

誤差逆伝播法

2017年11月10日



今回の内容

- 誤差逆伝播法解説。



誤差逆伝播法

- 誤差逆伝播法(バックプロパゲーション)について、解説。
<https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
上記のページを参照している。
- また、このページのGitHubのリンクから、プログラム(neural-network.py)をダウンロードする(メールにも添付してある)。
Pythonで実装されたプログラムがある。

誤差逆伝播法（バックプロパゲーション）

- 誤差逆伝播法（BP）は多層パーセプトロンの学習に使われる学習アルゴリズム。ある学習データが与えられたとき、多層パーセプトロンの出力が学習データと一致するように各層の間の結合荷重を修正するという学習法である。
- 多層パーセプトロンは誤差逆伝播法によって教師あり学習をおこない、パターン識別や関数の近似などに用いられる。
- 多層パーセプトロン：ユニットが複数の層を構成するように並び、入力から出力への一方向へのみ信号の伝達がおこなわれるネットワーク。

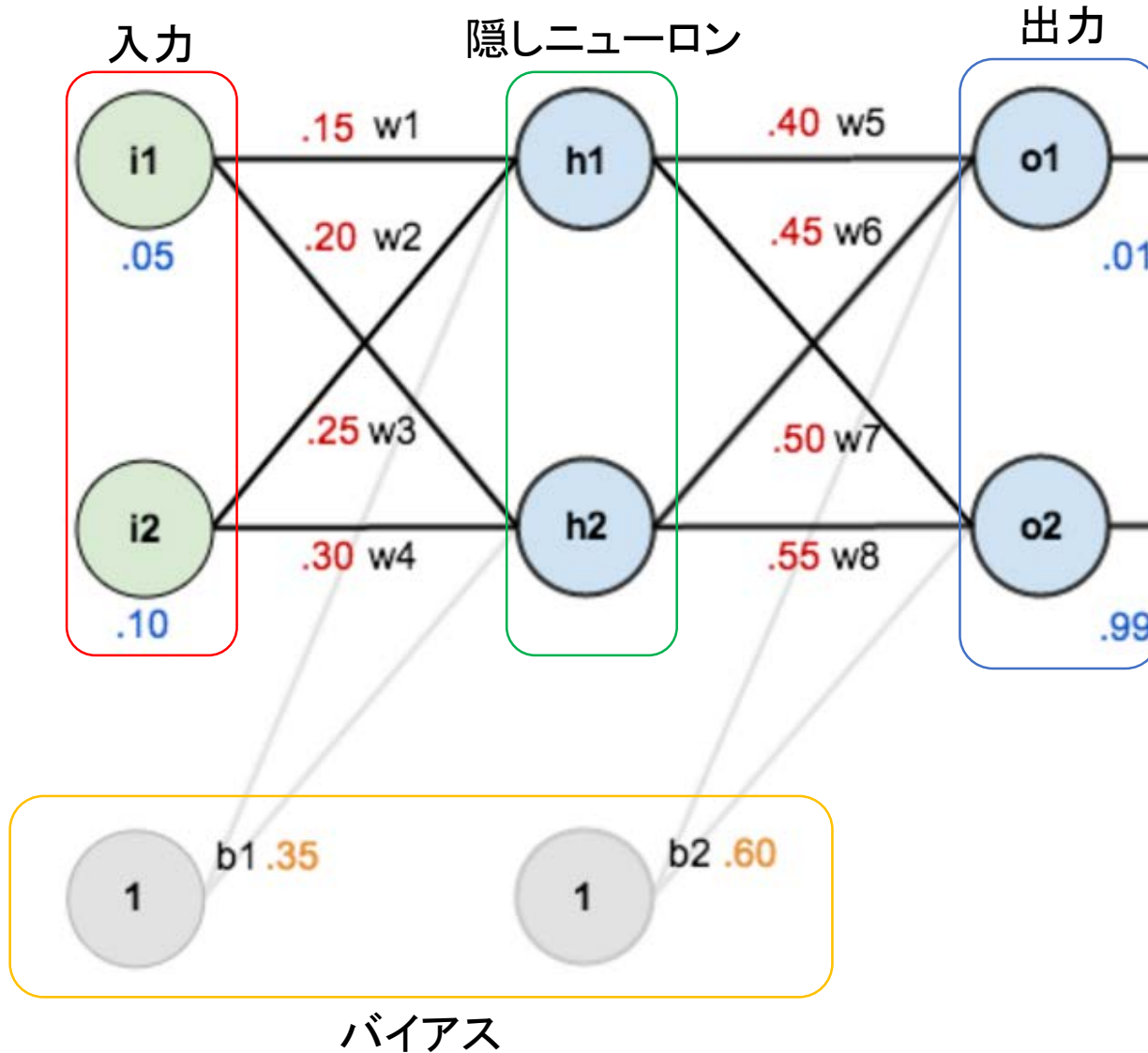
<http://www.brain.kyutech.ac.jp/~furukawa/data/bp.html>

