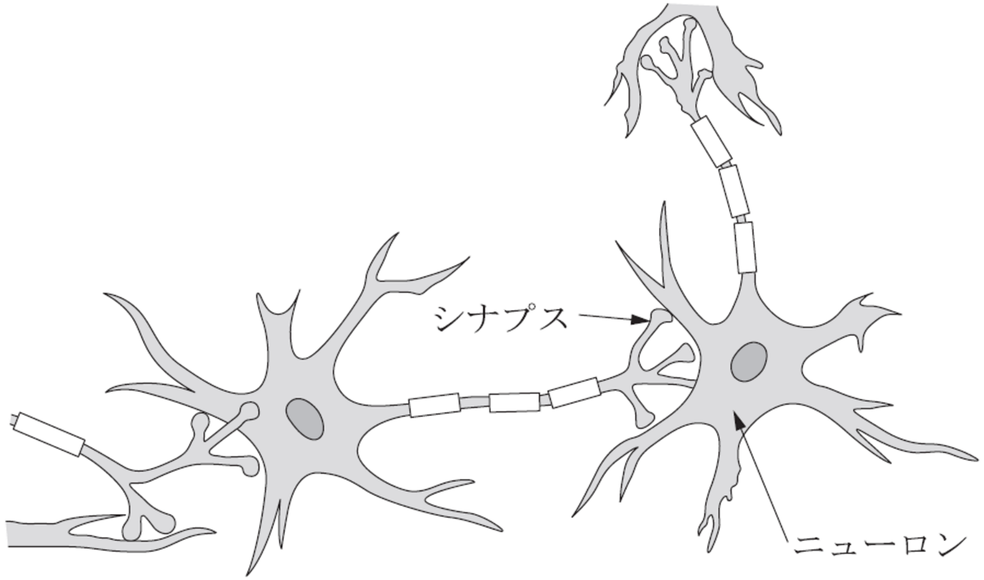
7. 限界は破れるか（2） ニューラルネットワーク

- 誤差評価に基づく学習
  - 誤差最小の線形識別面を学習できる
  - 非線形識別面の学習は可能だが、どのような非線形関数にするかを事前に設計する必要がある
- 誤差最小・任意形の識別面を学習することはできないか
  → ニューラルネットワーク

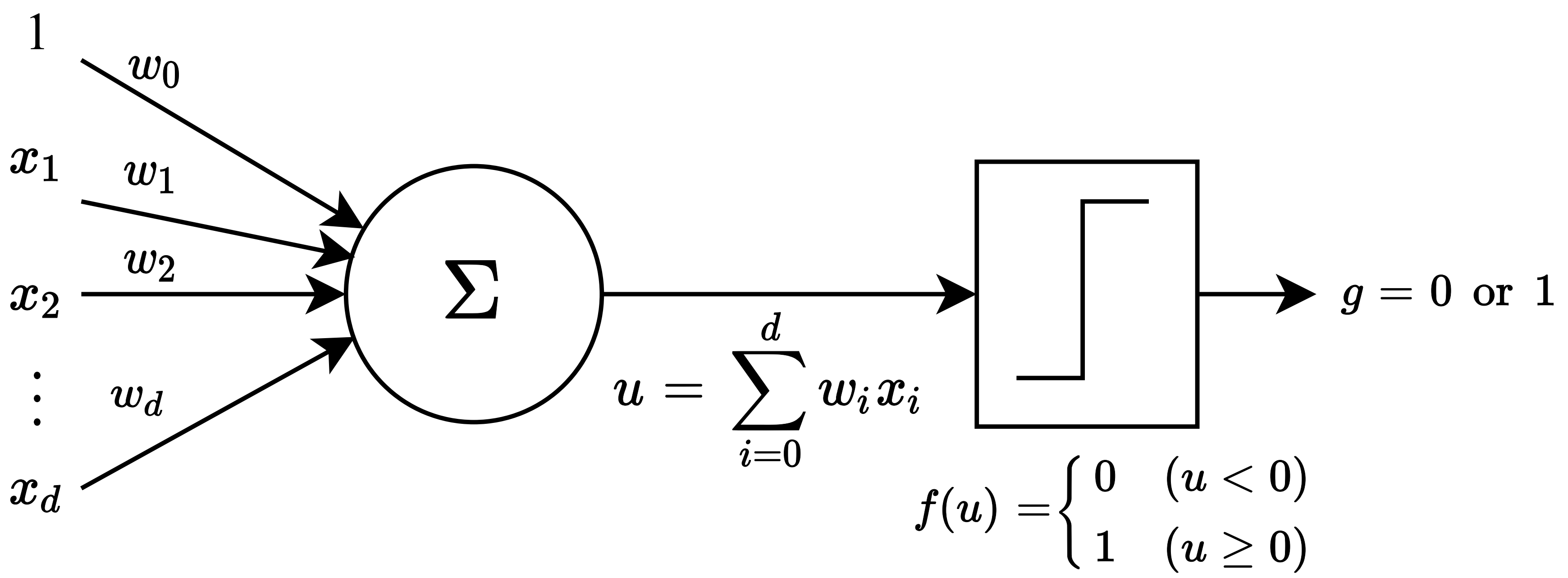
7.1 ニューラルネットワークの構成

- 神経細胞の計算メカニズム
  - ニューロンがシナプス結合によって複雑に結合
  - 入力された電気信号の重み付き和の値によって、各ニューロンの活性／非活性が決まる



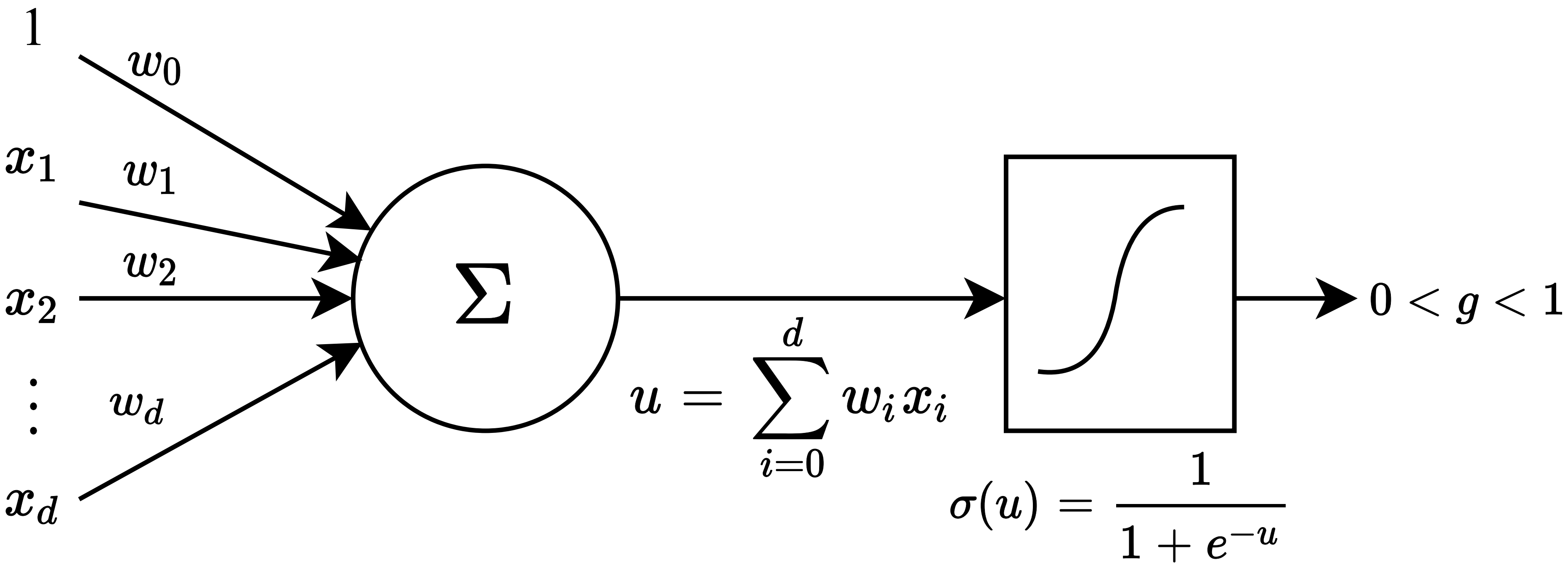
7.1 ニューラルネットワークの構成

- 閾値論理ユニットによるニューロンのモデル化
  - $\boldsymbol{w}^T\boldsymbol{x} = 0$ という特徴空間上の識別面を表現（$\boldsymbol{w}$, $\boldsymbol{x}$ は $d+1$ 次元）
    → パーセプトロン（データが線形分離可能なときのみ学習可能）



