階層型のネットワーク

浅川伸一 <asakawa@twcu.ac.jp>

1　結合係数の更新式 —ヘッブ則とデルタ則—

ニューラルネットワークモデルでは、シナプスの結合係数を変更することをニューラルネットワークの学習と呼びます。学習時における結合係数の変化を記述したルールを学習則 (learning rule) と呼びます。ここでは 2 つのニューラルネットワークの学習規則を紹介します。これら 2 つ学習規則はいずれも歴史的な価値があり、かつ、現在でもそまま形式で用いられたり、特定の目的に合致するように拡張、変形した形式で用いられている基本的な学習則です。

1.1　ヘッブ則

Hebb の原文では以下のような記述です。

> When an axon of cell A is near enough to excite cell B and repeatedly or consistently takes part in firing it, some growth process or metabolic changes takes place in one or both cells such that A's efficiency, as one of the cell B, is increased.
>
> —Hebb,D.O.,(1949), The Organization of Behavior—

「同時に発火したニューロン間のシナプス結合は強められる」ことを主張しているのがヘッブ則 (Hebbian rule) です (Hebb, 1949)。ヘッブの学習則とも表記されます。ヘッブ則は以下のように定式化できます。 ニューロンの発火状態を 1、休止状態を 0 と表現することにし、ニューロン y_i から ニューロン x_j へのシナプス結合係数を w_{ij} とします。このときヘッブの学習則は、シナプス結合係数の変化 Δw_{ij} として表現され、

$$\Delta w_{ij} = \lambda x_j y_i, \tag{1}$$

と書くことができます。ここで $\lambda(\geq 0)$ を学習定数といいます。x_j と y_i は 1 と 0 の 2 とおりの状態にしかならないため、可能な組み合わせは 4 通りになります。このうち $\Delta w_{ij} \neq 0$ となる組み合わせは、$(x_j = 1)$ かつ $(y_i = 1)$

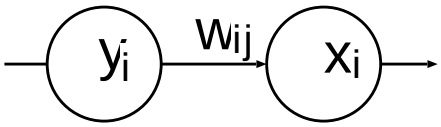

Figure 1: ヘッブの学習則

		x_j	
		1	0
y_i	1	λ	0
	0	0	0

Table 1: ヘッブの学習則

の場合だけです。

y が 1 または 0 の値しか取らないことに注意すれば、(1) は

$$\Delta \boldsymbol{w} = \begin{cases} \lambda \boldsymbol{x} & \text{if } y \text{ is fire,} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

と書くことができます。すわなち結合係数は、入力ニューロンと出力ニューロンの同時発火頻度に比例して、入力ニューロン x の λ 倍だけ増大することを意味します。ある時刻 t で、入力 x が y を発火させたとき、次回 $t+1$ に同じ入力が与えられるとより強く発火させる効果を持つようになると考えることもできます。逆に言えば同じ刺激を入力し続けると $|\boldsymbol{w}|$ が限りなく大きな値になってしまうことも意味しています。このようにヘッブ則には、学習回数を重ねても一定の値に収束しないという特徴を持っています。

ヘッブ則の変形はさまざまに考えられていて、例えば同時に発火しなかった時にシナプス結合を弱めるアンチヘッブ則、減衰項を加える方法、入力と出力の差 (の 2 乗) を利用する方法、などが考案されています。ヘッブ則の変形 (の一部) は、自己組織化の説明のところでも説明します。

1.2 デルタ則

デルタ則は LMS 則、あるいは Widrow–Hoff の学習則などと呼ばれることもあります。

デルタ則の説明のために、n 個の入力層ユニットと 1 個の出力ユニット y からなる単純な 2 層のネットワークを考えてみましょう。出力ユニット y の活動は、入力ユニットからの信号 $x_i (i = 1 \ldots n)$ の重みつき荷重和 $\sum w_i x_i$ で定まるとします。望ましい出力 (教師信号) を y^* とすれば、デルタ則は、望ましい出力 (教師信号) y^* と実際の出力 y の差 (デルタ) に入力信号 x を掛

けた形

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta\, \delta_t \boldsymbol{x_t} = \boldsymbol{w_t} + \eta\, (y_t^* - y_t)\, \boldsymbol{x}_t, \tag{3}$$

で表現されます。ここで η は学習係数と呼ばれる定数です。

デルタ則とは δ^2 を最小にする規準を導入した学習則です。すなわち、入力信号と望ましい出力とが与えられたとき

$$f(\boldsymbol{w}) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} \delta^2 = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} (y_i^* - y_i)^2 \tag{4}$$

という関数の極限を考え、この関数 $f(\boldsymbol{w})$ を最小化することを考えます。(4) は $\boldsymbol{w}$ の 2 次関数とみなすことができるので、$f \geq 0$ であり、$f = 0$ となるのは、すべての y_i^* に対して $y_i^* - y_i = 0$ のとき、すなわち完全に学習が成立したときだけです。そこで、任意の初期値 $\boldsymbol{w}_0$ から出発して漸化式 $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \Delta \boldsymbol{w_t}$ に従って逐次結合係数を更新して行くことを考えるのです。$f(\boldsymbol{w})$ は入力データと結合係数 w とで決まるので、w を微小に変化させたとき (微分係数) の $f(\boldsymbol{w})$ 変化量の逆方向 (f を w で微分しマイナスをかける) に逐次 w を変化させていくことで極小解に達する (図 2) ことが予想できます。これは、勾配降下法と呼ばれる最適化問題の解法の一つです。

より厳密に (3) 式が $f(\boldsymbol{w})$ の最小値に確率収束することを証明することができます。δ は確率変数であると考えれば、$E\left[\delta^2\right]$ すなわち誤差の 2 乗の期待値を最小にするような $\boldsymbol{w}$ を求める問題となります。実際 $f(w)$ を $\boldsymbol{w}$ で微分すると

$$\begin{aligned}
\frac{d}{d\boldsymbol{w}} f(\boldsymbol{w}) &= \frac{d}{d\boldsymbol{w}} \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} \delta_i^2 \\
&= \lim_{n\to\infty} \frac{1}{n} \frac{d}{d\boldsymbol{w}} \sum_{i=1}^{\infty} (y_i^* - y_i)^2 \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} 2\,(y_i^* - y_i)\, \frac{d}{d\boldsymbol{w}} (-\boldsymbol{y}) \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} 2\,(y_i^* - y_i)\, (-\boldsymbol{x}) \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} 2\delta_i\, (-\boldsymbol{x}) \\
&= -2E\left[\delta \boldsymbol{x}\right]
\end{aligned}$$

