まえがき

深層学習（ディープラーニング）は，多層のニューラルネットワークを用いた機械学習の方法論です．近年，音声，画像あるいは言語を対象とする人工知能（AI）の諸問題に対し，他の方法を圧倒する高い性能を示すことがわかり，多くの研究者や技術者の関心を集めています．人工知能に対する期待が最近かつてない高まりを見せていますが，その背景にも深層学習の成功があります．

ニューラルネットワーク自体の研究は，80 年代にも盛んに行われていました．しかしその後，90 年代半ばからつい最近に至るまで，関心が著しく低下した「冬の時代」がありました．同時期の初め頃から発展したサポートベクトルマシンと比較すると，ニューラルネットワークは，性能を引き出すには多くのパラメータを調整しなければならず，また理論的な裏付けに乏しい欠点があり，多くの人が関心を失いました．

この冬の時代を乗り越えて，今，再びニューラルネットワークが注目されています．転機になったのは，多層ネットワークの学習に関する Hinton らの研究でした．さらに，90 年代と比較すると計算機の能力が大きく向上し，またウェブや検索エンジンの発達にともない大規模な訓練データを用意しやすくなったことも，高い性能の達成を可能にしました．最近では一般のメディアにも深層学習が取り上げられるようになり，そこではバラ色の未来が語られています．時間の経過とともに研究が進めば，いずれ人に匹敵する知能を実現できるようになるというのです．人工知能が人にとっての脅威となるという懸念さえ語られるほどです．

しかしながら，現在の熱狂は，先述の「冬の時代」の反動といえる部分があります．ノーマークだったニューラルネットワークが，いくつかの難問を鮮やかに解決して見せたのです．関連分野の研究者が受けた衝撃はかなり大きなものでした．しかし落ち着いて考えてみると，確かにできることは増えたものの，人の知能が目標であるならば，まだできていないことの方が圧倒的に多いというのが本当のところです．熱狂と無関心を行き来した過去の経緯を考えると，今はニューラルネットワークへの期待が高すぎる時期なのか

も知れません．

そんな状況だからこそ，現在，地に足のついた地道な研究開発が求められています．深層学習の性能の高さは誰もが知るところとなりましたが，理論的な理解は追いついていません．例えば，なぜ多層だと性能が出るのかという問いに対し，納得できる説明は見つかっていません．個々の問題に対する有効性は適用してみなければわからず，うまくいかなかった場合に何が悪いのかがわかりません．この点では 80 年代からあまり進歩してしないともいえ，今後の研究の進展が望まれます．

現在，深層学習の研究は，性能向上のための手法の開発と応用の開拓が急速なペースで進みつつあります．本書の執筆にあたっては，評価の定まらない最新手法や応用事例を取り上げるよりも，基本的な事項をなるべく広くカバーすることとしました．

最後に，本書の執筆にあたってお世話になった皆様に感謝いたします．麻生英樹氏と安田宗樹氏には，本書を草稿の段階で査読いただき，数多くのご助言をいただきました．講談社サイエンティフィクの横山真吾氏には，執筆開始から出版に至るまでさまざまな支援をいただきました．また，本シリーズ編者の杉山将氏には，豪華執筆陣に混じって執筆する貴重な機会を与えてくださったことを，この場を借りて改めて感謝いたします．

2015 年 3 月

岡谷貴之

Chapter 1

はじめに

1.1 研究の歴史

1.1.1 多層ニューラルネットへの期待と失望

高度な情報処理の実現を目指して，生物の神経回路網を模倣した人工ニューラルネットワーク（以下ニューラルネット）が，長年にわたって研究されてきました．ただしその研究の歴史は平坦なものではありませんでした．1940 年代に研究が開始 [24, 48, 57] されて以来これまで，研究が一旦はブームになり，その後下火になるということを 2 度ほど繰り返してきました．中でも 80 年代半ばから 90 年代前半にかけて起こった 2 度めのブームは，多層ニューラルネットの学習法である**誤差逆伝播法**（**back propagation**）の発明 [59] をきっかけとしたもので，研究は大きな広がりを見せました．しかしこのブームも，90 年代後半にはすっかり終焉してしまいます．その間ごく最近まで，ニューラルネットの研究は極めて低調なものでした [64]．

