

The Multi-Layer Perceptron

Toru Wakahara

[1] 多層パーセプトロンの構成

- しきい値ユニット
- sigmoidal ユニット

[2] 多層パーセプトロンの写像能力

- 3層の場合
- 2層パーセプトロンの普遍性

[3] 多層パーセプトロンの学習

- エラー関数の微分
- 誤差逆伝播法

多層パーセプトロン

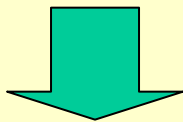
重みベクトルを多層に持つ feed-forward 系

ユニット活性化関数

(1) 単一変数

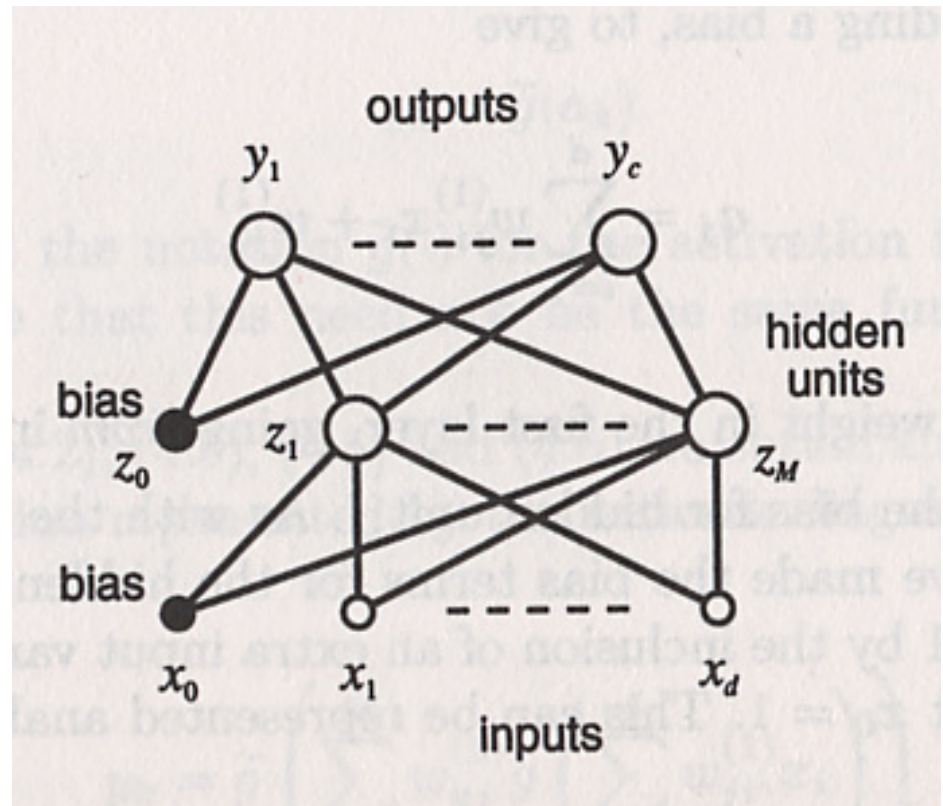
(2) 非線形

しきい値ユニット



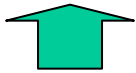
(3) 微分可能

sigmoidal ユニット



2層ネットワーク

