Sigmoid function から人工知能へ

— Google の人工知能を理解するために —

横浜市立大学・国際総合科学部・数学教室
藤井一幸

概要

このノートで、人工知能の研究で代表的な活性化関数である sigmoid function （これは非常に良い関数である）の種々の性質をリストする。次に、これを用いて多層のニューラルネットワーク (Deep Learning) の簡潔な説明をする。近い将来、人工知能の Deep Learning は大学生の必須の知識になるかも知れない。

このノートに於ける我々の目的は、Google の人工知能 DeepMind を理解するために必要な準備をすることである[1]。

[I] はじめに

正直に書くと、人工知能にはあまり興味はなかった。しかし 2015–2016 年に、Google の人工知能 **DeepMind** が囲碁の top （韓国人、名前は忘れた）を木端微塵（コッパミジン）にしたニュースに接して心の底から驚いた。

まさか、囲碁で、人間が、負けるとは！

チェスや将棋は時間の問題であったが、囲碁は当分人間の方に advantage があると思っていた。多くの人がそう思っていたに違いない。

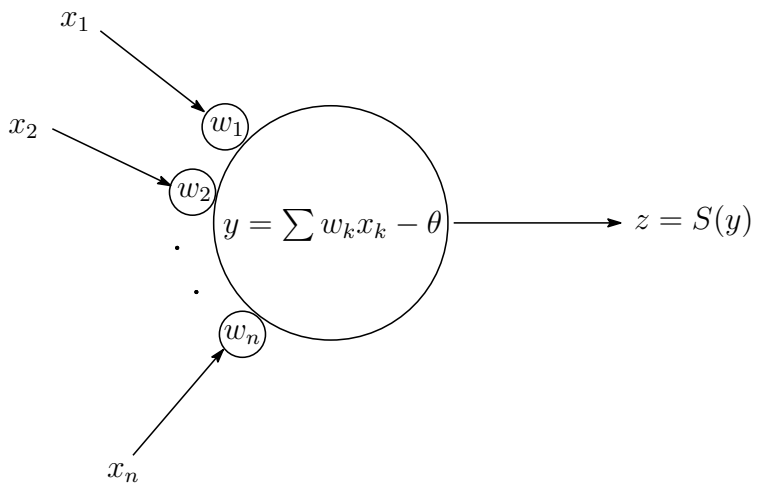
Google の人工知能 DeepMind とは一体何なのか？ 一体何をしているのか？ 血が騒ぐのである。

そのためにも人工知能の中核である Deep Learning を深くマスターしなければならない。このノートで Deep Learning を学生にもわかるように解説する。主に [2] : 5 章 及び [3] を参考にした。

[II] 準備：ニューロンモデル

少しだけ準備をしよう [6], [7]。ニューロンの数理モデルは以下の図のようになる：

[1] 本当に準備が出来たのか心もとないが。Google の猫 [1] を見よ

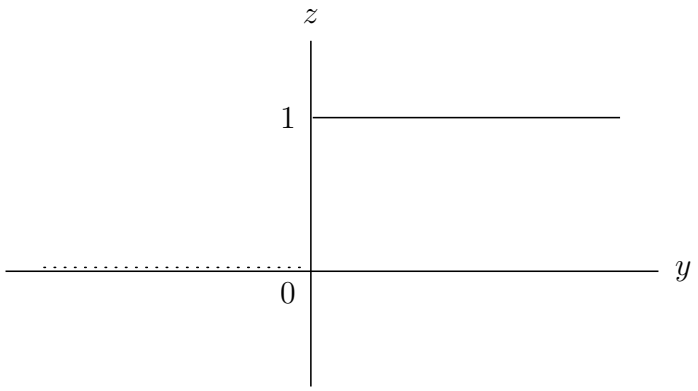


ここに、$x_1, x_2, \cdots, x_n$ はニューロンへの入力、$w_1, w_2, \cdots, w_n$ は**シナプス結合の荷重**、z はニューロンの出力である。また、θ は閾値（シキイ チ）である。

更に $S(y)$ は単位ステップ関数

$$S(y) = \begin{cases} 1 & y \geq 0 \\ 0 & y < 0 \end{cases}$$

で、原点を除けばヘヴィサイド関数 $Y(y)$ である。$S(y) = 1$ のとき細胞は点火する。



ヘヴィサイド関数 $Y(y)$ は

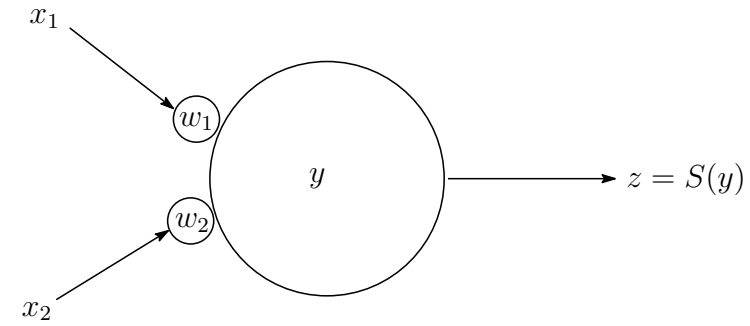
$$Y(y) = \begin{cases} 1 & y > 0 \\ 0 & y < 0 \end{cases}$$

又は、原点を埋めて

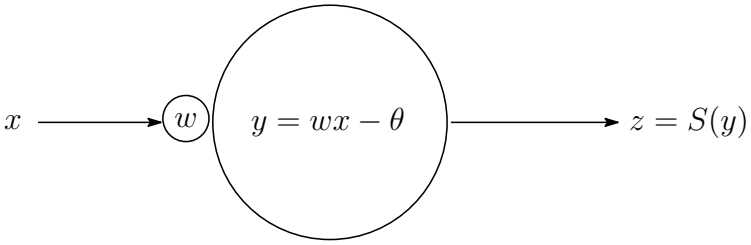
$$Y(y) = \begin{cases} 1 & y > 0 \\ \frac{1}{2} & y = 0 \\ 0 & y < 0 \end{cases}$$

である。

ここで、ニューロンモデルで、$\{w_1, w_2, w\}$ と $\{x_1, x_2, x\}$ 及び θ をうまく選べば論理演算が構成出来ることをコメントしておく。



$$y = w_1x_1 + w_2x_2 - \theta$$



以下 $a, b \in \mathbf{Z}_2 = \{0, 1\}$ とする。

(1) AND ゲートの構成

論理積 (AND) の真理値表は

a	b	AND
0	0	0
1	0	0
0	1	0
1	1	1

である。ここで

$$x_1 = a, x_2 = b,\ w_1 = w_2 = 1,\ \theta = 1.5$$

ととる。このとき

$$S(a + b - 1.5) = \min\{a, b\} = ab$$

となる。

(2) OR ゲートの構成

論理和 (OR) の真理値表は

a	b	OR
0	0	0
1	0	1
0	1	1
1	1	1

である。ここで

$$x_1 = a, x_2 = b,\ w_1 = w_2 = 1,\ \theta = 0.5$$

ととる。このとき

$$S(a + b - 0.5) = \max\{a, b\} = a + b - ab$$

となる。

(3) NOT ゲートの構成

否定論理 (NOT) の真理値表は

a	NOT
0	1
1	0

である。ここで

$$x_1 = a,\ w = -1,\ \theta = -0.5$$

ととる。このとき

$$S(-a + 0.5) = \bar{a} = 1 - a$$

となる。

(4) 排他的論理ゲート (XOR) の構成

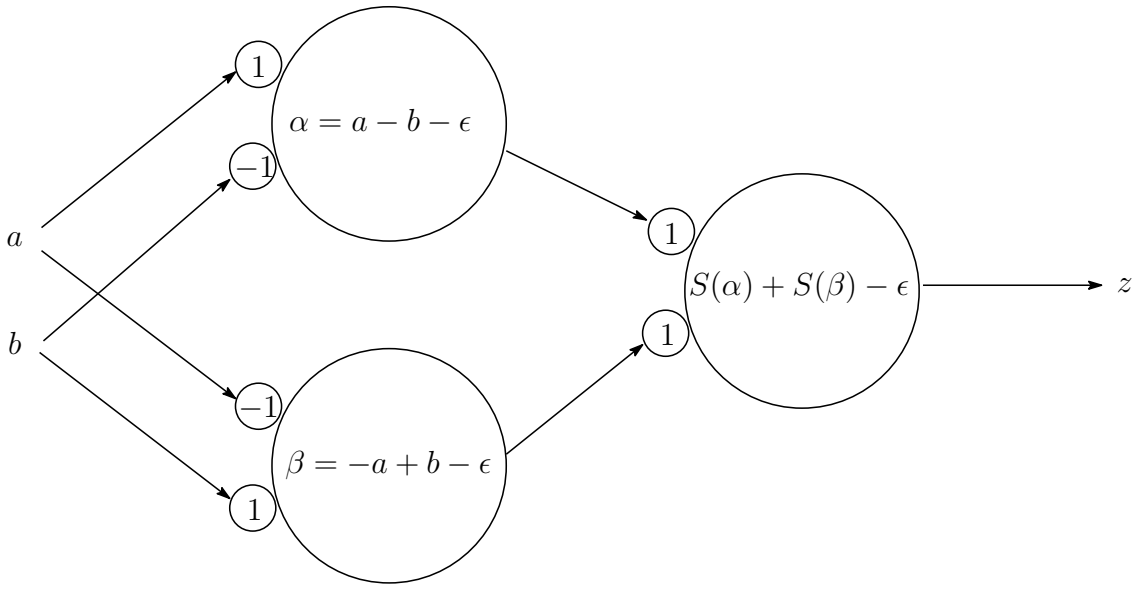
排他的論理ゲート (XOR) の真理値表は

a	b	XOR
0	0	0
1	0	1
0	1	1
1	1	0

である。上の表を式で表わせば

$$XOR(a, b) = a + b - 2ab \quad (a,\ b \in \mathbf{Z}_2)$$

とで、ニューロンモデルで構成すると



となる。即ち

$$z = S(S(\alpha) + S(\beta) - \epsilon), \quad 0 < \epsilon \ll 1$$

である（チェックせよ）。

単位ステップ関数のマズイところは、例えば

$$S(0.001) = 1, \quad S(-0.001) = 0$$

となるのである。従ってもっと fuzzy なものに変える必要があり、その一つの例が シグモイド関数 である。

[III] ステップ関数からシグモイド関数へ

誤差逆伝搬学習 (by Rumelhart) によるニューラルネットワークでは活性化関数の微分を用いるので、上の単位ステップ関数 $S(y)$ では意味をなさない。そこで用いられるのが以下のシグモイド関数 $\sigma(x)$ である。この関数を用いて 情報の非線形圧縮 を行うのである。

