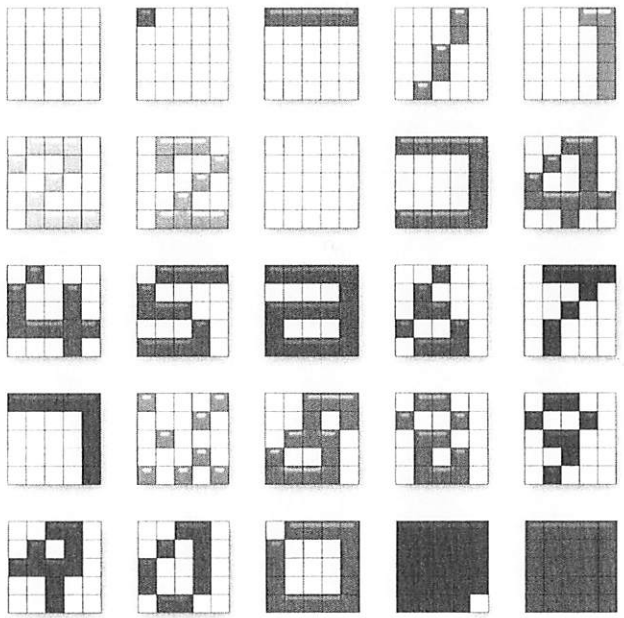
わかりやすい

パターン認識

第2版

石井健一郎・上田修功・前田英作・村瀬 洋 共著



Ohmsha

クラス間変動行列	$\mathbf{S}_B$
全変動行列	$\mathbf{S}_T$
自己相関行列	$\mathbf{R}$
事前確率	$P(\omega_i)$
事後確率	$P(\omega_i\|\boldsymbol{x})$
条件付き確率密度	$p(\boldsymbol{x}\|\omega_i)$
$\boldsymbol{x}$ の確率密度	$p(\boldsymbol{x})$

第1章
パターン認識とは

1.1 パターン認識系の構成

パターン認識（pattern recognition）とは，観測された**パターン**（pattern）をあらかじめ定められた複数の概念のうちの一つに対応させる処理である．この「概念」を**クラス**（class）[*1]と呼んでいる．例えばアルファベットの認識であれば，入力パターンを 26 個のクラスのいずれかに対応させる処理ということになる．パターンと言うと，人間の視覚に入ってくる 2 次元のパターンを思い浮かべるかもしれないが，パターン認識で扱う対象はもっと広い．例えば，音声のような時系列信号を処理して五十音や単語に対応させる音声認識もパターン認識の一分野であるし，心電図の波形を分析して異常のあるなしを判定するのも同様である．また，視覚や聴覚だけでなく，嗅覚や触覚などさまざまなセンサーを用いて状況を判断することもパターン認識と言うことができる．

人間は高度なパターン認識能力を備えており，この知的機能を機械で実現しようという試みは，計算機の出現以来研究者の中心的課題の一つであった．しかし，研究の進展に伴い，当初の期待に反してこの問題が見掛けほど簡単ではないことも，しだいに明らかになってきた．パターン認識の研究そのものを疑問視する時期もあったが，この分野の研究が依然活発であるのは，人間の知的機能を機械で実現しようという純然たる知的興味に加えて，パターン認識が潜在的に持つ高い実用的価値によるところが大きい．文字・音声・画像を対象とした認識装置は，多少の制約はあるものの実用化され，さまざまな分野で使われている．また，その高度化に対する要求や期待も高まりつつあり，パターン認識の研究は今後もますます活発になっていくものと思われる．

[*1] **カテゴリー**（category）と呼ぶこともある．

機械でパターン認識系を構成する場合，一般に図 1.1 の形をとることが多い．パターンが入力されると，まず**前処理**（preprocessing）部でノイズ除去，正規化などの処理を行う．続いて**特徴抽出**（feature extraction）部では，膨大な情報を持つ原パターンから識別に必要な本質的な特徴のみを抽出する．この特徴をもとに，**識別**（classification）部では識別処理を行う．識別処理は，入力パターンに対して複数のクラスのうちの一つを対応させることによって行われる．そのため，あらかじめ**識別辞書**（classification dictionary）を用意し，抽出された特徴を識別演算部においてこの辞書と照合することにより，入力パターンが所属するクラスを出力する．本書では，この辞書照合の部分を「識別」と呼び，パターンが入力されてから出力されるまでの前処理，特徴抽出処理，識別処理を総称して**認識**（recognition）と呼ぶことにする．

