知的情報処理
9. ニューラルネットワーク

理工学部管理工学科
櫻井彰人

本日の内容

- なぜ、「神経回路網」の研究と応用があるのか？
- 脳と神経回路網のイメージ
- 神経素子・ネットワークのモデル
  - どういったモデルにしたか
- 学習アルゴリズム
  - 誤差逆伝播法
- 中間層表現
  - 新しい動き: Deep Learning
- その他のニューラルネットワーク
  - SOM(Kohonen map) など

何か?

- "… 人間の脳で使われていると思われる原理を用いて、精妙に作られたシステム." ― James Anderson
- "... ニューラルネットワークは、単純な処理素子を多数結合して作ったシステムである。その機能は、ネットワークの構造、結合の強さ、各素子での機能に依存して決まる." ― DARPA Neural Network Study (1988)
- "ニューラルネットワークは、単純な素子を相互結合したものである。各素子の機能は、動物の神経素子をまねたものであり、ネットワークとしての機能は素子間の結合強度に依存する。結合強度は、学習データのパターンに適応する、または、それを学習することにより、定められる." ― Kelvin Gurney

なぜか？

- 知的な機械を作りたいから（今までもそうであったし、これからも、恐らく、ず～っとそうであろう） ― 西ではゴーレム（ドラクエにも出てくるそうな。大魔神もゴーレムをもとにしているとか）、江戸時代のからくり人形、鉄腕アトム、ペッパー等々）
- しかも、コンピュータの能力も急速に発展し、脳に近づいている（かもしれない）
  - ただし、脳を計算機（の大親分）にたとえるのは間違いかもしれない。
  - その昔、時計がその時代の最も複雑な機械であったころ、脳は時計をたとえに説明された
- 本物の脳が用いるニューロンの速度は遅い。現在のCPUと比べ $1/10^5$ ～ $1/10^6$.
  - それにも関わらず、現在のコンピュータより遥かに優秀。なぜだ？
- だったら、脳の真似をしてみよう、（人工）ニューラルネットワークと名づけて。
  - 単純素子の超並列結合で試してみよう



http://matome.naver.jp/odai/2144230177036982401/2144230612743091003
http://bitbspdata.blogspot.jp/2014/10/blog-post_28.html
http://anguirus.exblog.jp/14023026/
http://www.praguecityline.com/jewish-praque/rabi-low-rabbi-who-constructed-the-praque-golem

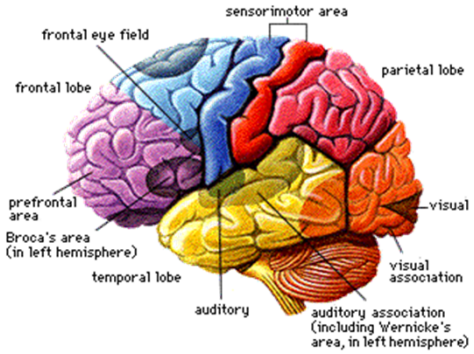
工学的だけではない

- “ニューラルネットワークは実用的な価値があると同時に、人間行動のモデルとしての価値もある.” -- Anderson

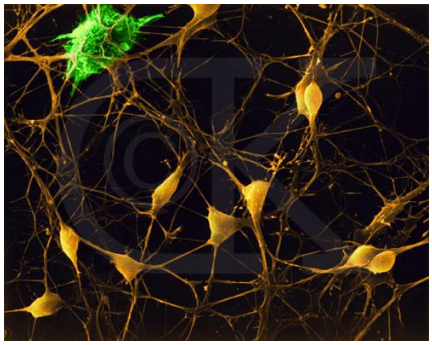
目次

http://www.willamette.edu/~gorr/classes/cs449/figs/oldbrain.gif

http://www.willamette.edu/~gorr/classes/cs449/figs/brain2.jpg

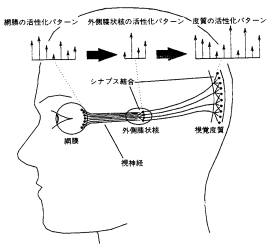


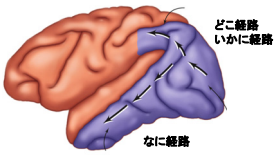
http://www.emc.maricopa.edu/faculty/farabee/BIOBK/brain.gif



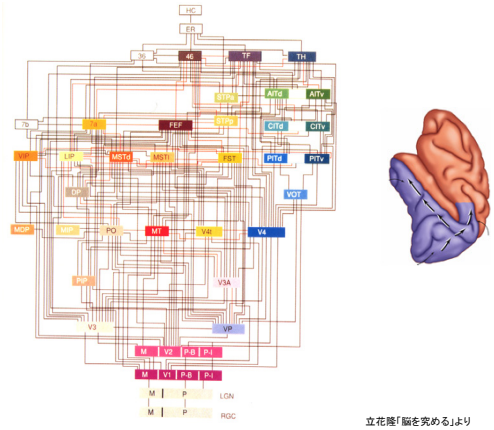
http://www.emc.maricopa.edu/faculty/farabee/BIOBK/08705a.jpg

