

# しおん研-ゼロつくゼミ

第3回

# 今日やること (P.66~P.109)

- ▶ 出力活性化関数の意味と実装
- ▶ Mnistデータとは
- ▶ Mnistデータを用いた3層ニューラルネットワークの推論過程の実装 (今回も、パラメータの学習は行いません)
- ▶ パラメータ学習のフレームワークと損失関数
- ▶ 勾配降下法 (ちょっとだけ)

# ニューラルネットワークの出力層活性化関数

- ▶ ニューラルネットワークの用途によって異なる。

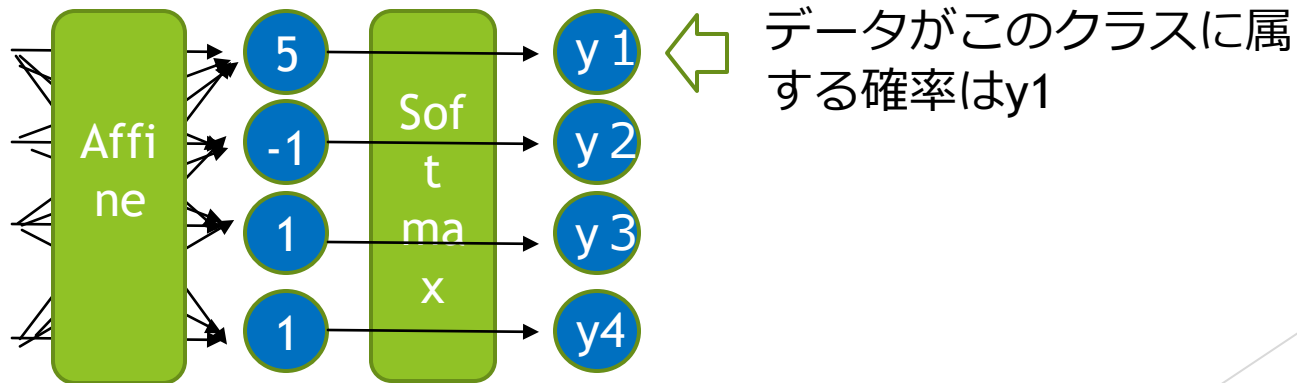
回帰問題：数値を予測する問題

分類問題：データがどのクラスに属するか、を決定する問題

- ▶ 出力層の活性化関数は、分類問題ならSoftmax関数、回帰問題なら恒等関数をあてる。

(なんで?)

- ▶ 分類問題における出力の意味づけ：「ターゲットのデータが各クラスに属する確率」を表す



## Softmax 関数

活性化関数はソフトマックス関数と呼ばれる。

$$\text{活性化関数: } \varphi(u_k) = \frac{e^{u_k}}{\sum_{i=1}^K e^{u_i}}$$

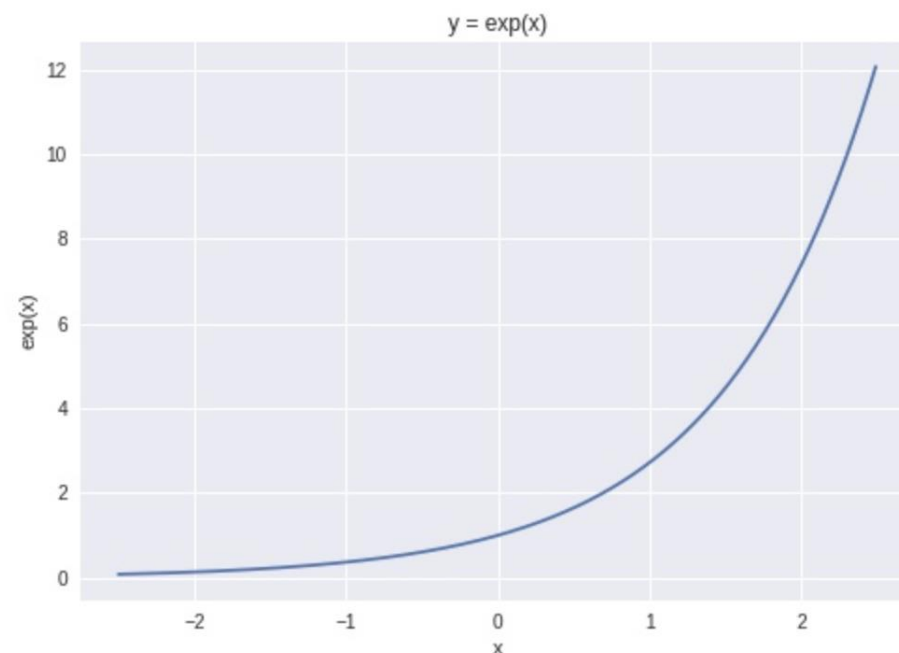
$u = [u_1, u_2, \dots, u_K]$  (流れてきたデータ一つ分)