この 80 年代半ばからのブームが終わってしまった背景に，2 つの理由を挙げることができます．

第一に，誤差逆伝播法によるニューラルネットの学習は，**図 1.1** 左のような 2 層程度のニューラルネットでこそ，うまくいきましたが，それ以上多層になると期待したような結果を得られませんでした．訓練サンプルの学習はできても汎化性能が上がらない，過適合（あるいは過学習）が壁となりました．誤差逆伝播法とは，サンプルに対するネットワークの誤差（目標出力と実際の出力との差）を，入力層から逆に伝播させ，各層の重みの勾配を計算するという方法です．この計算の際，入力層から離れた「深い」層に進むに連

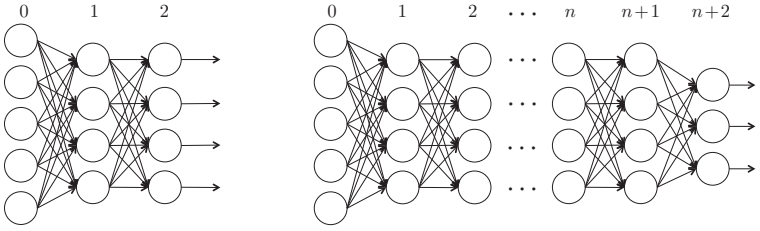

図 1.1　「浅い」ネットワーク（2層ネットワーク）と「深い（ディープな）」ネットワーク（多層ネットワーク）.

れ，勾配が急速に小さくなったり，あるいは急速に大きくなって発散してしまう，いわゆる勾配消失問題という呼ばれる現象が起こるようになります．このことが，多層ネットワークの学習を困難なものにしました．

第二に，ニューラルネットには層数やユニット数その他学習のためのパラメータが多数あるのに，それらが最終的な性能とどのように結びつくかがよくわかりませんでした．性能を引き出すためにこれらをどのように決めたらよいのか，ノウハウはあっても理論がなく [64]，この点が後発の機械学習の方法に見劣りしました．これらの理由から，ニューラルネットの研究は 90 年代後半には下火になってしまいました．

ただし最初の理由，つまり多層ネットワークの学習が難しいことに対しては，当時 1 つの例外があったことを指摘しておく必要があります．確かに，層間が密に結合された多層ネットワークの学習は困難でした．しかし，画像を対象とする**畳込みニューラルネット**（**convolutional neural network**, **CNN**）は，80 年代後半には 5 つもの層からなる多層ネットワークの学習に成功していました [42]．畳込みニューラルネットは，特定の画像処理を行う働きを持つ層を何層か重ねた構造を持ち，そのために層間の結合は疎らなネットワークです．Fukushima らのネオコグニトロン [17] をルーツに持ち，これに LeCun らが誤差逆伝播法による学習を取り入れ，完成させました [43]．当時，LeCun らの畳込みニューラルネットは手書き文字認識に応用されて高い性能を実現し，よく知られた存在でしたが，ニューラルネットの研究が下火になるにつれて，向けられる関心も小さくなっていきました．

1.1.2 多層ネットワークの事前学習

その後 90 年代後半から 2000 年代前半にかけて，ニューラルネット全般に対する関心は随分と低いものでしたが，Hinton らの**ディープビリーフネットワーク**（**deep belief network**, **DBN**）の研究 [28] が，これを覆すブレークスルーとなりました．DBN は，一般的なニューラルネットに似た多層構造を持つグラフィカルモデルです．その振る舞いは確率的に記述され，主にデータの生成モデルとして使われます．その学習は，一般的なニューラルネットとは異なる原理に基づいて行わるものの，多層になると DBN の学習は同じように困難でした．

