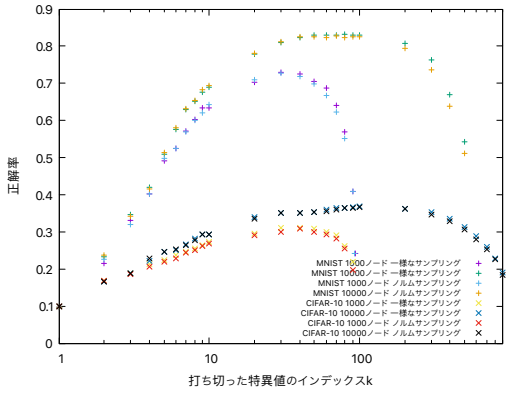


図 8 サンプル数を 10^3 ノードでは 10^2 回、10^4 ノードでは 10^3 回で固定し、採用する特異値の数を変化させたときの正解率

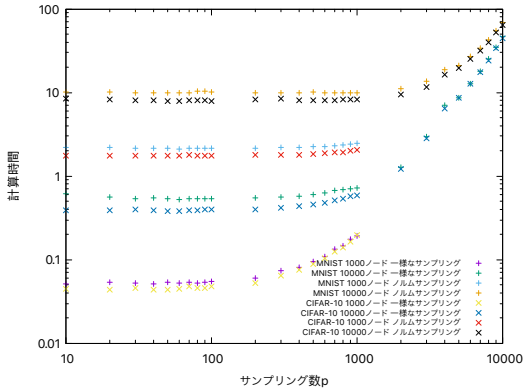


図 9 特異値を 10 個採用で固定し、サンプル数を変化させたときの計算時間

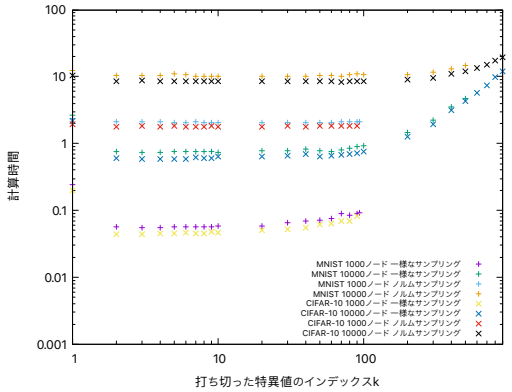


図 10 サンプル数を固定し、採用する特異値の数を変化させたときの計算時間

9 と図 10 に示す。 ノルムでサンプリングを行う場合、まずセグメント木構造を準備する必要があるため、全ての場合において、一様な確率でサンプリングをする場合よりも時間がかかっていることがわかる。lstsq と修正 FKV アルゴリズムの正解率と計算時間の比較を表 1 に示す (p はサンプル数をあらわす)。ノルムでのサンプリングを行った場合、10^3 ノードと 10^4 ノードのどちらの場合でも、セグメント木構造の準備にかかる時間を除くと、従来の特異値分解手法 (lstsq) より高速に計算ができていることがわかる。

表 1 10^3 ノード、特異値 10 個で近似したときの正解率と計算時間の比較 (データセットは MNIST、ノルムサンプリングのみセグメント木構造の準備にかかった時間をわけて示している)

ノード数	手法	正解率	計算時間 [s]
10^3	lstsq	0.687	1.00
10^3	$p = 10^2$、ノルムサンプリング	0.640	1.96+0.09
10^3	$p = 10^2$、一様確率サンプリング	0.646	0.0555
10^4	lstsq	0.693	105
10^4	$p = 10^2$、ノルムサンプリング	0.648	9.53+0.67
10^4	$p = 10^2$、一様確率サンプリング	0.644	0.535

表 2 パラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化し、特異値 10 個で近似したときの正解率の比較 (データセットには MNIST をもちいている)

ノード数	手法	正解率
10^3	$p = 10^2$、ノルムサンプリング	0.704 ± 0.0707
10^3	$p = 10^2$、一様確率サンプリング	0.584 ± 0.0771
10^4	$p = 10^3$、ノルムサンプリング	0.844 ± 0.0136
10^4	$p = 10^3$、一様確率サンプリング	0.745 ± 0.0406

表 3 パラメータ $\boldsymbol{A}, \boldsymbol{b}$ を最適化し、特異値 10 個で近似したときの正解率の比較 (データセットには CIFAR-10 をもちいている)

ノード数	手法	正解率
10^3	$p = 10^2$、ノルムサンプリング	0.224 ± 0.0175
10^3	$p = 10^2$、一様確率サンプリング	0.236 ± 0.0196
10^4	$p = 10^3$、ノルムサンプリング	0.309 ± 0.00957
10^4	$p = 10^3$、一様確率サンプリング	0.233 ± 0.0177

一方、セグメント木構造の準備にかかる時間も含めると、10^3 ノードの場合は従来手法よりも計算が遅いという結果になったが、修正 FKV アルゴリズムのプログラムが十分に最適化されていない状態で、この程度の差に収まっているのは十分に有望な結果であると考えられる。一様な確率でサンプリングを行った場合、セグメント木構造の準備をする必要がなくなるため、従来手法よりも高速であり、正解率もノルムでのサンプリングを行った場合とほぼ変わらなため、パラメータを乱数で設定しただけのエクストリーム機械学習では、修正 FKV アルゴリズムの工夫は有効ではないといえる。データセットに CIFAR-10 をもちいた場合でも、同様の結果が得られた。