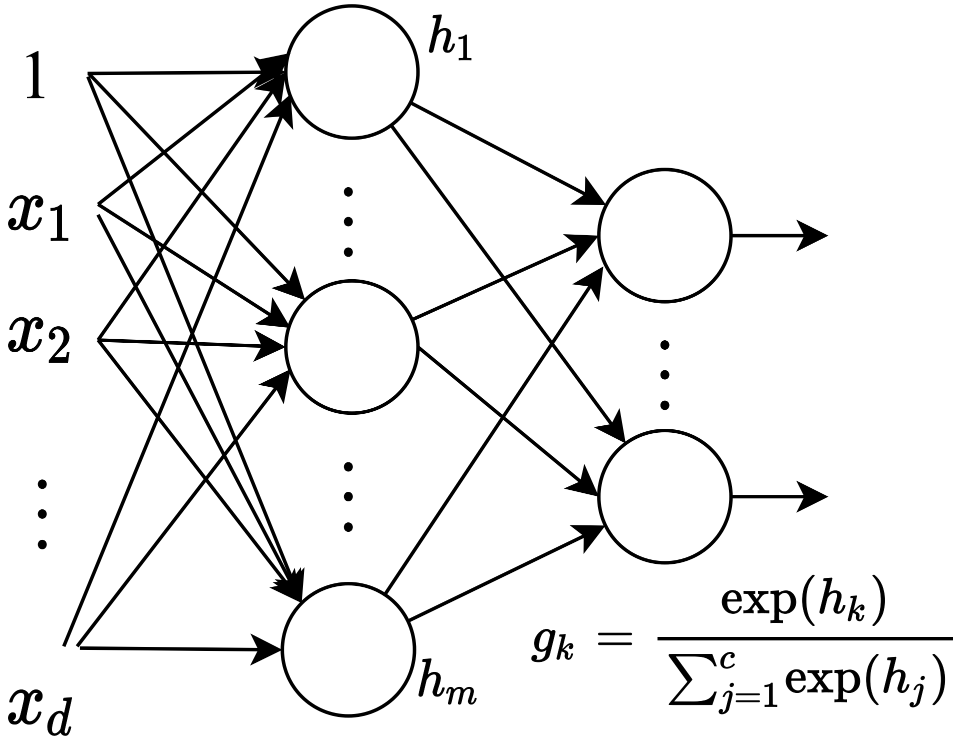
7.1 ニューラルネットワークの構成

- 活性化関数をシグモイド関数に差し替え
    - 入力の重み付き和を大小関係は変えずに0〜1の値に変換
    - データが線形分離不可能でも誤差最小の線形識別面が学習可能



7.1 ニューラルネットワークの構成

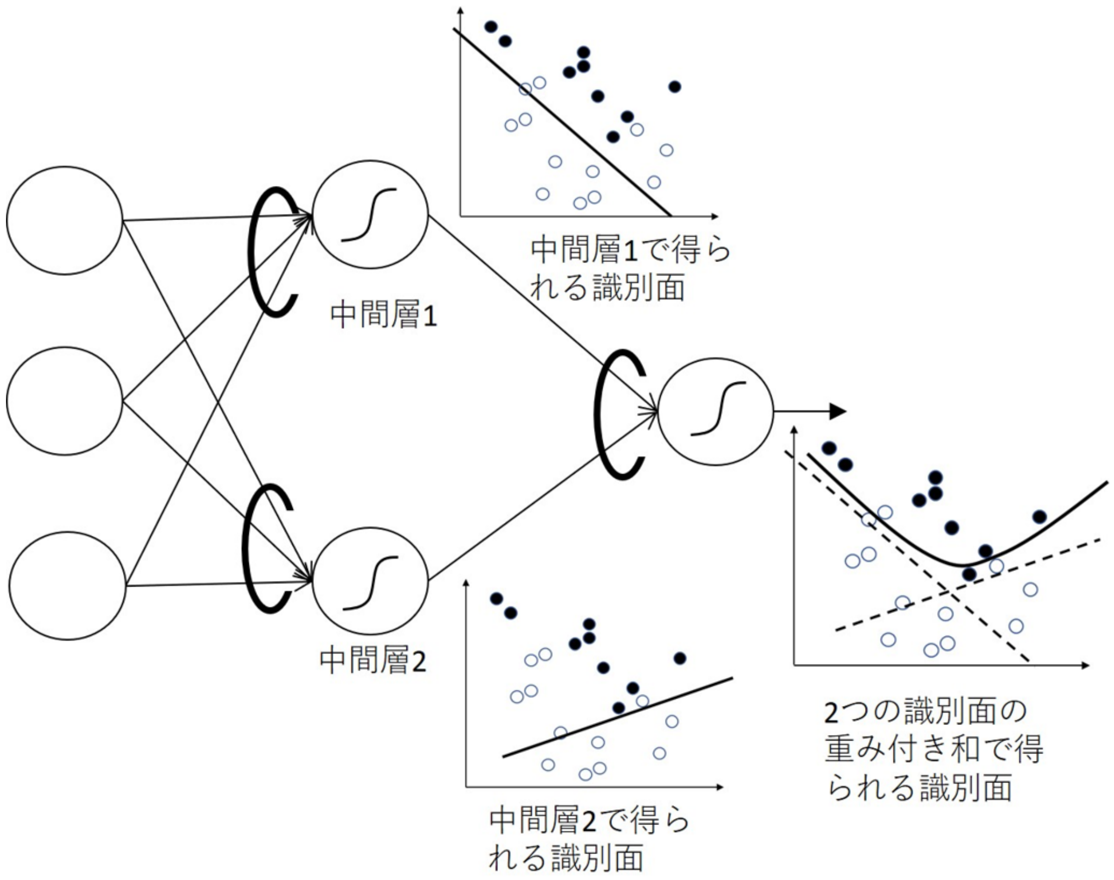
- フィードフォワード型ニューラルネットワーク
  - ユニットを階層状に結合することで非線形識別面を表現
  - 多クラス識別の出力層には活性化関数としてsoftmax関数を用いる



