

2025-09-30



超初心者のためのテンソルネットワーク講座



<https://github.com/todo-group/tn-basics>

超初心者のためのテンソルネットワーク講座

- ・テンソルネットワークの基礎
 - ・次元の呪い
 - ・特異値分解と低ランク近似
 - ・テンソルとテンソルネットワーク
- ・テンソルネットワークの利用
 - ・テンソルネットワーク表現
 - ・量子状態のユニタリ時間発展
 - ・階層構造の圧縮表現

参考文献

・日本語の文献

- ・「テンソルネットワークの基礎と応用: 統計物理・量子情報・機械学習」(SGCライブラリ 169), 西野友年 サイエンス社 (2021)
- ・数理科学 2022年2月号特集「テンソルネットワークの進展」
- ・「テンソルネットワーク形式の進展と応用」西野友年, 大久保毅 日本物理学会誌 2017年10号
- ・「テンソルネットワークによる情報圧縮とフラストレート磁性体への応用」大久保毅、物性研究 2018年

・英語の文献

- ・R. Orús, “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”, Annals. of Physics 349, 117 (2014). (レビュー)
- ・Tao Xiang, “Density matrix and tensor matrix renormalization”, Cambridge University Press, 2023. (教科書)
- ・tensornetwork.org の “Review Articles and Learning Resources”に載っている文献・ビデオ

次元の呪い

次元の呪い (curse of dimensionality)

- n 元超立方体(1辺の長さ 2, 体積 2^n)に対する n 次元単位球の体積の割合

$$q = \frac{\pi^{n/2}/\Gamma(\frac{n}{2} + 1)}{2^n} \sim (\pi/n)^{n/2}$$

- $n = 10$ で 0.2%, $n = 20$ で 10^{-8} , $n = 100$ で 10^{-70}
- モンテカルロ積分で球の体積を計算しようとすると, 標準偏差に対する平均値の割合は指数関数的に小さい

$$\frac{q}{\sqrt{q(1-q)}} \sim \sqrt{q}$$

- 次元が高くなるにつれて指数関数的に大きな M が必要となる
- 通常の数値積分(台形公式等)でも同様

古典統計物理

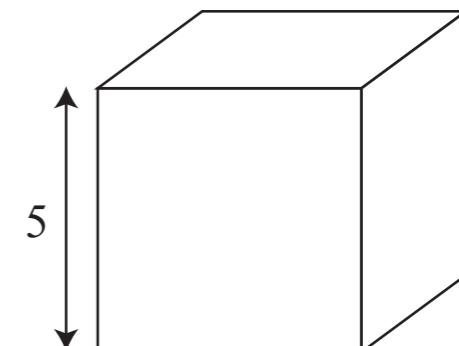
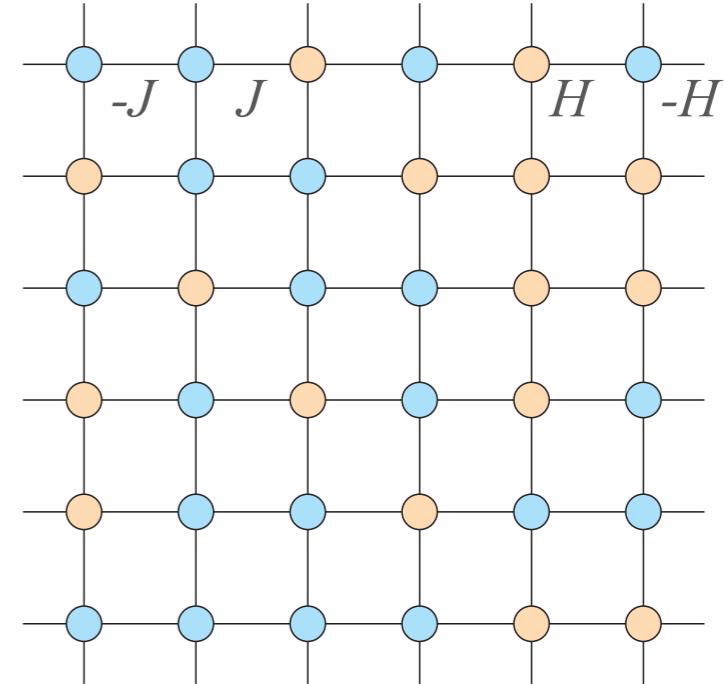
・イジング (Ising) 模型

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - H \sum_i \sigma_i$$

- 物理量の期待値の計算(例: 磁化)

$$m = \frac{\text{Tr } \sigma_i \exp[-\mathcal{H}/kT]}{\text{Tr} \exp[-\mathcal{H}/kT]}$$

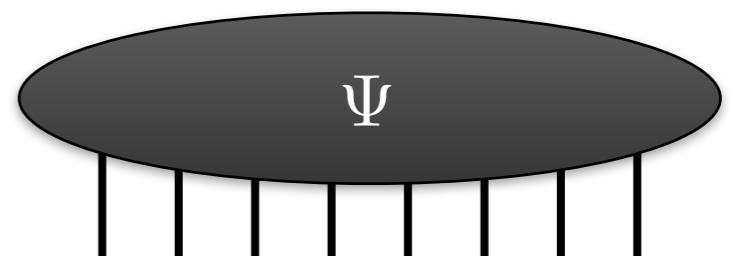
- 単純立方格子(一辺の長さ 5)
 - スピン数 $125 \rightarrow 2^{125} \approx 10^{38}$ の和
- 期待値の計算にかかる時間
 - $10^{38} \div (10^{18}/\text{sec}) \approx 3\text{兆年}$



量子多体系の波動関数

- 量子多体系状態 $\mathcal{H}|\Psi\rangle = E|\Psi\rangle$

$$|\Psi\rangle = \sum_{\{i_1, i_2, \dots, i_N\}} \Psi_{i_1 i_2 \dots i_N} |i_1 i_2 \dots i_N\rangle$$



- N スピン系
 - ハミルトニアン $\rightarrow 2^N \times 2^N$ の行列
 - 波動関数の係数 $\rightarrow 2^N$ の長さのベクトル (or N 本足のテンソル)
- N が大きくなると行列の次元、波動関数のサイズは指数関数的に増加
 - 量子多体系に共通する困難

量子回路

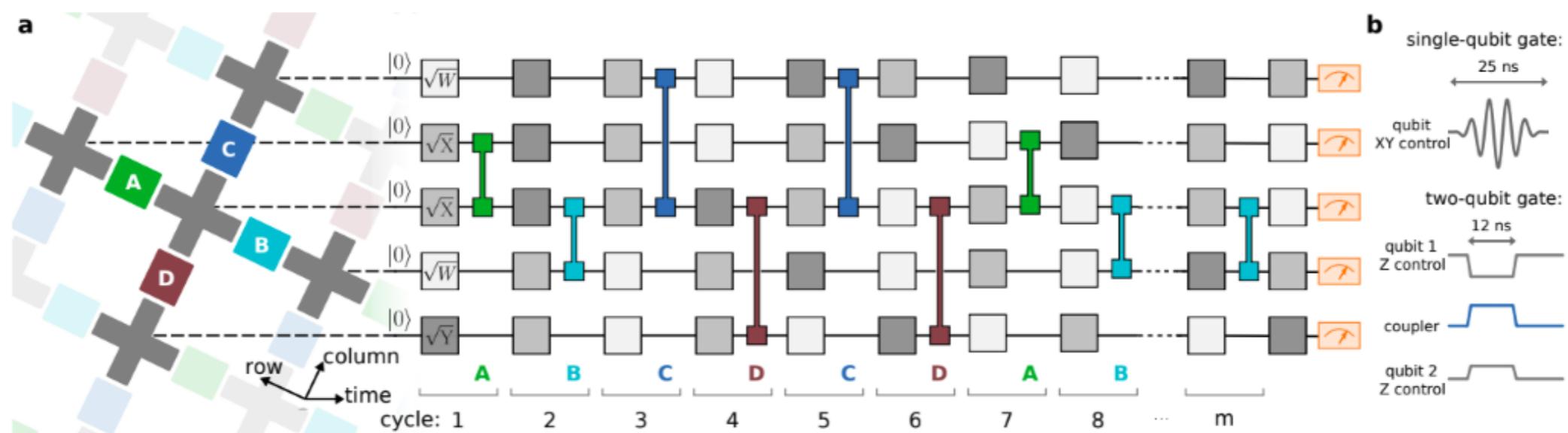
- 量子回路 = 量子ビットに演算するゲート操作の回路図

- 量子ビット(qubit)

- $|0\rangle$ と $|1\rangle$ の重ね合わせの状態をとる

- N -qubitの量子回路 $\rightarrow N$ 個の $S = 1/2$ スピン

- ヒルベルト空間の次元: 2^N



Arute, F., Arya, K., Babbush, R. et al. Nature 574, 505 (2019)

