目次



# ★ 9 パターン情報処理の応用例 (2) — 画像処理いろいろ（承前）

## ★ 9.2 画像の幾何変換

### ★ 9.2.1 画素値の補間と画像の拡大

画像の拡大縮小その他の幾何変換を行うためには，画素値を補間する必要がある．話を簡単にするために，まずは 1 次元のパターンの伸長を例にとる．

$x = 0, 1, 2, 3, 4$ の 5 点で $f(0), \ldots, f(4)$ という値をとるパターンを考え（図 1 左），このパターンを，$x = 0$ を基準として，適当な倍率 $a$ で変数の軸方向に伸長させたいとする．図中の 5 つの赤丸は，$a = 1.25$ の場合（$a > 1$ なのでパターンは伸びる）にこれらの点がどこに移るかを示している．例えば，元の座標（$x$ 軸）で $x = 1$ の位置にあった値 $f(1)$ は，新しい座標（$X$ 軸）では $X = 1.25$ に移っている．同様に，$f(2)$ は，$X = 1.25 \times 2 = 2.5$ の位置に移っている．このとき，元の座標 $x$ と新しい座標 $X$ の間には，次の関係が成り立っている．

$$X = ax \qquad\qquad x = \frac{X}{a} \qquad\qquad (1)$$

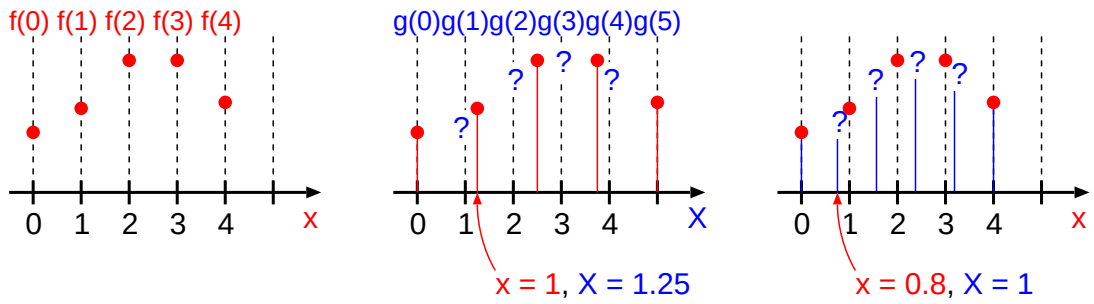



図 1: 1 次元パターンの伸縮

いま考えているのは変数が離散的なパターンなので，これでおわりというわけにはいかない．伸長結果の方も，変数が整数（$X = 0, 1, \ldots$）のときの値で表す必要があるからである．例の場合，図より $g(0) = f(0)$ および $g(5) = f(4)$ だから，これら 2 点の値はすぐにわかる．しかし，$g(1)$ から $g(4)$ までの 4 点は

$$g(1) = f(0.8) \qquad g(2) = f(1.6) \qquad g(3) = f(2.4) \qquad g(4) = f(3.2) \qquad (2)$$

となっており，これらの値はわからない．このように，離散的なパターンを変数軸方向に伸縮させようとすると，$f(0.8)$ や $f(1.6)$ のような値を何とかして計算する必要がある．

この $f(0.8)$ のような値を計算するには，その前後で値のわかっているもの ($f(0)$ や $f(1)$) を利用するのがよさそうである．既知の値を利用して間の値を補うので，このような計算を**補間**という．最も単純な補間の方法は,「最も近い既知の値をそのまま用いる」というものである．これを**最近傍法**という．もう少し工夫を考えると，前後 2 点を結ぶ直線を引いて考えることもできる．これを**線形補間法**という．図 2 左は，式 (2) の例でこれらの補間法を説明したものである．

補間: interpolation. **内挿**とも．最近傍法: nearest neighbor method. 線形補間法: linear interpolation method.