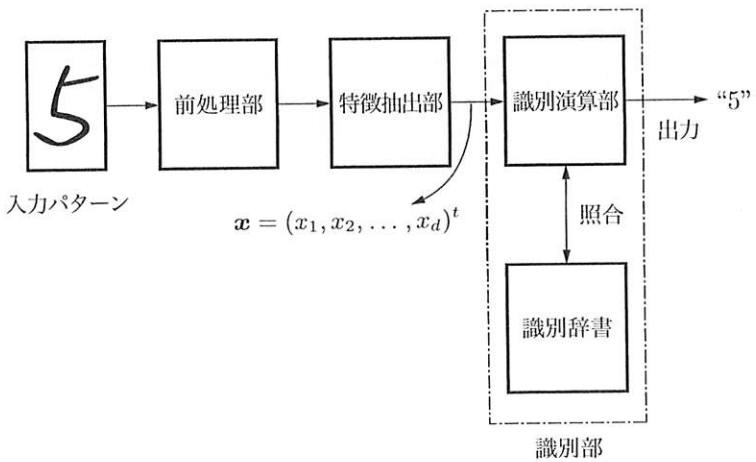


図 1.1　パターン認識系の構成

1.2　特徴ベクトルと特徴空間

(1)　特徴ベクトル

図 1.1 で示したように，認識のためには，原パターンから本質的な特徴を抽出する必要がある．認識系の中でも特徴抽出は，認識性能を左右する極めて重要な処理である．従来，特徴抽出は人間の直感によるヒューリスティックな方法に頼っていたが，**深層学習**（deep learning）による自動化も進みつつある．

特徴としては，さまざまなものが考えられる．例えば文字認識の場合，文字線の傾き，幅，曲率，面積，ループの数などは，しばしば用いられる特徴である．おのおのの特徴は数値で表され，通常はそれらを組にしたベクトルが用いられる．いま d 個の特徴を用いることにすると，パターンは次式のような d 次元ベクトル $\boldsymbol{x}$ として表される．

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^t \tag{1.1}$$

ここで，t は転置を表す．上のベクトルを**特徴ベクトル**（feature vector），特徴ベクトルによって張られる空間を**特徴空間**（feature space）と呼ぶ．したがって，パターンは図 1.2 に示す $\boldsymbol{x}$ のように，特徴空間上での 1 点として表される．また，対象としているクラスの総数を c とし，各クラスを $\omega_1, \omega_2, \ldots, \omega_c$ で表す．同一クラスに属するパターンは互いに類似しているから，特徴空間上でパターンは図 1.2 のようにクラスごとにまとまった塊として観測されるはずである．この塊を**クラスタ**（cluster）と呼ぶ．

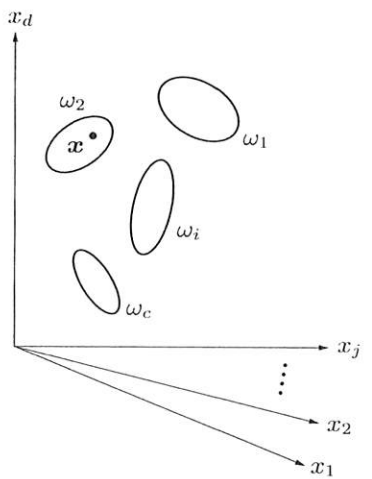


図 1.2　特徴空間での各クラスのパターンの分布

ここで，2次元的な広がりを持つパターンの認識を考える．文字を認識したり，X線写真から病変を判定したりといった処理がこれに相当する．例えば図 1.3 (a) は濃度情報を持つ2次元パターンの例である．このようなパターンに対し，以下のような単純な特徴を考えてみよう．まず，濃度値を有限個のレベルに制限し，実際の値をそれらのうち最も近いレベルに置き換えることにする（図 1.3 (b)）．さらに，パターンを図のような**メッシュ**（mesh）状に区切り，各メッシュをある濃度値で代表させる．j 番目のメッシュの濃度を x_j とすると，パターンは式 (1.1) のベクトルで記述できる．ここで，次元数 d はメッシュ総数に等しい．濃度のレベル数を q とすると，式 (1.1) で記述できるパターンは全部で q^d 通りとなる．図 1.3 (c) はこのようにして得られたパターンである．

