Vol.2022-AVM-116 No.13
2022/2/25

ストリームデータの部分空間クラスタリングにシーケンスデータの特徴を反映させた方法の検討

大和田 悠生[1,a)]　堀江 光彦[2]　笠井 裕之[1,2]

概要：近年，データをいくつかの部分空間に分類する部分空間クラスタリング (SC) が広く利用されている．その中でも，意味のある順序構造をもつシーケンスデータに特化し，その制約を組み込んだ手法が，性能の高さから注目を集めており，その代表例として OSC (Ordered Subspace Clustering) がある．ただし OSC は，計算量の二次関数的な増大のため，スケーラビリティに限界がある．

ここで，データが随時新規に到着し，部分空間の構造が一定でないストリームデータについて考える．ストリーム中の構造の変化に厳密に対応できる手法はこれまでほとんど考案されてこなかったが，StreamSSC (Stream Sparse Subspace Clustering) は，部分空間を代表する集合を抽出し適宜更新することで，前述の課題に対応した．またその過程で，計算量の大幅な削減にも成功している．

本稿では，時間情報の制約を考慮することでシーケンスデータに対応することが可能な StreamSSC の拡張手法について検討する．実際には，ストリームデータの多くがシーケンスデータの特徴を持つことを念頭に置いて，ストリームデータの枠組みの中で，OSC によるクラスタリングを考えることで，スケーラビリティの増大を目指す．数値実験から，提案手法が既存の OSC より計算時間の点で優っていることを示す．

1. はじめに

現在，機械学習をはじめとする多くのアプリケーションでは，高次元のデータを扱う場面が頻発し，このようなデータを扱うことが，アルゴリズムの性能に悪影響を与えている．ここで，多くの場合，高次元データは低次元の部分空間から抽出されたデータである [1]．例えば，異なる照明条件下で撮影された同一の人物の顔写真は，9 次元の線形部分空間で特徴づけられることが知られている [2]．低次元構造の復元は，高次元データを扱う上で大きな助けとなるが，実世界のデータは，別々の部分空間に属するデータの組み合わせであることが多い．部分空間クラスタリングとは，そのような複数の低次元部分空間から抽出されたデータの集合を，各部分空間に応じてクラスタリングする手法である．

本稿では，部分空間クラスタリングの中でも特にシーケンスデータに適用された手法について考察する．シーケンスデータとは，意味のある順序を持ち，各データが順序の通りに並び，再生されたときにはじめて意味を成すデータの集合である．画像が時系列で順序立てて集まり，再生される映像データは，時系列データの代表的な例といえる．シーケンスデータに共通する特徴として，順序の近い要素が密接に関連しているということが挙げられる．映像の例でも，一つのシーンの終了まで，連続するフレームが類似すると考えることができる．シーケンスデータの部分空間クラスタリングにおいては，この類似性をペナルティとして組み込むことで，多くの手法が高い性能を発揮している．その中でも後述する OSC [3] は，類似性を制約項として追加し，最適化を行うことで，制約なしのクラスタリング手法と比較して，精度の大きな向上を達成したことで広く知られている．ただし計算量の増大に悩まされる手法も多く，スケーラビリティの増加は，実世界データに応用するに当たっての重要な課題となっている．

本稿ではさらに，スケーラビリティの改善へのアプローチとして，ストリームデータの研究についても記述する．ストリームデータとは，各データ点が随時新規に到着し，全体の構造が，最後の点の到着か，データ点入手を打ち切るまで確定しないデータであり，例として，配信されている動画や，気象・交通情報などをリアルタイムで取得する場合などが挙げられる．ストリームデータの部分空間クラ

[1] 早稲田大学 基幹理工学部 情報通信学科
Department of Communications and Computer Engineering,School of Fundamental Science and Engineering, Waseda University
[2] 早稲田大学大学院 基幹理工学研究科 情報理工・通信専攻
Department of Computer Science and Communications Engineering, Graduate School of Fundamental Science and Engineering, Waseda University
[a)] yusei0122789@asagi.waseda.jp

