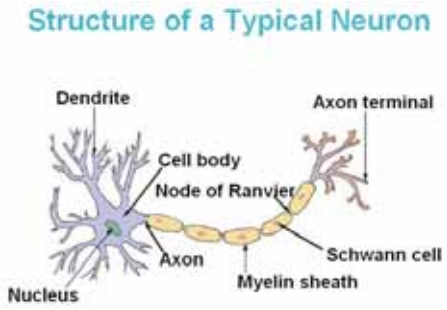
# コネクショニストモデル

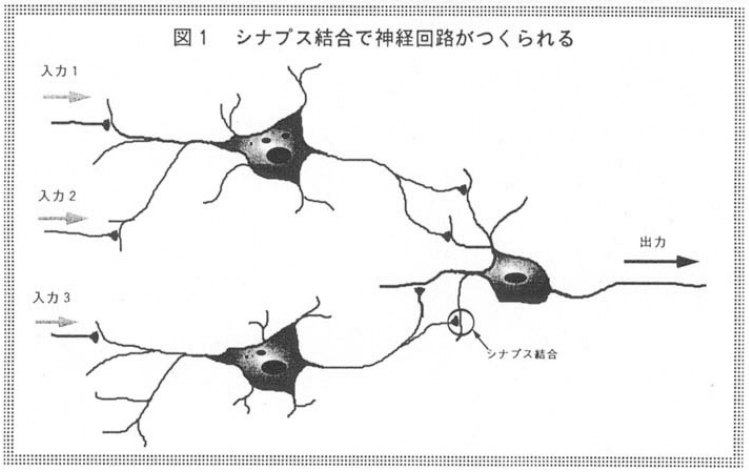
谷田勇樹（M1）

# 神経細胞（ニューロン）とシナプス

- 脳の神経細胞は100億〜1000億個。
- 神経細胞は入力刺激が入ってくると活動電位を発生させ、他の細胞に情報を伝達する。ひとつの神経細胞に複数の神経細胞から入力したり、活動電位がおきる閾値を変化させることにより、情報の修飾が行われる。
- 神経細胞どうしの結合部分（間隙部分）をシナプスという。

# ニューロンとシナプス

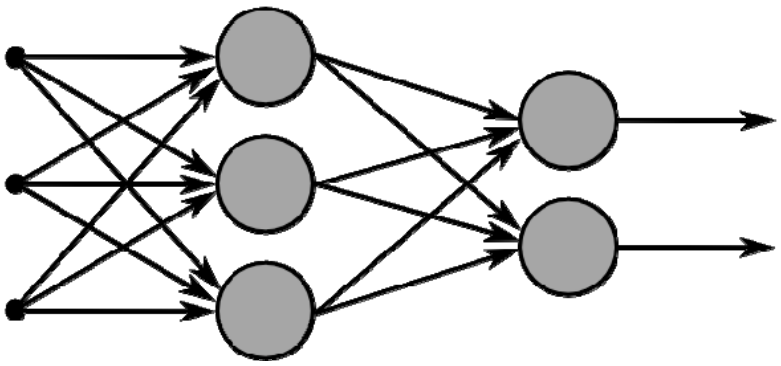



**出典**: **右図**: < http://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Neuron.jpg >
**左図**: < http://www.tmig.or.jp/J_TMIG/kouenkai/koza/51koza_1a.html >