(a) 原画像　(濃度レベル数 $q=8$) (b) 量子化　(メッシュ総数 $d=16\times16$) (c) 量子化+標本化

図 1.3　濃度情報を持つパターンの量子化と標本化

上で述べた処理のうち，前半は**量子化**（quantization）処理であり，後半は**標本化**（sampling）処理である．したがって，上で述べた処理は特徴抽出処理というより，単なるディジタル化処理と見ることもできる．ここではこのような場合も含めて特徴抽出と見なし，特に区別はしないことにする．

(2)　特徴ベクトルの多様性

以下では，このような特徴を手書き数字認識に適用してみる．クラス数は 10 ($c=10$) である．ここで入力されたパターンを 5×5 の 25 メッシュ ($d=25$) で標本化することにする．文字は基本的に白黒の二値パターンであるので，特徴

ベクトルの要素は

$$\begin{cases} x_j = 1 & (\text{黒：文字部分}) \\ x_j = 0 & (\text{白：背景部分}) \end{cases} \quad (j = 1, \ldots, d) \tag{1.2}$$

の二値と考えてよい．よって，レベル数は $q=2$ であるから，25 メッシュで表現できるパターンは $2^{25} = 33\,554\,432$ 通りとなる．図 1.4 にパターンの例を示す．図の (a) から始まって (y) までで，さまざまなパターンが表現できる．図から，5×5 メッシュは，数字を表現するにはかなり粗い標本化であることがわかる．

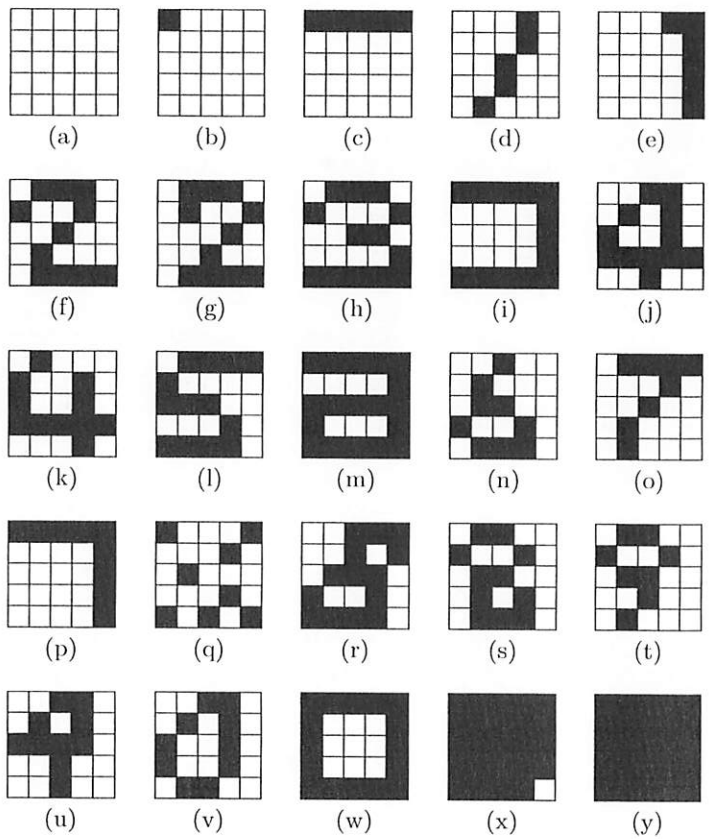


図 1.4　5×5 メッシュによる二値パターンの例

最も単純な識別系の構成法は，33 554 432 通りのすべてのパターンをそのクラス名とともに識別辞書として格納することである．これは，25 ビットデータのおのおのにクラス名が割り当てられた参照テーブルを作ることと等価である．この例では，図 1.1 の識別辞書は参照テーブルに対応し，識別演算部は参照テーブルの照合処理に対応している．特徴抽出部で標本化されたパターンは，必ず識別辞書中のいずれかのパターンと一致するので，一致したパターンのクラスを識別結果として出力することになる．ただし，図 1.4 から明らかなように，辞書中の 33 554 432 のパターンには数字としては意味のないパターンも多数含まれている．このようなパターンに対しては，11 番目のクラスとして**リジェクト**(reject)[*2] を割り当てておけばよい．**図 1.5** は，数字のパターンとして存在可能な領域とリジェクト領域で構成される特徴空間を表しており，この空間の 1 点が 33 554 432 通りのいずれかに対応している[*3]．