(実装はGoogle Colabにて)

# どうしてこれが「確率」になるのか？

- ▶ データを $[0, 1]$ の区間に収めるため
- ▶ データの大小を一致させるため

$$(-3 < 1 = 1 < 5)$$



(出典

<https://gensasaki.hatenablog.com/entry/2018/08/30/042807>)

# 実際にNNでデータを推測してみよう①

## ～mnistデータの読み込み～

mnistデータ とは？

機械学習の開発によく使われる、手書き文字のデータセット。Kerasなどのパッケージにもあるが、今回はゼロつく添付のライブラリから取ってくる。

(Deep\_learning\_scratch > detasets > mnist )

データセットの中身

- ・ 画像（訓練用： $1 \times 28 \times 28 \times 60000$  テスト用： $1 \times 28 \times 28 \times 10000$ ）
- ・ ラベル（訓練用：60000 (or  $10 \times 60000$ )  
テスト用：10000 (or  $10 \times 60000$ ))

実装は Google  
Colaboratory で



# 実際にNNでデータを推測してみよう②

## ～バッチ処理～

- ▶ 今まで「入力データ」として扱ってきたデータは全て1次元配列（1つのデータ）
- ▶ 実際は複数のデータをまとめて処理する必要がある
- ▶ もし仮に、100枚の画像データを同時に分類するなら・・・？



(数字は次元量)



# ニューラルネットワークの実装 (第2回スライドより)

- ▶ 入力のノードのベクトルを  $X = [x_1, x_2, \dots, x_{n1}]$  とする
- ▶ 重みつき入力信号の総和を  $A = [a_1, a_2, \dots, a_{n2}]$  とする
- ▶ 重みパラメータを  $W = [[w_{11}, w_{21}, \dots, w_{n11}],$

$[w_{12}, w_{22}, \dots, w_{n12}]$

$[w_{1n1}, w_{2n}, \dots, w_{n2n1}]$  とする

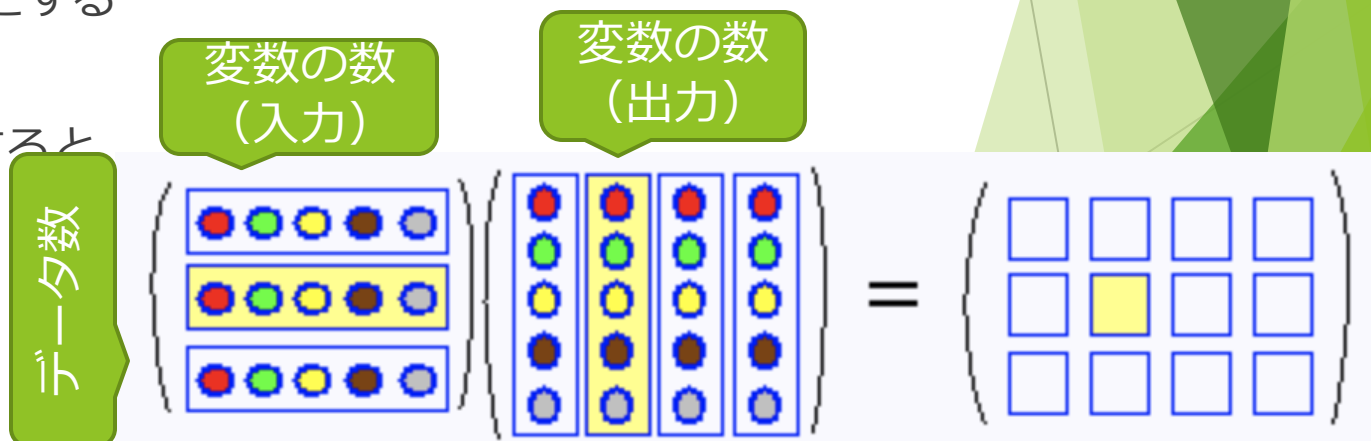
バイアスを  $B = [b_1, b_2, \dots, b_{n2}]$  とする

