VII-2-3. ニューラルネットワークモデルによる判別分析

VII-2-3-1. モデルの構造

ソフトマックス関数を使うことによって、多クラスを同時判別できます。判別することだけが目的であればこれは十分実用的な方法でしょうが、これではまだこの章の最初に掲げた問題、図 96 のような多クラスの分布をどのように判別するかという問題に完全に答えていません。どの人種かを明瞭に判別するという目的からすれば、実用的にはそれで構わないのかもしれませんが、線が交差したときの解釈が難しいでしょう。もうすこし、本当の分布に近い曲線で表してみたくなります。この場合、説明変数が 2 次元平面上の存在し、求める係数は定数項も含めて 3 つです。$y = f(x_1, x_2)$という式はx_1とx_2の一次式にしかなりませんから、判別の境界は直線にならざるを得ません。曲線にして楕円の大きさ、歪み、傾きなどを表現するには推定する係数を増やさなくてはならないでしょう。そこで、図 118 に示したような 2 層のニューラルネットワークモデルを考えます。入力総和を変換する活性化関数は曲線ですから、それらを空間的に組み合わせた切断面は曲線になります。一層目はデータが信号として入る細胞の層（中間層）で、2 層目は最終的にあるクラスに属する確率と属さない確率を計算する出力層です。ですから出力層の神経細胞はクラスの数だけあります。このように、シグナルを次の階層に伝えていくモデルをフィードフォワードニューラルネット（FNN）と言います。FNN では中間層を何層でも作れますが、あまりたくさん作っても

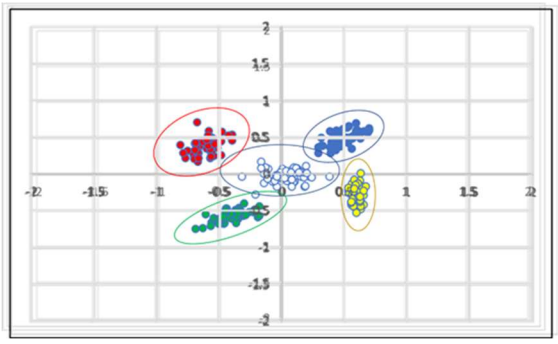

図 96（再掲）. 2 次元データの分布の事例

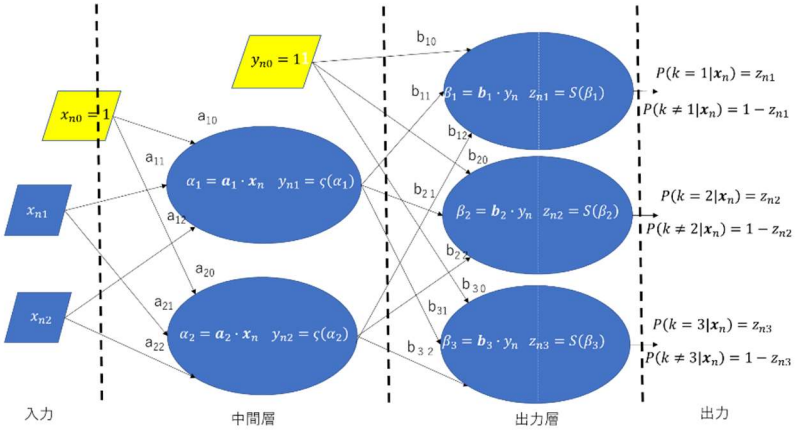

図 118. 3 クラス同時判別のニューラルネットワーク 2 層モデル

意味がないので、中間層は一つで全体を２層とし、中間層に２つの神経細胞を仮定します。出力層の細胞の役割はあるクラスである確率（あるクラスでない確率）を計算することです。中間層の役割はわかりにくいかもしれません。実際、中間層がなくても出力層の活性化関数をソフトマックス関数にすれば、多クラスの同時判別が出来ることはすでに確認済みです。中間層の役割は、入力されるデータに特定の視点から意味を与えることです。この表現は抽象的でよくわからないかもしれません。今まで例にしてきた人種別の体長と体重のデータを例に考えます。体長とか体重は cm とか kg とか数値で表される物理量ですが、数値はそのままでは意味を持ちません。太っているとか、ほっそりしているとか、背が高いというのが感覚的な意味です。味覚を例にすると、感覚を生み出しているのは食品に含まれる化学成分ですが、味蕾は成分分析だけをしているのではありません。成分は物理量ですが感覚は物理量ではありません。味蕾は化学物質の物理量的な刺激から感覚を作り出しているのです。基本的な味覚に甘味、塩味、旨味、酸味、苦味の５味がありますが、これらの食品に含まれる甘味成分のすべてを合計したものが実際に感じる甘味にはなりません。甘味と苦味、塩味と旨味のように、各味覚同士の干渉があったり、サッカリンのように低濃度では強い甘みを感じさせるのに、高濃度になると苦味を感じさせるものがあったりします。ニューラルネットワークモデルの第一層では、シグナルを一次結合してスカラーの入力総和を作り、これを活性化関数で変換して出力します。この作業は感覚を作る作業にあたります。この作業を空間的イメージとして捉えるために作ったのが図 119 です。３次元の図までしか書けないので図 119 はデータの項目がx_1, x_2の二つだけ（$D = 2$）ですが、一般的には説明変数項目はもっと多いでしょう。次のような手順で計算していると思ってください。まず、$0 = \boldsymbol{a} \cdot \boldsymbol{x}$という$D - 1$次元の境界平面を作ります。この平面と直交する軸が$\alpha_1, \alpha_2$で、境界平面からの距離です。与えられたデータと仮説的に与えた境界平面の一と傾き（境界平面の法線ベクトル）から距離α_1, α_2がきまり、この値を活性化関数で出力シグナルに変換します。条件を満たせば活性化関数には何を使っても良いのですが、シグモイド関数は良く使われる活性化関数の一つです。図 119 の垂直軸がy_nです。図は手書きなので線が多少凸凹していますがシグモイド関数のつもりです。このy_nの値が、次の層の細胞に出力されて、他の細胞でから入力する刺激とともに細胞が重み(b)を付けて一次結合して、　入力総和$\beta_1 \cdots \beta_K$となります。これが再び何らかの活性化関数で変換されて確率として出力されます。ここではこの活性化関数にソフトマックス関数を使います。

