2012 年度春学期　画像情報処理　第１４回

第５部・パターン認識と最近の話題／ パターン認識について

パターン認識は，画像工学のなかで統計学にもっとも密接に結びついた分野です。パターン認識とは，パターンを形成するベクトルをいくつかのカテゴリーに分類することということができますが，これらのベクトルはランダム現象によるゆらぎを受けますから，統計学的枠組みで分類法を考える必要があります。

パターン認識とは

パターン認識とは，対象となるパターン群をいくつかのカテゴリーに分類することです。ふつうは，どのカテゴリーに分類するかがすでにわかっているようなパターン例を「学習」して分類の方法を決め，そのうえで，どのカテゴリーに分類するかが未知のパターンを分類して，認識の結果とします。例えば，手書きの文字を認識することを考えてみましょう。事前に「A」であることがわかっている手書き文字の例，「B」であることがわかっている例，…を学習しておきます。そして，未知の文字が入力されると，それを分類して，「A」であるか，「B」であるか，…を答えます。

通常，パターンの数はカテゴリーの数よりもはるかに多くなります。「A」という文字を手書きで書くと無数の変形が考えられますが，これらはすべて「A」というカテゴリーに分類する必要があります。もっとも簡単な場合として，入力される手書き文字が「A」と「B」の２通りしかないとしましょう。このとき，パターン認識装置は各々の入力手書き文字を「A」または「B」のどちらかに分類しなければなりません。このもっとも簡単なパターン認識でさえも，手書き文字は通常ゆがんでおり，またどちらのカテゴリーに分類すべきか判断に苦しむ文字が入力されることもしばしばあるので，難しい問題となります。

パターン認識では，各々のパターンから「特徴ベクトル」が抽出されます。特徴ベクトルとはパターンを少数の変量で要約したもので，例えば画像パターンの場合でいえば平均輝度，エッジ（画像中にある物体の輪郭）方向の分布，空間周波数分布などが用いられます。特徴ベクトルを用いるのは，入力パターンの次元数を小さくするためです。例えば，256×256 画素の画像は，そのままでは 65536 次元ベクトルとなります。65536 次元空間でパターンを分類すると，入力パターンの細かい部分まで分類に用いるため，入力パターンがちょっとゆがんでいるだけでも分類誤りを生じてしまいます。そこで，入力パターンの細かい違いを要約するために特徴ベクトルを導入するわけです。

特徴ベクトルの抽出の問題は，画像パターンなら画像処理の問題として，音声パターンなら音声処理の問題として研究されています。

「次元のわな」

統計学の観点にもとづいてパターンを分類するには，各カテゴリーに属するパターン群の特徴ベクトルの，分布の性質を知らなければなりません。しかし，その分布は実際には不明で，わかっているのはすでに分類されている標本だけです。ですから，前節のように，この母集団分布を標本から推定する必要があります。

パターン認識問題の場合，特徴ベクトルを用いたとしても，パターンを分類するためにさまざまな特

徴を調べる必要があるので，特徴ベクトルの次元は高くなりがちです。一方，学習に用いる，すでに分類されている標本ベクトルは，それほど多くは用意できません[1]。高次元空間で少ない標本ベクトルから母集団分布を推定するのは，標本がすき間だらけに配置されていることになるので難しく，推定は不正確になります。これを「次元のわな」(curse of dimensionarity) とよびます。

ニューラルネットワーク — 「学習」のはじまり

ニューラルネットワークは，人間の脳細胞と神経回路を模した演算によって，人間の行なう複雑なパターン認識などを実現しようというものです。ニューラルネットワークの特徴のひとつが，その学習能力です。学習とは，ネットワークの入出力例（パターン認識でいえば，文字の画像と読み取り結果など）をネットワークに呈示し，その入出力関係をネットワークが行なうようにネットワークを調節することです。