copyright: Dennis Kunkel at http://www.DennisKunkel.com

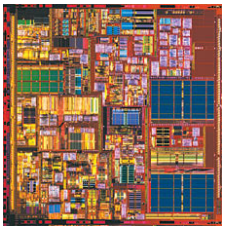


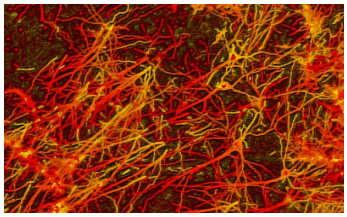


チャーチランド「認知哲学」から



立花隆「脳を究める」より





http://www.willamette.edu/~gorr/classes/cs449/figs/neurons.jpg

http://www.intel.com/pressroom/archive/photos/p4_photos.htm

	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）

能力面でも違いが

- 人間なら2分～3分
  - 1秒前後/（演算し書く）
- コンピュータなら？
  - 10^{-9}秒前後/演算
- もっと大変な数値計算なら、もっと大きな差がでる

http://drill100.com/images/Drill100.gif

能力の違い

- 単純計算は猛烈に速い
- 大量な計算でも速い
- 正確、信頼性抜群
- 多くの情報の統合は苦手 or できない
- 一部が壊れると全体が機能しない
- プログラムするしかない

（外界との交流がない等の人工知能自体の課題もある）

- 単純な計算は遅い
- 大量計算なんてできない
- 不正確、信頼性低い
- 多くの情報の統合は得意
- （壊れ方にもよるが）かなり壊れても大丈夫。
- プログラムはできないが、自律学習する

そこで

- 脳の基本原理（わかっていないのだが）を真似てコンピュータを作れば、脳のように賢くロバストかつ学習するコンピュータができるではないか？
- ポイント（と考えたこと）は
  - 単純要素をたくさんつなぎ、超並列動作
  - 結合の仕方、結合度合いを適応的に
    - 演算と同様に、結線が大切
  - データは分散して記憶
    - 少々壊れても推測可能

そこでモデル化

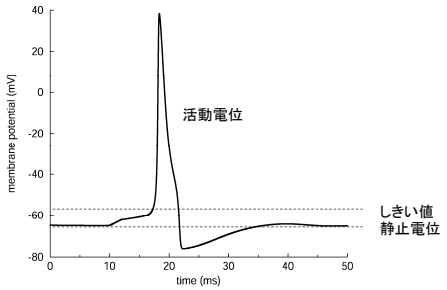
- 細胞一個のモデル化
  - 形態の観測、
  - 動作の測定等
- ネットワークのモデル化
  - 接続形態の観測、
  - 変化（学習）の測定等

今日の問題

- CPUの図とニューロンネットワークの図が載ったスライド（どれか？）を見て、CPUとニューロンネットワークとの類似と相違を述べよ
- コンピュータと人間の脳との類似と相違を、機能面から述べよ。講義資料の記述を引用してよい。

目次

- 脳と神経回路網
- **神経素子・ネットワークのモデル**
  - **多数素子の超並列実行とロバストネス**
  - **学習**
- 学習アルゴリズム
- 応用
  - テーマ別: 音声認識、自動運転、画像認識
  - 技術別: 関数近似、分類、連想、次元低減
- 中間層表現
- さまざまなニューラルネットワーク



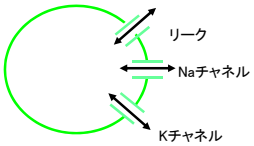
Hodgkin-Huxley方程式

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

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

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

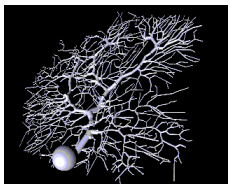
$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V) n$$



V: 膜電位、m, h(Na), n(K): チャンネルが開く確率

Santiago Ramón y Cajal による
プルキンエ細胞のスケッチ

http://en.wikipedia.org/wiki/Dendrite



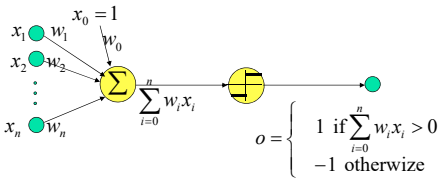
GENESISによるモデル

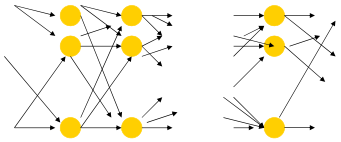
http://www.genesis-sim.org/GENESIS/cnslecs/purkcell.gif

簡略化

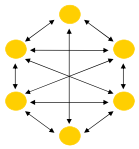
- 思い切って簡略化
- 本質は、
  - 多くの入力
  - 閾値関数的動作
    - 「閾値」以下なら出力は何も出さない
    - 「閾値」以上になると突然動作
    - 動作したあと暫くお休み
  - それを大量につなぐ

McCullogh-Pitts モデル(1943)

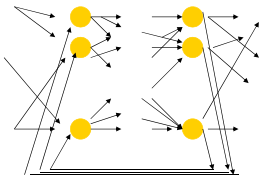




階層型(フィードフォワード)



相互結合



再帰型(リカレント)

パーセプトロン Perceptron

- Rosenblatt, F. (1957). "The perceptron: A perceiving and recognizing automaton (project PARA).", Technical Report 85-460-1, Cornell Aeronautical Laboratory.
- Rosenblatt, F. (1962). "Principles of Neurodynamics.", Spartan Books, New York.