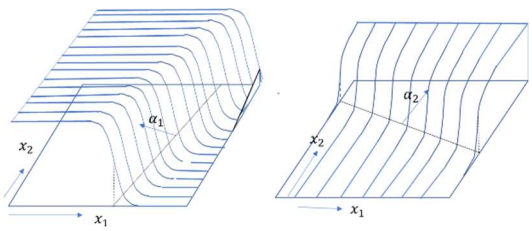

図 119. ニューロンで入力総和を作り、出力シグナルに変換する作業

VII-2-3-2. 誤差逆伝搬法(Back propagation)

モデルが決定し、すでに属するクラスがわかっているデータが与えられれば、データを使って、$\boldsymbol{a},\boldsymbol{b}$の重さ（係数ベクトル）を最適化するのですが、機械学習では、平均交差エントロピー誤差を最小化する$\boldsymbol{a},\boldsymbol{b}$を探すというやり方をします。$\boldsymbol{a},\boldsymbol{b}$で平均交差エントロピー誤差の微分が 0 になる$\boldsymbol{a},\boldsymbol{b}$を見つければ良いのですが、連立微分方程式を代数的に解くことができないので、適当に係数に初期値を与えて、順次更新していき微分値が 0 になるところ見つけるという勾配降下法を使います。これもいろいろなやり方がありますが、乱暴にやるのならば、初期値からちょっとずれた値を与えて交差エントロピー誤差を計算し、初期値の交差エントロピー誤差との値の差から、勾配を推測して更新する方向を探してずらしていくことを繰り返すいわゆる数値積分が考えられます。これだと大変時間がかかります。解析的にきちんと微分した方が計算速度は速くなります。これも様々なやり方があるようですが、良く知られているのは誤差逆伝播法（バックプロパゲーション）です。平均エントロピー誤差を$\boldsymbol{a},\boldsymbol{b}$の関数として書くと

$$E(\boldsymbol{a},\boldsymbol{b}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\log_e z_{nk}$$

これを、 n 番目のデータセットだけに着目して書くと、

$$E(\boldsymbol{a},\boldsymbol{b}) = \frac{1}{N}\sum_{n=1}^{N} E_n(\boldsymbol{a},\boldsymbol{b})$$

$$E_n(\boldsymbol{a},\boldsymbol{b}) = -\sum_{k=1}^{K} t_{nk}\log_e z_{nk}$$

となります。バックプロパゲーションの意味と手順を理解するために、中間層の一つの細胞(細胞 j)、出力層の一つの細胞を経る経路(図 120)で、FNN シグナルがどのように変換されるのかを具体的に数式で追います。

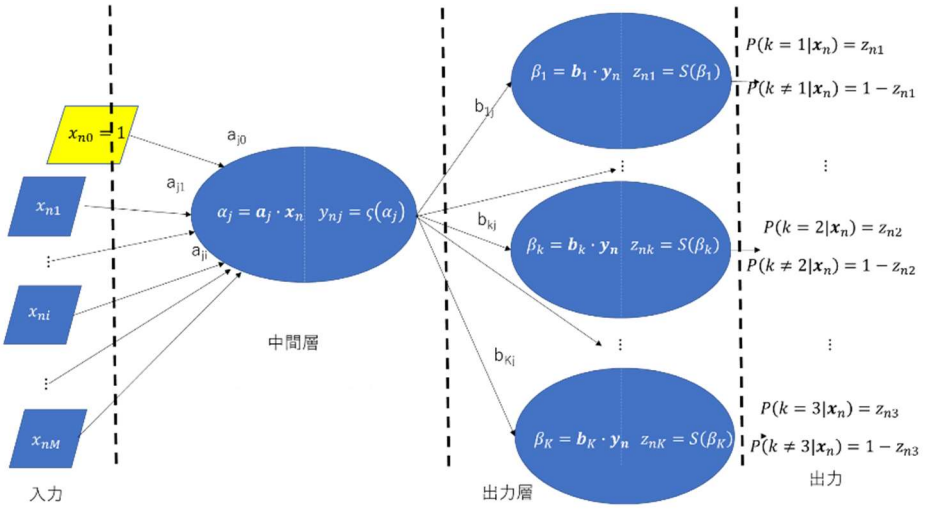

図 120. 中間層（第 1 層）の一細胞からのシグナルの流れ

第一層

$$\alpha_{nj} = \sum_{j=0}^{M} a_{ji}\, x_{ni} = \overrightarrow{\boldsymbol{a}_j} \cdot \overrightarrow{\boldsymbol{x}_n} \qquad \text{（細胞 j での入力総和）}$$

