社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－CVIM－166 (36)
2009／3／14

誤差による変動幅を考慮した決定木に関する研究

中田 裕介† 和田 俊和†

†和歌山大学システム工学研究科 〒640-8510 和歌山県和歌山市栄谷 930
E-mail: †{nakata, twada}@vrl.sys.wakayama-u.ac.jp

あらまし クエリとの類似度を最大化するデータを，大量のデータ中から探索することを「最大類似度探索」と呼ぶ．一般に，類似度は距離の公理を満足する尺度に変換できないため，最大類似度探索の代替として最近傍探索を用いることはできない．最大類似度探索では，高速性とともに，ノイズ等によってパターン間の類似度が変化した場合にも妥当な探索が行えるという正確性も要求される．本稿では，パターンの変動による類似度の変化を許容する決定木を構築し，それを用いて最大類似度探索を行う方法を提案する．本手法で用いる決定木は，各ノードにおいて参照パターンとの類似度を計算し閾値と比較（テスト）することによって，サーチパスを決定する．したがって，その類似度が変動した場合，サーチパスが変化するため，誤った解を探索するが，この変動幅があらかじめ分かっていれば，誤りができるだけ生じないようにテストの順序を決定できる．これは，各パターンの類似度の変化区間をできるだけ分断せず，しかも木の深さが平均的に浅くなるように決定木を作ることを意味する．本手法では，閾値に応じて変化するポジティブ・ネガティブのラベル付けを行い，このラベルに基づいてエントロピーゲインを計算し，この値を最大化するテストを順次選ぶことで，変動の影響を受けにくい決定木を構築する．

キーワード 最大類似度探索，決定木，エントロピーゲイン，

A Robust Decision Tree Based Maximum-Similarity Search

Yusuke NAKATA† Toshikazu WADA†

† Graduate School of Systems Engineering, Wakayama University 930 Sakaedani, Wakayama, 640-8510 Japan
E-mail: †{nakata, twada}@vrl.sys.wakayama-u.ac.jp

Abstract This paper presents a method to construct a decision tree for maximum-similarity search. Maximum-similarity search is to find a data providing the maximum similarity with a given query from a database. For this task, the search speed and the robustness against distorted query are both important. Our decision tree algorithm determines the search path by comparing a given threshold with the similarity value between a reference and a query at each inner node. This comparison is often referred as “testing”. The similarity value, however, can be affected by distortion of the query, and thus, the search path can be changed. The goal of this research is to construct a robust decision tree against distorted queries. This problem is equivalent to find near optimal testing sequences by analyzing the given intervals of the similarity fluctuations. In this report, we propose 1) a dynamic two-class labeling that depends on the threshold and 2) an algorithm for finding the locally optimal threshold maximizing the “entropy gain” under the labeling.

Keyword Maximum similarity search, Decision tree, Entropy gain

1. はじめに

最大類似度探索の問題には，指紋等の生体情報の検索，類似画像や音声の検索，などの実用的意義が高いものが多数存在しており，特に指紋等に関しては大規模なデータベースに対して高速な検索を行う方法が強く求められている．

この問題に類似した問題としては，最近傍探索がある．最近傍探索では，距離の公理に含まれる三角不等式を利用した距離計算対象の削減や，Persevalの等式を利用した距離計算の打ち切り，などが利用され，高速な探索アルゴリズムが開発されてきた．特に，近年では厳密な最近傍ではなく近似最近傍解を求めるアルゴリズムが詳しく研究され，ANN[1][2]，LSH[3][4][5]，PCH[6][7]などが例として挙げられる．

一方，最大類似度探索では，類似度が満たすべき条件が厳密な意味で定義されていないため，高速化，精度，省メモリの条件をすべてクリアした探索法は存在しない．この分野の先駆け的な研究としては前田ら[8]の研究がある．前田らは，指紋データを対象として，全データ間の類似度を要素とするマッチングスコア行列を用意し，これを利用してクエリとマッチする行列を高速に探索する方法を提案している．この方法は，クエリが歪んだ場合にもほぼ正しい結果を探索することができる半面，マッチングスコア行列が巨大になるため，登録するデータの2乗のオーダのメモリを消費するという欠点がある．これに対して，中田ら[9]は最大類似度探索で取り扱う非固定次元データをm個のリファレンスデータとの類似度を求めることによりm次