となってこの関係を用いれば、漸化式

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \delta_t \boldsymbol{x}_t = \boldsymbol{w}_t + (y_t^* - y_t)\, \boldsymbol{x}_t \tag{5}$$

が $\boldsymbol{w}$ の最小値に確率収束します。ところで $y = \boldsymbol{w}'\boldsymbol{x}$ ですから、この関係を (4) に代入すれば最小二乗法[1] の導出と同じ論旨の展開の仕方です。

[1] もちろん最小二乗法はガウス (Gauss) の発明による。心理学を学んだ者であれば誰でも、学

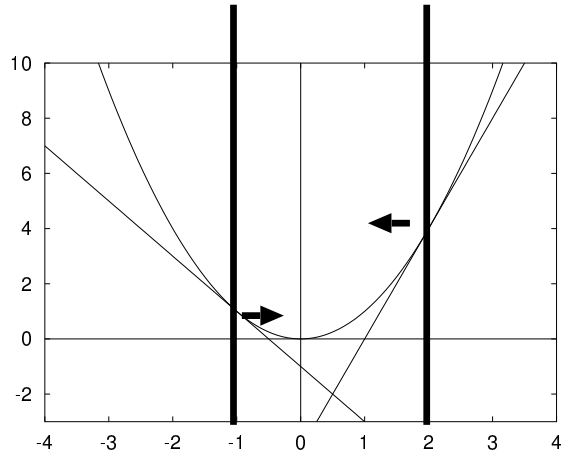

Figure 2: 最急降下法 (例えば伊理 (1981)) は接線の傾きと反対方向に向かって進めば極小点に達するというアイデアに基づいている。図にあるとおり接線の傾きが正であれば負の方向に、反対に接線の傾きが負であれば正の方向に少しだけ進めば極小点に近付くことができる。この操作を繰り返せばやがて極小点に達することができる。移動量は学習係数に比例する。学習係数が小さいと極小点に達するまでに時間がかかるが、学習係数が大きいと極小点を飛び越してしまい収束しない。統計的最適化理論、数値計算などの分野との関連から、2 次の微分係数の逆数を利用して移動量を決定する方法も提案されている

デルタ則の欠点は 2 層の結合係数しか扱えないことです。デルタ則を多層の回路での学習に適用できるようにしたものが 2.3 節で紹介する一般化デルタルール、あるいは、バックプロパゲーション法とよばれる学習則です。

2　階層型のネットワーク

パーセプトロンに代表されるフィードフォワード型の結合を持つ階層型のネットワークは、パターン認識 (pattern recognition)、情報圧縮 (data compression)、運動制御 (motion control)、雑音除去 (noise reduction)、および時系列予測 (time series prediction) などへの理論的、または応用的研究が試みられています。ここでは、パーセプトロンの学習について、幾何学的表現を用いて詳しく解説し、線形分離可能な問題について紹介します。続いてバックプロパゲーション法を導入し、階層型のネットワークについての幾つかの話題を解説します。

部の実験演習で必ずお世話になる考え方であり、データの当てはめ (回帰) や相関係数の計算など、最小二乗法を理解していなければ単位をもらえない。

2.1 階層型ネットワークの行列表現

以降では表記を簡単にするために線形数学の表記、すなわちベクトルと行列による表記方法を導入します。n 個の入力信号の組 $(x_1, x_2, \cdots, x_n)$ をまとめて $\boldsymbol{x}$ のようにボールド体で表すことにします。本章では一貫してベクトル表記には小文字のボールド体を、行列には大文字のボールド体を用いることにする。例えば n 個の入力信号の組 $(x_1, x_2, \cdots, x_n) = \boldsymbol{x}$ に対して、同数の結合係数 $(w_1, w_2, \cdots, w_n) = \boldsymbol{w}$ が存在するので、加算記号 $\sum$ を使って表現していた任意のユニットへの全入力 $\mu = \sum w_i x_i$ はベクトルの内積を用いて $\mu = (\boldsymbol{w} \cdot \boldsymbol{x})$ と表現されます。なお、横一行のベクトルを行ベクトル、縦一列のベクトルを列ベクトルと呼ぶことがあります。本章では行ベクトルと混乱しないように、必要に応じて列ベクトルを表現する際には $(x_1, x_2, \cdots, x_n)^T = \boldsymbol{x}$ とベクトルの肩に T を使って表現することもあります。そして、これらをベクトル表現や行列表現で表せば、表記も簡単になり、行列の演算法則を活用することもできます。そのため、ニューラルネットワークに関する文献でも行列表現が用いられることが多いのです。

図 3 のような単純な 2 層の回路を例に説明する。図 3 には、 3 つの入

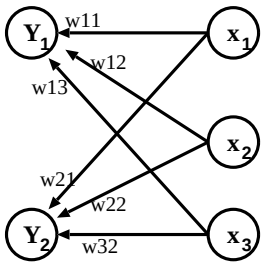

Figure 3: ネットワークの行列表現

力ユニットと 2 つの出力ユニットの活性値（ニューロンの膜電位に相当する）x_1, x_2, x_3 と y_1, y_2 および入力ユニットと出力ユニットの結合係数を表す $w_{11}, w_{12}, \cdots, w_{32}$ が示されています。これらの記号をベクトル $\boldsymbol{x}$, $\boldsymbol{y}$ と行列 $\boldsymbol{W}$ を使って表すと $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ となります。図 3 の場合、ベクトルと行列の各要素を書き下せば、

