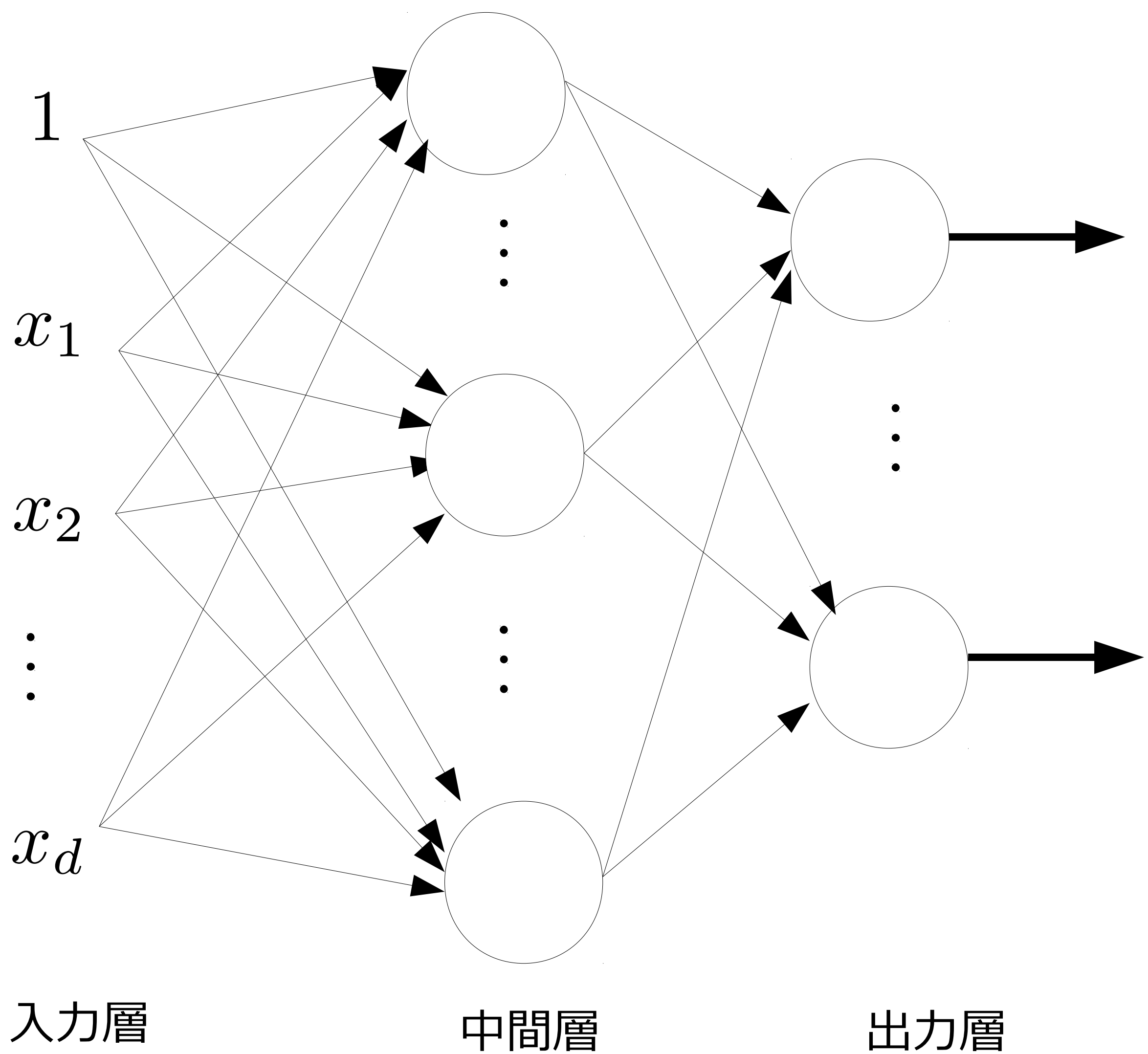
8. ニューラルネットワーク

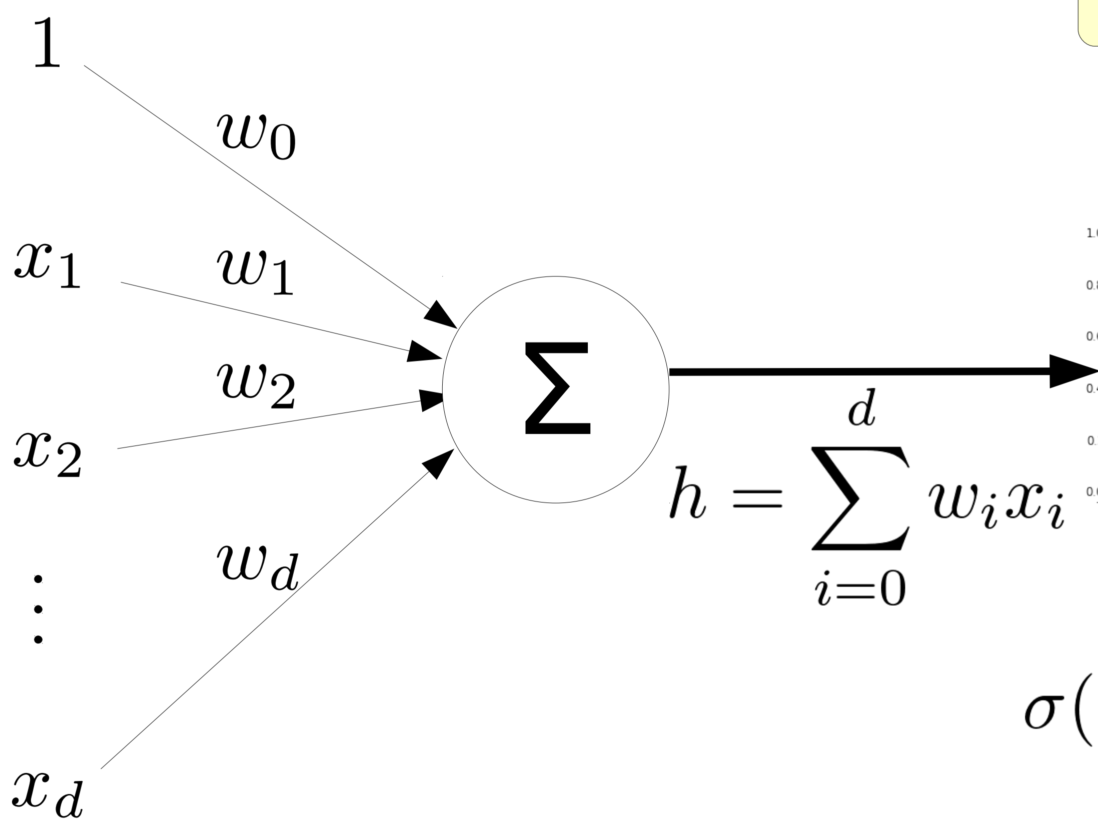
- 非線形関数ユニットの階層的組み合わせ



8.1 ニューラルネットワークの計算ユニット

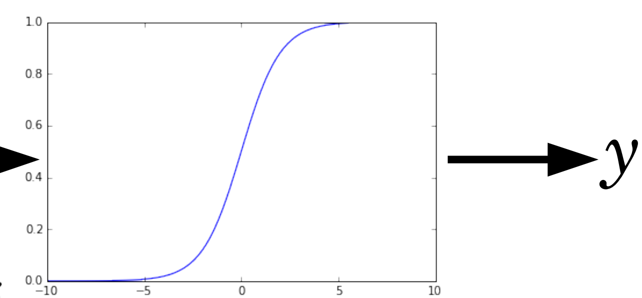
- 活性化関数にシグモイド関数を採用
  - 多層の誤差修正に対応するために、勾配計算の際に微分可能な活性化関数を用いる

