社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
DE2006-3, PRMU2006-41(2006-06)

# 分散コーディングによる高次元の最近傍探索

小林 卓夫† 中川 正樹‡

†１Ｘ研究所 〒351-0005 埼玉県朝霞市根岸台 7-7-1-402
‡東京農工大学大学院工学研究科 〒184-8588 東京都小金井市中町 2-24-16
E-mail: †xeno@remus.dti.ne.jp, ‡nakagawa@cc.tuat.ac.jp

**あらまし** 本論文では、一つのベクトルを多数のベクトルで表現し、符号化して記憶するという分散コーディングの考えを利用して、高次元球面空間の高速な近似最近傍探索アルゴリズムを構築する。プログラムに実装し、人工データを用いて実験をした。高次元、大量のプロトタイプにおいて、代表的な最近傍探索プログラム”ANN”よりも優れた性能が得られた。

**キーワード** 近似最近傍探索，分散コーディング，k-d tree，Locality Sensitive Hashing

# Higher-dimensional Nearest Neighbor Search by Distributed Coding

Takao KOBAYASHI† and Masaki NAKAGAWA‡

†1X Laboratory 7-7-1-402 Negishidai, Asaka-shi, Saitama, 351-0005 Japan
‡Graduate School of Technology, Tokyo University of Agriculture and Technology
2-24-16 Naka-cho, Koganei-shi, Tokyo, 184-8588 Japan
E-mail: †xeno@remus.dti.ne.jp, ‡nakagawa@cc.tuat.ac.jp

**Abstract** In this paper we propose a fast approximate nearest neighbor search algorithm in a high dimensional spherical space using an idea called “distributed coding” which is to represent a vector by a set of many vectors and encode them efficiently. We implemented the algorithm and tested it with synthetic data. The results show that the proposed method exceeds a popular approximate nearest neighbor library, “ANN” in search time and accuracy in the case of higher-dimension and a large number of prototypes.

**Keyword** Approximate Nearest Neighbor, Distributed Coding, k-d tree, Locality Sensitive Hashing

## 1. はじめに

近年、さまざまな領域で情報の洪水ともいえる大量のデータが発生している。大規模データの中から類似データを見つけることは、情報の有効利用や何らかの知識獲得を行う課題に対して中心となる技術であり、実用的な類似データ探索技術の必要性が高まってきている。

類似データ探索は、データを数値ベクトルで表現すれば最近傍探索問題(Nearest Neighbor Search problem: NNS)に帰結される。NNS は計算幾何学分野において古くから研究されてきている。また、NNS の近似アルゴリズムである Approximate Nearest Neighbor Search (ANNS)も熱心に研究されている。しかしながら、計算幾何学における NNS、ANNS の研究の中心は計算量の理論的な分析であり、理論的に優れたアルゴリズムであっても、現実的な適用が難しい場合もある。

ANNS が応用できる実用的な分野の一例としてパターン認識がある。パターン認識では大抵の場合、特徴次元数は 20 以上、学習サンプルは時には数万から数百万以上になることがある。高次元、大量のサンプルの条件では、学習および識別時間が障害となるので、少ない学習サンプルで未知パターンを精度良く推定できる手法がよく研究されている。その一方、ドキュメント解析で、非常に大量のサンプル（約 1 億）を集めて、k-d tree またはハッシュ法により解析を高速に行なうアプローチの可能性が指摘されている[1]。

今後、高次元・大量データを扱う探索問題はますます重要になると思われる。そこで本研究では、次のような特徴を持ち、実用的に効果のある ANNS アルゴリズムを提案し、プログラムを実装する。

特徴の１つ目は、高次元（20 以上）、大量のデータ（1 万～100 万）で高速に動作することである。２つ目は、汎用ＰＣの性能で動作可能であること、および明快でシンプルなアルゴリズム構造を持つことである。