ここまでは 1 次元パターンの伸長を考えたが，2 次元の画像を拡大する場合も，話は同じである．原点を基準として $x$ 軸方向に $a$ 倍，$y$ 軸方向に $b$ 倍することで元画像の位置 $(x, y)$ の画素が位置 $(X, Y)$ に移るとしたら，

$$(X, Y) = (ax, by) \qquad (x, y) = \left(\frac{X}{a}, \frac{Y}{b}\right) \tag{3}$$

という関係が成り立つので，これを利用して最近傍法や線形補間法を適用すればよいのである（図 2 右）．ただし，前後 2 点ではなく近傍 4 点の画素値を利用する．例えば，$(x, y)$ の画素値を求める際には，

$$(\lfloor x \rfloor, \lfloor y \rfloor) \quad (\lfloor x \rfloor + 1, \lfloor y \rfloor) \quad (\lfloor x \rfloor, \lfloor y \rfloor + 1) \quad (\lfloor x \rfloor + 1, \lfloor y \rfloor + 1) \tag{4}$$

という 4 点を用いる．ここで $\lfloor x \rfloor$ は，$x$ を超えない最大の整数を表す (☆ 1).

☆ 1) 例えば $\lfloor \pi \rfloor = \lfloor 3.99 \rfloor = 3$ である．床関数といい，正の実数の場合「切り捨て」に相当する．ガウスの記号 $[\cdot]$ で表すこともある．また，$\lceil x \rceil$ というのもある（天井関数）.

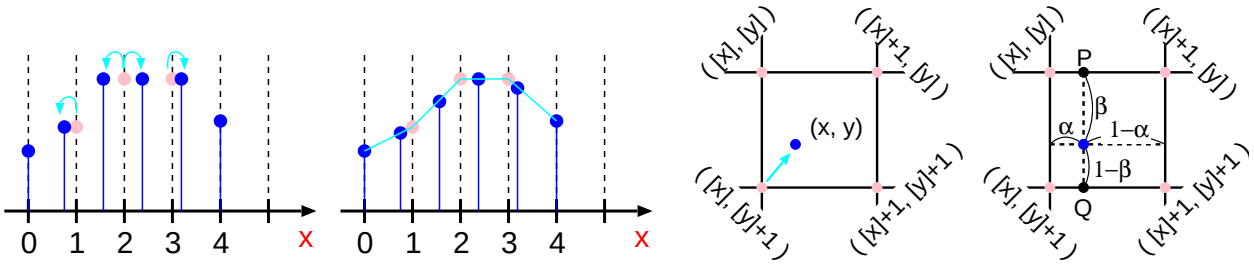



図 2: 最近傍法と線形補間法．左の 2 つは最近傍法／線形補間法による 1 次元パターンの補間を示し，右の 2 つは最近傍法／線形補間法による画素値の補間の例を示す.

線形補間法での画素値の補間法をちゃんと説明すると次のようになる．点 $(x, y)$ が図のような位置にあるとする． $0 \leq \alpha, \beta \leq 1$ である．点 $P, Q$ は左右の点を $\alpha : 1 - \alpha$ に内分する位置にあるので，その画素値は，$x$ 軸方向の線形補間より

$$(P \text{ の画素値}) = (1 - \alpha) f(\lfloor x \rfloor, \lfloor y \rfloor) + \alpha f(\lfloor x \rfloor + 1, \lfloor y \rfloor) \tag{5}$$

$$(Q \text{ の画素値}) = (1 - \alpha) f(\lfloor x \rfloor, \lfloor y \rfloor + 1) + \alpha f(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1) \tag{6}$$

と表せる．これらを用いて，今度は $y$ 軸方向の線形補間を考えると，

$$((x, y) \text{ の画素値}) = (1 - \beta)(P \text{ の画素値}) + \beta(Q \text{ の画素値}) \tag{7}$$

が得られる (☆ 2).

☆ 2) 縦横二方向で線形補間を行なうので，この方法は双線形補間法 (bi-linear interpolation method，双一次〜，バイリニア法などとも) と呼ばれることもある.