ロジスティック識別



$$h = \sum_{i=0}^{d} w_i x_i$$

活性化関数



シグモイド関数

$$\sigma(h) = \frac{1}{1 + e^{-h}}$$

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

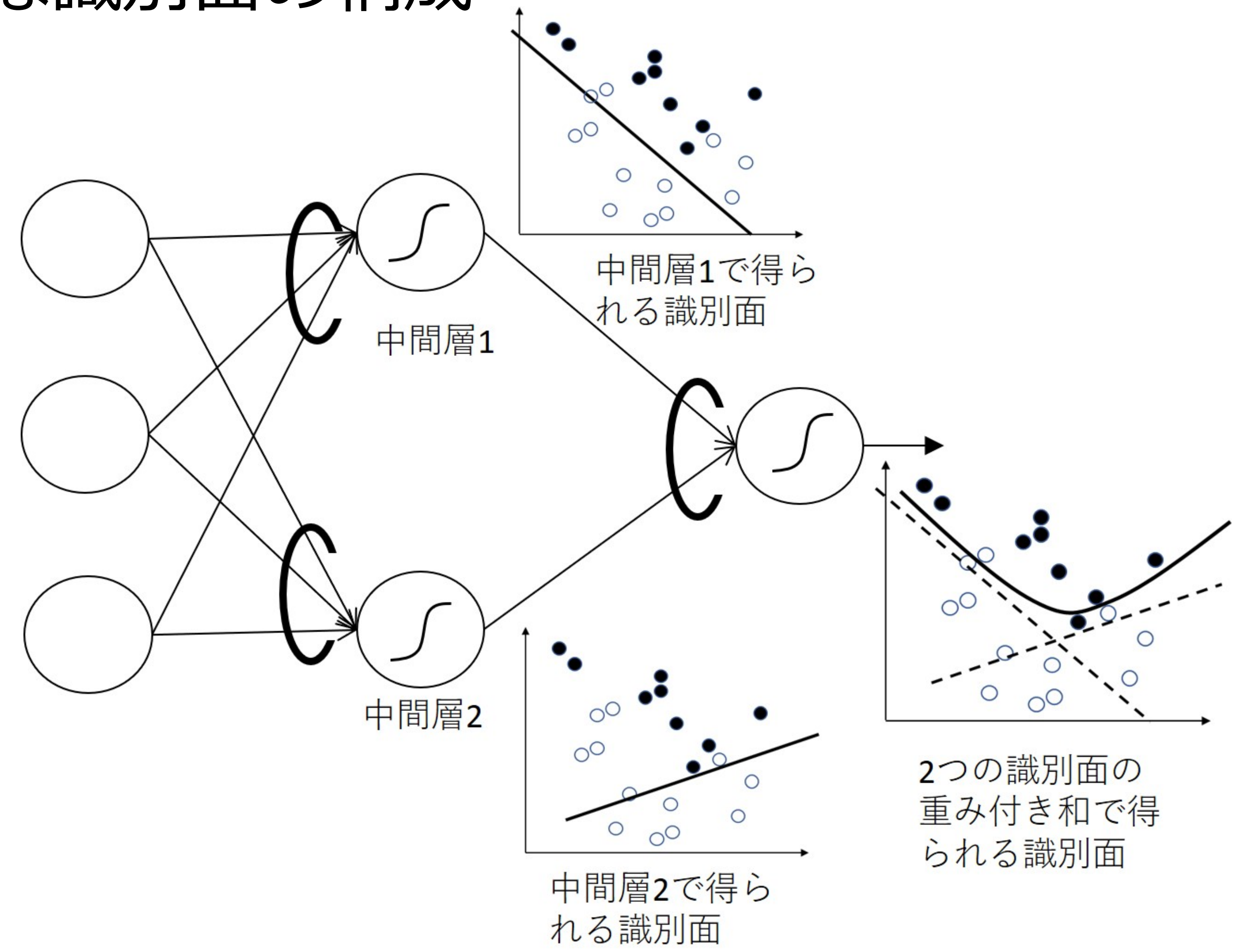
8.2 フィードフォワードネットワーク

- フィードフォワードネットワークのユニット
  - 中間層の活性化関数：シグモイド関数
  - 出力層の活性化関数：シグモイド関数または softmax 関数

$$f(h_i) = \frac{\exp(h_i)}{\sum_{j=1}^{c} \exp(h_j)}$$

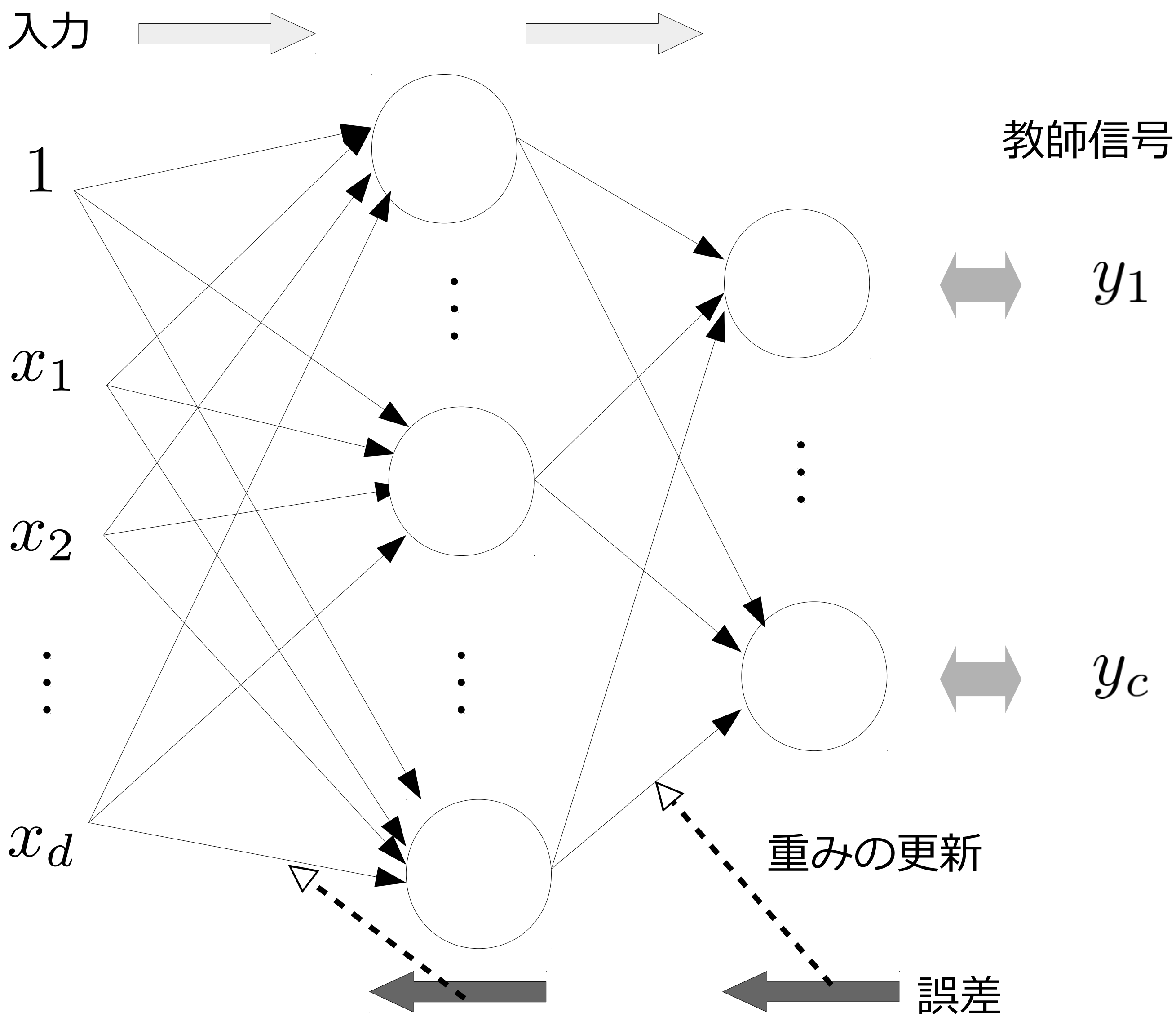
8.2 フィードフォワードネットワーク

- 複雑な識別面の構成



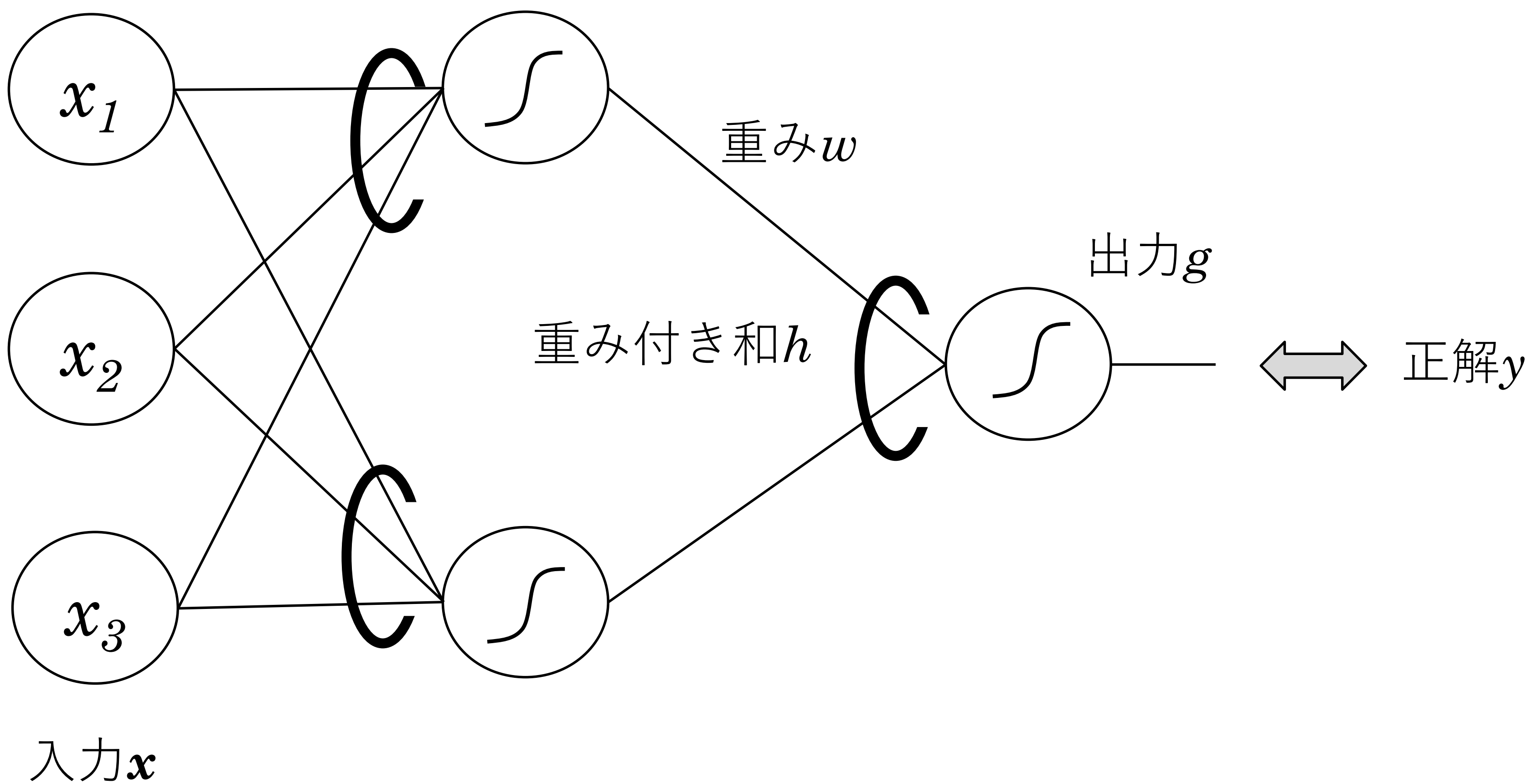
8.2.2 誤差逆伝播法による学習