スタリングにおける課題は，膨大な量のデータの，変化する部分空間構造を適宜検出する必要がある点であり，この課題を完全に克服した手法はほとんど存在しなかった．しかし近年，後述する StreamSSC [4] が，動的なデータへの対応と計算量の削減という観点で良い結果を出しており，注目されている．

これを踏まえて，本稿ではシーケンスデータの部分空間クラスタリングを，ストリームデータの枠組みで行う方法について考察する．基礎となるアイデアは，先ほど例に挙げた配信動画や気象・交通情報などをはじめ，リアルタイムで取得するデータは時間変化をするものが多く，つまりはシーケンスデータの一種と捉えられるというものである．具体的には，StreamSSC の枠組みを拡張し，OSC をオンラインで実装することで，課題であったスケーラビリティの増加を追求する．実験を行い，提案手法が既存の OSC と比較して，計算時間を削減できているかを観察することで，有効性を確認する．

2. 先行研究

2.1 部分空間クラスタリング

全ての部分空間クラスタリングに共通する重要なアイデアは, 対象データの低次元部分空間構造をどのように学習するかである．本稿では, 近年の主流となっている, スペクトルベースの手法に焦点を当てる．これは, データポイント間の類似性を表す親和性行列を学習し，NCut [5] などのスペクトル・クラスタリングアルゴリズムを適用するというもので, 適切な親和性行列の構築が, 各手法における性能の差として現れる．さらに, スペクトルベースの手法の多くに共通するアプローチとして,「高次元データが複数の低次元部分空間に属するとき, 各データは他のデータの線形結合で表現可能」という, 自己表現特性 [1] という考えがある．対象となるデータを $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$ とし，自己表現特性を数式化すると, 係数行列 $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N] \in \mathbb{R}^{N\times N}$ を用いて,

$$\mathbf{x}_i = z_{1i}\mathbf{x}_1 + z_{2i}\mathbf{x}_2 + ... + z_{Ni}\mathbf{x}_N \tag{1}$$

$$\mathbf{x}_i = \mathbf{X}\mathbf{z}_i \tag{2}$$

$$\mathbf{X} = \mathbf{X}\mathbf{Z} \tag{3}$$

と表される．このように，$\mathbf{Z}$ が $\mathbf{X}$ の新しい表現として導出できる．この $\mathbf{Z}$ はデータ間の関連性を，係数の大きさとして反映したものになり，$\mathbf{x}_i$ と $\mathbf{x}_j$ の相関を表す値は z_{ij} と z_{ji} の 2 つ存在する．よって，自己表現特性を用いた場合の親和性行列 $\mathbf{W}$ は，$\mathbf{W} = \frac{|\mathbf{Z}|+|\mathbf{Z}|'}{2}$ と取ることができる．

2.1.1 SSC (Sparse Subspace Clustering)

SSC [1] は，自己表現特性をはじめに組み込んだクラスタリング手法であり，係数 $\mathbf{z}_i$ の最も疎な表現を見つけることを目標としている．これは，$\mathbf{X}$ のデータ列 $\mathbf{x}_i$ に対して，$\mathbf{z}_i$ は非ゼロ成分が $\mathbf{x}_i$ と同じ部分空間に属するデータポイントに対応する解を得られ，理想的には非ゼロ成分の個数が，$\mathbf{x}_i$ が属する部分空間の次元と一致するという考え [1] によるものである．具体的には，以下の最適化問題を考える．

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_0 \quad \text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{Z} \tag{4}$$

