8. ニューラルネットワーク

- 本章の説明手順

一部、第 9 章の内容が入ります

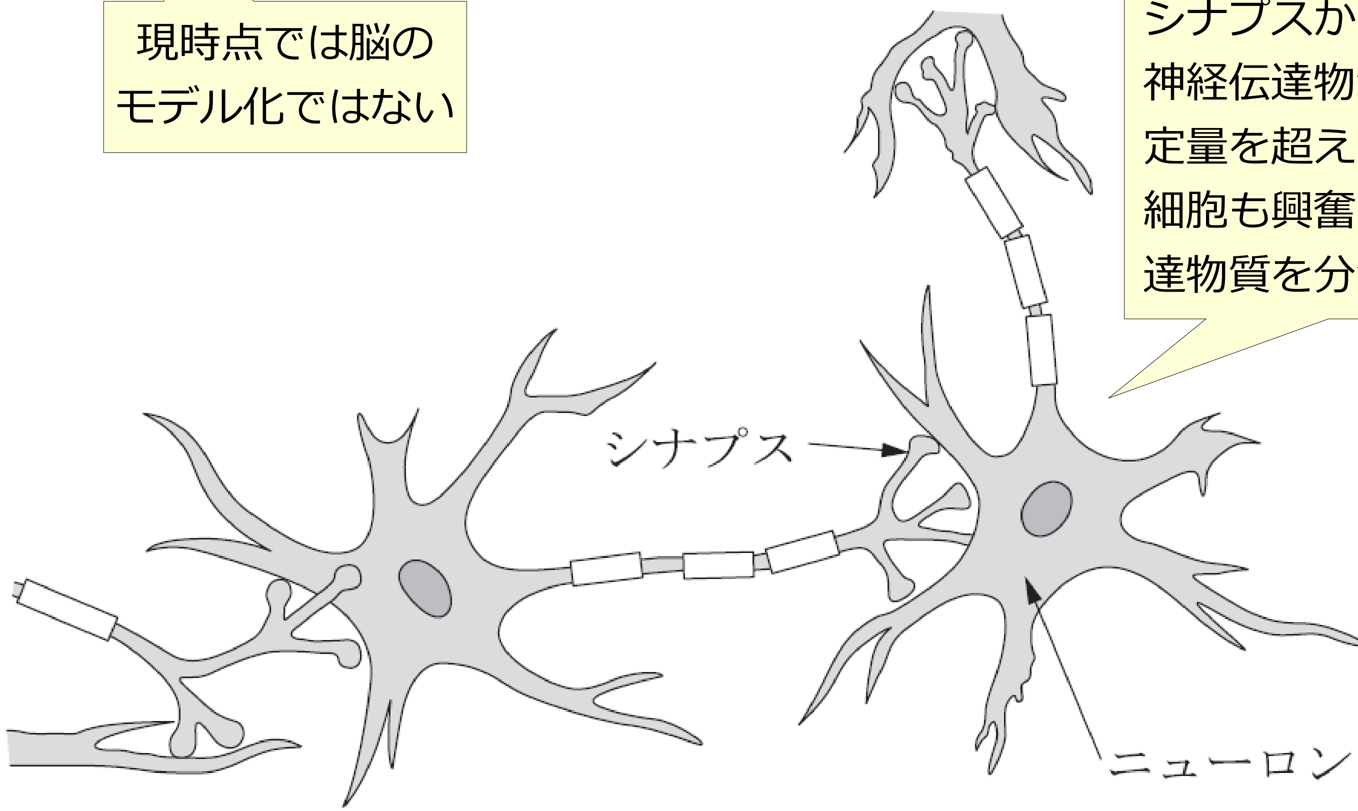
1. ニューラルネットワークによる非線形識別面の実現
2. ニューラルネットワークにおける学習＝誤差逆伝播法
3. ニューラルネットワークにおける学習の枠組み
   - keras プログラミング
4. 多階層ニューラルネットワークにおける学習

