初心者のための人工ニューラルネットワーク

Carlos Gershenson
C.Gershenson@sussex.ac.uk

1. Introduction

このティーチングパッケージの目的は、人工ニューラルネットワーク(ANN)についての予備知識がない人のために、ANNを簡単に紹介することである。最初にネットワークのモデルについて簡単に紹介し、次にANNの一般的な用語について説明する。その応用として、バックプロパゲーション・アルゴリズムを説明する。バックプロパゲーション・アルゴリズムは広く使われており、他の多くのアルゴリズムがそこから派生しているからである。

ユーザは代数と関数およびベクトルの取り扱いを知っている必要がある。微分積分は推奨されるが，必須ではない．本パッケージの内容は，高等教育を受けた人であれば理解できるはずである．ANNって何だろうという興味本位の人、ANNに慣れ親しみたい人、ANNをより本格的に勉強するときに、ANNに対する明確な概念をすでに持っている人に役立つだろう。また、バックプロパゲーション・アルゴリズムの詳細で正式な説明なしに、それを適用したいだけの人にも、この教材は役に立つだろう。本書は「ダミーのためのANN」ではないが、もちろん論文でもない。形式的な説明の多くは、簡略化のために省略されている。詳細な説明と実演は、参考文献にある。付属の練習問題は理論の理解を補うものである。この簡単なイントロダクションを発展させるために、オンラインリソースを強く推奨する。

2. Networks

複雑な問題を解く効率的な方法の一つは、「分割と征服」のレンマに従うことである。複雑なシステムは、それを理解できるようにするために、より単純な要素に分解することができる。また、単純な要素を集めて複雑なシステムを作り出すこともできる（Bar Yam, 1997）。ネットワークは、これを実現するためのひとつのアプローチである。ネットワークには数多くの種類があるが、どれも次のような構成要素によって特徴づけられる。

ノードは計算ユニットと見なすことができる。ノードは入力を受け取り、それを処理して出力を得る。この処理は非常に単純なもの(入力の合計など)もあれば，非常に複雑なもの(ノードに別のネットワークが含まれている...)もあります.

接続はノード間の情報の流れを決定する。情報が一方向にしか流れない一方向性と、情報が双方向にしか流れない双方向性がある。

接続を介したノードの相互作用は、ネットワークの要素では観察できない、ネットワークの大域的な振る舞いをもたらす。このグローバルな振る舞いは創発的であると言われている。これはネットワークの能力がその要素の能力を凌駕することを意味し、ネットワークを非常に強力なツールにしている。

ネットワークは、物理学、コンピュータ・サイエンス、生化学、倫理学、数学、社会学、経済学、電気通信、その他多くの分野で、幅広い現象のモデルに使われている。タンパク質、コンピュータ、コミュニティなど、多くのシステムをネットワークとして見ることができるからだ。他にネットワークと見なせるシステムは？なぜですか？

3. 人工ニューラルネットワーク

あるタイプのネットワークでは、ノードを「人工ニューロン」と見なす。これは人工神経回路網（ANN）と呼ばれる。人工ニューロンは、自然のニューロンから着想を得た計算モデルである。自然のニューロンは、ニューロンの樹状突起や膜にあるシナプスを介して信号を受け取る。受け取った信号が十分に強いと（ある閾値を超えると）、ニューロンは活性化され、軸索を通して信号を発する。この信号は別のシナプスに送られ、他のニューロンを活性化するかもしれない。