ただし，式 (4) の厳密な解を求めるには，総当たりで計算を行うほかなく，$\mathbf{Z}$ の要素数に応じて計算量が指数オーダーで増加するので，実際には ℓ_1 ノルムを用いて近似解を求める．ノイズと外れ値の存在を考慮し，実際の目的関数は以下のように表される．

$$\min_{\mathbf{E},\mathbf{S},\mathbf{Z}} \|\mathbf{Z}\|_1 + \frac{\lambda_1}{2}\|\mathbf{E}\|_F^2 + \lambda_2\|\mathbf{S}\|_1 \quad \text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E} + \mathbf{S}, \mathrm{diag}(\mathbf{Z}) = \mathbf{0} \tag{5}$$

$\mathbf{Z}$ の疎な表現のための制約は第一項で課され，それにノイズの影響を加味して定式化している．この手法は，解の保証と実際の性能という観点で優れているため，今日の部分空間クラスタリングにおいて最もよく知られている手法である．

2.1.2 OSC (Ordered Subspace Clustering)

OSC [3] は，SSC にシーケンスデータの特徴を反映させた手法である．疎な表現を追求する基本方針は同様に，シーケンスデータの順序構造を，$\mathbf{Z}$ の隣接する列が類似するとして定義している．定式化は，以下の式のようになる．

$$\min_{\mathbf{Z}} \frac{1}{2}\|\mathbf{X} - \mathbf{X}\mathbf{Z}\|_F^2 + \lambda_1\|\mathbf{Z}\|_1 + \lambda_2\|\mathbf{Z}\mathbf{R}\|_{2,1} \quad \text{s.t. } \mathrm{diag}(\mathbf{Z}) = \mathbf{0} \tag{6}$$

ここで,

$$\mathbf{R} \in \mathbb{R}^{N\times N-1} = \begin{bmatrix} -1 & & \\ 1 & \ddots & \\ & \ddots & -1 \\ & & 1 \end{bmatrix}$$

であり，$\mathbf{Z}\mathbf{R} = [\mathbf{z}_2 - \mathbf{z}_1, \mathbf{z}_3 - \mathbf{z}_2, ..., \mathbf{z}_N - \mathbf{z}_{N-1}]$ が $\mathbf{Z}$ の列の差分を与える．$\|\mathbf{A}\|_{2,1} = \sum_{i=1}^{n}\|\mathbf{a}_i\|_2$ を用いることで，列ごとの差分だけでなく，$\mathbf{Z}$ 全体に類似性の制約を課すことが可能となっている．OSC はシーケンス部分空間クラスタリングにおいて，SSC をはじめとするシーケンスデータの制約がない手法と比較して高い性能を発揮しているが，最適化の過程で計算量がデータ量に対して二次関数的に増大するため，大規模データにそのまま適用することは難しい．その後，最適化手法を改善することで精度を保ちつつ計算量の改善を試みる提案がなされている [6] ものの，新たな変数やパラメータが追加されていることもあり，簡単な手法ではない．

2.2 ストリームデータ

先行研究において，ストリームデータを厳密にクラスタリングできる手法はほとんど考察されていない．SSC を含む，従来の部分空間クラスタリング手法は，静的なデータしか扱うことができず，その後のスペクトルクラスタリングにも，クラスターの個数が必要である等，ストリームデータには適応できない．ストリームデータに関連付けられる手法として，部分空間クラスタリング手法の１つである LRR [7] のメモリ使用量を削減し，データを受け取りつつ実行可能にした OLRSC [8] や，メモリ使用量がデータ量に依存しない主成分分析 (PCA) の一種である ORPCA [9] がある．これらの手法は，データ全体が到着していない段階から計算を行える点が優れているが，従来の部分空間クラスタリング手法と同様，あらかじめ部分空間の個数を知っておく必要があり，データを受信する途中で新規の部分空間に属するデータが検出された場合に対応できないため，ストリームデータに直接対応できるものではなかった．

