次元削減を用いた高次元データに対する S-FINCH の高速化

牛尼 索造 [1]　藤原 靖宏 [2]　塩川 浩昭 [3]

S-FINCH はデータストリームに対する階層型クラスタリング手法である．S-FINCH のための空間索引を用いることで低次元データストリームに対しては高速なクラスタリング処理が可能であるが，高次元データストリームに対しては索引が機能せず処理速度を改善することが出来ない．そこで本稿では次元削減を用いて計算コストを削減することにより高次元データストリームに対する S-FINCH の高速化を行なう．また，多様な実データでの実験により次元削減手法とクラスタリング性能の関係について評価を行い 5% 未満の NMI スコアの損失で最大 9 倍の高速化を達成することを確認した．

1　はじめに

クラスタリングとはデータ集合を性質の類似したデータからなる部分集合へとデータ分割する手法である．多くの科学の分野において膨大なデータから性質の類似するデータのグループを見つけ出す処理は非常に重要である．特に近年では，取り扱われるデータの件数や次元数が増加しており，高次元データに対するクラスタリング処理技術への需要が高まっている [1–3].

高次元データに対するクラスタリング手法として Sarfraz らによって提案された FINCH [4] が近年注目を集めている．FINCH は古典的な k-means 法 [5] などとは異なり，データポイント間の近傍グラフ構造を利用することで，パラメータフリーかつ高精度な階層クラスタを構築することが可能な手法である．より具体的な FINCH の処理内容は次のとおりである．まず FINCH は多次元のデータポイントの集合を入力として受け取る．次に，各各データポイントの間のユークリッド距離に基づいて，最近傍となるデータポイントとの間，または最近傍を共有するデータポイント間にエッジを張ることで得られる近傍グラフを構築する．その後，この近傍グラフの中から強連結成分を抽出することで，クラスタを形成する．以降では各クラスタ内から代表データポイントを導出し，同様の近傍グラフ構築処理を行う．これにより，FINCH は階層的なクラスタ構造を抽出する．前述の通り，FINCH は従来の k-means 法 [5] などのクラスタリング手法と比較して，1) 高い精度で真のクラスタを自動的に決定できる，2) ハイパーパラメータを必要としない，3) 異なるドメインのデータでも一般的に利用できる，4) 膨大なデータセットに拡張できるという優位性を持っていることが知られている [4].

[1] 学生会員　筑波大学 理工情報生命学術院 システム情報工学研究群
sushiama@kde.cs.tsukuba.ac.jp
[2] 正会員　NTT コミュニケーション科学基礎研究所
yasuhiro.fujiwara@ntt.com
[3] 正会員　筑波大学 計算科学研究センター
shiokawa@cs.tsukuba.ac.jp

上述の背景から，FINCH は高次元のデータセットに対する効果的なクラスタリング手法のひとつと考えられており，画像処理やコンピュータビジョン分野を中心に利用されている [6–8]．しかしながら，映像データのような高次元データストリームを扱う際に，FINCH は膨大な計算コストを必要とするという課題があることが知られている [9]．より具体的には，次のようなオンラインクラスタリング問題 [10–12] を考える．高次元なデータポイントが時々刻々と到着するデータストリームを想定し，これを入力とする．ある時刻において，それまでに到達したデータストリームを含めた全てのデータを考慮したクラスタリングの結果を効率的に求めたい．FINCH はデータの更新を想定していない手法であるため，データストリームを処理する場合にはデータが追加毎に近傍グラフを再構築し，階層クラスタを抽出する必要がある．近傍グラフ構築では既に到着したデータポイント全てに対する近傍探索処理が生じることから，FINCH はオンラインクラスタリングにおいて膨大な処理時間を必要してしまう．

この問題を解決するため，Cunningham らによって S-FINCH [9] が提案された．S-FINCH は，データが追加された時にクラスタ構造を全て再構築するのではなく，追加されたデータに応じてクラスタを差分更新することでクラスタリング時間を短縮する．このように，差分更新を導入することで，S-FINCH は FINCH と比較して高速なオンラインクラスタリングが可能とした．しかしながら，S-FINCH は大規模な高次元ストリームデータを対象としたクラスタリングにおいて多くの処理時間を必要とする．その具体的な理由は，近傍グラフの差分更新処理にある．S-FINCH はクラスタを差分更新するために，対応する近傍グラフの差分更新を行う必要がある．近傍グラフの差分更新では，新たに到着したデータポイントとそれ以前に到着している全てのデータポイントとの間においてユークリッド距離を計算する必要がある．すなわち，データストリームが配信されてから n 番目のデータポイントが配信された際には，距離計算において $\mathcal{O}(n)$ の時間計算量が生じる．また，S-FINCH は階層的クラスタリング手法であるため，データポイントの到着によって他の階層のクラスタ構造も変化する．この階層は高々 $\log_2 n$ であることから，この処理は $\mathcal{O}(n \log n)$ の時間計算量を要する．したがって，n 件のストリームデータをクラスタリング処理するために S-FINCH は $\mathcal{O}\big(n^2 \log n\big)$ 時間を要する．オンラインクラスタリングでは時間経過とともにデータポイントが蓄積され，n の値が増加する．ゆえに，S-FINCH は大規模なデータストリームに対して，依然として膨大な処理時間を要することになる．