$$y_{nj} = \varsigma(\alpha_{nj}) \qquad \text{（細胞 j からの出力シグナル）}$$

次のステップで K 個の細胞に出力されますが、出力層の細胞 k のルートを追います。

$$\beta_{nkj} = b_{kj} y_{nj}$$

β_{nkj}はβ_{nk}の中でy_{nj}に由来する部分

$$z_{nkj} = S(\beta_{nkj})$$

$$E_{nk} = -t_{n1} \log_e z_{n1j} - \cdots - t_{nK} \log_e z_{nKj}$$

$$= -\sum_{k=1}^{K} t_{nk} \log_e z_{n1K}$$

となります。つまり、1 経路を追うと入れ子になった 4 つの関数の複合関数として、交差エントロピー誤差が計算されるということです。連鎖律を使って、E_nの$\boldsymbol{a}, \boldsymbol{b}$による偏微分をします。

$$\frac{\partial E_{nkj}}{\partial a_{ji}} = \frac{\partial E_{nkj}}{\partial z_{nkj}} \frac{\partial z_{nkj}}{\partial \beta_{nkj}} \frac{\partial \beta_{nkj}}{\partial y_{nj}} \frac{\partial y_{nj}}{\partial \alpha_{nj}} \frac{\partial \alpha_{nj}}{\partial a_{ji}}$$

$$\frac{\partial E_{nk}}{\partial b_{kj}} = \frac{\partial E_{nkj}}{\partial z_{nkj}} \frac{\partial z_{nkj}}{\partial \beta_{nkj}} \frac{\partial \beta_{nkj}}{\partial b_{kj}}$$

2 つの式の右辺の前から 2 番目までの偏微分は共通しています。最初の微分は対数の微分で二番目の偏微分は活性化関数の微分です。この例ではソフトマックス関数を活性化関数にしています。この二つは

$$\frac{\partial E_{nkj}}{\partial \beta_{nkj}} = \frac{\partial E_{nkj}}{\partial z_{nkj}} \frac{\partial z_{nkj}}{\partial \beta_{nkj}}$$

とまとめることが出来ます。右辺の第一番目の偏微分は以下の通りです。

$$\frac{\partial E_{kn}}{\partial z_{nkj}} = -\sum_{k=1}^{K} \frac{t_{nk}}{z_{nkj}}$$

2 番目のソフトマックス関数の微分はすでに解説しましたが、以下の通りです。

$$\frac{\partial z_l}{\partial \beta_{nk}} = z_{nl}(1 - z_{nl})$$

$$\text{when } \ l = k$$

$$\frac{\partial z_{nl}}{\partial \beta_{nk}} = -z_{nk} z_{nl}$$

$$\text{when } \ l \neq k$$

以上より

$$\frac{\partial E_{nj}}{\partial \beta_{nkj}} = \frac{\partial E_{nj}}{\partial z_{nkj}} \frac{\partial z_{nkj}}{\partial \beta_{nkj}} = \frac{\partial}{\partial \beta_{nk}} (-t_{n1} \log_e z_{n1} - \cdots - t_{nl} \log_e z_{nl} - \cdots - t_{nK} \log_e z_{nK})$$

$$= \left(-t_{n1}\frac{1}{z_{n1}}\frac{\partial z_{n1}}{\partial \beta_{nk}} - \cdots - t_{nk}\frac{1}{z_{nk}}\frac{\partial z_{nk}}{\partial \beta_{nk}} - \cdots - t_{nK}\frac{1}{z_K}\frac{\partial z_{nK}}{\partial \beta_{nk}}\right)$$

$$= \left(t_{n1}\frac{1}{z_{n1}}z_{nk}z_{n1} + \cdots - t_{nk}\frac{1}{z_{nk}}z_{nk}(1-z_{nk}) + \cdots + t_{nK}\frac{1}{z_{nK}}z_{nk}z_{nK}\right)$$

$$= (t_{n1}z_{nk} + \cdots - t_{nk}(1-z_{nk}) + \cdots + t_{nK}z_{nk})$$

$$= (t_{n1}z_{nk} + \cdots + t_{nk}z_{nk} + \cdots + t_{nK}z_{nk} - t_{nk})$$

$$= (t_{n1} + \cdots + t_{nk} + \cdots + t_{nK})z_{nk} - t_{nk}$$

$$= z_{nk} - t_{nk}$$

$$\because t_{n1} + \cdots + t_{nk} + \cdots + t_{nK} = 1$$

結論的には、

$$\frac{\partial E_{nk}}{\partial \beta_{nk}} = z_{nk} - t_{nk}$$

という極めてシンプルな形になります。この結論そのものは計算しなくても予測可能です。クラスがわかっているので、t_{nk}はデータがクラスｋに属していれば１、属していなければ０なのですから、クラスkに属するデータについて計算した確率z_{nk}は実際の値１よりも$|1-z_{nk}|$だけ過小評価で、クラスkに属さないデータについて計算した確率z_{nk}はz_{nk}だけ過大評価だというだけのことです。つまり、第二相の細胞ｋの予測値の誤差なのです。これを誤差$\delta_k^{(2)}$とします。

$$\delta_k^{(2)} = \frac{\partial E_{nk}}{\partial \beta_{nk}} = z_{nk} - t_{nk}$$

これは最尤法で交差エントロピー誤差を誤差とした場合です。ちなみに最小二乗法を用いると、誤差関数は

$$\delta_k^{(2)} = \frac{\partial E_{nk}}{\partial \beta_{nk}} = (z_{nk} - t_{nk})h(\beta_{nk})$$