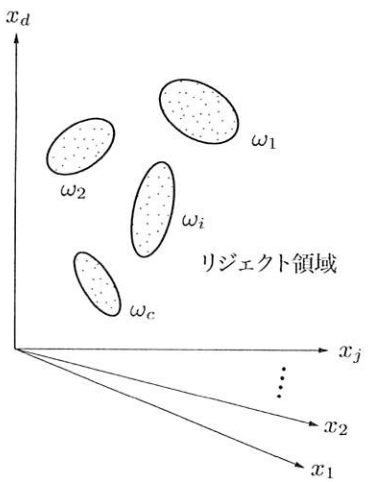


図 1.5　特徴空間とリジェクト領域

[*2] リジェクトには 2 種類がある．一つは，どのクラスにも属さないと判定される場合のリジェクトであり，ここで述べた例が相当する．もう一つは，複数のクラスが候補に挙がり，そのいずれとも判定しがたい場合のリジェクトである．例えば，図 1.4 (u) は “4” と “9” のいずれとも判定しがたいので，リジェクトせざるを得ない．**演習問題 1.1，1.2** を参照．

[*3] ただし，ここで扱う特徴は二値であるから，パターンは特徴空間上の超立方格子点を占めることになる．図では離散的な表現をとっていないが，特徴が連続的な値を持つ一般的な場合への拡張を考え，今後も図では連続的な表現を用いることにする．

さて，この識別辞書を作成するのにどれだけの手間が必要であろうか．この識別辞書の作成自体を自動化することはできない．なぜなら，一つひとつのパターンにクラス名を割り当てる作業そのものが，識別処理にほかならないからである．結局，識別辞書作成は人間の手作業によるしかない．いま，一人の人間が一つのパターンを見てクラス名を入力するのに 1 秒かかり，1 日 8 時間この作業を続けるとして，33 554 432 通りのすべてのクラス名を入力し終わるのには，単純な計算によれば 3 年 2 か月かかることになる．何人かで分担することにすれば実現不可能な作業ではないが，いずれにしても膨大な労力を要求されることになる．以上は，5×5 という粗い標本化での見積もりである．一方，数字のパターンを表現するには，少なくとも 50×50 メッシュ程度は必要である．このようにメッシュ数が増えたり，さらに濃度を持ったりすると，作業時間は天文学的数字となり，もはや識別辞書は現実的な時間内では作成不可能である．ここでは，メッシュの白黒という単純な特徴を考えたが，他の特徴を用いたとしても状況は同じである．

1.3　プロトタイプと最近傍決定則

(1)　プロトタイプ

上で述べた識別辞書構成法は，特徴ベクトルとして生起しうるすべての可能性を網羅し記憶する方法であり，理論的には可能であっても記憶容量，識別時間の点で非現実的である．

次善の策として，すべての可能性を網羅する代わりに，代表的なパターンのみを記憶する方法が考えられる．このようなパターンを**プロトタイプ**（prototype）と呼ぶ．入力パターンは特徴空間上でこれらのプロトタイプと比較され，最も距離が近いプロトタイプ，すなわち**最近傍**（nearest neighbor）の属するクラスを識別結果として出力する．これは，特徴空間上で近接しているパターン同士はその性質も互いに似ている，という仮定に基づいている．距離としてはユークリッド距離が使われることが多い．このような識別方法を**最近傍決定則（NN 法**：nearest neighbor rule）という．ここで，NN 法を定式化しておこう．

いま，n 個のパターンがその所属するクラスとともに $(\boldsymbol{x}_1, \theta_1), (\boldsymbol{x}_2, \theta_2), \ldots,$

$(\boldsymbol{x}_n, \theta_n)$ と与えられていたとする．ただし

$$\theta_p \in \{\omega_1, \omega_2, \ldots, \omega_c\} \qquad (p = 1, \ldots, n) \tag{1.3}$$