1.1　本研究の貢献

本稿では高次元データストリームに対する高速なオンラインクラスタリングの実現を目的とし，S-FINCH に対して空間索引と次元削減を用いた高速化手法を提案する．提案手法では索引構造を導入することで，クラスタおよび近棒グラフの差分更新処理の効率化を図る．我々は先行研究 [13] において空間索引を用いた S-FINCH の高速化手法を提案した．この手法では低次元のデータストリームに対して，近傍グラフ構築で生じる距離計算回数を効果的に削減できることを明らかにした．しかしながら，一般的に，高次元データでは空間索引などの距離空間での利用を前提と

した索引構造は効果的ではないことが知られており，高次元のストリームデータに対しては十分な高速化が達成できないという課題があった．

そこで本稿の提案手法では，上述の空間索引構造に対して次元削減手法を併用することでこの課題の解消を図る．より具体的には，提案手法は高次元なデータセットに対して次元削減を行うことで，高次元データストリームを低次元空間へと射影する．低次元空間に射影されたデータストリームに対して上述の空間索引を構築することで，効果的に不要な距離計算を削減しつつ，クラスタおよび近傍グラフの差分更新の効率化を行う．

結果として提案手法は以下の性質を持つ．

高速な近傍グラフ更新： 近傍グラフを更新する際に必要となる距離計算回数を空間索引を用いることによって削減できる．その結果として，近傍グラフの更新に要する計算時間が短縮される．

高速なクラスタリング： 近傍グラフ更新が高速化されることにより，S-FINCH と比較して高速にクラスタリング結果の更新を行うことができる．

正確なクラスタリング結果： 提案手法は各データポイント間の距離を出来るだけ保存する次元削減手法を採用する．そのため，S-FINCH や FINCH と比較して同等の精度のクラスタリング結果を獲得することができる．

本稿の構成は，次の通りである．2 章で本稿の前提となる知識について概説する．3 章にて提案手法の詳細について説明し，4 章において提案手法の評価と分析を行う．5 章にて，本稿をまとめ，今後の課題について論ずる．

表 1: 記号と定義

記号	定義
N	データポイントの数
n	あるステップに与えられたデータポイントの数
d	データポイント x の次元数
κ_i^1	データポイント i の最近傍データポイント
$L(x)$	データポイント x のクラスタラベル
$parent(x)$	高さ i のデータポイント x のクラスタの高さ $i+1$ での代表データポイント
$children(x)$	$\{y \mid parent(y) = x\}$
$dist(v, w)$	データポイント v, w の距離
$min\_dist(v, W)$	データポイント v と空間 w の距離の最小値
$NN(x)$	x の最近傍データポイント
$NNset(x)$	x が最近傍であるデータポイント
$V_{h,sh,x}$	高さ h のデータポイント x の子孫のうち，高さ sh にあるデータポイントを包含する空間
$W_{h,x}$	高さ h のデータポイント x を中心とし，x の最近傍までの距離を半径とする超球
$S_{h,sh,x}$	高さ h のデータポイント x の子孫のうち，高さ sh にあるデータポイント y の $W_{h,y}$ を包含する空間

2 事前準備

この節では FINCH [4] とそのオンラインクラスタリング拡張手法である S-FINCH [9] について説明する．表 1 では本稿で用いる主な記号とその定義を示す．

FINCH [4] は d 次元のデータポイントが N 件ある多次元データセット $\boldsymbol{S} \in \mathbb{R}^{N \times d}$ を入力として受け取る．これに対して，S-FINCH は入力として d 次元のデータポイントが N 件連続している多次元データストリームを $\boldsymbol{L} = [L_1, L_2, \ldots, L_N]$（ただし，$L_i \in \mathbb{R}^d$）を受け取る．距離尺度として，FINCH および S-FINCH はユークリッド距離を用いる．すなわち，2 つのデータポイントのユークリッド距離が小さいとき，それらは類似したデータポイントであることを意味する．多次元データポイント v, w があるとき，これらの距離を $dist(v, w)$ を表記する．加えて，多次元空間 X があるとき，v から X までの距離の最小値を $min\_dist(v, X)$ を表記する．データポイント x の最近傍とは，x に最も近いデータポイント y のことを指し $NN(x) = y$ と表記するこれに対して，データポイント x の被最近傍とは，x が最近傍であるデータポイント集合 $NNset(x)$ のことを指し $NNset(x) = \{y \mid NN(y) = x\}$ 定義する．