$h(\beta_{nk})$は出力層の活性化関数

となります。もとにもどって

$$\frac{\partial E_{nk}}{\partial b_{kj}} = \frac{\partial E_{nkj}}{\partial z_{nkj}}\frac{\partial z_{nkj}}{\partial \beta_{nkj}}\frac{\partial \beta_{nkj}}{\partial b_{kj}} = \frac{\partial E_{nk}}{\partial \beta_{nk}}\frac{\partial \beta_{nkj}}{\partial b_{kj}} = \delta_k^{(2)}\frac{\partial \beta_{nkj}}{\partial b_{kj}}$$

で、β_{nkj}の式は

$$\beta_{nkj} = b_{kj}y_{nj}$$

だから

$$\frac{\partial \beta_{nkj}}{\partial b_{kj}} = y_{nj}$$

$$\frac{\partial E_{nk}}{\partial b_{kj}} = \delta_k^{(2)}y_{nj}$$

となります。つまり、E_nの重さ係数b_{kj}は、y_{nj}を重さとして、誤差$\delta_k^{(2)}$を振り分けた値にな

ります。したがって、勾配法の次のステップでのb_{kj}の更新値は

$$b_{kj}(\tau+1) = b_{kj}(\tau) - lr\frac{\partial E_{nk}}{\partial b_{kj}} = b_{kj}(\tau) - lr\delta_k^{(2)} y_{nj}$$

$$lr: \textit{学習率}$$

となります。

次に、第１層の係数∂a_{ji}による、下記の偏微分について考えます。

$$\frac{\partial E_{nkj}}{\partial a_{ji}} = \frac{\partial E_{nkj}}{\partial z_{nkj}}\frac{\partial z_{nkj}}{\partial \beta_{nkj}}\frac{\partial \beta_{nkj}}{\partial y_{nj}}\frac{\partial y_{nj}}{\partial \alpha_{nj}}\frac{\partial \alpha_{nj}}{\partial a_{ji}}$$

右辺の１番目と２番目の偏微分が、$\frac{\partial E_{nk}}{\partial \beta_{nk}} = \delta_k^{(2)} = z_{nk} - t_{nk}$であることは上記の説明の通りです。$\beta_{nkj} = b_{kj} y_{nj}$だから、

$$\frac{\partial \beta_{nkj}}{\partial y_{nj}} = b_{kj}$$

で、

$$\frac{\partial E_{nkj}}{\partial y_{nj}} = \frac{\partial E_{nkj}}{\partial z_{nkj}}\frac{\partial z_{nkj}}{\partial \beta_{nk}}\frac{\partial \beta_{nkj}}{\partial y_{nj}} = \delta_k^{(2)} b_{kj}$$

$$\frac{\partial E_{nkj}}{\partial a_{ji}} = \delta_k^{(2)} b_{kj}\frac{\partial y_{nj}}{\partial \alpha_{nj}}\frac{\partial \alpha_{nj}}{\partial a_{ji}}$$

ですが、kは１からＫまでＫ個あります。E_{nj}はデータ$\boldsymbol{x}_n$の対数尤度ですから、E_{nj}の全変化量は１からＫまでの細胞による変化量の総和だから、

$$\frac{\partial E_{nj}}{\partial a_{ji}} = \left(\sum_{k=1}^{K}\frac{\partial E_{nkj}}{\partial y_{nj}}\right)\frac{\partial y_{nj}}{\partial \alpha_{nj}}\frac{\partial \alpha_{nj}}{\partial a_{ji}}$$

です。$\frac{\partial y_{nj}}{\partial \alpha_{nj}}$は活性化関数の偏微分ですから、活性化関数を$h(\alpha_{nj})$と表せば、

$$\frac{\partial E_{nj}}{\partial \alpha_{nj}} = \left(\sum_{k=1}^{K}\delta_k^{(2)} b_{kj}\right)\frac{\partial y_{nj}}{\partial \alpha_{nj}}$$

$$\frac{\partial y_{nj}}{\partial \alpha_{nj}} = h'(\alpha_{nj})$$

$$\frac{\partial E_{nj}}{\partial \alpha_{nj}} = \left(\sum_{k=1}^{K}\delta_k^{(2)} b_{kj}\right) h'^{(\alpha_{nj})}$$

出力層（第二層）にならって、これを誤差とすると細胞ｊにおける誤差は

$$\delta_j^{(1)} = h'^{(\alpha_{nj})}\sum_{k=1}^{K}\delta_k^{(2)} b_{kj}$$

と表せます。つまり、偏微分$\frac{\partial E_{nk}}{\partial \beta_{nk}}$は誤差になるので、これを誤差として、線形結合を反対向

きにたどって、出力層の細胞 k における j 細胞からのシグナルの寄与分を計算し、これを逆方向に中間層の細胞 j に送るという作業を k＝1 から k ＝K までのすべての細胞について合計し、細胞 j に伝播するというのが、誤差逆伝搬法(back propagation)です。
細胞 j では、

$$\alpha_{nj} = \sum_{i=0}^{M} a_{ji} x_i$$

だから、

$$\frac{\partial \alpha_{nj}}{\partial a_{ji}} = x_i$$

したがって、

$$\frac{\partial E_{nj}}{\partial a_{ji}} = \delta_j^{(1)} x_{ni}$$

となります。以上をまとめて、勾配法の次のステップでの第 1 層a_{ji}の更新値は

$$a_{ji}(\tau+1) = a_{ji}(\tau) - lr\frac{\partial E_{nj}}{\partial a_{ji}} = a_{ji}(\tau) - lr\delta_j^{(1)} x_{ni}$$

lr: *学習率*

なお、活性化関数として、シグモイド関数をつかうと、

$$h(\alpha_{nj}) = \varsigma(\alpha_{nj}) = \frac{1}{1+e^{-\alpha_{nj}}}$$

$$h'^{(\alpha_{nj})} = \varsigma'^{(\alpha_{nj})} = \varsigma(\alpha_{nj})\left(1-\varsigma(\alpha_{nj})\right)$$

$$\delta_j^{(1)} = \varsigma(\alpha_{nj})\left(1-\varsigma(\alpha_{nj})\right)\sum_{k=1}^{K}\delta_k^{(2)} b_{kj}$$

