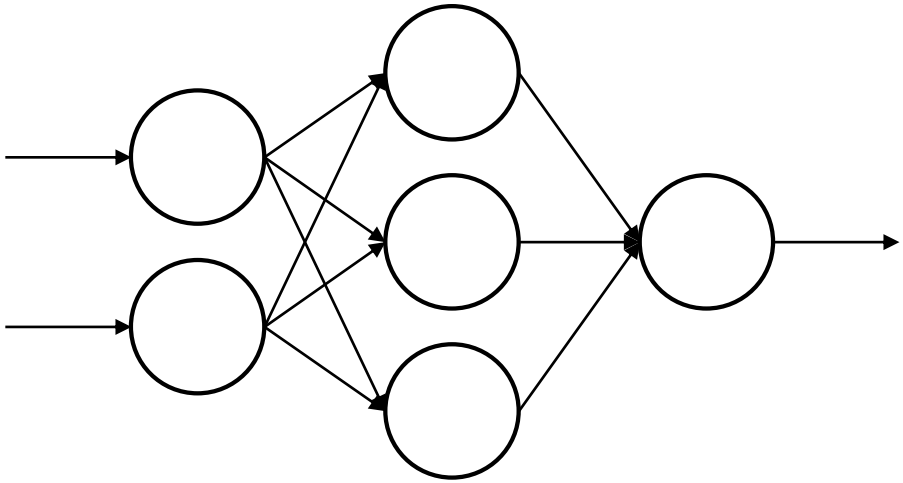
ニューラルネットワークの原理
Principle of Neural Network (NN)