実際に画像を拡大した例を図 3 に示す．最近傍法では隣接画素に同じ値が代入されることがあるためにガタガタした印象であるが，線形補間の方は少しぼけているがより自然な拡大画像となっている．

このように線形補間法ではある程度画質のよい拡大を実現できる．しかし実際の画像処理では，よりよい画質を求めて，非線形関数やスプラインを利用したり，4 近傍だけでなくさらにその周囲の画素値も利用したりする改良手法が用いられることもある (☆ 3)．

☆ 3) 3 次多項式を用いるバイキュービック法がよく知られている（興味のある人は調べてみよう）．たいていの画像処理ソフトウェアでは，バイキュービック法を含めた 3 手法の中から使用する手法を選べるようになっている．

図 3: 画像の拡大．左から順に，元画像，最近傍法による拡大，線形補間による拡大．縦横とも 1.25 倍．

**Q1.** $f(3,6)=88, f(4,6)=64, f(3,7)=72, f(4,7)=104$ として，$f(3.75,6.5)$ の値を最近傍法および線形補間法で求めよ．

### ★ 9.2.2　画像の縮小とエイリアシング

画像の縮小は，拡大と同様に式 (3) を考えて（ただし $0<a,b<1$(☆ 4)）画素値の補間を行なえばよい．しかし，$a,b$ が小さい場合は，画素値を間引く，すなわち標本化することになるので，エイリアシングを起こさないように注意する必要がある．一般的には，縮小する前に平滑化を行なって高周波数の成分を除去する（ローパスフィルタリングする）方法が用いられる．

☆ 4) もちろん，$1<a$ かつ $0<b<1$ とすれば横に拡大しつつ縦に縮小することになる．また，$a,b$ を負の値にすれば画像を反転することになる．

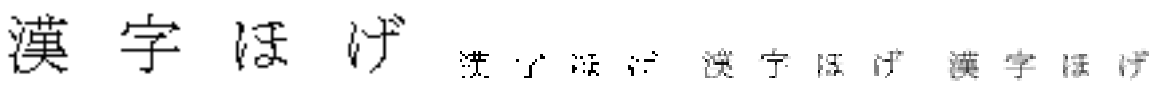


図 4: 画像の縮小．左から順に，元画像，最近傍法，線形補間，ローパス＋最近傍法．縦横とも 0.5 倍．

### ★ 9.2.3　アフィン変換

画素値の補間さえできれば，式 (3) のように元画像の座標 $(x, y)$ と変換後の新しい座標 $(X, Y)$ とを関係づける式を定めることで，様々な幾何変換が可能となる．特に，以下の式による変換がよく用いられる．これを（2 次元の）**アフィン変換**という．

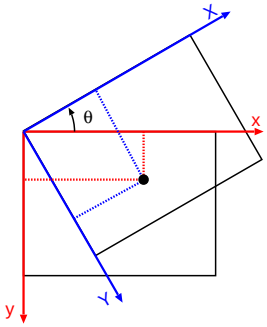
幾何変換: geometric transformation.
アフィン変換: affine transformation.

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} x \\ y \end{pmatrix} + \boldsymbol{t} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{8}$$

アフィン変換は，行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{t}$ の決め方次第で**拡大縮小**，**平行移動**，**回転**，**反転**などを表せる．すなわちこれらを一般化した変換となっている．例えば，

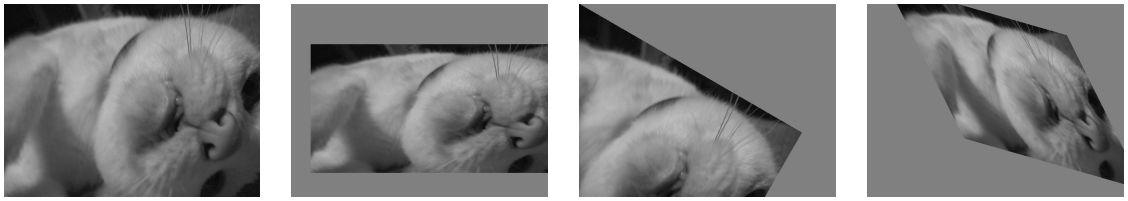
$$\boldsymbol{A} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad \boldsymbol{t} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{9}$$

