一般社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-057-3 (11/20)

オンライン音環境認識のための低次元埋め込み手法の高速化

Acceleration of Low-dimensional Embedding Method for Online Auditory Scene Analysis

新里 顕大[1*]　小島 諒介[2]
Kenta Shinzato[1]　Ryosuke Kojima[2]

[1] 京都大学工学部情報学科数理工学コース
[1] Undergraduate School of Informatics and Mathematical Science, Kyoto University
[2] 京都大学大学院医学研究科ビッグデータ医科学分野
[2] Department of Biomedical Data Intelligence, Graduate School of Medicine, Kyoto University

Abstract: Summarizing and visualizing auditory scenes are essential tasks for environmental monitoring applications. In this paper, we address the mapping of sound events into low-dimensional space by unsupervised manner considering long-term recordings. Because recorded data is accumulated continuously or periodically in long-term recording data analysis, computational costs and the incremental nature of the analysis methods are significant. We propose an unsupervised auditory scene analysis system, which uses an incremental low-dimensional embedding technique SONG for visualization. To realize real-time analysis by using this method, we improve the implementation of SONG algorithm by accelerating the algorithm utilizing just-in-time compile and matrix computation techniques. In this study, we investigated the bottleneck of this algorithm by the profiling and quantitatively evaluated the effectiveness of our accelerated implementation.

1　はじめに

長期的な環境モニタリングは，継続的な環境調査や動物の生態調査において重要である．特に，環境中の音をマイクロフォンで収録・分析することは，視界の悪い環境など視覚ベースのセンサが効果的でないような場合にも有効であることから注目されている．一方で，実環境には様々なノイズが含まれるため，収録した音から有用な音の情報を分離・抽出することは重要な研究課題である．特に音環境認識では，複数のマイクロフォンからなるマイクロホンアレイを用いることでより多くの情報を抽出することが可能である [1]．マイクロホンアレイを用いた研究は，ロボット聴覚の分野で多くなされており，例として Voice Activity Detection (VAD), Sound Source Localization (SSL), Sound Source Separation (SSS), Sound Source Identification (SSI) などの技術が挙げられる [2, 3]．VAD, SSL, SSS は古くから信号処理の分野で研究されており，教師あり・教師なし機械学習によるアプローチも研究されている．一方で，SSI に関しては教師あり機械学習の手法を用いることが一般的である．しかし，教師ありアプローチでは, 対象音源が明確にカテゴリ分類されている必要があり，それらの教師データは予めアノテーションをする必要があり，これを人間が行う場合には，その労力や正確性が問題となる．

近年，事前にカテゴリ分類がされていない，あるいは曖昧な音源を認識する手法として低次元埋め込みの手法が注目されている [4, 5, 6]．低次元埋め込みでは，高次元のベクトルを低次元，特に可視化の用途では 2 次元や 3 次元のベクトルとして表現することを考える．低次元埋め込みを使った可視化は既に，バイオインフォマティクスやデータサイエンスの分野などで広く応用されている [7, 8]．

我々は，野生生物のモニタリングや調査を想定した低次元埋め込みを用いた音環境モニタリングシステムを提案する．野生生物の音環境モニタリングでは，環境変化の調査や希少な事象の調査のために長時間の記録が欠かせないが，このような用途では，インクリメンタル性が重要である．インクリメンタルな手法を考える理由の一つは長時間の録音では季節や時間帯の変化などにより環境が動的に変化することで， 過去のデータには存在しない，新たな音源クラスが生じることを

*連絡先： 京都大学工学部情報学科数理工学コース
京都市左京区聖護院川原町 54
E-mail: sinzato.kenta.82r@st.kyoto-u.ac.jp

考慮する必要があるためである．加えて，長時間の連続録音では録音された音声データのサイズが膨大であるため，全データを一度に処理し，解析することは現実的ではない．また，インクリメンタルな処理を実現することで，重要なイベントのみをデータとして蓄積するといった応用も可能である．

