

3層型ニューラルネットワークによる学習と識別

《学修項目》

- 3層型ニューラルネットワーク(MLP)の構成
- 誤差逆伝播法による教師つき学習
- クラス分類
- 特徴量抽出

《キーワード》

ニューラルネットワーク、入力層、中間層、出力層、学習アルゴリズム、シグモイド関数、学習定数、活性化関数、重みの更新、誤差逆伝播法、教師つき学習、クラス分類、パターン認識、特徴量抽出

《参考文献、参考書籍》

- [1] 東京大学MIセンター公開教材 「AI基礎：3-4 深層学習の基礎と展望」 《利用条件CC BY-NC-SA》
- [2] 東京大学MIセンター公開教材 「AI基礎：3-5 認識」 《利用条件CC BY-NC-SA》

1.はじめに

ここでは、パターン認識分野の数多くの問題に対して、有効な結果を得ている[59] 教師つき学習のできる階層型誤差逆伝搬モデルについて解説する。特にここでは、簡単のために3層型のモデルを用いてみよう。以下に3層型ニューラルネットワーク (MLP : Multi-Layer Perceptron) の学習アルゴリズムの概略を示す。

ニューラルネットワークは、ある値の列を入力として与えた場合に、希望する値の列を出力する、ある種の関数であると考えることができる。例えば、犬猫判別ニューラルネットワークは、犬と猫の写真画像を入力として与えたら、犬であるか猫であるかの判別結果を出力として返してくれる関数である。

それでは、犬と猫を判別するニューラルネットワークを例にとって、どのようにネットワークを構成するのかを見ていこう。まず、値の列を入力としたいので、入力画像を縦横等分割し、区切られた領域（画素）の輝度値（白黒の濃淡値）を抽出し、それらを1列に並べて入力列としてみる（本来、画像から何らかの特徴量を抽出してから、入力列とする方法を採用するが、ここでは簡単のため、画像そのものを入力列としてみる）。

2.3層型ニューラルネットワークの構成

ニューラルネットワークは、図1に示すように、○印で示したユニットと呼ばれる部分とそれを結合する線で構成される。先ほどの入力画像から得られた数値列は、図1の最下層の○印に1つずつの値が渡される（例えば、入力値の数が10個なら、○の数も10個である）。この入力を与える最下層のことを入力層と呼ぶ。

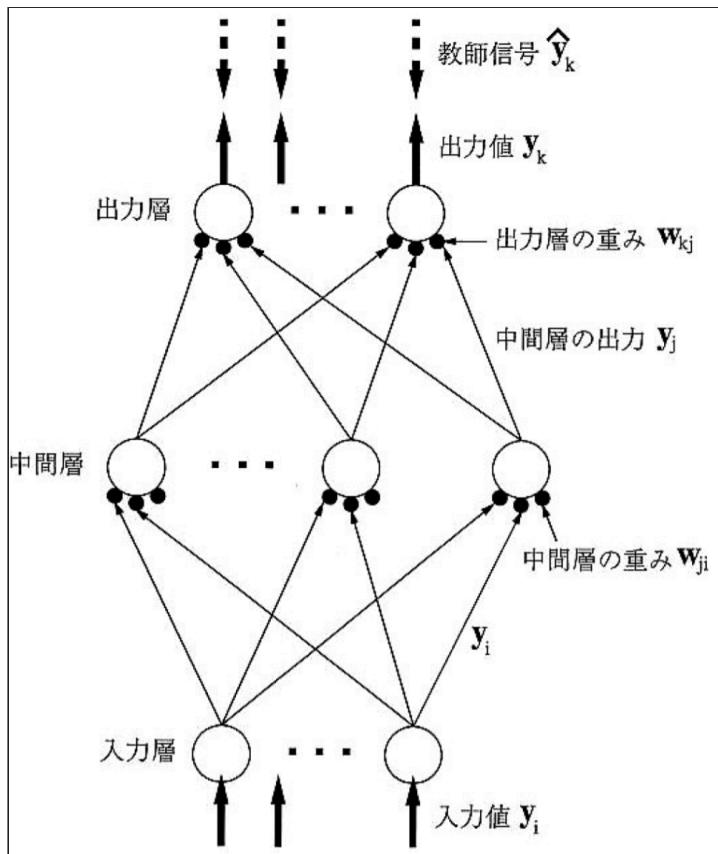


図1 3層型ニューラルネットワークの構成

入力層の値は、線で結合されている1段上のユニット（○印）に伝達される。その際、単純に伝達するわけではなく、その値に重みの値（図1では重みを●印で示した）を乗算して渡す。1つのユニットには、1段下のユニットから多数の値が伝達されるので、それらを足し合わせて、そのユニットの出力とする。これらを下の層から上の層に向かって伝達していく。その際、これらの伝達はユニット値と重みの積和計算のみで行われるため、線形の関数しか表現することができず、複雑な判別を行うことができない。

