# 情報意味論 (7) 神経回路網

理工学部管理工学科
櫻井彰人

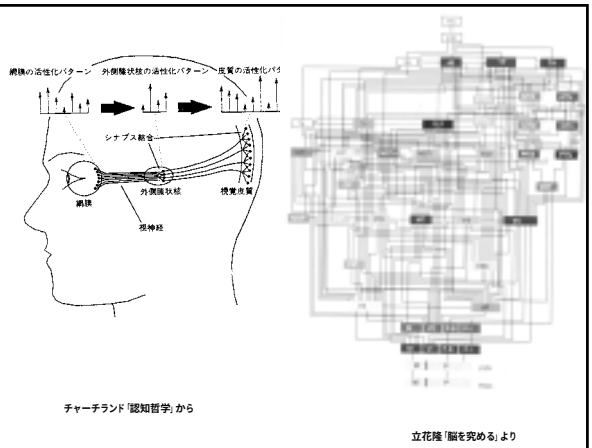
## 神経回路網
## ニューラルネットワーク

- 動物の神経・神経回路にヒントを得る
- 人工の神経素子 (neuron) とそのネットワーク
  - 多くの場合はソフトウェアで実現
- 適応する、学習する
- 連続値が扱える

## 目次



| | 処理素子 | 素子サイズ | 使用エネルギー | 処理速度 | 計算のスタイル | 耐故障性 | 学習 | 知能・意識 |
|---|---|---|---|---|---|---|---|---|
|  | $10^{14}$ シナプス | $10^{-6}$m | 30W | 100Hz | 並列分散 | あり | あり | 通常はあり |
|  | $10^{8}$トランジスタ | $10^{-6}$m | 30W | $10^{9}$Hz | 逐次集中 | なし | 少し | なし (今のところは) |

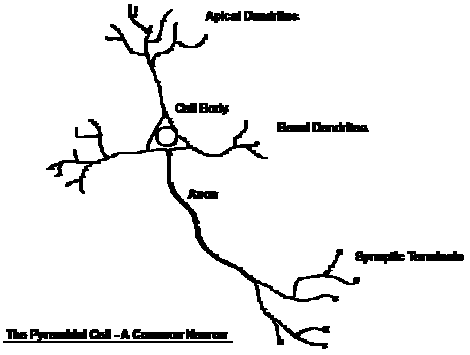



チャーチランド「認知哲学」から

立花隆「脳を究める」より

## 神経のネットワーク

- 人間の脳内の結合については良く分かったいない
- 新皮質内では、緊密に繋がっていると推察されている
- フィードバック結合が必ずあるとも言われている




http://www.emc.maricopa.edu/faculty/farabee/BIOBK/neurdrwing.gif

# 目次

- 概要
  - 脳と神経とそのモデル
  - 人工神経回路網 or ニューラルネットワーク
  - ニューラルネットワークの学習
  - 中間層表現
  - その他の話題
- パーセプトロン
- 多層神経回路網

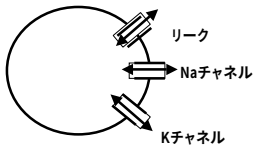
## Hodgkin-Huxley方程式

$$C_m \frac{dV}{dt} = -g_L(V - E_L) - g_{Na} m^3 h(V - E_{Na}) - g_K n^4 (V - E_K) + I$$

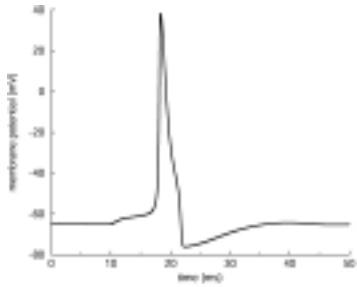
$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h$$

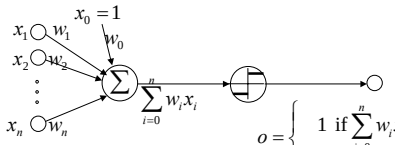
$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n$$



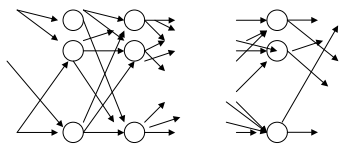

$V$:膜電位、$m$, $h$(Na), $n$(K): チャンネルが開く確率



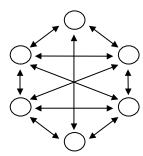
## McCullogh-Pitts モデル(1943)



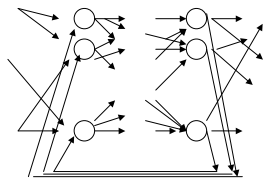

$$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwize} \end{cases}$$



階層型 (フィードフォワード)

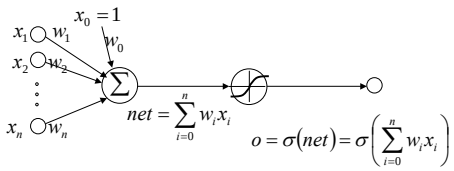


相互結合



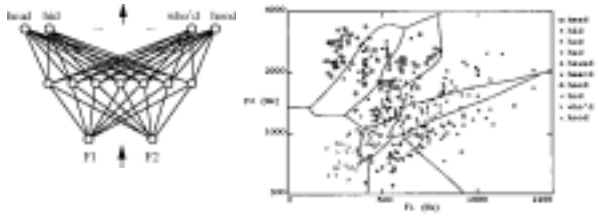
再帰型 (リカレント)

# シグモイド素子




**極めて頻繁に使われる** σ：　$\sigma(x) \equiv \frac{1}{1+e^{-x}}$

# 音声認識



http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html

# Autonomous Land Vehicle in a Neural Net (ALVINN)

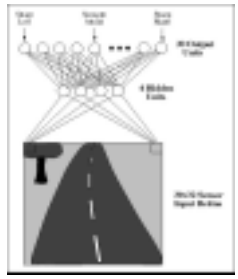
"No Hands Across America"

