ニューラルネット

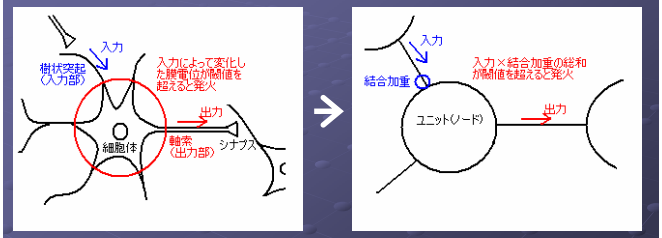
5/17　発表

M1　内野亮平

ニューラルネットとは何か

- 神経系をモデルとした，神経細胞に相当するユニット同士の結合とその結合の重み付けによって表現されたモデル．Rumelhart & McClellandが提唱．複数のユニットによる並列的な処理を特徴としており，並列分散処理モデル（PDP：Parallel Distributed Processing）やコネクショニストモデルとも呼ばれる．

神経系の仕組みとニューラルネットの簡単な対応図



このように，1つのユニットにおける発火が次々と他のユニットに伝わっていく（詳しい仕組みは後述）

伝わり方には，他のユニットの発火を促す興奮性のものと，発火を抑える抑制性のものの2種類がある．

なぜニューラルネットが必要とされたのか

- それまでの記号処理的・直列処理的モデル（PS：Production System）の限界

ー記号処理的モデルとは

・認知過程を，1つ1つの処理を特定のルールに基づいて順に行うものとして捉えたモデル．

・合理的な思考や問題解決，言語の使用といった高次の認知過程の解明に貢献

→しかし，全ての認知過程を記号処理モデルで表すことができるわけではなかった．

ニューラルネットの特徴

- 多くの処理を同時に行う分散処理
- 学習を繰り返すことでユニット同士の結合強度を変化させ，最適な値へと近づけていく．

→記号処理的モデルでは扱えなかった学習・知覚などをモデル化できる．（ex：顔の認識）

- 理論の確認だけでなく，結合強度の変化による実験的操作，内部構造の分析といった探索的なアプローチも可能

例：脳損傷のシミュレーション

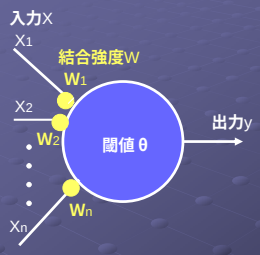
脳損傷のシミュレーション

- 脳損傷による症状のメカニズムを理解することは，人の認知過程を理解する上でも重要
- しかし，実際に人間の脳を破壊するわけにはいかない

→いったん学習が成立したネットワークを部分的に破壊することで，脳損傷の症状をコンピュータ上に再現することが可能

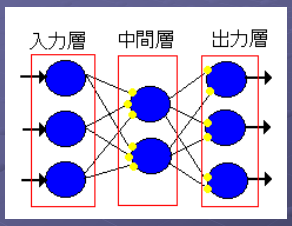
→どこをどのように破壊したか，それによって処理がどのように変化したかから症状のメカニズムを探る．

ニューラルネットの基本的な仕組み



- Xは入力，yは出力の値を表している．
- Wは入力に重み付けを行う結合強度で，学習によって変化する．
- 入力×結合強度の総和が閾値θを超えた時に出力yがなされる．

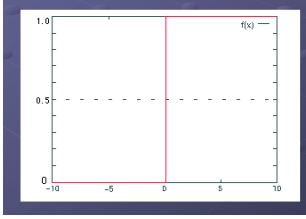
パーセプトロン



- Rosenblatt (1958) の提唱した階層型ネットワーク
- 入力層，中間層，出力層の3層構造
- 中間層は入力層の活性パターンを変換して出力層に送る役割を持つ．結合強度は固定
  (左図だと，3つのノードからなる入力を2つのノードに変換している)
- データの流れは入力から出力への一方向

パーセプトロンの入出力パターン

- 入力，出力のパターンは0か1 (階段関数，下図) で，入力X×結合強度Wの総和(net)が閾値θを超えた時のみ出力y＝1となる．
- 出力と望ましい出力 (教師信号) とを比較し，違っている時に結合強度と閾値を変化させて学習していく