8. ニューラルネットワーク

- ニューラルネットワークとは
  - 神経細胞の情報伝達メカニズムを単純化したモデル

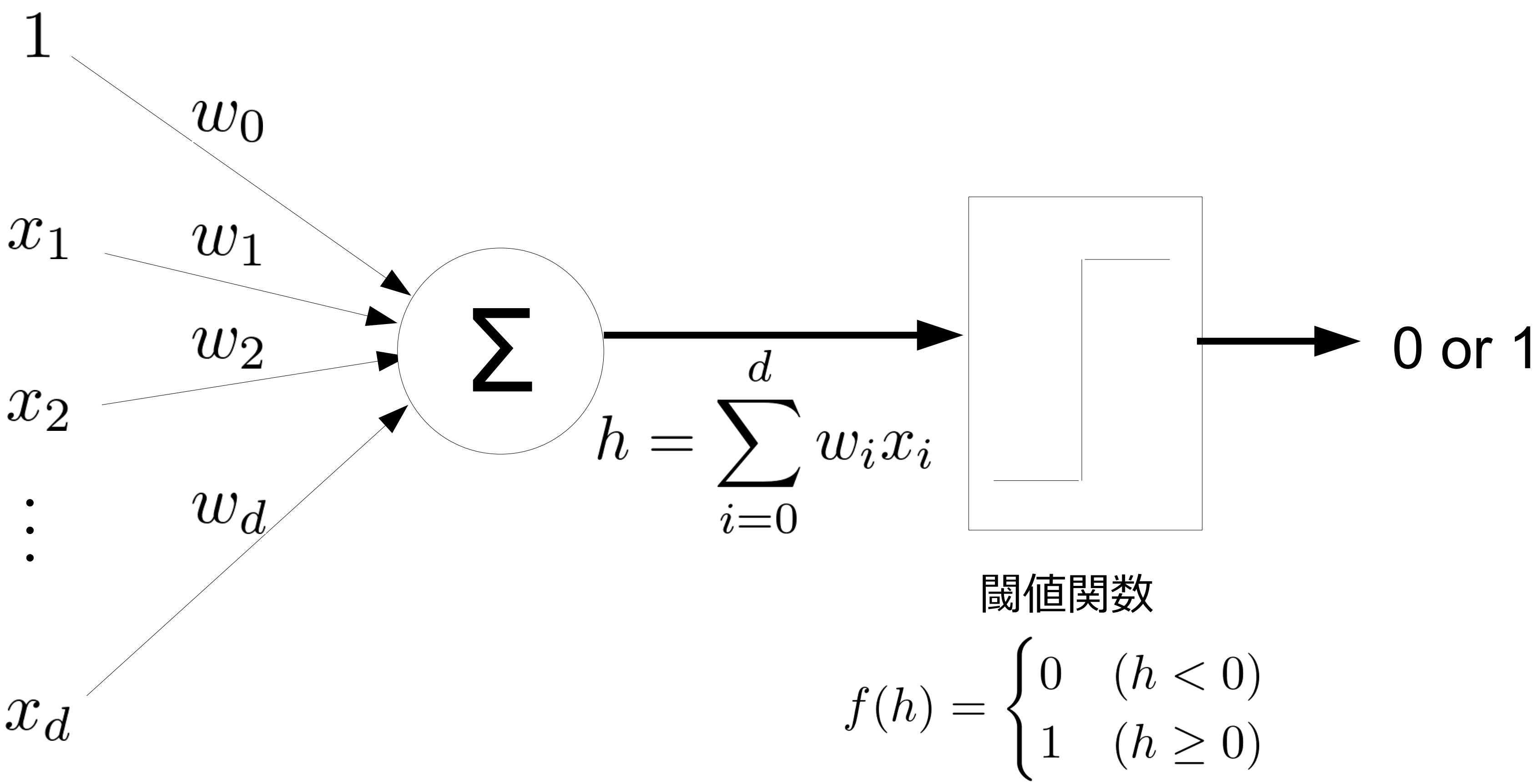
現時点では脳のモデル化ではない

シナプスから受け取る神経伝達物質の量が一定量を超えると、その細胞も興奮して神経伝達物質を分泌する



8.1 ニューラルネットワークの計算ユニット

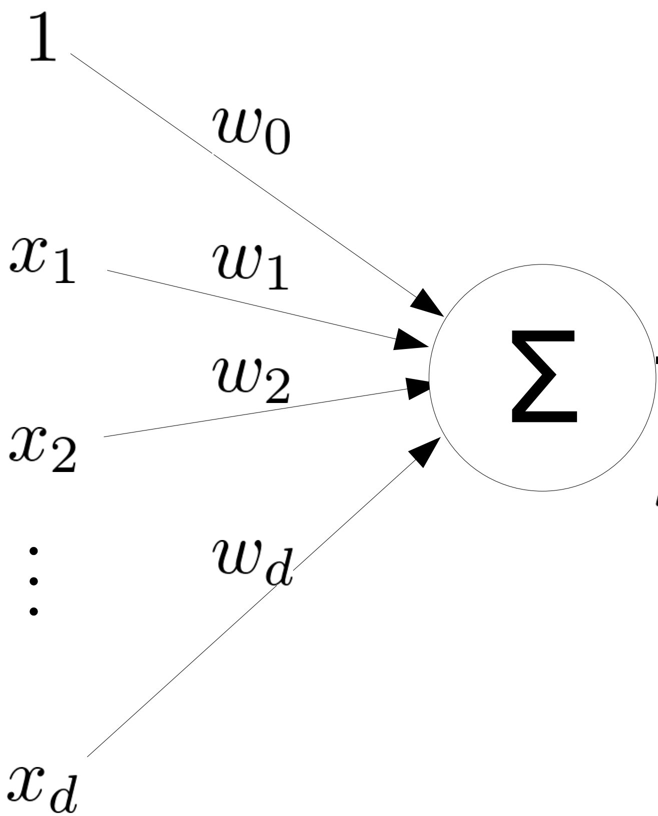
- 初期のニューロンモデル（ McCulloch&Pitts モデル）
  - 活性化関数に閾値関数を用いたパーセプトロン
  - $\boldsymbol{w}^T\boldsymbol{x} = 0$ という特徴空間上の線形識別面を表現



8.1 ニューラルネットワークの計算ユニット

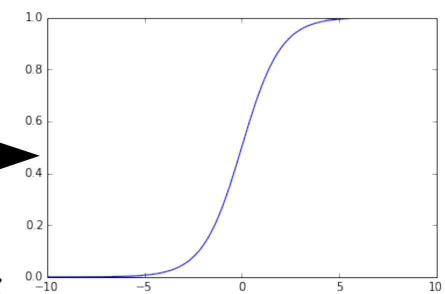
- 多階層で学習可能なユニットへ
  - 活性化関数に微分可能なシグモイド関数を用いる