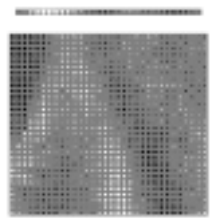
**2850マイルのうち50マイルを除き自動運転により大陸横断.** 70mph **で.**

http://www-2.cs.cmu.edu/afs/cs/user/tjochem/www/nhaa/nhaa_home_page.html

# ALVINN：自動運転

- **高速道路で米国横断**



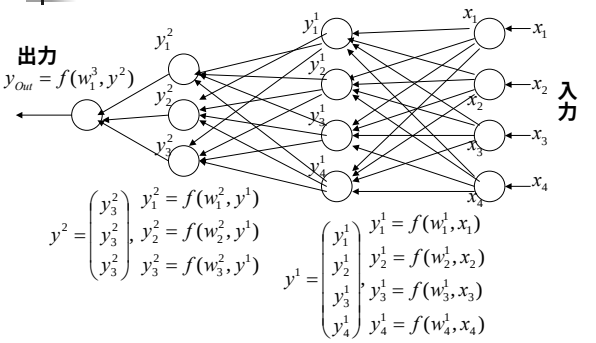


http://www.cs.cmu.edu/afs/cs/project/alv/member/www/projects/ALVINN.html (unavailable)

# 例: *NetTalk*

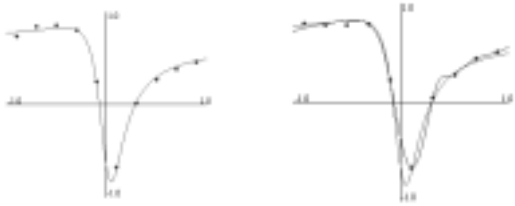
- Sejnowski and Rosenberg, 1986
- Backprop **の初期の大規模アプリケーション**
  - **テキストから音声への変換を目指す**
    - **獲得したモデル**: **文字列から音素と強勢** *stress marks*
    - **出力は音声合成器に送られる**
  - **約1000語の語彙で訓練したところ良好な結果をえた**
- **非常に工夫した**（**入力から出力への**）**符号化**
  - **入力**: 7-**文字のウィンドウ**; **中央にある文字の音素をその周囲の文字を文脈として決定**; **分散** distributed (i.e., sparse) **表現**: 200 bits
  - **出力**: **調音器官への情報** (e.g., "voiced"), **強勢**, **最も近い音素**; **分散表現**
  - 40 **隠れ素子**; 10000 **荷重数**（**全部**）
- **実験結果**
  - **語彙**: 1463**語中** 1024 **語で訓練**
  - 78% **の精度**
- http://www.boltz.cs.cmu.edu/benchmarks/nettalk.html (1999**年ごろまではあった**)

# 基本的な計算



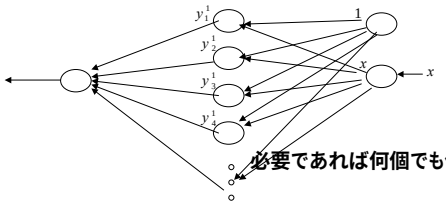
# ニューラルネットの原理

- 一側面：関数近似・回帰である
  - 例：1入力・1出力とする

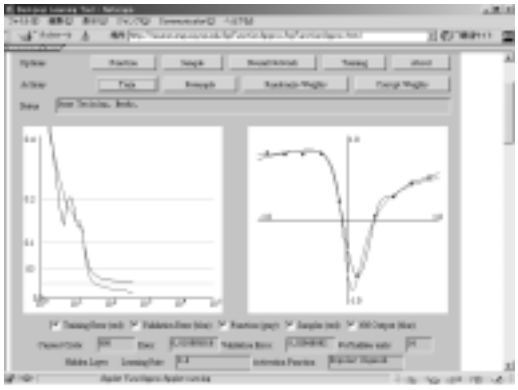


# 近似定理

- ニューラルネットワークの中間素子数を必要なだけ用意できるなら、任意の滑らかな関数を任意の精度で近似することができる

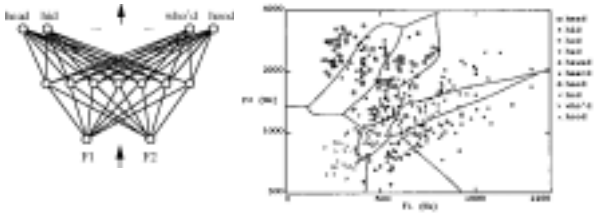



必要であれば何個でも使う



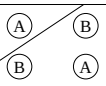
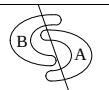
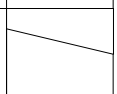
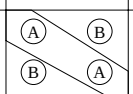
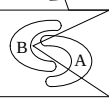
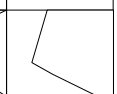
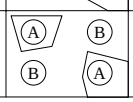
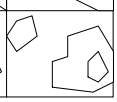
http://neuron.eng.wayne.edu/bpFunctionApprox/bpFunctionApprox.html

# 音声認識（再掲）



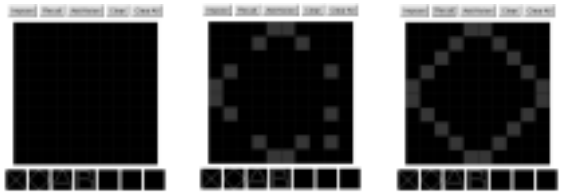
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html

# 領域と境界面

| 構造 | 境界面の形 | 例1<br>対XOR問題 | 例2 | 例3 |
|---|---|---|---|---|
| 中間層なし<br>  | 超平面 |  |  |  |
| 中間層2素子<br>  | 2超平面、それらを滑らかにしたもの |  |  |  |
| 中間層多素子<br>  | 任意（但し、素子数に依存） |  |  |  |

# 連想

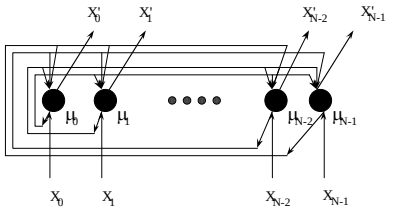
- (A,a), (B,b), (C,c),… というデータを記憶し、(A,?) と聞かれたら ?=a と答える
- 変形した・ノイズの乗った図形から元の図を復元する。



http://www.physics.syr.edu/courses/modules/MM/sim/hopfield.html

## Hopfieldネットワーク

- 相互結合型。通常、時を刻みながら、過去の自分達の値を入力として、次の出力（これが次の入力となる）を決める。




## Kohonenマップ

- 説明省略

http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-sample.html

## 目次

- 概要
  - 脳と神経とそのモデル
  - 人工神経回路網 or ニューラルネットワーク
  - ニューラルネットワークの学習
  - 中間層表現
  - その他の話題
- パーセプトロン
- 多層神経回路網

