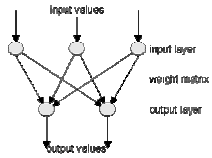
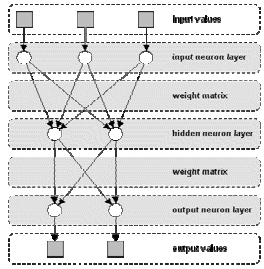
情報意味論
NNとベイズ学習の復習

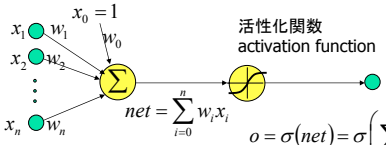
慶應義塾大学理工学部
櫻井 彰人

階層型(多層パーセプトロン)





シグモイド素子



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

または : $\tanh(x)$ or $\tanh\left(\frac{x}{2}\right) \equiv 2\sigma(x) - 1 = \frac{1-e^{-x}}{1+e^{-x}}$

まとめ1: 教師付き学習

- 結合荷重の求め方
  - 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、

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

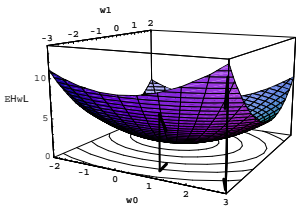
  - この*E*を最小化する*W*を求めればよい

まとめ2: 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？
  - まずは、微分できないことには話しにならん
  - パーセプトロンではだめ。
- 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

まとめ3: 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！

まとめ4: 反復最小化方法の計算式

- 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、

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

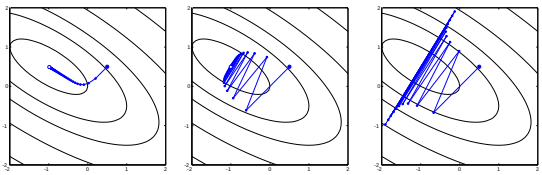
$$w_i^{j,new} = w_i^j + \Delta w_i^j$$

とする。η は学習係数（上手に決めないといけない定数）

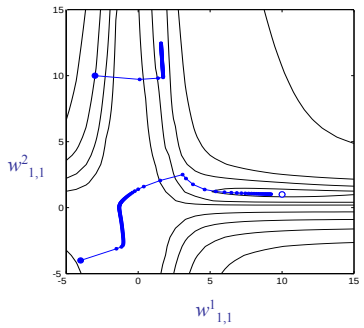
誤差逆伝播法の収束

- 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) へ近づいて行く
  - 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
      - $\Delta W^{new} \leftarrow \Delta W + \alpha \Delta W^{new}$; 加速係数 α は1より小さい定数
    - 確率的最急降下 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; うまく混合する
  - 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)
- 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]

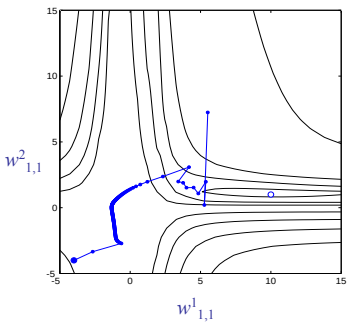
学習パラメータによる違い



収束過程の例



学習係数が大きすぎる場合



バッチモードとオンラインモード

- バッチモード
  - 全学習データに対して誤差項(E)を計算し、それを最小化する方向を求める

$$E(W) = \frac{1}{N}\sum_{i=1}^{N} E_i(W) = \frac{1}{N}\sum_{i=1}^{N} (F(W, x_i) - y_i)^2$$

  - 勿論、実際の計算は、学習データ一個ごとに、Eiの傾きを求め、それを総和する。
- オンラインモード
  - 上記Eiに対して、荷重の変更を実行してしまう
- 比較
  - バッチモードが正統、オンラインモードは非正統にみえる。
    - なお、最小点付近にいるなら、両者はほぼ同等。
  - 実データによる実験では、オンラインモードの方がよい。
  - 一般にはオンラインモードの方が、何らかの意味で、探索にランダム性が入り、局所最適解に入ることが少なくなる。
    - なお、アルゴリズムによってはオンラインモードが考えられないものもある
  - パーセプトロン学習アルゴリズムはオンラインモードしかない。

