機械学習ゼミ
(深層学習)

2018/07/06　金子法子，小川晃平，河井智弘

取扱内容

(担当：金子　pp.3-33)
第2章　機械学習と深層学習
第3章　ニューラルネットワーク

(担当：小川　pp.34-79)
第4章　勾配降下法による学習
第5章　深層学習の正則化
第6章　誤差逆伝播法

(担当：河井　pp.80-109)
第7章　畳み込みニューラルネット
第8章　再帰型ニューラルネット

準拠：これならわかる深層学習入門（瀧雅人）

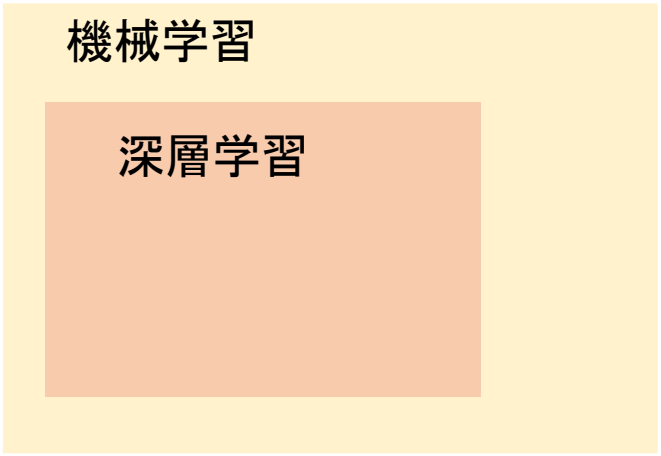
機械学習ゼミ
2章：機械学習と深層学習

2018.07.06

福田研究室M3

金子法子

2.1 なぜ深層学習か？



コンピュータプログラムに知的な作業をさせたい！
特に明示的なプログラムを書かなくていい！

機械学習	人間がこなすような様々な学習や知的作業を計算機に実行させるための手法やその研究 E（経験），T（タスク），P（パフォーマンス）からなる（T.M.ミッチェル）
学習	TについてPで測られた実行能力がEを通じて向上すること
深層学習	機械学習のうち一歩進んだ技術，生物系の神経系の挙動を模したニューラルネット計算を行う
AI	人と同等レベル以上のパフォーマンスのできる深層学習のこと（≒ 深層学習）

2.3 統計入門

機械学習の基本は統計学！

2.3.1 標本と推定

2.3.2 点推定

2.3.3 最尤推定

基礎ゼミなどで扱った内容が多いので割愛します！
忘れた人は読んでみてください.

2.4 機械学習の基礎

→この章に入る前に，基本的な用語について学びます．

機械学習：

データ集合を学習アルゴリズムで処理することでコンピュータプログラムに学習させ，それによってプログラムがタスクをよりうまくこなせるようになることを目標にする

◆ 機械学習において特殊な言い方をする用語

データ	訓練データ（training data），訓練サンプル，観測
データ集合	訓練集合

◆ 機械学習で使われる用語

学習機械（learning machine） アーキテクチャ（architecture）	学習を担うプログラムのこと （Ex. フィッティング関数）
学習アルゴリズム	学習プロセスを決めるアルゴリズム
コスト関数，損失関数， 目的関数，誤差関数	パフォーマンスの良さを測る尺度

◆ 機械学習で行われる主なタスク

クラス分類	データをいくつかのグループに分けること
回帰	データから対応するデータの値を予測すること

2.4 機械学習の基礎

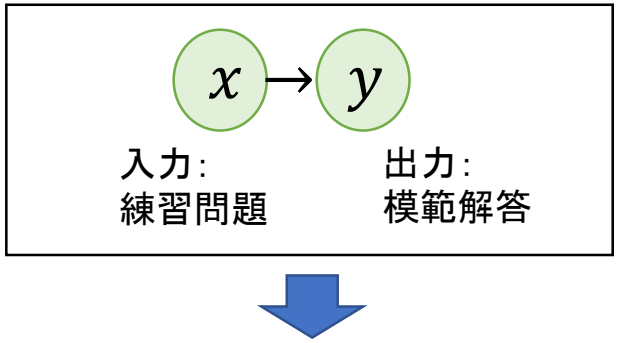
2.4.1 教師あり学習←深層学習の基礎となる考え方

赤字：統計学的な親しみやすい言い方で言い換えます

教師あり学習：

入力xと出力yが対になっており，その関係性から，新しいxに対してもyを予測できるようにすること

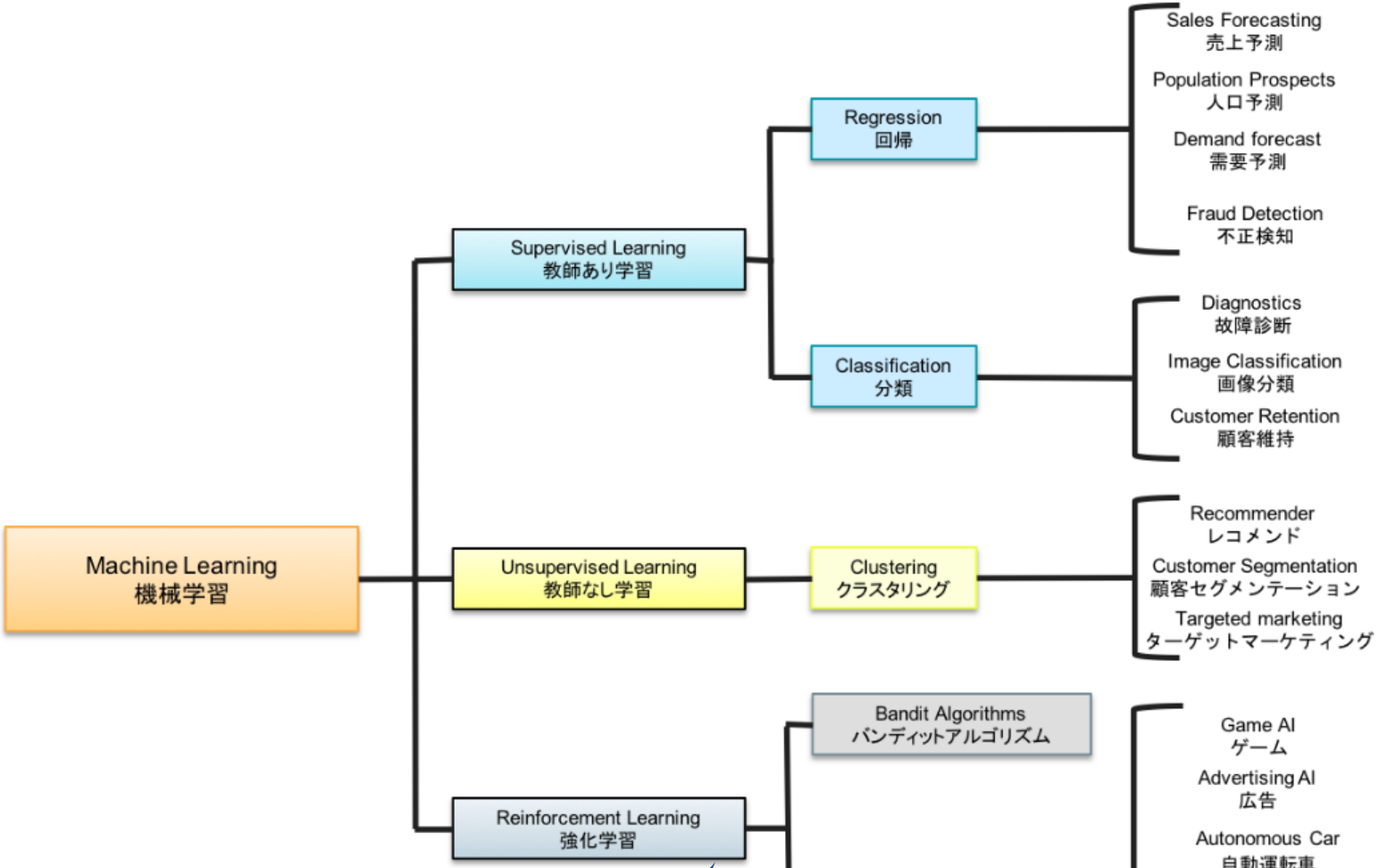
説明変数xと被説明変数yがついになっており，その関係性から関数モデルをつくり，データからパラメータ推定を行うこと



解き方を学んでみたことのない問題を解けるようにする

この次のセクションから統計の基礎的な話になるので，あまりに基礎的なところはとばします□

2.4 機械学習の基礎



正解を与える代わりに将来の価値を最大化することを学習するモデル
(Ex)Alpha碁

2.4 機械学習の基礎

2.4.1 (1)(2) 質的変数と量的変数について（略）

2.4.2 最小二乗法による線形回帰（概ね略）

誤差関数（平均事情誤差）の最小化

$$\boldsymbol{w}^* = argmin_w E_D(\boldsymbol{w}) = argmin_w \frac{1}{N}(\hat{y}(\boldsymbol{x_n}; \boldsymbol{w}) - y_n)^2$$

訓練誤差：手持ちのデータのサンプル平均で作った誤差
汎化誤差：訓練データだけなく任意の可能なデータに対して測られた誤差

→理想は汎化誤差だが，機械学習を使えば訓練誤差を汎化誤差に近づけられる！

2.4.3 線形回帰の確率的アプローチ

説明変数が与えられたときに
目標変数がどのような値をとりやすいか？

回帰に対する関数的アプローチ（最小二乗法）⇔確率的なアプローチ

確率的アプローチ：
説明変数$\boldsymbol{x}$,目標変数yも確率変数であり，推定量$\hat{y}$は条件付分布$P(y|x)$のモデル化で得られる

2.4 機械学習の基礎

2.4.3 線形回帰の確率的アプローチ

誤差関数も確率変数となる

誤差関数 (期待誤差，期待損失)

$$E(\hat{y}(\boldsymbol{x}), y) = (\hat{y}(\boldsymbol{x}) - y)^2$$

パフォーマンスを測る数値として，期待値を用いる

期待値

取りうるすべての離散値(x, y)に対して和をとる

$$E_{P_{data}}[E(\hat{y}(\boldsymbol{x}), y)] = \sum_x \sum_y (\hat{y}(\boldsymbol{x}) - y)^2 P_{data}(\boldsymbol{x}, y)$$

$(\boldsymbol{x}, y)$というデータ点が生成される確率

期待誤差を最小化する推定量$\hat{y}$は，
「とある説明変数の実現値xに対してだけ $\hat{y}(\boldsymbol{x}) \to \hat{y}(\boldsymbol{x}) + \delta\hat{y}$ としても期待誤差は変化しない」ので

$$0 = \delta E_{P_{data}}[E(\hat{y}(\boldsymbol{x}), y)]\Big|_y = \sum_y (\hat{y}(\boldsymbol{x}) - y) P_{data}(\boldsymbol{x}, y)\Big|_{\hat{y}^*}$$

頑張って変形すると，

条件付き確率で表せる！

$$\hat{y}^*(\boldsymbol{x}) = E_{Pdata(\boldsymbol{x}|y)}[y|\boldsymbol{x}]$$

→最適な$\hat{y}^*$＝$P_{data}(\boldsymbol{x}, y)$（生成分布）に関する条件付期待値である！

2.4 機械学習の基礎

2.4.3 線形回帰の確率的アプローチ

誤差関数も確率変数となる

$P_{data}(\boldsymbol{x}, y)$をデータから予測する必要がある

① 生成モデル：
データ背後にある生成メカニズムを直接，同時分布$P(\boldsymbol{x}, y)$としてモデル化
(a) モデル化した分布$P(x, y)$をデータから推定
(b) 事前確率$P(x) = \sum_y P(\boldsymbol{x}, y)$とベイズの公式から条件付分布$P(y, x)$を計算

$$P(y|\boldsymbol{x}) = \frac{P(x, y)}{P(x)}$$

② 識別モデル：
条件付分布を直接モデル化する
→あとで詳しくやります

2.4.4 最小二乗法と最尤法

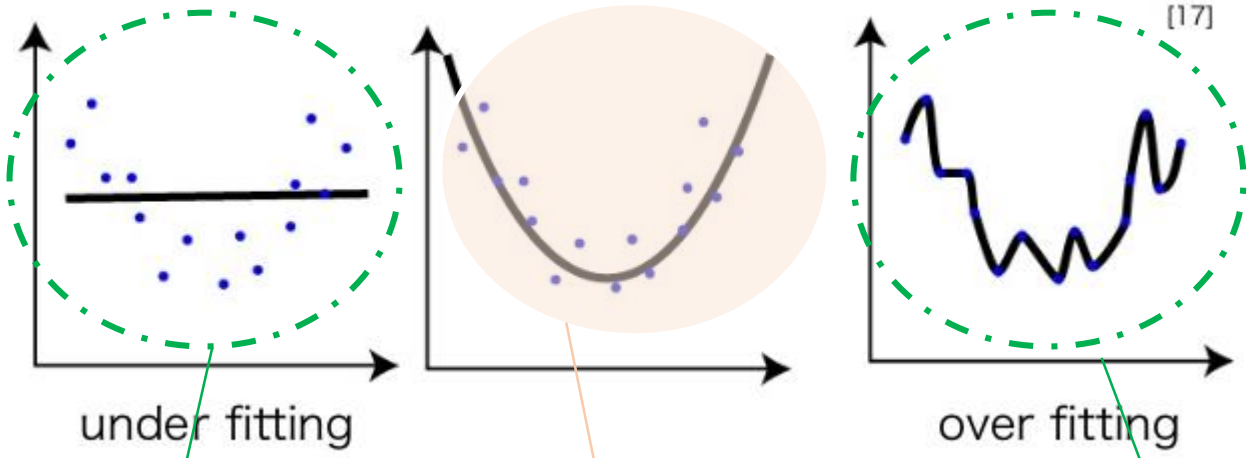
→省略！

2.4 機械学習の基礎

2.4.5 過適合と汎化

ハイパーパラメータ：モデル選択時に決める，学習されないパラメータ

多項式モデル $\hat{y} = \sum_{j=0}^{M} w_j x^j$ を決める際の説明変数の数の話



M
自由度

未学習
→訓練誤差
「勉強していない学生」

過適合，過剰適合，過学習
→汎化誤差
「丸暗記しかできていない学生」

望ましい状態
→見つける確固たる方法はなく，検証用データをモデルの良し悪しを見て判断する

2.4 機械学習の基礎

2.4.6 正則化

正則化：自由度を実質的に減らしてしまう手法のこと，
モデル変更なしで学習アルゴリズムを変更することで，自由度を減らす
誤差関数の変更をすることが多い

正則化パラメータλ：正則化の効果を表す

$$E_{new}(w) = E(w) + \lambda R(w)$$

(Ex) 重み減衰：強制的にパラメータを0に制限することで自由度を減らす手法

パラメータ数を減らすペナルティ項

$$E_{ed}(w) = E(w) + \lambda w^T w$$

重みベクトル$w^T w$のノルムの最小化
→ w^*では多くの成分がゼロになる

重み減衰をもちいて回帰したものをRidge回帰と呼ぶ

(Ex) LASSO回帰：不要な重みづけを0に近づける手法を用いて回帰したもの
least absolute shrinkage and selection operator

重みづけを減らすペナルティ項

$$E_{\mathrm{LASSO}}(w) = E(w) + \lambda \sum_i |w_i|$$

→5章で詳しくやるようです，ご紹介程度に

2.4 機械学習の基礎

2.4.3 クラス分類　←クラス分類の教師あり学習について

クラス分類：与えられた入力xをKこの種類(クラス)$C_1, C_2, \ldots, C_k$に分けること

① 二値分類：
分類先は2つ，C_1に属するとき$y = 1$，C_2に属するとき$y = 0$

② 多クラス分類：
分類先は複数，K個のクラスに対して，$y = 1,2, \ldots K$

ベクトル表記に直す(1-of-K 符号化に写像)

$$t(y) = (t(y)_1\ t(y)_2\ \cdots t(y)_k)^T$$

One-hot表現

例えば
$Ex.\ t(y = 1) = (1\ 0\ 0 \cdots 0)^T$

クロネッカーのデルタを用いて，

$$\delta_{i,j} = \begin{cases} 1\ (i = j) \\ 0\ (i \neq j) \end{cases}$$

$$t(y)_k = \delta_{y,k}$$

2.4 機械学習の基礎

2.4.8 クラス分類へのアプローチ

目標：与えられた入力に対して，離散的な目的変数を予測する

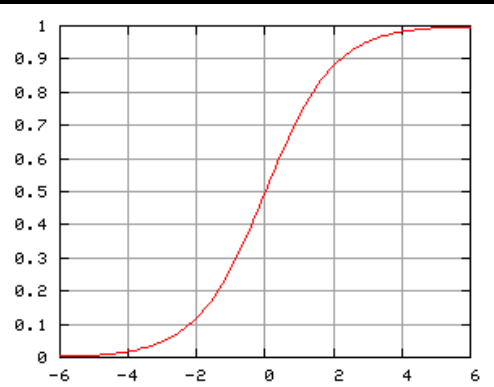
① 関数モデル：
入力と出力の間の関係を関数としてモデル化する手法
3章のニューラルネットもこれ

② 生成モデル：
データに潜むランダム性を同時分布$P(x, y)$のモデル化で表現し，データにフィットさせる

③ 識別モデル
直接的に$P(C_k|x)$をモデル化して，データ学習させる
→以降識別モデルに関するお話です．

2.4 機械学習の基礎

2.4.9 ロジスティック回帰（2クラス分類）



標準シグモイド関数
（ロジスティック関数）

2クラス分類を考える→$P(C_1|x) + P(C_2|x) = 1$が成立

シグモイド関数を導入

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

条件付き分布

$$P(C_1|x) = \sigma(u)$$

$$u = log\frac{P(C_1|x)}{1-P(C_1|x)} = log\frac{P(C_1|x)}{P(C_2|x)}$$

対数オッズ比

$$e^u = P(C_1|x)/P(C_2|x)$$

オッズ比

$e^u \geq 1$で
C_1である確率≥ C_2である確率

オッズ比uが線形を仮定した確率モデル：ロジスティック回帰

パラメータをまとめて表したベクトル

$$u = \boldsymbol{w}^T\boldsymbol{x} + b$$

対数尤度最大化は以下のようにかけて，**交差エントロピー**と呼ばれる

深層学習でよく出てくる誤差関数

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n log P(C_1|\boldsymbol{x_n}) + (1-y_n)\log(1-P(C_1|\boldsymbol{x_n})))$$

2.4 機械学習の基礎

2.4.10 ソフトマックス回帰（多クラス分類）←ロジスティック回帰の一般化

多クラス分類→$P(y|\boldsymbol{x}) = \prod_{k=1}^{K}\left(P(C_k|\boldsymbol{x})\right)^{t(y)_k}$

マルチヌーイ分布
カテゴリカル分布

条件付き分布

$$P(C_K|\boldsymbol{x}) = \frac{1}{\sum_{k=1}^{K} e^{u_k}} \qquad u_k = log\frac{P(C_k|\boldsymbol{x})}{P(C_K|\boldsymbol{x})}$$

ソフトマックス関数
多変数関数のひとつ
$softmax_k(u_1, u_2, \ldots, u_k) = e^{u_k} / \sum_{k'=1}^{K} e^{u_{k'}}$

ここで$P(C_k|\boldsymbol{x})e^{u_k} = P(C_k|\boldsymbol{x})$より，各クラス分布は

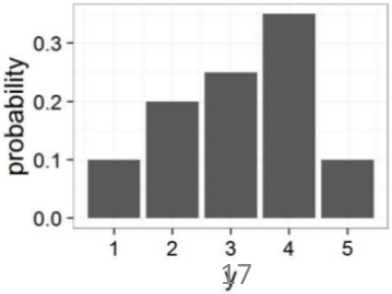
$$P(C_k|\boldsymbol{x}) = softmax_k(u_1, u_2, \ldots, u_k)$$

