ＡＩ基礎セミナー

第８回

機械学習（２回目：機械学習の概要と教師あり学習（分類））

改訂履歴

日付	担当者	内容
2021/05/08	M.Takeda	Git 公開

目次

(3) 脳神経系のモデル化
    (3.1) 神経細胞間の信号伝達の様子
    (3.2) 神経細胞間の信号伝達のﾓﾃﾞﾙ化
    (3.3) ﾆｭｰﾗﾙﾈｯﾄﾜｰｸと適用ﾓﾃﾞﾙの選択

(4) 前処理
    (4.1) ﾃﾞｰﾀを機械学習ﾓﾃﾞﾙに入力するまでの流れ
    (4.2) ﾃﾞｰﾀの種類について
    (4.3) 何を入力ﾃﾞｰﾀとするか
    (4.4) 入力ﾃﾞｰﾀのｸﾘｰﾆﾝｸﾞ
    (4.5) 入力ﾃﾞｰﾀの欠損値の扱い
    (4.6) 入力ﾃﾞｰﾀの標準化
    (4.7) 入力ﾃﾞｰﾀの加工による作成
    (4.8) 入力ﾃﾞｰﾀの符号化

(5) 活性化関数

(6) 損失関数
    (6.1) 損失関数・誤差関数・ｺｽﾄ関数・目的関数
    (6.2) 尤度と損失関数
    (6.3) 損失関数一覧

(7) ﾓﾃﾞﾙの構造
    (7.1) 順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ
    (7.2) 畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ
        (7.2.1) 畳み込み層
        (7.2.2) ﾌﾟｰﾘﾝｸﾞ層

(8) ﾓﾃﾞﾙの学習

(8.1) 勾配降下法と学習率

(8.2) 勾配降下法の改良版

(8.2.1) 確率的勾配降下法

(8.2.2) AdaGrad

(8.2.3) Adam

(8.3) 誤差逆伝搬法

(8.4) 「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」の誤差逆伝搬法

(8.4.1) 各層のﾕﾆｯﾄの誤差

(8.4.2) ﾕﾆｯﾄの誤差とｺｽﾄ関数の偏微分との関係式

(8.4.3) 出力層のﾕﾆｯﾄの誤差

(8.4.4) 隣接する層間のﾕﾆｯﾄの誤差についての逆漸化式

(8.4.5) 全ての中間層のﾕﾆｯﾄの誤差δを計算する

(8.4.6) ﾕﾆｯﾄの誤差δを用いて勾配降下法で学習する流れ（例）

(8.5) ﾓﾃﾞﾙの学習の検証

(8.5.1) ﾓﾃﾞﾙの学習の検証方法

(8.5.2) 過学習の対策とﾄﾞﾛｯﾌﾟｱｳﾄ

(9) まとめ

(10) 確認問題

(11) 確認問題回答用紙

(1) はじめに

・「ｾﾐﾅｰ7回目 (機械学習 1回目)」では、機械学習による画像認識を実装例を示して紹介しましたが、
「ｾﾐﾅｰ8回目 (機械学習 2回目)」では、機械学習の基本概念と「教師あり学習 (分類)」についての解説を行います。

・今回のｾﾐﾅｰは、「ｾﾐﾅｰ7回目」で扱ったﾓﾃﾞﾙとその学習方法の実装や他の「教師あり学習 (分類)」の文献を見て、
内容が手に取って読めるようになる、ということを目標とします。

・確認問題は本文を要約したような問題となっております。
解いてみて理解度を確認してみてください。

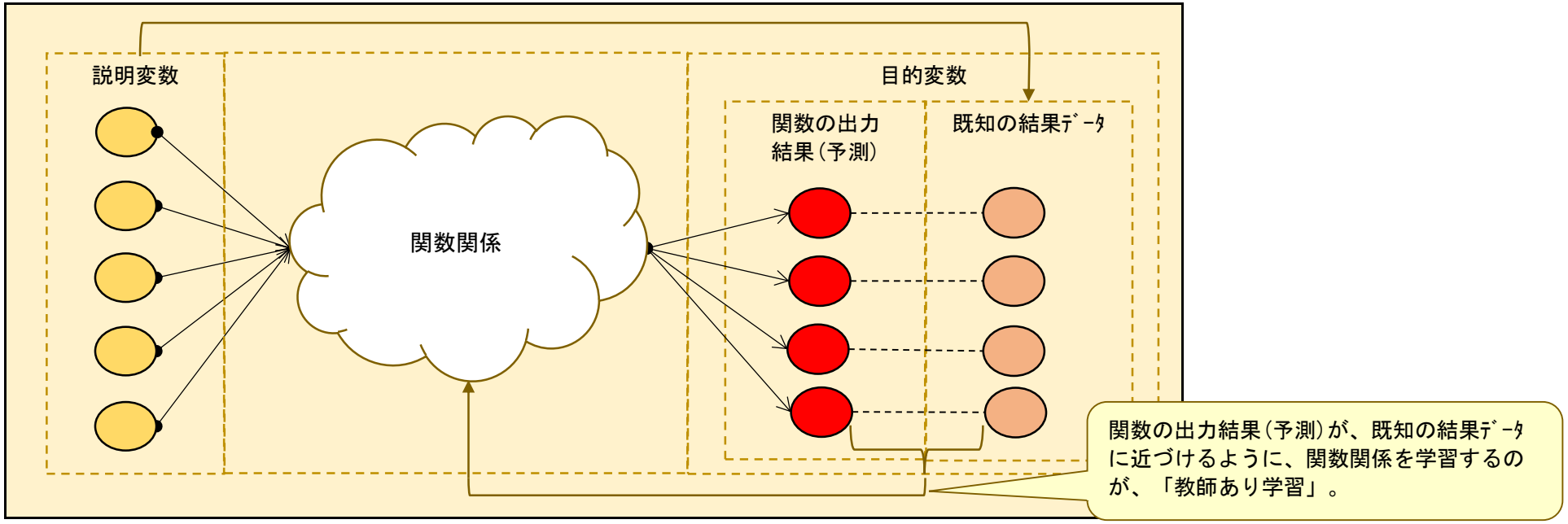
(2) 機械学習の手法

・「機械学習（ｷｶｲｶﾞｸｼｭｳ, machine learning, ML)」とは、人間の学習の仕組みを機械（特にｺﾝﾋﾟｭｰﾀ）で実現する技術・手法の総称です。
機械学習では、大量のﾃﾞｰﾀ間の統計的関係性をﾓﾃﾞﾙとして定式化して、そのﾊﾟﾗﾒｰﾀを訓練用ﾃﾞｰﾀで学習することにより推定します。
そしてそのﾓﾃﾞﾙを予測や分類といった問題に適用します。
機械学習の手法は、「教師あり学習」「教師なし学習」「強化学習」の３種類に分類できます。

(2.1) 教師あり学習

・何かの因果関係を探る場合、統計ﾓﾃﾞﾙでは、原因となる変数群を元に結果となる変数群を説明しようと試みます。
原因となる変数のことを「説明変数（ｾﾂﾒｲﾍﾝｽｳ、explanatory variable)」と言い、
結果となる変数のことを「目的変数（ﾓｸﾃｷﾍﾝｽｳ、response variable)」と言います。

・「教師あり学習（ｷｮｳｼｱﾘｶﾞｸｼｭｳ, Supervised learning)」は　異なるﾃﾞｰﾀ間の関係性を、一方を入力（説明変数）、
他方を出力（目的変数）とする関数関係として記述し、関数の出力を既知の結果ﾃﾞｰﾀ（目的変数の既知ﾃﾞｰﾀ）に近づけるように、
関数関係を学習する方法です。
この場合、関数の出力結果(予測)に対して、既知の結果ﾃﾞｰﾀが教師ﾃﾞｰﾀの役割をします（下図参照）。



・教師あり学習において、目的変数に対応する確率変数が連続確率変数の場合、関数関係を推定する学習を
「回帰（ｶｲｷ, Regression)」と呼びます。
「回帰」により分析することを「回帰分析（ｶｲｷﾌﾞﾝｾｷ, Regression analysis)」と呼びます。
「回帰」では、観測ﾃﾞｰﾀをもとに入力ﾃﾞｰﾀ（説明変数）と出力ﾃﾞｰﾀ（目的変数）の数値間の関連性を導き出し、
入力ﾃﾞｰﾀに対する出力ﾃﾞｰﾀを数値で予測します。

・教師あり学習において、目的変数に対応する確率変数が離散確率変数の場合、関数関係を推定する学習を
「分類（ﾌﾞﾝﾙｲ、Classification)」と呼びます。
「分類」での関数の出力結果は、各ﾗﾍﾞﾙが付与される確率を与えるものであり、
確率が最大となるﾗﾍﾞﾙが正解ﾗﾍﾞﾙ（教師ﾃﾞｰﾀ）と一致するように学習します。

(2.2) 教師なし学習

・入力変数（説明変数）に対する、出力変数（目的変数）が与えられていない状態で、
入力ﾃﾞｰﾀ自体の背後にある性質や特徴や構造などを分析することを、
「教師なし学習（ｷｮｳｼﾅｼｶﾞｸｼｭｳ、Unsupervised Learning)」と言います。

・「教師なし学習」の例として、「ｸﾗｽﾀﾘﾝｸﾞ（Clustering)」によるﾃﾞｰﾀ分類や、
「次元削減（ｼﾞｹﾞﾝｻｸｹﾞﾝ、Dimensionality Reduction)」による特徴量抽出などが挙げられます。

・「ｸﾗｽﾀﾘﾝｸﾞ」は入力ﾃﾞｰﾀを分類するものですが、
「教師あり学習」の「分類」では、分類済みの入力ﾃﾞｰﾀで分類方法を学習するのに対し、
「教師なし学習」の「ｸﾗｽﾀﾘﾝｸﾞ」では、未分類の似たﾃﾞｰﾀを集めてﾃﾞｰﾀ構造を発見します。
「ｸﾗｽﾀﾘﾝｸﾞ」のｱﾙｺﾞﾘｽﾞﾑとしては、「k平均法（ｹｲﾍｲｷﾝﾎｳ、k-means clustering)」や
「混合ｶﾞｳｽ分布（ｺﾝｺﾞｳｶﾞｳｽﾌﾞﾝﾌﾟ、Gaussian mixture models、GMM)」などがあります。

・「次元削減」は入力ﾃﾞｰﾀの次元を落とし、できるだけ「情報」を保ちながらより小さな次元の情報へ対応させることです。
「次元削減」の例として、多変量ﾃﾞｰﾀ解析の分野で提案された「主成分分析（ｼｭｾｲﾌﾞﾝﾌﾞﾝｾｷ、principal component analysis、PCA)」や
「因子分析（ｲﾝｼﾌﾞﾝｾｷ、factor analysis、FA)」がよく知られています。

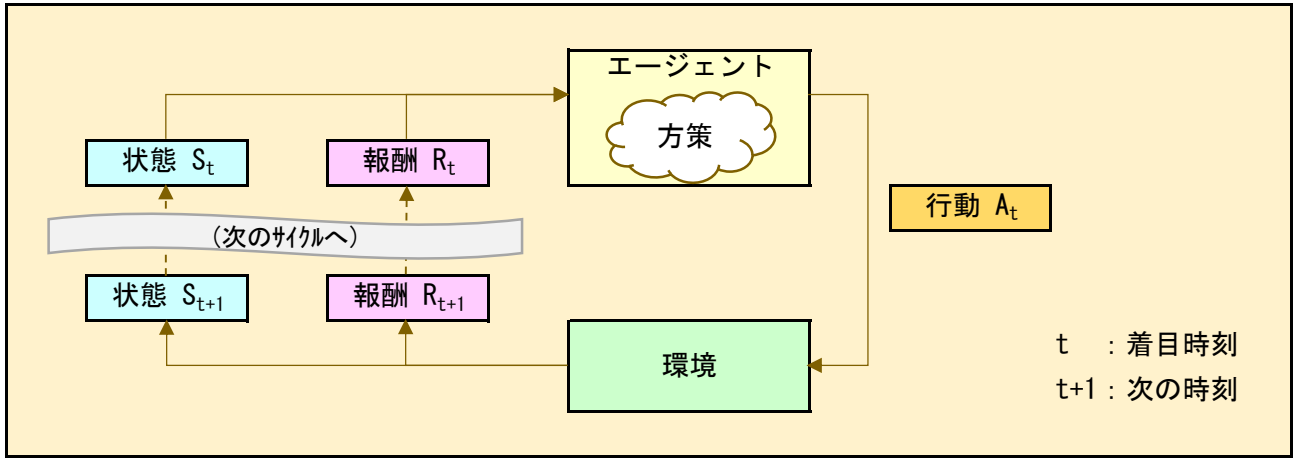
(2.3) 強化学習

・「強化学習（ｷｮｳｶｶﾞｸｼｭｳ、Reinforcement learning)」は、
「認知」した状況の下で最適な「行動」を「判断」すること、すなわち「制御」を学習する方法です。

・ｹﾞｰﾑなどでは、『その時の「状態」を認知し、最終的な「報酬」を最大化するように、次にどんな「方策」を講じるか判断して
「行動」に移す』ということを繰り返します。「強化学習」では、この流れを学習して、制御を行います。

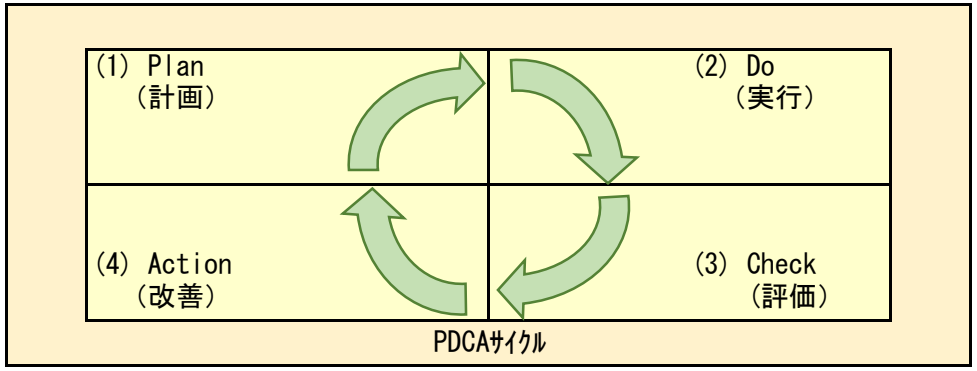
・強化学習では、「エージェント（agent、制御器)」が「環境（environment)」との相互作用を通じて情報収集しながら、
「環境」を制御する方法を学習します。
「環境」は、状態とその変化を規定する制御対象の系(システム)のことです。
「エージェント」は、環境の置かれた状態にもとづいて行動を選択しながら環境を制御する主体で、制御プログラムにあたります。

・「エージェント」がある時刻 t において「行動(action) A_t」を通して「環境」に影響を及ぼす際に、
「エージェント」は、時刻 t における「状態（state) S_t」をみて、
何らかの「方策（policy)」に基づいて「行動 A_t」を決定します。
一方、「環境」は、「エージェント」の「行動 A_t」がもたらす影響に応じて「報酬（reward) R_{t+1}」を「エージェント」に返すと同時に、
「状態（state)」を更新($S_t \Rightarrow S_{t+1}$)します。
こうした「エージェント」と「環境」のやりとりを繰り返しつつ、
最終的な「報酬」を最大化するように、「エージェント」の「方策」を最適化することが、強化学習の目的です。



・「強化学習」は、「PDCAサイクル（ﾋﾟｰﾃﾞｨｰｼｰｴｰｻｲｸﾙ、PDCA cycle、plan-do-check-act cycle)」と似ています。
これは、「Plan（計画)」「Do（実行)」「Check（評価)」「Action（改善)」の四段階を繰り返すことにより、
業務を継続的に改善する手法です。

(1) Plan（計画）　：従来の実績や将来の予測などをもとにして業務計画を作成する。
(2) Do（実行）　：計画に沿って業務を行う。
(3) Check（評価）　：業務の実施が計画に沿っているかどうかを評価する。
(4) Action（改善）　：実施が計画に沿っていない部分を調べて改善をする。



(2.4) 転移学習

・上記の分類とは異なりますが、学習済みﾓﾃﾞﾙの一部を学習しなおして使う「転移学習 (ﾃﾝｲｶﾞｸｼｭｳ、Transfer Learning)」という学習方法があります。
転移学習は、ある領域で学習させたﾓﾃﾞﾙを、別の領域に適応させる技術です。
具体的には、広くﾃﾞｰﾀが手に入る領域で学習させたﾓﾃﾞﾙを少ないﾃﾞｰﾀしかない領域に適応させたり、
ｼﾐｭﾚｰﾀ環境で学習させたﾓﾃﾞﾙを現実に適応させたりする技術です。

【出典・参考】

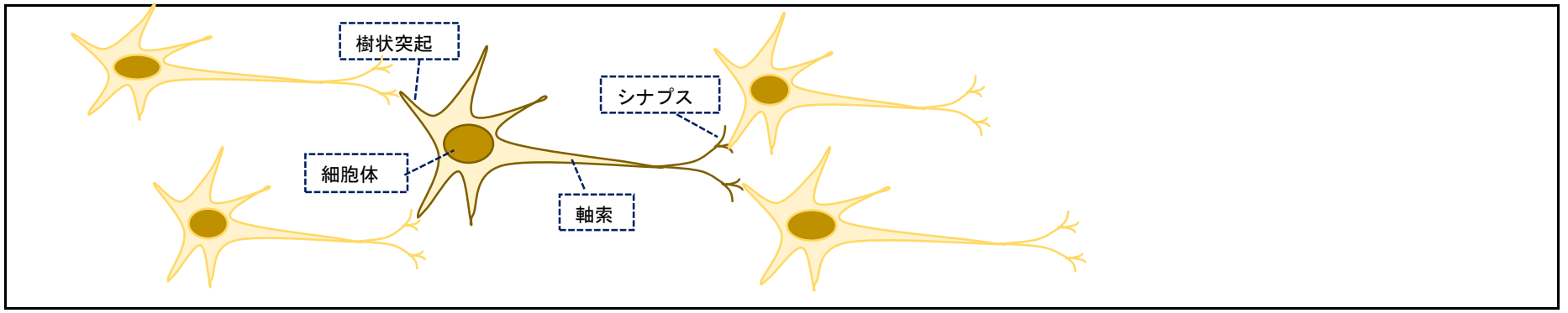
機械学習⇒ https://ja.wikipedia.org/wiki/機械学習
教師あり学習⇒ https://ja.wikipedia.org/wiki/教師あり学習
教師あり学習⇒「現場で使える！強化学習・深層学習による探索と制御入門」(2019年07月 翔泳社)
教師なし学習⇒「現場で使える！Python 機械学習入門」(2019年05月 翔泳社 大曾根圭輔 他著)
k平均法⇒ https://ja.wikipedia.org/wiki/K平均法
次元削減⇒ https://qiita.com/aya_taka/items/4d3996b3f15aa712a54f
強化学習⇒「現場で使える！強化学習・深層学習による探索と制御入門」(2019年07月 翔泳社)
PDCAｻｲｸﾙ⇒ https://flhouse.co.jp/article/178/pdca
PDCAｻｲｸﾙ⇒ https://ja.wikipedia.org/wiki/PDCAサイクル
転移学習⇒ https://qiita.com/icoxfog417/items/48cbf087dd22f1f8c6f4

(3) 脳神経系のモデル化

（※ この章は、「ｾﾐﾅｰ2回目」の内容を含みます。）

(3.1) 神経細胞間の信号伝達の様子

・２０世紀の初頭に、人間の脳は「ﾆｭｰﾛﾝ（neuron）」と呼ばれる神経細胞が、「ｼﾅﾌﾟｽ（synapse）」と呼ばれる結合部位を介して、
多数結合してできているﾈｯﾄﾜｰｸであることが示されました（下図）。
神経細胞は主に、細胞核のある細胞体、他の細胞からの入力を受ける樹状突起、他の細胞に出力する軸索から構成されています。
樹状突起が受け取った信号は細胞体で処理され、軸索を通って、次の神経細胞に伝達されます。

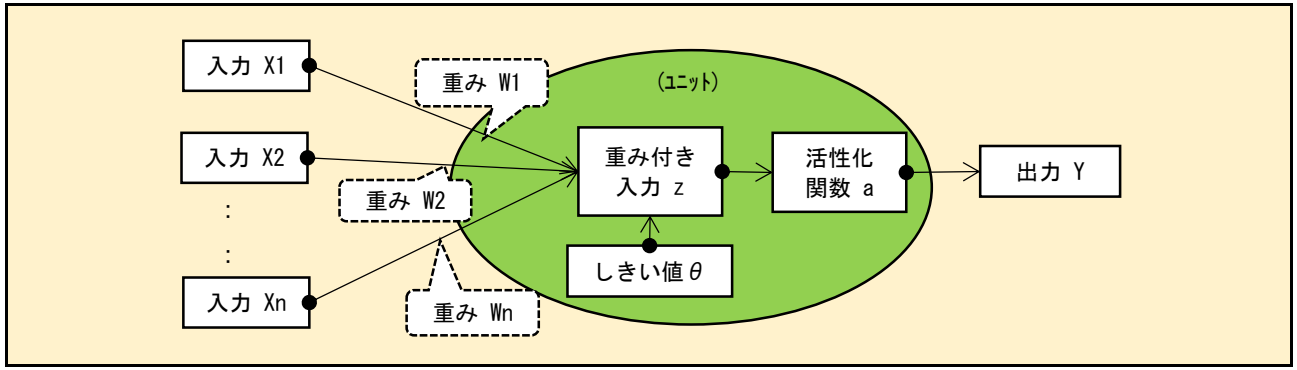


細胞体　　：細胞核とそれを取り巻く細胞質からなり、細胞としての機能はほとんどここで行われます。
　　樹状突起と軸索が会合する部位でもあります。
樹状突起　：細胞体から木の枝のように分岐しながら広がる構造であり、他の神経細胞などから信号を受け取る働きをします。
　　一つの神経細胞に、樹状突起は複数あります。
軸索　　：細胞体から延びている突起状の構造で、神経細胞において信号の出力を担います。
　　一つの神経細胞に、軸索は基本的には一本だけあります。
ｼﾅﾌﾟｽ　　：前の細胞の軸索終末と後ろの細胞の樹状突起の間の情報を伝達する部分にある、伝達構造です。

【出典・参考】
神経細胞⇒ https://ja.wikipedia.org/wiki/神経細胞
「深層学習 Deep Learning」人工知能学会監修（近代科学社 2015年11月）

(3.2) 神経細胞間の信号伝達のﾓﾃﾞﾙ化

・個々の神経細胞間の信号伝達の様子を以下のようにﾓﾃﾞﾙ化します。これを「形式ﾆｭｰﾛﾝ」「ﾕﾆｯﾄ」「ﾉｰﾄﾞ」または「ｾﾙ」と呼びます。



・関連する変数名・関数名は、以下のとおり：

Xi　：ﾆｭｰﾛﾝ i （1～n）からの入力信号
Wi　：ﾆｭｰﾛﾝ i （1～n）からの入力信号Xi の重み
θ　：入力信号の重み付き線形和（ $\Sigma^{n}_{i=1}\ (X_i * W_i)$ ）に対し、出力信号の有無を判別する為のしきい値
b　：バイアス（しきい値θの符号を変えたもの b = −θ）
z　：重み付き入力
a　：活性化関数（ｶｯｾｲｶｶﾝｽｳ、activation function、伝達関数(ﾃﾞﾝﾀﾂｶﾝｽｳ、transfer function)ともいいます）
Y　：出力信号

と表現した時、重み付き入力 z は以下の式で与えられます：

$$z = \{ X_1*W_1 + X_2*W_2 + \cdots + X_n*W_n \} - \theta$$
$$= \{ \Sigma^{n}_{i=1}\ (X_i * W_i) \} - \theta$$
$$= \{ \Sigma^{n}_{i=1}\ (X_i * W_i) \} + b$$

入力信号の重み付き線形和(括弧{}内の値) ＜ しきい値θ ならば、z ＜ 0
入力信号の重み付き線形和(括弧{}内の値) ≧ しきい値θ ならば、z ≧ 0

・出力信号 Y は、重み付き入力 z を入力変数として、活性化関数 a で、以下の式で与えられます：
活性化関数 a は、重み付き入力 z に応じて、出力信号 Y の大きさを規定する関数です：

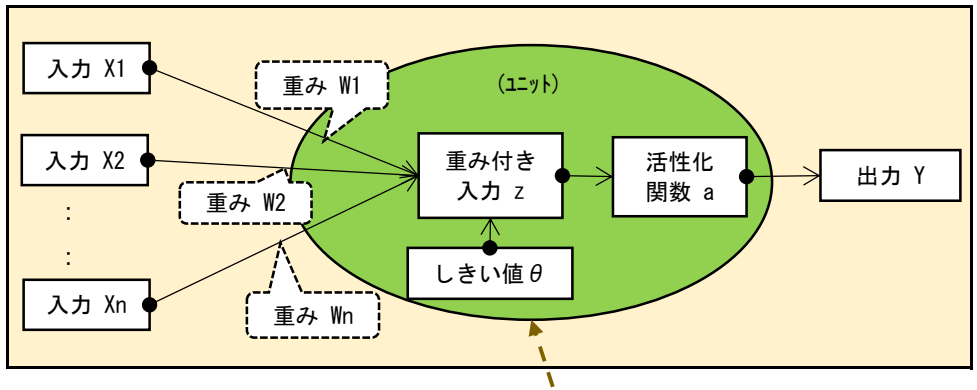
$$Y = a(z)$$
$$= a(\{ \Sigma^{n}_{i=1}\ (X_i * W_i) \} + b)$$

(3.3) ﾆｭｰﾗﾙﾈｯﾄﾜｰｸと適用ﾓﾃﾞﾙの選択

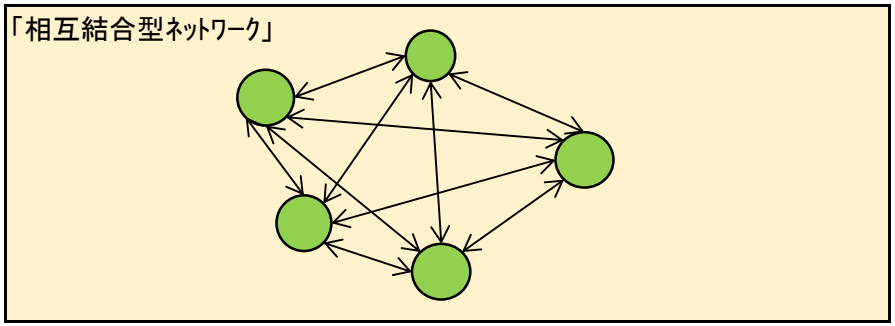
・「ﾕﾆｯﾄ（形式ﾆｭｰﾛﾝ）」を複数つなげて「数理的な神経回路網」を構成したものが
「ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ（Neural Network, NN）」で、「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」と「相互結合型ﾈｯﾄﾜｰｸ」があります。

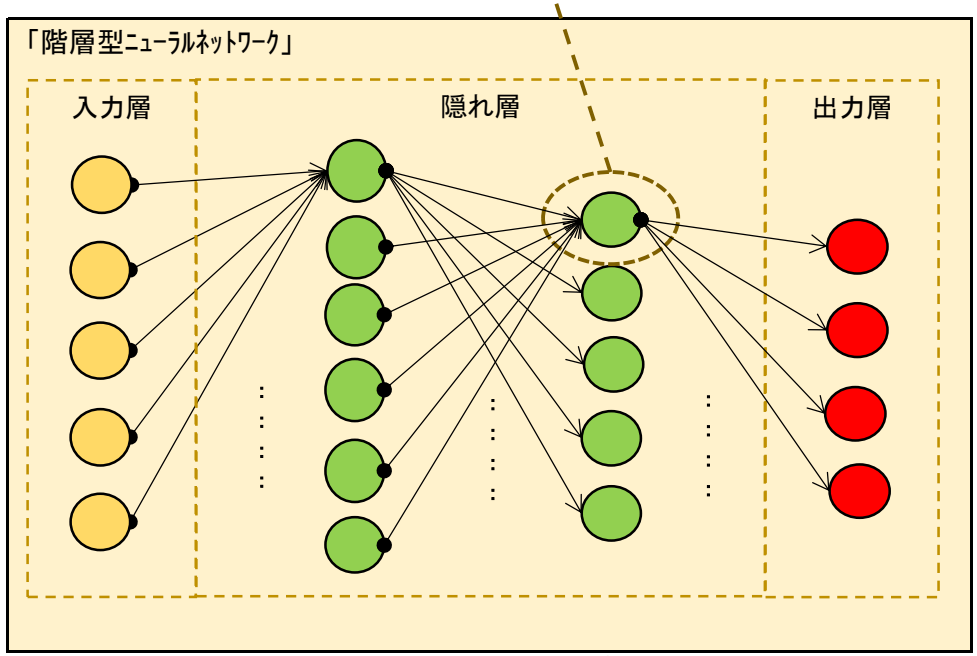
・「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」は、右図のように数層から構成されます。

入力層	ﾆｭｰﾗﾙﾈｯﾄﾜｰｸへの情報を取り込む層
隠れ層	入力側からの信号に対して、重みづけ評価を行いながら、出力側へ信号伝達する層（中間層とも言います）
出力層	ﾆｭｰﾗﾙﾈｯﾄﾜｰｸが算出した結果を出力する層



・「相互結合型ﾈｯﾄﾜｰｸ」は、下図のように、層がなくﾕﾆｯﾄ同士が互いに
結合しています。このﾕﾆｯﾄの持つ値により、ある状態を記憶することができ、
これを連想記憶と言います。
相互結合型ﾈｯﾄﾜｰｸは、制約ﾎﾞﾙﾂﾏﾝﾏｼﾝを起源としており、
それを深層化した深層ﾎﾞﾙﾂﾏﾝﾏｼﾝ、深層信念ﾈｯﾄﾜｰｸなどがあります。





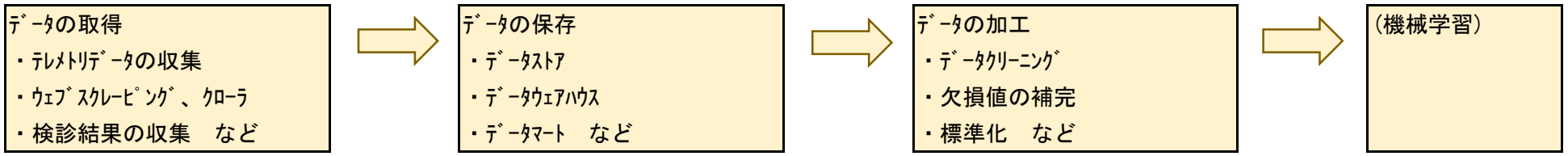
：入力層のﾕﾆｯﾄ
：中間層のﾕﾆｯﾄ
：出力層のﾕﾆｯﾄ

・目的の為にどのような「ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」のﾓﾃﾞﾙを適用するか
については、「ｾﾐﾅｰ2回目　機械学習のモデルとアルゴリズム」
を参考にしてください。