誤差逆伝播法のアルゴリズム

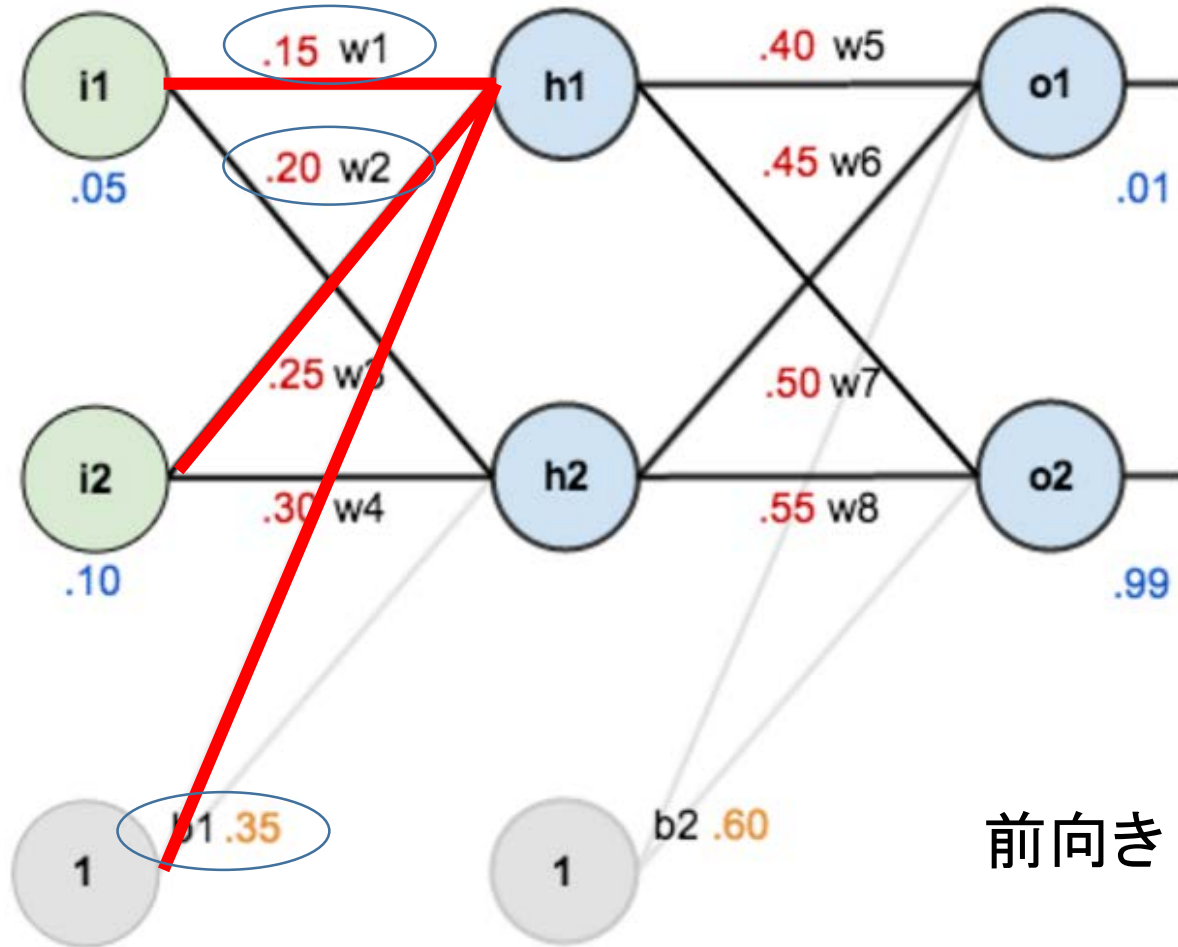
1. ニューラルネットワークに学習のためのサンプルを与える。
2. ネットワークの出力とそのサンプルの最適解を比較する。各出力ニューロンについて誤差を計算する。
3. 個々のニューロンの期待される出力値と倍率 (scaling factor)、要求された出力と実際の出力の差を計算する。これを局所誤差と言う。
4. 各ニューロンの重みを局所誤差が小さくなるよう調整する。
5. より大きな重みで接続された前段のニューロンに対して、局所誤差の責任があると判定する。
6. そのように判定された前段のニューロンのさらに前段のニューロン群について同様の処理を行う。