# ノードとコネクション

- コネクショニストモデルでは1つのニューロンを1の「ノード（ユニット）」で表す。
- あるノードは複数のノードから入力（値）を受ける。



**出典**: Wiki < http://upload.wikimedia.org/wikipedia/commons/e/e1/MultiLayerNeuralNetwork.png >

# ノードへの入力値

**あるノードは複数のノードから入力を受ける。**

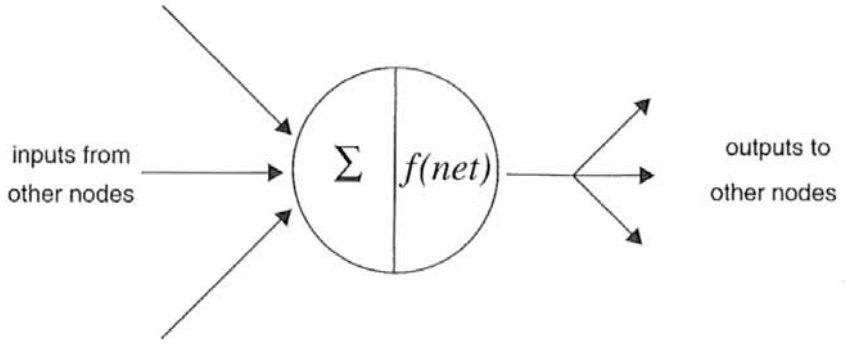
- **あるノード”i”への入力値**(net)

$$net_i = \sum_j w_{ij} a_j$$

- 入力をうけるノードをi
- 出力側のノードをj
- 活性値をa
- ノード間の結合の重みをw (受け取る値にどれだけの情報価を与えるか)
- ノードiの活性値は $a_i$
- ノードjからノードiへの結合は $w_{ij}$

- **その合計値に何らかの関数をかけて、あるノードの活性値を出す。**

node’s activation = f(net)



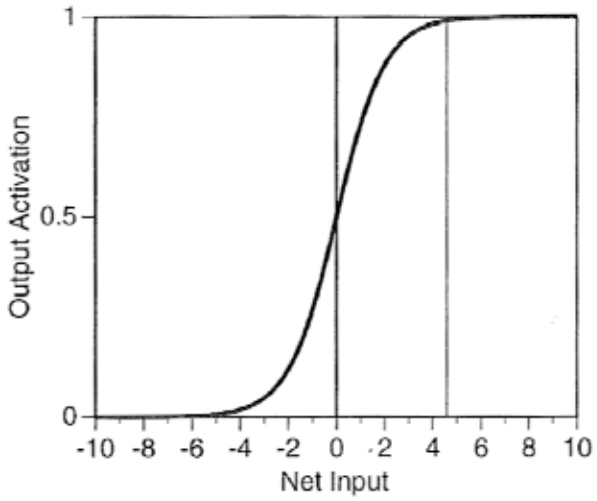
# ロジスティック関数

- ノードはほとんどの場合、単に入力と同期して活性せず、閾値があり、活性値はそれを超えなければ発火しない。多くのニューラルネットワークで、活性関数は入力に対して非線形で、ノードの活性は以下のロジスティック関数（シグモイド関数：ロジスティック関数の解のひとつ）で表される。

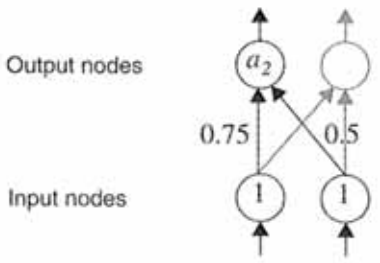
$$a_j = \frac{1}{1 + e^{-net_i}}$$

# シグモイド関数

- 入力が大きければ（正にも負にも）、活性値は0か1（全か無かの法則）。入力が小さいと入力に対して敏感に反応し（グラフの傾きが大きい）、異なる入力に対してより弁別できる。



# 実際に計算してみる




$a_2$への入力は1×0.75＋1×0.5＝1.25

シグモイド関数より1.25のNet InputはOutput Activationが0.777となる。

- このネットワークの活性化は入力側から出力側への一方向である。このようなネットワークを「フィードフォワード・ネットワーク」と呼ぶ。

# アーキテクチャ

ネットワークの構造、結合のパタンはどのようなものかということ。アーキテクチャによってネットワークの振る舞いは変わってくる。

- ユニットレベル：ノードの特性（出力関数、学習ルールなど）
- 局所レベル：feed-forward・再帰型、層数など
- 大域レベル：多数のネットワークの結合・統合の制約