(4) 前処理

(4.1) ﾃﾞｰﾀを機械学習ﾓﾃﾞﾙに入力するまでの流れ

・ﾃﾞｰﾀを機械学習ﾓﾃﾞﾙに入力するまでの流れは以下の様になります：



・「ﾃﾚﾒﾄﾘﾃﾞｰﾀ (telemetry data)」とは、「ﾃﾚﾒｰﾀ(遠隔計測装置)」を使って、遠隔地の測定結果をｾﾝﾀに送信して収集したﾃﾞｰﾀのことです。
「ｱﾒﾀﾞｽ (AMeDAS：Automated Meteorological Data Acquisition System：自動気象ﾃﾞｰﾀ収集ｼｽﾃﾑ)」で収集されるﾃﾞｰﾀや、
ｶﾞｽや電気のﾒｰﾀｰ、自動販売機の売り上げ管理ｼｽﾃﾑなどで収集されるﾃﾞｰﾀがそれにあたります。

・「ｳｪﾌﾞｽｸﾚｲﾋﾟﾝｸﾞ (Web scraping)」は、ｳｪﾌﾞｻｲﾄから情報を抽出するｺﾝﾋﾟｭｰﾀｿﾌﾄｳｪｱ技術のことです。
「ｸﾛｰﾗ (Crawler)」とは、ｳｪﾌﾞ上の文書や画像などを周期的に取得し、自動的にﾃﾞｰﾀﾍﾞｰｽ化するﾌﾟﾛｸﾞﾗﾑのことです。

・「ﾃﾞｰﾀｽﾄｱ (data store)」とは、ﾃﾞｰﾀを決められた形式で記憶装置などに永続的に保存・蓄積するｿﾌﾄｳｪｱ、および、そのような機能・処理のことです。ﾃﾞｰﾀを保存する機能一般を抽象的に表す用語で、ﾌｧｲﾙｼｽﾃﾑやﾃﾞｰﾀﾍﾞｰｽもﾃﾞｰﾀｽﾄｱの具体的な実装の一つということができます。

・「ﾃﾞｰﾀｳｪｱﾊｳｽ (Data Ware House, DWH)」とは、意思決定のために、基幹系などの複数ｼｽﾃﾑから、必要なﾃﾞｰﾀを収集し、
目的別に再構成して時系列に蓄積した統合ﾃﾞｰﾀﾍﾞｰｽで、ﾃﾞｰﾀ分析や意思決定に役立てます。

・「ﾃﾞｰﾀﾏｰﾄ (data mart)」とは、企業などで情報ｼｽﾃﾑに記録・蓄積されたﾃﾞｰﾀから、利用部門や用途、目的などに応じて
必要なものだけを抽出、集計し、利用しやすい形に格納したﾃﾞｰﾀﾍﾞｰｽのことです。「ﾏｰﾄ (mart)」は「小売店」の意味です。

(4.2) ﾃﾞｰﾀの種類について

・ﾃﾞｰﾀの種類について、表現する情報の性質に基づき、数学・統計学的に分類するのが「尺度水準 (ｼｬｸﾄﾞｽｲｼﾞｭﾝ、Level of measurement)」です。
以下に、一覧表にしておきます：

尺度水準	説明
名義尺度 (ﾒｲｷﾞｼｬｸﾄﾞ、 nominal scale)	「名義尺度」とは、分類して名称を付与することに意味があり、順序には意味がない尺度のことを言います。 （例）性別、血液型、電話番号、ｶｰﾄﾞのＩＤ番号
順序尺度 (ｼﾞｭﾝｼﾞｮｼｬｸﾄﾞ、 ordinal scale)	「順序尺度」とは、間隔には意味がないが、大小や順序に意味がある尺度のことを言います。 「順位尺度 (ｼﾞｭﾝｲｼｬｸﾄﾞ)」とも言います。 （例）売上げﾗﾝｷﾝｸﾞの順位、成績の5段階評価
間隔尺度 (ｶﾝｶｸｼｬｸﾄﾞ、 interval scale)	「間隔尺度」とは、大小関係に加えて数値の差が意味をもつ尺度で、間隔に意味があります。 原点は相対的な意味しか持ちません。「距離尺度」とも言います。 「順序尺度」の性質も備えています。 （例）温度(℃)
比率尺度 (ﾋﾘﾂｼｬｸﾄﾞ、 ratio scale)	「比率尺度」とは、数値の大小関係・差・比に意味がある尺度で、絶対原点をもつ間隔尺度をいいます。 「比例尺度(ﾋﾚｲｼｬｸﾄﾞ)」「比尺度」とも言います。 「順序尺度」、「間隔尺度」の性質も備えています。 （例）絶対温度(K)、体重、身長、年齢

(4.3) 何を入力ﾃﾞｰﾀとするか

（※ この節は、「ｾﾐﾅｰ6回目」からの再掲を含みます。）

・何を入力ﾃﾞｰﾀとするかについては、その適用業務の専門知識や経験などに基づいて決めることになります。

・入力ﾃﾞｰﾀとなる説明変数を選ぶ際に、変数間の多重共線性に気を付ける必要があります。
「多重共線性（ﾀｼﾞｭｳｷｮｳｾﾝｾｲ、multicollinearity)」(略称「ﾏﾙﾁｺ」)とは、
統計ﾓﾃﾞﾙの説明変数間の相関関係が高いときに解析計算が不安定になる、といった性質です。
これにより、正しく推計できなくなるといった現象が起こります。

（行列の逆行列計算では、行（または列）の間に線形の関係（一方の行が他方の行の定数倍であるなど）がある場合、
行列式が０になり計算できませんが、それに近い現象が発生しているとみて良いでしょう。）

・「多重共線性」の問題の最も一般的な解消法は、「相関関係が高いと考えられる説明変数を外すこと」です。

（例）コンビニの月間のアイスクリームの売り上げ
・目的変数:(1) コンビニの月間のアイスクリームの売り上げ
・説明変数:(1) 来客数
(2) 最高気温が30℃以上の日数
(3) 降雨日数
(4) 月間降水量
(5) 平均気温
という分析ﾓﾃﾞﾙを考えたとき、説明変数で「(3) 降雨日数」と「(4) 月間降水量」は相関が高く、
多重共線性が発生する可能性が高いとみられます。
この場合、(3)か(4)の何れか一方を説明変数として不採用とします。

(4.4) 入力ﾃﾞｰﾀのｸﾘｰﾆﾝｸﾞ

・入力ﾃﾞｰﾀ（説明変数）を統計処理するに先立って、
入力ﾃﾞｰﾀの中から重複や誤記、表記のぶれなどを探し出し、削除や修正などを行い、入力ﾃﾞｰﾀの品質を高める必要があります。
この作業を「ﾃﾞｰﾀｸﾘｰﾆﾝｸﾞ (data cleaning)」または「ﾃﾞｰﾀｸﾚﾝｼﾞﾝｸﾞ (data cleansing)」と言います。

・「ﾃﾞｰﾀｸﾘｰﾆﾝｸﾞ」は、使用する単位や書式や定義域(入力値の範囲)等を基準にして行います。
以下に様々な例を示します：

(例1)　電話番号の文字列を、半角のみにし、ﾊｲﾌﾝを取り除く。（"０１－２３４５－６７８９" ⇒ "0123456789"）
(例2)　住所の誤字を直す。（"東京都太田区" ⇒ "東京都大田区"）
(例3)　数字文字列の3桁区切り表記を、数字列に直す。（"123,456,789" ⇒ "123456789"）
(例4)　欠けているﾃﾞｰﾀを補完する。（「"123",,"789"」 ⇒ 「"123","456","789"」）
(例5)　名簿の重複ﾚｺｰﾄﾞを削除する。（「{"1","山田"},{"1","山田"},{"2","佐藤"}」 ⇒ 「{"1","山田"},{"2","佐藤"}」）
(例6)　長さの単位を[cm]から[mm]へ統一する。（「175」 ⇒ 「1750」）
(例7)　異常な地上気圧を(誤記と見做して)補正する。（「10240」 ⇒ 「1024」）
(例8)　異常な地上気圧を(計測機器故障と見做して)削除する。（「10240, 998, 30, 1013」 ⇒ 「998,1013」）

(4.5) 入力ﾃﾞｰﾀの欠損値の扱い

・入力ﾃﾞｰﾀ（説明変数）が数種類からなる場合、各入力ﾚｺｰﾄﾞに必ずしも全ての種類のﾃﾞｰﾀが揃っている訳ではありません。
欠けているﾃﾞｰﾀのことを「欠損値（ｹｯｿﾝﾁ、missing value)」と言います。

・「欠損値」の例を以下に示します：
(例1)　　ｱﾝｹｰﾄﾃﾞｰﾀの未回答項目
(例2)　　健康診断時の未検査項目

・欠損値がある場合、統計処理に支障があるため、以下の様な対応をします：
(1) 欠損値を含む行(ﾚｺｰﾄﾞ)を削除する。
(2) 欠損値を含む列(説明変数、目的変数)を削除する。
(3) 欠損値を何らかの値(指定値、平均値、内挿値、外挿値など)で補完する。

・Python のﾗｲﾌﾞﾗﾘ「Pandas」の「DataFrame」ｸﾗｽで欠損値対応のﾒｿｯﾄﾞが提供されているので、実装時に参照すると良いでしょう。
（「Missing data handling」の機能）

(4.6) 入力ﾃﾞｰﾀの標準化

・平均と分散についての公式を用いて、平均μ、標準偏差σの確率変数 X を、平均０、標準偏差１の確率変数 Z へ変換することが出来ます。
この操作を「標準化 (ﾋｮｳｼﾞｭﾝｶ, standardization)」と言い、
Z を「標準化確率変数 (ﾋｮｳｼﾞｭﾝｶｶｸﾘﾂﾍﾝｽｳ, standardized random variable)」
と言います。確率変数 X から次式により標準化確率変数 Z を求めます：

標準化確率変数

$$Z = (X - \mu) / \sigma$$

μ ：平均値
σ ：標準偏差

・入力ﾃﾞｰﾀ（説明変数）が数種類からなる場合で、入力ﾃﾞｰﾀ毎の平均や分布が大きく異なる場合などに標準化を行います。
これには、以下の様な効果があります。

(1) 特定の特徴に偏った結果が出ることを防ぐことが出来ます。
(2) 異なる項目のﾃﾞｰﾀであってもその大小を比較できるようになります。

(4.7) 入力ﾃﾞｰﾀの加工による作成

・ﾓﾃﾞﾙの学習時用の入力ﾃﾞｰﾀが少ない場合、ﾃﾞｰﾀに以下の様な加工を加えることにより、入力ﾃﾞｰﾀを増やすことが出来ます。

(1)　画像の場合、回転移動、拡大・縮小、平行移動、輝度変換およびそれらの組合せで別の画像を作成する。
（これは、Keras の ImageDataGenerator ｸﾗｽを用いて実装可能です）

(2)　画像の場合、元の画像にノイズを加えることにより別の画像を作成する。
（これは、numpy を用いて、ﾏｽｷﾝｸﾞﾉｲｽﾞﾃﾞｰﾀやｶﾞｳｼｱﾝﾉｲｽﾞﾃﾞｰﾀを加える、といった実装で可能です）

(4.8) 入力ﾃﾞｰﾀの符号化

・入力ﾃﾞｰﾀを符号化してﾓﾃﾞﾙに取り込んだ方が良い場合があります。

・出力が K種類への分類ﾓﾃﾞﾙの場合、目的変数をK次元のﾍﾞｸﾄﾙとします。
ある入力ﾃﾞｰﾀが第k番目(k=1～K)のｸﾗｽへ分類されることが分かっている場合、
目的変数ﾍﾞｸﾄﾙの k番目の要素だけ 1 でそれ以外を 0 とする符号化を行うと便利です。
これを「1-of-K 符号化法 (ﾜﾝｵﾌﾞｹｰﾌｺﾞｳｶﾎｳ、One-of-K encoding)」と言います（「one-hot encoding」とも言います）。

（例）
5つのｸﾗｽに分類するﾓﾃﾞﾙの場合、n番目の入力ﾃﾞｰﾀ Xn がｶﾞ3番目のｸﾗｽに分類された場合、
目的変数ﾍﾞｸﾄﾙ Tn を「3」と表現するのではなく、次のように表現します：
Tn = (0,0,1,0,0)

【出典・参考】

全般⇒「現場で使える！Python 機械学習入門」(2019年05月 翔泳社 大曾根圭輔 他著)

ﾃﾚﾒﾄﾘﾃﾞｰﾀ⇒ https://kotobank.jp/word/テレメトリー-577329

ｳｪﾌﾞｽｸﾚｰﾋﾟﾝｸﾞ ⇒ https://ja.wikipedia.org/wiki/ウェブスクレイピング

ﾃﾞｰﾀｳｪｱﾊｳｽ⇒ http://www.advanlink.co.jp/crmconsultation/crm_dwh.html

ﾃﾞｰﾀﾏｰﾄ⇒ http://e-words.jp/w/ﾃﾞｰﾀﾏｰﾄ.html

ﾃﾞｰﾀｽﾄｱ⇒ http://e-words.jp/w/ﾃﾞｰﾀｽﾄｱ.html

全般⇒「現場で使える！Python 機械学習入門」(2019年05月 翔泳社 大曾根圭輔 他著)

尺度水準⇒ https://ja.wikipedia.org/wiki/尺度水準

尺度水準⇒ https://to-kei.net/basic/glossary/scale/

多重共線性⇒ https://xica.net/vno4ul5p/

多重共線性⇒ https://support.minitab.com/ja-jp/minitab/18/help-and-how-to/modeling-statistics/regression/supporting-topics/model-assumptions/multicollinearity-in-regression/

行列式の計算⇒「ベクトルと行列」 (1976年05月 現代数学社 横田一郎、他著)

ﾃﾞｰﾀｸﾘｰﾆﾝｸﾞ ⇒ https://fukaichi.jp/web-marketing-glossary-list/data-cleaning

ﾃﾞｰﾀｸﾘｰﾆﾝｸﾞ ⇒ http://e-words.jp/w/ﾃﾞｰﾀｸﾚﾝｼﾞﾝｸﾞ.html

欠損値⇒ https://www.weblio.jp/content/欠損値

欠損値,Pandas⇒ https://techacademy.jp/magazine/17697

Pandas⇒ https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

標準化確率変数⇒ https://k-san.link/standardized/

標準化⇒ https://bellcurve.jp/statistics/course/19647.html

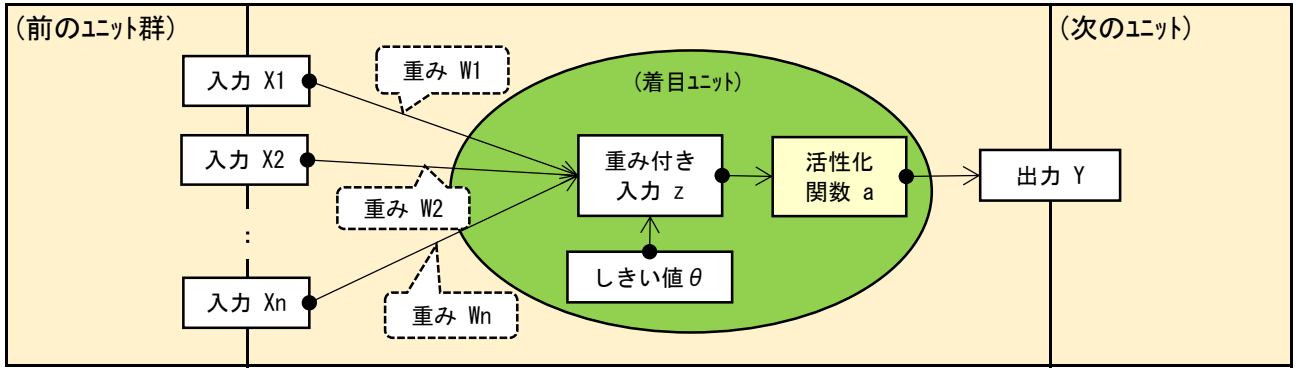
入力ﾃﾞｰﾀの加工⇒

「現場で使える！TensorFlow開発入門 Kerasによる深層学習ﾓﾃﾞﾙ構築手法」 (2018年04月 翔泳社 太田満久、須藤広大、黒澤匠雅、小田大輔　共著)

(5) 活性化関数

(※ この章は、「ｾﾐﾅｰ2回目」の内容を含みます。)

・あるﾕﾆｯﾄに着目した時、そのﾕﾆｯﾄへの入力信号の重み付き線形和にﾊﾞｲｱｽを加えたもの(下図で「重み付き入力 z」)を、
どのように活性化して次のﾕﾆｯﾄに渡すかを、「活性化関数 (ｶｯｾｲｶｶﾝｽｳ、Activation function)」で指定します。
「活性化関数」は「出力関数 (ｼｭﾂﾘｮｸｶﾝｽｳ、Output function)」とも言います。



・「活性化関数」には様々なものが考案されており、主なものを一覧表に示します。
このうち「ｾﾐﾅｰ7回目」で使用しているものは、「ReLU」「sigmoid」「ｿﾌﾄﾏｯｸｽ関数」の3つです。

・「ｿﾌﾄﾏｯｸｽ関数 (softmax function)」は、k個の入力値 Xi (i=1～k)の順序関係を保ちながら、
Xi を確率としての値 Yi (i=1～k、$0 \leqq Y_i \leqq 1$、$\Sigma_{i=1}^{k} Y_i = 1$) に変換する関数で、
k個のｸﾗｽへ分類する際の所属の確率を示すのによく用いられます。
「ｼｸﾞﾓｲﾄﾞ関数 (sigmoid function)」は「ｿﾌﾄﾏｯｸｽ関数」で分類するｸﾗｽ数を2個に限定したものに相当します。

(1) k 個のｸﾗｽへ分類する時のｿﾌﾄﾏｯｸｽ関数

$$Yi = \exp(X_i) \ / \ \Sigma^{k}_{j=1} \exp(X_j) \qquad (i=1 \sim k)$$

(2) 2個のｸﾗｽへ分類する時のｿﾌﾄﾏｯｸｽ関数

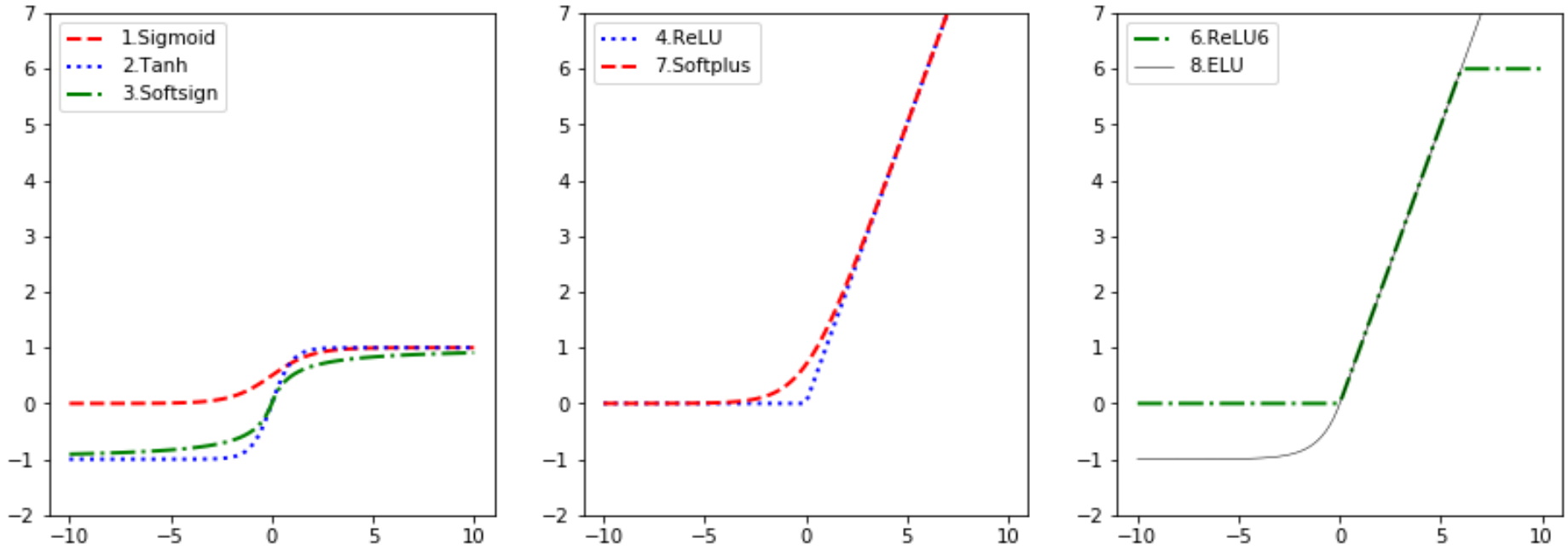
$$Y_1 = \exp(X_1) \ / \ \{\exp(X_1)+\exp(X_2)\} = 1/\{1+\exp(X_2-X_1)\} = 1/\{1+\exp(-Z)\} \qquad (Z=X_1-X_2) \qquad \cdots \text{ｼｸﾞﾓｲﾄﾞ関数}$$

$$Y_2 = \exp(X_2) \ / \ \{\exp(X_1)+\exp(X_2)\} = 1 - Y_1$$

・「ReLU (正規化線形関数)」は計算が容易で(正(負)の領域では微分が 1(0))、「sigmoid関数」で問題となる学習時の「誤差逆伝播法」での
「勾配消失問題 (ｺｳﾊﾞｲｼｮｳｼｯﾓﾝﾀﾞｲ、vanishing gradient problem)」もなく、一般的に使用されています。

No.	活性化関数名	和名	英語	式「Y = a(z)」	値域	tensorflowの関数名	備考
1	sigmoid	シグモイド関数（ロジスティック関数）	sigmoid function (logistic function)	1 / { 1 + exp(-z) }	0 ～ 1	sigmoid	微分可能で解析的に扱いが容易だが、誤差逆伝播法の誤差項が、学習中に0に近づいていく傾向にある為(勾配消失問題 Vanishing Gradient problem)、あまり使用されていません。 「ｿﾌﾄﾏｯｸｽ関数」で分類するｸﾗｽ数を2個に限定したものに相当します(1986年)。
2	tanh	ハイパボリックタンジェント（双曲線正接関数）	hyperbolic tangent	{ exp(z) - exp(-z) } / { exp(z) + exp(-z) }	-1 ～ +1	tanh	活性化関数は原点を通すべきと言う考えから、標準シグモイド関数よりもそれを線形変換した tanh の方が良いと提案されました。(1998年)
3	softsign	ソフトサイン関数	softsign function	z / { abs(z) + 1 }	-1 ～ +1	softsign	正弦関数に対する連続近似として位置づけられます。 (2010年 Xavier Glorot)
4	ReLU	正規化線形関数（ランプ関数）	Rectified Linear Unit	max(0、z)	0 ～ ∞	relu	ReLU (Rectifired Linear Units) は、入力値が負の値の場合は0、正の値の場合は入力値をそのまま出力します。 最も一般的で基本的な方法です。(2011年 Xavier Glorot)
5	Clipped ReLU	Clipped ReLU	Clipped Rectified Linear Unit	min(max(0,z),clipVal) clipVal：0以上の浮動小数点数。正の部分の最大値。	0 ～ clipVal	---	Clipped ReLU は、ReLUの特別なバージョンで、出力値が一定の大きさ以上にはならないように変更されたものです。ReLUでは、入力が正の値のときに線形な変換が適用されますが、Clipped ReLU では入力に対して、出力は頭打ち状態 (Clipping) になります。
6	ReLU6	正規化線形関数（ランプ関数）	Rectified Linear Unit 6	min(max(0,z),6)	0 ～ +6	relu6	ReLU関数で上限を6としたもの。 Clipped RELUの一つで、計算速度が速く、値の消失や発散の問題にも悩まされません。
7	softplus	ソフトプラス関数	softplus function	log(exp(z) + 1)	0 ～ ∞	softplus	ReLU関数を滑らかにしたものです。
8	ELU		Exponential Linear Unit	exp(z) -1 for z< 0 z for z>=0	-1 ～ ∞	elu	ELUは、ReLUの特別なバージョンで、負の入力に対して指数関数（Exp）から1を引いた値を適用しています。
9	LeakyReLU	LeakyReLU	Leaky Rectified Linear Unit	alpha * z for z< 0 z for z>=0 alpha：0以上の浮動小数点数。負の部分の傾き。	-∞ ～ ∞	leaky_relu	LeakyReLUは、ReLUの特別なバージョンで、ﾕﾆｯﾄがアクティブでないとき（z < 0）でも微少な勾配を可能とします。これにより勾配消失を防ぎ、学習速度を向上させる効果が期待できます。DCGAN（Deep Convolutional Generative Adversarial Network）と呼ばれるモデルなどで利用されます。
10	softmax	ソフトマックス関数	softmax function	$Y_i = \exp(X_i) / \Sigma^{k}_{j=1}\exp(X_j)$ (i=1 ～ k)	0 ～ 1	softmax	出力が k 個ある場合に、k個の入力値 Xi (i=1～k)の順序関係を保ちながら、確率としての値 Yi (i=1～k、$0 \leqq Yi \leqq 1$、$\Sigma_{i=1}^{k} Y_i=1$) に変換する関数。k個のｸﾗｽへ分類する際の所属の確率を示すのによく用いられます。

・上記で紹介した「活性化関数」のうち、幾つかを図示します。



（活性化関数とその描画の実装例）

・活性化関数はﾗｲﾌﾞﾗﾘで提供されているか、提供されているﾗｲﾌﾞﾗﾘのﾒｿｯﾄﾞの引数として活性化関数の名前を指定するだけでも実装できますが、ここでは、敢えて関数の定義のまま実装します。

（ ﾘｽﾄ08-(05)-1_様々な活性化関数 ）

```
#*****************************************************************/
# ﾘｽﾄ08-(05)-1_様々な活性化関数
#*****************************************************************/
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#(1) ｼｸﾞﾓｲﾄﾞ関数
def f_sigmoid(x):
    return 1 / ( 1 + np.exp(-x) )

#(2) ﾊｲﾊﾟﾎﾞﾘｯｸﾀﾝｼﾞｪﾝﾄ（双曲線正接関数）
def f_tanh(x):
    return  (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))

#(3) ソフトサイン関数
def f_softsign(x):
    return  x / (np.abs(x) + 1)

#(4) 正規化線形関数（ランプ関数）
def f_ReLU(x):
    xcnt = len(x)
    ylist = np.zeros(xcnt)
    for ii in range(xcnt):
        if x[ii] < 0:
            ylist[ii] = 0
        else:
            ylist[ii] = x[ii]
    return ylist

#(6) 正規化線形関数（ランプ関数、上限＝6）
def f_ReLU6(x):
    xcnt = len(x)
    ylist = np.zeros(xcnt)
    for ii in range(xcnt):
        if x[ii] < 0:
            ylist[ii] = 0
        elif x[ii] > 6:
            ylist[ii] = 6
        else:
            ylist[ii] = xlist[ii]
    return ylist
```

```
#(7) ソフトプラス関数
def  f_softplus(x):
    return  np.log(np.exp(x) + 1)

#(8) Exponential Linear Unit
def  f_ELU(x):
    xcnt = len(x)
    ylist = np.zeros(xcnt)
    for ii in range(xcnt):
        if x[ii] < 0:
            ylist[ii] = np.exp(x[ii]) -1
        else:
            ylist[ii] = x[ii]
    return ylist

# x データ列
xlist = np.linspace(-10, 10, 100)

# y データ列
y_sigmoid = f_sigmoid(xlist)
y_tanh = f_tanh(xlist)
y_softsign = f_softsign(xlist)
y_ReLU = f_ReLU(xlist)
y_ReLU6 = f_ReLU6(xlist)
y_softplus = f_softplus(xlist)
y_ELU = f_ELU(xlist)

# グラフ表示
plt.figure(figsize=(14,5))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

plt.subplot(1,3,1)
plt.plot(xlist, y_sigmoid, 'r--', label='1.Sigmoid', linewidth=2)
plt.plot(xlist, y_tanh, 'b:', label='2.Tanh', linewidth=2)
plt.plot(xlist, y_softsign, 'g-.', label='3.Softsign', linewidth=2)
plt.ylim([-2,7])
plt.legend(loc='upper left')
```

```
plt.subplot(1,3,2)
plt.plot(xlist, y_ReLU, 'b:', label='4.ReLU', linewidth=2)
plt.plot(xlist, y_softplus, 'r--', label='7.Softplus', linewidth=2)
plt.ylim([-2,7])
plt.legend(loc='upper left')

plt.subplot(1,3,3)
plt.plot(xlist, y_ReLU6, 'g-.', label='6.ReLU6', linewidth=2)
plt.plot(xlist, y_ELU, 'k-', label='8.ELU', linewidth=0.5)
plt.ylim([-2,7])
plt.legend(loc='upper left')

plt.show()
```

【出典・参考】

活性化関数全般　⇒ http://yagami12.hatenablog.com/entry/2017/09/17/111935
活性化関数全般　⇒ https://www.renom.jp/ja/notebooks/tutorial/basic_algorithm/activation_types/notebook.html
keras ﾗｲﾌﾞﾗﾘ　⇒ https://keras.io/ja/activations/
keras ﾗｲﾌﾞﾗﾘ　⇒ https://keras.io/ja/layers/advanced-activations/
ReLU6　⇒ https://www.tensorflow.org/api_docs/python/tf/nn/relu6
ReLU6　⇒ http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf
LeakyReLU　⇒ https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf
LeakyReLU　⇒ https://www.tensorflow.org/api_docs/python/tf/nn/leaky_relu
ELU　⇒ https://arxiv.org/pdf/1511.07289v1.pdf
sigmoid関数とｿﾌﾄﾏｯｸｽ関数の関係⇒ https://mathtrain.jp/softmax