オッズ比uが線形を仮定した確率モデル：ソフトマックス回帰

$$u_k = w_k^T\boldsymbol{x} + b_k(k = 1,2,\ldots,K-1)$$

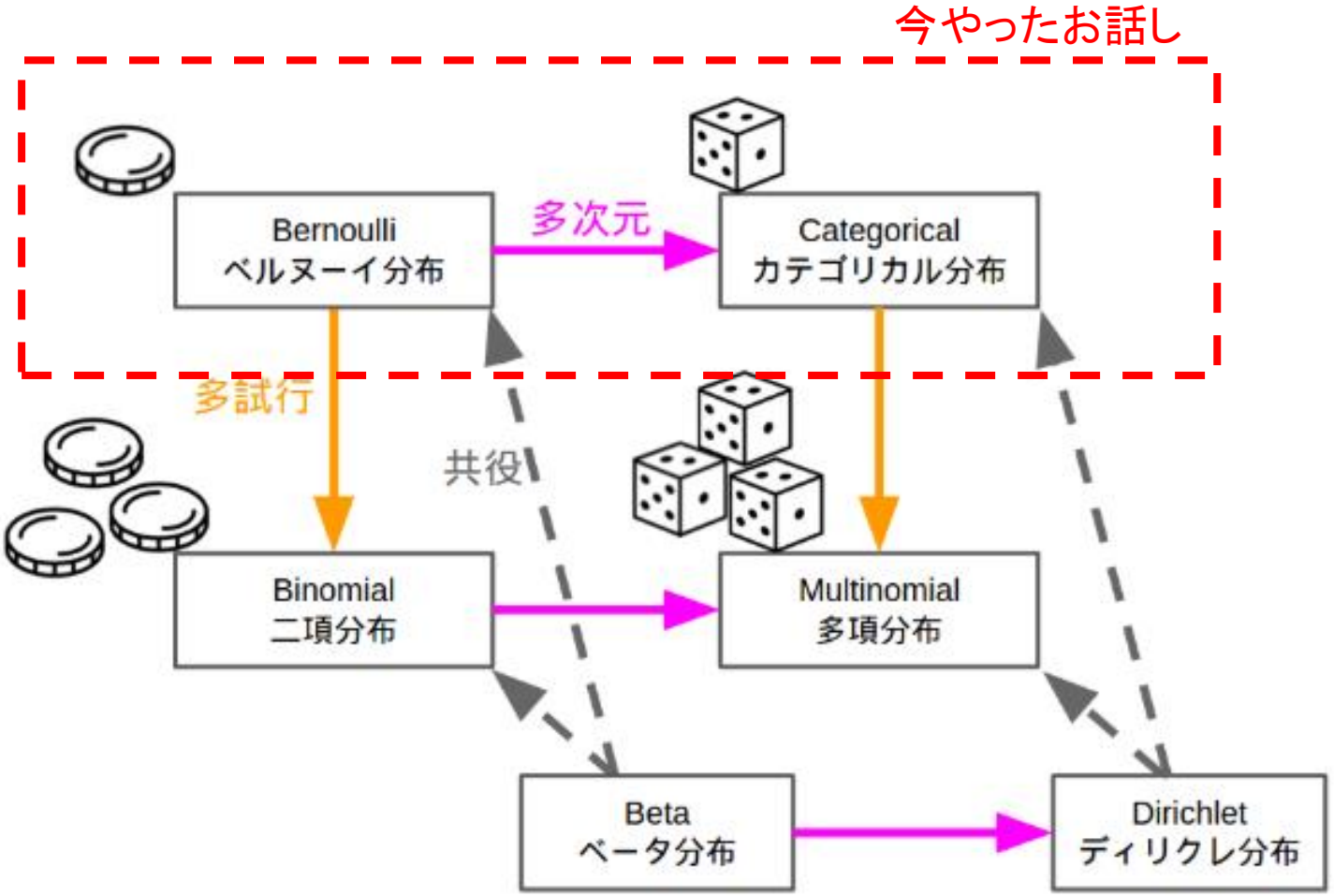
ロジスティック回帰と同様にして交差エントロピーは

$$E(w_1, w_2, \ldots, w_{K-1}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t(y_n)P(C_k|\boldsymbol{x_n})$$



2.4 機械学習の基礎

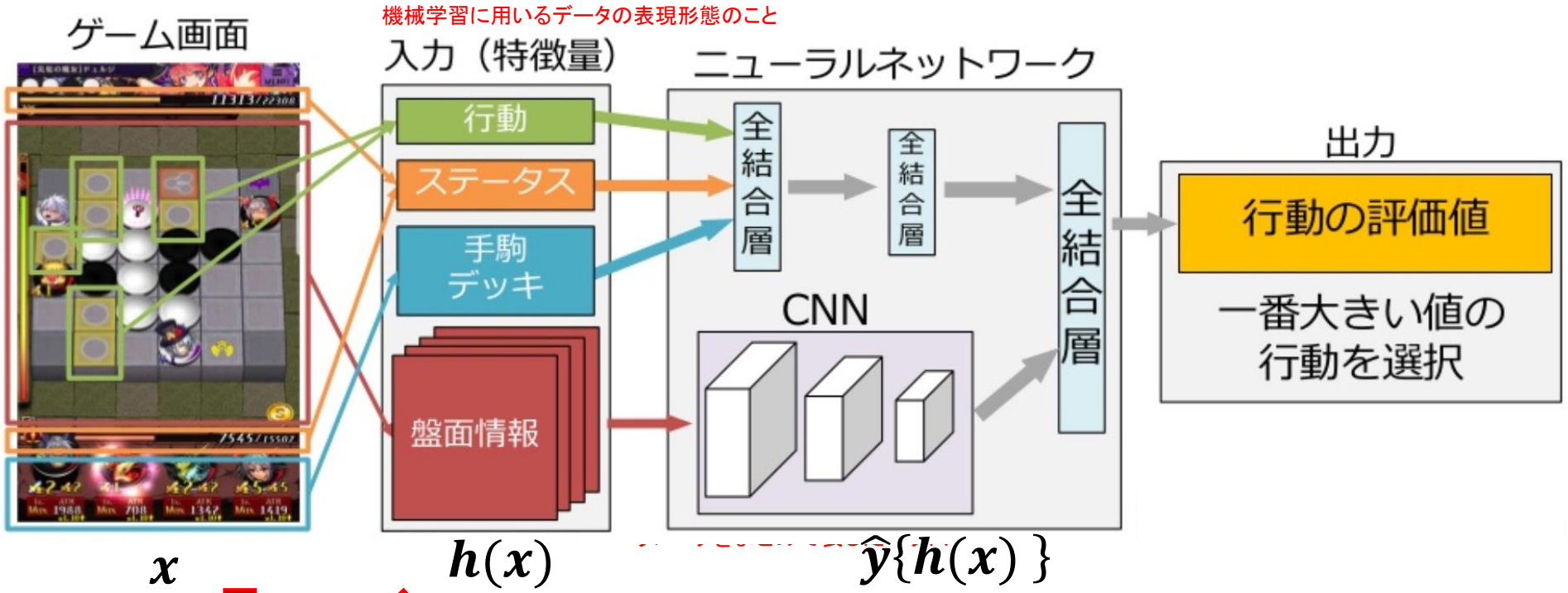
2.4.10 いろんな分布のお話(おまけ)



2.5 表現学習と深層学習の進展

2.5.1 表現学習

扱いやすい入力データばかりではない！



DeNA TechCon2018 ゲーム体験を支えるための強化学習

表現工学・特徴量工学

→表現学習(目的に応じた特徴量を学習を通じて獲得しようとするアプローチ)

$$x \rightarrow h(x) \rightarrow \hat{y}(h(x))$$

真相学習で得られる表現：真相表現

2.5 表現学習と深層学習の進展

2.5.2 深層学習の登場

2006 G.ヒントン (コンピュータ科学者)がボルツマンマシンの深層化

2012 G.ヒントンが画像認識コンペティションで深層学習を用いて圧勝

2012 Googleの猫□

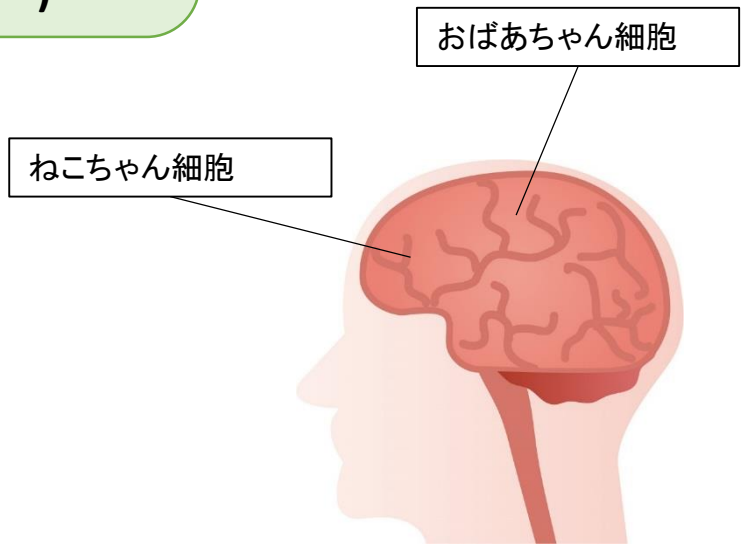
Youtubeからのランダム画像で人工ニューラルネットに教師なし学習→
ニューラルが獲得した猫の概念を画像化された

^ ^

(=ΦωΦ=)

http://zellij.hatenablog.com/entry/20130608/p1

ヒントンさん



機械学習ゼミ
3章：ニューラルネット

2018.07.06

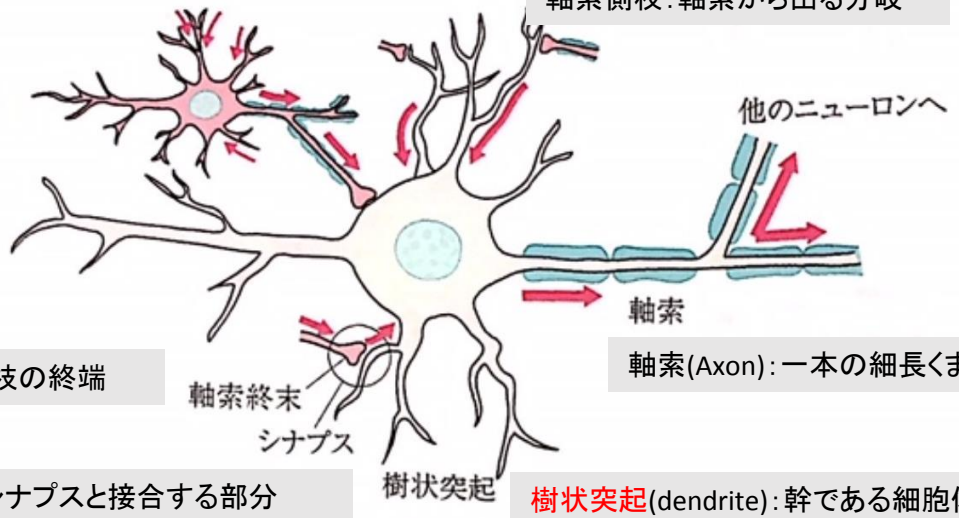
福田研究室M3

金子法子

3.1　神経細胞のネットワーク

→生物みたいな話からはじめます

軸索側枝：軸索から出る分岐



軸索終末：側枝の終端

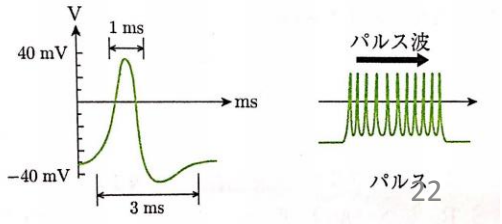
軸索(Axon)：一本の細長くまっすぐな突起

シナプス(synapse)：軸索端末が他のシナプスと接合する部分

樹状突起(dendrite)：幹である細胞体から伸びる

ニューロンとシナプス

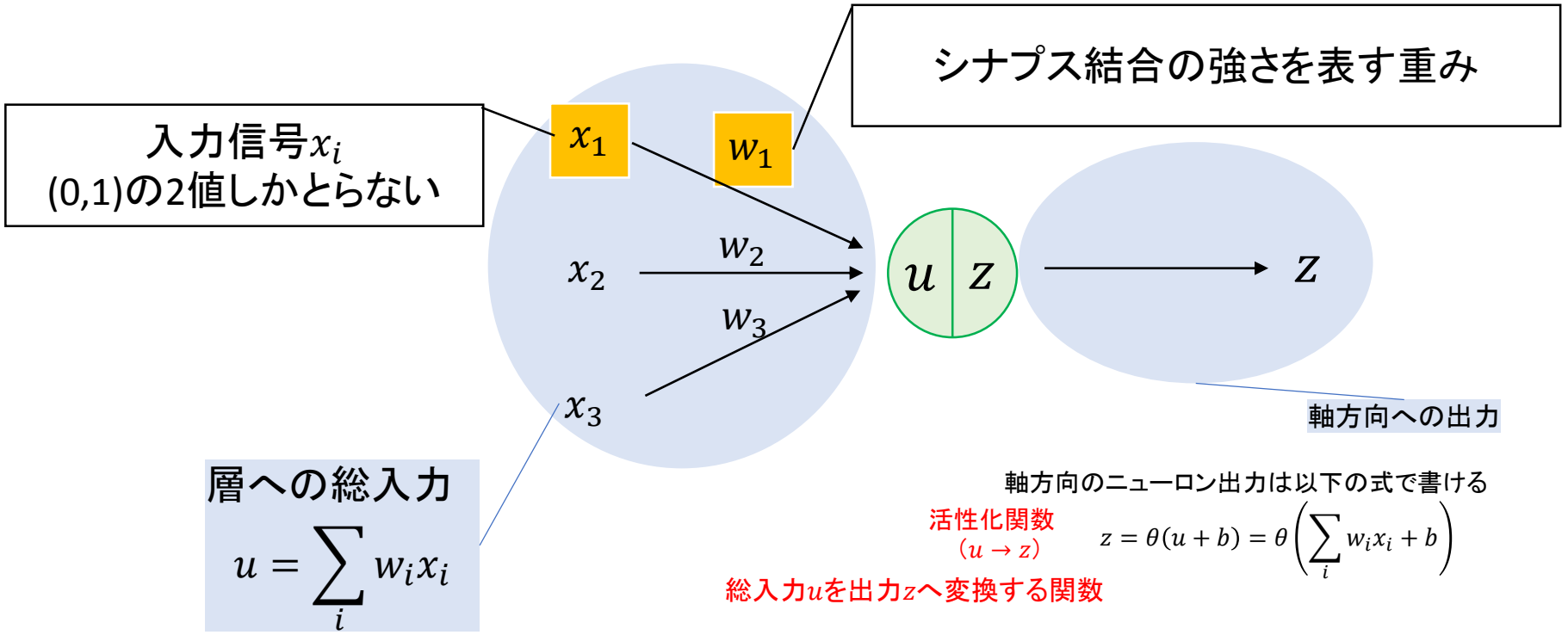
電気信号(パルス)を伝える電線のような役割



3.2 形式ニューロン

1943: W.マカロック（外科医），W.ピッツ(神童)が源流
形式ニューロンと人工ニューロンを定義
「ニューロンの活動も数理論理学的な手法でモデル化できる」

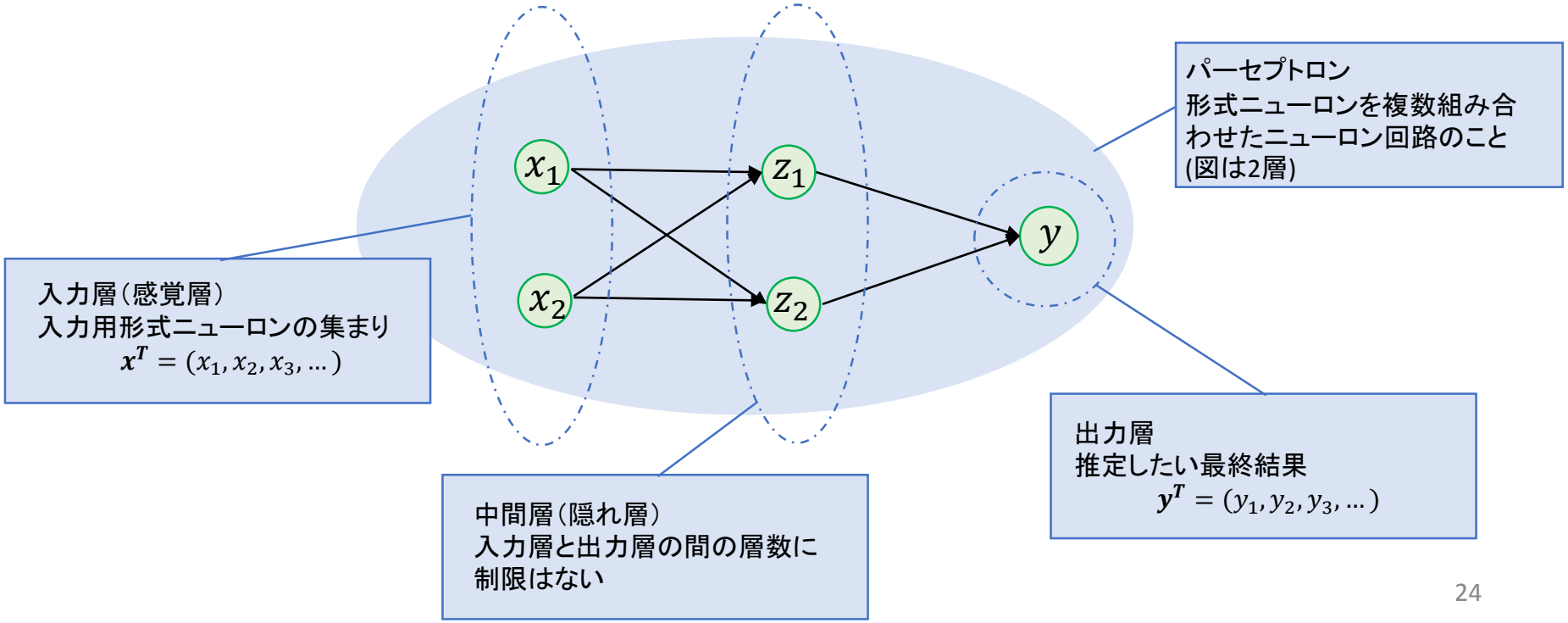
形式ニューロン…実際のニューロンの活動をモデル化したもの
機能：他の多数の形式ニューロン$i = 1,2, \ldots$から入力信号 x_i (0,1変数) を受け入れる



3.3 パーセプトロン

3.3.1 形式ニューロンによるパーセプトロン

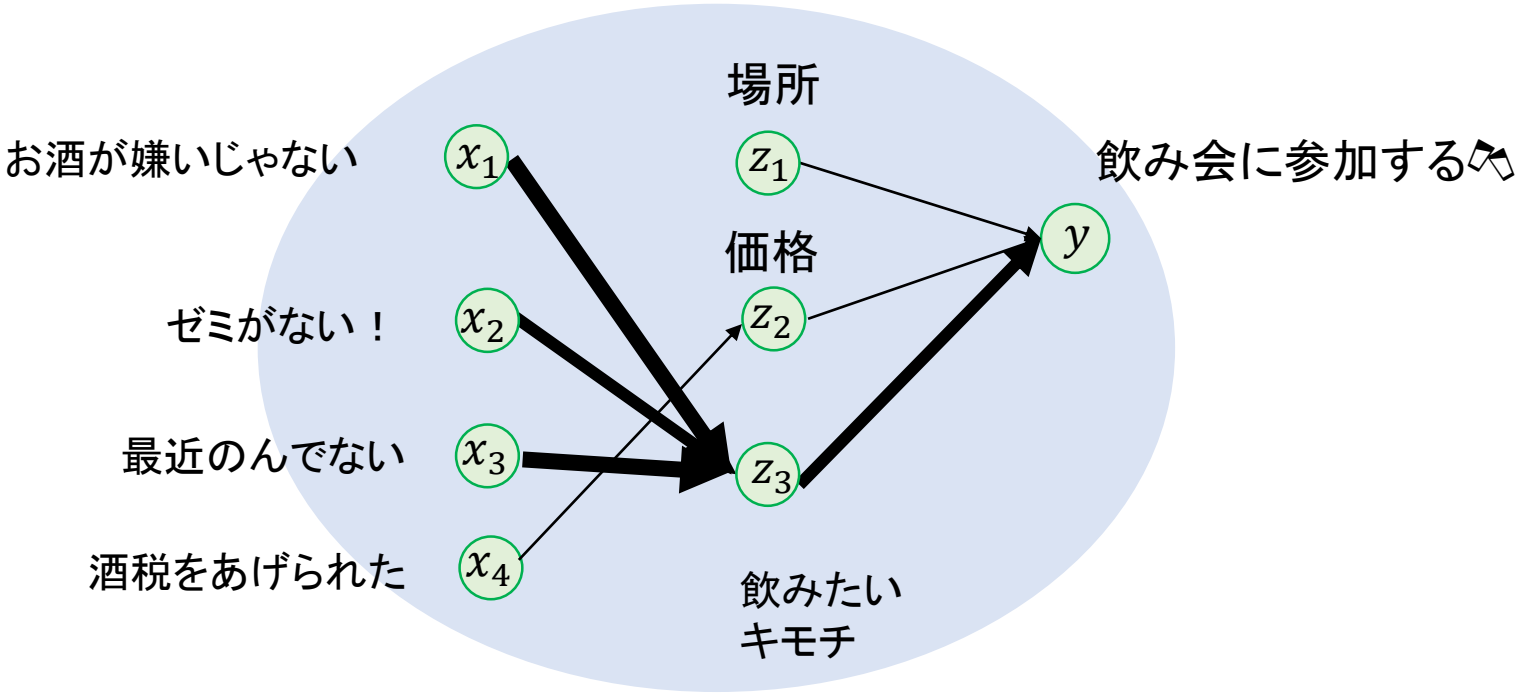
1943: W.マカロック（外科医），W.ピッツ(神童)
「固定値：重みwとバイアスbをもつ形式ニューロンを考える」
↓
1958: ローゼンプラット（心理学者）
「形式ニューロンのw, bは固定せず，期待問題をよく処理できるように訓練させる」
教師あり学習の始まり（詳細は4章）