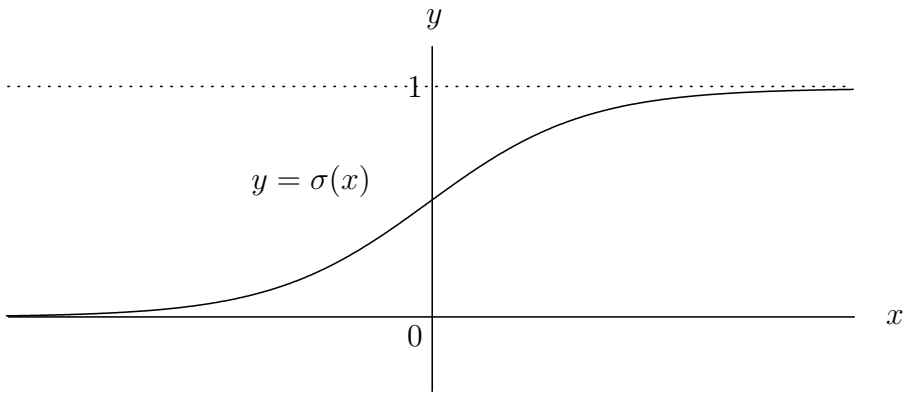
定義 シグモイド関数は

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (x \in \mathbf{R})$$

で与えられる。この関数は

$$\sigma(-\infty) = 0, \ \sigma(0) = \frac{1}{2}, \ \sigma(\infty) = 1$$

で、グラフは以下の通り ：



活性化関数を二値 $\{0,1\}$ から連続値 $[0,1]$ に変えるのは、ブール論理から量子論理に change するようなものである。そして量子論理は 重ね合わせ と consistent なのである。

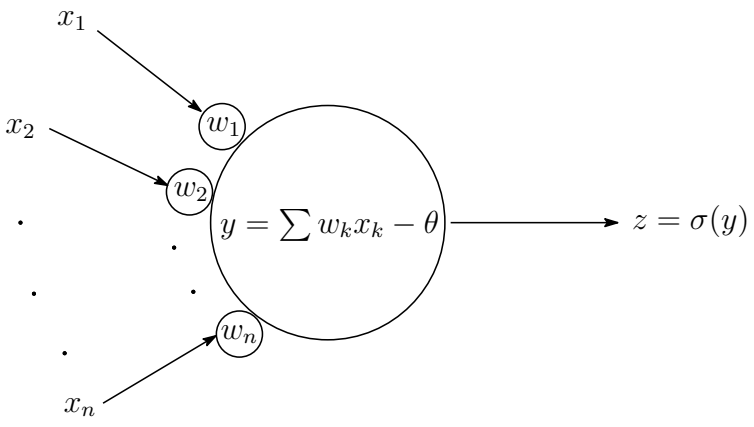
コメント 数学の世界では、これは standard logistic function と呼ばれている非常に有名な関数である。人工知能の世界では、sigmoid function と呼んでいる。

一般の logistic function $f(x)$ は、三つのパラメータを用いて

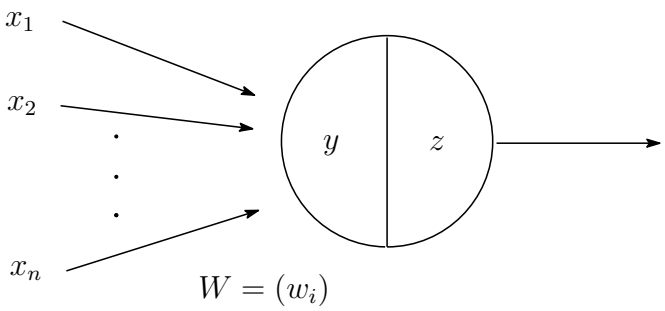
$$f(x) = \frac{L}{1 + e^{-\lambda(x - x_0)}}$$

で与えられる。$L = 1, \lambda = 1, x_0 = 0$ が standard logistic function である。
これがいかに重要な関数であるかについては、例えば藤井の lecture note [8] を見よ。

このとき、新しいニューロンの数理モデルは



となる。これを以後以下のように簡略化するが、とくに問題はないと思う。



以下シグモイド関数の性質をリストしてゆく（各自証明せよ)。

(1) 範囲

$$0 < \sigma(x) < 1$$

(2) パリティの関係式

$$\sigma(-x) = 1 - \sigma(x) \iff \sigma(-x) + \sigma(x) = 1$$

(3) 微分

$$\sigma' = \sigma(1-\sigma) \implies \sigma' \leq \frac{1}{4}$$

(4) 2 回微分

$$\sigma'' = \sigma(1-\sigma)(1-2\sigma)$$

(5) 変曲点

$$2\sigma(x) = 1 \implies x = 0$$

(6) 3 回微分

$$\sigma^{(3)} = \sigma(1-\sigma)(1-6\sigma+6\sigma^2)$$

(7) 4 回微分

$$\sigma^{(4)} = \sigma(1-\sigma)(1-2\sigma)(1-12\sigma+12\sigma^2)$$

(8) 逆関数

$$\sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right) \quad (0 < x < 1)$$

定義 情報理論のエントロピー関数（情報量）は

$$H(x) = -x\log x - (1-x)\log(1-x) \quad (0 < x < 1)$$

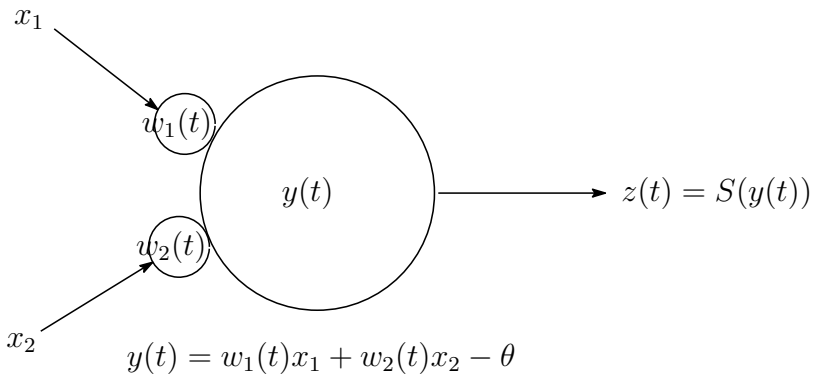
で与えられる。

(9) ある関係式

$$\sigma^{-1}(x) = -H'(x)$$

[IV] シナプス結合の荷重の決定

最初に簡単な例を用いてシナプス結合の荷重の決定方法を説明しておく。
For $t = 0, 1, 2, \cdots$ に対して $\{w_1(t), w_2(t)\}$ を次々に更新してゆく。



$$y(t) = w_1(t)x_1 + w_2(t)x_2 - \theta$$

更新方法は以下の通り ：

$$w_i(t) \longrightarrow w_i(t+1) = w_i(t) + \epsilon(d - z(t))x_i.$$

ここで

$$\theta = 0.5,\ x_1 = 0, x_2 = 1,\ d = 1,\ \epsilon = 0.2$$

ととる（θ は閾値、d は目標である教師信号、ϵ は学習係数）。

(0) $t = 0$ ここで、例えば

$$w_1(0) = 0.0,\ w_2(0) = 0.1$$

と取る。このとき

$$z(0) = S(0.0 \times 0 + 0.1 \times 1 - 0.5) = S(-0.4) = 0$$

である。

(1) $t = 1$ 変更は

$$\begin{aligned} w_1(1) &= w_1(0) + 0.2(1-0)0 = 0 \\ w_2(1) &= w_2(0) + 0.2(1-0)1 = 0.3 \end{aligned}$$

