ニューラルネットワーク (ISBN 4-254-11612-8) 自習ノート

目次

1 ニューラルネットワークとは何か

1.1 生物に学ぶ

1.1.1 脳の神経細胞

神経細胞には大きくわけて

- 錐体細胞 (pyramidal cell)
- プルキンエ細胞 (purkinje cell)
- 星状膠細胞 (astrocyte)

の 3 つがある。この 3 つのうち錐体細胞は脳の 70 から 85% を占める。神経細胞を三角で表すことがあるのは、この錐体細胞が三角形に見えるからである。

1.1.2 神経細胞の構造と機能

神経細胞は大きく

細胞体 (soma) 核 (nucleus) を細胞膜 (cell membrance) が包んだもの
樹状突起 (dendrite) 樹状突起は細胞体からつきだしているもの
軸索 (axon) 他の神経細胞の細胞体もしくは樹状突起へ繋がっているもの

によって構成される。軸索と他の神経細胞との結合部分をシナプス (synapse) という。

体液内と細胞内にはイオンが存在している。体内のイオンのうち神経細胞の働きに大きく影響を与えるのはナトリウムイオンとカリウムイオンである。通常、体液内にはナトリウムイオンが多く存在し、細胞内にはカリウムイオンが多く存在する。また細胞膜にはイオンチャネルが存在し、イオンチャネルが開いていると特定のイオンがイオンチャネルを通過することができる。細胞内と体液内には電位差があり、体液内に対する細胞内の電位を膜電位 (membrance potential) という。

神経細胞の細胞膜に存在するカリウムイオンチャネルは通常開いており、カリウムイオンが常に細胞内と体液内を行き来することができる。他に、電位依存性のナトリウムイオンチャネルや電位依存性のカリウムイオンチャネルが存在し、どちらも通常は閉じている。また細胞は DNA によってタンパク質を常に製造しており、タンパク質は負の電荷を帯びている。よって体液内のカリウムイオン (1 荷の正イオン) が体液内と細胞内の電位差によって細胞内に入ってくる。しかし、一定数のカリウムイオンが細胞内に入ると濃度の差によりカリウムイオンは細胞内から体液内へと移動する。通常時はこのカリウムイオンの出入りが安定した状態になる。この状態の膜電位を静止電位 (resting potential) といい一般に $-70\mathrm{mV}$ ほどである。

膜電位がある一定の値 (おおよそ $-50\mathrm{mV}$ から $-40\mathrm{mV}$) まで上がると、細胞膜上の電位依存性のナトリウムイオンチャネルが開く。すると電位差によりナトリウムイオン (1 荷の正イオン) が体液内から細胞内に入ってくる。これにより膜電位は大きく上昇し、膜電位が約 $40\mathrm{mV}$ ほどになると電位依存性のナトリウムイオンチャネルは閉じる。それとほぼ同時に電位依存性のカリウムイオンチャネルが開く。このとき、膜電位が正であることと細胞内にカリウムイオンが多く存在することから、カリウムイオンは細胞内から体液内へと移動する。これにより膜電位は大きく下がる。膜電位が約 $-70\mathrm{mV}$ 程度 (もうちょっと小さく?) になると電位依存性のカリウムイオンチャネルは閉じる。これ以降はカリウムイオンチャネルのみが開いている状態のため通常の状態に戻り膜電位は静止電位に戻る。膜電位が一定の値を越えてからのこの一連の流れが発生することを発火 (fire) するもしくは興奮 (excite) するという。また、神経細胞が発火する膜電位の値のことをしきい値 (threshold) といい、発火したと

きの膜電位の値を活動電位 (action potential) という。

発火している時間はだいだい同じであり、1 度しきい値を越えれば自動的に膜電位が増減するため同じような膜電位の増減のしかたをする。また、発火している最中に重ねて発火するようなことは起きないため発火すればどの神経細胞も同じような膜電位の増減のしかたをする。膜電位がしきい値を越えなかった場合も時間が経てば、膜電位は静止電位まで下がる。

この膜電位の増減は細胞膜上をつたわる。この膜電位の上昇が軸索をとおりシナプスへ伝わることで、シナプスが別の神経細胞へ信号を伝える。シナプスが他の神経細胞へ信号を伝える際、シナプスは伝達物質を用いるが、これには興奮性の物質と抑制性の物質がある。どちらの物質を出すかはシナプスによって決まっている。神経細胞は他の神経細胞とのシナプスから伝達物質を受け取ると膜電位が上昇する。

神経細胞を信号処理器と捉えると、軸索 (もしくはシナプス) は出力、細胞膜や樹状突起は入力とみることができる。複数の他神経細胞から伝達物質を受取り膜電位がしきい値を越えると、発火しそれを他の神経細胞へ伝えるという過程は非線形なふるまいであり、これが脳の高性能な処理能力の所以であるとされる。

1.2　神経細胞のモデル

1.3　シナプスの可塑性

1.4　ニューラルネットワークの分類

1.4.1　階層型ニューラルネットワーク

1.4.2　相互結合型ニューラルネットワーク

1.5　ニューラルネットワークの特徴

1.5.1　並列分散処理

1.5.2　学習と自己組織化

2　階層型ニューラルネットワーク

階層型ニューラルネットワークは心理学者ローゼンブラットによって提案されたパーセプトロンから発展したものである。パーセプトロンはパターンを学習・識別することができるニューラルネットワークであり、形式ニューロンとシナプスの可塑性を用いている。また、小脳においてパーセプトロンと類似した機能を有する部分があると指摘されている。本章ではパーセプトロンはら発展した階層型ニューラルネットワークについて述べる。ニューラルネットワークによる応用のうちで最も多いのが、階層型ニューラルネットワークのバックプ

ロパゲーションである。

2.1 パーセプトロン

2.1.1 単純パーセプトロン

パーセプトロンは 3 つの層からなる階層型ニューラルネットワークである。パーセプトロンを構成する 3 つの層はそれぞれ感覚ユニット (sensory unit)、連合ユニット (associate unit)、反応ユニット (response unit) と呼ばれる。パーセプトロンの基本形である単純パーセプトロンは感覚ユニットから連合ユニット、連合ユニットから反応ユニットへと一方向に繋がっている。また反応ユニットは 1 つのニューロンによって構成され、連合ユニットの全てのニューロンと接続される。

感覚ユニットの各ニューロンは入力を $x \in \{0,1\}$ としたとき出力値 z は

$$z = x$$

と表わされる。連合ユニットと反応ユニットのニューロンはマッカロとピッツの形式ニューロンであり、入力 (神経細胞が受け取るシナプス前神経細胞の出力) を $x_k \in \{0,1\}$ $(k = 1,2,\cdots,n)$、$w_k \in \mathbb{R}$ をシナプス結合荷重、y を膜電位の変化量としたとき、出力値 z は

$$y = \sum_{k=1}^{n} w_k x_k \tag{2.1}$$

$$f(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases} \tag{2.2}$$

$$z = f(y - h) \tag{2.3}$$

という計算式で表わされる。ここで感覚ユニットのニューロン数を m とし、i 番目 $(i = 1,2,\cdots,m)$ のニューロンの出力を $o_1^I,\cdots,o_m^I$、連合ユニットのニューロン数を n として感覚ユニットの i 番目のニューロンと連合ユニットの j 番目 $(j = 1,2,\cdots n)$ のユニットとのシナプス結合荷重を $w_{i,j}^{I,M}$ とすると連合ユニットの j 番目のニューロンへの入力は

$$\sum_{i=1}^{m} w_{i,j}^{I,M} o_i^I$$

と表わすことができる。この式は式 (2.1) に対応し連合ユニットの各ニューロンの出力値は式 (2.2)、式 (2.3) によって計算されるため、θ_j^M を連合ユニットの j 番目のニューロンのしきい値とすると連合ユニットのニューロンの出力値 $o_1^M,\cdots,o_n^M$ は

$$o_j^M = \begin{cases} 1, & \sum_{i=1}^{m} w_{i,j}^{I,M} o_i^I - \theta_j^M > 0 \\ 0, & \sum_{i=1}^{m} w_{i,j}^{I,M} o_i^I - \theta_j^M \leq 0 \end{cases}$$

となる。同様に連合ユニットの j 番目のニューロンと反応ユニットのニューロンとのシナプス結合荷重を $w_j^{M,O}$、反応ユニットのニューロンのしきい値を θ^O とすれば、反応ユニットのニューロンの出力値 o^O は

$$o^O = \begin{cases} 1, & \sum_{j=1}^{n} w_j^{M,O} o_j^M - \theta^O > 0 \\ 0, & \sum_{j=1}^{n} w_j^{M,O} o_j^M - \theta^O \leq 0 \end{cases} \tag{2.4}$$

となる。また、この単純パーセプトロンにおいて学習するパラメータは $w_j^{M,O}$ $(j = 1, 2, \cdots, n)$ と θ^O のみであり、$w_{i,j}^{I,M}$ $(i = 1, 2, \cdots, m, j = 1, 2, \cdots, n)$ と θ_j^M は固定して学習を行う。

ここで $w_{n+1}^{M,O} = -\theta^O$、$o_{n+1}^M = 1$ とすれば

$$\begin{aligned} \sum_{j=1}^{n} w_j^{M,O} o_j^M - \theta^O &= \sum_{j=1}^{n} w_j^{M,O} o_j^M + w_{n+1}^{M,O} o_{n+1}^O \\ &= \sum_{j=1}^{n+1} w_j^{M,O} o_j^M \end{aligned}$$

と書きなおすことができる。また、新たに $\mathbf{w}^{M,O} \in \mathbb{R}^{n+1}, \mathbf{o}^M \in \{0,1\}^{n+1}$ を

$$\mathbf{w}^{M,O} = \left(w_1^{M,O}, w_2^{M,O}, \ldots, w_n^{M,O}, w_{n+1}^{M,O}\right) = \left(w_1^{M,O}, w_2^{M,O}, \ldots, w_n^{M,O}, -\theta^O\right)$$

$$\mathbf{o}^M = \left(o_1^M, o_2^M, \ldots, o_n^M, o_{n+1}^M\right) = \left(o_1^M, o_2^M, \ldots, o_n^M, 1\right)$$

とするとベクトル同士の内積によって

$$\sum_{j=1}^{n} w_j^{M,O} o_j^M - \theta^O = \sum_{j=1}^{n+1} w_j^{M,O} o_j^M = \mathbf{w}^{M,O} \cdot \mathbf{o}^M \tag{2.5}$$

が成立する。

2.1.2 単純パーセプトロンの学習

単純パーセプトロンの学習法のうち、代表的なものである誤り訂正法 (learning by error-correction) について説明する。誤り訂正法は、複数の入力と教師信号 (入力を与えた際に求められる出力値) の組を用いて学習を行う。おおまかな手順は以下の通りである。

1. パーセプトロンに入力を与えて出力値を計算する
2. 得られた出力値と教師信号を比較してシナプス結合荷重としきい値によるベクトル $\mathbf{w}^{M,O}$ を修正する
3. 修正がなくなるまで 1 へ戻る