特異値分解と低ランク近似

特異値分解 (Singular Value Decomposition)

- 任意の $m \times n$ 實行列 A は

$$A = U\Lambda V^T$$

の形に (一意に) 分解できる ($k = \min(m, n)$)

- U : $(m \times k)$ 行列 (列ベクトルは互いに正規直交)
 V : $(n \times k)$ 行列 (列ベクトルは互いに正規直交)
 $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_k)$ ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k \geq 0$) 特異値
- ベクトル表示 (行列をランク 1 の行列で分解)

$$A = \sum_{i=1}^k \lambda_i u_i v_i^T$$

$k < m, n$ の時: $U^T U = V^T V = E_k$ だが $UU^T \neq E_n$, $VV^T \neq E_n$

特異値分解の例

Hands-on Lab: 1_svd

$$\begin{pmatrix} 1 & 2 & 3 \\ 6 & 4 & 5 \\ 8 & 9 & 7 \\ 10 & 11 & 12 \end{pmatrix} = \begin{pmatrix} -0.14 & -0.62 & -0.05 \\ -0.34 & 0.37 & 0.81 \\ -0.55 & 0.54 & -0.58 \\ -0.75 & -0.44 & 0.06 \end{pmatrix} \times \begin{pmatrix} 25.35 & 0 & 0 \\ 0 & 2.15 & 0 \\ 0 & 0 & 1.71 \end{pmatrix} \begin{pmatrix} -0.56 & -0.59 & -0.59 \\ 0.68 & 0.09 & -0.73 \\ 0.48 & -0.81 & 0.35 \end{pmatrix}$$

行列の低ランク近似

- ランク r ($r < k$) の行列のうち、行列 A を「最も良く近似」する \tilde{A} を選ぶ
- 「最も良く近似」 = フロベニウスノルム $\|A - \tilde{A}\|_F$ を最小化

$$\|X\|_F^2 \equiv \sum_{ij} x_{ij}^2$$

- 特異値のうち大きなものから r 個とり、残りを零とした

$$\tilde{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_r, 0, \dots, 0)$$

を使い

$$\tilde{A} = U \tilde{\Lambda} V^T$$

を作れば良い (Eckart-Young の定理)

行列の低ランク近似の例

Hands-on Lab: 1_svd

$$\begin{pmatrix} -0.14 & -0.62 & -0.05 \\ -0.34 & 0.37 & 0.81 \\ -0.55 & 0.54 & -0.58 \\ -0.75 & -0.44 & 0.06 \end{pmatrix} \begin{pmatrix} 25.35 & 0 & 0 \\ 0 & 2.15 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$\times \begin{pmatrix} -0.56 & -0.59 & -0.59 \\ 0.68 & 0.09 & -0.73 \\ 0.48 & -0.81 & 0.35 \end{pmatrix} = \begin{pmatrix} 1.04 & 1.93 & 3.03 \\ 5.33 & 5.12 & 4.52 \\ 8.47 & 8.21 & 7.34 \\ 9.95 & 11.1 & 12.0 \end{pmatrix}$$

- それなりに悪くない近似が得られる
- 誤差 (フロベニウスノルム) = 1.71 (無視した特異値の二乗和の平方根)

Eckart-Young の定理

- A を近似する行列 X (ランク $\leq r$) を考えると

$$\|A - X\|_{\text{F}}^2 = \sum_{ij} (a_{ij} - x_{ij})^2 = \text{tr}[(A - X)(A - X)^T]$$

を最小化すればよい。完全 SVD について $UU^T = E_m$, $VV^T = E_n$ より

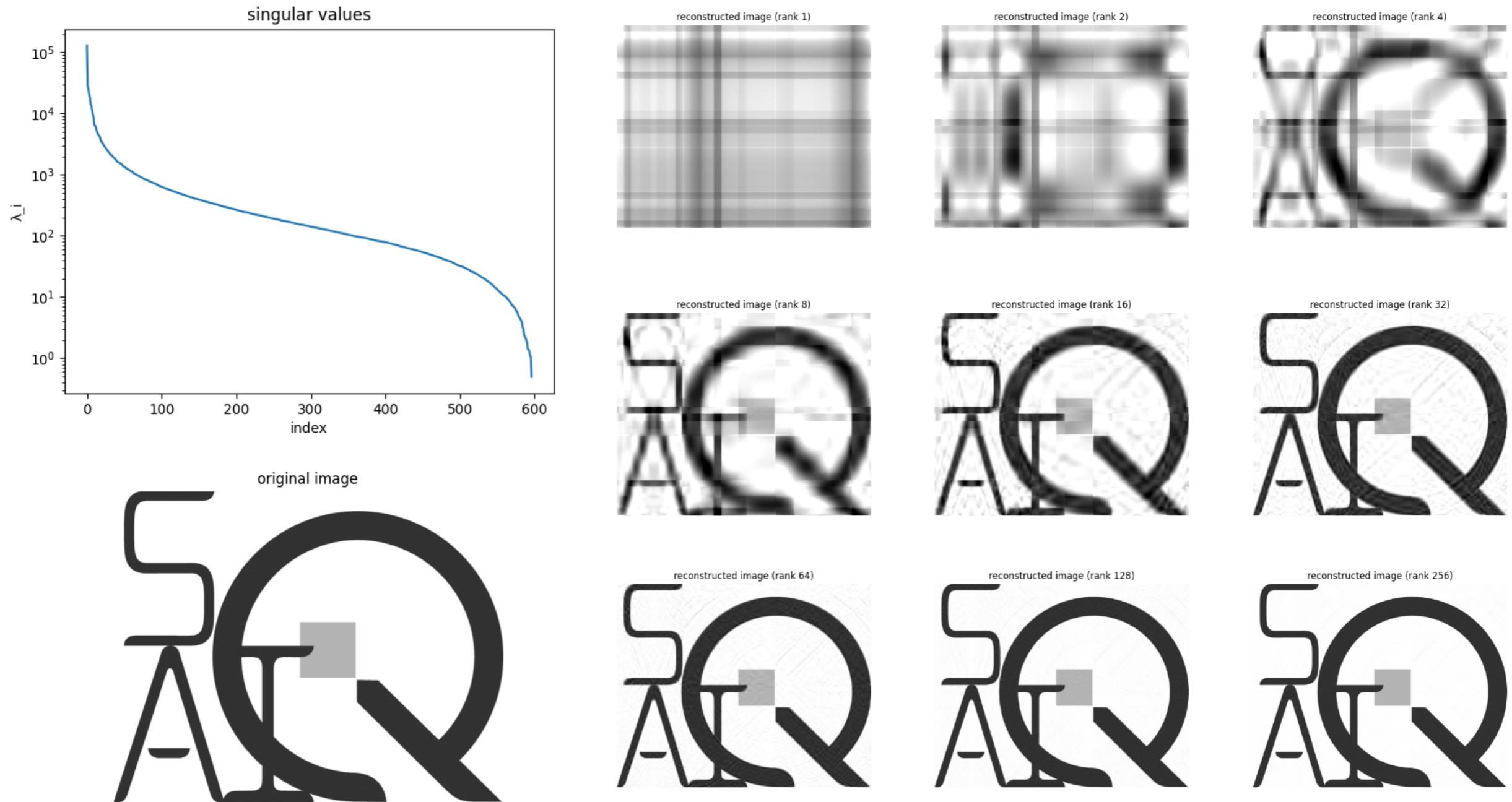
$$\begin{aligned}\|A - X\|_{\text{F}}^2 &= \text{tr}[UU^T(A - X)VV^T(A - X)^T] \\ &= \text{tr}[(\Lambda - G)(\Lambda - G)^T] \quad (G \equiv U^T X V) \\ &= \sum_i^k (\lambda_i - g_{ii})^2 + \sum_i \sum_{j \neq i} g_{ij}^2\end{aligned}$$

ランク r 以下で、 $\|A - X\|_{\text{F}}^2$ を最小化するには、 $g_{ii} = \lambda_i$ ($i = 1 \cdots r$)、それ以外は全て零とすればよい