BP法で満足か？

- とんでもない！
- "最適化法" でできることがわかってしまえば、最急降下法よりよい（よさそうな）ものは、いくらでもある。
- 様々な方法が試みられた
- それなりにうまくはいくのだが、目を見張るほどではない
  - 特に問題なのは計算時間
    - 高速な手法は、|W|*|W|（|W|は荷重の個数）の行列の逆行列の計算が必要とするから
    - 逆行列を、荷重の逐次更新とともに、逐次近似する方法が用いられる
  - それに見合うだけの、成功率と局所解回避率が得られない
  - Neural Networks は単純な形のようだが、結構性質が悪い。特に「特異点」があって、収束を遅くしたり、行列が特異になったりする
- 私が調べ・試みた中で最良のものは、Levenberg-Marquardt 法

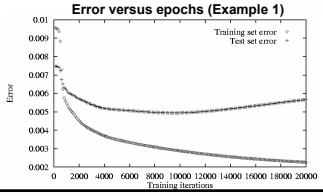
Timothy Masters, Advanced Algorithms for Neural Networks: A C++ Sourcebook, John Wiley & Sons (1995).

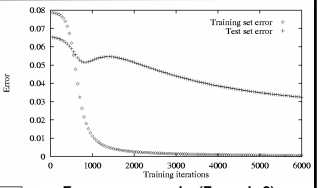
Feedforward ANNs: 表現力とバイアス

- 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

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)

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
- 過学習は存在しない（非常に小さい確率でのみ存在する）という議論がある

S. Amari, N. Murata, K.-R. Muller, M Finke and H. H. Yang, Asymptotic Statistical Theory of Overtraining and Cross-Validation, IEEE Transactions on Neural Networks, Vol. 8, No. 5, pp. 985-996, 1997.

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

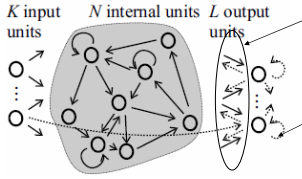
$$E(\vec{w}) \equiv \sum_{d \in D} \sum_{k \in outputs} \left[\left(t_{k,d} - o_{k,d}\right)^2 + \mu \sum_{j \in inputs} \left(\frac{\partial t_{k,d}}{\partial x_d^j} - \frac{\partial o_{k,d}}{\partial x_d^j} \right)^2 \right]$$

そのほかのニューロンモデル

- 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

ESN: 少し変わったNN

ESN: Echo State Network
Jaeger H. and Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science, 304:78-80, 2004.



この結合だけ修正（実線）

→隠れ層から出力層への結合のみ

点線は無くてもよい結合

結合加重

隠れ層と出力層：可変

その他：ランダムに決定し、以降固定

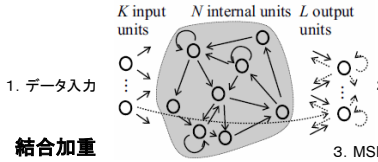
内部ユニットの出力

$$x(n+1) = f(W^{in}u(n+1) + Wx(n) + W^{back}y(n))$$

ESNの学習例

入力： $u(n) = \sin(n/5)$ →10πの間隔で入力・教師データを取得

出力： $y(n)_{teach} = 1/2\sin^7(n/5)$



1. データ入力

2. 教師データとのMSEを計算

3. MSEを最小化する結合加重を計算

結合加重

隠れ層内部：0 +0.4 -0.4に0.95 0.025 0.025の確率で決定

入力層と隠れ層：1 -1に等確率で決定

フィードバック結合：1 -1に等確率で決定

ESNの学習例:
The Mackey Glass system

- Chaotic attractorの学習・・・dynamical system の学習のためのテスト.ポピュラーだが難しい

$$\dot{y}(t) = \alpha y(t-\tau)/(1+y(t-\tau)^{\beta}) - \gamma y(t)$$

・パラメータは以下のように設定

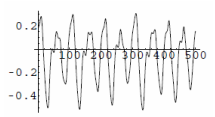
$$\alpha = 0.2, \beta = 10, \gamma = 0.1$$

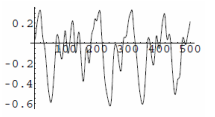
・Mildな動き $\tau = 17$　・Wildな動き $\tau = 30$