シナプス結合荷重としきい値によるベクトル $\mathbf{w}^{M,O}$ を修正した $\mathbf{w}'^{M,O}$ は

$$\Delta\mathbf{w}^{M,O} = \eta\left(t^O - o^O\right)\mathbf{o}^M$$

によって計算される $\Delta\mathbf{w}^{M,O}$ を用いて

$$\mathbf{w}'^{M,O} = \mathbf{w}^{M,O} + \Delta\mathbf{w}^{M,O} \tag{2.6}$$

と表わされる。ここで $t^O \in \{0,1\}$ は入力に対する教師信号、$\eta \in (0,1]$ は学習率 (learning rate) である。学習率は学習に大きく影響を与える定数であり、実際に学習する際には学習率 η は適切に決める必要がある。

t^O, o^O 共に 0,1 のどちらかであるため $t^O - o^O$ がとりうる値は

$$t^O - o^O = \begin{cases} 1, & t^O = 1, o^O = 0 \\ -1, & t^O = 0, o^O = 1 \\ 0, & t^O = o^O \end{cases}$$

であり $\Delta\mathbf{w}^{M,O}$ は

$$\Delta\mathbf{w}^{M,O} = \begin{cases} \eta\mathbf{o}^M, & t^O = 1, o^O = 0 \\ -\eta\mathbf{o}^M, & t^O = 0, o^O = 1 \\ \mathbf{0}, & t^O = o^O \end{cases}$$

と書くことができる。また、教師信号 t^O は入力に対して o^O がとるべき値であるため、式 (2.6) はパーセプトロンが入力に対して誤った値を出力したときにシナプス結合荷重としきい値を修正するという式である。さらに修正後の $\mathbf{w}'^{M,O}$ を用いて同じ入力に対する式 (2.5) の値を計算すると

$$\begin{aligned} \mathbf{w}'^{M,O} \cdot \mathbf{o}^O &= \left(\mathbf{w}^{M,O} + \Delta\mathbf{w}^{M,O}\right) \cdot \mathbf{o}^M \\ &= \mathbf{w}^{M,O} \cdot \mathbf{o}^O + \Delta\mathbf{w}^{M,O} \cdot \mathbf{o}^M \\ &= \mathbf{w}^{M,O} \cdot \mathbf{o}^O + \left[\eta\left(t^O - o^O\right)\mathbf{o}^M\right] \cdot \mathbf{o}^M \\ &= \mathbf{w}^{M,O} \cdot \mathbf{o}^O + \eta\left(t^O - o^O\right)\mathbf{o}^M \cdot \mathbf{o}^M \\ &= \mathbf{w}^{M,O} \cdot \mathbf{o}^O + \eta\left(t^O - o^O\right)\left\|\mathbf{o}^M\right\|^2 \end{aligned}$$

となり $t^O - o^O$ がとりうる値から、$t^O = 0, o^O = 1$ のときは式 (2.5) の値は $\eta\left\|\mathbf{o}^M\right\|^2$ だけ減ることで o^O が 0 になりやすくなり、$t^O = 1, o^O = 0$ のときは式 (2.5) の値は $\eta\left\|\mathbf{o}^M\right\|^2$ だけ増えることで o^O が 1 になりやすくなり、それ以外の場合は式 (2.5) の値と o^O の値は変化しないことがわかる。

学習をつづけることを時間の経過とらえることで式 (2.5) はヘブの強化則ということができ、式 (2.5) はシナプス結合荷重を変化させているためシナプスの可塑性を利用しているともいうことができる。

式 (2.6) によるシナプス結合荷重の修正を複数の入力と教師信号の組に対して繰り返し行なうことで学習を進める。誤り訂正法による単純パーセプトロンの学習の詳細な手順は以下の通りである。

Step1　複数の入力パターン $\mathbf{i}_p^I = (i_{p,1}^I, i_{p,2}^I, \cdots, i_{p,m}^I) \in \mathbb{R}^m$ $(p = 1, 2, \cdots, P)$ と、p 番目の入力パターンと対応する教師信号 $t_p^O \in \{0, 1\}$ を用意する。

Step2　シナプス結合荷重 $\mathbf{w}^{M,O} = \left(w_1^{M,O}, w_2^{M,O}, \cdots, w_{n+1}^{M,O}\right)$ の初期値を乱数によって小さい値に設定する。そして、学習率 $\eta \in (0, 1]$ を与える。

Step3　ある $p \in \{1, 2, \cdots, P\}$ に対して、p 個目の入力パターン $\mathbf{i}_p^I$ に対する連合ユニットの出力値 $o_{p,j}^M$ $(j = 1, 2, \cdots, n)$ を

$$o_{p,j}^M = \begin{cases} 1, & \sum_{i=1}^m w_{i,j}^{I,M} i_{p,i}^I - \theta_j^M > 0 \\ 0, & \sum_{i=1}^m w_{i,j}^{I,M} i_{p,i}^I - \theta_j^M \leq 0 \end{cases}$$

によって計算する。

Step4　得られた連合ユニットのニューロンの出力 $o_{p,j}^{M,O}$ $(j = 1, 2, \cdots, n)$ によって表わされるベクトル $\mathbf{o}_p^M = \left(o_{p,1}^M, o_{p,2}^M, \cdots, o_{p,n}^M, 1\right)$ とシナプス結合荷重から反応ユニットのニューロンの出力値 o_p^O を

$$o_p^O = \begin{cases} 1, & \mathbf{w}^{M,O} \cdot \mathbf{o}_p^M > 0 \\ 0, & \mathbf{w}^{M,O} \cdot \mathbf{o}_p^M \leq 0 \end{cases}$$

によって計算する。

Step5　反応ユニットのニューロンの出力値の計算に用いた入力パターンベクトル $\mathbf{t}_p^I$ と対応する教師信号 t_p^O と得られた反応ユニットの出力値 o_p^O を用いて計算される $\Delta\mathbf{w}^{M,O}$ を

$$\Delta\mathbf{w}^{M,O} = \eta \left(t_p^O - o_p^O\right) \mathbf{o}_p^M$$

によって計算する。そして、シナプス結合荷重 $\mathbf{w}^{M,O}$ に $\Delta\mathbf{w}^{M,O}$ を加算することでシナプス結合荷重を修正する。

Step6　任意の $p \in \{1, 2, \cdots, P\}$ に対して、入力パターンベクトル $\mathbf{i}_p^I$ から計算される反応ユニットの出力値 o_p^O と t_p^O が等しいならば終了する。そうでない場合は Step3 から Step5 までを繰り返す。

実際に学習を行う際には Step6 において、シナプス結合荷重の変化 $\Delta \mathbf{w}^{M,O}$ が一定の値より小さくなったら終了としたり、もしくは繰り返し回数に上限を設けて繰り返し回数が上限を越えたときに終了とするなどの条件も用いる。Step2 において学習率 η の値は大きいほど学習が早く進むがシナプス結合荷重や、教師信号と計算された反応ユニットの出力値との差が発散、振動しやすくなる。そのため η には適切な値を用いる必要がある。適切な η の値は問題の対象によって変化する。

単純パーセプトロンは入力パターンを 2 つの集合に分類することができる機械であり、分類する事を認識という。単純パーセプトロンにおける認識とは、入力パターンに対して反応ユニットのニューロンの出力値を計算することであり、出力値が 0 か 1 かによって入力パターンが 2 つの集合に分類される。式 (2.4) から反応ユニットのニューロンの出力値は式 (2.5) の値が正か負かによって決まることがわかる。式 (2.5) が

$$\sum_{j=1}^{n} w_j^{M,O} o_j^M - \theta^O = \sum_{j=1}^{n+1} w_j^{M,O} o_j^M = \mathbf{w}^{M,O} \cdot \mathbf{o}^M = 0$$

という $n+1$ 次元の超平面を表わしていると考えると、認識することは入力パターンと対応する $\mathbf{o}^M$ を超平面によって 2 つの集合にわけることと言いかえることができる。認識することは、入力パターン $\mathbf{i}^I \in \{0,1\}^m$ を座標変換によって $\mathbf{o}^M \in \{0,1\}^{n+1}$ へと変換し $\mathbf{o}^M$ を n 次元の超平面によって 2 つの集合のどちらかに分類わけすることと言いかえることができる。

また、単純パーセプトロンにおいて学習するパラメータはシナプス結合荷重 $\mathbf{w}^{M,O}$ のみであるため、入力パターン $\mathbf{i}^I$ から $\mathbf{o}^M$ への座標変換のパラメータは変化せずに n 次元の超平面の係数のみが変化することとなる。したがって、単純パーセプトロンの学習は入力パターンを適切に識別可能な n 次元超平面を求めることと同値となる。問題によってはどのような超平面を用いても正しく識別することができないものが存在し、逆に正しく識別することができる超平面が存在するとき線形分離可能という。

定義 1：線形分離可能 (linear separable) $F^+, F^- \in \{0,1\}^n$ $(n \in \mathbb{N})$ としたとき任意の $\mathbf{o}^+ \in F^+, \mathbf{o}^- \in F^-$ に対して

$$\mathbf{w} \cdot \mathbf{o}^+ > 0$$
$$\mathbf{w} \cdot \mathbf{o}^- \leq 0$$

となる $\mathbf{w} \in \mathbb{R}^n$ が存在することを F^+ と F^- は線形分離可能であるという。

パーセプトロンでは線形分離可能でない問題に対して適切な識別を行なうことはできない。したがってパーセプトロンが扱うのは線形分離可能な問題である。また線形分離可能な問題に対してであれば誤り訂正法を用いて必ず適切なパラメータを学習することができることが示される。

定理 1：パーセプトロンの収束定理 (perceptron convergence theorem)

定理 *1* の証明. 正しい識別を与えるシナプス結合荷重のうち任意の $\mathbf{o} \in F^+ \cup F^-$ に対して

$$\mathbf{o} \cdot \mathbf{w}^* \neq \mathbf{0}$$

となる $\mathbf{w}^* \in \mathbb{R}^{n+1}, \mathbf{w}^* \neq \mathbf{0}$ をひとつ選ぶ。実数空間が連続であることから $\mathbf{w}^*$ は必ず存在する。

また、シナプス結合荷重の初期値を $\mathbf{w}_0$、t 回修正を繰り返したシナプス結合荷重を $\mathbf{w}_t$ とすると式 (2.6) から、ある $\mathbf{o}^M \in \{0,1\}^{n+1}$ が存在して

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \left(t^O - o^O\right) \mathbf{o}^M \tag{2.7}$$

と書くことができる。ここで t^O は $\mathbf{o}^M$ と対応する教師信号であり、o^O は $\mathbf{o}^M$ を連合ユニットのニューロンの出力値からなるベクトル、$\mathbf{w}_t$ を連合ユニットと反応ユニット間のシナプス結合荷重として計算された反応ユニットの出力値である。

式 (2.7) の両辺と $\mathbf{w}*$ の内積をとり、変形すると

$$\begin{aligned} \mathbf{w}^* \cdot \mathbf{w}_{t+1} &= \mathbf{w}^* \cdot \left[\mathbf{w}_t + \eta \left(t^O - o^O\right) \mathbf{o}^M\right] \\ &= \mathbf{w}^* \cdot \mathbf{w}_t + \eta \left(t^O - o^O\right) \mathbf{w}^* \cdot \mathbf{o}^M \end{aligned} \tag{2.8}$$

