情報処理学会論文誌　Vol. 50　No. 8　1883–1891 (Aug. 2009)

効率的な類似検索のためのピボット学習法

木村　学[†1]　斉藤和巳[†2]　上田修功[†3]

与えられたクエリから類似したオブジェクトを同定する類似検索は，情報検索，パターン認識などの広い分野での重要な技術であり，多くのアプリケーションが存在する．効率的な類似検索を行う 1 つのアプローチとして，我々は，クエリとデータ群の各オブジェクトとの間で行われる類似計算の回数を減らすことに焦点を当て，特にピボットの集合に注目する．本論文では，ピボットの集合をデータから選択する従来手法とは異なり，データが存在しうる距離空間から機械学習アプローチにより学習する新たな手法を提案する．2 種類の人工データ，2 種類の実データを使用した実験により，提案手法が代表的な従来手法と比較して，類似計算の平均的な実行回数を減らすことで高速化が可能であることを示す．

Pivot Learning for Efficient Similarity Search

Manabu Kimura,[†1] Kazumi Saito[†2]
and Naonori Ueda[†3]

Similarity search, the task of finding objects similar to a given query object, is an important operation in multimedia databases, and has many applications in a wider variety of other fields as well. An existing approach to similarity search utilizes a set of pivots to reduce the number of similarity computation between a query and objects in a database. In this paper, unlike conventional methods which choose pivots from existing data objects using combinatorial optimization techniques, we propose a novel method that learns a set of pivots from objects not in the database, in virtue of iterative numerical nonlinear optimization. In our experiments using two synthetic and two real data sets, we show that the proposed method significantly reduced the average number of similarity computations, compared with some representative conventional methods.

1. はじめに

近年，画像，音声，ビデオクリップ，3 次元オブジェクト，文書などのマルチメディアデータが大量蓄積され，検索技術の高速化がますます重要になってきている．本論文では，類似検索に注目し検索を効率化するための手法について検討する．

類似検索とは，クエリとなるオブジェクトに類似したデータオブジェクトをデータベースなどに蓄積されたデータ集合の中から同定するタスクを指す．多くの場合，オブジェクト間の類似度は，距離関数を用いる．距離関数は，非負性，対称性，三角不等式の性質を満足する．この性質を基に多くの高速化技術が提案されている．

類似検索を高速化する一般的な方法として，全データオブジェクトとの距離が記録されたピボットと呼ばれるオブジェクトを利用する方法がある．この方法は，距離計算が高コストな多様な実問題での効率化をターゲットとし，検索時に，ピボットを利用してクエリとデータオブジェクト間の距離下界を低コストで求め，ある条件を満たすオブジェクトを下界の数値から枝狩りし，実際の距離計算を避けることで高速化する．オブジェクトを枝狩りできるかどうかはピボットに依存する．そのため，多くの枝狩りを可能にするピボット集合を事前に求めておくことが高速化のカギとなる．

これまでに提案されたピボットを計算する方法は，データオブジェクト集合からピボットを選択する点で共通しており，目的関数，最適化方法の観点から整理できる．詳細は，後で述べる．その中でも，Bustos ら[2] は，効率的な検索を実現するピボット集合を選択するための目的関数，そして，その目的関数をインクリメンタル法により最適化する手法を提案している．同手法は，それまでの手法よりも，検索時間が改善することを実証している．

しかしながら，後で示すように，データオブジェクト集合を含む距離空間には，データオブジェクト集合と同等かそれ以上に探索効率の高いピボット集合が存在しうる．本論文では，距離空間からピボット集合を抽出する問題設定を新たに提起し，距離計算のさらなる削減を可能とする優れたピボット集合の抽出手法を提案する．具体的には，Bustos らが提案したピボット選択のための目的関数を距離空間全体に拡張し，機械学習アプローチにより効

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
†2 静岡県立大学経営情報学部
School of Administration and Informatics, University of Shizuoka
†3 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

率的に最適化する．

本論文の構成は以下となる．2 章では，従来手法とその問題点について述べる．3 章では，提案手法の詳細について説明し，4 章で，その計算量について考察する．5 章では，2 種類の人工データ，2 種類の現実データを使用し，提案手法の性質を調べ，提案手法が代表的な従来手法と比べて距離計算の平均的な実行回数を大幅に削減できることを示す．6 章はまとめである．

2. 従来のピボット抽出手法とその問題点

2.1 レンジクエリタスク

$(\mathbb{X}, d)$ をオブジェクトの空間 $\mathbb{X}$ および $\mathbb{X}$ 上に定義された距離関数 d によって定まった距離空間とおく．d は非負性，対称性，三角不等式の性質を満たす．距離空間 $(\mathbb{X}, d)$ 上に，N 個のオブジェクト集合，すなわち，データベースがあるとし，それを $\mathbb{U} = \{u_1, \cdots, u_N; u_n \in \mathbb{X}\}$ とする．

本論文では，与えられたクエリオブジェクト $q \in \mathbb{X}$ から距離 r 以内にあるオブジェクトを同定するタスクを扱う．このタスクは “レンジクエリ” と呼ばれる．厳密には，$d(q,u) \leq r$ を満たすオブジェクト $u \in \mathbb{U}$ をすべて列挙する問題である．レンジクエリタスクでは，厳密性を保証したうえで，いかに距離計算回数を削減するかが重要となる．

2.2 レンジクエリタスクにおけるピボットを利用した枝狩り

ピボット p は，検索の前処理の段階で $\mathbb{U}$ 上のすべてのオブジェクト $u \in \mathbb{U}$ との距離 $d(p,u)$ を計算しておくオブジェクトである．本節では，検索時に K 個のピボット集合 $\mathcal{P}_K = \{p_1, \cdots, p_K; p_k \in \mathbb{U}\}$ を用いて，$d(q,u)$ を計算することなく $d(q,u) > r$ を判定する枝狩りの方法について説明する．