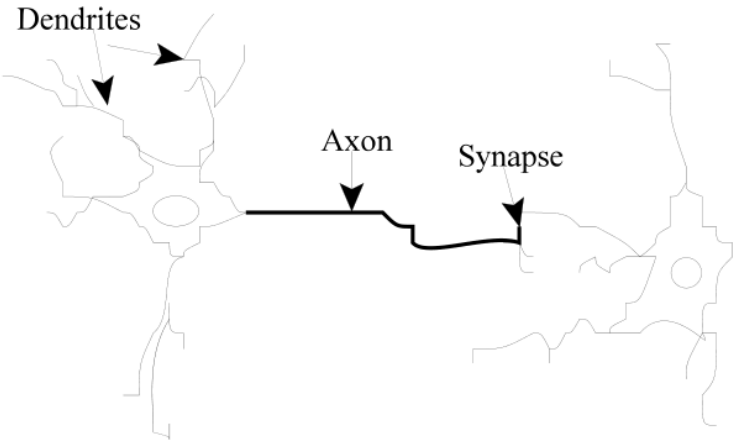


図1. 自然ニューロン（想像図）。

人工ニューロンをモデル化するとき、実際のニューロンの複雑さは高度に抽象化される。人工ニューロンは基本的に入力（シナプスのようなもの）で構成され、これ に重み（それぞれの信号の強さ）を掛け合わせ、ニューロンの活性化を決定する数学 的関数で計算する。もう1つの関数（IDの場合もある）は、人工ニューロンの出力を計算する（ある閾値に依存する場合もある）。ANNは人工ニューロンを組み合わせて情報を処理する。

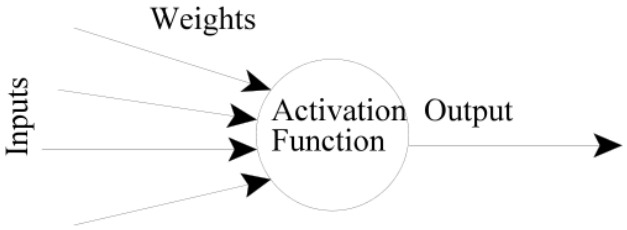


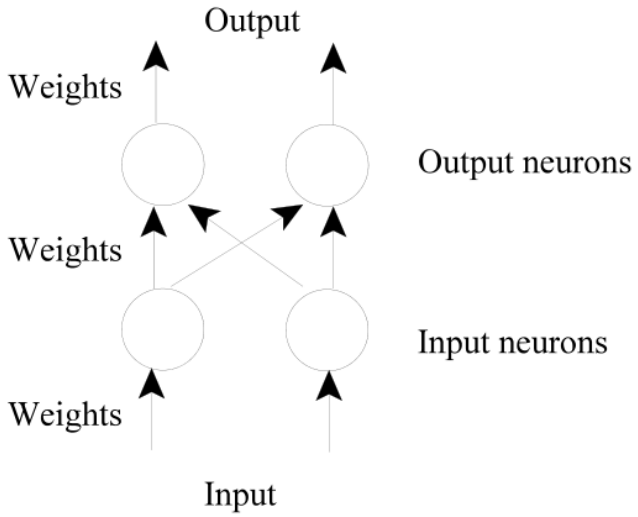
図2. 人工ニューロン

人工ニューロンの重みが大きいほど、それに乗算される入力は強くなる。重みは負にすることもできるので、負の重みによって信号が抑制されると言える。重みによって、ニューロンの計算は異なる。人工ニューロンの重みを調整することで、特定の入力に対して望む出力を得ることができる。しかし、数百、数千のニューロンからなるANNの場合、必要な重みをすべて手作業で見つけるのは非常に複雑である。しかし、ネットワークから望ましい出力を得るために、ANNの重みを調整できるアルゴリズムを見つけることができる。この重みを調整するプロセスは、学習または訓練と呼ばれる。
ANNの種類と用途は非常に多い。McCullochとPitts(1943)による最初のニューラル・モデル以来、ANNとみなされるモデルは何百種類も開発されている。それらの違いは、関数、許容値、トポロジー、学習アルゴリズムなどである。また、各ニューロンがここでレビューしているものよりも多くの特性を持つハイブリッド・モデルも数多く存在する。スペースの問題から、適切な重みを学習するためにバックプロパゲー ション・アルゴリズム（Rumelhart and McClelland, 1986）を使って学習する ANN だけを紹介する。

ANNの機能は情報を処理することであるため、主にそれに関連する分野で使用される。実際の神経回路網をモデル化し、動物や機械の行動や制御を研究するために使われるANNだけでなく、パターン認識、予測、データ圧縮などの工学的な目的で使われるANNもあり、その種類は多岐にわたる。

