RITSUMEIKAN

機械学習 第８回 ニューラルネットワーク

立命館大学 情報理工学部

福森 隆寛

Beyond Borders

講義スケジュール

担当教員１：福森（第1回～第15回）

1	機械学習とは、機械学習の分類
2	機械学習の基本的な手順
3	識別（１）
4	識別（２）
5	識別（３）
6	回帰
7	サポートベクトルマシン
8	**ニューラルネットワーク**

9	深層学習
10	アンサンブル学習
11	モデル推定
12	パターンマイニング
13	系列データの識別
14	半教師あり学習
15	強化学習

担当教員２：叶昕辰先生（第16回の講義を担当）

今回の講義内容

- 取り扱う問題の定義
- ニューラルネットワークの計算ユニット 神经网络计算单元
- フィードフォワード型ニューラルネットワーク 前馈神经网络 Feed-forward
  - ニューラルネットワークの構成 神经网络的结构
  - 誤差逆伝播法(ごさぎゃくでんぱんほう)による学習 错误反向传播学习法
- ニューラルネットワークの深層化
  - 勾配消失(しょうしつ)問題 梯度消失问题
  - 活性化(かっせいか)関数 激活函数
- 演習問題

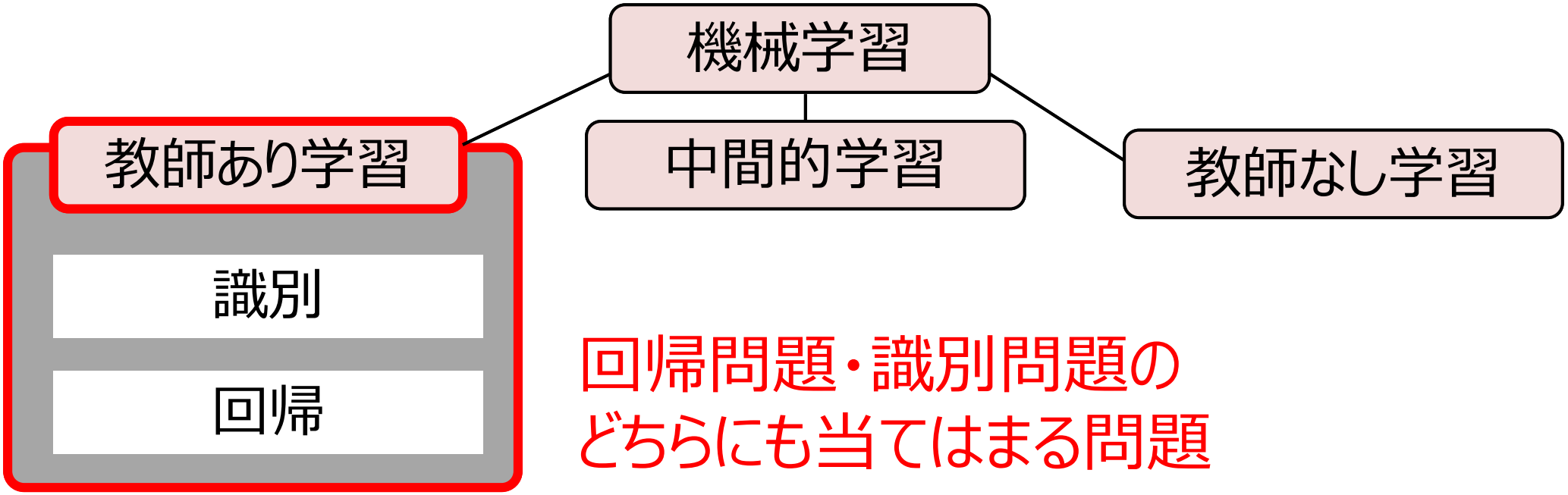
取り扱う問題の定義：教師あり問題

- 特徴ベクトルを入力して、それをクラス分けする識別器、または、それに対応する数値を出力する関数を作る

※ 教師あり学習の問題での学習データは、以下のペアで構成される

入力データの特徴ベクトル ← $\{\boldsymbol{x}_i, y_i\}, \quad i = 1,2,\ldots,N$ → 学習データの総数

正解情報

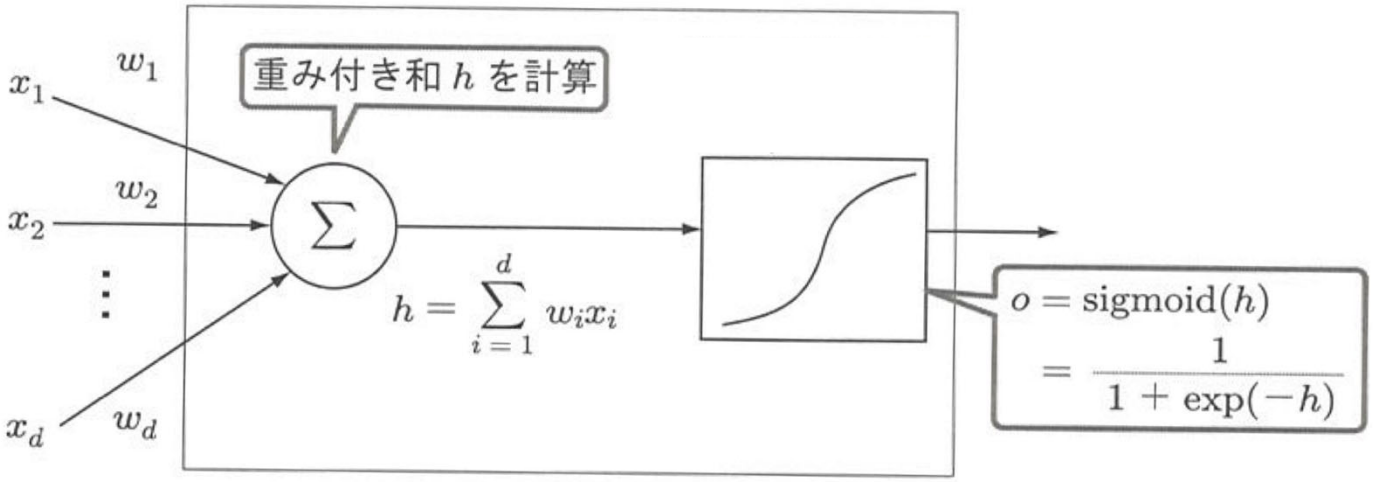


回帰問題・識別問題の
どちらにも当てはまる問題

ニューラルネットワークの計算ユニット

ニューラルネットワーク Neural Network

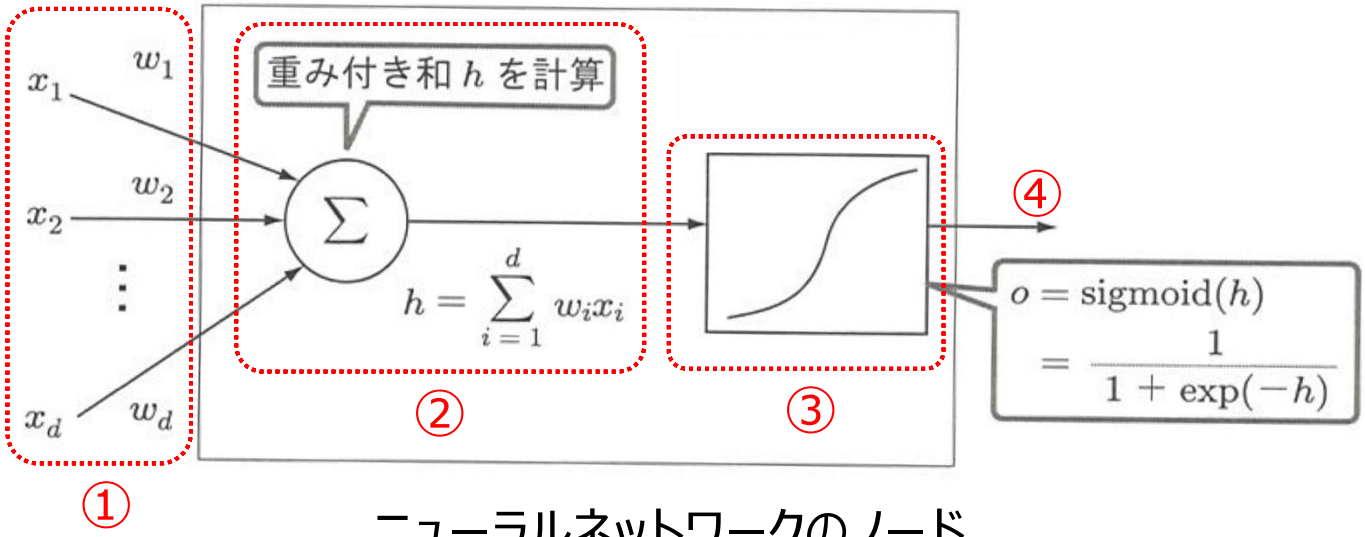
- 下図のような計算ユニットを階層的に組み合わせて、入力から出力を計算するメカニズム
  - 単純な論理演算で実現できることが特徴
- 生物の神経細胞（せいぶつ しんけいさいぼう）のはたらきを単純化したモデルであると考えられている