となる。式 (2.8) は修正が発生したときの式であるため $t^O \neq o^O$ であり $\left(\mathbf{w}* \cdot \mathbf{o}^M\right) \left(\mathbf{w}_t \cdot \mathbf{o}^M\right) \leq 0$ である。したがって式 (2.4) から

$$o^O = \begin{cases} 0, & \mathbf{w}* \cdot \mathbf{o}^M > 0 \\ 1, & \mathbf{w}* \cdot \mathbf{o}^M \leq 0 \end{cases}$$

$$t^O = \begin{cases} 1, & \mathbf{w}* \cdot \mathbf{o}^M > 0 \\ 0, & \mathbf{w}* \cdot \mathbf{o}^M \leq 0 \end{cases}$$

であり、$\eta > 0$ であることと $\mathbf{w}^*$ の定義から

$$\eta \left(t^O - o^O\right) \mathbf{w}^* \cdot \mathbf{o}^M > 0$$

が成立する。ここで δ を

$$\delta = \min_{\mathbf{o} \in F^+ \cup F^-} \mathbf{w}^* \cdot \mathbf{o}^M$$

とすると δ は非ゼロな実数となり、任意の $\mathbf{o} \in F^+ \cup F^-$ に対して

$$\eta \left(t^O - o^O\right) \mathbf{w}^* \cdot \mathbf{o}^M \geq \eta\delta \tag{2.9}$$

となる。そして式 (2.8) と式 (2.9) から、任意の t ($t = 1, 2, \cdots$) に対して

$$\begin{aligned} \mathbf{w}^* \cdot \mathbf{w}_t + \eta \left(t^O - o^O \right) \mathbf{w}^* \cdot \mathbf{o}^M &\geq \mathbf{w}^* \cdot \mathbf{w}_t + \eta\delta \\ \rightleftharpoons \mathbf{w}^* \cdot \mathbf{w}_{t+1} &\geq \mathbf{w}^* \cdot \mathbf{w}_t + \eta\delta \end{aligned}$$

となる。したがって、初期値 $\mathbf{w}_0$ に n 回修正を行なった $\mathbf{w}_n$ に対して

$$\mathbf{w}^* \cdot \mathbf{w}_n \geq \mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta \tag{2.10}$$

が成立する。

また式 (2.6) から $\|\mathbf{w}_t\|^2$ は $\mathbf{w}_t$ によって

$$\begin{aligned} \|\mathbf{w}_{t+1}\|^2 &= \left\| \mathbf{w}_t + \eta \left(t^O - o^O \right) \mathbf{o}^M \right\|^2 \\ &= \|\mathbf{w}_t\|^2 + 2\eta \left(t^O - o^O \right) \mathbf{w}_t \cdot \mathbf{o}^M + \eta^2 \left(t^O - o^O \right)^2 \left\| \mathbf{o}^M \right\|^2 \end{aligned} \tag{2.11}$$

と表わすことができ、式 (2.8) と同様に式 (2.11) は修正が発生したときの式であるため $t^O \neq o^O$ であり $\left(\mathbf{w}^* \cdot \mathbf{o}^M \right) \left(\mathbf{w}_t \cdot \mathbf{o}^M \right) \leq 0$ である。したがって式 (2.4) から

$$o^O = \begin{cases} 1, & \mathbf{w}_t \cdot \mathbf{o}^M > 0 \\ 0, & \mathbf{w}_t \cdot \mathbf{o}^M \leq 0 \end{cases} \tag{2.12}$$

$$t^O = \begin{cases} 0, & \mathbf{w}_t \cdot \mathbf{o}^M > 0 \\ 1, & \mathbf{w}_t \cdot \mathbf{o}^M \leq 0 \end{cases} \tag{2.13}$$

であり、$\eta > 0$ であることとから

$$2\eta \left(t^O - o^O \right) \mathbf{w}_t \cdot \mathbf{o}^M \leq 0 \tag{2.14}$$

が成立する。ここで M を

$$M = \max_{\mathbf{o} \in F^+ \cup F^-} \left\| \mathbf{o}^M \right\|$$

とすると $0 < \eta \leq 1$ と式 (2.12)、式 (2.13)、式 (2.14) から、任意の t ($t = 1, 2, \cdots$) と $\mathbf{o}^M$ に対して

$$\begin{aligned} \|\mathbf{w}_t\|^2 + 2\eta \left(t^O - o^O \right) \mathbf{w}_t \cdot \mathbf{o}^M + \eta^2 \left(t^O - o^O \right)^2 \left\| \mathbf{o}^M \right\|^2 &\leq \|\mathbf{w}_t\|^2 + \eta^2 \left(t^O - o^O \right)^2 \left\| \mathbf{o}^M \right\|^2 \\ \|\mathbf{w}_t\|^2 + \eta^2 \left(t^O - o^O \right)^2 \left\| \mathbf{o}^M \right\|^2 &\leq \|\mathbf{w}_t\|^2 + \eta^2 \left(t^O - o^O \right)^2 M^2 = \|\mathbf{w}_t\|^2 + \eta^2 M^2 \end{aligned}$$

となり、式 (2.11) から

$$\|\mathbf{w}_{t+1}\|^2 \leq \|\mathbf{w}_t\|^2 + \eta^2 M^2$$

が得られる。したがって、初期値 $\mathbf{w}_0$ に n 回修正を行なった $\mathbf{w}_n$ に対して

$$\|\mathbf{w}_n\|^2 \leq \|\mathbf{w}_0\|^2 + n\eta^2 M^2 \tag{2.15}$$

が成立する。

さらに $\mathbf{w}^*$ が非ゼロベクトルであることから、式 (2.10) は $\mathbf{w}^*$ と $\mathbf{w}_n$ のなす角を θ とすると

$$\begin{aligned} \mathbf{w}^* \cdot \mathbf{w}_n &\geq \mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta \\ \rightleftharpoons \|\mathbf{w}^*\|\|\mathbf{w}_n\| \cos\theta &\geq \mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta \\ \rightleftharpoons \|\mathbf{w}^*\|\|\mathbf{w}_n\| &\geq \mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta \\ \rightleftharpoons \|\mathbf{w}_n\| &\geq \frac{\mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta}{\|\mathbf{w}^*\|} \end{aligned} \tag{2.16}$$

と書きかえることができる。

式 (2.15) と式 (2.16) から

$$\frac{(\mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta)^2}{\|\mathbf{w}^*\|^2} \leq \|\mathbf{w}_n\|^2 \leq \|\mathbf{w}_0\|^2 + n\eta^2 M^2 \tag{2.17}$$

となる。ここで式 (2.17) を見たす $\|\mathbf{w}_n\|^2$ が存在するためには

$$\frac{(\mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta)^2}{\|\mathbf{w}^*\|^2} \leq \|\mathbf{w}_0\|^2 + n\eta^2 M^2 \tag{2.18}$$

が成立している必要があり、式 (2.18) を変形すると

$$\begin{aligned} (\mathbf{w}^* \cdot \mathbf{w}_0 + n\eta\delta)^2 &\leq \|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2 + n\eta^2 M^2\|\mathbf{w}^*\|^2 \\ \rightleftharpoons n^2\eta^2\delta^2 + n\left[2\eta\delta\,(\mathbf{w}^* \cdot \mathbf{w}_0) - \eta^2 M^2\|\mathbf{w}^*\|^2\right] + (\mathbf{w}^* \cdot \mathbf{w}_0)^2 - \|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2 &\leq 0 \end{aligned} \tag{2.19}$$

となる。また、式 (2.19) の左辺は下に凸な放物線であるため

$$n^2\eta^2\delta^2 + n\left[2\eta\delta\,(\mathbf{w}^* \cdot \mathbf{w}_0) - \eta^2 M^2\|\mathbf{w}^*\|^2\right] + (\mathbf{w}^* \cdot \mathbf{w}_0)^2 - \|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2 = 0 \tag{2.20}$$

の解を n_-, n_+ $(n_- \leq n_+)$ とすると式 (2.19) が成立することと

$$n_- \leq n \leq n_+$$

が成立することは同値である。

ここで式 (2.20) の判別式 D は

$$\begin{aligned} D &= \left[2\eta\delta\,(\mathbf{w}^* \cdot \mathbf{w}_0) - \eta^2 M^2\|\mathbf{w}^*\|^2\right]^2 - 4\eta^2\delta^2\left[(\mathbf{w}^* \cdot \mathbf{w}_0)^2 - \|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2\right] \\ &= \eta^4 M^4\|\mathbf{w}^*\|^4 - 4\eta^3\delta M^2\,(\mathbf{w}^* \cdot \mathbf{w}_0)\,\|\mathbf{w}^*\|^2 + 4\eta^2\delta^2\|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2 \\ &= \left[\eta^4 M^4\|\mathbf{w}^*\|^2 - 4\eta^3\delta M^2\,(\mathbf{w}^* \cdot \mathbf{w}_0) + 4\eta^2\delta^2\|\mathbf{w}_0\|^2\right]\|\mathbf{w}^*\|^2 \end{aligned} \tag{2.21}$$

となり、式 (2.21) は $\mathbf{w}^* = \frac{2\delta}{\eta M^2}\mathbf{w}_0$ のときのみ 0 となり、それ以外の場合に正となることがわかる。さらに $(\mathbf{w}^* \cdot \mathbf{w}_0)^2 - \|\mathbf{w}^*\|^2\|\mathbf{w}_0\|^2 \leq 0$ であるため

$$n_- \leq 0 \leq n_+$$

が必ず成立し $\mathbf{w}^* = \frac{2\delta}{\eta M^2}\mathbf{w}_0$ でないとき n は有限の値 n_+ によってバウンドされる。また、$\mathbf{w}^* = \frac{2\delta}{\eta M^2}\mathbf{w}_0$ であるとき $n_- = n_+ = 0$ であり、$\mathbf{w}^*$ の正の実数倍ベクトルは正しく分離するため $n = 0$ となる。

以上より、シナプス結合荷重の初期値 $\mathbf{w}_0$ がゼロベクトルであるときに誤り訂正法によるパーセプトロンの学習が有限回の修正で収束することが示された。 □

線形分離可能な問題に対しては単純パーセプトロンは適切に学習することができることが示された。実際に単純パーセプトロンを用いる場合、感覚ユニットと連合ユニット間のシナプス結合荷重は学習せずに固定する。感覚ユニットへ線形分離可能でない入力群が与えられる場合でも、感覚ユニットと連合ユニット間でランダムに定められたシナプス結合荷重による座標変換によって、連合ユニットの出力ベクトルの空間上では線形分離可能な問題へ変換することができるようになる。連合ユニットのニューロン数 n が十分に大きいとき、入力パターン数が $2n$ 以下であるどのような問題であっても線形分離可能となることが示されている。

また、パーセプトロンの学習則である誤り訂正法をデルタ則 (delta rule) ということもある。

2.2 バックプロパゲーション

2.2.1 一般化デルタ則

前節で示したデルタ則 (誤り訂正法) を一般化した、一般化デルタ則を導出する。まず入力から出力を計算する過程を一般化する。

入力層でない層のあるニューロン j は、そのニューロンよりも前の層の任意のニューロンの出力 o_i (i は 1 から前の層のニューロン数までの整数) を受けとる。このとき、ニューロン j は前の層のニューロンとのシナプス結合荷重を前の層のニューロンに乗じ和をとっている。よってニューロン j への入力 u_j を

$$u_j = \sum_i w_{i,j} o_i \tag{2.22}$$

と定める。ここで $w_{i,j}$ は前の層のニューロン i とニューロン j のシナプス結合荷重とした。またしきい値を用いていないが、前節で用いたように前の層に出力値が 1 で固定されたニューロンを仮定することでしきい値を表現することはできる。ニューロン j の出力値 o_j

