特別掲載

データ活用の基本②

―深層学習の基礎知識（ニューラルネットワーク）―

研究員　山名　一史

目　次

1．はじめに

前回に引き続き、収集・蓄積されたデータを活用するための基本的な知識として、本稿ではニューラルネットワークについて概説する。ニューラルネットワークは、深層学習の基礎的な構成要素であり、画像認識や音声認識、推薦システムの設計や強化学習などの分析を行う上で不可欠な知識である。

本稿の目的は、具体的なデータ活用事例の紹介やデータ活用の是非に関する議論ではなく、データ活用の背後にある統計的な考え方を説明することである。なお、一般の読者を想定し、数式の利用は最小限にとどめた。

本稿の構成は以下の通りである。第２節では脳の学習機能を模倣する方法として、ニューラルネットワークについて概説し、第３節ではパーセプトロンとその学習について取り上げる。さらに、単層のパーセプトロンを重ね合わせることで、より複雑な計算を可能にする試みとして、第４節ではディープニューラルネットワーク、第５節では畳み込みニューラルネットワークを取り上げる。

2．ニューラルネットワークとは

データ活用の基本①で説明したように、顧客属性などの入力変数から入院リスクや保険加入率といった出力変数の値を予測したい場合、統計モデルを作成し、モデル内のパラメータを何らかの方法で推定し、モデルに基づいて定量的な予測を行う、というのが標準的な流れとなる。モデルを作成する際、線形モデルなどの単純な構造のモデルを用いて十分な予測性能が得られるのであれば、モデルをむやみに複雑化するべきではない。しかし、取り組む問題自体が複雑な場合、入力変数と出力変数との間に非線形性を伴った複雑な関係が存在している可能性が高い。こうした場合、単純な構造のモデルでは変数間の関係をうまく近似できず、十分な予測性能を得られないことが少なくないため、より複雑な構造のモデルを用いて分析や予測をする必要が生じる。

複雑な構造のモデルを扱う最大の問題点は、構造が複雑になるにしたがって、モデル内パラメータの推定も困難になることである。場合によっては、推定が不可能になることも少なくない。そのため、分析者としては分析で使うモデルをできるだけ単純な構造にとどめておきたいというのが本音である。それでは、こうした相反する要望を満たす方法はあるだろうか。以下では、こうした要望を満たす可能性のある方法として、脳の学習機能を模倣するという試みについて検討してみよう。

脳は脳細胞（神経細胞）から構成されており、細胞体と樹状突起および軸索から構成される脳細胞はニューロンと呼ばれる。軸索の末端はシナプスと呼ばれ、他のニューロンの樹状突起に近接している。ニューロンは他のニューロンからシナプスを介して樹状突起から電気信号を受け取り、軸索を通じてシナプスから他のニューロンに電気信号を放出する。この一連の情報伝達メカニズムは、樹状突起によって情報を「入力」し、細胞体によって情報を「処理」し、シナプスによって情報を「出力」する、という三段階のプロセスに分けることができる。

1943年、Warren S. McCullochとと Walter Pittsは、ニューロンの情報伝達機能を二項分類の処理系と捉え、McCulloch－Pittsニューロン（MCPニューロン）という単純な計算モデルで表現した。二項分類とは、集合を０か１かの２種類に分類することであることから、個々のMCPニューロンは１つ以上のバイナリ（０か１か）入力を受け取り、１つのバイナリ出力を行うことしかできないことを意味している。そのため、MCPニューロンで構成される個々のネットワークは、単純な論理演算しか行うことができない。

単純な論理演算の具体例として、ニューロン１と２がそれぞれニューロン３に近接している場合を考えてみよう。ニューロン１と２がどちらも１を出力した場合にのみニューロン３が活性化するようなネットワークを構成すると、論理積（AND）が実行可能になる。同様に、ニューロン１と２のどちらか一方（または両方）が１を出力した時にニューロン３が活性化するようなネットワークを構成すると、論理和（ＯＲ）が実行可能になる。

MCPニューロンで構成される個々のネットワークは単純な論理演算しかできない。しかし、個々のネットワークを組み合わせると、たとえば大脳皮質のように数百億個のニューロンによって巨大なネットワーク構造を組織すれば、個々のネットワークの単純な構造を維持しつつ、ネットワーク全体では高度かつ複雑な演算ができるだろう。このように、ニューロンが結合されたネットワークを用いて複雑な演算を行うようなモデルをニューラルネットワークという。