そこで、各ユニットの出力に活性化関数と呼ばれる関数を介して、非線形関数化することが行われる。このようにして、入力層から入力された値は、上の階層のユニットに向かって計算され、最上層で目的の値の列を出力する。ここで、最上層を出力層と呼び、入力層と出力層の間の層のことを隠れ層あるいは中間層と呼ぶ。なお、図1のように上下層の各ユニットが相互に全て結合している層のことを全結合層と呼ぶ。

それでは、次に、どのように出力層の値を目的の値に近づけるか考えてみよう。

今、犬と猫の判別をする場合、出力層のユニット数が2個であり、犬である場合は、1つ目のユニットの値が1、2つ目のユニットの値が0になり、一方、猫である場合は、それとは逆に、0と1の値になるようにしたいと考えよう。ある入力画像の値を入力層に与えた場合、上記のネットワークの伝達の計算を行うと、何らかの出力値がout層の2つのユニットに得られる。ここで、入力した画像の正解がわかっていると仮定しよう。つまり、犬であった場合は出力層の値は「1, 0」が正解、猫の場合は「0, 1」が正解である。このような正解データのことを教師信号と呼ぶ。

ニューラルネットワークでは、現在の出力値と教師信号との誤差を計算し（誤差関数）、その値が小さくなる方向に、ネットワークの重み（●印）の値を修正する作業を行う。この計算は、通常のネットワーク計算とは逆順に、つまり出力層から入力層に向かって、重みの値を修正するように行われるため、誤差逆伝播法と呼ばれている。

このような計算を大量の入力画像とそれに対する教師信号のデータを使って行い、犬猫判別ニューラルネットワークを構成する。これにより、構成したネットワークに、犬猫の画像を入力すれば、出力

層に、その結果を出力することができる（この例の場合、出力層でより大きな値を出力するユニットに相当するクラスを識別クラスとすれば良い）。

3. ニューラルネットワークの学習

それでは、ここまで説明してきた内容を、記号と数式を使って説明してみよう。

3.1 中間層、出力層

今、入力パターン $\{y_i | i = 1, 2, \dots, \text{入力値の数}\}$ をネットワークの入力層（図1を参照）に与えた場合、中間層の出力値 $\{y_j | j = 1, 2, \dots, \text{中間層のユニット数}\}$ は、次式で計算される。

$$y_j = f \left(\sum_i w_{ji} \bullet y_i \right) \quad (1)$$

ここで、 $f(x)$ は活性化関数(*1) と呼ばれ、本手法ではシグモイド関数 $f(x) = 1/(1 + \exp(-x))$ を使用する。また、 w_{ji} は中間層の重みの値を示しており、初期値は適当な値を代入しておく。

(*1) 活性化関数としては、シグモイド関数以外に、tanh、ReLU、Leaky ReLUなどの関数が使われる。

出力層の出力値 $\{y_k | k = 1, 2, \dots, \text{出力層のユニット数}(k_n)\}$ は、中間層同様、次式で求められる。

$$y_k = f \left(\sum_j w_{kj} \bullet y_j \right) \quad (2)$$

ここで、 w_{kj} は出力層の重みの値であり、中間層の場合と同じく、初期値は適当な値を代入しておく。

3.2 誤差逆伝搬アルゴリズム

次に出力値 y_k と実際に期待する値（教師信号） \widehat{y}_k の2乗誤差の総和を暫時小さくするために、誤差逆伝搬アルゴリズム (*2) により重みの更新を行う（具体的には、誤差関数の値が小さくなる方向に重みを修正する。この計算は誤差関数を重みパラメータで偏微分することによって得られるが、ここではその導出方法は割愛する）。出力層の重みの更新量は、 η を学習定数 (*3) とした場合、次式で計算する。

$$\Delta w_{kj} = \eta (\widehat{y}_k - y_k) y_k (1 - y_k) y_j \quad (3)$$

(*2) ここでは誤差関数として二乗和誤差を採用しているが、他にクロスエントロピー誤差を採用する場合もある。

(*3) 学習定数とは、学習の速度を決定する定数である。小さな値を設定すると学習の進行が遅くなり、局所最適解に収束してしまう可能性が出てくる。一方、大きな値に設定すると学習は速く進むが、最適解に収束しづらくなる。このため、最適な値を設定する必要がある。

中間層の重みの更新量は次式で計算する。

$$\Delta w_{ji} = \eta y_j (1 - y_j) y_i \sum_k (\widehat{y}_k - y_k) y_k (1 - y_k) w_{kj} \quad (4)$$

多数の入力パターン y_i と教師信号 \widehat{y}_k の組をネットワークに与え、上記のアルゴリズムに従い、出力値と教師信号の2乗誤差の総和が十分小さくなるまで、重みの更新を繰り返すことにより学習が行われる。

