解説

空間分割を用いた識別と非線形写像の学習

(1) 空間分割による最近傍識別の高速化

和田 俊和
和歌山大学システム工学部　twada@ieee.org

■はじめに

今回を含めて2回に分け，空間分割を用いた最近傍識別の高速化と，非線形写像の学習方法について解説記事を書くことになった．正確さよりも分かりやすさを追求し，事例をたくさん盛り込むように努力するつもりなので，お付き合いいただきたい．今回は，分かりやすさを最優先にするため，解説と研究上のエピソードを交えた形式で議論を進めることにする．

まず，コンピュータビジョンや画像データを対象としたパターン認識の問題に関して研究を行ってきた者がなぜ，1967年頃に登場した古い識別器に心を奪われたかについて，説明させていただきたい．

自分自身の研究を振り返ると，被写体である物体やそれが写された画像領域などに関して，やたらと「モデル」を使ってきたことに気づく．効率よく研究を進めるためには，正しい選択ではあるが，常に違和感が残っていた．我々のように認識の問題を扱う者の中には，人間のように知的な機械を作ることを究極の目的にしている人が多いが，私も多分に漏れず，その1人である．そういう者がいったん「人間はこんなことはやってないだろう」と思いはじめると，その研究はパズルとしての意味しか持たなくなってしまう．

この空虚さを埋めるために，役立つ技術を作ることばかりを追い求めるようになるのが健全なのであるが，私もそうなりかけていた．その頃Support Vector Machine (SVM)[4]を知り，適当な確率分布モデルでパターン全体を記述するよりも，パターン間の誤識別を起こしやすい境界付近を記述することの方が識別にとって有効であるというVapnikの考えに強く惹かれた．カーネルという手品でぐにゃぐにゃ曲がった識別面を作るという特長よりも，「端っこ」を重視する考えの方が気に入ったわけである．

そうして，なんとかSVMを越えるものは考えつかないかと考えていたころ，よくよく考えてみると，最も古いパターン認識手法の1つである最近傍識別器[1]も，この「端っこ」を重視する，というか，端っこだけで識別面が決まる識別器であることに気がついた．歴史は巡るというが，SVMやADA Boost[5]などの近代的な識別法の根底にある哲学と，古典的な識別器の根底にあるものとがほとんど同じであることに気がついたときは，大変感心した．

また，よく考えてみると，最近傍識別器では識別対象に関するモデル化をほとんど行っていないことに気づく．多くの識別器が学習データである事例を基にして，識別器のコントロールパラメータを求めたり，構造を決定する．ところが，最近傍識別器ではそういったことをまったく行わない．つまり，モデルやルールを帰納することなく，事例自体で事例を説明するのである．このような方法は「事例ベース」と形容されるが，結局のところ，純粋にこう呼べるものとしては，最近傍識別器ぐらいしか思い当たらない．

これらが，最近傍識別器を気に入った理由である．研究者である以上，普通なら，最近傍識別器に代わる新しいものを作ろうとするのであるが，この識別器ほどシンプルで直接的な方法はないので，とにかく最近傍識別器のすばらしさを知ってもらおうというスタンスで，研究をすすめてきた．まず，本題に入る前に最近傍識別器に関する一通りの話を書いておくことにする．

■最近傍識別器の基礎

最近傍識別器というのは，入力データにその帰属クラスのラベルを付けた「プロトタイプ」と呼ばれるデータをたくさん記憶しておき，入力が入ってくると，一番近

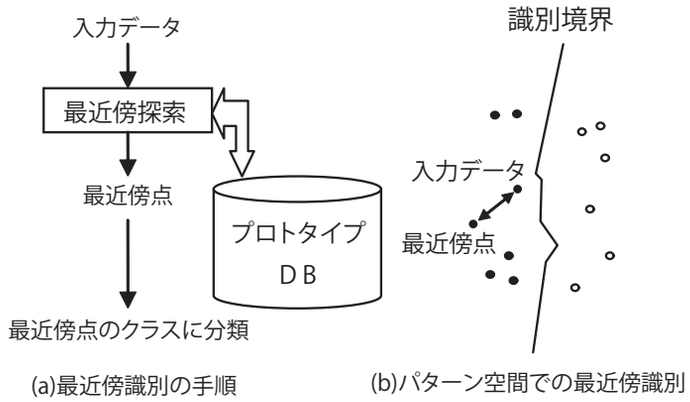


図-1 最近傍識別

いプロトタイプを探し出す「最近傍探索」を行い，見つかったプロトタイプに付いているラベルに従って識別を行うという単純極まりないものである（**図-1**）☆1．この変種として，k 番目までの最近傍プロトタイプを集めてきて，それらに付いているラベルで多いものを採るという，k-最近傍識別器[1]もある．