ニューロンとニューラルネットワーク

ニューラルネットの研究は，まず神経素子すなわち**ニューロン (neuron)** をモデル化することから始まりました。生理学的研究から，

1. ニューロンは他のニューロンから電気的・化学的刺激を受けとる。
2. 受け取る刺激の総和が一定量を超えると「発火」して，他のニューロンに刺激を与える。

というモデルが提示されました。これを図 1 のような記号で表します。単純化するため，i 番のニューロンの x_i の状態 (status) を，$x_i = 1$ →発火している・$x_i = 0$ →発火していないとし，発火しているとき 1 の量の刺激が他のニューロンへ送られます。また，刺激を受け取る際，神経素子間の結合には重みが設定されており，ひとつのニューロンには，各結合から（重み$\times$ 1 または 0）の刺激が届きます。すなわち，j 番のニューロンの状態を x_j とし，j 番と i 番のニューロン間の結合重みを wji とすると，j 番のニューロンから i 番のニューロンに届く刺激の量は $w_{ji}x_j$ となります。そして，その総和 $\sum_j w_{ji}x_j$ が一定量（しきい値 (threshold)）を超えると, この i 番のニューロンが発火します。すなわち，しきい値を T として，しきい値関数 f を

$$f(x) = \begin{cases} 1 & x \geq T, \\ 0 & x < T. \end{cases} \tag{1}$$

とするとき，$f(\sum_j w_{ji}x_j) = 1$ ならば i 番のニューロンが発火，すなわち $x_i = 1$ とし，それ以外は $x_i = 0$ とします。

ニューラルネットによる情報処理では，このようなニューロンを多数結合し，ニューロンの初期状態を与えることで情報を入力し，上のような刺激の伝達によってニューロンの状態を変化させて，変化後のニューロンの状態を出力とします。ニューロンの結合の形（トポロジー）には，おおまかにいって**階層型**と**相互結合型**の 2 種類があります。

図 2 の (a) が階層型ネットワークで，視神経における情報処理をモデルとして生まれたものです。1 つの層にあるニューロン全体で 1 つの情報を保持し（例えば，ニューロンを画素と考えて視覚的パター

[1] もっとも，この状況は近年大きく変わりつつあります。情報ネットワーク技術の発達で，大量のデータを集めることは，昔に比べてはるかに容易になっています。このことにより，パターン認識の技術もいま大きく変わりつつあります。