ロジステック識別と同じ



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

活性化関数



シグモイド関数

g

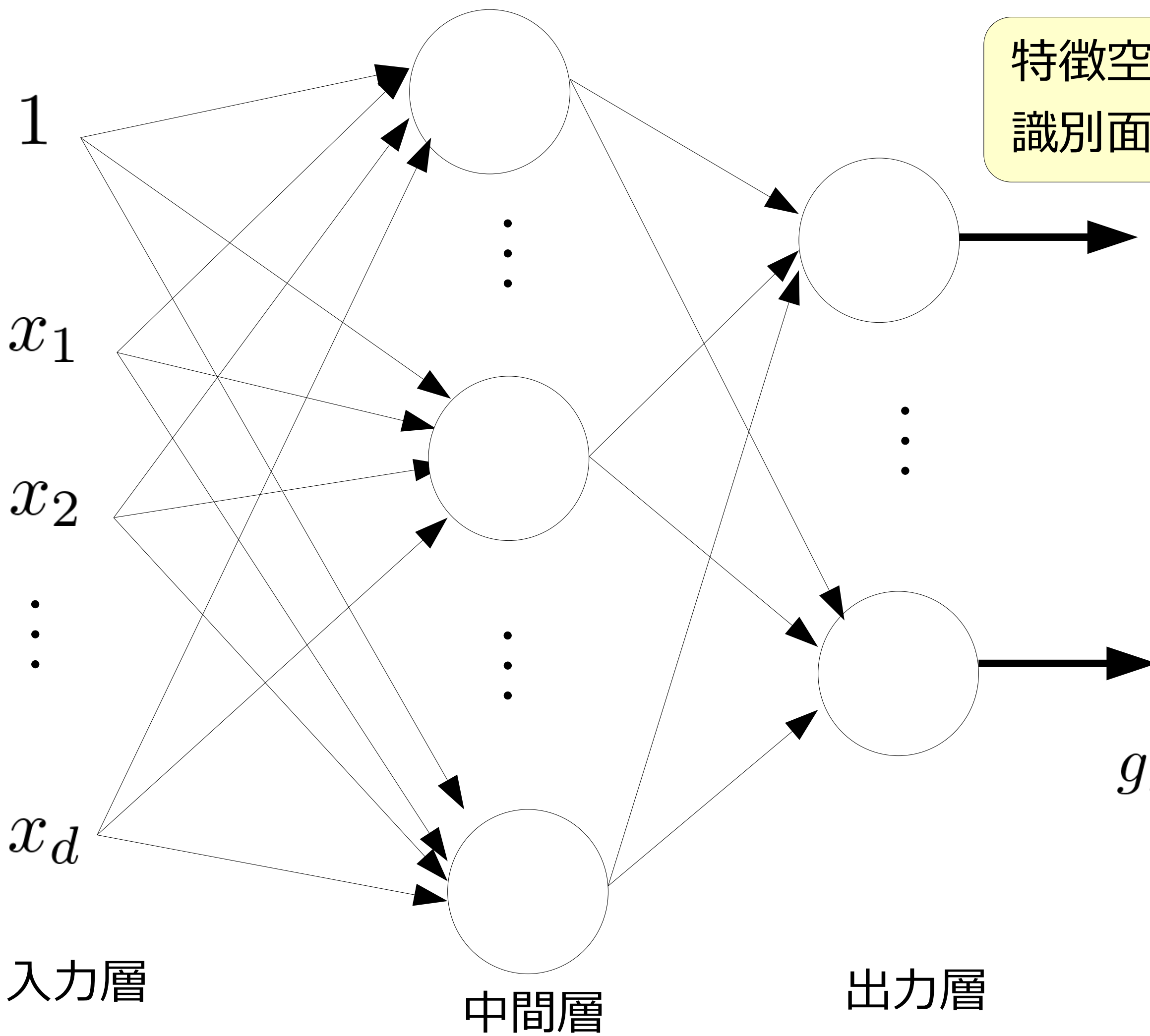
$0 \leq g \leq 1$

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

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

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型
  - 非線形関数ユニットの階層的組み合わせ



特徴空間上で複雑な非線形識別面が実現できる

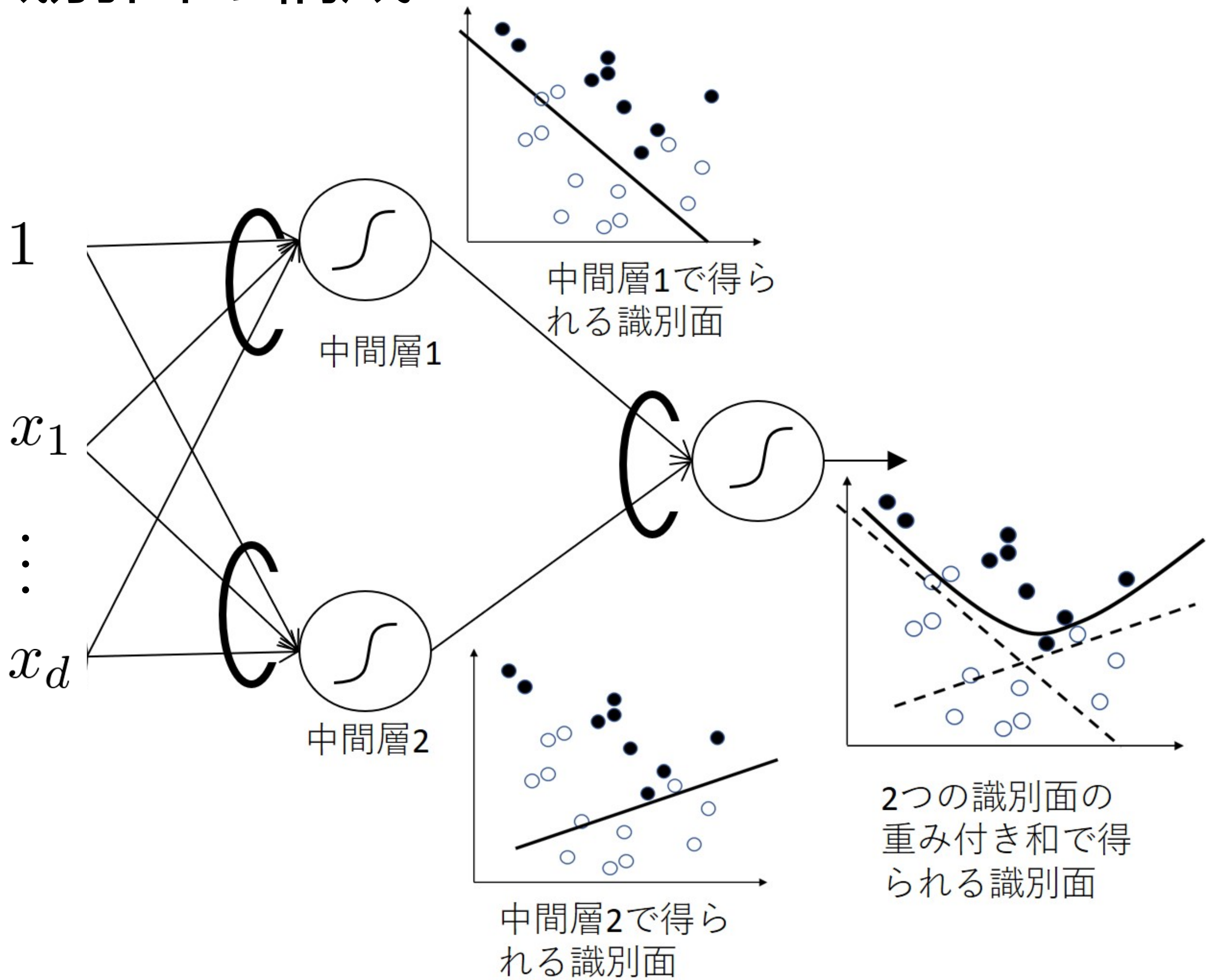
多クラス識別の出力層には活性化関数として以下のsoftmax 関数を用いる

$$g_k = \frac{\exp(h_k)}{\sum_{j=1}^{c} \exp(h_j)}$$

2 クラスの場合はシグモイド関数と同じ

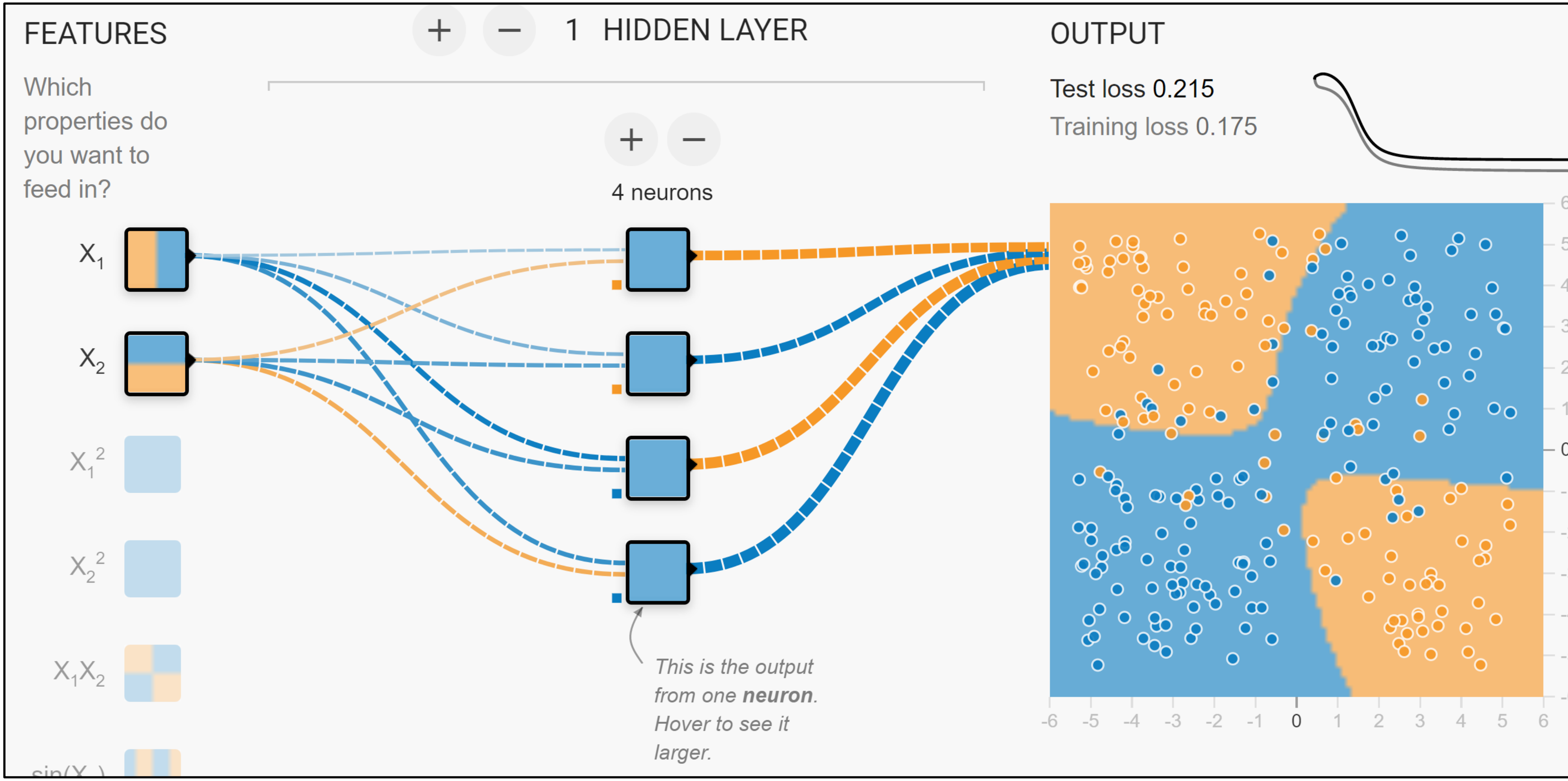