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{6}$$

のようになります。

行列の積は、左側の行列の i 行目の各要素と右側の行列（ベクトルは 1 列の行列でもある）の i 列目の各要素とを掛け合わせて合計することなので、

以下のような、加算記号を用いた表記と同じことです。

$$\begin{array}{rcl} y_1 & = & w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \quad = \quad \sum_i w_{1i}x_i \\ y_2 & = & w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \quad = \quad \sum_i w_{2i}x_i \end{array} \tag{7}$$

これを、m 個の入力ユニットと n 個の出力ユニットの場合に一般化すれば、

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{8}$$

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \tag{9}$$

と表現できます。しきい値の扱いについては、常に 1 を出力する仮想的なユニット $x_0 = 1$ を考えて W に組み込むことがあります。

実際の出力は $\boldsymbol{y}$ の各要素に対して

$$f(y) = \frac{1}{1 + e^{-y}} \tag{10}$$

のような非線型変換を施すことが多いです。

階層型のネットワークにとっては、(10) の非線型変換が本質的な役割を果たします。なぜならば、こうした非線形変換がなされない場合には、ネットワークの構造が何層になったとしても、この単純なシナプス結合係数を表す行列を $\boldsymbol{W}_i$ ($i = 1, \cdots, p$) としたとき、$\boldsymbol{W} = \prod_{i=1}^{p} \boldsymbol{W}_i$ と置くことによって本質的には 1 層のネットワークと等価になるからです。

$$\boldsymbol{y} = \boldsymbol{W}_p \boldsymbol{W}_{p-1} \cdots \boldsymbol{W}_1 \boldsymbol{y} = \left(\prod_{i=1}^{p} \boldsymbol{W}_i \right) \boldsymbol{y} \tag{11}$$

2.2 パーセプトロン

パーセプトロン perceptron とはローゼンブラット (Rosenblatt, 1958) によって提案された図 4 のような 3 層の階層型ネットワークモデルです。パーセプトロンはマッカロック・ピッツの形式ニューロンを用いて学習則にヘッブ則 (1) 式を使ったモデルで、単純な認識能力を獲得することができます。パーセプトロンは 3 層の階層型ネットワークでそれぞれ、S(sensory layer), A(associative layer), R(response layer) と呼ばれる層からなっています。$S \to A \to R$ のうち パーセプトロンの本質的な部分は $A \to R$ の間の学習にあります。最下層の入力層は外界からの刺激入力を表現しています。中間層では入力情報の変換が行われ、最上位層である出力層で認識に到ることになります。

ごく簡単にいえば、入力パターンに現われる P^+ と P^- という 2 つのパターンを識別する問題を考えたとき、パーセプトロンとは P^+ が入力された

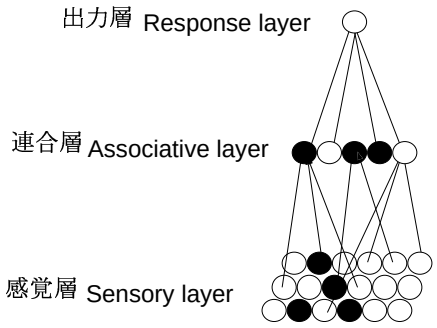

Figure 4: パーセプトロンの模式図

とき 1, P^- のとき 0 を出力する機械であるということができます。あるいは、出力ユニットが 1 つしかない場合を考えれば、パーセプトロンは入力層に現われるパターンを 1 と 0 とに 2 分する機械であるということができます。出力層の i 番目のユニットへの入力 (膜電位の変化) u_i は

$$u_i = \sum_j w_{ij} x_j - h_i = \boldsymbol{w}_i \cdot \boldsymbol{x}_i - h_i, \tag{12}$$

と表現されます。ここで中間層の j 番目のユニットの出力 y_i とこのユニットとの結合係数を w_{ij}、しきい値を h_i としました。このユニットの出力 y_i(活動電位、スパイク) は、

$$y_i = \lceil u_i \rceil \qquad \begin{cases} 1 & \text{if } u_i \geq 0, \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

と表さます。すなわち、活性値 u_i は、式 (13) のような関数によって、1 または 0 の出力に変換されることを意味します。活性値が 0 より大きければ 1 を出力し、それ以外の場合は 0 となります。

2.2.1 パーセプトロンの学習

パーセプトロンの学習は連合層から反応層への結合係数の変化として表現されます。最初の入力がなされる感覚層から連合層への結合係数は一定で変化しません。すなわちパーセプトロンは 3 層のネットワークではあるけれど、学習を考える際には、連合層と出力層の間だけ考えれば良いわけです。

パーセプトロンが特定のパターンに対してだけ出力するようになるためには、学習によってネットワーク内部の結合係数を変化させる必要があります。その際、パーセプトロンには種々の入力パターンが与えられ、それぞれが検出すべきパターンであるか否かが「教師信号」として与えられます。具体的には、入力信号を 2 分する問題を学習する場合、検出すべきパターンの時に

教師信号として 1 が、それ以外のパターンの場合には、教師信号として 0 が与えらます。ここでいう教師信号とは、のデルタ則の説明での望ましい出力に相当します。

パーセプトロンの学習は中間層 (連合層) から出力層への結合荷重の変化として表現されます。入力層から中間層への結合荷重については考慮されないことに注意が必要です。パターン c に対する教師信号 (望ましい出力) を t_c と書くことにするとパーセプトロンの学習、すなわち結合係数の更新式は

$$\Delta w_{ji} = \eta \delta x_{c,j} = \eta \left(t_{c,j} - y_{c,j} \right) x_{c,j}. \tag{14}$$

と表すことができます。η は学習係数と呼ばれる定数です。パーセプトロンの出力と教師信号と差分 $\delta = t_{c,j} - y_{c,j}$ のことを誤差信号と呼びます。上式は w_{ji} の更新式として

$$w_{ji}(n+1) = w_{ji}(n) + \eta \Delta w_{ji}(n) = w_{ji}(n) + \eta \left(t_{c,j} - y_{c,j} \right) x_{c,j}. \tag{15}$$