今，**図 1** に示すように，クエリ q と $u \in \mathbb{U}$ に対し，2 つのピボット p_1，p_2 が配置されているものとする．距離の公理（三角不等式）より $d(q,u) \geq |d(q,p_1) - d(p_1,u)|$ が成り立つ．図より，明らかに，$d(q,u) > r$ であるが，$|d(q,p_1) - d(p_1,u)| < r$ ゆえ，上記不等式から $d(q,u) > r$ と判定できない．一方，p_2 に対しては，$d(q,u) \geq |d(q,p_2) - d(p_2,u)| > r$ ゆえ，$d(q,u) > r$ と判定できることが分かる．すなわち，K 個のピボット $\mathcal{P}_K = \{p_1, \ldots, p_K\}$ の場合，クエリ q とオブジェクト u 間の距離 $d(q,u)$ の下限値は

$$D_{(q,u)}(\mathcal{P}_K) \stackrel{\text{def}}{=} \max_{1 \leq k \leq K} |d(q,p_k) - d(p_k,u)|$$

で得られるので，$d(q,u) \leq r$ となる $u \in \mathbb{U}$ を検索する場合，$\mathbb{U}' = \{u; u \in \mathbb{U}$ かつ $D_{(q,u)}(\mathcal{P}_K) > r\}$ とすると，$|\mathbb{U}'|$ 個のオブジェクトは，$d(q,u) > r$ として枝狩り可能で，距離計算 $d(q,u)$ が不要となる．$K \ll N$ とすると，クエリ q に対するレンジクエリの場合，全探索では N 回の距離計算が必要であるのに対し，上記枝狩り法では，q と $p_i \in P$ との距離計算（K 回）および $(N - |\mathbb{U}'|)$ 個のオブジェクトとの距離計算で済む．つまり，枝狩り数：$N - (K + (N - |\mathbb{U}'|)) = |\mathbb{U}'| - K$ より，$|\mathbb{U}'| > K$ である限り，ピボット法では，全探索に比べ $|\mathbb{U}'| - K$ 回の距離計算を削減できる．明らかに，$|\mathbb{U}'|$ の数は K 個のピボットの配置に依存する．換言すれば，$|\mathbb{U}'|$ を最大化する $\mathcal{P}_K$ の決定法が本研究の課題となる．

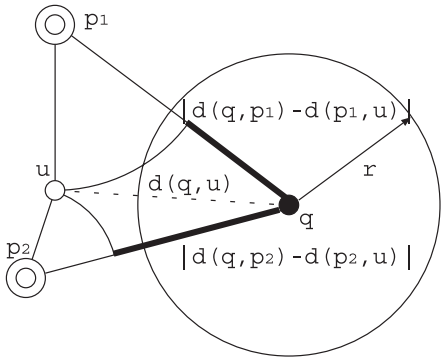

図 1　ピボットを利用した枝狩り
Fig. 1　Pruning by using pivots.

2.3 評価基準

検索時に要求される距離計算の実行回数は，全データオブジェクトの下限値 $D_{(q,u)}(\mathcal{P}_K)$ を決定するため共通に計算される $\{d(p_1,q), \cdots, d(p_K,q)\}$ の数 K と，$D_{(q,u)}(\mathcal{P}_K) > r$ を満足しなかったオブジェクト群 $\mathbb{U} - \mathbb{U}'$ に対して実行される距離計算 $d(q,u)$ の数との和 $K + (N - |\mathbb{U}'|)$ によって表される．以後，この和を総類似検索コストと呼び，本論文でのピボット抽出手法の評価基準として用いる．

2.4 従来手法の詳細

これまでに提案されたピボット集合 $\mathcal{P}_K = \{p_1, \cdots, p_K; p_k \in \mathbb{U}_K\}$ の抽出方法について整理する．効率的な探索を実現するために設計された，ピボット集合 $\mathcal{P}_K$ に対する目的関数，そして，最適化方法，2 つの観点から整理することができる．

最適化手法に関して，既存手法は，繰返し計算に基づき，ループのつど $\mathbb{U}$ の中からピボットを 1 つずつ確定するインクリメンタルな手段をとる．すなわち，k 回目のループ時に，す

でに選んだ $k-1$ 個のピボットに基づき，それぞれの手法の方針により k 個目のピボットを選ぶ．目的関数に関しては，Bustos ら[2)]，Micó ら[8)]，各々が提案したものがある．

Bustos らは，枝狩りのための条件 $D_{(q,u)}(\mathcal{P}_K) > r$ を満足する $u \in \mathbb{U}$ の数をできる限り増加させるために，$\mathbb{U}$ 全体での $D_{(q,u)}(\mathcal{P}_K)$ の和を表現する目的関数を以下のように提案している．

$$F_{\mathcal{A}}(\mathcal{P}_K) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{A}|} \sum_{i=1}^{W} D_{(a_i,a'_i)}(\mathcal{P}_K). \tag{1}$$

ここで，$\mathcal{A} = \{(a_1, a'_1), \cdots, (a_W, a'_W)\}$ は $\mathbb{U}$ からランダムに選ばれる W 個のオブジェクトのペアの集合である．この目的関数 (1) を，BNC（*Bustos Navarro Chávez*）基準と呼ぶ．BNC 基準に前述の最適化手法を組み合わせ，BNC incremental 法[2)] が提案されている．ただし，各ループで選択するピボットは，$\mathbb{U}$ からサンプリングする R 個のデータオブジェクトの中から選ぶ．