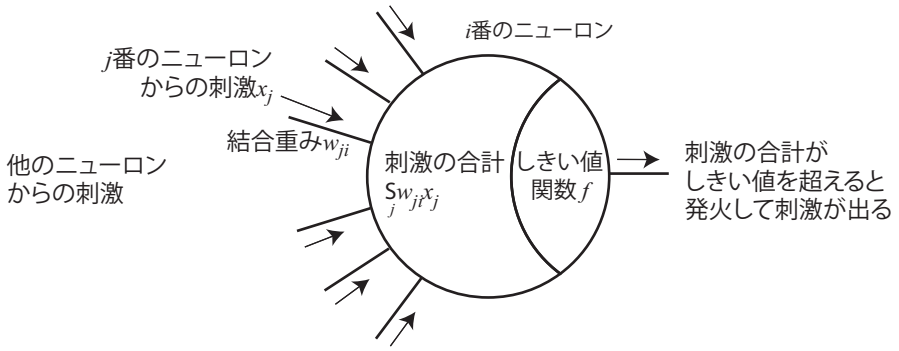

図 1: ニューロンのモデル

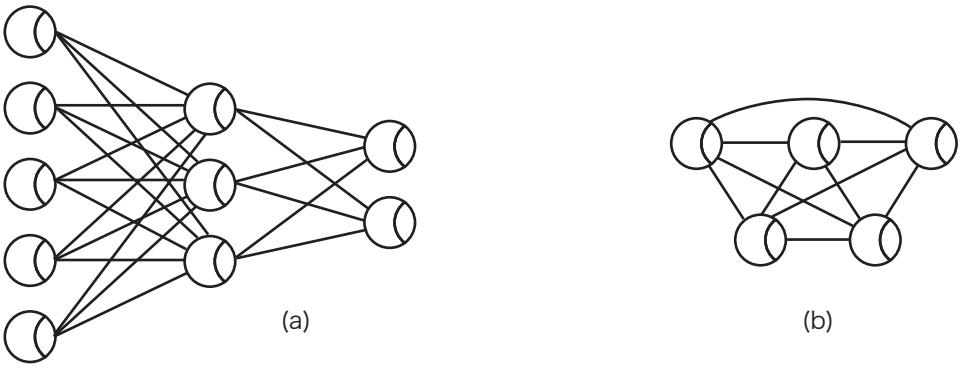

図 2: ニューラルネットワークのトポロジー。(a) 階層型 (b) 相互結合型

ンとする)，これがある層から次の層へ一定方向に刺激となって流れてゆきます。そして，最終層に刺激が届いたときの最終層のニューロンの状態が，このニューラルネットワークの出力結果となります。階層型ネットワークは，画像処理やパターン認識に多く用いられます。

一方，図 2 の (b) が相互結合型ネットワークで，こちらは層はなく，各ニューロンが互いに各ニューロンと結合しています。こちらの場合は，結合重みを一種の「プログラム」として問題に合わせて適切に設定し，ニューロンの初期状態を設定した後互いに刺激を交換しあいます。その動作が収束してそれ以上刺激のやりとりが起こらなくなったとき，その時のニューロンの状態を出力結果（あるいは問題の解）とします。相互結合型ネットワークは，最適配置問題を解いたり，連想記憶（記憶したいニューロンの状態に応じて結合重みを設定しておき，ニューロンの初期状態を与えて刺激を交換しあうと，記憶した状態の中で初期状態にもっとも近い状態に収束する。すなわち，初期状態からそれにもっとも近い記憶が「想起」される）として用いられます。

以下，この講義では，パターン認識に関連の深い，階層型ニューラルネットワークを取り扱います。

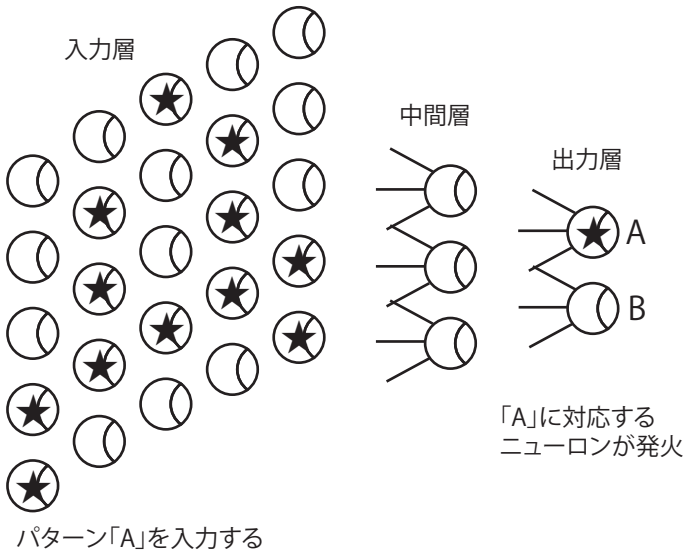

図 3: 階層型ニューラルネットワークによるパターン認識

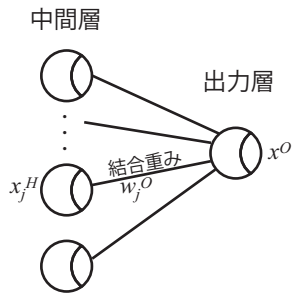

図 4: ２層間の結合重みの学習

パーセプトロン学習とその限界

パーセプトロン (perceptron) とは，ニューラルネットワーク研究の初期に提案された，もっとも簡単な階層型ニューラルネットワークで，図 3 のように入力層，中間層，出力層の 3 層からなっています。その利用法は，例えば入力層に「A」という文字パターンが入力されると出力層で「A」に対応するニューロンが発火し，以下，「B」のパターンにはBのニューロンが，「C」のパターンにはCのニューロンが，。。。というようにしてパターン認識を行うというものです。