$net=f\left(\sum_{i=1}^{n} Xi\ Wi-\theta\right)$

if net ＞ 0 then y = 1

If net ≦ 0 then y = 0

パーセプトロンの問題点

- 中間層の結合加重が変わらず，入力層のパターンを変換しているだけなので，実際には中間層と出力層の2層間の学習だけを見ればよい．
- つまり，学習という意味では2層のネットワークと変わらない．

→しかし，2層しかないと解決可能な問題が限られてくる

⇒どのような問題点があるかを，実際にtlearnでシミュレートして確かめてみる

tlearnの使い方

- tlearn.exeとその他の全ての関連ファイルを,任意のフォルダにまとめて入れる．
- tlearn.exeをダブルクリックすると,tlearnが起動する．
- 新しくネットワークを作る場合,Network＞New Project＞ファイル名入力＞開く
- 既存のネットワークを開く場合は, Network＞Open Project＞ファイル選択＞開く
- まずは既存のandファイルを開いてみる

→and.cf,and.date,and.teachの3つのファイルが開く

.cfファイルの見方

and.cf

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

入力層

ノードの数についての記述
- 入力層以外のノード数
- 入力層のノード数
- 出力層のノード数
- 出力層のノード番号

ノード同士の結合についての記述
- グループ化された結合の数
- 入力層1,2から1番ノードへの入力
- 0からの入力は閾値を表す

その他の設定
- Probe Selected Nodesで見るノード
- 最初の結合強度の制限値

.cfファイルの図式化

and.cf

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

Display＞Network Architectureでネットワーク構造を確認できる

.dataと.teachファイルの見方

and.dat

```
distributed
4
0 0
1 0
0 1
1 1
i1 i2
```

←学習項目のパターン数→
←学習項目の内容
各項目での正しい答え(教師信号) →

and.tch

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

ここではANDの演算ができるかを見たいので，

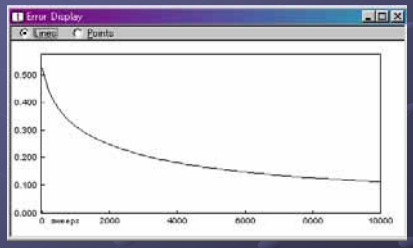
i1・i2ともに1を入力してる時だけ1を出力するように学習させる．

ネットワークの学習

- Network＞Training Optionで設定用のウインドウが開く
- Training Sweeps：学習回数　1000回のままでも良いが，多いほうがしっかり学習できる
- more＞Halt if RMS error falls belowで目安となるエラー率を設定できる．0.05と入力するとエラー率が5%以下になった時点で学習を止める．ここでは0でよい．
- その他は現時点ではそのままで問題ない．
- OKを選択し，Network＞Train the Networkで学習開始．
- Training Completedと出たら学習終了

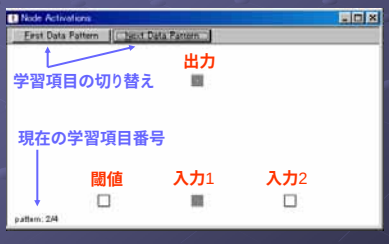
結果の見方：エラー率

- Display＞Error Displayでエラー率の推移が見れる．エラー率が低下が見られたら，全体として学習がうまくいっていることになる．



結果の見方：各学習項目での出力

- Display＞Node Activationsで各項目でのノードの活性状態が図示される．□は出力0の状態で，■の大きさが出力の大きさを表している．最大で出力約1となる．



■ 弱く活性 ＝出力小

□ 強く活性 ＝出力大

←この場合だと，入力1が0

入力2が1の時，出力値が非常に低くなっていることが分かる

結果の見方：各学習項目での出力

- Network＞ Verify Network has learnedで，学習項目ごとの具体的な出力の数値を確認することができる．

Output

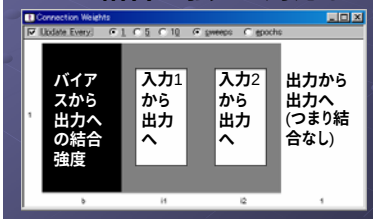
```
Output activations
using and-10000.wts and
  0.003
  0.120
  0.120
  0.857
```