$$y_k = \tilde{g}(a_k)$$



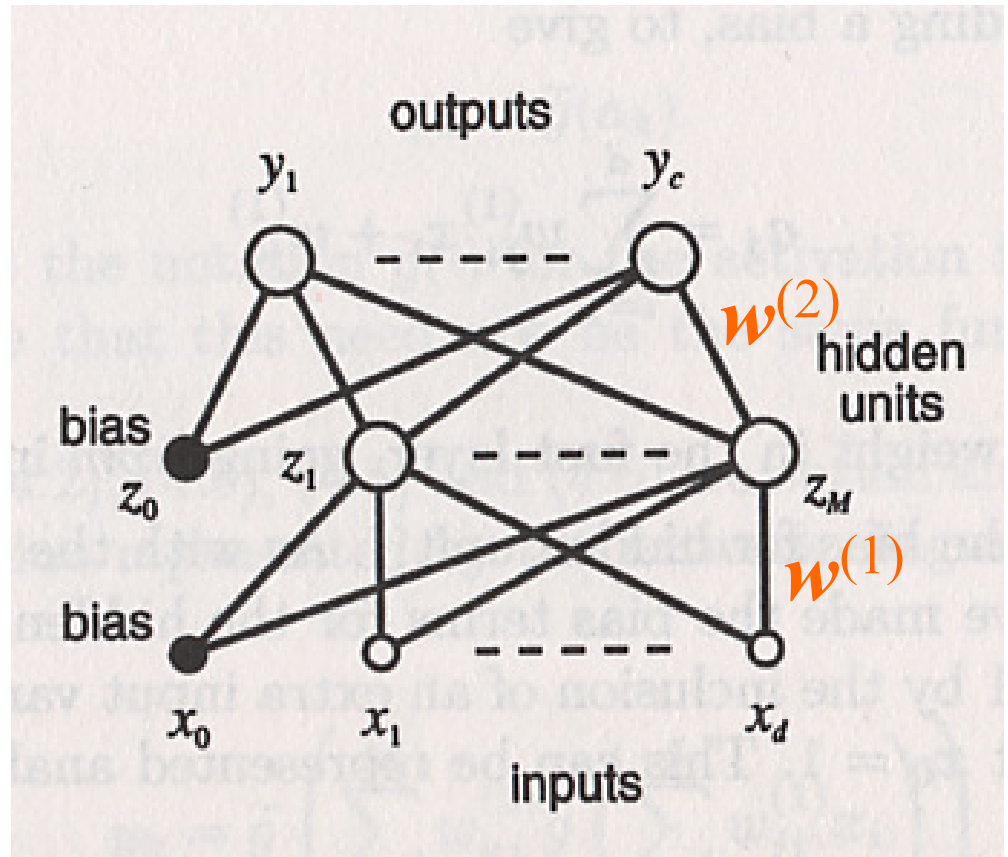
$$a_k = \sum_{j=0}^M w_{kj}^{(2)} z_j$$



$$z_j = g(a_j)$$



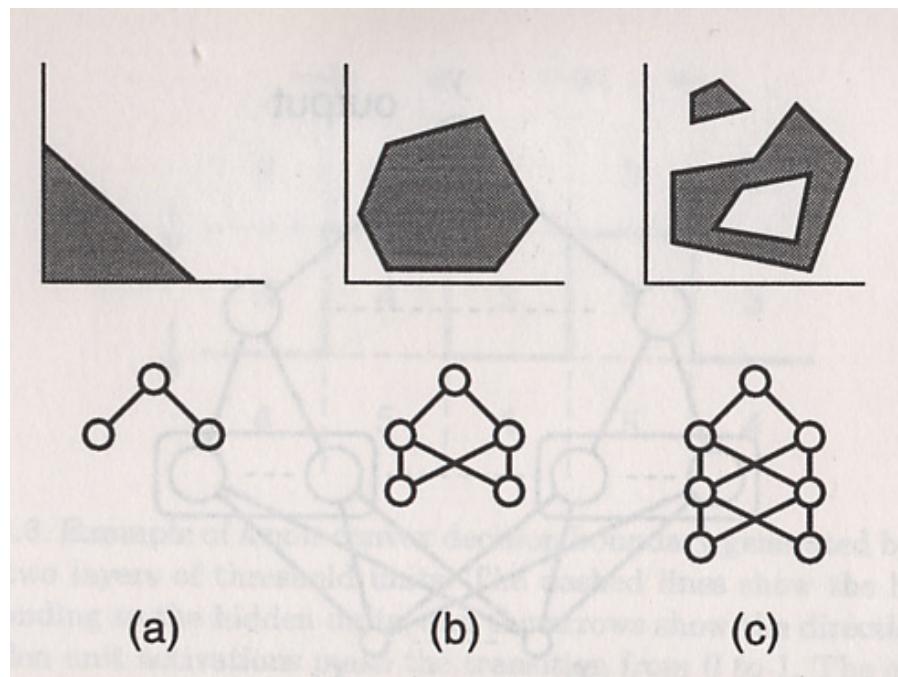
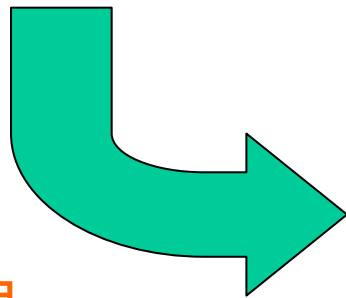
$$a_j = \sum_{i=1}^d w_{ji}^{(1)} x_i + w_{j0}^{(1)} = \sum_{i=0}^d w_{ji}^{(1)} x_i$$



しきい値ユニットをもつ 多層ニューラルネットの決定境界

$$g(a) = \begin{cases} 0 & \text{when } a < 0 \\ 1 & \text{when } a \geq 0 \end{cases}$$