パーセプトロンをはじめとする階層型ニューラルネットワークの特徴は，結合重みを学習によって定めることです。学習とは，ニューラルネットワークが望みの出力をするようにするための結合重みの調整を，入出力の例のみを用いて行うことです。パーセプトロンにおいては，学習は中間層と出力層の間だけで，以下のような手順で実現されます。以下，簡単のため出力層のニューロンは 1 つだけでその状態を x^O とします（図 4)。

1. 所望の入出力の例をいくつか用意します。これは例えば，入力層に「A」というパターンを入力したとき出力層のニューロンの状態が「1」になってほしい，といったものです。各入出力例において，入力層に入力例を与えたときの，中間層の j 番目のニューロンの状態を x_j^H とします。

2. 中間層と出力層間の初期結合重み w_j^O を適当に（例えばランダムに）決めておきます。

3. 入力の例を入力層に与え，その時の出力 x^O を調べます。

4. 出力層での所望の出力を y^O とします。もし，所望の出力が $y^O = 1$ なのに現実の出力が $x^O = 0$ である $(x^O - y^O = -1)$ ならば，それは出力層のニューロンにやって来る刺激の総和 $\sum_j w_j^O x_j^H$ が足りないからです。そこで，中間層のうち状態 x_j^H が 1 であるようなニューロンとの結合重み w_j^O を増やします。また，もし，所望の出力が $y^O = 0$ なのに現実の出力が $x^O = 1$ である $(x^O - y^O = +1)$ ならば，それは出力層のニューロンにやって来る刺激の総和 $\sum_j w_j^O x_j^H$ が多すぎるからなので，中間層のうち状態が 1 であるようなニューロンとの結合重みを減らします。以上のことは，中間層

の j 番のニューロンと出力層のニューロンとの結合の新しい重み $w_j'^O$ を

$$w_j'^O = w_j^O - (x^O - y^O)x_j^H. \tag{2}$$

にしたがって更新することで実現できます。

5. 所望の出力が得られるようになるまで，3，4 を繰り返します。

この方法をパーセプトロン学習といいます。この方法は簡単ですが，学習アルゴリズムが収束せずに結合重み w_j^O がいくつかの組合せの間を循環してしまう場合があることが知られています。これは，1 回の学習で w_j^O から $w_j'^O$ へ修正する修正量が大きすぎ，急激に結合重みを修正しすぎるからです。そこで，しきい関数 f を，例えば次のシグモイド関数

$$f(x) = \frac{1}{1 + \exp(-\lambda(x - T))}, \tag{3}$$

のような連続関数に置き換えます（図 5，λ はパラメータで，$\lambda to \infty$ のとき (3) 式の f はしきい関数になります)。そして，ニューロンの状態 x_j^H や x^O を連続値とし，(2) 式に小さな正の実数 ε（学習係数といいます）を導入して

$$w_j'^O = w_j^O + \varepsilon(y^O - x^O)x_j^H. \tag{4}$$

のように「徐々に」学習する方法があります。これは δ-**ルール**とよばれています。

さて，パーセプトロンでは，出力層のニューロンの状態が (1) 式で述べたとおり

$$x^O = f(\sum_j w_j^O x_j^H). \tag{5}$$

となります。この操作は，中間層のニューロンの状態の重み付き線形和を求め，それがしきい値 T より大きいか小さいかで出力層のニューロンの状態を決めることを意味しています。