これに対し Hinton らは DBN を，まず層ごとに**制約ボルツマンマシン** (**restricted Boltzmann machine**, **RBM**) と呼ばれる単層ネットワークに分解した上で，貪欲法の考え方に従い，これらの RBM を入力層に近い側から順番に教師なしで学習してゆく方法を提案しました．そしてこの方法によって，DBN の学習がうまく行えることを示しました．さらに，こうして得られる DBN は，一般的な順伝播型のニューラルネットに転換することが可能で，そうやって得られたニューラルネットは，多層であっても通常の方法で（過適合を起こさず）うまく学習できました．目的とするネットワークの学習前，層ごとに学習を行うことでパラメータのよい初期値を得ておくこのやり方は，**事前学習**（**pretraining**）と呼ばれています．

その後，DBN や RBM ではなく，より単純な**自己符号化器**（**autoencoder**）を使っても多層ネットワークの事前学習が可能であることがわかりました [4]．自己符号化器は，入力に対し計算される出力が，入力になるべく近くなるように訓練されるニューラルネットです．自己符号化器の目標出力は入力そのものであり，したがってその学習は教師なしで行われます．例えば任意の単層ネットワークを考えたとき，入力を順方向に伝播させて一旦出力を求め，今度はこれを逆方向に伝播させて入力側に戻す操作を考えると，これは自己符号化器の 1 つの形を与えます．この形の自己符号化器を使うと，DBN と同様の方法で，多層ニューラルネットの事前学習を行えるとわかったのです．目的とする多層ニューラルネットを単層に分割し，入力層から順番に各層を，上述の自己符号化器と見て教師なしで学習し，各層のパラメータの初期値を得ます．その後，全層のニューラルネットを教師ありで学習します．多層ニューラルネットは，パラメータをランダムに初期化すると

学習がうまく行えませんが，層ごとに教師なし学習を行う事前学習によって得たパラメータを初期値に使えば，学習がうまくいくことがわかったのです．

1.1.3　特徴量の学習

画像や音声など自然界のデータは一般に高次元空間に存在しますが，その空間内にまんべんなく広がっているわけではなく，強い偏りを持ちながら複雑に広がっていると考えられます．例えば風景を写した自然画像は，白色雑音（を画像にしたもの）とは，その性質が大きく違います．事前学習の成功の後に研究者の関心がまず向かったのは，多層ネットワークがこのような自然界のデータを学習したとき，そのようなデータの持つ構造が，どのようにネットワークの多層構造によって捉えられ，表現されるかでした．

自然画像から切り出したパッチ（小領域）の集合を対象に，**スパース符号化**（**sparse coding**）によって辞書（基底）の学習を行うと，哺乳類の脳の初期（低次）視覚野で取り出されているとされるガボールウェーブレット状の基底が得られることは，以前から知られていました [53]．自己符号化器に，冗長な基底の少数の組合せで入力を表現するというスパース符号化の考えを取り入れることで，多層ネットワークは学習によって興味深い階層構造を形成することがわかってきました．例えば Lee らは，自然画像のパッチを 2 層以上のネットワーク（RBM を重ねたもの）で学習すると，霊長類の視覚野の V2 領域に見られるとされる特徴 [36] に類似した特徴が得られることを報告しています [45]．また，畳込みニューラルネットの構造を組み込んだ DBN を使い，事物を写した自然画像を学習すると，画像の特徴が層と対応した階層性を伴う形で学習されることも報告しています [46]．Le らは，同様の構造を持つより大規模化なネットワークで自己符号化器を構成し，YouTube の動画から無作為に取り出した 100 万枚の画像を学習させると，特定の物体だけに選択的に反応するユニット *1 が自然に生成されることを示しました [41]．

1.1.4　深層学習の隆盛

その後，音声認識や画像認識のベンチマークテストで，多層ニューラルネットの有効性が確かめられ，過去の記録を次々にぬりかえるようになりま

*1　脳における物体の認識およびその内部表現に関する，神経科学の古典的な仮説である「おばあさん細胞」に相当するものです．

した．こうして，**深層学習**（**ディープラーニング**）の有効性が広く認知されるようになったのです（以降，深層学習が対象とする多層ニューラルネットのことを，ディープニューラルネット，略してディープネットと呼ぶことにします）．