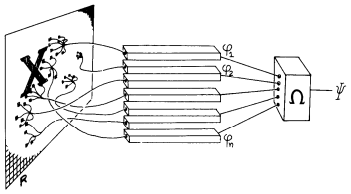


FIGURE 1. The one-layer perceptron analyzed by Minsky and Papert. (From *Perceptrons* by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press. Reprinted by permission.)

できること

- 何ができるか?
  - 文字(アルファベット)認識
  - いくつかのパターン認識課題 (形の認識等.)
  - しかも、パーセプトロン学習規則は、それが解くことができる全ての課題について、解を発見することができる、と証明できる
- 実は、学習ができることがポイント

目次

- 脳と神経回路網
- 神経素子・ネットワークのモデル
  - 多数素子の超並列実行とロバストネス
  - 学習
- **学習アルゴリズム**
- 応用
  - テーマ別: 音声認識、自動運転、画像認識
  - 技術別: 関数近似、分類、連想、次元低減
- 中間層表現
- その他のニューラルネットワーク

パーセプトロン Perceptron

- 下図の Ωは, 特徴抽出器の出力値の荷重付き線形和に閾値関数を施す
- 学習は、出力値が望みのものになるように、荷重を変更する

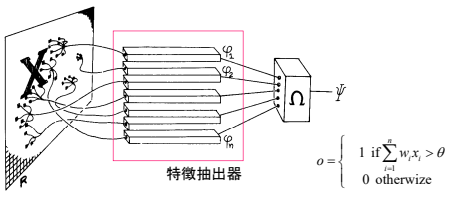


FIGURE 1. The one-layer perceptron analyzed by Minsky and Papert. (From *Perceptrons* by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press. Reprinted by permission.)

閾値は「一定」素子の荷重で

$$\sum_{i=1}^{n} w_i x_i > \theta$$

$$\sum_{i=1}^{n} w_i x_i - \theta > 0$$

$$\sum_{i=0}^{n} w_i x_i > 0 \qquad w_0 = -\theta \quad x_0 = 1$$

Rosenblatt の学習アルゴリズム

初期化: $\vec{w}$ は、任意の $\vec{x} \in F = F^+ \cup F^-$ とする

Repeat

順番に $\vec{x} \in F$ を選ぶ

If $\vec{w} \cdot \vec{x} > 0$ and $\vec{x} \in F^+$ then continue;

If $\vec{w} \cdot \vec{x} \leq 0$ and $\vec{x} \in F^+$ then FixPlus して continue;

If $\vec{w} \cdot \vec{x} \leq 0$ and $\vec{x} \in F^-$ then continue;

If $\vec{w} \cdot \vec{x} > 0$ and $\vec{x} \in F^-$ then FixMinus してcontinue;

until 間違えなくなる(FixPlusもFixMinusも呼ばなくなる)

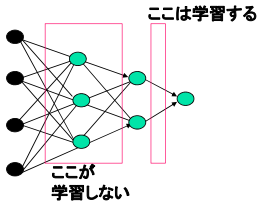
FixPlus: $\vec{w} := \vec{w} + \vec{x}$

FixMinus: $\vec{w} := \vec{w} - \vec{x}$

パーセプトロンの限界

- 何ができないか? 実は表現はできるが学習ができない
  - パリティ
  - 結合性
  - XOR 問題
  - 線形分離可能でない問題
- Marvin L. Minsky and Seymour Papert (1969), "Perceptrons", Cambridge, MA: MIT Press
- McCulloch & Pitts ニューロンのネットワークは Turing 機械と等価; でも 'それで?':
  - 学習させる方法を知らない
  - 予想: 任意のネットワークを学習させるアルゴリズムは、単に、存在しない

課題は



結果(出力値)が間違っているとき、結合荷重を変更すべきである。すべての荷重を変更して構わないが、問題はその変更量を決める方法が分からない

PDP

- "Perceptrons" のせいで、この分野の研究が20年遅滞したという…
- 転機: D.E. Rumelhart, J.L. McClelland, eds., "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", MIT Press, 1986.
  - 論文の集成, 数学的なものから哲学的なものまで
  - うまくいった実験結果をたくさん示している一方:
  - 誤差逆伝播学習アルゴリズム back propagation learning algorithm: 結局のところ多くのニューラルネットワークの学習を可能とした.
  - [実は, 類似の技法は, この間, 発見されていた (Amari 1967; Werbos, 1974, "dynamic feedback"; Parker, 1982, "learning logic") ので、再発見という言葉が適していよう. しかし、この分野を再出発させたことは大きな成果である.]