のような漸化式として表現されることがあります。ベクトル表現すれば

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta \left(\boldsymbol{t}_c - \boldsymbol{y}_c \right) \boldsymbol{x}_c. \tag{16}$$

ここで、$\boldsymbol{w}(n)$ は n 回目の学習終了時点での結合係数を入力層のユニットだけ並べて $(w_1, w_2, \ldots, w_N)$ としたものです。$\boldsymbol{t}_c$ はパターン c に対応する教師信号を表し、$\boldsymbol{y}_c$ はパターン c に対するパーセプトロンの出力を表します。$\boldsymbol{x}_c$ は入力信号です。

入力ユニット数 2、出力ユニット数 1 の 2 層で構成される単純なネットワークを考えれば、このネットワークへの入力全体は 2 次元平面で表現できます。この平面のことを入力空間と呼ぶことがあります。入力ユニットが n 個あれ

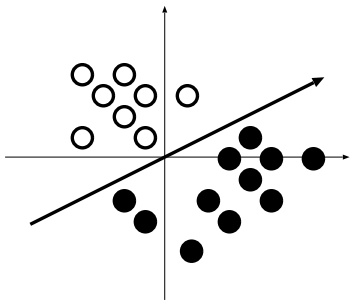

Figure 5: パーセプトロンの入力の幾何学的表現と判別直線

ば n 次元空間が入力空間になります。入力層のユニット数が 2 であれば平面になります。パーセプトロンにおける学習とはこの入力空間を 2 分割するような領域に分割する判別直線を見つけることであるということができます。n 次元の入力空間におけるパーセプトロンによるパターン分類とは、出力細胞

が 1 つの場合、n 次元の入力パターン空間を $\boldsymbol{wy} = \theta$ を満たす境界 ($n-1$ 次の超平面) によって 2 つの部分空間に分割することであると言えます。

いま、しきい値が 0、すなわち入力データを 2 群に分ける直線が原点を通る場合を考えることにしましょう。すなわち、図 5 で判別直線より上の白丸には 1 を、判別直線より下の領域にある黒丸には 0 を出力するような学習を考えるわけです。$w_1x_1 + w_2x_2 > 0$ であれば 1 を出力し $w_1x_1 + w_2x_2 \leq 0$ ならば 0 を出力するとは、ベクトル $(x1, x2)$ と ベクトル $(w1, w2)$ との内積の正負の判断をしているのと同義です。なぜならベクトルの内積とは 2 通りの表現があって

$$\boldsymbol{w} \cdot \boldsymbol{x} = w_1x_1 + w_2x_2 = |\boldsymbol{w}|\,|\boldsymbol{x}| \cos\theta \tag{17}$$

だからです。上式の最右辺の正負は 2 つのベクトルのなす角 $\cos\theta$ のみによって決まります。$\cos\theta$ が正となる範囲は 2 つのベクトルのなす角が $-\pi/2$ から $\pi/2$ までですから 図 6 のように斜線をつけた領域にある全てのベクトル

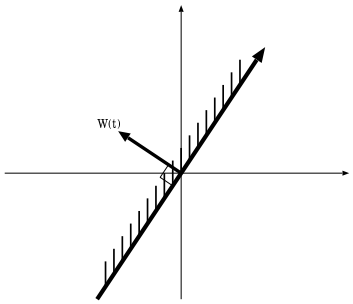

Figure 6: 2 群を判別するとき斜線の領域で正になるとは、法線ベクトルとの内積が正であることを意味する

は、2 群を判別する直線の法線ベクトル (2 つの結合係数を要素とするベクトル) との内積が正となります。入力信号と結合係数ベクトルのなす角 θ が $-\pi/2 < \theta < \pi/2$ の範囲のとき $\cos\theta > 0$ となり、そうでなければ負になるからです。

このパーセプトロンに学習すべきデータが一つ入って来たと仮定しましょう。図 7 は 1 と出力すべき ($y = 1$ すなわち法線ベクトルとの内積が正であるべき) データを、誤って 0 と出力してしまったという事態です。このとき、式 (16) 内の $(\boldsymbol{t} - \boldsymbol{y})$ は $1 - 0 = 1$ になるので結合荷重の更新式、すなわち法線ベクトルの更新式は

$$\boldsymbol{w}(n) + \eta\,(\boldsymbol{t}_c - \boldsymbol{y}_c)\,\boldsymbol{x}_c = \boldsymbol{w}(n) + \eta\boldsymbol{x}_c. \tag{18}$$

となってベクトルの足し算にります (図 8)。これによって判別直線が回転し (図 9)、今度は法線ベクトル $\boldsymbol{w}$ とデータ $\boldsymbol{x}$ とのなす角が 90 度以内になりま

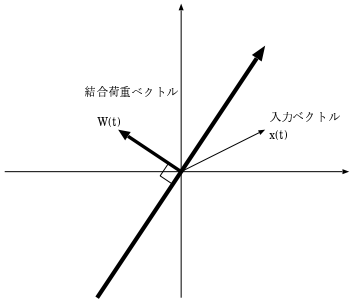

Figure 7: 学習すべきデータ

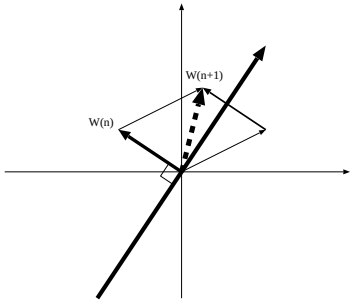

Figure 8: パーセプトロンの学習則による結合荷重ベクトル w の更新

す。0 と出力すべきデータを、誤って 1 と出力してしまった場合はベクトルの引き算になります。

最後に判別直線が原点を通らない場合、すなわち閾値が 0 ではない場合を説明します。(図 10)。この場合は、原点を通る直線を並行移動したことになるので、原点を通る判別直線では 1 と答えるべきの領域 (図 10 中の白マル) に入ってしまっている点を底アゲして 0 と答えるべき領域 (図 10 中の黒マル) の領域にするために行われると考えることができます。