しかし、式(3)、(4)のままでは、重みの更新が極端過ぎて学習が不安定になることがある。そこで、ここでは過去の学習の慣性を利用した次の式を用いることとする (*4)。

$$\Delta w_{kj}(t+1) = \eta(\widehat{y}_k - y_k)y_k(1-y_k)y_j + \alpha\Delta w_{kj}(t) \quad (5)$$

$$\Delta w_{ji}(t+1) = \eta y_j(1-y_j)y_i \sum_k (\widehat{y}_k - y_k)y_k(1-y_k)w_{kj} + \alpha\Delta w_{ji}(t) \quad (6)$$

ここで α は安定化定数と呼ばれ、この値が大きい程過去の学習傾向を重視することになる。 t は学習回数を示す。

また、シグモイド関数 $f(x) = 1/(1 + \exp(-x))$ の f 軸の位置も学習の要素とするために、入力層と中間層には常に1を出力するユニットを用意した（これは、バイアス項として機能する）。

(*4) ここで用いた最適化アルゴリズムの方法は、Momentumと呼ばれる手法である。他に確率的勾配降下法(SGD)、AdaGrad、RMSProp、Adamなどの方法がある。

3.3 学習パラメータ、学習・識別アルゴリズムのまとめ

学習過程に必要となるパラメータをまとめると、次のとおりである。

- 中間層、出力層の重みの初期値 w_{int} (非対称な解も得られるように、通常は小さな乱数値を与える)
- 学習定数 η
- 安定化定数 α
- 中間層のユニット数

これらは、実験を通して最適な値を見つける必要がある。

以上をまとめると、学習アルゴリズムと識別アルゴリズムの概略は次のような形式となる。

学習アルゴリズム

1. 中間層と出力層の重みを乱数値によって初期化する。
2. 入力パターン(合計n個)の各パターンpについて、下記の処理を行なう。
 - i. パターンpを入力層に入力し、式(1)に従って中間層の出力値 y_j を求める。
 - ii. 式(2)に従って出力層の出力値 y_k を求める。
 - iii. 出力層の重みの更新を式(5)を用いて行なう。
 - iv. 中間層の重みの更新を式(6)を用いて行なう。
3. 出力ユニットの平均2乗誤差 式(7) が、ある設定値以上である場合は、処理2.に戻って繰り返す。設定値より小さい場合は終了。

$$\frac{1}{n} \sum_p \left\{ \frac{1}{k_n} \sum_k (\hat{y}_k - y_k)^2 \right\} \quad (7)$$

識別アルゴリズム

学習アルゴリズムで得られたネットワークを用いて、下記の処理を行なう。

1. 識別したいパターンを入力層に入力し、式(1)に従って中間層の出力値 y_j を求める。
2. 式(2)に従って出力層の出力値 y_k を求める。
3. 出力値に応じて、識別判定を行なう。

なお、識別判定は、そのままの出力値を用いて判定する場合(恒等関数による判別)とソフトマックス関数を用いて判定する場合などがある。

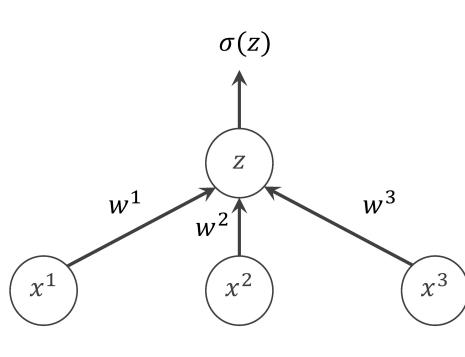
4. ニューラルネットワークの基礎理論、最適化問題 [1]

4.1 ニューラルネットワーク、結合層、多クラス分類問題（分類器）

ニューロン

ニューラルネットを構成する最小単位。

各ニューロンは接続されたニューロンから枝の重み w をかけ総和をとったのちに活性化関数 σ を適用し出力値を定めます。



$$z = w^\top x = \sum_{i=1}^d w^i x^i.$$

σ : 活性化関数.
ReLU, sigmoidと呼ばれる活性化関数などがよく用いられます。これらの活性化関数は次で定義されます。

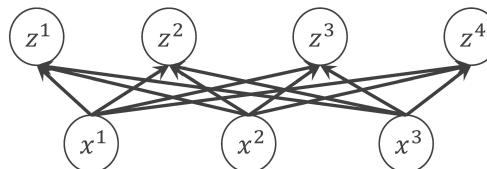
$$\sigma(z) = \begin{cases} \max\{0, z\} & (\text{ReLU}), \\ \frac{1}{1 + e^{-z}} & (\text{sigmoid}). \end{cases}$$

入力 $x = (x^1, x^2, \dots, x^d)^\top$.

重み $w = (w^1, w^2, \dots, w^d)^\top$.