PDPの成功理由

- 素子の出力関数(閾値関数)を微分可能な関数(sigmoid関数)にかえた
- 学習問題を誤差最小化問題に変換した

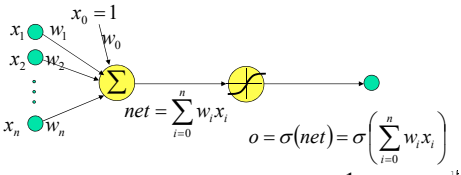
$$E(w) = \sum_{\text{すべてのデータ } x_k} \left(f(x_k; w) - x_k \text{の目標値} \right)^2$$

- (非線形・多変数の)誤差最小化問題を極めて naïve な方法でといた(最急降下法)

誤差の最小化

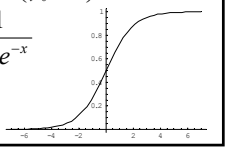
- 完全（誤差=0）を求めてはいけない
  - 我々の能力には限りがある
  - データにも誤りがあるかもしれない
  - そもそも、データの発生は確率的現象かもしれない
- （安易ではあるが）目標値と実際値の差の2乗を、全データについて足したものを誤差と考えよう
- それを最小化する荷重を決めればよい！

シグモイド素子



極めて頻繁に使われる σ：

$$\sigma(x) \equiv \frac{1}{1+e^{-x}}$$



最小化方法

- 微分して0とおいた方程式を解けばよい！本当か？

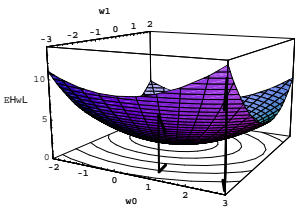
$$E(w) = \sum_{\text{すべてのデータ } x_k} \left(f(x_k; w) - x_k\text{の目標値}\right)^2$$

$$\frac{\partial E}{\partial w} = 0$$

- f は非線形関数ゆえ、この方程式は非線形連立方程式となる。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

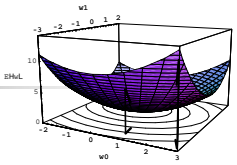
（反復）最小化法

- （最小化には様々な方法が提案されているが）
- 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！

具体的には



- 最急降下方向

$$-\frac{\partial E}{\partial w}$$

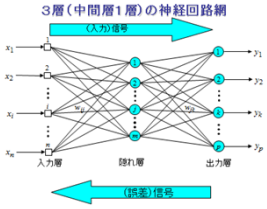
- その方向に沿って、少し、w を変更する

$$w^{new} \leftarrow w^{current} - \eta \frac{\partial E}{\partial w}$$

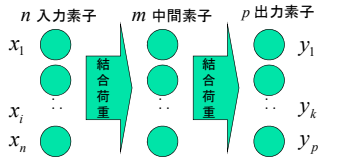
- 学習係数 η>0 は上手に定める必要あり

ニューラルネットワークの場合

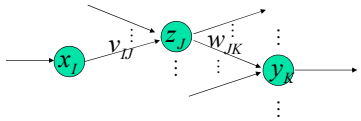
- 最急降下方向は結構複雑な関数であるが、上手に式変形すると、プログラムしやすい形で書ける。



誤差逆伝播法という
(error back-propagation)



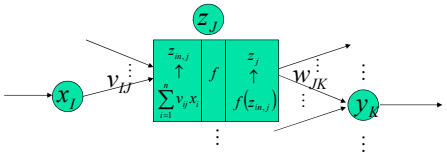
$$E = \sum_{k=1}^{p} [t_k - y_k]^2$$



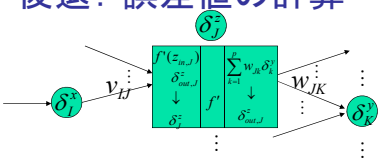
前進：出力値の計算

$$z_{in,j} = \sum_{i=1}^{n} v_{ij} x_i$$

$$z_j = f(z_{in,j})$$



後退：誤差値の計算

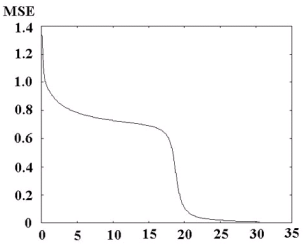


$$\delta_{out,J}^{z} = \sum_{k=1}^{p} w_{Jk} \delta_k^y$$

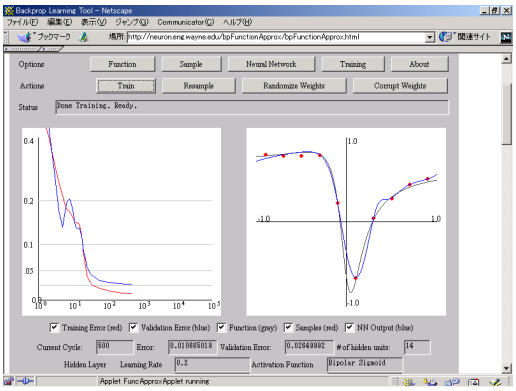
$$\delta_j^z = f'(z_{in,J}) \delta_{out,J}^z$$