2.2.1 StreamSSC

StreamSSC [4] は，SSC をストリームデータに適応させた手法である．当手法では，最初に到着した数個のデータを用いて，1 度 SSC を行ってクラスタリング結果を出力し，その後到着したデータに関しては，既存の部分空間に属しているかどうかを判別し，新規の部分空間が検出されるたびに，既知の部分空間に追加するという方法をとることで，変化するデータに対して随時クラスタリング結果を出力することに成功している．SSC に必要なクラスター数は、自己調整戦略 [10] を用いて自動推定することで，入力として必要なものはデータ本体だけに抑えられている．効果としては，先に挙げた全体の到着を待たないクラスタリングが可能な点の他に，初期化と更新で SSC を行うときに必要なデータ量が，データ全体の到着を待って 1 回 SSC を実行するよりも非常に少ないため，同じデータに対してクラスタリングを実行する際も，短時間で終了できる点がある．

3. 提案手法

3.1 課題およびアプローチ

先述した通り，OSC はシーケンス部分空間クラスタリングにおいて，高い性能を発揮できる一方で，計算量の爆発的な増大のため，大規模データへの対応には適さない．最適化の改善に取り組んだ手法は，一定の成果を挙げているものの，変数の追加により，アルゴリズムの複雑さが増しているため，簡単な作業とは言えない．ここで，実世界におけるストリームデータは，リアルタイムで発生するデータを随時取得するものが多く，つまりは時間に基づいた順序を持つシーケンスデータと考えることができる．そこで本稿では，OSC のスケーラビリティ改善へのアプローチとして，ストリームデータへの適応を考える．具体的には，StreamSSC の枠組みの中で，シーケンスデータの特徴を OSC 等の制約で付加した StreamOSC を考案する．主目的は OSC と同等の性能を維持しつつ，計算量を削減することだが，副次的な成果として，OSC がなしえなかった動的なデータへの対応も達成する．

3.2 アルゴリズム

このセクションでは，StreamOSC のアルゴリズムについて，StreamSSC から引き継いだ，核となる部分や，シーケンスデータに適合させて改良した部分を中心に解説する．入力として，K 個の部分空間 $[S_1, S_2, \cdots, S_K]$ から抽出された n 個のデータの集合 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ と，初期化サイズ T_0，部分空間の次元の上界 d，部分空間の変化の検出に用いられる閾値 γ を与える．$\mathbf{X}$ のうち，同じ部分空間 S_i に属するデータの集合は，部分空間別のデータ $\mathbf{C}_{fin} = [\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_K]$ のうち，$\mathbf{C}_i$ に含まれる．最終的に得たいのは，各データが属する部分空間のラベル $\mathbf{l} = [l_1, l_2, \cdots, l_n]$ である．以下に，クラスタリングの流れを順を追って示す．

(1) 初期化として，$[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{T_0}]$ に対して OSC を実行し，出力ラベル $[l_1, l_2, \cdots, l_{T_0}]$ を得る．ここで，$[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{T_0}]$ 内に含まれる部分空間の個数が必要になるため，StreamSSC と同様に，自己調整戦略 [10] を利用して，初期化されたクラスター数 $k = k_0$ と，部分空間別に分類したデータ $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_{k_0}]$ を得る．

(2) 各部分空間から，代表セット $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_k]$ を抽出する．この作業は，ストリームデータを扱う上で非常に重要となる動作であり，StreamSSC の根幹をなしている．なぜなら，ストリームデータは構造が最後まで確定しないデータであるため，到着したデータ全てをメモリに格納することは，サイズが著しく大きいデータを受け取ったときを考慮すると非現実的であるからである．ここで重要なのは，$\mathbf{C}_i$ のうちいくつのデータを代表セット $\mathbf{R}_i$ として抽出するかであり，部分空間を代表するには，$\mathbf{R}_i$ が S_i の基底を含んでいる，すなわち $span(\mathbf{R}_i) = S_i$ である必要がある．ここで，S_i 上のデータ点が一様に分布しており，S_i の次元が d_i である場合，$span(\mathbf{R}_i) = S_i$ をみたす代表セット $\mathbf{R}_i$ の最小要件は，C_i から無作為に選ばれた d_i 個のデータである [4]．これにより，簡単に代表セットを選ぶことが可能になるが，実世界データにおいては，各部分空間の次元と，部分空間上のデータの分布は大抵の場合未知である．よって，当手法ではあらかじめ設定した上界 d に基づき，C_i から一様に d 個のサンプルを抽出する．つまり，この段階で $\mathbf{R}$ は $k_0 \times d$ 個のデータで構成されている．d の値を，代表セットが部分空間を代表するに十分で，かつ部分空間全体より