とおけば，右図のように $\theta$ 回転する変換となる．図 5 に例を示す．はじめの例（左から 2 つ目の画像）は，元画像を縦に $\frac{2}{3}$ に縮小してから右下に平行移動したものとなっている．次の例は，右図に示すように原点周りに $\frac{\pi}{6}$ だけ回転したものとなっている (☆ 5)．最後のものは，一般のアフィン変換の例である．




☆ 5) 上図のように反時計周りに回転した座標系で元の画像を観察すると，画像自体は時計回りに回転して見える．

紹介はしないが，幾何変換としては，アフィン変換でも表せないようなもっと複雑なもの（例えばレンズを通して歪めるような変換）もいろいろある．$(x, y)$ と $(X, Y)$ とを関係づける式を定めて画素値を補間する，という方法はどんな幾何変換でも共通である．



$\boldsymbol{A} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{2}{3} \end{pmatrix}$, $\boldsymbol{t} = \begin{pmatrix} 50 \\ 100 \end{pmatrix}$　　$\boldsymbol{A} = \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}$, $\boldsymbol{t} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$　　$\boldsymbol{A} = \begin{pmatrix} 0.7 & 0.4 \\ 0.2 & 0.8 \end{pmatrix}$, $\boldsymbol{t} = \begin{pmatrix} 50 \\ -50 \end{pmatrix}$

図 5: 画像のアフィン変換．左端は元画像．元画像の範囲からはみ出した所はグレイで塗りつぶしてある．

## ★9.3 Hough 変換

Hough 変換 (☆ 6) は，画像の中から直線や円といった図形を検出する方法の一つである．図 6 に，Hough 変換による直線検出の原理と方法を示す．

☆ 6) Hough 変換 (ハフ変換，Hough transform): P. Hough が考案した．

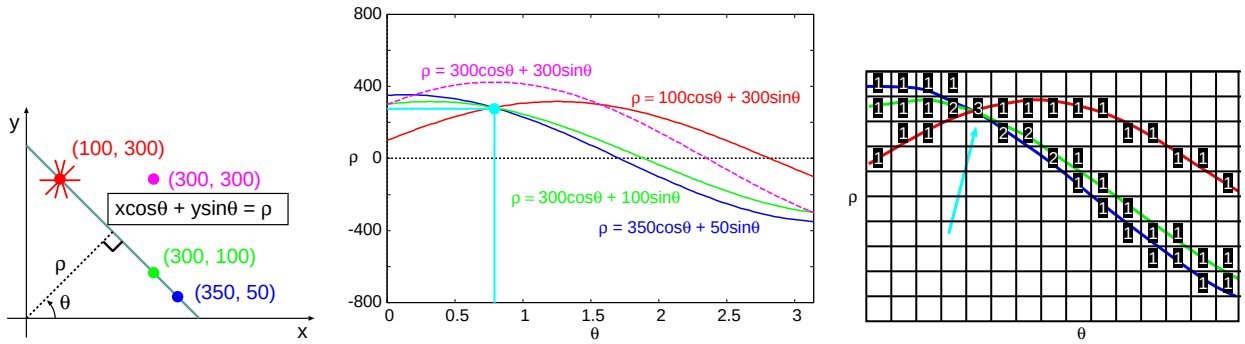



図 6: Hough 変換.

---

図 6 左: 平面上の点 $(x, y)$ を通る直線は，図に示す $\rho, \theta$ という二つの値を用いて，

$$x\cos\theta + y\sin\theta = \rho \tag{10}$$

という式で表せる (☆ 7)(☆ 8)．図中の灰色は，点 $(100, 300), (300, 100), (350, 50)$ を通る直線であり，この例では．$(\theta, \rho) = (\frac{\pi}{4}, 200\sqrt{2})$ となる．

図 6 中: ある点を通る直線は，様々な $\rho$ と $\theta$ の組み合わせで無数にあるが，それら全体は $(\theta, \rho)$ 平面上で正弦曲線を描く．例えば，点 $(100, 300), (300, 100), (350, 50)$ のそれぞれを通る直線たちは図に示すような 3 つの曲線を描き，その交点の $(\theta, \rho)$ の値は，この 3 つの点を通る直線を表す．