で、

$$z(1) = S(0 \times 0 + 0.3 \times 1 - 0.5) = S(-0.2) = 0$$

である。

(2) $t=2$ 変更は

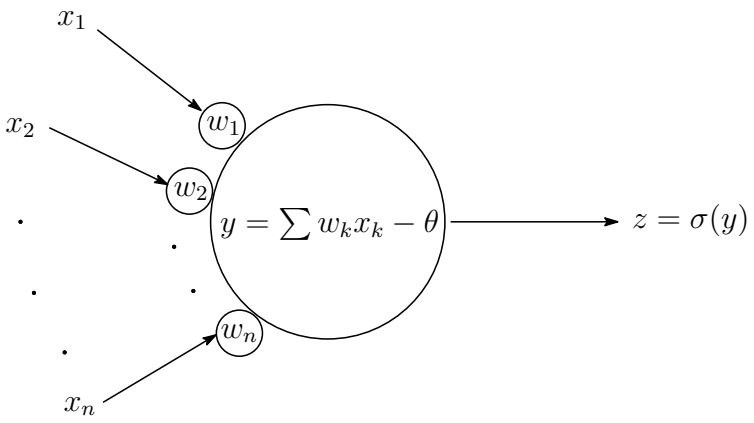
$$\begin{aligned} w_1(2) &= w_1(1)+0.2(1-0)0=0 \\ w_2(2) &= w_2(1)+0.2(1-0)1=0.5 \end{aligned}$$

で

$$z(2)=S(0\times 0+0.5\times 1-0.5)=S(0)=1$$

となる。2 step 後に出力信号が教師信号に一致した。

わかりやすくするために、もう一度新しいニューロンモデルを書いておく。



一番重要なことはシナプス結合の荷重を決定することである。そのために m 個の入力データと教師信号を準備する。今は $m<n$ としておく。以下では計算を見やすくするため $\theta=0$ とおく。

$j=1,2,\cdots,m$ に対して

入力データ : $\mathbf{x}^{(j)}=(x_1^{(j)},x_2^{(j)},\cdots,x_n^{(j)})^T$

教師信号 : $d^{(j)}$

とおく。入力データに対してその出力が、与えられた信号（：教師信号と言う）と（可能な限り）一致するようにシナプス結合の荷重を決めたい。そのために、誤差の 2 乗和

$$E=E(w)=\frac{1}{2}\sum_{j=1}^{m}(d^{(j)}-z^{(j)})^2$$

が最小になるように w を決めればよい。ここに

$$z^{(j)}=\sigma(y^{(j)})=\sigma\left(\sum_{k=1}^{n}w_k x_k^{(j)}\right)$$

である。これを繰り返しの学習で実現してゆく。そのために $w = w(t)$ の関数として

$$E = E(w) = E(w(t)) \ ; \ z^{(j)}(t) = \sigma(y^{(j)}(t)) = \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right)$$

とおく（t は繰り返しの回数で, $t = 0, 1, 2, \cdots$ である）。初期値 $\mathbf{w}(0)$ は適当に決めておく。

[A] 誤差逆伝搬学習法

この方法は、$\{w_1(t), w_2(t), \cdots, w_n(t)\}$ の $w_k(t)$ を**勾配降下法**を用いて

$$w_k(t+1) = w_k(t) - \epsilon \frac{\partial E(w(t))}{\partial w_k(t)} \quad (0 < \epsilon < 1)$$

と変更して

$$E = E(w(t+1))$$

とし、この操作を繰り返す（適当な回数で打ち止め）:

$$E(w(0)) \implies E(w(1)) \implies \cdots \implies E(w(T)) \equiv E(w)$$

計算してみよう。$E = E(w)$ として公式 (3) より（t を省略して）

$$\begin{aligned}
\frac{\partial E}{\partial w_k} &= \sum_{j=1}^{m} (d^{(j)} - z^{(j)}) \left(-\frac{\partial z^{(j)}}{\partial w_k}\right) \\
&= -\sum_{j=1}^{m} (d^{(j)} - z^{(j)}) \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right) \left\{1 - \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\right\} x_k^{(j)} \\
&= -\sum_{j=1}^{m} (d^{(j)} - z^{(j)}) x_k^{(j)} \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right) \left\{1 - \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\right\}
\end{aligned}$$

なので

$$\begin{aligned}
w_k(t+1) &= w_k(t) - \epsilon \frac{\partial E(w(t))}{\partial w_k(t)} \\
&= w_k(t) + \epsilon \sum_{j=1}^{m} (d^{(j)} - z^{(j)}(t)) x_k^{(j)} \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right) \left\{1 - \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right)\right\}
\end{aligned}$$

となる。これを代入すると

$$z^{(j)}(t) = \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right) \implies z^{(j)}(t+1) = \sigma\left(\sum_{k=1}^{n} w_k(t+1) x_k^{(j)}\right)$$

で、誤差

$$E = E(w(t+1)) = \frac{1}{2} \sum_{j=1}^{m} (d^{(j)} - z^{(j)}(t+1))^2$$

は相当に複雑な形になる。

コメント 簡単な例で説明しよう。x は $0 \leq x \leq 1$ と仮定する。$E(w)$ を

$$E = E(w) \equiv \frac{1}{2}(d - \sigma(wx))^2$$

とおく。微分を計算すると、公式 (3) より

$$\frac{\partial E}{\partial w} = -(d - \sigma(wx))\sigma'(wx)x = -x(d - \sigma(wx))\sigma(wx)\,(1 - \sigma(wx))$$

なので、

$$w \longrightarrow \tilde{w} = w - \epsilon\frac{\partial E}{\partial w} = w + \epsilon x(d - \sigma(wx))\sigma(wx)\,(1 - \sigma(wx))$$

となり、新しい誤差は

$$E(\tilde{w}) = \frac{1}{2}(d - \sigma(\tilde{w}x))^2 = \frac{1}{2}[d - \sigma\{wx + \epsilon x^2(d - \sigma(wx))\sigma(wx)\,(1 - \sigma(wx))\}]^2$$

となる。

関数 $\sigma(x)$ は単調増加なので $d - \sigma(wx) > 0$ である限り

$$\sigma\{wx + \epsilon x^2(d - \sigma(wx))\sigma(wx)\,(1 - \sigma(wx))\} > \sigma(wx)$$

で、従って

$$E(w) > E(\tilde{w})$$

となる。ではどのくらい改善されているのだろう？

ϵx^2 は充分小さいので、Taylor 展開で 1 次まで取ると

$$\begin{aligned} \sigma(\tilde{w}x) &\approx \sigma(wx) + \sigma'(wx)\epsilon x^2(d - \sigma(wx))\sigma(wx)\,(1 - \sigma(wx)) \\ &= \sigma(wx) + \epsilon x^2(d - \sigma(wx))\sigma(wx)^2\,(1 - \sigma(wx))^2 \end{aligned}$$

となる（高次の項は十分に小さいと仮定して良い）。故に

$$d - \sigma(\tilde{w}x) \approx (d - \sigma(wx))\left\{1 - \epsilon x^2\sigma(wx)^2\,(1 - \sigma(wx))^2\right\}$$

より

$$\begin{aligned} E(\tilde{w}) &\approx \frac{1}{2}(d - \sigma(wx))^2\left\{1 - \epsilon x^2\sigma(wx)^2\,(1 - \sigma(wx))^2\right\}^2 \\ &= E(w)\left\{1 - \epsilon x^2\sigma(wx)^2\,(1 - \sigma(wx))^2\right\}^2 \end{aligned}$$

で、$E(w) > E(\tilde{w})$ がわかる。

何回か繰り返せば零に近づくのであるが、問題点（？）があるので説明しよう。ϵ は小さいので $\epsilon = 0.2 = 1/5$ と取る（ある教科書では そうなっている）。

(i) $x = 1$ の場合 ： 公式 (3) より

$$1 - \epsilon\sigma^2(1-\sigma)^2 = 1 - \epsilon\{\sigma(1-\sigma)\}^2 \geq 1 - \frac{1}{5} \times \frac{1}{16} = \frac{79}{80}$$

なので

$$\left\{1 - \epsilon\sigma^2(1-\sigma)^2\right\}^2 \geq \left(\frac{79}{80}\right)^2 \approx 0.975$$

となる。即ち

$$E(w) > E(\tilde{w}) > 0.975E(w)$$

で、最大 2.5 % しか改良されないのである。更に

$$(0.975)^{50} \approx 0.28, \quad (0.975)^{100} \approx 0.08$$

なので、50 乗しても元の（最大）28 %程度である。

(ii) $x = \frac{1}{2}$ の場合 ： 同様にして

$$1 - \frac{\epsilon}{4}\sigma^2(1-\sigma)^2 = 1 - \frac{\epsilon}{4}\{\sigma(1-\sigma)\}^2 \geq 1 - \frac{1}{20} \times \frac{1}{16} = \frac{319}{320}$$

なので

$$\left\{1 - \frac{\epsilon}{4}\sigma^2(1-\sigma)^2\right\}^2 \geq \left(\frac{319}{320}\right)^2 \approx 0.994$$

となる。即ち

$$E(w) > E(\tilde{w}) > 0.994E(w)$$

で、最大 0.6 % しか改良されないのである。更に

$$(0.994)^{100} \approx 0.55, \quad (0.994)^{200} \approx 0.30$$

なので、200 乗しても元の（最大）30 %程度である。

大規模なシステムを何回も繰り返すと、丸め誤差の問題が起きてくるハズである。

[A2] 極値の数

ここで、このシステムの極値の数を評価しよう。結論から言えば、良いデータの場合最小値が一つのみである（他の極値は存在しない）。

入力データに関する仮定をおく。

仮定（良いデータ） $m < n$ として

$$\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}\right\}$$

は一次独立である。

特に問題はないと思う（下のコメントを見よ)。この仮定より、ある $k_1 < k_2 < \cdots < k_m$ が存在して

$$\begin{vmatrix} x_{k_1}^{(1)} & x_{k_1}^{(2)} & \cdots & x_{k_1}^{(m)} \\ x_{k_2}^{(1)} & x_{k_2}^{(2)} & \cdots & x_{k_2}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_m}^{(1)} & x_{k_m}^{(2)} & \cdots & x_{k_m}^{(m)} \end{vmatrix} \neq 0$$