ただし，ひとことで深層学習といっても，実際にはいくつかの異なる方法論の集まりであって，問題に応じてそれらが使い分けられていることには注意する必要があります．例えば，音声認識では層間ユニットが全結合したネットワークがよく使われ，上述の事前学習が一般的に行われています．一方で画像認識では，畳込みニューラルネットが主流であり，そこでは一般に事前学習は必要とされません．また，自然言語処理や音声認識の特定のタスクでは，**再帰型ニューラルネット** (**recurrent neural network**, **RNN**) が使われています．形の上ではいずれも多層ネットワークと言えますが，仕組みと学習方法はかなり異なるものです．

とはいえ，複数の分野で近年，多層ニューラルネットがこのように高い性能を発揮できるようになった背景には，共通の理由があります．まず，現実の問題は複雑であり，これを解くにはその複雑さに見合う規模のニューラルネットが必要です．そして，そんな大きなネットワークが過適合を起こさず学習できるためには，（事前学習がいくら有効でも）一定以上の規模のデータを要します．90 年代と違って今は，ウェブを始めとするインフラが整っており，十分な量の学習データを集めることができます．同様に，90 年代と比べると，GPU やマルチコア化された CPU，さらにその集合体である PC クラスタに見られるように，計算機の計算能力は飛躍的に向上しています [9]．これらはディープネットが成功する必要条件だったといえるでしょう．

また上述のように，事前学習は，深層学習のブームのきっかけを開きましたが，最近では必須の技術というわけではなくなりつつあります．主に画像認識に限られるとはいえ，畳込みニューラルネットは元々事前学習を要しません．今使われている畳込みニューラルネットは，80 年代末の LeCun らのネットワークの構造と学習方法を，ほぼそのまま引き継いでいます．音声認識などで主流となる全結合型のネットワークでも，ドロップアウト [67] のように，層ごとに教師なし学習を重ねる事前学習に頼らず，勾配消失問題や過適合を避け得る方法がいくつか提案されています．さらに，重みの初期化を慎重に行うだけでも，以前考えられていたよりはずっとうまく学習できるこ

とも指摘されています [27]．このように，最近のディープネットの成功の本質は，現実世界の大規模な問題を相手に，多層ニューラルネットを試して見たところ，思わぬ性能が出ることがわかったということかもしれません．

1.2 本書の構成

2 章では順伝播型ネットワークを扱います．入力から出力まで決まった一方向に情報が伝達されるタイプで，最も基本的かつ応用範囲が広いニューラルネットです．3 章では，順伝播型ネットワークの学習方法を説明します．特にディープネットで定番の学習方法となっている確率的勾配降下法に焦点を当て，基礎理論と方法を述べます．その際，最適化の目標となる誤差関数の最小化に，ネットワークのパラメータに関する微分（誤差勾配）を求める必要があります．4 章では，そのための方法である誤差逆伝播法を説明します．プログラムの実装上の考え方についても簡単に触れます．5 章では自己符号化器を扱います．教師なし学習を行うニューラルネットで，主にデータをよく表す特徴量を学習し，データのよい表現を得ることを目的とします．鍵となるスパース正則化やデータの白色化についても説明します．また自己符号化器を利用したディープネットの事前学習の方法を説明します．

6 章では，画像への応用で欠かせない存在となっている畳込みニューラルネット（CNN）を扱います．畳込み層やプーリング層などの特別な構造を持つ層について詳細に述べ，具体的な応用例を紹介します．7 章では再帰型ネットワーク（RNN）を扱います．可変長の時系列データを効率よく扱うためのニューラルネットです．基本となる構成と学習方法を説明し，音声認識や手書き文字認識などへの応用で成功を収めている長・短期記憶（Long Short-Term Memory）と，入力と長さの異なる系列を推定することのできるコネクショニスト時系列分類法 (CTC) を説明します．そして 8 章ではボルツマンマシンを扱います．他の章で扱ったニューラルネットと異なり，双方向的なユニット間の結合を持ち，挙動が確率的に記述されることが特徴です．データの生成モデルとして利用される他，順伝播型ネットワークに転換され利用されることもあります．