特異値分解による画像圧縮

Hands-on Lab: 2_image-compression

- グレースケール画像(597×744)を行列とみなして低ランク近似

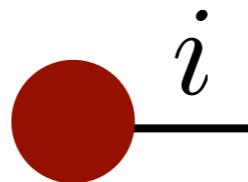


テンソルとテンソルネットワーク

テンソルとテンソルのダイアグラム表記

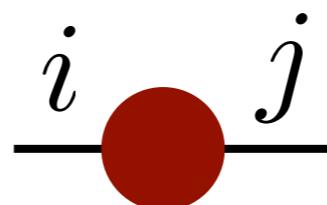
- ・テンソル: n 次元的な数字の並び (=多次元配列)
- ・ベクトル

$$\vec{v} : v_i$$



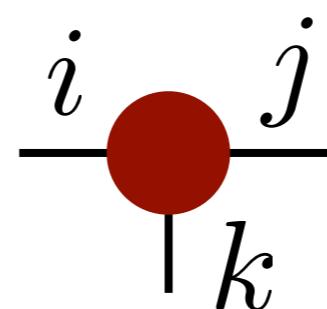
- ・行列

$$M : M_{i,j}$$



- ・3階テンソル

$$T : T_{i,j,k}$$



- ・ n 階テンソル $\rightarrow n$ 本の足

- ・スカラー(数字) \rightarrow 足なし



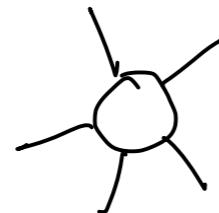
テンソルのデータ量

Hands-on Lab: 3_tensor

- 全ての足の次元を χ と仮定すると

- 2本足テンソル = $\chi \times \chi$ 行列
- 5階テンソル

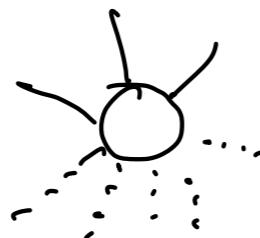
$$O(\chi^2)$$



$$T_{ijkln}$$

$$O(\chi^5)$$

- n 階テンソル



$$O(\chi^n)$$

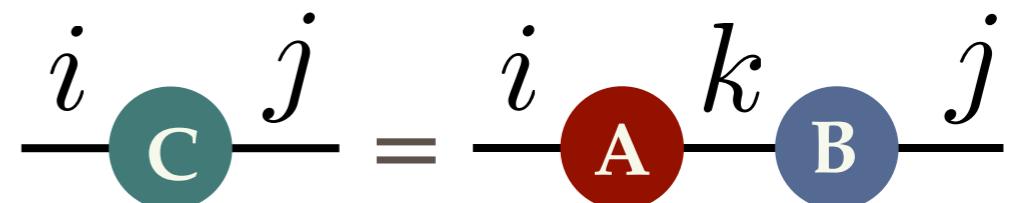
- 足の本数が増えるにつれて、データ量(=要素数)は指数関数的に増える
 - 注: 足の次元(ボンド次元と呼ぶ)は χ あるいは D あらわすことが多い

テンソルの縮約

- ・テンソルの縮約 (contraction)
 - ・共通する足について和を取る
- ・2階テンソル同士の縮約 → 結果は2階テンソル
 - ・行列行列積

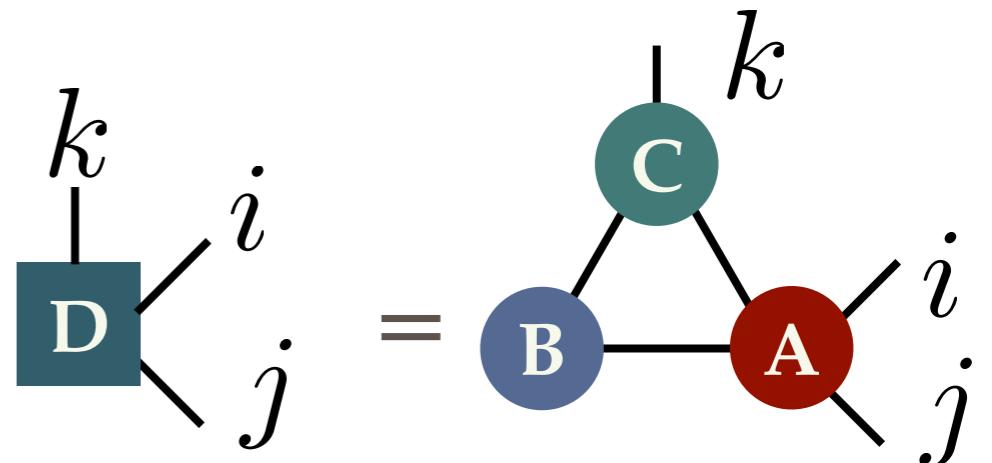
$$C_{i,j} = (AB)_{i,j} = \sum_k A_{i,k} B_{k,j}$$

$$C = AB$$



- ・一般のテンソルの縮約も同様に表現できる

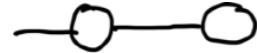
$$D_{i,j,k} = \sum_{\alpha,\beta,\gamma} A_{i,j,\alpha,\beta} B_{\beta,\gamma} C_{\gamma,k,\alpha}$$



テンソル縮約の計算量

Hands-on Lab: 4_contraction

- ・行列ベクトル積



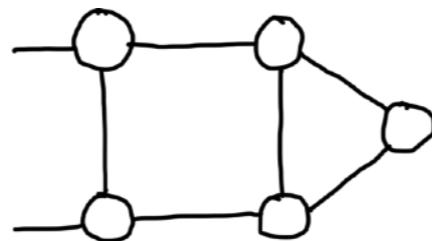
$$w_i = \sum_j T_{ij} v_j \quad O(\chi^2)$$

- ・行列行列積



$$A_{ij} = \sum_k T_{ik} R_{kj} \quad O(\chi^3)$$

- ・テンソルの縮約

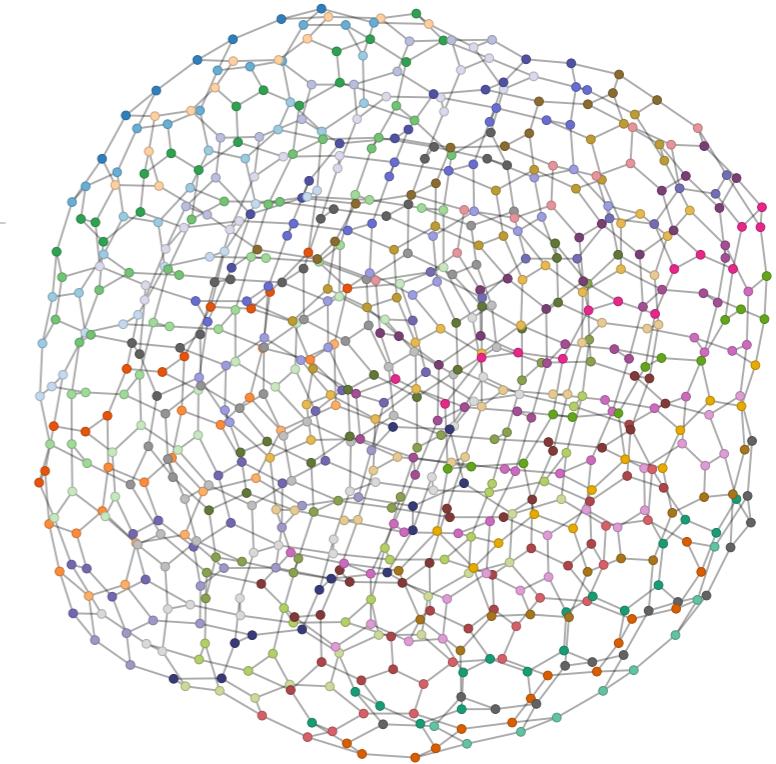


$$O(\chi^?)$$

- ・縮約の計算量は縮約の順番に強く依存する

縮約順序の最適化

- ・一般的なルール
 - ・たくさんの足をもつテンソルが途中で現れないように
- ・最適な縮約順序を見つける問題は(少なくとも)
「#P困難」であることが知られている
 - ・大きなテンソルネットワークの場合、最適解を見つけるのは事実上不可能
- ・さまざまな経験的な方法が提案されている
 - ・R. Schutski, T. Khakhulin, I. Oseledets, D. Kolmakov, Phys. Rev. A, 102, 1 (2020)
 - ・J. Gray, S. Kourtis, Quantum, 5, 1 (2021)



テンソルネットワーク表現

様々なテンソルネットワーク表現

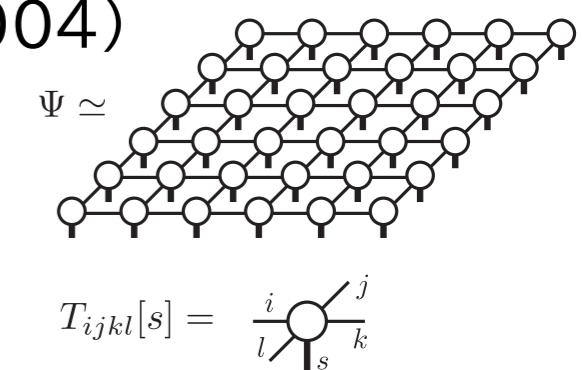
- 量子多体系の量子状態: $|\Psi\rangle = \sum \Psi_{s_1, s_2, \dots, s_N} |s_1, s_2, \dots, s_N\rangle$

• F. Verstraete, J. I. Cirac, arXiv, cond-mat/0407066 (2004)

• G. Vidal, Phys. Rev. Lett. 101, 110501 (2008)