元のベクトルに変換し，最近傍探索によって省メモリかつ高速に最大類似度探索が実現できることを示した．この方法は，一般的な問題ではある程度うまく機能するが，使用する類似性尺度によっては不正確な結果しかもたらさないことがある．例えば，指紋でよく用いられる類似度は，本人以外の指紋との類似度は極めて低く，同一人物の指紋同士であれば非常に高い値となる．このように，異なる組み合わせでは値が低く，正しい組み合わせのときだけ類似度の値が上昇する場合は，前述のように固定ベクトルを発生させて最近傍探索を行う方法では探索が不安定になる．特に，クエリが僅かでも歪むと類似度が低いデータとの類似度が大きく変化し，リファレンスに最大類似度をもたらすデータが含まれていない限り探索は不正確になる．

本研究では，「あらかじめ各データの歪みとそれによる類似度変化の幅が分かっている」ということを仮定し，これを利用してクエリの歪みに対して安定した探索を行うことを目的としている．当然のことながら，メモリ消費は前田らの方法よりも低く抑える必要がある．

低消費メモリでデータの探索を高速に行うには，ハッシュや木探索を用いる方法が考えられるが，ハッシュ型のアルゴリズムではハッシュビンの幅をデータ変動幅に応じて変更することが必要になる．これは，1）オーバラップしたハッシュビンを設定するなどの問題が出てくるためアルゴリズムが煩雑になること，2）常に決まったリファレンスとの比較しか行われずクエリに応じたリファレンスの切り替えが行えない，といった問題点がある．これに対して木探索では，1）クエリに応じたリファレンスの切り替えが行える，2）探索すべきデータのインデックスを複数の葉に分散させ冗長性を持たせることで探索の安定化を図ることができる，というメリットがある．このため，本研究では木探索に基づく最大類似度探索を行うことにする．

木探索で最大類似度探索を行うということは，何らかの条件判断（テスト）を繰り返し，最終的に入力が何と最も近いのかを判定することであり，これは決定木によって実現することができる．本稿では様々なリファレンスデータとの類似度を閾値処理することをテストとする決定木を構築し，これによって最大類似度を与える最大類似度探索アルゴリズムを提案する．この際に，各リファレンスに対する類似度がデータの歪みによって変動する幅を事前に調べておくことができる．これを利用して出来上がる決定木を最適化することが狙いである．

決定木では，あるテストを行うことによって，全データを 2 つの集合に分けるという操作を行う．この 2 つの集合は互いに素でなくてもよい．この分割を繰り返し，単一のデータしか含まないノードを作り，これを葉ノードとする．すべての葉ノードが出来上がった時点で決定木の構築は終わり，これを用いて探索を行うことができる．

通常，決定木を構築する際にテストの順番を適当に設定すると，非常に大きな木構造が出来上がり，これが汎化性能と動作速度の低下を引き起こすことが知られている．この問題を根本的に解決するには，全てのテストの組み合わせを試さなければならない．このような計算は非常に煩雑であるため，実際には，エントロピーゲイン あるいは gini gain と呼ばれる量を最適化するテストを採用することで準最適な決定木の構築が行われる．このような指標を用いることによって，あるリファレンスデータが決まると，指標を最適化する閾値が自動的に決まる．したがって，リファレンスと閾値の組で決まるテストが，リファレンスだけで決められるようになる．さらに，リファレンスを入れ替えた際に，各テストがもたらす指標の大小を比較することで，どのリファレンスデータを用いるべきかが判断できるようになる．本研究では，エントロピーゲインを用いて各テストのスコアリングを行い，分割されたデータに対して同じ計算を繰り返すことによって決定木の構築を行う．