それにもかかわらず，無限にプロトタイプが与えられた場合には，いかなる識別器でも不可避的な誤り確率である Bayes 誤り確率の 2 倍未満が達成できるという優れた性質を持っている．プロトタイプ数に対する誤り確率の低さという点では，k-最近傍識別器の方が優れているが，以降では 1-最近傍識別器について話を進める．

Condensing と Editing

ここまでの内容は，パターン認識の教育を受けた人は大体知っている話だと思うが，最近傍識別器の本質を知るにはさらに深い内容を知る必要がある．

最近傍識別器の動作は上述のとおり簡単である．では，最近傍識別器が形成する識別面の形状はどうかというと，計算幾何学の分野で主に研究されている Voronoi 分割を使えばこれも簡単に説明することができる．**図-2** に示すように，全プロトタイプを母点とする Voronoi 分割を行い，異なるクラスラベルが付けられたプロトタイプを母点とする Voronoi 領域が互いに接する面が識別境界となる[3]．

このことから，ある Voronoi 領域の周りを同じクラスラベルがつけられたプロトタイプを母点とする Voronoi 領域が取り囲んでいる場合，その Voronoi 領域を消しても識別面は変化しないということが分かる．したがって，大量のプロトタイプを記憶しなければならないという最近傍識別器の欠点の 1 つが解決できそうである．

☆1 通常の最近傍識別器の学習はトレーニングデータを記憶するだけの lazy learning であるが，後述する KDDT では識別の高速化のための eager learning が行われる．

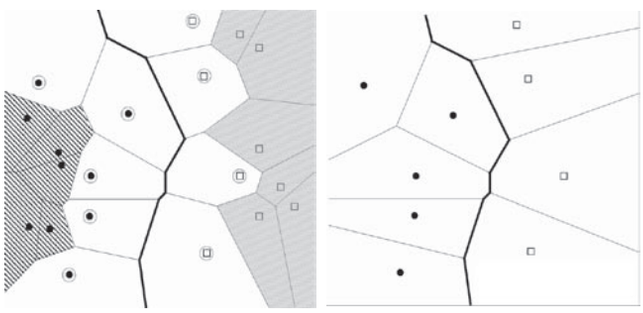


図-2 Voronoi Condensing

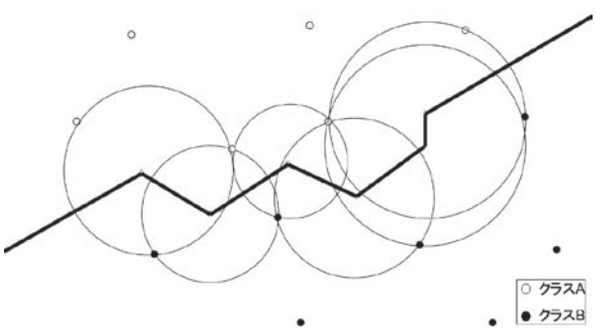


図-3 Direct Condensing

ここまで書くと，うまく行きそうであるが，実は話はそう簡単ではない．Voronoi 分割を行うためには，かなりの計算が必要なのである．具体的には，空間の次元数を d，プロトタイプ数を N とした場合，$O(N^{\lfloor d/2+1 \rfloor})$ の計算が必要になる[3]．実際に，任意次元数の Voronoi 分割を行うソフトウェア QuickHull を用いて試したところ，1,000 個，5 次元程度のプロトタイプまでしか計算はできない．

このままでは，最近傍識別器の良さが伝わらないので，なんとかしようと編み出したのが Direct Condensing[6]という手法である．これは，空間全体の Voronoi 分割を行わず，Voronoi Condensing と同じ結果を得る手法である．詳細は割愛するが，要は**図-3** に示すように，クラ