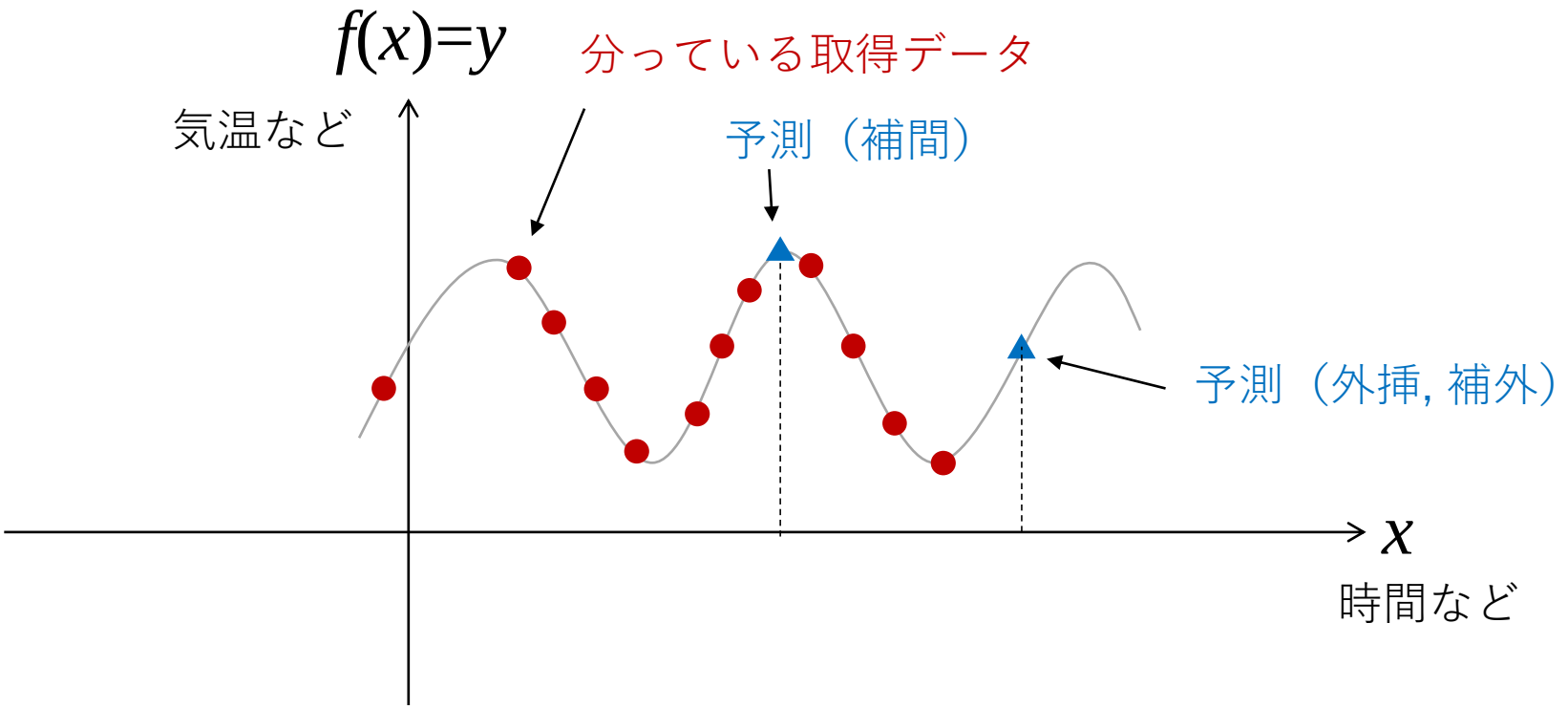


2020/6/12 平野拓一

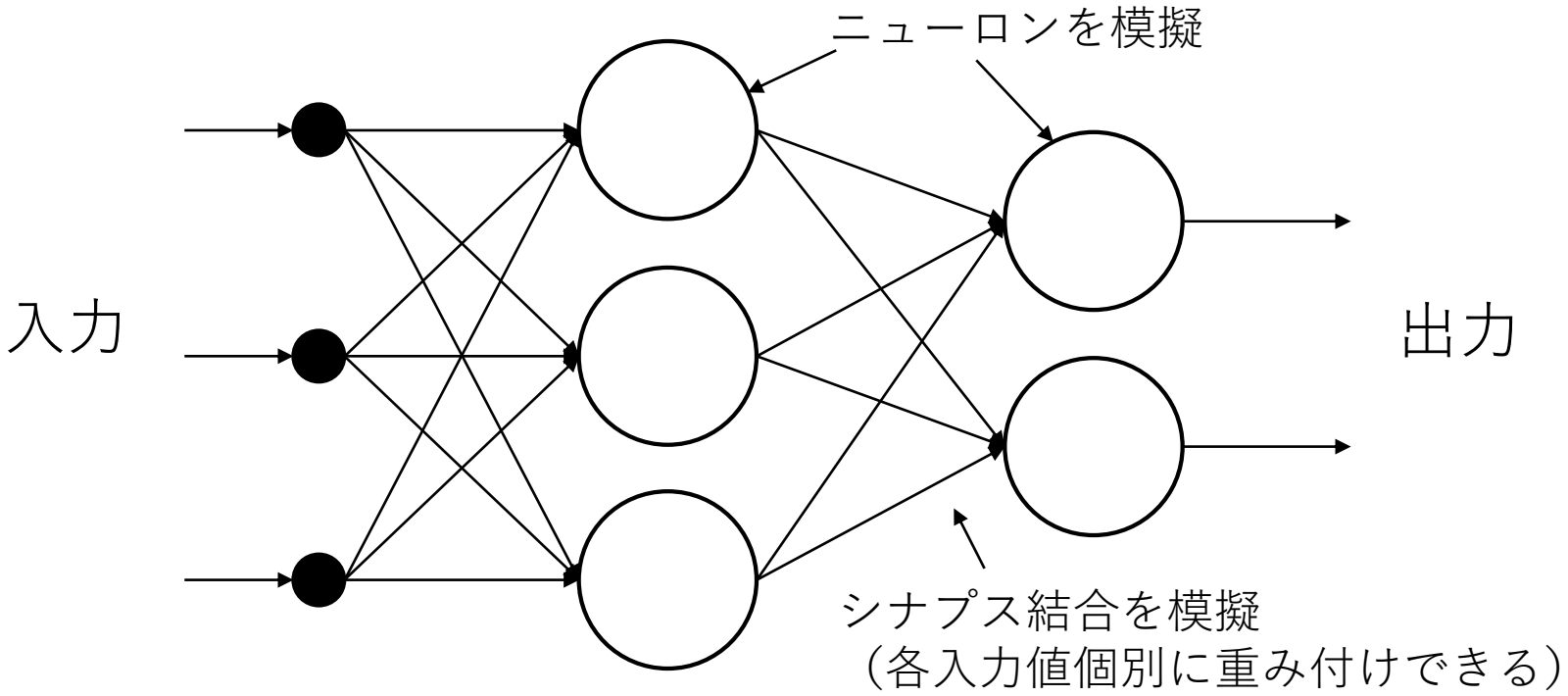
Neural Network

- 脳のニューロンを模擬
- 小脳のモデル
  （大脳のモデルは出来ていないから、まだまだ人間の仕事すべてが、AI(人工知能)に取って代わられることはない。今のメディアはAIを誇大広告しすぎである。（内容がわかっていないというのもあるだろうが））
- 一言で説明すると、汎化能力のある（つまり、補間、外挿機能がうまくいく）関数の近似を活用する技術である。
- 上の関数を応用して、見た目AIのように見せているだけである。（画像認識、データ予測など）

