名古屋工業大学 先進セラミックス研究センター　井田 隆　　2023 年 2 月 2 日更新

１５．人工神経網
Artificial Neural Network

１５−１　ニューラルネットワークとは？
What is Artificial Neural Network ?

日本のコンピュータ技術の分野で「ニューラルネットワーク」という言葉で表されていることは，**人工神経網**（アーティフィシャル ニューラル ネットワーク） artificial neural network (ANN) の意味であり，**人工ニューロン** artificial neuron をつなぎ合わせて「神経網のようなもの」を作ろうとする考え方に基づいています。

Fig. 15.1.1 に単純なニューラルネットワークの例を示します。図中の○は**ノード**（ノウド） node（節（ふし））と呼ばれ，ノードの間を結ぶ矢印は，信号の進む方向を表します。この例は，**入力層**（インプット レイヤ） input layerと中間層（**隠れ層**（かく ヒドゥン） hidden layer），**出力層**（アウトプット） output layer の３層からなり，これを「３層ニューラルネットワーク（トリプル レイヤ） triple layer neural network」と呼ぶ流儀（りゅうぎ）と，人工ニューロンをノードとして含む層は２層なので「２層ニューラルネットワーク（ダブル） double layer neural network」と呼ぶ流儀とがあります。

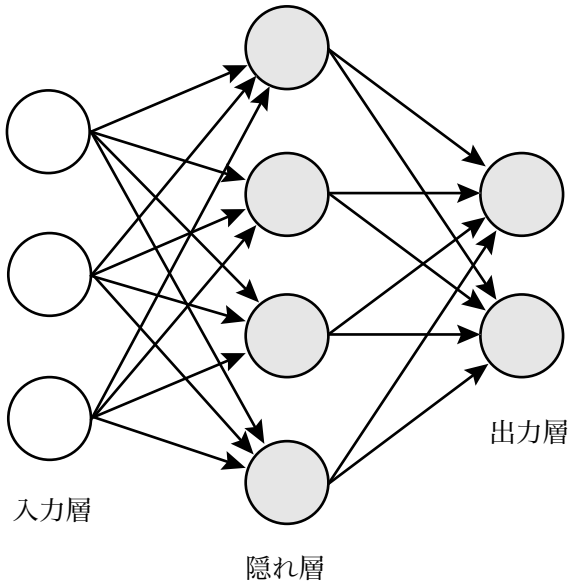

Fig. 15.1.1　単純なニューラルネットワーク

「ニューラルネットワーク」では，原則的に，隣り合う層にあるノードの間でのみ接続を行います。そのようにノード（ニューロン）間の接続のしかたに制限を加えることで，単純に「隠れ層の数を増やす」ことだけで「より複雑な論理に基づく処理が可能になる」ことを期待できます（Fig. 15.1.2）。

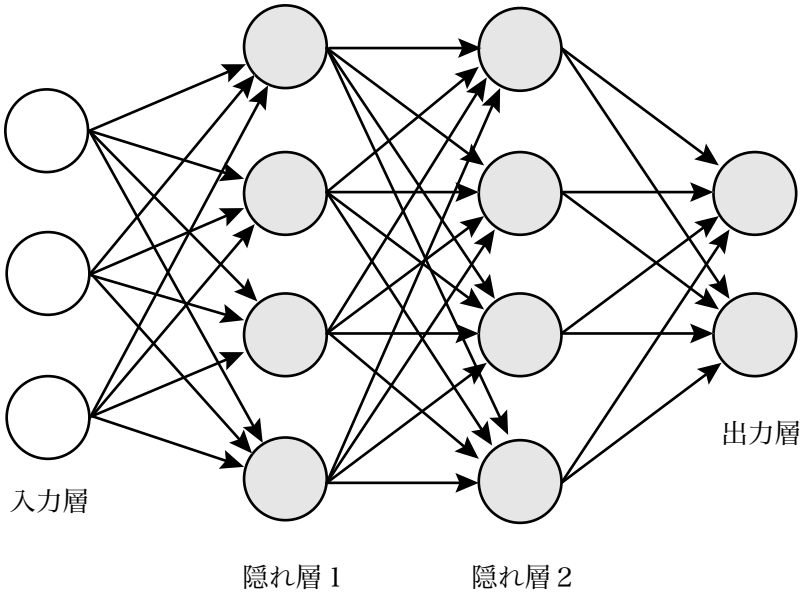

Fig. 15.1.2　隠れ層を増やしたニューラルネットワーク

１５−２　回帰問題と分類問題
Regression Problem and Classification Program

機械学習の分野では，人工神経網のような「機械」に解かせようとする問題に，**回帰問題** regression problem と**分類問題** classification problem があると言われます。回帰問題は解が連続的な数値として表される問題で，分類問題は解が離散的な数値として表される問題とも言われます。