低次元埋め込みアルゴリズムの進歩は目覚ましく，特に t-SNE [5] や UMAP [6] は，データの可視化，すなわち二次元への埋め込みを行うためのスタンダードな手法となっている一方で，従来手法の多くは，インクリメンタルに設計されていないため，新たなデータが与えられた際にすべてのデータについて再計算を行わなければならない．近年，低次元への埋め込みをインクリメンタルに構築できる Self-Organizing Nebulous Growths (SONG) [9] が提案され，これにより，t-SNE や UMAP に匹敵する可視化がインクリメンタルに可能であると報告されている．

長時間録音に対しインクリメンタルな音環境認識のシステムを構築するにあたり，求められる性能としてインクリメンタル性に加え, リアルタイム性がある．インクリメンタルな音響解析では，収録した音の長さと同程度の処理時間で低次元埋め込み処理が実行できることが理想的なので， システムの各処理は高速に動作することが求められる．本研究では，音環境認識を想定した埋め込みアルゴリズムの各処理に要する時間の分析を行い，実行速度面で改善が可能である部分に関し最適化を施し，評価を行う．

2 収録音の低次元埋め込みおよび可視化

低次元埋め込みを用いた環境音モニタリングシステム [10] は前処理・特徴抽出と可視化のための低次元埋め込みの 2 つのステップから構成される．まず，2.1 章で収録データから特徴量を抽出する方法について述べ，2.2 章でインクリメンタルな低次元埋め込みおよび可視化のための SONG 法について述べる．

2.1 前処理

我々は，これまでにインクリメンタルなマイクロホンアレイで収録を行い，定位，分離，可視化を行うデータフロー的な設計を行ったシステムを開発した [10]．本研究では，このシステムに組み込むことを想定し，特に，低次元埋め込みとその前処理部分に関する検討を行う．以降では分離後の音データから特徴量を計算する方法について述べる．

まず，音のフレーム毎の音響特徴量を抽出する．本稿では，メルスペクトログラムによる音響特徴量抽出を利用する．メルスペクトログラムは，メルスケーリングされたスペクトログラムであり，本研究で用いる特徴量はメルスペクトログラムの 2 つの隣接フレーム間の差のデルタ特徴とそのさらに差分の二重デルタ特徴を合わせて用いる．その後, スライディングウィンドウを利用して複数のフレームに関連付けられた特徴を構造化する．本稿における実験では, 10 フレームの窓を使用しているため, 最終的な特徴ベクトルは各フレームの特徴ベクトルの 10 倍の長さになる．本稿では, このスライディングウィンドウによって計算されたベクトルを「サンプル」と呼ぶことにする．実際の次元数や詳細は 4.2 章で述べる．

各サンプルの低次元埋め込みを行う前に，上述の特徴ベクトルに対し正規化と次元削減を行う．これらの前処理は低次元埋め込みを効果的に計算するために経験的に重要である [5]．我々のプロトタイプの設計では，PCA (Principal Component Analysis) を用いて特徴ベクトルを 20 次元のベクトルに削減している．この PCA によってパラメータを得る過程はインクリメンタルではないが，インクリメンタルな設計の PCA [11] を利用することで，真にインクリメンタルなシステムを構築できる．

2.2 SONG アルゴリズムの概要

Self-Organizing Nebulous Growths (SONG) [9] は，t-SNE や UMAP 等と同等に高次元のデータを低次元空間に埋め込むアルゴリズムであり，主に高次元でのデータを可視化するために用いられる． これらのアルゴリズムではデータ間の距離に基づいて，類似するデータ同士が低次元空間上に射影された際にも近距離になるよう埋め込みを行う．SONG は t-SNE や UMAP とは異なり，パラメトリックな手法と呼ばれている．SONG では埋め込みを行う際にパラメータを利用しており，一度埋め込みを計算した後に新たにデータを追加した場合にも，そのパラメータを用いて新たな埋め込みが容易に追加可能というインクリメンタルな性質を備えている．

SONG アルゴリズムでは，入力次元のユークリッド空間中にコーディングベクトル（以下 CV と呼ぶ）と呼ばれる点を複数用意し， データのクラスターを代表する点として扱う．2 つの CV の間には非負の値として関連度を設け，ある入力データの k-近傍以内の CV 同士の関連度を高くし，入力データの近くにそのデータに関連付けられた CV を移動させることで，関連する CV 同士がより密になるようにする．逆に，着目データ点の k-近傍でない CV と最近傍の CV とは関連度を下げる．このように，関連度の高い CV 同士が近くなり，関連しない CV 同士は離れるように CV を配置し，