Chapter 2

順伝播型ネットワーク

順伝播型ネットワークは最も基本的かつ最もよく使われているニューラルネットワークです．本章では，ネットワークの構成要素と，回帰やクラス分類など問題ごとに異なる出力層の設計，および誤差関数の選択など，基本的な事項を説明します．

2.1 ユニットの出力

順伝播型（ニューラル）ネットワーク（**feedforward neural network**）[*1]は，層状に並べたユニットが隣接層間でのみ結合した構造を持ち，情報が入力側から出力側に一方向にのみ伝播するニューラルネットワークです．文献によっては**多層パーセプトロン**（**multi-layer perceptron**）と呼ばれます[*2]．

ネットワークを構成する各ユニットは図 2.1 のように，複数の入力を受け取り，1 つの出力を計算します．同図の場合，4 つの入力 x_1, x_2, x_3, x_4 を受け取りますが，このユニットが受け取る総入力 u は，

$$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

のように，各入力にそれぞれ異なる**重み**（**weight**）w_1, w_2, w_3, w_4 を掛けた

---

*1 フィードフォワード（ニューラル）ネットワークという訳も一般的に使われています．

*2 多層パーセプトロンという呼び名が意図的に避けられる場合があります．単一のパーセプトロンを多層化したかのように誤解される危険があることと，順伝播型ネットワークではユニットの入出力関数がオリジナルのパーセプトロンと違ってステップ関数に限定されないことなどがその理由です．

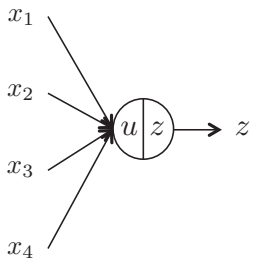

図 2.1　ユニット 1 つの入出力．$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$．$z = f(u)$．

ものをすべて加算し，これに**バイアス**（**bias**）と呼ばれる 1 つの値 b を足したものになります．このユニットの出力 z は，総入力 u に対する，**活性化関数**（**activation function**）と呼ばれる関数 f の出力です．

$$z = f(u)$$

順伝播型ネットワークでは図 2.2 のように，このようなユニットが層状に並べられ，層間でのみそれらは結合を持ちます．左の層のユニットの出力が右の層のユニットの入力になるという形で，この結合を通じて信号は左の層から右の層へと一方向に伝わります．同図のネットワークでは，右の層の 3 つのユニット（$j = 1, 2, 3$）はそれぞれ，左の層の 4 つのユニット（$i = 1, 2, 3, 4$）からの出力 x_1, x_2, x_3, x_4 を入力として受け取ります．ユニット間の結合は全部で $3 \times 4 = 12$ 本ありますが，その 1 つ 1 つの結合に異なる重み w_{ji} が与えられています．式で表すと，3 つのユニットが受け取る入力はそれぞれ

$$u_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_1 \tag{2.1a}$$

$$u_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_2 \tag{2.1b}$$

$$u_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_3 \tag{2.1c}$$

のように計算され，これらに活性化関数を適用したものが出力

$$z_j = f(u_j) \quad (j = 1, 2, 3) \tag{2.2}$$

となります．

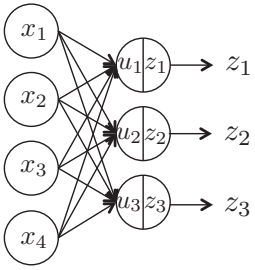

図 2.2 2 層に並べられたユニットを持つネットワーク．式 (2.1) 参照．

第 1 層のユニットを $i = 1, \ldots, I$，第 2 層のユニットを $j = 1, \ldots, J$ で表すと，第 1 層のユニットの出力から第 2 層のユニットの出力が決まるまでの計算は，次のように一般化されます $(j = 1, \ldots, J)$.