関数の補間と外挿について



ニューラルネットワーク

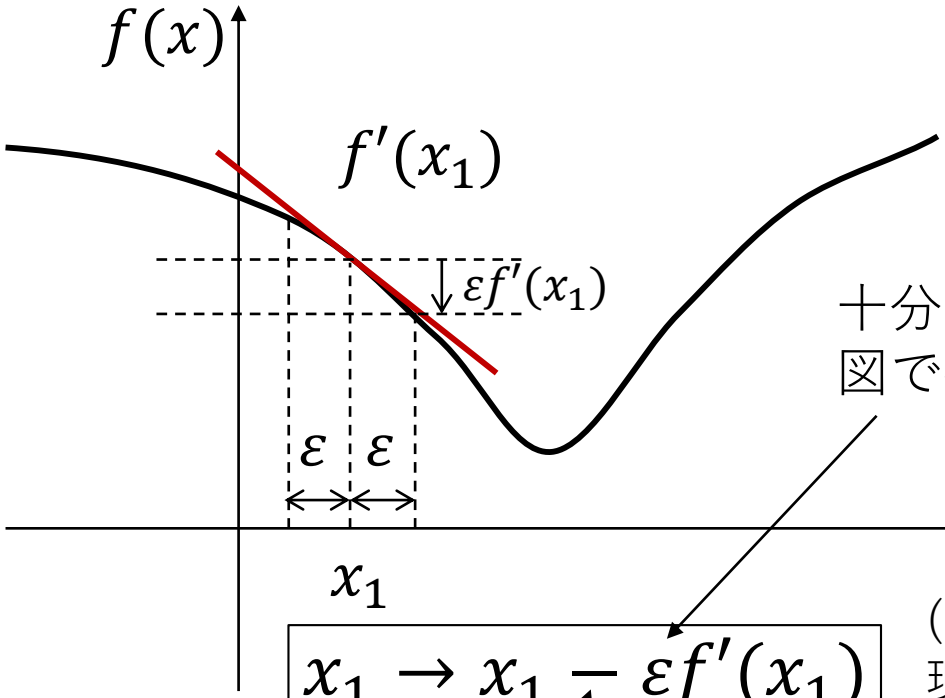


- ニューラルネットワークの技術は脳のニューロンのシナプス結合の原理を数値シミュレーションしたものである。
- ニューロンは複数のニューロンからの入力値に重みを付けて受け取る。その重み付け総和がある値（閾値）を超えたら1つの出力を出すという動作をする。
- 入力にある値を入れると、出力に値がでる。いくつかは入力と出力の組み合わせを与えて、学習させる。学習結果は各ニューロン間の結合の重み係数として得られる。

勾配法

- 極小あるいは極大を探索する数値計算法
- ニューラルネットワークでは、学習を行うために、その成功の可否を評価する評価関数を定義し、評価関数を最小化するために用いられる。
- 方程式や微分方程式を解く問題も、極小値を求める問題に帰着できる場合があり（変分法、それを基にした有限要素法など）、勾配法は一般的な手法としてよく用いられる。

勾配法



十分小さな値。十分というのは、図で谷を飛び越えない程度の値。

$$x_1 \rightarrow x_1 - \varepsilon f'(x_1)$$

(G1) $f'(x)$が解析的に表現されていると計算効率が良い。

$f(x)$が減る方向に動くため 。 ((G2) $f(x)$は下に凸がいい)

$f'(x)$をかけているのは傾きが大きい場所では大きく修正するため。 ((G3) $f(x)$は最小値で滑らかがいい)

xの更新量が小さくなったら、終了。 ($f(x)$は**<u>下に凸の滑らかな(G4)単峰性</u>**がいい)

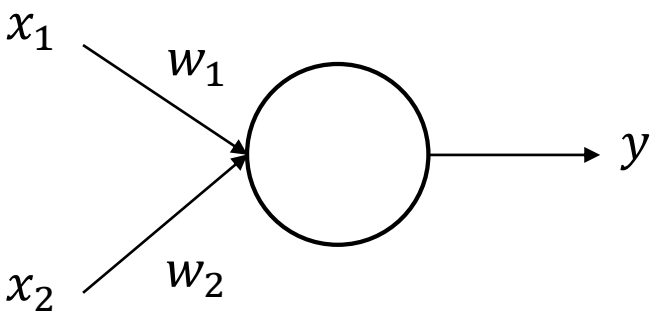
(G1)-(G4)がすべて満たされれば、 勾配法は絶対に上手くいく。
(ただし、(G1)は計算効率のためであって、必ずしも必要ではない)

ニューロンモデル

- シナプス結合
- シグモイド関数

ニューロンモデル

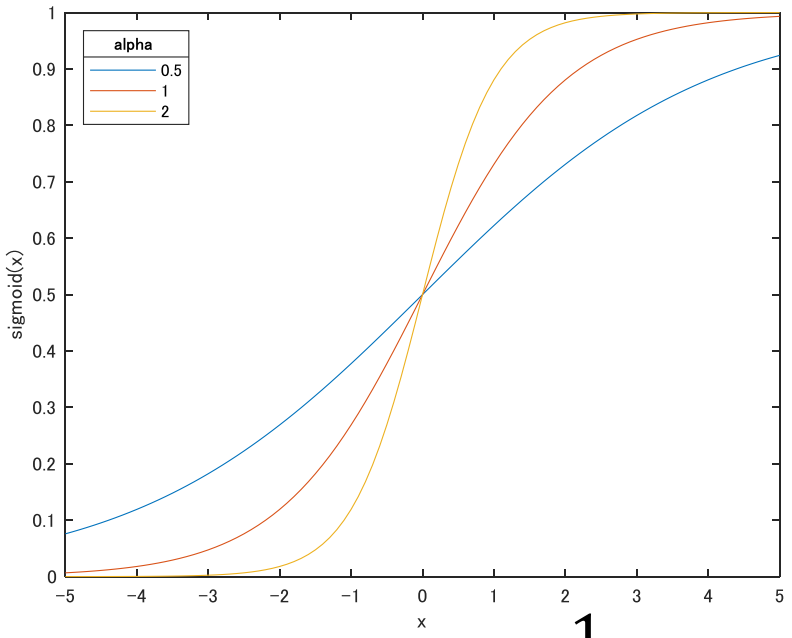
ニューロンの発火をシグモイド関数で表現する。シグモイド関数の良いところはパラメータで急峻度合いを調整でき、かつ滑らかなので微分ができるため誤差の勾配が解析的に求まるところである（さらに勾配はその値で計算可能で演算が削減できる）。