FINCH [4] および S-FINCH はクラスタリング結果として階層的なクラスタ構造を出力する．そしてこのクラスタ構造の階層の数，つまり階層の高さを H とする．本論文ではこのクラスタ構造の一番下の階層を高さ 1 と，一番上の階層を高さ H とする．それぞれの手法に入力として与えられたデータの各データポイントはまず高さ 1 の階層でクラスタを構成することになる．ここで入力として与えられた各データポイントは 1 つのデータポイントからなるクラスタとして扱う．これにより，高さ 1 の階層ではクラスタ構造のデータポイントは全てクラスタの代表点となる．最上層ではない階層においてデータポイント x がクラスタ X_3 に属するとき，クラスタ X_3 のことを x の親と呼び，$parent(x) = X_3$ と表記する．また，x はクラスタ X_3 の子となり，$x \in children(X_3)$ と表記する．データポイント x の子となるデータポイント $children(x)$ や，さらに $children(x)$ の子となるデータポイント $children(children(x))$ 全てを子孫データポイントと呼ぶ．すなわち，データポイント x の子孫となるデータポイントの集合を $descendants(x)$ とするとき，$children(x) \in descendants(descendants)$ であり，さらに $descendants(children(x)) \in descendants(x)$ となる．データポイント x の親となるデータポイント $children(x)$ や，さらに $children(x)$ の親となるデータポイント $children(children(x))$ 全てを祖先データポイントと呼ぶ．ここで，データポイント x の祖先となるデータポイントの集合を $ancestor(x)$ とするとき，$children(x) \in ancestor(ancestor)$ であり，さらに $ancestor(children(x)) \in ancestor(x)$ となる．

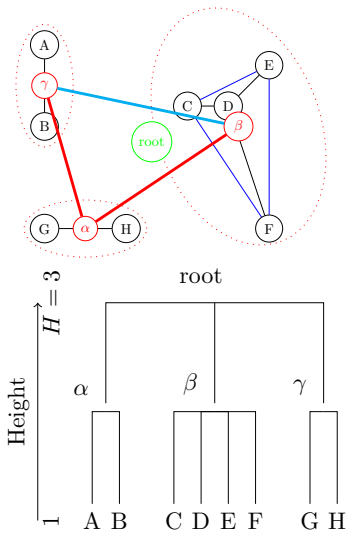

図 1: FINCH でクラスタを生成するときのグラフと対応するデンドログラム

2.1 FINCH

FINCH は， Sarfraz らによって提案された出力するクラスタ数やハイパーパラメータの指定を必要とせず，データポイントの近傍関係を利用して凝集型階層的クラスタリングを行う手法である．FINCH は与えられたデータセットからグラフを生成し，各連結成分を 1 つのクラスタとすることを繰り返しクラスタ階層を生成する．生成されたクラスタの数が 1 ではないとき，クラスタの構成要素となる各データポイントの平均，つまり重心を代表点とする．この各クラスタの代表点の集合を次の処理でのデータセットとする．与えられたデータセットをクラスタリングすることを 1 つのステップとし，ステップを繰り返すことで階層的なクラスタ構造を生成する．

次に連結成分を求めるために生成するグラフについて説明する．データセットの各データポイント間の距離を計算し，最近傍を求め，式 (1) で求まる隣接行列からグラフを生成する．

$$A(i,j) = \begin{cases} 1 \; if \; j = NN(i) \quad or \; i = NN(j) \\ \qquad\qquad\qquad\quad or \; NN(i) = NN(j) \\ 0 \; Otherwise \end{cases} \tag{1}$$

入力例とグラフの例を図 1 に示す．A から H は FINCH に入力として与えられる各データポイントを，α から γ は次のステップにおいて入力として与えられる各クラスタの構成要素の平均データポイントを示す．データポイント C, E, F を互いに結ぶ青い線，及びデータポイント α, γ を結ぶ水色の線は最近傍関係にはないものの，同じデータポイントを最近傍として持つデータポイントの間に張られるエッジを示している．それを除く赤と黒の実線は最近傍関係に基づいて張られるエッジを示している．例えば，データポイント A の最近傍はデータポイント B であり，データポイント γ の最近傍はデータポイント α である．これは式 (1) において $NN(i) = NN(j)$ となる場合に該当する．

FINCH の時間計算量を解析するために，各ステップでの時間計算量を求める．あるステップの入力として n 件のデータポイントがある場合, 最近傍探索に $\mathcal{O}(n^2d)$，連結成分の探索に $\mathcal{O}(n)$ の時間計算量が必要となり，1 ステップに $\mathcal{O}(n^2d)$ の時間計算量を要する．各データポイントはそれぞれの最近傍と同じの連結成分に含まれ 1 つのクラスタとなるため，ステップごとに入力となるデータセットのサイズは半分以下になる．そのため，FINCH に与えられた N 件の入力データセットに対して階層の数は高々 $\lceil \log_2 N \rceil$ となり，全体としての時間計算量は $\mathcal{O}(N^2 d \log N)$ となる．