$$u_j = \sum_{i=1}^{I} w_{ji} x_i + b_j$$
$$z_j = f(u_j)$$

ベクトルと行列を用いて表記すると，これは

$$\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{2.3a}$$
$$\mathbf{z} = \mathbf{f}(\mathbf{u}) \tag{2.3b}$$

のように表現できます．ただし，各ベクトルと行列は次のように定義しています．

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_J \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_I \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_J \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_J \end{bmatrix},$$
$$\mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1I} \\ \vdots & \ddots & \vdots \\ w_{J1} & \cdots & w_{JI} \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} f(u_1) \\ \vdots \\ f(u_J) \end{bmatrix}$$

2.2　活性化関数

ユニットが持つ活性化関数には通常，単調増加する非線形関数が用いられます．色々なものがありますが，古くから最もよく使われているのが，**ロジスティックシグモイド関数**（**logistic sigmoid function**）あるいは**ロジスティック関数**（**logistic function**）と呼ばれる

$$f(u) = \frac{1}{1+e^{-u}}$$

です．本書ではロジスティック関数と呼びます．この関数は実数全体 $(-\infty, \infty)$ を定義域に持ち，図 2.3 のように $(0, 1)$ を値域とします．ロジスティック関数の代わりに類似の双曲線正接関数

$$f(u) = \tanh(u)$$

を使うことがあります．関数の値域は $(-1, 1)$ と変わりますが，$\tanh(u) = (e^u - e^{-u})/(e^u + e^{-u})$ とも表せるように，ロジスティック関数とよく似た性質を持ちます．いずれの関数も，入力の絶対値が大きな値をとると出力が飽和し一定値となること，その間の入力に対して出力が徐々にかつ滑らかに変化することが特徴であり，一般に**シグモイド関数**（**sigmoid function**）と

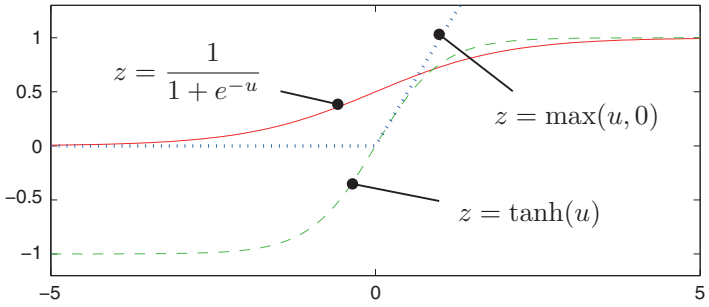

図 2.3　典型的な活性化関数とその形．

総称されます．これらは，生物の神経細胞が持つ性質 *3 をモデル化したものです．

近年これらの活性化関数に代わり，**正規化線形関数**（**rectified linear function**）

$$f(u) = \max(u, 0)$$

がよく使われています [18, 38, 52]（あるいは単に rectifier とも呼ばれる）．これは図 2.3 のように，$z = u$ の線形関数のうち $u < 0$ の部分を $u = 0$ で置き換えただけの単純な関数です．単純で計算量が小さく，さらに上述の 2 つの関数よりも学習がより速く進み，最終的にもよりよい結果が得られることが多いため，現在最もよく使われています．シグモイド関数では入力の変動の大きさに気を使う必要がありますが（大きすぎると出力がほとんどの場合 0 か 1 のどちらかになってしまう），正規化線形関数ではそういったことがありません（例えばバイアスが 0 なら，入力を 2 倍にすると出力はそのまま 2 倍になります）．なお，この関数を持つユニットのことを ReLU(Rectified Linear Unit) と略記することがあります．この関数の性質については 8.7.2 項で改めて述べます．

以上の関数が最も標準的なものですが，これらの他にもいくつかの活性化関数があります．1 つは線形写像や恒等写像

$$f(u) = u \tag{2.4}$$

です．ニューラルネットでは，各ユニットの活性化関数が非線形性を持つことが本質的に重要ですが，部分的に線形写像を使う場合があります．例えば 2.4.2 項で説明する回帰問題のためのネットワークでは，出力層に恒等写像を用います．またクラス分類を目的とするネットワークでは，出力層の活性化関数に通常，ソフトマックス関数を使います．これら出力層の活性化関数については 2.4.4 項で改めて述べます．