# アーキテクチャの例

(『認知発達と生得性』P43より抜粋)

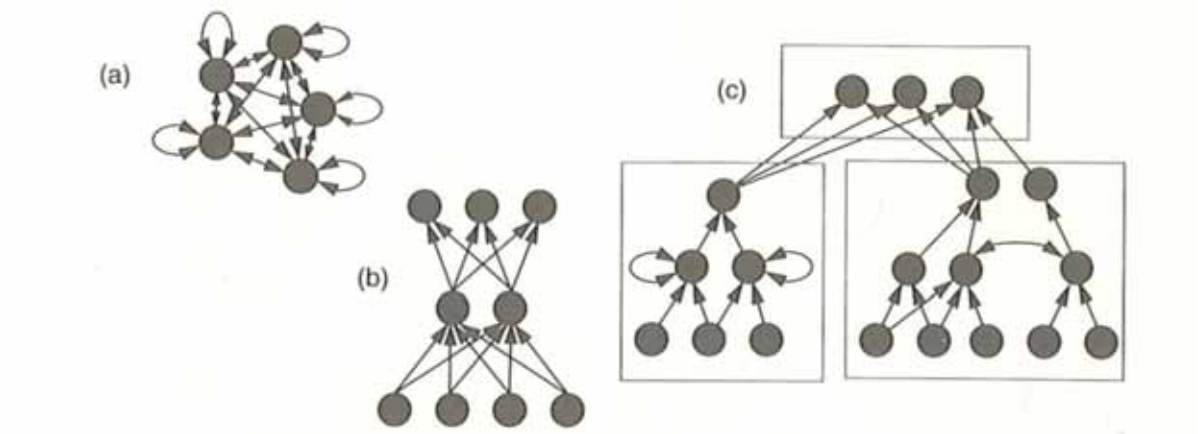



図 2.1　コネクショニストのネットワークのいくつかのタイプ．(a) 完全に双方向性結合のネットワーク，(b) 3層の一方向性ネットワーク，(c) 複数のモジュールからなる複雑なネットワーク．矢印は興奮や抑制が伝わる方向を示している．

# 学習

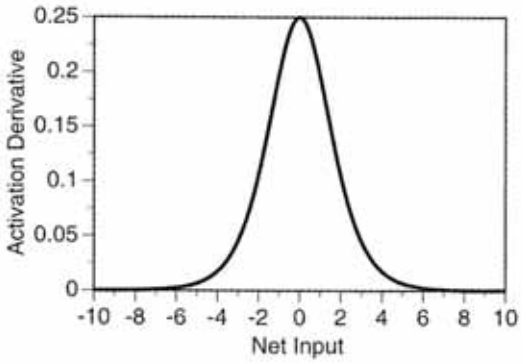
- 学習とは重み値を変化し、出力値を目標値に近づけること。目標値と出力値との誤差を小さくする過程。
- はじめ、ネットワークの結合の重みは通常平均が0、また上限・下限を設定された（多くは±1）ランダムな値。
- 重みは刺激の呈示によって（目標値を出力するように）変化し、ネットワーク自身で学習。
- 学習ルールの代表格が「誤差逆伝搬（バック・プロパゲーション）」

# (誤差の産出バック・プロパゲーション)

$$\delta_{ip} = (t_{ip} - o_{ip}) f'(net_{ip}) = (t_{ip} - o_{ip}) o_{ip}(1 - o_{ip})$$

- 訓練パタンpの目標値：$t_{ip}$
- 出力値：$o_{ip}$
- 誤差：$\delta_{ip}$
- $f'(net_{ip})$ はシグモイド関数の微分係数。
  微分係数は$o_{ip}(1 - o_{ip})$ となる。

右図は入力値に対する微分係数。




誤差は実際の出力値とシグモイド関数の微分係数（傾き）から産出される。ノードへの入力が大きければ傾きは小さくなり、誤差も小さくなる。また結合が弱い場合は入力が小さくなるので、傾きは大きくなり、誤差も大きくなる。