3.3 パーセプトロン

3.3.1 形式ニューロンによるパーセプトロン

1943: W.マカロック（外科医），W.ピッツ(神童)
「固定値：重みwとバイアスbをもつ形式ニューロンを考える」
↓
1958: ローゼンプラット（心理学者）
「形式ニューロンのw, bは固定せず，と期待問題をよく処理できるように訓練させる」
教師あり学習の始まり（詳細は4章）

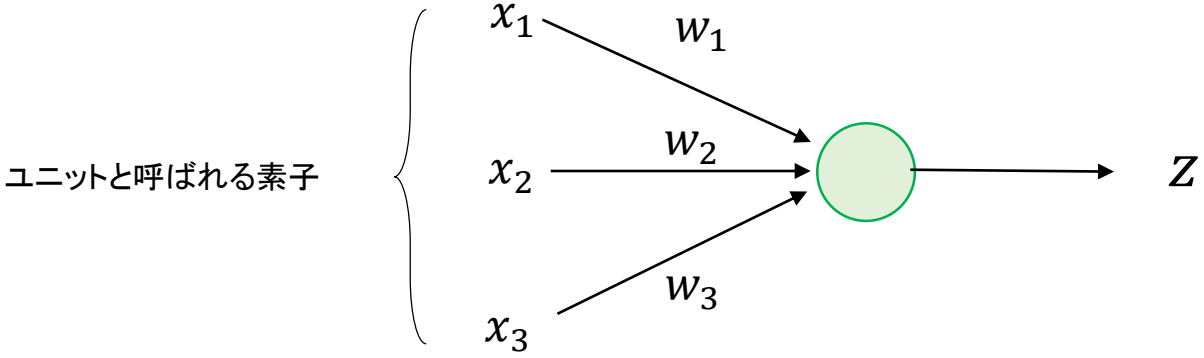


3.4 順伝番型ニューラルネットワークの構造

→この章では，現代型ニューラルネットについて

3.4.1 ユニットと順伝番型ニューラルネットワーク

形式ニューロン	パーセプトロン
• 入出力は(0,1)の2値しかとらない • 活性化関数として(0,1)の2値を示す階段関数を使用 （Ex. ヘビサイドの階段関数）	• 入出力にすべての実数値をとる • 活性化関数として微分可能な増加関数を採用できる （Ex. シグモイド関数）



結合強度の違いを考慮して
総入力は

$$u = \sum_i w_i x_i$$

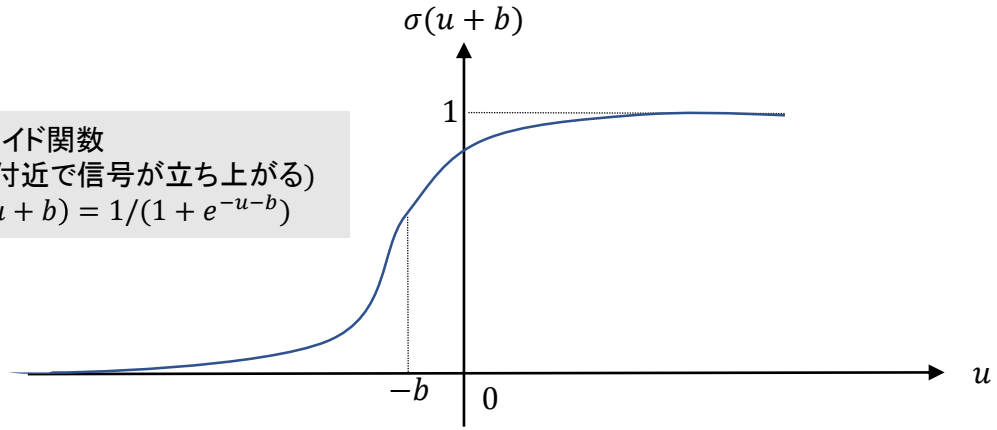
活性(activation)

活性化関数fで変換した出力は

$$z = f(u + b)$$

バイアス(bias)

$$= f\left(\sum_i w_i x_i + b\right)$$

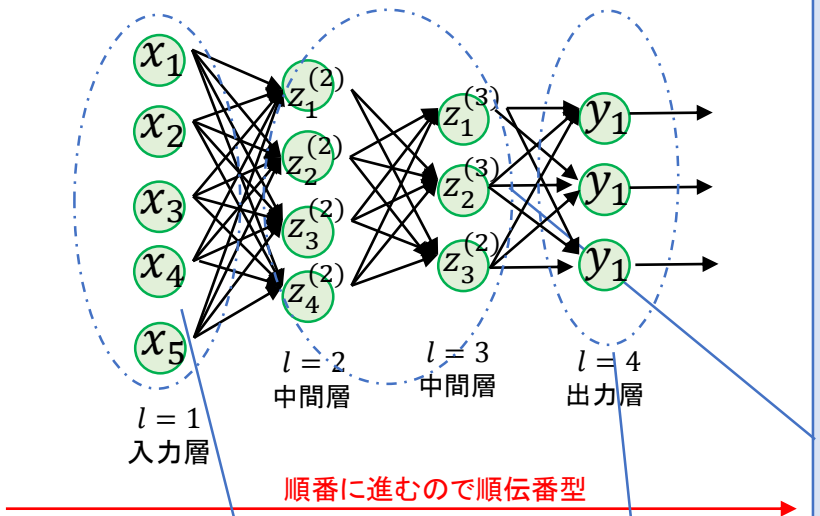


シグモイド関数
(閾値付近で信号が立ち上がる)
$\sigma(u+b) = 1/(1+e^{-u-b})$

3.4 順伝番型ニューラルネットワークの構造

3.4.1 ユニットと順伝番型ニューラルネットワーク

ユニットを層状につなぎ合わせた
アーキテクチャ，ニューラルネット



順番に進むので順伝番型

3.4.2 入力層（感覚層）
入力用形式ニューロンの集まり
$\boldsymbol{x}^{\boldsymbol{T}} = (x_1, x_2, x_3, x_4, x_5)$
出力は第1層からの出力として
$z_i^{(1)} = x_i, \boldsymbol{z}^{(\boldsymbol{1})} = \boldsymbol{x}$

3.4.4 出力層
最後のL層が出力層に当たり$L-1$層から$\boldsymbol{h} = \boldsymbol{z}^{(l-1)}$を入力され，最終出力$\boldsymbol{y} = \boldsymbol{z}^{(L)}$を得る
$\hat{\boldsymbol{y}} = \boldsymbol{z}^{(L)} = f^{(L)}(\boldsymbol{u}^{(L)}), \quad \boldsymbol{u}^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{h}$

3.4.5 関数
まとめると
$\hat{\boldsymbol{y}} = f^{(L)}\left(\boldsymbol{W}^{(L)} f^{(L-1)}\left(\boldsymbol{W}^{(L-1)} f^{(L-2)}\left(\cdots \boldsymbol{W}^{(2)} f^{(1)}\right)\right)\right)$

3.4.2 中間層（隠れ層）
第l層中のj番目のユニットの活性：
第$l-1$層の各ユニットからの入力$z_i^{(l-1)}$を入力とする

$$u_j^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)}$$

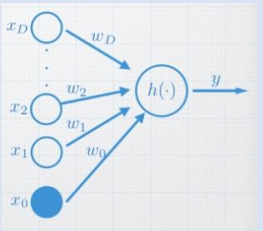
ユニットの出力は活性$u_j^{(l)}$にユニット固有のバイアス$b_j^{(l)}$を加え，
活性化関数$f^{(l)}$を用いて

$$z_j^{(l)} = f^{(l)}\left(u_j^{(l)} + b_j^{(l)}\right) = f^{(l)}\left(\sum_i w_{ji}^{(l)} z_i^{(l-1)} + b_j^{(l)}\right)$$

※活性化関数$f^{(l)}$は層ごとに共通の関数であるのが一般的

→式を行列表示する
第l層の全ユニットの活性：$\boldsymbol{u}^{(l)} = \left(u_j^{(l)}\right)$，出力値：$\boldsymbol{z}^{(l)} = \left(z_j^{(l)}\right)$，
第$l-1$層のユニットiと第l層のユニットjの間の結合の重みw_{ji}^{l}を$(j.i)$成分とする重み行列：

$$\boldsymbol{W}^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots \\ w_{21}^{(l)} & w_{22}^{(l)} & \\ \vdots & & \ddots \end{pmatrix}$$



バイアス：$\boldsymbol{b}^{(l)} = \left(b_j^{(l)}\right)$として表記すると，
中間層の行う演算処理は以下のようにまとめられる

重み和の計算

$$\boldsymbol{u}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)}, \qquad \boldsymbol{z}^{(l)} = f^{(l)}(\boldsymbol{u}^{(l)} + \boldsymbol{b}^{(b)})$$

バイアスも重みに含めることができ，（証明省略）

$$\boldsymbol{u}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)}, \qquad \boldsymbol{z}^{(l)} = f^{(l)}(\boldsymbol{u}^{(l)})$$

3.5 ニューラルネットによる機械学習

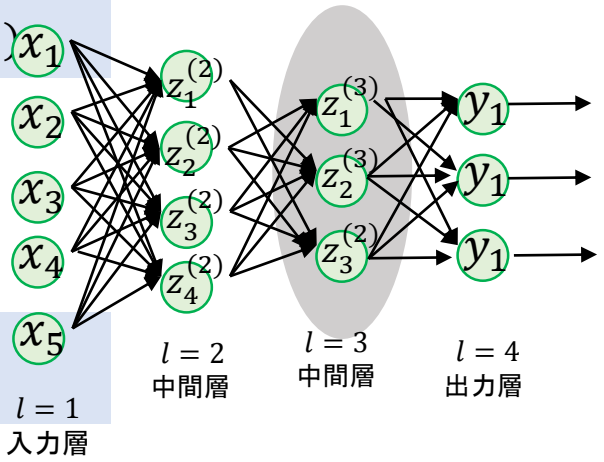
順伝播型ニューラルネットは入力xを受け取ると出力$\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{W}^{(2)}, \dots, \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(2)}, \dots, \boldsymbol{b}^{(L)})$を得る(前)
重み$\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}$をまとめてパラメータ$\boldsymbol{w}$とする.
→教師あり学習とみなすことができる

学習：訓練データ集合$D = \{(\boldsymbol{x_n}, \boldsymbol{y_n})\}_{n=1,\dots,N}$ に対して誤差関数$E(\boldsymbol{w})$の最小化をする $\boldsymbol{w}^* = \underset{w}{\operatorname{argmin}} E(\boldsymbol{w})$

ここで，最後の中間層を考えると，

$$\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{W}^{(2)}, \dots, \boldsymbol{W}^{(L-1)}) = \boldsymbol{z}^{(L-1)}(\boldsymbol{x}; \boldsymbol{W}^{(2)}, \dots, \boldsymbol{W}^{(L-1)})$$

入力$\boldsymbol{x}$に対する深層表現$\boldsymbol{h}$　　$L-1$層からの出力



3.5.1 回帰

深層表現$\boldsymbol{h}$について線形回帰するとき自明な活性化関数(恒等写像)を用いた出力層で

定義3.1 (線形ユニット)

$$\boldsymbol{y} = \boldsymbol{W}^{(L)}\boldsymbol{h}$$

一般的な非線形回帰では，

$$y(\boldsymbol{x}; w) = f^{(L)}(W^{(L)}h)$$

平均二乗誤差は，

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n; w) - y_n)^2, w^* = argmin_\theta E(w)$$

回帰関数のパラメータのみにあらず中間層の重みパラメータも推定！

3.5 ニューラルネットによる機械学習

3.5.2　2値分類

2値分類の実現→出力層を$h = z^{(L-1)}$のロジスティック回帰を与えるようデザイン

◆ 出力層の構造

定義3.2 (シグモイドユニット)

$$y(x; w) = P(y = 1|x; w) = \sigma\left(\sum_i w_i^{(L)} h_i\right)$$

$\hat{y} = P(y = 1|x)$を推定する

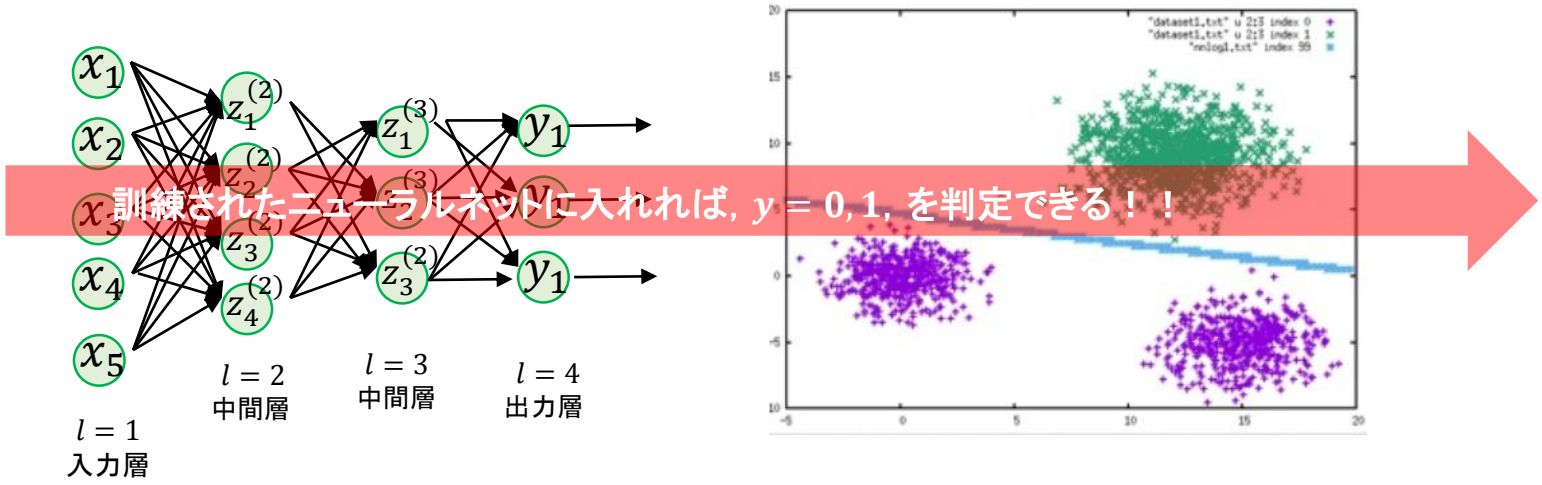
$\because E_{P(y|x)}[y|x]$

$= \sum_{y=0,1} P(y|x) = P(y = 1|x)$

◆ シグモイドユニットの活性化関数＝ロジスティック回帰におけるシグモイド関数

◆ ユニットへの総入力＝対数オッズ比

$$f^{(L)} = \sigma, u^{(L)}(x; w) = \frac{P(y = 1|x; w)}{1 - P(y = 1|x; w)}$$



3.5 ニューラルネットによる機械学習

3.5.3 多クラス分類

2章でやったとおり!!!!

◆ 各ユニットは出力値として，入力xが$y = k$のクラスに属する確率$P(y = k|\boldsymbol{x})$を推定する

定義3.3 (ソフトマックスユニット)

$$y_k(\boldsymbol{x}; \boldsymbol{w}) = P(y = k|\boldsymbol{x}; \boldsymbol{w}) = softmax_k(u_1^{(L)}, u_2^{(L)}, \dots, u_k^{(L)})$$
$$\boldsymbol{u}^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{h}$$

◆ 交差エントロピー(誤差関数)

$$E(\theta) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{n,k} log y_k(x_n; \theta)$$

3.6 活性化関数

活性化関数は経験的に決める

3.6.1 シグモイド関数と仲間たち

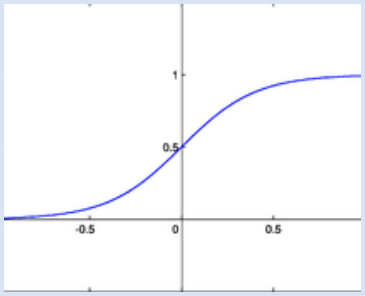
シグモイド関数

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

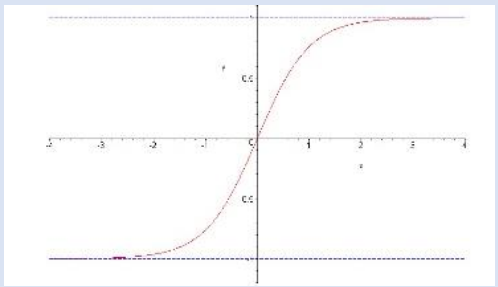
微分すると$\sigma'(u)=\sigma(u)(1-\sigma(u))$
※あんまり使われない

双曲線正接関数

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



※シグモイド関数



※双曲線正接関数

3.6.2 正規化線形関数

→学習をスムーズに進行させる

$$f(u)=\max\{0,u\}=\begin{cases}u(x>0)\\0(x\leq 0)\end{cases}$$

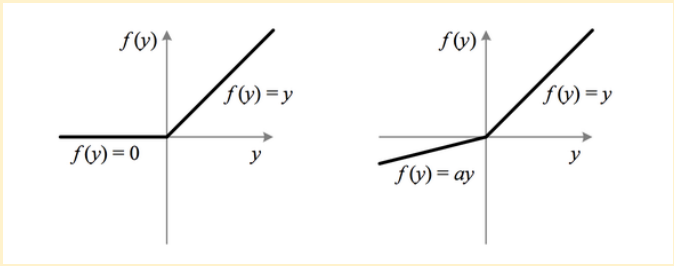
→これを持つユニットをReLUと呼ぶ

＊一般化した形

$$f(u)=\alpha\min\{0,u\}+\max\{0,u\}$$

α：ハイパーパラメータ

→ αも重みとして学習させるときpReLU



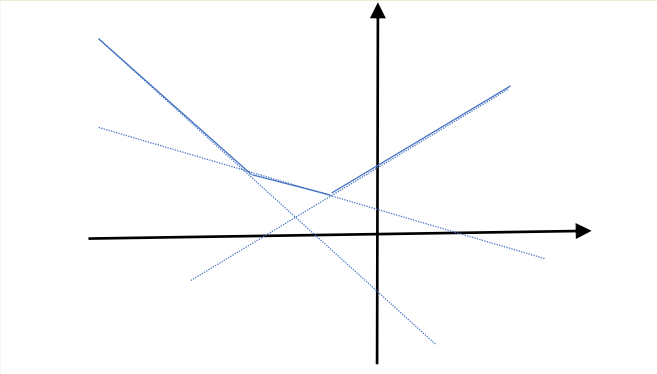
※ReLU　　※pReLU

3.6 活性化関数

活性化関数は経験的に決める

3.6.3 マックスアウト
関数形を学習できる区分線形関数
総入力u_j, 同じマックスアウト層に属するk個のユニットの総入力をすべて使い出力する

$$f(u)_j = maxu_i, \mathrm{I}_{k,j} = \{(j-1)k+1, (j-1)k+2, \dots, jk\}$$



区分線系近似の形になる
最近よく使われている！

参考文献

- これならわかる深層学習入門(1～3章)
- AITCオープンラボ: 第1回 機械学習勉強会
- DeNA TechCon2018 ゲーム体験を支えるための強化学習
- 作って遊ぶ機械学習
- Wikipedia(主にグラフを引用)