←上から順に，学習項目1, 2, 3, 4の時の出力値

左図では学習項目4の時のみ高い出力となっており，ANDが正しく学習されていることが分かる

結果の見方：結合加重

- Display＞Connection Weightsで，横軸ノードの縦軸ノードへの結合加重の強さが図示される．真っ白の背景は結合なし，灰色背景に■は正の結合，□は負の結合を表し，大きさが結合の強さに対応している．



結果の見方：結合加重

- File＞Open＞and-○○.wtsで具体的な数値を見れる．上の図では左から並んでいたものが，こちらでは上から並ぶ

```
# TO NODE 1
-5.7737150192
3.7801330090
3.7816770077
0.0000000000
```

上から順に，バイアスノードから出力1への結合強度(つまり閾値)，入力1から出力1，入力2から出力1，出力1から出力1（つまりなし）

この表から，片方の入力しか1の値でない場合，閾値によって引かれる値のほうが大きいため，十分な出力が得られないことが分かる．

OR，XORの学習

- 次にOR，XORの学習をtlearnでシミュレートを行う．
- Andの.cf，.data，.teachを参考に，どういう記述をすれば考えてみよう．

OR:2つの入力のうち，少なくとも1つが1なら1を出力する．

XOR：2つの入力のうち，どちらか一方のみが1であった時，1を出力する．

OR，XORの記述

or.tch

```
distril
4
0
1
1
1
```

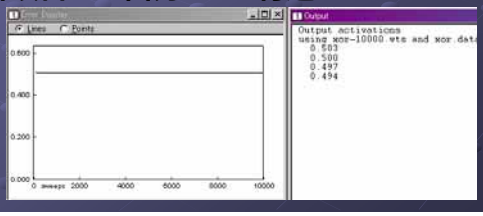
xor.tch

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

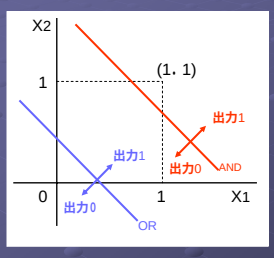
- それぞれの.teachを上のように書き換えればよい．（Network＞Open Project＞or，xorで既成のものが開ける．xor改はまだ開けない．）
- OR，XORそれぞれを10000回ほど学習させて結果を見てみよう．

OR，XORの結果

- エラー率，活性状態などを見て分かるように，ORの学習はうまくいくが，XORの学習はうまくいかない．
- XORにおいて，エラー率は約50%で推移し，各項目での出力も0．5付近で止まっている．



XORが学習できない理由



- ANDもORも，図のように1つの直線で分けること（線形分離）が可能だが，XORは不可能．

→非線形分離問題は解決不可能

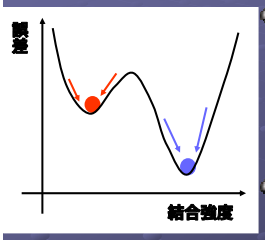
⇒誤差逆伝播法の登場

誤差逆伝播法
error back-propagation

- パーセプトロンの限界を補うための方法．非線形モデルを扱えるようにする．
- 最急下降法を用いて，出力と教師信号の誤差が小さくなるように結合強度を変化させる．
- その際に，出力層だけでなく中間層の結合強度も変化する．

25

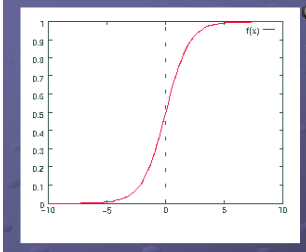
最急下降法



- 左の図の坂が「出力と教師信号の差」を，ボールが結合荷重の変化を表しているとすると，ボールは局所的に差が小さくなる点を求めて動き続けることが分かる．
- 最急下降法とは，このボールの動きのように，誤差が小さくなるような結合強度を求める方法のひとつである．