左右の平行移動（閾値）を制御

$$y = \mathrm{sigmoid}(x - \theta)$$

$$x = w_1 x_1 + w_2 x_2$$



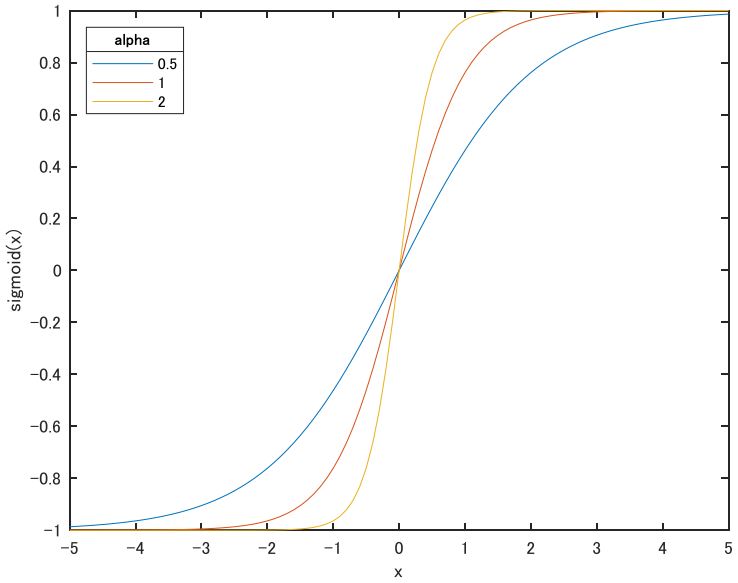
$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-\alpha x}}$$

ニューロン出力を表すこのような関数は活性化関数と呼ばれる。シグモイド関数はその一例である。

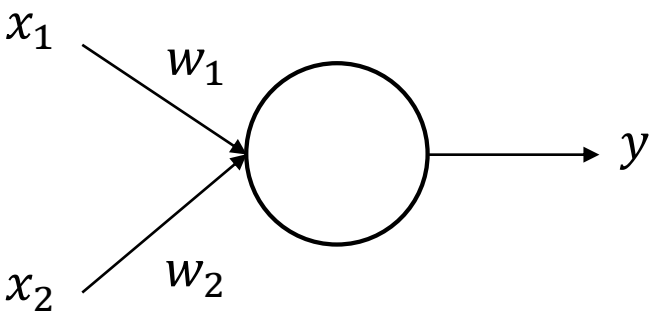
活性化関数

シグモイド関数以外にも、いくつかの種類の活性化関数が提案されている。次のハイパボリックタンジェントもその一例である。これは、シグモイド関数を上下方向に2倍拡大して縦方向に-1平行移動しただけである。-1～1の値を取る。

$$\tanh(\alpha x) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}}$$
$$= \frac{1-e^{-2\alpha x}}{1+e^{-2\alpha x}} = \frac{2-(1+e^{-2\alpha x})}{1+e^{-2\alpha x}}$$
$$= 2\mathrm{sigmoid}(2x) - 1$$

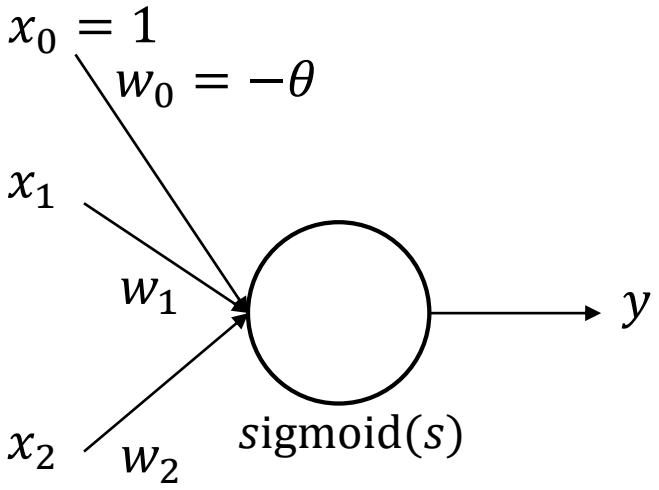


単一ニューロン



$$y = \mathrm{sigmoid}(x - \theta)$$
$$x = w_1 x_1 + w_2 x_2$$

θもwとして
扱う工夫



$$y = \mathrm{sigmoid}(-\theta + w_1 x_1 + w_2 x_2)$$
$$= \mathrm{sigmoid}(w_0 x_0 + w_1 x_1 + w_2 x_2)$$
$$= \mathrm{sigmoid}(s)$$