わざわざ記憶する必要はない

誤差の減少の様子

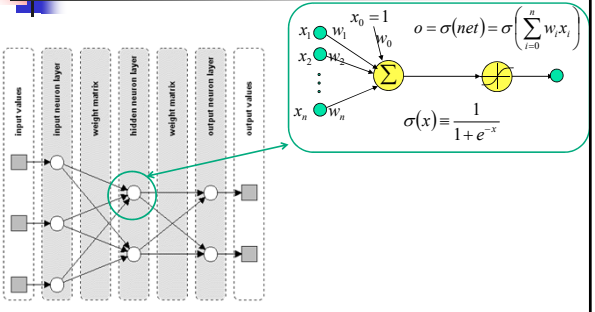


（比較的単純な問題の場合）



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

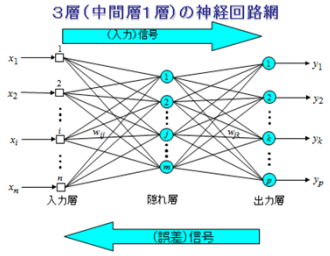
中間の纏め1



$$o = \sigma(net) = \sigma\left(\sum_{i=0}^{n} w_i x_i\right)$$

$$\sigma(x) \equiv \frac{1}{1+e^{-x}}$$

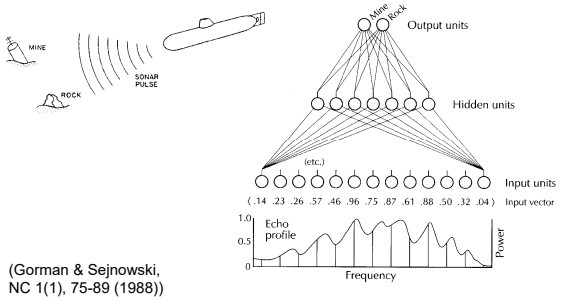
中間の纏め2



目次

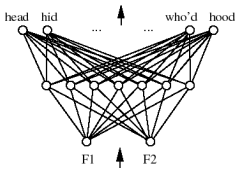
- 脳と神経回路網
- 神経素子・ネットワークのモデル
  - 多数素子の超並列実行とロバストネス
  - 学習
- 学習アルゴリズム
- **応用**
  - **テーマ別: 音声認識、自動運転、画像認識**
  - **技術別: 関数近似、分類、連想、次元低減**
- 中間層表現
- その他のニューラルネットワーク

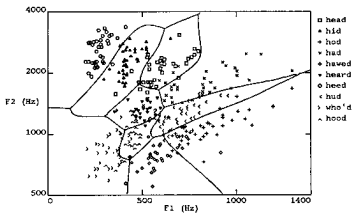
何ができるか？ 何でもできるが、



(Gorman & Sejnowski, NC 1(1), 75-89 (1988))

音声認識

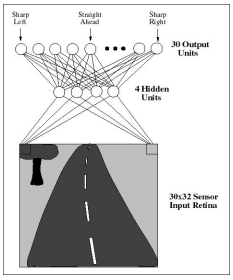


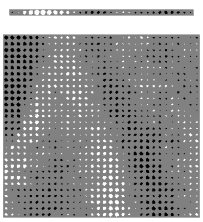


http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html

ALVINN：自動運転

- 高速道路で米国横断
  (Dean Pomerleau 1995)





http://web.mit.edu/6.034/wwwbob/L9/node10.html

学習できるだけではなく

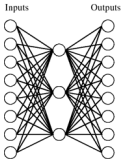
- 面白いことがわかった。それは、
- 中間層には、プログラマが意図しなかった内部表現が発生する（ことがある）
- よくよく見ると「意味深い」表現であったりする
- 実は、「情報の圧縮」すなわち、「意味抽出」が行われうることが示せる

目次

- 脳と神経回路網
- 神経素子・ネットワークのモデル
  - 多数素子の超並列実行とロバストネス
  - 学習
- 学習アルゴリズム
- 応用
  - テーマ別: 音声認識、自動運転、画像認識
  - 技術別: 関数近似、分類、連想、次元低減
- **中間層表現**
- その他のニューラルネットワーク

中間層表現 – 簡単な実験

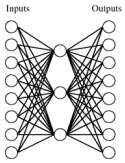
- これは学習できるか？



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

中間層表現(2)

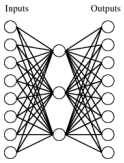
- 学習結果



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

中間層表現(3)

- 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

これで分かったこと

- 分散表現が実現できた。けれども
  - ロバストか？
  - 少々の破壊(荷重の変更)には yes
  - 大きな破壊には再学習が必要
- 学習ができた。
- (最新ハードウェアで実装すれば)速い

問題は残る

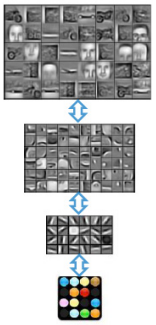
- 学習が遅い
  - 特にスケーラブルでない
  - つまり問題が大きくなるとかなり遅くなる
- 従って大きな問題に適用できない
  - 最初の夢(脳の実現！)は夢のまま
  - ニューラルネットワークだけで大きなシステムを作ることができない
- しかし、工夫がある
  - 例えば、高速化方法
    - テクニックではなく、本質的な方法
  - 例えば、組み合わせ
    - 人間の脳だって、機能モジュールの組み合わせ！
  - 例えば、support vector machine
- 問題があるにせよ、(機械学習の手法としては)素性の分からない問題をまず解いてみるには、よい方法である