Micó らは，ピボット集合がお互いに離れたものにするために，ピボット集合が外れ値である度合いを表現する以下の目的関数を提案している．

$$G(\mathcal{P}_K) = \sum_{k=1}^{K} d(u, p_k). \tag{2}$$

加えて，ループのつど，この目的関数 (2) を最大にするオブジェクトを $\mathbb{U}$ の中から選ぶアルゴリズムも提案している．この方法は，Outlier 法[8)] と呼ばれる．

目的関数を持たない手法として Brin が採用した MaxMin 法[1)] がある．これは，Micó らの手法と同様にお互いに離れたピボット集合を抽出することを目的とし，以下のようにループのつど新しいピボット p_k を求める．

$$p_k = u^* = \arg\max_{u \in \mathbb{U}} \min\left\{d(u, p_1), \cdots, d(u, p_{k-1})\right\}.$$

Bustos らは，BNC incremental 法と MaxMin 法を比較する実験は行ってないが，本論文では，提案手法の性能を上記すべての従来手法と比較して評価する．

2.5 問　題　点

前節で述べた従来手法は，データオブジェクト集合からピボットを選択する点で共通している．しかしながら，$\mathbb{X}$ は $\mathbb{U}$ を包含するため，$\mathbb{X}$ には $\mathbb{U}$ 内の要素と同等かそれ以上に効率的な探索を実現するピボットが存在する．これを説明するために，以下の定理を導入する．

定理 1　任意の関数 $H(\cdot)$，および部分集合 $\mathbb{U}_s \subset \mathbb{U}$ に対して，$H(\mathbb{X}_s) \geq H(\mathbb{U}_s)$ となる $\mathbb{X}_s \subset \mathbb{X}$ が存在する．

証明　$\mathbb{U}_s$ が $\mathbb{X}$ 全体で $H(\cdot)$ を最大とする場合，$\mathbb{U} \subset \mathbb{X}$ より，$H(\mathbb{U}_s) = H(\mathbb{X}_s)$ となる $\mathbb{X}_s (= \mathbb{U}_s)$ を得ることができる．$\mathbb{U}_s$ が $\mathbb{X}$ 全体で $H(\cdot)$ を最大としない場合，$\mathbb{X} - \mathbb{U}$ すなわち，$\mathbb{X} - \mathbb{U}$ を包含する $\mathbb{X}$ に $H(\cdot)$ を最大とする $\mathbb{X}_s$ が存在する．したがって，任意の $\mathbb{U}_s$ に対して，$H(\mathbb{X}_s) \geq H(\mathbb{U}_s)$ となる $\mathbb{X}_s$ が存在する．

定理 1 より，目的関数が適切であれば，$\mathbb{U}$ よりも $\mathbb{X}$ に効率的な探索を実現するピボットが存在するといえる．

また，ピボットをインクリメンタルに求める最適化方法は，並列処理，新たに追加されるデータオブジェクトへの対応が容易ではない．並列処理に関しては，各ピボットを独立に最適化することができないため，同時に複数のピボットへの最適化が不可能である．追加データへの対応に関しては，データが追加されたときにピボットを全データに対して最適なものにするには零から再計算する必要があり，それまでの計算が無駄になる．

次章では，これらの問題点を克服する，新たなピボット抽出方法を提案する．

3. 提 案 手 法

提案するピボット学習法は，オブジェクト空間 $\mathbb{X}$ に属する任意のピボットに関する目的関数を，K-Means Clustering alogorithm[4)] でセントロイドを計算するように各々のピボットを独立に更新し，目的関数を逐次的に最適化する．これは，前章で述べた，インクリメンタルによる最適化方法の問題点を克服する．ピボットを独立に計算できるので，ピボットを並列に最適化することが可能である．新たなデータの追加に対しても，1 度学習したピボットを次の学習の初期点として使用し，零から最適化することなくより短時間でピボットを最適化できる．

3.1 節では，変数空間を $\mathbb{X}$ としたピボットに関する目的関数を設計し，それを独立更新可能なものに変形する．本ピボット学習法は，すべての距離空間で適用可能である．3.2 節では，ユークリッド空間での実現例を説明する．

3.1 目 的 関 数

データセット $\mathbb{U}$ が用意されたときに，入力されるクエリ q，レンジ r に対しての枝狩り数を最大化するピボット集合を抽出したい．これに留意して，枝狩り条件 $D_{(q,u)}(\mathcal{P}_K) > r$ を満たす u の数の見積りを目的関数とする．クエリとレンジに関する 2 つの確率分布 $s(q)$ と $s(r)$ を仮定し，以下の形で目的関数を定義する．

$$C(\mathcal{P}_K) \stackrel{\text{def}}{=} \int_{q\in\mathbb{X}, r\in\mathcal{R}^+} |\{u; D_{(q,u)}(\mathcal{P}_K) > r, u \in \mathbb{U}\}| \times s(q)s(r)\ dq\ dr. \tag{3}$$