# (重みの変化バック・プロパゲーション)

産出された誤差が小さくなるように結合の重みづけを変える。

$$\Delta w_{ij} = -\eta \frac{\partial E_p}{\partial w_{ij}}$$

（書き換えると$\Delta w_{ij} = \eta \delta_{ip} o_j$）

- 重みの変化量：$\Delta w_{ij}$
- partial derivative（重みの変化に対して誤差がどれだけ変化するか）：$\frac{\partial E_p}{\partial w_{ij}}$
- 学習レイト：η（イータ）

「デルタルール」

- 変化量を小さくするため、η は1.0以下（毎試行後に重みを変化させるときに最適な値を行きすぎてしまわないよう急激な変化をさせないため）
- 変化量は誤差による。
- 送り側のノード からの入力についても考慮しなければならない。なぜなら誤差はあるノードだけではなく、そこに入力を送るノードからどれだけの情報を得ているかに関係するからである。もし送り側のノードの活性が大きく、活性化に大きな影響を与えているならば、もちろん誤差への影響も大きい。

# (重みの変化バック・プロパゲーション)

- ルールに沿って全ノードの誤差が計算され、結合の重みが変化させられる(ただしまだ変化はさせない)。さらに、あるのなら隠れ層にも同じ式を用いて重みの変更をする。
- しかし $\delta_{ip} = (t_{ip} - o_{ip}) f'(net_{ip}) = (t_{ip} - o_{ip}) o_{ip}(1 - o_{ip})$ では教師信号がないために誤差を算出することができない。ただし隠れノードの誤差が大きければそのノードの全体の誤差への寄与は大きいので、活性化しているノードの誤差を合計することで隠れユニットの誤差を計算できる。その式が $\delta_{ip} = f'(net_{ip}) \sum_k \delta_{kp} w_{ki}$
- この手順を繰り返して、ネットワークの出力側から入力側へと計算していく。入力層の手前まで計算したあとで初めて実際に重みが変化される。

# (誤差勾配)

(『認知発達と生得性』P57より抜粋)

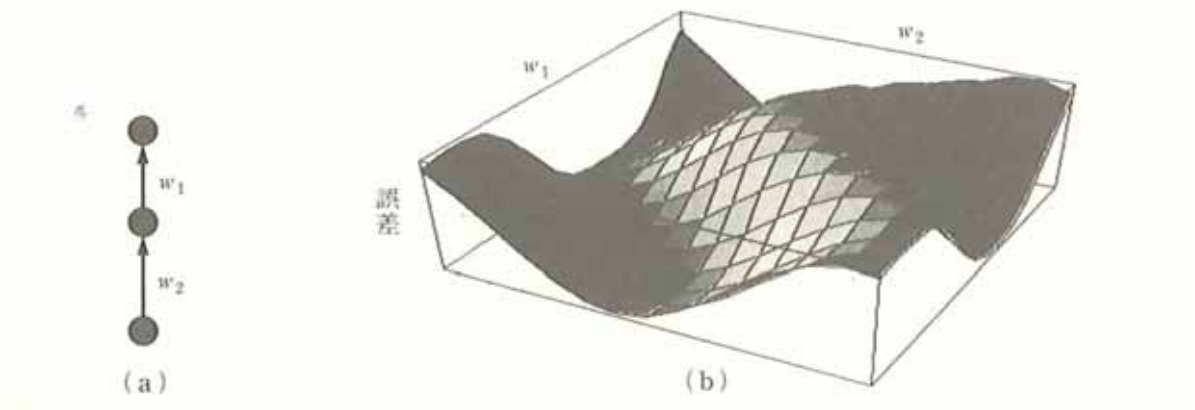
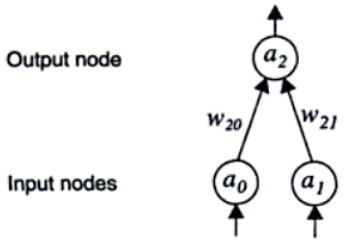