<https://ja.wikipedia.org/wiki/%E3%83%90%E3%83%83%E3%82%AF%E3%83%97%E3%83%AD%E3%83%91%E3%82%B2%E3%83%BC%E3%82%B7%E3%83%A7%E3%83%B3>

誤差逆伝播法



誤差逆伝播法 (h_1 導出)



誤差逆伝播法 (h_1)

- 各隠れ層のニューロンへの全ての入力を使い(net_{h1})、活性化関数(シグモイド関数)に入れる。

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

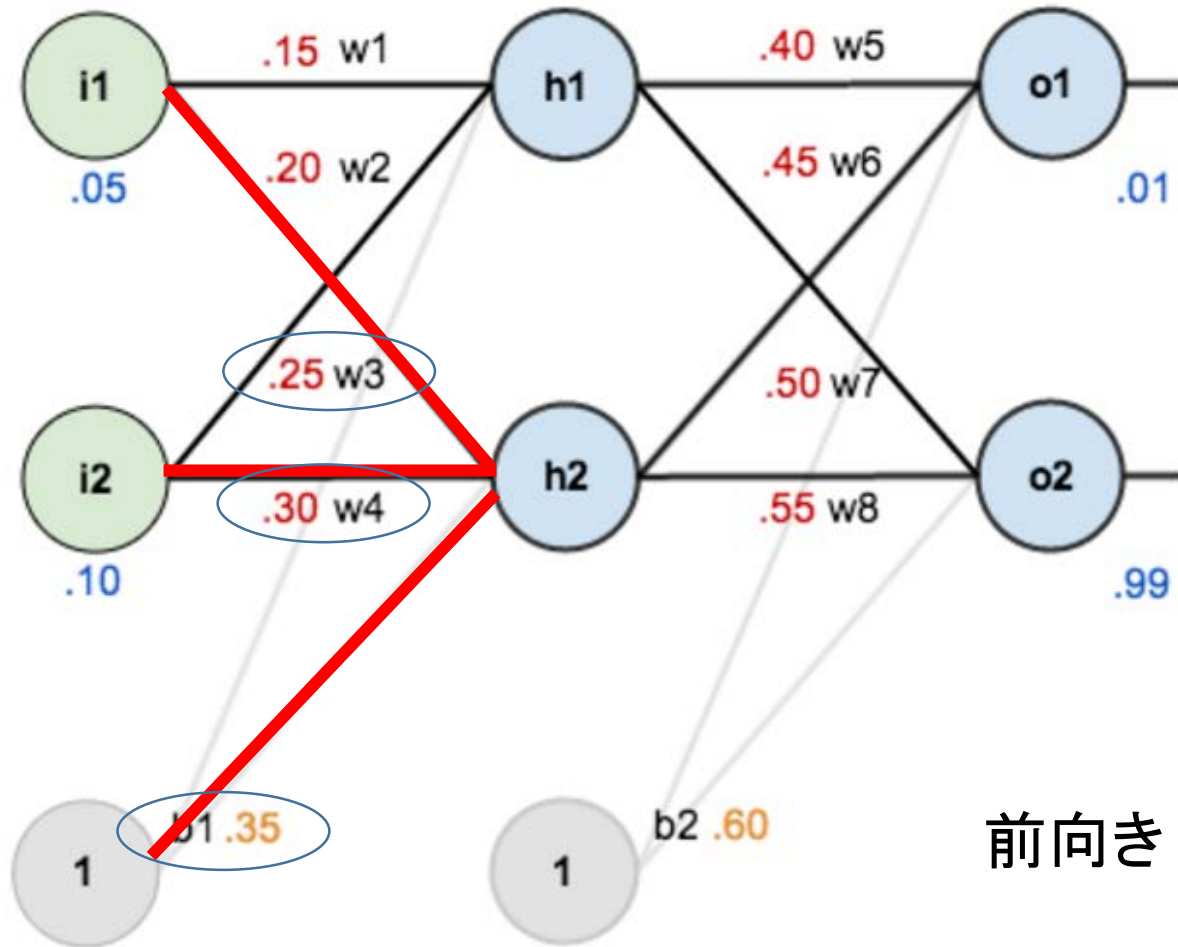
$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