• R. Orús, Ann. Phys. 349, 117 (2014) $Z = \sum_{\{S_i\}} e^{-\beta \mathcal{H}(\{S_i\})}$

TPS(PEPS)



- 統計力学模型の分配関数:

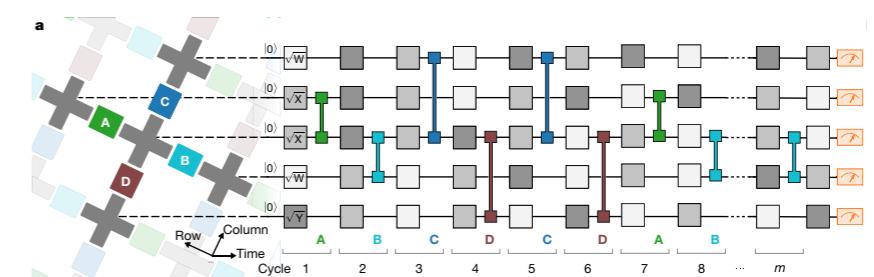
• M. Levin and C. P. Nave, Phys. Rev. Lett. 99, 120601 (2007)

• G. Evenbly and G. Vidal, Phys. Rev. Lett. 115, 180405 (2015)

- 量子回路

• F. Arute, et al., Nature 574, 505 (2019)

• Y. Lie , et al., SC '21 Proceedings, 3 (2021)

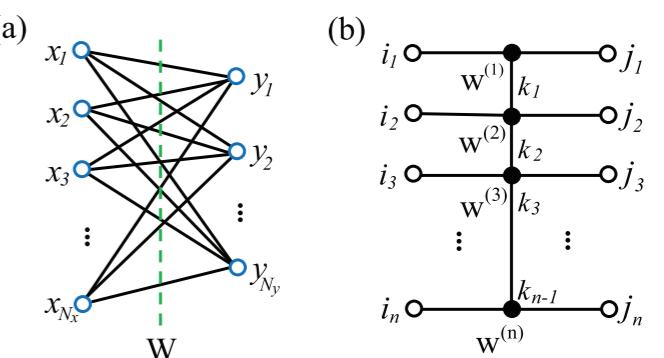
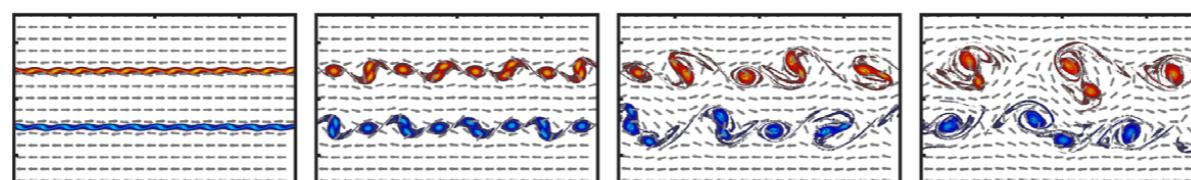


- データ科学・機械学習

• E. Stoudenmire and D. J. Schwab, NIPS 29, 4799 (2016)

• Z.-F. Gao et al., Phys. Rev. Research 2, 023300 (2020)

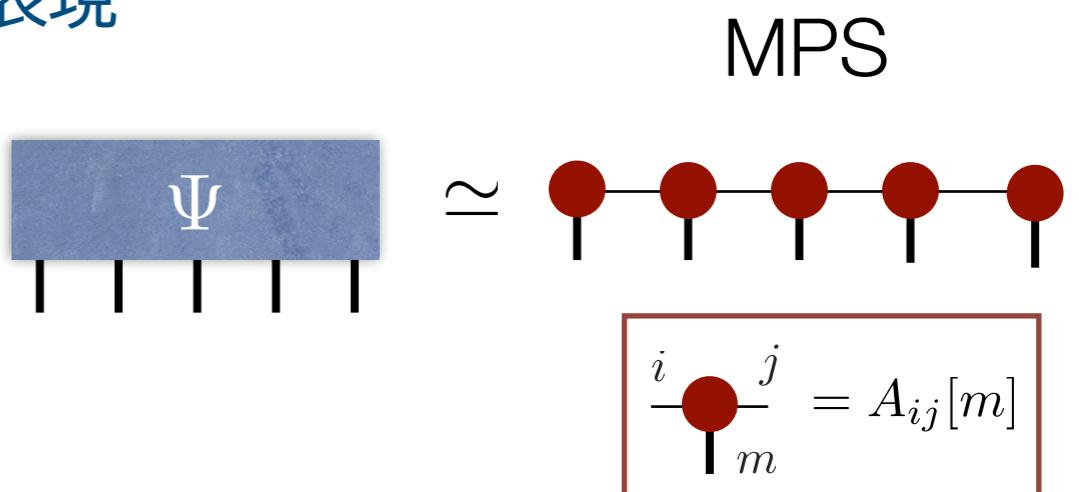
• Gourianov et al., Nat. Comput. Sci. 2, 30 (2022)



MPS (Matrix Product State)

- 一次元的につながったテンソルネットワーク
 - 応用数理分野では「Tensor Train」と呼ばれる
- N 本足のテンソル(ベクトル)を行列の積で表現

$$\Psi_{i_1 i_2 \dots i_N} \simeq A_1[i_1] A_2[i_2] \cdots A_N[i_N]$$



- MPSから外に出ている足を「物理的な足」と呼ぶ
- テンソルの間を繋ぐ足を「仮想的な足」と呼ぶ
- 仮想的な足について縮約をとると、元のテンソルと同じ形に戻る

Bell状態

- ・ 波動関数

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

- ・ 重ね合わせ係数は長さ4のベクトル $(1/\sqrt{2}, 0, 0, 1/\sqrt{2}) \rightarrow 2 \times 2$ 行列 $\begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}$

- ・ テンソルネットワーク(MPS)表現



- ・ 2 qubit \rightarrow 物理的な足は2本 (それぞれボンド次元 $d = 2$)
- ・ Bell状態を表すには、仮想的な足のボンド次元は $\chi = 2$ でよい

$$\begin{array}{c} \text{Diagram: } \begin{array}{c} \text{Red circle} \xrightarrow{0} \text{---} \\ |_0 \end{array} = \begin{array}{c} \text{Red circle} \xrightarrow{1} \text{---} \\ |_1 \end{array} = \begin{array}{c} \text{---} \xrightarrow{0} \text{Red circle} \\ |_0 \end{array} = \begin{array}{c} \text{---} \xrightarrow{1} \text{Red circle} \\ |_1 \end{array} = \frac{1}{2^{1/4}} \end{array}$$

- ・ これ以外の要素はゼロ

GHZ狀態

Hands-on Lab: 5_mps2statevector

- ・ N スピン系のGHZ (Greenberger–Horne–Zeilinger) 状態

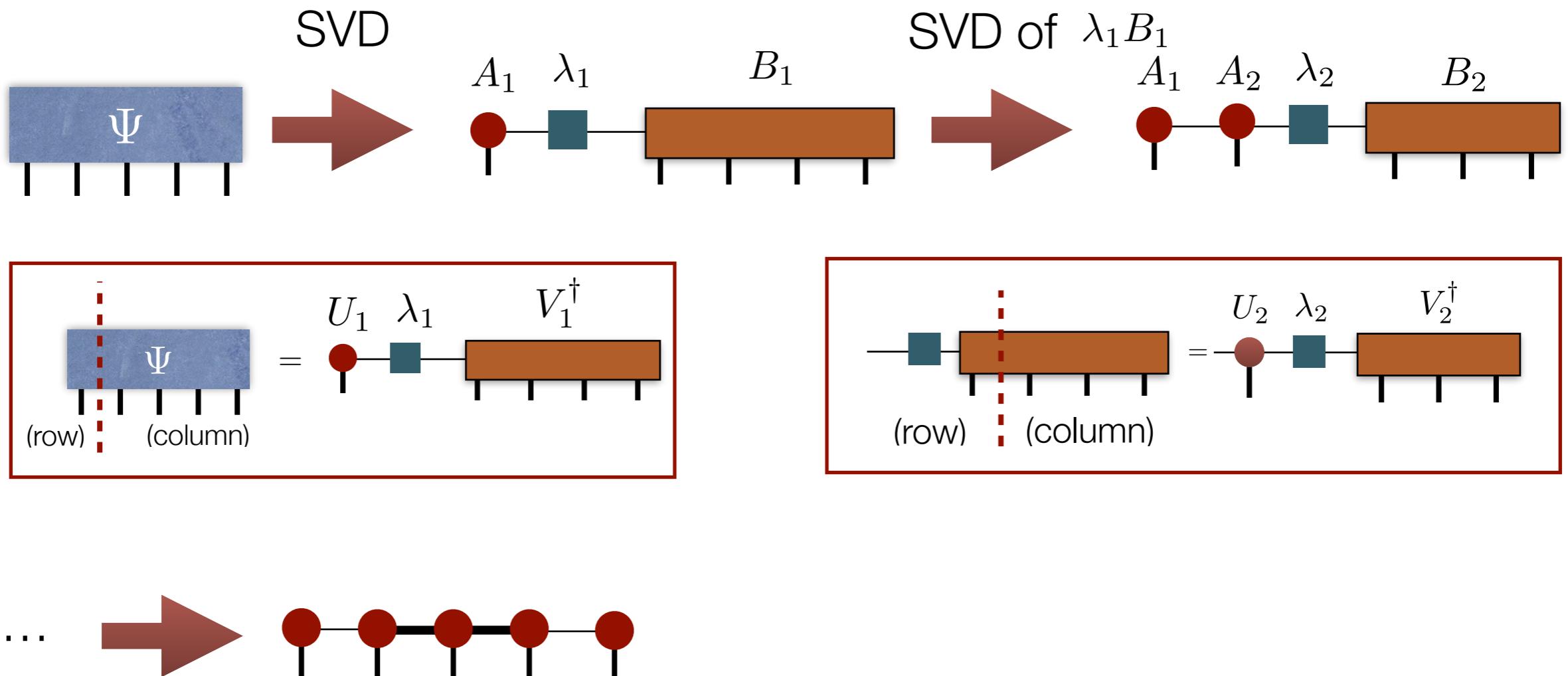
$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|000\dots 0\rangle + |111\dots 1\rangle)$$