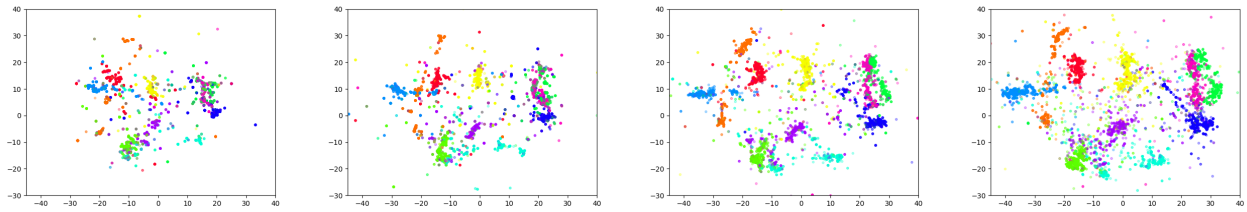

図 1: SONG 法により MNIST データが可視化される様子. 左から順に 10, 20, 50, 100 エポック後の埋め込み.

各 CV から低次元空間の点への写像を構成することで, 低次元のへの埋め込みを実現する.

より具体的には, SONG アルゴリズムは以下の 4 つのステップから構成され, それらを繰り返すことで低次元のへの埋め込みを行う. 初期時では, CV および埋め込み点はランダムに初期化されている.

ステップ A 入力データからランダムに一点選び $\boldsymbol{x}$ とする. $\boldsymbol{x}$ の 最近傍 の CV である $\boldsymbol{c}'$ と, その他の CV との間の関連度を更新する. 具体的には, $\boldsymbol{x}$ の k-近傍以内の点と $\boldsymbol{c}'$ との関連度を 1 にし, それ以外の点とは関連度を $\varepsilon(<1)$ 倍し, さらにその操作によって関連度が閾値以下になった場合は関連度を 0 にする.

ステップ B ある コーディングベクトル $\boldsymbol{c}$ との関連度が 0 ではない CV の集合を $R(\boldsymbol{c})$ で表し, これらを $\boldsymbol{c}$ と関連する点と呼ぶ. $R(\boldsymbol{c}')$ の各点を $\boldsymbol{x}$ に近くなるように移動させる. この際, $\boldsymbol{x}$ の近傍の点ほど大きく移動させることで関連度の情報を崩さないようにする. これには適切な損失関数を設定し, 最急降下法で最適化する.

ステップ C $\boldsymbol{c}'$ に対応する低次元空間の埋め込み点を $\boldsymbol{y}'$ とする. $\boldsymbol{y}'$ とその他の埋め込み点の位置を関連度の情報をもとに更新する. 具体的には, $\boldsymbol{y}'$ と, $R(\boldsymbol{c}')$ に対応する埋め込み点との距離が関連度に応じて近くなるようにする. 一方で $\boldsymbol{c}'$ と関連しない点 (CV の $R(\boldsymbol{c}')$ の補集合 $\overline{R(\boldsymbol{c}')}$ で表される点) からランダムにサンプリングしたコーディングベクトルに対応する埋め込み点と $\boldsymbol{y}'$ は離れるように, それぞれ適切な損失関数を設定し, 最急降下法で各埋め込み点の位置を最適化する.

ステップ D CV の数が不足している場合には, $\boldsymbol{x}$ と $\boldsymbol{x}$ の k-近傍の CV の重心座標に新たな CV を追加し, 同様に埋め込み点も追加する. この操作は, 各 $\boldsymbol{x}$ に対し $\|\boldsymbol{x}-\boldsymbol{c}'\|$ の累積が閾値を超えた場合に行う.

このステップ A-D を, 全入力データに対し行う操作を 1 エポックと数える.

我々は SONG の提案論文 [9] を元に, 独自に NumPy を用いた Python 実装を行い, これを公開している[1].

3 SONG 法の分析と高速化

ここでは, SONG 法のナイーブな実装 (以降, 初期実装と呼ぶ) とその改善した実装に関して, SONG 法の各処理における計算時間の分析を行う.