しかしながら，$s(q)$ と $s(r)$ は未知であるので，直接 $C(\mathcal{P}_K)$ を最大化することはできない．式 (3) を計算可能なものに変更する．まず，分布 $s(r)$ に関係なく枝狩り数を最大化するために，$D_{(q,u)}(\mathcal{P}_K)$ を最大化するものに変更する．ここで，$D_{(q,u)}(\mathcal{P}_K) \le d(q,u)$ であり，$D_{(q,u)}(\mathcal{P}_K)$ の下限としての精度向上と見なすこともできる．次に，$s(q)$ を経験分布に置き換え，leave-one-out の要領で各オブジェクト u をクエリ q と見なし，$\mathbb{U}$ 上のすべてのペア $\mathcal{B} = \{(u_1,u_2),(u_1,u_3),\cdots,(u_{N-1},u_N)\}$ について計算するものにする．以上の修正により以下の関数 $F_{\mathcal{B}}(\mathcal{P}_K)$ を定義できる．

$$\begin{aligned} F_{\mathcal{B}}(\mathcal{P}_K) &\stackrel{\text{def}}{=} \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} D_{(u_i,u_j)}(\mathcal{P}_K) \\ &= \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \max_{1\le k\le K} |d(p_k,u_i) - d(p_k,u_j)|. \end{aligned} \tag{4}$$

ここで，

$$\mathcal{S}_k(\mathcal{P}_K) \stackrel{\text{def}}{=} \{(i,j); k = \arg\max_{1\le k\le K} |d(p_k,u_i) - d(p_k,u_j)|, (u_i,u_j) \in \mathbb{U}\} \tag{5}$$

とおくと，

$$F_{\mathcal{B}}(\mathcal{P}_K) = \sum_{k=1}^{K} \sum_{(i,j)\in\mathcal{S}_k(\mathcal{P}_k)} |d(p_k,u_i) - d(p_k,u_j)| \tag{6}$$

を得る．式 (4) は，各 (i,j) で最適な k での和を求めている．一方，式 (5) のように，各 (i,j) に対し最適な k を定義しておけば，式 (6) の計算で式 (4) を求めることができる．つまり等価な表現である．式 (6) の右辺で k に関する足し合わせ $\sum_{k=1}^{K}$ の内側の項を

$$F_{\mathcal{B}}^{(k)}(\mathcal{P}_K) \stackrel{\text{def}}{=} \sum_{(i,j)\in\mathcal{S}_k(\mathcal{P}_k)} |d(p_k,u_i) - d(p_k,u_j)| \tag{7}$$

とおく．式 (7) の右辺は，$\{d(p_k,u_i); i = 1,\cdots,N\}$ の線形和の形で書くことができる．線形和の係数を $\{n_k(i); i = 1,\cdots,N\}$ とおく．$n_k(i)$ は $d(p_k,u_i)$ とそれ以外の項の大小関係で計算できる．$d(p_k,u_i)$ が他の項以上に大きいときの数を $n_k^+(i)$，他の項以下に小さいときの数を $n_k^-(i)$，すなわち，

$$\begin{aligned} n_k^+(i) &= \left|\{(i,j) \in \mathcal{S}_k(\mathcal{P}_K); d(p_k,u_i) \ge d(p_k,u_j)\}\right|, \\ n_k^-(i) &= \left|\{(i,j) \in \mathcal{S}_k(\mathcal{P}_K); d(p_k,u_i) \le d(p_k,u_j)\}\right|, \end{aligned}$$

とおくと，$n_k(i)$ は

$$n_k(i) = n_k^+(i) - n_k^-(i),$$

と計算できる．したがって，式 (7) は

$$F_{\mathcal{B}}^{(k)}(\mathcal{P}_K) = \sum_{i=1}^{N} n_k(i) d(p_k,u_i) \tag{8}$$

と変形できる．

式 (8) による $F_{\mathcal{B}}(\mathcal{P}_K) = \sum_k^K F_{\mathcal{B}}^{(k)}(\mathcal{P}_K)$ の計算は，相殺される加減算を抑える効果もある．実際，単純な加算回数に着目すれば，式 (6) を直接求めるのに，$N(N-1)/2$ ペアのオブジェクトペア群に対して $|d(p_k,u_i) - d(p_k,u_j)|$ の和が必要である．一方，式 (8) による $F_{\mathcal{B}}(\mathcal{P}_K)$ の計算では，NK 回の必要な項のみの和で計算できる．ピボットに基づいたアプローチは，$N \gg K$ を仮定しているので大幅な効果となる．このような相殺項の除去は，大規模データを扱うのに重要と考える．

3.2　ユークリッド空間での実現例

L 次元ベクトル空間を想定し，オブジェクトおよびピボットをそれぞれ $\{\boldsymbol{x}_i : i = 1,\cdots,N\}$，$\{\boldsymbol{p}_k : k = 1,\cdots,K\}$ とする．距離関数は，

$$d(\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \sqrt{\sum_{l=1}^{L}(x_l - y_l)^2}$$

で定義する通常のユークリッド距離を用いる．

3.2.1　計 算 手 順

提案するアルゴリズムの手順を以下に整理する．

step 1.　$\mathbb{U}$ からランダムに K 個オブジェクトを選ぶことにより，$\{\boldsymbol{p}_1,\cdots,\boldsymbol{p}_K\}$ を初期化する．

step 2.　以下のステップを T 回繰り返す．

step 2-1.　それぞれの $\boldsymbol{p}_k$ について，分割式 (5) で $\mathcal{S}_k(\mathcal{P}_K)$ を求め，$n_k(i)$ を計算して目的関数 (8) を定める．

step 2-2. 各ピボット $\boldsymbol{p}_k$（$k=1,\ldots,K$）を，$\boldsymbol{p}_k \leftarrow \boldsymbol{p}_k + \Delta\boldsymbol{p}_k$ [⋆1]により更新する．$\Delta\boldsymbol{p}_k$ はステップ幅であり，連立方程式 $H\Delta\boldsymbol{p}_k + \nabla F_{\mathcal{B}}^{(k)}(\boldsymbol{p}_k)$ [⋆2] $= 0$ を解くことで求める[⋆3]．すなわち，ニュートン法[5] による更新である．ここで，$\nabla F_{\mathcal{B}}^{(k)}(\boldsymbol{p}_k)$ および H は，$F_{\mathcal{B}}^{(k)}(\boldsymbol{p}_k)$ に対する勾配ベクトルおよびヘス行列であり，以下となる．