- ### ・テンソルネットワーク表現 ($\chi = 2$)

$$\begin{array}{c} 1 \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} 2 \\ \text{---} \\ \text{---} \end{array} = 2^{-1/(2N)}$$

行列積状態への変換

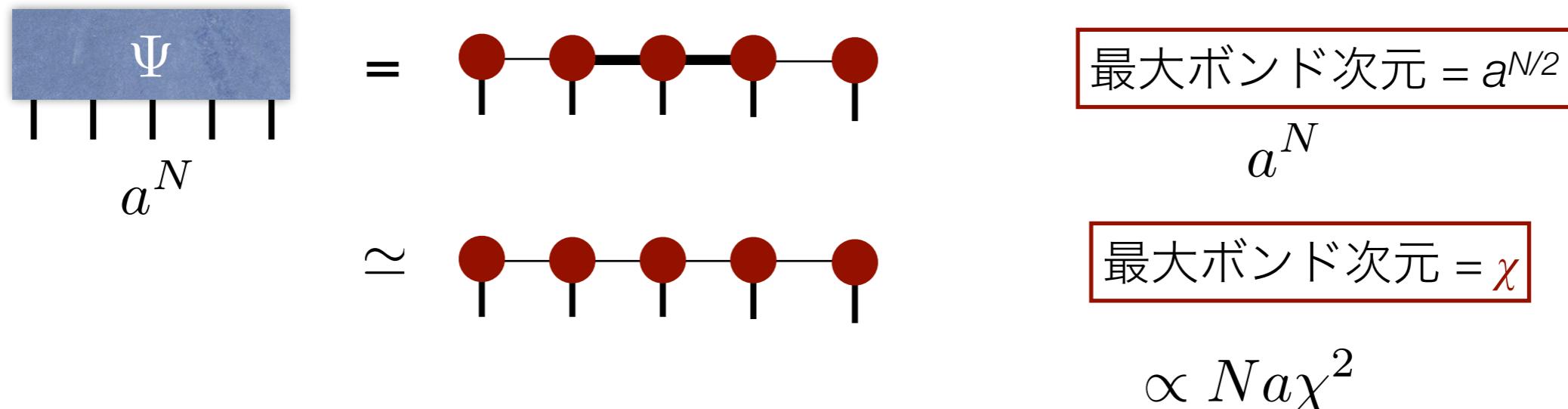
- 任意のテンソル(ベクトル)は特異値分解を繰り返すことで厳密なMPS表現に常に変換できる



- ボンド次元は場所に依存 \rightarrow 最大のボンド次元 $= O(a^{N/2})$

テンソルの低ランク近似

Hands-on Lab: 6_statevector2mps

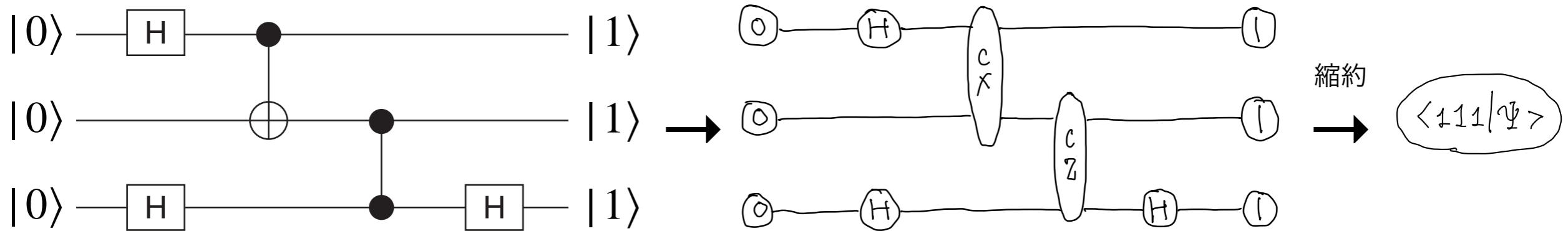


- ・ゼロの特異値が現れた場合には、それに対応する行や列は捨ててよい
 - ・テンソルネットワークによるテンソルの圧縮表現
- ・十分に小さな特異値を捨てることで、元のテンソルを精度良く近似可能
 - ・テンソルネットワークによるテンソルの低ランク近似
 - ・ N の指数関数のデータ量を N の多項式にまで大幅に減らせる!
 - ・(どこまで効率的に近似可能かどうかは、表そうとしている「状態」の性質(=エンタングルメントの強さ)による)

量子系のユニタリ時間発展

量子回路のテンソルネットワークシミュレーション

- 量子回路 ⇒ テンソルネットワーク

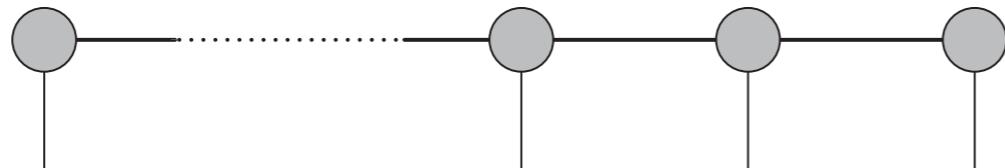


- 1ビットゲート → 2本足テンソル (2×2)
- 2ビットゲート → 4本足テンソル ($2 \times 2 \times 2 \times 2$)
- 初期状態・出力状態 → 1本足テンソル(ベクトル)の組
- 出力状態の振幅 → テンソルネットワークを縮約することで得られる
- 量子回路では必ず左から右に時間が進む
 - テンソルネットワークの縮約はどのような順番で計算しても結果は同じ
- テンソルネットワークの古典コンピュータでの計算は非常にコストが高い
 - 縮約順序を工夫することで計算量を劇的に減らせる可能性
 - 近似的な縮約も可能(低ランク近似、テンソルくりこみ群)

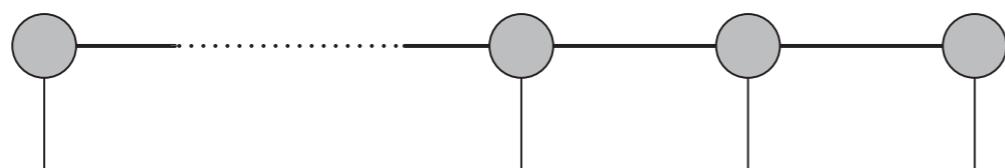
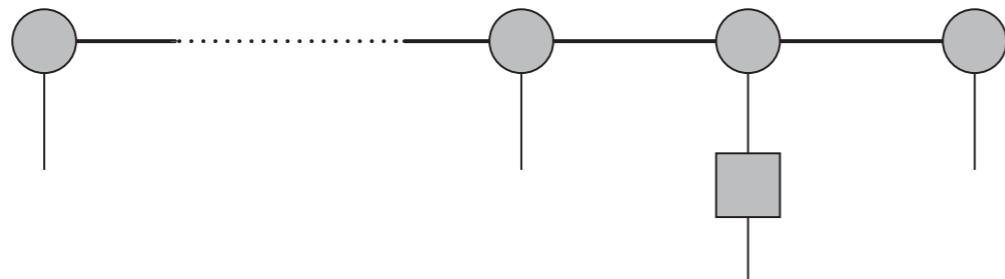
MPSによる量子回路シミュレーション: TEBD