ニューラルネットワークのノード

ニューラルネットワークの計算ユニット

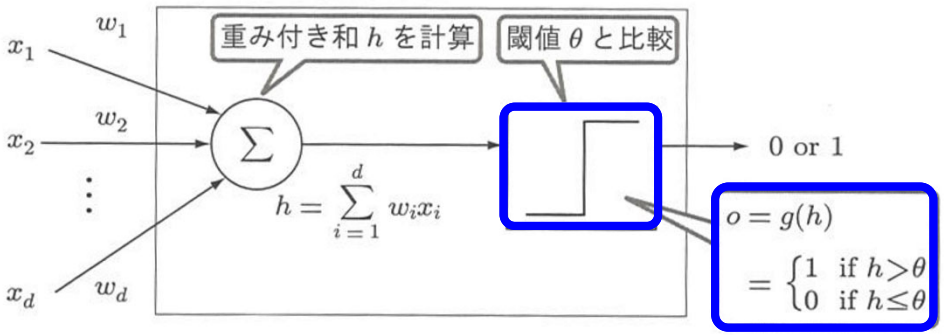
- ニューラルネットワークのノードの動き
  1. 神経細胞への入力x_iは、それぞれ異なる重みw_iをもつ
  2. 入力x_iと重みw_iの重み付き和としてhが計算される
  3. 重み付き和hを活性化関数（神経細胞）に代入し、その出力信号oを出力する　激活函数
  4. 出力信号oが他の神経細胞へ重み付きで伝搬（でんぱん）される



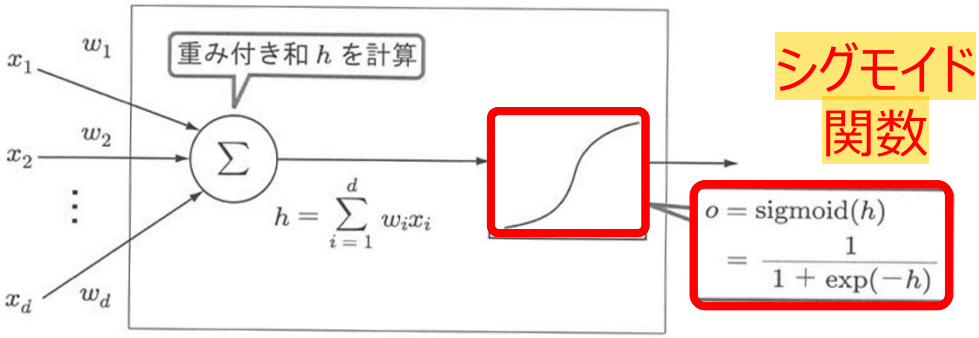
ニューラルネットワークのノード

ニューラルネットワークの計算ユニット

- 単層パーセプトロンとの違い
  - 閾値関数
    - 単層パーセプトロン：ステップ関数
    - ニューラルネットワークのノード：シグモイド関数 sigmoid函数
      - シグモイド関数は微分可能なので、使用原因： 可微， 可以反向传播
        後ほど説明する誤差逆伝搬法を利用できる



単層パーセプトロン　　ステップ関数

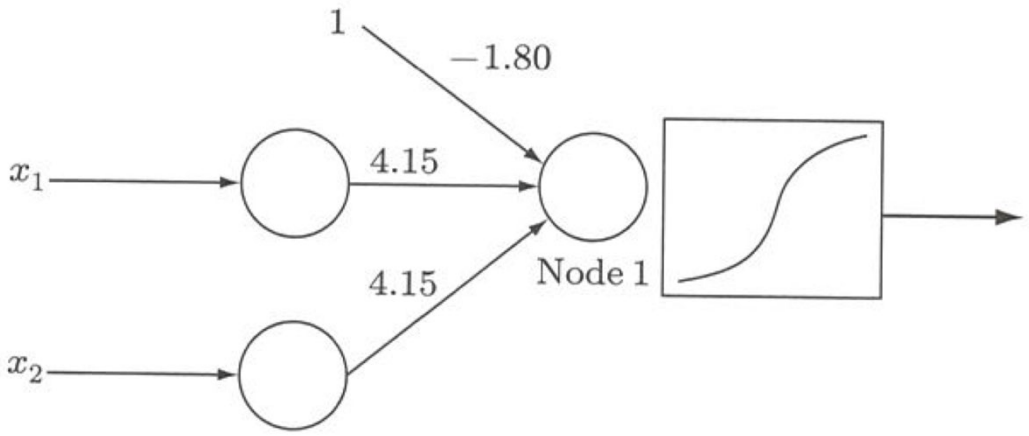


シグモイド関数

ニューラルネットワークのノード

演習問題8-1（10分間）

- 以下のニューラルネットワークのノードが与えられたとする
  1. 入力x_1とx_2が取りうる値が0と1のどちらかであるとき、x_1とx_2の全ての組み合わせに対する出力を計算せよ
  2. 1.の結果より、どのような入力が与えられると高い出力が得られやすいか考察しなさい