である．NN 法は次式のように書ける．

$$\min_{p=1,\ldots,n} \{D(\boldsymbol{x}, \boldsymbol{x}_p)\} = D(\boldsymbol{x}, \boldsymbol{x}_k) \quad \Longrightarrow \quad \boldsymbol{x} \in \theta_k \tag{1.4}$$

ここで

$$\boldsymbol{x}_k \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \tag{1.5}$$
$$\theta_k \in \{\theta_1, \theta_2, \ldots, \theta_n\} \tag{1.6}$$

であり，$D(\boldsymbol{x}, \boldsymbol{x}_p)$ は $\boldsymbol{x}$ と $\boldsymbol{x}_p$ との距離を表す．言うまでもなく，式 (1.4) における $\boldsymbol{x}_k$ は $\boldsymbol{x}$ の最近傍である．より一般的な方法として，入力パターンに最も近い k 個のプロトタイプをとり，その中で最も多数を占めたクラスを識別結果として出力する方法がある．これを ***k*-NN 法** (k-nearest neighbor rule) と呼ぶ．上で述べた NN 法は，1-NN 法に相当する．

図 1.6 は NN 法の処理を示したものである．図で各クラスに所属するパターンの存在領域が楕円で示されている．また，プロトタイプは白丸で示されてい

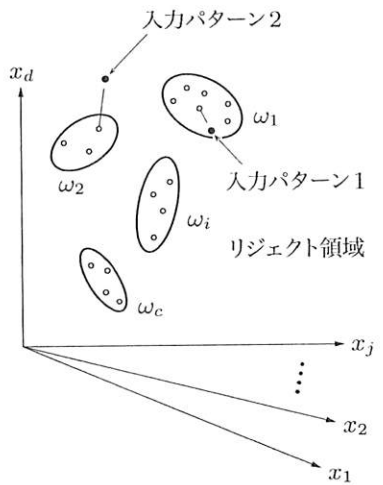


図 1.6　プロトタイプと最近傍決定則

る．例えば，入力パターン 1 は，その最近傍がクラス ω_1 に属することからクラス ω_1 と判定される．最近傍との距離があまり大きい場合には，文字としては意味のないパターンの可能性があるので，リジェクトする．例えば，入力パターン 2 は，そのまま忠実に NN 法を適用すればクラス ω_2 と判定されることになるが，パターンは楕円の外側に存在し，距離があらかじめ定められたしきい値を超えているのでリジェクトと判定されることになる．このような処置を施すことにより，先の方式と同様に，特徴空間上のすべての点にクラス名を割り当てたことになる．**図 1.7** に数字のプロトタイプの例を示す．

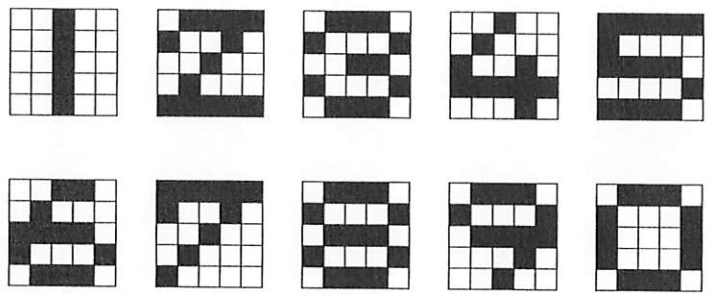


図 1.7　プロトタイプの例

この方法はプロトタイプのみを扱えばよいので，識別辞書作成作業は格段に軽減され，記憶容量，識別時間の問題も解消される．先の方式と比べれば，はるかに効率的である．いわば一を聞いて十を知るといった機能を持たせようというわけである．基本的なことだけを記憶してあとは類推により問題解決を図るという取り組み方は，人間の知的活動の中でもしばしば見られる戦略である[*4]．

coffee break

❖ NN 法を見直そう

NN 法は何の変哲もない識別法であるが，統計的には極めて興味ある性質を持つことが知られている（5.4 節 [1] 参照）．そのため，古来パターン認識の研究者が好んで取り上げたテーマであり，論文も膨大な数に上っている（文献 [Das91] は NN 法に関する論文のみを紹介している）．ただ，コンピュータの処理能力，記憶

[*4] 裁判の判決に対する過去の判例の役割，囲碁の定石も同様と考えられる．

容量が貧弱であった当時においては，NN 法はパターン認識の理論的側面を追求するための材料として捉えられていたにすぎず，その実用的な価値を認知される状況にはなかった．逆にそのような状況にあったからこそ，強力で洗練された学習アルゴリズムを求めて，多くの研究者が研究に取り組んできたと言うこともできる．

しかし，コンピュータの性能が飛躍的に向上した今日においては，むしろ NN 法が実用性においても十分現実的な手段になりうる時代になったと言える．実際，機械翻訳などに NN 法的手法を取り入れて成功を収めた例が報告されている [佐藤 97].