- これをシグモイド関数に入れる。

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

- この隠れ層ニューロンからの出力 out_{h1} を、出力層への入力にする。

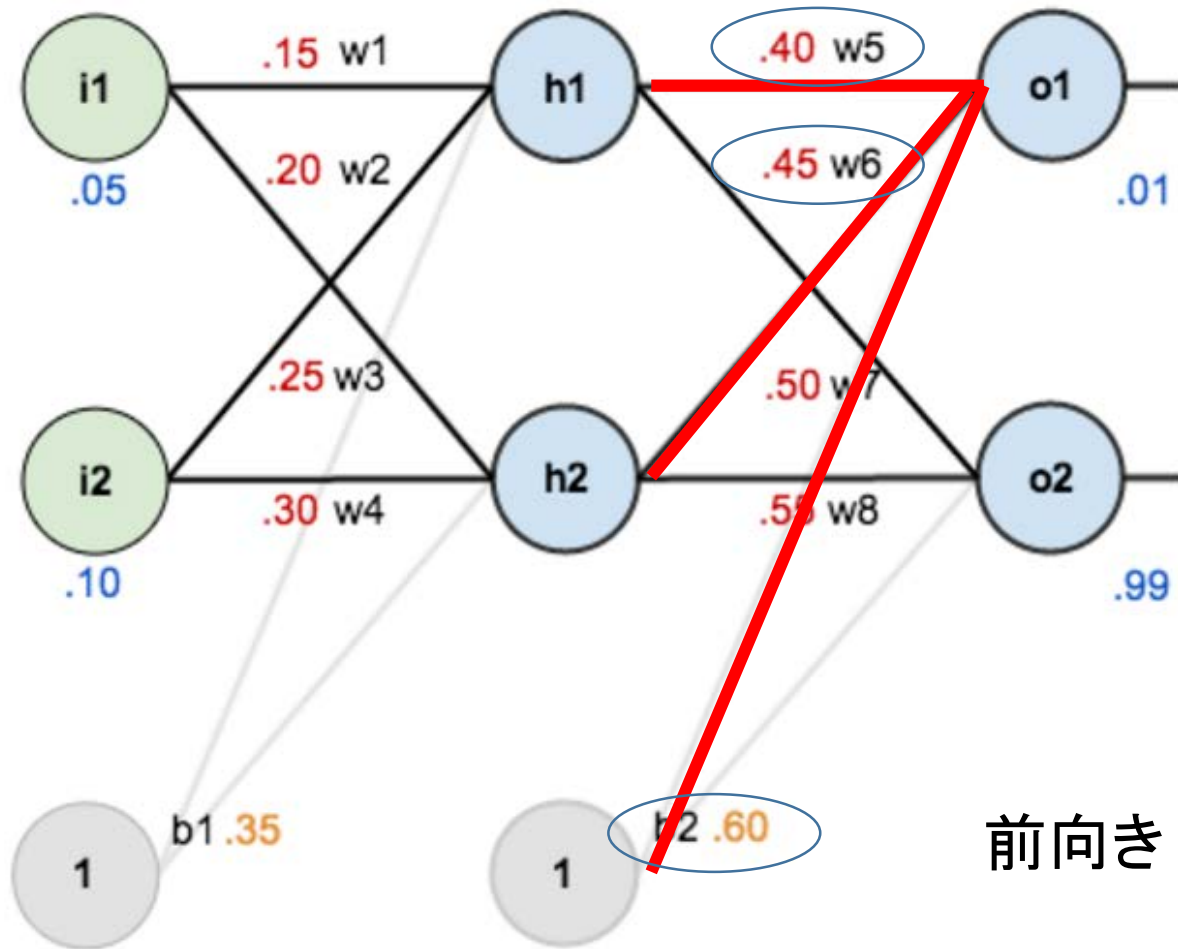
誤差逆伝播法 (h_2 導出)