2.2 S-FINCH

FINCH における逐次的なデータ更新を行うストリーム処理を達成することを目的に，近年 Cunningham によって S-FINCH が提案された．FINCH はデータの追加に対応できるクラスタリング手法ではないため FINCH を愚直にストリーム処理に対応させると，データが追加される度に過去のデータも含め全体のデータセットに対して FINCH を実行する必要があり $\mathcal{O}(N^3 d \log N)$ の 時間計算量を要する．そこで S-FINCH では，データが追加される前のクラスタ構造を用いることで時間計算量を大きく改善する．

S-FINCH は 3 つのステージで動作する．新しいデータポイント x が追加されたとする．x の最近傍データポイントを q，x が追加されたことにより x が最近傍になったデータポイント，つまり x にとっての逆最近傍データポイント集合を S とする．ステージ 1 で x に対する，q, S を求める．q, S に合わせてグラフも更新する．ステージ 2 で x のクラスタラベル L_x を更新する．ステージ 3 で連結成分になんらかの更新があったクラスタの代表点を更新し，次の階層のクラスタの更新を行う．この 3 つのステージを下の階層から順に実行することで新しく追加されたデータポイントも含めた FINCH と同じクラスタ階層が得られる．

ここで，ステージ 1 では x と直前までに追加された各データポイントとの距離を計算する必要があるため，x が追加される前までの各データポイントの個数を $n-1$ とすると $\mathcal{O}(nd)$ の時間計算量を要する．ステージ 2 では q, S の最近傍の更新により連結成分が更新されたクラスタを高々 $n-1$ 個の探索で求めることができるため，$\mathcal{O}(n)$ の時間計算量を要する．ステージ 3 ではステージ 2 で探索したクラスタのラベルを保持しておくことで解決される．階層構造の高さは高々 $\log_2 n$ であるため，1 つのデータ追加に $\mathcal{O}(nd \log n)$ の時間計算量を要する．この手順を図 2 に示した．これにより S-FINCH は $\mathcal{O}(N^2 d \log N)$ の 時間計算量となるため，FINCH に比べて 時間計算量を改善できる．しかし，S-FINCH は各データの追加やクラスタが変更後の更新のときに既存の全各データポイントとの距離を計算する必要がある．これにより追加されたデータ量が増えるにつれて膨大な時間を要することが問題となっている．

3 提案手法

本節では提案手法について述べる．提案手法は高速な S-FINCH 処理のために高次元なデータストリームに対して空間索引と次元削減を併用する．まず 3.1 節で提案手法の基本的なアイデアについて述べ，その詳細を 3.2 節から 3.4 節で説明する．

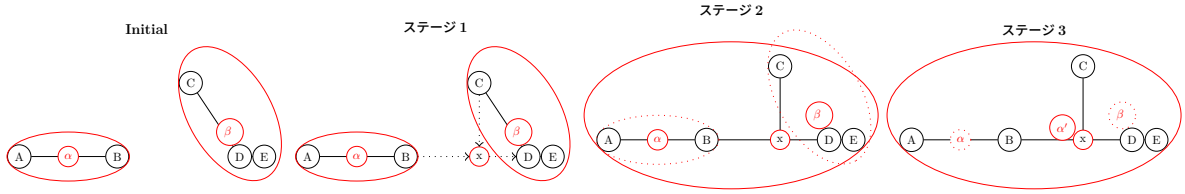

図 2: S-FINCH の詳細

3.1 基本アイデア

本研究の目的は高速・高精度に高次元なデータストリームに対してクラスタリングを行うことである．この目的のために，本稿では空間索引と次元削減を併用した手法を提案する．先行研究 [13] において，我々はは S-FINCH に対して空間索引を導入することで一定の高速化を達成したが，高次元のデータストリームに対しては依然として多くの計算時間が必要であった．この要因は高次元空間では同程度の距離を持つ多くのデータポイントが存在し得ることにある．多くのデータポイントが同程度の距離を持つことで，空間索引を用いても効果的に近傍グラフ構築を効率化することが難しくなる．そこで提案手法では高次元データストリームを低次元空間に射影することでこの問題を解消を図る．

このアプローチには以下の利点がある．第一に，提案手法は高次元なデータセットを次元削減するため，データの入力形式は従来と変わらない．したがって，ユーザは本手法を実行するために特別な前処理やコーディングを必要としない．第二に，次元削減によってオンラインクラスタリング処理で生じる距離計算コストを低減できる．結果として，提案手法は高次元データストリームに対しても高速かつ柔軟性の高いクラスタリングを可能とする．

以降では 3.2 節と 3.3 節において提案手法が用いる次元削減手法および空間索引の詳細について述べる．また，3.4 節において提案手法のアルゴリズムの全体像を示す．