図 6 右: $(\theta, \rho)$ 平面を格子状に分割したものに対応する二次元配列を用意し，画像中の各点について，その点を通る直線たちを表す曲線上の配列要素に一票を加える，という投票を行うと，画像中に存在する直線に対応した配列要素に票が集まることになる．

---

☆ 7) $y = ax + b$ のような表し方も考えられるが，その場合，$a$ の動く範囲を無限大までとらないと任意の直線を表せないため実用的でない．式 (10) の $\theta, \rho$ ならいずれも有限の範囲で済む．

☆ 8) 画像上の座標は，左下を原点にして右と下に $x$ 軸 $y$ 軸それぞれの正の向きをとって表すことが多いが，ここでは簡単のため上に $y$ 軸正の向きを選んでいる．

☆ 9) フィルタリングによって輪郭部分を抽出し，二値化する等.

図 6 では少数の点しか描いていないが，実際には画像から何らかの方法 (☆ 9) で抽出した多数の特徴点のそれぞれについて $(\theta, \rho)$ 平面上で投票を行い，得票数の多い $(\theta, \rho)$ を選び出す処理を行うことになる．この場合，1 位のみでなく，一定以上の票数を集めた $(\theta, \rho)$ を全て直線の候補とするといった扱いを考えると，画像中から複数の直線を検出することもできる．図 7 に実際の画像での直線検出例を示す.

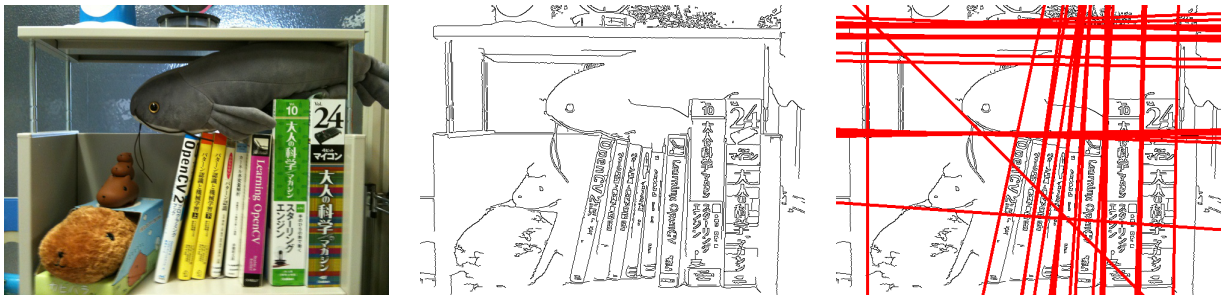


図 7: Hough 変換による直線検出例．左から，入力画像，フィルタリングによって抽出した特徴点，直線検出結果.

ここでは直線検出を例にあげたが，Hough 変換は，いくつかの変数の組み合わせで表せるような図形であれば適用可能である (☆ 10)．ただし，変数の数が多くなると投票する空間の次元数が大きくなって難しくなる．また，単純な直線検出の場合でも，前処理である特徴点抽出の結果によって検出結果が大きく変わるので，有効な結果を得るためには前処理の方法等も注意深く設定する必要がある．

☆ 10) 例えば円ならば，中心の座標と半径の計 3 つの変数の組で表せる．

## ★ 9.4　テンプレートマッチング

画像中にあるものが存在するかどうかを調べる方法を考えよう．最も原始的な方法は，見本画像（**テンプレート**と呼ぶ）を用意し，入力画像中にそれと似た領域がないかどうか比較する，というものである．テンプレートと入力画像との比較は，位置を少しずつずらしながら何度も行なう（図 8 左上）．このような方法を，**テンプレートマッチング**という．

テンプレートマッチング: template matching.

テンプレートマッチングでは，テンプレートと入力画像の部分領域との**類似度**（似ている度合）を位置毎に測り，この値があらかじめ定めた基準を満たせば，テンプレートに写っているのと同じものがその位置に存在すると判断する．類似度の計算には，テンプレートと入力画像の一部との**平均二乗誤差** (MSE) がよく用いられる (☆ 11)．これは次式で与えられる．

類似度: similarity.
平均二乗誤差: mean squared error.

