分散コーディングに基づく大規模データの高速探索技術とパターン認識への応用

1X研究所　小林卓夫

1992年

企業内にて文字認識の研究開発に従事

学習サンプル数に対して線形時間の学習、かつ定数時間の識別アルゴリズムを作成

2002年頃～

理論的な考察を行う

近似最近傍探索アルゴリズム作成

レジュメ

1. 分散コーディングの考え方
2. 高次元の近似最近傍探索
3. 高速なパターン認識

1. 分散コーディングの考え方

d次元の球面空間 S^{d-1}

$$\forall x = (x_1, \ldots, x_d) \in S^{d-1}, |x| = r$$

Q: 有限ベクトル集合，　k: 自然数

ベクトル x から Q への k-最近傍を

$Q(x, k)$

とする。（k=1 のときは特に $Q(x)$ とする。）

分散コーディング

Q_h (h=1,…,n_Q) : 離散的なベクトル集合とする

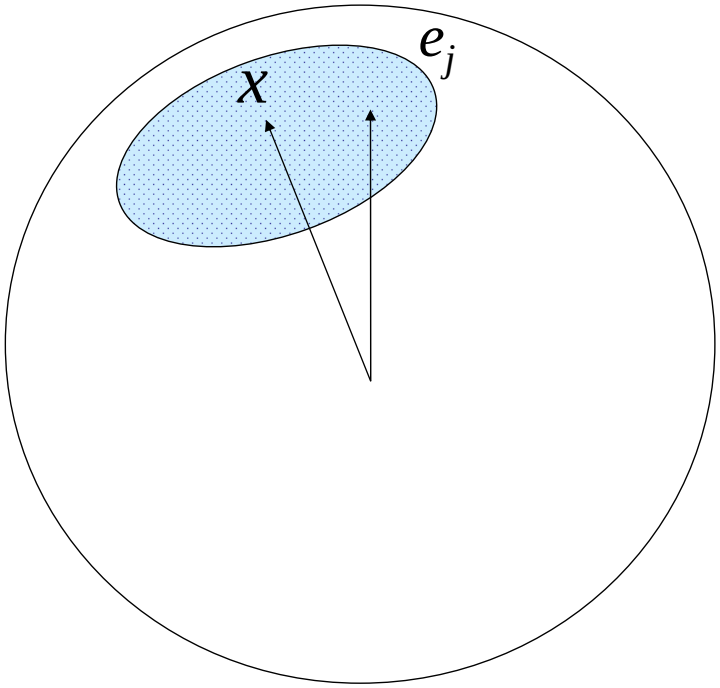
（※便宜的に、互いに共通部分がないものとする）

写像 D:

$$D(x) = \{e_j\} = \bigcup Q_h(x, k_h)$$

$\{e_j\}$を x の分散表現と呼ぶ。　$|\{e_j\}| = \Sigma k_h$

分散表現の概念的な図



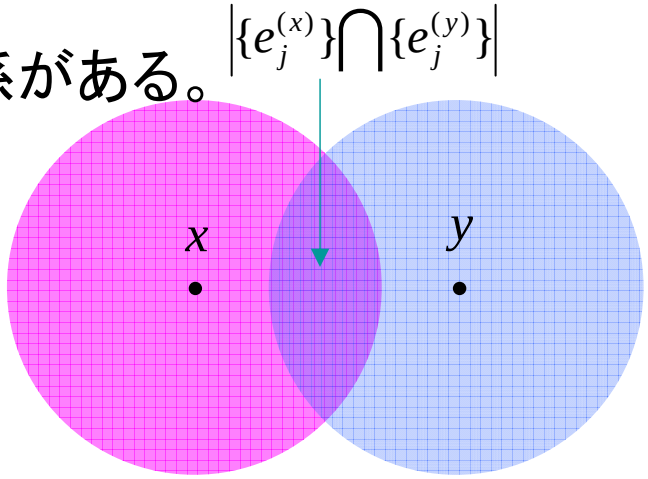
x, y の分散表現をそれぞれ

$$\{e_j^{(x)}\}, \{e_j^{(y)}\}$$

とするとき、

$$\left|\{e_j^{(x)}\} \bigcap \{e_j^{(y)}\}\right|$$

は、距離 $d(x, y)$ と関係がある。



（ちなみに…）

「Qの中で、xの最近傍を選ぶ」

||

Qによって空間Sをボロノイ領域分割

||

xをQでベクトル量子化

||

xを(locality sensitiveな)ハッシュコード化

(⇒Locality Sensitive Hashing: P. Indyk)

2. 高次元の近似最近傍探索

最近傍探索とは、（k-最近傍探索）

$$q, \{p_i\} = P, |P| = m$$

が与えられたとき、

$$T = \{t_1, \ldots, t_k\} \subset P,$$

$$d(q, t_j)_{t_j \in T} \leq d(q, p_i)_{p_i \in P-T}$$

となるTをみつけることである。

近似最近傍探索とは、

$$A = \{a_1, \ldots, a_k\} \subset P$$

$$\text{探索精度} : \frac{|P(q,k) \cap A|}{k}$$

探索精度がなるべく高いAを、
なるべく短い時間で探すことが目的。

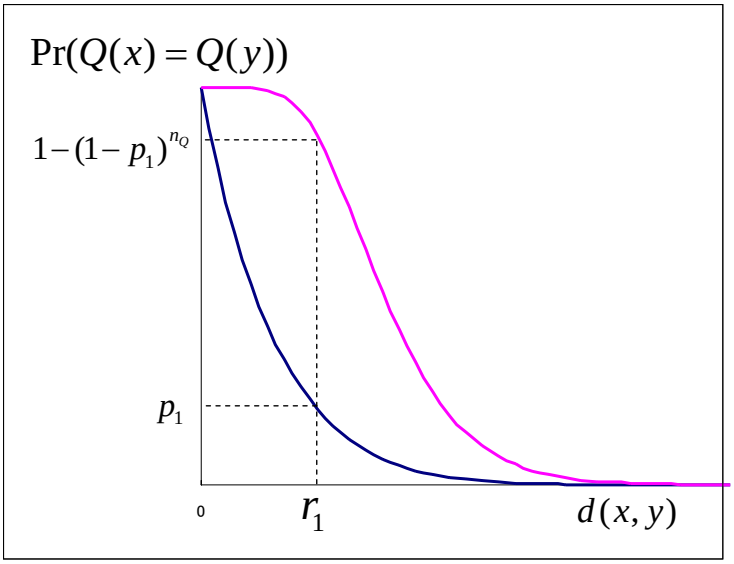
(※本研究では平均的な探索精度を用いている)

“Locality Sensitive”

$$\Pr(Q(x) = Q(y))$$

は、$d(x, y)$ が大きくなると、急激に小さくなる。

(高次元の場合)



Qを独立に
多数用意する
ならば…

探索処理では、クエリ q は

$$\left|\{e_j^{(q)}\} \bigcap \{e_j^{(p_i)}\}\right| \geq 1$$

であるプロトタイプp_iとだけ距離計算を行えばよい。(これは P のうちのほんのわずかであることが期待できる)

課題：

◎どのような領域分割を行うか？
（＝どのようなハッシュ関数を使うか？）

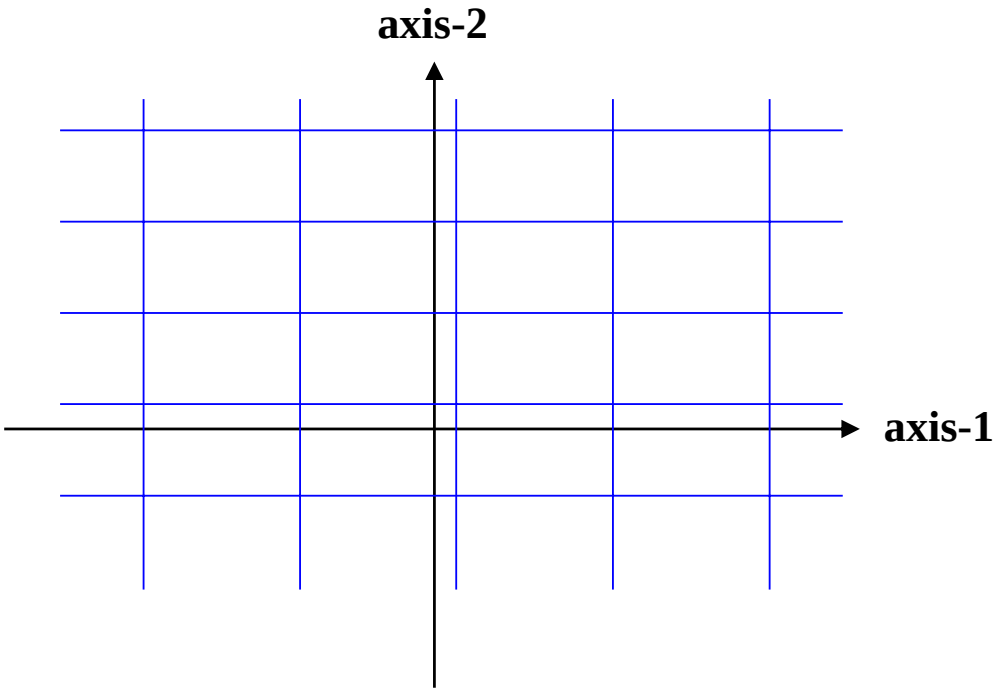
◎どのような領域分割の仕方が存在するか？
（＝どのようなハッシュ関数が存在するか？）