以下本論文では、2 章でまず ANNS の定義と既存手法について述べる。3 章で分散コーディングという考え方を用いて具体的な ANNS のアルゴリズムを示す。4 章で人工データを用いて探索性能の測定をする。こ

こでは提案手法と、代表的な ANNS アルゴリズムである k-d tree[2]とを比較する。5 章では応用について述べる。6 章でまとめとする。

## 2. ANNS の定義と既存手法

NNS (ANNS) の定義は次の通りである：

「距離空間上$(S, D)$に $m$ 個のプロトタイプからなる集合 $P=\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m\}$とクエリ $\boldsymbol{q}$ が与えられたとき、クエリに最も近いプロトタイプを（なるべく精度良く）見つけよ。」

距離空間にはさまざまなものがあるが、$(R^d, L_2)$（実数空間におけるユークリッド距離）は実用的に意義がある。

以下に、2 つの既存の ANNS 手法を紹介する。

k-d tree は代表的な NNS アルゴリズムである。座標軸に垂直なパーティションで領域を分割することを繰り返して木構造を作成する。探索は、クエリの属する領域の近くだけを探索する。探索する範囲を少なくして精度を犠牲にすることで ANNS とすることができる。

Locality Sensitive Hashing (LSH)[3]は、ANNS を ε-PLEB という問題に変換し、距離の近い点どうしはハッシュ値が同じになる確率が高いという性質を持つハッシュ関数のアイデアを使用している。LSH はハミング距離空間の探索問題に適している。

さて、パターン認識ではしばしば類似度として内積が用いられる。この場合、パターンの特徴ベクトルは $d$ 次元球面上$(S^{d-1})$の点になる。こうする理由は、パターン認識では多くの場合ベクトルの大きさは特徴量として本質的ではなくベクトルの方向が重要であると考えられているからである。$(S^{d-1}, L_2)$では、内積を求めることで 2 点間の距離がわかる。

本論文では、一般のデータ探索問題を $S^{d-1}$ 上の NNS に落とし込むことができるという前提で、$(S^{d-1}, L_2)$の ANNS アルゴリズムを具体的に構築する。

また、探索精度を定義し、多数のクエリに対する平均の探索精度と平均の探索時間によって提案手法の評価を行う。

提案手法は LSH に似ているが、LSH はハミング距離空間以外に直接適用するのが難しい。

## 3. 分散コーディングによる ANNS

### 3.1. 分散コーディング

$\boldsymbol{x}$ を $S^{d-1}$ のベクトルとする。$Q_1, Q_2, \ldots, Q_{nQ}$ のそれぞれを離散的な超球面空間とする。$k=1\sim n_Q$ に対して、$Q_k$ の中で $\boldsymbol{x}$ に近い点をそれぞれ $n_k$ 個選ぶ。この点の全体を$\{\boldsymbol{e}_j\}$ $(j=1\sim \sum n_k)$とし、$\boldsymbol{x}$ の分散表現と呼ぶことにする。

2 つのベクトル $\boldsymbol{x}, \boldsymbol{y}$ の分散表現をそれぞれ$\{\boldsymbol{e}_j\}, \{\boldsymbol{f}_j\}$とするとき、$|\{\boldsymbol{e}_j\} \cap \{\boldsymbol{f}_j\}|$は、$\boldsymbol{x}$ と $\boldsymbol{y}$ の距離が近いとき大きく、遠ざかるに連れて急激に減少する。各 $\boldsymbol{e}_j$ は符号化することにより、格納、探索が高速に行なえるように記憶領域に保持することができる。

このような原理をもとにした各種手法を包括して分散コーディング(Distributed Coding: DC)と呼ぶことにする。

DC では、分散表現、符号化した分散表現とプロトタイプの記憶領域への格納、およびクエリの探索にさまざまな方法を採用することができる。そして、最近傍探索やパターン認識など、さまざまなアルゴリズムを構築することが出来る。