となります。
データセットの数は N セットで、定数項を作るためのダミー変数$x_{n0} = 1$を含めて、
次のような配列になっていたとします。$\boldsymbol{T}$は各クラスの所属で、one-hot エンコーディングで書かれています。

$$\boldsymbol{x} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \vdots & x_{NM} \end{pmatrix}, \qquad \boldsymbol{T} = \begin{pmatrix} t_{11} & \cdots & t_{1K} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{NK} \end{pmatrix}$$

まとめると、誤差逆伝播法による勾配降下法の作業手順は次のようになります。
1．$\boldsymbol{a}, \boldsymbol{b}$の初期値$a(\tau), \boldsymbol{b}(\tau)$を適当に決める。

$$a(\tau) = \begin{pmatrix} a(\tau)_{10} & a(\tau)_{11} & \cdots & a(\tau)_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ a(\tau)_{M0} & a(\tau)_{M1} & \vdots & a(\tau)_{MM} \end{pmatrix}, \boldsymbol{b}(\tau) = \begin{pmatrix} b(\tau)_{10} & b(\tau)_{11} & \cdots & b(\tau)_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ b(\tau)_{K0} & b(\tau)_{K1} & \vdots & b(\tau)_{KM} \end{pmatrix}$$

2．$\boldsymbol{x}_n$は$\boldsymbol{x}_n = (1 \quad x_{n1} \quad \cdots \quad x_{n1})$となるベクトルで、$n = 1$から、$n = N$までの$\boldsymbol{x}_n$について、

$E_n(\boldsymbol{a}(\tau), \boldsymbol{b}(\tau), \boldsymbol{x}_n)$を計算する。この過程で、$\alpha_{nj}, y_{nj}, \beta_{nk}, z_{nk}$が得られる。

3．$\delta_j^{(1)}$ $\delta_k^{(2)}$　　$\alpha_{nj}, y_{nj}, \beta_{nk}, z_{nk}$から$\delta_{nj}^{(1)}$ $\delta_{nk}^{(2)}$を求める。

4．$\frac{\partial E_{nj}}{\partial a_{ji}} = \delta_{nj}^{(1)} x_{ni}$, $\frac{\partial E_{nk}}{\partial b_{kj}} = \delta_{nk}^{(2)} y_{nj}$を求める。

5．平均交差エントロピー誤差を求める。

$$\frac{\partial E_j}{\partial a_{ji}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E_{nj}}{\partial a_{ji}} = \frac{1}{N}\sum_{n=1}^{N}\delta_{nj}^{(1)} x_{ni}, \quad \frac{\partial E_k}{\partial b_{kj}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E_{nk}}{\partial b_{kj}} = \frac{1}{N}\sum_{n=1}^{N}\delta_{nk}^{(2)} y_{nj}$$

6．$\boldsymbol{a}, \boldsymbol{b}$を更新して、1 に戻る。

$$a_{ji}(\tau+1) = a_{ji}(\tau) - lr\frac{\partial E_j}{\partial a_{ji}}$$

$$b_{kj}(\tau+1) = b_{kj}(\tau) - lr\frac{\partial E_k}{\partial b_{kj}}$$

lr: *学習率*

戻ったステップ 1 では、初期値は

$$a_{ji}(\tau) = a_{ji}(\tau+1)$$
$$b_{kj}(\tau) = b_{kj}(\tau+1)$$

これを、すべての$\frac{\partial E_j}{\partial a_{ji}}$, $\frac{\partial E_k}{\partial b_{kj}}$が近似的に 0 になるまで繰り返す。

以上が誤差逆伝搬法による極値の探索法です。

VII-2-3-3. 2 層ニューラルネットワークモデルによる多クラス判別分析

多クラス判別分析をエクセルでやる人はまずいないでしょう。解析ソフトを使うか、自分でプログラムを書いて計算するでしょう。自分でプログラミングをするにしても、最近は無償で様々なプログラムがライブラリーとして公表されていますから、それらを利用することも可能です。ネットで探せば無料で利用できるプログラムがどこかに落ちています。筆者はこの前まで R を使っていたのですが、この章を書くために Python を勉強しました。機械学習ならば Python で使いやすい様々なライブラリーが提供されていると聞いたからです。Keras は Python で書かれた TensorFlow 上で動く、ニューラルネットワークモデル作り実行するためのライブラリーで、簡単に多層のモデルが作れて、計算の実行速度も速く、どのような計算をしているのかはブラックボックスですが、誤差逆伝搬法で計算するよりも１０倍ぐらい計算速度が速くなります。実用的には Keras を使えばよいのですが、何かの参考になることを期待して、Keras を使ったプログラムの前に、数値微分による計算と、解析的微分を使った誤差逆伝搬法による計算プログラムも作りました。これらのプログラムのコードリストを、このブログのカテゴリー「やさしい水産学(自習室)」の「参考資料」-「python」にあげておきました。数値微分は「VII-2-3-1.NNM による判別分析（数値微分)」、解析的