そこで，中間層のニューロンが x_1^H, x_2^H の 2 つしかない場合を考え，両ニューロンの状態を図 6 の座標平面で表します。このとき，(5) 式の計算は図 7(a) の座標平面上で

$$w_1^O x_1^H + w_2^O x_2^H = T, \tag{6}$$

という直線をひき，現在の中間層のニューロンの状態 x_1^H, x_2^H がこの直線のどちらにあるかによって x^O が 1 か 0 かを決定していることになります。

ところが，例えば「$x^O =$［x_1^H と x_2^H の排他的論理和（XOR)］」という演算では，図 7(b) のように，x^O が 1 になるような x_1^H と x_2^H の値の領域と，x^O が 0 になるような x_1^H と x_2^H の値の領域とを，直線で区切ることができません（このような演算を**線形非分離**といいます)。したがって，どのように学習しても中間層と出力層の 2 層では［x_1^H と x_2^H の排他的論理和］という演算を得ることはできません[2]。パーセプトロンでは入力層と中間層との間の結合重みは固定されていますから，入力層と中間層との間で行う演算の設定によっては，どう学習しても求めるパターン認識ができない，ということが起こります。

これは，パーセプトロンの限界の 1 つです。これを指摘した Minsky と Papert は，さらに入力層と中間層の間の構成に現実的な制限を課すと表現できない問題がたくさんあることを述べ，中間層と出力層の間の学習だけでは，大したことは学習できないことを示しました。その結果，ニューラルネットワークの研究は一気に下火になってしまったのです。この限界を超えるには，中間層と出力層間だけでなく，ネットワークの層間の結合全体を学習する方法を考える必要があります。それを実現したのが，次章で述べる誤差逆伝播法です。

[2]「パーセプトロン学習が収束しない場合」とは，所望の入出力例が線形非分離であるときです。δ-ルールでは，この場合，入出力例との誤差の 2 乗が最小である結合重みに収束します。

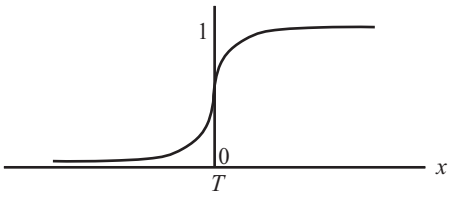

図 5: シグモイド関数

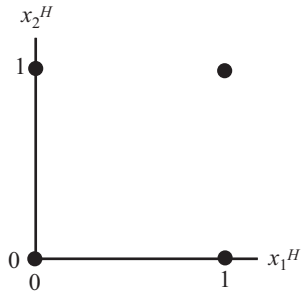

図 6: ニューロンの状態。(x_1^H, x_2^H) の組は図の 4 つの●のいずれかにある。

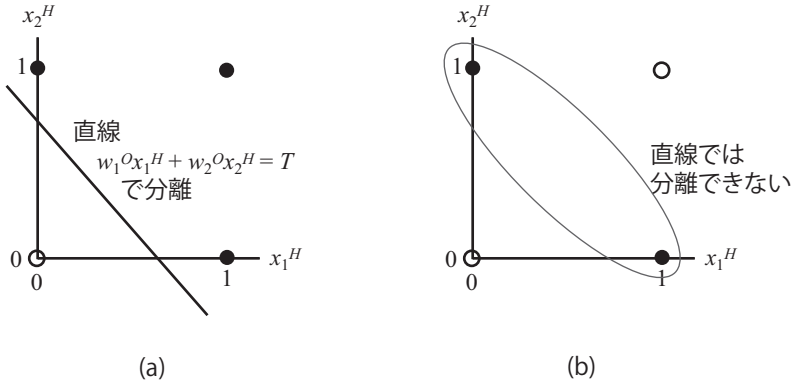

図 7: 線形分離性。(a) ●と○は線形分離。(b) 排他的論理和。●と○は線形非分離。

誤差逆伝播法

前節のパーセプトロン学習や δ-ルールにおいては，「所望の入出力例」は入力層と出力層の状態としてしか与えられていません。ですから，学習によって最小化する「入力例に対する，所望の出力と現実の出力との誤差」は，出力層についてしか得られません。パーセプトロン学習や δ-ルールにおいては，この誤差を使って出力層とその 1 つ手前の層（全体で n 層あるとすれば，第 $(n-1)$ 層）との結合重みを調整しました。