と出来る。

$$E = \frac{1}{2}\sum_{j=1}^{m}(d^{(j)} - z^{(j)})^2 = \frac{1}{2}\sum_{j=1}^{m}(d^{(j)} - \sigma(y^{(j)}))^2, \quad \sigma(y^{(j)}) = \sum_{k=1}^{n} w_k x_k^{(j)}$$

($\theta = 0$ for simplicity) を思い出して計算をしよう。$1 \leq k \leq n$ に対して

$$0 = \frac{\partial E}{\partial w_k} = -\sum_{j=1}^{m}(d^{(j)} - z^{(j)})\sigma'(y^{(j)})x_k^{(j)}$$

で、簡単のため

$$A^{(j)} = (d^{(j)} - z^{(j)})\sigma'(y^{(j)})$$

とおく。方程式は簡潔に

$$\sum_{j=1}^{m} A^{(j)} x_k^{(j)} = 0 \quad \text{for} \quad 1 \leq k \leq n$$

と表せる。このとき上述の仮定から

$$\sum_{j=1}^{m} x_k^{(j)} A^{(j)} = 0 \quad \text{for} \quad k = k_1, k_2, \cdots, k_m$$

で、行列を用いて表すと

$$\begin{pmatrix} x_{k_1}^{(1)} & x_{k_1}^{(2)} & \cdots & x_{k_1}^{(m)} \\ x_{k_2}^{(1)} & x_{k_2}^{(2)} & \cdots & x_{k_2}^{(m)} \\ \vdots & \vdots & \cdots & \vdots \\ x_{k_m}^{(1)} & x_{k_m}^{(2)} & \cdots & x_{k_m}^{(m)} \end{pmatrix} \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(m)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

となる。行列の行列式は nonzero なので

$$\begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(m)} \end{pmatrix} = \begin{pmatrix} (d^{(1)} - z^{(1)})\sigma'(y^{(1)}) \\ (d^{(2)} - z^{(2)})\sigma'(y^{(2)}) \\ \vdots \\ (d^{(m)} - z^{(m)})\sigma'(y^{(m)}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

で、更に $\sigma'(y) \neq 0$ なので、全ての $j = 1, 2, \cdots, m$ に対して

$$d^{(j)} = \sigma(y^{(j)})$$

となる。即ち、極値はただ一つで、最小値 $E = 0$ のみである。従って w_k の値を 勾配降下法 で次々に修正してゆくとき、極小値に trap されることはない。

逆に一次独立性をチェックしていないとき、誤差 E が動かなくなったら極小値に trap されている可能性が高く、入力データ $\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}\right\}$ が一次従属であることを示唆している。このときは入力データの一部を別のデータに変えなければならない。

では、どのようにして一次独立な入力データを選ぶのかを説明しておく。いま $n > m$ で n は非常に大きいとする。もし m が十分に小さければ、入力データ

$$\left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}\right\}$$

に対して $m \times m$ Gram 行列式

$$G(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}) = \begin{vmatrix} \langle \mathbf{x}^{(1)} | \mathbf{x}^{(1)} \rangle & \langle \mathbf{x}^{(1)} | \mathbf{x}^{(2)} \rangle & \cdots & \langle \mathbf{x}^{(1)} | \mathbf{x}^{(m)} \rangle \\ \langle \mathbf{x}^{(2)} | \mathbf{x}^{(1)} \rangle & \langle \mathbf{x}^{(2)} | \mathbf{x}^{(2)} \rangle & \cdots & \langle \mathbf{x}^{(2)} | \mathbf{x}^{(m)} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}^{(m)} | \mathbf{x}^{(1)} \rangle & \langle \mathbf{x}^{(m)} | \mathbf{x}^{(2)} \rangle & \cdots & \langle \mathbf{x}^{(m)} | \mathbf{x}^{(m)} \rangle \end{vmatrix}$$

を計算する。これが nonzero ならば、入力データは一次独立である。これをチェックするのは（m が十分に小さければ）問題はない。

少し説明しよう。簡単にするため $m = 3$ の場合を考え、更に

$$\mathbf{x}^{(1)} = \mathbf{a},\ \mathbf{x}^{(2)} = \mathbf{b},\ \mathbf{x}^{(3)} = \mathbf{c}$$

とおく。

$$G(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{vmatrix} \langle \mathbf{a} | \mathbf{a} \rangle & \langle \mathbf{a} | \mathbf{b} \rangle & \langle \mathbf{a} | \mathbf{c} \rangle \\ \langle \mathbf{b} | \mathbf{a} \rangle & \langle \mathbf{b} | \mathbf{b} \rangle & \langle \mathbf{b} | \mathbf{c} \rangle \\ \langle \mathbf{c} | \mathbf{a} \rangle & \langle \mathbf{c} | \mathbf{b} \rangle & \langle \mathbf{c} | \mathbf{c} \rangle \end{vmatrix}$$

であった。今 $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ を一次従属と仮定する。このとき、例えば

$$\mathbf{c} = \alpha \mathbf{a} + \beta \mathbf{b} \quad (\alpha, \beta \text{ は実数})$$

と表せる。これを上式に代入して計算すると

$$G(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{vmatrix} \langle \mathbf{a} | \mathbf{a} \rangle & \langle \mathbf{a} | \mathbf{b} \rangle & \langle \mathbf{a} | \mathbf{c} \rangle \\ \langle \mathbf{b} | \mathbf{a} \rangle & \langle \mathbf{b} | \mathbf{b} \rangle & \langle \mathbf{b} | \mathbf{c} \rangle \\ \alpha \langle \mathbf{a} | \mathbf{a} \rangle + \beta \langle \mathbf{b} | \mathbf{a} \rangle & \alpha \langle \mathbf{a} | \mathbf{b} \rangle + \beta \langle \mathbf{b} | \mathbf{b} \rangle & \alpha \langle \mathbf{a} | \mathbf{c} \rangle + \beta \langle \mathbf{b} | \mathbf{c} \rangle \end{vmatrix}$$

で、行列式の基本演算より

$$G(\mathbf{a},\mathbf{b},\mathbf{c}) = \begin{vmatrix} \langle \mathbf{a}|\mathbf{a}\rangle & \langle \mathbf{a}|\mathbf{b}\rangle & \langle \mathbf{a}|\mathbf{c}\rangle \\ \langle \mathbf{b}|\mathbf{a}\rangle & \langle \mathbf{b}|\mathbf{b}\rangle & \langle \mathbf{b}|\mathbf{c}\rangle \\ 0 & 0 & 0 \end{vmatrix} = 0$$

となる。即ち

$$\{\mathbf{a},\mathbf{b},\mathbf{c}\} \text{ は一次従属 } \Longrightarrow\ G(\mathbf{a},\mathbf{b},\mathbf{c}) = 0$$

である。これの対偶を取ると

$$G(\mathbf{a},\mathbf{b},\mathbf{c}) \neq 0\ \Longrightarrow\ \{\mathbf{a},\mathbf{b},\mathbf{c}\} \text{ は一次独立}$$

となる。

コメント $n > m$ とし $M(n,m;\mathbf{R})$ を $\mathbf{R}$ 上の $n\times m$ 行列全体とし、その部分集合 $M_{n,m}(\mathbf{R})$ を

$$M_{n,m}(\mathbf{R}) = \left\{(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\cdots,\mathbf{x}^{(m)}) \in M(n,m;\mathbf{R}) \mid \left\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},\cdots,\mathbf{x}^{(m)}\right\} \text{ は一次独立}\right\}$$

とおく。$M_{n,m}(\mathbf{R})$ は $M(n,m;\mathbf{R})$ の中で open dense であることが知られている。即ち、$\mathbf{R}^n$ からランダムに m 個ベクトルを選べば、ほとんど全て一次独立である。

[B] 線形代数的学習法（藤井流）

ヒトマネは好きでないので、自己流でやってみる。そのため、各 $j = 1,2,\cdots,m$ に対して

$$E = 0\ \Longrightarrow\ d^{(j)} = z^{(j)} = \sigma(y^{(j)})$$

と取る（これが一番簡単）。このとき公式 (8) より

$$\log\left(\frac{d^{(j)}}{1-d^{(j)}}\right) = \sigma^{-1}(d^{(j)}) = y^{(j)} = \sum_{k=1}^{n} w_k x_k^{(j)} = \sum_{k=1}^{n} x_k^{(j)} w_k$$

となる。これを連立方程式で書き変えると

$$\begin{cases} x_1^{(1)}w_1 + x_2^{(1)}w_2 + \cdots + x_n^{(1)}w_n = \log\left(\frac{d^{(1)}}{1-d^{(1)}}\right), \\ x_1^{(2)}w_1 + x_2^{(2)}w_2 + \cdots + x_n^{(2)}w_n = \log\left(\frac{d^{(2)}}{1-d^{(2)}}\right), \\ \cdots \\ x_1^{(m)}w_1 + x_2^{(m)}w_2 + \cdots + x_n^{(m)}w_n = \log\left(\frac{d^{(m)}}{1-d^{(m)}}\right) \end{cases}$$