十分に小さくなるような値に設定することで，本手法は最大のパフォーマンスを発揮できる．

(3) 新規に到着したデータ $\mathbf{x}_t \in \mathbb{R}^m$ に対して，$\mathbf{R}$ を用いて既存の部分空間に属すかどうかを判定する．具体的には，以下の最適化問題を考える．

$$\min_{\boldsymbol{\alpha}} \ \|\mathbf{x}_t - \mathbf{R}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \qquad (7)$$

ここで，

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_k \end{bmatrix} \in \mathbb{R}^{(k\times d)}, \boldsymbol{\alpha}_i \in \mathbb{R}^d$$

である．つまり，$\mathbf{x}_t$ を $\mathbf{R}$ 内のデータの線形結合で表したときの係数ベクトルが $\boldsymbol{\alpha}$ であり，$\boldsymbol{\alpha}_i$ は i 番目の部分空間の代表に関連する係数となる．ここで，$\mathbf{x}_t$ が i 番目の部分空間に属するならば，$\boldsymbol{\alpha}_i$ 以外の $\boldsymbol{\alpha}$ の成分は 0 となり，いずれの部分空間にも属さない場合，$\boldsymbol{\alpha}$ の非ゼロ成分は分散する [11]．この基準に従って，$\mathbf{x}_t$ の判定を行い，既存の部分空間に属す場合は，その番号ラベルをクラスタリング結果として出力する．

判定中に，新規の部分空間に属すと思われる $\mathbf{x}_t$ が見つかった場合，部分空間構造の変化を確認するステップに移行する．具体的には，追加で L 個のデータに関して判定を行い，新規の部分空間だと思しき個数が閾値 γ を越えた場合のみ，次のステップの，部分空間の更新を実施する．これは，データに含まれるノイズによって，一部に誤判定が生じることがあるため，その影響を最小限に抑えるための動作である．ここで，部分空間の更新が行われた場合，新規部分空間の代表セットは，追加分含め $L+1$ 個のデータから，部分空間ごとに d 個ずつ選ばれるので，L はそれに足るだけの個数を用意する必要がある．今回は，後述する通り，部分空間が一度に 2 個以上検出される場合を考えず，$L = d + 3$ と設定した．判定の結果，部分空間の更新を行わなかった場合，ここで $\mathbf{x}_t$ から $\mathbf{x}_{t+L}$ までの分類結果を出力する．

(4) 閾値を越えた場合，部分空間構造と，現在保持している代表セットを更新する．方法としては，$\mathbf{R} \cup \mathbf{W}_t^L$ ($\mathbf{W}_t^L = [\mathbf{x}_t, \cdots, \mathbf{x}_L]$) に対して OSC を実行し，新規部分の出力ラベル $[l_t, l_{t+1}, \cdots, l_L]$ を得る．ここで，再度部分空間の個数が必要となるため，StreamSSC では再び自己調整戦略 [10] を用いている．しかし，シーケンスデータには連続するデータは関連性が深いという特徴があるので，異なる新規の部分空間に属するデータが，短い間に連続するという状況は考えにくい．そこで，当手法では，部分空間の個数を単純に 1 つずつ足していく ($k = k + 1$) 方針をとった．この方針は，計算量の多い自己調整戦略の過程を，初期化の一回に抑えられる点で効果がある．その後，新規の代表セット $\mathbf{R}$ を $\mathbf{R} \cup \mathbf{W}_t^L$ 内で部分空間ごとに分けられたデータから d 個ずつ抽出し，更新は完了する．