最適化したパラメータ $\boldsymbol{A}, \boldsymbol{b}$ をもちいて、ノルムでサンプリングした場合と、一様な確率でサンプリングした場合の正解率の比較を行った。結果を表 2 に示す (p はサンプル数をあらわす)。乱数で初期化しただけのパラメータをもちいた場合と違い、正解率に有意な差が生まれた。これより、図 3 の、MNIST データセットでパラメータを最適化した場合のような特異値を持つ行列を用意することができれば、ノルムでのサンプリングが有効になることがわかった。データセットを CIFAR-10 に変更し、同様の比較を行った。結果を表 3 に示す (p はサンプル数をあらわす)。10^4 ノードでは、ノルムでのサンプリングと、一様なサンプリングの両者の結果に有意な差が生まれたが、10^3 ノードでは差が生まれなかった。原因としては、パラメータの

最適化が不十分であり、図 3 の、10^3 ノード、CIFAR-10 データセットで、パラメータを最適化した場合のような特異値をもつ行列は、ノルムでのサンプリングが有効になるほど低ランクではなかったためであると考えられる。

4. 結論

本研究では、量子特異値分解を脱量子化した、修正 FKV アルゴリズムのエクストリーム機械学習への応用を提案した。修正 FKV アルゴリズムがよい近似を与えるための条件である、行列の低ランク性が実データをもちいたエクストリーム機械学習において当てはまるのかということを調べた。また、実際に修正 FKV アルゴリズムをもちいて学習を行った場合、精度と実計算時間はどのようになるのか、ということを調査した。

エクストリーム機械学習において、低ランク近似を行った場合、学習の精度はデータセットとして MNIST、CIFAR-10 のどちらをもちいた場合でも、ノード数を少なくした場合のように急激に低下しなかった。これより、エクストリーム機械学習において低ランク近似が有効であることがわかった。また、修正 FKV アルゴリズムはサンプリングにより大きな特異値を含む行や列を確率的に抽出するため、厳密に行列の特異値を大きい順に k 個求めて低ランク近似をするということはできないが、それでも、10^2 サンプル程度のサンプリングを行えばランク 10 で近似してエクストリーム機械学習を行う上では問題ない精度の低ランク近似ができていた。これより、修正 FKV アルゴリズムの、サンプリングによって行と列を乱択し、次元圧縮をして特異値分解するというアルゴリズムがエクストリーム機械学習の擬似逆行列計算において有効であることがわかった。しかし、パラメータを乱数で初期化したエクストリーム機械学習においては、ノルムによって大きな特異値を抽出する必要がなく、ただ一様な確率で行と列をサンプリングするだけでよいことがわかった。エクストリーム機械学習でノルムによるサンプリングに優位性を持たせるには、乱数で初期化したパラメータを最適化する必要がある。実計算時間の面では、本研究で実装した修正 FKV アルゴリズムは最適化されていないものであるが、セグメント木構造が用意されているとすると、NumPy に実装されている関数よりも高速であったことは大きな成果であるといえる。

本研究では、特異ベクトルを実際に復元して擬似逆行列を計算し、隠れ層と出力層の結合重みを決定しているが、これでは元の行列の次元をもとに戻してしまっているため、推論解を得るときに行列の次元に応じた計算時間が必要となっている。この問題については、既に線形回帰の場合についてさらに計算を高速にする手法が提唱されている。[12] エクストリーム機械学習においても、隠れ層 $\boldsymbol{H}$ と出力層 $\boldsymbol{Y}$ の関係 $\boldsymbol{Hw} = \boldsymbol{Y}$ を線形回帰問題と考え、文献 [12] と同じように確率的勾配降下法をもちいて重み $\boldsymbol{w}$ を求めることで、さらに高速に計算できると考えられる。また、パラメータを乱数で設定するエクストリーム機械学習においては、セグメント木構造を用いたノルムによるサンプリングは不要であることがわかったが、最初に修正 FKV アルゴリズムを提唱した、推薦システム [1] や、時系列データでの学習 [13] において、これらの工夫が有効かどうかわからないため、それらについても調べる必要がある。

参考文献

[1] Tang, E.: A quantum-inspired classical algorithm for recommendation systems, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 217–228 (2019).
[2] Kerenidis, I. and Prakash, A.: Quantum Recommendation Systems, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, Vol. 67, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, p. 49 (2017).
[3] Frieze, A., Kannan, R. and Vempala, S.: Fast Monte-Carlo algorithms for finding low-rank approximations, *Journal of the ACM (JACM)*, Vol. 51, No. 6, pp. 1025–1041 (2004).
[4] Gilyén, A., Lloyd, S. and Tang, E.: Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension, *arXiv preprint arXiv:1811.04909* (2018).
[5] Huang, G.-B., Zhu, Q.-Y. and Siew, C.-K.: Extreme learning machine: a new learning scheme of feedforward neural networks, *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, Vol. 2, IEEE, pp. 985–990 (2004).
[6] Deng, L.: The MNIST Database of Handwritten Digit Images for Machine Learning Research (2012).
[7] Krizhevsky, A., Hinton, G. et al.: Learning multiple layers of features from tiny images (2009).
[8] Krizhevsky, A.: CIFAR-10 and CIFAR-100 datasets (2009). `https://www.cs.toronto.edu/~kriz/cifar.html`.
[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al.: Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830 (2011).
[10] Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).
[11] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J. et al.: Array programming with NumPy, *Nature*, Vol. 585, No. 7825, pp. 357–362 (2020).
[12] Gilyén, A., Song, Z. and Tang, E.: An improved quantum-inspired algorithm for linear regression, *arXiv preprint arXiv:2009.07268* (2020).
[13] Jaeger, H. and Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *science* (2004).