となる。又は行列とベクトルを用いて

$$\begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \cdot & \cdot & \cdots & \cdot \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \log\left(\frac{d^{(1)}}{1-d^{(1)}}\right) \\ \log\left(\frac{d^{(2)}}{1-d^{(2)}}\right) \\ \vdots \\ \log\left(\frac{d^{(m)}}{1-d^{(m)}}\right) \end{pmatrix}$$

と表せる（かなり美しい形になる[2]）。

$m < n$ のとき、full rank と仮定してよいので、m 個の $\{\tilde{w}_{\tau(1)}, \cdots, \tilde{w}_{\tau(m)}\}$ が決まる。残りの $n-m$ 個は $T=0$ のままである。即ち、シナプス結合の変更 $\{w_j\} \longrightarrow \{\tilde{w}_j\}$ は

$$\begin{cases} \tilde{w}_j = \text{the solution} & \text{for } \ j = \tau(1), \tau(2), \cdots, \tau(m) \\ \tilde{w}_j = w_j & \text{for } \ j \in \{1, 2, \cdots, n\} \setminus \{\tau(1), \tau(2), \cdots, \tau(m)\} \end{cases}$$

となる。我々の場合は $T=1$ で終了する。

コメント この手法は自然で、学生達にとってわかりやすいと思う。何故今まで知られていないのか不思議である（もっとも、全ての文献をチェックしたわけではないが）。

最後にミニバッチ（minibatch）を用いる手法を説明しておく（これは技術的に重要である）。一般に入力データの数 M は huge である。そうでないと deep な学習は出来ないハズ。このとき $M \gg n$ となり線形代数的学習法は破綻する。しかし、物事を一気に進める必要はないのである。

毎回 M から**ランダム**に少数の m 個を選び、上述の線形代数的学習法を行う。この操作を K 回繰り返し、シナプス結合の荷重を決めるのである。これを**ミニバッチ** (minibatch) と呼ぶ。

問題点 問題点は、M と K の関係が clear でないことである。例えば、数論の Gauss 記号を用いて $[M/m] + 1 = K$ か？

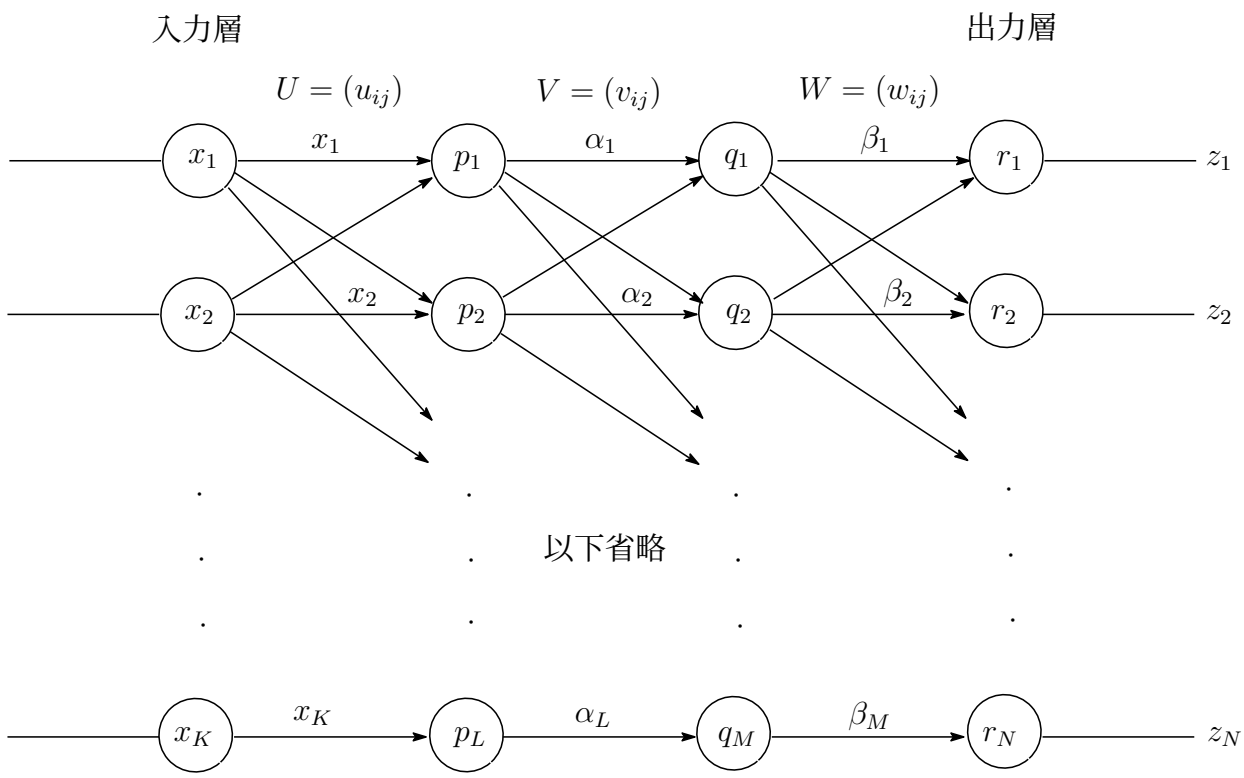
[V] Deep Learning 入門（教師有学習）

人工知能の heart である Deep Learning （深層学習）の簡潔な紹介をする。深層学習は一般に、入力層（K 個）、（複数の）中間層（$M_1, M_2, \cdots, N$ 個）、出力層（N 個）よりなるニューラルネットワーク・システムである。

[A] 通常の方法（逆伝播学習法）

まず、通常の方法を通常の図を用いて説明する。以下の図を見よ。

[2]数学者にとって**美しい**と言うことは、重要な driving force なのである



記号を説明しよう。シナプス結合荷重は、行列

$$\begin{aligned} U &= (u_{ij}) \ : \ L \times K \text{ 行列} \\ V &= (v_{ij}) \ : \ M \times L \text{ 行列} \\ W &= (w_{ij}) \ : \ N \times M \text{ 行列} \end{aligned}$$

で表わされる（これを “教育 ”することが目標である）。

シナプス結合は、出力層で

$$\begin{aligned} r_i &= \sum_{j=1}^{M} w_{ij}\beta_j - \theta, \quad 1 \le i \le N \\ z_i &= \sigma(r_i) \end{aligned}$$

入力層で

$$\begin{aligned} q_i &= \sum_{j=1}^{L} v_{ij}\alpha_j - \theta, \quad 1 \le i \le M \\ \beta_i &= \sigma(q_i) \end{aligned}$$

及び

$$\begin{aligned} p_i &= \sum_{j=1}^{K} u_{ij}x_j - \theta, \quad 1 \le i \le L \\ \alpha_i &= \sigma(p_i) \end{aligned}$$

となる。

ここで、教師信号ベクトルを

$$\mathbf{d} = (d_1, d_2, \cdots, d_N)^T$$

とおくと、２乗誤差は

$$E = E(U, V, W) = \frac{1}{2}\sum_{k=1}^{N}(d_k - z_k)^2$$

で与えられる。

このシステムの “時間発展”を考えよう。離散時間を $t = 0, 1, 2, \cdots$ として

$$\begin{aligned} t &\longrightarrow t+1 \\ U(t) &\longrightarrow U(t+1) \\ V(t) &\longrightarrow V(t+1) \\ W(t) &\longrightarrow W(t+1) \\ E(t) &\longrightarrow E(t+1) \end{aligned}$$

を決定し、t が十分大きくなると

$$E(t) \approx 0$$

を期待するのである（学習終了！）。

いま、t steps まで構成されていると仮定する：

$$E(t) = \frac{1}{2}\sum_{k=1}^{N}(d_k - z_k(t))^2.$$

このとき、シナプス結合荷重の時間発展は、**勾配降下法**を用いて

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial w_{ij}(t)}, \\ v_{ij}(t+1) &= v_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial v_{ij}(t)}, \\ u_{ij}(t+1) &= u_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial u_{ij}(t)} \end{aligned}$$

で与えられる。計算をする前に合成関数の微分をコメントしておく。

コメント 例で説明する。合成関数

$$x \longrightarrow f(x) \longrightarrow g(f(x)) \longrightarrow E(g(f(x)))$$

を考えよう。このとき微分公式は

$$\begin{aligned} \frac{dE}{dx} &= \frac{dE(g(f(x)))}{dg(f(x))}\frac{dg(f(x))}{df(x)}\frac{df(x)}{dx} \\ &= E'(g(f(x)))g'(f(x))f'(x) \end{aligned}$$

となる。合成関数は複雑でも、その微分は比較的簡単なのである。

余計なコメント シグモイド関数 $\sigma(x)$ の微分公式は

$$\sigma' = \sigma(1-\sigma)$$

であった。これに別証を与えておく。定義より

$$1 + e^{-x} = \frac{1}{\sigma} \iff e^{-x} = \frac{1}{\sigma} - 1 = \frac{1-\sigma}{\sigma}$$

である。これを微分すると

$$-e^{-x} = -\frac{\sigma'}{\sigma^2}$$

で、整理すると

$$\sigma' = \sigma^2 e^{-x} = \sigma^2 \frac{1-\sigma}{\sigma} = \sigma(1-\sigma)$$