8.2 フィードフォワードネットワーク

- 複雑な識別面の構成



8.2 フィードフォワードネットワーク

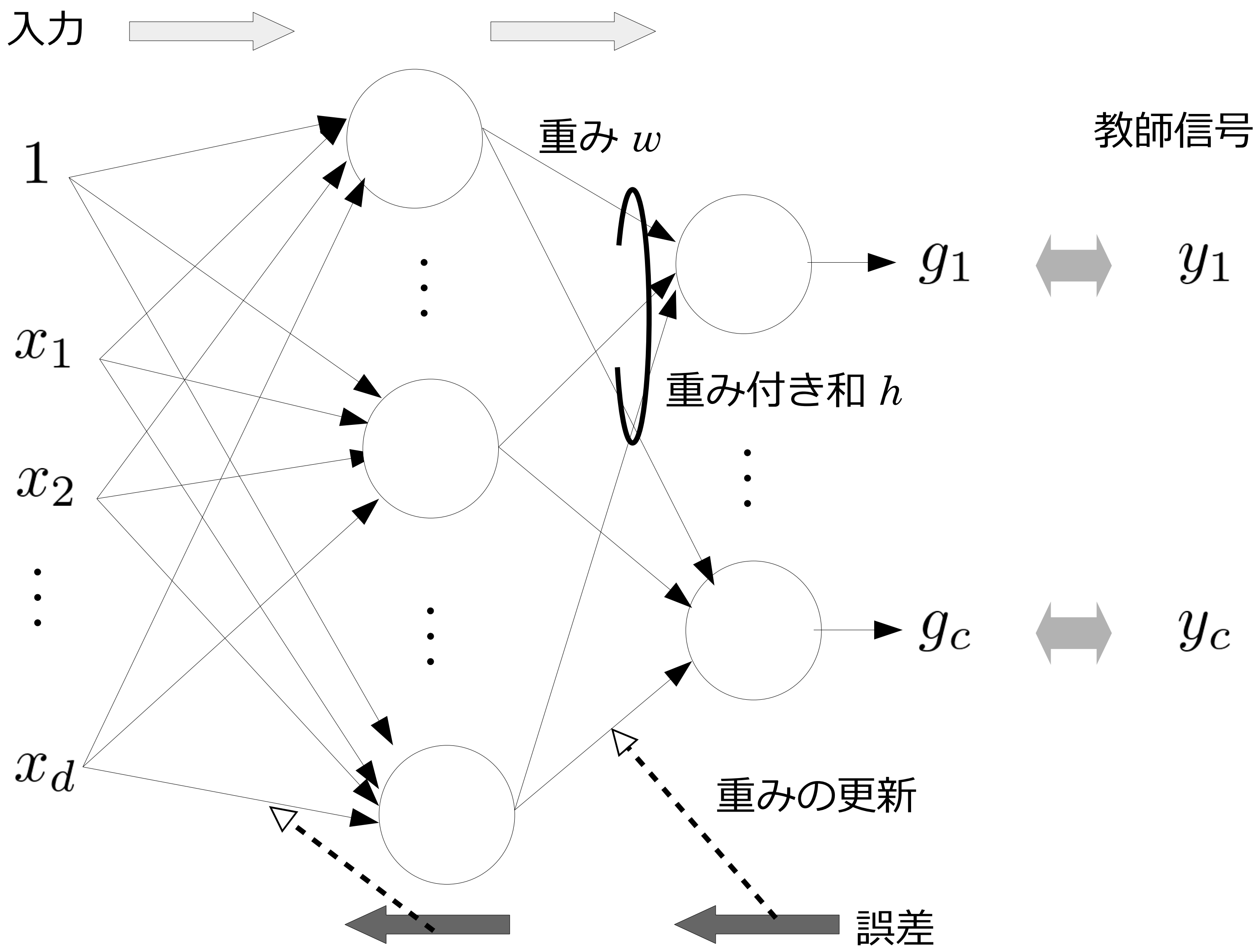
- 複雑な識別面の構成



http://playground.tensorflow.org/

8.2.2 誤差逆伝播法による学習

- フィードフォワード型ネットワークの学習



8.2.2 誤差逆伝播法による学習

- データ集合 D

$$\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}\ (i = 1, \ldots, N) \quad \boldsymbol{y}_i :$$ c 次元の one-hot ベクトル

- 特定のデータ $\boldsymbol{x}$ に対する二乗誤差

$$E(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{j=1}^{c}(g_j - y_j)^2$$

- 確率的最急勾配法による重み w の更新

$$w' \leftarrow w - \eta \underline{\frac{\partial E(\boldsymbol{w})}{\partial w}}$$

8.2.2 誤差逆伝播法による学習

- 合成微分の公式の適用

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

重み w で接続しているユニットの入力 g_{in}
中間層の場合は、

$$h = \sum_i w_i x_i$$

から x が得られる

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

g は w で結合しているユニットの出力

活性化関数の微分
$g(1-g)$

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

j は出力層のユニット

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

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 $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ 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 j 番目のユニットに対してエラー量 ε_j 計算