ニューラルネットワークのノード

入力		出力
x_1	x_2	
0	0	0.142
0	1	0.913
1	0	0.913
1	1	0.998

演習問題 解答例

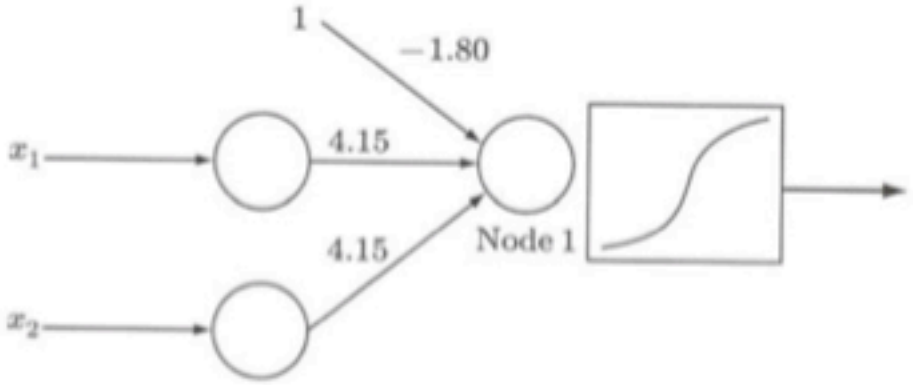
1. 入力x_1とx_2が取りうる値が0と1のどちらかであるとき、
x_1とx_2の全ての組み合わせに対する出力を計算せよ

x_1, x_2 に対する出力の値は、以下の式で算出できる

$$f(x_1, x_2) = \frac{1}{1 + \exp\{-(1 \cdot (-1.80) + 4.15 \cdot x_1 + 4.15 \cdot x_2)\}}$$

$f(x_1 = 0, x_2 = 0) \cong 0.142$　　$f(x_1 = 0, x_2 = 1) \cong 0.913$

$f(x_1 = 1, x_2 = 0) \cong 0.913$　　$f(x_1 = 1, x_2 = 1) \cong 0.998$



ニューラルネットワークのノード

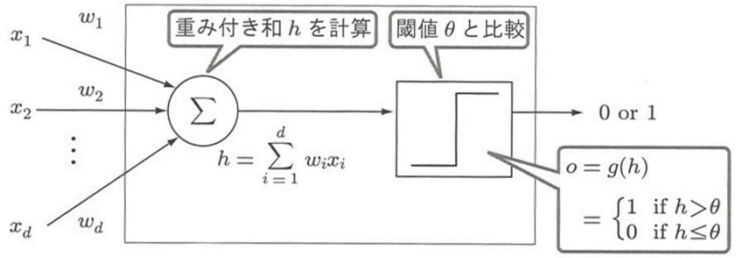
入力		出力
x_1	x_2	
0	0	0.142
0	1	0.913
1	0	0.913
1	1	0.998