$$s = w_0 x_0 + w_1 x_1 + w_2 x_2$$

単一ニューロン（誤差）

$(x_1, x_2) = (a_1, a_2) \rightarrow b$　　学習データ（教師データ、訓練データ）

$E = (y(s) - b)^2$　　出力誤差

$\mathrm{sigmoid}(s)$

勾配の計算

合成関数の微分

$$\frac{\partial E}{\partial w_n} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial w_n}$$

w_nによる誤差の変化率

$$\begin{cases} \dfrac{\partial E}{\partial y} = 2(y(s) - b) \\ \dfrac{\partial y(s)}{\partial s} = \dfrac{\partial}{\partial s}\left(\dfrac{1}{1 + e^{-\alpha s}}\right) = \dfrac{\alpha e^{-\alpha s}}{(1 + e^{-\alpha s})^2} = \alpha\,\mathrm{sigmoid}(s)\big(1 - \mathrm{sigmoid}(s)\big) \\ \qquad\qquad = \alpha y(s)(1 - y(s)) \\ \dfrac{\partial s}{\partial w_n} = x_n \end{cases}$$

あとは勾配法を適用して、重みを決定する。

学習アルゴリズム

w_nを乱数で設定する。

全学習データ$(\boldsymbol{a}^{(\ell)}, \boldsymbol{b}^{(\ell)})$に対し、次の処理をする

順番に学習データを選ぶ。