となる。

計算を始めよう。

(1) $W(t) = (w_{ij}(t))$ の case : 計算すると

$$\begin{aligned} \frac{\partial E(t)}{\partial w_{ij}(t)} &= \frac{\partial E(t)}{\partial z_i(t)} \frac{\partial z_i(t)}{\partial r_i(t)} \frac{\partial r_i(t)}{\partial w_{ij}(t)} \\ &= -(d_i - z_i(t))\sigma'(r_i(t))\beta_j(t) \\ &= -(d_i - z_i(t))\sigma(r_i(t))(1-\sigma(r_i(t)))\beta_j(t) \end{aligned}$$

となり、

$$w_{ij}(t+1) = w_{ij}(t) + \epsilon(d_i - z_i(t))\sigma(r_i(t))(1-\sigma(r_i(t)))\beta_j(t)$$

を得る。

(2) $V(t) = (v_{ij}(t))$ の case : 計算すると

$$\begin{aligned} \frac{\partial E(t)}{\partial v_{ij}(t)} &= \sum_{k=1}^{N} \frac{\partial E(t)}{\partial z_k(t)} \frac{\partial z_k(t)}{\partial r_k(t)} \frac{\partial r_k(t)}{\partial \beta_i(t)} \frac{\partial \beta_i(t)}{\partial q_i(t)} \frac{\partial q_i(t)}{\partial v_{ij}(t)} \\ &= -\sum_{k=1}^{N} (d_k - z_k(t))\sigma'(r_k(t))w_{ki}(t)\sigma'(q_i(t))\alpha_j(t) \\ &= -\sum_{k=1}^{N} (d_k - z_k(t))\sigma(r_k(t))(1-\sigma(r_k(t)))w_{ki}(t)\sigma(q_i(t))(1-\sigma(q_i(t)))\alpha_j(t) \end{aligned}$$

となり、

$$v_{ij}(t+1) = v_{ij}(t) + \epsilon \sum_{k=1}^{N} (d_k - z_k(t))\sigma(r_k(t))(1-\sigma(r_k(t)))w_{ki}(t)\sigma(q_i(t))(1-\sigma(q_i(t)))\alpha_j(t)$$

を得る。

(3) $U(t) = (u_{ij}(t))$ の case : 計算すると

$$\begin{aligned}\frac{\partial E(t)}{\partial u_{ij}(t)} &= \sum_{l=1}^{N}\sum_{k=1}^{M}\frac{\partial E(t)}{\partial z_l(t)}\frac{\partial z_l(t)}{\partial r_l(t)}\frac{\partial r_l(t)}{\partial \beta_k(t)}\frac{\partial \beta_k(t)}{\partial q_k(t)}\frac{\partial q_k(t)}{\partial \alpha_i(t)}\frac{\partial \alpha_i(t)}{\partial p_i(t)}\frac{\partial p_i(t)}{\partial u_{ij}(t)} \\ &= -\sum_{l=1}^{N}\sum_{k=1}^{M}(d_l - z_l(t))\sigma'(r_l(t))w_{lk}(t)\sigma'(q_k(t))v_{ki}(t)\sigma'(p_i(t))x_j\end{aligned}$$

となり、

$$\begin{aligned}u_{ij}(t+1) &= u_{ij}(t) + \epsilon\sum_{l=1}^{N}\sum_{k=1}^{M}(d_l - z_l(t))\sigma(r_l(t))(1-\sigma(r_l(t)))w_{lk}(t)\times \\ &\qquad \sigma(q_k(t))(1-\sigma(q_k(t)))v_{ki}(t)\sigma(p_i(t))(1-\sigma(p_i(t)))x_j\end{aligned}$$

を得る。

今までの計算は入力データは一つであったが、現実は非常に多くのデータがある。それらを全て考慮する必要がある。いまその個数を A としよう。
$1 \le a \le A$ として

$$\begin{aligned}&\text{入力データ}\quad \mathbf{x}^{(a)} = \left(x_1^{(a)}, x_2^{(a)}, \cdots, x_K^{(a)}\right),\ \mathbf{d}^{(a)} = \left(d_1^{(a)}, d_2^{(a)}, \cdots, d_N^{(a)}\right), \\ &\text{出力データ}\quad \mathbf{z}^{(a)} = \left(z_1^{(a)}, z_2^{(a)}, \cdots, z_N^{(a)}\right).\end{aligned}$$

これらを全て入力すると、t steps での誤差関数は

$$E(t) = \sum_{a=1}^{A} E^{(a)}(t)$$

where

$$E^{(a)}(t) = \frac{1}{2}||\mathbf{d}^{(a)} - \mathbf{z}^{(a)}(t)||^2 = \frac{1}{2}\sum_{k=1}^{N}\left(d_k^{(a)} - z_k^{(a)}(t)\right)^2$$

となる。この場合計算は簡単で、例えば

$$w_{ij}(t+1) = w_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial w_{ij}(t)} = w_{ij}(t) - \epsilon\sum_{a=1}^{A}\frac{\partial E^{(a)}(t)}{\partial w_{ij}(t)}$$

なので、単に

$$w_{ij}(t+1) = w_{ij}(t) + \epsilon\sum_{a=1}^{A}(d_i^{(a)} - z_i^{(a)}(t))\sigma(r_i^{(a)}(t))(1-\sigma(r_i^{(a)}(t)))\beta_j^{(a)}(t)$$

となる。$v_{ij}(t)$ も $u_{ij}(t)$ の計算も全く同様である。

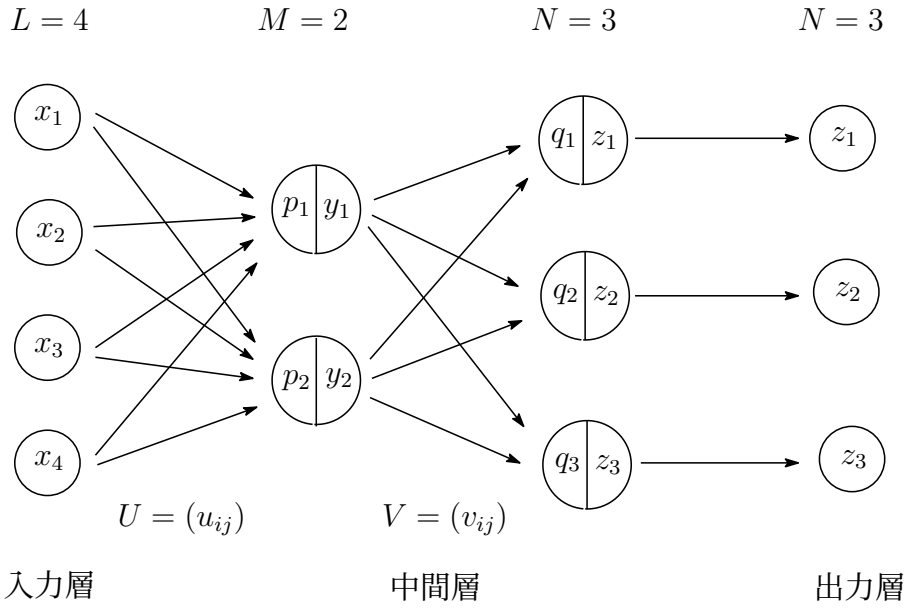
入力データ数 A が huge なとき、この時間発展の計算はコンピュータに非常な負荷を与えるであろう。それを避けるために minibatch を用いればよい。

[A2] 極値の数

もう一度重要な点を書いておく。このシステムには、極値は最小値 $L = 0$ のみである。従って、他の極値（極小値や極大値）は存在しない。

[B] 藤井流

一般論を書くのは大変なので、以下に簡単な例で説明する。



中間層 I

$$\begin{aligned} p_1 &= u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + u_{14}x_4 - \theta \longrightarrow \sigma(p_1) = y_1 \\ p_2 &= u_{21}x_1 + u_{22}x_2 + u_{23}x_3 + u_{24}x_4 - \theta \longrightarrow \sigma(p_2) = y_2 \end{aligned}$$

シナプス結合

$$U = U(u_{ij}) = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \end{pmatrix}$$

中間層 II

$$q_1 = v_{11}y_1 + v_{12}y_2 - \theta \longrightarrow \sigma(q_1) = z_1$$

$$
\begin{aligned}
q_2 &= v_{21}y_1 + v_{22}y_2 - \theta \longrightarrow \sigma(q_2) = z_2 \\
q_3 &= v_{31}y_1 + v_{32}y_2 - \theta \longrightarrow \sigma(q_3) = z_3
\end{aligned}
$$

シナプス結合

$$
V = V(v_{ij}) = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ v_{31} & v_{32} \end{pmatrix}
$$

誤差関数

$$
E = E(U, V) = \frac{1}{2}\sum_{j=1}^{3}(z_j - d_j)^2
$$

ここで $j = 1, 2, 3$ に対して

$$
z_j = d_j
$$

と選ぶと

$$
d_j = z_j = \sigma(q_j) \implies q_j = \sigma^{-1}(d_j) = \log\left(\frac{d_j}{1 - d_j}\right)
$$