- 誤差逆伝播法



8.2.2 誤差逆伝播法による学習

- 以下の構造を持つニューラルネットワークを仮定



8.2.2 誤差逆伝播法による学習

- データ集合

$$D : (\boldsymbol{x}_i, y_i) \quad (i = 1, \ldots, N)$$

- 誤差関数（二乗誤差）

$$E(\boldsymbol{w}) \equiv \frac{1}{2} \sum_{\boldsymbol{x}_i \in D} (g_i - y_i)^2$$

- 最急勾配法による重みの更新

$$w \leftarrow w - \eta \underbrace{\frac{\partial E(\boldsymbol{w})}{\partial w}}_{}$$

この項を求める

8.2.2 誤差逆伝播法による学習

- 合成微分の公式の適用

$$\frac{\partial E(\boldsymbol{w})}{\partial w} = \frac{\partial E(\boldsymbol{w})}{\partial h}\frac{\partial h}{\partial w}$$

重み w で接続している
ユニットの出力
中間層なら

$$h = \sum_i w_i x_i$$

から x が得られる

$$\epsilon = \frac{\partial E(\boldsymbol{w})}{\partial h} = \frac{\partial E(\boldsymbol{w})}{\partial g}\frac{\partial g}{\partial h}$$

活性化関数の微分
$g(1-g)$

出力層の場合 $\quad \frac{\partial E(\boldsymbol{w})}{\partial g} = g - y$

j は出力層のユニット

中間層の場合 $\quad \frac{\partial E(\boldsymbol{w})}{\partial g} = \sum_j \frac{\partial E(\boldsymbol{w})}{\partial h_j}\frac{\partial h_j}{\partial g} = \sum_j \epsilon_j w_j$