しかし，エントロピーゲインを計算するには，データを複数のクラスに分類する必要がある．ここで，各データそのものをクラスと考えると，クラス数が膨大である半面，各クラスを構成するトレーニングサンプルが少なすぎるため，安定した計算が行えない．そこで本研究では，類似度の変動幅を利用して，全データを閾値に応じて 2 クラスに分類する方式を提案する．これにより，安定にエントロピーゲインが計算できるようになり，決定木の構築が行える．

以下，関連研究を紹介し，次に提案手法並びに実験結果について述べる．

2. 関連研究

ここでは，前田ら中田らの手法について述べる．

2.1. 前田らの方法

前田らの提案したマッチングスコア行列を用いた最大類似度探索は高速性と精度の両面で優れた手法である．

マッチングスコア行列とは登録された N 個のデータ $x_1 \ldots x_N$ の全ての組み合わせに対する類似度を要素として持つ行列である．データ a と b のマッチングスコアを $M(a,b)$ で表す．この場合，マッチングスコア行列 X は以下のように表わされる．

$$X = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,N} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N,1} & v_{N,2} & \cdots & v_{N,N} \end{pmatrix} \tag{1}$$

但し $v_{i,j} = M(x_i, x_j)$， $i, j = (1,2,3,\cdots,n)$ である．

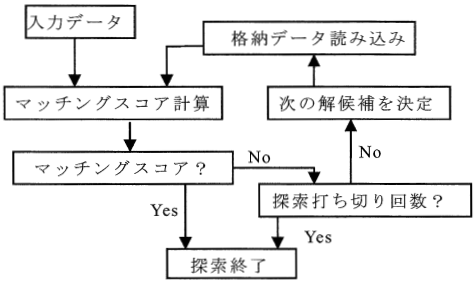

図 1. 前田らの方法の探索フローチャート

上記のマッチングスコア行列を用いて，以下に示す手順で入力との類似度が最大となるベクトルを求めることができる．

1. 入力データを x とする． ループ変数 m を１とし，現在の解候補の添え字 $r(m)$ を一番に登録されているデータ，すなわち $r(m)=1$ とする．
2. 入力データ x と現在の解候補データ $x_{r(m)}$ とのマッチングスコア $M(x, x_{r(m)})$ を計算する．
3. もしマッチングスコア $M(x, x_{r(m)})$ が事前に設定した閾値を超えているならば，解を $r(m)$ として探索終了．
4. 繰り返し変数 m が事前に設定した閾値を超えているならば，入力データは登録されていないという結果を出力し，探索を終了する．
5. この解候補を決めるために，これまで計算してきたマッチングスコアによって作られるベクトル

$$\boldsymbol{v}(x,m)=(M(x,x_{r(1)}),M(x,x_{r(2)}),..,M(x,x_{r(m)})) \quad (2)$$

と，データベース内の各データに関して求めた同様のベクトル

$$\boldsymbol{v}(x_k,m)=(M(x_k,x_{r(1)}),M(x_k,x_{r(2)}),..,M(x_k,x_{r(m)})) \quad (3)$$

の二つのベクトルの間で，次に示す相関値

$$R(\boldsymbol{v}(x,m),\boldsymbol{v}(x_k,m))=\frac{2<\boldsymbol{v}(x,m),\boldsymbol{v}(x_k,m)>}{\|\boldsymbol{v}(x,m)\|^2+\|\boldsymbol{v}(x_k,m)\|^2} \quad (4)$$

を計算する．

6. $$r(m+1)=\arg\max_k R(\boldsymbol{v}(x,m),\boldsymbol{v}(x_k,m)) \quad (5)$$

および，$m = m + 1$ として，ステップ２に戻る．

以上が，前田らの方法であるが，次のような問題点もある．

１． データ登録時に，全登録データ間の類似度（マッチングスコア）を計算し，これをマッチングスコア行列と呼ばれる形式で記憶しておく必要があるため，登録に要する時間が非常に長い．具体的には N 個のデータに 1 個のデータを追加する際に，N 回の類似度計算が必要になる．