◇領域分割が粗いと、
１つの領域にたくさんのプロトタイプが入る。
⇔領域分割が細かいと、ほとんどの領域は空。

◇細長い領域だと long tail (fat tail)。
（なるべくならば ”球”に近い形がよい。）

◇ハッシュ関数が速く計算できること

LSH (P. Indyk)

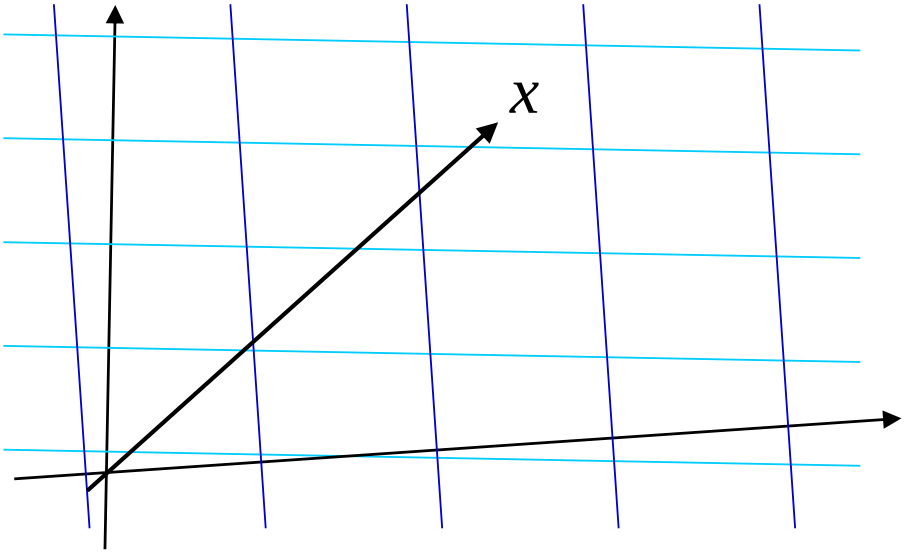


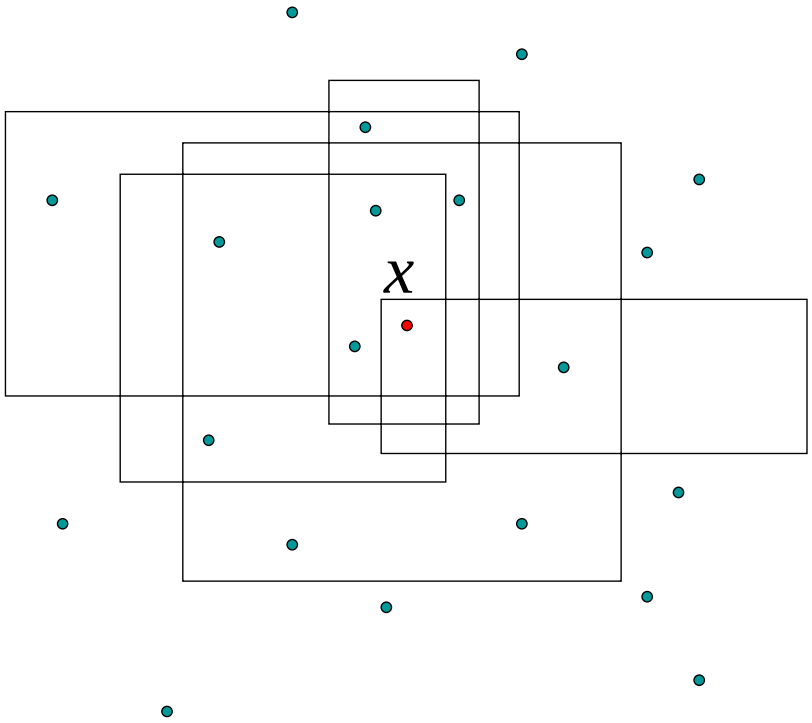
E^{2}LSH (Exact Euclidean LSH) (P. Indyk)

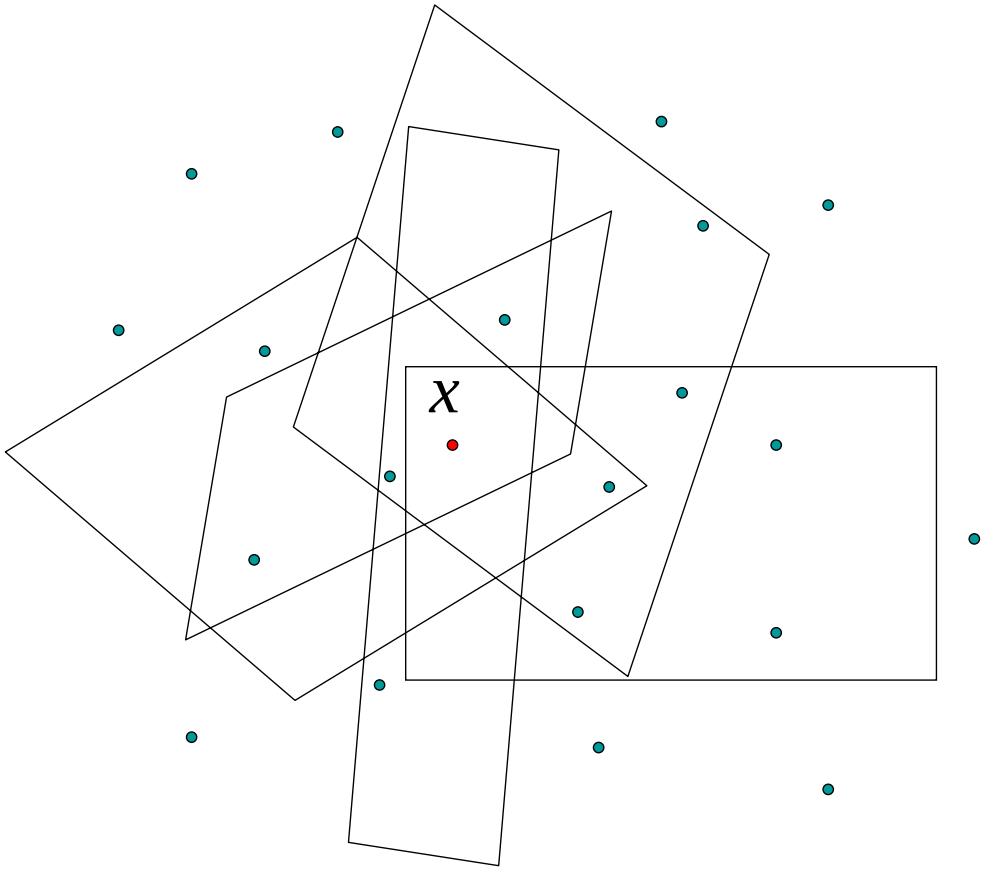
$$h_{a,b}(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor$$

a : 方向はランダム
$|a|$: ガウス分布
b : 平行移動操作 ($0 \leqq b < w$)

$$H(x) = \left(h_{a_1,b_1}(x), ..., h_{a_n,b_n}(x)\right)$$







一様ランダムな直交行列を使用する方法 (小林)

$\boldsymbol{v} = (v_0, v_1, \ldots, v_{d-1}) \in S^{d-1}$

$B(\boldsymbol{v}, k) = b_0 b_1 \ldots b_{k-1}$ (s.t. **if** $v_i > 0$ **then** $b_i = 1$ **else** $b_i = 0$)