図 2.9 (a) 調節可能な2つの重みをもつ単純なネットワーク．(b) ある仮想的な学習パターンにおける，すべての可能な重みの組合せ（$w_1$, $w_2$）に対する誤差が表すグラフ．曲面のなかの低い部分が誤差の小さい領域に対応する．

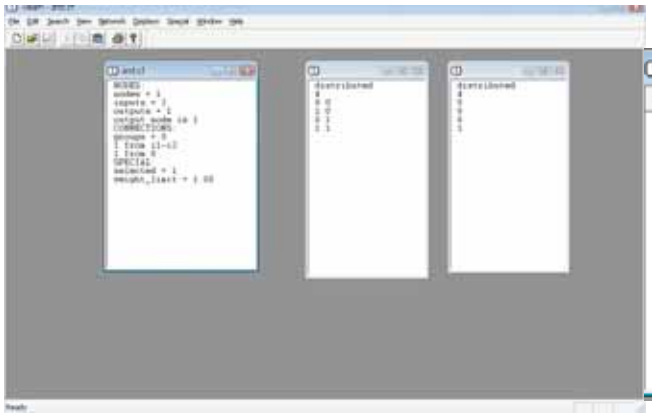
# Tlearnでのシミュレーション

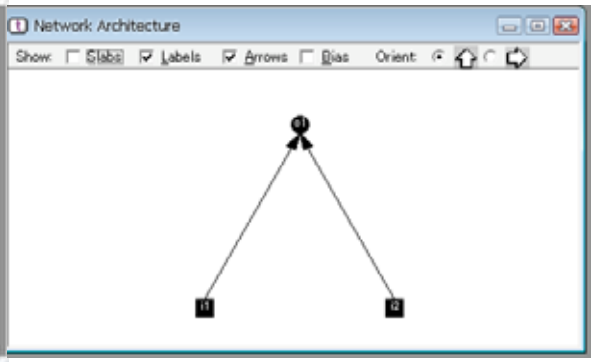
| Input Activation | | Output Activation | | |
|---|---|---|---|---|
| Node0 | Node1 | Node3 | | |
| | | AND | OR | XOR |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

- AND, ORは隠れ層のないネットワークで可能。




# AND関数






and.cf

```
NODES:
nodes = 1
inputs = 2
outputs = 1
output node is 1
CONNECTIONS:
groups = 0
1 from i1-i2
1 from 0
SPECIAL:
selected = 1
weight_limit = 1.00
```

