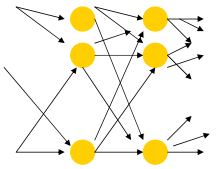
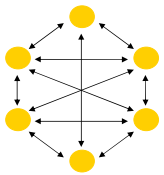
情報意味論 (6) 神経回路網

理工学部管理工学科
櫻井彰人

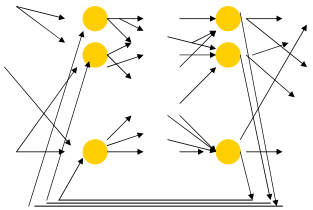
	処理素子	素子サイズ	使用エネルギー	処理速度	計算のスタイル	耐故障性	学習	知能・意識
	10^{14} シナプス	10^{-6}m	30W	100Hz	並列分散	あり	あり	通常はあり
	10^{8}トランジスタ	10^{-6}m	30W	10^{9}Hz	逐次集中	なし	少し	なし（今のところは）



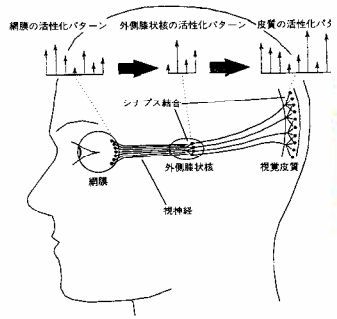
階層型（フィードフォワード）



相互結合



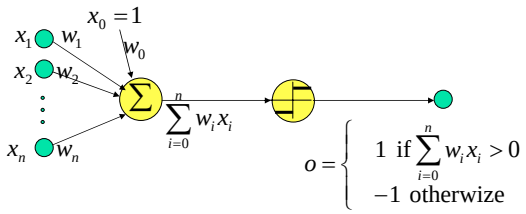
再帰型（リカレント）

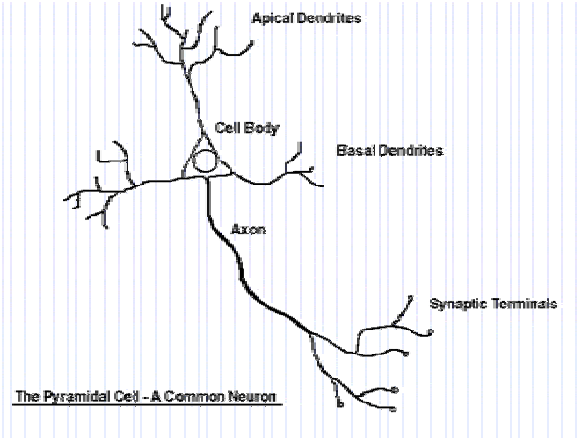


チャーチランド「認知哲学」から

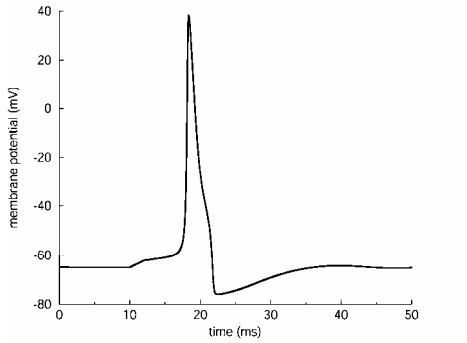
立花隆「脳を究める」より

McCullogh-Pitts モデル(1943)





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



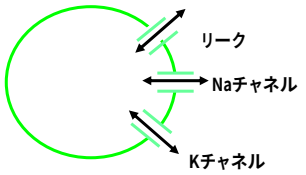
Hodgkin-Huxley方程式

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

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

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

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



V:膜電位、m, h(Na), n(K): チャンネルが開く確率

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

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 $\{(x_t,y_t)|1\le t\le N\}$ 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、

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

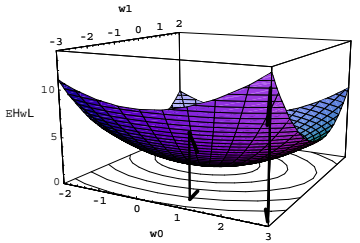
  - この*E*を最小化する*W*を求めればよい

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、

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

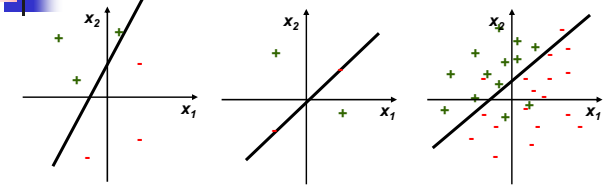
$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