フィードフォワード型ニューラルネットワーク

ロ ニューラルネットワーク（多層パーセプトロン）

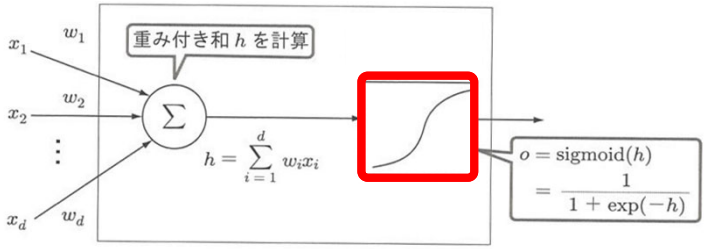
■ パーセプトロンをノードとして、階層状に結合したもの

線形識別面を構成

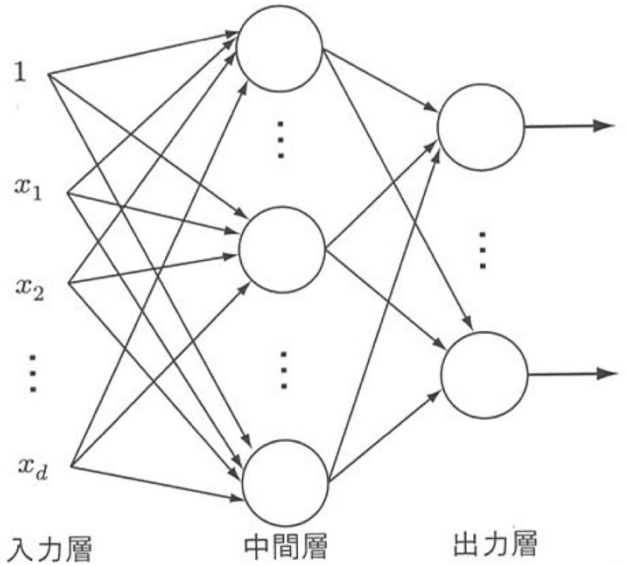


誤り訂正学習における
単層パーセプトロン

以下のように計算ユニットを変更して
さらに階層状に重ねると…



非線形識別面を実現



ニューラルネットワーク

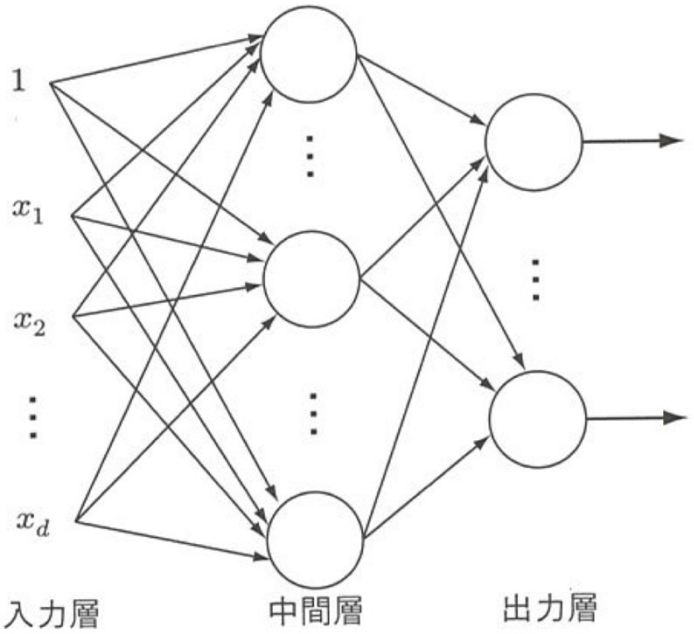
フィードフォワード型ニューラルネットワーク

- ちなみに…
  - 脳のニューロンは、綺麗（きれい）な階層状で結合しておらず
    - 階層内の結合
    - 階層を飛ばす結合
    - 入力側に戻る結合

    が入り乱（みだ）れていると考えられる
  - このような任意の結合をもつニューラルネットワークモデルでは学習が非常に複雑になるので、本講義では、簡単のために3階層のフィードフォワード型ネットワークを対象とする

フィードフォワード型ニューラルネットワーク

- 3階層のフィードフォワード型モデルを考える
  - **入力層**：特徴ベクトルを入力
    - ノード数＝特徴ベクトルの次元数
  - **出力層**：識別結果を出力
    - ノード数＝識別対象のクラス数（識別問題の場合）
  - **中間層**
    - **隠れ層**とも言う
    - ノード数＝入力層と出力層の
      ノード数に応じた適当な数