☆ 11) MSE が小さい方が類似度が大きい．この他，類似度としては正規化相関係数もよく用いられる．

$$\mathrm{MSE}(x,y) = \frac{1}{wh}\sum_{i=0}^{w-1}\sum_{j=0}^{h-1}(I(x+i,y+j)-T(i,j))^2 \tag{11}$$

ここで，$\mathrm{MSE}(x,y)$ は，入力の $(x,y)$ を左上隅とする $w\times h$ 画素の領域とテンプレートとの MSE であり，値が小さいほど類似していると考える．また，$I(a,b), T(a,b)$ は入力とテンプレートの位置 $(a,b)$ の画素値を表し，$w,h$ はテンプレートの幅と高さを表す．図 8 にテンプレートマッチングの結果の例を示す (☆ 12)．

☆ 12) この実験ではテンプレートの位置をずらしているだけであるが，実用的には，物体の大きさや向きの変化にも対応するため，入力画像を拡大縮小したり回転したものともマッチングするなど，より複雑な処理が必要となる．

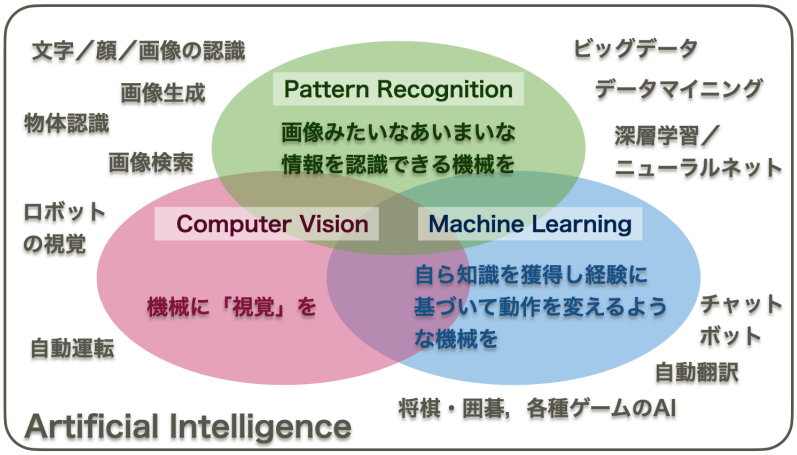

左の図は，各位置での MSE の値を明るさで表したもの．黒いほど MSE が小さい（類似度が高い）．

図 8: テンプレートマッチング．下段の 3 つの画像では，MSE の値が基準より小さかった位置にテンプレートと同サイズの赤枠を表示している．左のものほど MSE の基準が小さい（厳しい）．

# ★10　パターン認識と機械学習 (1) — パターン認識，機械学習とは

今回から，**パターン認識** (☆ 13) と**機械学習** (☆ 14) を取り上げる．以下の図は，これら 2 つの分野と関連の深い**コンピュータビジョン** (☆ 15) を加えて，これらの分野とその周辺のキーワード（のうちごく一部）を示したものである．

☆ 13) パターン認識: pattern recognition
☆ 14) 機械学習: machine learning
☆ 15) コンピュータビジョン: computer vision




## ★10.1　パターン認識とは

パターン認識とは，コンピュータを用いて，画像や音声のようなパターン情報の中の規則性を自動的に見つけ出し，それに基づいてパターンをいくつかのカテゴリに分類するような情報処理のことをいう（第 1 回講義資料も参照のこと）．音声認識，文字認識，画像認識などが典型例である．例えば...