パーセプトロンによるパターン分類とは、出力細胞が 1 つの場合、入力パターン空間 (n 個の入力があれば n 次元空間) を $\boldsymbol{wy} = \theta$ を満たす境界 ($n-1$ 次の超平面) によって 2 つの部分空間に分割することであると言える。逆に言えば、入力パターン空間が一本の判別直線によって分割できないようなパターンをパーセプトロンは学習することができないことを意味しています。このことを線形分離可能性 (linear separability) という。

以上をまとめると、出力層と入力層との間の結合係数を入力データ空間におけるベクトルと考えれば、パーセプトロンの出力は結合係数ベクトルと入力データとの内積が 0 より大きければ 1 を、小さければ 0 を出力する機械であるみなすことができます。また、パーセプトロンの学習は、入力信号ベ

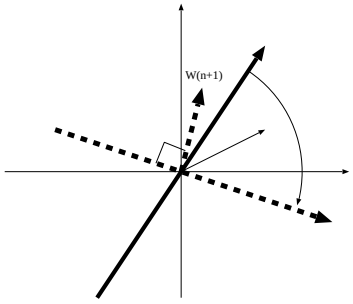

Figure 9: 判別直線の回転

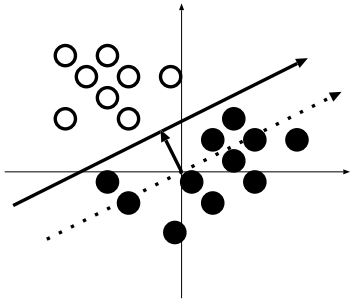

Figure 10: しきい値の意味

クトルと結合係数ベクトルの内積の大小によって結合係数ベクトルを回転させることだと言うことができます。このようにして訓練されたパーセプトロンでは、入力データ空間上で線形分離可能な問題ならば、必ず学習が可能であることが重要です。逆に言えば、線形分離不可能な問題でも、適切に次元を設定することで線形分離可能な空間に写像するような中間層表現を得ることができればパーセプトロンの持つ限界を越えることができるというアイデアに結び付きます。

ミンスキーとパパート (Minsky & Papert, 1988) はパーセプトロンのベクトル表示について悲観的な考え方を持っているようですが、ここでは理解のしやすさを優先してベクトル表示による説明を用いました。

パーセプトロンのような階層型のネットワークは、中間層のユニットを基底関数とみなすことで、関数近似法の一つと考えることができます。このことから、他の数学的な関数近似手法との関連[2]が指摘できます。一般にこれらの数学的手法は入力変数の次元数 n が増えれば、一定の近似誤差内で関数を近似するためには、必要なパラメータ数が n 乗のオーダーで増加することが知られており、このことは次元の呪い (curse of dimensionality) と呼ばれてい

[2]たとえばテーラー展開、フーリエ級数展開、スプライン関数など。

ます。多層パーセプトロンによる関数近似では、この次元の呪いを避けることができることが指摘されています。

2.2.2 パーセプトロンの情報処理能力

ミンスキーとパパート (Minsky & Papert, 1988) によるパーセプトロンの能力を示した定理を幾つか紹介しておきましょう。

パーセプトロンの収束定理: 解くべき問題が線形分離可能であるならば、パーセプトロンの学習は有限回で停止し、そのとき得られた結合係数はすべての入力パターンに対して正解を与えるものになっている。

パーセプトロンの循環定理: 解くべき問題が線形分離可能でないとき、パーセプトロンは一般に収束しないが w_{ji} の大きさは有界であり有限個の結合状態を循環する。

パーセプトロンの群不変定理: ある変換群のもとで幾何学的な性質を判断するパーセプトロンに、その変換群のもとで同値な w_{ji} を使うことで同値な値に対して同じ出力を得ることができる。

その他、パーセプトロンに代表される階層型のネットワークの能力についての研究が多くの研究者によって進めらてきました (概説は (上坂, 1993) など)。たとえば、

論理関数の計算に関する完全性: それぞれのユニットがしきいユニットである中間層のユニットを必要なだけたくさん使えば、三層のネットワークによって任意の 2 値論理関数を実現できる。

連続関数のシミュレータとしての完全性 中間層のユニットを必要なだけたくさんつかえば、ユニット間の結合 (およびジグモイド状関数のしきい値) を適切に設定することによって、任意の連続な関数 $f([0,1]^n \to (0,1)^m)$ を任意の精度で近似することができる (船橋賢一, 2000)。

いかに複雑な関数であっても絶対可積分であれば 2 層のパーセプトロンで実現できることが証明されています。

などがあります。

2.2.3 パーセプトロンの限界

パーセプトロンの学習 (2.2.1 節) でみたとおり、パーセプトロンは線形分離可能な問題でなければ解くことができません。パーセプトロンでは、入力層から中間層への結合についてはランダムであり学習が定義されていないからです。ネットワークの出力と教師信号と間の誤差を中間層以下のより下層

へ伝播させることができれば線形分離不可能な問題を正しく学習できかも知れない訳です (バックプロパゲーションの節 2.3 を参照)。

ミンスキーとパパート (Minsky & Papert, 1988) は図形の連結性を識別する簡単な問題も有限直径パーセプトロン (diameter–limited perceptron) では解けないことも例示しています。たとえば、図 11 のような問題です。x_{00} の

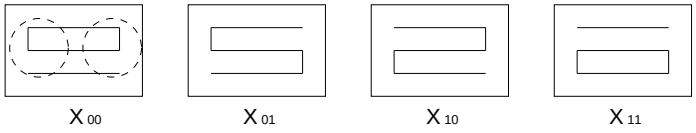

Figure 11: 図形の連結性を認識する問題

2 つの丸 (点線) で描かれている場所を見ている (受容野の位置と考えればよい) 2 つのユニット (A_1, A_2) があったとします。このパーセプトロンが連結図形 (X_{10}, X_{01}) と非連結図形 (X_{00}, X_{11}) とを $W_1A_1 + W_2A_2 + \theta > 0$ によって識別できないことが分ります。

ミンスキーとパパートは、この他にも結合係数が非常に大きくなってしまう問題、実質的な問題を解くためには膨大な数の中間層のユニット数が必要となる問題なども指摘しています。