は非線形な実数上の 1 変数関数 f によって

$$o_j = f(u_j) \tag{2.23}$$

として求められる。この関数 f は活性化関数もしくは入出力関数といわれる。ここで f をステップ関数とすることで形式ニューロンの式と等価とすることができる。また、

$$f(x) = \frac{1}{1 + e^{-\epsilon x}}$$

と定義されるシグモイド関数は定義域が実数全体であり、終域が実数上の開区間 $(0,1)$ かつ単調増加連続関数であるという性質からよく用いられる。

次に学習する過程を一般化する。ここで、ある入力パターン p (p は 1 から入力パターン数までの整数) に対してパーセプトロンの出力層のニューロン j が出力すべき値を $t_{p,j}$、実際のニューロン j (j は 1 から出力層のニューロン数までの整数) の出力値を $o_{p,j}$ とし、誤差関数 E を二乗誤差によって

$$E = \frac{1}{2} \sum_{p,j} \left(o_{p,j} - t_{p,k} \right)^2 \tag{2.24}$$

と定める。この誤差関数 E の値が 0 となれば入力パターン全てを正しく識別できていることとなる。したがって、誤差関数 E の値を小さくすることを学習ということができる。

誤差関数 E は出力層のニューロンの値 $o_{p,j}$ に関する関数である。しかし出力層のニューロンの値 $o_{p,j}$ は出力層と前の層との間のシナプス結合荷重 $w_{i,j}$ の関数であるため、誤差関数 E も $w_{i,j}$ に関する (陰 (implicit) に定義された) 関数であるといえる。したがって、各シナプス結合荷重 $w_{i,j}$ に適切な $\eta > 0$ を用いた

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} \tag{2.25}$$

を加算することで誤差関数 E の極小値 (もしくは鞍点) へ近づくことができる。これは最急降下法 (steepest descent method もしくは gradient decent method) と呼ばれる。

また、式 (2.22) と式 (2.23) から連鎖律により

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{p} \frac{\partial E}{\partial o_{p,j}} \frac{\partial o_{p,j}}{\partial u_{p,j}} \frac{\partial u_{p,j}}{\partial w_{i,j}}$$

が得られる。ここで $u_{p,j}$ は入力パターン p に対して計算された出力層のニューロン j の入力値である。さらに式 (2.22) と式 (2.23) と式 (2.24) から

$$\frac{\partial E}{\partial o_{p,j}} = o_{p,j} - t_{p,j}$$

$$\frac{\partial o_{p,j}}{\partial u_{p,j}} = \left.\frac{df}{dx}\right|_{x=u_{p,j}}$$

となり、さらに $o_{p,i}$ を出力層の前の中間層の入力パターン p に対するニューロン i の出力値とすると

$$\frac{\partial u_{p,j}}{\partial w_{i,j}} = o_{p,i}$$

となる。よって式 (2.25) の各因子へ代入することで

$$\Delta w_{i,j} = -\eta \sum_{p} \left(o_{p,j} - t_{p,j}\right) \left.\frac{df}{dx}\right|_{x=u_{p,j}} o_{p,i} \tag{2.26}$$

が得られる。

式 (2.26) は一般化デルタ則と呼ばれる。一般化デルタ則では全ての入力パターンに対する誤差を計算してから、シナプス結合荷重の修正量を決め修正している。しかし、η が非常に小さい場合は入力パターン毎に修正量を決めて修正する方法でも、全体の修正量はほぼ等しくなる。

2.2.2 誤差逆伝搬法

誤差逆伝搬法はフィードフォワード型の階層型ニューラルネットワークの教師あり学習 (supervised learning) の代表的な方法である。入力パターンに対して計算された出力値と教師信号との誤差が小さくなるように、シナプス結合荷重を修正する。

誤差逆伝播法を用いることで入力層と中間層間での高次への写像も学習することができるため、線形分離可能でない問題に対しても適切なパラメータを学習することができる。

実際の学習は、まず入力パターンに対して各ニューロンの出力値を計算し、その後計算された誤差から逆向きにシナプス結合荷重の修正値を計算する。生物のニューラルネットワークでは信号が逆向きに伝わることはないため、誤差逆伝搬法は実際の生物のモデルとしては適切でない。しかし工学上の応用がききやすいため広く利用されている。

以下の表に誤り訂正法と誤差逆伝搬法の違いをまとめる。

表 1　誤り訂正法と誤差逆伝搬法の違い

	誤り訂正法	誤差逆伝搬法
対象	単純パーセプトロン	フィードフォワード型の階層型ニューラルネットワーク
出力値	2 値	実数上の開区間 $(0,1)$
入出力関数	ステップ関数	シグモイド関数
学習する値	中間層と出力層の間のシナプス結合荷重と出力層のしきい値	任意のシナプス結合荷重としきい値
終了条件	一定回数繰り返しても収束しない場合	2 乗誤差が一定値以下になること

m 層のフィードフォワード型の階層型ニューラルネットワークを学習する場合を考える。学習に用いる入力パターンは P 個のときを考える。パターン p に対しての k 層目の j 番目のニューロンの出力値を $o_{p,j}^{k}$、入力値を $i_{p,j}^{k}$、$k-1$ 層目の i 番目のニューロンと k 層目の j 番目のニューロンの間のシナプス結合荷重 $w_{i,j}^{k-1,k}$、k 層目の j 番目のニューロンの入出力関数を f_j^k、しきい値を θ_j^k、k 層目のニューロン数を n_k とすると

$$o_{p,j}^{k} = f_j^k(i_{p,j}^{k}) \tag{2.27}$$

$$i_{p,j}^{k} = \sum_{i=1}^{n_{k-1}} w_{i,j}^{k-1,k} o_{p,i}^{k-1} - \theta_j^k \tag{2.28}$$

という関係式が成立する。ここで k 層目に出力値が常に 1 である n_k+1 番目のニューロンの存在を仮定することで式 (2.28) は

$$i_{p,j}^{k} = \sum_{i=1}^{n_{k-1}+1} w_{i,j}^{k-1,k} o_{p,i}^{k-1}$$

と書きかえることができる。

パターン p に対する m 層目の j 番目のニューロンの教師信号を $t_{p,i}^{m^k}$ し、誤差関数 E を

$$E = \sum_{p} E_p$$

$$E_p = \frac{1}{2} \sum_{i=1}^{n_m} \left(t_{p,i}^{m} - o_{p,i}^{m} \right)^2$$

と定義する。ここで E_p はパターン p に対する誤差ということができる。

最急降下法によってシナプス結合荷重に

$$\Delta_p w_{i,j}^{k-1,k} = -\eta \frac{\partial E_p}{\partial w_{i,j}^{k-1}} \tag{2.29}$$

を加算することで修正する。η は半開区間 $(0,1]$ 上の実数である。

式 (2.29) の右辺の偏微分は連鎖律によって

$$\frac{\partial E_p}{\partial w_{i,j}^{k-1}} = \frac{\partial E_p}{\partial i_{p,j}^{k}} \frac{\partial i_{p,j}^{k}}{\partial w_{i,j}^{k-1}}$$

と書くことができ、

$$\frac{\partial i_{p,j}^{k}}{\partial w_{i,j}^{k-1}} = o_{p,i}^{k-1}$$

であるため

$$\frac{\partial E_p}{\partial w_{i,j}^{k-1}} = \frac{\partial E_p}{\partial i_{p,j}^{k}} o_{p,i}^{k-1}$$

となり、

$$\delta_{p,j}^{k} = -\frac{\partial E_p}{\partial i_{p,j}^{k}}$$

と置くことで

$$-\frac{\partial E_p}{\partial w_{i,j}^{k-1}} = \delta_{p,j}^{k} o_{p,i}^{k-1}$$

となる。最急降下法を用いるために非ゼロな 1 以下の実数定数を学習率 η として、$w_{i,j}^{k-1,k}$ のパターン p に対する修正量 $\Delta_p w_{i,j}^{k-1,k}$ は

$$\Delta_p w_{i,j}^{k-1,k} = \eta \delta_{p,j}^{k} o_{p,i}^{k-1}$$

と表わすことができる。

また、$\delta_{p,j}^{k}$ は連鎖律と式 (2.27) によって

$$\begin{aligned}
\delta_{p,j}^{k} &= -\frac{\partial E_p}{\partial i_{p,j}^{k}} \\
&= -\frac{\partial E_p}{\partial o_{p,j}^{k}} \frac{\partial o_{p,j}^{k}}{\partial i_{p,j}^{k}} \\
&= -\frac{\partial E_p}{\partial o_{p,j}^{k}} \left. \frac{df_j^k(x)}{di_{p,j}^{k}} \right|_{x=i_{p,j}^{k}}
\end{aligned}$$

と書くことができる。ここで

$$\frac{\partial E_p}{\partial o_{p,j}^k} = \begin{cases} o_{p,j}^m - t_{p,j}^m & k = m \\ \sum_{l=1}^{n_k+1} \left(\frac{\partial E_p}{\partial i_{p,l}^{k+1}} \frac{\partial i_{p,j}^{k+1}}{\partial o_{p,j}^k} \right) = -\sum_{l=1}^{n_k+1} \left(\delta_{p,l}^{k+1} w_{j,l}^{k,k+1} \right) & \text{otherwise} \end{cases}$$

となり、まとめると層数 m の階層型ニューラルネットワークの学習は

$$\Delta_p w_{i,j}^{k-1,k} = \eta \delta_{p,j}^k o_{p,i}^{k-1}$$

$$i = 1, 2, \cdots, n_k + 1, \quad j = 1, 2, \cdots, n_k, \quad k = 2, 3, \cdots, m$$

$$\delta_{p,j}^k = -\frac{\partial E_p}{\partial o_{p,j}^k} \left. \frac{df_j^k(x)}{di_{p,j}^k} \right|_{x = i_{p,j}^k}$$

$$\frac{\partial E_p}{\partial o_{p,j}^k} = \begin{cases} o_{p,j}^m - t_{p,j}^m & k = m \\ \sum_{l=1}^{n_k+1} \left(\frac{\partial E_p}{\partial i_{p,l}^{k+1}} \frac{\partial i_{p,j}^{k+1}}{\partial o_{p,j}^k} \right) = -\sum_{l=1}^{n_k+1} \left(\delta_{p,l}^{k+1} w_{j,l}^{k,k+1} \right) & \text{otherwise} \end{cases}$$

として表わされる $\Delta_p w_{i,j}^{k-1,k}$ を $w_{i,j}^{k-1,k}$ に各パターンに対して加算することで行われる。

したがって、誤差逆伝搬法において入出力関数は C^1 級関数である必要がある。誤差逆伝搬法で一般に用いられるシグモイド関数は

$$f(x) = \frac{1}{1 + e^{-\epsilon x}}$$

と定められる実数全体から実数区間 $(0,1)$ への単調増加連続関数である。また、シグモイド関数の 1 階導関数は

$$\frac{df(x)}{dx} = \epsilon f(x) (1 - f(x))$$

である。

学習の際に修正するとき、各パターン毎に修正する方法と全パターンの修正量をまとめて後から修正する方法の 2 つがある。前者を逐次修正法 (またはオンライン学習)、後者を一括修正法 (またはバッチ学習もしくはフルバッチ学習) という。また、逐次修正法と一括修正法を合わせて、いくつかのパターン分の修正をまとめて行なうミニバッチ学習という方法も用いられる。厳密に逐次修正法は全パターンに対する誤差の極小値を得るわけではないが学習率 η を小さくすることで全パターンに対する誤差の極小値を得ることができる。