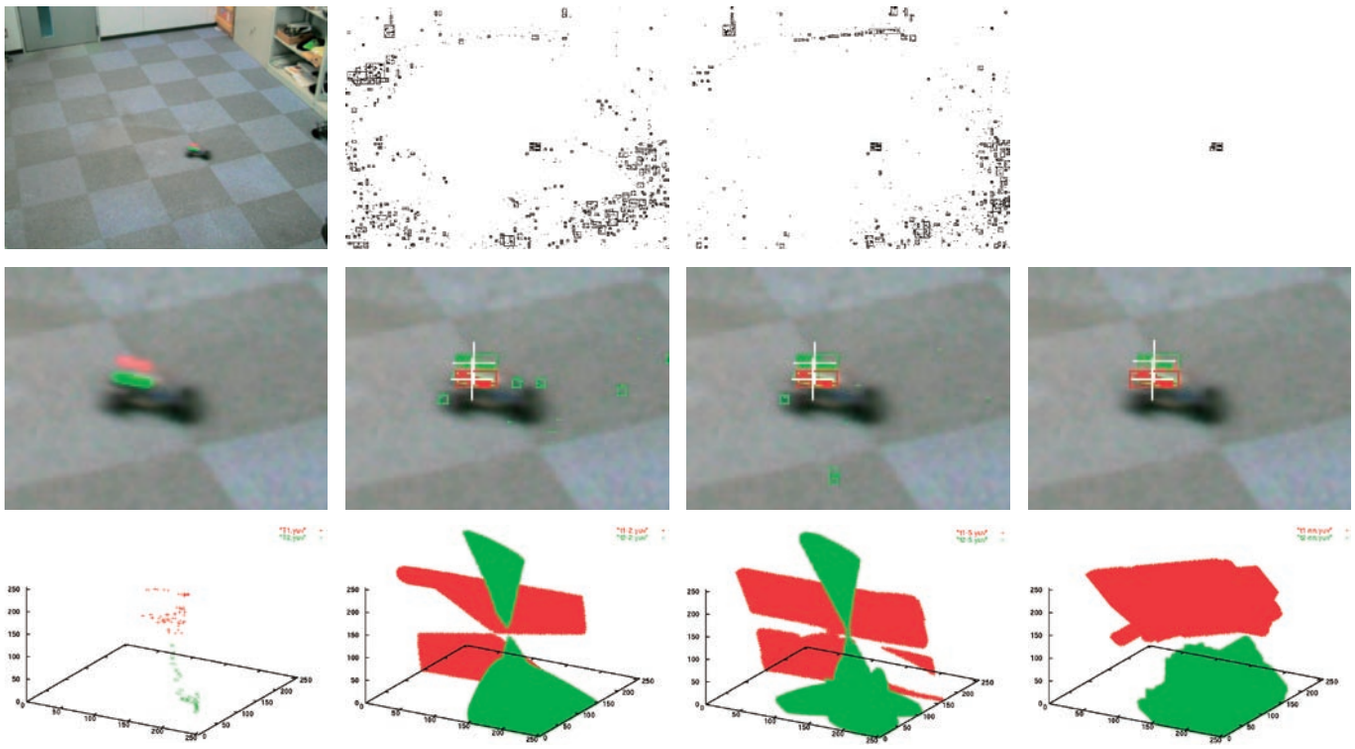


図 -4　2 色の LED 検出結果（上段左から：原画像，EM-2, EM-5, NN）（中段：上段の拡大表示）（下段左から：ターゲット色の教示色，EM-2, EM-5, NN で検出に使われた LUT）

スラベルの異なるプロトタイプを表面上に持つ超球を識別境界付近で求め，あとはグラフ探索と同じ要領で同様の超球を順に辿っていくのである．この結果求められる超球上のプロトタイプは，Voronoi Condensing によって求められるプロトタイプと一致する．この計算によって，Voronoi Condensing を行うことができる空間の次元数は増加するが，現在のところ，実用的なデータに対しては，25〜30 次元程度までにとどまっている．

これ以外にもプロトタイプを削減する手法[2]はたくさんある．特に，Gabriel Editing[3]☆2 と呼ばれる手法は，近接性の尺度が異なるだけで Voronoi Condensing とほとんど同じ方法である．この手法では，Gabriel グラフというものを用いる．こちらの計算量は，かなりまして，$O(dN^3)$ である．これを用いた場合の実験結果は，Voronoi Condensing よりもかなりプロトタイプ数が減るが，識別性能はほとんど変わらないという優れた性質を持っている．Gabriel Editing を行った最近傍識別器と SVM とが非常によく似ているといった研究報告もある[7]．考えてみれば，Voronoi Condensing も Gabriel Editing も，そして SVM も分布の端にあるデータを求めようとしているのだから，この発想も道理を得たものと言えよう．