- ディープラーニングの仕組みhttps://products.sint.co.jp/aisia/blog/vol1-5

- 教師あり学習と教師なし学習 (Vol.9)
  https://products.sint.co.jp/aisia/blog/vol1-9

機械学習ゼミ

深層学習（４章~6章）

M1　小川晃平

目次

第４章　勾配降下法による学習

第５章　深層学習の正則化

第６章　誤差逆伝播法

第４章　勾配降下法による学習

ニューラルネットによる機械学習，その他の機械学習

“誤差関数の最小化問題”を解くことで実行される

複雑で解答が困難，数値的な方法に頼るほかない

深層学習の標準的手法

「**勾配降下法**」について紹介

4.1　勾配降下法

勾配降下法の考え方
⇒上からボールを落とし一番低いところへ辿り着くまで待つ

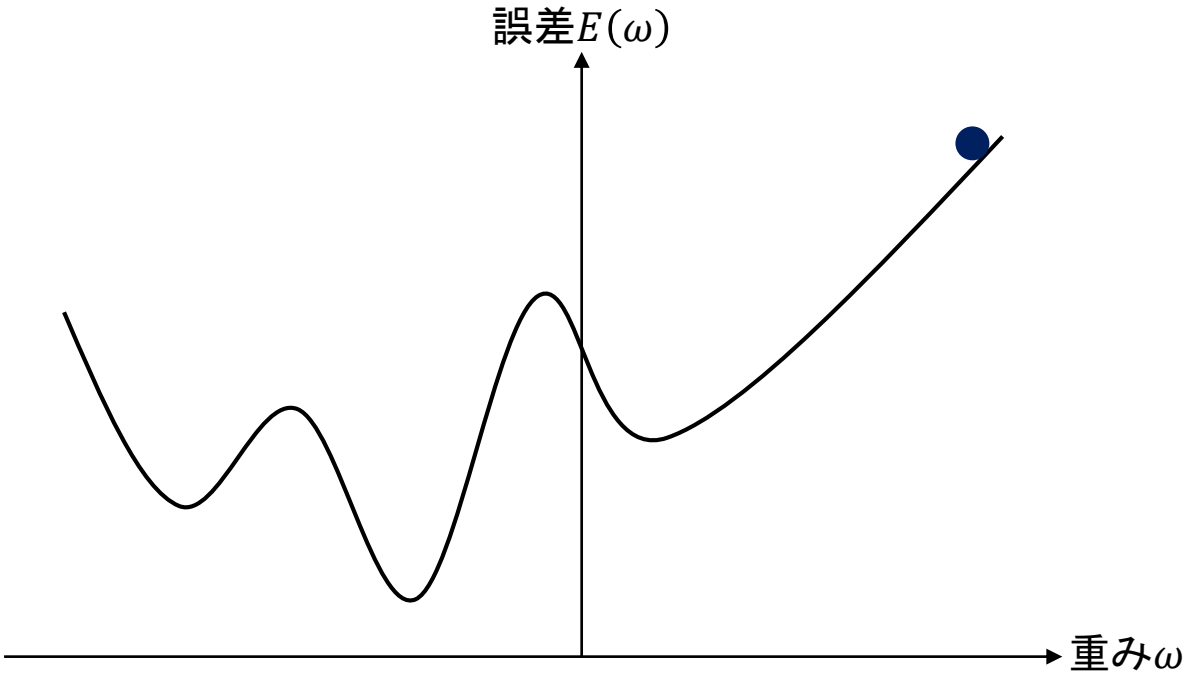


図4.1　勾配降下法による極小値の見つけ方

$$\nabla E(\boldsymbol{\omega}) = \frac{\partial E(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} \equiv \left(\frac{\partial E(\boldsymbol{\omega})}{\partial \omega_1}, \cdots, \frac{\partial E(\boldsymbol{\omega})}{\partial \omega_D} \right)^{\mathsf{T}} \qquad (4.8)$$

- 現在位置の勾配の逆方向（傾き：正，値：負）に移動する

4.1　勾配降下法

時刻tで$\boldsymbol{\omega}^{(t)}$にあったボールを勾配の逆方向へ動かす際のルール

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} + \Delta\boldsymbol{\omega}^{(t)}, \Delta\boldsymbol{\omega}^{(t)} = -\eta\nabla E\left(\boldsymbol{\omega}^{(t)}\right) \qquad (4.9)$$

$t = 0,1,2\cdots$と順次繰り返すことで，誤差関数グラフの底へ降りていく

＊１ステップの移動距離$\Delta\boldsymbol{\omega}^{(t)}$は“学習率”$\eta$で決定される

学習率の取り方はトライアンドエラーに基づく…

- 学習率が大きい
  ⇒刻みが荒すぎて誤差関数の形状を捉えられず，収束しない
- 学習率が小さい
  ⇒学習が一向に進行しない

適度な学習率を見つけることが学習を進めるうえで重要

4.1　勾配降下法

＊最小値と極小値を区別していない！

深層学習における誤差関数は非常に複雑な非凸関数
⇒大域的極小値（真の最小値）以外にも局所的極小値が沢山

▼局所的最適解の問題

極小値が膨大で，深層学習では真の最小値に辿り着けない

ところが…
誤差関数のよい極小値を見つけることができれば十分とされている

なぜ？
現在でも解明されていない深層学習における謎のひとつ

多くの場合“極小値さえ見つければ十分”であると理解してください

4.1　勾配降下法

誤差関数の値があまりに大きい臨界点に嵌ることは問題
⇒回避するためにランダムな要素を取り入れ弾き出す

訓練データ$\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1,\cdots,N}$

誤差関数は各訓練サンプル要素で計算した誤差の和で表現される

$$E(\boldsymbol{\omega}) = \sum_{n=1}^{N} E_n(\boldsymbol{\omega}) \qquad (4.10)$$

毎回の更新ですべての訓練サンプルを必要としていた⇒“バッチ学習”

一方で…

勾配による更新は一部の訓練サンプルを用いる⇒“ミニバッチ学習”

各時間tで用いる訓練サンプル$\mathcal{B}^{(t)}$の部分集合を用意
$\mathcal{B}^{(t)}$をミニバッチと呼び，通常は学習前にランダムに作成しておく

4.1　勾配降下法

時刻tにおける更新ではミニバッチ上で平均した誤差関数を用いる

$$E^{(t)}(\boldsymbol{\omega}) = \frac{1}{\left|\mathcal{B}^{(t)}\right|} \sum_{n \in \mathcal{B}^{(t)}} E_n(\boldsymbol{\omega}) \qquad (4.13)$$

$n \in \mathcal{B}^{(t)}$ …ミニバッチに含まれる訓練サンプルのラベル

$\left|\mathcal{B}^{(t)}\right|$…ミニバッチ中のサンプル要素の総数

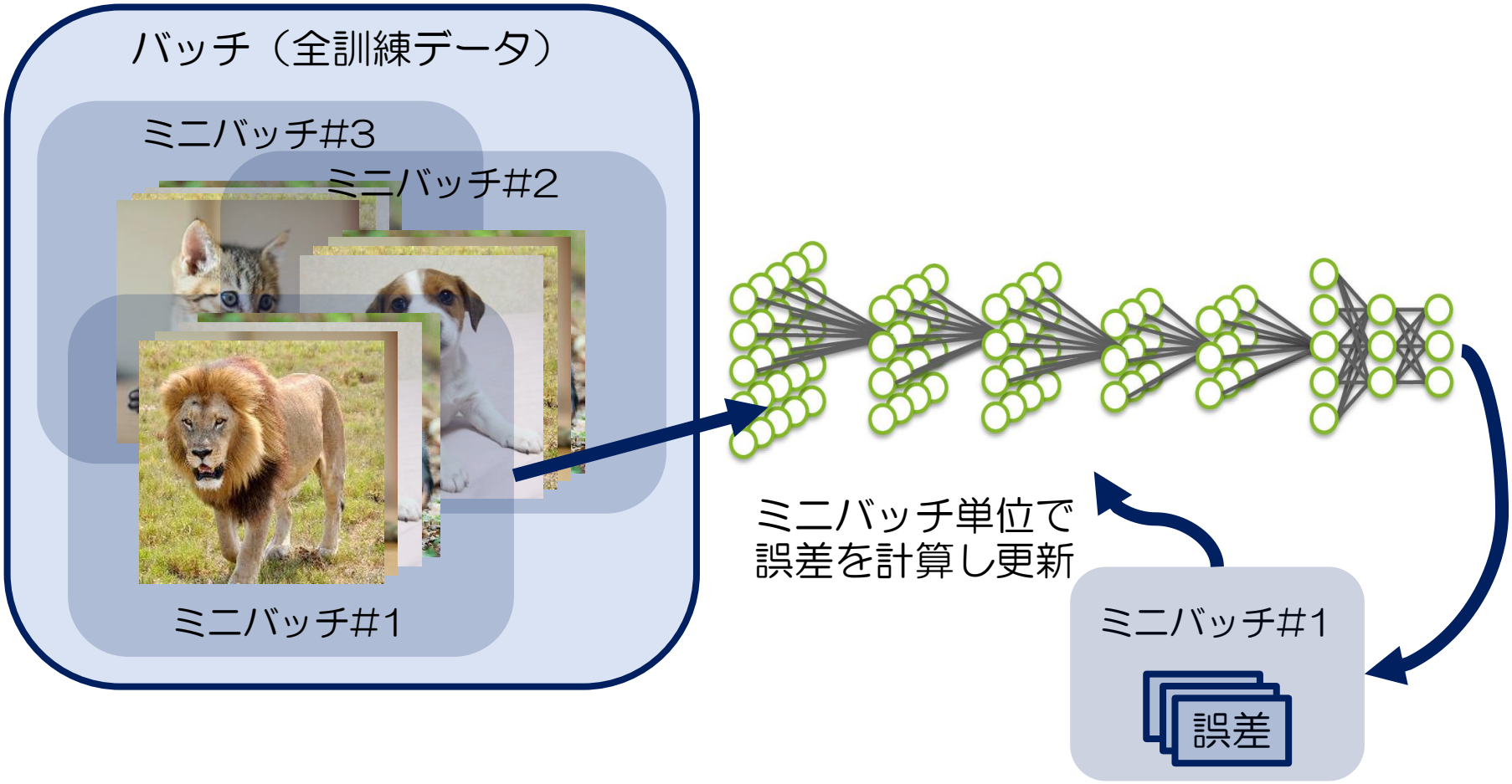
パラメータの更新…

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} + \Delta\boldsymbol{\omega}^{(t)}, \Delta\boldsymbol{\omega}^{(t)} = -\eta \nabla E^{(t)}\left(\boldsymbol{\omega}^{(t)}\right) \qquad (4.14)$$

$*\left|\mathcal{B}^{(t)}\right| = 1$の場合を“確率的勾配降下法（SGD）”という

- 時間ごとに$E^{(t)}(\boldsymbol{\omega})$が変化するため，望ましくない臨界点に嵌り込む可能性が低くなる
- 訓練データが大きいと類似したデータが偏る可能性があるが，重複の無駄を省くことができる

4.1　勾配降下法



- 全訓練データを使った１サイクルを“エポック”と呼ぶ
- すべてのミニバッチを使い切るとエポックは終了となる

4.1　勾配降下法

＊学習率ηが時間を通じて一定だと仮定していた！

1ステップごとの更新サイズが一定のまま
⇒収束したい点になかなか辿り着かない，収束までに時間がかかる

“ミニバッチ法”や“SGD”は誤差関数の推定値を
ミニバッチ上での期待値として近似的に与えていた

ミニバッチによる推定値を用いる間はランダムな効果が
消えないため誤差関数の極小値へは落ち着かない

極小値に近づくにつれて学習率を小さくする必要がある
⇒時間に依存する学習率の導入

4.1　勾配降下法

時間に依存する学習率の導入

$$\Delta \boldsymbol{\omega}^{(t)} = -\eta^{(t)} \nabla E^{(t)}\left(\boldsymbol{\omega}^{(t)}\right) \qquad (4.15)$$

凸誤差関数上のSGDに関しては収束を保証する$\eta^{(t)}$の条件が存在

$$\sum_{t=1}^{\infty} \eta^{(t)} = \infty \,, \sum_{t=1}^{\infty} \left(\eta^{(t)}\right)^2 < \infty \qquad (4.16)$$

厳密にこの条件を満たす必要はなく，よく用いられる学習率の取り方は…

$$\eta^{(t)} = \begin{cases} \dfrac{t}{T}\eta^{(T)} + \left(1 - \dfrac{t}{T}\eta^{(0)}\right) & (t \leq T) \\ \eta^{(T)} & (t \geq T) \end{cases} \qquad (4.17)$$

ある大きな時刻Tを選んでおき，それまでは線形に減衰させる
それ以降は学習率を正の一定値に固定

4.2　改良された勾配降下法

“局所的最適解の問題”以外にも課題が存在する

- 振動

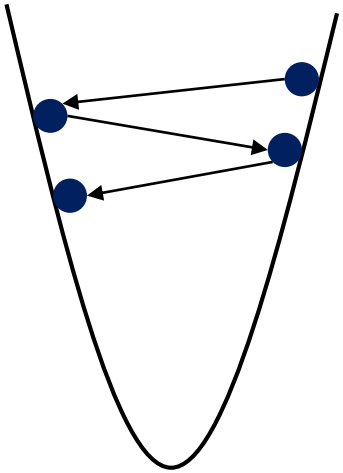


図4.2(a)　谷での振動

誤差関数が深い谷を作っている状況
⇒勾配方向にパラメータを更新しても
　極小値へ降りていかない

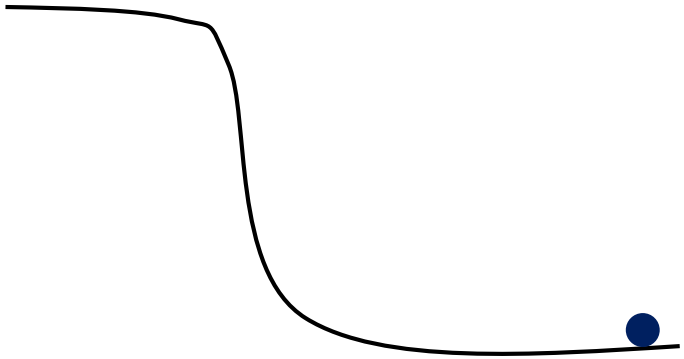
4.2　改良された勾配降下法

- プラトー（広く平らな領域）

誤差関数の勾配がなくなる
⇒更新が止まってしまう

図4.2(b)　プラトーでの停止

- 絶壁



誤差関数が急に切り立った絶壁
⇒壁に当たった瞬間に極めて
　大きな勾配によって吹き飛ばされる

図4.2(c)　絶壁での反射

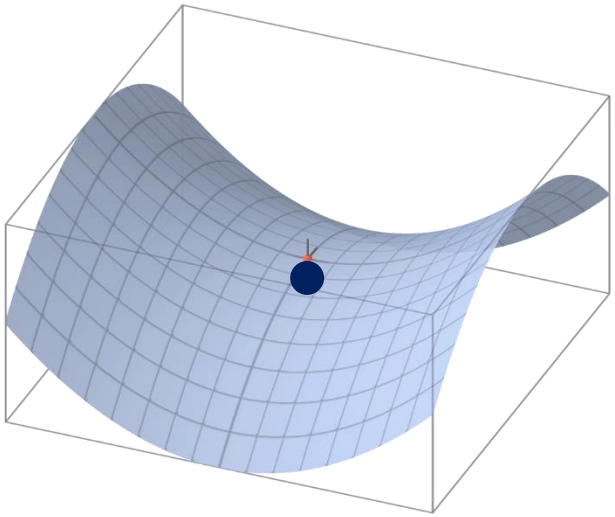
4.2　改良された勾配降下法

これらの問題への対策
⇒“勾配クリップ”

$$\Delta\boldsymbol{\omega}^{(t)} = -\eta \frac{g_0}{|\nabla E|} \nabla E\left(\boldsymbol{\omega}^{(t)}\right) \qquad (4.19)$$

勾配の大きさ$|\nabla E|$に閾値g_0を設定して，それが閾値を超えた場合（$|\nabla E| \geq g_0$）に勾配によるパラメータ更新の大きさを上式で調整する

- 鞍点



ある方向にずれると下り始める
別の方向へずれると上り始める

鞍点周辺は比較的平坦な領域が
広がっていることが多く，危険性が高い

学習アルゴリズムの工夫が必要

4.2　改良された勾配降下法

“モーメンタム法”
勾配法の振動を抑制し，極小値への収束性を改善する手法

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} + \Delta\boldsymbol{\omega}^{(t)} \qquad (4.20)$$

$$\Delta\boldsymbol{\omega}^{(t)} = \mu\Delta\boldsymbol{\omega}^{(t-1)} - (1-\mu)\eta\nabla E\left(\boldsymbol{\omega}^{(t)}\right) \qquad (4.21)$$

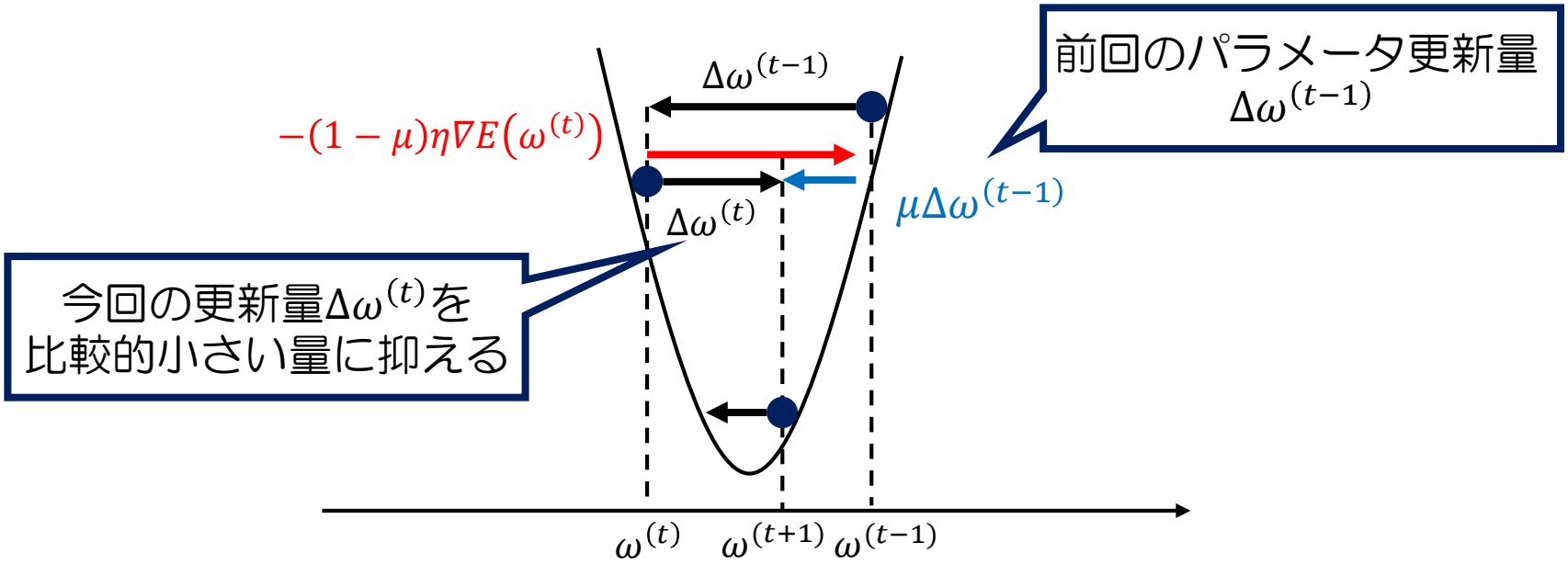


図4.6　モーメンタムによる谷での振動の抑制

4.2　改良された勾配降下法

“ネステロフの加速勾配法”
⇒モーメンタム法の修正版で勾配の値を評価する位置が異なる

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} + \Delta\boldsymbol{\omega}^{(t)} \qquad (4.23)$$

$$\Delta\boldsymbol{\omega}^{(t)} = \mu\Delta\boldsymbol{\omega}^{(t-1)} - (1-\mu)\eta\nabla E\left(\boldsymbol{\omega}^{(t)} + \mu\boldsymbol{\omega}^{(t-1)}\right) \qquad (4.24)$$

$\boldsymbol{\omega}^{(t)} + \mu\boldsymbol{\omega}^{(t-1)}$によって次時刻のt+1での位置を大雑把に見積り
そこでの勾配の大きさを用いている

少し先での位置を用いることで，勾配が変化する前に，
あらかじめその変化に対応できるよう改良されている

4.2　改良された勾配降下法

＊勾配降下方には１つの学習率しかない
　⇒実際にはパラメータ空間の座標方向によって誤差関数の勾配は異なる

ex)　ω_1方向は急勾配，ω_2方向は緩勾配であれば，学習率が１つの勾配法でははω_1方向のみ学習が進行してしまう
仮に複数の学習率が導入できたなら，収束するまでの時間は短縮できるが，トライアンドエラーの負担が大きくなる

“AdaGrad”
$$\Delta\boldsymbol{\omega}_i^{(t)} = -\frac{\eta}{\sqrt{\sum_{s=1}^{t}(\nabla E(\omega^{(s)})_i)^2}}\nabla E\left(\boldsymbol{\omega}^{(t)}\right)_i \qquad (4.25)$$

左辺：$\Delta\boldsymbol{\omega}^{(t)}$の第i成分　，　$\nabla E\left(\boldsymbol{\omega}^{(t)}\right)_i$：勾配の第i成分

学習率を過去の勾配成分の二乗和の平方根で除している
⇒勾配値が大きな方向：学習率を減少
　勾配値が小さな方向：学習率を増大

＊欠点：初期の勾配が大きいと更新量$\Delta\boldsymbol{\omega}_i^{(t)}$がすぐに小さくなる，0になると有限値に戻らない

4.2　改良された勾配降下法

“RMSprop”
⇒“AdaGrad”の欠点を解消

$$v_{i,t} = \rho v_{i,t-1} + (1-\rho)\left(\nabla E\left(\boldsymbol{\omega}^{(t)}\right)_i\right)^2 \qquad (4.26)$$

$$\Delta\boldsymbol{\omega}_i^{(t)} = -\frac{\eta}{\sqrt{v_{i,t}+\epsilon}}\nabla E\left(\boldsymbol{\omega}^{(t)}\right)_i \qquad (4.27)$$

初期値$v_{i,0} = 0$，$\epsilon = 10^{-6}$として分母が0にならないようにする

最近の勾配の履歴のみが影響するため，更新量が消えることがない
鞍点を抜けた後は，モーメンタム法と組み合わされたりする

＊欠点：学習率に鋭敏である事実に改善が見られなかった
（トライアンドエラーの負担が減っていない）

4.3　重みパラメータの初期値の取り方

重みパラメータの初期値（初めにボールを置く場所）
⇒勾配降下法での学習がうまくいくかどうかを左右する

実際の研究ではガウス分布や一様分布から
初期値をサンプリングする

分布の分散の大きさ次第で学習の成否は様々となる

適した分散の大きさの選び方とは...