### 3.2. $S^{d-1}$ の領域分割

$S^{d-1}$ の任意のベクトル $\boldsymbol{v}=(v_i)$に対して、$k$ 桁の 2 進数 $B(\boldsymbol{v}, k)=b_0 b_1 \ldots b_{k-1}$ を

**If** $v_i>0$ **then** $b_i=1$, **else** $b_i=0$ $\quad (i=0\sim k\text{-}1)$ (1)

という操作で作成する。この $B(\boldsymbol{v}, k)$の値によって $S^{d-1}$ は $2^k$ 個の等面積の領域に分割される。この場合、分割境界は必ず座標軸に直交する超平面になっている。

直交行列 $T$ を使って、$T\boldsymbol{v}=\boldsymbol{w}$ と変換してから(1)の操作を行なえば、領域は任意の位置に移動される。直交行列をランダムに選べばいろいろな領域分割を行なうことができる。

こうして得られる領域分割操作で、2 つのベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2$ が同じ領域に入る確率、すなわち

$\Pr(\ B(\boldsymbol{v}_1, k)=B(\boldsymbol{v}_1, k)\ )$

は、$\boldsymbol{v}_1, \boldsymbol{v}_2$ のなす角（球面距離）によって決まり、距離が大きくなると急激に 0 に近づく。2 つのベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2$ のなす角を $\theta(\boldsymbol{v}_1, \boldsymbol{v}_2)$と書く。図 1 に $d$=30、$k$=5, 10, 20 で、なす角と確率 の関係を示す。

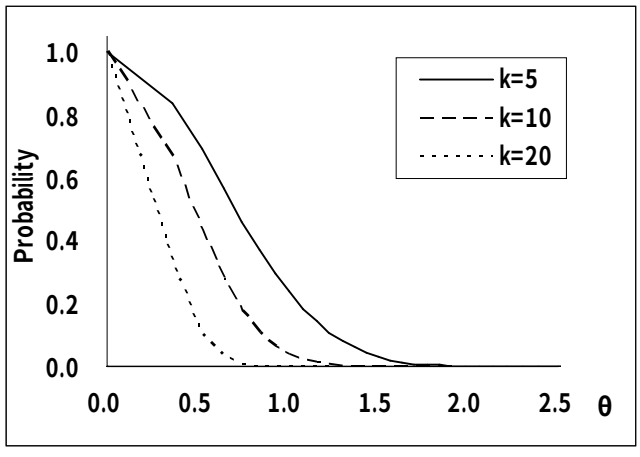


図 1　$\theta(\boldsymbol{v}_1, \boldsymbol{v}_2)$と $\Pr(\ B(\boldsymbol{v}_1, k)=B(\boldsymbol{v}_1, k)\ )$ の関係 ($d$=30)

Fig.1 Relation between $\theta(\boldsymbol{v}_1, \boldsymbol{v}_2)$ and $\Pr(B(\boldsymbol{v}_1, k)=B(\boldsymbol{v}_1, k))$ (d=30).

この性質を用いてANNSアルゴリズムを作ることができる。ランダムな $n_T$ 個の直交行列 $T_1, T_2, \ldots, T_{nT}$ を用意する。あらかじめ適切な $k$ を選択する。

**前処理**：プロトタイプ $\boldsymbol{p}_i$ ($i$=1～$m$)に対してそれぞれ $T_j$ ($j$=1～$n_T$)により変換し、(1)の操作をして $B(T_j\boldsymbol{p}_i, k)$を作る。$(B(T_j\boldsymbol{p}_i, k), j)$をキーにして $i$ を記憶領域に格納する。

**探索**:クエリ $\boldsymbol{q}$ を $T_j$ ($j$=1～$n_T$)により変換し(1)の操作をし $B(T_j\boldsymbol{q}, k)$を作る。記憶領域に$(B(T_j\boldsymbol{q}, k), j)$に一致するものを探して、そこに紐付けられているプロトタイプを収集する。収集されたプロトタイプの全てとクエリとで距離計算を行い最も近いものを探索結果とする。