2.3 バックプロパゲーション (誤差逆伝播法)

2.3.1 XOR 問題、線形分離不可能な問題

パーセプトロンでは絶対に解けない問題に排他的論理和 (XOR) 問題があります。排他的論理和とは、2 つの入力のうちいずれか一方のみが 1 のとき 1 を出力する問題です。図 12 左を見るとわかるとおり、XOR 問題は一本の判別直線で白マルと黒マルを分離できない、すなわち線形分離不可能な問題です。

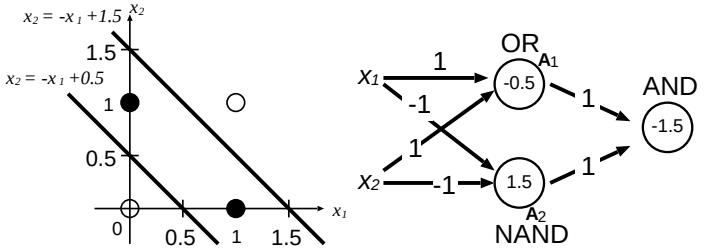

Figure 12: XOR 問題の幾何学的表現と XOR 問題を解くためのネットワーク

図 12 右は図 12 左の幾何学表現を対応するネットワークとして表現したも

のです。一番左が入力層、一番右が出力層、中間の 2 つが中間層です。ユニットの中に書かれた数値は各ユニットのしきい値を示しています。中間層の 2 つのユニットのうち上の OR と書かれたユニットは、$x_1 + x_2 - 0.5 > 0$ のとき発火します。この式を書き換えると、$x_2 > -x_1 + 0.5$ となるので図 12 左の下斜線より上の領域に対応します。一方、中間層の下の NAND(not and) と書かれたユニットは、$-x_1 - x_2 + 1.5 > 0$ のとき発火しますから、移項して $x_2 < -x_1 - 1.5$ とすれば、図 12 左の上斜線より下の領域に対応していることが分かります。さらに、AND と書かれた出力ユニットは、2 つの中間層ユニットからの情報の論理積 (AND) を計算するユニットになっています。そこで、2 つの中間層ユニットの両方が発火する場合だけ、出力ユニットも発火し、1 を出力する。これは、図 12 左では、「下の斜線より上の領域」でかつ「上の斜線より下の領域」に対応します。すなわち、図中の黒丸の領域だけが分離されることになります。このような 2 本の直線は図中にいくらでも引けることから、XOR 問題の解も無数に存在することが分かります。

入力層		中間層		出力
x_1	x_2	a_1	a_2	r
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Table 2: 図 12 に対応する XOR 問題の真偽表

図 12 左にあるとおり、中間層のユニット 1 個は 1 つの線形判別関数に相当すると考えられます。中間層から出力層への結合では各出力の論理積 AND を計算していることに相当します。n 個の中間層を用意すれば原理的には $\frac{n^2+n+2}{2}$ 個のカテゴリー分類が可能です。パーセプトロンが XOR 問題を解くことができない理由は入力層から中間層にいたる結合係数を変更する手段がないことなのです。

2.3.2 誤差逆伝播法 (一般化デルタルール)

XOR 問題でも見たように、パーセプトロンの問題点は学習が出力層と中間層の間だけで行われ、下の層に伝播しないことです。この点を改良したのがバックプロパゲーション (一般化デルタルール) と呼ばれる学習則です。

m 層のネットワークを考え、k 層の i 番目のユニットへの総入力を x_i^k, このユニットの出力を y_i^k, $k-1$ 層の i 番目のユニットから k 層の j 番目のユ

ニットへの結合係数を $w_{ij}^{k-1,k}$ と表記する。各ユニットの出力は

$$y_i^k = f\left(x_i^k\right) = \frac{1}{1+e^{-x_i^k}} \tag{19}$$

$$x_j^k = \sum_i w_{ij}^{k-1,k} y_i^{k-1}, \tag{20}$$

で定義されているものとします。あるデータ $\boldsymbol{x}$ と教師信号 $\boldsymbol{t}$ が与えられたとき教師信号と出力との 2 乗誤差を

$$E = \frac{1}{2}\sum_j \left(y_j^m - t_j\right)^2 = \frac{1}{2}\sum_j \delta^2 \tag{21}$$

と表記します。総和記号 $\sum$ の前の 1/2 は微分したときに式を簡単にする程度の意味しかないので本質的ではありません。この誤差関数 E は、教師信号と出力との差の 2 乗に比例して大きくなります。そこで、E が減少する方向に $\boldsymbol{w}$ の値を逐次更新することがバックプロパゲーション法の基本的な発想なのです。(21) 式の誤差 E は各 y_j^m の 2 次関数とみなすことができるので $E \geq 0$ であり、$E = 0$ となるのは、すべての y_j^m に対して $y_j^m - t_j = 0$ のとき、すなわち完全に学習が成立したときだけです。

$$\Delta w_{ij}^{k-1,k} = -\epsilon \frac{\partial E}{\partial w_{ij}^{k-1,k}} \tag{22}$$

(22) 式で E は y_j^m の関数ですが、さらに y_j^m は x_j^m の関数であり、さらにさらに x_j^m は $w_{ij}^{m-1,m}$ の関数ですから合成関数の微分公式により

$$\Delta w_{ij}^{m-1,m} = -\epsilon \frac{\partial E}{\partial w_{ij}^{m-1,m}} \tag{23}$$

$$= -\epsilon \frac{\partial E}{\partial y_j^m}\frac{\partial y_j^m}{\partial x_j^m}\frac{\partial x_j^m}{\partial w_{ij}^{m-1,m}} \tag{24}$$

$$= -\epsilon \left(y_j^m - t_j\right) y_j^m \left(1 - y_j^m\right) y_i^{m-1} \tag{25}$$

$$= -\epsilon \delta_j^m y_i^{m-1}. \tag{26}$$

となります (ただし $\delta_j^m = \left(y_j^m - t_j\right) y_j^m \left(1 - y_j^m\right)$)。もし仮に (19) 式で与えられている出力関数が線形関数 $y(x) = x$ であれば、(26) 式は $\Delta w_{ij} = \left(t_j - y_j\right) y_i^{m-1}$ となってパーセプトロンの学習式と一致します。