方法は「VII-2-3-2.NNM による判別分析（解析微分）」、Keras を使う方法は「VII-2-3-3.NNM による判別分析（keras）」という名前のコードリストです。計算時間もわかるようにしておきました。使ったサンプルデータは「Sample data」にあります。

数値微分でも解析微分でも Keras でも、リストの最初のリスト（i）csv データの読み込みです。ここではデータをエクセルで整理して csv で保存することを想定しています。このリストの後半に、#データをテストデータとトレーニング用のデータに分割して保存というコードがあります。データを全部使うのではなくてデータの一部を使うことを想定しているのです。TrainingRatio はデータの抽出率で、デフォルトは 1 になっていて、すべてのデータを使うことになっています。TrainingRatio を 0.5 にすれば全データの前半半分を使うことになります。こんなことをする理由は 2 つあります。一つは時間の節約です。Sample2 のデータサイズは 250 です。後から出てきますが、数値微分でやった場合、勾配降下法を 1000 ステップ実行すると私のパソコンで 15 分ぐらいかかりました。TrainingRatio を 0.5 にすれば、データサイズが半分になって計算時間が短くなります。解析微分でやれば 1 分ぐらいで終了しますし、keras ならば数秒で終わりまが、機械学習が想定している分析のデータサイズはもっと大きいはずです。その場合、Keras でやってもかなり時間がかかるでしょう。ここここで例として取り上げてる 2 層モデルは極めて単純でモデルの形を複雑にする必要はありませんが、3 層以上のモデルの場合、各層の細胞数をどのようにするのか、そもそも、フィードフォワードモデルが適当なのか等々様々な問題が出てきて、モデルの形を試行錯誤的に試すことになります。そのような場合に、すべてのデータを使っていちいち分析するのは時間の浪費です。一部のデータを抽出して、それを training data として分析することを考える必要があります。もう一つの理由は、オーバーフィッティングの有無を確認するためです。オーバーフィティッティングとは、重回帰分析などで、偶発的に起きる変動を意味のある変動と誤解して、説明変数を無意味に増やしてしまうことです。オーバーフィッティングが起きているか否かは、Training data と test data を入れ替えて結果を比較するとか、training data の分析結果から得られた係数を使って、test data での確率を予測して、要求するレベルで正答率を確認すれば分かります。

２番目のリスト（ii）は選んだデータの分布の確認です。ここも 3 つのプログラムで共通です。つづいて、モデルの形を決めて一つ一つのプロセスの計算式を指定します。この部分(iii)も数値微分と、解析的微分で numerical と analytical では違いがありません。keras ではモデルの形を実行段階で指定しますので、リスト(iii)は必要なライブラリーのインポートだけになっています。数値微分と、解析的微分では次の段階（iv）で微分の仕方を指示しますが、当然、そのリストの内容は異なります。計算の実行段階（VII-2-3-1-v, VII-2-3-2-v, VII-2-3-3-iv）で計算を実行し係数を推定します。数値微分と解析微分では勾配降下法を定義して実行させますが、計算途中で One step ごとに交差エントロピー誤差を記録しています。デフォルトでは学習率 LR を 0.5 とし、繰り返し計算を 1000 ステップ行うことにして、N_step を 1000 にしています。どのプログラムでも、計算経過がわかるように一回の試行ごとに交

差エントロピー誤差を計算しています。Keras では「#Sequantial モデルの作成」でニューラルネットワークモデルの形を指定します。この例では、model=Sequential()のところで、モデルの形として Sequential を指定しています。Sequential は層が単純に積み重なっていて、一つの層に 1 組の配列が入力し、次の層に一組の配列が出力される構造という意味です。
model.add(Dense(M,input_dim=M,activation='sigmoid',kernel_initializer='uniform'))　は第一層の指定で、細胞数に M、入力データの次元 M、活性化関数にシグモイド関数、初期値に一様の値を与えることを指定しています。次の
model.add(Dense(K,activation='softmax',kernel_initializer='uniform'))では、第二層に細胞数 K、活性化関数にソフトマックス関数を使い、初期値に一様値を指定しています。次の
sgd=keras.optimizers.SGD(lr=0.5,momentum=0.0,decay=0.0,nesterov=False)では、最適化の方法として確率的勾配降下法 SGD(stochastic gradient descent)を指定しています。SGD は勾配降下法の一つで、全微分して最大斜度の方向係数を移動する最急降下を改良して、データの一部の微分値を確率的に使って最急降下法をする方法で、計算が早く局所的な最小値に陥りやすいという最急降下法の弱点を補って、早く確実に収束点を見つけます。ここで学習率 lr も指定します。次に
model.compile(optimizer=sgd,loss='categorical_crossentropy',metrics=['accuracy'])　で　、optimizer に sgd を渡し、交差エントロピー誤差と正誤率を記録します。
実行の段階では、
history=model.fit(X_train,T_train,epochs=1000,batch_size=100,verbose=0,validation_data=(X_test,T_test))のところで、計算回数(epoch)を 1000、一回に計算するデータ数（batch size を 100）にしています。
VII-2-3-1-vi、VII-2-3-1-vii、VII-2-3-1-viii、VII-2-3-2-vi、VII-2-3-2-vii、VII-2-3-3-v、VII-2-3-3-vi は途中経過と結果の図示です。VII-2-3-1-ix、VII-2-3-2-viii、VII-2-3-3-vii は推定された重さ（係数）の取得と出力です。Keras では、第一層の係数は model.layers[0]に、第二層の係数は　model.layers[1]に入っています。これを l1、l2 に格納します。変数の係数はそれぞれついての層について Ax=(l1.get_weights()[0])、Bx=(l2.get_weights()[0]）で得られ、定数項は A0=(l1.get_weights()[1]）、B0=(l2.get_weights()[1])で得られます。
それぞれの計算法で得られた結果を比較します。表 66 は推定された係数の表です。analytical と numerical は全く同じ値です。数値微分と解析的微分で微分の仕方は違いますが、微分値が大きく異なることはないでしょう。勾配降下法が全く同じだから、またっく同じ値になったとしてもそんなに意外なことではありません。keras は勾配降下法が違います。普通は keras の方が早くより確実に収束に向かいます。図 121 に analytical で計算した時と keras で計算したときの平均エントロピー誤差の変化を示しました。Keras の方が速く収束に向かうっことがわかります。また、keras では平均交差エントロピー誤差が 1000 回でほぼ一定になっているように見えます。回数でも速さに差がありますが、時間ではもっと差があります。筆者のパソコンで計算したところ、推定の実行に数値微分では 838 秒、解析微分で