探索処理において高速にキーを探索できるようなデータ構造には、ハッシュテーブル、ツリー、ソートされた配列などがある。

このアルゴリズムは、$n_T$ およびビット数 $k$ を適切に選ぶことにより、平均的に少数のプロトタイプとだけ距離計算を行えばよいので、探索時間を大幅に短縮できる可能性がある。

しかし、個々のクエリに対しては、クエリの近くにプロトタイプがたくさんある場合はクエリと同じ領域に含まれるプロトタイプの数が多くなるので距離計算回数が多くなり、逆にクエリの近くにプロトタイプがほとんどない場合は領域にわずかしかプロトタイプが含まれないので探索精度が低くなるという問題がある。

そこで、クエリの属する領域に入っているプロトタイプ数が多いときは領域を分割することを繰り返して、領域内のプロトタイプ数が常に一定の範囲に収まるようにする。修正したアルゴリズムは次の通りである。

**前処理**:プロトタイプ $\boldsymbol{p}_i$ に対してそれぞれ $T_j$ により変換し、(1)の操作をして $d$ 個の $B(T_j\boldsymbol{p}_i, k)$ ($k$=1～$d$)を作る。各$(B(T_j\boldsymbol{p}_i, k), j)$をキーにして $i$ を記憶領域に格納する。このとき、キーが$(B(T_j\boldsymbol{p}_i, k), j)$であるプロトタイプの個数がすぐにわかるようなデータ構造にする。このようなデータ構造としてはツリーまたはテーブル構造を採用することにより記憶領域を節約できる。

**探索**：探索の閾値 $h$ を決める。クエリ $\boldsymbol{q}$ を $T_j$ ($j$=1～$n_T$)により変換し、$k$=1 として(1)の操作により $B(T_j\boldsymbol{q}, k)$を作る。$(B(T_j\boldsymbol{q}, k), j)$をキーにして記憶領域を参照し、そのキーに属するプロトタイプの数を調べる。プロトタイプ数が $h$ より多ければ $k$ を 1 増やして繰り返す。そうでなければ、そのキーに紐付けられているプロトタイプを全て収集する。収集されたプロトタイプの全てとクエリとで距離計算を行い最も近いものを探索結果とする。

### 3.3. 使用メモリサイズ

記憶領域には、$i$=1～$m$ に対して、それぞれ $j$=1～$n_T$ について $B(T_j\boldsymbol{p}_i, d)$と $i$ を格納する。従って、必要なサイズは $m\times n_T\times(d+log_2 m)$ビットであり、仮に $m$=100 万、$d$=20、$n_T$ =200 とすると、

$1{,}000{,}000\times 200\times(20+log_2 1{,}000{,}000)$

$2^{20}\times 8{,}000$ bits = 1GB

となり、現在の PC のメモリにも収まる。

## 4. 人工データによる実験

### 4.1. 実験の方法

**第 3.2 節**で提示したアルゴリズムをプログラムに実装し、人工的なデータを用いて探索精度と探索時間の評価を行なう。

実験用のベクトルデータは次の方法で作成した。

$d$ 次元の原点 $O$ から勝手な方向に $\mu$ だけ離れた点 $A$ を 1 つ取る。$A$ を中心とする多次元正規分布でランダムにベクトルを生成する。このベクトルの大きさを 1 に揃える。$\mu$=0.0 とすれば生成したベクトルは $S^{d-1}$ 上に一様分布する。$\mu$>0.0 のとき $OA$ 方向に偏った分布となる。実験では $\mu$=0.0, 2.0 を採用し、プロトタイプとクエリが一様分布の場合と偏りがある場合の性能を測定した。なお、プロトタイプとクエリの出現分布は同じであるものとする。