(6) 損失関数

(6.1) 損失関数・誤差関数・ｺｽﾄ関数・目的関数

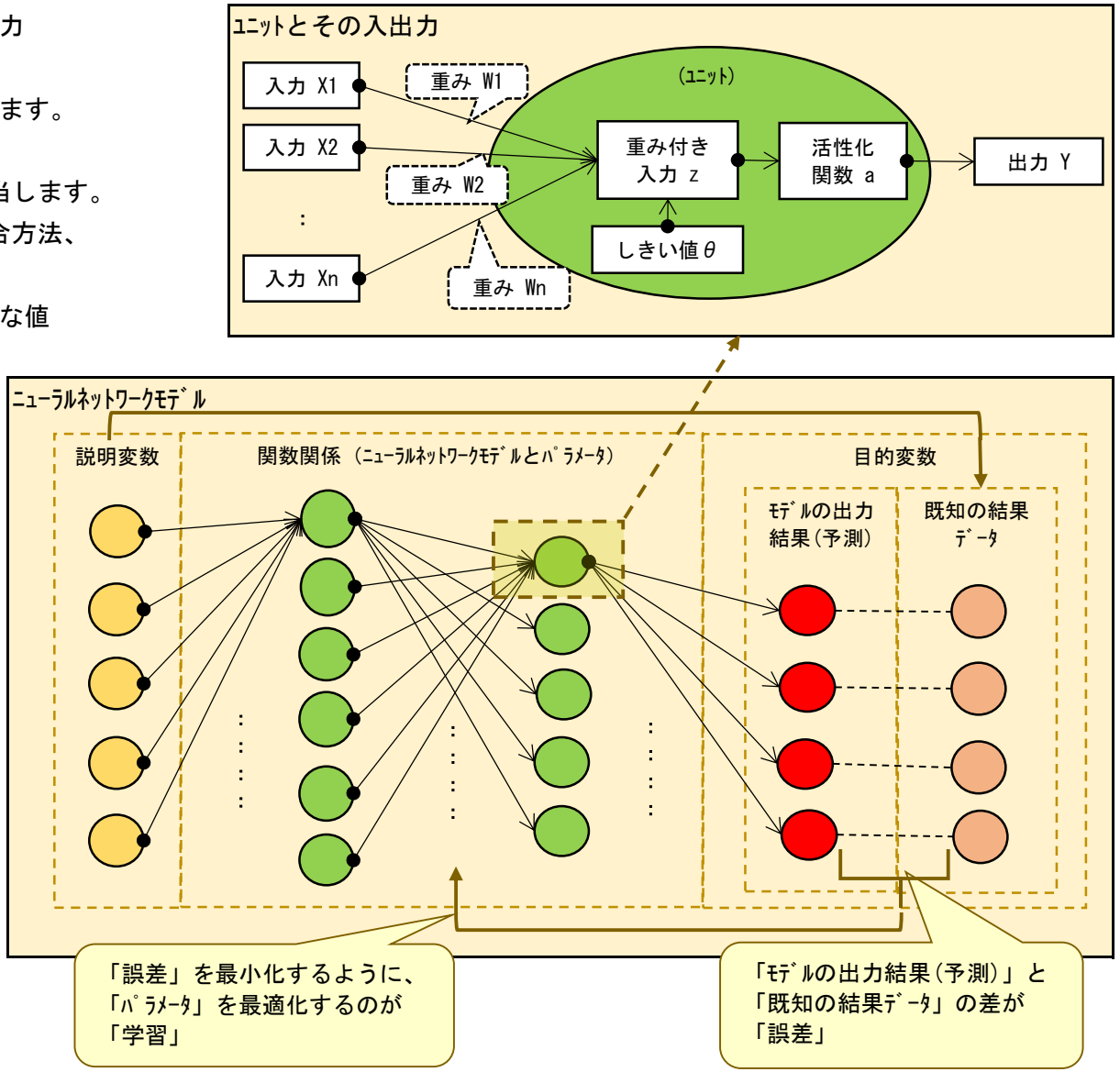
・「教師あり学習」では、一方を入力（説明変数）、他方を出力（目的変数）とする関数関係として記述し、関数の出力を観測ﾃﾞｰﾀ（目的変数）に近づけるように、関数関係を学習します。

・この関数関係は「ﾆｭｰﾗﾙﾈｯﾄﾜｰｸﾓﾃﾞﾙ」とその「ﾊﾟﾗﾒｰﾀ」に相当します。
「ﾆｭｰﾗﾙﾈｯﾄﾜｰｸﾓﾃﾞﾙ」は使用する層やﾕﾆｯﾄの数、ﾕﾆｯﾄ間の結合方法、使用する活性化関数などのﾈｯﾄﾜｰｸ構成で、
「ﾊﾟﾗﾒｰﾀ」は、各ﾕﾆｯﾄの重みWiとしきい値θなどの調整可能な値のことです。

・説明変数と目的変数の関係を良く説明できるﾓﾃﾞﾙが「良いﾓﾃﾞﾙ」であり、「ﾓﾃﾞﾙの出力結果(予測)」と「既知の結果ﾃﾞｰﾀ」との差(「誤差（ｺﾞｻ、error)」)が最小になるように、「ﾊﾟﾗﾒｰﾀ」を最適化するのが「学習」です。

・「学習」を通して最小化したいのはこの誤差であり、誤差を表す関数のことを、
「誤差関数（ｺﾞｻｶﾝｽｳ、error function)」、又は
「損失関数（ｿﾝｼﾂｶﾝｽｳ、loss function)」、又は
「ｺｽﾄ関数（ｺｽﾄｶﾝｽｳ、cost function)」と言います。

これらは「最小化する」という目的があるので、
「目的関数（ﾓｸﾃｷｶﾝｽｳ、objective function)」
とも言います。



(6.2) 尤度と損失関数

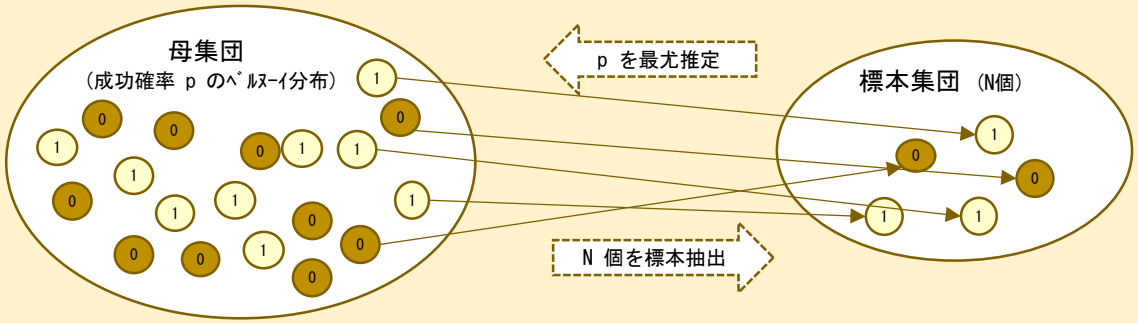
・統計学において、未知または既知のある確率分布の下で、確率変数の結果が出現する場合に、
確率変数の出現結果からみて、元々の確率分布を推測する時の「尤もらしさ（ﾓｯﾄﾓﾗｼｻ）」を表す数値を、
「尤度（ﾕｳﾄﾞ、likelihood）」といい、「尤度」を計算する関数を「尤度関数（ﾕｳﾄﾞｶﾝｽｳ、likelihood function）」と言います。

・この「尤度」を最大化するように元の確率分布を推測することを「最尤推定（ｻｲﾕｳｽｲﾃｲ、maximum likelihood estimation、MLE）」と言います。
「離散型確率分布」の場合は「確率質量関数」を、「連続型確率分布」の場合は「確率密度関数」を推定することになります。

・例として、ﾍﾞﾙﾇｰｲ分布の最尤推定方法を示します。　（ﾍﾞﾙﾇｰｲ分布については、「ｾﾐﾅｰ6回目」を参照のこと）

ﾍﾞﾙﾇｰｲ分布の最尤推定方法

標本 $\{ X_1, X_2, \cdots, X_N \}$ はある母集団からの N 個の無作為標本であるとします。
その母集団が従う確率分布がﾍﾞﾙﾇｰｲ分布で、１回のﾍﾞﾙﾇｰｲ試行で成功する(1となる)確率を p（未知数）とします。
この時、失敗する(0となる)確率は 1-p となります。



この試行で標本 $\{ X_1, X_2, \cdots, X_N \}$ の観測値 $\{ t_1, t_2, \cdots, t_N \}$ の各 t_n は {1,0} の何れかを各々 {p, 1-p} の確率で
取ります。各 X_n が値 t_n を取るというﾍﾞﾙﾇｰｲ分布の確率質量関数 $P(X_n=t_n)$ は次式となります：

$$P(X_n=t_n) = p^{t_n} (1-p)^{1-t_n} \quad \text{for } t_n \in \{0,1\},\ n=1 \sim N \qquad \cdots(式6.2\text{-}1)$$

この式を元に、標本 $\{ X_1, X_2, \cdots, X_N \}$ の観測値が $\{ t_1, t_2, \cdots, t_N \}$ となる確率(同時確率)は、
各観測結果 X_n についての確率(式6.2-1)の直積で以下の式になります：

$$P(X_1=t_1, X_2=t_2, \cdots, X_N=t_N) = \prod_{n=1}^{N} P(X_n=t_n) = \prod_{n=1}^{N} p^{t_n} (1-p)^{1-t_n}$$
$$= p^{z} (1-p)^{N-z} \qquad (z\text{は}t_n=1\text{となる}t_n\text{の総和：}z=\sum_{n=1}^{N} t_n)$$

これが観測値が { $t_1, t_2, \cdots, t_N$ } の下での成功確率 p を推定する為の尤度関数 L となります。

$$L(p|t_1, t_2, \cdots, t_N) = \prod_{n=1}^{N} p^{tn} (1-p)^{1-tn} \quad \cdots(式6.2-2)$$

$$= p^{z} (1-p)^{N-z} \quad (zは t_n=1 となる t_n の総和：z=\sum_{n=1}^{N} t_n) \quad \cdots(式6.2-3)$$

p の最尤推定値は、尤度関数 L を最大にする p の値になります。
ここで自然対数関数 log(x) が x について単調増加であり、関数値の大小関係が x の大小関係を保つという性質から、
尤度関数 L を最大にする p の値は、この対数をとった「対数尤度関数」の値も最大にすることが分かります。
対数尤度関数は以下の様になります（log関数の性質を用いて式を変形します）：

$$\log\{ L(p|t_1, t_2, \cdots, t_N) \} = \log\{ \prod_{n=1}^{N} p^{tn} (1-p)^{1-tn} \} \quad \cdots(式6.2-2)より$$

$$= \sum_{n=1}^{N} \{ t_n * \log p + (1-t_n) * \log(1-p) \} \quad \cdots(式6.2-4)$$

$$= \log\{p^{z} (1-p)^{N-z} \} \quad \cdots(式6.2-3)より$$

$$= z * \log p + (N-z) * \log(1-p) \quad \cdots(式6.2-5)$$

対数尤度関数 log(L) の p についての偏微分が 0 を与える p が最尤推定値であることの条件となります：

$$\partial \log(L) / \partial p = z/p - (N-z)/(1-p) = 0$$

$$\therefore \quad p = z/N$$

$$\therefore \quad p = 1/N \sum_{n=1}^{N} t_n \quad \cdots(式6.2-7)$$

これは、標本平均に一致しています。
例えば、標本{ $t_1, t_2, \cdots, t_N$ } のうち、50% が 1（成功） ならば p=0.5 が最尤推定値となります。

・「尤度」を最大化することは、ﾓﾃﾞﾙの学習時の評価指数である「損失関数」の値を最小化することになります。
この関係から、上記の例のようなｸﾗｽ分類では「対数尤度関数（ﾀｲｽｳﾕｳﾄﾞｶﾝｽｳ、log likelihood function)」にﾏｲﾅｽ1(-1) をかけて
更に標本ｻｲｽﾞN で平均を取った関数を「損失関数」として扱います。

「対数尤度関数」(式6.2-4)に -1/N をかけて得られた関数は、
次に述べる損失関数一覧の「2ｸﾗｽ分類の場合の交差ｴﾝﾄﾛﾋﾟｰ誤差」CE(式6.3-4)になります。

$$CE = -1/N \sum_{n=1}^{N} \{t_n \log(y_n) + (1-t_n) \log(1-y_n)\} \quad \cdots(式6.3-4)$$

(6.3) 損失関数一覧

・「損失関数」には様々なものが提案されており、次表に代表的なものを幾つか掲載します。

No.	損失関数名	英語	略称	式	用途・特徴など
1	平均二乗誤差 (ﾍｲｷﾝﾆｼﾞｮｳｺﾞｻ)	Mean Squared Error	MSE	$MSE = 1/N \sum_{n=1}^{N} (t_n - y_n)^2$ …(式6.3-1) t_n ：実際の値 y_n ：ﾓﾃﾞﾙの予測値 N ：ﾃﾞｰﾀ個数 (n=1～N)	回帰問題において平均二乗誤差は最も一般的で、線形回帰モデルやニューラルﾈｯﾄﾜｰｸ、決定木といった様々なモデルにおいて用いられます。 平均二乗誤差の性質として外れ値に対して敏感であることが挙げられます。
2	平均絶対誤差 (ﾍｲｷﾝｾﾞｯﾀｲｺﾞｻ)	Mean Absolute Error	MAE	$MSE = 1/N \sum_{n=1}^{N} \lvert t_n - y_n \rvert$ …(式6.3-2) t_n ：実際の値 y_n ：ﾓﾃﾞﾙの予測値 N ：ﾃﾞｰﾀ個数 (n=1～N)	平均絶対誤差の性質として外れ値に強いことが挙げられます。
3	平均二乗対数誤差 (ﾍｲｷﾝﾆｼﾞｮｳﾀｲｽｳｺﾞｻ)	Mean Squared Logarithmic Error	MSLE	$MSLE = 1/N \sum_{n=1}^{N} \{\log(1+t_n) - \log(1+y_n)\}^2$ …(式6.3-3) t_n ：実際の値 y_n ：ﾓﾃﾞﾙの予測値 N ：ﾃﾞｰﾀ個数 (n=1～N)	平均二乗対数誤差を用いたモデルは予測が実値を上回りやすくなるという傾向があります。
4	交差ｴﾝﾄﾛﾋﾟｰ誤差 (ｺｳｻｴﾝﾄﾛﾋﾟｰｺﾞｻ)	Cross Entropy Error	CE	2ｸﾗｽ分類の場合 $CE = -1/N \sum_{n=1}^{N} \{t_n \log(y_n) + (1-t_n)\log(1-y_n)\}$ …(式6.3-4) t_n ：入力x_n のｸﾗｽ1への所属確率の実際の値 (0,1) y_n ：入力x_n のｸﾗｽ1への所属確率の予測値 (0～1) N ：ﾃﾞｰﾀ個数 Kｸﾗｽ分類の場合 $CE = -1/N \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log \{ q(Y=k\|X=x_n) \}$ …(式6.3-5) t_{nk} ：入力x_n のｸﾗｽkへの所属確率の実際の値 (0,1) $q(Y=k\|X=x_n)$ ：入力x_n のｸﾗｽkへの所属確率の予測値 (0～1) N ：ﾃﾞｰﾀ個数 (n=1～N) K ：分類のｸﾗｽ数 (k=1～K)	ｸﾗｽ分類に一般的に用いられます。

【出典・参考】

損失関数⇒ https://to-kei.net/neural-network/nn_loss_function/

損失関数⇒ https://www.renom.jp/ja/notebooks/tutorial/basic_algorithm/lossfunction/notebook.html

尤度⇒「統計学」森棟,照井,中川,西埜,黒住 共著 有斐閣 2017年12月 改訂版第3刷

目的関数⇒http://research.nii.ac.jp/~uno/opt_1.htm

(7) ﾓﾃﾞﾙの構造

・ここでは「教師あり学習（分類）」のﾓﾃﾞﾙについて、「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」のうち以下の2つを用いた分類ﾓﾃﾞﾙを紹介します。

(1) 「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ（ｼﾞｭﾝﾃﾞﾝﾊﾟﾝｶﾞﾀﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Feed forward neural network、FFNN）」

(2) 「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ（ﾀﾀﾐｺﾐﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Convolutional neural network、CNN）」

・「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」は、基本的に以下の層から構成されます。

層名	説明
入力層	ﾆｭｰﾗﾙﾈｯﾄﾜｰｸへの情報を取り込む層
隠れ層	入力側からの信号に対して、重みづけ評価を行いながら、出力側へ信号伝達する層（中間層とも言います）
出力層	ﾆｭｰﾗﾙﾈｯﾄﾜｰｸが算出した結果を出力する層

(7.1) 順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ

(※ この節の一部は、「ｾﾐﾅｰ2回目、7回目」からの再掲を含みます。)

・「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (ｼﾞｭﾝﾃﾞﾝﾊﾟﾝｶﾞﾀﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Feed forward neural network、FFNN)」は、以下の特徴を持つﾆｭｰﾗﾙﾈｯﾄﾜｰｸです。

(1) 隣接する層間のすべてのﾕﾆｯﾄが完全に接続されている(完全結合)

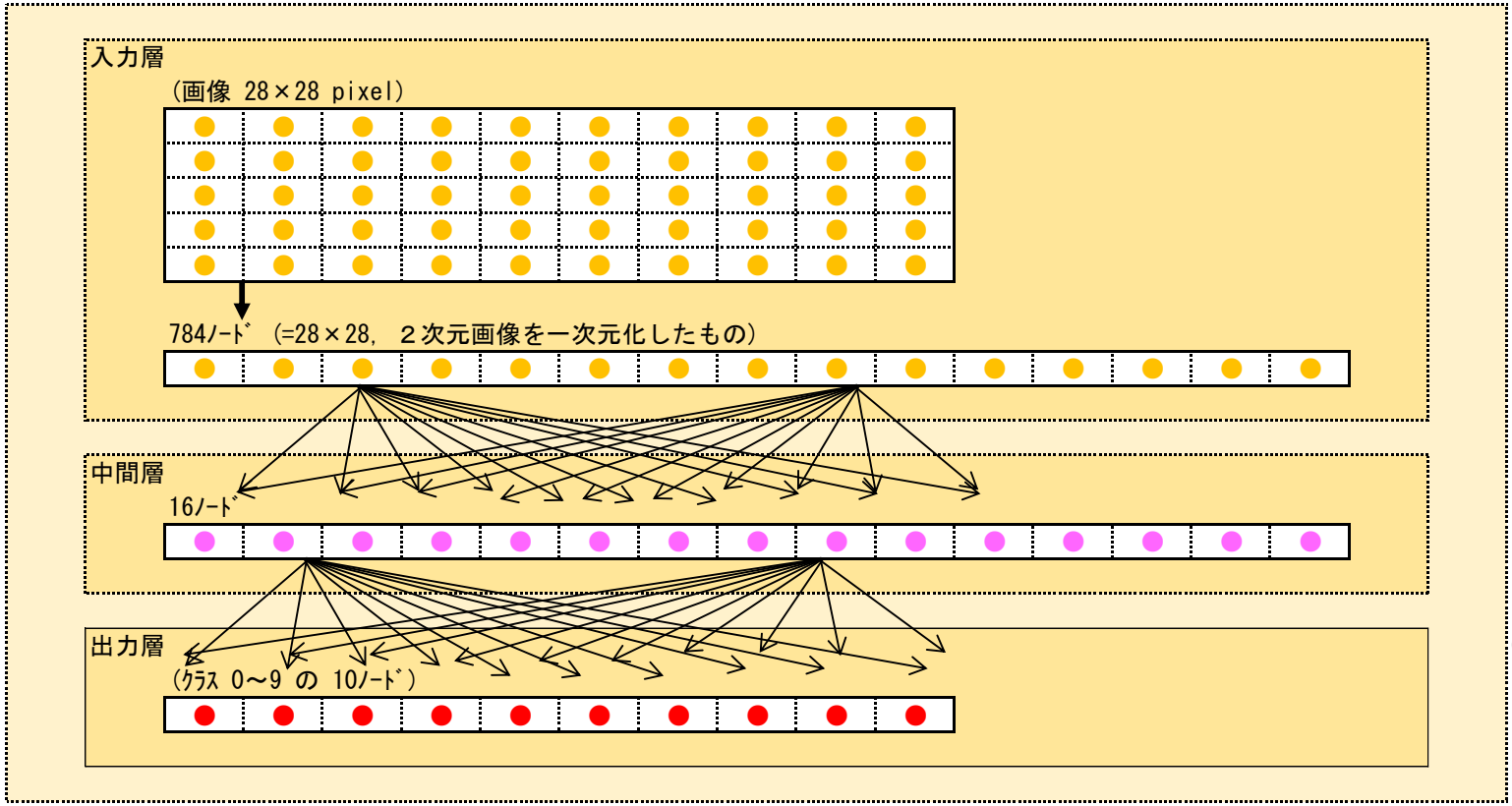
(2) 入力層から出力層に向けて一方通行で流れる

(3) 入力層と出力層の間に、１つ以上の隠れ層がある

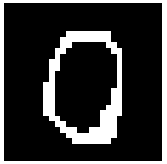
・「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」で、隠れ層の層数を多くしたものが
「深層順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (ｼﾝｿｳｼﾞｭﾝﾃﾞﾝﾊﾟﾝｶﾞﾀﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Deep Feed forward neural network、DFF)」です。

・このﾀｲﾌﾟのﾈｯﾄﾜｰｸは「誤差逆伝播法 (ｺﾞｻｷﾞｬｸﾃﾞﾝﾊﾟﾎｳ、Backpropagation)」で学習します。

・以下に前回のｾﾐﾅｰで使用したﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ１」を、FFNNの例として再掲します。



(入力画像)



(28×28 ﾓﾉｸﾛ)

(出力：ｸﾗｽへの所属確率)
0: 0.45, 1: 0.01, 2: 0.06
3: 0.05, 4: 0.02, 5: 0.18
6: 0.01, 7: 0.01, 8: 0.21
9: 0.01

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ１」とその学習方法は以下のようなものです：

(1) ﾓﾃﾞﾙは「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (FFNN)」であり、入力層から出力層に向けて一方通行で流れる(Sequential)。

(2) 入力ﾃﾞｰﾀは、(Y方向ｻｲｽﾞ, X方向ｻｲｽﾞ, 入力ﾁｬﾈﾙ数) = (28,28,1) のﾓﾉｸﾛ画像 (0～9の手書き数字1文字) であるが、
二次元画像を一次元化した784 (=28×28)個の各ﾃﾞｰﾀを、入力層の各ﾉｰﾄﾞとする。
各入力値は、0～255 の 256 諧調のところを、255 で割って 0～1 に正規化したものを用いる。
入力層の各ﾉｰﾄﾞは中間層の各ﾉｰﾄﾞとの間で、完全に接続させる(全結合)。

(3) 入力と出力の間に、ﾉｰﾄﾞ数が 16 から構成される１つの中間層 (隠れ層) を置く。
中間層の活性化関数として「ReLU (正規化線形関数, RectifiedLinear Unit)」を用いる。
中間層の各ﾉｰﾄﾞは出力層の各ﾉｰﾄﾞとの間で、完全に接続させる(全結合)。

(4) 出力層のﾉｰﾄﾞ数は、10 (入力画像の文字の種類(0～9)の数)である。
ﾃﾞｰﾀ型は「float32 (入力画像枚数,10)」で、入力画像の各ｸﾗｽへの所属確率[0～1]を表す。
出力層の活性化関数として「ｿﾌﾄﾏｯｸｽ関数 (softmax)」を用いる。

(5) 損失関数(誤差を計算する関数)として、「交差ｴﾝﾄﾛﾋﾟｰ誤差 (cross entropy)」を用いる。

(6) 学習方法(ﾓﾃﾞﾙの最適化方法)として、勾配降下法の一種である「Adam」を用いる。
学習時には「ｴﾎﾟｯｸ数」として10、「ﾊﾞｯﾁｻｲｽﾞ」として「1000」で学習を行う。

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ１」の定義と学習の実装部分は以下のようなものです：

（ ﾘｽﾄ07-(03)-1_手書き文字認識ﾓﾃﾞﾙ１ ）からの抜粋と要約

```
IMG_SIZE_Y = 28         # 画像サイズ(Y方向)
IMG_SIZE_X = 28         # 画像サイズ(X方向)
IMG_CLASS_NO = 10       # 画像の分類クラス数(0～9 の10クラス)

#=============================================================
# 「手書き文字認識ﾓﾃﾞﾙ１」を定義する
#=============================================================
np.random.seed(1)
predModel = Sequential()
predModel.add(Dense(16, input_dim=inTrain1.shape[1], activation=actvFunc))
predModel.add(Dense(IMG_CLASS_NO, activation='softmax'))
predModel.compile(loss='categorical_crossentropy',
    optimizer=Adam(), metrics=['accuracy'])
predModel.summary()

#=============================================================
# 「手書き文字認識ﾓﾃﾞﾙ１」で学習する
#=============================================================
epochNo = 10
batchSize = 1000

history = predModel.fit(inTrain1, outTrain1,
    epochs=epochNo, batch_size=batchSize,
    verbose=0, validation_data=(inTest1, outTest1))
score = predModel.evaluate(inTest1, outTest1, verbose=0)
```

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ１」はﾓﾃﾞﾙｸﾗｽの「summary()」ﾒｿｯﾄﾞによりそのｻﾏﾘを出力でき、以下のようになります：