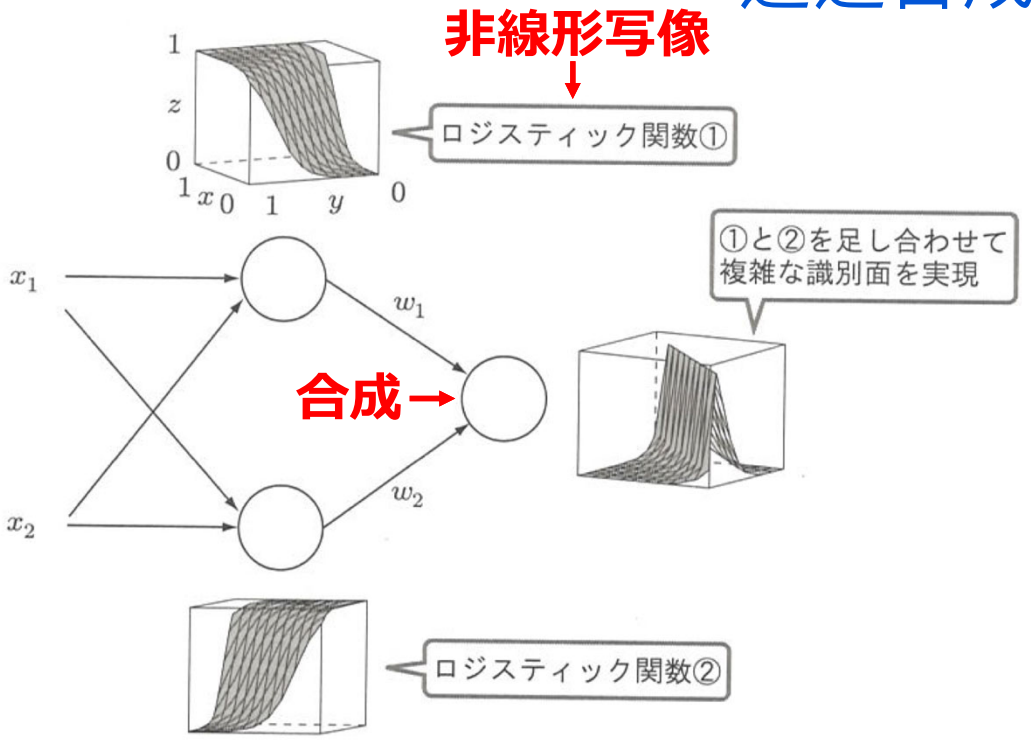


3階層フィードフォワード型
ニューラルネットワーク

フィードフォワード型ニューラルネットワーク

- ニューラルネットワークのようにノードを段階的に組むと非線形識別面を実現できる理由

通过合成非线性映射实现非线性识别面



ニューラルネットワークによる
非線形識別面の実現

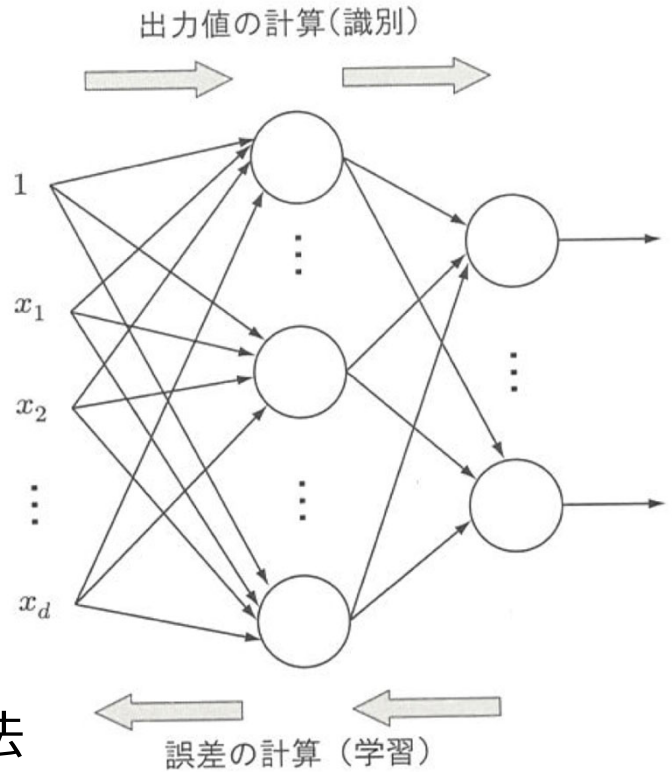
非線形写像（ごうせい）の合成によって
非線形識別面を実現

- 個々のノードの出力は、**ロジスティック関数の出力**（**非線形写像**）
- この関数の出力に重みを加えて足し合わせることで、データの境界の形に合わせた識別面を構成

フィードフォワード型ニューラルネットワーク

- ニューラルネットワーク（非線形識別面）のパラメータを獲得するための考え方
  - 出力層が正しい値を出力しなければ、その値の算出に寄与した中間層の重みが、出力層の更新量に応じて修正されるべき

- 誤差逆伝播法
  - 出力層の誤差を計算し、その誤差を中間層に伝播させて学習する手法



誤差逆伝播法

フィードフォワード型ニューラルネットワーク

- ニューラルネットワークでも、出力とターゲットの二乗誤差$(y_i - o_i)^2$が最小になるように学習
- 重み$\boldsymbol{w}$の学習

  誤差の二乗和 $E(\boldsymbol{w})$

$$E(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{\boldsymbol{x}_i \in D}(y_i - o_i)^2 = \frac{1}{2}\sum_{\boldsymbol{x}_i \in D}\left[y_i - \frac{1}{1+\exp\{-(\boldsymbol{w}\cdot\boldsymbol{x}_i)\}}\right]^2$$

  ここでは、最急勾配法を用いて、重み$\boldsymbol{w}$を学習

$$w_j \leftarrow w_j - \eta\frac{\partial E}{\partial w_j}$$

フィードフォワード型ニューラルネットワーク

重み$\boldsymbol{w}$の学習（つづき）

誤差の二乗和 $E(\boldsymbol{w})$ の勾配方向

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{\boldsymbol{x}_i \in D}(y_i - o_i)^2$$

$$= \frac{1}{2}\sum_{\boldsymbol{x}_i \in D}\frac{\partial}{\partial w_j}(y_i - o_i)^2$$

$$= \sum_{\boldsymbol{x}_i \in D}(y_i - o_i)\frac{\partial}{\partial w_j}(y_i - o_i)$$

$$= -\sum_{\boldsymbol{x}_i \in D}(y_i - o_i)\frac{\partial o_i}{\partial w_j}$$

$$\frac{\partial}{\partial w_j}(y_i - o_i) = \frac{\partial y_i}{\partial w_j} - \frac{\partial o_i}{\partial w_j}$$

$\frac{\partial y_i}{\partial w_j} = 0$ なので

$$= -\frac{\partial o_i}{\partial w_j}$$

フィードフォワード型ニューラルネットワーク

重み$\boldsymbol{w}$の学習（つづき）

誤差の二乗和 $E(\boldsymbol{w})$ の勾配方向

$$\frac{\partial E}{\partial w_j} = -\sum_{x_i \in D} (y_i - o_i) \frac{\partial o_i}{\partial w_j} = -\sum_{x_i \in D} (y_i - o_i) o_i (1 - o_i) x_{ij}$$

出力o_iを重みw_jで偏微分したものは、以下の合成微分で求める

$$\frac{\partial o_i}{\partial w_j} = \frac{\partial o_i}{\partial h_i} \frac{\partial h_i}{\partial w_j}$$