- ・波動関数のMPS表現

- ・ n -量子ビット波動関数をMPS(3本足テンソル- $\times n$)で近似



- ・1量子ビットゲート（近似なしで実行可能）



2量子ビットゲート

Hands-on Lab: 7_tebd

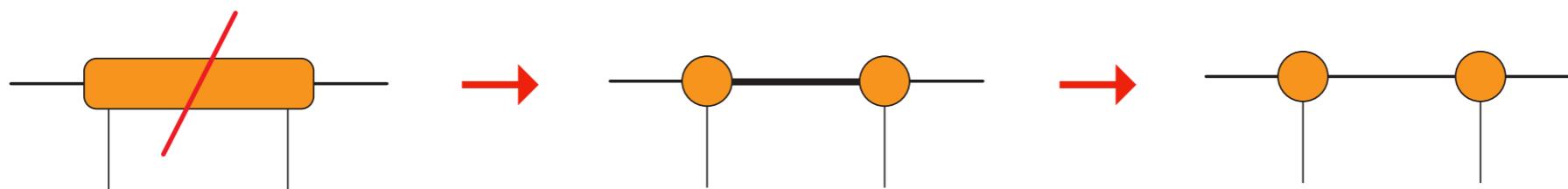
- MPSに2量子ビットゲートを掛ける(縮約)

- 2量子ビットゲートをかけると形が変わる



- テンソルを分解してもとのMPSの形にもどす

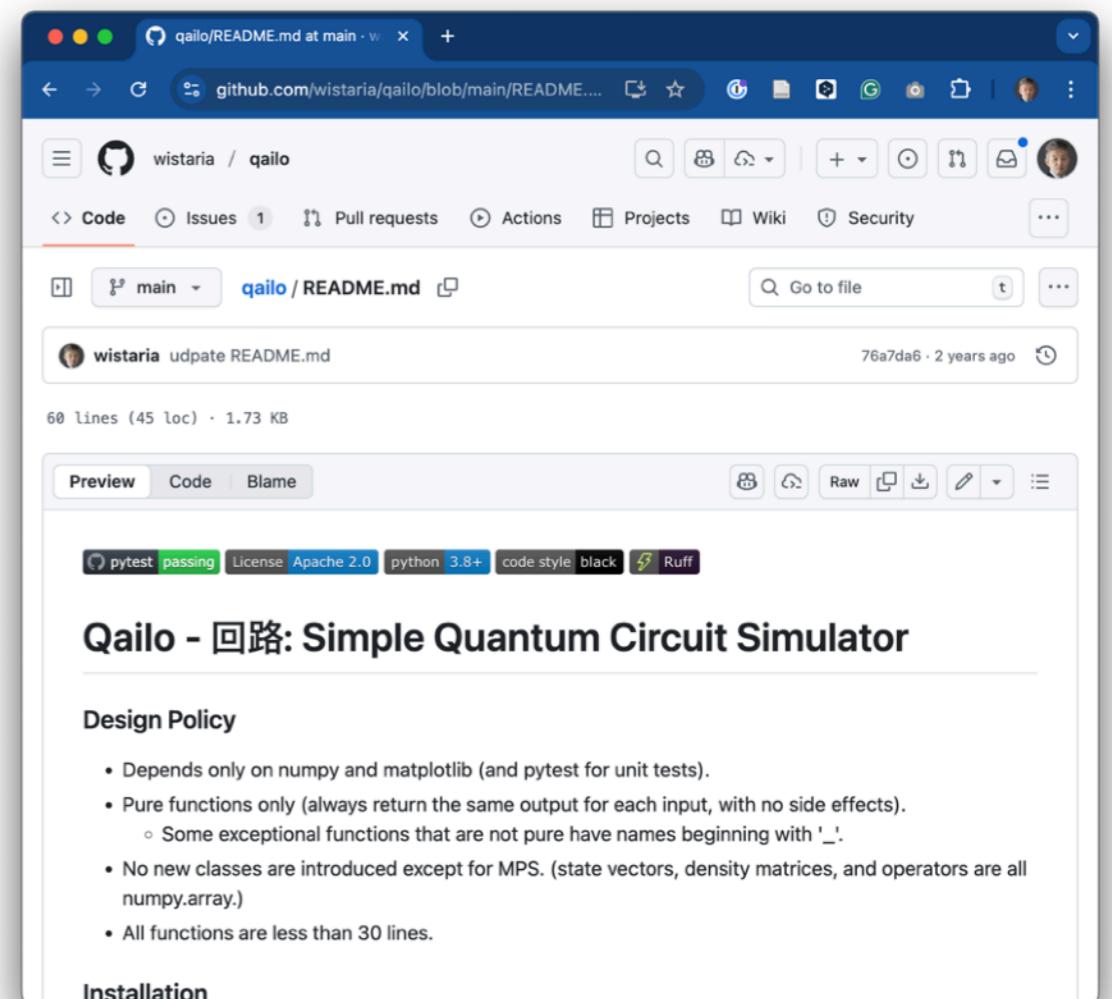
- 分解するだけではボンド次元が増加してしまうので、間をつなぐボンド次元が χ を超えないように低ランク近似する



- (低ランク近似の際にMPS全体が最適化されるように近似すると、精度が飛躍的に向上する→カノニカル形式の利用)

Qailo - 回路: Simple Quantum Circuit Simulator

- ・オープンソースソフトウェア on GitHub
 - ・<https://github.com/wistaria/qailo>
- ・Qailoの設計方針
 - ・numpyとmatplotlibのみに依存(ユニットテストにはpytestも使用)。
 - ・純粋関数のみ(各入力に対して常に同じ出力を返し、副作用なし)
 - ・純粋でない例外的な関数は名前を “_” で始める
 - ・MPS を除き新規クラスは導入しない(状態ベクトル、密度行列、演算子は全て numpy.array)
 - ・全関数を30行未満とする



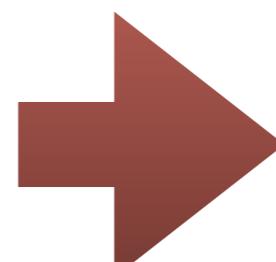
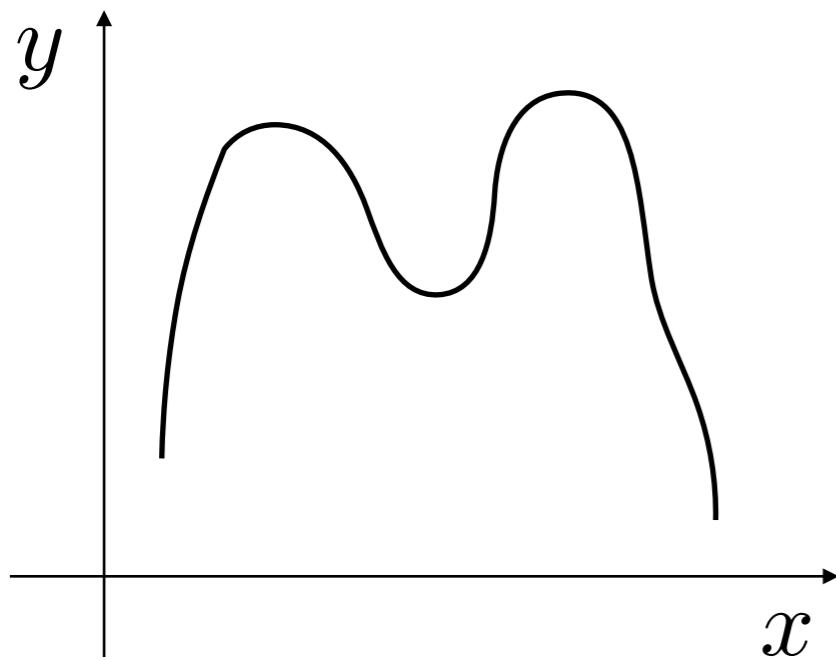
階層構造の圧縮表現