■最初の試み

さて，下準備が一通りできたので，また研究をどういう風に進めていったかについて話を戻したい．SVM に凝っていた当時，ラジコンに取り付けた赤と緑の LED の部分を色で検出するプログラムを作っていた．

まずは，画像から LED の色をサンプルし，同じ色が出てきたら，それを検出するというプログラムを作成してみた．すると，色をサンプルした画像以外で LED はまったく検出されなかった．画素値はかなり大きく変動するので，このような結果になることは当然であったが，こんなにもうまくいかないとは思わなかった．検出する色の範囲を膨張させたりすると，床が検出されてしまったりして，この方針ではだめだということが分かった．

そこで，こんどは当時流行の SVM を使って学習をさせてみたところ，まあまあの結果が得られたので，誤検出や未検出の部分をトレーニングデータに追加していったところ，非常に長い学習時間がかかるようになり，これも対話的には使えないことが分かった．考えてみると，色の種類は非常に多いので学習サンプル数が非常に多くなるという最悪の条件で SVM を使っていたことになる．

☆2　Condensing と Editing の用語の使い分けは，この分野の草分けである Dasarathy が言い出したことで，それ以前は皆が Editing と呼んでいた．彼によると，結果を変えないものは Condensing であり，データを削除するなどして，汎化性能を向上させるというのが Editing である．

そして辿り着いたのが，最近傍識別器である．これなら，基本的に学習は要らないし，誤検出の部分の修正も対話的に行える．しかし，唯一の問題は最近傍色の探索時間である．1 枚の画像に含まれる画素の数は 640×480=307200 である．これだけの回数の最近傍探索を NTSC 規格の映像信号のフレームレート（1/30 秒）で終えるのはどんなコンピュータを使っても不可能であるので，色の 3 次元配列を用意してそれを最近傍識別した結果，つまりターゲットか非ターゲットかを表す値を振った Look Up Table（LUT）を用いることにした．

図 -4 にこの実験結果を示す．EM-2 は EM アルゴリズムを用いて，赤と緑および背景の色分布をそれぞれ 2 つ，合計 6 つの正規分布で当てはめた場合，EM-5 はそれぞれ 5 つ，合計 15 個の正規分布で当てはめた場合，そして NN が最近傍識別器の結果である．この結果からも，最近傍識別器が最も優れており，学習（LUT の更新）時間も約 0.5 秒と非常に短いことが確認された．このことから，事例ベースのコンピュータビジョンができそうな気がしてきた[8]．

■普通の最近傍識別器の高速化

先の色ターゲット検出を少し拡張すると，たとえば背景の色と入力画像の 2 つの色を合わせて 6 次元ベクトルを構成して，これでターゲット検出を行うことができると思いついた．しかし，さすがに 6 次元配列では計算機のキャッシュに納まらないどころか，下手をすれば仮想記憶が動きそうで，とても実時間処理は行えそうにない．したがって，LUT は使用できない．

この時点で，本来の最近傍識別器の高速化法についてまじめに研究しなければいけないという気持ちになり，調査を開始した．ところが，この分野の研究では，最近傍探索の高速化法は山ほどあるものの，最近傍識別器の高速化方法というものはほとんど出てこないことに気がついた．

とりあえず，最近傍探索を行うツールでは Approximate Nearest Neighbor（ANN）と呼ばれるツール[9]が評判が良いようなので，これを使ってみることにした☆3．他のツールも手に入る限り試してみたが，100 次元程度までのデータに対する処理では ANN が圧倒的に速いことが分かった．しかし，1 枚の画像内にある画素をビデオレートで識別するには遠く及ばないことも事実であった．

この問題を解決するためのヒントは，最近傍探索と最近傍識別の違いである．つまり，識別のためには常に最近傍プロトタイプを見つける必要はないのではないか？ということである．この話を進める前に，ANN などの空間分割を行う最近傍探索の大まかな計算の流れを説明しておくことにする．

■空間分割による最近傍探索

ANN では，入力空間を張る軸のうちの 1 つに直交する超平面で空間を 2 つの半空間に分割し，最終的に中に含まれるプロトタイプが 1 つになるまでこの操作を再帰的に繰り返す．空間分割を行う際に参照する軸の選択や，どの位置で分割を行うかなどに関してはさまざまな方法があるが，ここでは以下に述べる Sliding-midpoint split rule を紹介する．