誤差逆伝播法 (h_2)

- h_1 の時と同様に
- $net_{h2} = w_3 * i_1 + w_4 * i_2 + b_1 * 1$
 $= 0.25 * 0.05 + 0.3 * 0.1 + 0.35 * 1 = 0.3925$
- これをシグモイド関数に入れる。
- $out_{h2} = 1 / (1 + e^{-0.3925}) = 0.596884378$

誤差逆伝播法(出力層 o_1)



誤差逆伝播法 (o_1 導出)

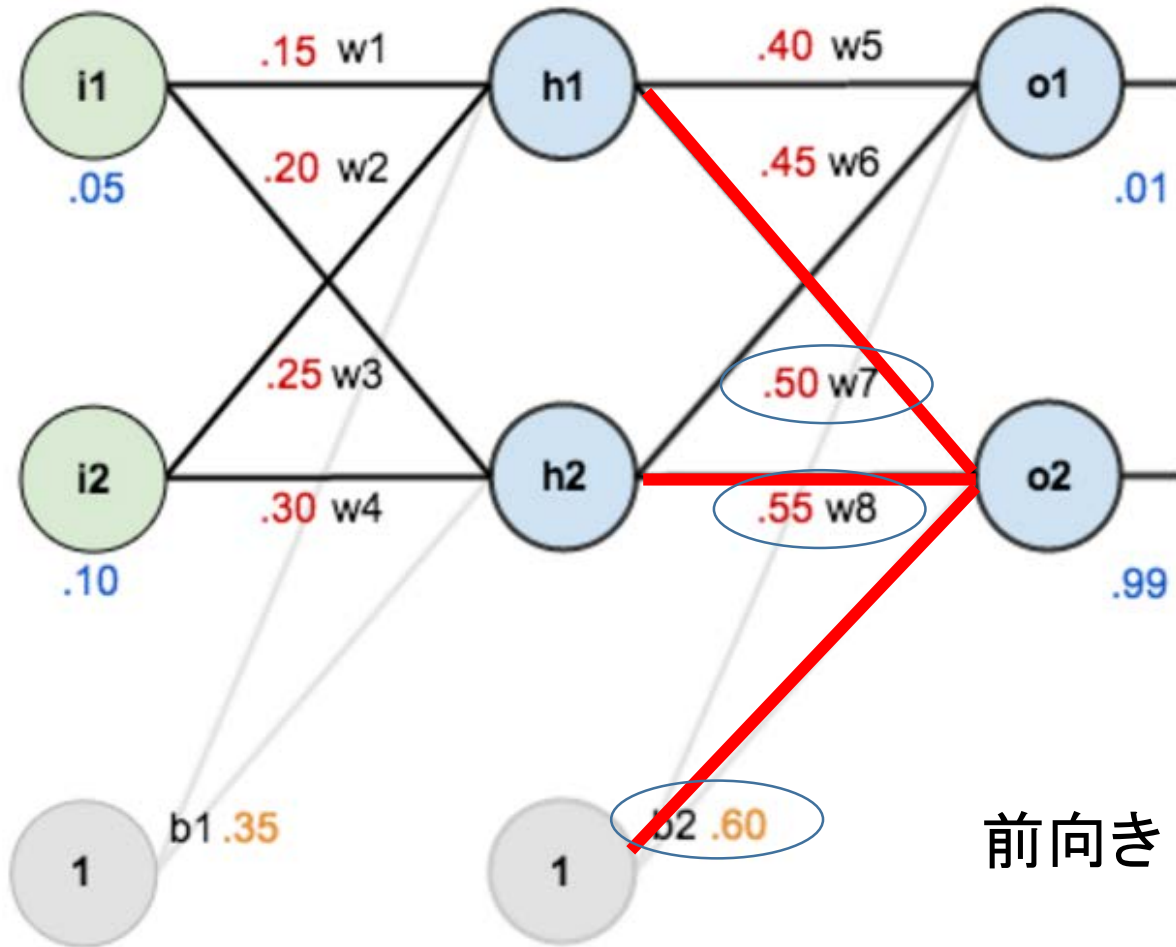
- 先程得た隠れ層の出力 out_{h1} , out_{h2} を入力とし、出力層での出力を出す。