2.2.3 応用例

誤差逆伝搬法は広く応用に使われている。応用されている事例をいくつか示す。

英語の発音学習システム a

2.2.4　ニューラルネットワークの構造とパラメータの与え方

2.2.5　誤差逆伝搬法の改良

3　相互結合型ニューラルネットワーク

3.1　相互結合型ニューラルネットワークの形態

相互結合型ニューラルネットワークは任意のニューロン同士が結合しているネットワークである。階層型ニューラルネットワークは一般に結合の向きで各ニューロンに順序関係が定まるのに対して、相互結合型ニューラルネットワークにおいては順序関係が定まらない。また、階層型ニューラルネットワークは一般に時系列を扱わないのに対して、相互結合型ニューラルネットワークは時刻の経過によって出力を計算する。

相互結合型ニューラルネットワークには外界から入力を受け取るユニット I、外界へ出力を出すユニット O、それ以外のユニット H が存在する。また必ずしも $I \cap O = \phi$ であるとは限らない。

3.2　連想記憶

人間における記憶の検索は連想によって行なわれているとされる。この連想による検索を再現するニューラルネットワークが相互結合型ニューラルネットワークである。ここで、ニューラルネットワークにおける連想記憶の定義を示す。

定義 2 : 連想記憶 (associative memory) 入力パターンの集合 I、出力パターンの集合 O に対してニューラルネットワーク n が写像 $n: I \to O$ として定まることである。また $I = O$ であるような連想記憶を自己相関連想記憶 (autoassociative memory)、$I \neq O$ であるような連想記憶を相互相関連想記憶 (heteroassociative memory) という。

代表的な相互結合型ニューラルネットワークであるホップフィールドモデルは最適化問題にも用いられている。

3.3　ホップフィールドモデル

3.3.1　ホップフィールドモデル (Hopfield model)

ホップフィールドモデルはアメリカの物理学者 J.J. Hopfield が導入したニューラルネットワークである。以下にホップフィールドモデルの特徴を示す。

- 複数のニューロンで構成され、各ニューロンは自分以外の全てのニューロンから出力

を受け取る

- エネルギー関数の極小化するように各ニューロンの内部状態が変化する
- ニューロン間のシナプス結合荷重が対称である
- ニューロンの出力は各ニューロンの内部状態に依存する
- ある時刻において、1 つのニューロンのみが他のニューロンからの出力を受け内部状態変化を起こす

ホップフィールドモデルは当初 2 値の出力値を扱うものとして提案されたが、その後連続値へと拡張された。このモデルは物理モデルから発展したものであり、脳のモデル化と言い難いものである。したがって、あくまでニューラルネットワークの特徴の解析から発生した 1 つの数理モデルであるととらえるべきである。

ニューラルネットワークの状態は各ニューロンの出力を並べてベクトルとすることで表わすことができる。

3.3.2 2 値ホップフィールドモデル

本節では $n \in \mathbb{N}$ 個の 2 値出力のニューロンからなるホップフィールドモデルを考える。離散時間 t $(t = 0, 1, 2, \cdots)$ のときにニューロン i $(i = 1, 2, \cdot, n)$ の内部状態 $u_i(t)$、出力値 $x_i(t)$ を

$$x_i(t+1) = \begin{cases} 1, & u_i(t) > 0 \\ x_i(t), & u_i(t) = 0 \\ 0, & u_i(t) < 0 \end{cases} \tag{3.1}$$

$$u_i(t) = \sum_{j=1}^{n} w_{i,j} x_j(t) - \theta_i \tag{3.2}$$

と定義する。ここで $w_{i,j}$ はニューロン i とニューロン j 間のシナプス結合荷重、θ_i はニューロン i の時刻 t におけるしきい値である。また $w_{i,j} = w_{j,i}, w_{i,i} = 0$ である。

式 (3.2) は神経細胞モデルにおいて膜電位の計算式と同等であるため、式 (3.1) とあわせてみると、膜電位を越えたら出力は 1 となり膜電位を越えない場合は出力は 0 となるととらえることができる。ホップフィールドモデルにおいてニューロンの内部状態とは階層型ニューラルネットワークでの膜電位と同じ意味の単語である。膜電位ではなく内部状態という単語を用いているのは、ホップフィールドモデルが物理から発展したモデルであるためである。

ニューラルネットワークの状態は各ニューロンの出力を並べたものであるため、n 個のニューロンによる 2 値ホップフィールドモデルならば 2^n の状態を持つ。

このニューラルネットワークのエネルギー関数 E は

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i \tag{3.3}$$

として定義される。式 (3.3) の値は任意の状態から式 (3.1) と式 (3.2) に従って内部状態変化を繰り返すことで必ず平衡状態へと遷移する。

Proof. ホップフィールドモデルにおいて 1 度の内部状態変化では 1 つのニューロンの内部状態のみが変化する。よって 1 度の内部状態変化でエネルギー関数の値が減少もしくは変らないことを示せばよい。

まず式 (3.3) をある k ($k = 1, 2, \cdots, n$) を用いて

$$\begin{aligned} E &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}x_i x_j + \sum_{i=1}^{n}\theta_i x_i \\ &= -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{i,j}x_i x_j + \sum_{i\neq k}\theta_i x_i - \frac{1}{2}x_k\sum_{i=1}^{n} w_{i,k}x_i - \frac{1}{2}x_k\sum_{j=1}^{n} w_{k,j}x_j + \theta_k x_k \end{aligned}$$

と変形すると $w_{i,j} = w_{j,i}$ であるため

$$E = -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{i,j}x_i x_j + \sum_{i\neq k}\theta_i x_i - x_k\sum_{i=1}^{n} w_{i,k}x_i + \theta_k x_k \tag{3.4}$$

となる。

ここである t ($t = 0, 1, 2, \cdots$) においてニューロン k が式 (3.1) と式 (3.2) に従って内部状態変化を起こしたとすると、 k でない任意の i ($i = 1, 2, \cdots, n$) に対して

$$\begin{aligned} x_k(t+1) &= x_k(t) + \Delta x_k \\ x_i(t+1) &= x_i(t) \end{aligned}$$

が成立する。ここで $\Delta x_k \in \{-1, 0, 1\}$ である。式 (3.4) から時刻 t と $t+1$ のエネルギー $E(t), E(t+1)$ を計算するとそれぞれ

$$\begin{aligned} E(t) &= -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{i,j}x_i(t)x_j(t) + \sum_{i\neq k}\theta_i x_i(t) - x_k(t)\sum_{i=1}^{n} w_{i,k}x_i(t) + \theta_k x_k(t) \\ E(t+1) &= -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{i,j}x_i(t)x_j(t) + \sum_{i\neq k}\theta_i x_i(t) - x_k(t+1)\sum_{i=1}^{n} w_{i,k}x_i(t) + \theta_k x_k(t+1) \end{aligned}$$

となり、エネルギー関数の差分 ΔE は

$$\begin{aligned} \Delta E &= E(t+1) - E(t) \\ &= -\left[x_k(t+1) - x_k(t)\right]\sum_{i=1}^{n} w_{i,k}x_i(t) + \left[x_k(t+1) - x_k(t)\right]\theta_k x_k(t+1) \\ &= -\left[\sum_{i=1}^{n} w_{i,k}x_i(t) - \theta_k\right]\Delta x_k \end{aligned}$$

と書くことができ、式 (3.2) から

$$\Delta E = -u_k(t)\Delta x_k$$

が得られる。時刻 t から $t+1$ にかけてニューロン k の内部状態が変化しなかった場合は $\Delta x_k = 0$ であるため $\Delta E = 0$ となる。内部状態が変化した場合を考えると $u_k(t) > 0$ かつ $x_k(t) = 0$ というパターンと $u_k(t) < 0$ かつ $x_k(t) = 1$ というパターンが考えられる。$u_k(t) > 0$ かつ $x_k(t) = 0$ である場合、式 (3.1) から $x_k(t+1) = 1$ であることため $\Delta x_k = 1$ となり $\Delta E < 0$ となる。$u_k(t) < 0$ かつ $x_k(t) = 1$ である場合も、式 (3.1) から $x_k(t+1) = 0$ であることため $\Delta x_k = -1$ となり $\Delta E < 0$ となる。よってどのような場合であっても $\Delta E \leq 0$ が成立する。したがって任意の t $(t = 0, 1, 2, \cdots)$ に対してエネルギーの差分 $\Delta E = E(t+1) - E(t)$ は $\Delta E \leq 0$ となり、$E(t)$ を時間の関数とみると単調減少であることが言える。

また、エネルギー関数の値はニューラルネットワークの状態に対して一意に定まるため同じニューラルネットワークの状態に遷移することなない。さらにニューラルネットワークの状態が有限であることから、必ずエネルギー関数の値に平衡点が存在し、エネルギー関数の値は時間の増加に対して収束する。 □

3.3.3 連想記憶への応用

2 値ホップフィールドモデルは連想記憶に適用された。これを実現する方法として、不正確なパターンを入力とし、ニューラルネットワークの状態遷移を発生させ、最終的に到達したニューラルネットワークの状態を想起パターンとすることである。また以降 2 値ホップフィールドの出力値は 0, 1 でなく −1, 1 を用いる。

パターンをベクトルで表現し各成分を各ニューロンの出力値とすれば、パターンのベクトル表記はニューラルネットワークの状態とみることができる。この記憶させることを記銘させるという。P 個のパターンを記銘させようとし、記銘させたい s $(s = 1, 2, \cdots, P)$ 個目のパターンを $\mathbf{x}^s \in \{-1, 1\}^n$ と表わすと、ニューラルネットワークにパターン $\mathbf{x}^s$ を記銘させるということは、エネルギー関数が $\mathbf{x}^s$ で極小値をとるようにシナプス結合荷重を与えることと等しくなる。ここで、各ニューロンのしきい値が 0 である場合のシナプス結合荷重の設計を考える。

各ニューロンのしきい値を 0 とするとエネルギー関数 (式 (3.3)) は

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}x_i x_j$$

となる。ここで、P 個のパターン $\mathbf{x}^s = \left(x_1^s, x_2^s, \cdots, x_n^s\right) \in \{-1, 1\}^n$ を極小値とするシナプス結

合荷重の一つは

$$w_{i,j} = \sum_{s=1}^{P} x_i^s x_j^s \qquad i,j = 1,2,\cdots,n$$

である。これは、この式をエネルギー関数に代入することで

$$\begin{aligned}
E &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=1}^{P} x_i^s x_j^s x_i x_j \\
&= -\frac{1}{2}\sum_{s=1}^{P}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^s x_i x_j^s x_j \\
&= -\frac{1}{2}\sum_{s=1}^{P}\sum_{i=1}^{n} x_i^s x_i \sum_{j=1}^{n} x_j^s x_j \\
&= -\frac{1}{2}\sum_{s=1}^{P}\left(\sum_{i=1}^{n} x_i^s x_i\right)^2 \\
&= \frac{1}{2}\sum_{s=1}^{P}\left[-\left(\sum_{i=1}^{n} x_i^s x_i\right)^2\right]
\end{aligned}$$