and.data

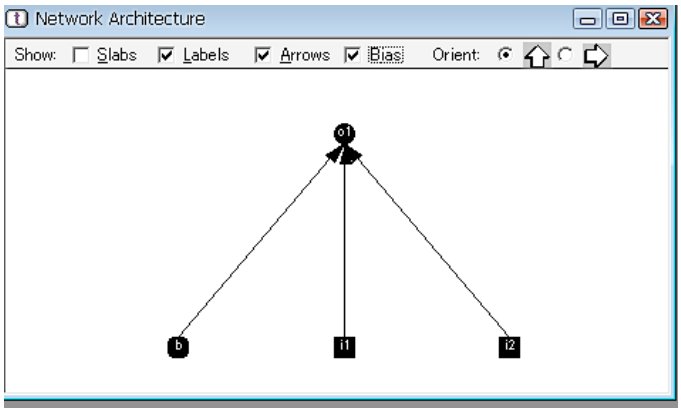
```
distributed
4
0 0
1 0
0 1
1 1
```

and.teach

```
distributed
4
0
0
0
1
```

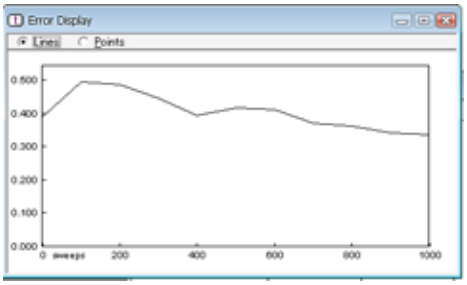
# バイアスノード

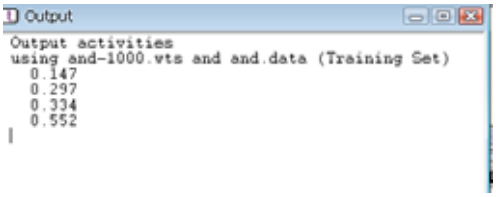
- シグモイド関数において、入力0のときデフォルトでは出力は0.5。
- しかし入力0で0や1を出力したいときもあろう（閾値の変更）。

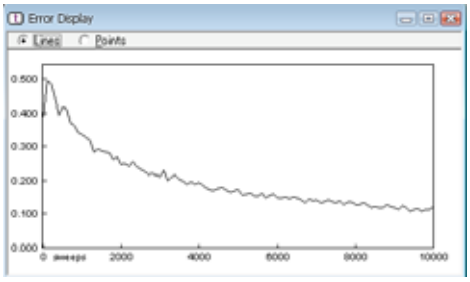


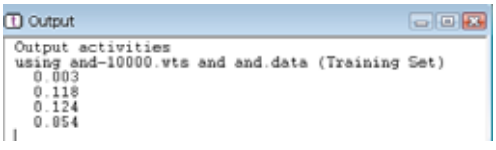

→全ノードにバイアスノードを結合させておけばよい。全ノードはそれぞれのバイアスノードと結合し、常に入力を受けている（この結合の重み値も学習で変えられる）。

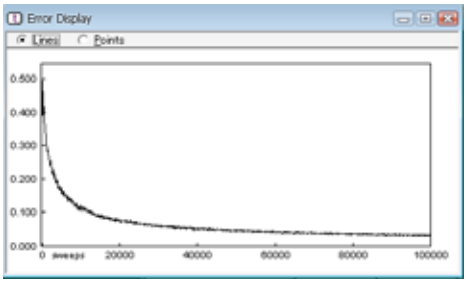
# ANDの結果

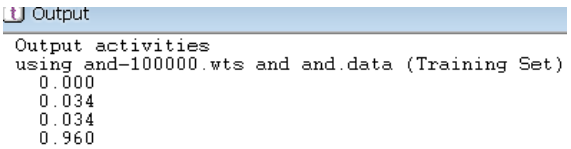


# XOR

(図は『認知発達と生得性』P51より抜粋)

- XORは2層のネットワークでは学習不可能。
- (1,1)の入力で0を出力するためには2つの重み値が小さければよいが、(0,1)や(1,0)のときに1を出力しなければならない。この場合重み値が小さいと出力1を得られない。
- つまり2層のネットワークでは解決不可能な課題。

→隠れ層を加えて3層に。

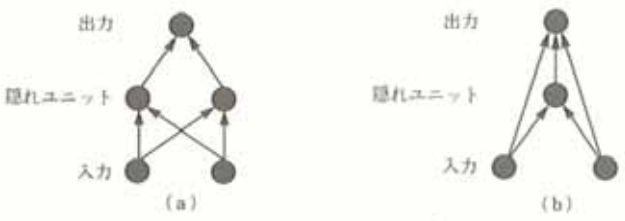
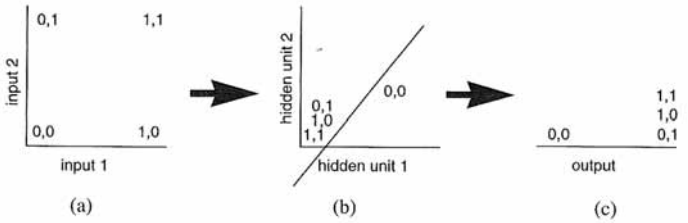