表 66. 推定された係数の比較

numerical				analytical				keras			
第1層				第1層				第1層			
細胞	a_0	a_1	a_2	細胞	a_0	a_1	a_2	細胞	a_0	a_1	a_2
1	0.192948	-6.25267	-1.56976	1	0.192948	-6.25267	-1.56976	1	-0.5715	5.835425	1.345716
2	-0.30494	1.632353	-7.29199	2	-0.30494	1.632353	-7.29199	2	-0.12354	-1.21315	7.472765
第2層				第2層				第2層			
細胞	b_0	b_1	b_2	細胞	b_0	b_1	b_2	細胞	b_0	b_1	b_2
1	4.563186	-5.52949	-5.1468	1	4.563186	-5.52949	-5.1468	1	-7.13747	7.093163	8.438554
2	-0.69589	6.021247	-6.67039	2	-0.69589	6.021247	-6.67039	2	-1.16122	-9.33266	9.015834
3	-4.80711	4.996871	5.10183	3	-4.80711	4.996871	5.10183	3	6.899177	-7.80041	-10.2782
4	-0.85279	-6.14654	6.392856	4	-0.85279	-6.14654	6.392856	4	-2.15405	10.22326	-7.56991
5	1.793752	0.665182	0.296646	5	1.793752	0.665182	0.296646	5	3.553574	-0.1111	0.444254

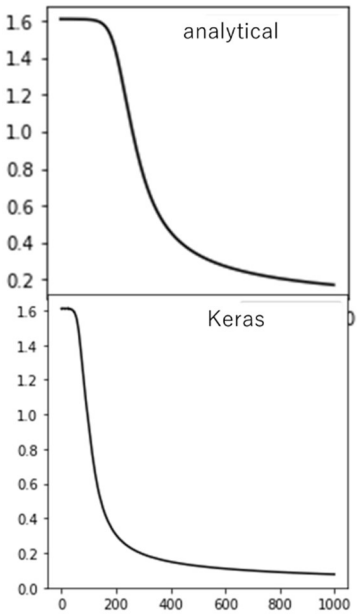

図 121. 平均交差エントロピー誤差の変化

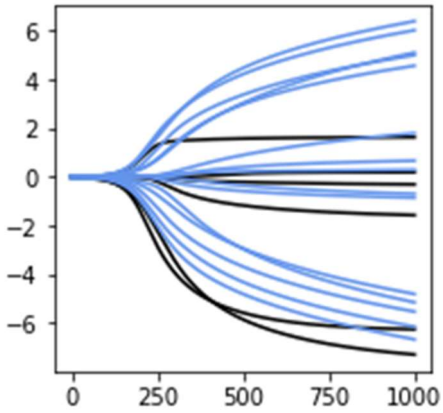

図 122. analytical で推定された係数の変化

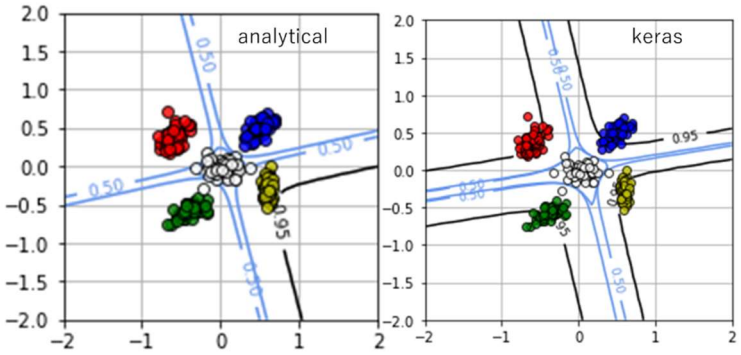

図 123.analyticalkeras の判別境界の比較

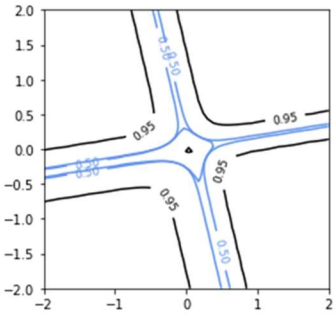

図 124. Keras で描いた境界線

は 62 秒かかりましたが、keras ではわずかに 3.5 秒でした。図 122 には解析微分で計算したときの各係数の時間変化を示しました。確率 0.5 の境界にあまり違いはありませんが、確率 0.95 の境界線には大きな違いがあって、解析微分では 0.95 の境界線が 1 本しかありません。その位置も適切ではないようです。図 124 は keras で描いた境界線を取り出した図です。中心部にクラス 5 の確率 0.95 の点のような領域が見えて、かなり精密で綿密に境界線が描かれています。keras を使うのが時間的にも精度の上でもお勧めです。