となり、x_i^s や x_i が -1 か 1 しかとらないことから、エネルギー関数が各 $\mathbf{x}^s$ を極小値とする 2 次関数の和となっていることがわかる。パターンの数 P が少ない等いくつかの制約の上ではこのエネルギー関数が任意の $\mathbf{x}^s$ を極小値とすることが直感的にわかる。また、$\mathbf{x}^s$ と $-\mathbf{x}^s$ はエネルギーの値が同じであるため $\mathbf{x}^s$ を記銘することと $-\mathbf{x}^s$ を記銘することは同じ意味を持つ。

なお、大きな n 似たいして記銘ベクトルをランダムに選ぶことで $\{-1,1\}$ の 2 値ホップフィールドモデルの記憶容量は

$$\frac{n}{2\log_2 n}$$

で近似できることが示されている。

3.3.4 連続値ホップフィールドモデル

本節では連続値ホップフィールドモデルについて考える。連続値ホップフィールドモデルと 2 値ホップフィールドモデルとの違いは、出力値がとりうる値の区間が $[0,1]$ であることである。また、出力値の変化に共なってニューロン i の内部状態 $u_i(t)$ の変化は微分方程式

$$\frac{du_i(t)}{dt} = -\frac{u_i(t)}{\tau} + \sum_{j=1}^{n} w_{i,j} x_j(t) - \theta_i \tag{3.5}$$

$$x_i(t) = f_i(u_i(t)) \tag{3.6}$$

で表わされる。ここで $i = 1, 2, \cdots, n$ であり、$f_i(u)$ は非線形な連続単調増加関数である。また、τ はニューロンの内部状態における時定数である

式 (3.5) の第 1 項目は $u_i(t)$ を指数関数的に減少させることで、$u_i(t)$ の発散を抑制し平衡値へと近づける効果がある。第 1 項を除けば式 (3.5) は式 (3.1) と同等の式とみることができ、膜電位に対応する値が正であれば $u_i(t)$ は時間的に増加させる、負であれば $u_i(t)$ は時間的に減少させる、0 であれば $u_i(t)$ の変化に寄与しないという効果がある。また

$$u_i(t) = \tau \left(\sum_{j=1}^{n} w_{i,j} x_j(t) - \theta_i \right)$$

のとき式 (3.5) は 0 となり平衡状態となり、$\tau = 1$ のとすると 2 値ホップフィールドモデルの式 (3.2) と一致する。

このネットワークに対してエネルギー関数は

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i + \frac{1}{\tau} \sum_{i=1}^{n} \int_0^{x_i} f_i^{-1}(x) dx \tag{3.7}$$

と定義される。ここで $f_i^{-1}(x)$ は $f_i(x)$ の逆関数である。式 (3.7) は 2 値ホップフィールドモデルのエネルギー関数 (式 (3.3)) に項を追加したものである。これはエネルギー関数が時間 t に対して単調減少関数であるために追加された項である。

2 値ホップフィールドモデルと同様に時間変化によるエネルギーの変化を考える。エネルギー関数 (式 (3.7)) を時間 t によって微分すると式 (3.4) によって

$$\begin{aligned}
\frac{dE}{dt} &= \sum_{i=1}^{n} \frac{\partial E}{\partial x_i} \frac{dx_i}{dt} \\
&= \sum_{i=1}^{n} \left(-\sum_{j=1}^{n} w_{i,j} x_j + \theta_i - \frac{1}{\tau} f_i^{-1}(x_i) \right) \frac{dx_i}{dt} \\
&= -\sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{i,j} x_j - \theta_i + \frac{1}{\tau} f_i^{-1}(x_i) \right) \frac{dx_i}{dt}
\end{aligned}$$

となり、f_i^{-1} が f_i の逆関数であることと式 (3.5) と式 (3.6) から

$$\frac{dE}{dt} = \sum_{i=1}^{n} \frac{du_i}{dt} \frac{dx_i}{dt}$$

となり、また

$$\frac{dx_i}{dt} = \frac{df_i}{du_i} \frac{du_i}{dt}$$

であるため

$$\frac{dE}{dt} = \sum_{i=1}^{n} \frac{df_i}{du_i} \left(\frac{du_i}{dt} \right)^2$$

が得られる。よって $\frac{df_i}{du_i} > 0$ であることから

$$\frac{dE}{dt} \leq 0$$

となり、エネルギー関数 (式 (3.7)) が時間に対して単調減少であることが示された。エネルギー関数が単調減少であるため、連続値ホップフィールドモデルのエネルギーは時間 $t \to \infty$ によって極小値となる。

ホップフィールドは単調増加関数としてシグモイド関数

$$f_i(u_i) = \frac{1}{1 + e^{-\frac{2u_i}{\mu_0}}}$$

を用いており、ここで μ_0 は基準活性化レベル (reference activation level) と呼ばれる。

3.3.5 最適化問題への応用

組み合わせ最適化問題 (combinatorial optimization problem) は、目的関数を最小化する組み合わせを探す問題である。ここで組み合わせ最適化問題の変数列をベクトル $\mathbf{x} \in \{0,1\}^n$ へ変換し、また目的関数と式 (3.3) が一致するような $w_{i,j}$ を求めることができれば、2 値ホップフィールドモデルの平衡状態は組み合わせ最適化問題の目的関数の極小値と同等である。よって 2 値ホップフィールドモデルは組み合わせ最適化問題を解きうるモデルであるといえる。本節では NP 困難である組み合わせ最適化問題の中で代表的な問題である TSP を解くことを考える。

遺伝的アルゴリズム (genetic algorithm) は 1960 年代に Holland によって導入されたアルゴリズムである。これは生物の進化の過程を模したアルゴリズムであり、1980 年代ごろから

活発に研究され始めた。特徴としては主に以下の 3 つがある。

- 複数の解候補を同時並行的に更新していく反復解法
- 解候補に対する交叉 (crossover) という演算が用いられる
- 柔軟性が高い

遺伝的アルゴリズムの登場以来、このアルゴリズムの特徴を取り入れて別のアルゴリズムが改良されたり、遺伝的アルゴリズムが別のアルゴリムを取り入れて改良されたりしている。

遺伝的アルゴリズムは生物の進化過程である遺伝を模したアルゴリムである。生物の個体は形質とよばれる形態的・生理的な特徴を継承していく。これは細胞が分裂するさいに細胞中の遺伝子 (gene) が複製されることによるものである。細胞分裂時には各遺伝子が一列に並んだ染色体 (chromosome) となり、各遺伝子が染色体上のどの位置に並ぶかは決まっており、この位置を遺伝子座 (locus) という。また遺伝子は劣勢な遺伝子や優勢な遺伝子等複数種類存在し、遺伝子座が同じであり種類の違う遺伝子を対立遺伝子 (allele) と呼ぶ。対立遺伝子の並びを遺伝子型 (genotype) と言い、遺伝子型と遺伝子の種類によって表面的に表われる形質を表現型 (phenotype) という。通常の細胞分裂では遺伝子型は変化しないが、2 本の染色体の一部が入れ換って遺伝子の組替えが生じたり、一部の遺伝子が変化したりすることによって新しい遺伝子型が形成される。ここで前者を交叉、後者を突然変異 (mutation) と呼ぶ。このような手順で生物は子孫を残す。生物いろいろな遺伝子を持つが、環境に適応し生き残ることができた個体が子孫を残すことになるため、子孫に遺伝する遺伝子は環境への適応度 (fitness) が高いものである可能性が高い。このように優秀な遺伝子が残される仕組みを選択・淘汰 (selection) と呼ばれる。

遺伝的アルゴリズムは上述した生物の進化過程の上で個体を解候補、適応度を目的関数へと対応づけ、選択・交叉・突然変異に対応した演算を実行することで目的関数により適した解を得ようとするアルゴリズムである。このアルゴリズムは生物の進化をモデル化したものではあるが、最適化の計算過程と生物進化の知見とは強い関係はない。

6.2 アルゴリズム

一般に遺伝的アルゴリズムは最適化問題に対する近似解法として用いられる。目的関数を z、実行可能領域を A としたときに A を終域とする全単射が定まるような始域 P の元として個体が定まる。個体 $p \in P$ は遺伝子 g の集まりとして表現され、どのように遺伝子を並べて個体を表現するかは比較的任意性がある。また 1 列に並んだ遺伝子 (?) を染色体と呼ぶ。最も単純な並べ方としては目的関数の変数を 1 次元的に並べたものであるが、交叉・突然変異等が適切に働くのであれば別の表現を用いることもできる。

また遺伝子による個体の表現 $p \in P$ を遺伝子型、遺伝子型から得られる解候補 $a \in A$ を表現型と言い、P の元から A の元へと写像することをデコーディング (decoding) という。これらの関係を図式に表すと

$$P \xrightarrow{decoding} A$$

となる。本節では個体と遺伝子型を等しいものとして扱う。

複数の個体 (individual) からなる個体群 (population)P に対して、以下の繰り返し計算を行うことで解を探索する。なお、遺伝的アルゴリズムでは 1 回の繰り返しを 1 世代 (generation) と呼ぶ。

Step1　初期個体群を生成し、世代を 1 とする

Step2　交叉則・突然変異則・選択則を現在の個体群に適用し、新たな個体群を生成する

Step3　終了条件を満たせば計算を終了しそれまでに得られた最もよい個体を解とする、そうでない場合は世代を 1 つ増やして Step2 へ戻る

Step3 での終了条件は一般に世代数の上限として定められる。ほかには、何世代か続けて現最良個体よりもよい個体が生成されないこととする等の方法がある。

Step2 での交叉則・突然変異則・選択則は遺伝演算子 (genetic operation) と呼ばれる。

交叉則　複数の個体の遺伝子をいくつか入れ換えて新しい個体を生成する演算である。ある $p_1, p_2, \cdots, p_n$ に交叉則を作用させ、$p'_1, p'_2, \cdots, p'_n$ が得られたとき、$p_1, p_2, \cdots, p_n$ それぞれを親、$p'_1, p'_2, \cdots, p'_n$ それぞれを子という。

突然変異則　ある個体の遺伝子を変更して新しい個体を生成する演算である。交叉則と同様に、p から p' が生成された場合は p を親、p' を子という。

選択則　個体群中の個体、もしくは個体群中の個体から交叉則や突然変異則によって生成された子から次の世代に必要な個体を選択する演算である。各個体への目的関数が大きい (小さい) 個体、もしくは適応度の高い個体を選択する。

交叉則や突然変異則によって生成された子は親と近い遺伝子を持つため、親の適応度が高ければ子の適応度も高くなることが期待される。また交叉則において親の適応度が高くない場合でも親の優れた遺伝子のみを受け継ぎより高い適応度の子が生成される可能性もある。