8.2.2 誤差逆伝播法による学習

- 誤差逆伝播法

1. リンクの重みを小さな初期値に設定
2. 個々の学習データ $(\boldsymbol{x}_i, y_i)$に対して以下繰り返し
   - 入力 $\boldsymbol{x}_i$ に対するネットワークの出力 $\boldsymbol{g}_i$ を計算

   a)出力層の k 番目のユニットに対してエラー量 ε 計算

   $$\epsilon_k \leftarrow g_k(1-g_k)(g_k-y_k)$$

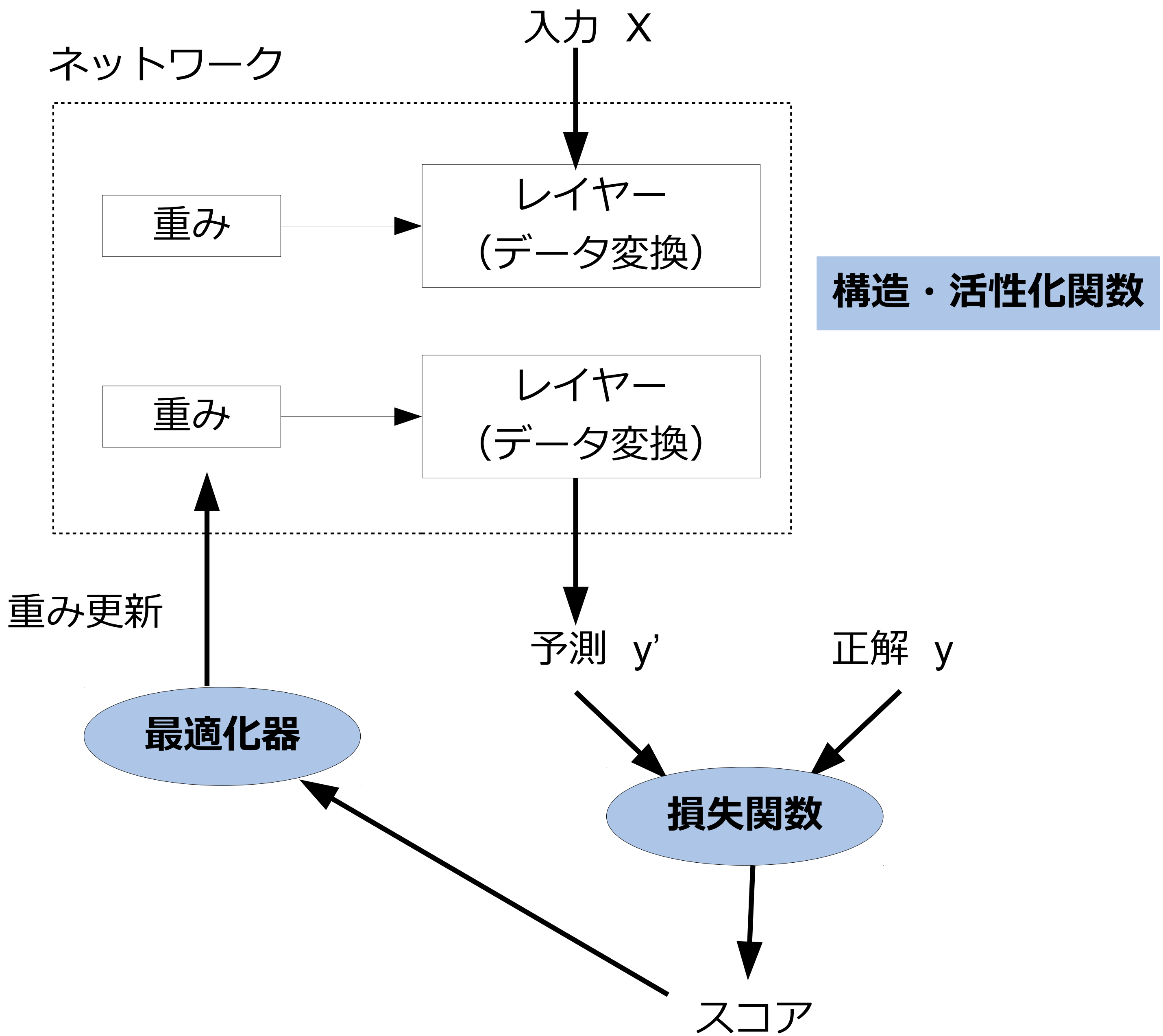
   b)中間層の h 番目のユニットに対してエラー量 ε 計算