正解の $k$ 個の最近傍を T=$\{\boldsymbol{t}_0, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{k-1}\}$とし、探索プログラムによって返された最近傍を A=$\{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}\}$とするとき、探索精度を次の式で定義する。

$$\text{探索精度} = \frac{|T \cap A|}{k} \qquad (2)$$

本論文では、多数のクエリで計測を行い，その平均の探索精度によって手法の性能を評価する。

また、既存の手法と提案手法を比較するため、公開されている k-d tree のソースコード "ANN"[4]を用いて同一の条件で性能の測定を行った。(以下では提案手法を DC と略記する。)

実験では、次元数 $d$=10, 20, 30, 40 プロトタイプ数 $m$=1 万、10 万、100 万、返却する最近傍の個数 $k$=1, 20 のそれぞれの条件について測定を行なった。

プログラムは C で記述し、VisualStudio で最適化オプションを付けてコンパイルし、実行モジュールを生成した。C++の ANN も同様に実行モジュールを生成した。

使用したマシンは、CPU:Celeron2.80Ghz、メモリ 2Gbyte、OS は Windows-Xp である。

### 4.2. 実験結果 (探索精度)

DC は、$n_T$ および $h$ を変えることにより、探索精度と探索時間が変化する。同様に、ANN では探索誤差パ

ラメータ *eps* を変えることにより、精度・時間が変わる。どちらの手法でも、探索精度と探索時間はトレードオフの関係になる。

探索パラメータをさまざまに変えて測定し、探索精度と探索時間の関係をプロットしたものを図 2 に示す。*k*=1, μ=0.0 における、{次元数、プロトタイプ数} = {10, 1 万}、{30, 100 万} の場合を示す。縦軸の探索時間は対数目盛りを使用している。

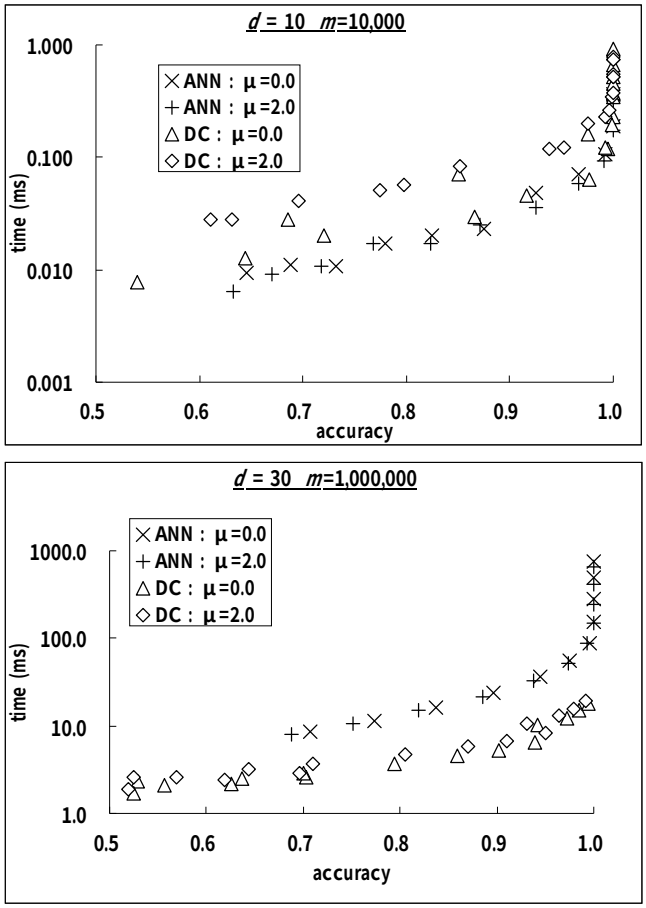


**図 2　探索精度と探索時間の関係** (k=1, μ=0.0)

Fig.2 Relation between the search accuracy and the search time (k=1, μ=0.0).