滑らかな関数の離散化

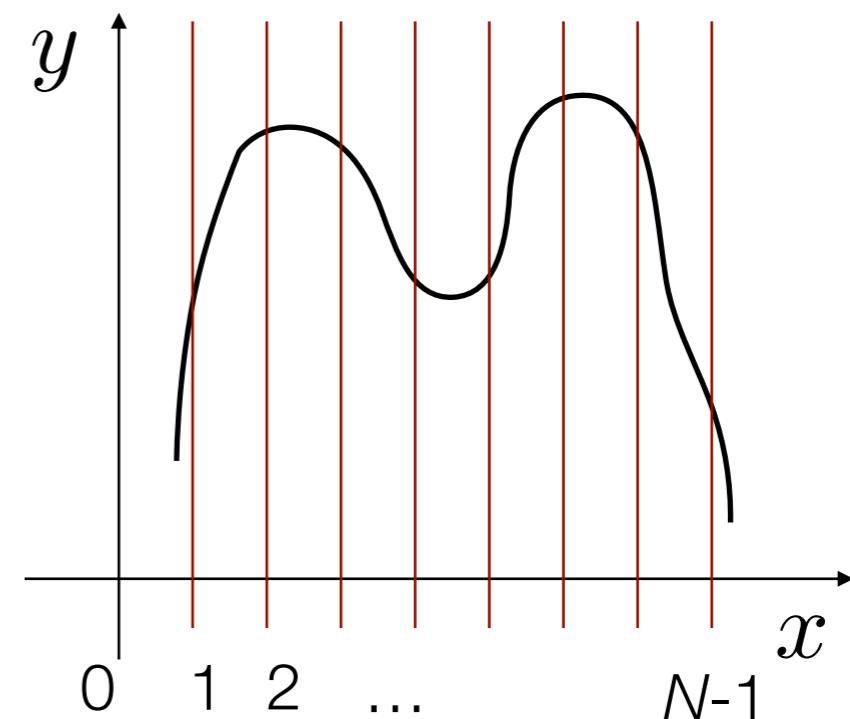
$$y = f(x) \quad x \in [0, 1)$$

$$x_i = \frac{i}{N}, i = 0, 1, \dots, N - 1$$

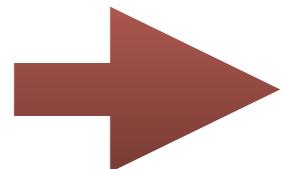
$$y_i = f(x_i)$$



離散化



離散化された関数値のテーブル y_i は N 成分ベクトル

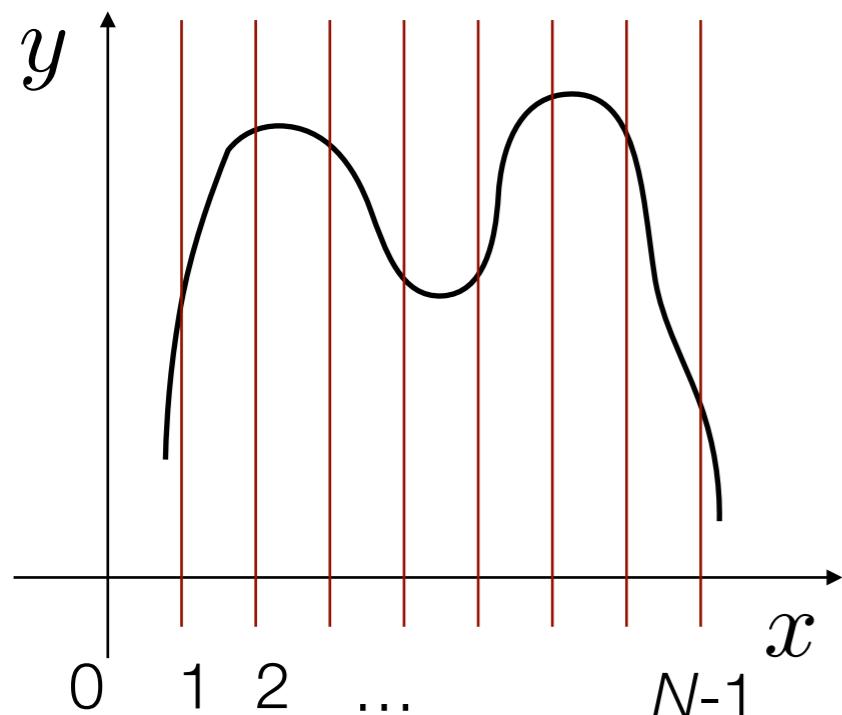


関数の情報圧縮？

関数のテンソル化

$$x_i = \frac{i}{N}, i = 0, 1, \dots, N-1$$

$$y_i = f(x_i)$$

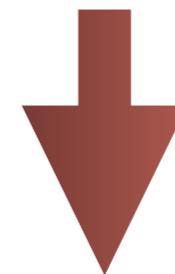


インデックス i の二進数表示

$$N = 2^M$$

$$i = i_0 2^{M-1} + i_1 2^{M-2} + \dots + i_{M-2} 2^1 + i_{M-1}$$

$$i_m = \{0, 1\}$$



$$x_i = x_{i_0, i_1, \dots, i_{M-1}} = \sum_{m=0}^{M-1} \frac{1}{2^{m+1}} i_m$$

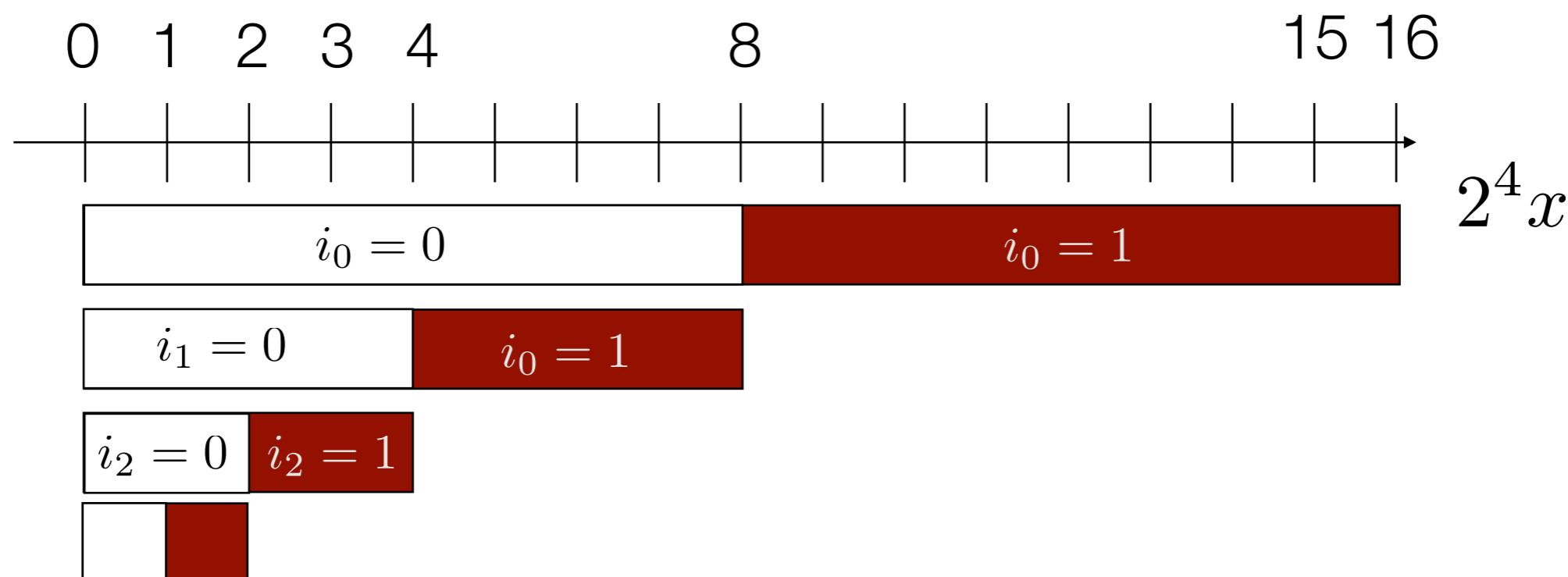
$$y_i = f(x_i) = y_{i_0, i_1, \dots, i_{M-1}}$$

2^M グリッド上の関数 = M 本脚のテンソル

二進数表記と長さスケール

$$x \in [0,1) の M ビット二進数表記 : x_i = x_{i_0, i_1, \dots, i_{M-1}} = \sum_{m=0}^{M-1} \frac{1}{2^{m+1}} i_m$$

各ビットは“長さスケール”に対応している



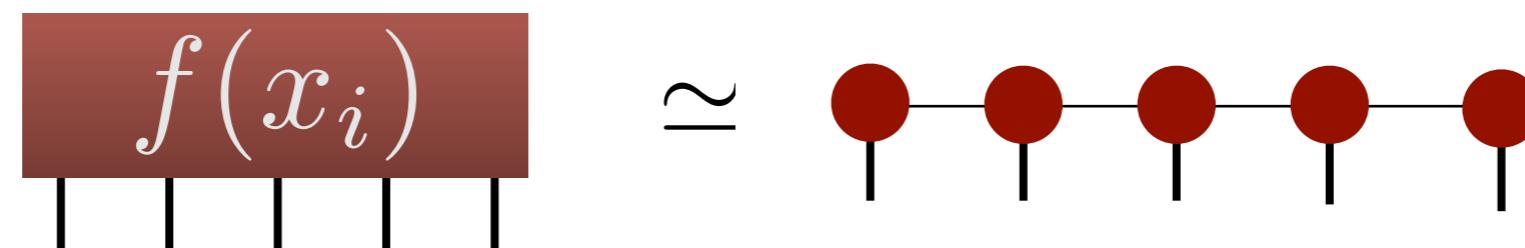
m 番目のビットの変化 = 長さ $\frac{1}{2^{m+1}}$ の変化