$$\nabla F_{\mathcal{B}}^{(k)}(\boldsymbol{p}_k) = \sum_{i=1}^{N} \frac{n_k(i)}{d(\boldsymbol{p}_k, \boldsymbol{u}_i)}(\boldsymbol{p}_k - \boldsymbol{u}_i),$$

$$H = \sum_{i=1}^{N} \frac{n_k(i)}{d(\boldsymbol{p}_k, \boldsymbol{u}_i)} \left(\boldsymbol{I}_L - \frac{(\boldsymbol{p}_k - \boldsymbol{u}_i)(\boldsymbol{p}_k - \boldsymbol{u}_i)^T}{d(\boldsymbol{p}_k, \boldsymbol{u}_i)^2} \right).$$

$\boldsymbol{I}_L$ は，L 次元単位行列を表す．

このアルゴリズムはユークリッド空間に特化したものであるが，各ピボットの更新式を定義できさえすれば，他の距離空間にも適用可能である．

提案アルゴリズムによるピボット学習過程の概要は以下となる．与えられたデータセット $\mathbb{U}$ と，ユーザが指定するピボットの個数 K と step 2 の反復回数 T がアルゴリズムへの入力である．まず，step 1 で規定されるように，K 個のピボットの初期値は，データセットからランダムに選択されるので，この段階では，ピボット集合はデータセット $\mathbb{U}$ に包含される．次いで，step 2 による更新を実行することにより，データセット $\mathbb{U}$ には限定されず，より広いオブジェクト空間 $\mathbb{X}$ において，目的関数を改善するようにピボットの場所が変わる．直感的には，我々の行った実験の範囲では，step 2 の反復を実行することにより，K 個のピボット集合の全体でデータセット $\mathbb{U}$ を外包するように各ピボットの場所が変化していく．アルゴリズムは，学習結果のピボット集合と，各ピボットとデータセット $\mathbb{U}$ のオブジェクトとの距離を出力して終了する．

3.2.2 補　　足

提案アルゴリズムは，step 2-1，step 2-2 を繰り返す．それぞれのステップが $F_{\mathcal{B}}(\mathcal{P}_K)$ を増加させる理由を述べる．

まず，step 2-1 での $\mathcal{S}_k(\mathcal{P}_K)$ による再分割が，$F_{\mathcal{B}}(\mathcal{P}_K)$ を増加させる理由を述べる．すべてのピボット $\mathcal{P}_K = \{p_k : k = 1, \cdots, K\}$ を $\mathcal{P}'_K = \{p'_k : k = 1, \cdots, K\}$ に更新後，オブジェクトのペア (u_i, u_j) に関する $|d(p_k, u_i) - d(p_k, u_j)|$ を最大化させるピボットは，更新前に最大化していた p_k が同じ k での更新後の p'_k であるとは限らない．再分割することで，すべてのオブジェクトのペアに関して $|d(u_i, p_k) - d(p_k, u_j)|$ を最大にするピボットを選びなおし，同等か，それ以上の値を得ることが可能となる．以上の議論は，ユークリッド空間以外でも成り立つ．

次に，step 2-2 でのニュートン法による更新の妥当性を議論する．ニュートン法を適用する場合，ヘス行列 H が負定値となる必要がある．ヘス行列 H がつねに負定値であることは保証されないものの，H のトレースが非正値であることが以下のように示される．

$$\begin{aligned} trace\{H\} &= (L-1)\sum_{i=1}^{N} \frac{n_k(i)}{d(\boldsymbol{p}_k, \boldsymbol{u}_i)} \\ &= (L-1) \sum_{(i,j)\in S_k(\mathcal{P}_K)} \left(\frac{1}{d(\boldsymbol{p}_k, \boldsymbol{u}_i)} - \frac{1}{d(\boldsymbol{p}_k, \boldsymbol{u}_j)} \right) \\ &= -(L-1) \sum_{(i,j)\in S_k(\mathcal{P}_K)} \frac{d(\boldsymbol{p}_k, \boldsymbol{u}_i) - d(\boldsymbol{p}_k, \boldsymbol{u}_j)}{d(\boldsymbol{p}_k, \boldsymbol{u}_i) d(\boldsymbol{p}_k, \boldsymbol{u}_j)} \leq 0. \end{aligned}$$

ヘス行列 H が負定値なら，トレースは負になる．しかし，逆がつねに成り立つとは限らない．しかしながら，実験では，$F_{\mathcal{B}}^{(k)}(\mathcal{P}_K)$ に対するヘス行列 H は，つねに負定値となっていた．

4. ピボット集合の抽出計算量の比較

提案手法と BNC incremental 法のピボット抽出における計算量について議論する．

提案手法は，$N(N-1)/2$ 個のペアワイズオブジェクトの距離に対し，その下限値の近似精度を最も向上させるように K 個のピボット集合での評価を行う．また step 2 では，T 回の反復処理で，ピボットを更新する．したがって，計算量は，$O(TKN(N-1))$ となる．BNC incremental 法と同様に，W 個のペアをサンプリングするならば，$N(N-1)$ を W に置き換えることになる．

BNC incremental 法は，RW の計算コストが必要な式 (1) を，K 回繰り返し評価する．したがって，計算量は $O(RKW)$ となる．ただし，W はサンプリングされたオブジェクトのペアの数であり，提案手法と同様に，オブジェクトのすべてのペアを対象とするならば，W を $N(N-1)$ に置き換えることになる．