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

誤差逆伝播法 (o_2 導出)



誤差逆伝播法 (o_2)

- o_1 の時と同様に

- $net_{o_2} = w_7 * out_{h_1} + w_8 * out_{h_2} + b_2 * 1$

$$= 0.5 * 0.593269992 + 0.55 * 0.596884378 + 0.6 * 1 = 1.2249214$$

- これをシグモイド関数に入れる。

- $out_{o_2} = 1 / (1 + e^{-1.2249214}) = 0.772928465$

総誤差計算

- 出力層の各ニューロンにおける、それぞれの出力と実際に期待した値との差の二乗に1/2をかけたものの合計を求める。

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

総誤差計算

- 総誤差は、今回は E_{o1} と E_{o2} の合計を求めればいいので、分けて考えて、

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

- E_{o1} と同様に

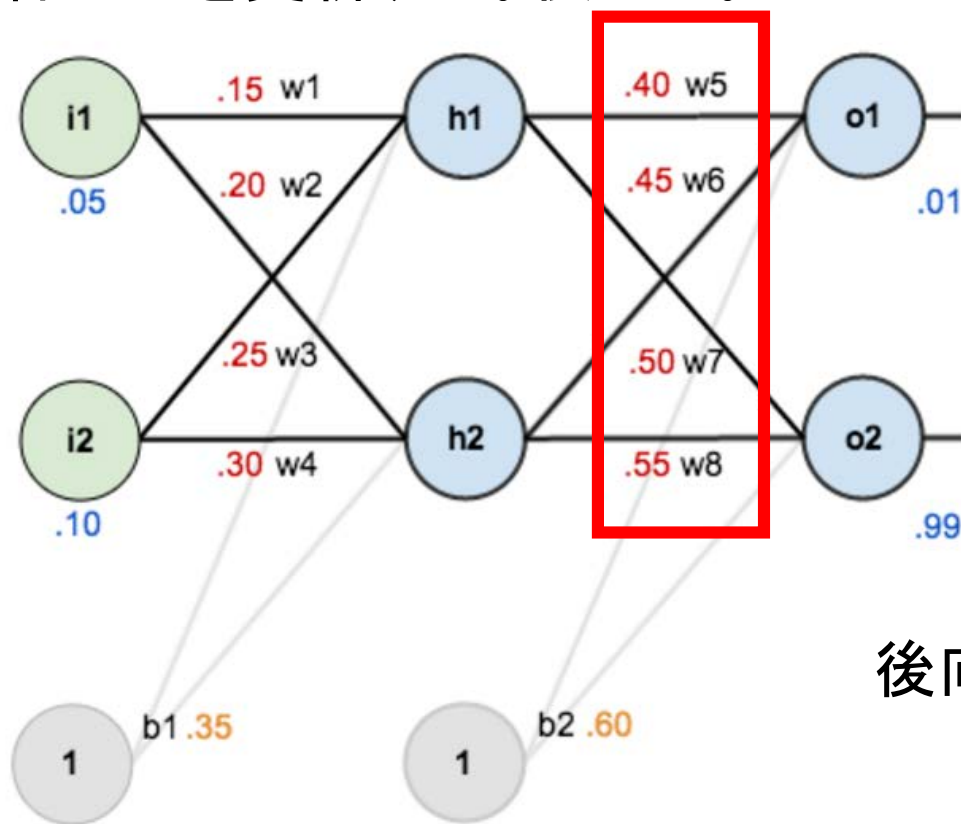
$$E_{o2} = 0.023560026$$

- 2つを足し合わせる。

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

重みの更新

- ここまでの計算結果を、今度は出力層から前の層へ次々と利用して、各重みを更新する。後向き。

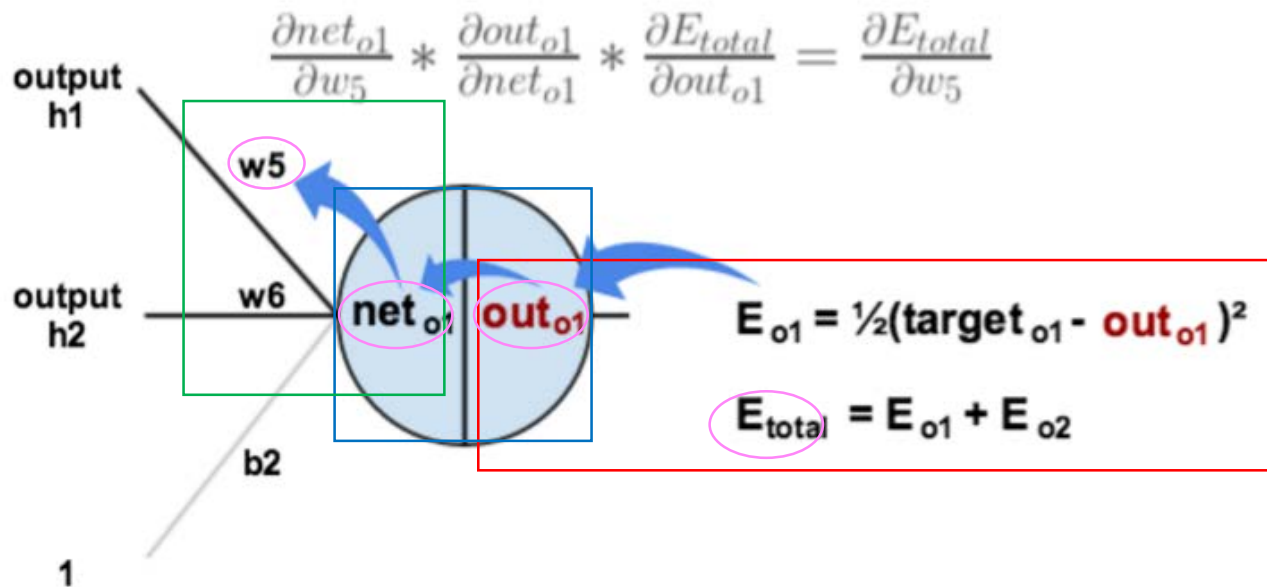


後向き

重みの更新(出力層)

- まず、出力層において、各重みを更新する。
 w_5 に関しては次式に基づき求める。

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$



重みの更新 (w_5)

- E_{total} を out_{o1} について微分した式の値を求める。

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

重みの更新 (w_5)

- out_{o1} の式を net_{o1} について微分したものの値を求める。

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

which has a nice derivative of:

$$\frac{\partial \varphi}{\partial z} = \varphi(1 - \varphi)$$



$$\varphi = 1/(e^{-z} + 1) = (e^{-z} + 1)^{-1}$$

$$\begin{aligned} \frac{\partial \varphi}{\partial z} &= e^{-z}/(e^{-z} + 1)^2 \\ &= 1/(e^{-z} + 1)(e^{-z} + 1 - 1)/(e^{-z} + 1) \\ &= \varphi(1 - \varphi) \end{aligned}$$

重みの更新 (w_5)

- 既に求めた net_{o1} の式を、 w_5 について微分したものの値を求める。

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

重みの更新(w_5)

- 3つを全てまとめて計算する。

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

重みの更新(出力層)