$$\epsilon_j \leftarrow g_j(1 - g_j)(g_j - y_j)$$

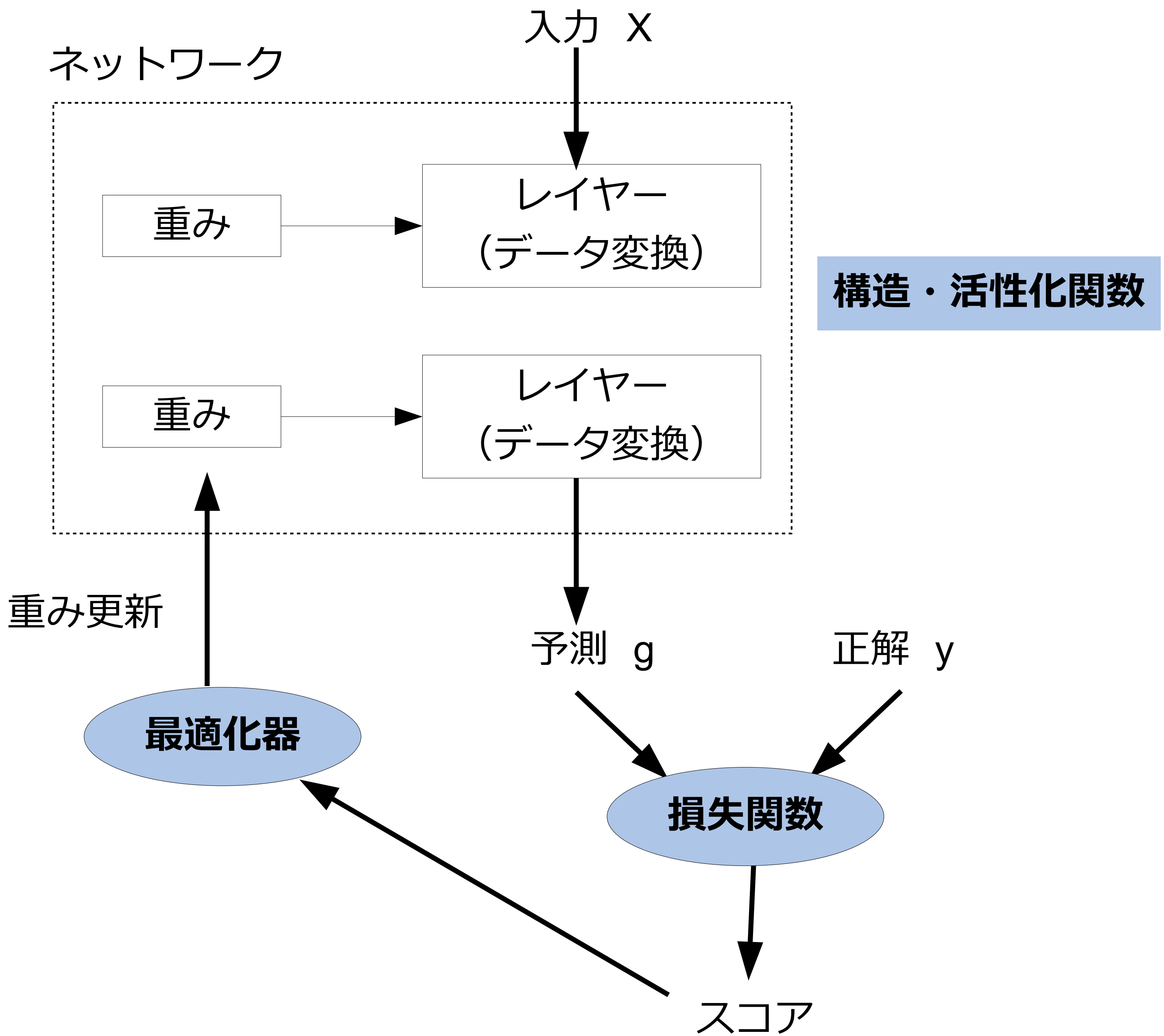
   b)中間層の k 番目のユニットに対してエラー量 ε_k 計算

$$\epsilon_k \leftarrow g_k(1 - g_k)\sum_{j=1}^{c} w_{kj}\epsilon_j$$

   c)重みの更新

$$w' \leftarrow w - \eta\epsilon g_{in}$$

ニューラルネットワークによる学習の枠組み



損失関数

- 回帰問題
  - 二乗誤差
  - 外れ値の影響を小さくしたい場合は 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

データ数が多いときは adam 、
少ないときは L-BFGS が勧められている

- 準ニュートン法 (L-BFGS)
  - 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

keras のコーディング

- NN の構造と活性化関数の指定

```
model = Sequential([
    Dense(512, input_shape=(784,)),
    Activation('sigmoid'),
    Dense(10),
    Activation('softmax')
    ])
```

最初の層のみ、ユニット数と入力の次元数の指定が必要

2 階層目以上はユニット数の指定のみ

- Dense: 密結合層
  - 隣接する層間のすべてのユニット間で結合をもつ
- Activation: 活性化関数
  - ‘softmax’: ソフトマックス関数
  - ‘sigmoid’: シグモイド関数 $f(x) = 1 / (1 + \exp(-x))$
  - ‘tanh’: 双曲線正接 $f(x) = \tanh(x)$
  - ‘relu’: rectified linear 関数 $f(x) = \max(0, x)$