とする。η は学習係数（上手に決めないといけない定数）

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 <*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
  - UNTIL 終了条件が成立 DO
    - Δw_i を 0 に初期化
    - FOR <*x*, *t*(*x*)> 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 *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



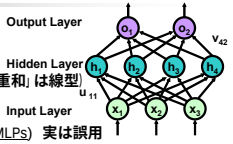
- 線型分離可能: 完全な分類ができる
  - Example A: パーセプトロン学習アルゴリズムが収束
- 線型分離不能: 近似できるのみ
  - Example B: 線型分離不能; delta 規則は収束, しかし3個正解よりはよくならない
  - Example C: 線型分離不能; delta 規則でよい結果
- 荷重ベクトル w = 誤分類した $x \in D$ の和
  - パーセプトロンアルゴリズム: w を最小化
  - Delta 規則: 最小化 *error* ≡ 分類境界からの距離 (i.e., 最大化 $\frac{\partial E}{\partial \vec{w}}$)

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 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 <x, $t(x)$> 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 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

非線形素子の多層ネットワーク

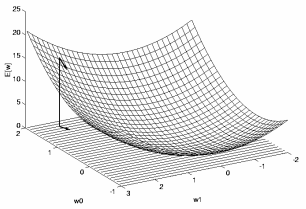
- 非線形素子
  - 復習: 活性化関数 *sgn* ($w \bullet x$) これは勿論非線形なのだが
  - 非線形活性化 activation 関数: *sgn* を一般化 (注:「入力の荷重和」は線型)
- 多層ネットワーク Multi-Layer Networks
  - ある特別の型: 多層パーセプトロン Multi-Layer Perceptrons (MLPs) 実は誤用
  - 定義: 多層フィードフォワードネットワーク multi-layer feedforward network は一つの入力層 input layer, 一または二以上の隠れ層 hidden layers, 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



勾配降下: 線形素子の場合

- 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

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

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

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

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

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

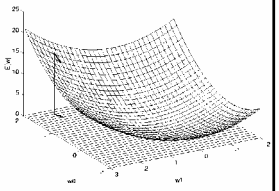
シグモイド素子の場合

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

誤差逆伝播の計算

各出力素子につき

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

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

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

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

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

各内部素子につき

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

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

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

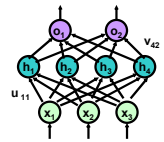
$$\frac{\partial z_k}{\partial w_{j,k}} = \frac{\partial \sum w_{j,k}x_j}{\partial w_{j,k}} = x_j \qquad \frac{\partial z_h}{\partial w_{j,h}} = \frac{\partial \sum w_{j,h}x_j}{\partial w_{j,h}} = x_j$$

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
      - 該当素子に事例 *x* を入力し, 出力 *o*(*x*) = σ(*net*(*x*)) を計算する
      - FOR それぞれの出力素子 *k* DO $\delta_k = o_k(x)(1 - o_k(x))(t_k(x) - o_k(x))$
      - FOR それぞれの隠れ素子 *j* DO $\delta_j = h_j(x)(1 - h_j(x))\sum_{k=outputs} v_{jk}\delta_j$
      - 荷重の更新: それぞれの $w = u_{i,j}$ ($a = h_j$) または $w = v_{j,k}$ ($a = o_k$)

$$w_{start\text{-}layer,\ end\text{-}layer} \leftarrow w_{start\text{-}layer,\ end\text{-}layer} + \Delta w_{start\text{-}layer,\ end\text{-}layer}$$

$$\Delta w_{start\text{-}layer,\ end\text{-}layer} \leftarrow r\,\delta_{end\text{-}layer}\,a_{end\text{-}layer}$$



  - RETURN 最後の 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. α 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

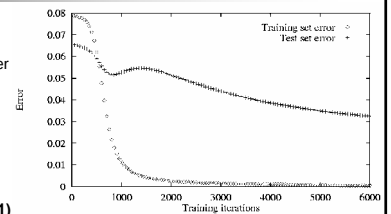
- **表現力**
  - **隠れ層1の**feedforward ANN
    - **任意の** Boolean function **が実現できる**(“**できる**” **ことは自明**. AND-OR **ネットワークを真似**)
    - **任意の 有界連続関数** bounded continuous function **(任意精度で近似)** [Funahashi, 1989; Cybenko, 1989; Hornik *et al*, 1989]
  - **シグモイド関数** (**でなくともよい**) : **基底関数** basis functions; (**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]