また，ロジスティック関数を区分的に直線で近似した次の関数を使うことがあります．

---

*3 高等生物の神経細胞では，情報は電気的パルスの時間密度によって表現され，細胞間でやりとりされます．この時間密度は最小値が 0，電気化学的な制約で決まる上限を最大値とします．

$$f(u) = \begin{cases} -1 & u < -1 \\ u & -1 \leq u < 1 \\ 1 & u \geq 1 \end{cases}$$

直線の組合せで非線形性を表現している点で正規化線形関数や最大化出力関数と似ていますが，値域に制限がある点ではシグモイド関数に近いといえます．

さらに，正規化線形関数ともつながりの深い**マックスアウト**（**maxout**）と呼ばれる関数 [19] があります．この活性化関数を持つユニット 1 つは，K 個の異なるユニットをまとめて 1 つにしたような構造を持ちます．それら K 個の 1 つ 1 つが異なる重みとバイアスを持ち，それぞれの総入力を $u_{j1}, \ldots, u_{jK}$ と別々に計算した後，それらの最大値をこのユニットの出力とします．

$$u_{jk} = \sum_i w_{jik} z_i + b_{jk} \quad (k = 1, \ldots, K)$$

$$f(u_j) = \max_{k=1,\ldots,K} u_{jk}$$

各ユニットがこの活性化関数を持つネットワークは，各種ベンチマークテストで，正規化線形関数を使ったネットワークを凌ぐ結果を残しています．ただし，パラメータ数が普通のユニットの K 倍あることから，それほどよく使われるというわけではありません．また，6 章で説明する畳込みネットワークが持つ最大プーリングという構造も，実質的に同じ計算を行います．

2.3　多層ネットワーク

図 2.2 のネットワークにもう 1 つの層を追加し，**図 2.4**(a) に示すような 2 層構造のネットワークを考えます [*4]．情報は左から右へと一方向に伝わり，この順に各層を $l = 1, 2, 3$ で表します．なお $l = 1$ の層を**入力層**（**input layer**），$l = 2$ を**中間層**（**internal layer**）あるいは**隠れ層**（**hidden layer**），$l = 3$ を**出力層**（**output layer**）と呼びます．

各層のユニットの入出力を区別するために，各変数の右肩に層の番号（$l = 1, 2, 3$）を付け，$\mathbf{u}^{(l)}$ や $\mathbf{z}^{(l)}$ のように書くことにします．この表記を

[*4] 本書では，ネットワークの層数は入力層をカウントしないで数えることにします．つまり図 2.4(a), (b) のネットワークの層数は 2 と数えます．

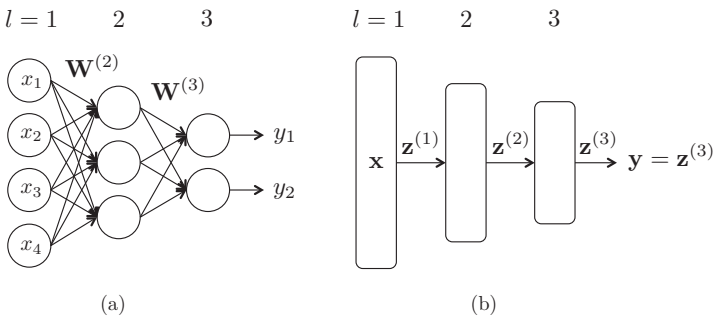

図 2.4 3 つのユニットを中間層に持つ 2 層ネットワークと，このネットワーク各層の入出力．(a) 詳細な構造．(b) 各層をひとまとめにしたネットワークの簡略表示．

使うと，中間層（$l=2$）のユニットの出力は式 (2.3) にならって次のように計算されます．

$$\mathbf{u}^{(2)} = \mathbf{W}^{(2)}\mathbf{x} + \mathbf{b}^{(2)}$$
$$\mathbf{z}^{(2)} = \mathbf{f}(\mathbf{u}^{(2)})$$