keras のコーディング

- model のコンパイル

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['acc'])
```

  - 損失関数、最適化器、評価指標（複数可）を指定
  - optimizer: 最適化手法
    - ‘sgd’ : 確率的最急降下法
    - ‘adam’ : Adaptive Moment Estimation
  - metrics: 評価指標
    - ‘acc’ : 正解率
    - ‘mae’ : 平均二乗誤差

keras のコーディング

- 学習

  model.fit(X_train, y_train, batch_size=200, epochs=3)

  - ミニバッチのサイズと繰り返し数を指定
  - 繰り返し毎に損失関数の値と、 metrics で指定した値が表示される

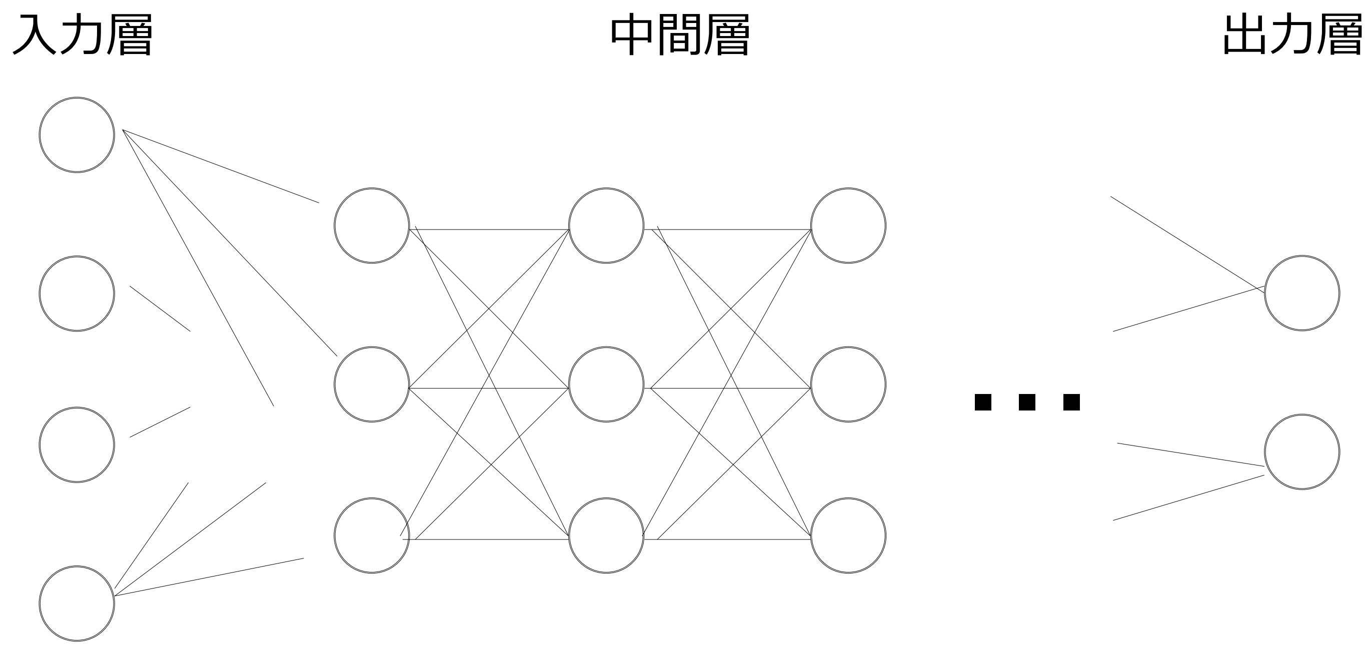
- 評価

  score = model.evaluate(X_test, y_test)

  - score[0] は損失関数の値
  - score[1] 以降は metrics で指定したもの

8.3 ニューラルネットワークの深層化

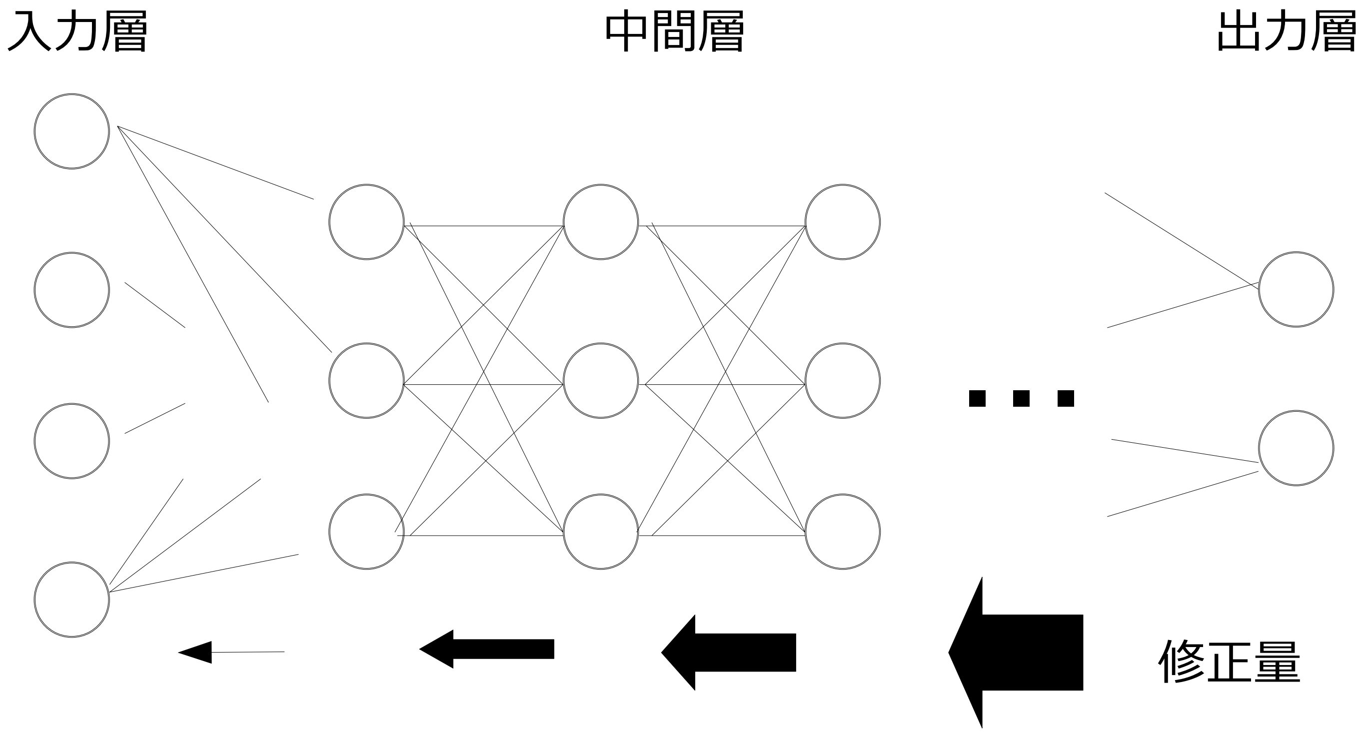
- ニューラルネットワークの構造の決定
  - 中間層の数：その層で実現される非線形変換の複雑さ