実験した範囲では、プロトタイプ数に関わらず、次元数=10 では DC は ANN より劣るが、次元数=20～40 では、DC の方が優るという結果になった。

本実験では、μ=0.0 と μ=2.0 で計測結果に有意な違いは見られなかった。従って、これ以降 μ=0.0 の場合のみ結果を示す。

さて次に、プロトタイプ数、次元数のファクターに対して 2 つの手法の性能の違いがどのように変化するかを見る。

2 つの手法において、(2)で定義した探索精度に基づき、精度=0.9, 0.99 となるところのパラメータを求めそこでの探索時間を測定する。そして、両者の探索時間の比を求める。

図 3 ではプロトタイプ数というファクターに対して、両者の性能の開きを見ている。例えば、k=20 精度=0.99 のときプロトタイプ数を 1 万から 100 万に増加させると、DC に対する ANN の探索時間は約 2 倍から 4 倍以上に上昇する。(すなわち DC は ANN より 4 倍以上速くなると読むこともできる。)このようにプロトタイプ数が大きくなるほど、DC は ANN に比べて相対的に速度が速くなることがわかる。

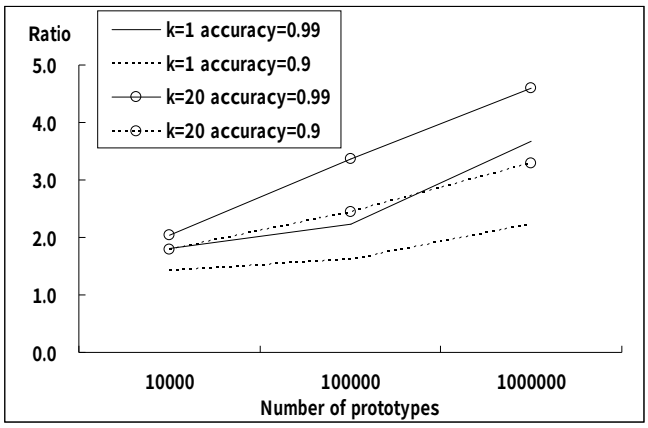


**図 3　プロトタイプ数に対する 2 つの手法の探索時間の比率**(ANN／DC) (*d*=20 **の場合**)

Fig.3 Relation between the number of prototypes and the search time ratio of ANN over DC (d=20).

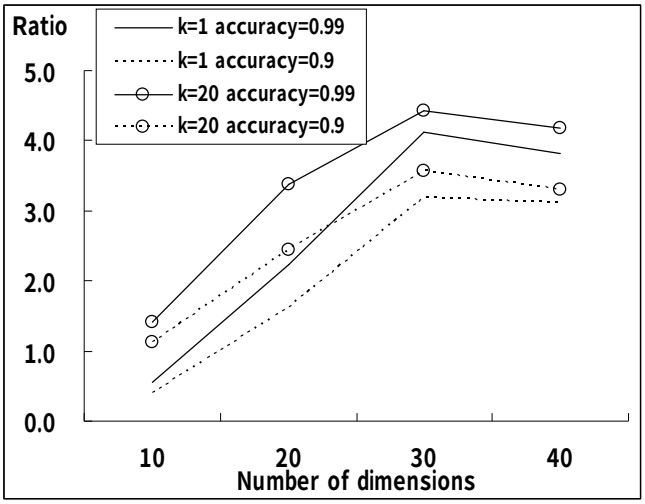


**図 4　次元数に対する探索時間の比率**(ANN／**提案手法**) (*m*=10 **万**)

Fig.4 Relation between the number of dimensions and the search time ratio of ANN over DC) (d=20).