３．単層ニューラルネットワークの構造と学習

1957年、Frank RosenblattがMCPニューロンに基づいて提案した計算方法（アルゴリズム）がパーセプトロンという単層のニューラルネットワークである。入力はバイナリ値ではなく実数値をとり、個々の接続部には重み（パラメータ）が与えられる。この重み値は、どの信号を相対的に重視するか、その度合いを示すものと考えることができる。そして、受け取った入力値と重み値との内積（加重和）に基づいて、出力信号が生成される。

パーセプトロンの構造を、分かりやすく数式にしてみよう。受け取った入力値とそれに対応する重みのベクトルをそれぞれ$\boldsymbol{x}, \boldsymbol{w}$とする。細胞体はこの内積$\boldsymbol{x}^T\boldsymbol{w}$を計算し、内積がある基準の値（これをしきい値という）Tを超えた場合（$\boldsymbol{x}^T\boldsymbol{w} \geq T$）、信号が生成（発火・活性化）され、軸索を通じてシナプスから出力される。これら一連の働きは、

$$y = \begin{cases} 1 & \boldsymbol{x}^T\boldsymbol{w} \geq T \\ 0 & \boldsymbol{x}^T\boldsymbol{w} < T \end{cases}$$

という単純な数式で表現することができる。ここで、出力yは内積$\boldsymbol{x}^T\boldsymbol{w}$の関数なので、$y = \phi(\boldsymbol{x}^T\boldsymbol{w})$という関数（活性化関数や決定関数と呼ばれる）を導入すると分かりやすい。また、しきい値を左辺に移動して、$z \equiv \boldsymbol{x}^T\boldsymbol{w} - T$という変数を新たに定義すると、

$$\phi(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

のように、上記の関係を簡単に書きかえることもできる。しきい値の判定に用いる$-T$は信号が出力される程度やニューロンの発火あるいは活性化のしやすさを表す項であるが、この項はバイアス（ユニット）と呼ばれる。

パーセプトロンでは活性化関数としてステップ関数（zが０以上なら１、０未満なら０を出力するような関数）が用いられる。また、ニューラルネットワークでは（標準）シグモイド（ロジスティック）関数やReLU（Rectified Linear Unit）関数などの非線形関数もよく用いられる。

パーセプトロンモデルは内積の計算としきい値の判定という極めて単純な要素で構成されており、単一のモデルで表現できる関係は限定的である。しかし、MCPニューロンのところで議論したように、これらを何重にも重ねて（層を深くして）いけば、より複雑な関係を表現することができるだろう。また、個々の要素が単純であることを活かして、多層のネットワークをうまく訓練する方法を考えることができれば、モデルの推定を簡単にできるかもしれない。

まずは、単純なパーセプトロンモデルを訓練する方法から考えてみよう。パーセプトロンモデルにおいて、推定したいパラメータは、個々の入力$\boldsymbol{x}$をどの程度出力に反映させるかを表す重み$\boldsymbol{w}$なので、推定（学習・訓練）の対象はこの重み値となる。最初に、重み値をゼロまたは任意の小さい値で初期化し（$\boldsymbol{w} = \boldsymbol{w}^0$）、初期の重み値に基づいて出力を予測しよう：

$$\hat{y}^0 = \phi(\boldsymbol{x}^T\boldsymbol{w}^0 - T).$$

計算された予測値と現実の出力値とを比較すれば、当たりをつけた初期の重み値が的はずれなものかそうでないのかが分かる。次に、以下のようなルールにしたがって重み値を更新しよう：

$$\boldsymbol{w}^1 = \boldsymbol{w}^0 + \eta(y - \hat{y}^0)\boldsymbol{x}.$$

これは、予測値が外れていた場合（$y \neq \hat{y}^0$）は重み値を更新し、当たっていれば（$y = \hat{y}^0$）現在の重み値を引き継ぐというルールである。ここで、もし予測値が外れていた場合、重み値をどれくらい更新するか、その更新の程度を調節する（ハイパー）パラメータ$\eta \in (0,1]$は学習率と呼ばれる。学習率が大きすぎると、重み値が振動を続け、いつまでたっても更新が終わらないリスクが高くなる。一方、学習率が小さすぎると学習する速度が低下するため、リスクと速度を考慮したうえで適切な値を設定することが、実務的には重要となる。適切な学習率を設定し、出力比較と重み値更新のプロセスを十分に繰り返せば、予測性能を最大化するような重み値を推定できる。