可能な
決定境界



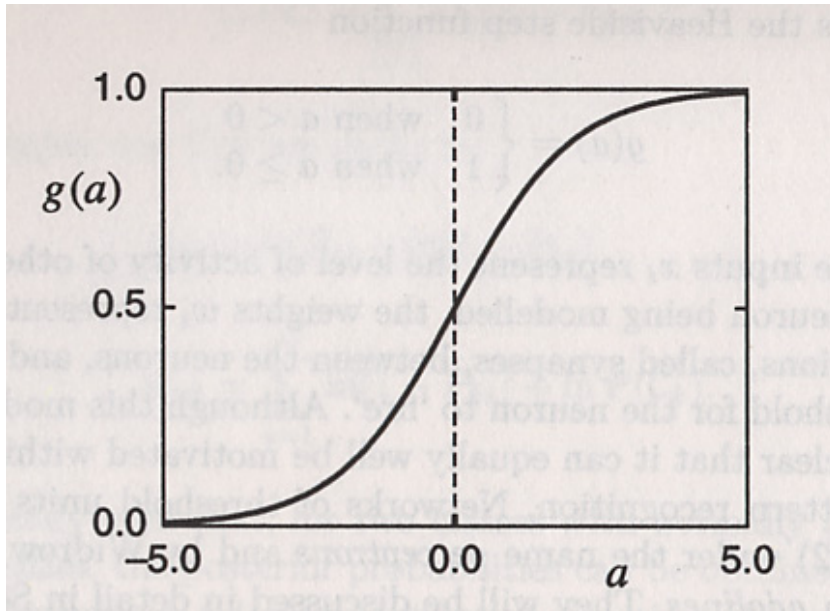
1層

2層

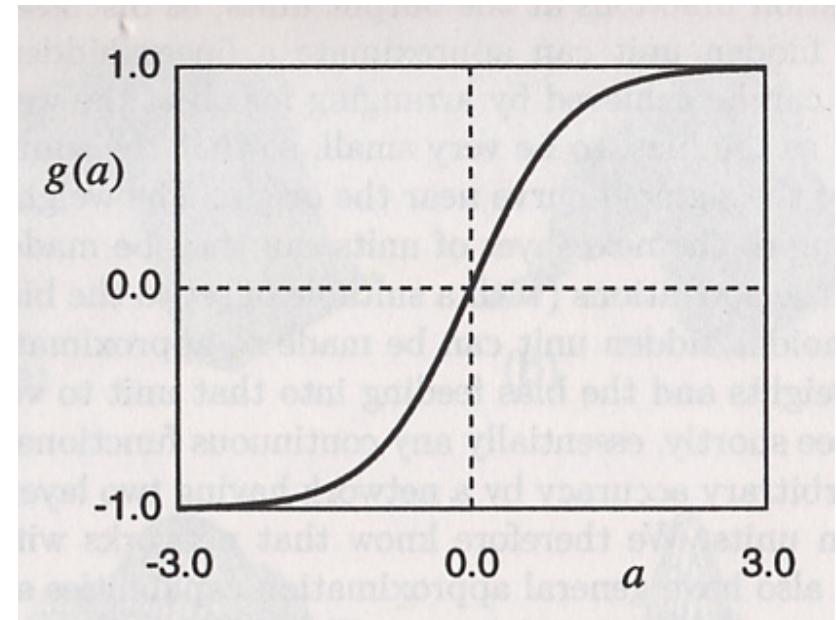
3層

sigmoidal ユニット

sigmoid



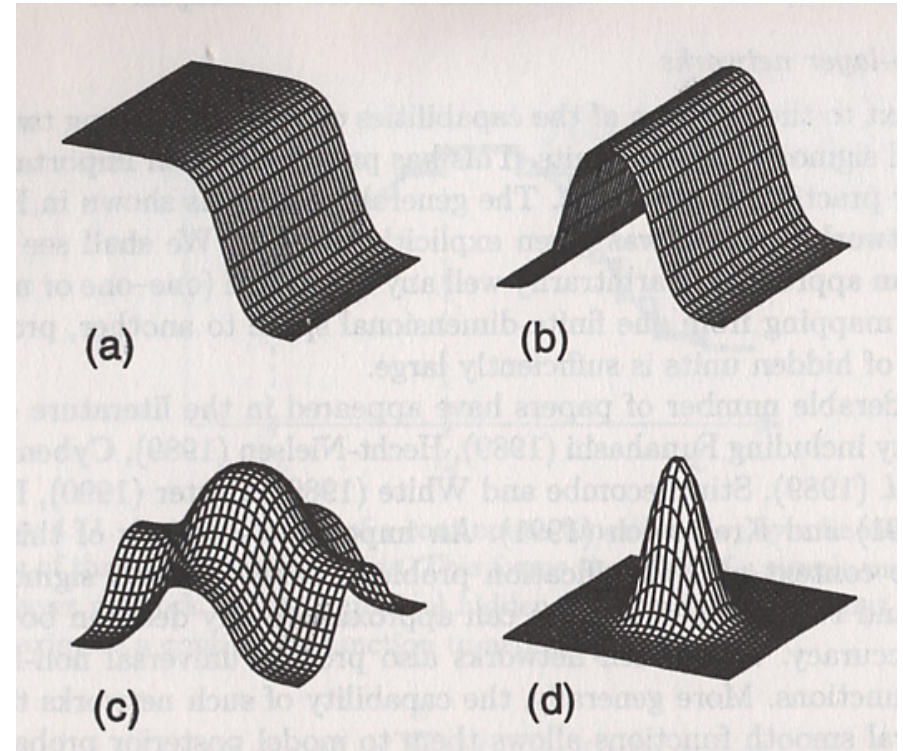
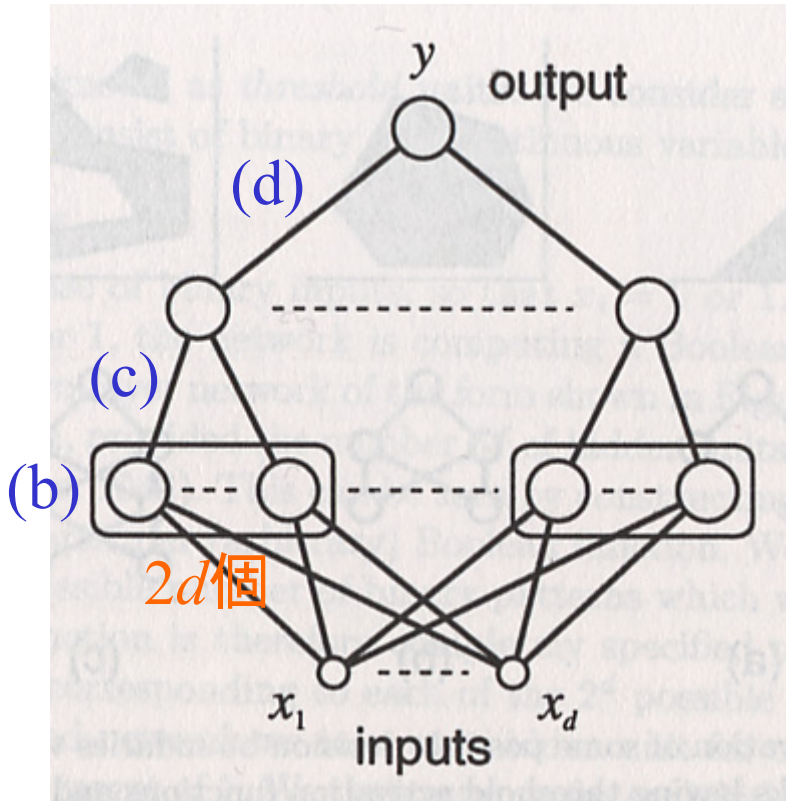
tanh