3.2 次元削減手法

次元削減手法は一般的に線形アプローチと 非線形アプローチに大別できる．FINCH,S-FINCH はクラスタの形成に際して最近傍及び逆最近傍を求めており，最近傍及び逆最近傍探索の精度がクラスタリング性能に直結する．そのため次元削減前後での各データポイント間の距離は可能な限り保存されていることが望ましい．そのため非線形アプローチより線形アプローチが適切であると考え次元削減手法 PCA を採用する．

3.3 空間索引

S-FINCH の処理の高速化を目的として提案した S-FINCH に対する空間索引 [13] を用いる．この空間索引はデータセットの次元が高い場合に，低い枝刈り性能を示すこと指摘されていた．3.2 節で計算した低次元のデータを基に低次元空間索引を構築することで，距離計算の精度つまり最近傍及び逆最近傍探索の精度を犠牲にして距離計算回数の枝刈りを実現する．3.1 節で述べた通り空間索引は S-FINCH に対する空間索引を採用する．本研究で採用した空間索引について 3.3.1 項から 3.3.3 項で説明する．

Algorithm 1: 次元削減を用いた S-FINCH 空間索引

Input: S : 入力データストリーム
Output: L : クラスタ構造

```
begin
    while ストリーム S が停止していない do
        d ← ストリーム S のうち未処理のデータ
        d' ← REDUCE_DIMENSION(d)
        nn, rNN ← FIND_NN_rNN(d', index)
        L ← UPDATE_SFINCH(d', index)
        index ← UPDATE_INDEX(d', index)
```

3.3.1 空間索引の構築

データポイント x の最近傍を探索するにあたってデータポイント a, b を包含する空間 V をおく．$dist(a,x), dist(b,x) \geq min(dist(x,V))$ であることから，V との最小距離を計算することで V が包含する各データポイントの距離の下限を求められる．これにより最近傍を計算するにあたりデータポイント x との距離が最小距離以下であるデータポイントを既に求めている場合，データポイント x と V が包含するデータポイントの距離計算を省略できることがある．また V が小さいほど距離の下限が真の距離に近くなる．そのため空間索引の構築では小さな V を，低いコストで求めるために S-FINCH の持つクラスタ構造を利用する．あるクラスタのデータポイント全てを包含する球を求め，それを各データポイントに対する V とする．

3.3.2 空間索引の更新

空間索引はデータポイント a, b, c が 1 つのクラスタであるとき，これらの各データポイントを包含する球 V を持つ．S-FINCH はオンラインクラスタリング手法であるため，このクラスタにデータポイントが追加・移動・削除されることがある．ここで移動は，削除を行った後に追加を行うことで同等の処理を行うことができるため省略する．

まずデータポイント a が削除されたとき，クラスタの構成要素は b, c となる．ここで b, c を包含する球 V' を再計算し空間索引を更新する．次にデータポイント d が追加されたとき，クラスタの構成要素は a, b, c, d となる．ここで a, b, c, d を包含する球 V'' を再計算し空間索引を更新する．

3.3.3 空間索引の探索

データポイント x の最近傍を探索するにあたって，空間 V_1, V_2, V_3 があるとき x とそれぞれの空間の最小距離を計算する．$min(dist(x,V_1))$ が最小だったとき V_1 に対応するクラス

タの構成要素との距離を計算する．ただし，a, b, c が構成要素であり，a との距離が最小だったとき x の最近傍は暫定的に a となる．次に，V_2 との最小距離が a との距離より大きい場合 $(dist(x,a) < min(dist(x,V_2)))$ は V_2 に対応するクラスタの構成要素は最近傍になり得ない．そのため V_2 に対応するクラスタの構成要素との距離計算を省略できる．一方，V_3 との最小距離が a との距離より小さい場合 $(dist(x,a) > min(dist(x,V_3)))$ は V_3 に対応するクラスタの構成要素は最近傍になり得る．V_3 に対応するクラスタの構成要素と都度距離計算を行い a より近い場合は暫定的な最近傍を更新する．これを全ての空間に対して行うことで最近傍をもとめる．

3.3.4 空間索引の階層化

3.3.2 項までの手法はクラスタ構造の階層を考慮していない．しかし 3.3.1 項から 3.3.3 項でデータポイントと定義していた箇所を球とみなすことで再帰的に空間索引を定義できる．3.3.2 項に関しては 2.2 節と同様に階層的な更新を実行でき，3.3.3 項は逆に進むことで探索が可能である．これにより，S-FINCH のような階層的なクラスタ構造に対しても空間索引を構築できる．

3.4 アルゴリズム

アルゴリズム 1 に提案手法の流れを示す．`REDUCE_DIMENSION` は 3.2 節，`FIND_NN_rNN` は 3.3.3 項，`UPDATE_SFINCH` は 2.2 節，`UPDATE_INDEX` は 3.3.2 項での処理を指す．まずストリームが停止していない場合 (2 行)，ストリームに存在するデータのうち未処理のデータポイントを取得する (3 行)．次に，取得したデータポイントの次元を削減し (4 行)，3.3.3 項の空間索引を基に最近傍及び逆最近傍を見つけ (5 行)，2.2 節に従ってデータポイントとその最近傍及び逆最近傍で S-FINCH のクラスタ構造を更新する (6 行)．さらに 3.3.2 項に従って空間索引を更新する (7 行)．