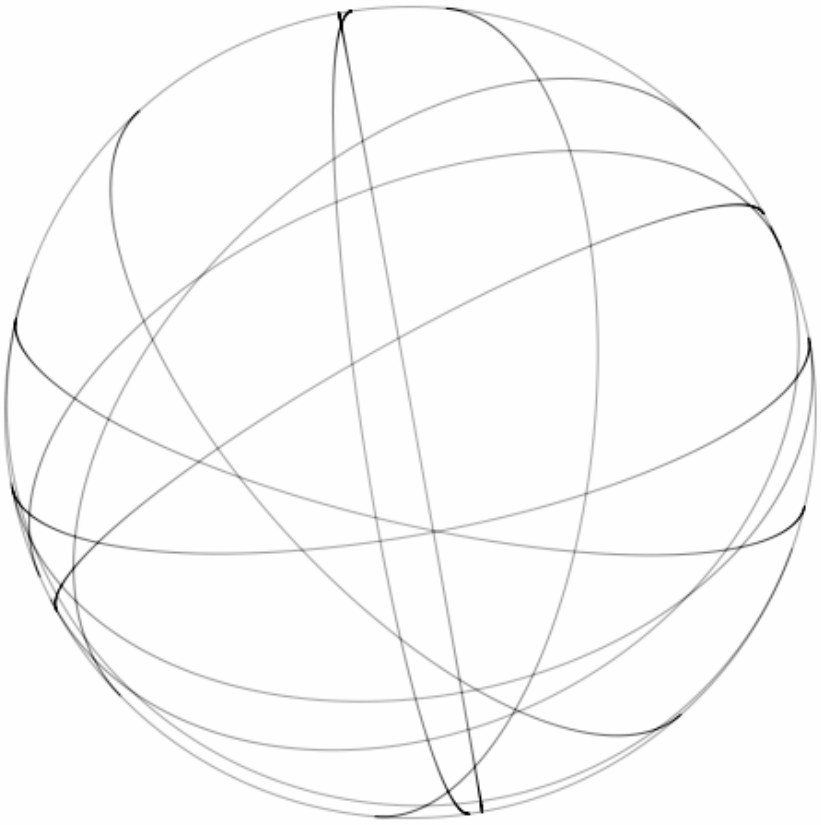
S^{d-1}を2^k個の等面積の領域に分けることができる。

直交行列Tを用いて、

$T\boldsymbol{v} = \boldsymbol{w}$

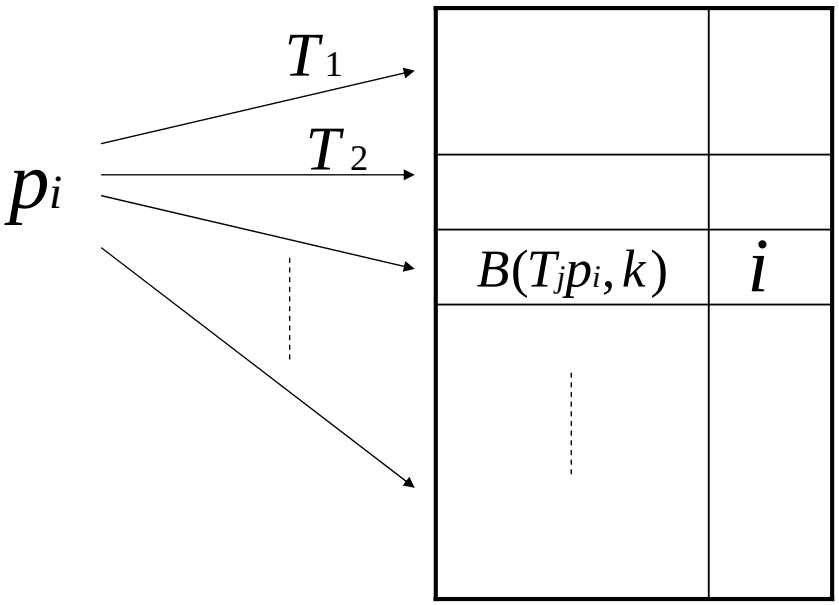
と変換すれば、いろいろな領域分割を行うことが出来る。

概念図（3次元, 2ビットの場合）

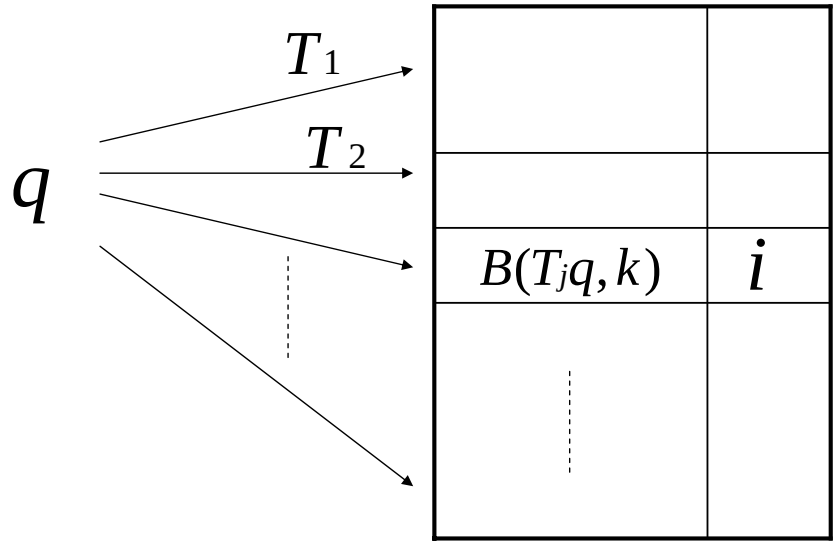


アルゴリズムの作成（1．1）（データ構造作成処理）

たくさんのランダムな直交行列 $T_1, T_2, T_3, \ldots$



アルゴリズムの作成(1．2) （探索処理）



最近傍候補の集合：

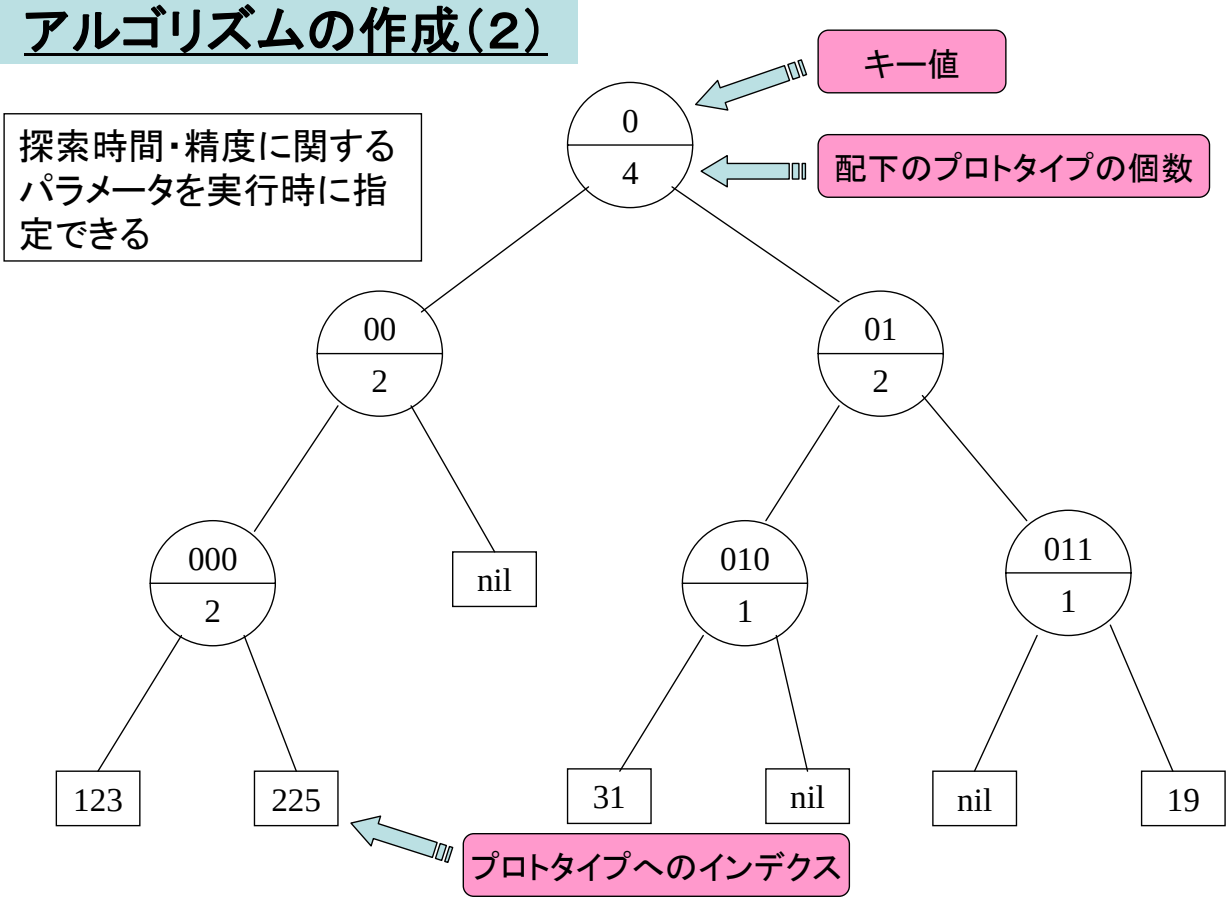
$$\{p_i \mid B(T_j q, k) = B(T_j p_i, k)\}$$

距離計算をする：

$$\min\{d(q, p_i)\}$$

アルゴリズムの作成(2)