まず，入力データ全体を囲む超直方体を求める．そしてその最長辺に直交する超平面でこの超直方体を分割する．分割位置は最長辺の中点である．この結果，2 つの超直方体が得られる．各超直方体に含まれるプロトタイプが 1 つになるまで同じ操作を繰り返すが，分割した結果一方の超直方体の内部のプロトタイプが 0 個になった場合には，内部に少なくとも 1 つのプロトタイプが含まれる位置まで分割面をスライドさせる．

この方法は，いい加減だとも思えるが，超直方体内部のプロトタイプの分布などをいちいち調べて「最適な」分割位置を求めようとする他の手法と比べて高速であり，しかも各辺の長さに極端な偏りがない空間分割が得られるという特長がある．

この空間分割の結果は，二分木に格納され，探索時にはこれが二分探索木として用いられる．具体的には，各ノードには，分割に用いた軸，分割面の位置（閾値），そして閾値以上と以下の 2 つの子ノードに対するポインタが格納されており，葉ノードにはプロトタイプが格納される．二分探索時には分割に用いた軸と，閾値を参照して，ノードを辿っていくが，各ノードでは入力ベクトルデータの 1 要素だけを調べているため，高速な探索が行える．空間分割を行うアルゴリズムの大半が超直方体分割を採用している主な理由はここにある．

探索時には，入力データに対する二分探索を行い，辿り着いた直方体の内部にあるプロトタイプを探すが，これだけでは済まない．見つけられたプロトタイプが入力データに対して最も近いとは即断できないからである．この問題を解決するための探索を ANN では Priority

☆3 ANN はその名の通り，誤差を許容することで，最近傍探索の速度を向上させるアルゴリズムであるが，今回は許容誤差を 0 にセットして，通常の最近傍探索器としてこれを用いた．

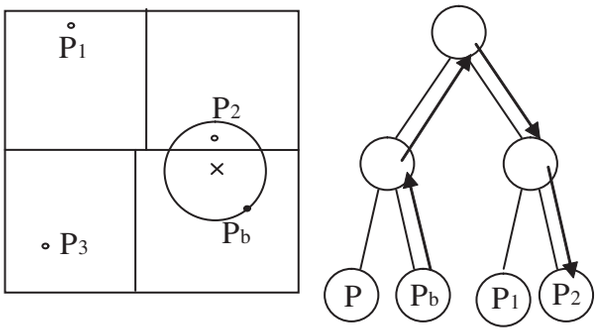


図 -5　Priority Search

Search と呼んでいる．

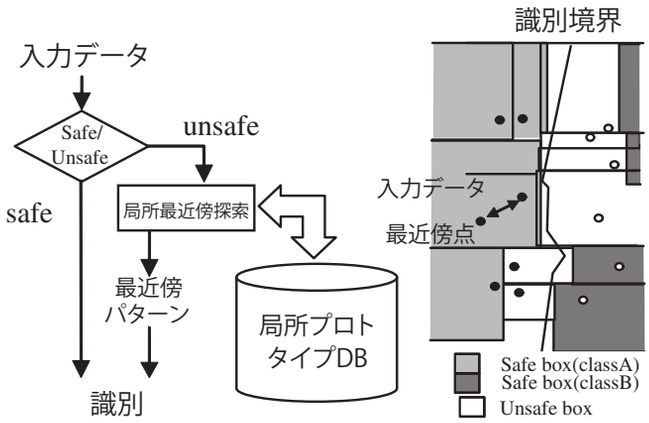
図 -5 に示すように，見つかったプロトタイプを中心とし，入力データを表面に含む超球を考える．この超球が超直方体の壁と交差しなければ，見つかったプロトタイプは入力データの最近傍プロトタイプであると言い切れる．そうでない場合，つまりこの超球が近隣の直方体と交差する場合には，それらの直方体に含まれる全プロトタイプとの距離を調べていかなければならない．これは，前述の二分探索木ではある種のバックトラックを行うことに相当する．

この Priority Search が探索時に最も時間のかかる処理であり，これを省くことができればかなりの高速化が行える．しかし，空間分割を行う最近傍探索アルゴリズムのほとんどが，辿り着いたノードから後戻りをしながら同様のサーチを行っており，これは不可避的なことなのである．最も計算回数が大きくなるのは，超球上の点として与えられたプロトタイプ集合に対して，入力データがその中心に与えられた場合である．この場合には，全探索が行われるため極端にスピードが低下する．