4 評価実験

提案手法の S-FINCH に対する優位性を実行時間とクラスタリング精度の観点から評価する．

4.1 実験設定

4.1.1 実験環境

本実験には CPU Intel Xeon E5-2690 2.6 GHz, メモリ 128 GB の Linux サーバを利用する．実験に利用したプログラムは C++ で実装し，g++ (GCC) 9.2.0 で実行速度の最適化を目的に `O2` オプションをつけコンパイルした．

4.1.2 データセット

本研究では，表 2 に示す 5 種類のデータセットを用いる．文献 [14] で指摘されているように，データストリームクラスタリングのための高品質なベンチマークデータセットは限られている．そのため，文献 [14] によって挙げられている MNIST, STL-10, Forest Cover Type などの静的データセットもストリームデータとみなして評価に利用する．各詳細は次のとおりである．

MNIST [15] は 28×28 ピクセル画像の手書き数字 (0–9) からなるデータセットである．画像認識手法のベンチマークとして広く用いられている．STL-10 [16] は 10 種類の物体のクラスの画像からなるデータセットである．Gas Sensor Array Drift Dataset [17] はガスセンサのドリフト特性を含む実験データであり，36 ヶ月に渡り 6 種類のガスの識別を目的として 16 チャネルで測定したデータセットである．本研究ではこれを特徴量化した 128 次元の特徴ベクトルを採用した．化学センシング領域における機械学習手法の評価に適している．Relative location of CT slices on axial axis [18] は CT 画像のスライス位置を推定するタスクのデータセットである．医用画像解析における位置推定のベンチマークとして活用されている．Forest Cover Type [19] は地形や土壌などの情報から森林被覆タイプを分類する大規模データセットである．地理空間データの代表的なベンチマークとして知られており，オンラインデータストリームクラスタリング分野で最も一般的なデータセットである．

4.1.3 評価指標

空間索引による枝刈り性能の評価として同じ区間のデータを処理したときの処理時間を採用し，クラスタリング性能の評価として Normalized Mutual Information (NMI) [20] を用いる．

4.2 高速性

削減する次元数を変化させた際の処理速度とクラスタリング精度を評価する．また，ここではデータセットの全てのデータを入力と同時に次元削減を行い，その処理時間は全体の処理時間の 1% 以下であったため実験結果には含めていない．

図 3 に削減後の次元による処理時間を，表 3 に削減後の次元による NMI スコアを示す．横軸は削減後の次元数，縦軸は処理時間の合計を記載している．各データセットの最も右のプロットは全ての次元を採用したもの，つまり次元削減を行っていないオリジナルのデータで実験したものとなっている．Forest Cover Type については実行が終了しなかったため全ての次元においてデータストリーム長 50000 で打ち切り，打ち切る直前 1000 件についての時間の合計を計測した．

全ての例において，1 次元など極端に小さい次元数に削減すると空間索引の大幅な性能改善が確認できる．しかし，それに伴い NMI は全ての次元で実行した場合に比べ非常に低くなる．また，削減後の次元を増やしていくと空間索引の性能は悪化していくが，NMI スコアも改善している．図 3 の MNIST や STL-10 などのように次元を増やしていくと空間索引の性能が悪化し，S-FINCH より処理に時間を要するようになるデータセットも存在する．これは，ベースとなっている我々の先行研究 [13] が抱える問題と同様に，高次元において空間索引の構築および探索に要する距離計算回数の増加により性能が悪化することに起因している．MNIST や STL-10 は我々の先行研究においても高次元での性能悪化が顕著であったデータセットであり，今回の実験でも同様の傾向が見られたと考えられる．しかし，削減後の次元を同一にした場合は MNIST や STL-10 においても提案手法が従来手法より高速に動作している．これは，次元削減により超球ベースの空間索引による枝刈り性能の恩恵を受けやすくなったためと考えられる．そのため特に我々の先行研究を適用するだけでは従来手法の S-FINCH に劣る場合でも，次元削減を組み合わせることで提案手法の性能を最大限に引き出すことが可能であるといえる．これらより，最適な削減後の次元があるわけではないが，次元を削減すると空間索引の性能が改善するため，同等のクラスタリング性能を得ながら処理時間の削減が可能であるといえる．

表 2: 実験に用いたデータセット

Dataset Name	Number of Instance	Number of Features	Number of Clusters
MNIST	10,000	784	10
STL-10	13,000	2048	10
Gas(Gas Sensor Array Drift Dataset)	13,910	128	6
CT(Relative location of CT slices on axial axis)	53,500	386	74
FCT(Forest Cover Type)	581,012	54	7