この方法を使ってさらにもう 1 つ手前の層（第 $(n-2)$ 層）との結合重みを調整するには，第 $(n-1)$ 層の各ニューロンでの誤差を知る必要があります。そのためには，出力層の 1 つのニューロンでの誤差を第 $(n-1)$ 層の各ニューロンに割り振らなければなりません。この割り振りを，次の考え方で行います。

1. 第 $(n-1)$ 層の各ニューロンのうち，出力層のこのニューロンとの結合重みが大きいニューロンにより多く誤差を割り振るほうが，割り振られた誤差が修正されたとき，その修正の出力層での効果が大きくなって好都合です。

2. 第 $(n-1)$ 層の各ニューロンの状態は，第 $(n-2)$ 層から送られてくる刺激の総和にシグモイド関数のような非線形関数を適用して決まります。このとき，刺激の総和の変化に対するニューロンの状態の変化率が大きい，つまり「感度のよい」ニューロンにより多く誤差を割り振るほうが，やはり割り振られた誤差の修正の効果が大きくなって好都合です。

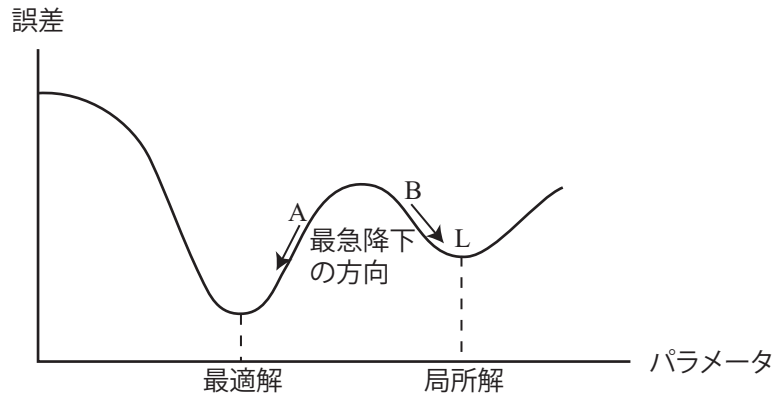

図 8: 最急降下と局所解

この考え方の詳細は，付録プリントに掲載します。以上の方法で３層以上の階層型ニューラルネットワークの学習を行う方法を，**誤差逆伝播法 (error back propagation method, EBP)** といいます。この名前は，出力層での所望の出力と現実の出力との差をもとに，手前の層に誤差を順に伝えていくことから来ています。

さて，ここまでの説明では誤差逆伝播法を「考え方」1., 2. から導きましたが，実はこの結合重みの修正法は，最終層での誤差の２乗和がその時点でもっとも多く減る方向に結合重みを修正することが証明できます。このことを，この修正法が**最急降下 (steepest descent)** になっているといいます。

結合重みのある組合せに対して，誤差２乗和は１通りに定まります。したがって，各結合重みを座標軸とする多次元空間の各点に，誤差２乗和の値が決まっているような「場」を考えることができます。結合重みの修正は，誤差２乗和がもっとも小さい点をめざして，多次元空間の中のある経路をたどってゆくことに相当します。最急降下とは，現在いる点から誤差２乗和がもっとも大きく減少する方向へ進むことを意味しています。

最急降下の方向は，f が**現時点で**もっとも減少する方向を示しているだけで，**最小値に向かう方向を示しているのではない**ことに注意してください。このため，最急降下法では，「f の値が，周囲よりは小さいが，遠く離れた場所の値よりは大きい」という**局所解**にしか到達できないことがあります。これを解決するために，最急降下するだけでなく「時々降下しない方向へも進む」などの方法が研究されています。