4.3　重みパラメータの初期値の取り方

“LeCunの初期化”

$$\omega_{ji}^{(l)} \sim \mathcal{U}\left(w; -\sqrt{\frac{3}{d_{l-1}}}, \sqrt{\frac{3}{d_{l-1}}}\right) \quad or \quad \mathcal{N}\left(w; 0, \frac{1}{\sqrt{d_{l-1}}}\right) \qquad (4.44)$$

重みの初期値を分散$\frac{1}{d_{l-1}}$の一様分布，ガウス分布からサンプリングする

前層と多くの結合を持つユニットに関しては，分散が小さくなるので
重みの初期値を小さく抑えることができる

4.3　重みパラメータの初期値の取り方

“Glorotの初期化”

$$\omega_{ji}^{(l)} \sim \mathcal{U}\left(w; -\sqrt{\frac{6}{d_{l-1}+d_l}}, \sqrt{\frac{6}{d_{l-1}+d_l}}\right) \quad or \quad \mathcal{N}\left(w; 0, \sqrt{\frac{2}{d_{l-1}+d_l}}\right) \qquad (4.45)$$

線形ユニットのみを持つニューラルネットワークの解析から提唱

ユニット数d_lの第l中間層の重み$\omega_{ji}^{(l)}$は
平均0，分散$\frac{2}{d_{l-1}+d_l}$の一様分布，ガウス分布からサンプリングする

活性化関数が左右対称である場合に有用な初期化

4.4　訓練サンプルの前処理

学習を加速して汎化に近づけるための，訓練データの下ごしらえ

“データの正規化”
⇒訓練サンプルの成分ごとに行う前処理

各成分iに対し，訓練サンプルの成分$\{x_{ni}\}_{n=1}^{N}$の平均と分散を一定値に規格化

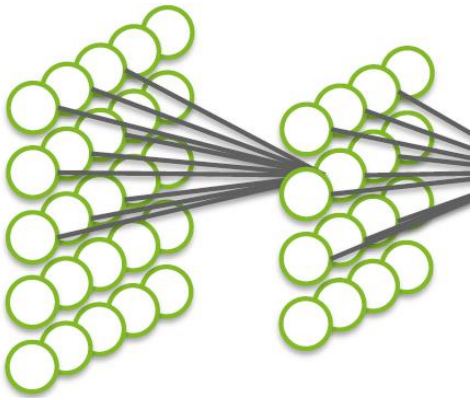
まず成分の平均値を0にするために，成分のサンプル平均を引く

$$\bar{x_i} = \frac{1}{N}\sum_{n=1}^{N} x_{ni} \Rightarrow x_{ni} - \bar{x_i} \qquad (4.52)$$

次に分散を1に規格化するため，各成分を標準偏差で除する

$$\sigma_i^2 = \frac{1}{N}\sum_{n=1}^{N}(x_{ni} - \bar{x_i})^2 \Rightarrow x_{ni}^{new} = \frac{x_{ni} - \bar{x_i}}{\sigma_i} \qquad (4.53)$$

第５章　深層学習の正則化



多層ニューラルネットは他のモデルよりも
多くの重みパラメータを持つ

膨大なパラメータを安直に学習させると
“過学習”が起きてしまう

過学習を防ぐための手法

「**正則化**」について紹介

5.1　汎化性能と正則化

機械学習の目的はモデルが導く予測をデータ生成分布と近づけること
⇒学習とはデータ生成分布上で得られる誤差関数・汎化誤差を最小化する

$$\mathrm{E}_{gen}(\boldsymbol{\omega}) = \mathrm{E}_{P_{data(X,Y)}}\left[\frac{1}{2}(\hat{y}(\boldsymbol{X};\boldsymbol{\omega}) - y)^2\right] \qquad (5.1)$$

しかし簡単に解くことはできない…
∴データ生成分布の情報が使えない代わりに，手持ち訓練データを使用

$$q(\boldsymbol{x}, y) = \frac{1}{N}\sum_{n=1}^{N}\delta_{x,x_n}\delta_{y,y_n} \qquad (5.2)$$

訓練誤差：経験分布を用いて，汎化誤差を近似的に見積もったもの

$$E(\boldsymbol{\omega}) = \mathrm{E}_{q(X,Y)}\left[\frac{1}{2}(\hat{y}(\boldsymbol{X};\boldsymbol{\omega}) - y)^2\right] = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(\hat{y}(\boldsymbol{x}_n;\boldsymbol{\omega}) - y)^2 \qquad (5.3)$$

＊実際の学習ではこの訓練誤差を使用

5.1　汎化性能と正則化

しかし本来最小化したかったのは汎化誤差
⇒本当に最小化したい誤差関数と実際代用する誤差関数の
　ミスマッチが“過学習”を引き起こす

過学習…学習のし過ぎで判断基準が厳しくなり，少しでもパターンが
　　異なると誤った出力をしてしまうこと

＊訓練誤差が減少する一方で汎化誤差が増加していれば過学習の兆候

ただ厳密には汎化誤差を知ることはできないので“テスト誤差”を見る

$$E_{test}(\boldsymbol{\omega}) = \frac{1}{N_{test}} \sum_{m=1}^{N_{test}} \frac{1}{2} (\hat{y}(\boldsymbol{x}_m; \boldsymbol{\omega}) - y_m)^2 \qquad (5.4)$$

5.1　汎化性能と正則化

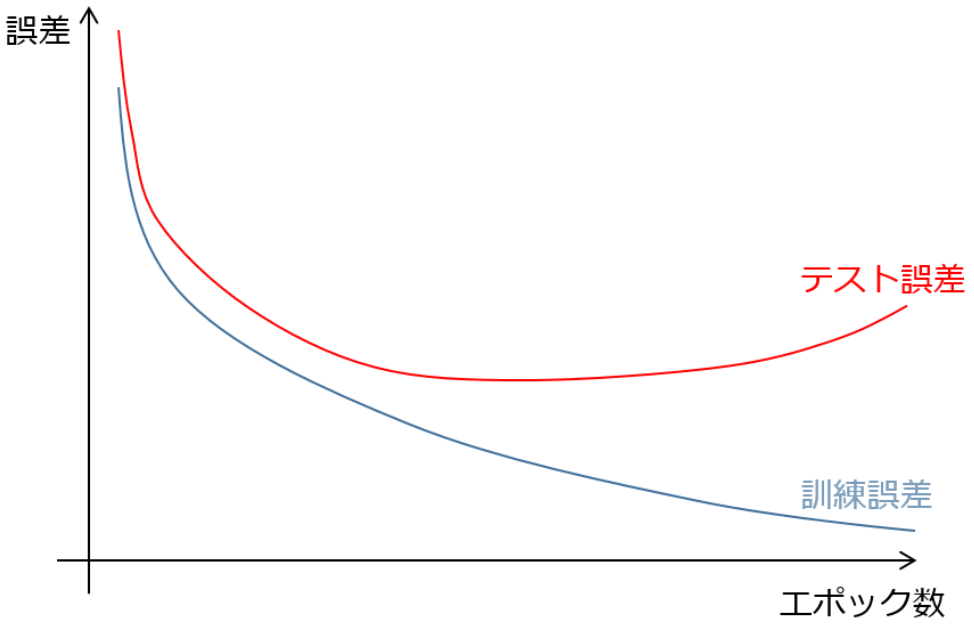


図5.1　学習曲線

- 学習が進むにつれ訓練誤差が0に近づいているため良好に見える
- テスト誤差はある点から増加しているため典型的な過学習といえる

単に訓練誤差を最小にすればいいという訳ではない

5.1　汎化性能と正則化

- 過学習が起きる原因

訓練データが有限であるため，モデルの大きな自由度がデータに含まれる統計的ばらつきをどんどん学習していく

訓練データの本質的な部分だけを取り組む必要がある
⇒“正則化”

正則化…学習アルゴリズムを修正することで汎化誤差を最小化させること

推奨されるモデル
　はじめから十分に大きな自由度を用意しておき，それに正則化を加える

$$E_{reg}(\boldsymbol{\omega}) = E(\boldsymbol{\omega}) + \alpha R(\boldsymbol{\omega}) \qquad (5.5)$$

$R(\boldsymbol{\omega})$ ：ペナルティ項
α ：ハイパーパラメータ

5.2　重み減衰

最もシンプルな正則化⇒“重み減衰”

誤差関数のある極小値ω^0近傍に注目し，２次関数で近似
ヘッシアン行列$H = diag(h_i)$と対角的だと仮定

$$E(\boldsymbol{\omega}) \approx E^0 + \frac{1}{2}\sum_i h_i\left(\omega_i - \omega_i^0\right)^2 \qquad (5.6)$$

この関数の極小値は$\omega_i = \omega_i^0$で，ここに重み減衰を加える

$E_{reg.}(\boldsymbol{\omega}) = E(\boldsymbol{\omega}) + \frac{\alpha\omega^2}{2}$の微分係数が消える点が新たな最小値なので
$0 = \nabla E_{reg.}(\boldsymbol{\omega})_i = h_i\left(\omega_i - \omega_i^0\right) + \alpha\omega_i$を解くと最小値の座標が得られる

$$\omega_i^* = \frac{h_i}{h_i + \alpha}\omega_i^0 = \frac{1}{1 + \frac{\alpha}{h_i}}\omega_i^0 \qquad (5.7)$$

αと比べてヘッシアン行列の成分が極めて小さい（$h_j \ll \alpha$）場合

$$\omega_j^* = \frac{1}{1 + \frac{\alpha}{h_j}}\omega_i^0 \approx 0 \qquad (5.8)$$

パラメータの値がほとんど0になる

5.3　早期終了

早期終了とは...

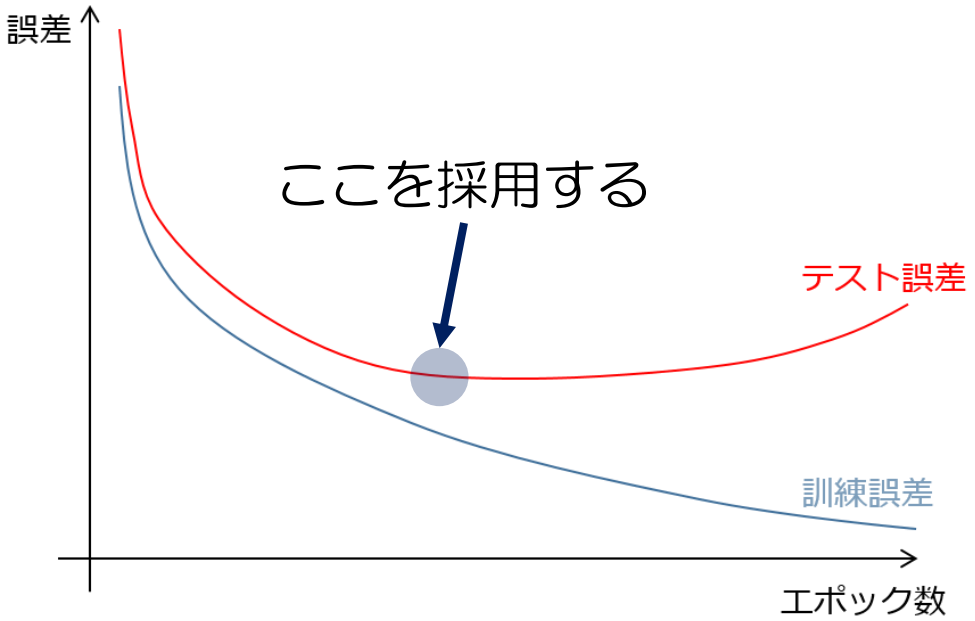


図5.1　学習曲線

過去のエポックで更新した重みパラメータを記憶したまま
学習を続け，テスト誤差が増加し始める点で学習を停止
⇒増加に転じた時点でのパラメータを採用する手法

5.3　早期終了

アルゴリズム…

周期と期間の長さを決定しておく
　周期：「テスト誤差を評価するのはパラメータ更新の何回に一度か」
　期間の長さ：「どの位のステップ数でテスト誤差が連続して
　　増加すれば学習を停止するか」

学習が始まり，テスト誤差が「期間の長さ」に達した時点で学習を停止
減少から増加へ転じた時点でのパラメータを最適値として採用する

＊データの一部を検証用に使用するため，全データを学習には使用しない
　⇒学習終了までのステップ数Tのみを記憶し，パラメータを初期化して
　　全データで再学習するという方法もある

5.4　重み共有

重み共有とは...

重みパラメータがとるべき前提知識がある場合に使える正則化の代表例

ニューラルネットワークの重みをすべて独立とせず
それらの間に拘束関係を課す

２箇所の重みの間に$\omega_{ji}^{(l)} = \omega_{pq}^{(m)}$という条件を課す

自由に調整できるパラメータ数を減らす，モデルの自由度を減らす正則化

⇒畳み込みニューラルネットワーク（第８章）が典型例

5.6　バギング

学習後の機械動作の安定性を向上させる手法“アンサンブル法”

その一例である…“バギング”

- 訓練サンプル集合から各要素の重複を許してサンプリングする
- ブートストラップサンプル集合を複数作成し，学習を行う
- 学習後はすべてのモデルの予測値をもとにして決定する

多数の予測値の平均を用いると性能が安定する理由…

ex)　m=1,…,Mでラベルされる多数のモデルを考える
各モデルの出力$y_{(m)} + \epsilon_m$には確率変数ϵ_mで表される誤差があるとする

この変数の期待値と分散，共分散は以下のようになる

$$\mathrm{E}[\epsilon_m] = 0\,,\mathrm{E}[\epsilon_m^2] = \sigma^2\,,\mathrm{E}[\epsilon_l\epsilon_m] = \sigma_{lm} \qquad (5.24)$$

5.6　バギング

モデルの誤差は以下のようになり，

$$\frac{1}{M}\sum_{m=1}^{M}\epsilon_m \qquad (5.25)$$

予測値のブレ（分散）は次のように与えられる

$$\mathrm{E}\left[\left(\frac{1}{M}\sum_{m=1}^{M}\epsilon_m\right)^2\right]=\frac{1}{M}\sigma^2+\frac{2}{M^2}\sum_{l\neq m}\sigma_{lm} \qquad (5.26)$$

右辺第１項はモデルの数Mが大きいほど小さくなる
⇒多数の予測値の平均を用いると性能が安定する

5.7　ドロップアウト

前節の手法はニューラルネットワークに適用可能だが計算コストが嵩む
モデル平均を直接用いずアンサンブル法を実現できないか…

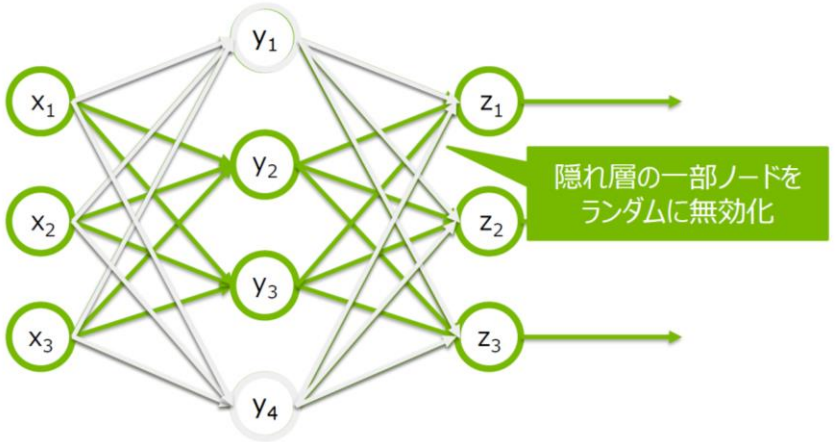
“ドロップアウト”

1. ミニバッチ$\mathcal{B}_{t=1,2,\dots}$を作成すると同時にマスク$\mu_{t=1,2,\dots}$をサンプリング

2. 学習時にマスクを用いネットワークを縮小する

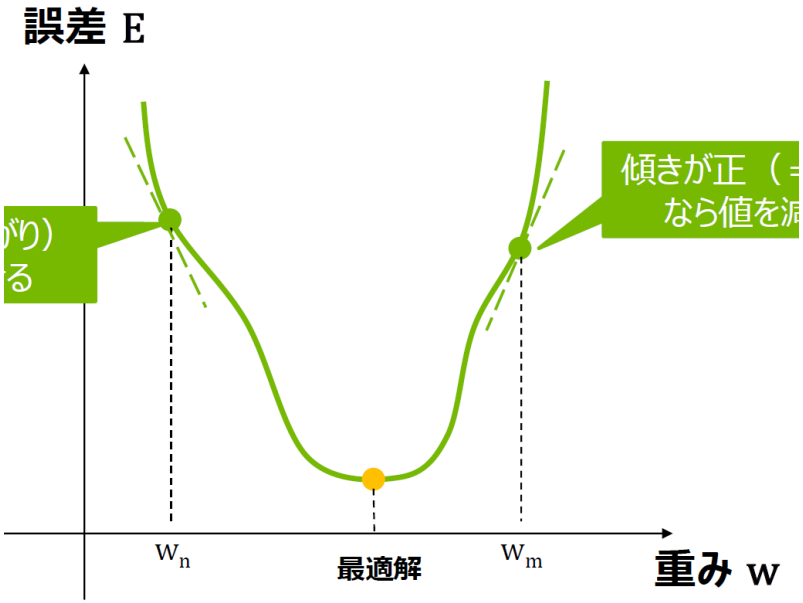
   （元のネットワークをマスクμ_tで部分ネットワークに変換し，
   それをミニバッチ$\mathcal{B}_t$で更新する）

3. マスクで取り除かれたユニットを復活させ，同様の操作を繰り返す



モデルの自由度を小さくし
過学習を回避

第６章　誤差逆伝播法



ニューラルネットワークの
学習では勾配降下法を用いた

勾配の計算が自明ではない

勾配を高速計算する

「**誤差逆伝播法**」について紹介

6.2　誤差逆伝播法

“誤差逆伝播法”
　デルタ則を多層のニューラルネットに拡張したもの

勾配降下法による学習を考える...
　勾配は誤差関数の微分係数で表される

$$\nabla_{\boldsymbol{\omega}} E(\boldsymbol{\omega}) = \frac{\partial E(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} \qquad (6.5)$$

微分のチェインルールを使うと...

$$\frac{\partial E(\boldsymbol{\omega})}{\partial \omega_{ji}^{(l)}} = \sum_{k=1}^{D_l} \frac{\partial E(\boldsymbol{\omega})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial \omega_{ji}^{(l)}} \qquad (6.6)$$

出力$\hat{\boldsymbol{y}}(\boldsymbol{x};\boldsymbol{\omega})$をパラメータ$\omega_{ji}^{(l)}$で偏微分する計算に
帰着するがこの計算は非常に複雑である

6.2　誤差逆伝播法

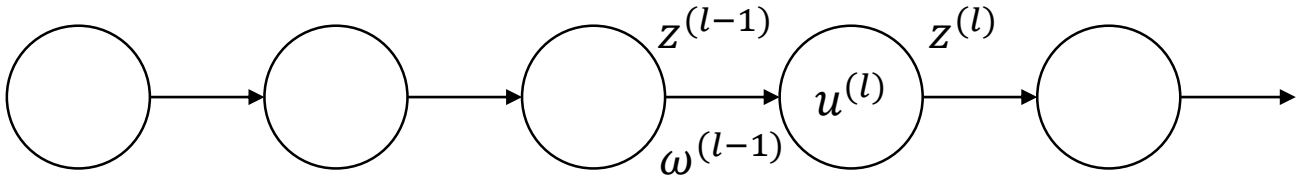


図6.1　１つのユニットからなる層を１列に繋げたニューラルネット

$$\hat{y} = f\left(\omega^{(L)} f\left(\omega^{(L-1)} f\left(\cdots \omega^{(l+1)} f\left(\omega^{(l)} f(\cdots f(x))\right)\right)\cdots\right)\right) \qquad (6.7)$$

$\omega^{(l)}$による微分係数を計算するとき
対応するパラメータの微小変動$\omega^{(l)} + \Delta\omega^{(l)}$は…
l層のノード出力$z^{(l)}$に影響，l+1層のノード出力を変動させ
l+1層，l+2層…と次々に影響し，
伝播プロセスを何層も繰り返し出力層に達する