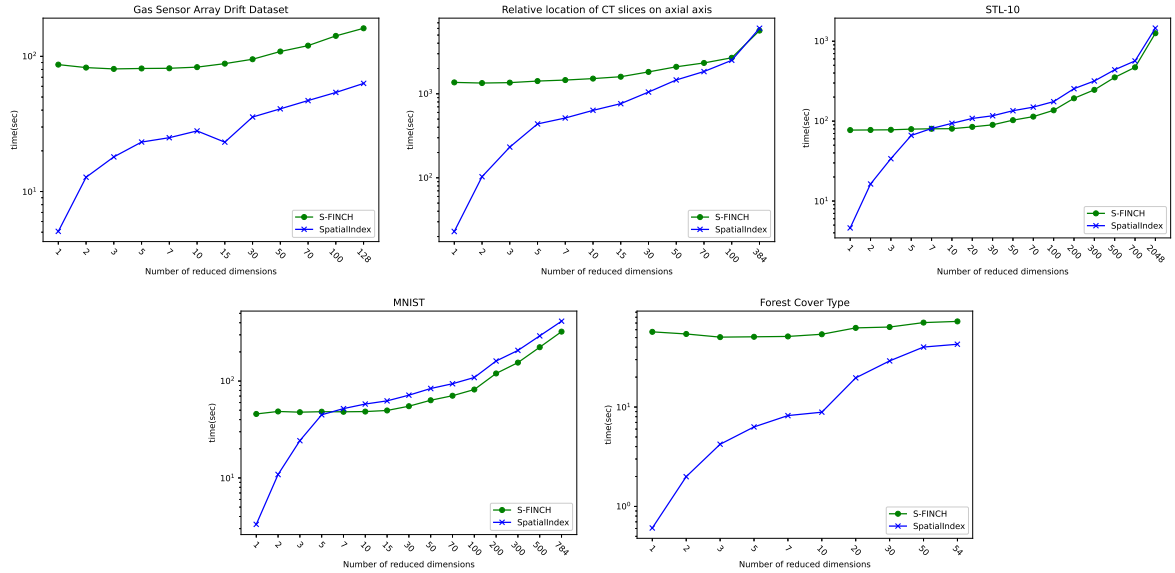

図 3: 削減後の次元数による処理速度の比較

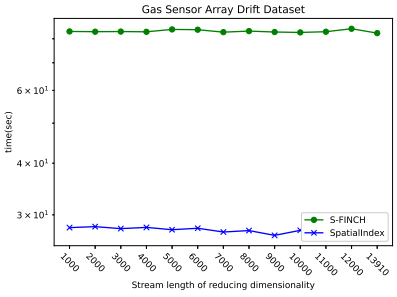

図 4: 次元削減のタイミングによる処理速度の比較

4.3 射影空間の評価

4.2 節 では，データセットの全てのデータを基に次元削減を行った．しかし，実際のオンラインクラスタリングでは到達しているデータのみで次元削減を行う必要がある．そこで，次元削減を行うタイミングを決定するために，次元削減のタイミングによる性能の評価を行う．そのため，データセットを分割し次元削減の射影空間を決定するためのデータを基に射影空間を決定し，その後に到達したデータは予め決められた射影空間に射影することで次元削減を行う評価する．

図 4 に，次元削減を実行した際のストリーム長さに対する処理時間を示す．図 4 からも明らかなように，次元削減おタイミングを変化させた場合でもオンラインクラスタリングに与える実行時間の変化は小さい．また，NMI スコアについても射影空間を決定したときのタイミングとの相関は見られず，最大値と最小値が 0.4703 と 0.4549 であることを確認した．この結果は図 3 の Gas Sensor Array Drift Dataset の NMI の次元削減した次元数による差よりは十分に小さく，射影空間を決定するタイミングは実行時間およびクラスタリング精度に大きく影響していないことがわかった．一方で，評価に用いたデータセットは時間変化い対する性質が均質であるため，性質の時間変化を伴うデータストリームに対する性能評価については今後の課題である．

5 おわりに

本稿では，高次元の大規模データストリームに対して高速に S-FINCH を実行する手法を提案した．提案手法では，S-FINCH に対する空間索引 [13] が課題としていた高次元なデータストリームに対しても効果的な計算コスト削減を実現するために，空間索引と次元削減を併用する手法を示した．本稿では実データを用いた評価実験を通じて，提案手法が既存手法よりも，最大 9 倍高速にオンラインクラスタリング処理が可能であることを確認し