なので

$$
\begin{aligned}
y_1v_{11} + y_2v_{12} &= \log\left(\frac{d_1}{1 - d_1}\right) + \theta, \\
y_1v_{21} + y_2v_{22} &= \log\left(\frac{d_2}{1 - d_2}\right) + \theta, \\
y_1v_{31} + y_2v_{32} &= \log\left(\frac{d_3}{1 - d_3}\right) + \theta
\end{aligned}
$$

及び

$$
\begin{aligned}
x_1u_{11} + x_2u_{12} + x_3u_{13} + x_4u_{14} &= \log\left(\frac{y_1}{1 - y_1}\right) + \theta, \\
x_1u_{21} + x_2u_{22} + x_3u_{23} + x_4u_{24} &= \log\left(\frac{y_2}{1 - y_2}\right) + \theta
\end{aligned}
$$

となる。

具体的手続き M 個のデータからランダムに 2 個選ぶ（もちろん一次独立）：

$$
\begin{aligned}
&\text{入力データ} \quad \mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}), \ \mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)}) \\
&\text{教師信号} \quad \mathbf{d}^{(1)} = (d_1^{(1)}, d_2^{(1)}, d_3^{(1)}), \qquad \mathbf{d}^{(2)} = (d_1^{(2)}, d_2^{(2)}, d_3^{(2)}).
\end{aligned}
$$

このとき誤差関数は

$$
E = E(U, V) = \frac{1}{2}\sum_{j=1}^{3}(z_j^{(1)} - d_j^{(1)})^2 + \frac{1}{2}\sum_{j=1}^{3}(z_j^{(2)} - d_j^{(2)})^2
$$

となる。ここで

$$z_j^{(1)} = d_j^{(1)},\ z_j^{(2)} = d_j^{(2)} \quad (j = 1, 2, 3)$$

と選ぶ。

(1) シナプス結合 V の決定

$$\begin{cases} y_1^{(1)} v_{11} + y_2^{(1)} v_{12} = \log\left(\frac{d_1^{(1)}}{1-d_1^{(1)}}\right) + \theta, \\ y_1^{(2)} v_{11} + y_2^{(2)} v_{12} = \log\left(\frac{d_1^{(2)}}{1-d_1^{(2)}}\right) + \theta \end{cases}$$

$$\begin{cases} y_1^{(1)} v_{21} + y_2^{(1)} v_{22} = \log\left(\frac{d_2^{(1)}}{1-d_2^{(1)}}\right) + \theta, \\ y_1^{(2)} v_{21} + y_2^{(2)} v_{22} = \log\left(\frac{d_2^{(2)}}{1-d_2^{(2)}}\right) + \theta \end{cases}$$

$$\begin{cases} y_1^{(1)} v_{31} + y_2^{(1)} v_{32} = \log\left(\frac{d_3^{(1)}}{1-d_3^{(1)}}\right) + \theta, \\ y_1^{(2)} v_{31} + y_2^{(2)} v_{32} = \log\left(\frac{d_3^{(2)}}{1-d_3^{(2)}}\right) + \theta \end{cases}$$

となる。従って連立方程式を解いて

$$V = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \\ v_{31} & v_{32} \end{pmatrix}$$

が決まる。更に

(2) シナプス結合 U の決定

$$\begin{cases} x_1^{(1)} u_{11} + x_2^{(1)} u_{12} + x_3^{(1)} u_{13} + x_4^{(1)} u_{14} = \log\left(\frac{y_1^{(1)}}{1-y_1^{(1)}}\right) + \theta, \\ x_1^{(2)} u_{11} + x_2^{(2)} u_{12} + x_3^{(2)} u_{13} + x_4^{(2)} u_{14} = \log\left(\frac{y_1^{(2)}}{1-y_1^{(2)}}\right) + \theta \end{cases}$$

$$\begin{cases} x_1^{(1)} u_{21} + x_2^{(1)} u_{22} + x_3^{(1)} u_{23} + x_4^{(1)} u_{24} = \log\left(\frac{y_2^{(1)}}{1-y_2^{(1)}}\right) + \theta, \\ x_1^{(2)} u_{21} + x_2^{(2)} u_{22} + x_3^{(2)} u_{23} + x_4^{(2)} u_{24} = \log\left(\frac{y_2^{(2)}}{1-y_2^{(2)}}\right) + \theta \end{cases}$$

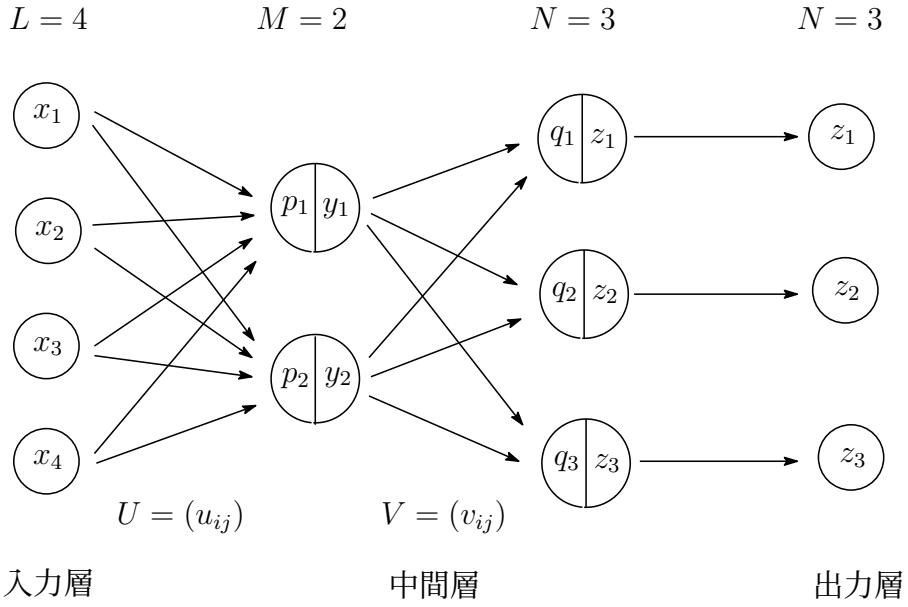
となるので連立方程式を解いて $\{u_{11}, u_{12}, u_{13}, u_{14}\}$ 及び $\{u_{21}, u_{22}, u_{23}, u_{24}\}$ の半分が求まる。従って

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \end{pmatrix}$$

の半分が決まる。残りの半分は元のままである。

目的 以上の繰り返しによるシナプス結合の強化

以下のことをわかりやすくするために、再び深層学習の図をリストする。



First step （ランダムに）データ $\{\mathbf{x}^{(1)}, \mathbf{d}^{(1)}\}, \{\mathbf{x}^{(2)}, \mathbf{d}^{(2)}\}$ を与える。

Second step 初期条件 $U(0)$ 及び $V(0)$ を適当にとる。

$$
\begin{aligned}
U(0) &= \begin{pmatrix} u_{11}(0) & u_{12}(0) & u_{13}(0) & u_{14}(0) \\ u_{21}(0) & u_{22}(0) & u_{23}(0) & u_{24}(0) \end{pmatrix}, \\
V(0) &= \begin{pmatrix} v_{11}(0) & v_{12}(0) \\ v_{21}(0) & v_{22}(0) \\ v_{31}(0) & v_{32}(0) \end{pmatrix}.
\end{aligned}
$$

このとき、数値

$$\{y_1^{(1)}, y_2^{(1)}\}, \quad \{y_1^{(2)}, y_2^{(2)}\}; \quad \{z_1^{(1)}, z_2^{(1)}, z_3^{(1)}\}, \quad \{z_1^{(2)}, z_2^{(2)}, z_3^{(2)}\}$$

が決まる。誤差関数は

$$E = E(U,V) = \frac{1}{2}\sum_{j=1}^{3}(z_j^{(1)} - d_j^{(1)})^2 + \frac{1}{2}\sum_{j=1}^{3}(z_j^{(2)} - d_j^{(2)})^2$$

となる

Third step 上述の (1), (2) より $V(1)$ と $U(1)$ が決まる。

$$U(1) = \begin{pmatrix} u_{11}(1) & u_{12}(1) & u_{13}(1) & u_{14}(1) \\ u_{21}(1) & u_{22}(1) & u_{23}(1) & u_{24}(1) \end{pmatrix}, \quad V(1) = \begin{pmatrix} v_{11}(1) & v_{12}(1) \\ v_{21}(1) & v_{22}(1) \\ v_{31}(1) & v_{32}(1) \end{pmatrix}.$$

但し、$U(1)$ の半分は $U(0)$ のままである。

この操作を K 回繰り返す：

$$\{U, V\}_1 \longrightarrow \{U, V\}_2 \longrightarrow \cdots \longrightarrow \{U, V\}_K$$

最終的に誤差関数 E は M 個のデータ $\{\mathbf{x}, \mathbf{d}\}$ (添え字は省略) に対して

$$E = E(U, V) = \frac{1}{2}\sum_{j=1}^{3}(z_j - d_j)^2 \approx 0$$

となる（であろう)。ここで 教師有学習 は終了する。

問題点 このシステムの問題点は、入力層と出力層が指定されたとき、中間層の個数を決定する standard が無いことである（あるのかも知れないが、見当たらない)。

[VI] 何故 Deep Learning は有効なのか？

Deep Learning は何故かくも有効なのか考えてみたい。

[A] 本当の変数

変換

$$(x_1, x_2, \cdots, x_n) \xrightarrow{W} (y_1, y_2, \cdots, y_m)$$

with

$$y_i = \sum_{j=1}^{n} w_{ij}x_j - \theta \Longleftarrow W = (w_{ij})$$

は数学の変数変換と考えられる。Big data $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})$ $(k = 1, 2, \cdots, N)$ の中から**本当の**又は**隠れた**変数を見つけようとしているのである。実際、具体例では見つけている。