層が深くなるほど，巨大な合成関数を計算しなくてはならない

6.2　誤差逆伝播法

ここで，$u^{(l)} = \omega^{(l)} z^{(l-1)}$の関係に着目

$\omega^{(l)}$の変動は$u^{(l)}$の値に直接影響を与えるので，チェインルールより

$$\frac{\partial E}{\partial \omega^{(l)}} = \frac{\partial E}{\partial u^{(l)}} \frac{\partial u^{(l)}}{\partial \omega^{(l)}} = \frac{\partial E}{\partial u^{(l)}} z^{(l-1)} \equiv \delta^{(l)} z^{(l-1)} \qquad (6.8)$$

つまり$\delta^{(l)} \equiv \frac{\partial E}{\partial u^{(l)}}$という量が分かれば勾配が決定する

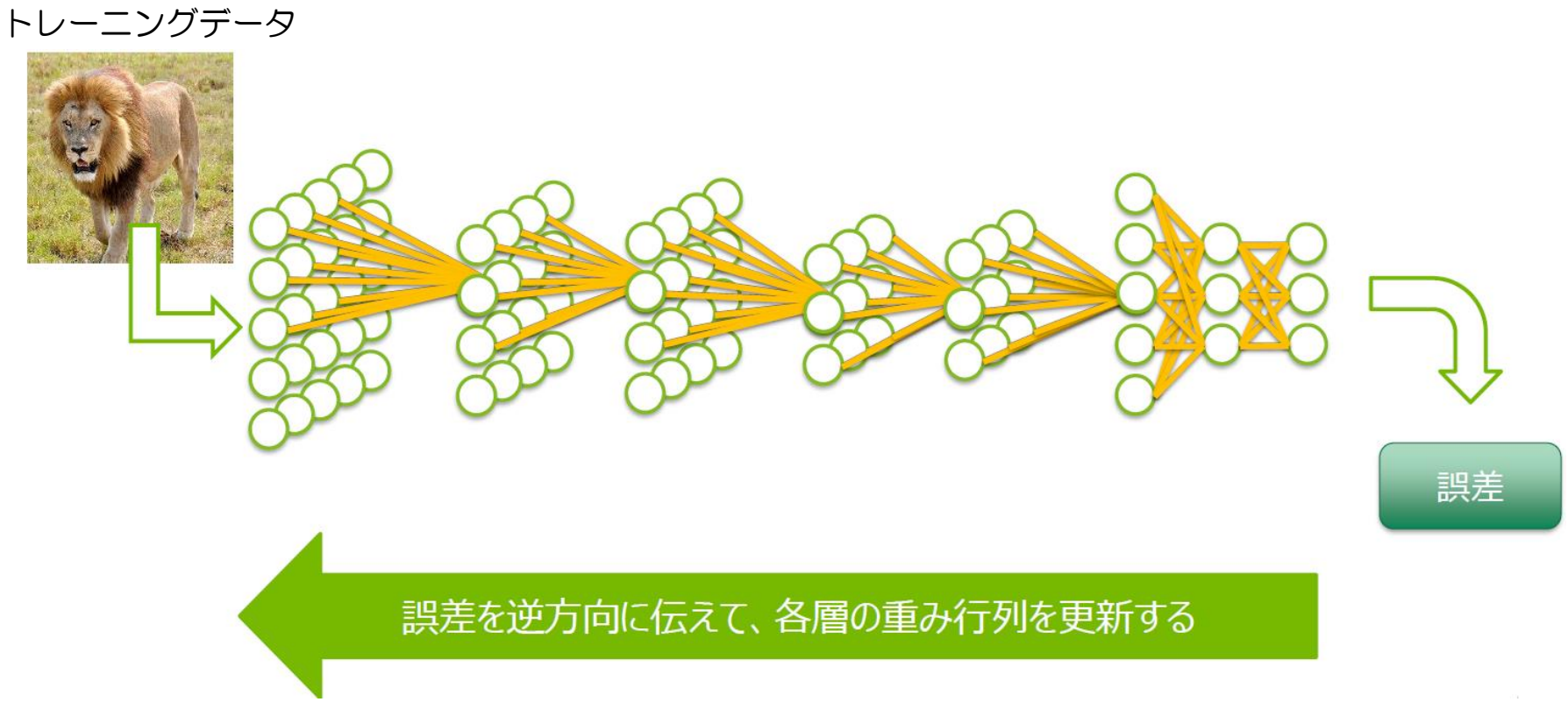
ところが$u^{(l+1)} = \omega^{(l+1)} f\left(u^{(l)}\right)$なので，このデルタが次層の$\delta^{(l+1)}$と関係する

$$\delta^{(l)} = \frac{\partial E}{\partial u^{(l)}} \frac{\partial u^{(l+1)}}{\partial u^{(l)}} = \delta^{(l+1)} \omega^{l+1} f'\left(u^{(l)}\right) \qquad (6.9)$$

これを漸化式として出力側から入力側に向けて順次解いていき勾配を求める
⇒誤差逆伝播法の考え方

6.2　誤差逆伝播法

処理のイメージ



6.2　誤差逆伝播法

巨大な合成関数の微分を一挙に計算せず分割して行った

注目している重み$\omega_{ji}^{(l)}$の変動がネットワークの局所的な構造を通して
周囲にどれくらいの影響を与えるのかを把握する

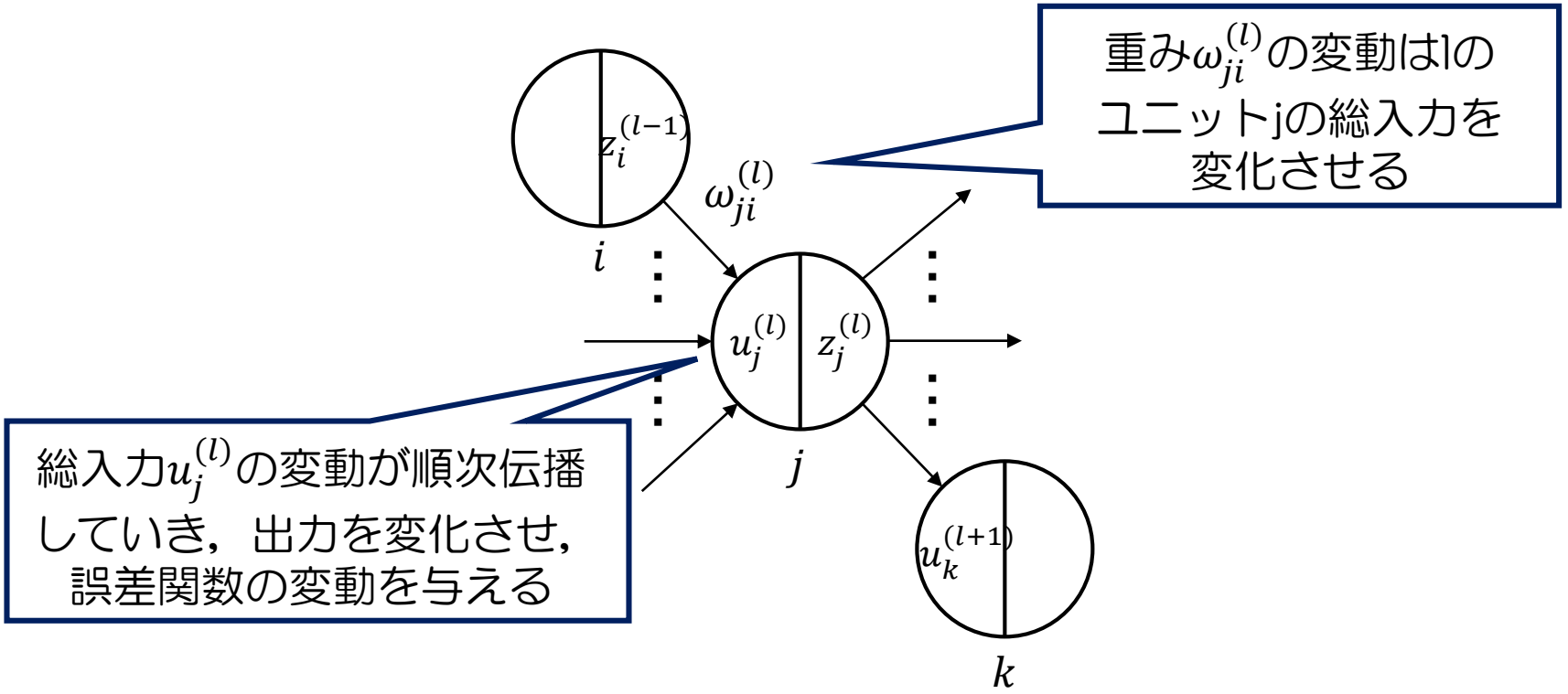


図6.2　結合の重み$\omega_{ji}^{(l)}$の変動が，各層の入出力に与える影響

6.2　誤差逆伝播法

誤差関数の変動を微分操作で表すと…

誤差関数$E_n\left(u_j^{(l)}\left(\omega_{ji}^{(l)}\right)\right)$は総入力$u_j^{(l)}$を通じて，
重み$\omega_{ji}^{(l)}$に間接的に依存しているため

$$\frac{\partial E_n}{\partial \omega_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial \omega_{ji}^{(l)}} \qquad (6.11)$$

上式の右辺一つ目の因子を“デルタ”と呼ぶこととする

$$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial u_j^{(l)}} \qquad (6.12)$$

ユニットjからの信号がどれほど誤差に効いているか図る尺度

6.2　誤差逆伝播法

また，式(6.11)の右辺二つ目の因子$\frac{\partial u_j^{(l)}}{\partial \omega_{ji}^{(l)}}$は出力値$z^{(l-1)}$と等しい

⇒訓練サンプル$\boldsymbol{x}_n$を入力し，信号を伝播させれば自動的に算出される

$$\frac{\partial E_n}{\partial \omega_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \qquad (6.13)$$

勾配を“デルタ”と“出力”に分けること→第１ステップ

第２ステップ→デルタの満たす漸化式を見つける

デルタ…誤差関数をユニットjへの総入力$u_j^{(l)}$で微分したもの

図6.2より$u_j^{(l)}$は同ユニットの出力$z_j^{(l)}$に変換され次のユニットに入力される

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial u_j^{(l)}} = \sum_{k=0}^{d_{l+1}-1} \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \qquad (6.14)$$

6.2　誤差逆伝播法

式(6.14)右辺の一つ目の因子$\frac{\partial E_n}{\partial u_k^{(l+1)}}$は$\delta_k^{(l+1)}$だと考えられる

$$u_k^{(l+1)} = \sum_{j'} \omega_{ji}^{(l+1)} f\left(u_{j'}^{(l)}\right) \qquad (6.15)$$

(6.14), (6.15)よりデルタの逆伝播則が得られる

$$\delta_j^{(l)} = \sum_{k=0}^{d_{l+1}-1} \delta_k^{(l+1)} \omega_{kj}^{(l+1)} f'\left(u_{j'}^{(l)}\right) \qquad (6.16)$$

6.2　誤差逆伝播法

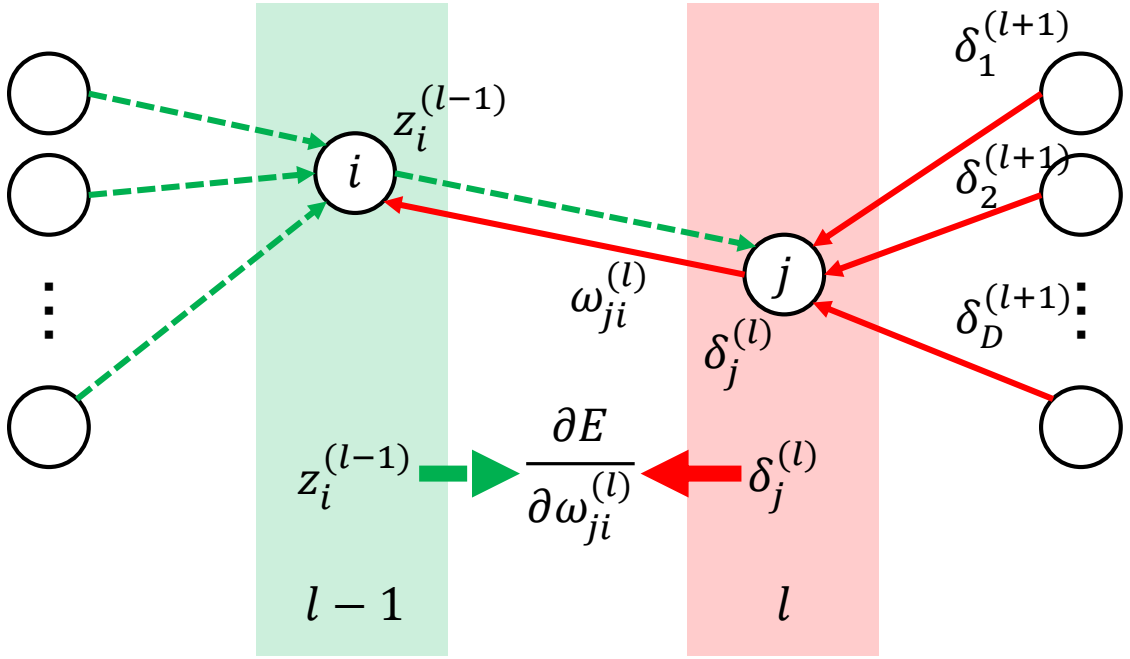


図6.3　誤差逆伝播による勾配計算の仕組み

入力側のl-1層にあるノードiからは順伝播で出力$z_i^{(l-1)}$がくる
出力側のl+1層から逆伝播でl層のノードjに対するデルタ$\delta_j^{(l)}$が与えられる
２つを組み合わせて重み$\omega_{ji}^{(l)}$に関する勾配が算出される

$$\frac{\partial E_n}{\partial \boldsymbol{\omega}^{(l)}} = \boldsymbol{\delta}^{(l)}\left(\boldsymbol{z}^{(l-1)}\right)^T \qquad (6.18)$$

まとめ

第４章　勾配降下法による学習

- ✓ 誤差を最小化する重みを見つける手法（モーメンタム法など）

第５章　深層学習の正則化

- ✓ 過学習を防ぐための手法（重み減衰・早期終了など）

第６章　誤差逆伝播法

- ✓ 誤差を逆方向へ伝えて，各層の重みを更新する手法

参考文献

瀧 雅人　『これならわかる深層学習　入門』　講談社

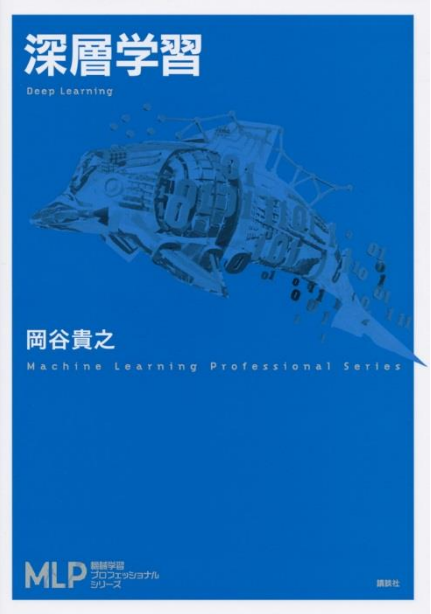
山崎 和博　「これから始める人のためのディープラーニング基礎講座」
https://www.slideshare.net/NVIDIAJapan/01-1000-nvdls18yamasaki

機械学習集中ゼミpart3

福田研M1　河井智弘

本発表で取り扱う内容

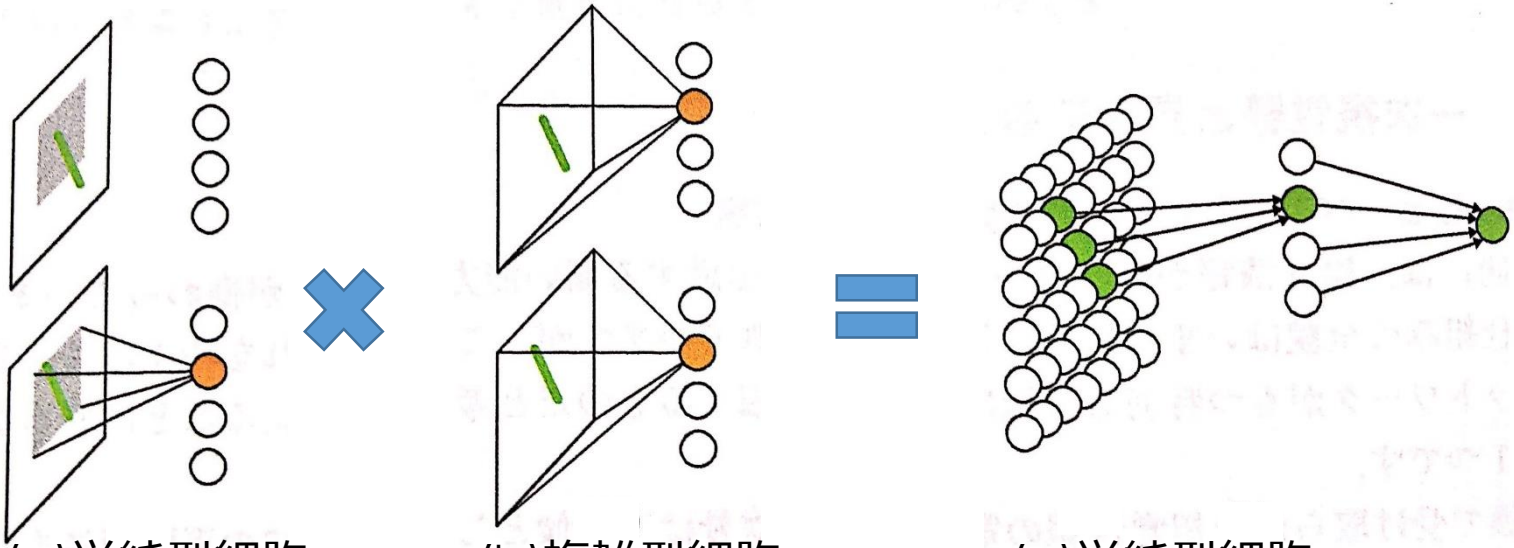
Chapter8: 畳み込みニューラルネット
(Convolutional Neural Network : CNN)
Chapter9: 再帰型ニューラルネット
(Reccurent Neural Network : RNN)



一次視覚野と畳み込み

- 画像認識をしたい
- 猫の視覚は以下の2種類の細胞により反応していることを発見

<左の四角領域(網膜)から視覚刺激を受け取った際のニューロンの変化>



(a)単純型細胞
⇒灰色領域でのみニューロンが発火
⇒需要野が狭い

(b)複雑型細胞
⇒需要野が広い

(c)単純型細胞
⇒(a)と(b)の組み合わせにより認識
⇒階層的

重み共有：結合の重みは各ユニット間で共通

⇒機械の画像認識では階層構造を表現するために畳み込みニューラルネットワークを導入しよう！

畳み込み層

畳み込み層：行列入力$z_{ij}^{(l-1)}$へフィルタ(画素値h_{pq})を作用させる

$$u_{ij}^{(l)} = \sum_{p,q=0}^{H-1} z_{i+p,j+q}^{(l-1)} h_{pq}$$

※ l：層

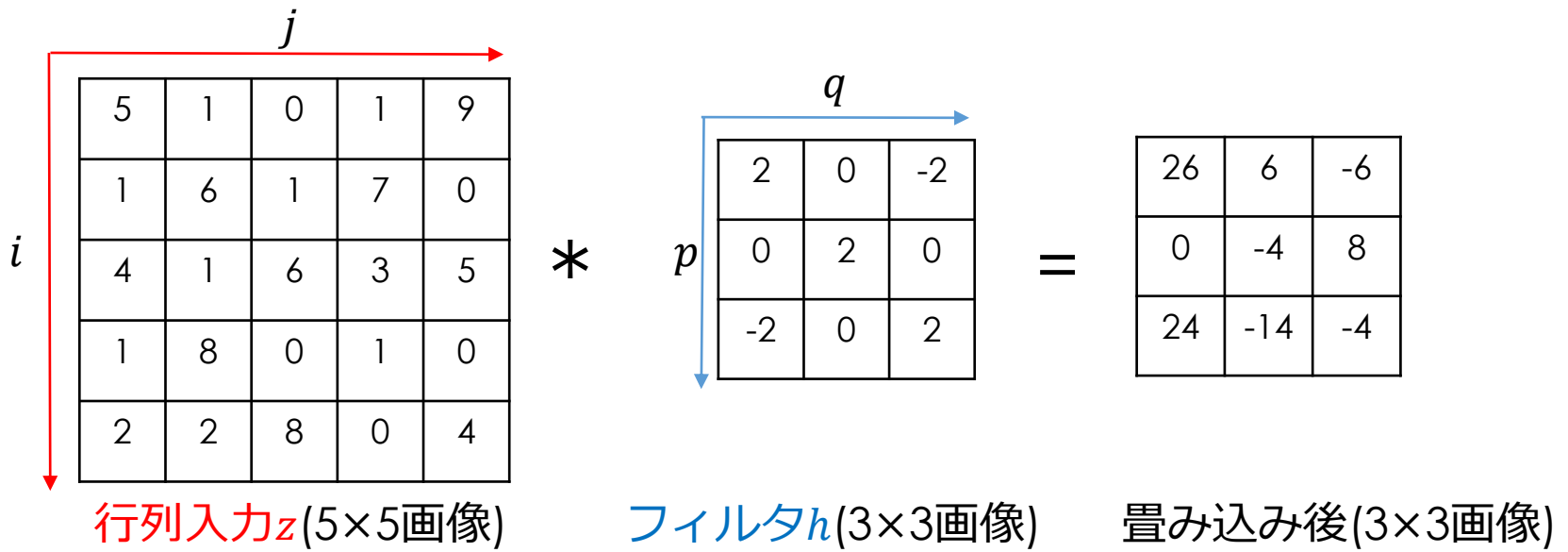
畳み込み後の画像サイズは
(W－H＋1)×(W－H＋1)