２． 探索時にも，上記マッチングスコア行列を参照するため，メモリ消費が極めて多い．具体的には N 個のデータの格納に，$O(N^2)$のメモリが必要になる．

これらの問題点があるため，前田らの手法は比較的小規模な問題にしか適用することができないという制限がある．

2.2. 中田らの方法

従来手法での問題点を解消する方法として，不定次元パターンを，部分空間法を用いて選択した独立性の高い複数のリファレンスデータとの類似度からデータを固定次元ベクトル化し，PCH による最近傍探索を行う手法である．

独立性の高いリファレンスデータを選択するのに部分空間を用いる．マッチングスコア行列の行をマッチングスコアベクトルと呼ぶことにする．このベクトルは，ある一つのリファレンスデータに対する全データの類似度から成るベクトルである．

1 つ目のマッチングスコアベクトルの選択方法は，マッチングスコアベクトルの要素の分散値 (σ^2) が一番大きいものを選ぶ．こうすることによって，他のデータとの偏りが少ないものを選ぶことが出来る．

$$\sigma^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 \quad (6)$$

マッチングスコアベクトルの要素　x_i
マッチングスコアベクトルの要素の平均　$\bar{x}$

2 つ目の軸となるマッチングスコアベクトルは 1 つ目の軸となるマッチングスコアベクトルとできるだけ無相関なベクトルを選ぶ．そのためには，1 つ目の軸となるマッチングスコアベクトルとの内積値が一番小さいものを選べばよい．

$$\text{内積値}=\frac{<\boldsymbol{A},\boldsymbol{B}>}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} \quad (7)$$

1 つ目の軸となるマッチングスコアベクトル　$\boldsymbol{A}$
2 つ目の軸となるマッチングスコアベクトル　$\boldsymbol{B}$
ここで，$<o,\times>$は内積を表す

3 つ目～n 個目の軸となるマッチングスコアベクトルは，今までに選んだ軸のマッチングスコアベクトルを考慮し，できるだけ無相関なものを選ぶために，以下のような方法を用いる．まず，今まで選んだマッチングスコアベクトルの正規直交行列 U をグラムシュミット正規直交化法で求める．次に U を含む部分空間との残差が最大となるマッチングスコアベクトルを選ぶ．

$$\boldsymbol{U}_1=\frac{\boldsymbol{V}_1}{\|\boldsymbol{V}_1\|} \quad (8)$$

$$\boldsymbol{U}_2=\frac{\boldsymbol{V}_2-<\boldsymbol{V}_2,\boldsymbol{U}_1>\boldsymbol{U}_1}{\|\boldsymbol{V}_2-<\boldsymbol{V}_2,\boldsymbol{U}_1>\boldsymbol{U}_1\|} \quad (9)$$

$$\boldsymbol{U}_n=\frac{\boldsymbol{V}_n-\sum_{i=0}^{n-1}<\boldsymbol{V}_n,\boldsymbol{U}_i>\boldsymbol{U}_i}{\left\|\boldsymbol{V}_n-\sum_{i=0}^{n-1}<\boldsymbol{V}_n,\boldsymbol{U}_i>\boldsymbol{U}_i\right\|} \quad (10)$$

但し，$<o,\times>$は内積を表す．このとき，残差は次式で表わされる．

$$\text{残差} = \left\| (I - \sum U_n U_n^T) V_n \right\| \qquad (11)$$

但し，Iは単位行列である．
この残差が最大となるマッチングスコアベクトルに対応するリファレンスデータを選択し，全データを固定次元化する．

これによって，得られる固定次元ベクトルに対して最近傍探索を行うのが，中田らの方法である．この方法は，省メモリ，高速性の2つの観点からみると，前田らの手法よりも優れている．しかし，指紋データに適用した場合，データが歪むと急速に正解探索の割合が減少してしまうという問題点がある．

3. 提案手法

本研究では，以上の問題点を解決するために，メモリ消費量を削減するために決定木を用いることにする．決定木の場合，前田らの手法のようにクエリによって動的にテストを切り替え，適切なサーチパスを生成できる可能性がある．