   $$\epsilon_h \leftarrow g_h(1-g_h)\sum_{k \in outputs} w_{kh}\epsilon_k$$

   c)重みの更新

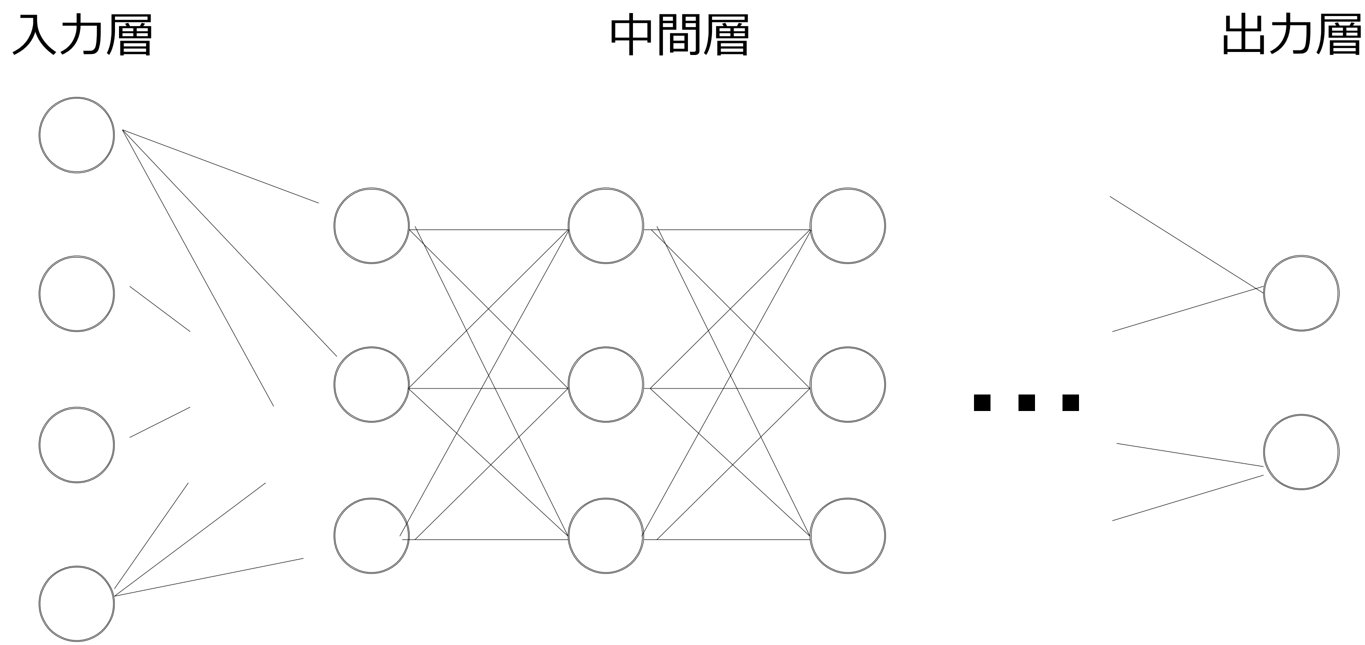
   $$w'_{ji} \leftarrow w_{ji} + \eta\epsilon_j x_{ji}$$

ニューラルネットワークによる学習の枠組み



8.3 ニューラルネットワークの深層化

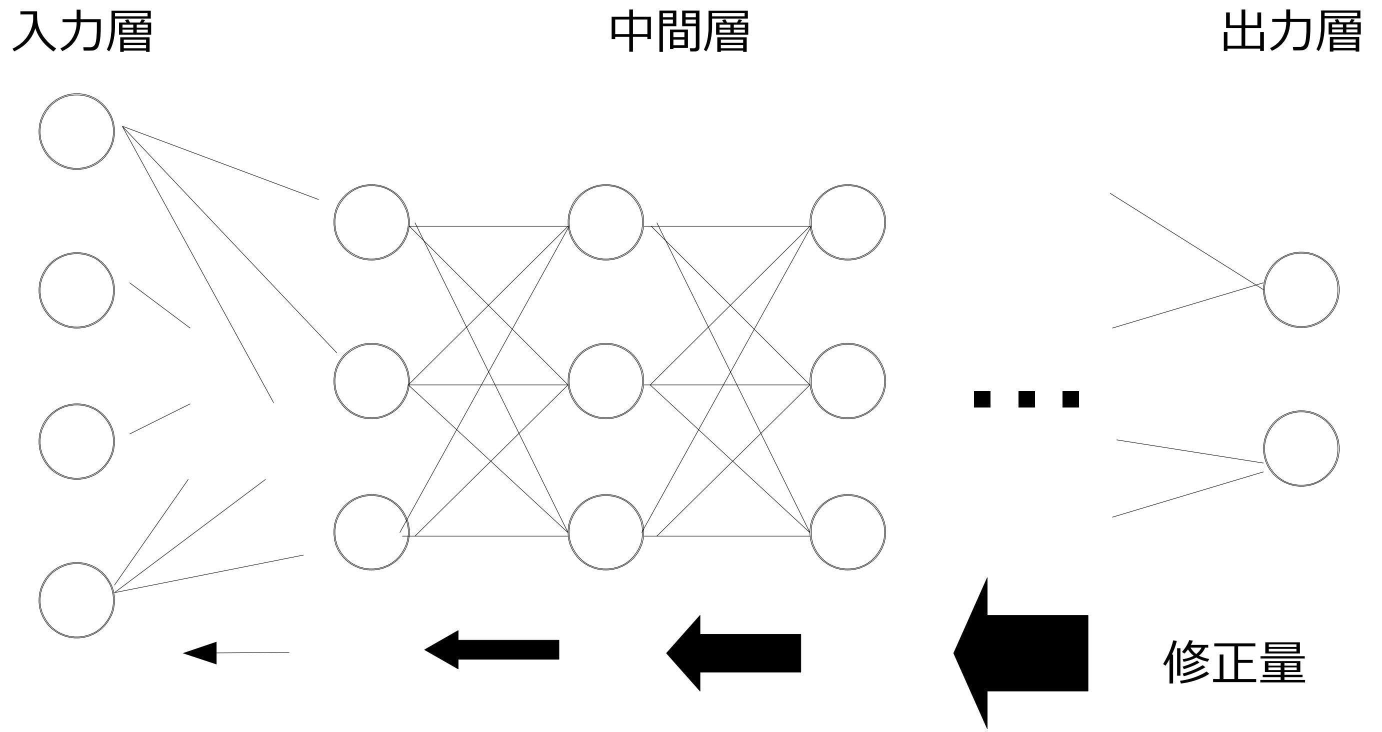
- ニューラルネットワークの構造の決定
  - 中間層の数：その層で実現される非線形変換の複雑さ
  - 階層数：低次の特徴表現から高次の特徴表現への段階的な変換を実現