- ▶ 出力ノードのを  $Y = [y_1, y_2, \dots, y_{n2}]$  とすると

- ▶ ニューラルネットワーク1層分 :

$$Z = XW + B \quad (A=XW)$$

$$Y = h(Z)$$







# 実際にNNでデータを 推測してみよう③ ～分類と可視化～

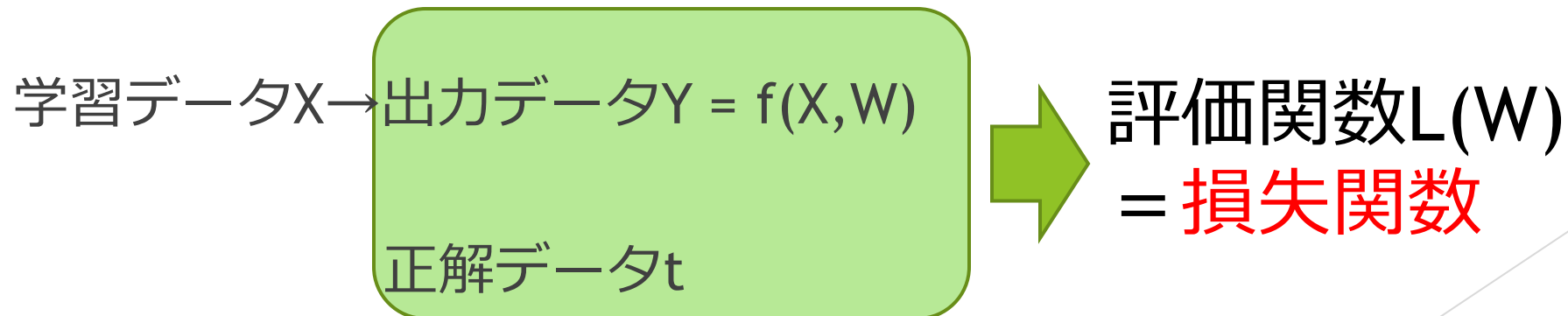
(実装はGoogle Colabで)

# パーセプトロンとニューラルネットワークの違い（復習）

	単層パーセプトロン	多層パーセプトロン	ニューラルネットワーク
層の数	1つ	複数	複数
活性化関数	Step関数	Step関数	Sigmoid/Relu/Softmax関数
パラメータの自動学習	なし 	なし 	なし 

# 学習のフレームワーク

- ▶ 目的は、推論に最適な（＝もっとも適切な出力を返す）パラメータを見つけること！
- ▶ そのためにはどうすればいい？
  - ある推論に対して、それがどれくらい正しいか評価する関数を定義し、その評価が最大になるようにパラメータを定めれば良いのでは？



# 損失関数として使われる誤差

one-hot 表現

▶ 出力値 :  $[y_1, y_2, \dots, y_n]$ 、正解ラベル :  $[t_1, t_2, \dots, t_n]$

▶ 回帰問題 : 2乗和誤差 
$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

▶ 分類問題 : クロスエントロピー誤差 
$$E = - \sum_k t_k \log y_k$$

これらを最小にするパラメータを見つける !

# クロスエントロピー誤差とは

$$E = - \sum_k t_k \log y_k$$

例：  $y = [y_1, y_2, \dots, y_n]$ ,  $t = [0, 0, \dots, 1, \dots, 0]$  (k番目のみが1のone-hotベクトル) であるとき