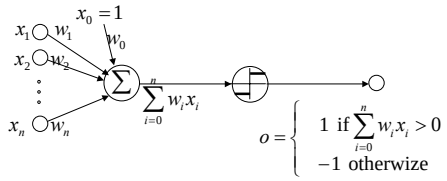
## 生物の学習（脳の中）

- 神経回路が変化している
- 結合が変わることより、結合の強さが変わることが多い
- Hebb則
  - あるシナプスから入力があったとき、神経細胞が発火した ⇒ そのシナプス結合強度上昇
- 実際は、タイミング等を含む複雑な変化であろうと考えられている

## ニューラルネットワークの学習

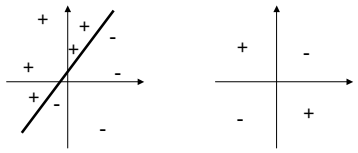
- 教師つき学習：
  - 入力値・出力値の対が（教師データとして）大量に与えられる
  - 入力値が入力されたら、対応する出力値を出力するよう、内部の結合荷重を設定する
- 教師なし学習：（機構はちょっと複雑になる）
  - （ロボットで考える）ある状態のとき、ある行動をとると褒美が与えられる
  - その状態のときはその行動をとりがちになるよう内部のパラメータを修正する

## パーセプトロン



## パーセプトロンの表現能力

- 非常に多くの（意味のある）論理関数（ブール関数）が表現可能
  - AND, OR, NOT, 多数決（一般に *m*-of-*n*）等
  - しかし、XOR は表現できない



## パーセプトロンの学習

- Rosenblatt の学習アルゴリズム
  - 一個の線形閾値素子からなる回路が対象
    **if** $w \cdot x \le 0 \wedge x \in P$ **then** $w \leftarrow w + x$
    **else if** $w \cdot x > 0 \wedge x \in N$ **then** $w \leftarrow w - x$
  - もし、ユークリッド空間 *R* 内の点集合 *P, N* が線形分離可能 *i.e.*, $\exists w\ (\forall x \in P\ w \cdot x > 0 \ /\ \forall x \in N\ w \cdot x \le 0)$ であれば、上記手続きは収束する
  - 特徴：収束が保証されている上に速い

## パーセプトロン学習の欠点

- ノイズに弱い
- 線型分離可能でない時に学習できない
- 多層回路網に拡張できなかった
  - 中間層素子に対する教師信号が特定できない
  - その結果、2ビットの XOR も実現できない
  - パーセプトロン・ブーム終焉の一つの契機
    - 線形閾値素子で代替しようとしていた、計算機用素子が安価で高速になったのも理由

## ノイズ・線型分離不能の対策

- 全ての入出力仕様が満足できなくとも、そこそこに満足できればよい
- 「満足程度」（不満足程度）の評価
  - 誤差関数：目標と現実の差を表現する関数
- 例によって、誤差の自乗和がよく使われる

$$E(W) = \frac{1}{N}\sum_{t=1}^{N}(F(W, x_t) - y_t)^2$$

## 教師付き学習

- 結合荷重の修正方法
  - 学習データを $\{(x_t, y_t) \mid 1 \le t \le N\}$ とする
  - またネットワークの入出力関係を $y = F(W, x)$
  - 誤差関数を設定する。通常は、

$$E(W) = \frac{1}{N}\sum_{t=1}^{N}(F(W, x_t) - y_t)^2$$

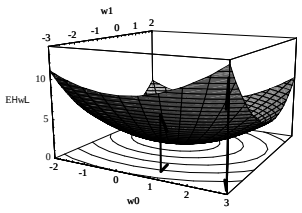
  - この *E* を最小化する *W* を求めればよい

## 誤差最小化の方法

- 微分して0とおいた方程式を解けばよい！本当か？
- 非線形連立方程式になり、到底、解けない
- 反復解法（少しずつ、解を改善していく方法）を考える。すなわち、$E(W_1) > E(W_2) > E(W_3) > \cdots$ となる $W_1, W_2, W_3 \cdots$ を求める方法を考える

## 反復最小化法

- 様々な方法が提案されている。
- 中でも最も単純なものが、最急降下法
  - 最大値を求めるなら、最急上昇法（あまり使わない）。




最急降下方向と等高線とのなす角度に注目！

## 反復最小化方法の数学

- 実際の計算はどうすればよいか？
- 微係数は、最急上昇方向であった！
- そこで、

$$\Delta w_i^j = -\eta \cdot \frac{\partial E}{\partial w_i^j}(W)$$

$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

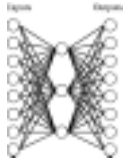
とする。η は学習係数（上手に決めないといけない定数）

## 目次

- 概要
  - 脳と神経とそのモデル
  - 人工神経回路網 or ニューラルネットワーク
  - ニューラルネットワークの学習
  - 中間層表現
- パーセプトロン
- 多層神経回路網

## 中間層での表現

- これは学習できるか？



| Input | | Output |
|---|---|---|
| 10000000 | → | 10000000 |
| 01000000 | → | 01000000 |
| 00100000 | → | 00100000 |
| 00010000 | → | 00010000 |
| 00001000 | → | 00001000 |
| 00000100 | → | 00000100 |
| 00000010 | → | 00000010 |
| 00000001 | → | 00000001 |

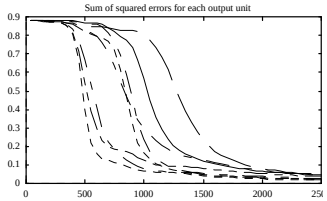
## 中間層での表現（2）

- 学習結果

| Input | | | | | | | | Output |
|---|---|---|---|---|---|---|---|---|
| 10000000 | → | .89 | .04 | .08 | → | 10000000 |
| 01000000 | → | .01 | .11 | .88 | → | 01000000 |
| 00100000 | → | .01 | .97 | .27 | → | 00100000 |
| 00010000 | → | .99 | .97 | .71 | → | 00010000 |
| 00001000 | → | .03 | .05 | .02 | → | 00001000 |
| 00000100 | → | .22 | .99 | .99 | → | 00000100 |
| 00000010 | → | .80 | .01 | .98 | → | 00000010 |
| 00000001 | → | .60 | .94 | .01 | → | 00000001 |

## 隠れ層での表現（3）

- 学習の進行の様子

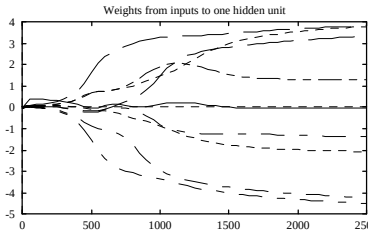


## 隠れ層での表現 (4)

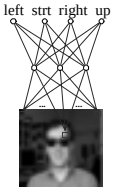
- 学習の進行の様子 (2)