初期実装とボトルネックについて改善を施したコードに対しプロファイルをとり, 各処理に要する累積時間が全体に占める割合を表した結果が図 2 である. 図中の A-C は 2.2 章の各ステップと対応する. A-1 は, ステップ A において $\boldsymbol{x}$ の k-近傍のコーディングベクトルを探索する関数の累積時間の割合を示す. A-2 は, ステップ A において $\boldsymbol{c}'$ とその他のコーディングベクトルとの関連度を更新するのに要した時間の割合を表す. 速度の計測実験は, MNIST データセットから 10,000 個サンプリングしたデータの次元を PCA で 20 次元に削減し, ノルムの正規化を施したものに対して, SONG 法を 100 エポック適用するという条件の下で行った.

実行速度においてボトルネックだった部分を解消するために, 我々は 1) 最急降下法の実装の改善 2) 実行時コンパイラ (Jsut-In-Time Compiler) による高速化を行い, 速度の向上を検証した.

以下では, 本研究で実施した実装の改善について述べる. 2.2 章のステップ B, C において, 初期実装では各 $R(\boldsymbol{c}')$ および $\overline{R(\boldsymbol{c}')}$ の要素に対して一つずつ最急降下法の処理を実行していたが, このプロセスを行列演算として実行することにより, ボトルネックであったステップ B, C の速度を改善するに至った. その結果が図 2 の SGD optimization に相当する. また, 行列計算化した最急降下法の処理を Numba [12] を用いた実行時コンパイラを適用することで, わずかな速度向上が見られた. この時, 各実行時間の割合は図 2 の JIT に示した通りであった. このとき, ステップ A については特に改善を施していないため変化は見られないが, ステップ B については 9 倍の速度向上がみられた.

[1] https://github.com/hoppiece/song

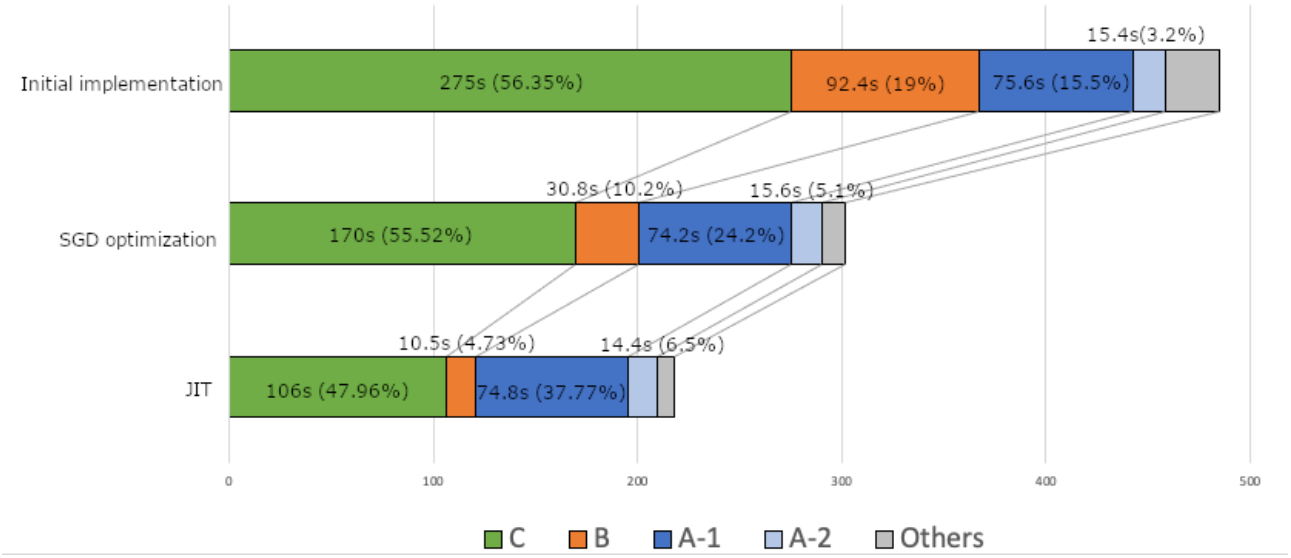

図 2: 各実装における SONG の各ステップに要した累積時間

本手法によって，全体でおおよそ 2 倍の高速化が実現できたが，SONG 法は UMAP と類似の操作で構成されており，実行時コンパイラによってさらに高度な高速化を行うと UMAP と同様にさらに高速化可能であると考えられる．この SONG 法のさらなる高速化法については 5 章で考察する．