$$g(a) = \frac{1}{1 + \exp(-a)}$$

$$g(a) \equiv \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

3層パーセプトロンの写像能力



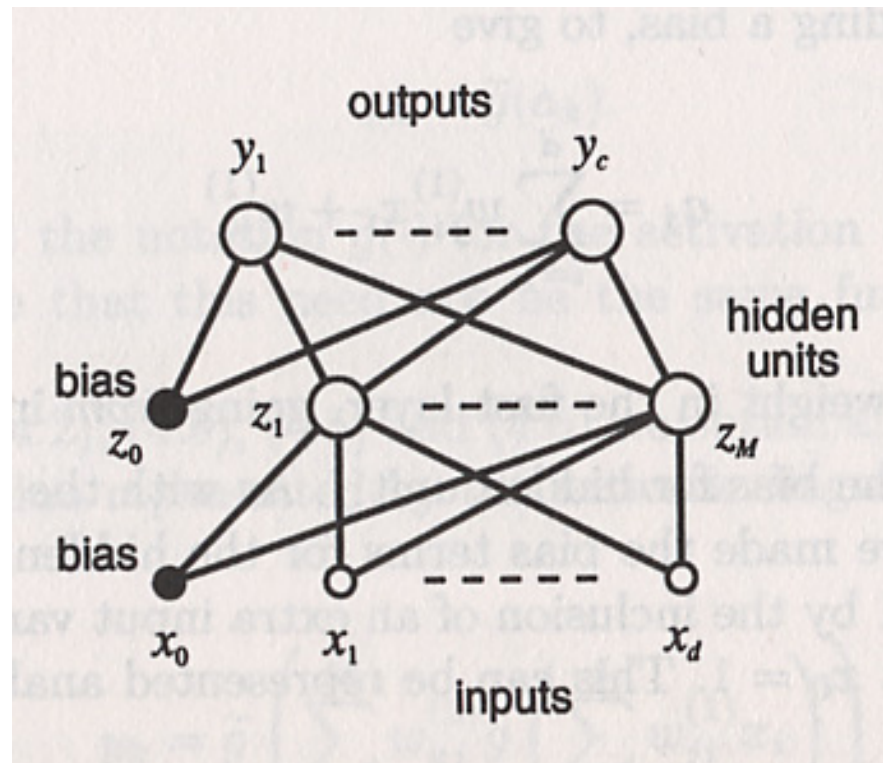
第1, 2層はsigmoidユニット, 第3層は線形ユニット

任意の連続写像は (d) の線形重ね合せ!

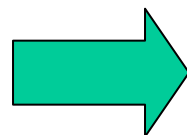
2層パーセプトロンの写像能力

$$y_k = g \left(\sum_{j=0}^M w_{kj}^{(2)} g \left(\sum_{i=0}^d w_{ji}^{(1)} x_i \right) \right)$$

$g(\bullet)$: sigmoidal function

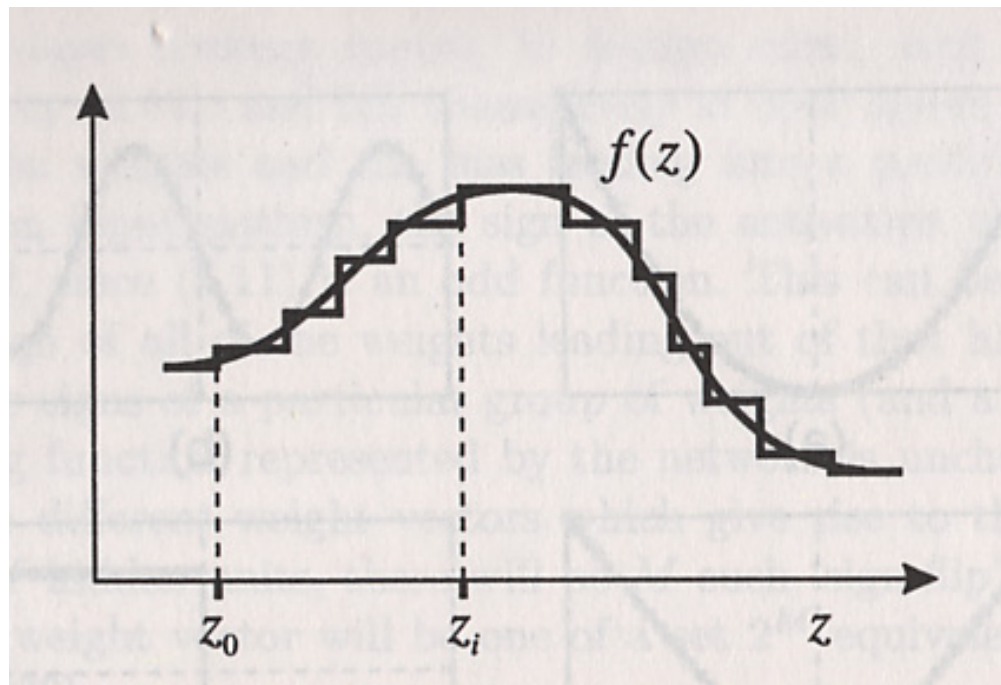


hidden units の数 M
が十分大きければ,
有限次元の連続写像
を任意精度で近似!