もう新しくない動き – deep learning

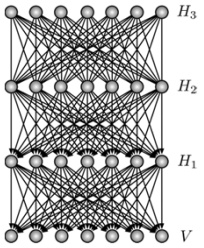
- 基本は、層の数が多い（少ない方は2層、3層、、多い方は、数十、数百）ニューラルネットワーク
- 昔は（deep learning ブームの前は）、存在はしたが、学習できなかったので、研究されなかった（パーセプトロンとなぜか同じ）
- アルゴリズムの工夫とハードウェアの進化（高速化、安価化）により、学習が可能になった
- 可能になったら、急に、いろいろできることが分かった
- それで一気にブームに

大枠

- 特徴量を学習する
  - Hand-craftはしない
- 抽象度が低い特徴から抽象度の高い特徴までを階層的に学習する
  - 抽象度の低い特徴は、類似タスクで利用可能



使用するネットワーク



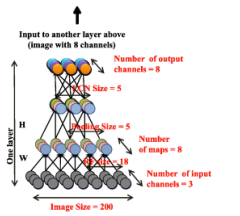


Figure 1. The architecture and parameters in one layer of our network. The overall network replicates this structure three times. For simplicity, the images are in 1D.

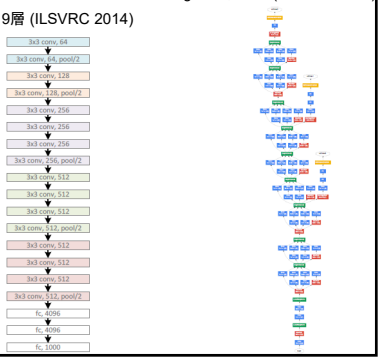
ちょっと脱線

層数の増加

AlexNet, 8層
(ILSVRC 2012)

VGG, 19層 (ILSVRC 2014)

GoogleNet, 22層 (ILSVRC 2014)



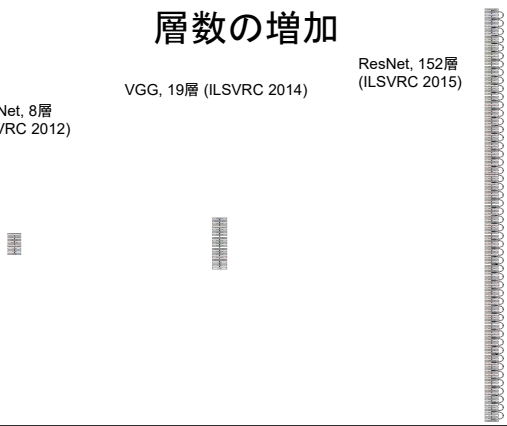
ちょっと脱線

層数の増加

AlexNet, 8層
(ILSVRC 2012)

VGG, 19層 (ILSVRC 2014)

ResNet, 152層
(ILSVRC 2015)



目次

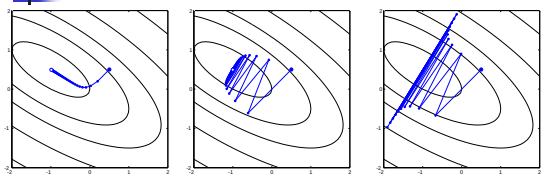
- 脳と神経回路網
- 神経素子・ネットワークのモデル
  - 多数素子の超並列実行とロバストネス
  - 学習
- 学習アルゴリズム
- 応用
  - テーマ別: 音声認識、自動運転、画像認識
  - 技術別: 関数近似、分類、連想、次元低減
- 中間層表現
- **その他のニューラルネットワーク**

その他のニューラルネットワーク

- 何種類もある
  - 次に2種類を紹介
- リカレントネットワーク
  - 時系列予測
  - 文法の学習
- パルス・ニューラルネットワーク
  - 神経らしく、パルスで動く
  - 10年前はコンピュータが遅くて研究できなかった
- カオスネットワーク
  - 実は、本物のニューロンは、これかも
- Liquid state machine
  - カオス系の予測精度が高い。

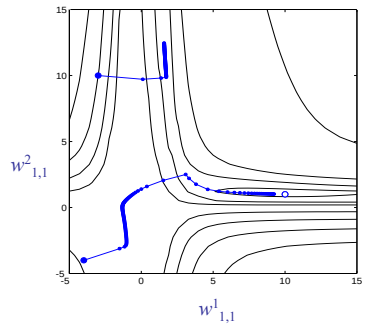
補足

学習パラメータによる違い



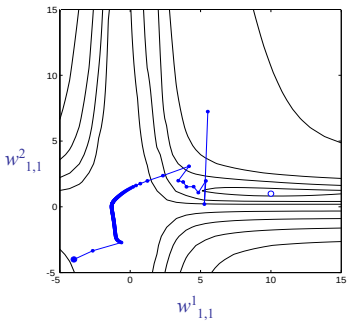
補足

収束過程の例



補足

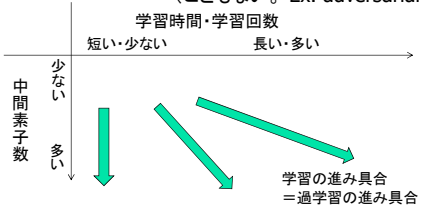
学習係数が大きすぎる場合



補足

過学習

- NNは通常は過学習しにくい
- 理論的には、勿論、過学習するのだが、実際には、それほどでもない（こともない。Ex. adversarial examples）