4 実験

4.1 MNIST による評価

MNIST データセット [13] において， エポックごとに埋め込みが自己組織化される様子が図 1 である．我々の以前の研究 [10] では, SONG 法による MNIST データセットの埋め込みと，その他の埋め込み手法に対し比較を行った．

4.2 SONG 法による音響データの可視化

本システムを評価するために鳥の歌の音声データセットを用意し，音響特徴量を用いた低次元埋め込みを行う．まず，既報論文で利用されているものと同様のデータセットを構築する [4]．単一のマイクで録音された鳥の鳴き声のデータセットが `http://taylor0.biology.ucla.edu/birdDBQuery/` [14] から利用可能である．用いたデータセット内には 4 種の鳥の鳴き声を含む 645 の録音ファイルが収録されている．このデータセットには，鳥の鳴き声に関する音節のセグメンテーションとアノテーションが含まれているが， 本研究では音響特徴の埋め込みに焦点を当てているため，セグメンテーションは正しく行われているものとしてデータセット中のセグメントを用いた．また，音響特徴を抽出するには短すぎるセグメントは削除した．このデータセットから抽出された音響データの大部分は 50 フレーム未満であったので，50 フレームのデータを用いた．最終的に，こうして得られた音データから，2.1 章に記したメルスペクトラムを用いた特徴ベクトルを 349386 個のベクトルを用意した．

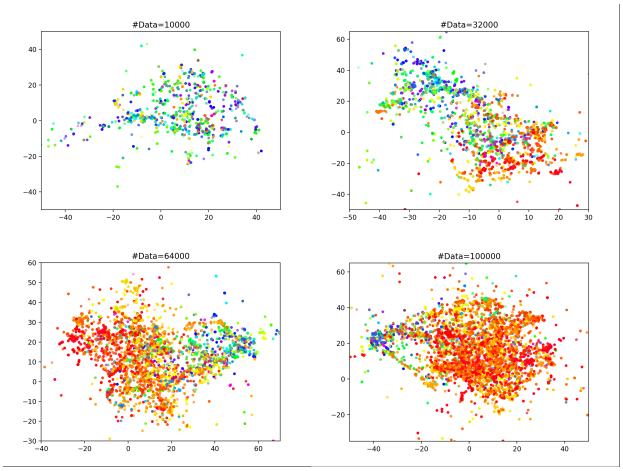

図 3: 鳥の鳴き声データの SONG 法による可視化．色は音節の種類を表す．左上，右上，左下，右下の順にデータ数が 10000, 32000, 64000, 100000 に対応する．

このデータから実行速度を評価するために，N 個のベクトル $(N = 10000, 32000, 64000, 100000)$ を取り出して，SONG 法を用いて可視化したものが図 3 となる．