ニューラルネットワークを使った機械学習では，回帰問題を解かせる場合には「なるべく正解に近い値を出すようにする」方針がとられ，分類問題を解かせる場合には「なるべく正解を出す確率が高くなるようにする」方針のとられる場合が多いようです（補足 15.2.A）。

ニューラルネットワークに回帰問題を解かせる場合に，正解が n 個の数値で表される（n 次元のベクトルを求めたい）なら，出力層のノード数を n 個に設定します。例えば Fig. 15.1.1 や Fig. 15.1.2 に示すケースでは，２つの数値で表現される２次元のベクトルを出力できます。この出力層ノードではシグモイド函数などの伝達函数（計算科学基礎 14-2 節）は使わず，人工ニューロンの前段部に相当する「重み付きの和」の値をそのまま出力します。「入力値をそのまま出力すること」を**恒等変換** identity transformation と呼び，ニューロンの働きでは後段部に相当する伝達函数として**恒等函数** identity function を用いると言い換えられることがあります。あまりニューロンらしくない働き方ですが，そのようなノードでも「ニューラルネットワーク」を使った機械学習の分野では「ニューロン」のうちに数えることが普通です。

ニューラルネットワークに分類問題を解かせる場合には，候補となるクラス（種類）class が n 種類なら，出力層の中のノード数を n 個に設定します。例えば 0 から 9 の「手書

き数字」の画像について「どの数字の意味か」特定したいのであれば，出力層に 10 個のノードを設置します。そして出力層のノードでは，「重み付きの和と伝達函数の組み合わせ」の代わりに**ソフトマックス函数** softmax function と呼ばれる函数を使うのが普通です。

１５−３　ソフトマックス函数　Softmax Function

ソフトマックス函数 softmax function は，形式的には多入力・1 出力の函数で，入力 $\{\mathbf{z}\} = \{z_j\}$，$(j = 1, \cdots, n)$ に対して，以下の数式で表されるような出力をします。

$$\sigma(\mathbf{z};\, k) = \frac{\exp(z_k)}{\sum_{j=1}^{n} \exp(z_j)} \tag{15.3.1}$$

ここで $\exp(x)$ は，オイラー数 Euler's number（自然対数の底；ネイピアの定数 Napier's constant とも）を e として，e^x を表し，**指数函数** exponential function と呼ばれる初等函数です。式 (15.3.1) は，出力値を y_k として，以下のように書いても同じことです。

$$\begin{aligned}
y_1 &= \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2) + \cdots + \exp(z_n)}, \\
y_2 &= \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2) + \cdots + \exp(z_n)}, \\
&\cdots, \\
y_n &= \frac{\exp(z_n)}{\exp(z_1) + \exp(z_2) + \cdots + \exp(z_n)}
\end{aligned} \tag{15.3.2}$$

実質的な意味としては入力 z_k をそのまま y_k として出力する恒等変換と大きく変わらないのですが，式 (15.3.1) あるいは式 (15.3.2) のような形式を用いれば，入力値 $\{z_j\}$ が正負・大小のどのような値をとったとしても，y_k の値は「常に正 $(y_k > 0)$」であり，なおかつ「和が 1 $(y_1 + y_2 + \cdots + y_n = 1)$」なので，出力 y_k に，「**確率** probability と同じ性格」を持たせることができます。

「確率と同じ性格を持つ」からと言って，それが「確率を表す」ことにはなりませんが，ニューラル・ネットワークがうまく働けば（学習できれば），本当に「正しさ」「確からしさ」を表す数値を出力できる可能性があります。これが分類問題にニューラル・ネットワークを使う時の出力層にソフトマックス函数が好んで用いられることの最も直接的な動機と思われます。