関数のテンソルネットワーク近似

$$x_i = x_{i_0, i_1, \dots, i_{M-1}} = \sum_{m=0}^{M-1} \frac{1}{2^{m+1}} i_m$$

$$y_i = f(x_i) = y_{i_0, i_1, \dots, i_{M-1}}$$

長さスケールを意識したテンソルネットワーク = MPS



Quantized/Quantics Tensor Train (QTT)

$N = 2^M$ の情報を $O(M = \log_2 N)$ に圧縮

指数関数的な圧縮の可能性

I. V. Oseledets, Dokl. Math. **80**, 653 (2009).

B. N. Khoromskij, Constr. Approx. **34**, 257 (2011).

I. V. Oseledets, Constr. Approx. **37**, 1 (2013)

QTT表現の例 1：指数関数

指数関数 : $f(x) = \exp [\lambda x]$

$$\rightarrow f(x_i) = \exp \left[\lambda \sum_m \frac{1}{2^{m+1}} i_m \right] = \prod_{m=0}^{M-1} \exp \left[\lambda \frac{1}{2^{m+1}} i_m \right]$$

二進数表記

インデックス毎
の指数関数の積

= 量子系の積状態

$$\exp [\lambda x] = \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet$$

指数関数 = ボンド次元 1 の QTT

$$\bullet = \exp \left[\lambda \frac{1}{2^{m+1}} i_m \right]$$

QTT表現の例 2：三角関数

Hands-on Lab: 8_function2qtt

$$\text{三角関数: } f(x) = \cos(\omega x) = \frac{e^{i\omega x} + e^{-i\omega x}}{2}$$

二つの指数関数の和 → ボンド次元2のQTT

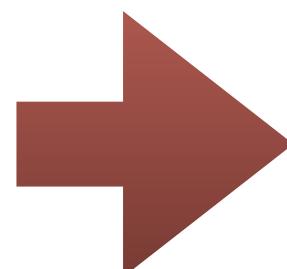
*以下の議論は複素数テンソルの場合

$$\begin{aligned} \cos(\omega x) &= \frac{1}{2} \left(\begin{array}{c} \text{Red nodes} \\ \vdots \end{array} + \begin{array}{c} \text{Blue nodes} \\ \vdots \end{array} \right) \\ &= \frac{1}{2} \left(\begin{array}{ccccc} \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} \\ \hline \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} & \text{Blue nodes} \end{array} \right) \begin{array}{c} \text{Red node} \\ \vdots \end{array} = \exp \left[i\omega \frac{1}{2^{m+1}} i_m \right] \\ \alpha \begin{array}{c} \text{Blue node} \\ \hline \text{Blue node} \end{array} \beta &= \begin{cases} \begin{array}{c} \text{Red node} \\ \vdots \end{array} & (\alpha = \beta = 0) \\ \begin{array}{c} \text{Blue node} \\ \vdots \end{array} & (\alpha = \beta = 1) \\ 0 & (\alpha \neq \beta) \end{cases} \begin{array}{c} \text{Blue node} \\ \vdots \end{array} = \exp \left[-i\omega \frac{1}{2^{m+1}} i_m \right] \end{aligned}$$

QTT表現の例 2：三角関数（補足）

$$\cos(\omega x) = \frac{1}{2} \begin{array}{c} \text{---} \\ | \\ | \\ | \\ | \end{array} \begin{array}{ccccc} \text{blue circle} & \text{---} & \text{blue circle} & \text{---} & \text{blue circle} \\ | & & | & & | \\ \text{blue circle} & \text{---} & \text{blue circle} & \text{---} & \text{blue circle} \end{array} \begin{array}{c} \text{red circle} \\ | \\ \text{---} \end{array} = \exp \left[i\omega \frac{1}{2^{m+1}} i_m \right]$$

$$\alpha \begin{array}{c} \text{---} \\ | \\ \text{blue circle} \end{array} \beta = \begin{cases} \begin{array}{c} \text{red circle} \\ | \\ \text{---} \end{array} & (\alpha = \beta = 0) \\ \begin{array}{c} \text{teal circle} \\ | \\ \text{---} \end{array} & (\alpha = \beta = 1) \\ 0 & (\alpha \neq \beta) \end{cases} \begin{array}{c} \text{---} \\ | \\ \text{teal circle} \\ | \\ \text{---} \end{array} = \exp \left[-i\omega \frac{1}{2^{m+1}} i_m \right]$$



水平の足に、ユニタリ変換 $U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$ の組

$\text{---} = I = U^\dagger U$ を挿入することで実数に変換可能

$$\begin{array}{c} \text{---} \\ | \\ \boxed{U} \end{array} \begin{array}{c} \text{---} \\ | \\ \text{blue circle} \end{array} \begin{array}{c} \text{---} \\ | \\ \boxed{U^\dagger} \end{array} \text{---} = \begin{pmatrix} \cos(\theta_m i_m) & \sin(\theta_m i_m) \\ -\sin(\theta_m i_m) & \cos(\theta_m i_m) \end{pmatrix}$$

$$\theta_m = \frac{\omega}{2^{m+1}}$$

QTT表現での微分

$$y = f(x) \rightarrow \text{微分} : \frac{dy}{dx} = \frac{d}{dx} f(x)$$

微分の差分近似 : $\frac{dy}{dx_i} \simeq \frac{y_{i+1} - y_{i-1}}{2\Delta x}$ $\Delta x = \frac{1}{N}$

$$= \frac{1}{2\Delta x} \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & -1 \\ -1 & 0 & 1 & 0 & & \vdots \\ 0 & -1 & 0 & 1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & \dots & 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_{N-1} \end{pmatrix}$$

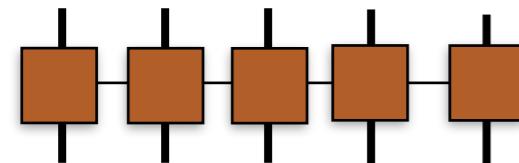
QTTでの効率的な微分計算?

*周期境界条件

QTT表現での微分：行列積演算子

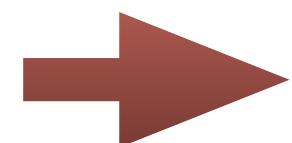
$$\frac{dy}{dx_i} \simeq \frac{1}{2\Delta x} \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & -1 \\ -1 & 0 & 1 & 0 & & \vdots \\ 0 & -1 & 0 & 1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 1 & \dots & 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_{N-1} \end{pmatrix}$$

行列積演算子
(MPO)

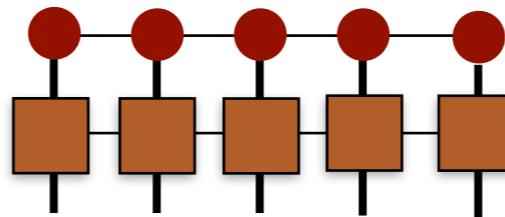


*ボンド次元=3

QTT



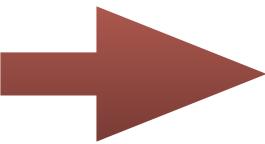
$$\frac{dy}{dx_i} \simeq$$



$O(M = \log_2 N)$
で微分が計算可能

QTT表現での微分：行列積演算子の構築（補足）

$$\frac{dy}{dx_i} \simeq \frac{y_{i+1} - y_{i-1}}{2\Delta x}$$



シフト演算子

$$y_{i+1} = (S^+ y)_i$$

$$y_{i-1} = (S^- y)_i$$

$$\frac{y_{i+1} - y_{i-1}}{2\Delta x} = \frac{1}{2\Delta x} [(S^+ - S^-)y]_i$$

$$S^+ = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & & \vdots \\ 0 & 0 & 0 & 1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 1 & \dots & 0 & 0 & 0 & 0 \end{pmatrix} \quad S^- = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & 0 & & \vdots \\ 0 & 1 & 0 & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 1 & 0 \end{pmatrix} = (S^+)^t$$

シフト演算子はボンド次元2の行列積演算子で容易に書ける
(cf. ハンズオンのコード)

まとめ

- ・ テンソルネットワークはいたるところに現れる
 - ・ 様々なデータをテンソルネットワークで厳密あるいは近似的に表現できる
 - ・ 特異値分解を使うことで最適な近似が可能に
- ・ テンソルネットワーク形式
 - ・ データ(=要素数)の削減
 - ・ 計算量の削減
- ・ ネットワークの選択
 - ・ テンソルネットワーク表現は一意ではない
 - ・ どのような形のテンソルネットワークで表すかにより効率が劇的に変わる
- ・ テンソルネットワークの可能性
 - ・ 量子状態、深い階層にまたがる現象、深層学習などにおいて、情報を効率的に圧縮し、大規模な計算が可能に