$$E = -\log y_k$$

となり、正解ラベルのとりうる確率のみに依存する。

複数のデータにおける損失は、各データのEの平均値となる。



# クロスエントロピー 誤差の実装 (Google Colabより)



# 分類問題にクロスエントロピー誤差を使う理由（考えてきてください）

- ▶ (c.f.) どうしてaccuracyじゃいけないの？



# バッチ学習とは

- ▶ 先程、複数データを1度に処理する方法を学習したが、流石に60000個もの訓練データを一度に学習するのは計算量が馬鹿にならない・・・

→訓練データから「ミニバッチ」と呼ばれるデータ集合をランダムに取り出し、ミニバッチにおける損失の平均を訓練データ全体の平均と近似する！

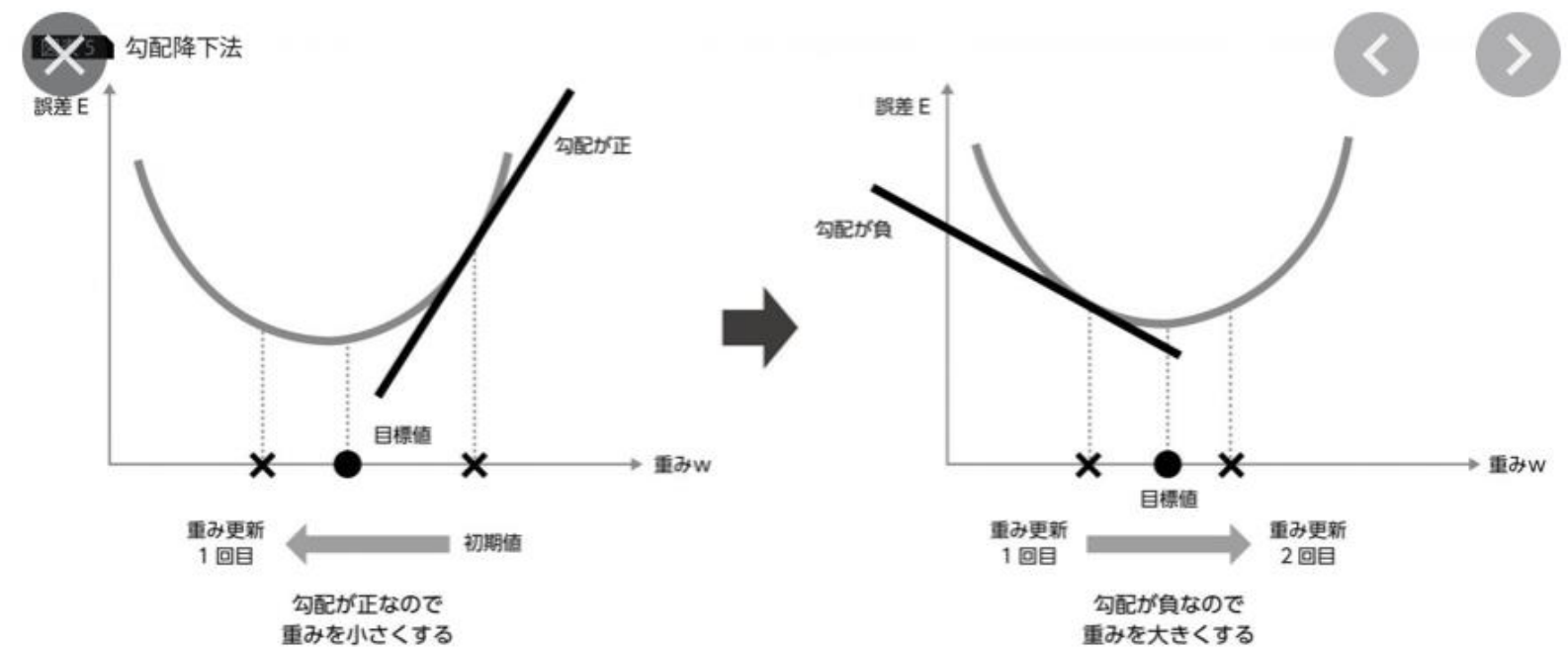
(やってることは標本平均の近似と同じ)

(c.f.) ミニバッチ単位で学習を行うのには、もう1つ大きな理由がある(っていうかこっちが本質的かも)がありますが、次回以降説明します。

# ミニバッチを用いた クロスエントロピー 誤差の実装

# 損失関数から、どうやって最小をとるパラメータを見つけるの？：それぞれのパラメータで微分！

- ▶ しかし、 $f(X, W)$ があまりにも複雑なので、 $L'(W) = 0$ 行列 となる解 $W$ をピンポイントで見つけるのはまあ無理。
- ▶ → 勾配降下法の利用！



出典：

<https://www.imagine.co.jp/再帰型ニューラルネットワークの「基礎の基礎」/>

# 勾配降下法の 3 ステップ

- ▶ 1、 $L(W)$ の勾配 (grad) を求める
- ▶ 2、勾配の値に応じてパラメータを更新する
- ▶ 1,2を繰り返す

(質問) ステップ 1 について

「どんな関数でも、ある値における微分値が必ず求まる」方法とは？