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

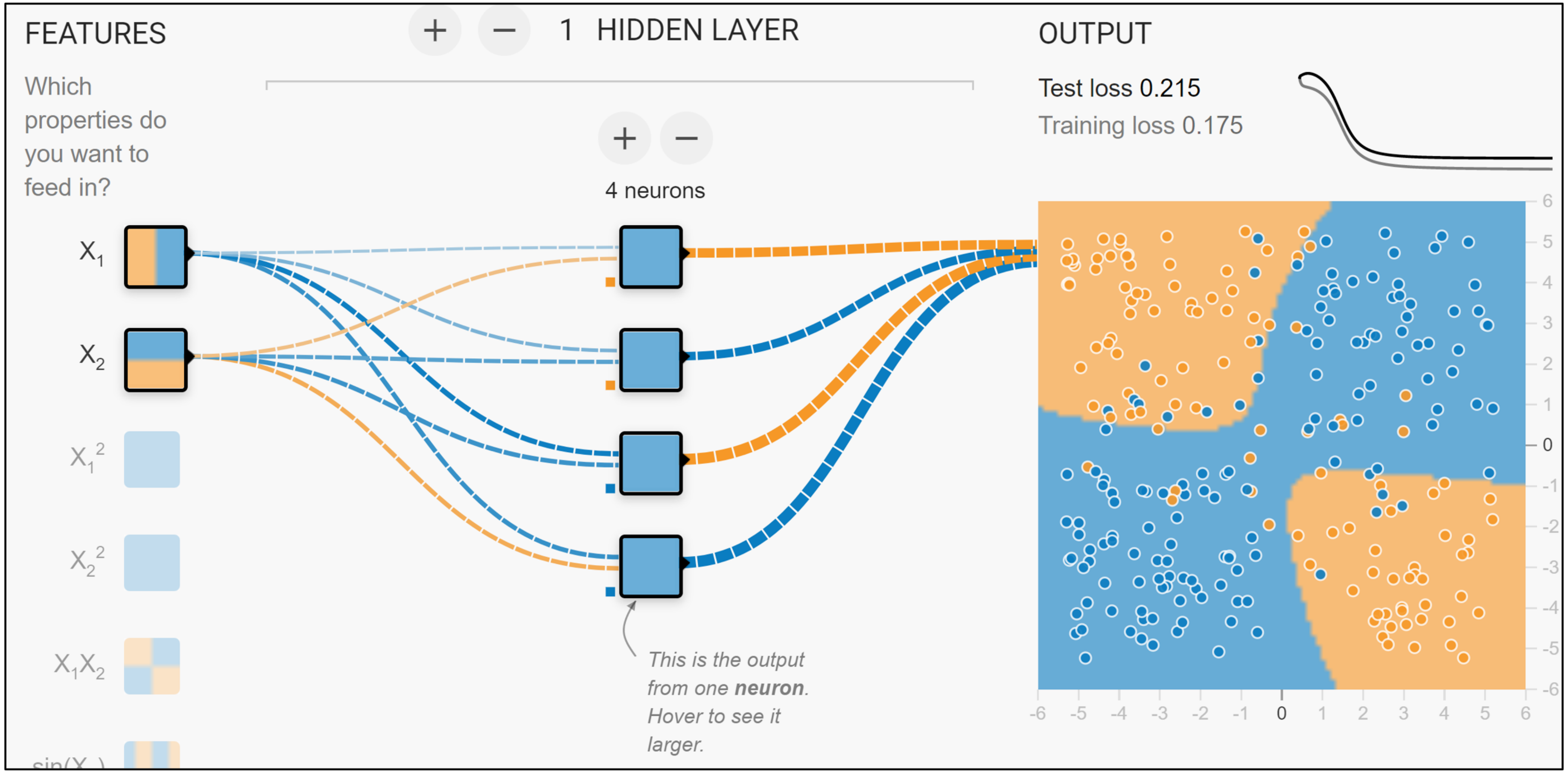
7.1 ニューラルネットワークの構成

- ニューラルネットワークによる非線形識別面の実現



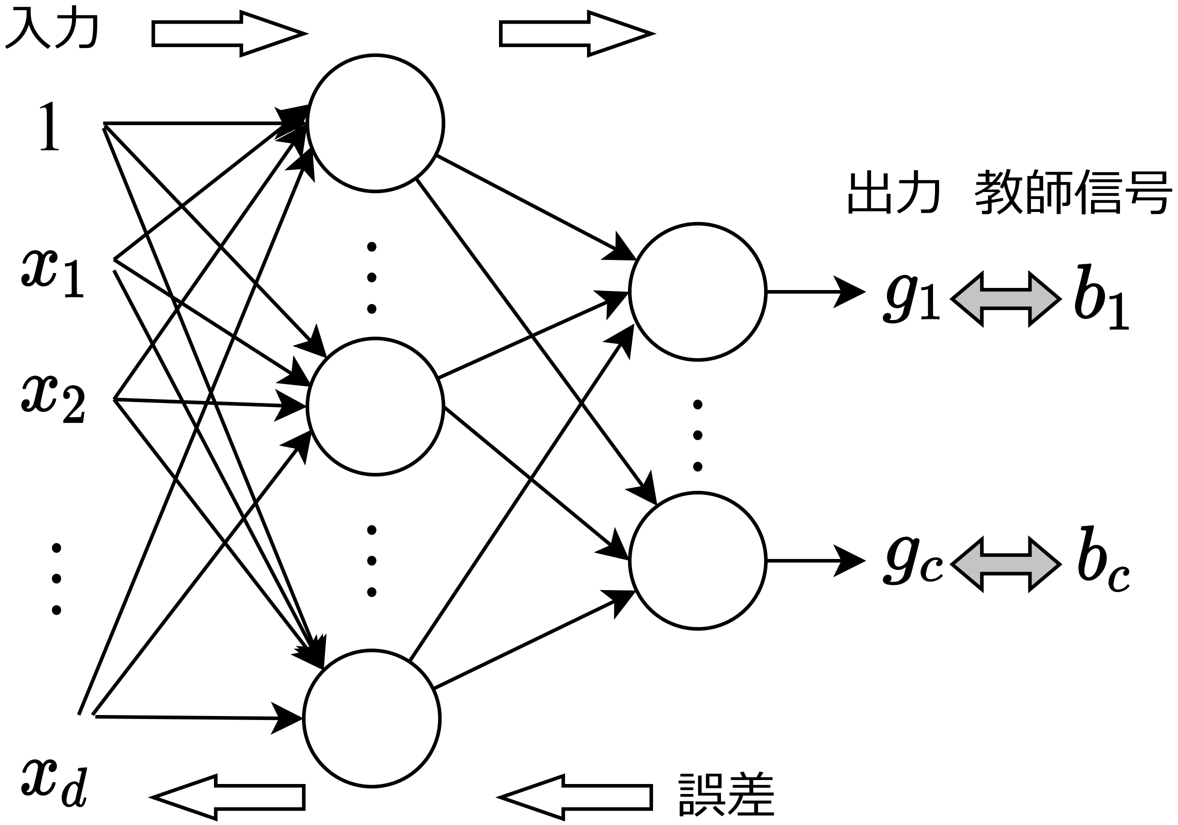
7.1 ニューラルネットワークの構成

- ニューラルネットワークによる非線形識別面の実現



7.2 誤差逆伝播法による学習

- 誤差逆伝播法の名前の由来



7.2 誤差逆伝播法による学習

- 結合重みの調整アルゴリズム
  - 特定のデータ $\boldsymbol{x}_p$ に対する二乗誤差

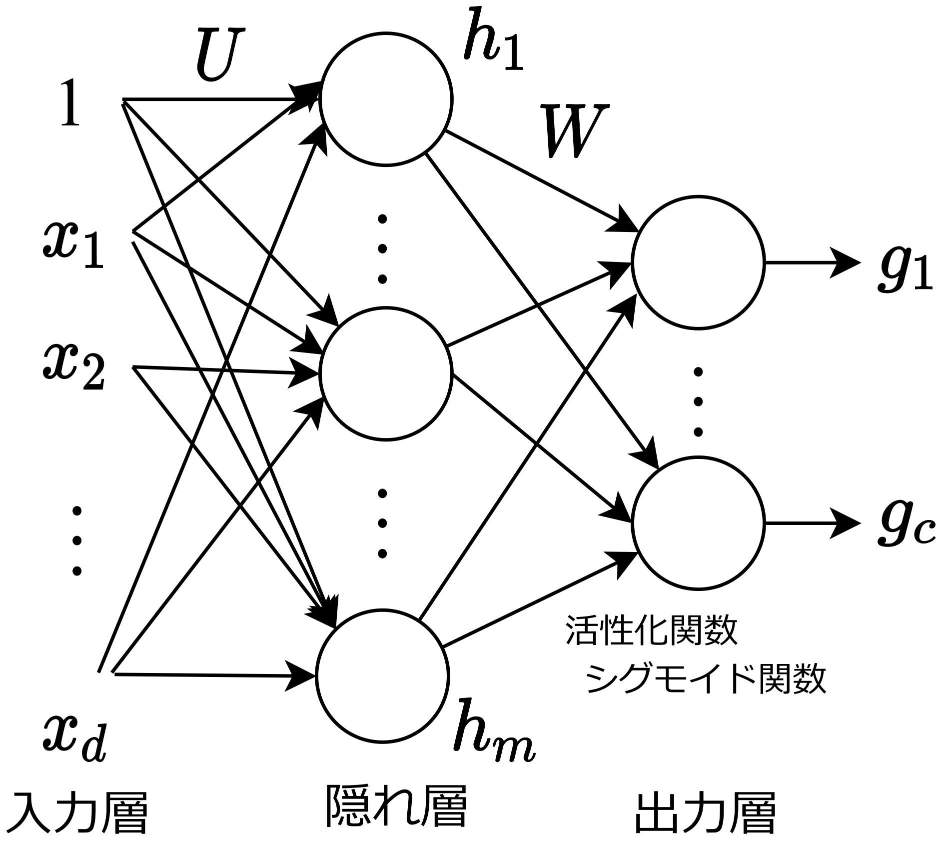
$$J(\boldsymbol{w}) \equiv \frac{1}{2}\sum_{i=1}^{c}(g_i(\boldsymbol{x}_p) - b_i)^2$$

  - 誤差 J は重み $\boldsymbol{w}$ の関数
    - $\boldsymbol{w}$ を J の勾配方向へ一定量だけ動かすことを繰り返して、最適解へ収束させる（→勾配降下法）

$$\boldsymbol{w}' \leftarrow \boldsymbol{w} - \rho\frac{\partial J}{\partial \boldsymbol{w}}$$

7.2 誤差逆伝播法による学習

- 学習対象のネットワーク



7.2 誤差逆伝播法による学習

- 入力値: $\boldsymbol{x} = (x_1, \ldots, x_d)^T$
- 隠れ層の値: $\boldsymbol{h} = (h_1, \ldots, h_m)^T$
- 出力値: $\boldsymbol{g} = (g_1, \ldots, g_c)^T$
- 入力ノードと隠れ層ノードとの結合重み: $\boldsymbol{U} = \{u_{j \leftarrow i}\}$
- 隠れ層ノードと出力ノードとの結合重み: $\boldsymbol{W} = \{w_{k \leftarrow j}\}$

隠れ層と出力層との間の重み $\boldsymbol{W}$ の調整

- 隠れ層の出力の重み付き和

$$s_k = \sum_j w_{k \leftarrow j} h_j$$

- 出力層の出力（活性化関数はシグモイド関数）

$$g_k = sigmoid(s_k) = \frac{1}{1 + exp(-s_k)}$$

隠れ層と出力層との間の重み $\boldsymbol{W}$ の調整

- J の勾配計算

$$\frac{\partial J}{\partial w_{k \leftarrow j}} = \frac{\partial J}{\partial g_k}\frac{\partial g_k}{\partial s_k}\frac{\partial s_k}{\partial w_{k \leftarrow j}}$$