（ ﾘｽﾄ07-(03)-1_手書き文字認識ﾓﾃﾞﾙ１ ）の実行結果の抜粋と要約

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 16)                12560
_________________________________________________________________
dense_4 (Dense)              (None, 10)                170
=================================================================
Total params: 12,730
Trainable params: 12,730
Non-trainable params: 0
_________________________________________________________________
```

【出典・参考】

⇒ https://qiita.com/ma-oshita/items/99b2cf313494adbb964d

⇒ https://ja.wikipedia.org/wiki/ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ#順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ

(7.2) 畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ

（※ この節の一部は、「ｾﾐﾅｰ2回目、7回目」からの再掲を含みます。）

・「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ（ﾀﾀﾐｺﾐﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Convolutional neural network、CNN)」は、階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸに、畳み込み構造を
入れ込んだﾓﾃﾞﾙです。「深層畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ（ｼﾝｿｳﾀﾀﾐｺﾐﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Deep Convolutional neural network、DCN)」は、
CNN の中間層（隠れ層）を深くしたものです。
これは、人の視覚をﾓﾃﾞﾙにした構成になっており、画像認識分野を中心に幅広く利用されているﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞ(深層学習)手法の一つです。

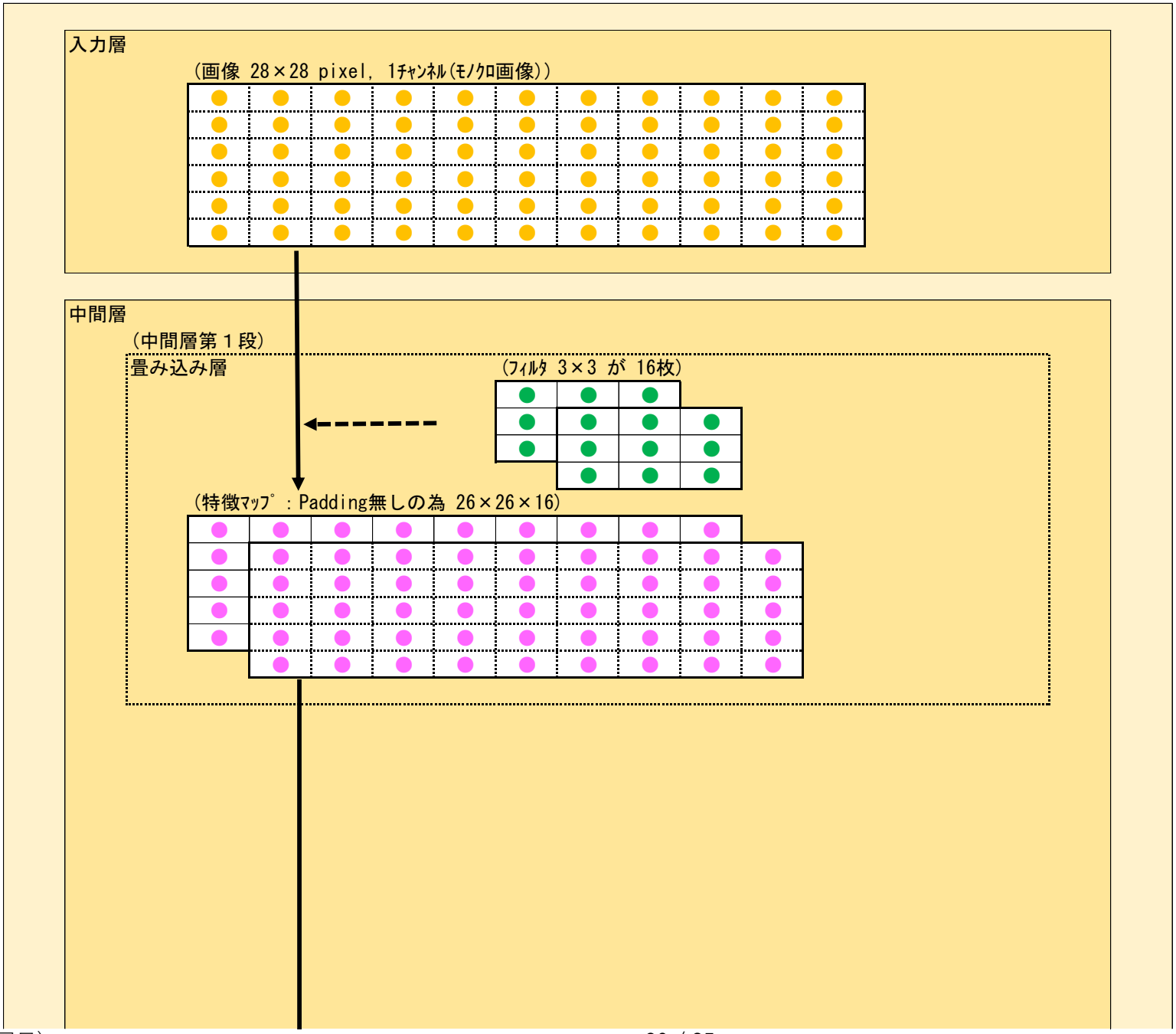
・「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」には、以下のような特徴があります：
　(1) 複雑なﾊﾟﾀｰﾝ認識問題にも、簡潔なﾈｯﾄﾜｰｸで対応できます。
　(2) 全体としてﾕﾆｯﾄ数が少なくて済むので、計算が容易です。

・このﾓﾃﾞﾙは、中間層(隠れ層)が、畳み込み層(ｺﾝﾎﾞﾘｭｰｼｮﾝ層)、正規化層、ﾌﾟｰﾘﾝｸﾞ層、全結合層で構成された複数の層からなります。
正規化層とﾌﾟｰﾘﾝｸﾞ層は必須ではありません。また、畳み込み層でﾌｨﾙﾀの「移動量（ｲﾄﾞｳﾘｮｳ、ｽﾄﾗｲﾄﾞ、stride)」を調整することにより
情報縮約を行って、ﾌﾟｰﾘﾝｸﾞ層を代行することもあります。

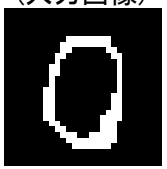
・各層は以下のようなものです。

層名	説明
畳み込み層　(ﾀﾀﾐｺﾐｿｳ、Convolution Layer)	入力側からの信号に対してﾌｨﾙﾀ処理を行い「特徴ﾏｯﾌﾟ（ﾄｸﾁｮｳﾏｯﾌﾟ, feature map)」を出力する層です。 ｺﾝﾎﾞﾘｭｰｼｮﾝ層ともいいます。使用するﾌｨﾙﾀは「ｶｰﾈﾙ（kernel)」「ｵﾍﾟﾚｰﾀ」「ﾏｽｸ」とも言います。
ﾌﾟｰﾘﾝｸﾞ層（ﾌﾟｰﾘﾝｸﾞｿｳ、Pooling Layer)	畳み込み層で作成された「特徴ﾏｯﾌﾟ」に対し情報縮約操作を行う層です。 ﾌﾟｰﾘﾝｸﾞには、以下のようなものがあります： ・最大値ﾌﾟｰﾘﾝｸﾞ（Max Pooling）　着目領域での最大値で縮約する ・平均値ﾌﾟｰﾘﾝｸﾞ（Average Pooling）着目領域での平均値で縮約する ・Lpﾌﾟｰﾘﾝｸﾞ（Lp Pooling）　着目領域での中央値を強調した値で縮約する
全結合層（ｾﾞﾝｹﾂｺﾞｳｿｳ、Fully Connected Layer)	隣接する層の各ﾕﾆｯﾄ同士が、全ての組合せで結合することを全結合と言います。 通常、出力層と前段の数層を全結合とし、これを全結合層としてﾓﾃﾞﾙ化します。
正規化層（ｾｲｷｶｿｳ、Normalization Layer)	各ｻﾝﾌﾟﾙの平均と分散をそれぞれ０と１になるように、入力側からの信号を変換する層です。 これで、入力ﾃﾞｰﾀ間のばらつきを補正します。
Flatten層（ﾌﾗｯﾄﾝｿｳ、Flatten Layer)	前段の層から出てきた多次元のﾃﾞｰﾀを一次元に変換する層です。 次段の層の各ﾕﾆｯﾄと全結合する準備などの為に入れます。

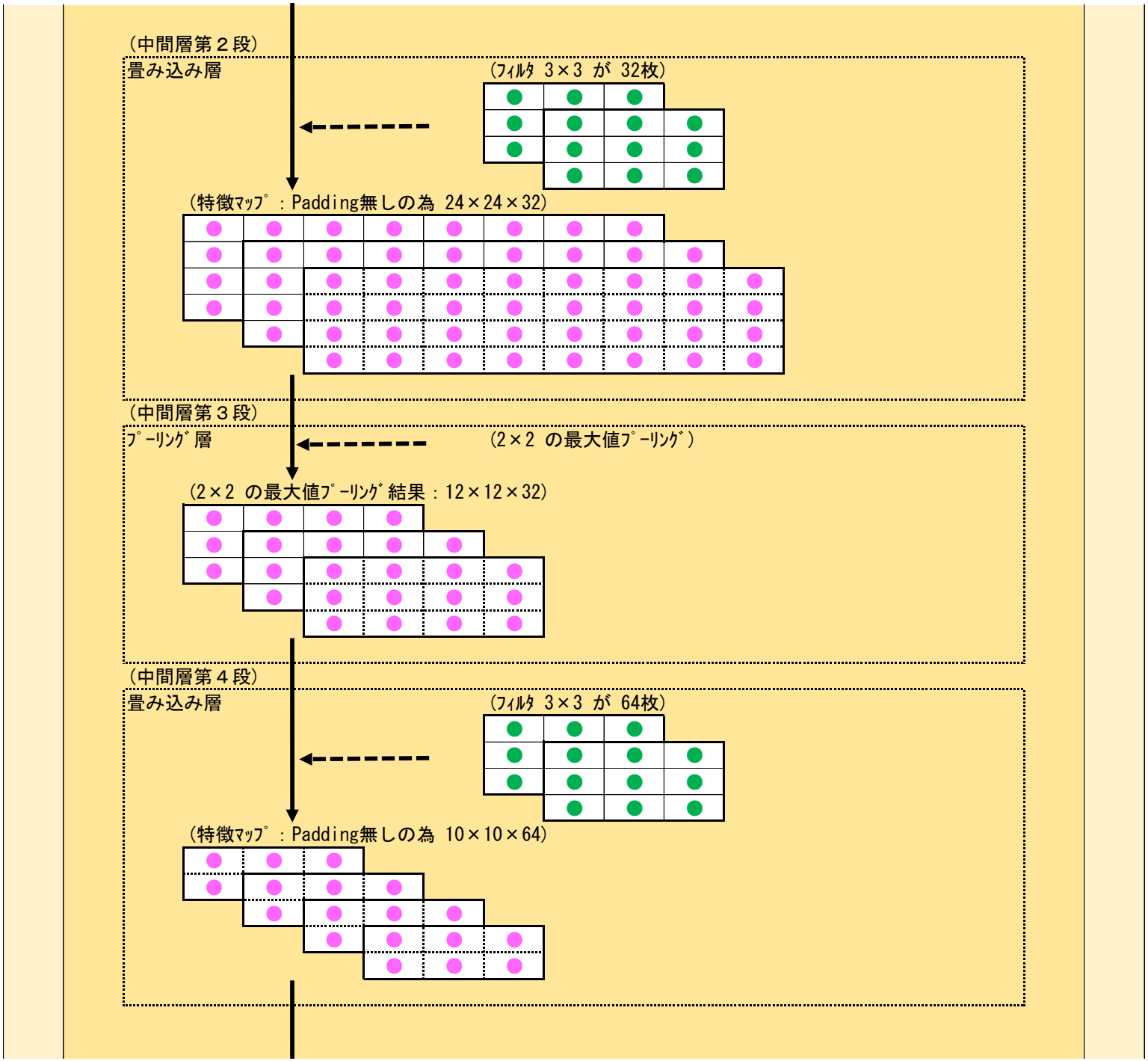
・以下に前回のｾﾐﾅｰで使用したﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ３」を、CNNの例として再掲します。

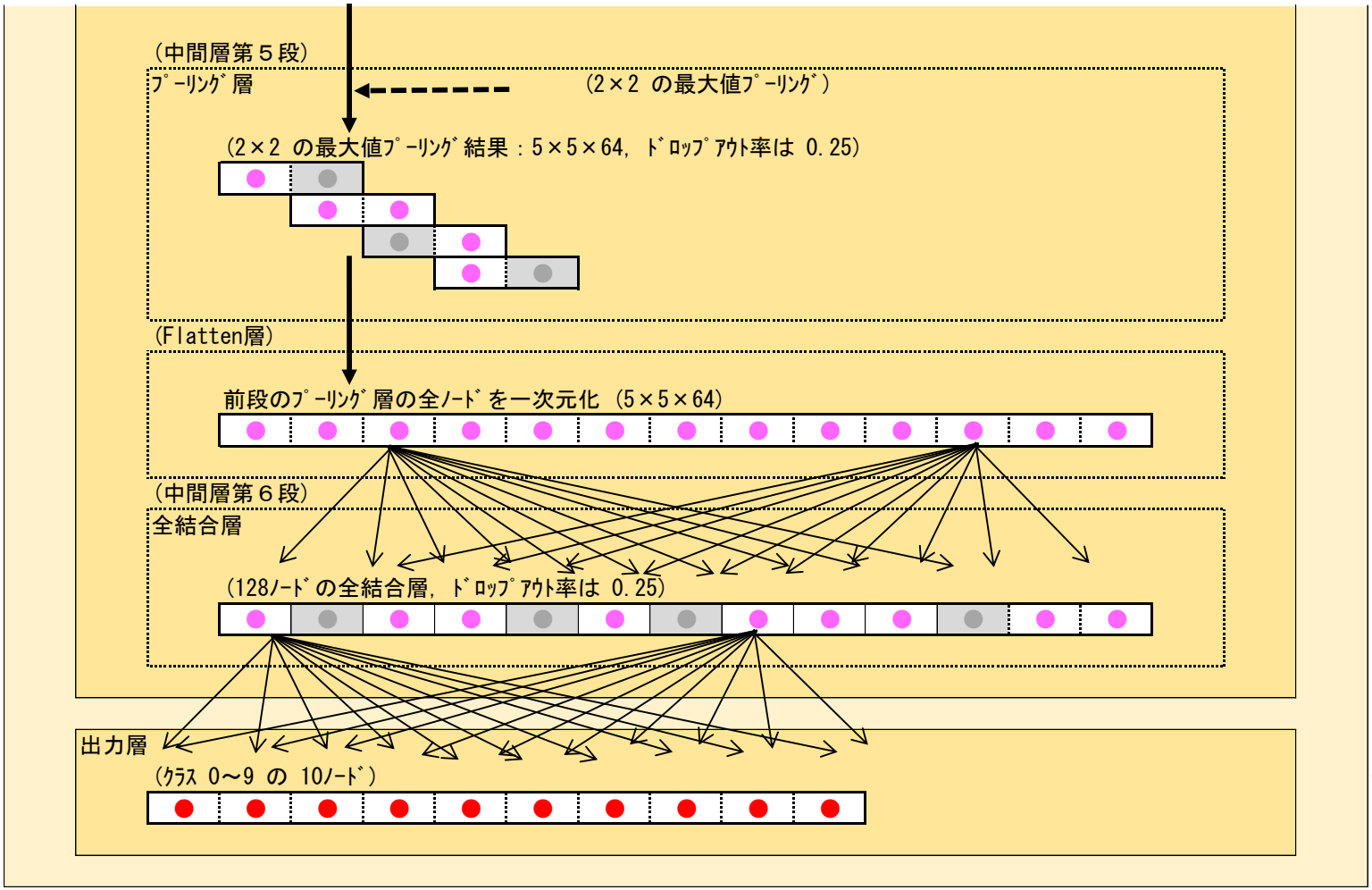


（入力画像）



（28×28 ﾓﾉｸﾛ）


(中間層第２段)
畳み込み層
(ﾌｨﾙﾀ 3×3 が 32枚)
(特徴ﾏｯﾌﾟ：Padding無しの為 24×24×32)
(中間層第３段)
ﾌﾟｰﾘﾝｸﾞ層
(2×2 の最大値ﾌﾟｰﾘﾝｸﾞ)
(2×2 の最大値ﾌﾟｰﾘﾝｸﾞ結果：12×12×32)
(中間層第４段)
畳み込み層
(ﾌｨﾙﾀ 3×3 が 64枚)
(特徴ﾏｯﾌﾟ：Padding無しの為 10×10×64)



(出力：ｸﾗｽへの所属確率)
0: 0.99, 1: 0.00, 2: 0.00
3: 0.00, 4: 0.00, 5: 0.00
6: 0.00, 7: 0.00, 8: 0.00
9: 0.01

：ﾄﾞﾛｯﾌﾟｱｳﾄ対象のﾉｰﾄﾞを示します

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ３」とその学習方法は以下のようなものです：

(1) ﾓﾃﾞﾙは「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」であり、入力層から出力層に向けて一方通行で流れる(Sequential)。

(2) 入力ﾃﾞｰﾀは、(Y方向ｻｲｽﾞ, X方向ｻｲｽﾞ, 入力ﾁｬﾈﾙ数) = (28,28,1) のﾓﾉｸﾛ画像 (0～9の手書き数字1文字) であり、
これをそのまま入力層の入力ﾃﾞｰﾀとする。
各入力値は、0～255 の 256 諧調のところを、255 で割って 0～1 に正規化したものを用いる。

(3) 入力と出力の間に、以下の６段で構成する中間層 (隠れ層) を置く。
(3.1) 中間層第１段は、畳み込み層で、ﾌｨﾙﾀ 3×3 を 16枚で、Padding無しで、畳み込む。
(3.2) 中間層第２段は、畳み込み層で、ﾌｨﾙﾀ 3×3 を 32枚で、Padding無しで、畳み込む。
(3.3) 中間層第３段は、2×2の最大値ﾌﾟｰﾘﾝｸﾞ層とする。
(3.4) 中間層第４段は、畳み込み層で、ﾌｨﾙﾀ 3×3 を 64枚で、Padding無しで、畳み込む。
(3.5) 中間層第５段は、2×2の最大値ﾌﾟｰﾘﾝｸﾞ層で、ﾄﾞﾛｯﾌﾟｱｳﾄ率を 0.25 とする。
この出力を、Flatten層を通して一次元化し、その各ﾉｰﾄﾞを、中間層第５段の各ﾉｰﾄﾞとの間で全結合させる。
(3.6) 中間層第６段は、全結合層で、ﾄﾞﾛｯﾌﾟｱｳﾄ率を 0.25 とする。
各ﾉｰﾄﾞを、出力層の各ﾉｰﾄﾞとの間で全結合させる。
(3.7) 中間層の活性化関数として「ReLU (正規化線形関数, RectifiedLinear Unit)」を用いる。

(4) 出力層のﾉｰﾄﾞ数は、10 (入力画像の文字の種類(0～9)の数)である。
ﾃﾞｰﾀ型は「float32 (入力画像枚数,10)」で、入力画像の各ｸﾗｽへの所属確率[0～1]を表す。
中間層の出力を「Flatten」層を通すことで、「28×28×8枚×入力ﾁｬﾈﾙ数」という
４次元のﾃﾞｰﾀを一次元化して、出力層との間で全結合させる。
出力層の活性化関数として「ｿﾌﾄﾏｯｸｽ関数 (softmax)」を用いる。

(5) 損失関数(誤差を計算する関数)として、「交差ｴﾝﾄﾛﾋﾟｰ誤差 (cross entropy)」を用いる。

(6) 学習方法(ﾓﾃﾞﾙの最適化方法)として、勾配降下法の一種である「Adam」を用いる。
学習時には「ｴﾎﾟｯｸ数」として10、「ﾊﾞｯﾁｻｲｽﾞ」として「1000」で学習を行う。

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ３」の定義と学習の実装部分は以下のようなものです：

（ ﾘｽﾄ07-(03)-3_手書き文字認識ﾓﾃﾞﾙ３ ）からの抜粋と要約

```
IMG_SIZE_Y = 28         # 画像サイズ(Y方向)
IMG_SIZE_X = 28         # 画像サイズ(X方向)
IMG_CLASS_NO = 10       # 画像の分類クラス数(0～9 の10クラス)

#=============================================================
# 「手書き文字認識ﾓﾃﾞﾙ３」を定義する
#=============================================================
np.random.seed(1)
predModel = Sequential()
predModel.add(Conv2D(16, (3, 3), input_shape=(IMG_SIZE_Y, IMG_SIZE_X, 1), activation='relu'))
predModel.add(Conv2D(32, (3, 3), activation='relu'))
predModel.add(MaxPooling2D(pool_size=(2, 2)))
predModel.add(Conv2D(64, (3, 3), activation='relu'))
predModel.add(MaxPooling2D(pool_size=(2, 2)))
predModel.add(Dropout(0.25))
predModel.add(Flatten())
predModel.add(Dense(128, activation='relu'))
predModel.add(Dropout(0.25))
predModel.add(Dense(IMG_CLASS_NO, activation='softmax'))
predModel.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
predModel.summary()

#=============================================================
# 「手書き文字認識ﾓﾃﾞﾙ３」で学習する
#=============================================================
epochNo = 10
batchSize = 1000
history = predModel.fit(inTrain3, outTrain3,
    epochs=epochNo, batch_size=batchSize,
    verbose=0, validation_data=(inTest3, outTest3))
score = predModel.evaluate(inTest3, outTest3, verbose=0)
```

・ﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ３」はﾓﾃﾞﾙｸﾗｽの「summary()」ﾒｿｯﾄﾞによりそのｻﾏﾘを出力でき、以下のようになります：

（ ﾘｽﾄ07-(03)-3_手書き文字認識ﾓﾃﾞﾙ３ ）の実行結果の抜粋と要約

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 26, 26, 16)        160
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               204928
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 229,514
Trainable params: 229,514
Non-trainable params: 0
_________________________________________________________________
```

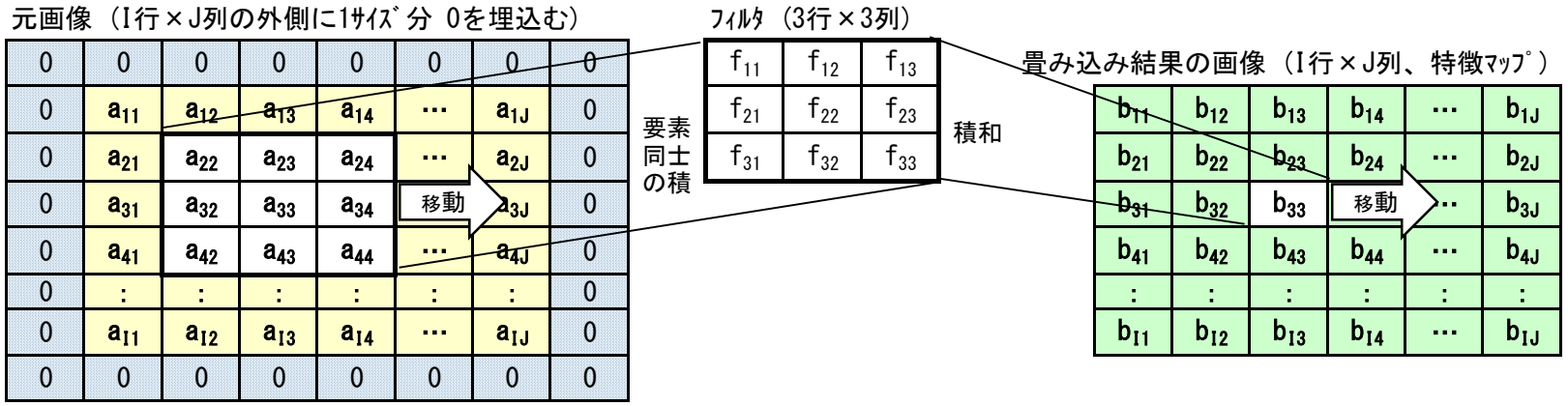
(7.2.1) 畳み込み層

・「畳み込み層（ﾀﾀﾐｺﾐｿｳ、Convolution Layer)」は、入力側からの信号に対して「畳み込み」によるﾌｨﾙﾀ処理を行い、「特徴ﾏｯﾌﾟ（ﾄｸﾁｮｳﾏｯﾌﾟ, feature map)」を作成する層です。

・「畳み込み」によるﾌｨﾙﾀ処理は、具体的には以下のように、元画像の領域にﾌｨﾙﾀを重ねて、要素同士の積和を畳み込み結果の領域の中心位置の値とする演算処理で、これを全領域に渡って移動しながら、畳み込み結果画像を得るものです。

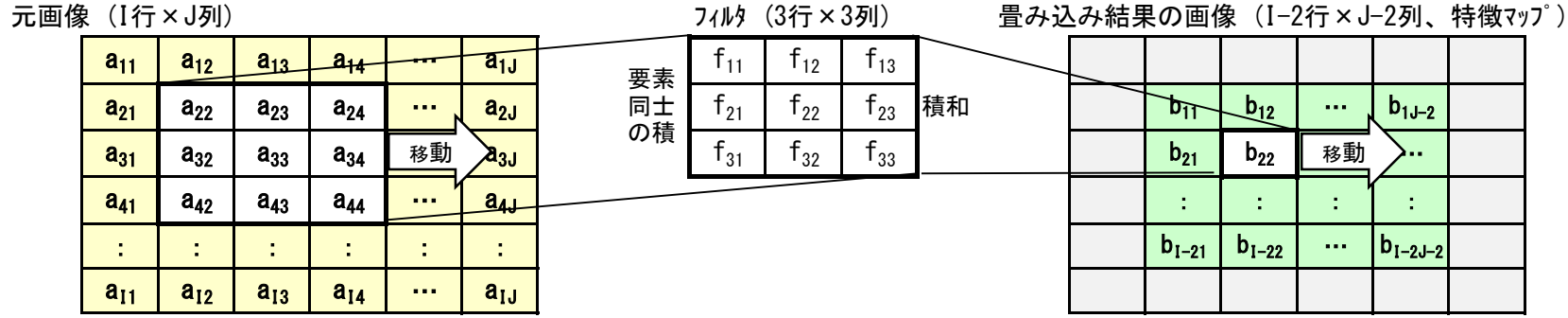
(1) padding 有りの場合　（元画像がI行×J列、ﾌｨﾙﾀが3行×3列、ｽﾄﾗｲﾄﾞのｻｲｽﾞは1）

$$b_{ij} = \sum_{m=1}^{3} \sum_{n=1}^{3} a_{i+m-2,\, j+n-2} \times f_{mn}$$　　i=1～I, j=1～J, 拡張部分($a_{0,*}$ / $a_{*,0}$ / $a_{I+1,*}$ / $a_{*,J+1}$)は0



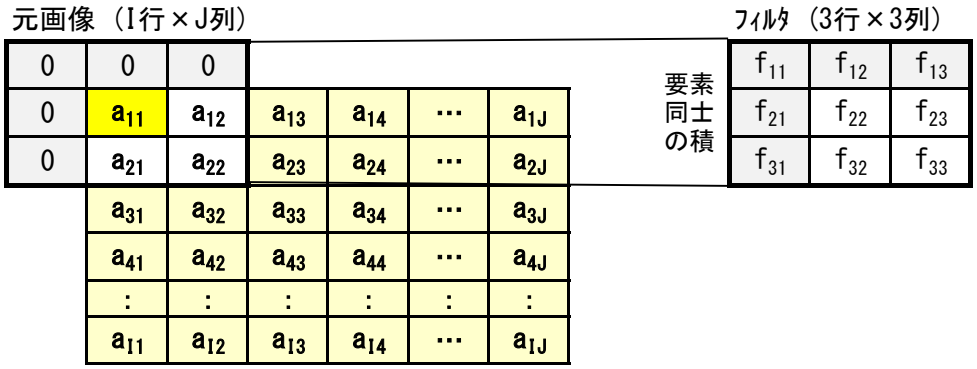
(2) padding 無しの場合　（元画像がI行×J列、ﾌｨﾙﾀが3行×3列、ｽﾄﾗｲﾄﾞのｻｲｽﾞは1）

$$b_{ij} = \sum_{m=1}^{3} \sum_{n=1}^{3} a_{i+m-1,\, j+n-1} \times f_{mn}$$　　i=1～I-2, j=1～J-2

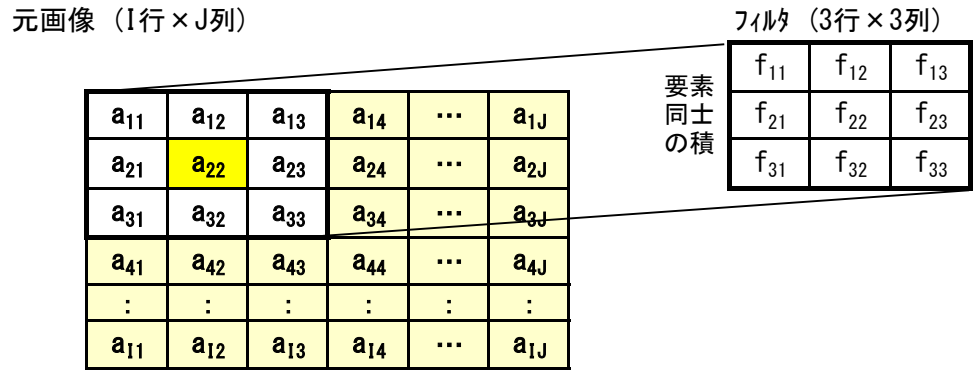


・「畳み込み」演算では、元画像の縁部分にﾌｨﾙﾀをかけると、ﾌｨﾙﾀの一部が元画像の領域からはみ出してしまいます。
この為、ﾌｨﾙﾀ演算結果として採用できるのは、元画像の領域からはみ出していない領域に限られ、
ﾌｨﾙﾀ演算結果のｻｲｽﾞが元画像よりも小さくなってしまいます。

・これを避けるために、元画像の縁から外側に、ﾌｨﾙﾀのはみ出し部分がないｻｲｽﾞ分だけ０などを埋め込みます。
この操作を「ﾊﾟﾃﾞｨﾝｸﾞ (padding)」と言い、ｻｲｽﾞを「ﾊﾟﾃﾞｨﾝｸﾞｻｲｽﾞ (padding size)」と言います。
「ﾊﾟﾃﾞｨﾝｸﾞ」によりﾌｨﾙﾀｰ適用後画像（特徴ﾏｯﾌﾟ）のｻｲｽﾞが小さくなるのを防ぐことができます。



… a_{11} を中心とした領域に
ﾌｨﾙﾀ (3行×3列) をかける際に、
ﾌｨﾙﾀは元画像の領域から一部がはみだしてしまいます。
1ｻｲｽﾞ分(ﾌｨﾙﾀの半分のｻｲｽﾞ)外側に0を埋めて対応します。



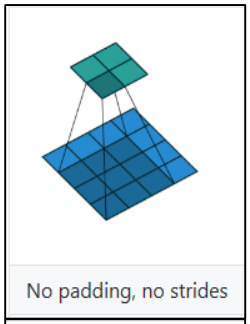
… a_{22} を中心とした領域に
ﾌｨﾙﾀ (3行×3列) をかける際に、
ﾌｨﾙﾀは元画像の領域内にあるので、
外側に0を埋める必要はありません。

・「畳み込み」演算時に、全領域に渡って移動しながら元画像の領域にﾌｨﾙﾀを重ねて演算を行いますが、
この時のﾌｨﾙﾀの「移動量（ｲﾄﾞｳﾘｮｳ、ｽﾄﾗｲﾄﾞ、stride)」を指定できます。
上記の説明では「ｽﾄﾗｲﾄﾞ」のｻｲｽﾞを「1」にしています。

・「移動量」を調整することにより結果的に情報縮約を行うことになり、これでﾌﾟｰﾘﾝｸﾞ層を代行することもあります。
例えば、「ｽﾄﾗｲﾄﾞ」として「2」を指定すると、ﾌｨﾙﾀｰ適用後画像（特徴ﾏｯﾌﾟ）が、元画像の2分の1に縮んだものとなります。

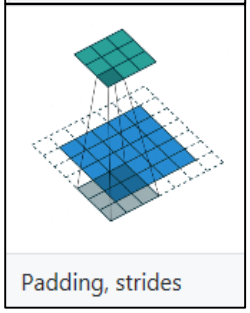
・以下に畳み込み時のﾌｨﾙﾀ・ｽﾄﾗｲﾄﾞ・ﾊﾟﾃﾞｨﾝｸﾞのﾊﾟﾀｰﾝを幾つか、引用で示します。
（「https://github.com/vdumoulin/conv_arithmetic」からの引用。参照するとｱﾆﾒｰｼｮﾝになっていて解り易いです）

例1



・(例1)は、4×4の画像(下図)に対して、ﾌｨﾙﾀのｻｲｽﾞを3×3、
ｽﾄﾗｲﾄﾞのｻｲｽﾞを1×1、ﾊﾟﾃﾞｨﾝｸﾞ無しで畳み込みを行った結果、
2×2の特徴ﾏｯﾌﾟが作成されたものです(上図)。

例2

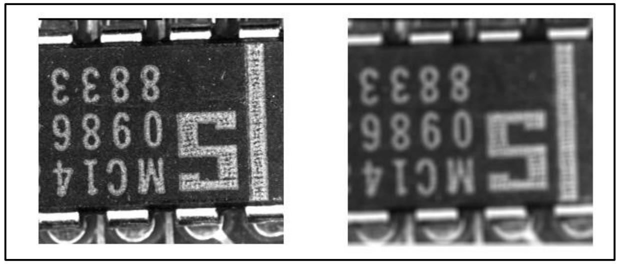


・(例2)は、5×5の画像(下図)に対して、ﾌｨﾙﾀのｻｲｽﾞを3×3、
ｽﾄﾗｲﾄﾞのｻｲｽﾞを2×2、ﾊﾟﾃﾞｨﾝｸﾞあり(片側ｻｲｽﾞ1)で畳み込みを行った結果、
3×3の特徴ﾏｯﾌﾟが作成されたものです(上図)。

・画像処理では様々なﾌｨﾙﾀがあり、これによって画像解析の為の前処理を行っています。
CNN では、学習を通してこのﾌｨﾙﾀを自動作成します。

・「畳み込み」は、画像編集ｿﾌﾄでは「効果」で「ぼかし」「ｴｯｼﾞ検出」などの機能として組み込まれているものが多く見受けられます。
これも画像処理ﾌｨﾙﾀであり、以下に幾つか紹介します。「https://imagingsolution.net/imaging/filter-algorithm/」からの引用です。

移動平均ﾌｨﾙﾀ （※1）



(3 × 3)

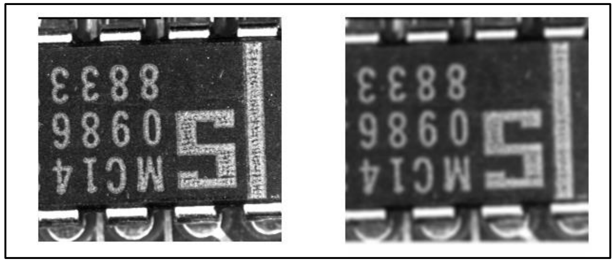
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

(5 × 5)

$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$
$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$	$\frac{1}{25}$

・移動平均ﾌｨﾙﾀ
（平均化ﾌｨﾙﾀ、単に平滑化ﾌｨﾙﾀともいう）では、
注目画素のその周辺の輝度値を用いて、
輝度値の平均を求め、
処理後画像の輝度値とするﾌｨﾙﾀです。

ｶﾞｳｼｱﾝﾌｨﾙﾀ （※1）



(3 × 3)

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

(5 × 5)