## 隠れ層での表現 (5)

- 学習の進行の様子 (3)




## 顔画像の学習 (再掲)

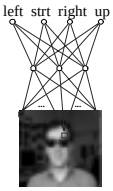



30 x 32

- 顔画像の例

## 顔画像の学習

- 学習後の荷重




30 x 32

## 中間層に発現する表現

- 中間層には、プログラマが意図しなかった内部表現が発生する (ことがある)
- よくよく見ると「意味深い」表現であったりする
- 実は、オンライン逐次学習法を用いるとBayes学習的なことがおこり、「情報の圧縮」すなわち、「意味抽出」が行われうることが示せる

## 他の誤差関数

- 大きな荷重にペナルティを

$$E(\vec{w}) \equiv \sum_{d \in D} \sum_{k \in outputs} \left(t_{k,d} - o_{k,d}\right)^2 + \gamma \sum_{i,j} {w_{j,i}}^2$$

- 関数の傾きも学習対象

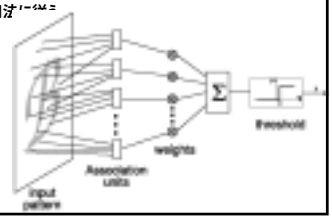
$$E(\vec{w}) \equiv \sum_{d \in D} \sum_{k \in outputs} \left[ \left(t_{k,d} - o_{k,d}\right) + \mu \sum_{j \in inputs} \left( \frac{\partial t_{k,d}}{\partial x_d^j} - \frac{\partial o_{k,d}}{\partial x_d^j} \right)^2 \right]$$

# 目次

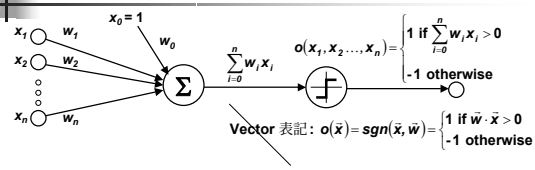
- 概要
  - 脳と神経とそのモデル
  - 人工神経回路網 or ニューラルネットワーク
  - ニューラルネットワークの学習
  - 中間層表現
- パーセプトロン
- 多層神経回路網

# 多義語: パーセプトロン

- パーセプトロン: 同じ言葉で別のものを指している
  - 線型閾値素子: 次のスライド
  - 元祖パーセプトロン: 下記. これが本当！
  - シグモイド素子: 次回
  - シグモイド素子のネットワーク: 多層パーセプトロンと呼ばれる. 次回
  - 線型閾値素子のネットワーク: 多層パーセプトロン. 稀
- 本講義では, 習慣に従い「間違った」用法で
- 元祖パーセプトロン
  - Rosenblatt 1962
  - Minsky and Papert 1969

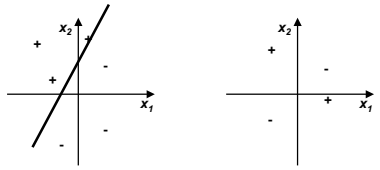



# パーセプトロン Perceptron




- パーセプトロン Perceptron: 単一ニューロンのモデル
  - 別名 線型閾値素子 Linear Threshold Unit (LTU) or Linear Threshold Gate (LTG)
  - 素子への純入力 net input: 線型関数 $net = \sum_{i=0}^{n} w_i x_i$
  - 素子の出力: 純入力に閾値関数 threshold function を施したもの (閾値 threshold $\theta = w_0$)
    - 純入力に施して出力を得る関数を 活性化関数 activation function と呼ぶ
- パーセプトロンネットワーク Perceptron Networks
  - パーセプトロン同士が 荷重つき結合 weighted links $w_i$ によって繋がっている
  - Multi-Layer Perceptron (MLP): 下の方

# パーセプトロンの決定境界




- パーセプトロン: 重要な関数がいくつも簡単に表現できる
  - 論理関数 (McCulloch and Pitts, 1943)
  - e.g., 簡単な荷重で $AND(x_1, x_2)$, $OR(x_1, x_2)$, $NOT(x)$
- 表現できない関数もある
  - e.g., 線型分離可能でないもの
  - 解: パーセプトロンのネットワーク

# パーセプトロン学習アルゴリズム

- 学習規則 ≡ 訓練規則 training Rule
  - 教師付き学習に特有の話ではない
  - 文脈: モデルの更新
- Hebbの学習則 Hebbian Learning Rule (Hebb, 1949)
  - アイデア: もし2個の素子が両方とも active ("firing")であれば, 結合荷重は増加する
  - $w_{ij} = w_{ij} + r\, o_i\, o_j$ , 但し $r$ は学習係数 learning rate で、定数である
  - 神経生理学的に, ほぼ, 指示されている
- パーセプトロン学習アルゴリズム Perceptron Learning Rule (Rosenblatt, 1959)
  - アイデア: 各入力ベクトルに対して出力値が与えられているなら, 荷重を漸進的に更新することにより, 当該出力値が出力できるようになる
  - 2値出力 (Bool値, Boolean-valued) を仮定; 単一パーセプトロン素子
  - $w_i \leftarrow w_i + \Delta w_i$
    $\Delta w_i = r(t - o)x_i$

    但し $t = c(x)$ は目標出力値, $o$ はパーセプトロンの現在の出力値, $r$ は学習係数, 正定数であれば何でも良い. 1でよいので、実は、パーセプトロン学習アルゴリズムでは, $r$ は不要
  - $D$ が線型分離可能 linearly separable であれば, 収束する. $r$ が十分小さいことを条件とする説明もあるがそれは誤り

# パーセプトロン学習アルゴリズム

- 単純な勾配降下 Gradient Descent アルゴリズムである
  - このアイデアは, 適当な表現を用いれば, 概念学習にも記号学習にも適用可能
- アルゴリズム *Train-Perceptron* ($D \equiv \{<x, t(x) \equiv c(x)>\}$)
  - 荷重 $w_i$ をランダム値に初期化する // パーセプトロン時は0に初期化してもよい
  - WHILE 正しい出力をしない事例がある DO
    - FOR それぞれの事例 $x \in D$
      - 現在の出力 $o(x)$ を計算
      - FOR $i = 1$ to $n$
        - $w_i \leftarrow w_i + r(t - o)x_i$ // perceptron learning rule. r is any positive #
- パーセプトロン学習可能性
  - 復習: $h \in H$ のときのみ学習可能 - i.e., 線型分離可能 linearly separable (LS) functions
  - Minsky and Papert (1969) *Perceptrons* : 元祖パーセプトロンの表現・学習の限界を示した
    - 注: 素子一個では parity ($n$-変数 XOR: $x_1 \oplus x_2 \oplus \ldots \oplus x_n$) 関数が表現できない, というのは既知
    - e.g., 画像の symmetry, connectedness は (元祖パーセプトロンで) 表現できない
    - "*Perceptrons*" のせいで ANN 研究が10年近く遅れたといわれもするが, どこまで真実か。