2層パーセプトロン
の普遍性

連続関数はしきい値関数の線形和 で近似できる！



$$f(z) \cong f_0 + \sum_{i=0}^N \{f_{i+1} - f_i\} H(z - z_i)$$

2層パーセプトロンの普遍性の証明 (1)

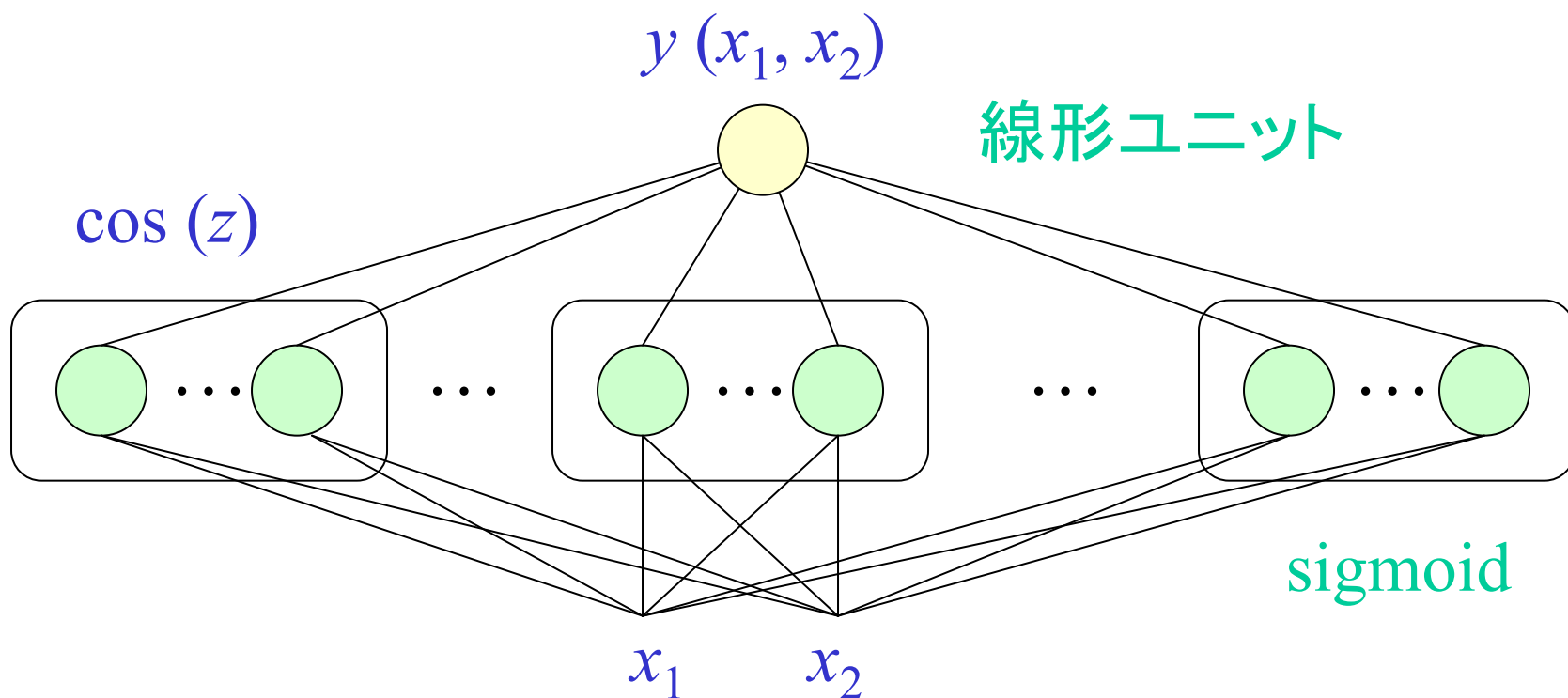
任意の連続写像 $y = y(x_1, x_2)$ の場合:

$$\begin{aligned} y(x_1, x_2) &\cong \sum_s A_s(x_1) \cos(sx_2) \\ &\cong \sum_s \sum_l A_{sl} \cos(lx_1) \cos(sx_2) \\ &= \sum_s \sum_l \frac{1}{2} A_{sl} (\cos(z_{sl}) + \cos(z'_{sl})) \end{aligned}$$

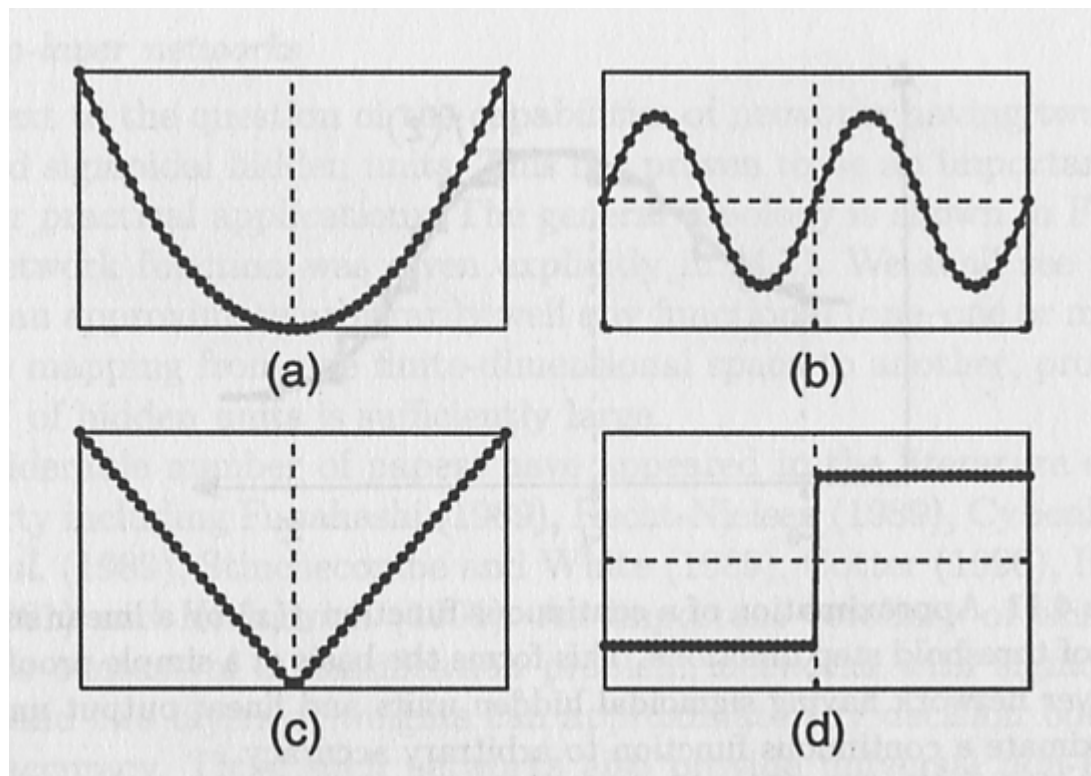
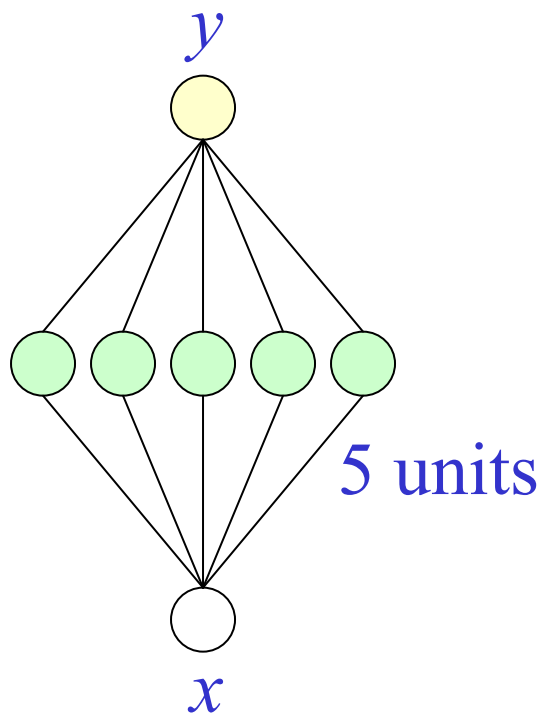
ここで、連続関数 $\cos(z)$ はしきい値関数の線形和で近似できることに注目する