  - 階層数：低次の特徴表現から高次の特徴表現への段階的な変換を実現



8.3.1 勾配消失問題

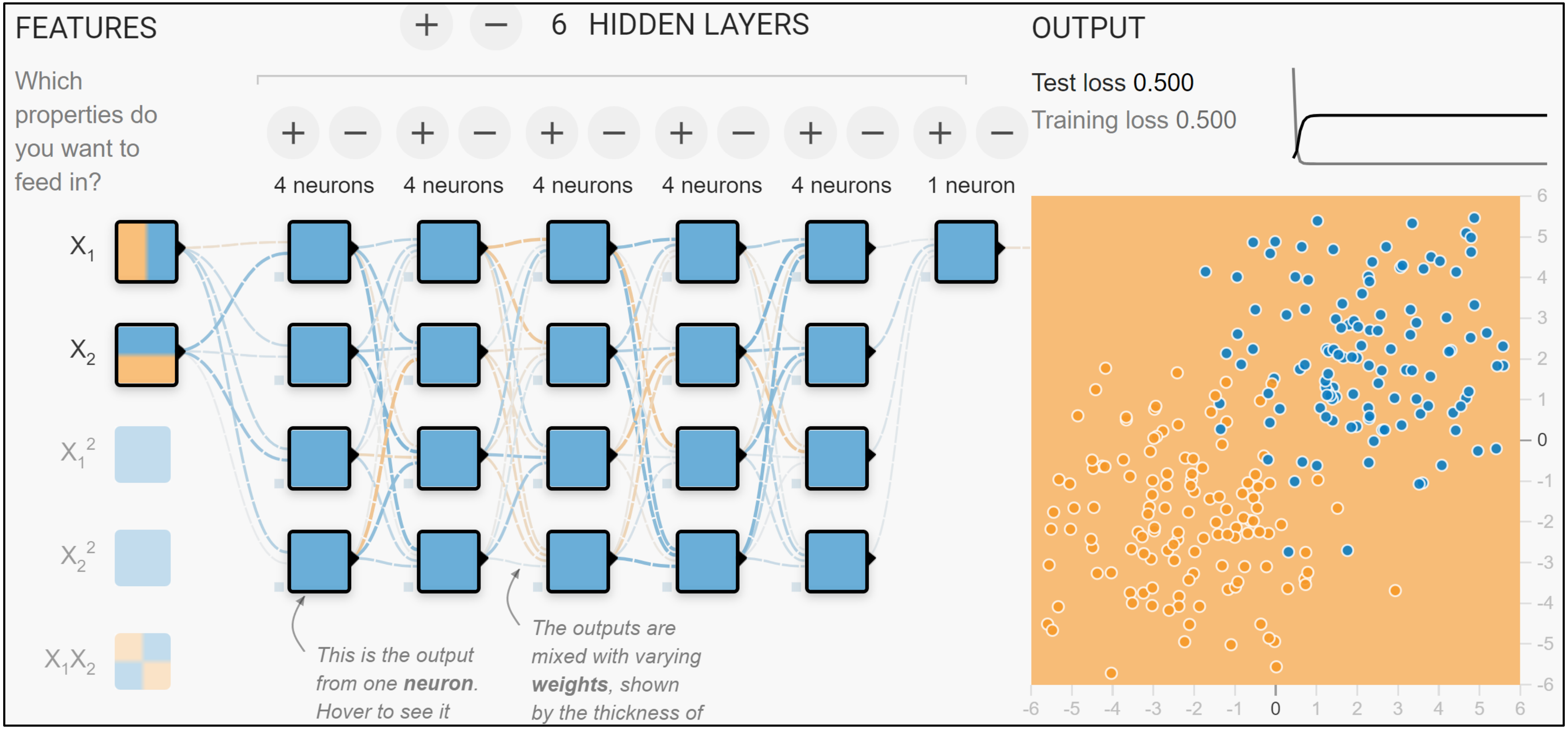
- 多階層における誤差逆伝播法の問題点
  - 修正量が消失／発散する

順方向：非線形　　逆方向：線形



8.3.1 勾配消失問題

- 多階層での学習

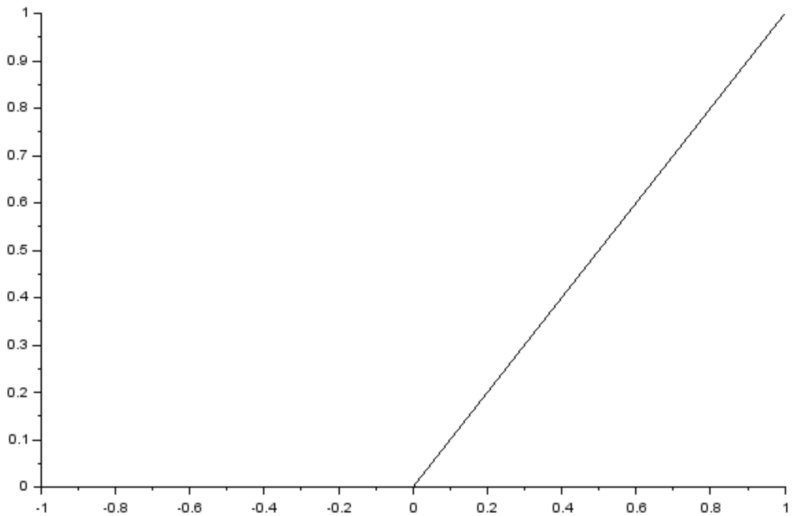


http://playground.tensorflow.org/

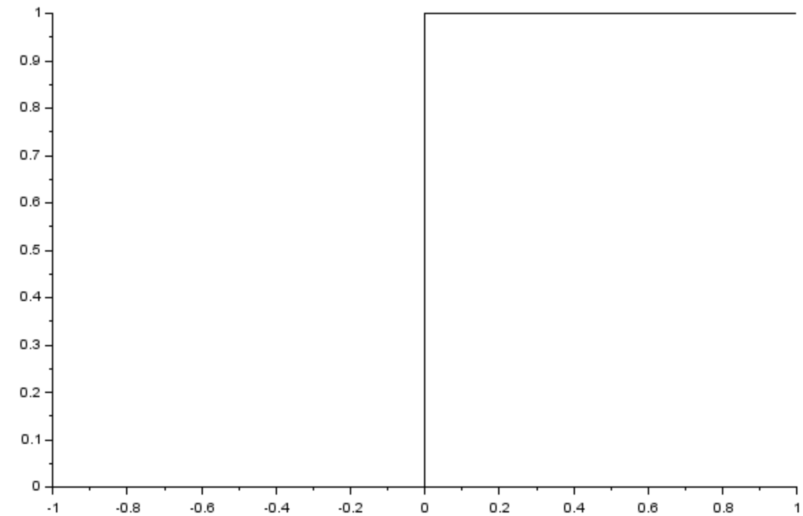
8.3.2 様々な活性化関数

- 活性化関数を rectified linear 関数に → ReLU

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



(a) rectified linear 関数



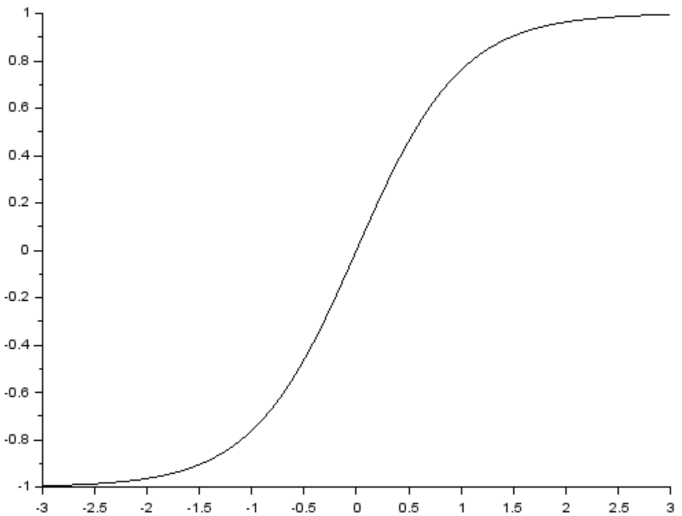
(b) (a) の導関数

- 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

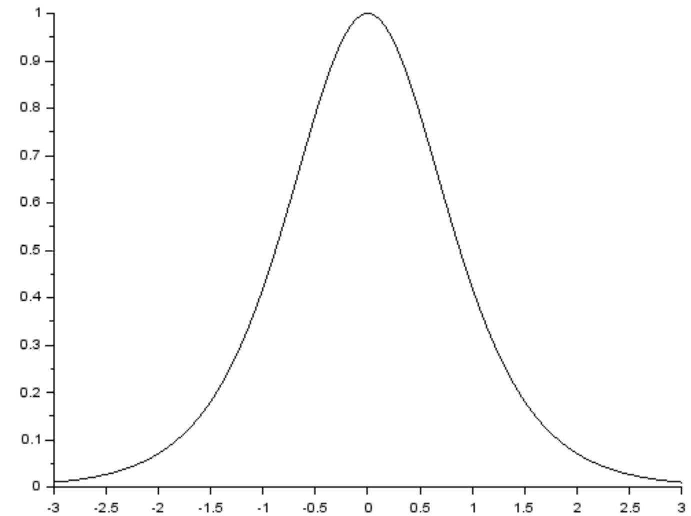
8.3.2 様々な活性化関数

- 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