8.3.1 勾配消失問題

- 多階層における誤差逆伝播法の問題点
  - 修正量が消失／発散する

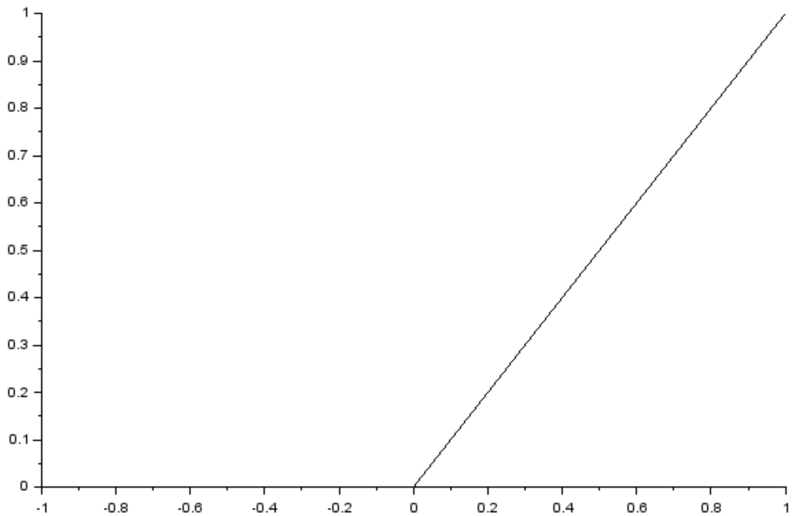
順方向：非線形　　逆方向：線形



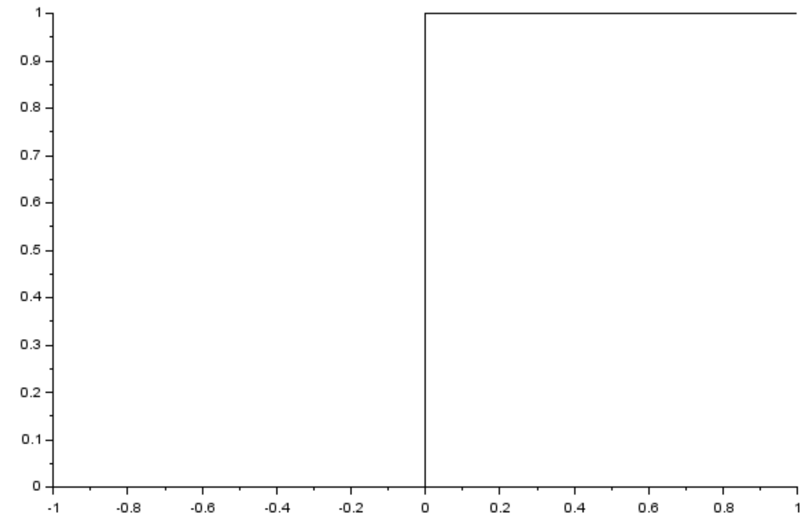
8.3.2 様々な活性化関数

- 活性化関数を rectified linear 関数に → RELU

$$f(x) = \max(0, x)$$



(a) rectified linear 関数



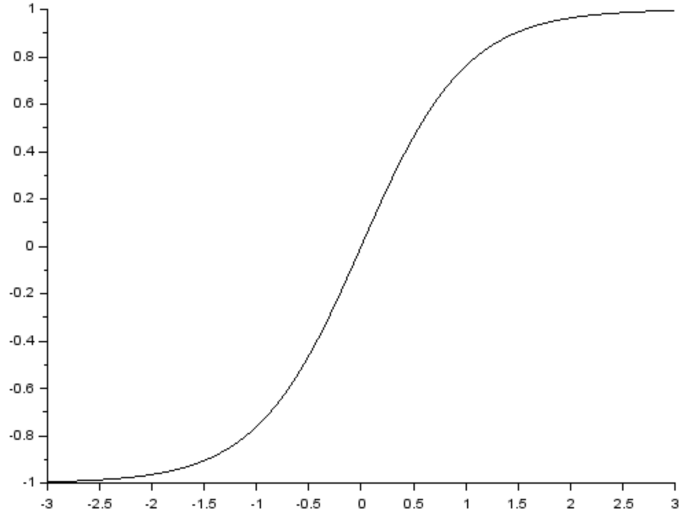
(b) (a) の導関数

- RELU の利点
  - 誤差消失が起こりにくい
  - 0 を出力するユニットが多くなる

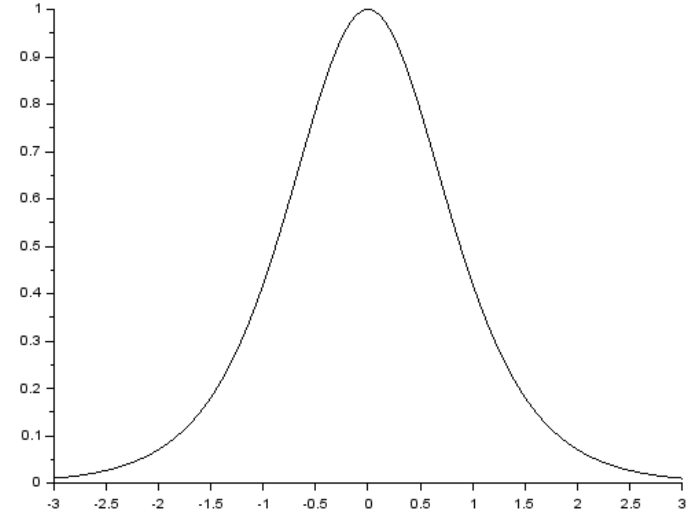
8.3.2 様々な活性化関数

- 活性化関数を双曲線正接 tanh 関数に

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{-x} + e^{x}}$$



(a) tanh　関数



(b) (a) の導関数

- tanh の利点
  - 誤差消失が起こりにくい

    cf) sigmoid は微分係数の最大値が 0.25

損失関数

- 回帰問題
  - 二乗誤差
  - 外れ値の影響を小さくする場合は Huber 損失
    - 一定の範囲内は二乗誤差、範囲外は線形損失
- 識別問題
  - クロスエントロピー

$$E(\boldsymbol{w}) \equiv - \sum_{\boldsymbol{x}_i \in D} y_i \log(g_i)$$

理論的には確率分布 y と確率分布 g の近さ

最適化器

- 最急勾配法
  - モーメンタム（慣性）の導入
    - 更新の方向に勢いを付けることで収束を早め、振動を抑制する

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} + \eta \frac{\partial E}{\partial \boldsymbol{w}}$$