各個体が 6 つの遺伝子で構成される 6 個体の個体群に対する遺伝的アルゴリズムの一例を示す。

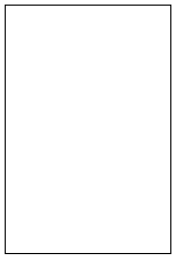

図 1　遺伝的アルゴリズムの動作例

図の説明

図 1 での例のみでは実際にアルゴリズムを実行することはできない。これは具体的な交叉や選択則を示していないからであるが、これらは適当に定める必要がある。この任意性によって遺伝的アルゴリズムは様々な問題へと適用できる柔軟性を持っている。また定め方次第で得られる解や収束速度に大きく影響する。基本的には最適化問題の特徴をうまく取り出して、それをアルゴリズムに組込むことで良い結果が得られやすい。以降、比較的よく用いられる手続きを説明する。

6.2.1　個体表現

遺伝的アルゴリズムの設計において最も重要なのが個体の表現方法である。個体の表現が重要なのは他のアルゴリズム等でも同じであり、問題の特徴をうまく取り入れる必要がある。したがって、他の問題で用いられる表現方法を用いることも可能である。また、問題の特徴を取り入れる必要があるため個々の問題に依存する。よって、いくつかの最適化問題の例と共に説明する。

0-1 表現　遺伝子を 0 もしくは 1 とし、個体を遺伝子が 1 次元的に並んだもとして表現する方法であり、最もよく利用される。

ナップザック問題 (knapsack problem)　ナップザック問題は最大化問題であり、N 個の荷物をどのようにナップザックへと入れるかを決める問題である。具体的には、各荷物 i $(i = 1, 2, \cdots, N_1)$ に価値 T_i とコスト C_i が与えられており、コストが一定値を越えないように最大の価値となるように荷物を選ぶ問題である。コスト

の最大を C とするとナップザック問題は

$$\max z = \sum_{i=1}^{N} x_i T_i$$
$$\text{s.t.} \sum_{i=1}^{N} x_i C_i \leq C$$
$$x_i \in \{0,1\}$$

と書かれる。

このナップザック問題では 0 と 1、すなわち x_i を i が 1 から N なで並べることで個体を表現する。

関数最適化問題　関数最適化問題は関数の最小値を求める問題である。ここでは

$$\min z = \sum_{i=1}^{N} x_i^2$$
$$\text{s.t. } -5.12 \leq x_i < 5.12$$

と書かれる最小化問題の個体表現を考える。

本来 x_i は実数値であるが、0-1 表現とするために x_i を 4 ビットの 2 進数で表わすこととすると個体は 0 もしくは 1 を $4N$ 個並べたものとなる。ある x_i を表わす各ビットを $b_i^1, b_i^2, b_i^3, b_i^4$ とすると

$$x_i = -\frac{5.12 + 5.12}{16} \sum_{j=0}^{3} b_i^j 2^j + 5.12$$

という変換によって x_i を得る。

遺伝子の列として b_i $(i = 1, 2, \cdots, 4N)$ が与えられたとき、各 x_j $(j = 1, 2, \cdots, N)$ へのデコーディングは

$$x_j = -\frac{5.12 + 5.12}{16} \sum_{k=1}^{4} b_{4(j-1)+k} 2^{k-1} + 5.12$$

として行なわれる。

0-1 表現の拡張　0-1 表現では 0 もしくは 1 を遺伝子として用いたが、そもそも 0 と 1 である必要はないので代替として A や B を用いたりすることもできる。また、0 と 1 の 2 つである必要もないので 3 種類以上の遺伝子を用いる場合もある。

0-1 のナップザック問題ではナップザックが 1 つであったがナップザックが複数の場合もあり、ナップザックの数を M とし j 個目のナップザックの容量を C_j とすると

$$\max z = \sum_{i=1}^{N} x_i^j T_i$$
$$\text{s.t. } \forall j, \sum_{i=1}^{N} x_i^j C_i \leq Cj$$
$$\forall i, \sum_{j=1}^{M} x_i^j = 1$$
$$x_i^j \in \{0,1\}$$

と書くことができる。このような場合遺伝子として $\sum_{j=1}^{M} x_i^j 2^{j-1}$ を用いることで遺伝子の種類は $1,2,\cdots,M$ の M 種類となる。

最適化問題では 2 進数で表わすのでなくそもそも遺伝子を $-3.84, 0.64, 0$ として個体表現を行うこともできる。

$f(x)$ あ $f(x)$ い

順序表現　最適化問題には最短経路探索等解が順序として与えられるものが存在する。このような問題での個体表現には順序そのものを用いることが多い。0-1 表現と違い、各対立遺伝子は個体のうちに 1 つづつしか存在することができない。

巡回セールスマン問題 (traveling salesman problem)　ある枝にコストが与えられたグラフに対してコストを最小化するオイラー閉路を得る問題である。巡回セールスマン問題は全てのノードを通る路を解とする問題であるため、解となる路の長さはノード数と等しくなる。

フローショップ・スケジューリング問題 (flow shop scheduling problem)　フローショップ・スケジューリング問題は N 個の製品と M 個の機械があり、どの製品も機械 1 から機械 M へと順に入れることで生産される。しかし各機械は同時に 1 つの製品しか生産することができず、また同時に 1 つしか機械を動かすことができない。この条件下で機械 1 にどのような順番で製品を入れれば生産の合計時間を最小化できるかを考えるものである。この場合も巡回セールスマン問題と同じように製品の並びによって個体を表現することができる。

その他の表現　0-1 表現、順序表現共に 1 列の遺伝子、すなわち 1 つの染色体による個体表現を例としたが、個体は複数の染色体で表現することもできる。また、列でなくグラフや行列を用いることもできる。さらに列である場合も遺伝子の数が固定である必要もない。特殊な染色体に対しては、その染色体の表現用の交叉則や突然変異則を用い

たりする。

6.2.2 交叉則

交叉則は遺伝的アルゴリズムの特徴的な演算である。この交叉則が有効に機能することでほかの解法よりもよりよい解が得られる場合がある。交叉則において重要なのでは個体表現が持つ元問題の特徴を残して子へと継承することである。そうでない場合はランダムに子を生成しているのと変わらない。交叉は世代の全ての個体に対して行われるのでなく、事前に定めた確率によって実行される。また、交叉する時にも乱数が用いられる。

以降、一般に用いられる 2 個の親から 2 個の子を生成する交叉の例を示す。

1 点交叉 (one-point crossover)　1 点交叉は 0-1 表現や拡張された 0-1 表現に対して用いられる交叉である。この交叉を順序表現に対して用いると順序表現が持つべき同じ遺伝子が存在してはならないという上限条件が崩れてしまうため、順序表現には用いることはできない。実際の生物では、2 つの染色体のある部分以降が交換されることで新たに染色体が作られる。この実際の生物をモデルとしたものが 1 点交叉である。

1 点交叉ではまず、染色体のうちのある 1 箇所をランダムに選びそこから染色体を 2 つにわけ部分遺伝子を生成する。わけた部分遺伝子を入れ換えることで新たな子の染色体を 2 つ生成する。

2 点交叉 (two-point crossover)　2 点交叉は 1 点交叉を拡張したものである。1 点交叉では 1 箇所をランダムに選びそこで染色体を 2 分したのに対し、2 点交叉ではランダムに染色体上の 2 箇所を選び、選んだ 2 箇所に挟まれている染色体を入れ換える。

一様交叉 (uniform crossover)　一様交叉は 1 点交叉は 2 点交叉等をより拡張したものである。一様交叉では各遺伝子座にてランダムに遺伝子を入れ換える。これは毎回交叉点の数がランダムに変化するような n 点交叉ということができる。

部分一致交叉 (partially matched crossover)　部分一致交叉は順序表現に対して用いられる交叉である。具体的には以下の手順で行われる。

Step1　親において、ランダムに 2 つの遺伝子座 $m_1, m_2 (m_1 < m_2)$ を選び、それらの遺伝子座の後ろに切れ目を入れる。

Step2　親 1 の 2 つの切れ目の外側の部分遺伝子列を子 1 にコピーし、親 2 の 2 つの切れ目の間の部分遺伝子列を子 1 にコピーする。また、$n = m_1 + 1$ とする。

Step3　子 1 の左から n 番目にある対立遺伝子が子 1 の中に 1 つしかない場合は、n 番目の遺伝子を親 1 の n 番目の遺伝子に置き換える。対立遺伝子が 2 つある場合は子 1 の中の n 番目でない同じ対立遺伝子の遺伝子座を探し、その位置の遺伝子を親 1 の n 番目の遺伝子で置き換える。

Step4　もし $n < m_2$ ならば $n = n + 1$ として Step3 へ戻る。

Step5　親 1 を親 2、親 2 を親 1、子 1 を子 2 に読み替えて Step2 から Step4 までの手順を繰り返し実行して子 2 を得る。

例として遺伝子を 3 分割する方法を示したが、一様交叉や n 点交叉のようにして分割して行うこともある。

順序交叉 (order crossover)　順序交叉は部分一致交叉と同様に順序表現に対して用いられる交叉である。遺伝子の長さが L である個体群に対しては具体的には以下の手順で行われる。

Step1　個体の遺伝子座を 2 つランダムに選び、$m_1, m_2 (m_1 < m_2)$ とする

Step2　親 1 の遺伝子 I_1 を遺伝子座 1 から m_1 までの遺伝子を I_1^l、$m_1 + 1$ から m_2 までの遺伝子を I_1^c、$m_2 + 1$ から L までの遺伝子を I_1^r とし、同様に親 2 の遺伝子 I_2 を I_2^l, I_2^c, I_2^r とわける。

Step3　I_1^c 内での対立遺伝子同士の順序関係が親 2 の遺伝子 I_2 内での順序関係と一致するように I_1^c 内の遺伝子を並び変えて $I_1^{c\prime}$ とする。

Step4　同様に I_2^c 内での対立遺伝子同士の順序関係が親 1 の遺伝子 I_1 内での順序関係と一致するように I_2^c 内の遺伝子を並び変えて $I_2^{c\prime}$ とする。

Step5　I_1^l と $I_1^{c\prime}$ と I_1^r を並べることで子 1 の遺伝子とし、I_2^l と $I_2^{c\prime}$ と I_2^r を並べることで子 2 の遺伝子とする。

部分一致交叉と同様に、一様交叉や n 点交叉のようにして分割して行うこともある。

6.2.3　突然変異則

突然変異則も交叉則と同様に遺伝的アルゴリズムの特徴的な点であり、問題の特徴を破壊せずに子へと継承するような演算であるべきである。また、突然変異則はタブーサーチ等ほかの解法でも同じ手法が用いられることもあるため、別の解法から流用することも可能である。

0-1 表現等に対する突然変異則　ランダムに一つ遺伝子を選択し、その遺伝子を別の種類の対立遺伝子へと変更する。変更する遺伝子の数は複数の場合もある。

交換突然変異 (swap mutation)　交換突然変異は順序表現等で個体の順序表現を崩さないように突然変異を起こす。この突然変異はランダムに選択した 2 つの遺伝子を交換するものである。

移動突然変異 (shift mutation)　移動突然変異も交換突然変異と同様に順序表現等に対して用いる突然変異である。この突然変異はランダムに選択した 1 つの遺伝子を、ランダムに選択したもう 1 つの遺伝子の前に移動させるものである。