W:重み行列

※フィルタのサイズは元の画像からはみ出せない

W H

例：5×5画像への3×3フィルタの畳み込み



行列入力z(5×5画像)　フィルタh(3×3画像)　畳み込み後(3×3画像)

畳み込みの例

5	1	0	1	9
1	6	1	7	0
4	1	6	3	5
1	8	0	1	0
2	2	8	0	4

＊

2	0	-2
0	2	0
-2	0	2

＝

26	6	-6
0	-4	8
24	-14	-4

行列の内積
5×2+1×0+0×(-2)+…

5	1	0	1	9
1	6	1	7	0
4	1	6	3	5
1	8	0	1	0
2	2	8	0	4

＊

2	0	-2
0	2	0
-2	0	2

＝

26	6	-6
0	-4	8
24	-14	-4

行列の内積
1×2+0×0+1×(-2)+…

Channel

チャネル…画像データの各位置(i, j)に対して付与された情報の自由度

(例) m:チャネルの種類　…色など：m=0,1,…M-1を設定
k :チャネル内の情報…色の種類 k=0,1,…K-1を表現

作用を受けた画像

$$u_{ijm}^{(l)} = \sum_{k=0}^{K-1} \sum_{p,q=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqkm} + b_{ijm}$$

バイアス

畳み込み層の最終的な出力(特徴マップ)
；活性化関数fを作用させる

$$z_{ijm}^{(l)} = f(u_{ijm}^{(l)})$$

シグモイド関数 or ReLU関数 など…

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

$$f(u) = \begin{cases} u & \text{for } x > 0 \\ 0 & \text{for } x \leqq 0 \end{cases}$$

Stride

ストライドS；畳み込み計算を減らすために，画素をS-1個余分に飛ばしで
フィルタを動かしていくこと→画像を大雑把に理解できる

$$u_{ijm}^{(l)} = \sum_{k=0}^{K-1} \sum_{p,q=0}^{H-1} z_{Si+p,Sj+q,k}^{(l-1)} h_{pqkm} + b_{ijm}$$

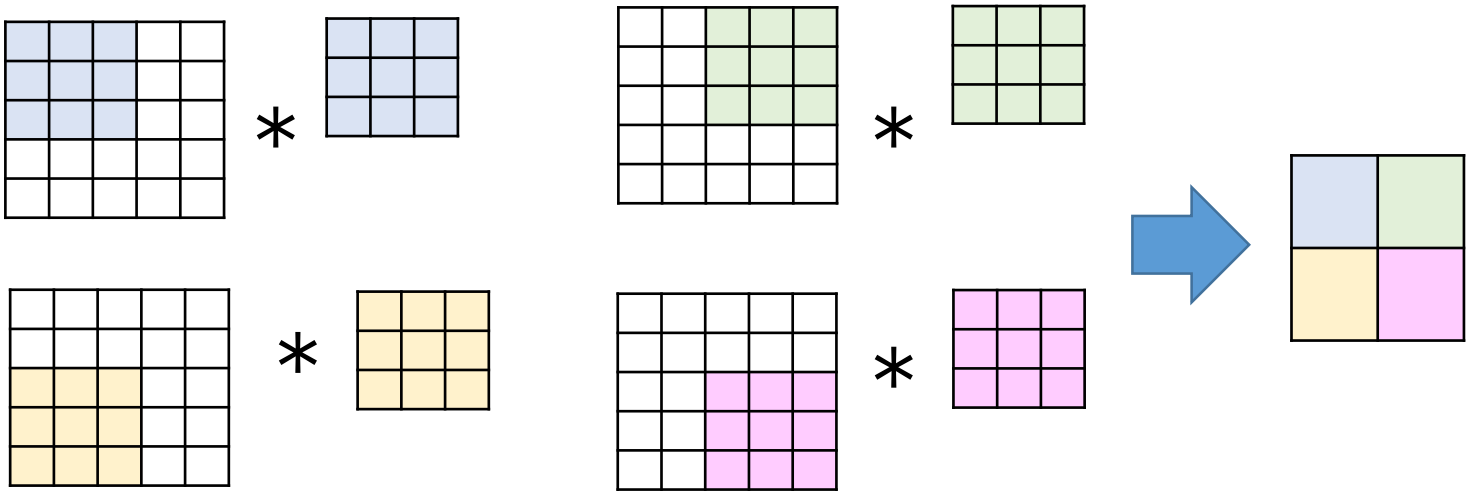
(8.8)式

畳み込み後の画像サイズは

$$\left\lfloor \frac{W-H}{S} \right\rfloor + 1 \times \left\lfloor \frac{W-H}{S} \right\rfloor + 1$$

※ $\lfloor x \rfloor$;床記号；xを超えない最大の整数

例：5×5画層へのストライドS=2での3×3フィルタの畳み込み



Padding

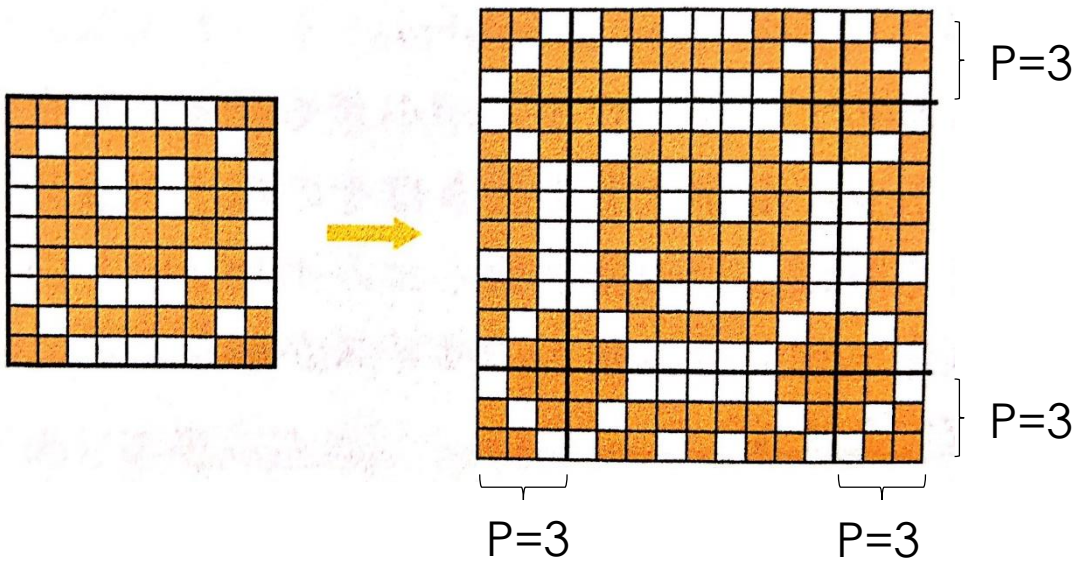
●ストライドにより計算量が減る分，画像サイズは小さくなる
⇒技術的に都合が良くない状況がある

パディングP…元の画像の周りに厚さPの淵を加える操作

畳み込み後の画像サイズは

$$\left[\frac{W-H+2P}{S}\right]+1 \times \left[\frac{W-H+2P}{S}\right]+1$$

例：9×9画像へのパディングP=3



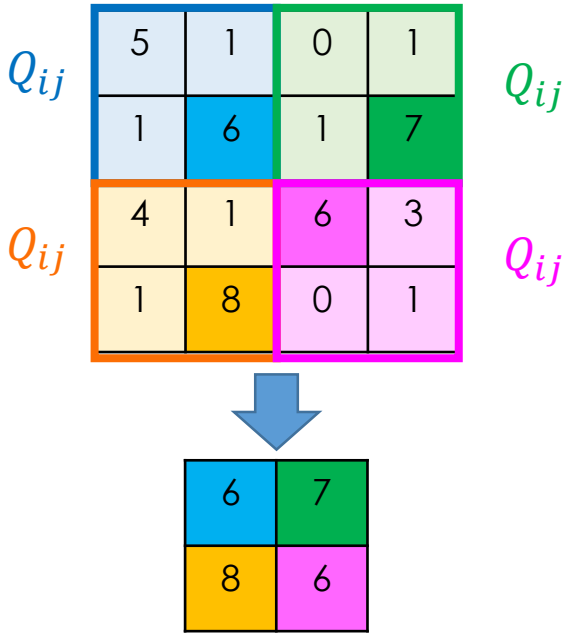
Pooling

■**プーリング**…画像サイズを縮小すること

Q_{ij}の範囲内で1つの代表データを採る

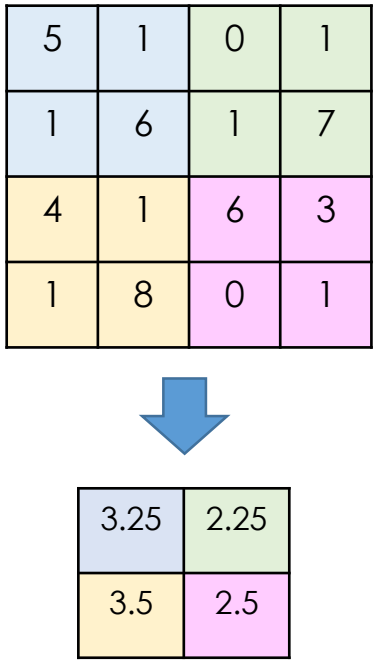
※H×H領域

●**最大プーリング** $u_{ijk}^{(l)} = \max_{(p,q)\in Q_{ij}} z_{pqk}^{(l-1)}$



⇒過学習に陥りやすい

●**平均プーリング** $u_{ijk}^{(l)} = \frac{1}{H^2} \sum_{(p,q)\in Q_{ij}} z_{pqk}^{(l-1)}$



⇒画像が粗くなる

Pooling / Unpooling

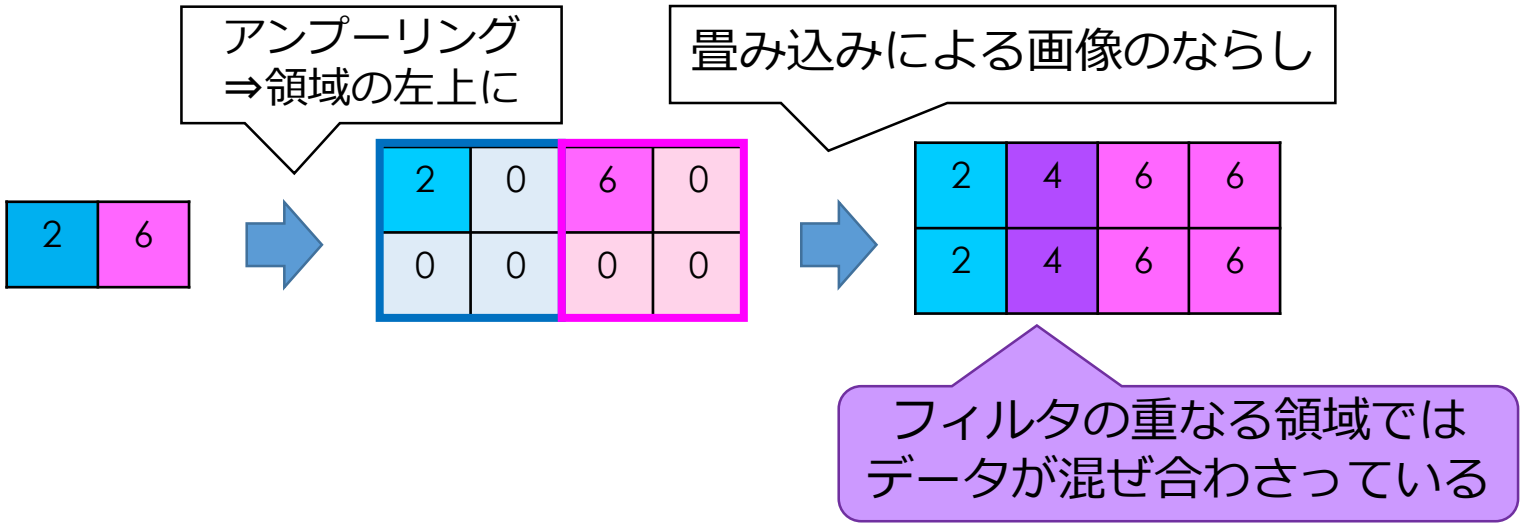
● **L^Pプーリング**：最大プーリングと平均プーリングの拡張

$$u_{ijk}^{(l)} = \left(\frac{1}{H^2} \sum_{(p,q)\in Q_{ij}} \left(z_{pqk}^{(l-1)} \right)^P \right)^{\frac{1}{P}}$$

P=1　：平均プーリング
P→∞：最大プーリング

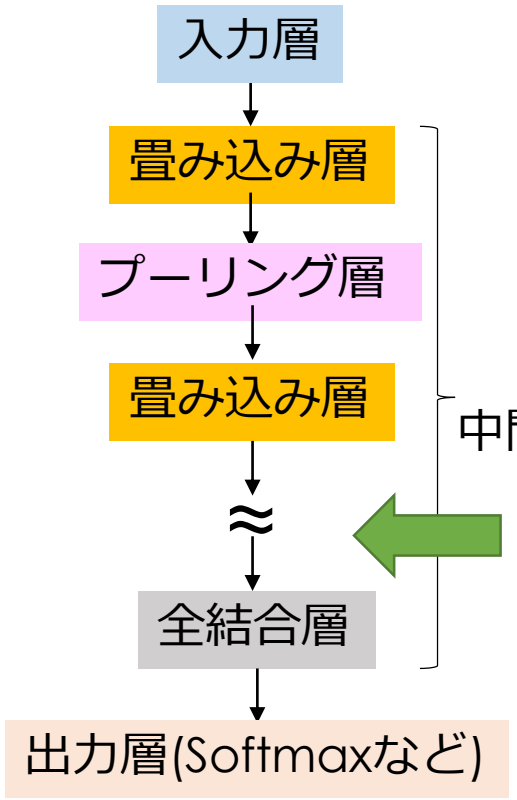
となる理由はp28補足へ

■ **脱畳み込み**：小さな画像から大きな画像を生成



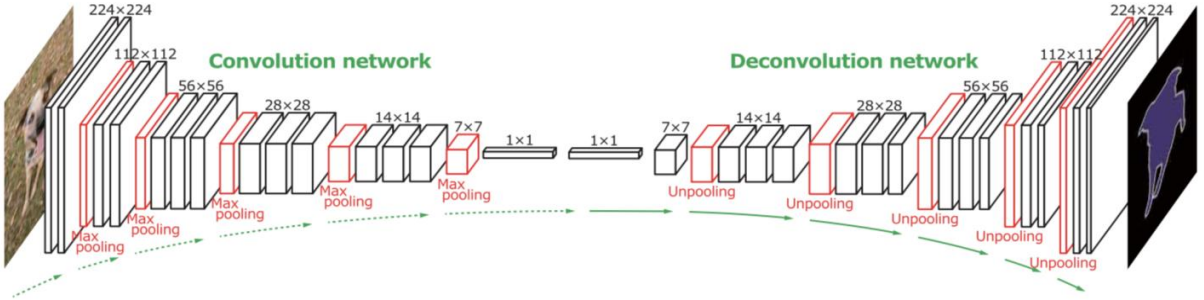
畳み込みニューラルネットまとめ

CNNは各層を繰り返し積み上げることで構成される



必要に応じて
活性化関数の層を挿入

前半：画素を縮小しながら計算⇒計算速度を速く
後半：欲しいサイズまで画素を広げる



参考：補足p29に正規化層について説明

●**転移学習**…学習済みパラメータを別の新たなタスクに対する初期値として設定
⇒学習がスムーズに進行する

CNNの誤差逆伝搬

●畳み込み層の誤差逆伝播
…デルタδの漸化式を作る

ストライドの導入式；(8.8)式

$$u_{ijm}^{(l)} = \sum_{k=0}^{K-1} \sum_{p,q=0}^{H-1} z_{Si+p,Sj+q,k}^{(l-1)} h_{pqkm} + b_{ijm}$$

$$\begin{cases} a = Si + p \\ b = Sj + q \end{cases} \text{とおく}$$

$$\delta_{abk}^{(l-1)} = \frac{\partial E}{\partial u_{abk}^{(l-1)}} = \sum_{i,j,m} \frac{\partial E}{\partial u_{ijm}^{(l)}} \frac{\partial u_{ijm}^{(l)}}{\partial z_{abk}^{(l-1)}} f'(u_{abk}^{(l-1)})$$

$\delta_{abk}^{(l)}$

$\frac{\partial u_{ijm}^{(l)}}{\partial z_{abk}^{(l-1)}} = \sum_{p=a-Si} \sum_{q=b-Sj} h_{pqkm}$ より，

$$\delta_{abk}^{(l-1)} = \sum_{i,j,m} \sum_{p=a-Si} \sum_{q=b-Sj} \delta_{abk}^{(l)} h_{pqkm} f'(u_{abk}^{(l-1)})$$

参考：デルタの逆伝播　(6.16)式

$$\delta_j^{(l)} = \frac{\partial E}{\partial u_j^{(l)}} = \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)} f'(u_j^{(l)})$$

CNNの誤差逆伝搬

●プーリング層の誤差逆伝播

誤差を順伝播で選択した画素の位置に分配、それ以外の画素には 0を設定

Unpoolingとの違い：代表値の元の場所を覚えている！

例：最大プーリング

$$w_{JI} = \begin{cases} 1 & \text{for} \max_{(p,q)\in Q_{ij}} z_{pqk}^{(l-1)} \\ 0 & otherwise \end{cases}$$

逆伝播

0	0	0	0
0	δ1	0	δ2
0	0	δ4	0
0	δ3	0	0

δ1	δ2
δ3	δ4

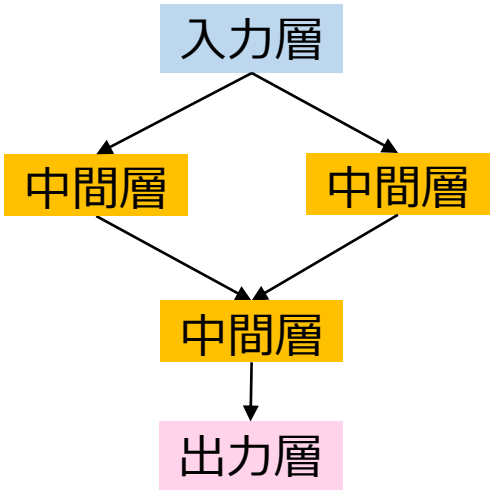
順伝播

5	1	0	1
1	6	1	7
4	1	6	3
1	8	0	1

6	7
8	6

再帰型ニューラルネット

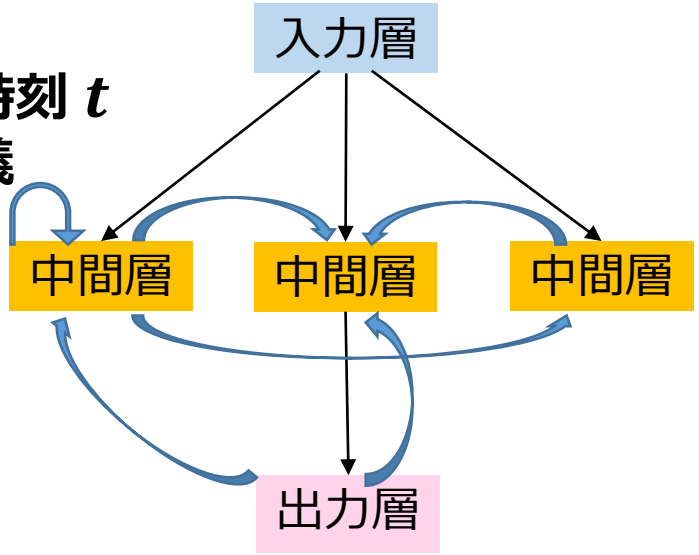
●8章：画像処理
⇒階層構造が定義できる
⇒順伝播型ニューラルネット



●9章：時系列データ
Ex)会話文
$\boldsymbol{x}^1$=This $\boldsymbol{x}^2$=is $\boldsymbol{x}^3$=an $\boldsymbol{x}^4$=appleときて
次の単語推測$\boldsymbol{x}^t$