2層パーセプトロンの普遍性の証明 (2)

さらに, しきい値関数は sigmoid 関数で近似できる
→ 任意の連続写像 $y = y(x_1, x_2)$ が近似できる



2層パーセプトロンの適用例



- : 線形ユニット
- : tanh ユニット

The BFGS quasi-Newton 法
で 1000 回学習

多層パーセプトロン学習の課題

[1] エラー関数の選択

$$E = \frac{1}{2} \sum_{n=1}^N \left\| \mathbf{y}(\mathbf{x}_n; \mathbf{w}) - \mathbf{t}^n \right\|^2, \quad \{ \mathbf{t}^n \}_{n=1}^N : \text{教師データ}$$

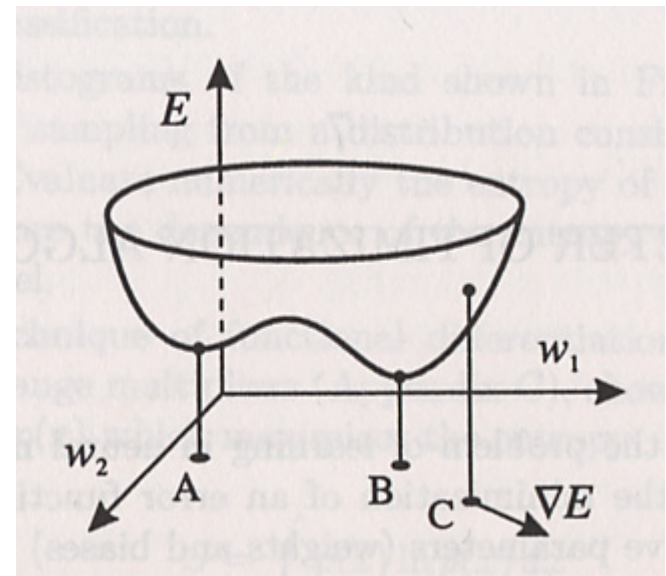
2乗和エラー関数

[2] パラメータ最適化手法

$$\Delta \mathbf{w}^{(\tau)} = -\eta \nabla E^n \Big|_{\mathbf{w}^{(\tau)}}$$

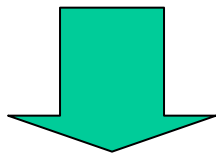
勾配法

(Gradient descent)

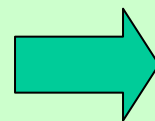


活性化関数の選択

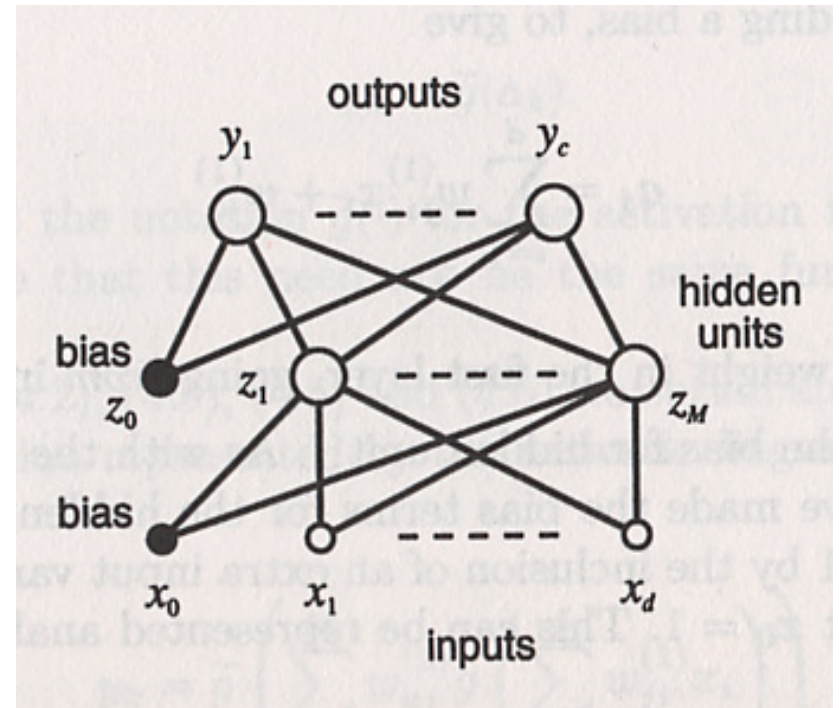
しきい値ユニットの場合:
 w の学習ができない!
The credit assignment
problem