探索時間・精度に関するパラメータを実行時に指定できる



人工データによる実験

●ベクトルデータをランダムに生成する。

（※本実験では一様ランダム）

・次元数＝10，20，30，40

・プロトタイプ数＝1万、10万、100万

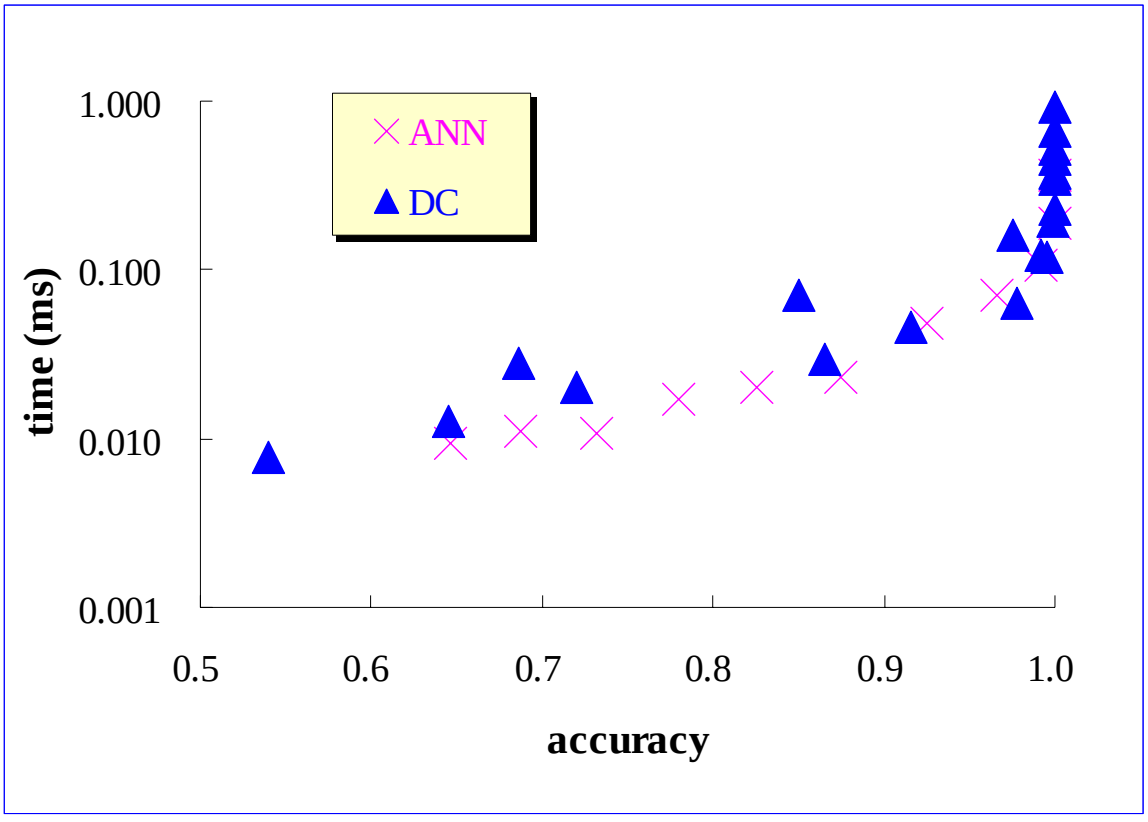
・クエリに対して返却する最近傍の数＝1個、20個

●探索時間　・探索精度　・メモリ使用量　を計測する。

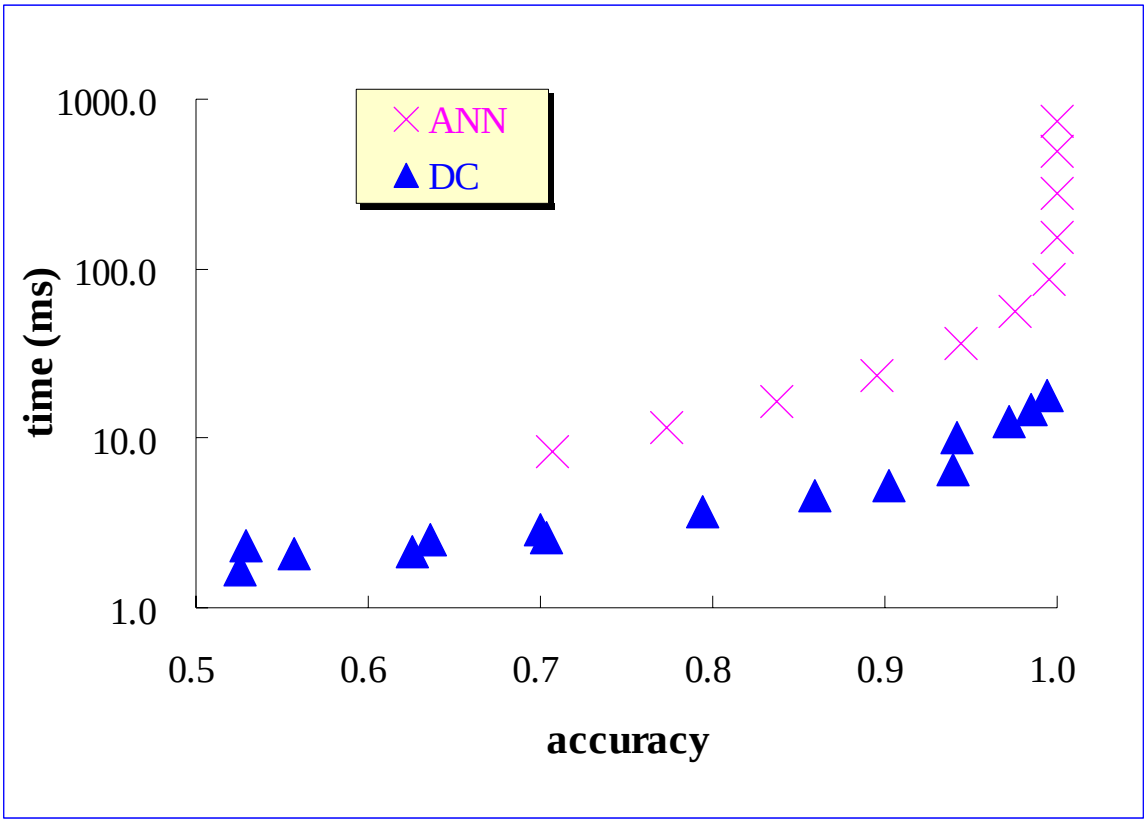
●既存の手法 (k-d tree) と比較する。　…（“ANN”）

* ANN: A Library for Approximate Nearest Neighbor Searching (D. M. Mount and S. Arya)