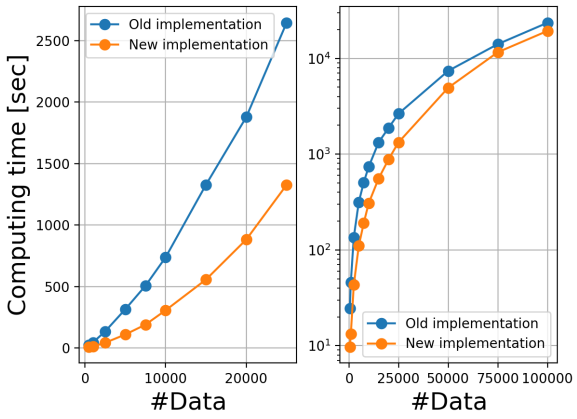

図 4: 入力データ数に応じた計算時間

上述のデータセットに関して，データ量に対しどのように増加するかを計測した結果が図 4 となる．実装は図 2 の JIT のものを用いた．データ数が 10^4 程度まではデータ数に対し線形よりやや悪いペースで実行時間が増加しているが，その後，計算時間が悪化している．その原因を分析するため様々なデータ数に対してプロファイルをとったところ， データ数が 10^4 未満の場合図 2 の JIT と同様の累積計算時間の割合を示したが，データ数が $30,000$ 程度からは，ステップ A の処理に大きく時間がかかっていることがわかった．データ数 50,000 の際の際にプロファイルをとったところ, 全体に要した時間 9770s に対して，ステップ A-1(近傍探索) が 6980 sec (70.2%) を占め，ステップ C が 2520 sec (25.31%)，ステップ B は 104 sec (1.04%)，ステップ A-2 は 181 sec (1.82%) となった．このことからデータ数が大きい際の近傍探索の速度を改善が今後の速度改善に関して新たな課題であることが分かったが，これについては 5 章で考察する．

5 考察

今回，特に基本的かつ効果的な 2 つの高速化を行った．SONG 法についてさらなる高速化の方法として次の 2 つを適用することが考えられる．

1. バッチ処理あるいは並列化による高速化
2. 近傍探索手法の高速化

の 2 つである．

バッチ化 (並列化) による高速化　有効な埋め込みアルゴリズム高速化の手法として，バッチ処理による高速化が考えられる．この方法では, 主にステップ B およびステップ C の計算に要する時間を改善することができる．2.2 のアルゴリズムでは，入力データ $\boldsymbol{x}$ を一つずつ選んで処理しているが，この処理をバッチ化し，複数のデータをまとめて扱うことで行列計算ライブラリの恩恵を受けることができる．しかし，$R(\boldsymbol{c}')$ の要素数が $\boldsymbol{x}$ ごとに異なるので，単純なテンソル計算に落とし込むには工夫が必要である．

類似のアプローチとして，バッチ内の $\boldsymbol{x}$ ごとに並列的に処理を行うという手法が考えられる．入力データが多い際に各 $\boldsymbol{x}$ を独立に処理できるので，並列化を用いた手法は計算機の並列演算性能に応じて高速化可能である．近年のニューラルネットワークの飛躍的な性能向上には GPU コンピューティングの利用が大きく寄与しているが，t-SNE や UMAP などの埋め込みアルゴリズにおいても GPU を利用した高速化の研究がされており [15, 16]，特にサンプル数が大きい ($> 10^5$) 場合に，既存の実装の 100 倍以上の高速化を実現している．SONG 法においても並列処理に GPU コンピューティングを用いることで，インクリメンタルな可視化の飛躍的な速度向上が期待できる．

近傍探索の高速化　SONG の高速化の方法として，図 2 での実験では全体の 37.7%，さらにデータ数 50,000 の際には全体の 70.2% を占める近傍探索の改善が重要である．2.2 節のアルゴリズム中のステップ A では, 入力データ近傍のコーディングベクトルを探索している．この近傍探索の速度を改善することは, 大規模なデータを可視化する際に重要であるほか，同様の改善は UMAP をはじめとした近傍探索を行う多くの埋め込み手法の改善にも応用可能である．

計算量に関して考えると，入力データ数を N，入力データの次元を D として，コーディングベクトルは入力データを代表する点のため，データ数に応じて増加することが経験的にわかっているので， コーディングベクトルの数を $O(N)$ と考えてもよい．

初期実装は線形探索により，クエリ点とすべてのコーディングベクトルとの距離を計算し，最小となるものを探索しているため, $O(ND)$ の計算量を要している．およそ $D < 30$ 以下の場合には，ユークリッド空間中の近傍探索を行う方法である KD 木 [17] や Ball 木 [18] を利用した近傍探索が有効であることが知られている．KD 木は探索空間を次元毎に二分割することで探索の効率を図っており，Ball 木は探索空間を次元毎ではなく超球面に沿って分割することで KD 木の高次元での非効率性を改善している． これらの手法は，探索のクエリに対して $O(D \log N)$ の時間計算量を要する (いずれも D がある程度小さい場合) が，木構造の構築にオーバーヘッドがあるので 前述のバッチ化により木の構築回数を減らすことで高速化の効果が相乗効果により大きく得られると考えられる．また，高次元で線形探索

以上の効率で厳密な最近傍探索を行うことは困難であることが知られている [19] が，GPU の特性を利用することで線形探索を高速に行う手法も考案されている [20].