6.2.4 選択則

選択則は優秀な個体を選択するように設計する。一般に目的関数の値が良いものを選べばよいが、目的関数の値が悪い値であっても良い遺伝子を持っている可能性はあるため、そのような個体を選べたほうがよい。しかし、どの遺伝子が優秀かは一般に事前にわからないため乱数を用いる。

一般に適応度は正の数であり、大きければ大きいほどよいものであるとする。最大化問題では目的関数をそのまま使えばよく、最小化問題では目的関数に負数を乗じ、一定の正数を足すことで適応度として用いることができる。また、最大化問題であっても一定の正数を加算する場合がある。

適応度比例選択 (fitness proportional selection) 実際の生物の適応度はその個体の繁栄度合いとみることができる。よって生物のモデルをそのまま遺伝的アルゴリズムに適用したのが適応度比例選択 (もしくはルーレット選択) である。適応度比例選択では個体群 S のある個体 $s \in S$ が選択される確率 e_s は、s の適応度を F_s とすると

$$e_s = \frac{F_s}{\sum_{s' \in S} F'_s}$$

と書かれる。ここで適応度 F_s $(s \in S)$ にある定数 $B > 0$ が足されていてある F'_s を用いて

$$F_s = F'_s + B$$

と書かれるとすると選択される確率は

$$e_s = \frac{F_s}{\sum_{s' \in S} F'_s} = \frac{F'_s + B}{\sum_{s' \in S} F'_{s'} + NB}$$

と書くことができ、$B \to \infty$ で極限をとると

$$\lim_{B \to \infty} e_s = \lim_{B \to \infty} \frac{F'_s + B}{\sum_{s' \in S} F'_{s'} + NB} = \frac{1}{N}$$

となる。また、e_s は B に対して単調減少関数であるため B が大きくなるにつれて F_s に関わらず e_s の値は一定へと近づくことがわかる。つまり B が大きければ大きいほど適応度が低い個体も選ばれやすくなるということになる。この B に関して、B の値が小さいときに選択の圧力が強いといい、B の値が大きくなると選択の圧力が弱くなるという。

選択の圧力は遺伝的アルゴリズムの性能に大きく影響するため、適応度比例選択を用いる場合は選択の圧力が適切になるように定める必要がある。

トーナメント選択 (tournament selection)　適応度比例選択では、選択の圧力を適切に定める必要があり、また実行時によい個体が選択されない可能性もある。このデメリットを解消する代表的な方法がトーナメント選択である。

トーナメント選択では個体群 S からランダムに $T \in \mathbb{N}$ 個の個体を取り出し、そのなかから値が最もよい個体を選択することで個体を 1 つ選ぶ。この操作を $|S|$ 回行うことで $|S|$ 個の個体を選ぶ。これにより、適応度の高い個体だけでなく適応度が低い個体が選択される可能性も高くなる。このトーナメントのサイズ T は一般に 2 がよく用いられる。

またこの方法だとトーナメントに参加できない個体が発生する可能性があるため、個体群 S の個体を T 個ならべて $|S|T$ 個の個体の集りを作り、これを並び変えて T 個の集りを再度作ることでサイズ T のトーナメントを $|S|$ 個作る方法も用いられる。この方法であれば全ての個体が必ずトーナメントに参加することができるため適応度が高い個体は必ず選択される。

適応度比例選択では適応度から計算される確率に従って個体が選択されるのに対して、トーナメント選択では適応度の大小とトーナメントのサイズに従って個体が選択される。

6.2.5　その他の設計

遺伝的アルゴリズムを行う際に決めなければならないのは交叉則や突然変異則、選択則だけではない。通常は個体数 S、世代数の上限 G、交叉則における交叉率 P_c、突然変異率 P_m を定めなければならず、選択則によっては選択の圧力やトーナメントのサイズを定める必要もある。また、初期個体の生成を行う際に問題の特徴から優れた個体が生成できるのであれば必要な世代数を少くすることもできる。

一般に個体数 S や総世代数 G は大ければ最適解が得られやすいが、計算時間も必要になる。個体数 S は数十から数百くらいの値がよく用いられ、総世代数 G は問題やアルゴリズムの構成によるが求められる計算時間に合わせて設定される。交叉則は遺伝的アルゴリズムの主となる演算であるため、交叉率 P_c は 0.6 以上の大きめの値が用いられる。一方、突然変異率が高いとランダムに個体を生成するのと変わらないため、突然変異則 P_m は小さめの値として 0.1 以下とされる場合が多い。

また、よりよい解を得るために交叉則・突然変異則・選択則だけでなく別の操作を行う場合がある。それらの操作の例を説明する。

エリート保存 (elite preserving)　これまで示した選択則では個体を選ぶだけであったため、交叉則や突然変異則によって前の世代の個体群での個体と一致する個体は一般に次の世

代に表われない。これでは最良の個体が消えてしまう可能性があるうえに、次の世代で悪い個体しか生成されなかった場合に効率が悪くなる。その対策として選択則と合わせてエリート保存という操作を行う場合がある。

エリート保存では交叉則、突然変異則を適用する前に個体群 S から適応度の高い E 個の個体を除外し個体群 S' を作る。そして得られた個体群 S に対して交叉則、突然変異則を適用して S' と同じサイズの個体群を生成する。次の世代の個体群は S に交叉則、突然変異則を適用して得られた集合に除外した E 個の個体を加えることで生成する。最初に優秀な個体を除外しておくことで、交叉則や突然変異則によって悪い個体しか生成されなくても次の世代では優秀な個体が残っていることになるため、最終的によい解を得ることができると期待できる。

Steady-state 型　適応度比例選択やトーナメント選択を用いる場合は、一般に全ての個体に交叉則や突然変異則を適用する機会を与えることで、全ての個体が入れ替わる可能性を作る。しかし、交叉則や突然変異則を適用しうる個体数に制限をもうけることで 1 世代で入れ替わる個体数に上限を与える方法もある。このような方法で行われる遺伝的アルゴリズムを Steady-state な遺伝的アルゴリズムという。

6.2.6　数値計算例

6.2.7　アルゴリズムの改良

hoge()　hoge

hoge()　hoge

7　Memetic Algorithm

Memetic Algorithm

しみずスライド p.23 -> 999001 個ってのは本当は 99001(?) 個では? しみずスライド p.23 -> 突然変異 0.9 -> 遺伝子 1 つ 1 つでなく記号列全体に対して定義するから高めの 0.9

Memetic Algorithm をわざわざ GA と区別することはそうそうない-> だって大域的探索と局所的探索のバランスが大切ってのは進化計算の常識だし...

NN の構造の決定に対して GA を使うのはまぁよくあるし、服部研でもちょっと昔はやってた

GA, ACO や PSO はパラメータを決めるときにたまに便利なのよ

8 Ant Colony Optimization

Ant Colony Optimization

これは実際に蟻の行動を観察したひとが"蟻が最短経路みつけとるやん!"ってなったのがもとらしい

局所的な探索が苦手なので 2-opt でその対策をしたといえる

9 Particle Swarm Optimization

Particle Swarm Optimization

入力 $p^{(k)}$ をとるシステム

$$\mathbf{x}^{(k+1)} = M\mathbf{x}^{(k)} + Np^{(k)}$$

を考える。ある $\mathbf{h}^{(k)}$ を

$$\mathbf{h}^{(k)} = \begin{cases} M^k N & k \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

とし、$p^{(k)}$ との畳み込みを計算すると

$$\begin{aligned} \sum_{m=-\infty}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} &= \sum_{m=0}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} \\ &= \sum_{m=1}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} + \mathbf{h}^{(0)} p^{(k-1)} \end{aligned}$$

と書くことができる。ここで $m' = m - 1$ とすると

$$\begin{aligned} \sum_{m=-\infty}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} &= \sum_{m=1}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} + \mathbf{h}^{(0)} p^{(k-1)} \\ &= \sum_{m'=0}^{\infty} \mathbf{h}^{(m'+1)} p^{(k-(m'+1)-1)} + \mathbf{h}^{(0)} p^{(k-1)} \\ &= \sum_{m'=0}^{\infty} M^{m'+1} N p^{(k-m'-2)} + N p^{(k-1)} \\ &= M \sum_{m'=0}^{\infty} M^{m'} N p^{((k-1)-m'-1)} + N p^{(k-1)} \end{aligned}$$

となる。ここで

$$\mathbf{x}^{(k)} = \sum_{m=-\infty}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)}$$

とすれば

$$\begin{aligned}\mathbf{x}^{(k)} &= \sum_{m=-\infty}^{\infty} \mathbf{h}^{(m)} p^{(k-m-1)} \\ &= M \sum_{m'=0}^{\infty} M^{m'} N p^{((k-1)-m'-1)} + N p^{(k-1)} \\ &= M \sum_{m'=0}^{\infty} \mathbf{h}^{(m')} p^{((k-1)-m'-1)} + N p^{(k-1)} \\ &= M\mathbf{x}^{(k-1)} + N p^{(k-1)}\end{aligned}$$

となるため、このシステムはインパルス応答 $\mathbf{h}^{(k)}$ が

$$\mathbf{h}^{(k)} = \begin{cases} M^k N & k \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

である線形時不変システムである。また、

$$\sum_{k=-\infty}^{\infty} \mathbf{h}^{(k)} = \sum_{k=0}^{\infty} M^k N = \left(\sum_{k=0}^{\infty} M^k \right) N$$

となるため $\sum_{k=0}^{\infty} M^k$ が収束すればこのシステムは有界な入力に対して有界な出力を返す。

M が 2×2 の行列で非負実数 w, ϕ によって

$$M = \begin{pmatrix} w & \phi \\ -w & 1-\phi \end{pmatrix}$$

と書かれる場合を考える。まず、M の固有値を求める。特性方程式

$$\det(M - \lambda I) = \begin{vmatrix} w-\lambda & \phi \\ -w & 1-\phi-\lambda \end{vmatrix} = 0$$

を解くと M の固有値 λ_1, λ_2 は

$$\begin{aligned}\lambda_1 &= \frac{w - \phi + 1 + \sqrt{(w-\phi+1)^2 - 4w}}{2} \\ \lambda_2 &= \frac{w - \phi + 1 - \sqrt{(w-\phi+1)^2 - 4w}}{2}\end{aligned}$$

となる。固有値 λ_1, λ_2 それぞれに対する固有ベクトル $\mathbf{v}_1, \mathbf{v}_2$ は

$$\begin{aligned}\mathbf{v}_1 &= \begin{pmatrix} -\frac{w+\phi-1+\sqrt{(w-\phi+1)^2-4w}}{2w} \\ 1 \end{pmatrix} \\ \mathbf{v}_2 &= \begin{pmatrix} -\frac{w+\phi-1-\sqrt{(w-\phi+1)^2-4w}}{2w} \\ 1 \end{pmatrix}\end{aligned}$$

である。よって固有ベクトルが存在するためある行列 P を用いて

$$M = P^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} P$$

と書くことができ、

$$M^k = P^{-1} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}^k P$$

が成立する。したがって、$|\lambda_1| \leq 1, |\lambda_2| \leq 1$ であれば $\lim_{k\to\infty} M^k$ は収束する。λ_1, λ_2 が前述の条件を満たすのは

$$0 \leq w < 1,\ 0 \leq \phi < 2w + 2$$

であるため、この条件を満たせば $\sum_{k=0}^{\infty} M^k$ は収束し、システムが有界な入力に対して有界な出力を返す。