keras のコードリスト VII-2-3-3-vi は拡大図の図示です。VII-2-3-3-vii は得られた係数の出力、VII-2-3-3-viii はそれらの保存です。VII-2-3-3-ix で得られた係数を使って新たに与えられたデータから各クラスに属する確率を計算します。例えば、data として表 67 左のようなデータを csv のデータシートで与えると、表 67 右のような確率が返ってきます。図 125 それらの確率の分布を図示したものです。最後に、sammple2 の個々のクラスの分布中心を変えずに、偏差を 2 倍に広げたデータ（sample3）で同じ計算をしてみたところ、図 126 が得られました。重なり合いが多くなって、0.95 以上の確率で予測できる領域はなくなりましたが、0.5 の確率ならば、かなりの範囲で予測可能です。しかし、クラス 5 の 0.5 以上の確率の領域はとても小さくなってしまいました。クラス 1〜4 を拾い出すのであれば、この分析はそれなりに有効ですが。この二つの変数だけでクラス 5 を他のクラスから取り分けて拾い出すことは難しいということです。クラス 5 を確実に拾い出すためには、新たに何らかの変数を加えなくてはならないでしょう。

表 67 確率の予測、データ（左）と帰ってくる確率（右）

x1	x2	p1	p2	p3	p4	p5
-1	-1	0	0.0003	0.9648	0.0001	0.0347
-0.5	-1	0	0.0003	0.9631	0.0001	0.0364
0	-1	0	0.0002	0.9121	0.0011	0.0866
0.5	-1	0.0006	0	0.0137	0.8515	0.1342
1	-1	0.0003	0	0.0002	0.9877	0.0118
-1	-0.5	0	0.0011	0.9317	0.0001	0.0672
-0.5	-0.5	0	0.0006	0.9435	0.0002	0.0557
0	-0.5	0	0.0003	0.7981	0.0056	0.196
0.5	-0.5	0.0006	0	0.002	0.9456	0.0519
1	-0.5	0.0003	0	0.0001	0.9882	0.0113
-1	0	0.0014	0.841	0.0014	0	0.1561
-0.5	0	0.0016	0.5697	0.0124	0	0.4163
0	0	0.0125	0.0171	0.0111	0.0031	0.9563
0.5	0	0.0513	0	0.0002	0.7148	0.2337
1	0	0.008	0	0.0001	0.9405	0.0514
-1	0.5	0.0015	0.9758	0	0	0.0227
-0.5	0.5	0.0035	0.9579	0	0	0.0386
0	0.5	0.6496	0.0802	0	0.0001	0.27
0.5	0.5	0.9773	0.0001	0	0.0007	0.022
1	0.5	0.9772	0.0001	0	0.0014	0.0213
-1	1	0.0015	0.9765	0	0	0.0219
-0.5	1	0.0075	0.94	0	0	0.0524
0	1	0.8952	0.0082	0	0.0001	0.0964
0.5	1	0.9865	0.0001	0	0.0003	0.0131
1	1	0.9884	0.0001	0	0.0004	0.0112

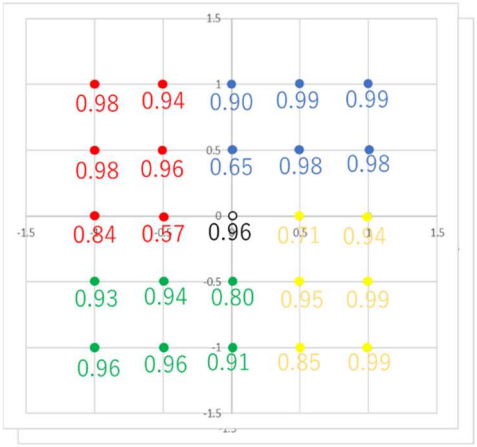

図 125. それぞれのクラスである確率の分布

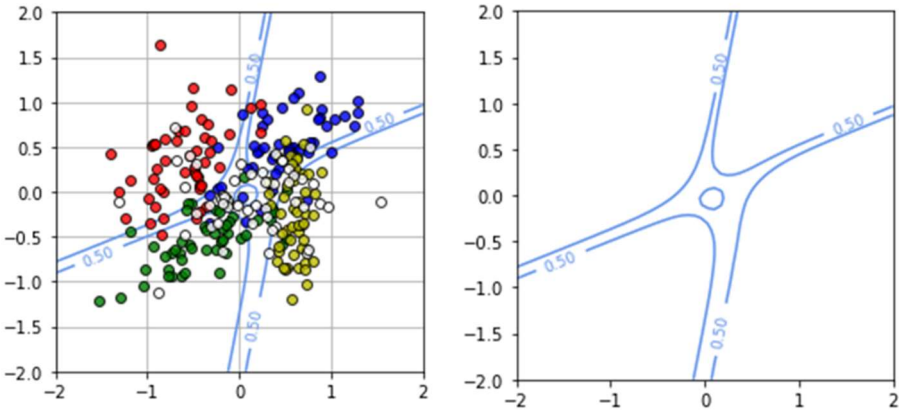

図 126. 分布範囲を広げた場合の境界線