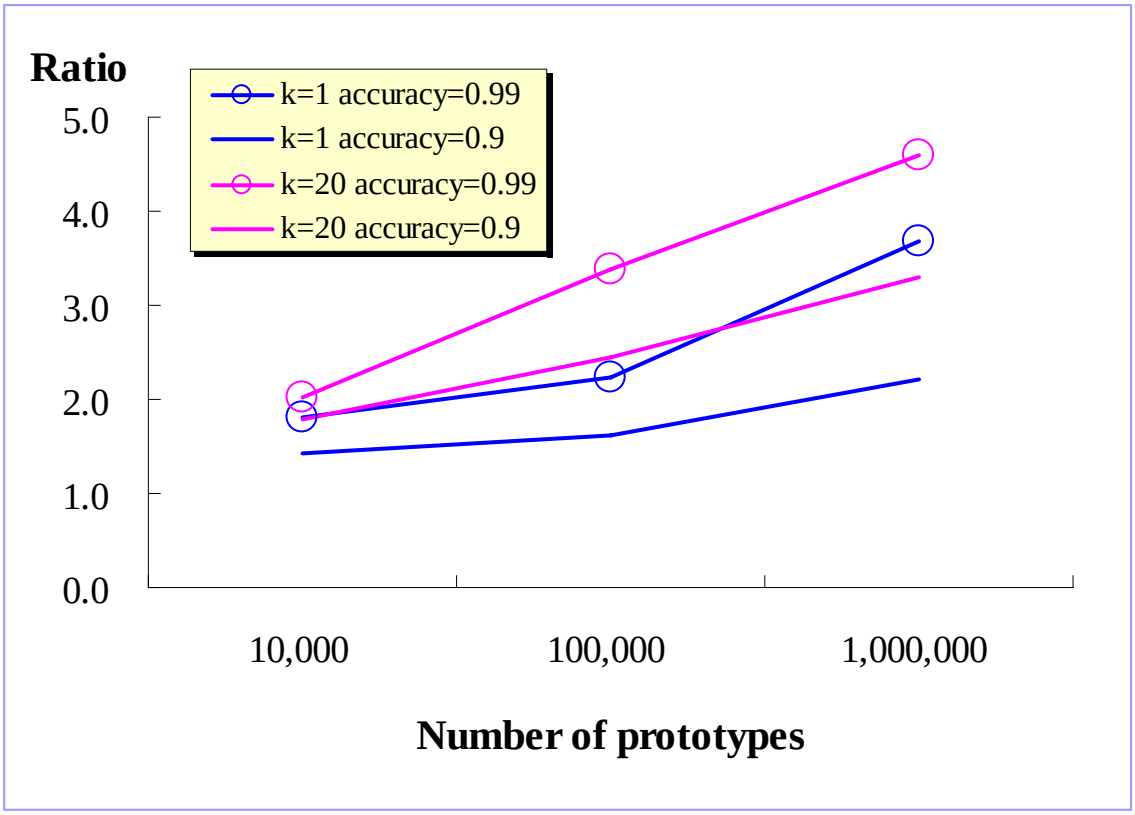
探索精度と探索時間の関係（次元数=10, プロトタイプ数=1万, k=1）



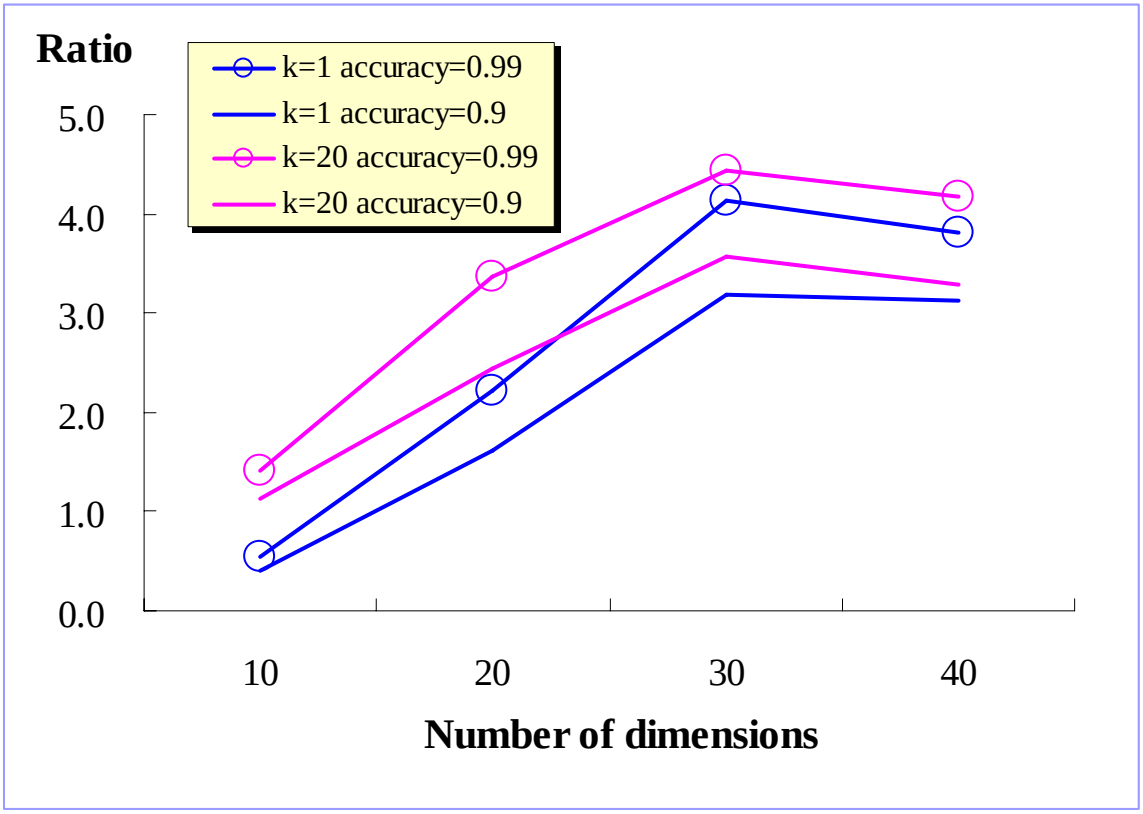
探索精度と探索時間の関係（次元数=30, プロトタイプ数=100万, k=1）



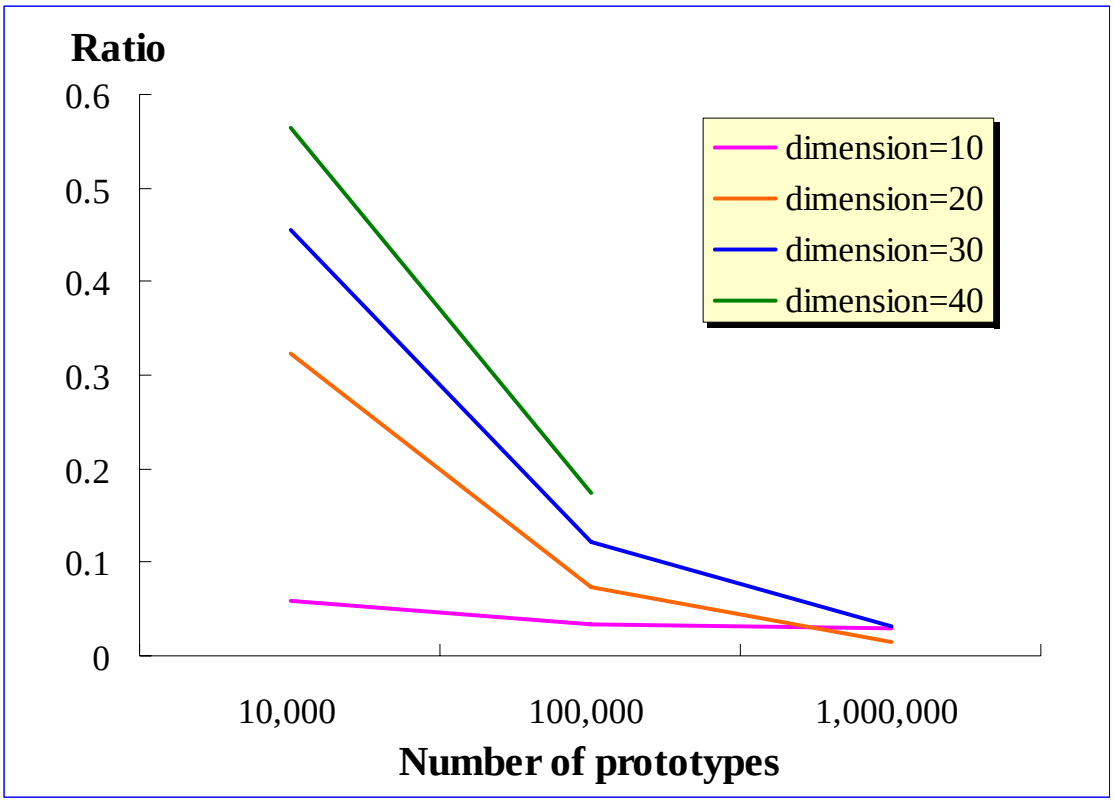
探索時間の倍率（ANN／DC）（次元数=20）



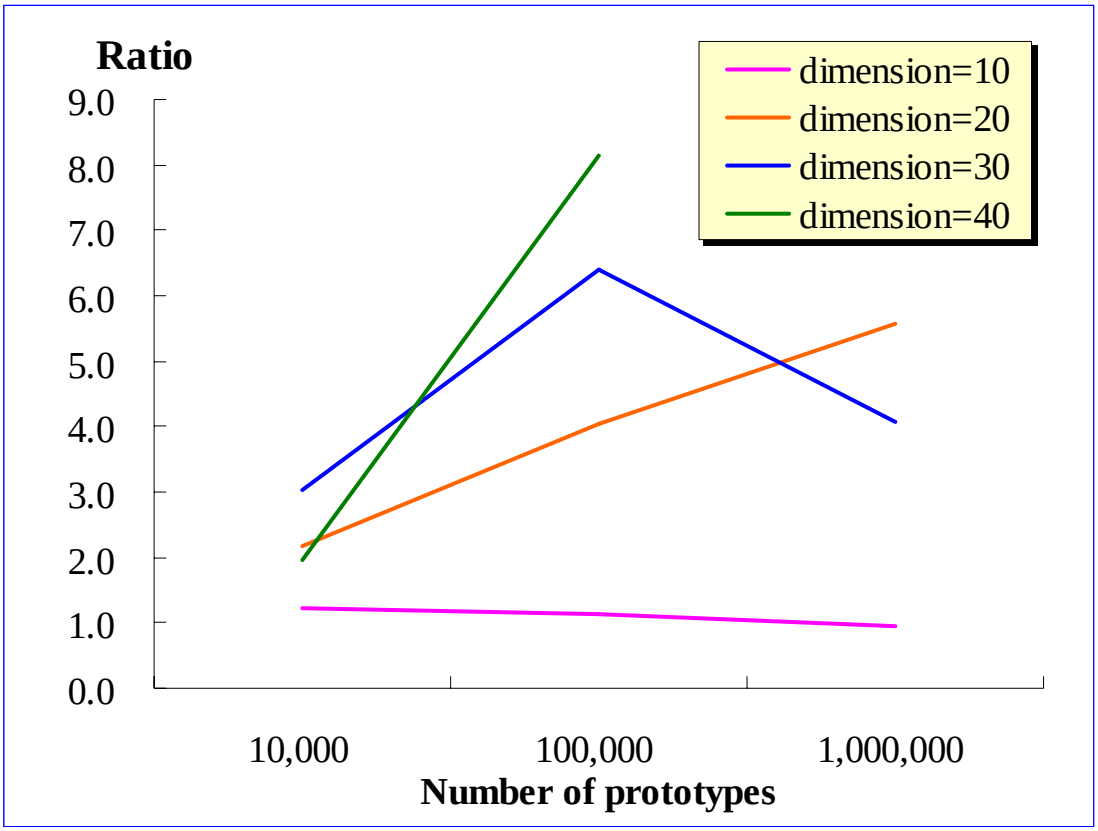
探索時間の倍率（ANN／DC）（プロトタイプ数=100,000）