パーセプトロンと同じ単層のニューラルネットワークとして、1960年にBernard WidrowとTedd Hoff（Marcian Edward Hoff Jr.）が発表したのがADALINE（ADAptive LInear NEuron）である。パーセプトロンが０または１という出力の正誤のみに基づいて重み値の更新を行っていたのに対し、ADALINEは入力値と重み値の内積を活性化関数とし、連続値に基づいて重み値の更新を行う点が最大の違いである：

$$\phi(z) = z = \boldsymbol{x}^T\boldsymbol{w}.$$

そのため、予測が当たったか外れたか、という質的な情報に加えて、予測値が実測値からどれくらい外れていたのか、という量的な情報を重み値の更新に利用できる点でパーセプトロンよりも学習効率に優れている。

これまでの議論から、重み値の学習には予

測値と実測値との乖離を小さくすることが重要であることがわかった。そこで、予測値と実測値との乖離を損失（コスト）と考えると、重み値の学習問題は損失最小化問題として定式化できることがわかる。こうした観点から、損失関数は微分可能であることが望ましい。なぜなら、損失最小化を行う際、関数の傾き（勾配）の情報を利用できるからである。この点、ADALINEは、損失関数を誤差平方和として定義できる：

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i}\left(y_i - \phi(z_i)\right)^2$$

ので、勾配降下法などの効率的な最適化ルーチンを適用することができる点に計算上の強みがある。他方、パーセプトロンで用いられるステップ関数は $z = 0$ で微分ができないため、効率的な最適化ルーチンを適用できないという弱みがある。

4．ディープニューラルネットワーク

単純なパーセプトロンを訓練する方法が分かったところで、単一のニューロンを多層に重ねた場合を考えてみよう。前項で述べたように、ニューロンを多層に重ねることで、ネットワーク全体でより複雑な関係を表現できるようにすることが最終的な目標である。たとえば、パーセプトロンを多層化したものは多層パーセプトロンと呼ばれる。パーセプトロンが入力層と出力層のみで構成されていたのに対し、多層パーセプトロンは層の重ね合わせによって入力層と出力層の間に１つ以上の隠れ層（中間層）を持つ点が最大の違いである。このように、隠れ層が１つ以上存在するネットワークをディープニューラルネットワークという。

ネットワークが多層になったところで、損失関数の最小化問題を解いて重み値を学習するという基本的な考え方は単層の場合と変わらない。しかし、多層化することによって損失関数は複雑な非凸関数になり、局所最小点を無数に含むようになるため、大域的な最小値を単純な最適化ルーチンで効率的に得ることは困難になる。

一般に、複雑な問題に対処するには、問題を単純なブロックに切り分けることが効果的なことが多い。幸運なことに、ディープニューラルネットワークは単純なネットワークが多層に重ね合わさった構造をしているので、切り分け自体は難しくない。複雑な損失関数を単純な要素に切り分ける方法としては、微分法における連鎖律が参考になる。

合成関数の導関数は、合成前の関数の導関数の積として表すことができる、というのが微文法における連鎖律であった。ディープニューラルネットワークの損失関数は、層ごとの損失関数を合成した関数なので、その導関数は層ごとの損失関数の導関数の積として表すことができる。このように、連鎖律に基づいて損失関数を切り分け、その勾配を効率的に計算する手法を自動微分という[1]。

自動微分を用いたからといって、必ず学習に成功するという保証はない。一般に、層が深くなるにつれて学習の難易度は高くなることが知られている。たとえば、層を追加するとともに勾配が小さくなり、学習が行き詰まる勾配消失の問題が典型的に知られている。ディープニューラルネットワークにおける勾配の不安定さに対処するためには特別なアルゴリズムを設計する必要があり、このように設計されたアルゴリズムはディープラーニングと総称される。

1　やや技術的な話をすると、自動微分には勾配を前からかけ合わせるフォワード型と後ろからかけ合わせるリバース型とがある。解析的には同型であるが、リバース型はフォワード型と異なり、行列どうしの乗算が不要なため、計算量の観点から有利であることが知られている。ニューラルネットワークの学習を行う際によく用いられるものとしてバックプロパゲーション（誤差逆伝播法）というアルゴリズムがあるが、名前からも分かる通り、これはリバース型の手法である。