- 右辺第1項: J の定義式から

$$\frac{\partial J}{\partial g_k} = \frac{\partial}{\partial g_k}\frac{1}{2}(g_k - b_k)^2 = (g_k - b_k)$$

- 右辺第2項: 活性化関数（シグモイド関数）の微分

$$\frac{\partial g_k}{\partial s_k} = g_k' = g_k(1 - g_k)$$

隠れ層と出力層との間の重み $\boldsymbol{W}$ の調整

- 右辺第3項: s_k の定義式から

$$\frac{\partial s_k}{\partial w_{k \leftarrow j}} = \frac{\partial}{w_{k \leftarrow j}} \sum_j w_{k \leftarrow j} h_j = h_j$$

- 右辺のまとめ

$$\frac{\partial E}{\partial w_{k \leftarrow j}} = (g_k - b_k) g'_k h_j$$

- 誤差項 δ_k の定義

$$\delta_k = (g_k - b_k) g'_k$$

入力層と隠れ層との間の重み U の調整

- 入力層の値の重み付き和

$$z_j = \sum_i u_{j \leftarrow i} x_i$$

- 隠れ層の出力

$$h_j = sigmoid(z_j)$$

- Jの勾配

$$\frac{\partial J}{\partial u_{j \leftarrow i}} = \frac{\partial J}{\partial h_j} \frac{\partial h_j}{\partial z_j} \frac{\partial z_j}{\partial u_{j \leftarrow i}}$$

入力層と隠れ層との間の重み U の調整

- 右辺第1項の計算
  - J に直接影響を与える変数は複数の g_k
  - g_k は s_k から、s_k は h_j から計算されている

$$\frac{\partial J}{\partial h_j} = \sum_k \frac{\partial J}{\partial g_k} \frac{\partial g_k}{\partial s_k} \frac{\partial s_k}{\partial h_j}$$

- 上式右辺の和の内部第1項と第2項の積を誤差項δ_kを使って書き換え

$$\frac{\partial J}{\partial g_k} \frac{\partial g_k}{\partial s_k} = (g_k - b_k) g'_k = \delta_k$$

- 和の内部第3項の計算

$$\frac{\partial s_k}{\partial h_j} = \frac{\partial}{\partial h_j} \sum_k w_{k \leftarrow j} h_j = w_{k \leftarrow j}$$

入力層と隠れ層との間の重み U の調整

- Jの勾配式の右辺第1項

$$\frac{\partial J}{\partial h_j} = \sum_k \delta_k w_{k\leftarrow j}$$

- 右辺第2項は活性化関数の微分: h'_j
- 右辺第3項: z_jの定義式から

$$\frac{\partial z_j}{\partial u_{j\leftarrow i}} = \frac{\partial}{u_{j\leftarrow i}} \sum_i u_{j\leftarrow i} x_i = x_i$$

- Jの勾配

$$\frac{\partial J}{\partial u_{j\leftarrow i}} = \sum_k (\delta_k w_{k\leftarrow j}) h'_j x_i$$

誤差逆伝播法（確率的勾配降下法）

1. ネットワークの重み $\boldsymbol{W}$, $\boldsymbol{U}$ を適当な初期値に設定
2. 全データに対する学習を1エポックとし、エポック数だけ以下繰り返し
   i. 入力 $\boldsymbol{x}_i$ に対するネットワークの出力を計算
   ii. 隠れ層と出力層の間の重み $\boldsymbol{W}$ の調整と誤差項 $\delta_k = (g_k - b_k)g'_k$ の計算

$$w'_{k\leftarrow j} \leftarrow w_{k\leftarrow j} - \rho \frac{\partial J}{\partial w_{k\leftarrow j}} = w_{k\leftarrow j} - \rho(g_k - b_k)g'_k h_j$$

   iii. δ_kを利用して入力層と隠れ層の間の重み $\boldsymbol{U}$ を調整

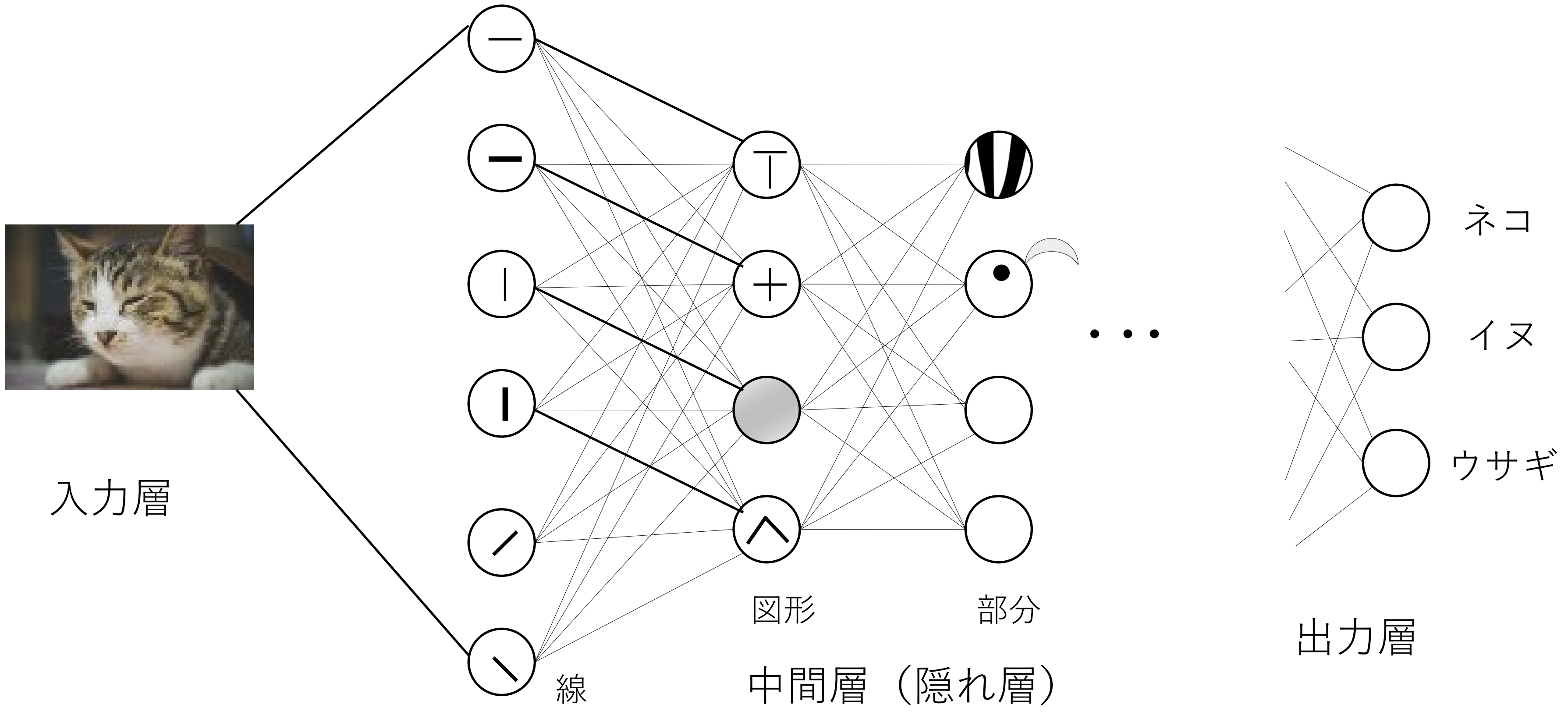
$$u'_{j\leftarrow i} \leftarrow u_{j\leftarrow i} - \rho \frac{\partial J}{\partial u_{j\leftarrow i}} = u_{j\leftarrow i} - \rho(\sum_k \delta_k w_{k\leftarrow j})h'_j x_i$$

7.2 誤差逆伝播法による学習

- 過学習に気をつけよう
    - ニューラルネットワークは非線形識別面を学習することができるので、学習データの誤識別率を限りなく0に近づけることができる
    - そのような識別面は、未知データに対して誤識別率が高いことが多い
    - 学習データに特化しすぎた識別面が学習される現象を過学習とよぶ