# 線型分離

- **定義**
  - $f(x) = 1$ if $w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq \theta$, 0 otherwise
  - $\theta$: **閾値**
- **線型分離可能性**
  - **注**: $D$ **が線型分離可能だからといって**, **真の概念** $c(x)$ **が線型分離可能とは限らない**
  - **選言** disjunction: $c(x) = x_1' \vee x_2' \vee \ldots \vee x_m'$
  - $m$ of $n$: $c(x)$ = at least 3 of ($x_1'$, $x_2'$, …, $x_m'$) ✓
  - **排他的** exclusive *OR* (*XOR*): $c(x) = x_1 \oplus x_2$ ✗
  - **一般の** DNF: $c(x) = T_1 \vee T_2 \vee \ldots \vee T_m$; $T_i = l_1 \wedge l_2 \wedge \ldots \wedge l_k$ ✗
- **表現の変換**
  - **線型分離可能でない問題を線型分離可能な問題に変換できるか**?
  - **それは意味のあることなのか**? **現実的なのか**?
  - **現実問題の重要な部分を占めるのか**?


✓ **Linearly Separable (LS) Data Set**

# パーセプトロン学習の収束

- **パーセプトロン学習の収束定理**
  - **主張**: **もし訓練データと** consistent **な荷重集合があれば** (i.e., **データが線型分離可能なら**), **パーセプトロン学習アルゴリズムは収束する**
  - **証明**: **探索空間が限界のある順序をなしている** ("**楔の幅**" **が厳密に減少していく**) – **参照** Minsky and Papert, 11.2-11.3
  - **注意** 1: **収束までの平均時間は**?
  - **注意** 2: **もし線型分離可能でなければどうなるのか**?
- **パーセプトロン循環定理**
  - **主張**: **訓練データが線型分離可能でなければ パーセプトロン学習アルゴリズムにより得られる荷重ベクトルは、ある有界集合内に留まる. 荷重が整数ベクトルなら, 有限集合内に留まる.**
  - **証明**: **もし十分に絶対値が大きい荷重ベクトルから始めると**, **絶対値は殆ど大きくなれないことが示せる**; **訓練事例の次元** $n$ **の数学的帰納法による** – Minsky and Papert, 11.10
- **よりロバストに**, **またより表現力を上げるには**?
  - **目的** 1: **もっとも良い近似を発見するアルゴリズムの開発**
  - **目的** 2: **表現の制約を超える新しいアーキテクチャの開発**

# 勾配降下: 原理

- **線型素子に対する勾配降下**
  - **線型素子を考えよう**:

$$o(\vec{x}) = net(\vec{x}) = \sum_{i=0}^{n} w_i x_i$$

  - **目的**: $D$ **に最もよく適合する関数を見出す**
- **近似アルゴリズム**
  - **目的を定量的に**: **訓練データ集合** $D$ **上での誤差を最小に**
  - **誤差関数**: **自乗誤差の和** sum of squared error (SSE)

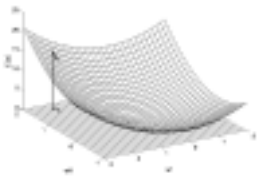
$$E[\vec{w}] = error_D[\vec{w}] = \frac{1}{2}\sum_{x \in D}(t(x) - o(x))^2$$

- **どうやって最小化するか**?
  - **簡単な最適化問題**
  - **荷重–誤差空間で、最も急な降下方向に動けばよい**
    - **計算としては、接平面を求める**
    - i.e. E **を荷重** ($w_i$) **で微分すればよい**

# 勾配降下: Delta/LMS (Widrow-Hoff) 規則の導出

- **定義**: **勾配** gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$



- **勾配降下学習規則**

$$\Delta \vec{w} = -r \nabla E[\vec{w}]$$

$$\Delta w_i = -r \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\left[\frac{1}{2}\sum_{x \in D}(t(x) - o(x))^2\right]$$

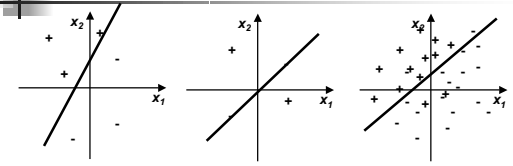
$$= \sum_{x \in D}\left[(t(x) - o(x))\frac{\partial}{\partial w_i}(t(x) - \vec{w} \cdot \vec{x})\right]$$

$$= \sum_{x \in D}\left[(t(x) - o(x))(-x_i)\right]$$

# 勾配降下: Delta/LMS 規則を用いたアルゴリズム

- **アルゴリズム** *Gradient-Descent* ($D$, $r$)
  - **訓練事例は**, **入力ベクトル** $x$ **と出力値** $t(x)$ **の対** $\langle x, t(x)\rangle$ **である**. $r$ **を学習率とする** (e.g., 0.05)
  - **荷重** $w_i$ **を乱数値で初期化する**
  - UNTIL **終了条件が成立** DO
    - $\Delta w_i$ **を** 0 **に初期化**
    - FOR $\langle x, t(x)\rangle$ in $D$ DO
      - **該当素子に事例** $x$ **を入力し**, **出力** $o$ **を計算する**
      - **FOR 荷重** $w_i$ **DO**
        - $\Delta w_i \leftarrow \Delta w_i + r(t - o)x_i$
        - $w_i \leftarrow w_i + \Delta w_i$
  - RETURN **最終** $\vec{w}$
- Delta **規則のメカニズム**
  - **勾配は微係数に基づく**
  - **重要**: **後ほど**, **非線形活性化関数** nonlinear activation functions (**別名** transfer functions).