誤差　$E = E(\boldsymbol{w})$

勾配ベクトル $\nabla E(\boldsymbol{w})$

勾配法を適用して修正

$$\boldsymbol{w} \to \boldsymbol{w} - \varepsilon \nabla E(\boldsymbol{w})$$

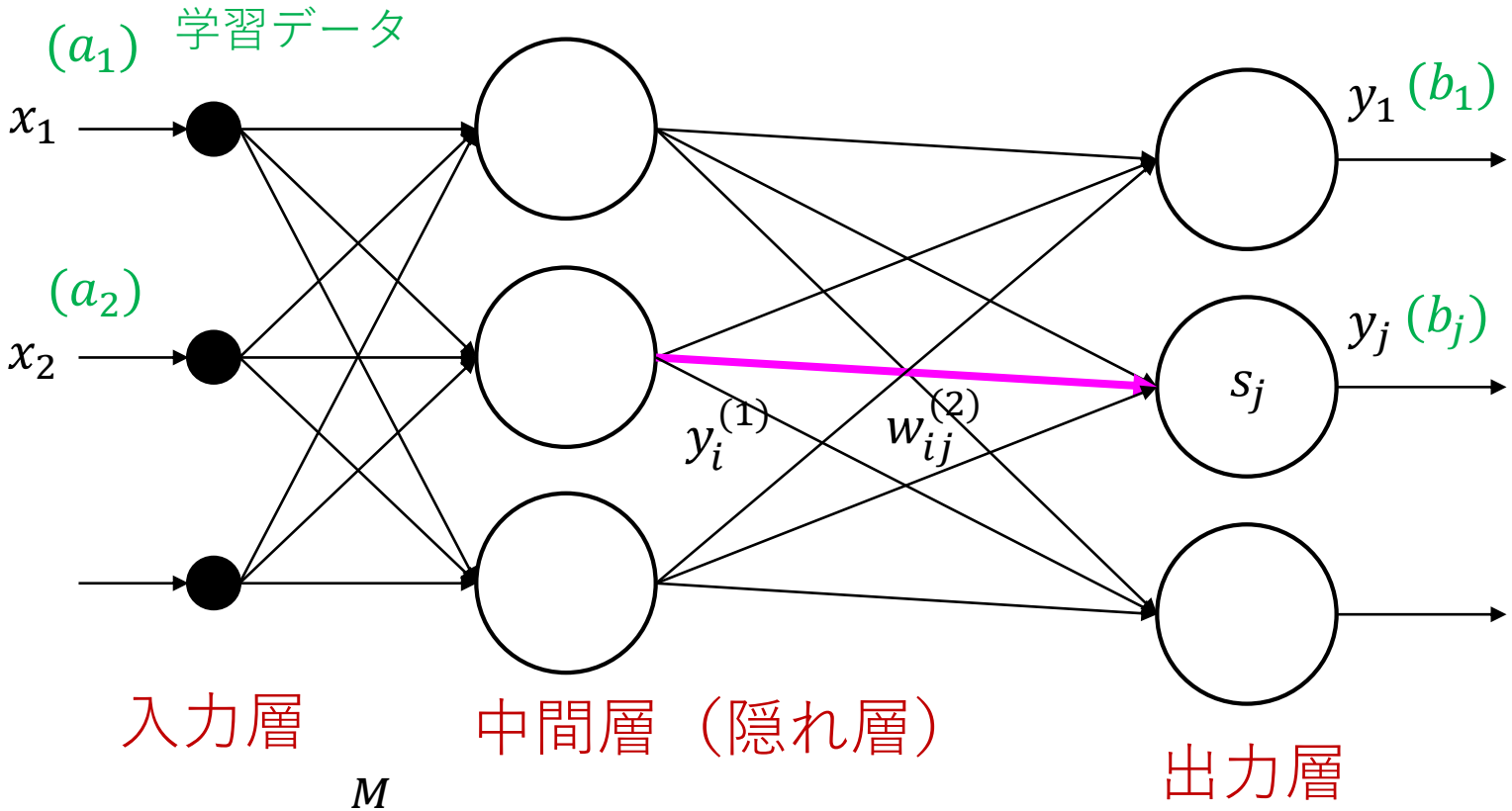
誤差が基準値より小さくなるまで
この修正を繰り返す。

全学習データの最大誤差求める。その誤差が基準値以下ならば終了。そうでなければもう一度$\boldsymbol{w}$を修正。

多層ニューロン

- 連鎖律による誤差の勾配の評価
- 誤差逆伝搬法

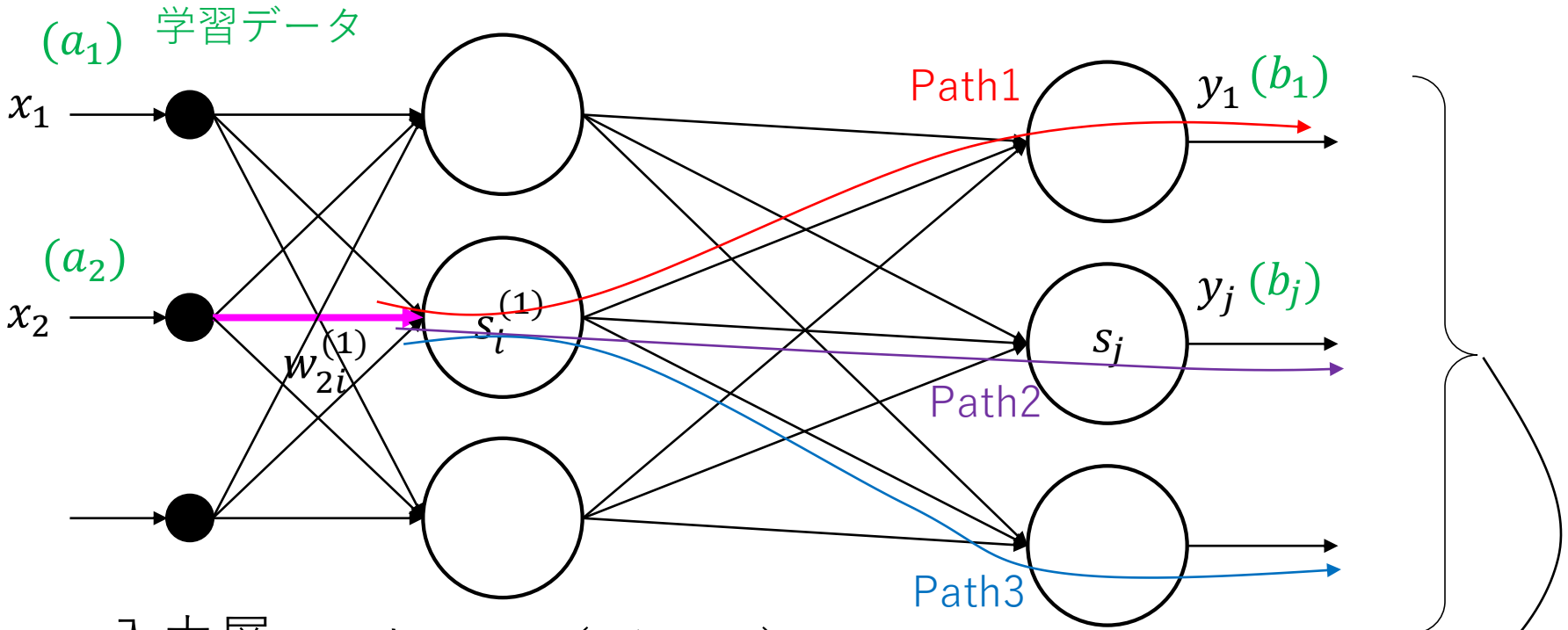
多層ニューロン(1)



$$E = \sum_{k=1}^{M} (y_k - b_k)^2$$

$$\frac{\partial E}{\partial w_{ij}^{(2)}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}^{(2)}} = 2(y_j - b_j)\alpha y_j(1 - y_j)y_i^{(1)}$$

多層ニューロン(2)



誤差 $E = \sum_{k=1}^{M} (y_k - b_k)^2$

連鎖律(Chain Rule) 付録参照

「多層ニューロン(1)」のように計算できる

$$\frac{\partial E}{\partial w_{2i}^{(1)}} = [\text{Path1の合成関数微分}] + [\text{Path1の合成関数微分}] + [\text{Path1の合成関数微分}]$$

多層ニューロン(3)