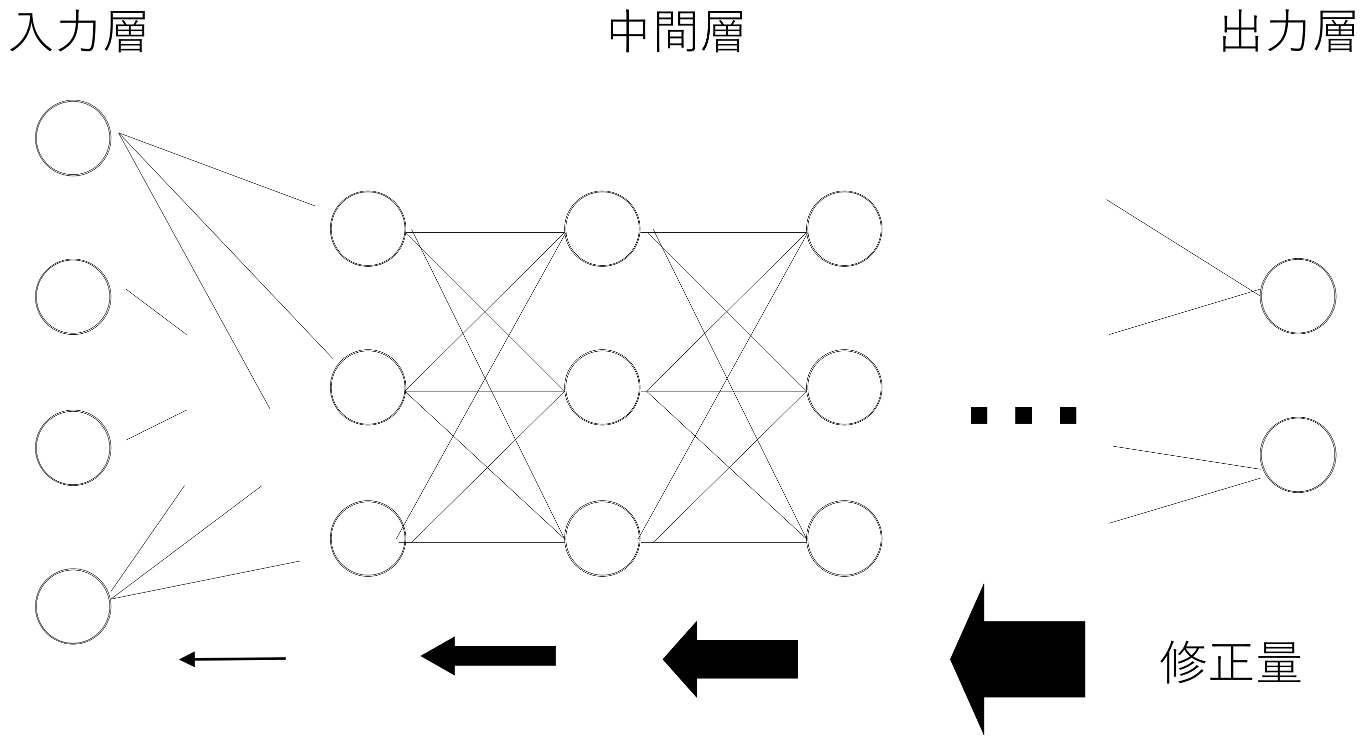
7.3 ディープニューラルネットワーク

- 深層学習：多階層ニューラルネットワークによる学習
  - ◆ 多階層での学習を可能にする工夫
  - ◆ 問題に特化したネットワーク構造の導入

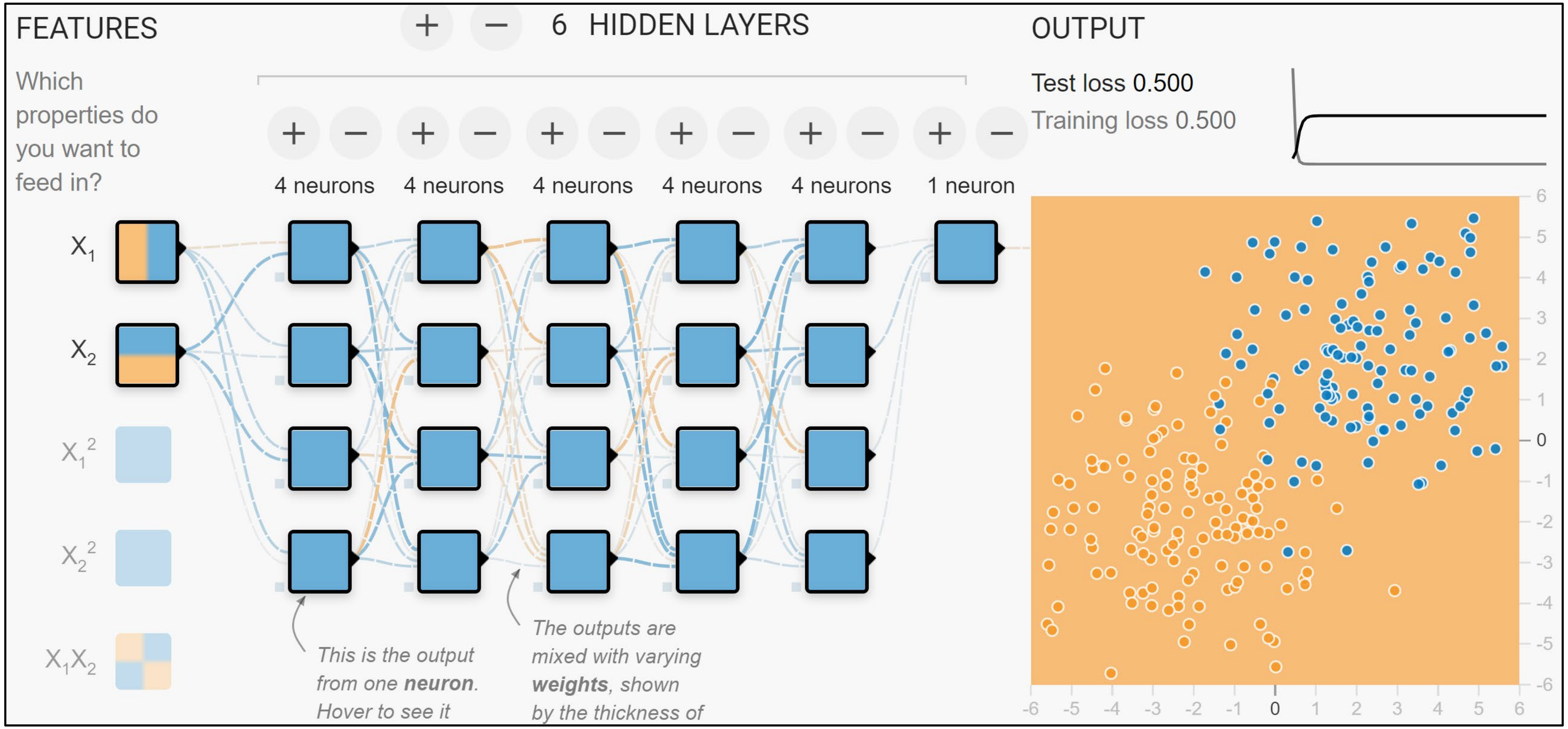


7.3.1 勾配消失問題とは

- 多階層における誤差逆伝播法の問題点
  - 入力層に近づくにつれて修正量が消失する



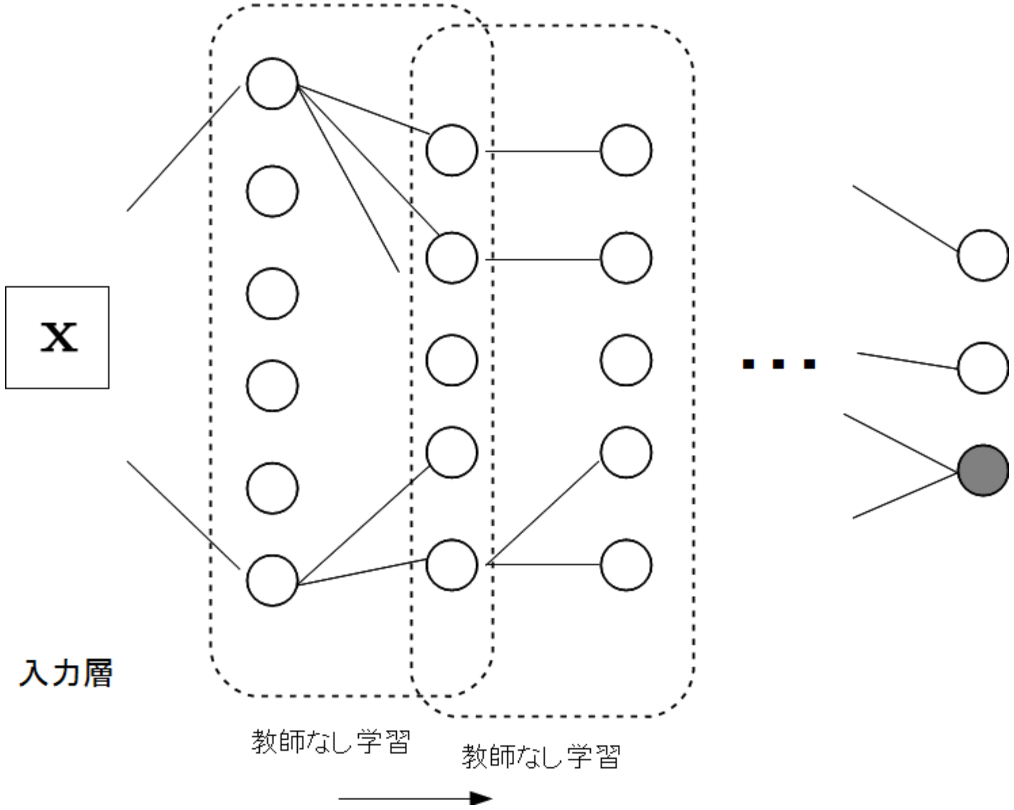
7.3.1 勾配消失問題とは



https://playground.tensorflow.org/

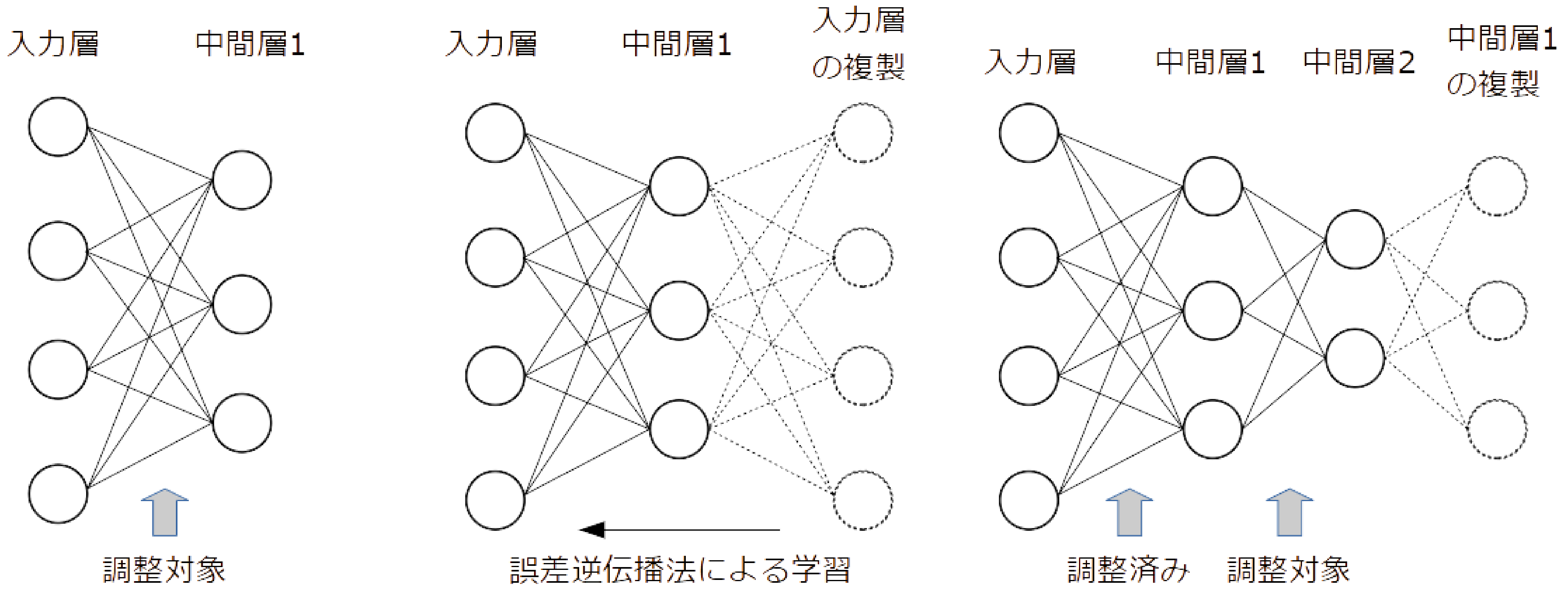