オブジェクト間のすべてのペアについての計算を考えた場合，BNC incremental 法と提案手法，それぞれの計算量は，$O(RKN(N-1))$，$O(TKN(N-1))$ である．両者の違い

⋆1 Δ は変数の前に付いて，その変数の増分を表す．
⋆2 ∇ はスカラを出力する関数の前に付くことで，引数の座標での勾配ベクトルを表す．
⋆3 連立方程式を解くにあたり LL 分解法[7] を利用した．

は，R と T である．前者 R の値は，文献 2) 中の評価実験では，50 に設定されている．一方，後者 T の値は，次章で述べるように 10 で十分な値となる．したがって，上述の R と T を決めたならば，我々の実験では，提案手法は，BNC incremental 法に比べて 5 分の 1 ほどの計算量となる．オブジェクトのペア集合をサンプリングした場合でも同様の議論が成り立つ．

5. 実　　験

5.1 デ ー タ

実験では 2 種類の人工データと 2 種類の実データを用いた．**表 1** に示すように，2 種類の人工データは，それぞれ，8，16 次元の単位立方体内に一様に分布した 50,000 オブジェクトにより構成される．実データの 1 種類目は，40,700 枚からなる NASA の画像データ[⋆1]である．詳細には，Bustos ら[2] と同様に，20 次元のベクトルに変換された画像オブジェクトを実験に用いた．実データの 2 種類目は，1993 年から 1995 年まで掲載された 64,585 文書からなる毎日新聞の国際面記事である．LSA（Latent Semantic Analysis）[6] により，20 次元のベクトルに変換された文書オブジェクトを用いた．

5.2 提案手法の評価

提案手法でピボット集合を抽出し，学習に用いたデータ，それとは異なる別途独立なテスト用のデータ，それぞれをクエリの集合と見なし類似検索を行い，それぞれの総類似検索コストの平均値を比較評価した．**表 2** は，16 次元人工データを用いた実験において，ピボット数を 20，40，60，80，100 に設定して抽出し，その学習用データ，それとは独立に生成した 16 次元単位立方体に一様に分布した 50,000 個のテスト用データを使い，それぞれの総類似検索コストを比較評価した結果である．どのピボット数でも，学習用データとテスト用データでの総類似検索コストは，ほぼ等しいことが分かる．したがって，学習用データをテストのために利用しても，妥当な評価結果が得られると考えられる．機械学習分野でのテスト用データに対する汎化性能（generalization performance）の概念でいえば，学習用データのオブジェクト数が十分に存在するため，それらでの性能と汎化性能がほとんど等しいことを意味すると考える．特に実データでは，オブジェクト数が限られている状況で，より大規模なデータでの比較実験を行うため，それらすべてを学習とテストの両方に利用して実験を行った．

⋆1 Sixth DIMACS Implementation Challenge: Available Software. http://www.dimacs.rutgers.edu/Challenges/Sixth/Software.html

表 1　データ名とそれらの概要
Table 1　Data names and their descriptions.

名称	内容
8 次元人工データ	ランダムベクトル 8 次元 50,000 データ
16 次元人工データ	ランダムベクトル 16 次元 50,000 データ
NASA データ	画像アーカイブ 20 次元 40,700 データ
毎日新聞データ	毎日新聞 5 年分 20 次元 64,585 データ

5.2.1 学習性能の評価

提案手法は逐次的に最適化を行うため，それぞれの実験データでのループ回数に対する目的関数 (4) の値の変化を評価した．詳細には 4 種類のデータセットにおいて，最大ループ回数を $T = 20$ に設定し，ループごとにそれぞれの目的関数 (4) の値を求めた．ピボット数 50，100 での結果を，それぞれ，**図 2**，**図 3** に示す．

人工データでは目的関数値が約 10 ループで十分に安定し，実データでは 5 ループ程度でも安定した値となっている．このことは，最初の数ループで効率良く最適化が行われていることを示唆している．以下では，提案手法のループ回数を $T = 10$ に設定し，実験を行った．

5.3 従来手法との比較評価

すでに述べた従来手法（BNC incremental 法，MaxMin 法，Outliers 法）との比較実験により，提案手法の性能を評価した．

まず，共通の実験設定について記述する．Bustos ら[2] と同様に，データから 0.01%のオブジェクトを同定するレンジクエリで評価した．すなわち，それぞれのデータにおいて，$|\{u : d(q,u) \le r\}| \approx 0.0001N$ となる r によるレンジクエリである．我々の評価尺度は，総類似検索コストの平均値である．つまり，leave-one-out cross-validation のように，データ中の各オブジェクトをクエリと見なし，それぞれの検索での総類似検索コストを平均した値で評価した．ピボットの数は $10, 20, 30, \ldots, 90, 100$ と変化させた．ただし，16 次元人工データのみ $10, 50, 100, 150, \ldots, 400, 450$ と変化させた．BNC incremental 法のパラメータの値は，Bustos らが示唆するよう $R = 50$ に設定した．また，実験条件を等しくするため，BNC incremental 法のオブジェクトのペア集合 $\mathcal{A}$ をデータ集合 $\mathbb{U}$ のすべてのペアとした．

前述の 4 種類のデータでの実験結果を図 4 から図 7 に示す．すでに指摘されているように[2]，Outlier 法の性能はデータに大きく依存する．**図 4** と**図 5** の人工データでは，他の従来法と比較して同等な性能を示すものの，**図 6** と**図 7** の実データでは，ピボットを追加し

表 2　16 次元人工データにおける学習用データとテスト用データでの総類似検索コストの比較
Table 2　The evaluation of the total complexity using training data and test data.