(2)　特徴空間の分割

それでは，機械にパターン認識機能を持たせるためには，プロトタイプをどのようにして設定すればよいのであろうか．先ほどの数字認識を例にとろう．まず，現実の手書き数字が特徴空間上でどのように分布しているかを知る必要がある．厳密な分布状況を直接求めることはできないので，実際に書かれた文字を収集し，それが現実の分布を反映しているデータと見なす．手書き文字には個人性がさまざまな変形となって現れるから，それらの変形をカバーするのに十分な量のパターンを収集しなくてはならない．そして，収集されたパターンを，実際起こりうるパターンの典型と見なし，これらのパターンを正しく識別できるように，プロトタイプの個数と特徴空間上での位置とを決定する[*5].

この目的を達成するための確実な方法は，収集された全パターンをそのままプロトタイプとすることである．これは**全数記憶方式**（complete storage）とでもいうべきもので，これを実現するには，計算機の処理能力と記憶容量が十分でなければならない．ただし，たとえ全数記憶方式をとったとしても，起こりうる実パターンのごく一部しかカバーできないことに注意しなくてはならない．もう少し効率的な方法として，少数のパターンをプロトタイプとして設定する方法が考えられる．人間の場合でも，要領の良い人は，暗記する事項を最小限に抑える工夫をするはずである．その極端な場合がクラス当たり 1 個のプロトタイプを用いる方法である．その場合，クラスを代表するプロトタイプとして，クラスの分布の重心を選ぶというのは合理的である．**図 1.8** は，重心をプロトタイプに選び，収集パターンを完全に分離できた例を示している．この例では，2 次元の特徴空間（$d = 2$）上に三つのクラス（$c = 3$）$\omega_1, \omega_2, \omega_3$ が存在する場合を取り上げている．NN 法を適用すると，クラス間を分離する境界は，二つのプロトタイプから等距離の線，すなわち垂直 2 等分線になる．図ではこれが太線で表されている．この境界を**決定境界**（decision boundary）といい，一般に d 次元特徴空間では $(d-1)$ 次元**超平面**（hyper plane）となる．また，図ではリジェクト領域が灰色で示され，各クラスとリジェクトとの境界が太線で示されている．このようにして，特徴空間は超平面によって三つのクラスとリジェクトの 4 領域に分割されたことになる[*6]．入力パターンがこれらのどの領域に存在するかによって，識別結果が異なってくるわけである．なお，上で用いた超平面という用語は以後もたびたび使用するので，ここで簡単に述べておく．

次元数 d が 3 の場合は特徴空間は 3 次元となり，$(d-1)$ 次元超平面は通常の 2 次元平面である．超平面とは，この 2 次元平面をそれ以外の次元へ一般化した概念である．すなわち，図 1.8 で示した例は 2 次元特徴空間であるので，超平面は 1 次元超平面，すなわち直線である．また，4 次元特徴空間の超平面は 3 次元超平面となる．

*5 本来なら，収集されたパターンの中からプロトタイプとしてふさわしいパターンを選ぶべきであるが，このようにプロトタイプを特徴空間上で動かしたほうがきめ細かな設計ができる．