→離散化

$$y(n+1) = y(n) + \delta\left(\frac{0.2y(n-\tau/\delta)}{1+y(n-\tau/\delta)^{10}} - 0.1y(n)\right)$$

学習するデータ例



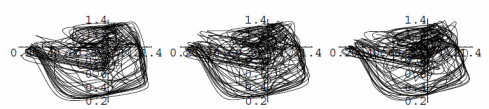


・Mildな動き $\tau = 17$　・Wildな動き $\tau = 30$

内部ユニットの出力：ノイズ入り

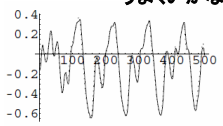
$$x(n+1) =$$
$$(1-\delta Ca)x(n) + \delta C(f(W^{in}u(n+1) + Wx(n) + W^{back}y(n) + v(n)))$$

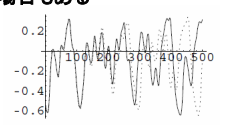
学習結果(Wildな動き) $\tau = 30$



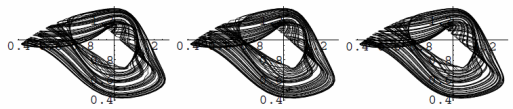
オリジナル　**21000step学習**　**3000step学習**

うまくいかない場合もある

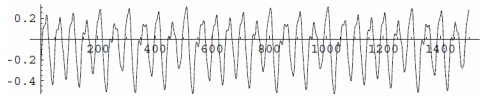




学習結果(Mildな動き) $\tau = 17$



オリジナル　**21000step学習**　**3000step学習**



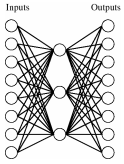
→かなりうまくいっている

補足：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

中間層での表現

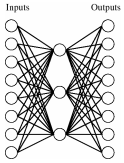
- これは学習できるか？



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）

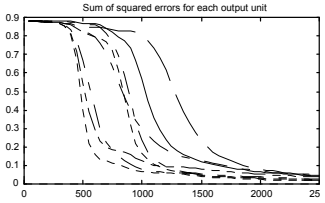
- 学習結果



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

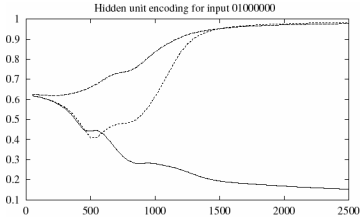
隠れ層での表現（3）

- 学習の進行の様子



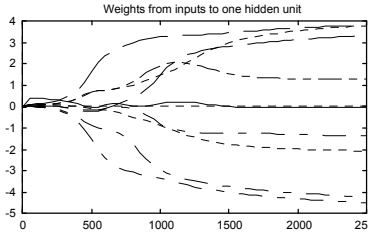
隠れ層での表現（4）

- 学習の進行の様子（2）

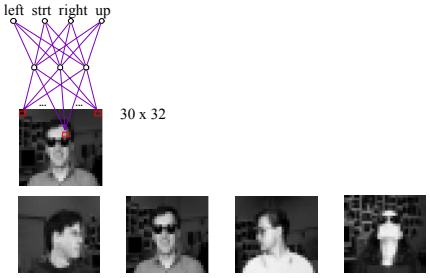


隠れ層での表現（5）

- 学習の進行の様子（3）



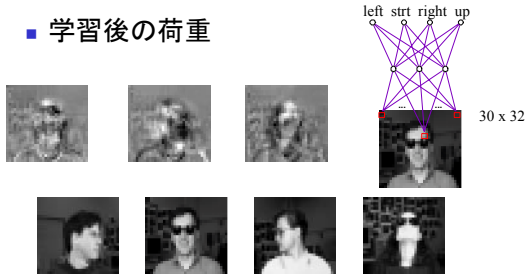
古い例: 顔画像の学習



- 顔画像の例

顔画像の学習

- 学習後の荷重



Bayes の定理

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

$$P(A,B) = P(A \mid B)P(B)$$
$$= P(B \mid A)\,P(A)$$

仮説選択に関して教えてくれること

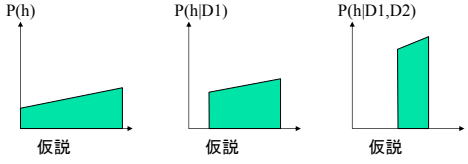
$$P(h \mid D) = \frac{P(D \mid h)\,P(h)}{P(D)}$$