数式を用いた説明と，最急降下であることの証明は付録プリントに掲載しています。

サポートベクタマシンとカーネル法

サポートベクタマシンは，空間中に配置された点の２つの集合を最適に分離する境界を，その集合に属する点の分布を表す確率分布モデルを考えることなく求める方法です。その基本的アイデアは大変簡単で，「境界を，それぞれの集合でもっとも境界に近い点のどちらからも，もっとも遠くなるように置く」というものです。この簡単な考え方はかなり古くからあるものですが，近年ふたたび脚光を浴びました。それは，空間をさらに高次元の空間に変換するのと同等の操作を行なう「カーネル法」という方法を導入することによって，線形でない「曲がった」境界を求めることができるようになり，より複雑

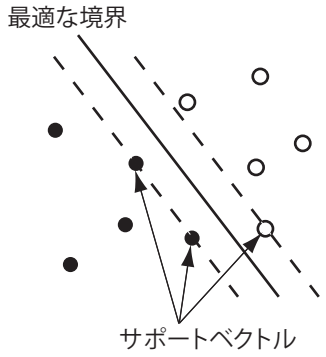

図 9: サポートベクタマシンによる最適な境界

な認識問題に対応できるようになったためです。

基本的なサポートベクタマシン

まず最初に，前節のニューラルネットワークについての説明で述べた「線形分離可能」な問題について考えます（図 7)。この問題について，それぞれの集合（○と●）を「最適に」分離する超平面を求めます。ここでいう「最適な」分離超平面とは，それぞれの集合に現在存在する点を完全に分離するだけでなく，それぞれの集合に存在するであろう「未知」の点をも分離することができる，という意味です。しかし，「次元のわな」として述べたように，通常のパターン認識問題では，空間の次元は既知の点の数よりもはるかに大きく，既知の点は空間中に非常に疎にしか分布していないので，確率分布の推定は実際は非常に難しいことになります。

そこで，確率分布の推定を必要としない，別の簡単な方法を考えます。この方法では，「最適」な境界を，「それぞれの集合のどちらからももっとも離れている境界」と考えます。言い換えると，この境界はそれぞれの集合の「ちょうど中間」を通ります。それぞれの集合の分布をあらわす確率分布は不明ですが，このような境界は，どちらの集合からももっとも離れているのですから，それぞれの集合の未知の点ももっともうまく分離できると期待されます。それぞれの集合に属する点のうち，この境界にもっとも近い点を**サポートベクトル**といいます。

このような境界は，それぞれの集合の凸閉包（集合に属するすべての点を囲む最小の凸図形）を結ぶ最短の線分の中点を通り，その線分に垂直な超平面となります。

$\boldsymbol{x}$ を空間中のある点（ベクトル）とするとき，境界超平面は，下の式で表される超平面のひとつです。

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0. \tag{7}$$

ここで，$\boldsymbol{w}$ は重みベクトル，b はバイアス項とよばれます。集合に属するあるベクトル $\boldsymbol{x}_i$ と境界超平面の距離は**マージン**とよばれ，つぎのように表されます。

$$\frac{|\boldsymbol{w}^T\boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|}. \tag{8}$$

(7) 式で表される超平面は，$\boldsymbol{w}$ と b に共通の定数をかけても同じものになります。そこで，次のような制約を導入します。

$$\min_i |\boldsymbol{w}^T\boldsymbol{x}_i + b| = 1. \tag{9}$$

最適な境界は, (8) 式の最小値を最大にするものです。(9) 式の制約により, この最大化は $1/\|\boldsymbol{w}\|^2 = 1/\boldsymbol{w}^T\boldsymbol{w}$ の最大化に帰着されます。すなわち，この最適化は

$$\begin{aligned} &minimize \quad \boldsymbol{w}^T\boldsymbol{w} \\ &subject\ to \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \end{aligned} \tag{10}$$