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

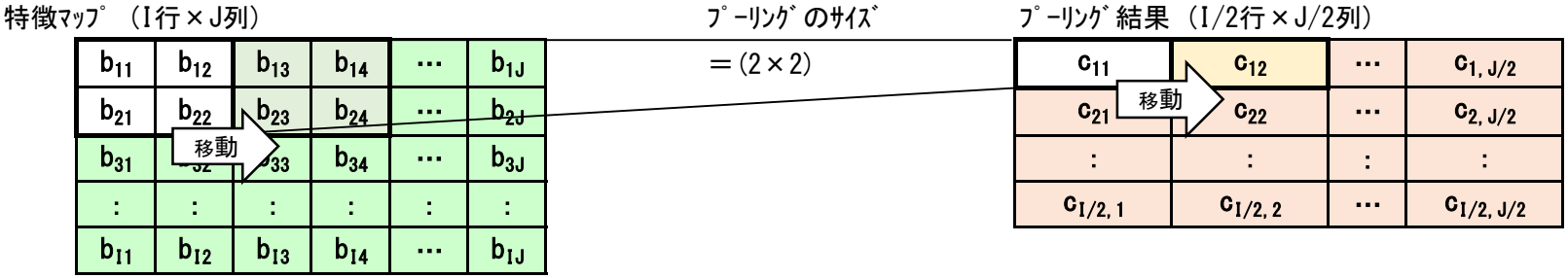
・ｶﾞｳｼｱﾝﾌｨﾙﾀ
は、注目画素に近いほど、
平均値を計算するときの重みを大きくし、
遠くなるほど重みを小さくなるようにｶﾞｳｽ分布の
関数を用いて割合を計算しているﾌｨﾙﾀです。
人間の目の焦点の当て方に近いと言われています。

（※1） 左が原画で右がﾌｨﾙﾀ処理後の画像

(7.2.2) ﾌﾟｰﾘﾝｸﾞ層

・「ﾌﾟｰﾘﾝｸﾞ層（ﾌﾟｰﾘﾝｸﾞｿｳ、Pooling Layer)」は、畳み込み層で作成された「特徴ﾏｯﾌﾟ」に対し情報縮約操作を行う層です。
「ﾌﾟｰﾘﾝｸﾞ」には、「最大値ﾌﾟｰﾘﾝｸﾞ」「平均値ﾌﾟｰﾘﾝｸﾞ」「Lpﾌﾟｰﾘﾝｸﾞ」といったものがあります。
何れも元画像の全領域を小領域（2×2、3×3などｻｲｽﾞを任意に指定）に分けて、各小領域での特徴量（最大値、平均値など）で縮約します。
縮約した結果のｻｲｽﾞは、元画像のｻｲｽﾞを小領域のｻｲｽﾞで割ったものになります。
これにより、特徴として重要な情報を残しながら元の画像を縮小し、画像の位置ずれに対する頑強さも持つことができます。

・以下にﾌﾟｰﾘﾝｸﾞのｻｲｽﾞが2×2の場合について示します。



・以下に主なﾌﾟｰﾘﾝｸﾞを一覧にして示します。

No.	ﾌﾟｰﾘﾝｸﾞ名	英語	式 （ﾌﾟｰﾘﾝｸﾞのｻｲｽﾞ＝2×2 の場合）	説明
1	最大値ﾌﾟｰﾘﾝｸﾞ（ｻｲﾀﾞｲﾁﾌﾟｰﾘﾝｸﾞ）	Max Pooling	$c_{ij} = \max\{ b_{m,n} \}$ $m=2(i-1)+1 \sim 2(i-1)+2,\ n=2(j-1)+1 \sim 2(j-1)+2$ $i=1 \sim I/2,\ j=1 \sim J/2$	元画像の全領域を、各小領域での最大値で縮約するものです。
2	平均値ﾌﾟｰﾘﾝｸﾞ（ﾍｲｷﾝﾁﾌﾟｰﾘﾝｸﾞ）	Average Pooling	$c_{ij} = 1/4 \times \sum_{m=2(i-1)+1}^{2(i-1)+2} \sum_{n=2(j-1)+1}^{2(j-1)+2} b_{m,n}$ $i=1 \sim I/2,\ j=1 \sim J/2$	元画像の全領域を、各小領域での平均値で縮約するものです。
3	Lpﾌﾟｰﾘﾝｸﾞ（ｴﾙﾋﾟｰﾌﾟｰﾘﾝｸﾞ）	Lp Pooling	$c_{ij} = \{1/4 \times \sum_{m=2(i-1)+1}^{2(i-1)+2} \sum_{n=2(j-1)+1}^{2(j-1)+2} (b_{m,n})^P \}^{1/P}$ $i=1 \sim I/2,\ j=1 \sim J/2$ P はべき指数	元画像の全領域を、各小領域での中央値を強調した値で縮約するものです。

【出典・参考】

畳み込み⇒ https://github.com/vdumoulin/conv_arithmetic

CNN⇒ https://deepage.net/deep_learning/2016/11/07/convolutional_neural_network.html

DCNN⇒ http://www.nlab.ci.i.u-tokyo.ac.jp/pdf/CNN_survey.pdf

DCNN⇒ https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

空間ﾌｨﾙﾀ⇒ http://www.clg.niigata-u.ac.jp/~lee/jyugyou/img_processing/medical_image_processing_03_press.pdf

ﾌｨﾙﾀ⇒ https://codezine.jp/article/detail/129

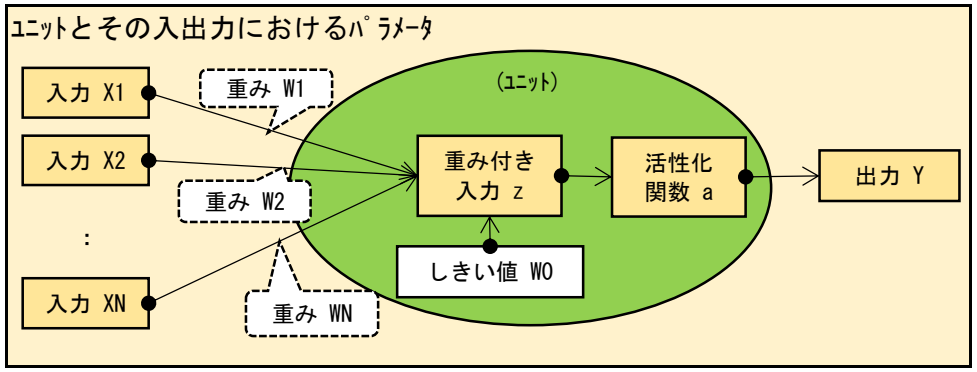
ﾌﾟｰﾘﾝｸﾞ ⇒ http://pynote.hatenablog.com/entry/dl-pooling

ﾌﾟｰﾘﾝｸﾞ ⇒「深層学習 Deep Learning」人工知能学会監修（近代科学社 2015年11月）

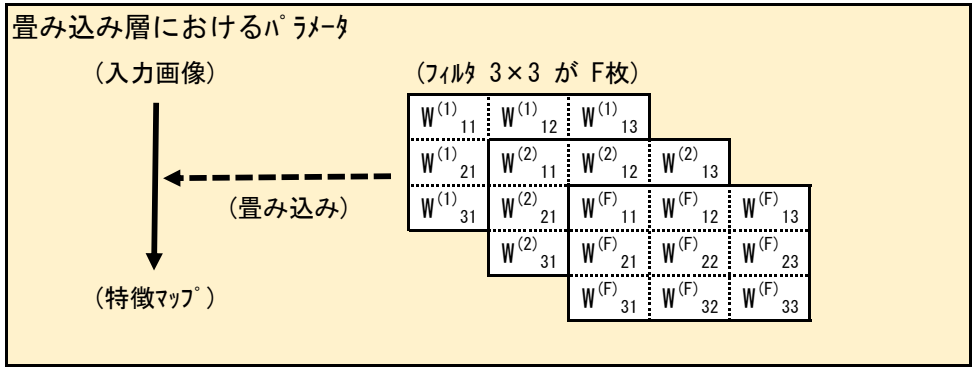
(8) ﾓﾃﾞﾙの学習

・教師あり学習では、説明変数と目的変数の関係を説明するﾓﾃﾞﾙを作成し、
「ﾓﾃﾞﾙの出力結果(予測)」と「既知の結果ﾃﾞｰﾀ」との差が最小化する(ｺｽﾄ関数(損失関数)の値が最小になる)ように、
ﾓﾃﾞﾙの「ﾊﾟﾗﾒｰﾀ」を最適化します。これが「学習」です。

・ﾓﾃﾞﾙの「ﾊﾟﾗﾒｰﾀ」は、各ﾕﾆｯﾄの重みW_i／しきい値W_0／ﾌｨﾙﾀの各値$W^{(f)}_{ij}$などの調整可能な値です。



・左図で、ﾕﾆｯﾄとその入出力におけるﾊﾟﾗﾒｰﾀは、
ﾕﾆｯﾄの重み W_i、しきい値 W_0 です。
(i＝1～N：入力の数分)



・左図で、畳み込み層におけるﾊﾟﾗﾒｰﾀは、
ﾌｨﾙﾀの各値 $W^{(f)}_{ij}$ です。
(f＝1～F：ﾌｨﾙﾀ枚数分、i＝1～3, j＝1～3：ﾌｨﾙﾀのｻｲｽﾞ分)

・では、どのようにﾓﾃﾞﾙの「ﾊﾟﾗﾒｰﾀ」を最適化するのでしょうか？
以下に、この最適化の方法を紹介します。

(8.1) 勾配降下法と学習率

(※ この節の一部は、「ｾﾐﾅｰ5回目」からの再掲を含みます。)

・最適化対象のﾊﾟﾗﾒｰﾀをθで表します。ﾊﾟﾗﾒｰﾀθの関数であるｺｽﾄ関数Ｃ(θ)が与えられた時、
これを最小にするθ* を求める最適化を、「勾配降下法 (ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method)」で行います。

・「勾配降下法」は、
現ｽﾃｯﾌﾟ(t)のﾊﾟﾗﾒｰﾀ $\theta^{(t)}$ における勾配と学習率 η_t から、
次ｽﾃｯﾌﾟ(t+1)のﾊﾟﾗﾒｰﾀ $\theta^{(t+1)}$ を次式により更新して、
ｺｽﾄ関数Ｃ(θ)が最小になる θ* を逐次近似により求める方法です。

勾配降下法によるﾊﾟﾗﾒｰﾀの最適化

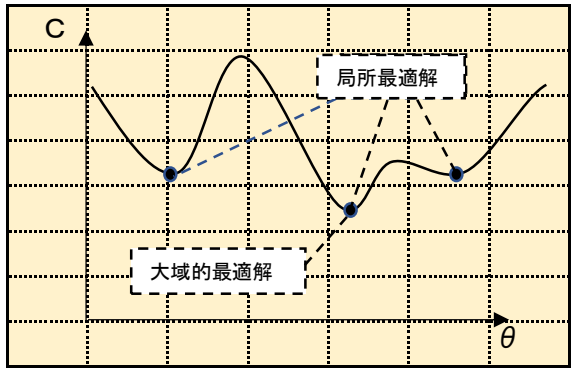
$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \times \partial C(\theta) / \partial \theta \,|_{\theta=\theta(t)} \quad \cdots(式8.1\text{-}1)$$

- Ｃ(θ)：ｺｽﾄ関数 (ﾊﾟﾗﾒｰﾀθの関数)
- $\theta^{(t)}$：ｽﾃｯﾌﾟ t におけるﾊﾟﾗﾒｰﾀθの値
- η_t：ｽﾃｯﾌﾟ t における学習率ηの値 (>0)
- $\partial C(\theta)/\partial\theta\,|_{\theta=\theta(t)}$：θ(t)におけるｺｽﾄ関数の勾配

(「勾配降下法」の詳細な説明は「ｾﾐﾅｰ5回目」を参照してください。)

・勾配降下法によるﾊﾟﾗﾒｰﾀの最適化の(式8.1-1)では、「学習率 (ｶﾞｸｼｭｳﾘﾂ、learning rate)」と呼ばれるﾊﾟﾗﾒｰﾀ η(ｲｰﾀ) があります。
これは、勾配降下法で、現ｽﾃｯﾌﾟ(t)のﾊﾟﾗﾒｰﾀθ(t)におけるｺｽﾄ関数Ｃ(θ) の勾配から、次ｽﾃｯﾌﾟ(t+1)のﾊﾟﾗﾒｰﾀ θ(t+1)を計算する時に、
勾配をどの程度の割合で変化分として反映させるか？という割合を示しています。
「学習率」は「学習係数」とも言います。

・「学習率」を大きくすると、最適解を通り過ぎて見落としてしまう可能性が高くなり、
「学習率」を小さくすると、最適解に辿り着くのに時間がかかったり、
最小値でない極小値にはまってしまうと抜け出られない場合があるという欠点があります。
(局所的に最小値を与える解を「局所最適解 (ｷｮｸｼｮｻｲﾃｷｶｲ、local optimal solution)」と言います
が、極小値が最小値とは限りません。)
これを「局所解問題 (ｷｮｸｼｮｶｲﾓﾝﾀﾞｲ、local minumum problem)」と言います。
尚、全領域にわたり最小値となる解を「大域的最適解 (ﾀｲｲｷﾃｷｻｲﾃｷｶｲ、global optimal soution)」
と言います。



・以降に述べる様々な改良版の「勾配降下法」と区別する場合、上記の「勾配降下法」を
「ﾊﾞｯﾁ勾配降下法 (ﾊﾞｯﾁｺｳﾊﾞｲｺｳｶﾎｳ, Batch gradient descent method)」と呼ぶこともあります。

・「ﾊﾞｯﾁ勾配降下法」は、学習データが多いと計算コストがとても大きくなってしまいます。
また、学習データが増えるたびに全ての学習データで再学習が必要となってしまうという欠点もあります。

(勾配降下法による最小値探索の例)

・以下の図は多変数$(x_1,\ x_2)$ の関数 y と、その勾配降下法による最小値探索の例です。

$$y(x_1, x_2) \quad = \quad x_1^{\,2} + x_2^{\,2} + 3x_2 + 4$$ ・・・(birdview)、(contour)の図

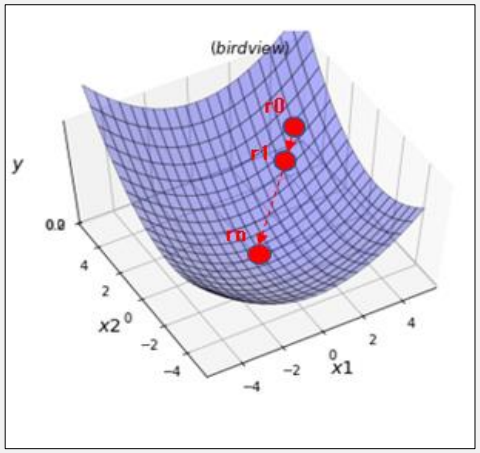
この関数が構成する面の勾配ベクトル∇y は、次式で与えられます。

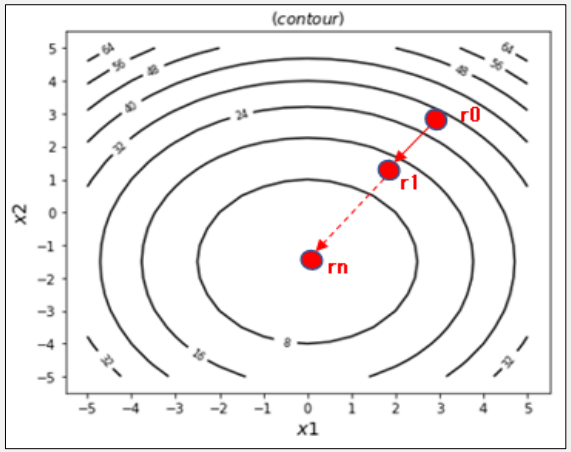
$$\nabla y \quad = \quad (\partial y/\partial x_1,\ \partial y/\partial x_2) \quad = \quad (2x_1,\ 2x_2+3)$$ ・・・(gradient)の図

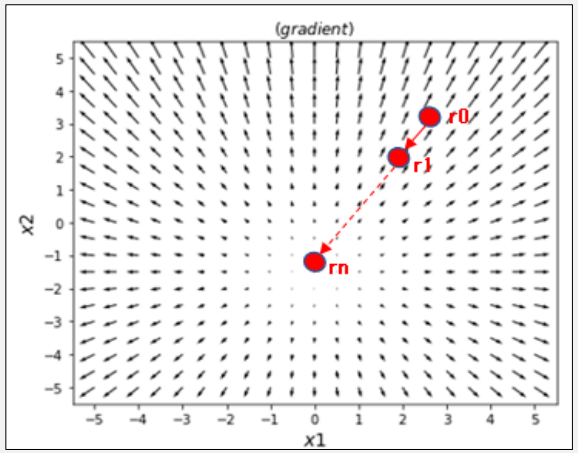
下図では、丸点とそれをつなぐ矢印は、初期位置 $r_0=(x_{10},\ x_{20})$ から始まって、

$$r_{i+1} \quad = \quad r_i - \eta\,(\partial y/\partial x_1,\ \partial y/\partial x_2)$$ $\eta\ (>0)$：学習率 ・・・各図

という移動を繰り返して、関数値が最小の点 $r_n=(x_{1n},\ x_{2n})$に落ち着く様子を示しています。







(8.2) 勾配降下法の改良版

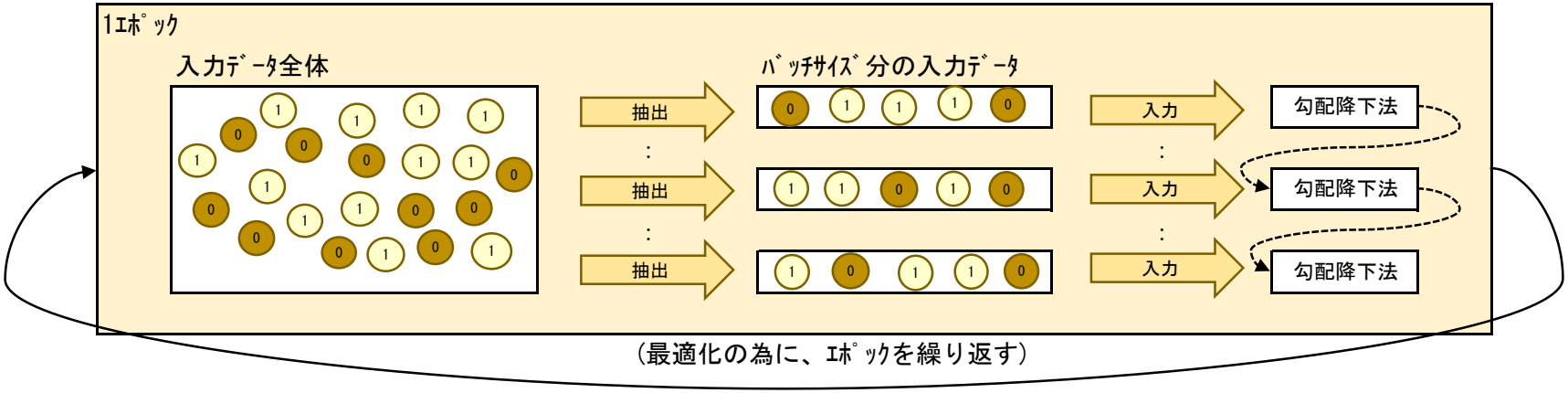
・「ﾊﾞｯﾁ勾配降下法」の欠点を改善するために、様々な改良版が考案されていますので幾つか紹介します。

(8.2.1) 確率的勾配降下法

・入力ﾃﾞｰﾀ全てに対する誤差関数の勾配を、更新の１ｽﾃｯﾌﾟごとに計算する場合、ﾃﾞｰﾀが大きいとその計算に時間がかかってしまいます。
そのような場合には、ﾃﾞｰﾀの一部で誤差関数の勾配を計算する、
「確率的勾配降下法（ｶｸﾘﾂﾃｷｺｳﾊﾞｲｺｳｶﾎｳ、Stochastic gradient descent method, SGD)」という方法が使われます。
「確率的勾配降下法」は「確率的勾配法」とも言います。

・「確率的勾配法」では入力ﾃﾞｰﾀ全体の一部で計算します。計算１回分に使用するﾃﾞｰﾀの個数を「ﾊﾞｯﾁｻｲｽﾞ（batch size)」と言います。
「ﾊﾞｯﾁｻｲｽﾞ」分の入力ﾃﾞｰﾀを使用した計算で、ﾊﾟﾗﾒｰﾀが１回更新されることになります。
入力ﾃﾞｰﾀ全体を用いて全体の誤差が最小になる方向へ進む「勾配降下法」とは異なり、
「ﾊﾞｯﾁｻｲｽﾞ」分の入力ﾃﾞｰﾀを使用した「確率的勾配法」は、訓練ﾃﾞｰﾀを少なくする為、勾配の推定が不安定となり、
ふらつきながら徐々に誤差の少ない方向へ進むことになります。
逆にこの不安定さのお陰で、局所解を抜け出せることもありうる(局所解問題の解消)、という長所があります。

・入力ﾃﾞｰﾀ全てを使いﾊﾟﾗﾒｰﾀ更新を行うとき、この更新の単位を「ｴﾎﾟｯｸ（epochs)」で表します。
「確率的勾配法」では、１「ｴﾎﾟｯｸ」のﾊﾟﾗﾒｰﾀ更新を行うのに「ﾊﾞｯﾁｻｲｽﾞ」分の入力ﾃﾞｰﾀを使用した更新を何度か繰り返します。
そしてﾊﾟﾗﾒｰﾀが最適化されるまで「ｴﾎﾟｯｸ」を何度か繰り返します(ﾓﾃﾞﾙによっては収束しない場合もあります)。



（例：MNIST ﾃﾞｰﾀｾｯﾄ・・・ｾﾐﾅｰ7回目を参照）

・訓練ﾃﾞｰﾀの件数は 60000 であり、ﾊﾞｯﾁｻｲｽﾞを 1000 とすると、
訓練ﾃﾞｰﾀを全て使うのに、 60000/1000＝60 回のﾊﾟﾗﾒｰﾀ更新が行われます。

・これが 1ｴﾎﾟｯｸでのﾊﾟﾗﾒｰﾀ更新です。
このｴﾎﾟｯｸを何度か繰り返して、ﾊﾟﾗﾒｰﾀの最適値を計算します。

(8.2.2) AdaGrad

・「AdaGrad（ｴｲﾀﾞｸﾞﾗｯﾄﾞ）」は、2011年にJohn Duchiらが提唱した手法であり、学習率を自動で調整するｱﾙｺﾞﾘｽﾞﾑです。
ｽﾃｯﾌﾟ(t)が進むにつれて、学習率η_tが徐々に小さくなるように調整します。

AdaGradで、ｽﾃｯﾌﾟ(t)における学習率η_t

$$\eta_t = \eta_0 / \left\{ \sqrt{\sum_{j=1}^{t} \left(\partial C(\theta) / \partial \theta \mid_{\theta=\theta(j)} \right)^2} \right\}$$

…(式8.2-2)

- $C(\theta)$：ｺｽﾄ関数（ﾊﾟﾗﾒｰﾀθの関数）
- θ：最適化対象のﾊﾟﾗﾒｰﾀ
- η_0：初期学習率
- $\partial C(\theta) / \partial \theta \mid_{\theta=\theta(j)}$：$\theta$(j)におけるｺｽﾄ関数の勾配
- j：ｽﾃｯﾌﾟ参照用添え字

(8.2.3) Adam

・「Adam（ｱﾀﾞﾑ、Adaptive moment estimation)」は、2015年にDiederik P. Kingma 氏が提唱した手法で、
AdaGrad や RMSProp, AdaDelta を改良したものです。
ｱﾙｺﾞﾘｽﾞﾑは『「深層学習 Deep Learning」近代科学社 人工知能学会監修 2015年11月』などに解説がありますので、参照して下さい。

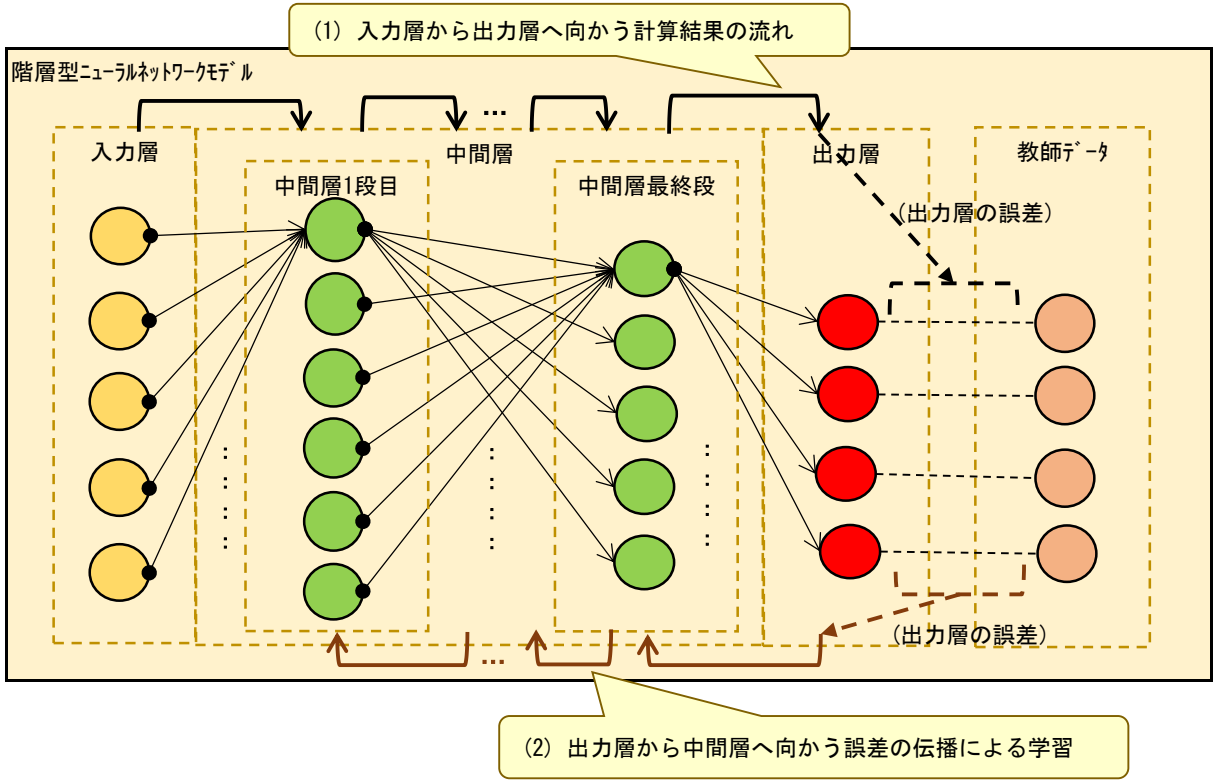
・「Adam」は、ｾﾐﾅｰ7回目の実装例でも使用しています。

(8.3) 誤差逆伝搬法

・「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」の学習方法として「誤差逆伝播法（ｺﾞｻｷﾞｬｸﾃﾞﾝﾊﾟﾎｳ、Backpropagation method)」が有名です。
手法そのものは以前からありましたが、1986年に「backwards propagation of errors（後方への誤差伝播)」の略から
ﾃﾞﾋﾞｯﾄﾞ･ﾗﾒﾙﾊｰﾄらによって命名されたものです。

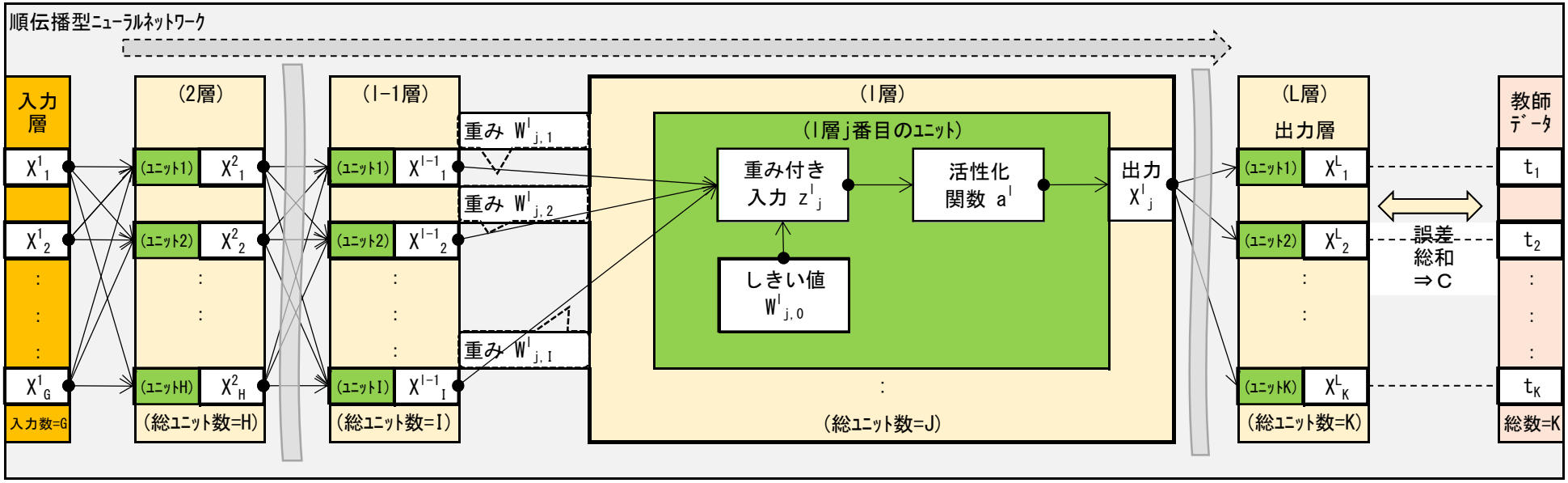
・「誤差逆伝播法」では、出力層での誤差(階層型ﾈｯﾄﾜｰｸの出力で生じる教師ﾃﾞｰﾀとの差)を使って、
ﾓﾃﾞﾙの出力の計算方向が中間層⇒出力層の方向なのとは逆向きに、
出力層⇒中間層の方向へと、勾配降下法を用いてﾊﾟﾗﾒｰﾀを更新してゆきます(下図)。

・勾配降下法では、誤差関数の偏微分を計算しますが、
これを偏微分の「連鎖律」を用いて解いてゆきます。
「連鎖律」の説明は「ｾﾐﾅｰ5回目」を御参照下さい。



(8.4) 「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」の誤差逆伝搬法

ここでは「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Feed forward neural networks、 FF, FFNN)」(下図)について、誤差逆伝搬法をより詳しく見てゆきます。