▌最近傍探索から最近傍識別へ

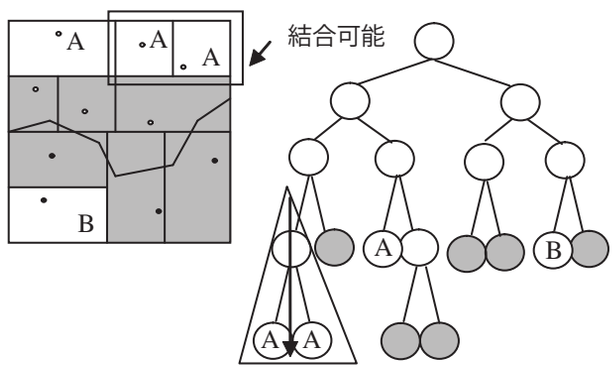
上記のアルゴリズムを，高速な最近傍識別器を作りたいと思って読むと，ある無駄なことをしていることに気がつく．つまり，最近傍識別器では各プロトタイプにクラスラベルが付いている．だから，最近傍プロトタイプが求まらなくても，正しいクラスラベルが求まりさえすれば問題ないのである．

このためには，ある超直方体が識別境界と交差しない場合（このケースを Safe と呼ぶ）にはその内部に「ここに入ったらクラスは○○です」などというクラスラベルを付けておけばいい．それ以外の場合，つまり識別境界と交差する超直方体に入力データが入った Unsafe の場合には，Priority Search を行わない限り識別は行えない．しかし，よく考えてみると識別面は異なるクラスラベルを持つプロトタイプの等距離面が合わさったもので



(a)最近傍識別の手順　　(b)パターン空間での最近傍識別

図 -6　最近傍識別の高速化



(a) box 表現　　(b) tree 表現

図 -7　Safe ノードのマージ

あるので，直方体内部に現れる識別面を形成している少数のプロトタイプをあらかじめ見つけておけば，それらとの距離比較で直接的に識別が行える．これを Local Nearest Neighbor Search（LNNS）と呼ぶ．

図 -6 に示すように，Safe/Unsafe の識別を行い，Unsafe の直方体に関しては LNNS を行うというのがこの手法の基本戦略であるが，実際に木構造を作ってみると，子の葉ノードがすべて Safe になっている部分がたくさんあることに気がつく．そういった場合には，**図 -7** に示すように，これらの子ノードを削除して親を新たな葉ノードにすればよい．Safe ノードに含まれるプロトタイプは，すべて削除することができるため，前述の Voronoi Condensing の結果と同数のプロトタイプを記憶するだけでよい．

この手法を我々は k-d Decision Tree（KDDT）[☆4] と名

☆4　機械学習やデータマイニングの分野では KDD（Knowledge Discovery and Data-mining）という言葉がよく用いられているが，当時はこのことを知らず，こんな大事なキーワードを使ってしまったことを少々反省している．