次に、次元数に対する性能を見ることにする。図 4 にプロトタイプ数=10 万の場合を示す。次元数が大き

くなるとDCの探索速度はANNに対して速くなっていく。

しかしながら、次元数=40 になると倍率は下がってくる。この理由を次のように推測した。ANN の次元数=40 の探索時間を計測した結果、総当り探索に近いかそれを大きく超える時間となっていた。おそらく、ツリーの全ノードの探索に近い状態となり、探索空間の上界に押さえられると思われる。そのため、DC との探索時間の比が相対的に下がった。

また、返却する最近傍の数 $k$ については、絶対的な性能では当然 1 よりも 20 の方が低い。しかし、ANN との性能の比較では、$k$=20 の方が相対的に性能が良いという結果が観察された。このことは図 4、図 5 からも読み取れる。現実のデータ探索では、クエリに対して複数の候補を得て後処理に利用するというニーズは少なくないから、この性質は実用的に有効であるといえる。

次に、DC と総当り探索とを比較する。

図 5 より、DC はプロトタイプ数が多いほど探索時間削減の効果が顕著になることがわかる。プロトタイプ数=100 万では、概ね総当り法の 1/30～1/60 の探索時間となっている。

以上のことから、DC は高次元及び大量のプロトタイプの場合になるほど良い性能を発揮することがわかる。

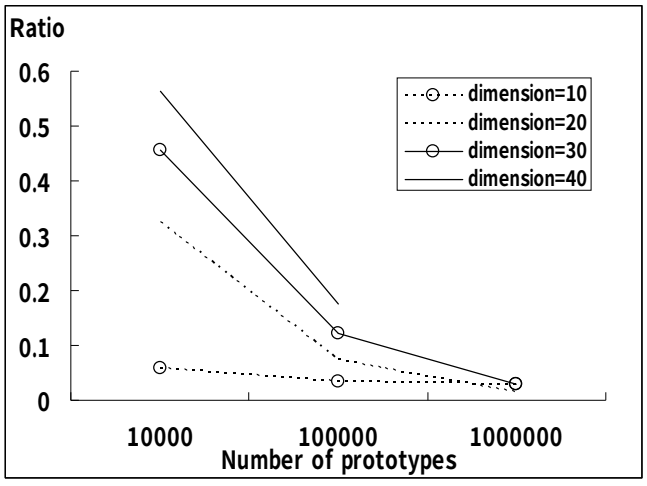


図 5 総当り探索と DC の探索時間の比率 (DC／総当り) (探索精度=0.9, k=1)

Fig.5 The search time ratio of DC over the brute-force method) (accuracy=0.9, k=1).

### 4.3. 実験結果 (メモリ使用量)

探索時のメモリ使用量は、ANN では次元数とプロトタイプ数のみによって決まる。一方、DC では探索精度によって必要なメモリ使用量が変化する。

DC において、探索精度を固定してメモリ使用量を計測すると、次元数、プロトタイプ数、返却する最近傍の数が大きくなるに従い、概ね ANN よりも相対的に多くのメモリを必要とする(図 6)。

例えば、d=20、m=100 万、k=1、探索精度=0.99 でのメモリサイズは約 1.3GB になる。実験環境の制約(搭載メモリ 2GB)により、次元数・プロトタイプ数が大きい場合の最適なメモリサイズが測定できなかったが、次元数=40 では、数ギガから 10 ギガバイト以上のメモリサイズを必要とすると推定される。ただし、図 6 で基準としているのは最良の探索パラメータにおけるメモリサイズであるので、探索速度または探索精度を若干落とすことによってメモリサイズをかなり小さくすることは可能であると思われる。

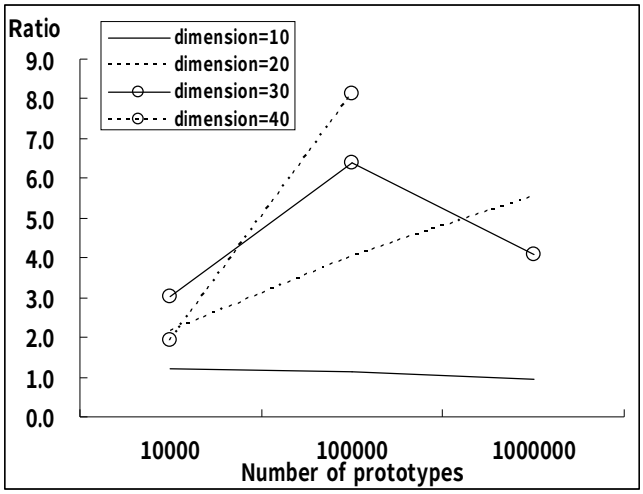


図 6 DC と ANN のメモリ使用量の比率 (探索精度=0.99, k=1)

Fig.6 The memory size ratio of DC over ANN) (accuracy=0.99, k=1).