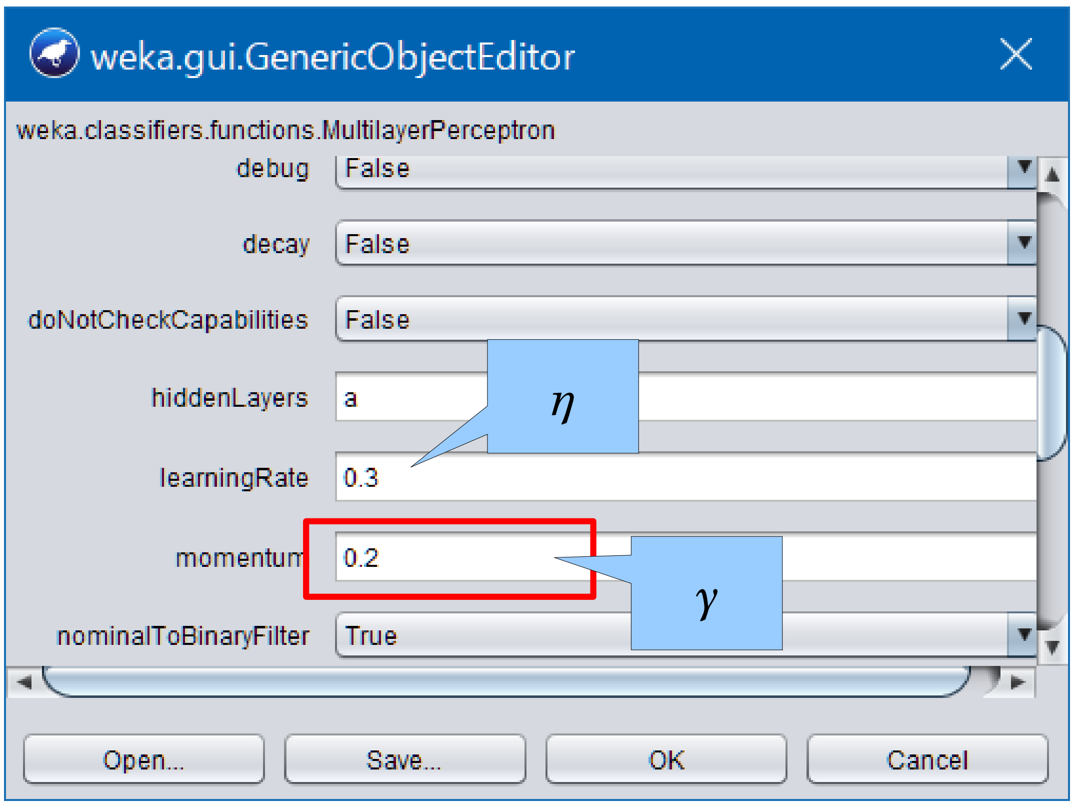
$$\boldsymbol{w}' = \boldsymbol{w} - \boldsymbol{v}_t$$

最適化器

- 準ニュートン法
  - 2 次微分（近似）を更新式に加える
- AdaGrad
  - 学習回数と勾配の 2 乗を用いた学習係数の自動調整
- RMSProp
  - 学習係数調整の改良：勾配の 2 乗の指数平滑移動平均を用いることで直近の変化量を反映
- Adam:Adaptive Moment Estimation
  - モーメントの拡張：分散に関するモーメントも用いる
  - まれに観測される特徴軸に対して大きく更新する効果

Weka でのニューラルネットワークの設定

- Weka での最適化器の調整
  - モーメンタム（慣性）$\boldsymbol{v}$



$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} + \eta \frac{\partial E}{\partial \boldsymbol{w}}$$

$$\boldsymbol{w}' = \boldsymbol{w} - \boldsymbol{v}_t$$

sklearn でのニューラルネットワークの設定

- sklearn の学習パラメータ (1/2)

```
MLPClassifier(
    activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
    beta_2=0.999, early_stopping=False, epsilon=1e-08,
    hidden_layer_sizes=(100,), learning_rate='constant',
    learning_rate_init=0.001, max_iter=200, momentum=0.9,
    nesterovs_momentum=True, power_t=0.5, random_state=None,
    shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
    verbose=False, warm_start=False)
```

- activation: 活性化関数
    - ‘identity’: 同一値関数 f(x) = x
    - ‘logistic’: シグモイド関数 f(x) = 1 / (1 + exp(-x))
    - ‘tanh’: 双曲線正接 f(x) = tanh(x)
    - ‘relu’: ランプ関数 f(x) = max(0, x)

sklearn でのニューラルネットワークの設定

- sklearn の学習パラメータ (2/2)
- solver: 最適化手法
  - ‘lbfgs’ : 準ニュートン法
  - ‘sgd’ : 確率的最急降下法
  - ‘adam’ : Adaptive Moment Estimation

データ数が多いときは adam 、
少ないときは lbfgs が
勧められている