5．畳み込みニューラルネットワーク

最後に、畳み込みニューラルネットワーク（Convolution Neural Network：CNN）を取り上げよう。CNNは人が物体を認識するときの視覚野の働きをヒントにしたモデルで、画像認識や音声認識などで用いられていることが多い。

統計的な予測を行う際は、予測に有用な入力変数を探し出すことが決定的に重要である。仮に適切な関数型が得られていたとしても、適切な入力がなければ、十分な予測性能は期待できない。入力変数を探し出すことは特徴量の抽出と呼ばれ、分析者が専門知識に基づいて恣意的に抽出するか、または何らかの特徴量抽出手法を用いるやり方が一般的である。この点、CNNはデータから特徴量を自動で学習する。

これまでに見てきた多層ニューラルネットワークでは、すべての入力変数が内積の計算としきい値の判定に用いられてきた。これは隣接する層のニューロンがすべて近接（連結）していることを仮定しており、こうした層は全結合層と呼ばれる。全結合層とは基本的に多層パーセプトロンを意味するので、すべての入力x_iが出力y_jに重みw_{ij}で結合されていることになる。全結合層はもっとも基礎的な結合層で扱いやすい。ただ、その構造からすべての入力を等しく扱わざるを得ないので、情報を処理する際に高次の情報が切り捨てられてしまうという問題がある。

高次の情報が切り捨てられる、という状況を考えるため、三次元座標の情報が入力された場合を考えよう。それぞれの数値の背後には縦・横・高さという情報が存在している。しかし、ここで内積を計算すると、座標に関する情報はすべて失われてしまう。もちろん、こうした高次の情報が学習に影響を与えないのであれば、どれだけ切り捨てても問題はない。しかし、たとえば画像認識を行う場合、座標が近いか遠いか、などの高次情報が学習に有用である可能性は否定できない。そのため、分析者としては、情報を処理する過程でこうした情報を可能な限り保持しながら分析を行いたい。このような動機から、畳み込みと呼ばれる演算を行う層をネットワークに導入し、高次の情報が保持されたディープニューラルネットワークがCNNである。

それでは、畳み込みとはどのような演算を指すのだろうか、１次元の畳み込み演算を使って具体的に確認しよう。ベクトル$\boldsymbol{x}$と$\boldsymbol{w}$との畳み込みは$\boldsymbol{y}=\boldsymbol{x}*\boldsymbol{w}$で表される。これまでと同様、$\boldsymbol{x}$は入力またはシグナルと呼ばれるのに対し、$\boldsymbol{w}$はフィルターやカーネルと呼ばれる。これは、「重み」に相当するものと考えておけばよいだろう。数学的に、畳み込み演算は

$$y_i=\sum_{k=-\infty}^{\infty}x_{i-k}w_k$$

と定義される。xは有限次元のベクトルなので、総和を計算するため、$\boldsymbol{x}$と$\boldsymbol{w}$のインデックスの外側は０などの定数で満たされていると仮定して計算を行う。このような処理はパディングと呼ばれており、０で満たすときはとくにゼロパディングと呼ばれる。実務上、パディングは出力サイズを調節するために用いられる。数式だけではイメージが湧きにくい読者もいると思うので、補論では具体的な計算例を紹介した。こちらも参照いただきたい。

最後に、畳み込みとともに多くのCNNで使われるプーリング、またはサブサンプリングとよばれる演算を紹介しておこう。いま、二次元の畳み込み演算によって以下のような入力が得られたとする。

$$\boldsymbol{x}=\begin{bmatrix}1&2&0&1\\1&0&2&3\\2&0&1&0\\1&1&1&0\end{bmatrix}$$

プーリング演算は、$n\times m$の領域を縮約する

処理で、代表的なものに最大値プーリングや平均値プーリングがあげられる。たとえば、2×2の最大値プーリングは、左上の$\begin{bmatrix}1 & 2\\ 1 & 0\end{bmatrix}$の最大値２、右上の$\begin{bmatrix}0 & 1\\ 2 & 3\end{bmatrix}$の最大値３、左下の$\begin{bmatrix}2 & 0\\ 1 & 1\end{bmatrix}$の最大値２、そして右下の$\begin{bmatrix}1 & 0\\ 1 & 0\end{bmatrix}$の最大値１を取り出した$P_{2\times2}=\begin{bmatrix}2 & 3\\ 2 & 1\end{bmatrix}$となる。プーリング演算は、入力の微小な変化に対して安定的な値を取るため、データに含まれるノイズに対して頑健な特徴量を抽出する方法である。