補足

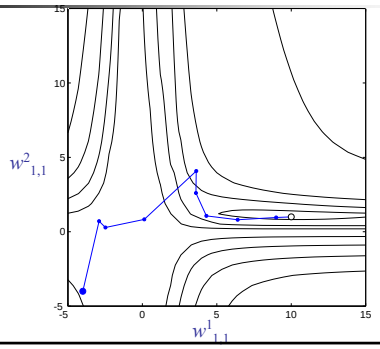
BP法で満足か？

- 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
  - なお、R の nnet では BFGS を用いている

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

補足

Levenberg-Marquardt 法



まとめ

- 脳と神経回路網
- 神経素子・ネットワークのモデル
  - 多数素子の超並列実行とロバストネス
  - 学習
- 学習アルゴリズム
- 応用
  - テーマ別: 音声認識、自動運転、画像認識
  - 技術別: 関数近似、分類、連想、次元低減
- 中間層表現
- さまざまなニューラルネットワーク

RにおけるNN

- R には、ニューラルネットワーク関連のパッケージとして、neuralnet、nnet などがある。
  - いずれにしても、遅く収束性の芳しくないアルゴリズム（誤差逆伝播法）が使われているのが惜しい。皆さんに誤解を与えてしまうので。

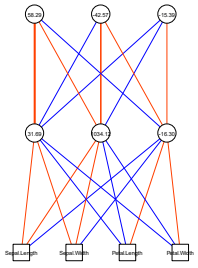
nnet の使用例

```
> library(nnet)
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> attach(iris)
> # iris は data.frame になっているので、そのまま使用。
> iris.nn <- nnet( Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
+                  size=3, maxit=1000)
# weights:  27
initial  value 177.100666
iter  10 value 24.409026
iter  20 value 7.183210
iter  30 value 5.221957
iter  40 value 4.923773
iter  50 value 4.922074
iter  60 value 4.921957
iter  60 value 4.921957
iter  60 value 4.921957
final  value 4.921957
converged
```

nnetの使用例（図示方法）

```
> # source("http://hosho.ees.hokudai.ac.jp/~kubo/log/2007/img07/plot.nn.txt")
> source("http://www.sakurai.comp.ae.keio.ac.jp/classes/IntInfProc-class/2017/plot.nn
> plot.nn(iris.nn)
> # confusion matrix
> table(Species,apply(predict(iris.nn),1,which.max))

Species       1  2  3
  setosa     50  0  0
  versicolor  0 49  1
  virginica   0  0 50
>
```



Rのバージョンによっては表示されない

```
> # 出力素子を 1 個 にしてみよう。各Species を 1,2,3 に割り振ることにする。
> #  as.integer(iris$Species) とすればよい
> # 勿論 Species をそのまま用いるとlogisitc function では表現できない。そこで
> # 出力素子を線形にしてみよう
> # 中間素子数も多くしてみよう(その必要は、実は、ない)
> iris.nn <- nnet(iris[-5],as.integer(iris$Species),linout=T,size=9, maxit=1000,decay=0.01)
# weights:  55
initial  value 2206.113647
iter  10 value 22.780551
iter  20 value 6.811561
iter  30 value 5.018212

iter 240 value 3.187806
iter 240 value 3.187806
final  value 3.187806
converged
> # 予測値は predict(iris.nn) で得られるが、それは実数値である。
> # それを四捨五入して（多分、1,2,3 となるだろう）、分類結果とする。
> table( Species, round(predict(iris.nn)) )

Species      1  2  3
  setosa    50  0  0
  versicolor  0 48  2
  virginica   0  1 49
>
> # 出力値を 1/3, 2/3,1 として学習をしてみよう
> iris.nn <- nnet(iris[-5],as.integer(iris$Species)/max(as.integer(iris$Species)),
+     linout=T,size=9, maxit=1000,decay=0.01)
# weights:  55
initial  value 178.965462
iter  10 value 0.906403
iter  20 value 0.670427

iter 270 value 0.585759
iter 280 value 0.585756
final  value 0.585756
converged
```

```
> table(Species,round(predict(iris.nn)*max(as.integer(iris$Species))))

Species      1  2  3
  setosa    50  0  0
  versicolor  0 47  3
  virginica   0  0 50
>
> # プロット
> plot(as.integer(Species), predict(iris.nn))
>
> # 予測誤差が計算できる. 大きな意味はない
> # 平均二乗誤差の平方根
> sqrt( sum((as.integer(Species) - predict(iris.nn))^2)/length(Species) )
[1] 1.446571
> # 平均絶対誤差
> sum( abs(as.integer(Species) - predict(iris.nn)))/length(Species)
[1] 1.333356
>
```