P(h) = 仮説 h の事前確率
P(D) = 訓練データ D の生起確率
P(h|D) = D が与えられたときの h の生起確率
P(D|h) = h が与えられたときの D の生起確率

データ D を生成したらしい仮説 h を選択することができる！

大切な注: 条件付確率は因果関係(もしあれば)を反映するわけではない

ノイズがないときの事後確率の進展



力ずく MAP 仮説学習

1. 各仮説 h について, 事後確率を計算する:

$$P(h \mid D) = \frac{P(D \mid h)\,P(h)}{P(D)}$$

2. 出力する仮説 h_{MAP} は、その中で事後確率最大のもの, 引き分け時はランダムに選択:

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h)\,P(h)$$

回帰分析＝ML推定

学習事例: <x_i,d_i> 但し

$d_i = f(x_i) + e_i$

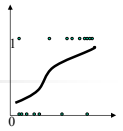
e_i はノイズ = iid なる正規分布に従う確率変数で、平均 =0 かつ分散は有限とする

$$p(D \mid h) = \prod_{i=1}^{N} e_i = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{d_i - f(x_i)}{\sigma}\right)\right)$$

ならば：

$$h_{ML} = \arg\max_{h \in H} \ln p(D \mid h) = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

ML推定=頻度の推定



- 密度≒頻度; d_i を頻度とすると

$$h_{ML} = \arg\max_{h \in H} \ln p(D \mid h)$$
$$= \arg\max_{h \in H} \sum_{i=1} d_i \ln f(x_i) + (1 - d_i) \ln(1 - f(x_i))$$

注: cross entropy $H(p,q) = -\sum_x p(x) \log q(x) = H(p) + D_{KL}(p \| q)$

すなわち、データの頻度分布との違いを最小化している

Bayes 最適な分類器

Bayes 最適な分類器は、(最適な)仮説を選ぶのではなく、最適な判断を選ぶ

$$\arg\max_{c_j \in \{+,-\}} \sum_{h_i \in H} P(c_j \mid h_i) P(h_i \mid D)$$

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h) P(h) \qquad h_{ML} = \arg\max_{h \in H} P(D \mid h)$$

注: Bayes 最適な分類器は H に含まれるとは限らない

注: 実行可能か？

注: 論文にはうまくいくと報告されているのだが、試してみるとMAPやMLと変わらない場合がある。どのような場合にそうなるか、興味のあるところである

オッカムの剃刀→MDL

- Occam's razor: "最短仮説を選べ"
- MDL: データ記述長最短の仮説を選べ

$$h_{MDL} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D \mid h)$$

ex. 木を記述するビット数
∝ 記述する符号の長さ

h が所与のとき、D を記述するビット数
∝ 誤分類データの個数

このままでは、使えない。使うようにした方法がある
1. Rissanen による統計的MDL
2. Kolmogorov/Chaitin のプログラム複雑度に基づくMDL

を、Lin & Vitanyi は実際のデータ圧縮で近似

MDL=MAP

- データの記述長 = −log データの生起確率
  仮説の記述長 = −log 仮説の生起確率

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h) P(h)$$
$$= \arg\min_{h \in H} -\log_2 P(D \mid h) - \log_2 P(h)$$
$$= \arg\min_{h \in H} L_{C2}(D \mid h) + L_{C_1}(h)$$

Naïve Bayes 分類器

- サンプル $x = \langle a_1, \ldots, a_n, v \rangle$ が与えられた(v は x が属するクラス)とき, vを推定したい($a_1, \ldots, a_n$は既知)

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, \ldots, a_n)$$
$$= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)}$$
$$= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)$$

naive

$$= \arg\max_{v_j \in V} P(a_1 \mid v_j) \cdots P(a_n \mid v_j) P(v_j)$$
$$\approx \arg\max_{v_j \in V} \hat{P}(a_1 \mid v_j) \cdots \hat{P}(a_n \mid v_j) \hat{P}(v_j)$$

- 各パラメータ(確率)の推定

$$\hat{P}(a_i \mid v_j) = \frac{count(\langle a_i, v_j \rangle) + mp}{count(v_j) + m} \qquad \hat{P}(v_j) = \frac{count(v_j)}{\sum_{v_j \in V} count(v_j)}$$