ピボット数	20	40	60	80	100
学習用データによる評価（総類似検索コスト）	9.750×10^4	4.865×10^3	3.044×10^3	2.268×10^3	1.828×10^3
テスト用データによる評価（総類似検索コスト）	9.762×10^4	4.690×10^3	3.044×10^3	2.271×10^3	1.830×10^3

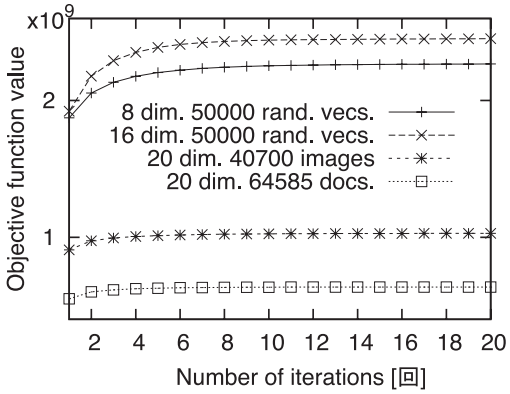

図 2　ピボット数 50 での提案手法の学習曲線
Fig. 2　Learning curve of proposed method at 50 pivots.

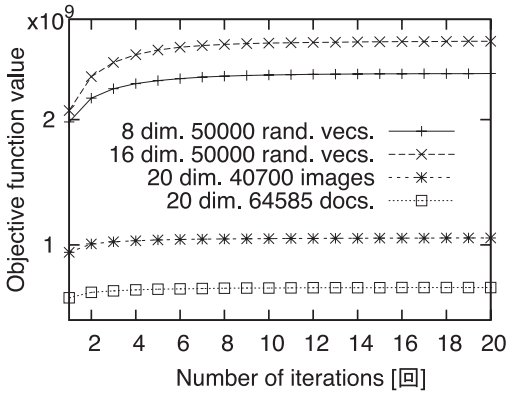

図 3　ピボット数 100 での提案手法の学習曲線
Fig. 3　Learning curve of proposed method at 100 pivots.

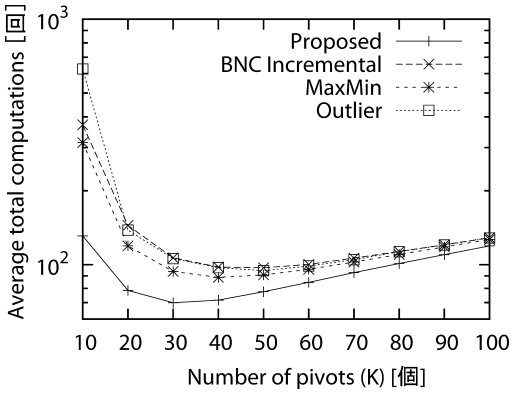

図 4　8 次元ランダムベクトルでの比較
Fig. 4　Comparison using 8 dimensional random vectors.

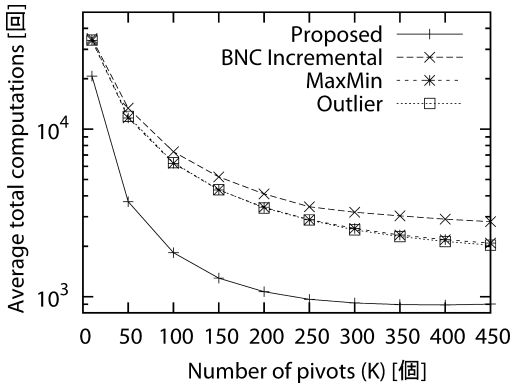

図 5　16 次元ランダムベクトルでの比較
Fig. 5　Comparison using 16 dimensional random vectors.

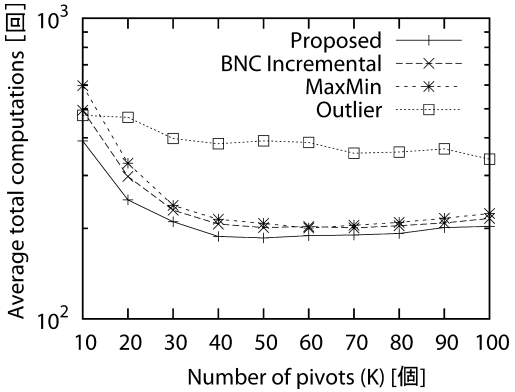

図 6　NASA の画像アーカイブでの比較
Fig. 6　Comparison using NASA images.

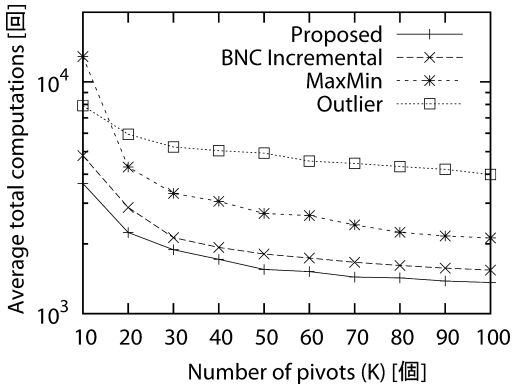

図 7　毎日新聞での比較
Fig. 7　Comparison using newspaper documents.