・これより以降は、以下のような記法を用いることにします。添え字が多いですがご了承下さい。

- L : 層の総数（入力層を第1層、出力層を第L層とします）
- l : 層番号（l=2～L、入力層は層の厚さとしては数えませんが、説明上第1層とします）
- I : l-1層の総ﾕﾆｯﾄ数
- J : l層の総ﾕﾆｯﾄ数
- i : l-1層でのﾕﾆｯﾄ番号（i=1～I）
- j : l層でのﾕﾆｯﾄ番号（j=1～J）
- X^{l-1}_i : l-1層i番目のﾕﾆｯﾄの出力値（=l層への入力信号）
- $W^l_{j,i}$: l層j番目のﾕﾆｯﾄについて、l-1層i番目のﾕﾆｯﾄからの入力信号の重み
- $W^l_{j,0}$: l層j番目のﾕﾆｯﾄについて、l-1層からの入力信号に付加するﾊﾞｲｱｽ
- z^l_j : l層j番目のﾕﾆｯﾄの重み付き入力（ $z^l_j = \sum_{i=0}^{I} W^l_{j,i} * X^{l-1}_i$ （但し $X^{l-1}_0=1$：ﾊﾞｲｱｽ項） ）
- a^l : l層の活性化関数 (activation function)
- X^l_j : l層j番目のﾕﾆｯﾄの出力値（ $X^l_j = a^l(z^l_j)$ ）
- C : ｺｽﾄ関数（平均二乗誤差(MSE)の場合 $C=(1/K)\sum_{k=1}^{K}(t^L_k - X^L_k)^2$ （K：は出力層のﾕﾆｯﾄ数） ）

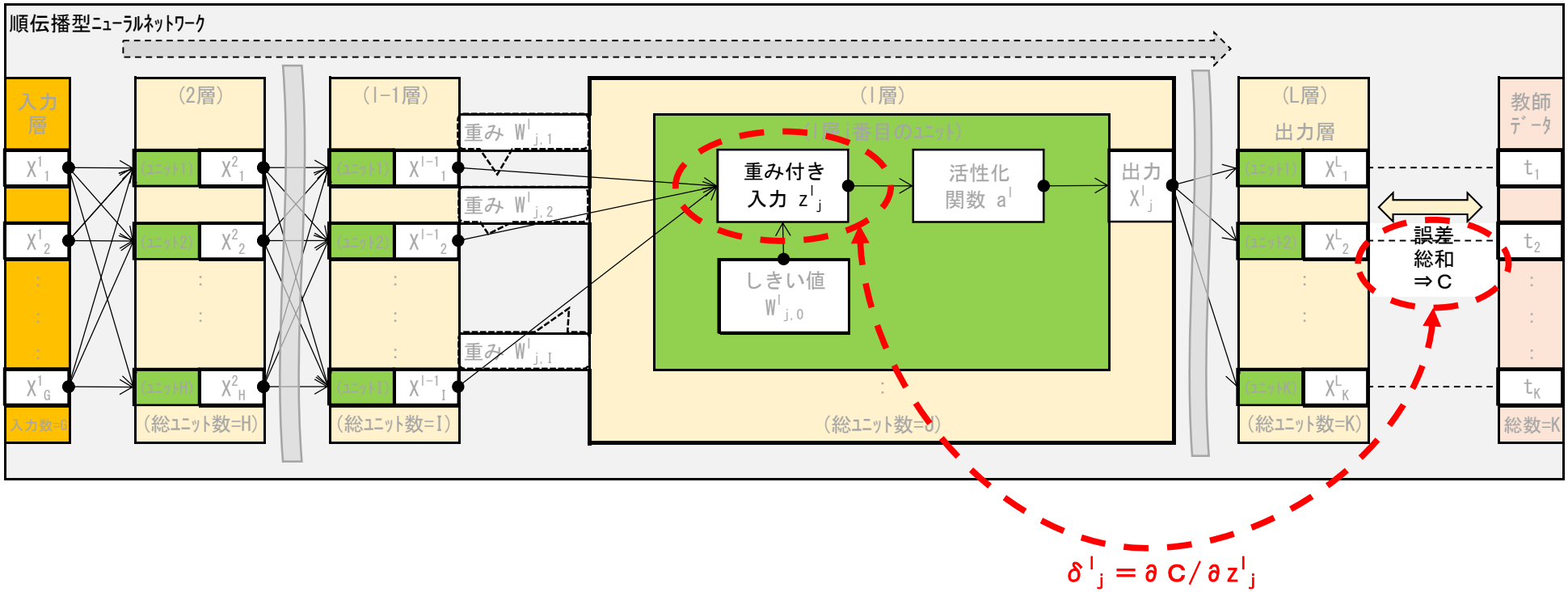
(8.4.1) 各層のﾕﾆｯﾄの誤差

・はじめに「l層j番目のﾕﾆｯﾄの誤差」δ^{l}_{j} なる概念を導入します。
これは、次の定義で与えられるもので、l層j番目のﾕﾆｯﾄの重み付き入力 z^{l}_{j} が、ｺｽﾄ関数Cに与える変化率を意味しています。
「誤差逆伝播法」では、このﾕﾆｯﾄの誤差「δ^{l}_{j}」についての逆漸化式を解くことにより、ﾊﾟﾗﾒｰﾀの最適化を行います。

「l層j番目のﾕﾆｯﾄの誤差」の定義

$$\delta^{l}_{j} = \partial C / \partial z^{l}_{j} \qquad \cdots(式8.4\text{-}1)$$

- l ：層番号（l=2～L。説明上入力層を第1層とします）
- j ：l層でのﾕﾆｯﾄ番号（j=1～J）
- δ^{l}_{j} ：l層j番目のﾕﾆｯﾄの誤差
- C ：ｺｽﾄ関数
- z^{l}_{j} ：l層j番目のﾕﾆｯﾄの重み付き入力



(8.4.2) ユニットの誤差とコスト関数の偏微分との関係式

・「l層j番目のユニットの誤差 δ^{l}_{j}」と、コスト関数Cの重み $W^{l}_{j,i}$ に関する偏微分について、以下の関係があります：

「l層j番目のユニットの誤差 δ^{l}_{j}」と、コスト関数Cの重み $W^{l}_{j,i}$ に関する偏微分との関係式

$$\partial C / \partial W^{l}_{j,i} = \delta^{l}_{j} \, X^{l-1}_{i} \quad \cdots(式8.4\text{-}2)$$

特に i=0（バイアス用のユニット番号）の場合、

$$\partial C / \partial W^{l}_{j,0} = \delta^{l}_{j} \quad \cdots(式8.4\text{-}3)$$

- C ：コスト関数
- l ：層番号（l=2～L、説明上、入力層を第1層とします）
- j ：l層でのユニット番号（j=1～J）
- i ：l-1層でのユニット番号（i=1～I， i=0 はバイアス用のユニット番号）
- $W^{l}_{j,i}$ ：l層j番目のユニットについて、l-1層i番目のユニットからの入力信号の重み
- $W^{l}_{j,0}$ ：l層j番目のユニットについて、l-1層からの入力信号に付加するバイアス
- X^{l-1}_{i} ：l-1層i番目のユニットの出力値（l層への入力信号）
- X^{l-1}_{0} ：l-1層からのユニットの出力のバイアス用の出力値（1固定）
- δ^{l}_{j} ：l層j番目のユニットの誤差

（説明）

l層j番目のユニットの重み付き入力 z^{l}_{j} は、次式で与えられます

$$z^{l}_{j} = \sum_{i=0}^{I} W^{l}_{j,i} * X^{l-1}_{i}$$

これより

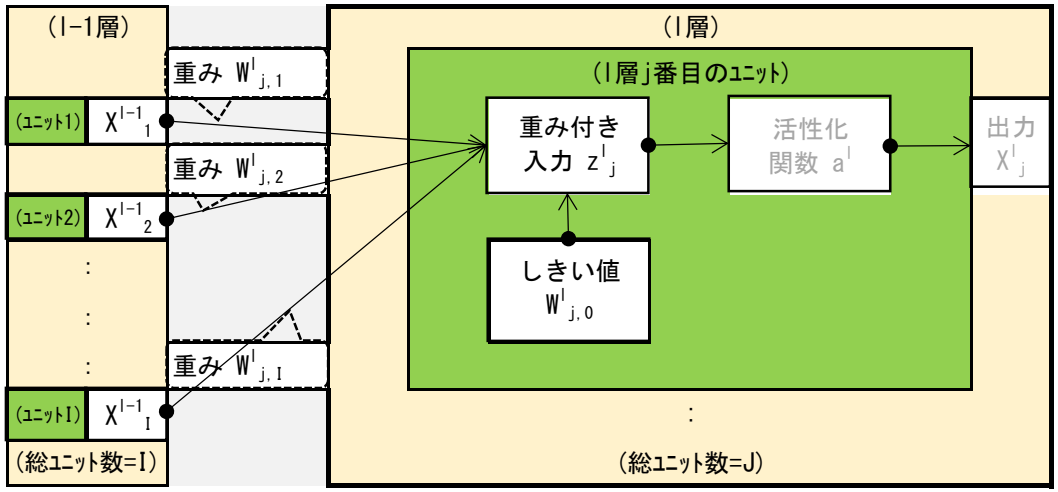
$$\partial z^{l}_{j} / \partial W^{l}_{j,i} = X^{l-1}_{i}$$

従って

$$\partial C / \partial W^{l}_{j,i} = (\partial C / \partial z^{l}_{j})(\partial z^{l}_{j} / \partial W^{l}_{j,i}) = \delta^{l}_{j} \, X^{l-1}_{i}$$

特に i=0（バイアス用のユニット番号）の場合、$X^{l-1}_{0}=1$ より、

$$\partial C / \partial W^{l}_{j,0} = \delta^{l}_{j}$$



(8.4.3) 出力層のﾕﾆｯﾄの誤差

・出力層(L層) j番目のﾕﾆｯﾄの誤差 δ^L_j は、以下のように具体的に計算できます：

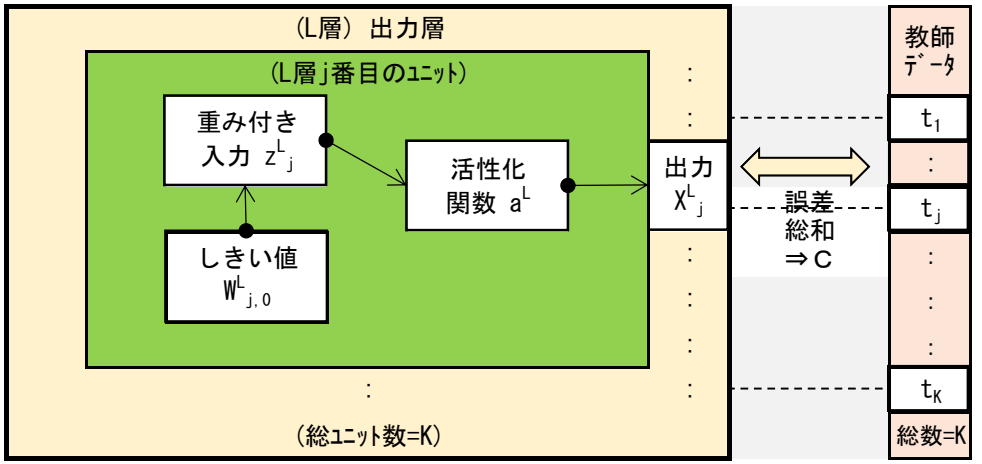
出力層(L層) j番目のﾕﾆｯﾄの誤差 δ^L_j の計算

$$\delta^L_j =_{(*0)} \partial C / \partial z^L_j$$
$$=_{(*1)} (\partial C / \partial X^L_j)(\partial X^L_j / \partial z^L_j) \quad \cdots(式8.4\text{-}4)$$

(*0) ：ﾕﾆｯﾄの誤差の定義

(*1) ：$X^L_j = a^L(z^L_j)$ の関係式から偏微分の連鎖律を適用

- δ^L_j ：出力層j番目のﾕﾆｯﾄの誤差
- L ：出力層の層番号
- j ：出力層でのﾕﾆｯﾄ番号（j=1～K）
- C ：ｺｽﾄ関数
- z^L_j ：出力層j番目のﾕﾆｯﾄの重み付き入力値
- X^L_j ：出力層j番目のﾕﾆｯﾄの出力値 $X^L_j = a^L(z^L_j)$
- a^L ：出力層の活性化関数



（ 出力層のﾕﾆｯﾄの誤差の計算例 ）

・出力層(L層) に関して、ｺｽﾄ関数Ｃとして平均二乗誤差(MSE)を適用する場合、
ｺｽﾄ関数Ｃは、「学習ﾃﾞｰﾀ」の入力値から算出した予測値 X^L_j と正解値 t_jの平均二乗誤差の総和に相当します。

$$C = 1/(N*K) \ \Sigma^N_{n=1} \ \Sigma^K_{j=1} \ (X^L_j[n] - t_j[n])^2$$

$$\therefore \ \partial C/\partial X^L_j = 2/(N*K) \times (X^L_j - t_j)$$

N,n ：学習ﾃﾞｰﾀ数とその参照添え字
K,j ：出力層のﾕﾆｯﾄ数とその参照添え字
$X^L_j[n]$ ：n番目のﾃﾞｰﾀについての出力層j番目ﾕﾆｯﾄの予測値
$t_j[n]$ ：n番目のﾃﾞｰﾀについての出力層j番目ﾕﾆｯﾄの教師ﾃﾞｰﾀ
X^L_j ：$\Sigma^N_{n=1} \ X^L_j[n]$
t_j ：$\Sigma^N_{n=1} \ t_j[n]$
z^L_j ：出力層j番目のﾕﾆｯﾄの重み付き入力
$=\Sigma_{i=0}{}^I \ W^L_{j,i} * X^{L-1}_i$ （但し $X^{L-1}{}_0=1$））

活性化関数 a として、
ｼｸﾞﾓｲﾄﾞ関数 $\sigma(z)=1/(1+\exp(-z))$ を持ちいる場合、

$$X^L_j = \sigma(z^L_j)$$

$$\therefore \ \partial X^L_j/\partial z^L_j = \partial \sigma(z^L_j)/\partial z^L_j$$

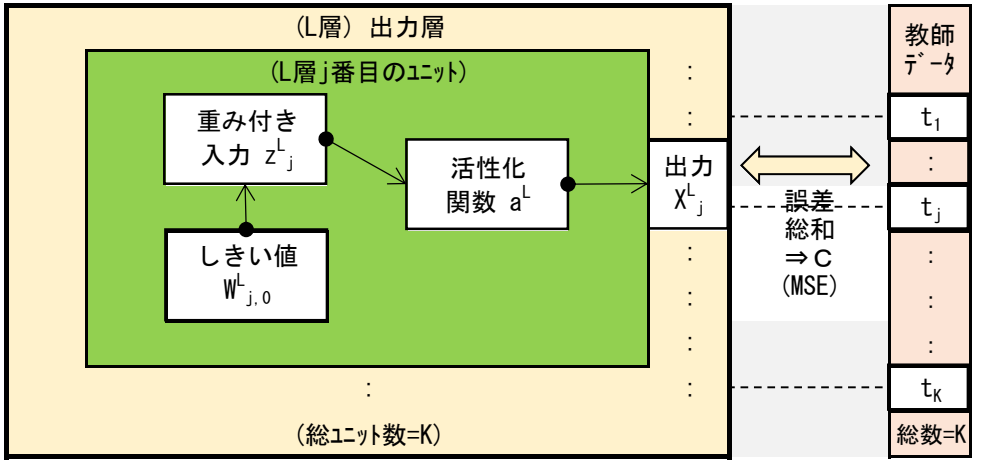
$$= \sigma(z^L_j) \ \{1 - \sigma(z^L_j)\}$$

従って、この時の「出力層(l=L) j番目のﾕﾆｯﾄの誤差」δ^L_j は、

$$\delta^L_j =_{(*1)} (\partial C/\partial X^L_j)(\partial X^L_j/\partial z^L_j)$$

$$=_{(*2)} 2/(N*K) \times (X^L_j - t_j) \times \sigma(z^L_j) \ \{1 - \sigma(z^L_j)\}$$

(*1) ：(式8.4-4)
(*2) ：上の関係式



(8.4.4) 隣接する層間のﾕﾆｯﾄの誤差についての逆漸化式

・中間層である l層j番目のﾕﾆｯﾄに関して、l-1層i番目のﾕﾆｯﾄとの間に、次の逆漸化式が成り立ちます：

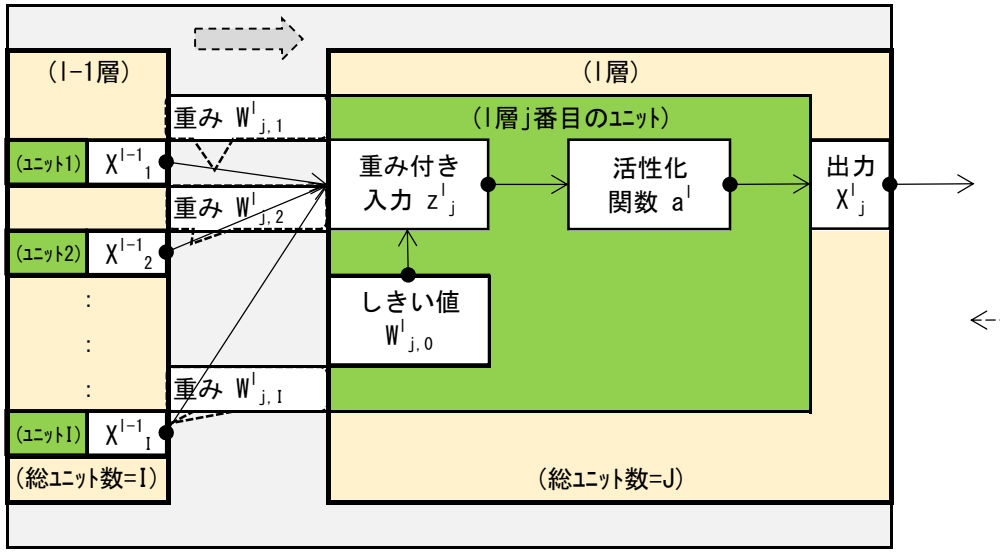
l層j番目のﾕﾆｯﾄと、l-1層i番目のﾕﾆｯﾄとの間に成立する逆漸化式

$$\delta^{l-1}_i = \{\sum_{j=1}^{J} \delta^{l}_j * W^{l}_{j,i}\} * \partial a^{l-1} / \partial z^{l-1}_i$$

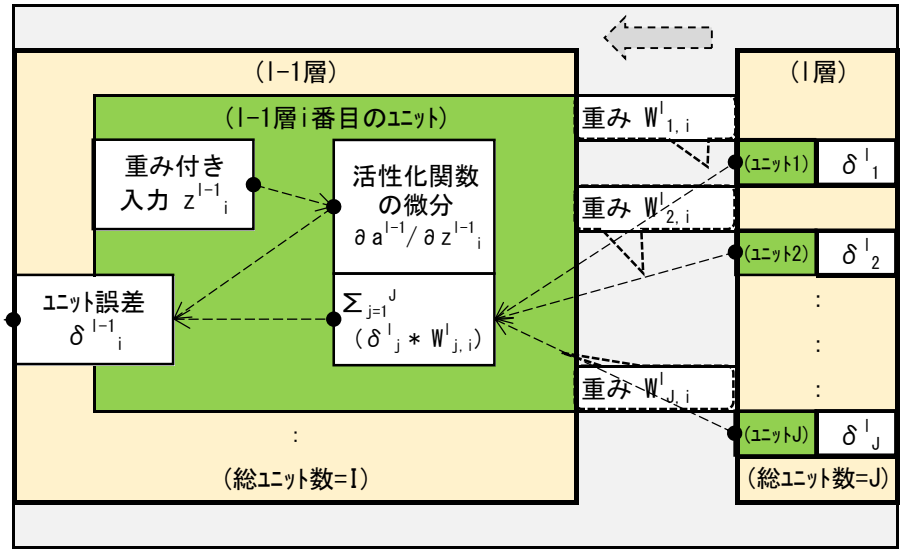
…(式8.4-5)

- l ：層番号(l=2～L、説明上入力層を第1層とします)
- δ^{l-1}_i ：l-1層i番目のﾕﾆｯﾄの誤差
- δ^{l}_j ：l層j番目のﾕﾆｯﾄの誤差
- I, i ：l-1層でのﾕﾆｯﾄ数, ﾕﾆｯﾄ番号 (i=1～I)
- J, j ：l層でのﾕﾆｯﾄ数, ﾕﾆｯﾄ番号 (j=1～J)
- $W^{l}_{j,0}$ ：l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽ
- $W^{l}_{j,i}$ ：l層j番目のﾕﾆｯﾄへのl-1層i番目のﾕﾆｯﾄからの入力信号の重み
- a^{l-1} ：l-1層の活性化関数
- z^{l-1}_i ：l-1層i番目のﾕﾆｯﾄの重み付き入力値

(出力計算時の流れ)



(誤差逆伝搬時の流れ)



・(式8.4-5)の導出について、補足説明を加えておきます。

$$\delta^{l-1}_i = {}_{(※0)}\ \partial C/\partial z^{l-1}_i$$
$$= {}_{(※1)}\ \sum_{j=1}^{J} (\partial C/\partial z^{l}_j)(\partial z^{l}_j/\partial X^{l-1}_i)(\partial X^{l-1}_i/\partial z^{l-1}_i)$$
$$= {}_{(※0,2,3)}\ \{ \sum_{j=1}^{J} \delta^{l}_j * W^{l}_{j,i} \} * \partial a^{l-1}/\partial z^{l-1}_i$$

(※0) : ﾕﾆｯﾄの誤差の定義

$$\delta^{l-1}_i = \partial C/\partial z^{l-1}_i$$
$$\delta^{l}_j = \partial C/\partial z^{l}_j$$

(※1) : 以下の関数関係に基づいて偏微分の連鎖律を適用

$$X^{l-1}_i = a^{l-1}(z^{l-1}_i)$$
$$z^{l}_j = \sum_{i=0}^{I} W^{l}_{j,i} * X^{l-1}_i$$
$$= \sum_{i=0}^{I} W^{l}_{j,i} * a^{l-1}(z^{l-1}_i)$$

(※2) : l層の第jﾕﾆｯﾄの重み付き入力

$$z^{l}_j = \sum_{i=0}^{I} W^{l}_{j,i} * X^{l-1}_i$$

より、X^{l-1}_i についての偏微分をとると、

$$\partial z^{l}_j/\partial X^{l-1}_i = W^{l}_{j,i}$$

(※3) : l-1層の第iﾕﾆｯﾄの出力は、活性化関数 a^{l-1} を用いて

$$X^{l-1}_i = a^{l-1}(z^{l-1}_i)$$

より、その偏微分をとると、

$$\partial X^{l-1}_i/\partial z^{l-1}_i = \partial a^{l-1}/\partial z^{l-1}_i$$

C : ｺｽﾄ関数
l : 層番号(l=2～L、説明上入力層を第1層とします)

I, i : l-1層でのﾕﾆｯﾄ数, ﾕﾆｯﾄ番号 (i=1～I)
δ^{l-1}_i : l-1層i番目のﾕﾆｯﾄの誤差
z^{l-1}_i : l-1層i番目のﾕﾆｯﾄの重み付き入力値
X^{l-1}_i : l-1層i番目のﾕﾆｯﾄの出力値
a^{l-1} : l-1層の活性化関数

J, j : l層でのﾕﾆｯﾄ数, ﾕﾆｯﾄ番号 (j=1～J)
δ^{l}_j : l層j番目のﾕﾆｯﾄの誤差
z^{l}_j : l層j番目のﾕﾆｯﾄの重み付き入力値
X^{l}_j : l層j番目のﾕﾆｯﾄの出力値
a^{l} : l層の活性化関数
$W^{l}_{j,0}$: l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽ
$W^{l}_{j,i}$: l層j番目のﾕﾆｯﾄへのl-1層i番目のﾕﾆｯﾄからの入力信号の重み

(8.4.5) 全ての中間層のﾕﾆｯﾄの誤差δを計算する

・「漸化式（ｾﾞﾝｶｼｷ、recurrence relation)」の場合は、初項と、第n項で第n+1項を表現する関係式を与えると、全ての項が計算できます。
これに対し「逆漸化式」の場合は、末項と、第n項で第n-1項を表現する関係式を与えると、全ての項が計算できます。

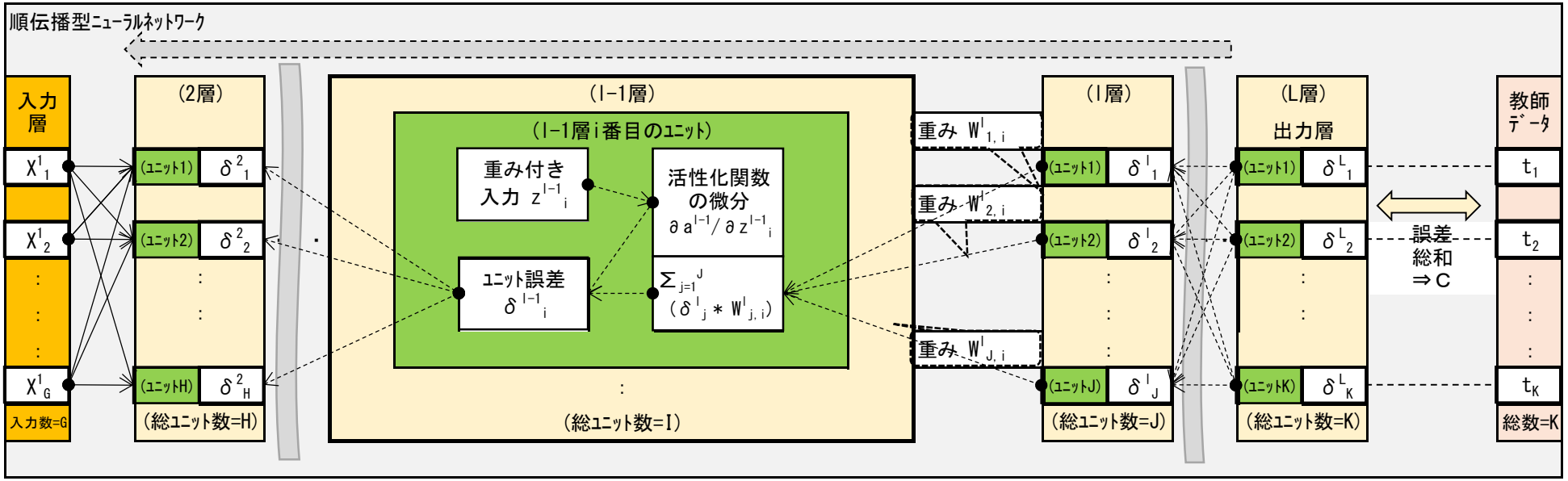
・「(8.4.3) 出力層のﾕﾆｯﾄの誤差」で末項「δ^{L}_{j}」を与え、
「(8.4.4) 隣接する層間のﾕﾆｯﾄの誤差についての逆漸化式」で、
第l-1層i番目のﾕﾆｯﾄの誤差「δ^{l-1}_{i}」を、第l層j番目のﾕﾆｯﾄの誤差「δ^{l}_{j}」で表現する関係式を与えることにより、
ﾕﾆｯﾄの誤差δは、全ての中間層のﾕﾆｯﾄについて計算することができます。

出力層j番目のﾕﾆｯﾄの誤差 δ^{L}_{j}

$$\delta^{L}_{j} = (\partial C / \partial X^{L}_{j})(\partial X^{L}_{j} / \partial z^{L}_{j})$$ …(式8.4-4) 再掲

l層j番目のﾕﾆｯﾄと、l-1層i番目のﾕﾆｯﾄとの間に成立する逆漸化式

$$\delta^{l-1}_{i} = \{\sum_{j=1}^{J} \delta^{l}_{j} * W^{l}_{j,i}\} * \partial a^{l-1} / \partial z^{l-1}_{i}$$ …(式8.4-5) 再掲



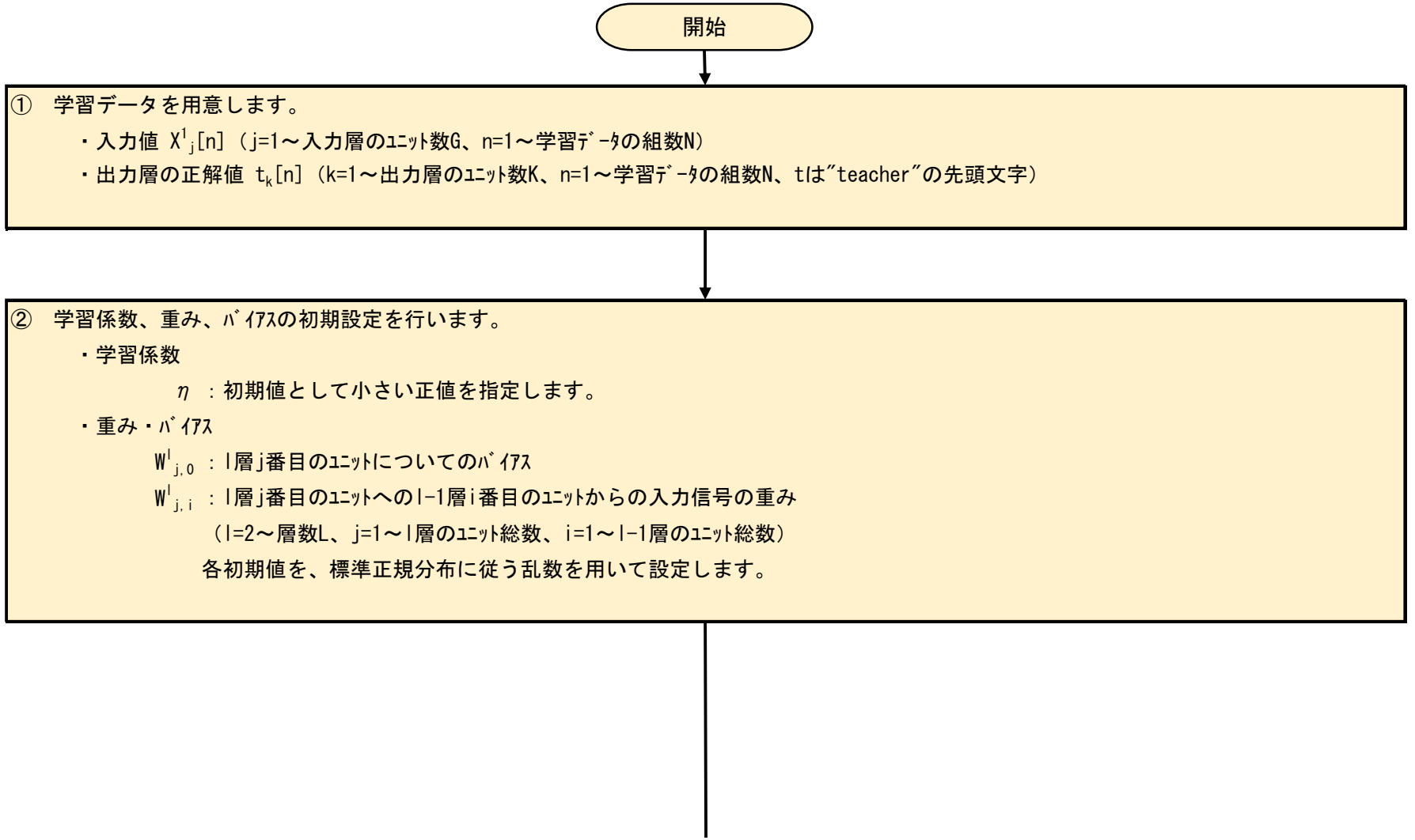
(8.4.6) ﾕﾆｯﾄの誤差δを用いて勾配降下法で学習する流れ（例）

・ﾕﾆｯﾄの誤差δを導入した「誤差逆伝播法」による学習(ﾓﾃﾞﾙのﾊﾟﾗﾒｰﾀ最適化）のｱﾙｺﾞﾘｽﾞﾑ例は、以下のようになります。

以下の例では、ｺｽﾄ関数Ｃ、第l層の活性化関数 a^l として、次のものを使用します。

(1) ｺｽﾄ関数Ｃとして二乗誤差(SE)