ここで、$o_i = \dfrac{1}{1 + \exp(-h_i)}$, $h_i = w_0 + w_1 x_{i1} + \cdots w_d x_{id}$

シグモイド関数　　入力の重み付き和

$$\frac{\partial o_i}{\partial h_i} = o_i (1 - o_i)$$

シグモイド関数の微分

$$\frac{\partial h_i}{\partial w_j} = x_{ij}$$

← i番目の学習データに対するj次元目の要素

入力の重み付き和の微分

フィードフォワード型ニューラルネットワーク

□ 重み$\boldsymbol{w}$の学習（つづき）

最急勾配法による
重み$\boldsymbol{w}$の更新式

$$w_j \leftarrow w_j - \eta \frac{\partial E}{\partial w_j}$$

と

誤差の二乗和 $E(\boldsymbol{w})$ の
勾配方向

$$\frac{\partial E}{\partial w_j} = -\sum_{\boldsymbol{x}_i \in D} (y_i - o_i) o_i (1 - o_i) x_{ij}$$

より出力層の重みの更新式は、以下の通りとなる

由于神经网络的训练通常使用随机梯度下降，所以经常删避免对所有数据求和的操作

$$w_j \leftarrow w_j + \eta \sum_{\boldsymbol{x}_i \in D} (y_i - o_i) o_i (1 - o_i) x_{ij}$$

通常、ニューラルネットワークの学習は確率的最急勾配法を用いるので
全データに対して和をとる操作を削除することが多い

さくじょ

フィードフォワード型ニューラルネットワーク

□ 誤差逆伝播法のアルゴリズム

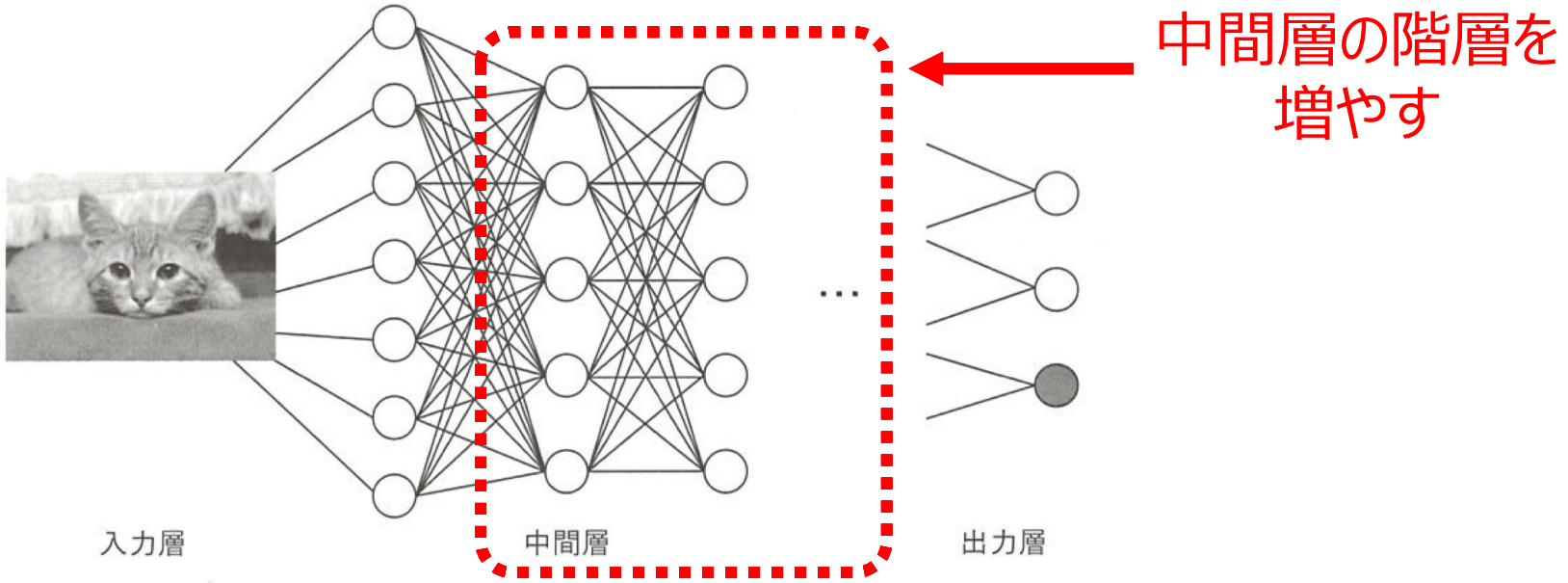
入力：正解付学習データ $D\{(\boldsymbol{x}_i, y_i)\}, i = 1, \ldots, N$ （数値特徴）
出力： 重み $\boldsymbol{w}$
重み $\boldsymbol{w}$ を適当な値で初期化
Repeat
　　for all $\boldsymbol{x} \in D$ **do**
　　$\boldsymbol{o} \leftarrow \mathrm{nn}(\boldsymbol{w} \cdot \boldsymbol{x})$
　　/* 出力層のk番目のユニットに対してエラー量δ_kを計算 */
　　$\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k)$
　　/* 中間層のh番目のユニットに対してエラー量δ_hを計算 */
　　$\delta_h \leftarrow o_k(1 - o_k) \sum_k w_{kh}\, \delta_k$
　　$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \delta_j \boldsymbol{x}$
　　end for
until 修正量が閾値以下 or 決められた回数