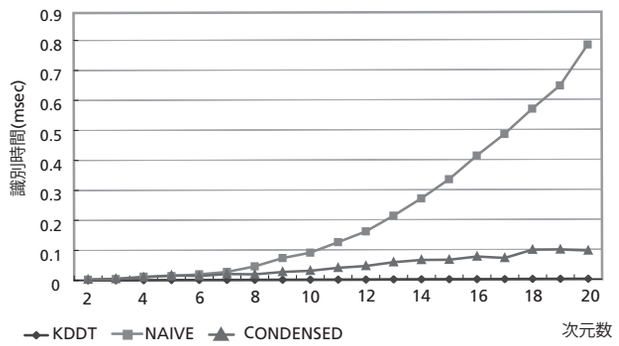


図 -8 次元に対する最近傍識別の速度変化

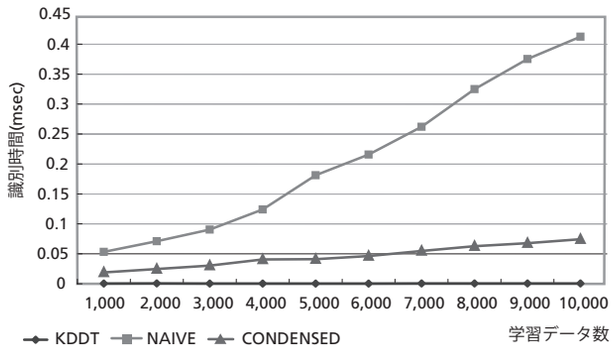


図 -9 データ数に対する最近傍識別の速度変化

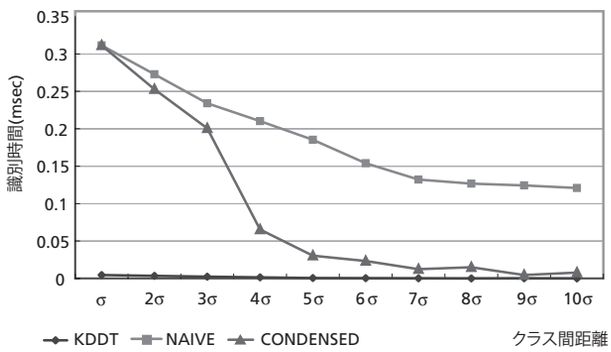


図 -10 クラス間距離に対する識別時間の変化

づけた[10]．KDDT を実装してテストしてみたところ，ANN による最近傍探索に基づく手法の数倍から数百倍の実行速度が得られた．

図 -8，**図 -9** には，ANN を用いて最近傍探索を行って識別を行う NAIVE と，Direct Condensing を用いてプロトタイプを減らした結果に対して同じ方法で識別を行った CONDENSED, そして KDDT を用いた場合の識別時間が示されている．図 -8 はプロトタイプ数を 3,000 として空間の次元数を変化させた場合，図 -9 は次元数を 10 としてプロトタイプ数を変化させた場合，それぞれの平均識別時間である．これは 100,000 個のテストデータを用いて計測した平均識別時間である．次元数を変化させた場合は単調に速度比は大きくなり，たとえば 20 次元のとき NAIVE に対して 583 倍，CONDENSED に対して 71 倍もの速度が出ている．また，プロトタイプ数の変化に対して NAIVE，CONDENSED ともに識別時間が増加しているのに対して，KDDT ではプロトタイプ 1,000 個に対して 0.00043ms，10,000 個で 0.000656ms と，速度低下がほとんど起きていない．

アルゴリズムの改良で数百倍という速度向上が実現できるという話は，近年ではあまり聞かなくなったが，これはそういったことができる稀なケースであろう．

最近傍識別器において過学習の問題をどう解決するのかという点が気になる読者もおられることと思う．先に述べた「無限にプロトタイプが与えられる場合に，Bayes 誤り確率の 2 倍未満が達成できる」という最近傍識別器の特長は，この問題がさほど深刻ではないことを意味している．これは，プロトタイプが十分あれば識別面の乱れは局所的にしか起きないため，この性質が成り立つのである．プロトタイプ数が十分でない場合は，*k*-最近傍識別器を用いるか，Editing によって異なるクラスの分布の重なり付近にあるプロトタイプを除去する，という方法で識別器の誤り確率を下げようとする試みもこれまで多数行われてきた．

異なるクラスの分布が重なるということが起きた場合，KDDT の識別速度はどのように変化するのかを調べるために，偏差 σ の正規分布に従うクラスを 2 種類考え，分布間距離を 1σ から 10σ まで変化させた場合の識別時間を求めた．これを，**図 -10** に示す．

2 つの分布が入り混じると Safe の超直方体が減少し，したがって Safe ノードの統合も行えなくなる．しかし，LNNS は依然として行えるため，1σ においても KDDT は NAIVE，CONDENSED に対して約 69 倍の速度を維持している．

ただし，この方法にも問題点があり，超直方体の Safe/Unsafe を判定したり，LNSS で用いる局所的なプ

ロトタイプを見つけるために Voronoi Condensing を行う必要があるのである．これは，Direct Condensing アルゴリズムを用いても 30 次元ほどのデータしか扱うことができないということを意味しており，これが唯一の弱点である．この問題を解決すべく，しばらくの間検討したが，現在もなお解決策は見つかっていない．

■空間分割を行わない最近傍探索へ

画像のようにそのままでは非常に高次元のベクトルとしてしか扱えないようなデータに対する最近傍探索問題では，空間分割そのものが使えなくなってしまう．

この問題を解決する策としては，距離計算のみで最近傍探索を行う以下の方法が考えられる．

- AESA[11] およびそれを拡張した手法
- VP-tree[12] およびそれを高速化した手法
- k-NN グラフ [13] などのグラフを用いた手法

これらの手法は，常に改善されており，また目的も異なるため，どれが最適であるかは一概には判定しがたい．

VP-tree は入力に対してある距離の範囲内にあるプロトタイプを集めてくるレンジサーチ用のアルゴリズムであり，1- 最近傍の探索にはあまり適さない．AESA は 3 角不等式を利用した最近傍候補の絞り込みを行う手法であり，距離計算回数は少ないが，プロトタイプ間の距離テーブルの参照回数が多いため，実際には必ずしも速くない．これらは，最近傍プロトタイプが正確に計算できるが，グラフを用いた場合には Delaunay グラフを用いない限り正確には最近傍プロトタイプは計算できない．このため，高次元空間では k-NN グラフなどがその代用として用いられ，この場合に高速な探索が行える．このように，どのような最近傍探索を行うか，正確さと速度のいずれを重視するか，などの基準によって用いられるべき手法が異なってくるという事情がある．