６．おわりに

本連載では、収集・蓄積されたデータを活用するための基本的な知識の概説を目的として、第一回では統計モデルの選び方、そして第二回では、より複雑な関係を記述するモデルとしてニューラルネットワークを扱った。

本連載で扱った内容に限らず、統計学や機械学習に関する知識を理解するためには、手を動かす、つまりモデルを実装して実際にデータ分析を行う必要がある。本連載を通じてデータ分析に興味をもった、あるいは実務プロセスにデータ分析を導入したいと思ったのであれば、小規模なプロジェクトでいいのでまずは始めてみることが大切だ。現実のデータ分析は、教科書の例のようにうまくはいかないが、試行錯誤することで得られるものは少なくないだろう。本連載の内容が、そうした取組みの一助となれば幸いである。

補論

畳み込み演算を具体的に理解するため、$\boldsymbol{x}=[1\,2\,3\,4\,5\,6],\boldsymbol{w}=\left[1\,\frac{1}{2}\,\frac{1}{3}\,\frac{1}{4}\right]$の畳み込み$\boldsymbol{x}*\boldsymbol{w}$を計算してみよう。シグナルとカーネルはインデックスを逆方向に参照するので、カーネルを回転して、$\boldsymbol{w}^r=\left[\frac{1}{4}\,\frac{1}{3}\,\frac{1}{2}\,1\right]$とする。次に出力要素ごとにシグナルと回転したカーネルとをかけ合わせていけば、

$$y_1=1\times\frac{1}{4}+2\times\frac{1}{3}+3\times\frac{1}{2}+4\times1$$

$$y_2=2\times\frac{1}{4}+3\times\frac{1}{3}+4\times\frac{1}{2}+5\times1$$

$$y_3=3\times\frac{1}{4}+4\times\frac{1}{3}+5\times\frac{1}{2}+6\times1$$

のように畳み込み演算を行うことができる。この例では、パディングを行っていないので、パディングサイズは０となる。また、カーネルはシグナル上を１ずつ移動している。この移動サイズは畳み込みのハイパーパラメータの一つで、ストライドと呼ばれている。

たとえば、ストライドが２のもとで、入力ベクトルのサイズを出力でも維持したい（つまり、６にしたい）場合を考えてみよう。このような場合は、

$$y_1=0\times\frac{1}{4}+0\times\frac{1}{3}+0\times\frac{1}{2}+0\times1$$

$$y_2=0\times\frac{1}{4}+0\times\frac{1}{3}+1\times\frac{1}{2}+2\times1$$

$$y_3=1\times\frac{1}{4}+2\times\frac{1}{3}+3\times\frac{1}{2}+4\times1$$

$$y_4=3\times\frac{1}{4}+4\times\frac{1}{3}+5\times\frac{1}{2}+6\times1$$

$$y_5=5\times\frac{1}{4}+6\times\frac{1}{3}+0\times\frac{1}{2}+0\times1$$

$$y_6=0\times\frac{1}{4}+0\times\frac{1}{3}+0\times\frac{1}{2}+0\times1$$

のように、出力サイズに応じてパディングサイズを４に設定すればよい。つまり、サイズが４のゼロパディングを行ったシグナル$\boldsymbol{x}=[0,0,0,0,1,2,3,4,5,6,0,0,0,0]$と回転したカ

ーネルとを掛け合わせた、という意味である。

（参考文献）

・斎藤康毅.『ゼロから作るDeep Learning－Pythonで学ぶディープラーニングの理論と実装』. オライリー・ジャパン. 2016.
・A. Géron.『scikit－learn、Keras、TensorFlowによる実践機械学習』. 第２版. オライリー・ジャパン. 2020.
・C. M. Bishop. 『パターン認識と機械学習 上・下』. 丸善出版. 2012.
・R. Sebastian, M. Vahid. 『Python機械学習プログラミング 達人データサイエンティストによる理論と実践』. 第３版. インプレス. 2020.
・T. Hastie, R. Tibshirani, J. Friedman. 『統計的学習の基礎 －データマイニング・推論・予測－』. 共立出版. 2014.