**例 PR1:** 母音の音声波形から「a,i,u,e,o」を識別する (波形を ‘a’, ‘i’, ‘u’, ‘e’, ‘o’ の 5 つのカテゴリに分類する)
**例 PR2:** 人の顔画像をいくつかのカテゴリに分類する (カテゴリの設定の仕方次第で様々な応用が考えられる)
**例 PR3:** 様々な動物の写った画像が与えられたときに，それぞれの画像に写っている動物の種類を識別したい (ネコ，ペンギン，カピバラ,…)

パターンをカテゴリ分類する処理とは少し異なるが，類似の情報処理として，次のような問題もパターン認識で扱う対象となる．

**例 PR4:** 顔画像から年齢を推定する
**例 PR5:** 大量の画像の中から，ある画像と似た画像を見つけ出す

近年のパターン認識では，研究レベルでも実用レベルでも，後述する機械学習を利用した方法を採ることがほとんどである．

## ★ 10.2 機械学習とは

機械学習とは，人間のように自ら知識を獲得し経験に基づいて動作を変えるような機械／コンピュータ／アルゴリズムの実現を目指す研究分野またはその技術を指す．人間が行なっているような知的な情報処理の仕組みを人工的に実現することを目指す**人工知能** (☆ 16) の一分野であるが，現在ではより幅広く，データの中から何らかの知識や法則性を自動的に見つけ出したり，データに基づいて何らかの判断を自動的に下したりするような情報処理全般を対象とするものとなっている．大量のデータが与えられたときに，そこに潜む規則性を自動的に見つけ出し，そこから何らかの情報を取り出すような情報処理は，機械学習の応用の例である．例えば... (以下の括弧の中は，取り出す情報がどのようなものかを表す)

☆ 16) 人工知能: artificial intelligence, AI

**例 ML1** 時々刻々変化する気温，湿度，電力使用量の数値から，少し未来の電力使用量の推定値を求めたい (未来の電力使用量 (実数値))

**例 ML2** 通販サイトの購買履歴データをもとに，顧客を買い物傾向の似たグループに分類したい (例えば「グループ 1」から「グループ $K$」までの $K$ 通りのいずれか)

先のパターン認識の例も，機械学習の問題の一つと考えることができる．

**例 PR1:** 母音の識別 ('a', 'i', 'u', 'e', 'o' の 5 つのカテゴリのいずれか)
**例 PR2:** 顔画像の認識 (顔/非顔，男性/女性, A さん/B さん/C さん..., etc.)
**例 PR3:** 画像中の動物の認識 (ネコ/ペンギン/カピバラ/…)
**例 PR4:** 顔画像から年齢を推定する (年齢の数値)

これらの処理は共通して，何らかのデータ $\boldsymbol{x}$ が入力されると適当な値 $\boldsymbol{y}$ を出力する関数として次式のように表すことができる．

$$\boldsymbol{y} \longleftarrow \boxed{f} \longleftarrow \boldsymbol{x} \qquad\qquad \boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{w}) \tag{12}$$

入力 $\boldsymbol{x}$ は，問題に応じて様々な形式で与えられる．出力 $\boldsymbol{y}$ (☆ 17) は，例 ML1 や PR4 のように多様な数値をとる場合もあれば，例 ML2 などのように何通りかの値しかとらない場合もある．ここで $\boldsymbol{w}$ は，関数 $f$ に含まれるパラメータ (変数) (☆ 18) である (☆ 19)．パラメータを変えると，同じ入力に対しても異なる出力を与えることになる．このように式で表現すると，機械学習とは,「入力に対して「望ましい出力」が得られるように，パラメータ $\boldsymbol{w}$ をデータから自動的に決定する方法」に関する分野/技術と言うことができる．

機械学習において，望ましい出力が得られるようにパラメータを調節する過程を**学習** (☆ 20) といい，その際に用いるデータを**学習データ** (☆ 21) という．学習データで望ましい出力が得られることは当然大事であるが，機械学習の目標は，学習後に，学習時に遭遇したことのない未知のデータが入力されても，望ましい出力が得られる (これを「**汎化能力**がある」という) ようにすることである．

☆ 17) ここの例は全て出力が一つの値をとるものであるが，一般には出力が複数の値の組であってもよいので，ここでは $\boldsymbol{y}$ をベクトルして表している．

☆ 18) パラメータ: parameter, 助変数，媒介変数.

☆ 19) 例えば，$y = f(x) = ax + b$ では $a$ と $b$ がパラメータ.

☆ 20) 学習: learning, 訓練 (training) とも．

☆ 21) 訓練データともいう.