3.1. Exercise

この演習は人工ニューラルネットワークの概念に慣れるためのものである。つの人工ニューロンからなるネットワークを構築する。つのニューロンはネットワークへの入力を受け取り、残りの2つのニュー ロンはネットワークからの出力を与える。



各矢印には、情報の流れを表す重みが割り当てられている。これらの重みは各矢印を通過する値に乗算され、各矢印を通過する信号の強さを増減させる。

このネットワークの神経細胞は、ただそれぞれの重みを合計するだけである。このネットワークのニューロンは、それぞれの入力を合計するだけである。入力ニューロンは1つの入力しか持たないので、その出力は、受け取った入力に重みを掛けたものになる。この重みが負ならどうなるか？この重みがゼロの場合はどうなるか？

出力層のニューロンは、両方の入力ニューロンの出力を受け取り、それぞれの重みを掛け合わせ、それらを合計する。その出力に別の重みが掛けられる。

ここで、すべての重みを1に等しく設定する。これは情報が影響を受けずに流れることを意味する。以下の入力に対するネットワークの出力を計算せよ：(1,1)，(1,0)，(0,1)，(0,0)，(-1,1)，(-1,-1)．

よし。次に、重みを0.5、0、-0.5の中から選び、ネットワークに沿ってランダムに設定する。上と同じ入力に対して出力を計算する。いくつかの重みを変えて、ネットワークの振る舞いが変わるのを見よう。どの重みがより重要か（その重みを変更すると、出力はより劇的に変化する）？

Now, suppose we want a network like the one we are working with, such that the outputs should be the inputs in inverse order (*e.g.* (0.3,0.7)->(0.7,0.3)).

That was an easy one! Another easy network would be one where the outputs should be the double of the inputs.

次に、ニューロンにしきい値を設定しましょう。つまり、ニューロンの前回の出力（入力の加重和）がニューロンのしきい値より大きければ、ニューロンの出力は1になり、そうでなければ0になる。すでに開発されているいくつかのネットワークにしきい値を設定し、これがそれらの動作にどのように影響するかを見てみよう。

さて、入力として0と1だけを受け取るネットワークがあるとする。最初の出力ニューロンの出力がネットワーク入力の論理和（AND）（両方の入力が1のときは1、それ以外は0）になり、2番目の出力ニューロンの出力がネットワーク入力の論理和（OR）（両方の入力が0のときは0、それ以外は1）になるように、ニューロンの重みとしきい値を調整する。要求された結果を与えるネットワークが1つ以上あることがわかるだろう。

さて、このような小さなネットワークの重みを調整するのは、それほど複雑なことではないのかもしれない。何百ものニューロンからなるネットワークが必要な場合、どのように重みを調整すれば望ましい出力が得られるだろうか？それを見つける方法はいくつかあるが、ここでは最も一般的な方法を紹介しよう。

4. バックプロパゲーションアルゴリズム

バックプロパゲーション・アルゴリズム（Rumelhart and McClelland, 1986）は、階層化されたフィードフォワードANNで使用される。つまり、人工ニューロンは層で構成され、信号を「前方」に送り、誤差は後方に伝搬される。ネットワークは入力層のニューロンによって入力を受け取り、出力層のニューロンによって出力される。1つ以上の中間隠れ層があってもよい。つまり、ネットワークに計算させたい入力と出力の例をアルゴリズムに与え、誤差（実際の結果と期待される結果の差）を計算する。バックプロパゲーション・アルゴリズムの考え方は、ANNが訓練データを学習するまで、この誤差を減らすことである。訓練はランダムな重みで開始され、その目標は誤差が最小になるように調整することである。

バックプロパゲーション・アルゴリズムを実装したANNの人工ニューロンの活性化関数は、加重和（入力x_iにそれぞれの重みw_{ji}を掛けたもの）である：

$$A_j(\bar{x},\bar{w}) = \sum_{i=0}^{n} x_i w_{ji} \qquad (1)$$

活性化は入力と重みにのみ依存することがわかる。もし出力関数が恒等式（output=activation）であれば、このニューロンは線形と呼ばれる。しかし、これには大きな制限がある。最も一般的な出力関数はシグモイド関数である：

$$O_j(\bar{x},\bar{w}) = \frac{1}{1+e^{A_j(\bar{x},\bar{w})}} \qquad (2)$$

シグモイド関数は，大きな正の数では1に非常に近く，0では0.5，大きな負の数では0に非常に近い．これによって、ニューロンの低出力と高出力（ゼロに近いか1に近い）の間のスムーズな遷移が可能になる。出力は活性化のみに依存し、その活性化は入力とそれぞれの重みの値に依存することがわかる。