誤差　$E = E(\boldsymbol{w})$

勾配ベクトル $\nabla E(\boldsymbol{w})$

勾配法を適用

$$\boldsymbol{w} \rightarrow \boldsymbol{w} - \varepsilon \nabla E(\boldsymbol{w})$$

$\nabla E(\boldsymbol{w})$を求めるには、すべての$\boldsymbol{w}$成分による微係数を計算する必要がある。
→「多層ニューロン(2)」のように、個々の$\boldsymbol{w}$成分を計算できるのだが、結局全成分計算しないとならないので、出力側から入力側に逆に流れを追い、「多層ニューロン(1)」で計算する値を書くニューロンに保持させていく。
（これが「誤差逆伝搬法」の名前の由来である）

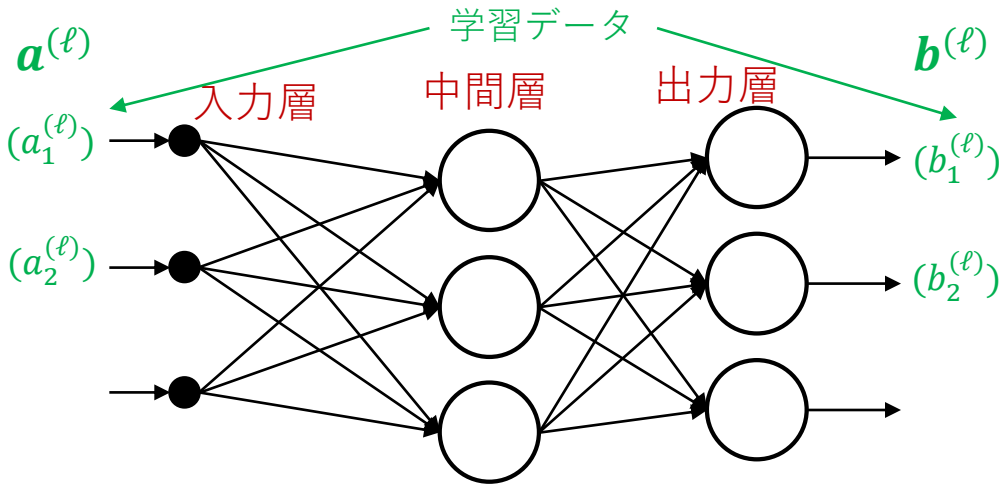
誤差逆伝搬法(Backpropagation)

全$\boldsymbol{w}$の成分の微係数を効率よく求める方法

1. 出力jに$2(y_j - b_j)$($\frac{\partial E}{\partial y_j}$の成分)を入れ、左の入力側に伝搬させる。
2. 連鎖律を考えたとき、出力層のニューロンでは、$\frac{\partial y_j}{\partial s_j}$の演算が行われるので、出力（右側）から伝搬してきた値に$\alpha y_j(1 - y_j)$($\frac{\partial y_j}{\partial s_j}$の成分)をかける。
3. さらに左に伝搬させると、出力層ニューロンの入力への重み係数による変化率は、右側から伝搬してきた値に、左のニューロンの出力y($\frac{\partial s_j}{\partial w}$の成分)をかけた値として求まる。(「多層ニューロン(1)」のスライド参照)
4. 2に戻り、左の入力層側まで同様の操作を繰り返すと全重み係数の変化率が求まる。