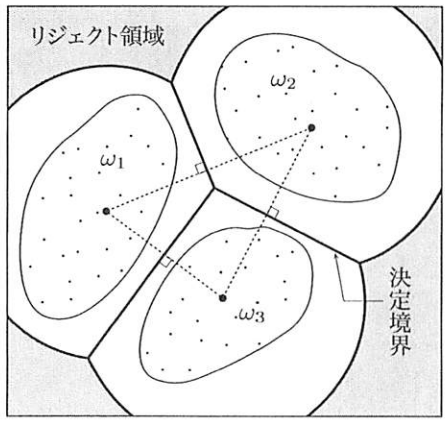


図 1.8　クラス当たり 1 個のプロトタイプによる特徴空間の分割

*6 リジェクト領域は超球面によって設定される．

このように，超平面とは d 次元空間の $(d-1)$ 次元**部分空間** (subspace) の呼称である．部分空間とは，特徴空間のベクトル（の組）によって張られる線形空間を指す．一般にその次元数は元の特徴空間の次元数より小さい．厳密な定義は線形代数の教科書を参照されたい．

coffee break

❖ 特徴抽出今昔

かつて文字認識の研究が活発だった 1970 年前後は，文字を認識するためのさまざまな特徴抽出法が提案された．しかし，どの手法も一長一短で，決定的な手法とはなり得なかった．当時，特徴抽出法は，直観と経験に基づき，人間が試行錯誤を通じて考案すべきものとされていた．本書初版の coffee break でも，「特徴抽出に王道なし」と題して，「特徴抽出は人間の経験と勘が物を言う世界であり，コンピュータで自動化することはできない」と述べた．

しかし，深層学習の登場以来，このような考え方には修正が求められるようになった．なぜなら，大量の学習パターンを与えれば，特徴抽出すらもコンピュータで自動化できることを示唆する結果が得られつつあるからである．ただし，深層学習で得られる特徴は，いったいどのような特徴を抽出しようとしているのか，またその高い性能が何に起因するのかを明示的に説明できないというもどかしさがある．一方，古典的な特徴抽出法は，例えば**付録 A.3** で示すように，特徴の意味や狙いが明確であり，深層学習で得られる特徴と対照的である．

大きな期待が寄せられている深層学習ではあるが，これまでブームになったり冬の時代を迎えたりというサイクルを繰り返してきた人工知能の歴史を振り返ると，特徴抽出が完全に自動化できると確信をもって言えるようになるには，もう少し時間が必要かもしれない．

演習問題

1.1　最近傍決定則（NN 法）を用いて，5×5 メッシュの数字パターンを識別する．パターンは，式 (1.2) に従って 25 次元の二値特徴ベクトルに変換される．プロトタイプはクラス当たり 1 個とし，図 1.7 で示したパターンを用いる．ただし，識別にあたっては，リジェクトを考慮した判定方法をとるものとする．この条件で**図 1.9** に示すパターン $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, $\boldsymbol{x}_4$ を識別し，その結果を示せ．

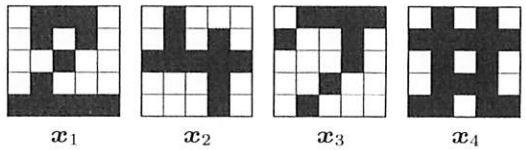


図 1.9　識別すべき入力パターン

1.2　文字パターンを識別するための特徴として，以下を考える．まず，二値化されたパターンを縦軸方向および横軸方向に射影し，黒メッシュのヒストグラムを求める．**図 1.10** は，図 1.7 に示したプロトタイプの一つである数字 “2” を例として用い，ヒストグラムを求めた結果である．

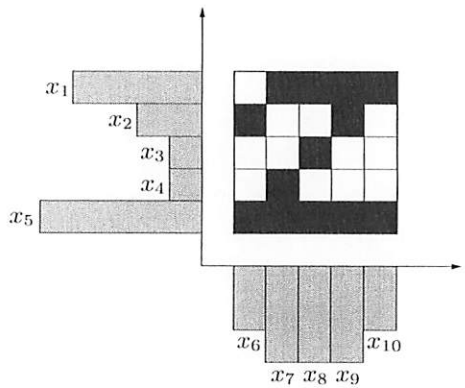


図 1.10　ヒストグラムに基づく特徴

この例のように，パターンが 5×5 メッシュであれば，ヒストグラムは図で示すように，10 次元ベクトル

$$\begin{aligned}\boldsymbol{x} &= (x_1, x_2, \ldots, x_{10})^t \\ &= (4, 2, 1, 1, 5, 2, 3, 3, 3, 2)\end{aligned}$$

として表せる．この 10 次元ベクトルを特徴として用い，**演習問題 1.1** と同条件で図 1.9 の 4 パターンを識別せよ．