次に中間層以下第 n 層 $(n \neq m)$ のユニット y_j^n の結合係数の更新には、

$$\frac{\partial E}{\partial y_j^n} = \sum_k \frac{\partial E}{\partial y_k^{n+1}}\frac{\partial y_k^{n+1}}{\partial x_k^{n+1}}\frac{\partial x_k^{n+1}}{\partial y_j^n} \tag{27}$$

$$= \sum_k \frac{\partial E}{\partial y_k^{n+1}}\frac{\partial y_k^{n+1}}{\partial x_k^{n+1}}\frac{\partial}{\partial y_j^n}\sum_i w_{ik}^{n,n+1} y_i^n \tag{28}$$

$$= \sum_k \delta_k^{n+1} w_{kj}^{n,n+1}, \tag{29}$$

を誤差信号 δ として再帰的に計算する。以上をまとめると、結合係数の修正量 $w_{ij}^{k-1,k}$ は

$$\Delta w_{ij}^{k-1,k} = -\epsilon \delta_j^k y_i^{k-1}. \tag{30}$$

$$\text{where } \delta_j^k = \begin{cases} \left(y_j^k - t_j\right) y_j^k \left(1 - y_j^k\right), & \text{if } k = m \\ \left(\sum_l \delta_l^{k+1} w_{jl}^{k,k+1}\right) y_j^k \left(1 - y_j^k\right), & \text{otherwise} \end{cases} \tag{31}$$

となります。式 (31) を見ると誤差の計算がデータ処理とちょうど逆の流れで入力層まで伝播するようになっています。これが誤差逆伝播法と呼ばれる所以です。

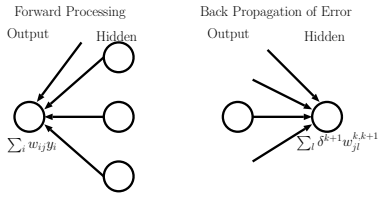

Figure 13: 誤差逆伝播の模式図

2.4 汎化能力と過学習

学習すべきデータ集合の中からいくつかのサンプルを選び訓練課題セットとしてニューラルネットワークに学習させるとします。このとき、学習させた学習課題セット以外のデータをテスト課題として選び、このテスト課題の成績を調べることで学習したルールの一般化能力を測定することができます。

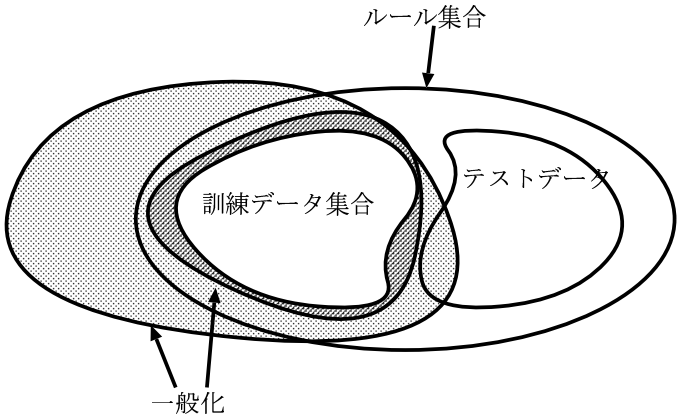

Figure 14: 汎化能力と過学習

学習した内容がルールの適用範囲と完全に重なることが理想ですが、図 14 で示したように一般化にはさまざまな可能性が考えられます。例えば入出力とも 0, 1 のデータで、入力層が N 個、出力層が 1 個である場合を考えてみましょう。入力パターンの総数は N 個の入力層の可能な組み合わせ、すなわち 2^N 個存在します。これらの入力集合を 0 か 1 かに分類する課題では、全部で 2^{2^N} とおりの分類が可能です。この中から M 個の入力を選んで訓練した場合には、残りの分類パターンはすべて一般化になるので $2^{2^N} - M$ とおりの一般化が可能になります。このことは入力層のユニット数 N が大きくなると、実質的に無限大の一般化が考えられることを意味します。

一方、訓練課題セットでは正解を得ることができるが、テスト課題では正解できないことがあります。学習が進行しすぎるとしばしば観察される現象で、過剰な学習がなされたことを意味する (過学習 overfitting)。図 14 は、誤った方向に一般化がなされた場合と、過学習によって訓練課題にだけ正解するようになった場合とを表したものです。

図 15 では正弦曲線 $y = 0.5 + 0.4 \sin(2\pi x)$ を 3 層のネットワークに学習させた例を示しました。この例では 0 から 1 までの 0.05 刻みの各 x 座標を入力信号として与え、対応する y の値を学習させました。実際の教師信号に若干のノイズを加えてあります。一般に教師信号に少量のノイズを加えたデータを学習させることで、ネットワーク一般化能力が向上すると言われています。ただし、データセット数に対して中間層の数が多いときに繰り返し学習を進行させると過学習が生じることがあります。図 15 中の 2 本の点線のうち、ほぼノイズを付加した教師信号完全に学習している点線では、過学習によって真の曲線を学習するのではなく真の関数とノイズとの合成積 convolution を学習しています。他方の点線では、ほぼ望み通りの結果を得ていますが[3]、最大値 0.9 付近、最小値 0.1 付近での真の関数とのずれがやや大きくなっていることが読み取れます。

2.5 中間層ユニット数の決定問題

入力情報が N ビットの情報を持っているとき、 全入力情報を損失無く表現するためには 2^N 個の中間層ユニットを用意すれば十分であることはすぐに分かる。ところが、これは中間層のユニット数決定のための必要条件ではありません。では最適な中間層のユニット数は幾つなのでしょうか？最適な中間層のユニット数を決定するためには、2.4 節の一般化の問題を踏まえて議論する必要があります (甘利、村田、Müller(1997), Elman ら (1996))。この問題は 2.4 節の一般化の問題と関連して論じられることが多いです。