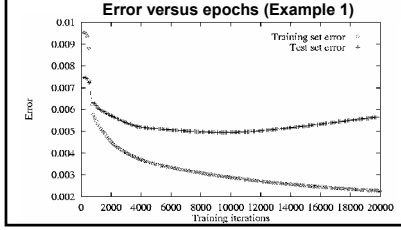
ANNs の過学習

- **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

他の誤差関数

- 大きな荷重にペナルティを

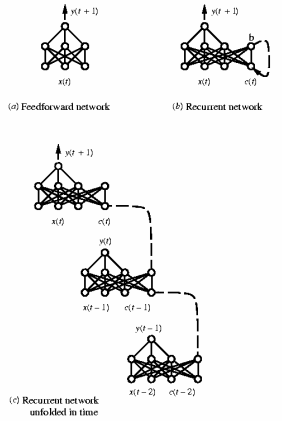
$$E(\vec{w}) \equiv \sum_{d \in D} \sum_{k \in outputs} \left(t_{k,d} - o_{k,d}\right)^2 + \gamma \sum_{i,j} w_{j,i}^{\ 2}$$

- 関数の傾きも学習対象

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

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

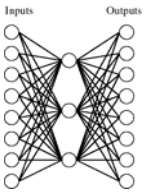


新しいニューロンモデル

- 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

中間層での表現

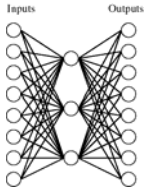
- これは学習できるか？



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)

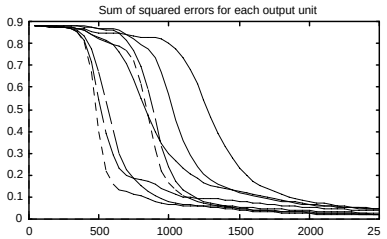
- 学習結果



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

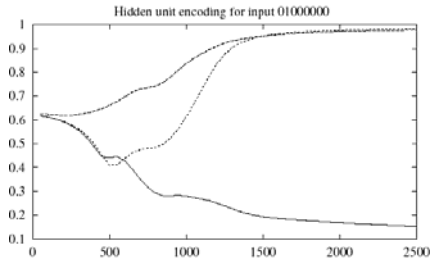
隠れ層での表現 (3)

- 学習の進行の様子



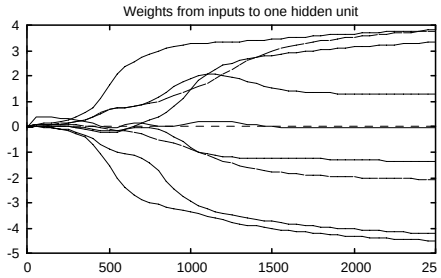
隠れ層での表現 (4)

- 学習の進行の様子 (2)

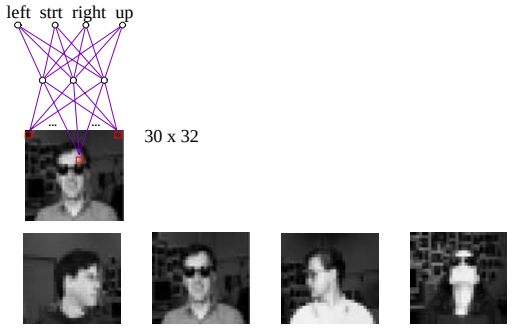


隠れ層での表現 (5)

- 学習の進行の様子 (3)



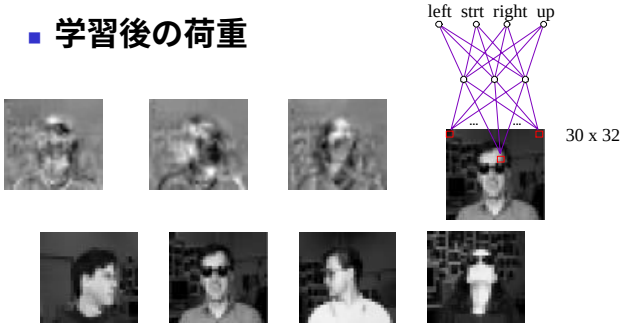
顔画像の学習 (再掲)



- 顔画像の例

顔画像の学習

- 学習後の荷重



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

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、、、