総当り探索に対するDCの探索時間の比率



メモリ使用量の比率（DC／ANN）（※探索精度=0.99 k=1）

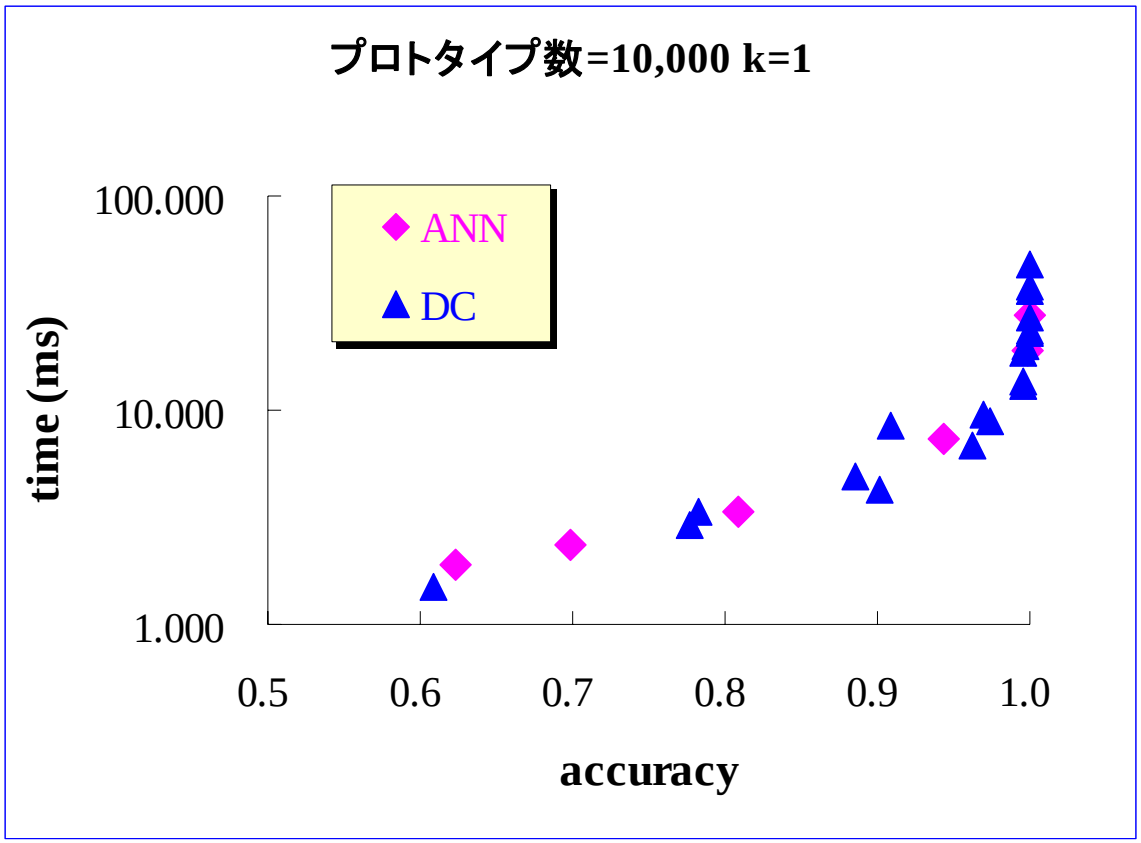


MNISTによる実験

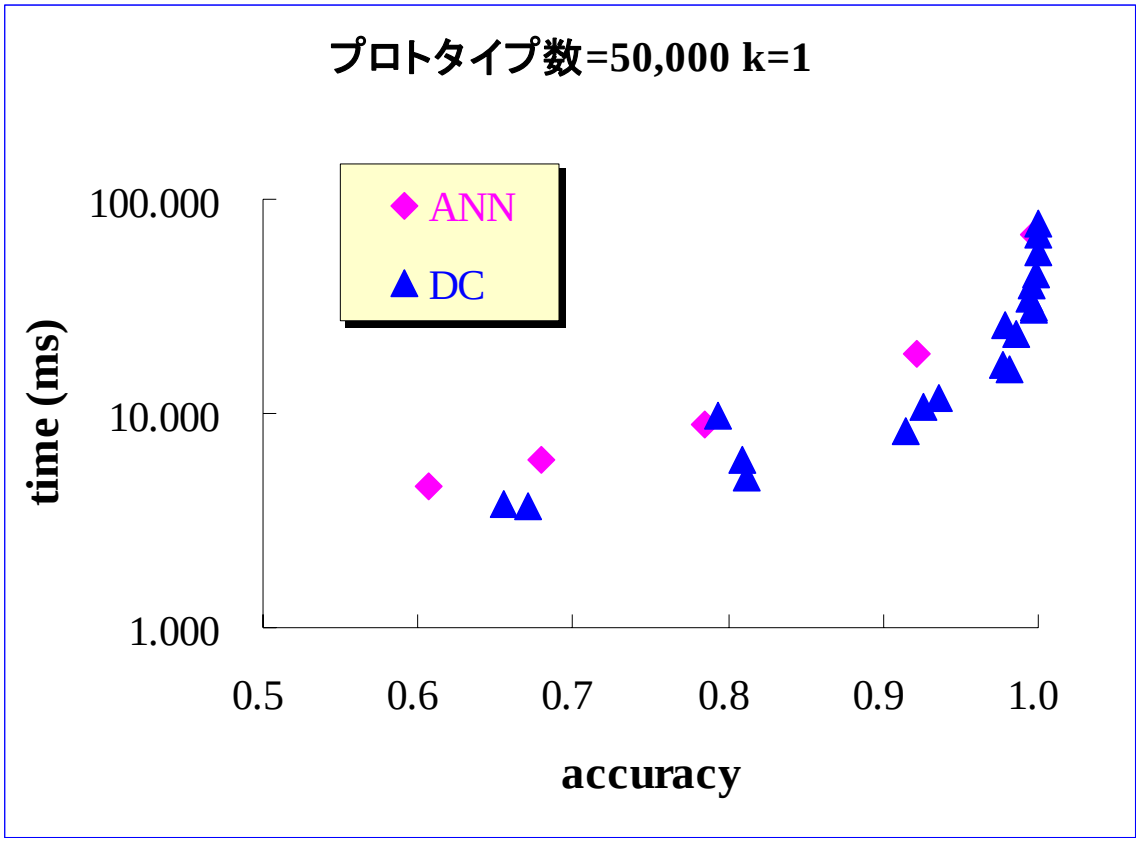
- 次元数=784次元（大きさ1に正規化）
- プロトタイプ数=1万、(3万)、5万
- クエリに対して返却する最近傍の数＝１個、２０個
- クエリ数=1万

* THE MNIST DATABASE of handwritten digits (Yann LeCun, and Corinna Cortes)