# 勾配降下: Delta/LMS 規則を用いたアルゴリズム




**Example A** **Example B** **Example C**

- **線型分離可能**: **完全な分類ができる**
  - Example A: **パーセプトロン学習アルゴリズムが収束**
- **線型分離不能**: **近似できるのみ**
  - Example B: **線型分離不能**; delta **規則は収束**, **しかし3個正解よりはよくならない**
  - Example C: **線型分離不能**; delta **規則でよい結果**
- **荷重ベクトル** $W$ = **誤分類した** $x \in D$ **の和**
  - **パーセプトロンアルゴリズム**: $\vec{w}$ **を最小化**
  - Delta **規則**: **最小化** $error$ = **分類境界からの距離** (i.e., **最大化** $\frac{\partial E}{\partial \vec{w}}$ )

# 漸進的 (確率的) 勾配降下

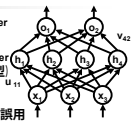
- **バッチ・モードの勾配降下**
  - UNTIL **終了条件が成立** DO
    1. **勾配を求める** $\nabla E_D[\vec{w}]$
    2. $\vec{w} \leftarrow \vec{w} - r\nabla E_D[\vec{w}]$
  - RETURN **最後の** $\vec{w}$
- **漸進的 (オンライン** online**) モードの勾配降下**
  - UNTIL **終了条件が成立** DO
    FOR each $\langle x, t(x)\rangle$ in $D$, DO
    1. **勾配を求める** $\nabla E_d[\vec{w}]$
    2. $\vec{w} \leftarrow \vec{w} - r\nabla E_d[\vec{w}]$
  - RETURN **最後の** $\vec{w}$
- **解釈:オンラインモード=バッチ・モードのエミュレート**
  - $E_D[\vec{w}] \equiv \frac{1}{2}\left[\sum_{x\in D}(t(x)-o(x))^2\right]$, $E_d[\vec{w}] \equiv \frac{1}{2}(t(x)-o(x))^2$
  - **漸進的勾配降下はバッチ勾配降下をいくらでもよく近似することができる,** もし $r$ **を必要に応じて小さくとれば.**
  - **しかし, 実際は, バッチモードより動作が望ましいことが**

# 目次

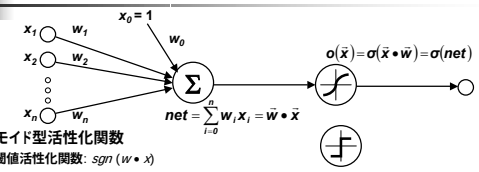
- **概要**
  - **脳と神経とそのモデル**
  - **人工神経回路網** or **ニューラルネットワーク**
  - **ニューラルネットワークの学習**
  - **中間層表現**
- **パーセプトロン**
- **多層神経回路網**

# 非線形素子の多層ネットワーク

- **非線形素子**
  - **復習**: **活性化関数** *sgn* $(w \bullet x)$ **これは勿論非線形なのだが**
  - **非線形活性化** activation **関数**: *sgn* **を一般化** (**注:「入力の荷重和」は線型**)
- **多層ネットワーク** Multi-Layer Networks
  - **ある特別の型**: **多層パーセプトロン** Multi-Layer Perceptrons (MLPs) **実は誤用**
  - **定義**: **多層フィードフォワードネットワーク** multi-layer feedforward network **は一つの入力層** input layer, **一または二以上の隠れ層** hidden layers, **そして一つの出力層** output layer
  - "**層**": **分野で数え方が異なる** (e.g., 1 **隠れ層** = 2-**層** (**計算量研究者**) = 3-**層** (ANN**研究者**))
  - **隠れ層と出力層のみがパーセプトロン素子** (**閾値または非線形活性化関数素子**)
- MLPs: **理論面から**
  - **ネットワーク** (**中間層一層以上**) **どんな関数もいくらでも近似できる** (**素子数には上限ないとして**)
  - 3-**素子** multi-layer ANNs **の学習でさえ** NP-hard (Blum and Rivest, 1992)
- MLPs: **実用面から**
  - **任意の関数に対して有効なネットワーク構造を見出したり***設計*したりするのは困難
  - **構造が分かっているときでも学習は計算**-intensive **である**




# 非線形活性化関数




- **シグモイド型活性化関数**
  - **閾値活性化関数**: *sgn* $(w \bullet x)$
  - **非線形活性化 (別名** transfer**) 関数**: *sgn* **に近く滑らか**
  - $\sigma$ **はよく使われる標準シグモイド** [S**字型**, sigmoid] **関数** $\sigma(net) = \frac{1}{1+e^{-net}}$
    - **値の範囲が** (0,1) **であることに注意**
  - **訓練するために必要な勾配規則が導ける**
    - **シグモイド素子一個の場合だけでなく**
    - **シグモイド素子の多層フィードフォーワードネットワーク** (**誤差逆伝播法使用**)
- **同様によく使われる** Hyperbolic Tangent **活性化関数** (**実質的に同一**)
  $\sigma(net) = \frac{\sinh(net)}{\cosh(net)} = \frac{e^{net}-e^{-net}}{e^{net}+e^{-net}}$

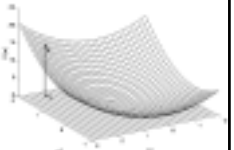
# シグモイド素子の誤差勾配

- **復習**: **誤差関数の勾配** $\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$
- **シグモイド活性化関数の勾配**

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\left[\frac{1}{2}\sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}(t(\vec{x})-o(\vec{x}))^2\right] = \frac{1}{2}\sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}\left[\frac{\partial}{\partial w_i}(t(\vec{x})-o(\vec{x}))^2\right]$$

$$= \frac{1}{2}\sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}\left[2(t(\vec{x})-o(\vec{x}))\frac{\partial}{\partial w_i}(t(\vec{x})-o(\vec{x}))\right] = \sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}\left[(t(\vec{x})-o(\vec{x}))\left(-\frac{\partial o(\vec{x})}{\partial w_i}\right)\right]$$

$$= -\sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}\left[(t(\vec{x})-o(\vec{x}))\frac{\partial o(\vec{x})}{\partial net(\vec{x})}\frac{\partial net(\vec{x})}{\partial w_i}\right]$$



- **しかし**:

$$\frac{\partial o(\vec{x})}{\partial net(\vec{x})} = \frac{\partial \sigma(net(\vec{x}))}{\partial net(\vec{x})} = o(\vec{x})(1-o(\vec{x}))$$

$$\frac{\partial net(\vec{x})}{\partial w_i} = \frac{\partial(\vec{w}\cdot\vec{x})}{\partial w_i} = x_i$$

- **従って**: $\frac{\partial E}{\partial w_i} = -\sum_{\langle \vec{x}, t(\vec{x})\rangle \in D}\left[(t(\vec{x})-o(\vec{x}))\cdot(o(\vec{x})(1-o(\vec{x})))\cdot x_i\right]$