(2) 第l層の活性化関数 a^l としてｼｸﾞﾓｲﾄﾞ関数



③ 現在の重みとﾊﾞｲｱｽを用いて、各層のﾕﾆｯﾄの出力値を第2層（中間層の第1層）から出力層に向かって順番に計算し（ﾌｫﾜｰﾄﾞﾌﾟﾛﾊﾟｹﾞｰｼｮﾝ）、
最後に出力層の出力値と学習ﾃﾞｰﾀとの二乗誤差(SE)を算出し、これをｺｽﾄ関数Cとします。
（l=2～層数L分、l=1:入力層、l=L:出力層、j=1～各層l毎のﾕﾆｯﾄ数、a^lは第l層の活性化関数）

・入力層の出力値

X^1_j ：入力層j番目ﾕﾆｯﾄの出力値

・中間層・出力層の出力値

X^l_j ：第l層j番目ﾕﾆｯﾄの出力値

$= a^l \left(\sum_{i=0}^{l-1層のﾕﾆｯﾄ数} (W^l_{j,I} * X^{l-1}_i[n]) \right)$

・ｺｽﾄ関数(二乗誤差(SE))

$C = \sum_{j=1}^{出力層のﾕﾆｯﾄ数} 1/2 \ (X^L_j[n] - t_j[n])^2$

n ：学習ﾃﾞｰﾀの組番号。n=1～学習ﾃﾞｰﾀの組数N
$W^l_{j,0}$ ：l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽ
$W^l_{j,i}$ ：l層j番目のﾕﾆｯﾄへのl-1層i番目のﾕﾆｯﾄからの入力信号の重み
X^{l-1}_0 ：l-1層からのﾕﾆｯﾄの出力のﾊﾞｲｱｽ用の出力値（1固定）
X^{l-1}_i ：l-1層i番目のﾕﾆｯﾄの出力値

④ 誤差逆伝播法から、出力層から中間層に向かって逆漸化式を用いて、各層のﾕﾆｯﾄの誤差δを計算します(ﾊﾞｯｸﾌﾟﾛﾊﾟｹﾞｰｼｮﾝ)。
(例では、第l層の活性化関数 a^l としてｼｸﾞﾓｲﾄﾞ関数「$\sigma(z)=1/(1+\exp(-z))$」を使用)

・出力層のﾕﾆｯﾄの誤差

δ^L_j ：出力層j番目のﾕﾆｯﾄの誤差（j=1～出力層のﾕﾆｯﾄ総数）

$= (\partial C/\partial X^L_j)(\partial X^L_j/\partial z^L_j)$

$= (X^L_j - t_j) \exp(-z^L_j)/(1+\exp(-z^L_j))$

z^l_j ：l層j番目のﾕﾆｯﾄの重み付き入力

$= \sum_{i=0}^{l-1層のﾕﾆｯﾄ総数} W^l_{j,i} * X^{l-1}_i$ （但し $X^{l-1}_0=1$）

・中間層のﾕﾆｯﾄの誤差

δ^{l-1}_i ：第l-1層i番目のﾕﾆｯﾄの誤差（l=3～L-1、i=1～l層のﾕﾆｯﾄ総数）

$= \left\{ \sum_{j=1}^{l層のﾕﾆｯﾄ数} (\delta^l_j \times W^l_{j,i}) \right\} * \partial a^{l-1}/\partial z^{l-1}_i$

$= \left\{ \sum_{j=1}^{l層のﾕﾆｯﾄ数} (\delta^l_j \times W^l_{j,i}) \right\} * \exp(-z^{l-1}_i) / (1 + \exp(-z^{l-1}_i))$

⑤　各層のﾕﾆｯﾄの誤差δからｺｽﾄ関数Ｃの、重み・ﾊﾞｲｱｽに関する偏微分を計算します(ﾊﾞｯｸﾌﾟﾛﾊﾟｹﾞｰｼｮﾝの続き)。

(l=2～層数L分、j=1～l層の各層毎のﾕﾆｯﾄ総数、i=1～l-1層の各層毎のﾕﾆｯﾄ総数)

・重みに関するｺｽﾄ関数の偏微分

l層j番目のﾕﾆｯﾄについて、l-1層i番目のﾕﾆｯﾄからの入力信号の重み$W^{l}_{j,i}$ に関するｺｽﾄ関数C の偏微分

$$\partial C / \partial W^{l}_{j,i} = \delta^{l}_{j} \; X^{l-1}_{i}$$

・ﾊﾞｲｱｽに関するｺｽﾄ関数の偏微分

l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽに関するｺｽﾄ関数の偏微分

$$\partial C / \partial W^{l}_{j,0} = \delta^{l}_{j}$$

$W^{l}_{j,i}$ ：l層j番目のﾕﾆｯﾄの、l-1層i番目のﾕﾆｯﾄからの入力信号の重み
$W^{l}_{j,0}$ ：l層j番目のﾕﾆｯﾄの、l-1層からの入力信号に付加するﾊﾞｲｱｽ
X^{l-1}_{i} ：l-1層i番目のﾕﾆｯﾄの出力値（l層への入力信号）

⑥　上記③～⑤の結果を全入力ﾃﾞｰﾀについて加え合わせ、ｺｽﾄ関数C_Tとその勾配∇C_T を算出します。

・全入力ﾃﾞｰﾀについて加え合わせたｺｽﾄ関数

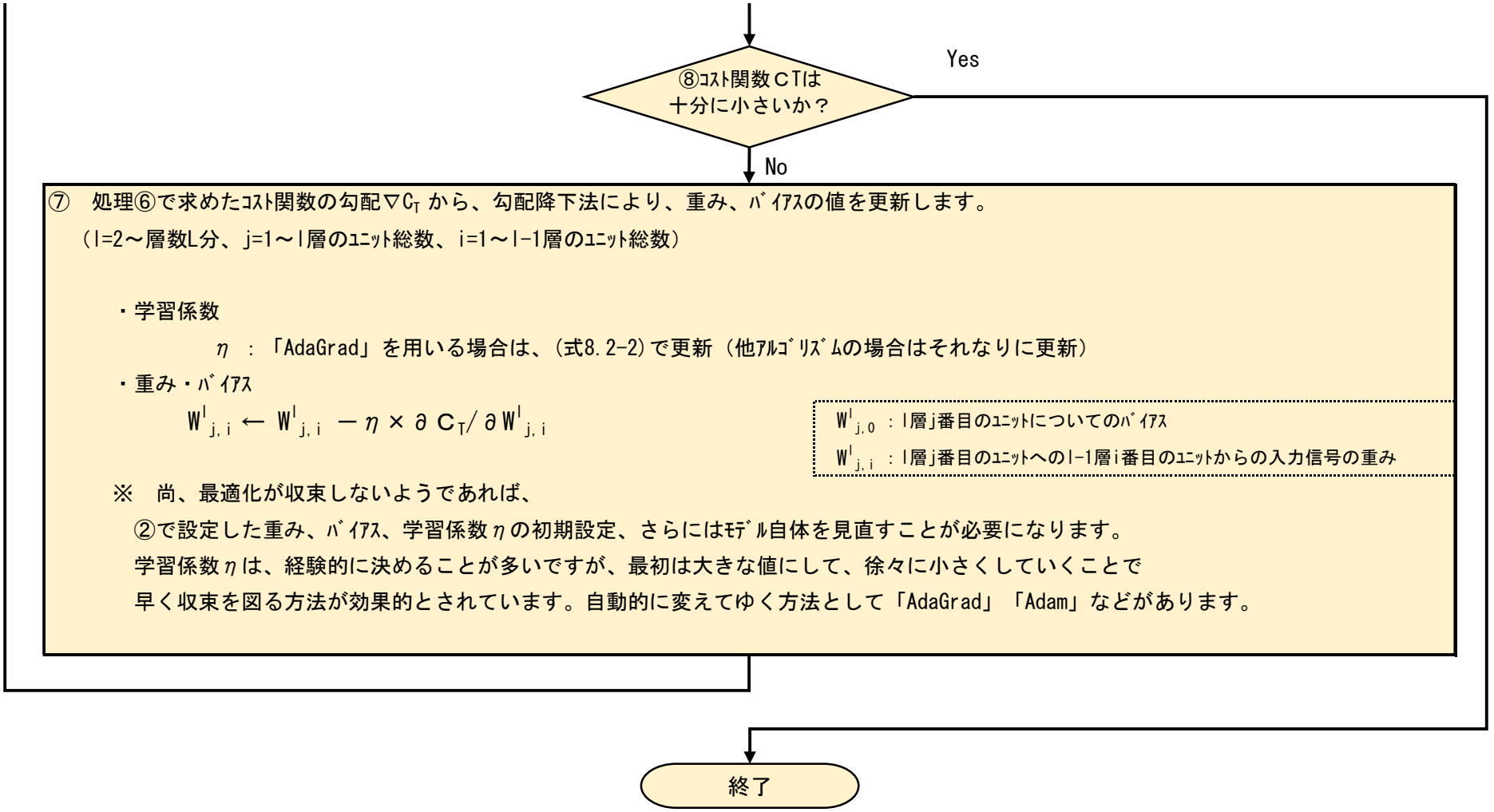
$C_T = \Sigma^{学習ﾃﾞｰﾀ数}$ 処理③で算出されたＣ

l ：層番号（l=2～層数L分）
j ：l層でのﾕﾆｯﾄ番号（j=1～l層のﾕﾆｯﾄ総数）
i ：l-1層でのﾕﾆｯﾄ番号（i=1～l-1層のﾕﾆｯﾄ総数）
n ：学習ﾃﾞｰﾀの組番号。n=1～学習ﾃﾞｰﾀの組数N

・全入力ﾃﾞｰﾀについて加え合わせたｺｽﾄ関数の勾配

$\nabla C_T = \Sigma^{学習ﾃﾞｰﾀ数}$ 処理⑤で算出されたＣの偏微分を成分とするベクトル

$= (\partial C_T / \partial W^{l}_{j,0}、\partial C_T / \partial W^{l}_{j,1}、・・・、\partial C_T / \partial W^{l}_{j,l-1層のﾕﾆｯﾄ総数})$


⑧ｺｽﾄ関数ＣＴは
十分に小さいか？
Yes
No
⑦　処理⑥で求めたｺｽﾄ関数の勾配▽C_T から、勾配降下法により、重み、ﾊﾞｲｱｽの値を更新します。
（l=2～層数L分、j=1～l層のﾕﾆｯﾄ総数、i=1～l-1層のﾕﾆｯﾄ総数）
・学習係数
η ：「AdaGrad」を用いる場合は、(式8.2-2)で更新（他ｱﾙｺﾞﾘｽﾞﾑの場合はそれなりに更新）
・重み・ﾊﾞｲｱｽ
$W^{l}_{j,i} \leftarrow W^{l}_{j,i} - \eta \times \partial C_T / \partial W^{l}_{j,i}$
$W^{l}_{j,0}$ ：l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽ
$W^{l}_{j,i}$ ：l層j番目のﾕﾆｯﾄへのl-1層i番目のﾕﾆｯﾄからの入力信号の重み
※　尚、最適化が収束しないようであれば、
②で設定した重み、ﾊﾞｲｱｽ、学習係数ηの初期設定、さらにはﾓﾃﾞﾙ自体を見直すことが必要になります。
学習係数ηは、経験的に決めることが多いですが、最初は大きな値にして、徐々に小さくしていくことで
早く収束を図る方法が効果的とされています。自動的に変えてゆく方法として「AdaGrad」「Adam」などがあります。
終了

(8.5) ﾓﾃﾞﾙの学習の検証

・ﾓﾃﾞﾙを作成し、学習を通してﾓﾃﾞﾙのﾊﾟﾗﾒｰﾀの最適化を行います。
学習時のﾃﾞｰﾀではｺｽﾄ関数Ｃが最小になるものの、学習時以外のﾃﾞｰﾀでは誤差が大きく出てしまうことがあります。
これは、学習時のﾃﾞｰﾀに特化した学習をしてしまって、汎用になっていないということであり、
これを「過学習（ｶｶﾞｸｼｭｳ、over fitting)」と言います。

(8.5.1) ﾓﾃﾞﾙの学習の検証方法

・使用しているﾓﾃﾞﾙと、そのﾊﾟﾗﾒｰﾀの学習状況を把握する為の検証方法を幾つか紹介します。
以下は、ﾃﾞｰﾀを、ﾓﾃﾞﾙの学習用と、評価用に分けて評価する方法です。
学習用ﾃﾞｰﾀは、「訓練ﾃﾞｰﾀ（ｸﾝﾚﾝﾃﾞｰﾀ、training data)」「訓練事例集合（ｸﾝﾚﾝｼﾞﾚｲｼｭｳｺﾞｳ、training set)」とも言います。
評価用ﾃﾞｰﾀは、「ﾃｽﾄﾃﾞｰﾀ（test data)」「ﾃｽﾄ事例集合（ﾃｽﾄｼﾞﾚｲｼｭｳｺﾞｳ、testing set)」とも言います。

検証方法	説明	長所・短所
ホールドアウト検証 (hold out validation)	・集められたﾃﾞｰﾀのうち、学習専用と評価専用の２つのグループに分けて、学習ﾃﾞｰﾀでモデルを作成し、評価ﾃﾞｰﾀで作成されたモデルの評価を行う。 これにより作成したモデルがどの位未知ﾃﾞｰﾀを判別できるかが評価でき、過学習を防ぐことにつながる。 ・尚、評価ﾃﾞｰﾀは学習ﾃﾞｰﾀに含めてはいけない。	・集めたﾃﾞｰﾀを学習専用と評価専用に分けるため、学習に使えるﾃﾞｰﾀの量が減るという欠点がある。 ・ﾃﾞｰﾀを学習用と評価用に分ける際、分け方に偶然偏りができた場合にはうまく評価ができないという欠点がある。。 ・ﾃﾞｰﾀ数が少ない時は、上記の欠点が顕著に表れる。 ・ﾃﾞｰﾀ量が多くなると「K-分割交差検証」の結果と殆ど変わらなくなる。
K-分割交差検証 (K-fold cross-validation)	・集められたﾃﾞｰﾀ（説明変数と目的変数）をＫ個に分割する。そのうちの１個を評価ﾃﾞｰﾀとし、残りのＫ－１個を学習ﾃﾞｰﾀとして、モデルのパラメータを算出して、評価ﾃﾞｰﾀでの平均二乗誤差を計算する。 これをＫ個分繰り返して、平均二乗誤差の平均をこのモデルの評価値とする。	・ﾃﾞｰﾀ量が多くなると計算に時間がかかるという欠点がある。
leave-one-out 交差検証 （一個抜き交差検証） (leave-one-out cross-validation (LOOCV))	・K-分割交差検証で、分割数Ｋをﾃﾞｰﾀ数Ｎとした場合の検証である。この時、評価ﾃﾞｰﾀ数は１、学習ﾃﾞｰﾀ数はＮ－１になる。 ﾃﾞｰﾀ数が特に少ない時にこの手法が使われる。	・ﾃﾞｰﾀ量が多くなると計算に時間がかかるという欠点がある。

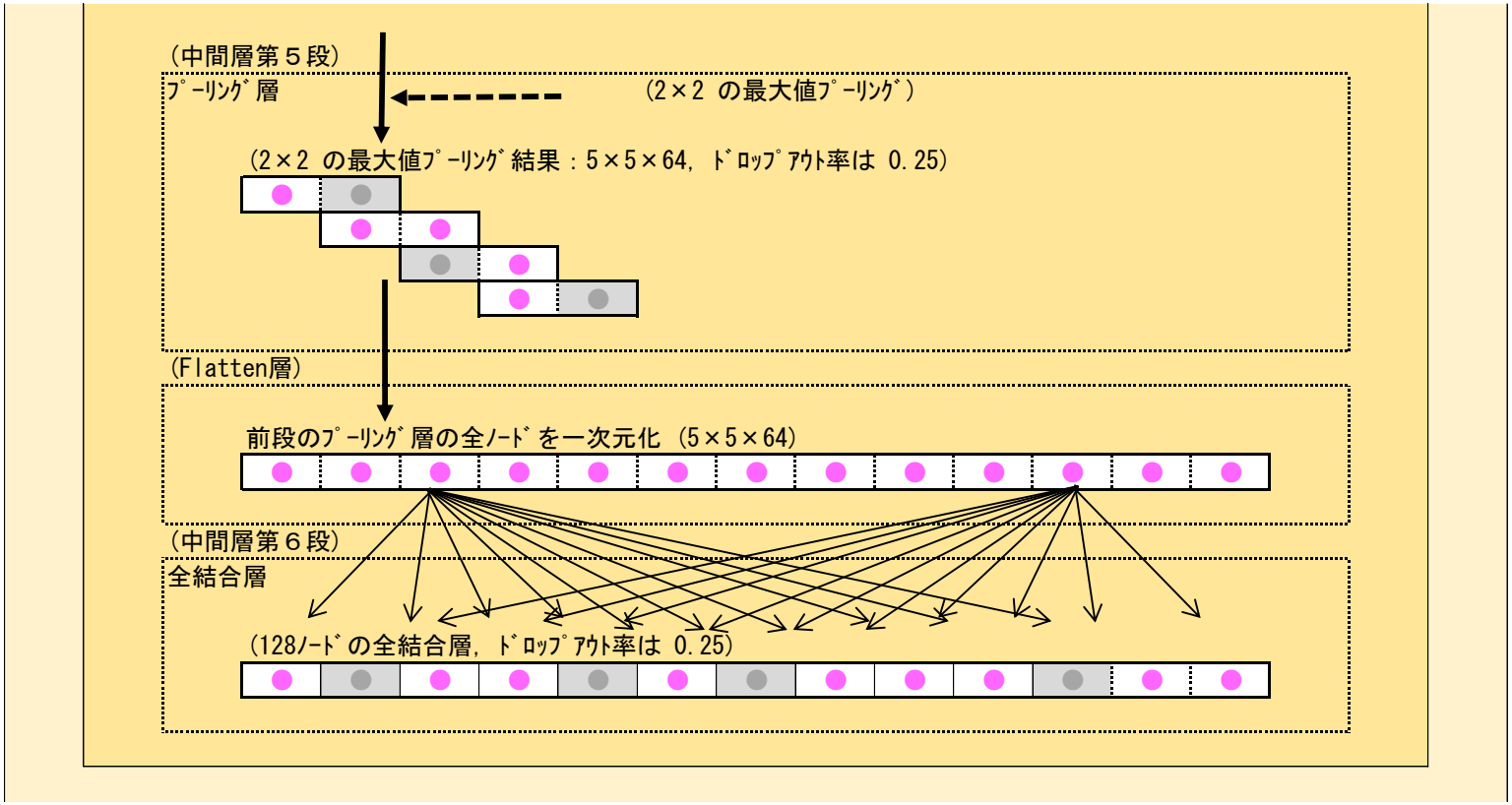
(8.5.2) 過学習の対策とﾄﾞﾛｯﾌﾟｱｳﾄ

・「過学習」の対策として、入力にﾉｲｽﾞを与えたり、学習時に正則化を適用したり、様々な方法が提案されてきましたが、
「ﾄﾞﾛｯﾌﾟｱｳﾄ (Dropout)」という方法が有効であることが知られています。

・「ﾄﾞﾛｯﾌﾟｱｳﾄ」は、学習時に入力層や中間層のﾕﾆｯﾄを、各層毎に一定の割合α (0≦α＜1) でﾕﾆｯﾄを選出し、
それ以外は無効な状態で学習を行います（選出されなかったﾕﾆｯﾄはないものとして学習を行います）。
これをﾐﾆﾊﾞｯﾁ毎にﾕﾆｯﾄを選出しなおして、学習を繰り返します。

・学習後に予測をする場合は、全てのﾕﾆｯﾄを使用しますが、学習時はαの割合のﾕﾆｯﾄしか存在しなかったので、
予測時にはﾄﾞﾛｯﾌﾟｱｳﾄを施した層の出力の重みをα倍にして出力が大きくなることを防ぎます。

・既述のﾓﾃﾞﾙ「手書き文字認識ﾓﾃﾞﾙ３」では、ﾌﾟｰﾘﾝｸﾞ層と全結合層について、ﾄﾞﾛｯﾌﾟｱｳﾄ率α＝0.25 で作成しています。



【出典・参考】

勾配降下法⇒ https://qiita.com/YudaiSadakuni/items/ece07b04c685a64eaa01

勾配降下法⇒ https://postd.cc/optimizing-gradient-descent/#batchgradientdescent

AdaGrad、Adam⇒ https://qiita.com/tokkuman/items/1944c00415d129ca0ee9

AdaGrad、Adam⇒ 「深層学習 Deep Learning」人工知能学会監修（近代科学社 2015年11月）

ﾓﾃﾞﾙの学習⇒https://ml4a.github.io/ml4a/jp/how_neural_networks_are_trained/

本章全般⇒「Pythonで動かして学ぶ！ あたらしい機械学習の教科書」伊藤真著（翔泳社 2018年1月）

確率的勾配降下法⇒ http://yaju3d.hatenablog.jp/entry/2017/08/27/233459

誤差逆伝搬法⇒ 「ﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞがわかる数学入門」涌井良幸・貞美著（技術評論社 2017.04）

ﾄﾞﾛｯﾌﾟｱｳﾄ⇒ 「深層学習 Deep Learning」人工知能学会監修（近代科学社 2015年11月）

(9) まとめ

・以上、「教師あり学習（分類）」を中心にして機械学習の基本的な技術的要素を解説してきましたが、如何でしたでしょうか？
今回の解説で、「ｾﾐﾅｰ7回目」の「ﾓﾃﾞﾙとその学習方法」が手に取るように読めるようになったのでは？と期待しています。

・誤差逆伝搬法については、「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」だけ取り上げましたが、
「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」について詳しく知りたい方は、以下の書籍などに詳しく書かれているので、参照してみてください。

「Pythonで動かして学ぶ！ あたらしい機械学習の教科書」伊藤真著（翔泳社 2018年1月）
「ﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞがわかる数学入門」涌井良幸・貞美著（技術評論社 2017.04）

・実装ﾚﾍﾞﾙではすでに提供されているＡＰＩを使いこなす・・といったことが中心となります。
ＡＰＩを使いこなすことができれば、仕事をこなすことは十分できるような時代になってはいますが、
今回のｾﾐﾅｰで取り上げたような基本的な背景を押さえておくことは、無駄にはならないと思います。

・今回取り上げた内容は機械学習の全体のほんの一部にすぎませんので、これを踏み台にして、
自分も含めて、どんどん取り組んで行っていただきたいと思います。

以上です。

(10) 確認問題

(1) 学習の種類

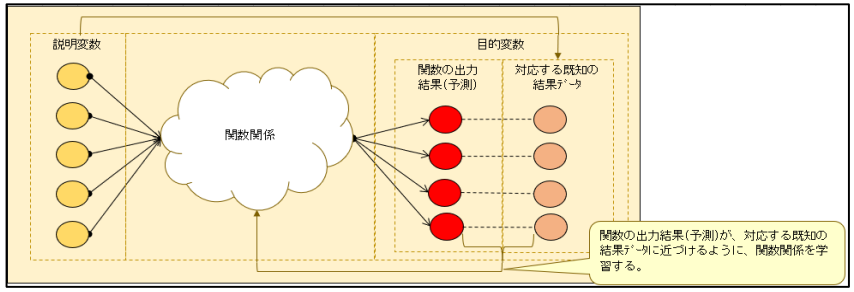
以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・(1.1)＿＿＿＿＿＿＿＿＿＿＿＿は　異なるﾃﾞｰﾀ間の関係性を、一方を入力（説明変数）、
他方を出力（目的変数）とする関数関係として記述し、関数の出力を既知の結果ﾃﾞｰﾀ（目的変数の既知ﾃﾞｰﾀ）に近づけるように、
関数関係を学習する方法です。この場合、関数の出力結果(予測)に対して、既知の結果ﾃﾞｰﾀが教師ﾃﾞｰﾀの役割をします(図10.1.1 参照)。

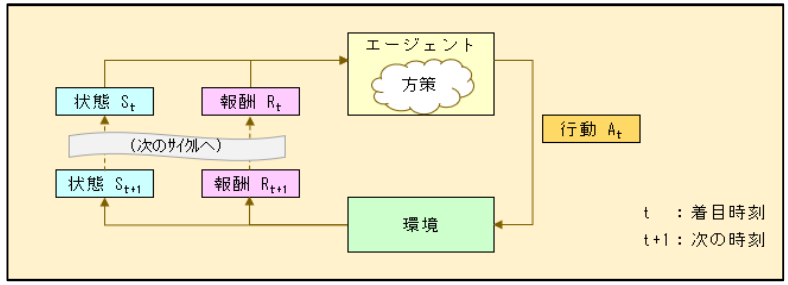
・(1.2)＿＿＿＿＿＿＿＿＿＿＿＿は、
入力変数（説明変数）に対する、出力変数（目的変数）が与えられておらず、入力ﾃﾞｰﾀ自体の背後にある性質や特徴や構造などを
分析・発見すること言います。例として、「ｸﾗｽﾀﾘﾝｸﾞ (Clustering)」によるﾃﾞｰﾀ分類や、
「次元削減 (ｼﾞｹﾞﾝｻｸｹﾞﾝ、Dimensionality Reduction)」による特徴量抽出などが挙げられます。

・(1.3)＿＿＿＿＿＿＿＿＿＿＿＿は、
「認知」した状況の下で最適な「行動」を「判断」すること、すなわち「制御」を学習する方法です(図10.1.2 参照)。
強化学習では、「エージェント (agent、制御器)」が「環境 (environment)」との相互作用を通じて情報収集しながら、
環境を制御する方法を学習します。「環境」は、状態とその変化を規定する制御対象の系(システム)のことです。
「エージェント」は、環境の置かれた状態にもとづいて行動を選択しながら環境を制御する主体で、制御プログラムにあたります。

(図10.1.1)



(図10.1.2)



(選択肢)

(a)　「強化学習 (ｷｮｳｶｶﾞｸｼｭｳ、Reinforcement learning)」
(b)　「教師あり学習 (ｷｮｳｼｱﾘｶﾞｸｼｭｳ, Supervised learning)」
(c)　「教師なし学習 (ｷｮｳｼﾅｼｶﾞｸｼｭｳ、Unsupervised Learning)」
(d)　「転移学習 (ﾃﾝｲｶﾞｸｼｭｳ、Transfer Learning)」

(回答)

(1.1)	
(1.2)	
(1.3)	

(2) 入力準備

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・入力ﾃﾞｰﾀ（説明変数）を統計処理するに先立って、
入力ﾃﾞｰﾀの中から重複や誤記、表記の揺れなどを探し出し、削除や修正などを行い、入力ﾃﾞｰﾀの品質を高める必要があります。
この作業を(2.1)＿＿＿＿＿＿＿＿または「ﾃﾞｰﾀｸﾚﾝｼﾞﾝｸﾞ (data cleansing)」と言います。

・入力ﾃﾞｰﾀ（説明変数）が数種類からなる場合、各入力ﾚｺｰﾄﾞに必ずしも全ての種類のﾃﾞｰﾀが揃っている訳ではありません。
欠けているﾃﾞｰﾀのことを(2.2)＿＿＿＿＿＿＿＿と言います。
欠損値がある場合、統計処理に支障があるため、当該項目を削除したり、何らかの値で補完したり、といった対応をします。

・平均と分散についての公式を用いて、平均μ、標準偏差σの確率変数 Xを、平均0、標準偏差1の確率変数 Zへ変換することが出来ます。
この操作を(2.3)＿＿＿＿＿＿＿＿と言います。これには、以下の様な効果があります。
(1) 特定の特徴に偏った結果が出ることを防ぐことが出来ます。
(2) 異なる項目のﾃﾞｰﾀであってもその大小を比較できるようになります。

・入力ﾃﾞｰﾀを符号化してﾓﾃﾞﾙに取り込んだ方が良い場合があります。
出力が K種類への分類ﾓﾃﾞﾙの場合、目的変数をK次元のﾍﾞｸﾄﾙとします。
ある入力ﾃﾞｰﾀが第k番目のｸﾗｽへ分類されることが分かっている場合、目的変数ﾍﾞｸﾄﾙの k番目の要素だけ 1でそれ以外を 0とする符号化を
(2.4)＿＿＿＿＿＿＿＿(または「one-hot encoding」)と言います。

（選択肢）

(a) 「1-of-K 符号化法 (ﾜﾝｵﾌﾞｹｰﾌｺﾞｳｶﾎｳ、One-of-K encoding)」
(b) 「ﾃﾞｰﾀｸﾘｰﾆﾝｸﾞ (data cleaning)」
(c) 「ﾊﾌﾏﾝﾌﾞﾛｯｸ符号化法 (Huffman block coding method)」
(d) 「欠損値 (ｹｯｿﾝﾁ、missing value)」
(e) 「標準化 (ﾋｮｳｼﾞｭﾝｶ, standardized)」

（回答）

(2.1)	
(2.2)	
(2.3)	
(2.4)	

(3) 活性化関数

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・あるﾕﾆｯﾄに着目した時、そのﾕﾆｯﾄへの入力信号の重み付き線形和にﾊﾞｲｱｽを加えたもの(下図で重み付き入力 z)を、
どのように活性化して次のﾕﾆｯﾄに渡すかを、(3.1)＿＿＿＿＿＿＿＿で指定します。
「活性化関数」は「出力関数 (ｼｭﾂﾘｮｸｶﾝｽｳ、Output function)」とも言います。