さて、学習プロセスの目標は、ある入力が与えられたときに望ましい出力を得ることである。誤差は実際の出力と希望する出力の差であるため、誤差は重みに依存し、誤差を最小化するために重みを調整する必要がある。各ニューロンの出力に対して誤差関数を定義することができる：

$$E_j(\bar{x},\bar{w},d) = \left(O_j(\bar{x},\bar{w}) - d_j\right)^2 \qquad (3)$$

誤差は常に正であり、差が大きければ大きく、小さければ小さくなるからである。ネットワークの誤差は、単純に出力層の全ニューロンの誤差の合計となる：

$$E(\bar{x},\bar{w},\bar{d}) = \sum_j \left(O_j(\bar{x},\bar{w}) - d_j\right)^2 \qquad (4)$$

バックプロパゲーション・アルゴリズムは、誤差が出力、入力、重みにどのように依存するかを計算する。これを求めたら、勾配降下法を使って重みを調整します：

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \qquad (5)$$

各重み( ) w_{ji} の調整は、定数エタ(0)にネットワークの誤差に対する前の重みの依存性(これはw_iに対するEの微分である)を掛けたものの負になる。調整の大きさは0 に依存し、関数の誤差に対する重みの寄与に依存する。つまり、重みの誤差への寄与が大きければ、寄与が小さい場合よりも調整は大きくなる。(5)は、適切な重みが見つかるまで（誤差が最小になるまで）使用される。導関数を知らなくても心配はいらない。導関数は関数であり、すぐに代数式に置き換えることができる。導関数を理解している場合は、自分で式を導出し、その結果をここで示したものと比較してください。バックプロパゲーションアルゴリズムの数学的証明をお探しの方は、この教材の範囲外なので、推薦図書で確認することをお勧めします。

つまり、w_{ji}に関してEの導関数を求める「だけ」でよい。これはバックプロパゲーション・アルゴリズムのゴールであり、これを逆に達成する必要があるからだ。まず、誤差が出力にどれだけ依存するかを計算する必要があり、これはO_jに対するEの微分である（(3)より）。

$$\frac{\partial E}{\partial O_j} = 2\left(O_j - d_j\right) \qquad (6)$$

(1)と(2)から)重みに依存する活性化(activation)にどれだけ出力が依存するか：

$$\frac{\partial O_j}{\partial w_{ji}} = \frac{\partial O_j}{\partial A_j}\frac{\partial A_j}{\partial w_{ji}} = O_j(1 - O_j)x_i \qquad (7)$$

(6)と(7)から)わかる：

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial O_j}\frac{\partial O_j}{\partial w_{ji}} = 2\left(O_j - d_j\right)O_j(1 - O_j)x_i \qquad (8)$$

そして、各重みの調整量は（(5)と(8)より）となる：

$$\Delta w_{ji} = -2\eta\left(O_j - d_j\right)O_j(1 - O_j)x_i \qquad (9)$$

(9)は2層のANNを学習するときにそのまま使える。さて、もう1つの層を持つネットワークを学習するためには、いくつかの考慮が必要である。前の層の重み（仮にv_{ik}とする）を調整したい場合、まず誤差が重みに依存するのではなく、前の層からの入力に依存することを計算する必要がある。これは簡単で、(7)，(8)，(9)のx_iをw_{ji}で変えればよい。しかし、ネットワークの誤差がv_{ik}の調整にどのように依存するかを見る必要もある。そこで

$$\Delta v_{ik} = -\eta \frac{\partial E}{\partial v_{ik}} = -\eta \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial v_{ik}} \tag{10}$$

Where:

$$\frac{\partial E}{\partial w_{ji}} = 2\left(O_j - d_j\right) O_j (1 - O_j) w_{ji} \tag{11}$$