このための具体的な以下の手法を提案する．

1. 登録データの全データに対し，決定木を作る際の評価基準であるエントロピーゲインを計算する．
2. エントロピーゲインが最大のものであるデータを木のノードにする．
3. 1．2．を再帰的に行うことで，決定木を構築する．
4. 3．で構築された決定木を用いて探索を行う．

3.1. エントロピーゲイン

決定木を構築する際の基本方針としては，木の葉ノード付近では間違いが起こっても，木の根ノード付近では間違いが出来るだけ起こらないようにデータを選択する．このように木を構築することで，間違ったノードに進んだ場合でも，その周辺を探索することで目的のノードに到達できるようにし，探索精度を向上させることが狙いである．その際の評価基準はエントロピーゲインを用いることとする．

エントロピーゲインとは，データマイニングではID3と呼ばれている手法で使われている評価基準である．エントロピーゲインは，あるデータを親ノードとした場合，親ノードと子ノードの平均情報量の期待値（以下よりエントロピーと呼ぶ）により計算でき，計算で求めた値が最大のデータを木のノードにするという手法である．

エントロピー$ent(p)$は，ポジティブデータの割合を$p = \#\mathrm{P}/(\#\mathrm{P}+\#\mathrm{N})$，ネガティブデータの割合を$1-p = \#\mathrm{N}/(\#\mathrm{P}+\#\mathrm{N})$としたとき

$$ent(p) = -p\log_2 p - (1-p)\log(1-p)$$

で計算することが出来る．このエントロピーを用いてエントロピーゲインは以下のように求めることができる．親ノードでの全データ数$n = \#\mathrm{P}+\#\mathrm{N}$，ポジティブデータ数$m = \#\mathrm{P}$とすると，子ノードでのデータ数$x$，ポジティブデータ数$y$は図2のように表される．

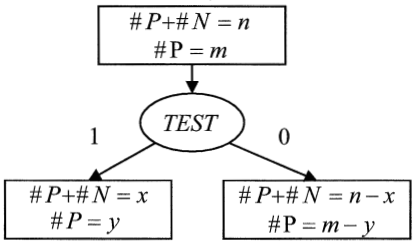

図2.　決定木におけるノード

このときエントロピーゲインは，親のエントロピーから，子のエントロピーを減算することにより求められるので，

$$Ent(x,y) = ent(\frac{m}{n}) - \frac{x}{n}ent(\frac{x}{n}) - \frac{n-x}{n}ent(\frac{m-y}{n-x})$$

で計算できる．

3.2. 動的なラベリング

3.1で説明したエントロピーゲインを実際どのように適用し，決定木を構築していくかを述べる．

誤差の範囲がわかっている場合，あるリファレンスとなるデータに対する全データが取りうるマッチングスコアの分布を図3に示す．ここで，決定木を構築するには，リファレンスデータごとの閾値が必要となってくる．こうすることで，親ノードとのマッチングスコアが，閾値より小さければ左の子ノードへ進み，閾値以上であれば右の子ノードへと進むような構造となる．

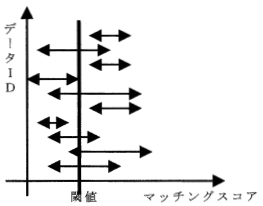

図3．あるリファレンスデータに対するスコア分布

エントロピーゲインを計算するためには，ポジティブデータとネガティブデータが事前に与えられてなければいけない．しかし，与えられているのは誤差のハインが与えられているマッチングスコアの値だけであるので，どのデータがポジティブで，どのデータがネガティブであるかは与えられてはいない．

この問題を解決するために，閾値が与えられたときにポジティブデータとネガティブデータを決定しなければいけない．この決定方法は，与えられているのはデータの誤差の範囲であるので，閾値に対してそのデータがどの程度の割合でポジティブデータに属しているかを計算し,割合が多い方に属しているようにする．