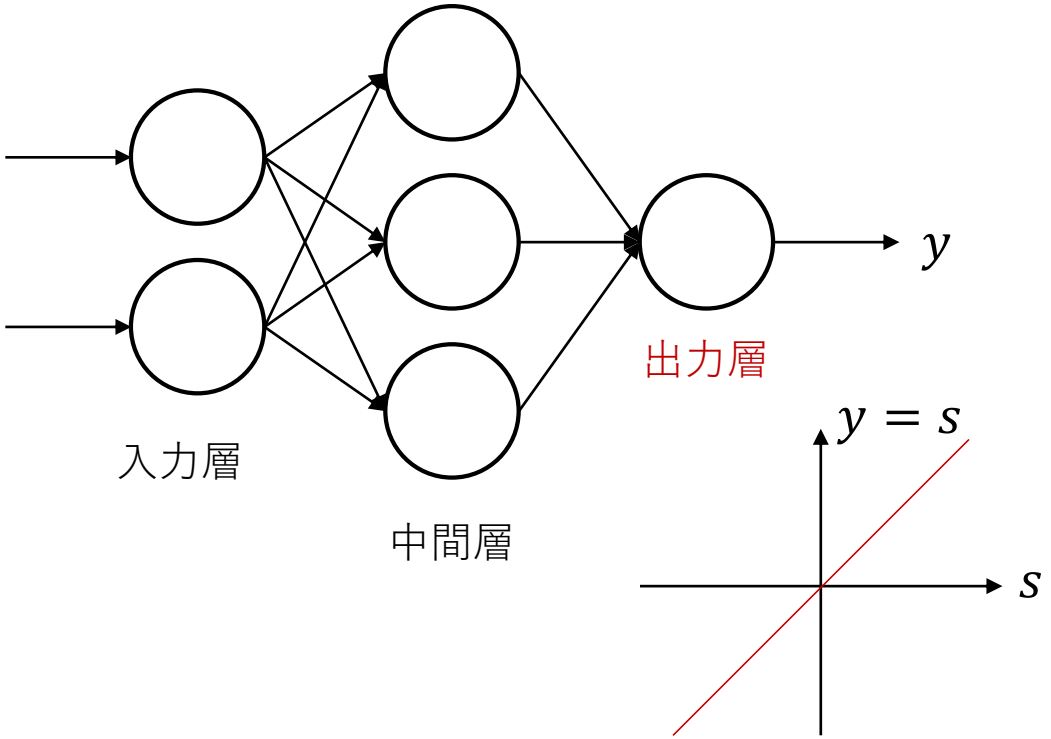
NNの応用

学習



- NNに複数($\ell = 1, \cdots, L$)の学習データ(入力$\boldsymbol{a}^{(\ell)}$, 出力$\boldsymbol{b}^{(\ell)}$)を出力するように誤差逆伝搬法を用いてニューロンの結合重み係数$\boldsymbol{w}$を求める（これが学習）。順番に学習させ、最後にもう一度全学習データの誤差を評価し、基準より小さければ学習終了、どれかが大きくなってしまっていたらも一度学習をやり直す。
- 関数の近似なので、応用例は多数ある。例えば、 $\boldsymbol{a}^{(\ell)}$ を画像のピクセル値とし、$\boldsymbol{b}^{(\ell)}$を画像の分類とすると画像認識となる。 $\boldsymbol{b}^{(\ell)}$を文字コードとすると文字認識となる。
- NNは学習データ以外のデータに対しても出力し、それが経験上うまくいく。これは、関数としての補間あるいは外挿であり、汎化能力と呼ばれる。あくまで経験上うまくいくだけで、もしかしたらうまくいかないかもしれない。それがNNの信頼性の正しい認識である。
- NNのニューロン数、層数などは経験的に上手くいく数を用いているだけ。

関数近似に用いるには



出力層で何もせず、そのまま入力値を出力すればよい。
つまり、活性化関数として線形出力する素子を用いればよい。

深層学習（ディープラーニング）とは

- これまでのニューラルネットワークの説明では、入力層、中間層（ここが処理のメイン）、出力層の３層構造であった。
- 中間層は1層ではなく、より多くの複数の層用いるのが深層学習である。
- ハードウェアの発展で演算能力が飛躍的に高まったので、深層学習の試行錯誤が可能となった。
- 深層学習とは言っても、ニューラルネットワークの中間層が複数層あるだけである。ただ、畳み込みニューラルネットワーク(CNN; Convolutional Neural Network)、再帰型ニューラルネットワーク(RNN; Recurrent Neural Network)という工夫を追加したものもある。

付録

連鎖律(Chain Rule)

多変数合成関数の微分公式

例）

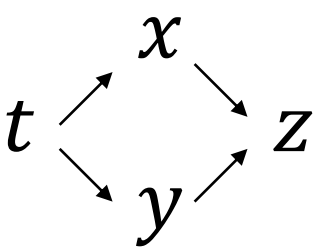
$$z = t^5 \quad \Rightarrow \quad \frac{\partial z}{\partial t} = 5t^4$$

$$t \longrightarrow z$$

ここで、次のような合成関数になっていたとする。

$$z = xy$$

$$\begin{cases} x = t^2 \\ y = t^3 \end{cases}$$



連鎖律

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial t}$$

$$= y(2t) + x(3t^2)$$

$$= 5t^4$$

媒介変数x, yの2変数以上でも和が増えるだけであり、中間に他の媒介変数が入ったら積が増えるだけである。もしこの問題で媒介変数が1つならば、高校で習う合成関数の微分となる（つまり、連鎖律はその拡張）。

参考書

1. 熊沢逸夫, 学習とニューラルネットワーク, 森北出版, 1998.

まとめ

- ニューラルネットワーク、深層学習はあくまで小脳モデルであり、関数の近似（＋汎化（補間、外挿））を活用しているにすぎない。（それでも、人間が複雑と思っていたことは結構単純なものが多く、それらの能力は超えることがあるだろうが）
- 現在、AI、AI・・・と盛んに言われているが、そもそもAIの定義さえしておらず、一見、人間にしかできないと思えていた作業を自動化したら「AIだ」と言っているだけで、昔から行われていることの範囲が少し広がっただけにすぎない。今の技術に知能はまだない（あるように見せかけている場合はある）。
- 人間が大脳で考えている仕組み、意識・感情というものはまだ解明されていない。大脳の仕組みがニューロンの活動として解明、モデル化されてコンピュータシミュレーションで実行できるようにならなければ、いくら演算が早くなり、メモリは増えて人間の脳のレベルを超えようとも人間のような能力を持ったとは言えないであろう（現在、そのような解明ができそうな予兆すら感じられず、まだまだ先だと思われる）。
- ただ、大脳の働きもニューロンによるものと思われるので（私はそう思う）、一度解明され、モデル化されれば、SF映画のように人間が一気に機械に負ける時代（シンギュラリティーと言われている）が来るであろう。