

データを圧縮、高速計算

東京大学 講師 松井勇佑

課題名 圧縮線形代数：
データ圧縮による省メモリ
高速大規模行列演算



問題

大量データ

処理

応用

$$\begin{bmatrix} 2.42 \\ 4.09 \\ 1.31 \\ 2.51 \end{bmatrix} \begin{bmatrix} 4.37 \\ 0.11 \\ 7.42 \\ 3.21 \end{bmatrix} \dots \rightarrow \begin{bmatrix} 0.16 & 0.75 & 0.19 & 0.53 \\ 0.10 & 0.37 & 0.21 & 0.91 \\ 0.72 & 0.76 & 0.47 & 0.91 \\ 0.17 & 0.89 & 0.13 & 0.51 \end{bmatrix} \begin{bmatrix} 0.22 \\ 1.34 \\ 0.19 \\ 1.25 \end{bmatrix} = \begin{bmatrix} 2.42 \\ 1.31 \\ 5.08 \\ 2.51 \end{bmatrix}$$

データ量が多すぎるので

④大量メモリ消費。遅い。

- 現状、大規模処理は力技（大量サーバ）
- Google等の大企業しか取り組めない

どのように未来を切り拓くか

大量データ

コード

処理

応用

$$\begin{bmatrix} 2.42 \\ 4.09 \\ 1.31 \\ 2.51 \end{bmatrix} \begin{bmatrix} 4.37 \\ 0.11 \\ 7.42 \\ 3.21 \end{bmatrix} \dots \xrightarrow{\text{圧縮}} \begin{bmatrix} 32 \\ 61 \\ 93 \end{bmatrix} \rightarrow \begin{bmatrix} 71 & 13 & 32 \\ 61 & 61 & 61 \end{bmatrix} = \begin{bmatrix} 2 \\ 93 \end{bmatrix}$$

(1) 圧縮して

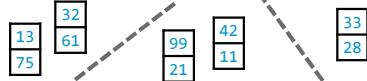
(2) 線形代数演算

- データ圧縮した世界での高速演算体系
- 全てのエンジニア/研究者に便利な道具を提供

成果 (ACT-I + ACT-I 加速)

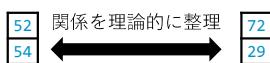
【道具1：実用】圧縮世界での高速処理

- クラスタリング [ACMMM 17]
 - ✓ 2017当時世界最速級・コード公開
 - ✓ 部分探索 [ACMMM 18]



【道具2：理論】添字計算

- 添え字（青い数字）に関係を持たせる
- 新しい理論的枠組み



【アウトリーチ】

- CV分野国内最大学会 (MIRU) および世界最大学会でチュートリアル (CVPR, 6月予定)
- その他招待講演 (東北大、産総研、DeNA、CyberAgent)



技術的詳細の要約

$$\begin{bmatrix} 2.42 \\ 4.09 \\ 1.31 \\ 2.51 \end{bmatrix} \xrightarrow{k^* = \operatorname{argmin}_k \|x - c_k\|_2^2} \boxed{32} \quad k^* = 32$$

▶ ベクトル量子化でデータを圧縮

- 「圧縮データ」に対し、**圧縮したまま演算**を行いたい
 - ✓ 【道具1：実用】圧縮したまま高速探索。高速クラスタリング
 - ✓ 【道具2：理論】添字に関係を持たせる（これまでにない操作）

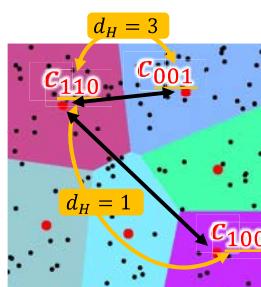
【道具1：実用】クラスタリング



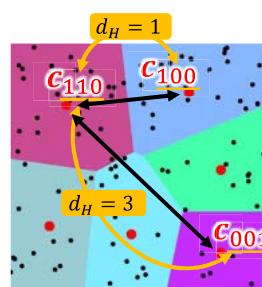
▶ 圧縮したデータの世界でk-means

- データを格子上の点におき、格子上の世界でクラスタリング
- ✓ k-meansに比べ、10xから100x高速。10xから100xメモリ効率良
- コード公開。プレスリリース。
- 手法：
 - 圧縮コードに対する探索 [Matsui+, ICCV15]
 - 圧縮コードに対する平均計算 [Matsui+, ACMMM 17]

【道具2：理論】添字に意味をもたせる



- この二つは全く同じ量子化化
- 添字だけが違う
- なのでハミング距離は違う



- 「添字割り振り」の自由度がある
- この自由度を使って、**新たな関係性**を埋め込む
- すると、これまでにない数学体系が作れる（かも）
- 例：
これまでと全く同じベクトル量子化器だが、添字をじっと眺めるとなぜか別の距離尺度が計算できる
- より応用的な例：
商品の類似画像検索で、圧縮された特徴ベクトルをじっと眺めるとなぜかカテゴリ検索が出来る

$$\begin{aligned} d(c_{011}, c_{110}) &= \|c_{011} - c_{110}\|_2 \\ d \sim d_H \text{ になるような添字割り当てを探す} \\ d_H(011, 110) &= \mathbf{1}^\top \begin{pmatrix} [0] \\ [1] \oplus [1] \\ [1] \end{pmatrix} \\ d \sim d_{WH} \text{ になるように } \mathbf{w} \text{ を最適化} \\ d_{WH}(011, 110) &= \mathbf{w}^\top \begin{pmatrix} [0] \\ [1] \oplus [1] \\ [1] \end{pmatrix} \end{aligned}$$

ユークリッド距離そのものを埋め込む例：
▶ ベスト割り振りを求める（置換の最適化）
▶ さらにベストな重みを求める

ユークリッド以外（別の距離尺度）を埋め込めるかもしれない