# 誤差逆伝播の計算

**各出力素子につき**

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

$w_{i,j} = w_{i,j} - \eta\delta_j x_i$

**各内部素子につき**

$\delta_h = o_h(1-o_h)\sum_{k\in output(h)} w_{k,h}\delta_k$

$w_{i,j} = w_{i,j} - \eta\delta_j x_i$

$$\frac{\partial E}{\partial w_{i,k}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial z_k}\frac{\partial z_k}{\partial w_{i,k}}$$

$$\frac{\partial E}{\partial o_k} = \frac{\partial}{\partial o_k}\frac{1}{2}\sum_{j\in output}(t_j-o_j)^2 = -(t_k-o_k)$$

$$\frac{\partial o_k}{\partial z_k} = \frac{\partial \sigma(z_k)}{\partial z_k} = o_k(1-o_k)$$

$$\frac{\partial z_k}{\partial w_{j,k}} = \frac{\partial \sum w_{j,k}x_j}{\partial w_{j,k}} = x_j$$

$$\frac{\partial E}{\partial w_{j,h}} = \sum_{k\in output(h)}\frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial o_h}\frac{\partial o_h}{\partial z_h}\frac{\partial z_h}{\partial w_{j,h}}$$

$$\frac{\partial E}{\partial z_k} = \delta_k$$

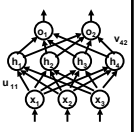
$$\frac{\partial z_k}{\partial o_h} = w_{h,k}$$

$$\frac{\partial o_h}{\partial z_h} = o_h(1-o_h)$$

$$\frac{\partial z_h}{\partial w_{j,h}} = \frac{\partial \sum w_{j,h}x_j}{\partial w_{j,h}} = x_j$$

# 誤差逆伝播アルゴリズム
## Error-Backpropagation Algorithm

- **直感的説明**: ある層での誤差に対する責任を前層に分散する
- **アルゴリズム** *Train-by-Backprop* (*D*, *r*)
  - **訓練事例は**, **入力ベクトル** *x* **と出力値** *t*(*x*) **の対** <*x*, *t*(*x*)> **である**. *r* **を学習率とする** (e.g., 0.05)
  - **荷重** $w_i$ **を乱数値で初期化する** (**絶対値でみて**) **小さい値の方がよいという意見もあるが未決着**)
  - UNTIL **終了条件成立** DO
    - FOR each <*x*, *t*(*x*)> in *D*, DO
      - **該当端子に事例** *x* **を入力し**, **出力** $o(x) = \sigma(net(x))$ **を計算する**
      - FOR **それぞれの出力端子** *k* DO **出力層**
        $$\delta_k = o_k(x)(1-o_k(x))(t_k(x)-o_k(x))$$
      - FOR **それぞれの隠れ端子** *j* DO **隠れ層**
        $$\delta_j = h_j(x)(1-h_j(x)) \sum_{k \in outputs} v_{jk}\delta_j$$ **入力層**
      - **荷重の更新**: **それぞれの** $w = u_{ij}$ ($a = h_j$) **または** $w = v_{jk}$ ($a = o_k$)
        $$w_{start\text{-}layer,\ end\text{-}layer} \leftarrow w_{start\text{-}layer,\ end\text{-}layer} + \Delta w_{start\text{-}layer,\ end\text{-}layer}$$
        $$\Delta w_{start\text{-}layer,\ end\text{-}layer} \leftarrow r\,\delta_{end\text{-}layer}\,a_{end\text{-}layer}$$
    - RETURN **最後の**I *u*, *v*




# 誤差逆伝播法と局所解

- Backprop (BP) **における勾配降下**
  - **ネットワーク全体の荷重ベクトルに対する操作**
  - **任意の有向グラフに一般化容易**
  - **性質**: **フィードフォワード** ANNs **への** backprop **は**, **誤差の局所的** (**大域的とは限らない**) **最小解を見出す** (**バッチモードの時**)
- Backprop: **実際面から**
  - **局所最適化で十分うまくいく** (**必要なら複数回実行**)
  - **慣性項** momentum **を用いることもある**. $\alpha$ **は慣性係数で** 1 **に近い定数** (e.g. 0.95 )
    $$\Delta w_{start\text{-}layer,\ end\text{-}layer}(n) = r\,\delta_{end\text{-}layer}\,a_{end\text{-}layer} + \alpha\Delta w_{start\text{-}layer,\ end\text{-}layer}(n-1)$$
  - **学習事例に対する誤差最小化 – 未知事例に対して汎化するか**?
  - **学習はしばしば遅い**: *D* **上の数千回の繰り返し** (**一回のスキャンを** epoch **という**)
  - **予測** (**訓練後にネットワークを動作させる**) **は非常に速い**
    - **分類**
    - **制御** (**実時間制御に利用可能**)

# Feedforward ANNs: 表現力とバイアス

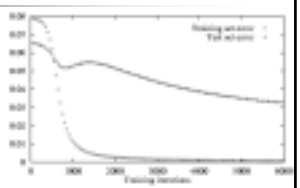
- **表現力**
  - **隠れ層1の**feedforward ANN
    - **任意の** Boolean function **が実現できる**("できる" **ことは自明**. AND-OR **ネットワークを真似**)
    - **任意の 有界連続関数** bounded continuous function (**任意精度で近似**) [Funahashi, 1989; Cybenko, 1989; Hornik *et al*, 1989]
  - **シグモイド関数** (**でなくともよい**) : **基底関数** basis functions; (**ほぼ**) **局所的な和で関数近似**
  - ANNs **が近似容易な関数**: Network Efficiently Representable Functions (NERFs) - **特徴づけはできていない** [Russell and Norvig, 1995]
- ANNs **の帰納バイアス**
  - *n*-**次元ユークリッド空間** (**結合荷重の空間** weight space)
  - **連続関数** (**荷重パラメータに関して連続**)
  - **選択バイアス**: **訓練事例の** "**滑らかな内挿**"
  - **よくは分かっていない**