7.3.2 多階層学習における工夫

- 事前学習法
  - ◆ 深層学習における初期パラメータ学習



7.3.2 多階層学習における工夫

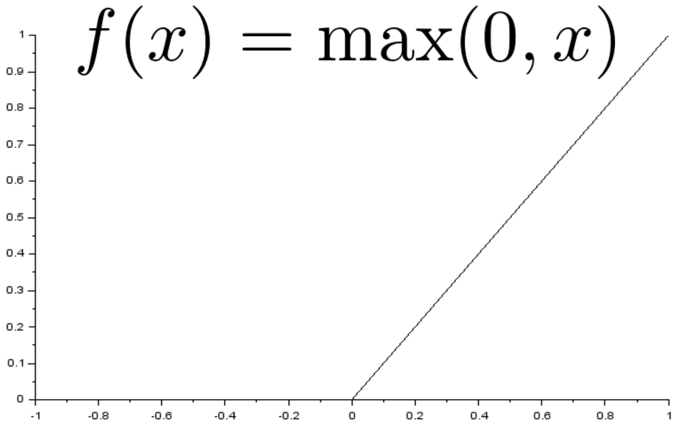
- 事前学習法のアイディア
  - ◆ 自己写像学習による重みの初期値設定



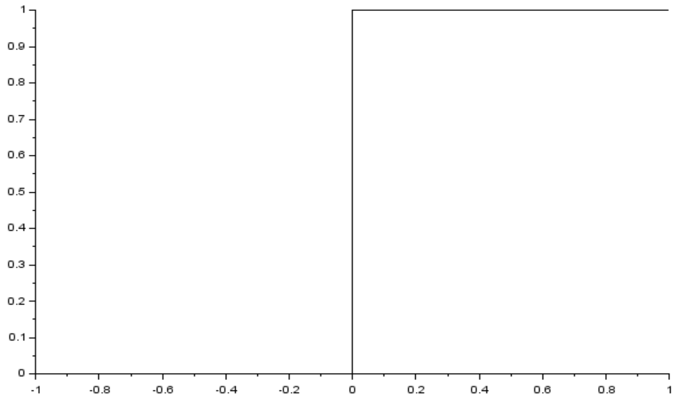
(a) 事前調整対象の重み　(b) オートエンコーダによる復元学習　(c) 1階層上の事前調整

7.3.2 多階層学習における工夫

- 活性化関数をrectified linear関数(ReLU)に変更
- 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