3.3 計算コスト

先述したように，StreamOSC の重要な貢献は，計算量や使用するメモリ量の削減である．データの全体像が不明なまま分類を実行する必要がある，ストリームデータのクラスタリングにおいて，受け取ったデータ全てを計算に用いるのは現実的でないが，部分空間の代表セットを用いることで，一定量のデータを破棄し続けながらクラスタリングができる．実際に，メモリに格納すべき最大データ量は，データ全体の部分空間の個数 k に対して，$kd + L$ 個と初期化サイズ T_0 個のうち大きいもので済み，計算量もそれに応じて減少させることができる．特に，OSC はデータ数に対して計算量が二次関数的に増大するため，結果として全てのデータに対して処理を行うとしても，一度に扱う量を減らせることが大きな効果を生む．

4. 評価実験

本章では，提案した StreamOSC の有効性を確かめるため，クラスタリング精度と実行時間の観点から，性能比較を行う．実験には，ランダムに生成した部分空間を用いた人工データを用いる．[6], [7] と同様に，複数の部分空間 $\{S_i\}_{i=1}^{\mathrm{Num}}$ を考え，その基底 $\mathbf{U}_i$ を $\mathbf{U}_{i+1} = \mathbf{T}\mathbf{U}_i$ と与える．ここで，$\mathbf{T}$ はランダムな回転行列，$\mathbf{U}_1 \in \mathbb{R}^{100\times 4}$ は 4 次元のランダムな直行基底である．これにより，互いに独立な 4 次元部分空間が Num 個得られる．次に，各部分空間からデータ $\mathbf{X}_i$ を $\mathbf{X}_i = \mathbf{U}_i\mathbf{Q}_i$ として得る．ここで，$\mathbf{Q}_i \in \mathbb{R}^{4\times 20}$ を，行ごとの分散が 0.001，隣り合う列間の分散が 0.0005 のランダムなガウス型多変量行列とおくことで，隣り合った要素が類似するというシーケンスデータの特徴を反映させることができる．最終的に，各データを連結させて，Num 個の部分空間をもつデータ $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \cdots . \mathbf{X}_{\mathrm{Num}}] \in \mathbb{R}^{100\times 20\mathrm{Num}}$ を得る．**図 1** に，Num=7 のときの人工シーケンスデータの例を示す．

実験では，Num=7 のときの OSC および StreamOSC のクラスタリング結果と，計算時間の統計情報を見て，精度の維持と時間削減がなされているかを確認する．実験を通じて，OSC のパラメータとして $\lambda_1 = 0.1$，$\lambda_2 = 1$ を，StreamOSC のパラメータとして $T_0 = 50$，$\gamma = 10$，$d = 10$ を用いた．Num=7 のときの OSC によるクラスタリング結果を**図 2**，StreamOSC によるクラスタリング結果を**図 3**，10 回の時間計測の統計情報を**表 1** に示す．StreamOSC は，人工データにおいて，OSC とクラスタリングで同等の性能を発揮できていた．この結果より，代表セットを用いて既

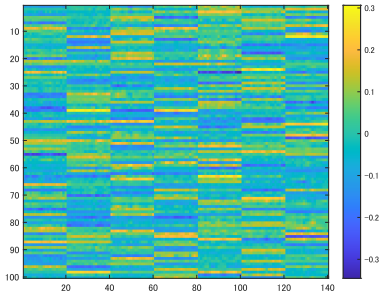

図 1: Num=7 のときの人工シーケンスデータ

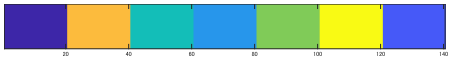

図 2: OSC によるクラスタリング結果

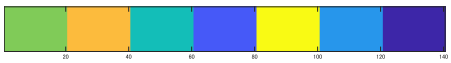

図 3: StreamOSC によるクラスタリング結果

表 1: Num=7 のときクラスタリングにかかった時間

	OSC	StreamOSC
Mean [s]	11.02829	**4.489982**
Var	2.491945	0.002563
Max [s]	13.58643	**4.544955**
Min [s]	9.615686	**4.362408**
Med [s]	10.17486	**4.488993**