いずれにしても，高次元空間において最近傍プロトタイプを高速に探索する技術の開発競争は今後もしばらく続きそうである．

■まとめ

最近傍識別の中で，空間分割を行う KDDT はそれが実行できる次元の範囲内ではおそらく最も高速な最近傍識別器であると言える．しかし，この手法は「次元の呪い」というやっかいなものにからまれた Voronoi Condensing という手法に依存していることが判明した．このため，比較的低次元の空間でしか適用できない．この次元の呪いを解くことはかなり困難であるが，実用的な回避策もある．Voronoi Condensing の代わりに，Gabriel グラフを用いた Gabriel Editing を使っても識別結果に大きな差は出ないので，この代用策で数百次元の空間を扱うことができるようになる．

最近，「人間も最近傍探索をやっているのではないか」と思うようになってきた．たとえば，人間の脳がニューラルネットであっても，その中で起きるアトラクタへの引き込みは，何がしかの距離尺度に基づく最近傍探索であると言える．そもそも，我々は具体的な事例から常にモデルを構築しているのであろうか？　たとえば，人の顔を認識する際，これまで見たことがある顔の中のどれかに近いから，顔だと判定しているに過ぎないという考え方も成立し得るのである．

次回は，この稿で述べた空間分割を非線形写像の学習に適用する方法について，中村恭之先生と一緒に解説することにする．

謝辞　Direct Condensing と KDDT は，和歌山大学加藤丈和助手と同大修士 2 回生の柴田智行君との研究成果である．実験補助にとどまらず，アルゴリズムの構想段階から彼らの助けが必要であった．

参考文献

1）Cover, T. M. and Hart, P. E.: Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory, Vol.IT-13, No.1, pp.21-27 (1967).
2）Hart, P. E.: The Condensed Nearest-neighbor Rule, IEEE Transactions on Information Theory, Vol.IT-4, No.5, pp.515-516 (1968).
3）Bhattacharya, B. K., Poulsen, R. S. and Toussaint, G. T.: Application of Proximity Graphs to Editing Nearest Neighbor Decision Rule, International Symposium on Information Theory, Santa Monica (1981).
4）Vapnik,V.: The Nature of Statistical Learning Theory, Springer-Verlag, New York (1995).
5）Freund, Y. and Schapire, R. E.: Experiments with a New Boosting Algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, pp.148-156 (1996).
6）和田，加藤：近接性グラフに基づく効率的 Condensing の理論，信学技報 PRMU, Vol.103, No.96, pp.13-18 (May 2003).
7）Zhang, W. and King, I.: A Study of the Relationship between Support Vector Machine and Gabriel Graph, In Proceedings of IEEE World Congress on Computational Intelligence-International Joint Conference on Neural Networks (WCCI'02-IJCNN'02), pages CD-ROM #1415, Honolulu, Hawaii (May 2002).
8）和田：最近傍識別器を用いた色ターゲット検出，情報処理学会論文誌：CVIM, Vol.44, No.SIG 17 (CVIM 8), pp.126-135 (2003).
9）Arya, S., Mount, D. M., Netanyahu, N.S., Silverman, R. and Wu, A. Y.: An Optimal Algorithm for Approximate Nearest Neighbor Searching, Journal of the ACM, Vol.45, pp.891-923 (1998).
10）Shibata,T., Kato,T. and Wada, T.: K-D Decision Tree: An Accelerated and Memory Efficient Nearest Neighbor Classifier, Proc. of 3rd ICDM, pp.641-644 (2003).
11）Vidal, E.: An Algorithm for Finding Nearest Neighbors in (approximately) Constant Average Time, PRL, Vol.4, pp.145-157 (1986).
12）Uhlmann, J. K.: Satisfying General Proximity/Similarity Queries with Metric Trees, Information Processing Letters, Vol.40, pp.175-179 (1991).
13）Sebastian, T. B. and Kimia, B. B.: Metric-based Shape Retrieval in Large Databases, Proc. of 16th ICPR, Vol.3, pp.291-296 (2002).

（平成 17 年 3 月 11 日受付）