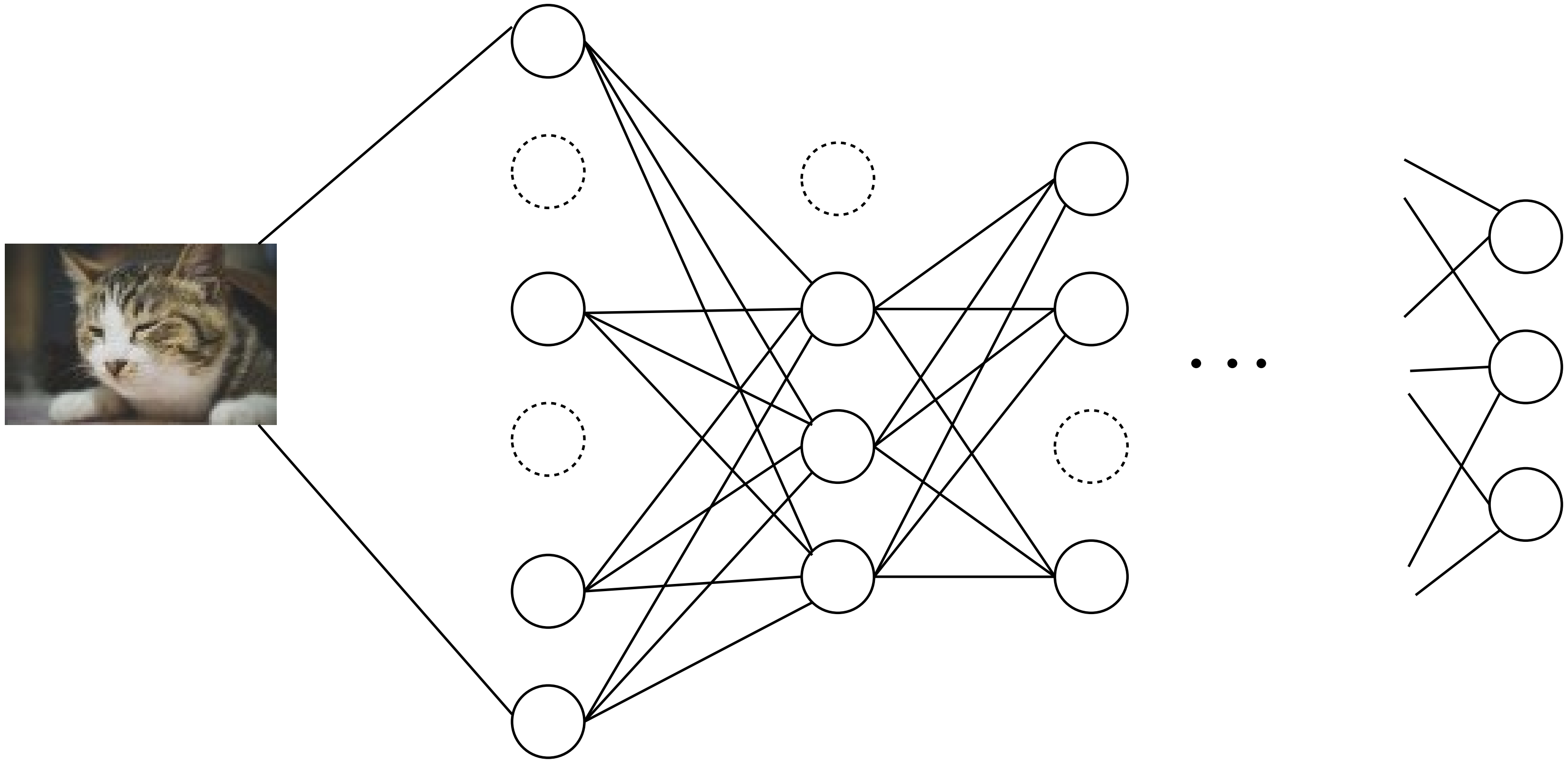
(a) rectified linear 関数



(b) (a)の導関数

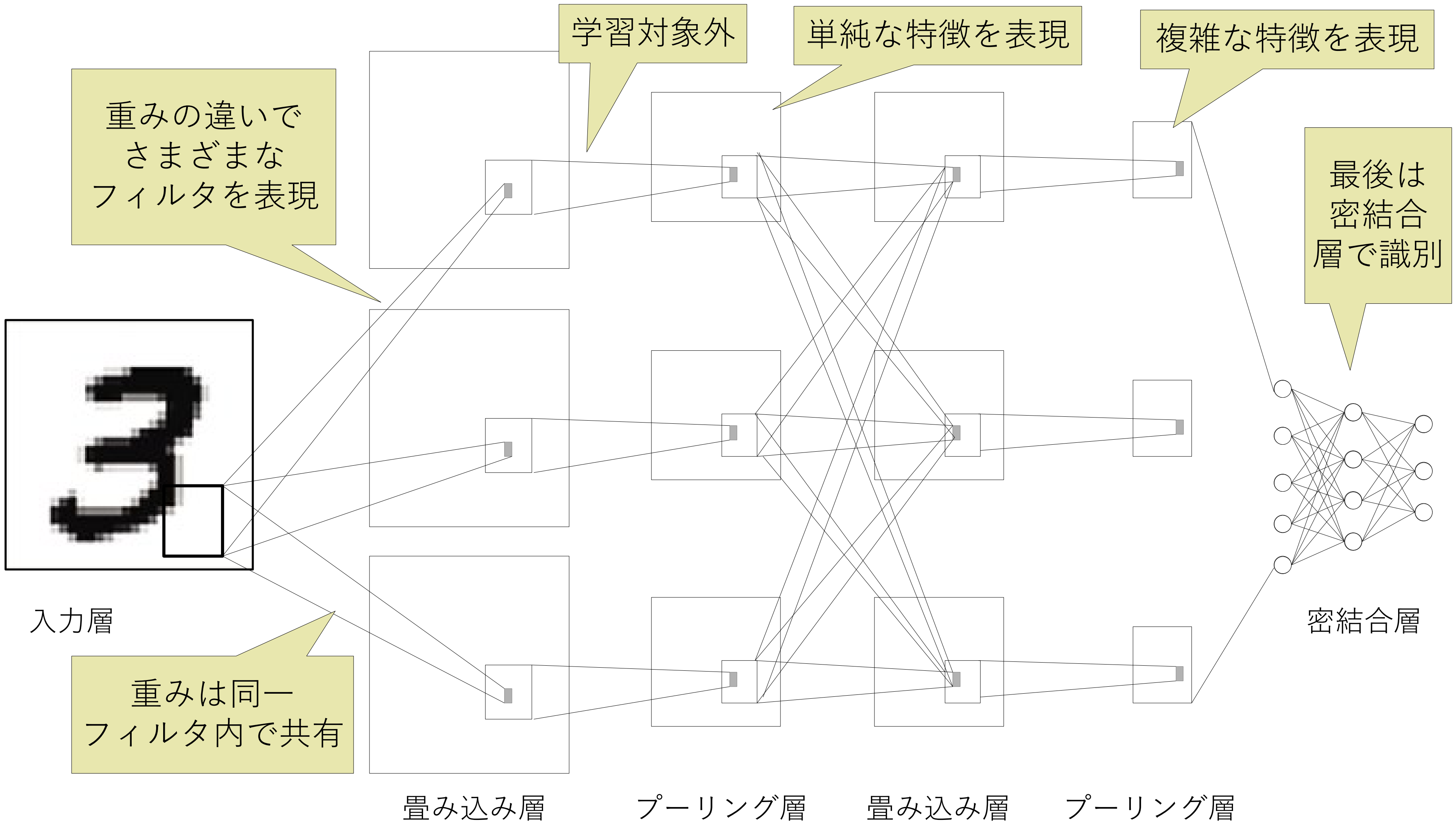
7.3.2 多階層学習における工夫

- 過学習の回避
  - ◆ ドロップアウト：ランダムに一定割合のユニットを消して学習を行う



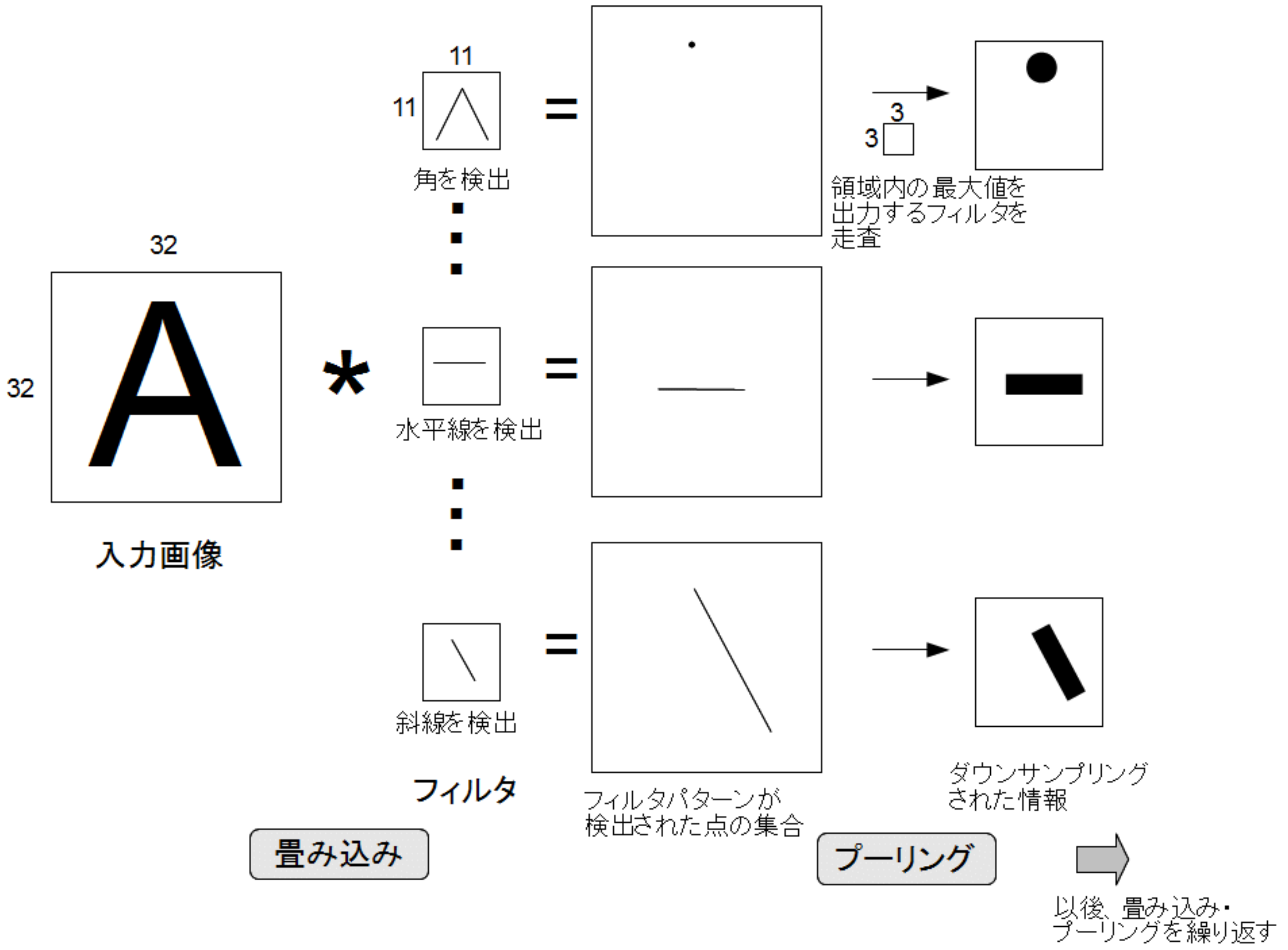
7.3.3 特化した構造をもつニューラルネットワーク

- 畳み込みニューラルネットワーク



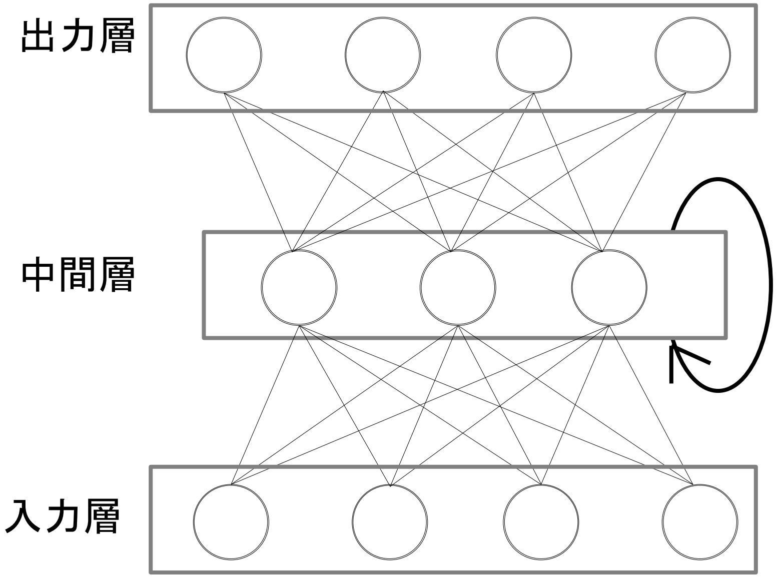
7.3.3 特化した構造をもつニューラルネットワーク

- 畳み込みニューラルネットワークの演算

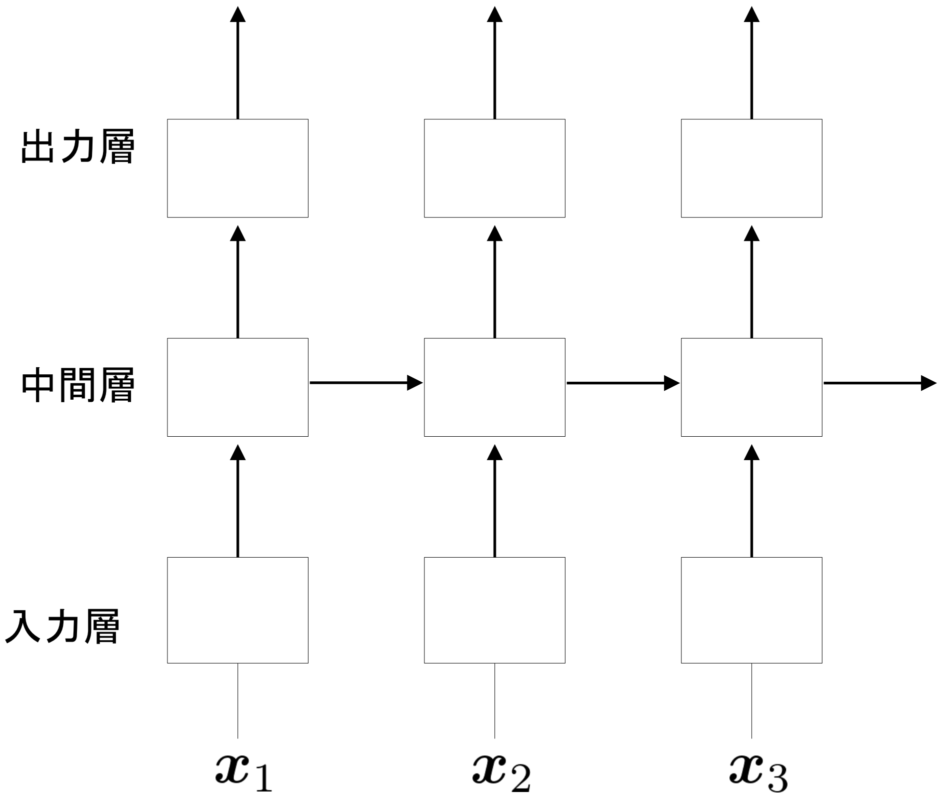


7.3.3 特化した構造をもつニューラルネットワーク

- リカレントニューラルネットワーク
  - ◆ 時系列信号の認識や自然言語処理に適する