表 3: 削減後の次元数によるクラスタリング性能の比較

	MNIST	STL-10	Gas	CT	FCT
1	0.3998	0.4376	0.3122	0.5687	0.2016
2	0.3998	0.5398	0.3149	0.5687	0.2016
3	0.3998	0.628	0.3822	0.5687	0.2016
5	0.4144	0.7057	0.421	0.6034	0.2016
7	0.5237	0.7587	0.4643	0.6311	0.2016
10	0.5799	0.7904	0.4703	0.6493	0.2016
15	0.6275	-	0.4474	0.6571	-
20	-	0.7655	-	-	0.2016
30	0.7107	0.7946	0.4845	0.6688	0.2016
50	0.7127	0.837	0.4847	0.6797	0.2016
70	0.6535	0.8427	0.4973	0.6869	-
100	0.7102	0.8465	0.4837	0.6899	-
200	0.6678	0.8134	-	-	-
300	0.7137	0.7988	-	-	-
500	0.711	0.8062	-	-	-
700	-	0.8043	-	-	-
次元削減無し	0.7269	0.8024	0.4845	0.6975	0.2016

た．また，提案手法は次元削減を導入しているが，クラスタリング結果の誤差は従来手法と比較して 5% 未満に抑制されていることも実験的に明らかにした．

一方で，提案手法が導入している次元削減手法にはいくつかの課題がある．例えば，提案手法ではクラスタリング処理の前に次元削減処理を導入しているが，これはデータストリームの性質が均質であることを前提としている．したがって，データストリームの性質が時間とともに動的に変化する場合では，次元削減の効果が低下し，クラスタリング精度の面において性能低下をもたらす可能性がある．このような問題への対処として，PCA の差分更新手法 [21] といったデータストリームに対応した次元削減手法の活用が考えられる．しかし，これらの手法の導入は現在の提案手法と比較して多くの計算オーバーヘッドをもたらすため，同手法の導入に際しては実践的なデータセットを用いた検証が必要である．これらの検証も含めた，データストリームの性質変化に対する検証や対応手法の検討は今後の課題のひとつである．

謝辞

本研究の一部は JST AIP 加速課題 JPMJCR23U2 の支援を受けたものである．

参考文献

[1] Hiroaki Shiokawa, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Scaling Fine-grained Modularity Clustering for Massive Graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pp. 4597–4604, 2019.
[2] Hiroaki Shiokawa. Scalable Affinity Propagation for Massive Datasets. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021)*, Vol. 35, No. 11, pp. 9639–9646, May 2021.
[3] Shohei Matsugu, Yasuhiro Fujiwara, and Hiroaki Shiokawa. Uncovering the Largest Community in Social Networks at Scale. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pp. 2251–2260, 2023.
[4] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8926–8935, June 2019.
[5] James MacQueen, et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, pp. 281–297. Oakland, CA, USA, 1967.
[6] Yan-Lin He, Lei Chen, Yuan Xu, Qun-Xiong Zhu, and Shan Lu. A new distributed echo state network integrated with an auto-encoder for dynamic soft sensing. *IEEE Transactions on Instrumentation and Measurement*, Vol. 72, pp. 1–8, 2023.
[7] Yu-Ting Chang, Qiaosong Wang, Wei-Chih Hung, Robinson Piramuthu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation via sub-category exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
[8] M. Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11220–11229, 2021.
[9] James Cunningham, Jim Davis, Kyle Tarplee, and Juan Vasquez. S-finch: An optimized streaming adaptation to finch clustering. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pp. 1343–1349, Aug 2022.
[10] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, Vol. 1, No. 2, pp. 141–182, Jun 1997.
[11] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. *Density-Based Clustering over an Evolving Data Stream with Noise*, pp. 328–339. SIAM, 2006.
[12] Charu C. Aggarwal, Philip S. Yu, Jiawei Han, and Jianyong Wang. - a framework for clustering evolving data streams. In Johann-Christoph Freytag, Peter Lockemann, Serge Abiteboul, Michael Carey, Patricia Selinger, and Andreas Heuer, editors, *Proceedings 2003 VLDB Conference*, pp. 81–92. Morgan Kaufmann, San Francisco, 2003.
[13] 牛尼索造, 藤原靖宏, 塩川浩昭. 大規模データストリームに対する高速な s-finch クラスタリング. 第 16 回データ工学と情報マネジメントに関するフォーラム (DEIM 2024), 4 2024.
[14] Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering: a review. *Artif. Intell. Rev.*, Vol. 54, No. 2, p. 1201–1236, February 2021.
[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.
[16] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Vol. 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.
[17] Alexander Vergara. Gas Sensor Array Drift Dataset. UCI Machine Learning Repository, 2012. DOI: `https://doi.org/10.24432/C5RP6W`.
[18] F. Graf, H.-P. Kriegel, M. Schubert, S. Poelsterl, and A. Cavallaro. Relative location of ct slices on axial axis. UCI Machine Learning Repository, 2011. DOI: https://doi.org/10.24432/C5CP6G.
[19] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: `https://doi.org/10.24432/C50K5N`.
[20] sklearn.metrics.normalized_mutual_info_score.
[21] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The Fast Convergence of Incremental PCA. In *Advances in Neural Information Processing Systems (NeurIPS 2013)*, Vol. 26, 2013.