近似的な最近傍探索の手法を可視化アルゴリズムに適用すると，D や N が大きい際にも高速に近傍探索が可能である．近似最近傍探索 (Approximate Nearest Neighbor, ANN) には情報検索や画像認識を始めとした広範な応用があるため，近年も盛んに研究されいる．Navigable Small World Graphs (NSW) [21] などのグラフ系の近傍探索手法では，探索データをノードとするグラフ構造を考え，探索データ間に k-近傍グラフを作成しておき，探索時には辺の張られたノード間の距離のみを計算することにより計算量を改善できる．SONG 法では，アルゴリズム中でコーディングベクトル間に関連度を辺の重みとした k-近傍グラフを構築するため，この情報を利用することで D, N が大きい際でも探索コストを大幅に抑えることが期待できる．

6　おわりに

本稿では，インクリメンタルな低次元埋め込み手法である SONG 法を用いた教師なし音環境認識システムを提案した．実装上の改善をすることで可視化フェーズにおいて初期実装と比較し 2 倍以上の高速化を実現した．さらに，改善した実装を用いて，実際の鳥の鳴き声データに関しても適用し，同様の高速化の効果があることを確認した．また，データ数を増やした場合の調査により，さらなる高速化の可能性について考察し，さらなる高速化の可能性とリアルタイムの音環境認識への応用可能性が示唆された．

謝辞

本研究は JSPS 科研費 No.20H00475, 19KK0260 の助成を受けた．実験には京都大学情報学研究科 先端数理科学専攻 応用解析学講座の計算機を利用した．

参考文献

[1] Jacob Benesty, Jingdong Chen, and Yiteng Huang. *Microphone array signal processing*, Vol. 1. Springer Science & Business Media, 2008.

[2] K. Nakadai, T. Lourens, H. G. Okuno, and H. Kitano. Active audition for humanoid. In *Proceedings of 17th National Conference on Artificial Intelligence (AAAI-2000)*, pp. 832–839. AAAI, 2000.

[3] Kazuhiro Nakadai and Hiroshi G Okuno. Robot audition and computational auditory scene analysis. *Advanced Intelligent Systems*, Vol. 2, No. 9, p. 2000050, 2020.

[4] Tim Sainburg, Marvin Thielk, and Timothy Q Gentner. Latent space visualization, characterization, and generation of diverse vocal communication signals. *bioRxiv*, 2020.

[5] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, Vol. 9, No. Nov, pp. 2579–2605, 2008.

[6] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[7] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nature communications*, Vol. 10, No. 1, pp. 1–14, 2019.

[8] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

[9] Damith Senanayake, Wei Wang, Shalin H. Naik, and Saman Halgamuge. Self Organizing Nebulous Growths for Robust and Incremental Data Visualization. *arXiv:1912.04896 [cs]*, June 2020. arXiv: 1912.04896.

[10] Kenta Shinzato and Ryosuke Kojima. An unsupervised auditory scene analysis system using incremental low-dimensional embedding. In *Proceedings of the 2021 IEEE/SICE International Symposium on System Integration(SII2021), (to appear)*, 2021.

[11] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, Vol. 77, No. 1-3, pp. 125–141, 2008.

[12] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[13] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[14] Julio G Arriaga, Martin L Cody, Edgar E Vallejo, and Charles E Taylor. Bird-db: A database for annotated bird song sequences. *Ecological Informatics*, Vol. 27, pp. 21–25, 2015.

[15] David M. Chan, Roshan Rao, Forrest Huang, and John F. Canny. t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data. *arXiv:1807.11824 [cs, stat]*, July 2018. arXiv: 1807.11824.

[16] Corey J. Nolet, Victor Lafargue, Edward Raff, Thejaswi Nanditale, Tim Oates, John Zedlewski, and Joshua Patterson. Bringing UMAP Closer to the Speed of Light with GPU Acceleration. *arXiv:2008.00325 [cs, stat]*, August 2020. arXiv: 2008.00325.

[17] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, Vol. 18, No. 9, p. 509–517, September 1975.

[18] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[19] 和田俊和. 高次元空間における近似最近傍探索技術の進歩とその展望 (特集; 大規模画像データ処理). 人工知能, Vol. 25, No. 6, pp. 761–768, 2010.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus, 2017.

[21] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018.