# 誤差逆伝播法の収束

- **大域的な最適解への収束は保証されない**
  - **比較**: **パーセプトロンの収束** (**最適な** $h \in H$ **に**, **但し** $h \in H$ **なる条件下**; i.e., **線型分離可能**)
  - **ある局所最適解** (**まあ大域的最適解ではなかろう**) **へ勾配降下していく**
  - backprop (BP) **に対する改善**(**かもしれない**)
    - **慣性項** ( **荷重更新規則を多少変更**): **浅い局所最小解はスキップするかも**
    - **確率的勾配効果** stochastic gradient descent : **局所解に捕まる確率が低下**
    - **複数個のネットを異なる荷重値で初期化**; **うまい混合解** mixture **を見つける**
  - **フィードフォワードネットワークの改善**
    - ANNs **のベイズ学習** Bayesian learning (e.g., simulated annealing) – **のちほど**
    - **複数のネットワークを対象とする他の大域最適化法**; **原理的にうまい方法はない**
- **収束過程**
  - 0**に近い初期値**, i.e., **線型に近いネットワークから開始**, **徐々に非線形ネットワークへ**: **未解明**
- **プラトーと収束速度**
  - **プラトーの解消**: **自然勾配法** natural gradient [Amari, 1998]

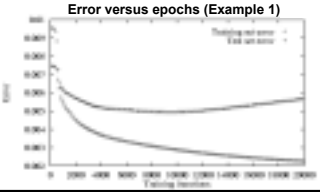
# ANNs の過学習

- **復習**: **過学習の定義**
  - *h*′ **は** *h* **と比較して**, worse on $D_{train}$, better
- **過学習**: **ある型**
  - **繰り返し過ぎ**
  - **回避**: **停止条件**
    (cross-validation: holdout, *k*-fold)
  - **回避策**: weight decay



**Error versus epochs (Example 2)**

**Error versus epochs (Example 1)**



# ANNs の過学習

- **過学習の考えられる他の原因**
  - **予め設定する隠れ素子数の個数**
  - **少なすぎると**, **十分に学習できない** ("underfitting")
    - **成長不足**
    - **連想**: **連立方程式で**、**式** (NNモデル) **の個数** (**自由度**) **より変数** (**真の概念**) **の個数が多い**
  - **多すぎると過学習**
    - **枝刈りされていない**
    - **連想**: 2**次多項式をより高次の多項式で近似する**
- **解**
  - **予防**: **属性部分集合選択** attribute subset selection (pre-filter **または** wrapper)
  - **回避**
    - cross-validation (CV)
    - Weight decay: **エポックごとに荷重を一定値で** (**絶対値を**) **減少させる**
  - **発見/回復**: random restarts: **初期値をランダムにかえて**, **荷重や素子の** addition and deletion

# 別の誤差関数

- **大きい荷重に罰金を (罰金係数** Penalty Factor $w_p$)

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{(\vec{x},\vec{t}(\vec{x})) \in D} \sum_{k \in outputs} \left[ \left( t_k(\vec{x}) - o_k(\vec{x}) \right)^2 + w_p \sum_{start-layer,end-layer} w^2_{start-layer,end-layer} \right]$$

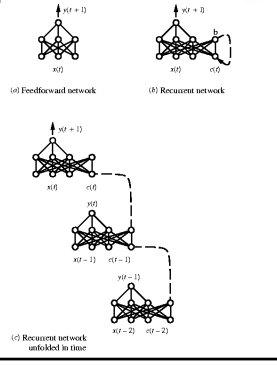
- **目標値だけでなく、傾きも使って訓練**

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{(\vec{x},\vec{t}(\vec{x})) \in D} \sum_{k \in outputs} \left[ \left( t_k(\vec{x}) - o_k(\vec{x}) \right)^2 + w_s \sum_{i \in inputs} \left( \frac{\partial t_k(\vec{x})}{\partial x_i} - \frac{\partial o_k(\vec{x})}{\partial x_i} \right)^2 \right]$$

- **複数の荷重をまとめる**
  - e.g., **音素認識** *Connectionist Speech Recognition* [Bourlard and Morgan, 1994]

# 再帰型ネットワーク

- ANNs **で時系列を表現する**
  - Feedforward ANN: $y(t+1) = net\,(x(t))$
  - **時間に関わる関係を捉える必要あり**
- **解へのアプローチ**
  - **有向サイクル**
  - **フィードバック**
    - **出力層から入力層へ** [Jordan]
    - **隠れ層から入力層へ** [Elman]
    - **入力層から入力層へ**
  - **時間的に離れたデータ間の関係を捉える**
    - $x(t' \le t)$ **と** $y(t+1)$ **の間**
    - $y(t' \le t)$ **と** $y(t+1)$ **の間**
  - **再帰型** ANNs **での学習**
    - Elman, 1990; Jordan, 1987
    - Principe and deVries, 1992
    - Mozer, 1994; Hsu and Ray, 1998




# 新しいニューロンモデル

- **状態をもったニューロン**
  - Neuroids [Valiant, 1994]
  - **それぞれの基本素子が状態をもつ**
  - **それぞれの更新規則は異なってもよい (または 状態に基づく異なった計算)**
  - **適応的なネットワークモデル**
    - **ランダムグラフの構造**
    - **基本素子は学習過程の一部として意味も受取る**
- **パルス・コーディング**
  - **スパイク・ニューロン** spiking neurons [Maass and Schmitt, 1997]
  - **活動度が出力の表現ではない**
  - **発火の列間の相のずれが意味をもつ**
    - **古い時間コーディング** temporal coding **では** rate coding **が用いられ、それは活動度で表現可能**
- **新しい更新規則**
  - **非加算的更新** [Stein and Meredith, 1993; Seguin, 1998]
  - **スパイク・ニューロン・モデル** spiking neuron model

# ANN **の課題と展望**

- **ハイブリッド・アプローチ**
  - **知識** knowledge **や 分析的学習** analytical learning **を** ANNs **に組み込む**
    - Knowledge-based neural networks [Flann and Dietterich, 1989]
    - Explanation-based neural networks [Towell *et al*, 1990; Thrun, 1996]
  - **不確実推論** uncertain reasoning **とANN 学習・推論との結合**
    - **確率的** ANNs
    - **ベイジアンネット** [Pearl, 1988; Heckerman, 1996; Hinton *et al*, 1997] – **のちほど**
- ANNs **の大域的最適化**
  - Markov chain Monte Carlo (MCMC) [Neal, 1996] - e.g., simulated annealing
  - **遺伝的アルゴリズムとの組合わせ** – **のちほど** (**できるか**?)
- ANN **学習結果の解釈**
  - ANNs **からの知識獲得**
    - **規則抽出** rule extraction
    - **決定境界曲面の抽出**
  - **決定支援** decision support **および** KDD **応用** [Fayyad *et al*, 1996]
- **他にもたくさん、、、**