そして、v_{ik}を持つニューロンへの入力u_kがあると仮定する（(7)より）：

$$\frac{\partial x_i}{\partial v_{ik}} = x_i (1 - x_i) v_{ik} \tag{12}$$

さらに別の層を追加したい場合は、同じように、誤差が最初の層の入力と重みにどのように依存するかを計算すればよい。各層は異なる数のニューロンを持つ可能性があるため、インデックスに注意する必要がある。

現実的な理由から、バックプロパゲーション・アルゴリズムを実装したANNは、ネットワークのトレーニング時間が指数関数的に増大するため、あまり多くの層を持たない。また、バックプロパゲーションアルゴリズムには、より高速な学習を可能にする改良が施されている。

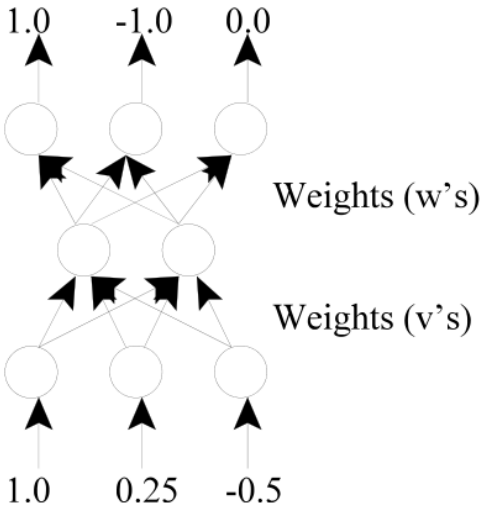
4.1. Exercise

プログラミングの方法を知っているなら、バックプロパゲーション・アルゴリズムを実装し、少なくとも以下のネットワークを訓練する。もしバックプロパゲーション・アルゴリズムの一般的な実装ができるのであれば、どうぞ（1層あたりのニューロン数、学習セット、偶数層は問いません）。

プログラミングの方法は知らないが、数学アシスタント(MatlabやMathematicaなど)の使い方を知っている場合、作業を容易にする関数を定義した後、以下のネットワークに適した重みを見つける。

If you do not have any computing experience, find the weights by hand.

The network for this exercise has three neurons in the input layer, two neurons in a hidden layer, and three neurons in the output layer. Usually networks are trained with large training sets, but for this exercise, we will only use one training example. When the inputs are (1, 0.25, -0.5), the outputs should be (1,-1,0). Remember you start with random weights.



5. Further reading

- 以下の名著はANNをより深く掘り下げている： Rojas, R. (1996). Rojas, R. (1996)： A Systematic Introduction. Springer, Berlin. Rumelhart, D. and J. McClelland (1986). 並列分散処理。MIT Press, Cambridge, Mass.

For further information on networks in general, and related themes, these books are quite useful and illustrative:

- Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Addison-Wesley.
- Kauffman, S. (1993) *Origins of Order*, Oxford University Press.

6. オンラインリソース

インターネット上には、ニューラルネットワークに関連する膨大なリソースがある。Javaアプレット（ソースコードあり）を使った素晴らしい例題付きのチュートリアルがEPFLで開発された（http:／／diww.epfl.ch／mantra／tutorial／english／）。他に、マドリッド政治大学（http:／／www.gc.ssr.upm.es／inves／neural／annl／anntutorial.html）とインペリアル・カレッジ（Imperial College）のチュートリアルがある。

of Science, Technology and Medicine University of London (http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html). The author also has a small amount of resources related to programming neural networks in Java (http://jlagunez.iquimica.unam.mx/~carlos/programacione.html).

7. Bibliography

Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Addison-Wesley.

Kauffman, S. (1993) *Origins of Order*, Oxford University Press.

McCulloch, W. and W. Pitts (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction.* Springer, Berlin.

Rumelhart, D. and J. McClelland (1986). *Parallel Distributed Processing*. MIT Press, Cambridge, Mass.

Young, D. *Formal Computational Skills Course Notes*. Http://www.cogs.susx.ac.uk/users/davidy/fcs