ても比較的にわずかな性能改善しか得られない．MaxMin 法に対しても，類似した傾向を指摘できる．BNC incremental 法は，実データにおいては，既存手法の中で高い性能を示している．

総類似計算コストは，クエリとピボット間の類似計算回数と，下界で枝狩りできなかったオブジェクトの数との和からなり，両者の間にはトレードオフの関係がある．ピボット数が少ないときは，枝狩りできないオブジェクト数が多くなるため，総類似計算コストが高くなる．ピボット数が多いときは，枝狩り数が増大するが，一方，クエリとピボット間の総類似計算コストが高くなる．最適なピボット数は，データの質に依存する．図 4 の場合，最適なピボット数は 30–40 である．図 5 では，BNC incremental 法の要求する空間コストが $O(N^2)$ と大きいため，総類似計算コストが増大するまでプロットできなかったが，図 4 と同様に，ピボット数を増大させると総類似計算コストは増大する．実際，ピボットの数 $K = 750$，$K = 1{,}000$ としたとき，Outlier 法の場合，1.873×10^3（$K = 750$）が 1.922×10^3（$K = 1{,}000$）となり，MaxMin 法の場合，1.869×10^3（$K = 750$）が 1.919×10^3（$K = 1{,}000$）となり，総類似計算コストが増大していることを確認している．

提案手法に関しては，図 4 と図 5 とで従来手法との総類似計算コストの差が顕著であるが，図 6 と図 7 では顕著でない．明らかに，手法の効果はデータの質に依存する．このデータの質を定量評価する尺度として，“距離空間の固有次元（Intrinsic Dimensionality）” を導入する．“距離空間の固有次元” とは，既存の類似検索手法の難しさを表す指標であり[3]，ペアワイズオブジェクトの距離の分布に対して，次式で定義される．

$$\rho = \frac{\mu^2}{2\sigma^2} \tag{9}$$

ここで，μ，σ は，ペアワイズオブジェクトの距離の分布に対する平均と分散である．固有次元 ρ の数値は，類似検索の難しさに比例する．実験データに対する固有次元を**表 3** に示す．表 3 より，提案手法は，既存手法にとって困難である固有次元が高いデータほど，既存手法よりも効率的に検索すると解釈できる．

表 3　実験データの固有次元
Table 3　Intrinsic dimensionality of experimental data.

データ	平均 μ	分散 σ^2	固有次元 ρ
8 次元人工データ	2.258	0.243	10.50
16 次元人工データ	3.231	0.238	21.90
NASA データ	1.478	0.212	5.163
毎日新聞データ	0.377	0.020	3.499

6. おわりに

本論文では，ピボットを利用した手法に基づく類似検索の高速化に向けて，連続空間上でピボットを探索する問題を新たに設定し，機械学習アプローチにより効率的にピボットを抽

出する手法を提案した．特にユークリッド空間を対象とし，ピボット抽出に非線形最適化手法を逐次的に適用する方法で効率化を実現した．2 種類の人工データと 2 種類の実データを使った実験では，提案手法が，代表的な従来手法と比較して，総類似検索コストの平均値を減らし高速化を実現できることを実証した．今後は，さらに多様な実験を行い，提案手法の評価を進める．

参 考 文 献

1) Brin, S.: Near Neighbor Search in Large Metric Spaces, *VLDB*, Dayal, U., Gray, P.M.D. and Nishio, S. (Eds.), pp.574–584, Morgan Kaufmann (1995).
2) Bustos, B., Navarro, G. and Chávez, E.: Pivot selection techniques for proximity searching in metric spaces, *Pattern Recognition Letters*, Vol.24, No.14, pp.2357–2366 (2003).
3) Chávez, E., Navarro, G., Baeza-Yates, R.A. and Marroquín, J.L.: Searching in metric spaces, *ACM Comput. Surv.*, Vol.33, No.3, pp.273–321 (2001).
4) Hastie, T., Tibshirani, R. and Friedman, J.H.: *The Elements of Statistical Learning*, Springer (2001).
5) Luenberger, D.G.: *Introduction to Linear and Nonlinear Programming*, Addison-Wesley (1973).
6) Manning, C.D. and Schtze, H.: *Foundations of Statistical Natural Language Processing*, The MIT Press (1999).
7) Meyer, C.D. (Ed.): *Matrix analysis and applied linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2000).
8) Micó, L., Oncina, J. and Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements, *Pattern Recognition Letters*, Vol.15, No.1, pp.9–17 (1994).

(平成 20 年 11 月 24 日受付)
(平成 21 年 5 月 13 日採録)

木村　学（学生会員）

昭和 57 年生．平成 17 年京都工芸繊維大学工芸学部電子情報工学科卒業．平成 19 年奈良先端科学技術大学院大学情報科学研究科情報処理学専攻博士前期課程修了，同年同大学院博士後期課程に進学．類似検索，機械学習，データマイニング，自然言語処理等の研究に従事．システム制御情報学会会員．

斉藤　和巳（正会員）

昭和 38 年生．昭和 60 年慶應義塾大学理工学部数理科学科卒業．同年 NTT 入社．平成 3 年より 1 年間オタワ大学客員研究員．平成 19 年より静岡県立大学経営情報学部教授．神経回路網，機械学習，複雑ネットワーク等の研究に従事．工学博士．平成 8 年情報処理学会論文賞，平成 10 年人工知能学会論文賞等受賞．電子情報通信学会，人工知能学会，日本神経回路学会各会員．

上田　修功（正会員）

昭和 33 年生．昭和 57 年大阪大学工学部通信工学科卒業．昭和 59 年同大学大学院修士課程修了．工学博士．同年 NTT 入社．平成 5 年より 1 年間 Purdue 大学客員研究員．画像処理，パターン認識・学習，ニューラルネットワーク，統計的学習，Web データマイニング等の研究に従事．現在，NTT コミュニケーション科学基礎研究所副所長主席研究員，奈良先端科学技術大学院大学客員教授．電気通信普及財団賞，電子情報通信学会論文賞，本会論文賞等受賞，電子情報通信学会，IEEE 各会員．