[B] シグモイド関数の良い性質

シグモイド関数 $\sigma(x)$ に関して以下の予想をもっている。

Conjecture $F(x)$ を $[0,1]$ に値をもつ $\mathbf{R}$ 上の任意の微分可能関数とする（連続関数でも良い）。このとき

$$F(x) \approx \sum_{j=1}^{N} c_j \sigma(\alpha_j x)$$

で、N はあまり大きくなくても良い（1000 億くらい[3]。1000 億は十分に大きいと言う人もいるかもしれないが、日本の国債の総額は 1000 兆以上なのである。1000 億は十分に小さいのである）。但し、c_j 及び α_j は適当な定数である。

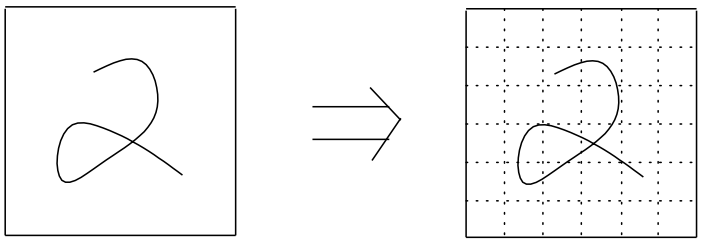
Deep Learning の背後には、この conjecture があると思う。

[VII] 多クラス分類 ··· 例

例として、手書き数字の人工知能による認識を示そう。目的は、下に書かれた（美しい？）数字の画像が、$0, 1, 2, \cdots, 9$ のどれであるかを答えることである[4]。

この画像を 32×32 ピクセルに分解する（その個数は $32 \times 32 = 1024$）。各ピクセルの白と黒の間の灰色濃度（グレー・スケール）を x とする（従って、x の値域は $[0,1]$ である）。従って、$\mathbf{x}$ はグレー・スケールをベースとした 1024 次元ベクトルである。

$$\mathbf{x} = (x_1, x_2, \cdots, x_{1024})^{\mathrm{T}}.$$

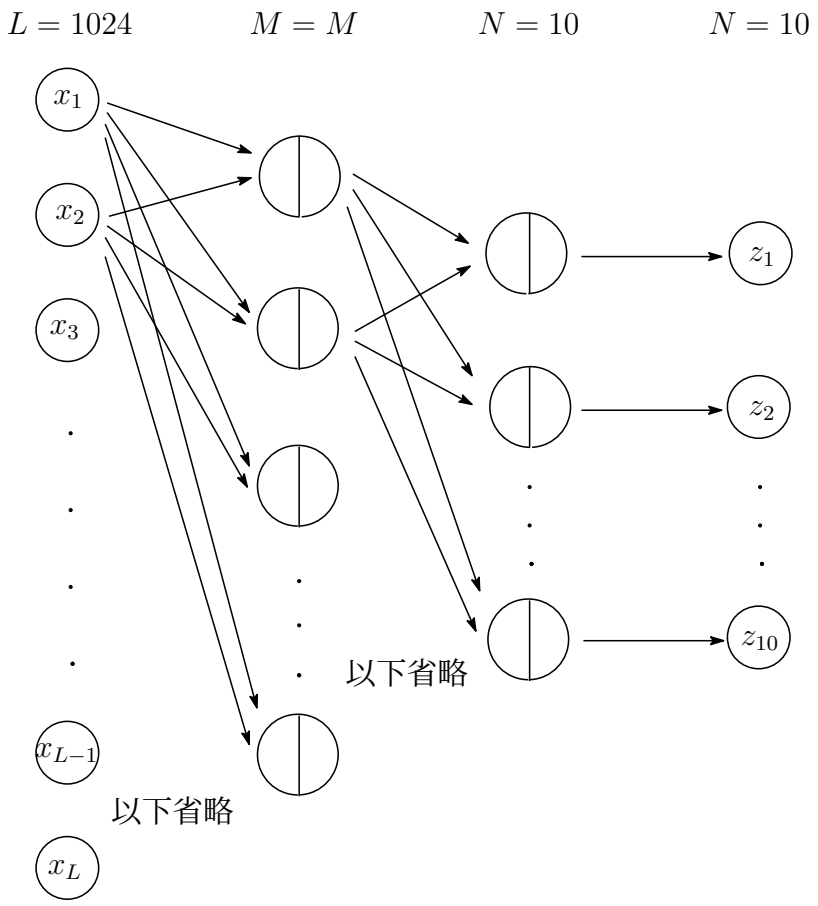


32×32

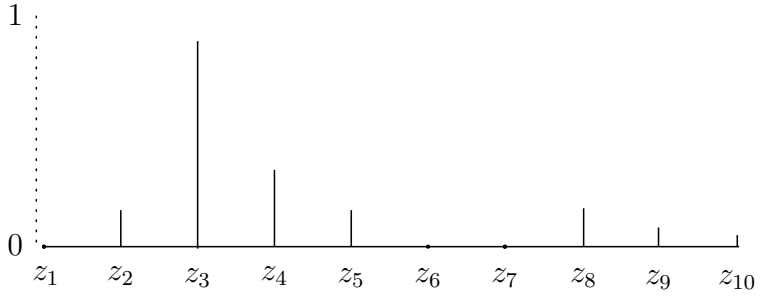
これを 教師有学習を終了 したニューラルネットワーク・システムにかけてみる（問題点は、中間層 M の個数を特定する論理が無いことである）：

[3]中枢神経全体の神経細胞の数は、1000 億から 2000 億の間と推定されている

[4]残念ながら、私の使っているソフトである winTpic では、厚みのある数字や文字は描けない



結論は、以下のようになるのであろう（注：$z_1 = 0$, $z_2 = 1$, $\cdots$, $z_{10} = 9$)。



続く（？）

[∞] 人工知能は人類を越えるか？

いろいろな（人工知能の）本で、“人工知能は人類を越えるか？” という問題が提起されている。人工知能がこのまま進化すれば、やがて人類の能力を越えてしまうのではないか と言うことである。私の考えでは、ある部分は正しく、ある部分では間違っている。

最先端の人工知能は Big Data をベースにしており、(自己の経験をベースにする) Small Data の人間思考よりも深く広いのは当然である。囲碁で人間が負けたことはその証明であろう (Google は過去の約 2000 万局の囲碁の勝負をデータベースに入れていた)。

しかし、人間には "**思考のジャンプ**" があるのだ。ある場面から他の場面へ一気に change することが可能である。具体例で示そう。

問題 今の人工知能が、前期量子力学に遭遇したとき、後期量子力学 (例えば、1925 年の Heisenberg の記念碑的論文) を生み出すことが出来るか？

まさに 思考のジャンプ そのものである。人工知能には無理だと信じている。もう少し抽象的 (数学的) な見方を示そう。

問題が、関数 $f(x)$ で表されたとする。各人間には Small Data から

$$f(x) = a_0 + a_1 x + a_2 x^2$$

までしかわからないとしよう。人工知能には Big Data を用いて高次の項がとらえられるので

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

までわかるのである。

しかし、ある問題 (例えば、量子力学) では、人間は一気に

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

がわかるのである。まさに人口知能である。結論として

「人工知能」 と 「人口知能」 は違うのである。

参考文献

[1] Quoc V. Le et al : Building High-Level Features Using Large Scale Unsupervised Learning, to appear in Proceedings of the 29^{th} International Conference on Machine Learning, Edinburgh, 2012.

[2] 溝口理一郎, 石田享（共編）: **人工知能**, オーム社, 2000.

[3] 岡谷貴之 : **深層学習**, 講談社, 2015.

[4] 中井悦司 : **機械学習理論入門**, 技術評論社, 2015.

[5] 三宅陽一郎 : **人工知能の作り方**, 技術評論社, 2017.

[6] 合原一幸, 神埼亮平（編）: **理工学系からの脳科学入門**, 東京大学出版会, 2008.

[7] 宮川博義, 井上雅司 : **ニューロンの生物物理**, 丸善出版, 2013.

[8] 藤井一幸 : 人口問題のフェルフルスト・モデルについて, 横浜市立大学講義録, 2014.

[9] 甘利俊一 : **脳・心・人工知能：数理で脳を解き明かす**, 講談社, ブルーバックス B-1968, 2016.

[10] 松山豊 : **人工知能は人間を越えるか**, KADOKAWA, 2015.

[11] 北研二, 津田和彦, 獅ノ堀正幹 : **情報検索アルゴリズム**, 共立出版, 2002.

[12] 笠原皓司 : **線形代数学**, サイエンス社, 2000.

[13] 名取享 : **線形計算**, 朝倉書店, 1998.

[14] K. Fujii and H. Oike : A superintroduction to Google matrices for undergraduates, Far East Journal of Mathematical Education, 14 (1) 55-68, 2015.

[15] K. Fujii and H. Oike : How to understand Google from the mathematical point of view, preprint, Yokohama City University, 2015.

[16] 藤井一幸 : **線形代数への誘い**, 横浜市立大学講義録, 2014.

[17] 藤井一幸 : **解析学の散歩道**, 横浜市立大学講義録, 2014.