微分可能な活性化関数
 $g(\cdot)$ の選択



エラー関数 E も
 w で微分可能



エラー関数 E の微分 (1)

活性化関数の働き:

$$a_j = \sum_i w_{ji} z_i \rightarrow z_j = g(a_j)$$

エラー関数:

$$E = \sum_n E^n \quad \text{where } E^n = E^n(y_1, \dots, y_c)$$

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j \frac{\partial a_j}{\partial w_{ji}} \quad \text{where } \delta_j \equiv \frac{\partial E^n}{\partial a_j}$$

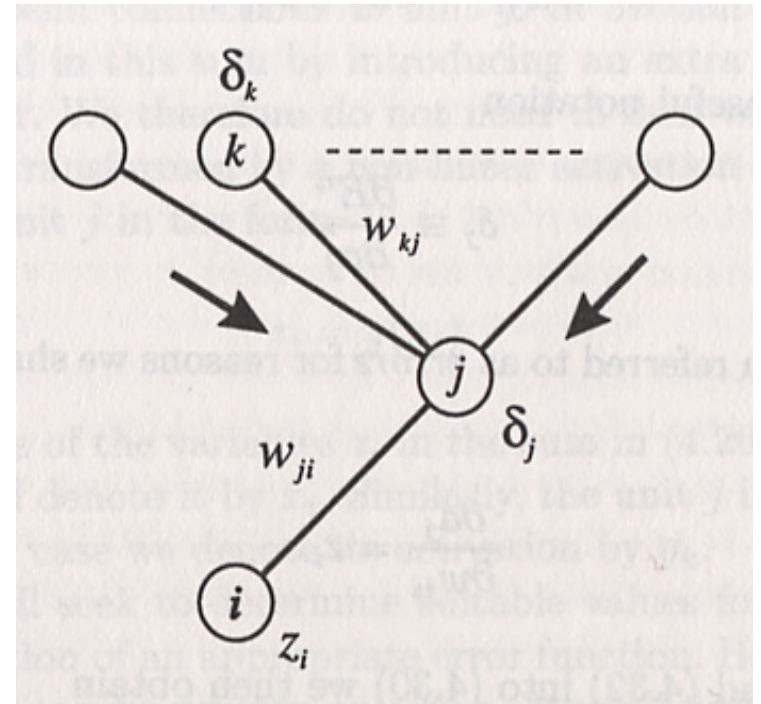
エラー関数 E の微分 (2)

$$a_j = \sum_i w_{ji} z_i \rightarrow \frac{\partial a_j}{\partial w_{ji}} = z_i \quad \therefore \frac{\partial E^n}{\partial w_{ji}} = \delta_j z_i \quad (1)$$

出力層の場合:

$$y_k = g(a_k) \text{ より}$$

$$\delta_k \equiv \frac{\partial E^n}{\partial a_k} = g'(a_k) \frac{\partial E^n}{\partial y_k}$$

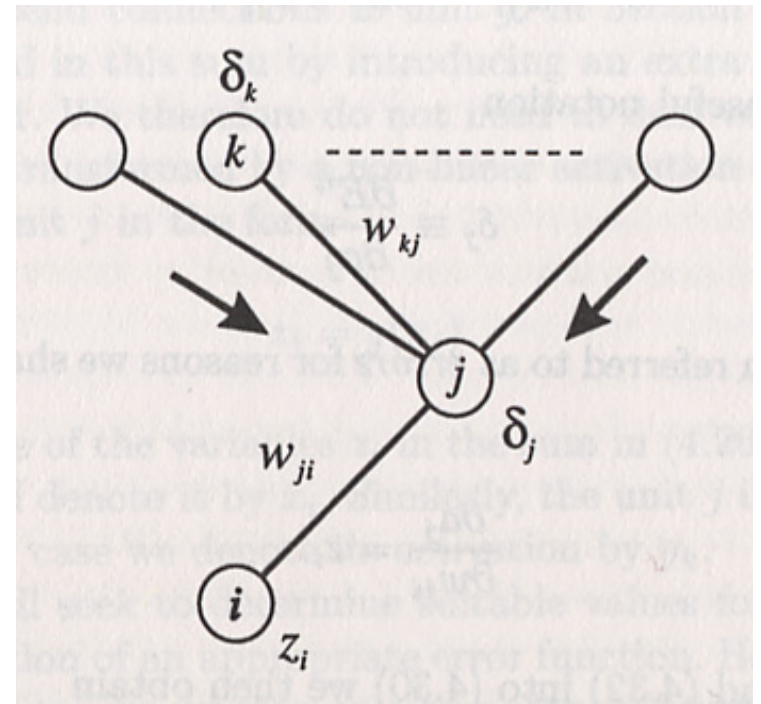


エラー関数 E の微分 (3)

$$a_k = \sum_j w_{kj} z_j = \sum_j w_{kj} g(a_j), \quad \delta_k \equiv \frac{\partial E^n}{\partial a_k} \quad \text{を用いて}$$

中間層の場合:

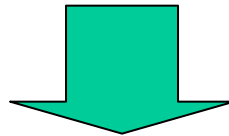
$$\begin{aligned} \delta_j &\equiv \frac{\partial E^n}{\partial a_j} = \sum_k \frac{\partial E^n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\ &= g'(a_j) \sum_k w_{kj} \delta_k \quad (2) \end{aligned}$$



エラー関数 E の微分のまとめ (1)

1st step: 入力 x_n に対して, 順方向に伝播させて
すべての $\{z_i\}, \{y_k\}$ を求める

$$a_j = \sum_i w_{ji} z_i \quad \rightarrow \quad z_j = g(a_j)$$



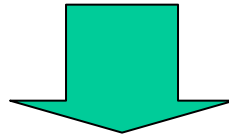
2nd step: 出力ユニットの δ_k を求める

$$\delta_k = g'(a_k) \frac{\partial E^n}{\partial y_k}$$

エラー関数 E の微分のまとめ (2)

3rd step: 出力ユニットの δ_k から逆方向に伝播させて、中間層のユニットの δ_j を求める

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k$$



4th step: エラー関数 E の微分を求める

$$\frac{\partial E^n}{\partial w_{ji}} = \delta_j z_i$$

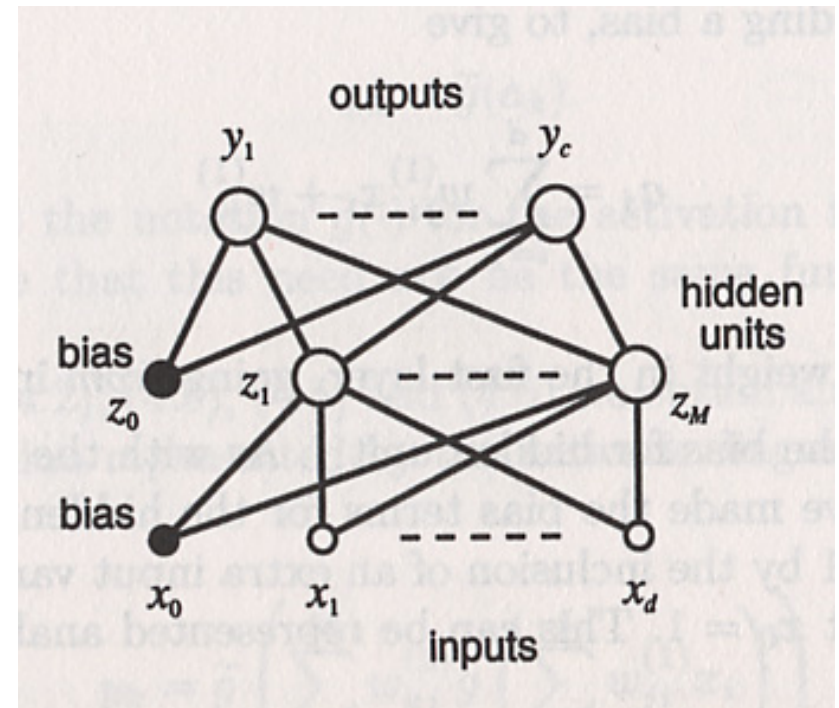
2層パーセプトロンの誤差逆伝播法 (1)

[1] エラー関数は2乗和

$$E^n = \frac{1}{2} \sum_{k=1}^c (y_k - t_k)^2$$

[2] 中間 & 出力ユニットは
sigmoid 関数

$$g(a) = \frac{1}{1 + \exp(-a)}$$



このとき, $g'(a) = g(a)(1 - g(a))$ であり計算が容易!

2層パーセプトロンの誤差逆伝播法 (2)

1st step: 出力ユニットの δ_k を求める

$$\delta_k = y_k(1 - y_k)(y_k - t_k)$$

2nd step: 中間層のユニットの δ_j を求める

$$\delta_j = z_j(1 - z_j) \sum_{k=1}^c w_{kj}^{(2)} \delta_k$$

3rd step: エラー関数 E の微分を求める

$$\frac{\partial E^n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \quad \frac{\partial E^n}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

2層パーセプトロンの誤差逆伝播法 (3)

エラー関数 E の微分 $\rightarrow w$ の修正

最急勾配法: $\Delta w = -\eta \nabla_w E^n$ を用いる

第2層: $\Delta w_{kj}^{(2)} = -\eta \delta_k z_j = -\eta y_k (1 - y_k) (y_k - t_k) z_j$

第1層: $\Delta w_{ji}^{(1)} = -\eta \delta_j x_i = -\eta z_j (1 - z_j) x_i \sum_{k=1}^c w_{kj}^{(2)} \delta_k$

2層パーセプトロンの誤差逆伝播法 (4)

On-line learning

Batch learning

第2層: $\Delta w_{kj}^{(2)} = -\eta \delta_k z_j$ $\Delta w_{kj}^{(2)} = -\eta \sum_n \delta_{n,k} z_{n,j}$

第1層: $\Delta w_{ji}^{(1)} = -\eta \delta_j x_i$ $\Delta w_{ji}^{(1)} = -\eta \sum_n \delta_{n,j} x_{n,i}$

局所解から
逃れる可能性あり

収束が安定化
学習速度は遅い

多層パーセプトロンを動かしてみよう

2層パーセプトロンの誤差逆伝播法による学習を体験してみる。ここでは、2値ビット列の恒等変換 (ex. 1001 → 1001) の学習を取り上げる。学習後に、未知の2値ビット列を入力して、学習の効果を調べてみる。

→	myNN.h	ヘッダファイル
	BPlearn.c	BP学習
	NNrecog.c	テストプログラム
	iden.lrn & iden.nn	学習データ