少々手間取るデータ

- 資料にあるデータ 09banknote.csv を対象として、ニューラルネットワークの学習を実行してみよう。このデータは
  - スイスの紙幣の偽札に関するデータであり、
  - もともとは下記の論文で用いられたものです。
  - 本データは, パッケージ alr3 にある banknote を csv ファイルに変換・保存したものです
- 実は、属性値の平均値が0から大きくはずれていたり、分散が大きいと nnet の学習アルゴリズムはうまく動かない。このデータはその一例となっています。

他のアルゴリズムでも起こります。そこで、例えば後で使用する svm などは常に正規化しています。

Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics: A practical approach.* London: Chapman & Hall.

```
> library(nnet)
> banknote <- read.csv ("09banknote.csv", header=TRUE)
> head(banknote)
  Length  Left Right Bottom  Top Diagonal Y
1  214.8 131.0 131.1    9.0  9.7    141.0 0
---略---
> attach(banknote)
>
> # 目標属性 Y の値の種類をみると
> table(banknote[,length(banknote)])

  0   1
100 100
> # 2値(0,1)なので、出力素子は1個であり、0と1が出力目標値になる。
>
> # banknote は data.frame になっているので、そのまま使用。
> # 中間素子は3個で試してみよう
> banknote.nn <- nnet( Y ~ ., banknote, size=3, maxit=1000)
# weights:  25
initial  value 50.180002
final  value 50.000000
converged
> # confusion matrix
> table( Y, round(predict(banknote.nn)) )

Y     1
  0 100
  1 100
>
> # うまくいかない。
> # 実は最初に見るべきであったのですが、各属性の値の分布をみてみよう
> # summary を見てもよいが
> summary(banknote)

---略---
```

乱数の種を設定・変更して変化があるか調べて下さい。

```
> # 平均と分散
> mean(banknote)
  Length     Left    Right   Bottom      Top Diagonal        Y
214.8960 130.1215 129.9565   9.4175  10.6505 140.4835   0.5000
> sd(banknote)
   Length      Left     Right    Bottom       Top  Diagonal         Y
0.3765541 0.3610255 0.4040719 1.4446031 0.8029467 1.1522657 0.5012547
> # 分散は小さくて扱いやすいが、平均値が大分0からはずれているのが分る。
> # そこで、各属性ごと、平均0, 分散1に正規化しよう。
> # ただし、最後の属性(分類先、今の場合Y)は正規化しない方がよさそう(してもよいがね)
> # ちょっと面倒になる。
> # 全ての属性を正規化してよいなら、
> #     normalize <- function (x) (x - mean(x))/sd(x)
> #     normalizedBanknote <- apply(banknote, 2, normalize)
> # でよい。
> # R に組み込まれている scale を使ってもよい(下記 normalize の代わりに scale を使う)
> normalize <- function (x) (x - mean(x))/sd(x)
> tmp <- apply(banknote[,1:length(banknote)-1], 2, normalize)
> normalizedBanknote <- cbind(data.frame(tmp), Y=banknote[,length(banknote)])
>
> # これで学習する
> banknote.nn <- nnet( Y ~ ., normalizedBanknote, size=2, maxit=1000)
---略---
converged
> table( Y, round(predict(banknote.nn)) )

Y     0   1
  0  99   1
  1   0 100
>
```

```
>
> # 良すぎるね。
> # 過学習かもしれない。では、10-fold cross validation で調べてみよう
>
> library(bootstrap)                # crossval を使用するために
>
> # 関数 crossval に必要な関数を定義する
> # なお、次の nnet の中の 2 は中間素子数の指定。
> theta.fit <- function (x,y) { return( nnet( x, y, ,2  ) ) }
> theta.predict <- function( fit, x) {predict( fit, data.frame(x)) }
>
> # では crossval を使ってみよう
> xy <- normalizedBanknote    # just for abbreviation
> results <- crossval(xy[,-length(xy)], xy[,length(xy)], theta.fit, theta.predict, ngroup=10)
---略---
converged
>
> # confusion matrix と分類精度
> (cm <- table( xy[,length(xy)], round( results$cv.fit )  ))

     0   1
  0 97   3
  1   0 100
> (accuracy10CV <- sum(diag(cm))/sum(cm))
[1] 0.985
```

cross validation については次回

本来は、中間素子数を変更して繰り返し実験をする必要があります。

本来は、乱数の種を変更して繰り返し実験する必要があります。

本日の問題

- 資料にあるデータ 09BreastCancer.csv を対象として、ニューラルネットワークの学習を実行してください。このデータは
  - 腫瘍の良性・悪性を判定する課題です。
  - 昔から使われているデータです。
  - パッケージ mlbench にあるデータ BreastCancer からIDフィールドを削除し、欠測値のある行は当該行を削除し、目標値 Class を0と1にするという変更をしました。
  - このデータの性質を見るには、パッケージ mlbench をインストールし、library(mlbench) そして ?BreastCancer として下さい。
- Banknote と同じように分析してみてください。

cross validation については次回