層

ニューロンを並列に束ね層を構成します。



ニューロンを接続する枝には重みパラメータ W があります。

入力ニューロンの値を $x = (x^1, x^2, \dots, x^d)^\top$, 出力ニューロンの値を $z = (z^1, z^2, \dots, z^h)^\top$ とすれば, 重みパラメータ W は行数 h , 列数 d の行列であります。

この層による入力 x の変換は $z = \sigma(Wx)$ とかけます。

注意: ここで活性化関数 σ は全出力ニューロンに適用されます。

このような接続関係がある層は全結合層と呼びます。

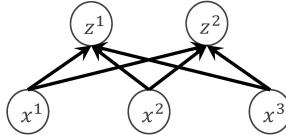
パラメータ W は基本的に経験損失最小化による学習で調整されます。

ロジスティック回帰

ロジスティック回帰の詳しい説明は1-4「データ分析」を参照。

多クラス分類問題に対する線形ロジスティック回帰は最も単純なニューラルネットワークの学習問題とみなせます。

クラス数を C としてラベルは $y = 1, \dots, C$ をとることにし入力データは $x = (x^1, x^2, \dots, x^d)^\top$ とします。ロジスティック回帰では行数 C , 列数 d のパラメータ W と C 次元のバイアスパラメータ $b = (b^1, b^2, \dots, b^C)^\top$ により $z = f_{W,b}(x) = Wx + b$ という変換をします。



ロジスティック回帰では入出力ペア (x, y) に対して次の損失関数を定めます：

$$l(f_{W,b}(x), y) = -\log \frac{\exp(z^y)}{\sum_{k=1}^C \exp(z^k)}.$$

これは一層のニューラルネットワークをモデルとし交差エントロピー損失を損失関数とする問題設定に他なりません。

4.2 機械学習系、最小化問題(損失最小化、最急降下法、確率的勾配下降法)

経験損失最小化

深層学習はニューラルネットワークモデルを用いた機械学習です。

ですので基本的に経験損失最小化問題を解きます：

$$\min_W \left\{ L(W) = \sum_{i=1}^n l(f_W(x_i), y_i) \right\}.$$

損失関数 l は二乗損失（回帰）や交差エントロピー損失（分類）を用います。

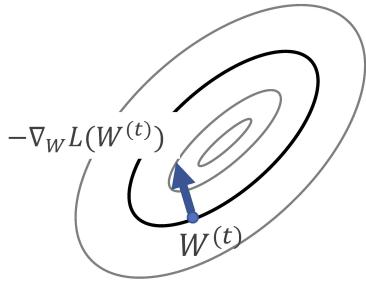
この最適化問題は確率的勾配下降法やその改良版であるADAMという手法で解くことが一般的です。

確率的勾配下降法は最急降下法を確率化したもので、機械学習において非常に有効な最適化手法です。

最急降下法

最急降下法は反復法の一種で、適当に決めた初期パラメータ $W^{(1)}$ から逐次的に以下の手続きでパラメータを更新します：

$$W^{(t+1)} = W^{(t)} - \eta \nabla_W L(W^{(t)}).$$



ここで $\nabla_W L(W^{(t)})$ は $W^{(t)}$ における経験損失関数 L の勾配です。 η はステップサイズや学習率とよばれるハイパーパラメータで、パラメータを変化させる幅を調整します。

勾配は関数の等高線と直交しますので負の勾配は関数を局所的に減少させる方向になります。

従って η がある程度小さいとき、最急降下法により損失関数が減少することが分かります。

ニューラルネットワークは行列による線形変換を繰り返すモデルでした。そのためパラメータについての勾配は微分における連鎖率というルールで計算されます。これは **深層学習と線形代数/微分積分との関係性** を指していて、深層学習を正しく理解するには微分積分や線形代数がある程度必要であるといえます。

確率的勾配降下法

経験損失関数はデータ数分の損失の和で定義されていました：

$$L(W) = \frac{1}{n} \sum_{i=1}^n l(f_W(x_i), y_i).$$

従って最急降下法のパラメータの更新規則は次のように書けます：

$$W^{(t+1)} = W^{(t)} - \frac{\eta}{n} \sum_{i=1}^n \frac{\partial l(f_W(x_i), y_i)}{\partial W}.$$

全データでの計算が必要で遅い

ここでパラメータの更新の度に全データ分の計算が必要となり非効率です。

そこで各 t において n 個のデータから乱択された 1 データ (x_{i_t}, y_{i_t}) のみを用いた近似的な勾配でパラメータを更新する手法が **確率的勾配降下法** です：

$$W^{(t+1)} = W^{(t)} - \eta \frac{\partial l(f_W(x_{i_t}), y_{i_t})}{\partial W}.$$

これにより計算効率が改善され大規模データに対する深層学習が可能となります。

memo