現実の数値計算システムを使ってソフトマックス函数を計算する場合には注意した方が良い面もあります（補足 15.3.A）。

１５−４　人工神経網の機械学習
Machine Learning of Artificial Neural Network

ニューラルネットワークに何か判断のようなことをさせる場合に，すべての人工ニューロンの持つすべての（バイアスを含む）重みパラメータ $\{w_{jk}\}$ について「良い判断結果」を出せるような値(あたい)を決めてやれば良いことになります。ただし，ニューラル・ネットワークの規模が大きくなれば，ヒトが理詰(りづ)めで最適な重みパラメータを決めることは事実上不可能になります。

ニューラルネットワークに「学習能力のようなもの」を持たせて，自動的に「良い結果を出すような重みパラメータ」を見つけさせる方法がなければ，実際には人工神経網は使い物になりません。

そのための方法として**教師あり学習** スーパーヴァイズド ラーニング supervised learning と呼ばれるやり方があります。このやり方は大量の（いろいろな）「問題と正解」が組となったデータ・セット（例題）を使って，なるべく正解を出す確率が高くなるように，あるいはなるべく正解に近い解答が出るように，ニューラル・ネットワークの含む「重みパラメータ」を調整します。このために使われるデータは**訓練データ** トレイニング デイタ training data あるいは**訓練セット** トレイニング セット training set と呼ばれます。

機械学習には**教師なし学習** アンスーパーヴァイズド unsupervised learning と呼ばれる分野もあり，「正解の付いていない問題」に相当(そうとう)するデータ・セットだけを与え，データが含む特徴を抽出させたり，データの間の類似性によって分類させたりします。

教師あり学習と教師なし学習は，手法としても違うのですが，機械に学習の結果として実現させようとする内容そのものが違います。「教師なし学習」の結果を何かの方法で評価して，修正させるようなプロセスを加えれば，機械が（半ば）自動的に正解を導くようにさせることも可能でしょうけれど，それはデータ・セットの解釈が違うだけの「教師あり学習」と同じこととも言えます。

補足 15

（補足 15.2.A）機械学習における回帰問題と分類問題（↵）

解が連続的な数値として表される問題であれば「なるべく正解に近い値を出すようにする」方針をとり，解が離散的な数値として表される問題であれば「なるべく正解を出す確率が高くなるようにする」方針をとるのは悪くない考え方と思いますが，この２つの方針は本質的に意味が違います。

機械学習の分野では，「なるべく正解に近い値を出すようにする」ために，実際には例えば「訓練データに付けられた正解」と「訓練データの入力に対して機械が出す出力値」との差の自乗の和を「損失函数」として，その値が最小になるように重みパラメータを調整する方法が使われます。これは L_2 ノルムを最小化する最小自乗法と呼ばれる方法と同じことです。「損失函数」の選び方によってロバスト推定もミニマックス推定もできます。

「なるべく正解を出す確率が高くなるようにする」のはモード（最頻値(さいひんち)）を代表値としてとることと同じように見えますが，訓練データの解答例も統計的な変動を含む性格を持つのが普通です。この場合，理想的に

最適化されたニューラルネットワークは，統計的な変動・確率分布までモデル化することになるでしょうから，単純に「なるべく正解を出す確率が高くなるようにする」のとは違う意味を持つことになるでしょう。(↩)

（補足 15.3.A）ソフトマックス函数について（↩）

現時点で普通に利用できる計算システムの多くは，数値を IEEE 規格の実数型（浮動小数点数 floating point number）として扱うことが前提となっています。指数函数 $\exp(x)$ は変数 x の値が大きいと急速に大きい数になり，式 (15.3.1)

$$\sigma(\mathbf{z};\, k) = \frac{\exp(z_k)}{\sum_{j=1}^{n} \exp(z_j)} \tag{15.3.1}$$

を使ってソフトマックス函数の値を求めようとすると，$\exp(x)$ の値が IEEE の浮動小数点数規格で表現できる数の範囲を超えてしまい，それ以上計算が進まない状況になる場合があります。

それを回避するために，例えば $\{z_j\}$ のうちの最大値を $z_{\max}$ として，

$$\sigma(\mathbf{z};\, k) = \frac{\exp(z_k - z_{\max})}{\sum_{j=1}^{n} \exp(z_j - z_{\max})} \tag{15.3.A.1}$$

のようにして計算する方法があります。 (↩)