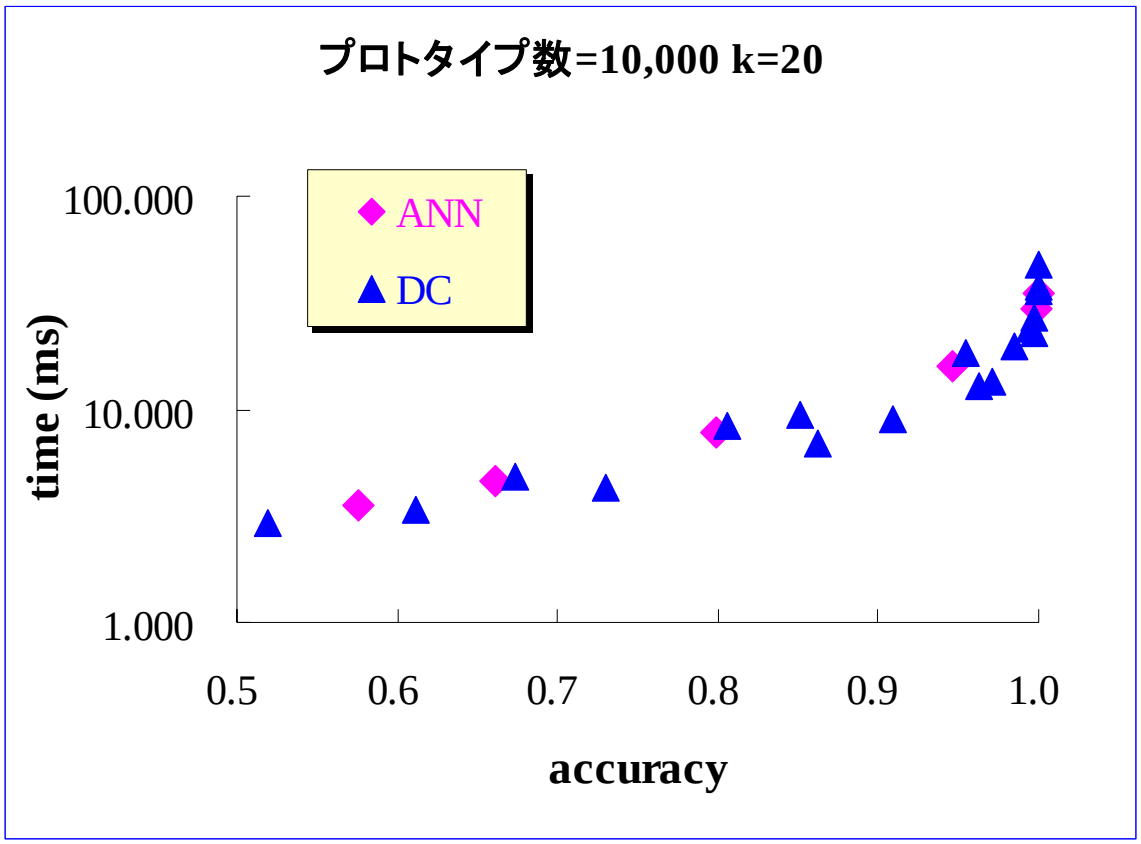
MNISTによる実験（次元数=784）



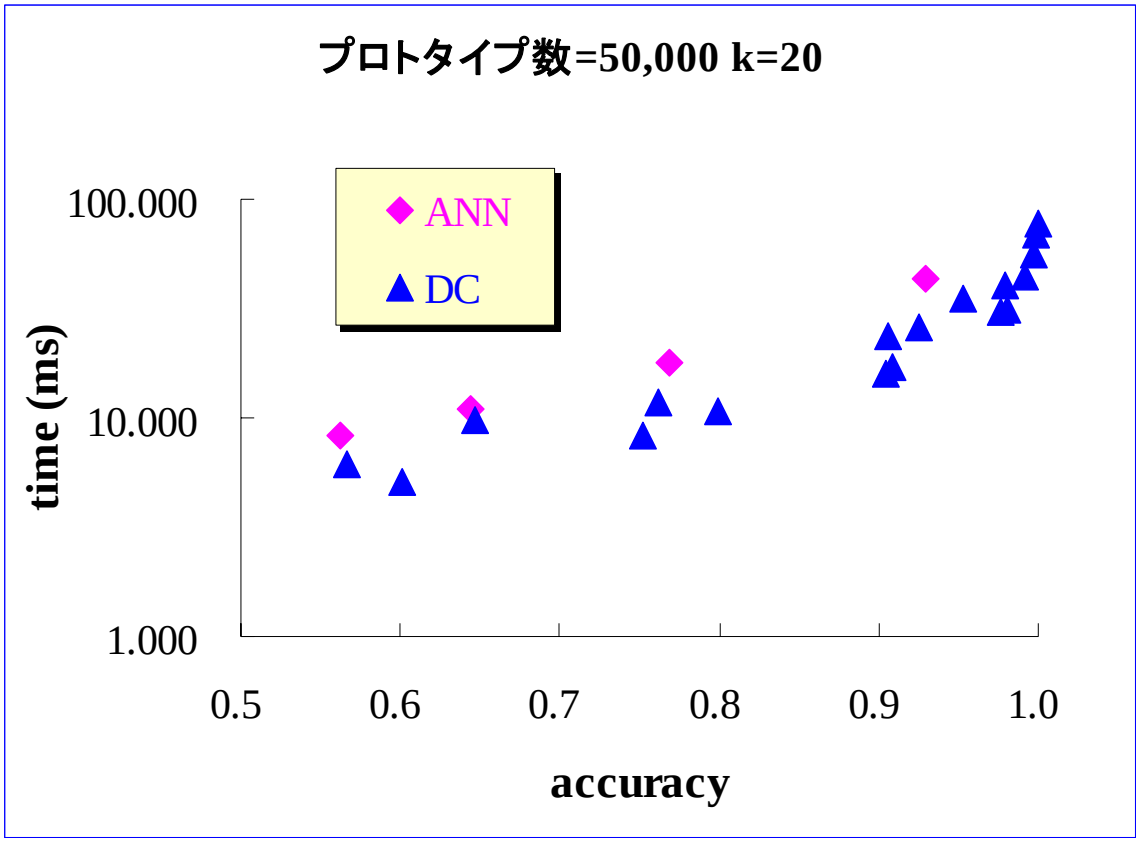
MNISTによる実験（次元数=784）



MNISTによる実験（次元数=784）



MNISTによる実験（次元数=784）



最近傍探索のまとめ：

高次元、大量のプロトタイプ数ほど、また、探索候補数が多くなるほど、ANNより優位性が増す。

3. 高速なパターン認識

未知パターンを x^*, クラスC_kに属する学習パターンを $x^{i,k}$ とし、分散表現をそれぞれ

$$\{e_j^*\}, \{e_j^{i,k}\} \qquad 1 \le i \le m_k, 1 \le k \le n_C$$

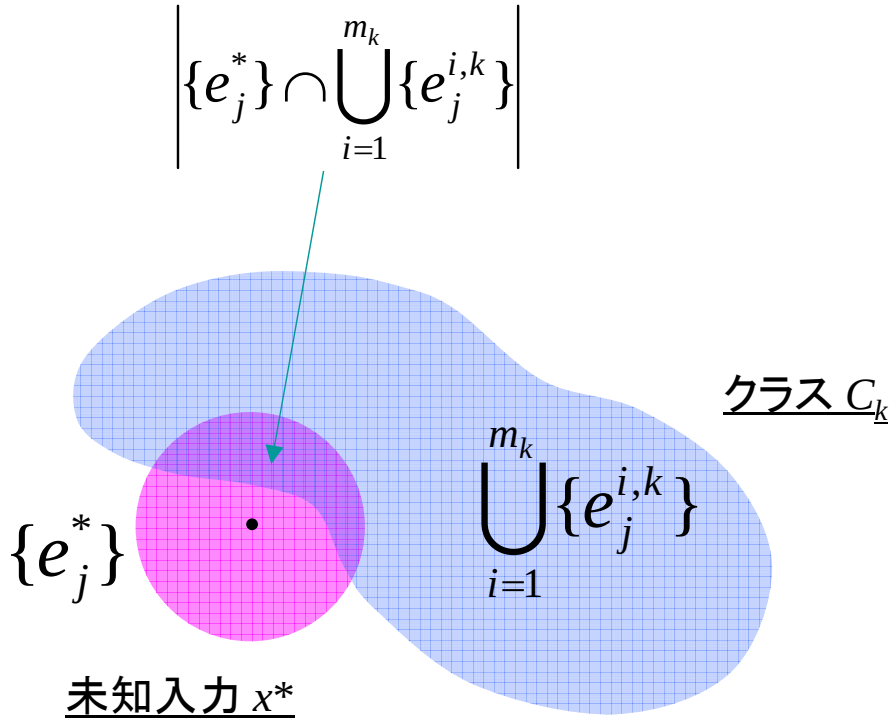
とする。

識別のための評価関数（類似度）を次のように定義する:

$$s_k(x^*) = \left| \{e_j^*\} \cap \bigcup_{i=1}^{m_k} \{e_j^{i,k}\} \right|$$

識別結果:

$$C_{kmax} \qquad kmax = \underset{1 \le k \le n_C}{argmax} \{s_k(x^*)\}$$



分散表現の一例:

$Facto^{\pm}(N) = \{\, x \mid x = (k'_0, k'_1, \ldots, k'_{N-1}),$

$k'_0\, k'_1 \ldots k'_{N-1}$は $-N+1\ -N+3 \ldots N-1$の順列$\}$

$$Combi^{\pm}(N, j) = \{e \mid e = (e_i), e_i = -1, +1, \sum_{e_i=1} e_i = j\}$$

と定義するとき、

$x \in Facto^{\pm}(N)$, $Q_h = Combi^{\pm}(N, h)$, $(h=1 \sim N-1)$

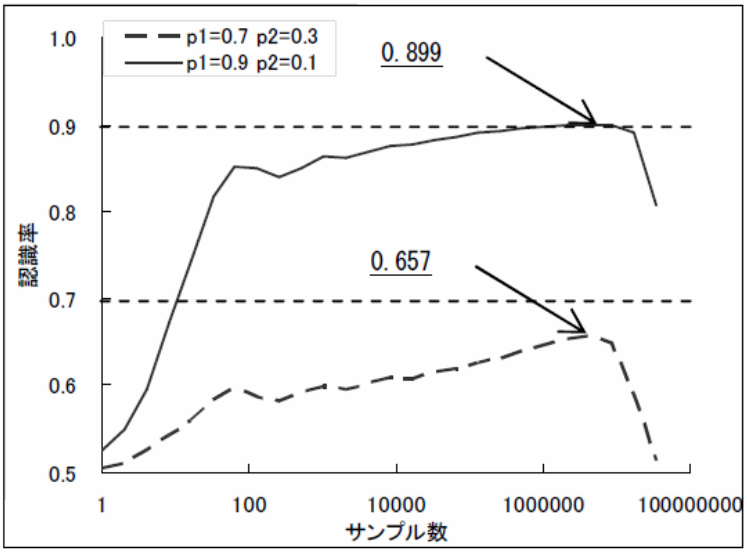
として分散表現を行う。

数値例:

$$(-5, -3, -1, +1, +3, +5) =$$
$$(-1, -1, -1, -1, -1, +1) +$$
$$(-1, -1, -1, -1, +1, +1) +$$
$$(-1, -1, -1, +1, +1, +1) +$$
$$(-1, -1, +1, +1, +1, +1) +$$
$$(-1, +1, +1, +1, +1, +1)$$