$\mathbf{W}^{(2)}$ は入力層と中間層間の結合の重みで，図 2.4 のネットワークでは 3×4 行列になります．また $\mathbf{b}^{(2)}$ は中間層のユニットに与えられたバイアスです．次に，出力層（$l=3$）のユニットの出力は，$\mathbf{x}$ を中間層の出力 $\mathbf{z}^{(2)}$ に置き換えて

$$\mathbf{u}^{(3)} = \mathbf{W}^{(3)}\mathbf{z}^{(2)} + \mathbf{b}^{(3)}$$
$$\mathbf{z}^{(3)} = \mathbf{f}(\mathbf{u}^{(3)})$$

のように計算されます．$\mathbf{W}^{(3)}$ は中間層と出力層間の重みで，図 2.4 のネットワークでは 2×3 行列です．$\mathbf{b}^{(3)}$ は出力層のユニットに与えられたバイアスです．

以上は，任意の層数 L のネットワークに一般化できます．層 $l+1$ のユニットの出力 $\mathbf{z}^{(l+1)}$ は，1 つ下の層 l のユニットの出力 $\mathbf{z}^{(l)}$ から

$$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)} \tag{2.5a}$$

$$\mathbf{z}^{(l+1)} = \mathbf{f}(\mathbf{u}^{(l+1)}) \tag{2.5b}$$

のように計算されます．したがって，ネットワークに対する入力 $\mathbf{x}$ が与えられたとき（層 $l = 1$ のユニットの出力を $\mathbf{z}^{(1)} = \mathbf{x}$ とし），式 (2.5a), 式 (2.5b) を $l = 1, 2, 3, \ldots, L-1$ の順に実行していくことで，各層の出力 $\mathbf{z}^{(2)}, \mathbf{z}^{(3)}, \ldots, \mathbf{z}^{(L)}$ をこの順に決定していくことができます．以下では，ネットワークの最終的な出力を

$$\mathbf{y} \equiv \mathbf{z}^{(L)}$$

と表記することにします．なお上では，表記の簡素化のため単一の活性化関数 $\mathbf{f}$ を使うかのように表記しましたが，各層で異なるものを選んでも構いません．特に出力層のユニットの活性化関数は，問題に応じて一般に中間層とは異なるものを選びます．

このように順伝播型ネットワークでは，与えられた入力 $\mathbf{x}$ に対し，入力層側から出力層側へ，上の計算を繰り返すことで情報を伝播させ，出力 $\mathbf{y}$ を計算します．この関係は，関数 $\mathbf{y} = \mathbf{y}(\mathbf{x})$ として表現できます．この関数の中身を決定するのは，各層間の結合重み $\mathbf{W}^{(l)}(l = 2, \ldots, L)$ とユニットのバイアス $\mathbf{b}^{(l)}(l = 2, \ldots, L)$ です．以下これらをネットワークのパラメータと呼びます．これらを変えれば，さまざまな関数を表現できることになります．このことを表すために，$\mathbf{y}(\mathbf{x}; \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(L)}, \mathbf{b}^{(2)}, \ldots, \mathbf{b}^{(L)})$ ，あるいは，これらネットワークのパラメータすべてを成分に持つベクトル $\mathbf{w}$ を定義し簡潔に $\mathbf{y}(\mathbf{x}; \mathbf{w})$ と書くことにします．

2.4　出力層の設計と誤差関数

2.4.1　学習の枠組み

上述のように，順伝播型ネットワークが表現する関数 $\mathbf{y}(\mathbf{x}; \mathbf{w})$ は，ネットワークのパラメータ $\mathbf{w}$ を変えると変化します．よい $\mathbf{w}$ を選ぶことで，このネットワークが望みの関数を与えるようにすることを考えます．

目標とする関数は，その具体的な姿形（すがたかたち）はわからないものの，関数の入力と出力のペアが複数与えられているとします．すなわち，1 つの入力 $\mathbf{x}$ に対する望ましい出力を $\mathbf{d}$ と書くと，そのような入出力のペアが複数，