26

最急下降法



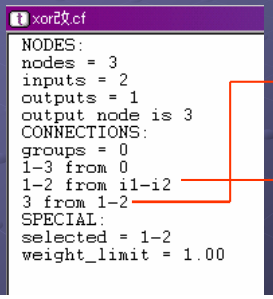
$$f(x)=\frac{1}{1+e^{-x}}$$

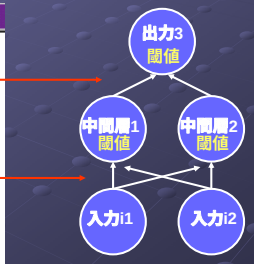
- 最急下降法では，最小値（極小値）を求めるために，関数上の各点の勾配を計算する必要がある．勾配を求めるためにはその関数を微分する必要があるため，ここでは階段関数ではなく微分可能なシグモイド関数が用いられる．
- これによって，出力値も0と1だけではなく，0から1までの実数値を取ることが可能となる．

27

Tlearnでの誤差逆伝播法による学習

- Network＞Open Project＞xor改を開く
- どんなネットワークかイメージできるだろうか





28

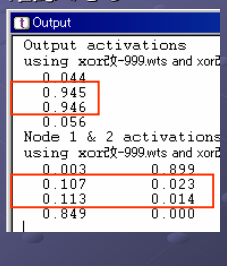
XORの学習

- Network＞Training Optionを開く
- Training Sweepsを10000回に設定
- Learning Rate：学習係数　結合強度の修正の程度を表す．今回は速く学習させるため，0.3に設定．
- Momentum：モーメント係数　結合強度の変化が前回の学習と同じ方向ならより大きく，違う方向なら小さく変化させる程度を表す．大きいほど学習が速く安定する．今回は速く学習させるため0.9に設定．
- Network＞Train the Networkで学習させる．
- 学習が終わったら，エラー率の推移・活性状態（特に中間層）・結合強度を確認してみる．

29

XOR結果

- エラー率が2000回くらいで低下していること，XORが学習できていることが分かる．
- Network＞Probe Selected Nodesで出力層，及び2つの中間層それぞれの項目毎の活性パターンが確認できる．



中間層が両方とも低い時にのみ出力層が活性化するように，工夫して学習していることが分かる．

（これ以外のパターンもあるが，ここではこのパターンに沿って話を進める）

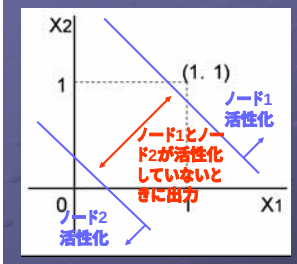
30

XOR結果2

```
# TO NODE 1
-5.9022793770
3.7829885483
3.8424088955
0.0000000000
0.0000000000
0.0000000000
# TO NODE 2
2.1829183102
-5.9342212677
-6.4513845444
0.0000000000
0.0000000000
0.0000000000
# TO NODE 3
3.8679935932
0.0000000000
0.0000000000
-7.8894743919
-7.7151923180
0.0000000000
```

- ノード1，2が中間層で3が出力ノード
- ノード1はi1・i2の2つとも発火した時にのみ，ノード2はi1・i2の2つとも発火していない時にのみ活性化する．
- 出力ノードは中間ノードが2つとも活性化していないときに出力する．

ここでの中間層の働き



- ノード1，2はAND・OR演算を用いて，XORで排除すべき2つのパターンで活性化するように学習する．
- 出力ノードは，ノード1・2の両方とも活性化していないときに出力するように学習する．
- つまり，1つの直線で分けることができない課題において，中間層が2本の直線の役割を担っている．