$\mathrm{nn}(\boldsymbol{w} \cdot \boldsymbol{x})$：
重み$\boldsymbol{w}$のニューラルネットワークに
$\boldsymbol{x}$を入力したときの出力層の出力を
成分として並べたベクトルを返す関数

中間層は、修正量δ_kを
重み付きで足し合わせる
（た　あ）

ニューラルネットワークの深層化

- 人間の神経回路網
  - 3階層のフィードフォワード型ネットワークより、遥かに複雑な構造であるはず
    - ニューラルネットワークの中間層を増やして（階層を深くして）性能を向上させようと考えるのは当然である



しかし、階層を深くすると、うまく学習できないことがある…なぜか？

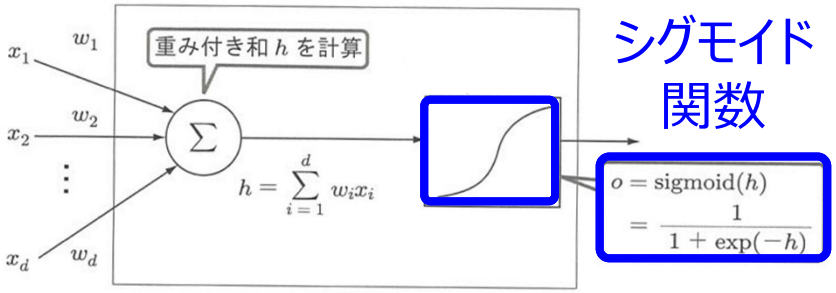
ニューラルネットワークの深層化

- **勾配消失問題** 校正量(梯度)会随着返回到输入层的方向越变越小
  - 中間層の階層を深くした多層ネットワークに対して誤差逆伝播法によって重みを学習したときに重みの修正量が入力層の方向へ戻るにつれて小さくなる問題
  - 特に入力層に近い層の重みは、更新を繰り返しても値があまり変わらない
    - 重みが変わらないということは、性能も一向に変わらない
- 勾配消失問題の解決手段
  - 事前学習法（次回に紹介）
  - 活性化関数に対する工夫

ニューラルネットワークの深層化

様々な活性化関数

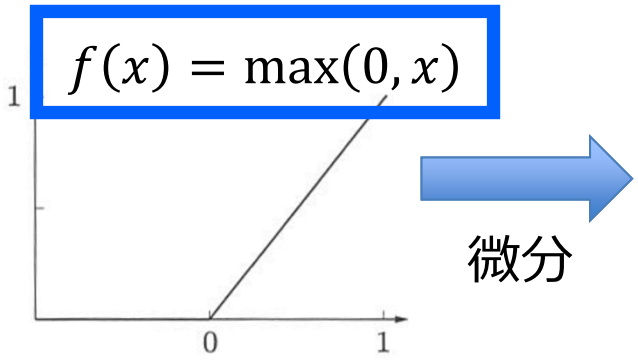
- ノードの活性化関数にシグモイド関数以外を使う
- **ReLU（rectified linear unit）**
  - Rectified linear関数（引数(ひきすう)が負のときは0、0以上のときはその値を出力する関数）を活性化関数としたユニット
  - 半分の領域(りょういき)で勾配が1（誤差が消失しない）
  - 勾配(こうそく)を高速に計算できる（0または1）
  - 多くのユニットの出力が0であるスパースなネットワークになる



シグモイド関数

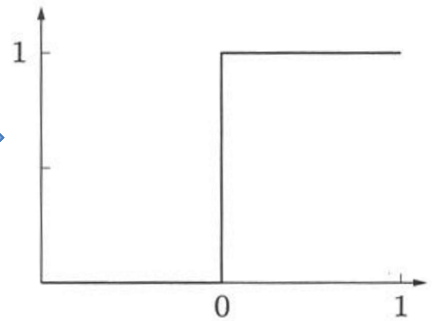
$o = \mathrm{sigmoid}(h) = \frac{1}{1 + \exp(-h)}$

従来のニューラルネットワークのノード



rectified linear関数

微差分



導関数(どうかんすう)

演習問題8-2（10分間）

- シグモイド関数を活性化関数にすると
  多層ニューラルネットワークの重みを学習するときに、
  勾配消失問題が発生(はっせい)しやすい理由を考えよ