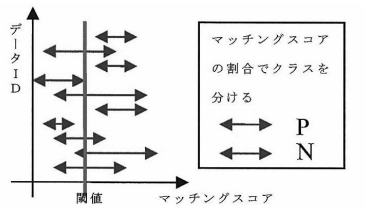

図 4.　動的なラベリング

以上のように，閾値によりポジティブデータとネガティブデータを決定した上で，閾値を移動させながらエントロピーゲインを計算する．最終的には得られたエントロピーゲインが最大の値をとるときの閾値として木を構築していく．木の葉ノードになるための条件とは，子ノードのデータ数 $x=1$ とし，再帰的に木構造を構築していく．

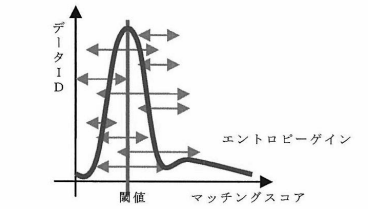

図 5. エントロピーゲイン曲線

3.3. 決定木を用いた探索

3.2 で構築された木構造を用いて探索を行う．

4. 実験

提案手法と，従来手法（前田らの手法）を比較するために実験を行った．実験環境は，以下に示す計算機を用いた．

CPU : Intel Pentium4 3.4GHz
Memory : 1GByte
OS : Fedora Core 6

4.1. データ

この実験で扱うデータ，National Institute of Standards and Technology（以下：ＮＩＳＴ）のオープンソースである NBIS パッケージに含まれる指紋データ 1000 個，類似度関数は同パッケージ内の Bozorth3 アルゴリズムを用いることとする．

4.2. 誤差

誤差については，ランダムを用いて発生させた人工のノイズを与えた．マッチングスコアを計算する Bozorth3 に入力するデータとは，指紋データから特徴を抽出したデータである xyt 形式のデータである．このデータの指紋の特徴であるマニューシャの x 座標，y 座標，角度 θ，品質 $(0\sim100)$ の 4 次元のデータが特徴の数だけ格納されている．この特徴の数は指紋データによってさまざまな値をとる．

ここで誤差を与えるのは，xyt 形式のファイルに与える．具体的には，以下の 3 種類を定義する．

- 挿入：ランダムに生成した特徴を 1 つ加える．
- 欠損：ランダムに指定した特徴を 1 つ削除する．
- 変動：x 座標，y 座標は変更せずに，角度 θ と品質をランダムで生成した値と置き換える．

以上の 3 種類の誤差を全特徴数から任意に与えたパーセンテージ分だけに対して行うことにより，指紋を採取する際の誤差の再現とすることとする．

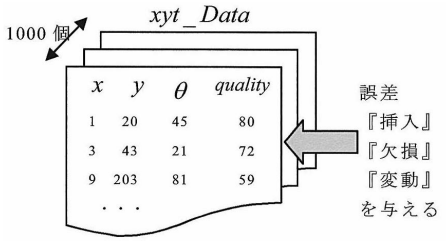

図 6.　指紋データとそれに加える誤差

4.3. 比較実験

従来手法でのパラメータは，探索打ち切り回数は 100 回，探索成功マッチングスコアは 40 として実験を行った．提案手法のパラメータとしては，探索成功マッチングコアは 40，木構造を構築する際に用いたデータは誤差なし，誤差 5%，10%，20%，30%，40%の 5 種類のデータで誤差の範囲を計算したデータを与えた．

入力データは 5%，10%，20%，30%，40%の誤差を与えた 1000 個の指紋特徴データである．ここでの誤差は，提案手法での木構造を構築した際とは別のものである．比較項目は，探索成功確率，探索時間，マッチングスコア計算回数，メモリ消費量の 4 点であり，