パーセプトロンとの違い

- 1つの出力ノードだけでは同時に処理できない2つのパターンを，2つの中間ノードに処理させる．この2段階の処理によってXOR演算を可能にしている．

→中間ノードは，出力ノードの問題解決を補助するための学習を行っている．

→中間ノードの結合強度が固定されているパーセプトロンではこのような学習が行えない．

- パーセプトロンの中間層にあらかじめそういう活性化をするように設定しておけば良いのでは？

→学習の結果，XORの演算ができることに意味があるので，最初からそう設定していたのでは意味がない．

誤差逆伝播法まとめ

- 誤差逆伝播法を用いることでXORを含めた論理演算が可能となった．
- しかし，先ほどの図でも分かるように，必ずしも最小値に落ち着くとは限らない．

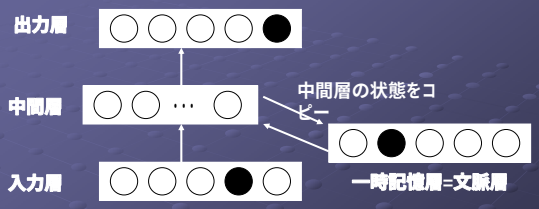
（ローカルミニマム；極小値に落ち着くこともありうる）

→この問題を解決する方法としてボルツマンマシンなどのモデルがあるが，ここでは省略．

エルマンネット

- 誤差逆伝播法の応用モデル．Elman(1990)が提唱．
- パーセプトロンだと同時生起パターンしか処理できなかったが，一時記憶層を加えることで時系列データも処理できるように改良した．
- 文法規則学習のシミュレーションに用いられる．

エルマンネットの図



- 各ノードが1つの単語に相当．
- 中間層の状態が一時記憶層にコピーされ，次の入力時に文脈情報として働く．
- 入力と一時記憶層の状態から，次に来る単語を予測し，出力する

エルマンネットによるシミュレーション Elman(1991)

- 複数の文を一単語ずつ入力して，文法規則を学習させた．

→複雑な文をいきなり学習させてもうまくいかなかったが，簡単な文から段階を踏むことで，複雑な文も学習できた．

→文法規則は段階を踏むことで学習可能

→しかし，子供が言語を学習する際に，必ずしも学習環境が段階づけられているとは限らない

エルマンネットによるシミュレーション2 Elman(1993)

- 環境側が段階付けられてなくても，学習者の作業記憶容量が段階的に増加することで文法規則が学習できるようになると仮定．
- 今度は一時記憶層に貯蔵できる単語数を，3~4語→4~5語→5~6語→・・・段階を踏んで増やして学習させた．
- 学習させる文は最初から複雑な文法のものを用いた

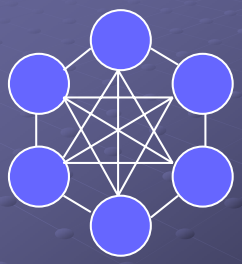
Elman(1993)の結果

- 記憶容量を段階を踏んで増やしていくことで，最初から複雑な文を入力しても，最後には文法規則が学習されることが分かった．
- ただし，初期の学習が十分になされていないと，後の学習が困難になることも分かった．

→子供の作業記憶容量が小さいことは，複雑な文法学習においては逆に有利にはたらくことが示された．

その他のニューラルネット

- 相互結合型ネットワーク



- 結合強度による特定のパターンを記憶
- 厳密な意味での出力層・入力層はなく，一部のノードの活性化が全体に広がり，特定のパターンに収束する．

ニューラルネットまとめ

- 並列処理的・非線形な関係をモデル化できる．
- コンピュータ上でシミュレートできるため，モデルの検証・操作が容易で，新たな知見が得られることもある．
- 今回は扱わなかったが，クラスター分析や主成分分析にも応用できる．
- ただし，ニューラルネットでは表現が困難なものもあるため,過信は禁物．

参考文献

岩田彰・松原俊之 (1996) ニューラルネットワーク入門 http://mars.elcom.nitech.ac.jp/java-cai/neuro

豊田秀樹 (1996) 『非線形多変量解析ーニューラルネットによるアプローチ』 朝倉書店．

守一雄 (1996) 『やさしいPDPモデルの話ー文系読者のためのニューラルネット理論入門』 新曜社．

守一雄・都築誉史・楠見孝(編著) (2001) 『コネクショニストモデルと心理学ー脳のシミュレーションによる心の理解』 北大路書房．