・活性化関数の一つ(3.2)＿＿＿＿＿＿＿＿は、k個の入力値 Xi (i=1～k)の順序関係を保ちながら、
Xi を確率としての値 Yi (i=1～k、各値は0～1の範囲で総和が1になる) に変換する関数で、
k個のｸﾗｽへ分類する際の所属の確率を示すのによく用いられます。

k 個のｸﾗｽへ分類する時のｿﾌﾄﾏｯｸｽ関数

$$Yi = \exp(X_i) \;/\; \Sigma^{k}_{j=1} \exp(X_j) \qquad (i=1\sim k)$$

・活性化関数の一つ(3.3)＿＿＿＿＿＿＿＿は「ｿﾌﾄﾏｯｸｽ関数」で分類するｸﾗｽ数を2個に限定したものに相当します。

2個のｸﾗｽへ分類する時のｿﾌﾄﾏｯｸｽ関数

$$Y_1 = \exp(X_1) \;/\; \{\exp(X_1)+\exp(X_2)\} = 1 \;/\; \{1+\exp(X_2-X_1)\} = 1 \;/\; \{1+\exp(-Z)\} \qquad (Z=X_1-X_2) \quad =ｼｸﾞﾓｲﾄﾞ関数$$

$$Y_2 = \exp(X_2) \;/\; \{\exp(X_1)+\exp(X_2)\} = 1 - Y_1$$

・活性化関数の一つ(3.4)＿＿＿＿＿＿＿＿は計算が容易で(正の領域では微分が 1)、
「sigmoid関数」で問題となる学習時の「誤差逆伝播法」での(3.5)＿＿＿＿＿＿＿＿もなく、
一般的に使用されています。

(選択肢)

(a) 「ReLU (正規化線形関数)」
(b) 「ｼｸﾞﾓｲﾄﾞ関数 (sigmoid function)」
(c) 「ｿﾌﾄﾏｯｸｽ関数 (softmax function)」
(d) 「過学習 (ｶｶﾞｸｼｭｳ、over fitting)」
(e) 「活性化関数 (ｶｯｾｲｶｶﾝｽｳ、Activation function)」
(f) 「勾配消失問題 (ｺｳﾊﾞｲｼｮｳｼｯﾓﾝﾀﾞｲ、vanishing gradient problem)」

(回答)

(3.1)	
(3.2)	
(3.3)	
(3.4)	
(3.5)	

(4) 損失関数・誤差関数・ｺｽﾄ関数・目的関数

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・説明変数と目的変数の関係を良く説明できるﾓﾃﾞﾙが「良いﾓﾃﾞﾙ」であり、
「ﾓﾃﾞﾙの出力結果(予測)」と「既知の結果ﾃﾞｰﾀ」との差(「誤差 (ｺﾞｻ、error)」) が最小になるように、
ﾓﾃﾞﾙの「ﾊﾟﾗﾒｰﾀ」を最適化するのが「学習」です。

・「学習」を通して最小化したいのはこの誤差であり、誤差を表す関数のことを、
「誤差関数 (ｺﾞｻｶﾝｽｳ、error function)」、又は(4.1)＿＿＿＿＿＿、又は
「ｺｽﾄ関数 (ｺｽﾄｶﾝｽｳ、cost function)」と言います。
これらは「最小化する」という目的があるので、(4.2)＿＿＿＿＿＿とも言います。

・統計学において、未知または既知のある確率分布の下で、確率変数の結果が出現する場合に、
確率変数の出現結果からみて、元々の確率分布を推測する時の「尤もらしさ (ﾓｯﾄﾓﾗｼｻ)」を表す数値を、
(4.3)＿＿＿＿＿＿または「尤度関数 (ﾕｳﾄﾞｶﾝｽｳ、likelihood function)」と言います。

・この「尤度」を最大化するように元々の確率分布を推測することを
(4.4)＿＿＿＿＿＿と言います。
「離散型確率分布」の場合は「確率質量関数」を、「連続型確率分布」の場合は「確率密度関数」を推定することになります。
「尤度」を最大化することは、ﾓﾃﾞﾙの学習時の評価指数である「損失関数」の値を最小化することになります。

・自然対数関数 log(x) が x について単調増加であり、関数値の大小関係が x の大小関係を保つという性質から、
「尤度」を最大にする x の値は、この対数をとった(4.5)＿＿＿＿＿＿の値も最大にします。

(選択肢)

(a) 「損失関数 (ｿﾝｼｯﾂｶﾝｽｳ、loss function)」
(b) 「目的関数 (ﾓｸﾃｷｶﾝｽｳ、objective function)」
(c) 「尤度 (ﾕｳﾄﾞ、likelihood)」
(d) 「最尤推定 (ｻｲﾕｳｽｲﾃｲ、maximum likelihood estimation、MLE)」
(f) 「対数尤度 (ﾀｲｽｳﾕｳﾄﾞ、Log likelihood)」

(回答)

(4.1)	
(4.2)	
(4.3)	
(4.4)	
(4.5)	

(5) 順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ(FFNN)、畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (CNN)

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・(5.1)＿＿＿＿＿＿＿＿は、以下の特徴を持つﾆｭｰﾗﾙﾈｯﾄﾜｰｸです。
(5.2)＿＿＿＿＿＿＿＿は、
FFNN の中間層（隠れ層）を深くしたものです。

(1) 隣接する層間のすべてのﾕﾆｯﾄが完全に接続されている(完全結合)
(2) 入力層から出力層に向けて一方通行で流れる
(3) 入力と出力の間に、１つ以上の隠れ層がある

・(5.3)＿＿＿＿＿＿＿＿は、階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸに、畳み込み構造を
入れ込んだﾓﾃﾞﾙで、人の視覚をﾓﾃﾞﾙにした構成になっており、画像認識分野を中心に幅広く利用されているﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞ（深層学習）
手法の一つです。以下のような特徴があります。
(5.4)＿＿＿＿＿＿＿＿は、
CNN の中間層（隠れ層）を深くしたものです。

(1) 複雑なﾊﾟﾀｰﾝ認識問題にも、簡潔なﾈｯﾄﾜｰｸで対応できます。
(2) 全体としてﾕﾆｯﾄ数が少なくて済むので、計算が容易です。

・上記何れのﾀｲﾌﾟのﾈｯﾄﾜｰｸも(5.5)＿＿＿＿＿＿＿＿で学習します。

（選択肢）

(a) 「誤差逆伝播法 (Backpropagation)」
(b) 「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Feed forward neural network、FFNN)」
(c) 「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Convolutional neural network、CNN)」
(d) 「深層順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Deep Feed forward neural network、DFF)」
(f) 「深層畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Deep Convolutional neural network、DCN)」

（回答）

(5.1)	
(5.2)	
(5.3)	
(5.4)	
(5.5)	

(6) 畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (CNN) を構成する層

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・「畳み込みﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (ﾀﾀﾐｺﾐﾆｭｰﾗﾙﾈｯﾄﾜｰｸ、Convolutional neural network、CNN)」は、
中間層(隠れ層)が、畳み込み層(ｺﾝﾎﾞﾘｭｰｼｮﾝ層)、正規化層、ﾌﾟｰﾘﾝｸﾞ層、全結合層で構成された複数の層からなります。
正規化層とﾌﾟｰﾘﾝｸﾞ層は必須ではありません。また、畳み込み層でﾌｨﾙﾀの「移動量 (ｲﾄﾞｳﾘｮｳ、ｽﾄﾗｲﾄﾞ、stride)」を調整することにより
情報縮約を行って、ﾌﾟｰﾘﾝｸﾞ層を代行することもあります。

・各層は以下のようなものです。

層名	説明
(6.1)	入力側からの信号に対してﾌｨﾙﾀ処理を行い、「特徴ﾏｯﾌﾟ (ﾄｸﾁｮｳﾏｯﾌﾟ, feature map)」を作成する層です。 ｺﾝﾎﾞﾘｭｰｼｮﾝ層ともいいます。ﾌｨﾙﾀは「ｶｰﾈﾙ (kernel)」「ｵﾍﾟﾚｰﾀ」「ﾏｽｸ」とも言います。
(6.2)	畳み込み層で作成された「特徴ﾏｯﾌﾟ」に対し情報縮約操作を行う層。 ﾌﾟｰﾘﾝｸﾞには、以下のようなものがあります： ・最大値ﾌﾟｰﾘﾝｸﾞ (Max Pooling) 着目領域での最大値で縮約する ・平均値ﾌﾟｰﾘﾝｸﾞ (Average Pooling) 着目領域での平均値で縮約する ・Lpﾌﾟｰﾘﾝｸﾞ (Lp Pooling) 着目領域での中央値を強調した値で縮約する
(6.3)	隣接する層の各ﾕﾆｯﾄ同士が、全ての組合せで結合することを全結合と言います。 通常、出力層と前段の数層を全結合とし、これを全結合層としてﾓﾃﾞﾙ化します。
(6.4)	各ｻﾝﾌﾟﾙの平均と分散をそれぞれ０と１になるように、入力側からの信号を変換する層。 これで、入力ﾃﾞｰﾀ間のばらつきを補正します。
(6.5)	前段の層から出てきた多次元のﾃﾞｰﾀを一次元に変換する層。 次段の層の各ﾕﾆｯﾄと全結合する準備などの為に入れます。

(選択肢)

(a) 「Flatten層 (ﾌﾗｯﾄﾝｿｳ、Flatten Layer)」
(b) 「ﾌﾟｰﾘﾝｸﾞ層 (ﾌﾟｰﾘﾝｸﾞｿｳ、Pooling Layer)」
(c) 「畳み込み層 (ﾀﾀﾐｺﾐｿｳ、Convolution Layer)」
(d) 「正規化層 (ｾｲｷｶｿｳ、Normalization Layer)」
(f) 「全結合層 (ｾﾞﾝｹﾂｺﾞｳｿｳ、Fully Connected Layer)」

(回答)

(6.1)	
(6.2)	
(6.3)	
(6.4)	
(6.5)	

(7) ﾓﾃﾞﾙの学習

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

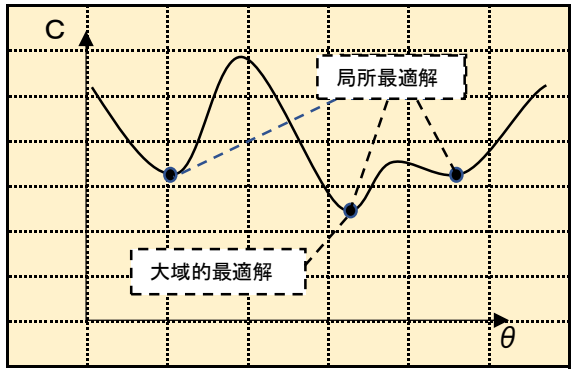
・教師あり学習では、説明変数と目的変数の関係を説明するﾓﾃﾞﾙを作成し、
「ﾓﾃﾞﾙの出力結果(予測)」と「既知の結果ﾃﾞｰﾀ」との差が最小化する(ｺｽﾄ関数(損失関数)の値が最小になる)ように、
ﾓﾃﾞﾙの「ﾊﾟﾗﾒｰﾀ」(各ﾕﾆｯﾄの重みW_i／しきい値W_0／ﾌｨﾙﾀの各値$W^{(f)}{}_{ij}$などの調整可能な値)を最適化します。これが「学習」です。

・最適化対象のﾊﾟﾗﾒｰﾀを一般化してθで表します。ﾊﾟﾗﾒｰﾀθの関数であるｺｽﾄ関数Ｃ(θ)が与えられた時、
これを最小にするθ^*を求める最適化を、(7.1)＿＿＿＿＿＿＿＿で行います。

・「勾配降下法」は、現ｽﾃｯﾌﾟ(t)のﾊﾟﾗﾒｰﾀ$\theta^{(t)}$における勾配と学習率η_tから、
次ｽﾃｯﾌﾟ(t+1)のﾊﾟﾗﾒｰﾀ$\theta^{(t+1)}$を次式により更新して、ｺｽﾄ関数Ｃ(θ)が最小になるθ^*を逐次近似により求める方法です。

勾配降下法によるﾊﾟﾗﾒｰﾀの最適化	
$\theta^{(t+1)} = \theta^{(t)} - \eta_t \times \partial C(\theta) / \partial \theta \mid_{\theta=\theta(t)}$	Ｃ(θ)：ｺｽﾄ関数 (ﾊﾟﾗﾒｰﾀθの関数) $\theta^{(t)}$：ｽﾃｯﾌﾟ t におけるﾊﾟﾗﾒｰﾀθの値 η_t：ｽﾃｯﾌﾟ t における学習率ηの値 (>0) $\partial C(\theta) / \partial \theta \mid_{\theta=\theta(t)}$：$\theta$(t)におけるｺｽﾄ関数の勾配

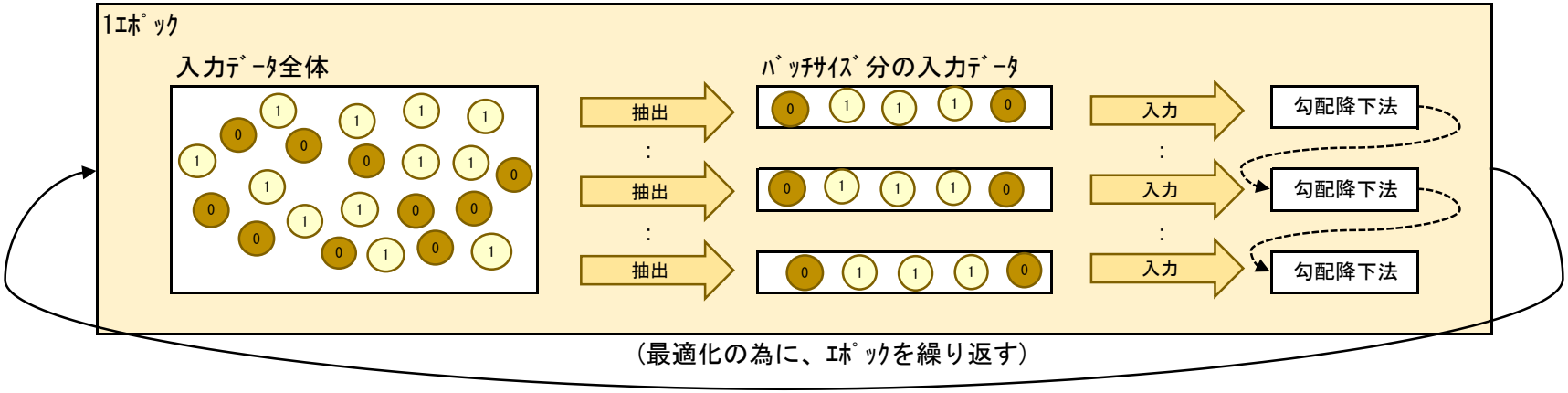
・勾配降下法によるﾊﾟﾗﾒｰﾀの最適化の式では、(7.2)＿＿＿＿＿＿＿＿と呼ばれるﾊﾟﾗﾒｰﾀη(ｲｰﾀ)があります。
これは、勾配降下法で、現ｽﾃｯﾌﾟ(t)のﾊﾟﾗﾒｰﾀθ(t)におけるｺｽﾄ関数Ｃ(θ)の勾配から、次ｽﾃｯﾌﾟ(t+1)のﾊﾟﾗﾒｰﾀθ(t+1)を計算する時に、
勾配をどの程度の割合で変化分として反映させるか？という割合を示しています。
「学習率」は「学習係数」とも言います。



・「学習率」を大きくすると、最適解を通り過ぎて見落としてしまう可能性が高くなり、
「学習率」を小さくすると、最適解に辿り着くのに時間がかかったり、
極小値が最小値とは限らないのに、最小値でない極小値
(これを「局所最適解 (ｷｮｸｼｮｻｲﾃｷｶｲ、local optimal solution)」と言います)
にはまってしまうと抜け出られない場合があるという欠点があります。
これを(7.3)＿＿＿＿＿＿＿＿と言います。
尚、全領域にわたり最小値となる解を「大域的最適解 (ﾀｲｲｷﾃｷｻｲﾃｷｶｲ、global optimal soution)」
と言います。

・入力ﾃﾞｰﾀ全てに対する誤差関数の勾配を、更新の１ｽﾃｯﾌﾟごとに計算する場合、ﾃﾞｰﾀが大きいとその計算に時間がかかってしまいます。
そのような場合には、ﾃﾞｰﾀの一部で誤差関数の勾配を計算する、
(7.4)＿＿＿＿＿＿＿＿という方法が使われます。
「確率的勾配降下法」では入力ﾃﾞｰﾀ全体の一部で計算しますが、計算1回分に使用するﾃﾞｰﾀの個数を(7.5)＿＿＿＿＿＿＿＿と言います。
「ﾊﾞｯﾁｻｲｽﾞ」分の入力ﾃﾞｰﾀを使用した計算で、ﾊﾟﾗﾒｰﾀが1回更新されることになります。

・入力ﾃﾞｰﾀ全てを使いﾊﾟﾗﾒｰﾀ更新を行うとき、この更新の単位を(7.6)＿＿＿＿＿＿＿＿で表します。
「確率的勾配法」では、1「ｴﾎﾟｯｸ」のﾊﾟﾗﾒｰﾀ更新を行うのに「ﾊﾞｯﾁｻｲｽﾞ」分の入力ﾃﾞｰﾀを使用した更新を何度か繰り返します。
そしてﾊﾟﾗﾒｰﾀが最適化されるまで「ｴﾎﾟｯｸ」を何度か繰り返します(ﾓﾃﾞﾙによっては収束しない場合もあります)。



（選択肢）

(a)　「勾配降下法 (ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method)」
(b)　「学習率 (ｶﾞｸｼｭｳﾘﾂ、learning rate)」
(c)　「局所解問題 (ｷｮｸｼｮｶｲﾓﾝﾀﾞｲ、local minumum problem)」
(d)　「確率的勾配降下法 (ｶｸﾘﾂﾃｷｺｳﾊﾞｲｺｳｶﾎｳ、Stochastic gradient descent method, SGD)」
(e)　「ﾊﾞｯﾁｻｲｽﾞ(batch size)」
(f)　「ｴﾎﾟｯｸ (epochs)」

（回答）

(7.1)	
(7.2)	
(7.3)	
(7.4)	
(7.5)	
(7.6)	

(8) 「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」の誤差逆伝搬法

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

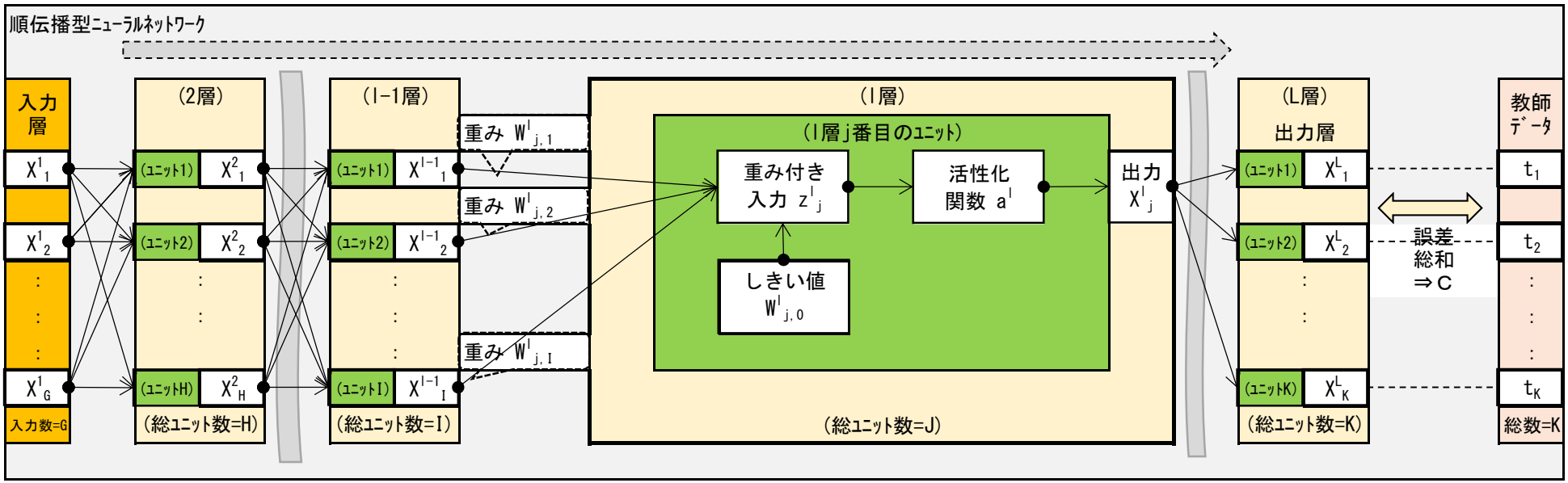
・「階層型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ」の学習方法として(8.1)＿＿＿＿＿＿＿＿＿＿が有名です。
手法そのものは以前からありましたが、1986年に「backwards propagation of errors（後方への誤差伝播)」の略から
ﾃﾞﾋﾞｯﾄﾞ･ﾗﾒﾙﾊｰﾄらによって命名されたものです。

・「誤差逆伝播法」では、出力層での誤差(階層型ﾈｯﾄﾜｰｸの出力で生じる教師ﾃﾞｰﾀとの差)を使って、
ﾓﾃﾞﾙの出力の計算方向が中間層⇒出力層の方向なのとは逆向きに、
出力層⇒中間層の方向へと、勾配降下法を用いてﾊﾟﾗﾒｰﾀを更新してゆきます(下図)。

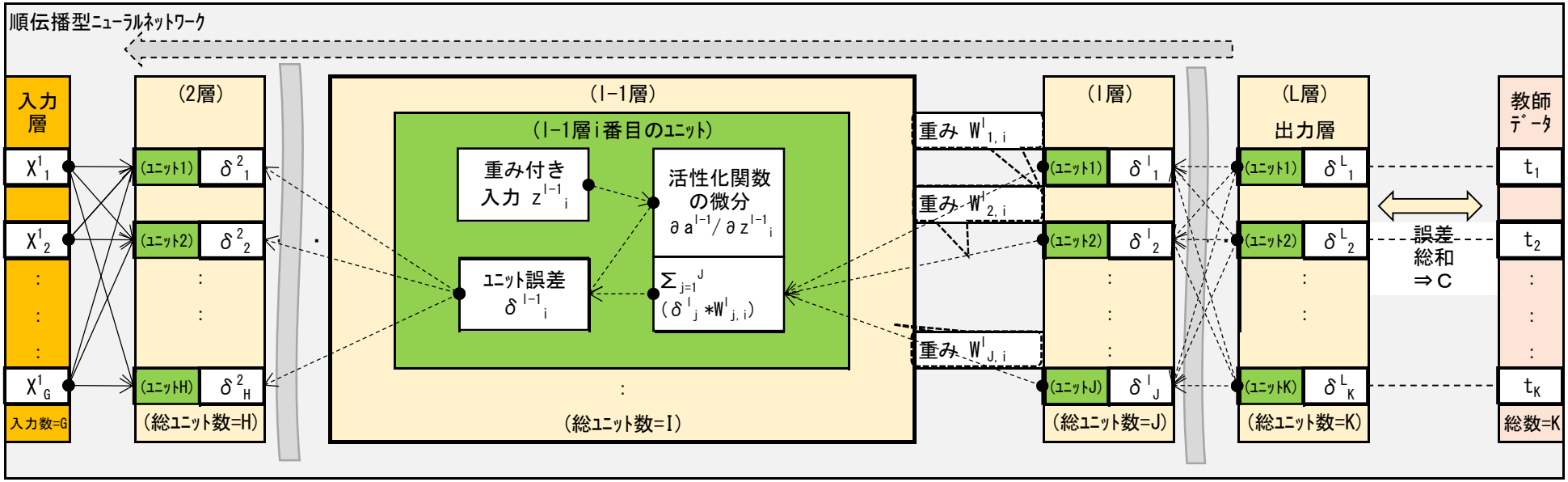
・下図では、以下のような記法を用います。

- L ：層の総数（入力層を第1層、出力層を第L層とします）
- J ：l層のﾕﾆｯﾄ数
- l ：層番号（l=2～L、入力層は層の厚さとしては数えませんが、説明上第1層とします）
- j ：l層でのﾕﾆｯﾄ番号（j=1～J）
- i ：l-1層でのﾕﾆｯﾄ番号（i=1～I）
- I ：l-1層のﾕﾆｯﾄ数
- X^{l-1}_i ：l-1層i番目のﾕﾆｯﾄの出力値（l層への入力信号）
- $W^l_{j,i}$ ：l層j番目のﾕﾆｯﾄについて、l-1層i番目のﾕﾆｯﾄからの入力信号の重み
- $W^l_{j,0}$ ：l層j番目のﾕﾆｯﾄについて、l-1層からの入力信号に付加するﾊﾞｲｱｽ
- z^l_j ：l層j番目のﾕﾆｯﾄの重み付き入力（ $z^l_j = \sum_{i=0}^{I} W^l_{j,i} * X^{l-1}_i$ （但し $X^{l-1}_0=1$ でﾊﾞｲｱｽ用））
- a^l ：l層の活性化関数（activation function)
- X^l_j ：l層j番目のﾕﾆｯﾄの出力値（ $X^l_j=a^l(z^l_j)$ ）
- C ：ｺｽﾄ関数（平均二乗誤差(MSE)の場合 $C=(1/K)\sum_{k=1}^{K}(t^L_k - X^L_k)^2$ （K：は出力層のﾕﾆｯﾄ数））
- δ^{l-1}_i ：l-1層i番目のﾕﾆｯﾄの誤差
- δ^l_j ：l層j番目のﾕﾆｯﾄの誤差

（出力計算時の流れ）



（誤差逆伝搬時の流れ）



・漸化式の場合は、初項と、第n+1項を第n項で表現する関係式を与えると、全ての項が計算できます。
これに対し、逆漸化式の場合は、末項と第n-1項を第n項で表現する関係式を与えると、全ての項が計算できます。

・出力層のﾕﾆｯﾄの誤差で末項「δ^{L}_{*}」を与え、
隣接する層間のﾕﾆｯﾄの誤差についての逆漸化式で、
第l-1層のﾕﾆｯﾄの誤差「δ^{l-1}_{*}」を、第l層のﾕﾆｯﾄの誤差「δ^{l}_{*}」で表現する関係式を与えることにより、
ﾕﾆｯﾄの誤差δは、全ての中間層のﾕﾆｯﾄについて計算することができます。

出力層j番目のﾕﾆｯﾄの誤差 δ^{L}_{j}

$$\delta^{L}_{j} = (\partial C/\partial X^{L}_{j})(\partial X^{L}_{j}/\partial z^{L}_{j})$$

l層j番目のﾕﾆｯﾄと、l-1層の各ﾕﾆｯﾄとの間に成立する逆漸化式

$$\delta^{l-1}_{i} = \{\sum_{j=1}^{J} \delta^{l}_{j} * W^{l}_{j,i}\} * \partial a^{l-1}/\partial z^{l-1}_{i}$$

・これを元に、重みに関するｺｽﾄ関数の偏微分とﾊﾞｲｱｽに関するｺｽﾄ関数の偏微分を計算して、
更に(8.2)を用いて、重み、ﾊﾞｲｱｽの値を更新します。

$\partial C/\partial W^{l}_{j,i} = \delta^{l}_{j} X^{l-1}_{i}$ ：l層j番目のﾕﾆｯﾄについて、l-1層i番目のﾕﾆｯﾄからの入力信号の重みに関するｺｽﾄ関数の偏微分

$\partial C/\partial W^{l}_{j,0} = \delta^{l}_{j}$ ：l層j番目のﾕﾆｯﾄについてのﾊﾞｲｱｽに関するｺｽﾄ関数の偏微分

（選択肢）

(a) 「誤差逆伝播法 (ｺﾞｻｷﾞｬｸﾃﾞﾝﾊﾟﾎｳ、Backpropagation method)」

(b) 「勾配降下法 (ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method)」

（回答）

(8.1)	
(8.2)	

(9) ﾓﾃﾞﾙの学習の検証

以下の空欄に最もあてはまる語句を選択肢から選び、その記号を回答欄に記入してください。

・ﾓﾃﾞﾙを作成し、学習を通してﾓﾃﾞﾙのﾊﾟﾗﾒｰﾀの最適化を行います。
学習時のﾃﾞｰﾀでは、ｺｽﾄ関数Ｃが最小になるが、学習時以外のﾃﾞｰﾀでは、誤差が大きく出てしまうことがあります。
これは、学習時のﾃﾞｰﾀに特化した学習をしてしまって、汎用になっていないということであり、
これを(9.1)＿＿＿＿＿＿＿＿と言います。

・「過学習」の対策として、入力にﾉｲｽﾞを与えたり、学習時に正則化を適用したり、様々な方法が提案されてきましたが、
(9.2)＿＿＿＿＿＿＿＿という方法が有効であることが知られています。

・「ﾄﾞﾛｯﾌﾟｱｳﾄ」は、学習時に入力層や中間層のﾕﾆｯﾄを、各層毎に一定の割合α（0≦α＜1）でﾕﾆｯﾄを選出し、
それ以外は無効な状態で学習を行います（選出されなかったﾕﾆｯﾄはないものとして学習を行います）。
これをﾐﾆﾊﾞｯﾁ毎にﾕﾆｯﾄを選出しなおして、学習を繰り返します。

・学習後に予測をする場合は、全てのﾕﾆｯﾄを使用しますが、
学習時はαの割合のﾕﾆｯﾄしか存在しなかったので、
予測時にはﾄﾞﾛｯﾌﾟｱｳﾄを施した層の出力の重みをα倍にして出力が大きくなることを防ぎます。

（選択肢）

(a) 「過学習（ｶｶﾞｸｼｭｳ、over fitting)」
(b) 「ﾄﾞﾛｯﾌﾟｱｳﾄ（Dropout)」
(c) 「ﾄﾞﾛｯﾌﾟｲﾝ（Dropin)」

（回答）

(9.1)	
(9.2)	

以上。

(11) 確認問題回答用紙

提出者　：

提出日　：　　年　　月　　日

回答　　（全 37 問）

No.	回答
(1.1)	
(1.2)	
(1.3)	
(2.1)	
(2.2)	
(2.3)	
(2.4)	
(3.1)	
(3.2)	
(3.3)	
(3.4)	
(3.5)	
(4.1)	
(4.2)	
(4.3)	
(4.4)	
(4.5)	

No.	回答
(5.1)	
(5.2)	
(5.3)	
(5.4)	
(5.5)	
(6.1)	
(6.2)	
(6.3)	
(6.4)	
(6.5)	
(7.1)	
(7.2)	
(7.3)	
(7.4)	
(7.5)	
(7.6)	

No.	回答
(8.1)	
(8.2)	
(9.1)	
(9.2)	

（※黄色の枠　のみ記入をお願いします）

※ ご意見・ご要望など有りましたら、下欄に記してください。