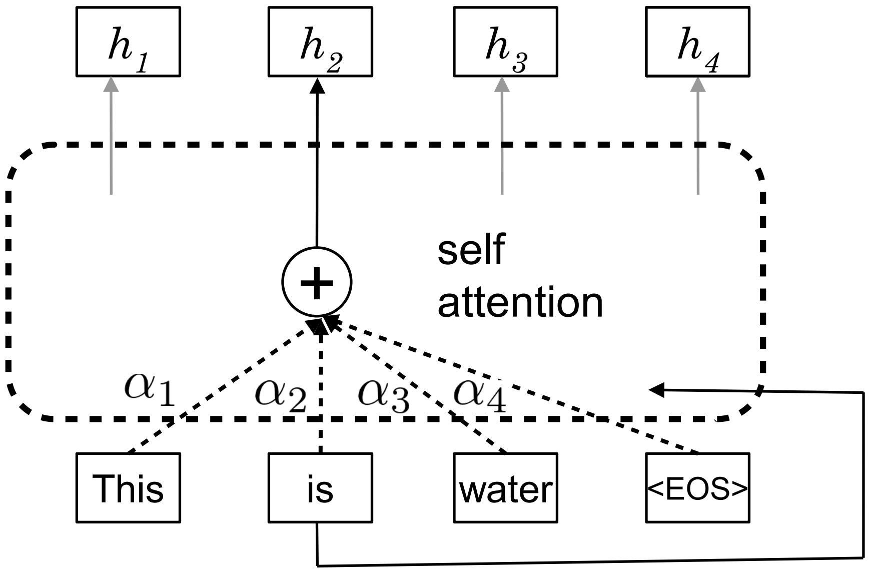
(a) リカレントニューラルネットワーク

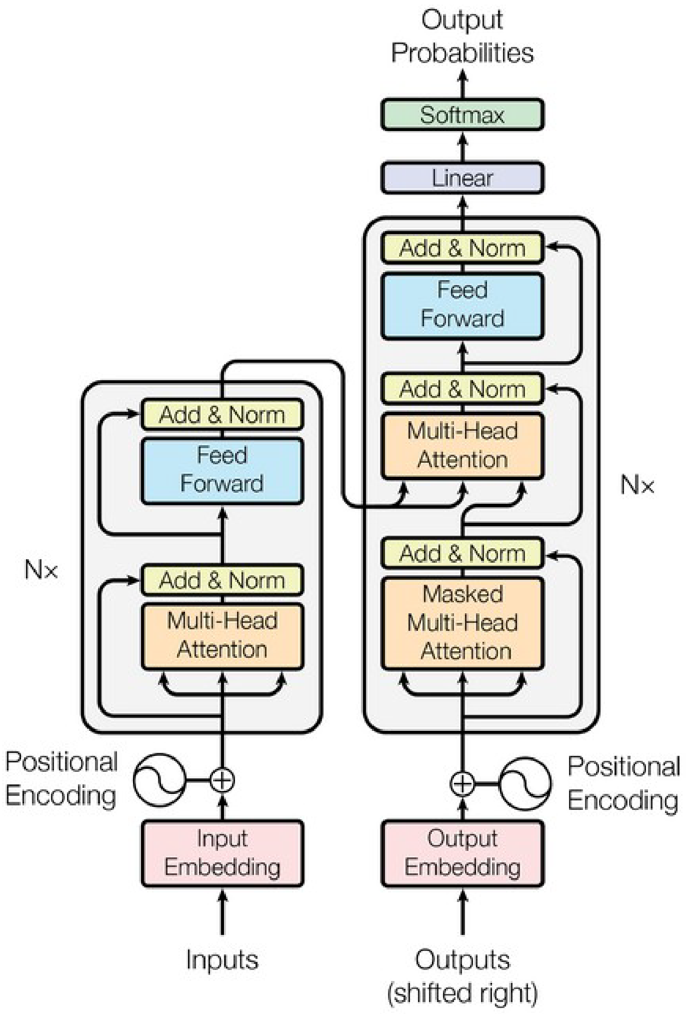


(b) 帰還路を時間方向に展開

7.3.3 特化した構造をもつニューラルネットワーク

- Transformer: Self-αttention + フィードフォワードNN
  - ◆ 自分の中間表現を作るときに、入力の他の部分との関係を計算
  - ◆ BERTなどの事前学習モデルに使われる





Ashish Vaswani et.al. : Attention Is All You Need, arXiv:1706.03762

まとめ

- ニューラルネットワークは誤差を最小にする確率的勾配降下法の枠組みで非線形識別面を学習できる
- 多階層のニューラルネットワークは誤差逆伝播法を用いる
- 勾配消失問題などで学習がうまくゆかないことがあったが、現在では様々な工夫により深層学習が可能になっている