- w_5 を次式に基づき更新する。 η (学習率: $0 < \eta \leq 1$)は0.5と今回は設定している。

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

- w_6, w_7, w_8 も同様にすると、

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

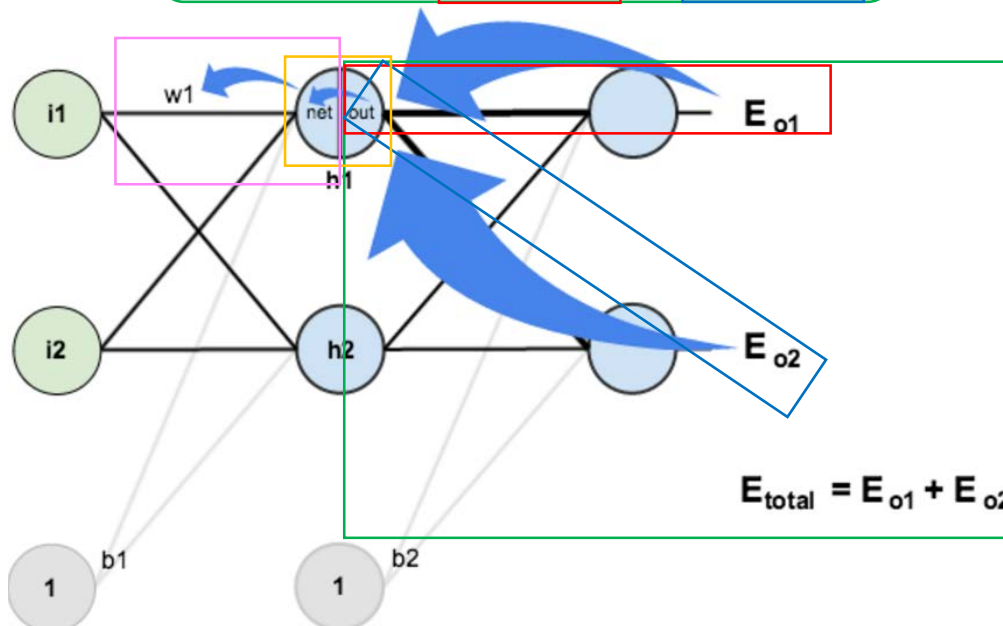
重みの更新(隠れ層)

- 次に、隠れ層において重みを更新する。 w_1 を例として。

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

↓

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



後向き

重みの更新 (w_1)

- まずは、緑の部分を求める。その中の赤の部分について。

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

- 部分ごとに求め、計算する。

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

重みの更新 (w_1)

- 青の部分について、同様に行う。

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

- まとめて

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

重みの更新 (w_1)

- 残りの黄色部分について。

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

which has a nice derivative of:

$$\frac{\partial \varphi}{\partial z} = \varphi(1 - \varphi)$$



$$\varphi = 1/(e^{-z} + 1) = (e^{-z} + 1)^{-1}$$

$$\begin{aligned} \partial \varphi / \partial z &= e^{-z} / (e^{-z} + 1)^2 \\ &= 1 / (e^{-z} + 1) (e^{-z} + 1 - 1) / (e^{-z} + 1) \\ &= \varphi (1 - \varphi) \end{aligned}$$

重みの更新 (w_1)

- 残りのピンク部分について。

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

重みの更新 (w_1)

- 全てをまとめる。

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

重みの更新(w_1)

- w_1 を更新する($\eta:0.5$)。

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

- 他にも同様に

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

繰り返し

- 更新した重みで、5ページ目からの流れを繰り返し、また誤差を出し、重みを更新するという流れを設定回数だけ繰り返していく。
- プログラムでは1万回繰り返し、実行してみると少しずつ総誤差の値が小さくなっていることが分かる。

演習

- スライド内で計算の過程を端折った部分や、2回目、3回目の誤差を紙に式を書き表しながら、計算して出してみる。
- プログラムの結果と比較して、大体合っているか確認してみると良い。
- プログラムを解読してみる。

次回

- ニューラルネットワーク、Deep Learningと、それに関する知識。