いずれも入力データを 1000 個与えたときの平均値である．

クエリに対して与えた誤差に対する探索精度，計算時間，類似度の計算回数，および消費メモリ量の比較を図 7-10 に示す．

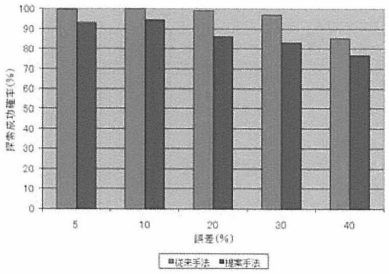

図 7．クエリに与える誤差と探索精度の関係

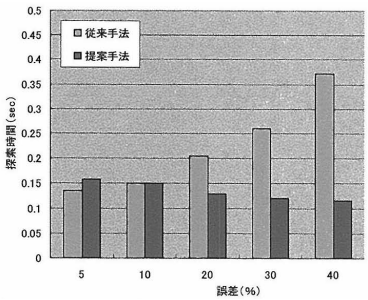

図 8．クエリに与える誤差と探索時間の関係

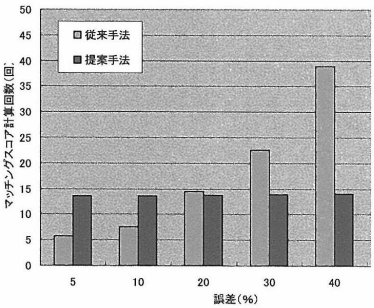

図 9．クエリに与える誤差と類似度計算回数の関係

実験結果より，探索の精度は従来手法よりも若干劣るものの，メモリ消費量および，計算速度は大幅に改善することができたと言える．

5. おわりに

本研究では，決定木を用いて類似度の変動幅を考慮した最大類似度探索法を提案した．決定木の構造，ならびに，各テストの閾値をエントロピーゲインに基づいて最適化することによって，パターン変動に対して頑健で，省メモリかつ高速な探索アルゴリズムの構築が行えた．

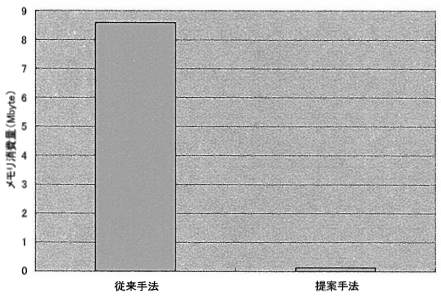

図 10．消費メモリ量の比較

今後は，単純に類似度の変動幅を与えるだけでなく，類似度変化の確率分布を与え，これをもとにしてエントロピーゲインを求め，より正確な最大類似度探索を行う方法について検討する予定である．

文　献

[1] S. Arya, , D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching,” Journal of the ACM, Vol.45, pp. 891-923, 1998

[2] ANN: Library for Approximate Nearest Neighbor Searching (http://www.cs.umd.edu/~mount/ANN/)

[3] P. Indyk and R. Motwani, “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality,” In Proceedings of the 30th ACM Symposium on Theory of Computing (STOC’98), pp.604-613, May 1998.

[4] M. Datar, P. Indyk, N. Immorlica, and V. Mirrokni,“Locality-Sensitive Hashing Scheme Based on p-Stable Distributions,” In Proceedings of the 20th Annual Symposium on Computational Geometry(SCG2004), June 2004.

[5] A. Andoni, P. Indyk, “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions”, In Proc. of FOCS'06, pp. 459-468, 2006

[6] 松下裕介，和田俊和，”Principal Component Hashing”，画像の認識・理解シンポジウム (MIRU2007)，pp.127-134，JULY 2007.

[7] Y. Matsushita, T. Wada, “Principal Component Hashing: An Accelerated Approximate Nearest Neighbor Search,” Pacific-Rim Symposium on Image and Video Technology 2009, PSIVT 2009 LNCS 5414, pp.374-385, 2009

[8] Takuji MAEDA, Masahito MATSUSHITA, and Koichi SASAKAWA, Regular Members , "Identification Algorithm Using a Matching Score Matrix”, IEICE TRANS.INF & SYST, VOL.E84-D.NO.7, pp.819-824, JULY 2001.

[9] 中田裕介，和田俊和，"ハッシュを用いた最大類似度探索に関する研究",画像の認識・理解シンポジウム（MIRU2008），pp. 1436-1443, 2008