評価関数s_kの識別能力はどの程度あるのか？

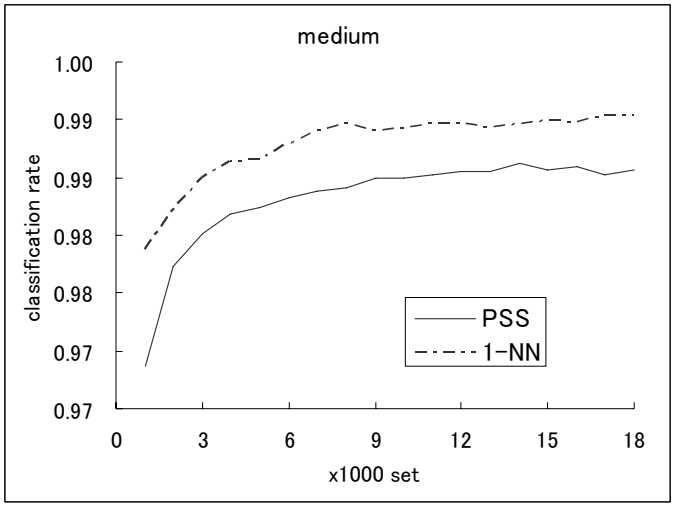
数値シミュレーション：クラス数=2、密度の比= p1:p2 として、サンプル密度と認識率の関係を調べる。



サンプル数を増やしていけば（臨界点に達するまでは）認識率は上昇していく。

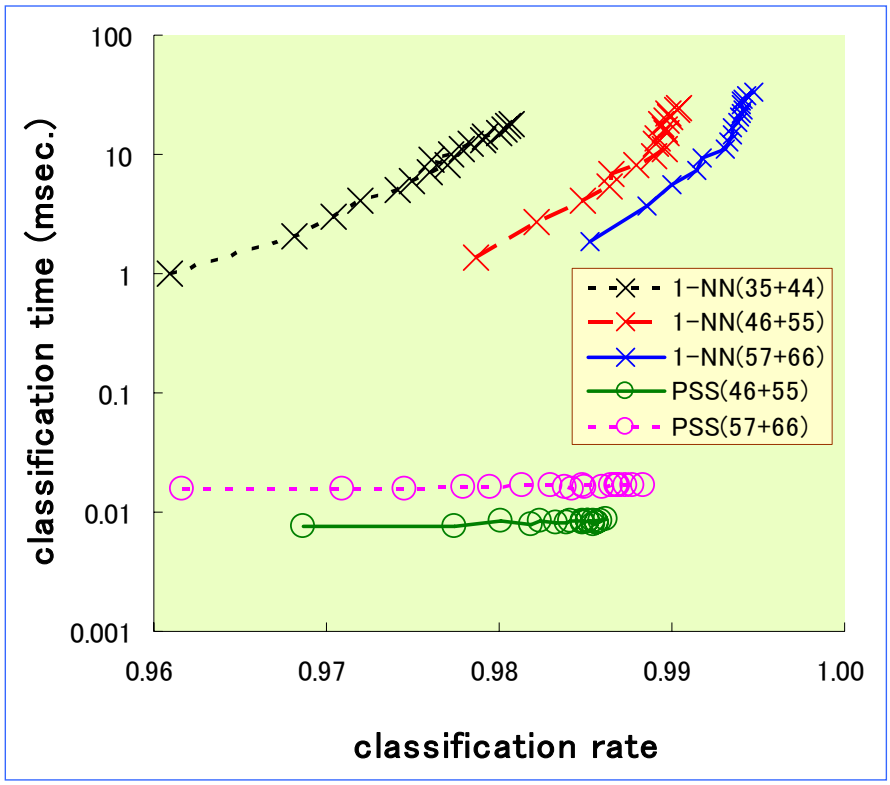
実データによる実験

BIRDS database を用いて1-NN法と比較（認識率）



学習サンプル数を増やせば1-NN法の認識精度と同等にすることができる

認識率と認識時間の関係



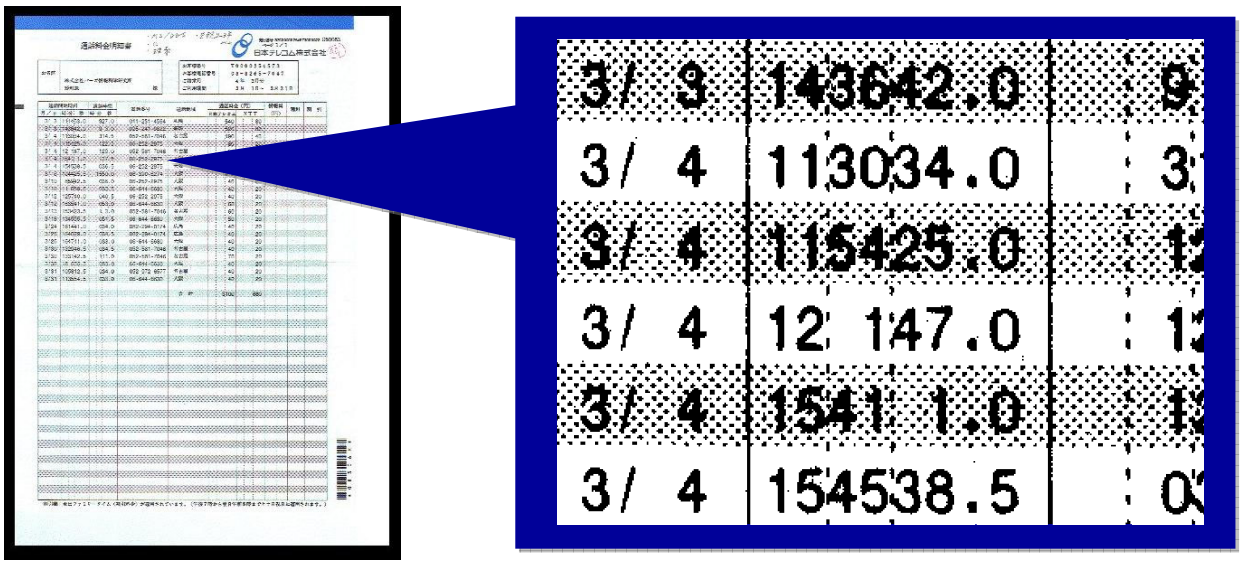
パターン認識のまとめ:

大量のサンプルの学習処理が高速であり、且つ、認識処理は非常に高速である。

大量のサンプルを学習させることによって、認識率を高くすることが出来る。

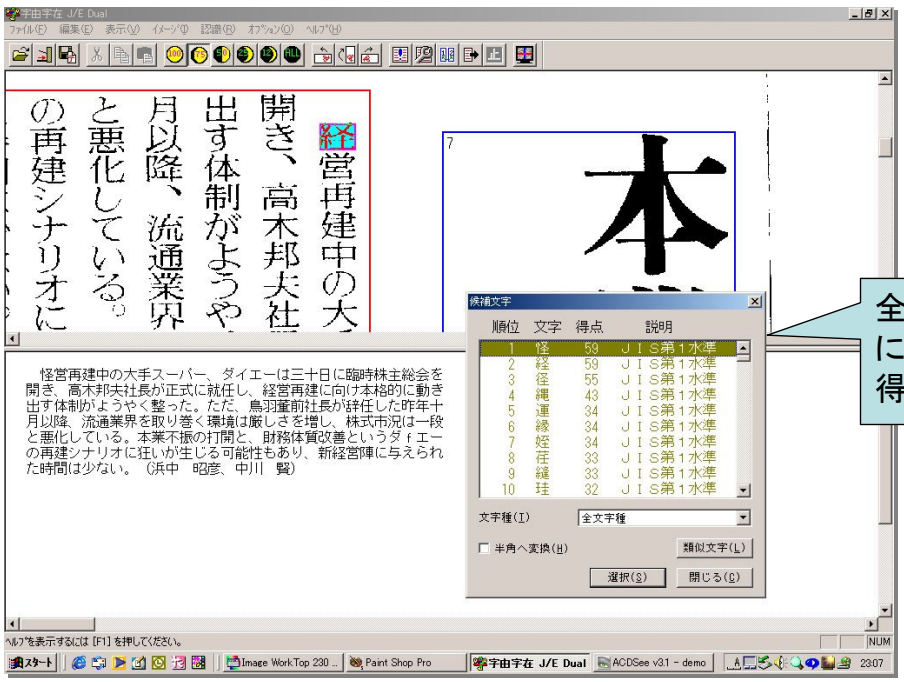
実用例：低品質原稿の文字認識

ビジネスニーズに応えるためには開発にスピードが要求される。
「学習が高速である」ことも重要なファクターであった。



実用例：日本語OCR

日本語文字種約3300カテゴリーの高速な認識の実例



全てのカテゴリーに対して類似度が得られている

本発表のまとめ

大規模データに対して有効な探索手法、学習・識別手法を提示した。

今後

・さまざまな改良手法の研究

・現実問題での有効性の実証、理論的な考察の両面

リソース

DC-ANN サンプルソースコード (PRMU2006/06版):

http://www.geocities.jp/onex_lab

お問い合わせ : 1xlab@rforce.co.jp

ANN:

http://www.cs.umd.edu/~mount/ANN/

LSH:

http://www.mit.edu/~andoni/

MNIST:

http://yann.lecun.com/exdb/mnist/

BIRDS文字パターンデータベース:

http://www.geocities.jp/onex_lab/birdsdb/birdsdb.html