図 2.7 XOR問題を解くことのできる2種類のネットワーク。いずれの場合にも内部に「隠れユニット」が必要である。

# XORが2層で不可能なことのベクトル的理解

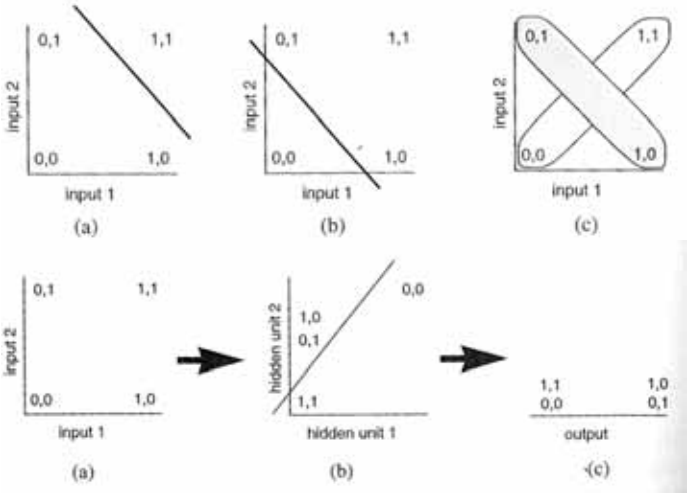
論理関数は線形分離問題。類似性に基づく。

•OR：同じカテゴリーに分類するものが線形分離可能。

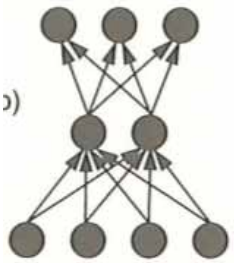



•XOR：同じカテゴリーとして出力しなければならない(1,1)と(0,0)の類似性低い（距離が遠い）。

→この2つの類似性が高まるような空間へと変換するのが隠れ層の役割。



# 隠れ層について



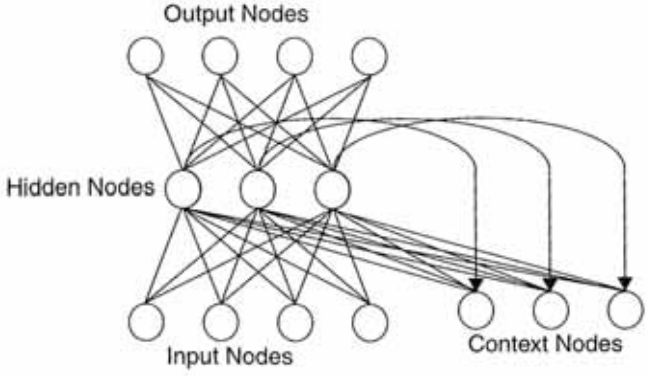
- 隠れ層は「内部表現」と考えられる。
- 例えば「自己連想(auto-association)」のネットワーク：入力と同じ出力を返す。

意味がないようだが、普通、隠れノードは入・出力ノードより少ない。にもかかわらず入力と同じ出力を返すには隠れ層でより少ない次元で情報を保持しなければならない。その情報はもしかしたら特徴によって分けられているかも。

# 単純回帰ネットワーク

(SRN:simple Reccurent Network)

- 前の試行の活動を次の試行に取り込む（時間的な幅をもつ課題、未来の予測）。
- 文章において、次にどんな単語が来るかの予測、など。



# 参考


- Kim Plunkett & Jeffrey L. Elman, 1997, Exercises in rethinking Innateness, MIT Press
- Elman, Bates, Johnson, Karmiloff-Smith, Parisi & Plunkett**著**, **乾敏郎**, **今井むつみ**, **山下博志訳**, 1998, **認知発達と生得性―心はどこから来るのか**, **共立出版**
- **ニューロンの絵**(p3)：< http://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Neuron.jpg >
- **シナプス結合の絵**(p3)：< http://www.tmig.or.jp/J_TMIG/kouenkai/koza/51koza_1a.html >
- **ノードの絵**(p4)：< http://upload.wikimedia.org/wikipedia/commons/e/e1/MultiLayerNeuralNetwork.png >