村田ら (1994) は、神経回路網を確率機械と捉えて、情報量規準を用いて中間層を定める手法を提案しました。例えば、神経回路網を確率機械と捉え

[3]実際に用いた中間のユニット数は 6

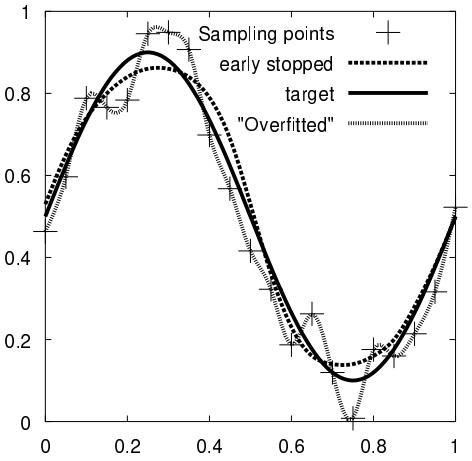

Figure 15: 正弦曲線を 3 層バックプロパゲーション法によって学習させた例。実線が $0.5 + 0.4\sin(2\pi x)$ の曲線であり、プラス (+) の記号で実際に用いられた教師信号 (ノイズ付加) が示されている。2 本の点線によって、中間層を少なくして学習終了基準を甘く設定した結果と、中間層の数を増やして意図的に過学習を起こさせた結果とを示した。

て、情報量規準を用いて中間層を定める手法が村田ら (Murata, Yoshizawa & Amari, 1994) によって提案されています。彼らは赤池の情報量規準 AIC (Akaike's Information Criterion, 坂本、石黒 & 北川 (1983)) を拡張した NIC (Network Information Criterion) を提案しました。データが与えられたときのモデルの対数尤度に自由パラメータ数の 2 倍を加えたものが赤池の情報量規準 AIC と呼ばれています。(データの当てはまりを表す量である) 対数尤度が同じならば自由パラメータ数の少ないモデルを選択すべきであることを AIC は主張しています。複数のモデル間で AIC を計算、比較して最適モデルを選択しようとするのが AIC の基本となるアイデアです。村田らの提案した NIC は、中間層のユニットをパラメータと考え、訓練データ上で、任意の入力信号に対するモデルからの出力と教師信号 (正解) との「ずれ」に、パラメータ数を加えたものとして定義されています。もし、モデルが真の入出力関係を実現可能であり、かつ、上記の「ずれ」が対数のマイナスで定義されているならば、NIC と AIC とは係数を除いて一致します。

2.6 中間層のユニット数の増減法

中間層のユニット数をネットワークに学習させるという手法も開発されています。最初は十分な数のユニットを用意して、全結合させ、学習中に不要な

結合を刈り込む手法を枝刈り法 (pruning あるいは weight elimination) といいます。結合係数 w_{ij} が徐々に 0 になるような傾向を持たせて、学習によって結合係数が補強されない限り不必要な結合が除去されるように方法を総称してこう呼びます。具体的には w_{ij} の更新後に

$$w_{ij}^{\mathrm{new}} = (1-\epsilon) w_{ij}^{\mathrm{old}} \tag{32}$$

を用います。このことはバックプロパゲーションで使われる誤差の 2 乗和 E_0 に w_{ij} の 2 乗和を加えた

$$E = E_0 + \gamma \frac{1}{2} \sum w_{ij}^2 \tag{33}$$

を新たに誤差関数として誤差逆伝播アルゴリズムによる結合係数の更新式

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{34}$$

を適用するのと定数項を除いて等価です。

枝刈り法は最適なネットワーク構造を探索するための手段として用いられることがある。枝刈り法のように、最初は全結合を作っておいて、あとで不要なものを刈り取ることは網膜から外側膝状体への投射、外側膝状体から第一次視覚野の間でも観測されている事実です。脳内でどのユニットがどの役割を果たすかが出生直後から決まっているわけではありません。出生後の環境によって柔軟に対応できるようにするためには、このほうが有利なのだとの解釈も成り立ちます。

一方、小数の中間層ユニットから出発して必要に応じてユニットを追加していく方法も開発されている。この方法は、構成法 (constuructive method) あるいは動的ノード生成法 (dynamic node creation) という。誤差の減少幅が小さくなって収束しないときに新しい中間層を加えることによって実現される (Reed & Marks II, 1999)。

文献

坂本, 石黒 & 北川 (1983). **情報量統計学**. 東京: 共立出版.

上坂, . (1993). **ニューロコンピューティングの数学的基礎**. 近代数学社.

甘利俊一, 村田昇 & Muller, R. (1997). 学習の数理モデル—汎化能力と過学習—. In 外山敬介 & 杉江昇 (Eds.), **脳と計算論** chapter 3, (pp. 37–53). 東京: 朝倉書店.

船橋賢一 (2000). 多層パーセプトロン. In 甘利 & 外山 (Eds.), **脳科学大辞典**. 朝倉書店.

Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Simth, A., Parisi, D. & Plunkett, K. (1996). *Rethinking Innateness: A connnectionist perspective on development.* Cambridge, MA: MIT Press. (邦訳「認知発達と生得性」, 乾, 今井, 山下訳, 共立出版).

Hebb, D. (1949). The organization of behavior. In J. Anderson & E. Rosefeld (Eds.), *Neurocomputing* chapter 4. New York: MIT press.

Minsky, M. & Papert, S. (1988). *Perceptrons, Expanded Edition* (2 Ed.). Cambridge, MA: MIT Press.

Murata, N., Yoshizawa, S. & Amari, S. (1994). Network information criterion — determining the number of hidden units for an artificial neural netwrork model. *IEEE Transactions on Neural Networks*, *5(6)*, 865–872.

Reed, R. D. & Marks II, R. J. (1999). *Neural Smithing: supervised learning in feedforward artificial neural networks.* Cambridge, MA: MIT Press.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psyochological Review*, *65*, 386–408. In J. A. Anderson and E. Rosenfeld (Eds.) Neurocomputing (1988), MIT Press.