## 5. 応用

分散コーディングの考え方は、データマイニングだけでなく、パターン認識へも適用することができる。

高速な ANNS は、パターン認識手法として識別精度が理論的に考察されている kNN 法[5]に利用することができる。一方、著者らは学習サンプルとの距離計算を行わないでクラスとの識別尺度を評価するモデルを提案し、その有効性を確認している[6]。

また、本論文のアルゴリズムを修正することにより、最近点対探索(Closest Pair Search) に応用が可能である。

## 6. まとめ

分散コーディングという考え方を用いて、高次元、

大量のプロトタイプに適用できる高速な近似最近傍探索手法を提案し、プログラムを実装し、実験により性能の評価を行った。

実験には人工データを用い、次元数、プロトタイプ数、返却する最近傍数のファクターについて調査し、探索精度と探索時間の関係を考察した。また、k-d tree を実装したプログラム ANN との比較を行った。

次元数=10 を超える場合で提案手法は ANN を上回る性能を示した。さらに、次元数、プロトタイプ数が大きいほど、性能の開きが大きくなる傾向が確認できた。

本提案手法は、高次元かつ大量のプロトタイプにおいて顕著な効果を発揮するので、現実の各種のデータ分析において有効であると考える。

また、提案手法はアルゴリズムがシンプルで、プログラムによる実装が容易であることも付け加える。このことは現実的な適用において利点となる。

また、本手法は現在の PC のメモリ容量で十分に動作可能である。

今後は、アルゴリズムに改良を加え、さらなる性能を向上させることと、最近傍探索以外の各種問題への応用を研究したい。

**謝辞**　実験作業に協力いただいた新井君に大変感謝いたします。本研究の一部は独立行政法人情報処理推進機構による未踏ソフトウェア創造事業の支援を受けている。また、本研究は萌芽研究 16650015 の（一部）補助による。

## 文　献

[1] H. S. Baird and M. R. Casey, “Towards Versatile Document Analysis Systems,” Proc. 7th IAPR Workshop on Document Analysis Systems, pp.280-290, Nelson, New Zealand, Feb. 2006.

[2] J. H. Friedman, J. L. Bentley, and R. A. Finkel. “An algorithm for finding best matches in logarithmic expected time.” ACM Transactions on Mathematical Software, 3(3):209-226, 1977.

[3] P. Indyk, and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” Proc. 30th ACM Symposium on Theory of Computing, pp.604-613, 1998.

[4] http://www.cs.umd.edu/~mount/ANN/

[5] T.M. Cover and P.E. Hart, “Nearest neighbor pattern classification,” IEEE Trans. Information Theory, vol.IT-13, no1, pp.21-27, Jan. 1967

[6] 小林卓夫, 中川正樹, “線形時間学習および定数時間識別の一パターン識別手法, ” 信学論, 採録決定済み.

## 付録：一様ランダムな $d$ 次直交行列を生成するアルゴリズム

$d$ 次直交行列は $d$ 個の互いに直交する単位ベクトルを行ベクトルとする行列である。従って、$d$ 個の直交する単位ベクトルの組をランダムに生成すればよい。アルゴリズムを以下に示す。

**Step-1:** 一様ランダムな方向の $d$ 個の $d$ 次元のベクトルを生成する（線形独立であるとする）。

**Step-2:** Gram-Schmidt の直交化法によりそれぞれを互いに直交させる（高々$O(d^3)$の乗算回数で行える）。

**Step-3:** 全てのベクトルを大きさ 1 に正規化する。