存の部分空間を表現し，その情報を到着したデータの判定に用いる方法が正しく機能していることが確認できた．また，実行時間の計測結果より，StreamOSC は OSC と比較して大幅な改善が見られた．既存手法が 140 個のデータ全てを一度に対象にして計算を行うのに対し，代表セットを用いて，初期化，更新と分けて計算を行うことで，一度に扱うデータ量を激減させることができる．

続いて，計算時間の変化を確認するため，Num を 5 から 8 まで変化させた際の OSC と StreamOSC の計算時間の差を比較する．先述の通り，各部分空間から 20 個のデータを抽出するため，全体のデータ数も 100 から 160 まで変化する．計測はそれぞれ 10 回行い，平均を用いてグラフを作成する．結果を**図 4** に示す．データ数の増大に伴う計算時間の増加は，OSC と StreamOSC で大きな開きがあり，データ数が多くなるほど，提案手法が有効に働くことが示された．

5. まとめ

本稿では，シーケンスデータの部分空間クラスタリング手法の 1 つである OSC のスケーラビリティ改善の提案を行った．ストリームデータに対するクラスタリング手法として，計算量の削減に成功している StreamSSC を活用

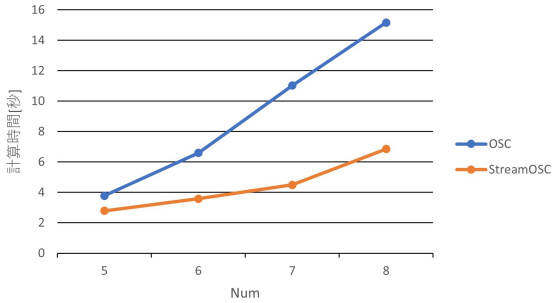

図 4: 部分空間の個数の増大にともなう計算時間の変化

し，シーケンスデータをストリームの枠組みでクラスタリングする StreamOSC を提案した．人工データに対する評価実験で，StreamOSC が OSC に対して，計算時間の観点で優っていることを確認した．今後は，ストリームデータのクラスタリングという観点から，StreamOSC の性能を StreamSSC と比較するとともに，他のシーケンス部分空間クラスタリング手法で今回の枠組みを生かした性能向上ができるかについて検証を進める．

参考文献

[1] E. Elhamifar and R. Vidal. Sparse subspace clustering:algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[2] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2):218–233, 2003.

[3] Stephen Tierney and Junbin Gao. Subspace clustering for sequential data. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014.

[4] Ken Chen, Yong Tang, Long Wei, Pengfei Wang, Yong Liu, and Zhongming Jin. Sparse subspace clustering for stream data. *IEEE Access*, 9:57271–57279, 2021.

[5] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[6] Stephen Tierney, Yi Guo, and Junbin Gao. Segmentation of subspaces in sequential data, 2015.

[7] Z. Lin G. Liu and Y. Yu. Robust subspace segmentation by low-rank representation. *International Conference on Machine Learning*, pages 663–670, 2010.

[8] Jie Shen, Ping Li, and Huan Xu. Online low-rank subspace clustering by basis dictionary pursuit. In *International Conference on Machine Learning*, pages 622–631. PMLR, 2016.

[9] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. *Advances in neural information processing systems*, 26:404–412, 2013.

[10] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

[11] Ehsan Elhamifar, Mahdi Soltanolkotabi, and Shankar Sastry. Approximate subspace-sparse recovery with corrupted data via constrained ℓ_1-minimization. *arXiv preprint arXiv:1412.7260*, 2014.