というものになります。ここで y_i は $\boldsymbol{x}_i$ が一方の集合に属するとき 1，もう一方の集合に属するとき -1 とします。もしも，この境界が各集合の点を正確に分離するならば，つねに $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 0$ となります。

このような条件付き最適化は，Lagrange の未定乗数法で解くことができます。その詳細は，付録プリントを見てください。

カーネル法

上のような議論は，線形分離可能な問題のみに適用されます。２つの集合が線形分離でない場合，２つの集合を完全に分離する超平面は存在しません。**カーネル法**は，こんな問題に対して，完全に非線形な「曲がった」境界を見つける方法です。

カーネル法の基本的アイデアは，元の空間自身をより次元の高い空間に変形することです。例えば，前回あげた線形非分離問題をみてみましょう（図 10(a)）。この２次元空間を，図 10(b) のような３次元空間に変形すれば，●の点と○の点は線形分離となります。

高次元空間への変換を Φ で表すことにします。変換された空間では，距離が定義されていないと，２つの集合を「最適に」分離する超平面を見つけることができません。また変換後の空間での距離は，元の空間での距離と関係がある（つまり，元の空間で「遠く離れた」２点は，変換後の空間でも「遠く離れている」）ほうが好都合です。このような条件を満たす変換を定義するために，点 $\boldsymbol{x}$ と点 $\boldsymbol{x}'$ の組に対して**カーネル関数** $K(\boldsymbol{x}, \boldsymbol{x}')$ を考えます。カーネル関数は，

$$K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}'). \tag{11}$$

という関係を満たすものです。この式は，カーネル関数が，変換 Φ で変換された高次元空間で測られた距離に相当するものであることを意味しています。境界を求めるのに必要な計算はすべて $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ を用いて可能で，変換後の空間や変換 Φ が実際にどういうものかは知る必要がありません。このことについては，付録プリントを見てください。

経験リスクと期待リスク

経験リスクとは，分離したい集合内の既知の点のうち，誤った分類をされる点の割合をさします。「最適」な境界を求めるのに，本当に最小にしなければならないのはこの経験リスクではありません。本当に最小化しなければならないのは，各集合の（既知のものも未知のものも含めた）全ての点のうち，誤った分類をされる点の割合です。こちらの割合のほうは**期待リスク**といいます。

線形分離可能な問題の場合，既知の点を線形分離可能なわけですから，経験リスクをゼロにする分離超平面が存在する，ということになります。「マージンが最大になる境界を見つける」というサポートベクタマシンの考え方は，経験リスクをゼロにする分離超平面のなかから，期待リスクを最小にするものを選ぶ，ということに相当します。

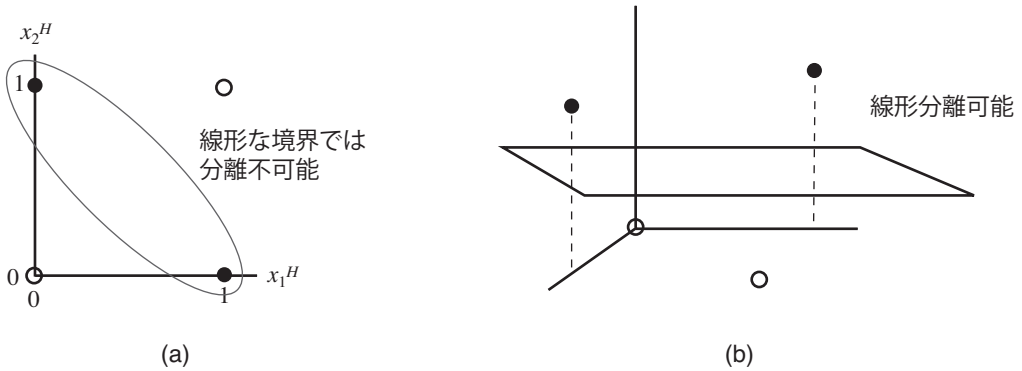

図 10: 高次元空間への変換