辞書では…
$\boldsymbol{x}^t$=(0 …0 1 0 … 0)
a peer pen penetrate

⇒階層構造が
定義できない
⇒再帰型ニューラル
ネットを定義
⇒ユニット入力時刻 t
ごとに伝播を定義



再帰型ニューラルネット(RNN)

RNNの層は一方通行ではない⇒時間ごとに変化を追う

●RNNの伝播(入力層・中間層)

$$\boldsymbol{u}^t = \boldsymbol{W}^{in}\boldsymbol{x}^t + \boldsymbol{W}\boldsymbol{z}^{t-1} + \widetilde{\boldsymbol{W}}\boldsymbol{z}^{t-1,out}, \qquad \boldsymbol{z}^t = f(\boldsymbol{u}^t)$$

入力層⇒中間層

中間層⇒中間層

出力層⇒中間層

活性化関数

●RNNの伝播(出力層)

$$\boldsymbol{v}^t = \boldsymbol{W}^{out}\boldsymbol{z}^t, \qquad \boldsymbol{z}^{t,out} = f^{out}(\boldsymbol{v}^t)$$

$\boldsymbol{x}^t$:時刻tでの入力(入力層の出力)
$\boldsymbol{u}^t$:中間層のユニット入力
$\boldsymbol{z}^t$:中間層のユニット出力
$\boldsymbol{W}^{in}$:入力層・中間層の帰還経路の重み
$\boldsymbol{W}$:中間層間の帰還経路の重み
$\widetilde{\boldsymbol{W}}$:出力層から中間層への帰還経路の重み

$\boldsymbol{v}^t$:出力層へのユニット入力
$\boldsymbol{z}^{t,out}$:出力層のユニット出力
$\boldsymbol{W}^{out}$:中間層と出力層の重み

実時間リカレント学習法(Real Time Reccurent Learning)

● 誤差の逆伝播計算方法①RTRL法

勾配降下法より，

$$\Delta w_{rs}^{t} = -\eta \frac{\partial E^{t}(\boldsymbol{w})}{\partial w_{rs}} = -\eta \sum_{k} \frac{\partial E^{t}(\boldsymbol{w})}{\partial y^{t}(\boldsymbol{w})} \frac{\partial y^{t}(\boldsymbol{w})}{\partial w_{rs}}$$

注目

※ここでは簡単のため，
クロネッカーのデルタ$\delta_{r,k} = \begin{cases} 0 \\ 1 \end{cases}$
で表される信号を考える

※簡単のために以下のユニットの組み合わせを表す

i：⇔入力層
j：⇔中間層
k：⇔出力層
r：⇔中間層or出力層
s：⇔全ての層

※ j'：他の中間層を指す

pを時間方向に順次仮定する漸化式

(導出はp30補足へ)

$$p_{rs}^{k}(t) = \frac{\partial y^{t}(\boldsymbol{w})}{\partial w_{rs}} = f^{out\prime}(v_{k}^{t}) \left(\delta_{r,k} z_{s}^{t} + \sum_{j} w_{kj}^{out} p_{rs}^{j}(t) \right)$$

：出力層

$$p_{rs}^{j}(t) = \frac{\partial z_{j}^{t}(\boldsymbol{w})}{\partial w_{rs}} = f'(u_{j}^{t}) \left(\delta_{r,j} x_{s}^{t} + \delta_{r,j} z_{s}^{t-1} + \sum_{j} w_{jj'} p_{rs}^{j'}(\mathrm{t}-1) \right)$$

入力層の信号

$t-1$時の挙動

：中間層

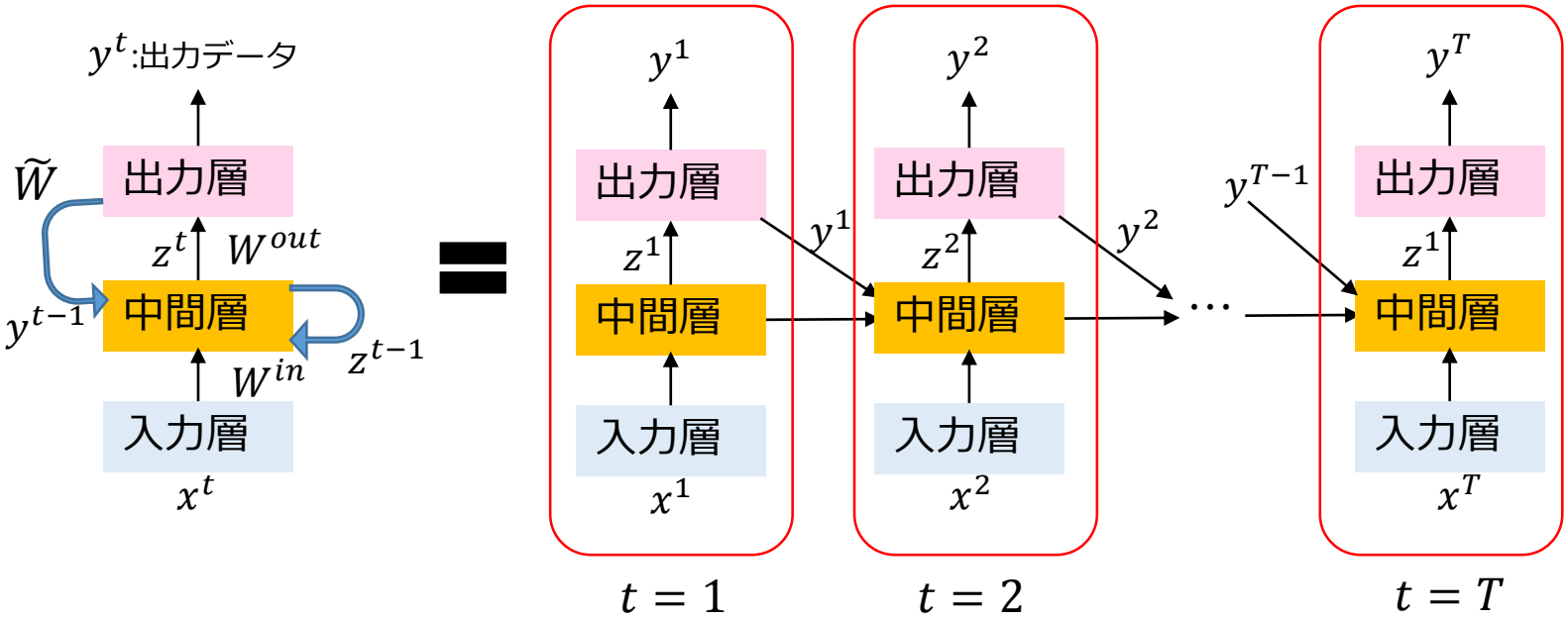
RTRL法では：得られたpを使い，各時刻で勾配降下法を行う

◎過去の情報の保持が少ないのでメモリ消費が少ない
×計算量は多い

通時的誤差逆伝播法(Back Propagation Through Time)

● 誤差の逆伝播計算方法②BPTT法

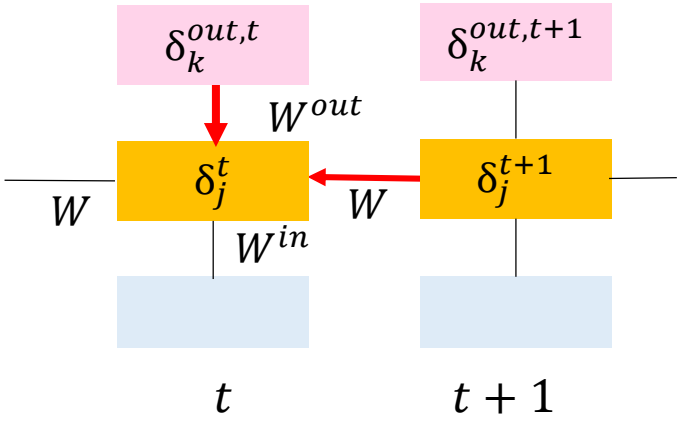
⇒時間ごとに層を設定するようなもの
⇒ループが無いので性質の良いグラフ



通時的誤差逆伝播法(Back Propagation Through Time)

●BPTT法

デルタの逆伝播⇒$\delta_k^{out,t}$，δ_j^{t+1}が影響



デルタの逆伝播：6章 (6.16)式

$$\delta_j^{(l)} = \frac{\partial E}{\partial u_j^{(l)}} = \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)} f'(u_j^{(l)})$$ より，

RNNの逆伝播　(9.17)式：

$$\delta_j^t = \left(\sum_k w_{kj}^{out} \delta_k^{out,t} + \sum_{j'} w_{jj'} \delta_{j'}^{t+1} \right) f'(u_j^t)$$

$\delta_k^{out,t}$はRNNの出力$\boldsymbol{y}^t$(順伝播の結果)と
目標出力$\boldsymbol{d}^t$から誤差関数Eにより計算

▽誤差Eの，各層の重みによる微分は

入力層　$$\frac{\partial E}{\partial w_{ji}^{in}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{ji}^{in}} = \sum_{t=1}^{T} \delta_j^t x_i^t$$

出力層　$$\frac{\partial E}{\partial w_{ji}^{out}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{ji}^{out}} = \sum_{t=1}^{T} \delta_j^{out,t} z_i^t$$

中間層間　$$\frac{\partial E}{\partial w_{jj'}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{jj'}} = \sum_{t=1}^{T} \delta_j^t z_{j'}^{t-1}$$

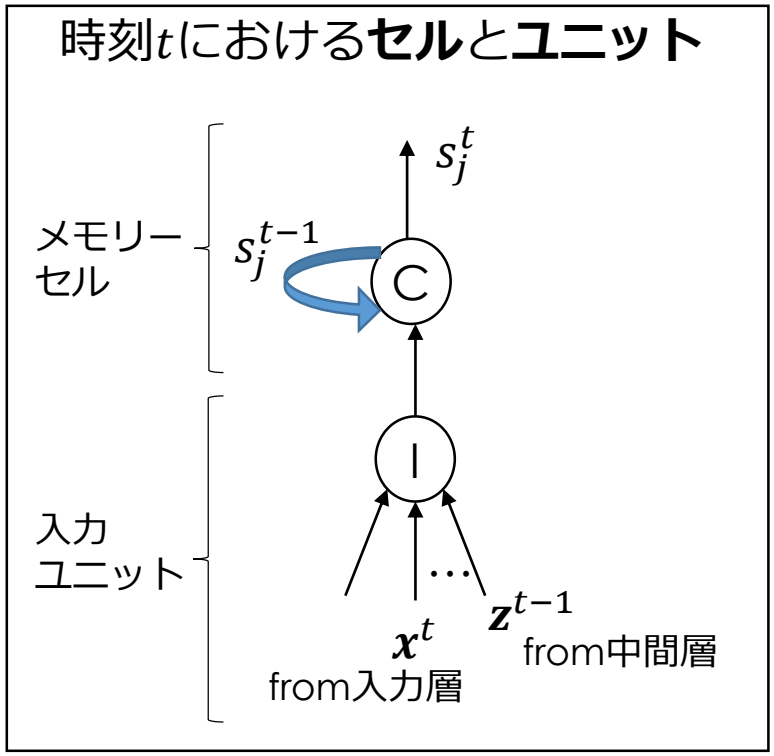
⇒各時刻で勾配降下法を行う

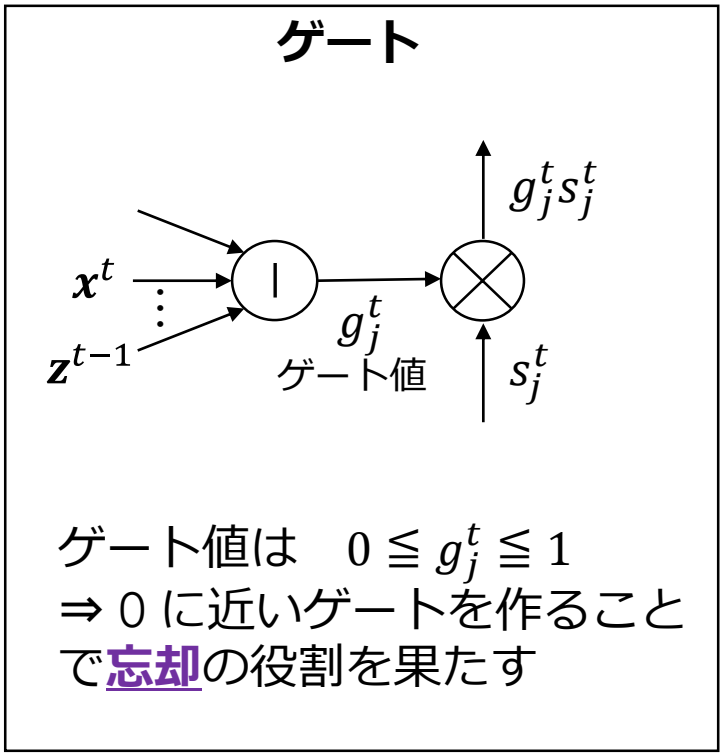
◎計算量が少ない

長・短期記憶(Long Short-Term Memory)

RNNは，ループが回り続けるので，学習が上手くいかないことも
⇒十分情報を活用し終えた過去データは消去したい！

LSTM：RNNの中間層ユニットをメモリー・ユニットで置き換えたもの.



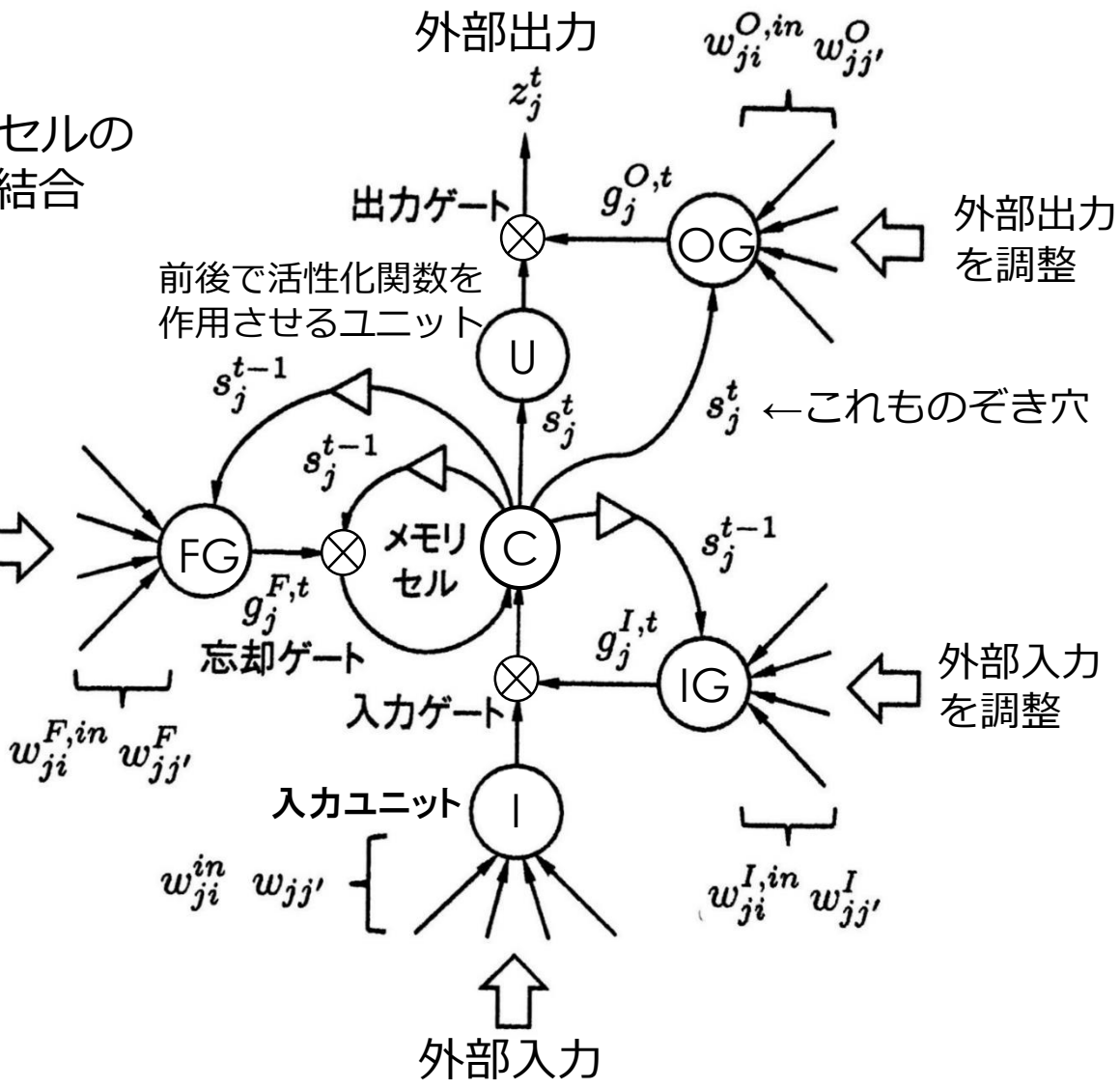


長・短期記憶(Long Short-Term Memory)

● LSTMの構造

のぞき穴(peephole)：セルの出力→ゲート へ流れる結合

◁：遅れて伝わるのぞき穴



長・短期記憶(Long Short-Term Memory)

●ゲート値について

忘却ゲート

活性化関数

$$g_j^{F,t} = f(u_j^{F,t}) = f\left(\sum_i w_{ji}^{F,in} x_i^t + \sum_i w_{jj'}^{F} z_{j'}^{t-1} + w_j^F s_j^{t-1}\right)$$

忘却ゲートの入力層

忘却ゲートの中間層

メモリセルからゲートの値を決めるユニット

入力ゲート

$$g_j^{I,t} = f(u_j^{I,t}) = f\left(\sum_i w_{ji}^{I,in} x_i^t + \sum_i w_{jj'}^{I} z_{j'}^{t-1} + w_j^I s_j^{t-1}\right)$$

出力ゲート

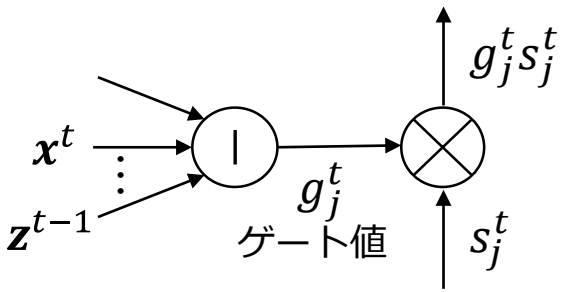
$$g_j^{O,t} = f(u_j^{O,t}) = f\left(\sum_i w_{ji}^{O,in} x_i^t + \sum_i w_{jj'}^{O} z_{j'}^{t-1} + w_j^O s_j^{t-1}\right)$$

メモリー・セルの状態変化

$$s_j^t = g_j^{I,t} f(u_j^t) + g_j^{F,t} s_j^{t-1}$$

最終出力は

$$z_j^t = g_j^{O,t} f(s_j^t)$$



LSTMの逆伝播

● 誤差逆伝播のデルタ

連鎖則より

前後で活性化関数を作用させるユニットU

$$\delta_j^{U,t} = g_j^{O,t} f'(u_j^{O,t}) \left(\sum_k w_{kj}^{out} \delta_k^{out,t} + \sum_{j'} w_{jj'} \delta_{j'}^{t+1} \right)$$

$$\delta_j^{U,t} = \frac{\partial E}{\partial u_j^{U,t}} = \sum_k \frac{\partial v_k^t}{\partial u_j^{U,t}} \frac{\partial E}{\partial v_k^t} + \sum_{j'} \frac{\partial u_{j'}^{t+1}}{\partial u_j^{U,t}} \frac{\partial E}{\partial u_{j'}^{t+1}} \text{ より}$$

出力ゲート OG

$$\delta_j^{O,t} = f'(u_j^{O,t}) f(s_j^t) \left(\sum_k w_{kj}^{out} \delta_k^{out,t} + \sum_{j'} w_{jj'} \delta_{j'}^{t+1} \right)$$

忘却ゲート FG

$$\delta_j^{F,t} = f'(u_j^{F,t}) s_j^{t-1} \delta_j^{C,t}$$

入力ゲート IG

$$\delta_j^{I,t} = f'(u_j^{I,t+1}) f(u_j^t) \delta_j^{C,t}$$

入力ユニットI

$$\delta_j^t = \frac{\partial E}{\partial u_j^t} = g^{I,t} f'(u_j^t) \delta_j^{C,t}$$

※通常，この活性化関数fにはシグモイド関数を用いる

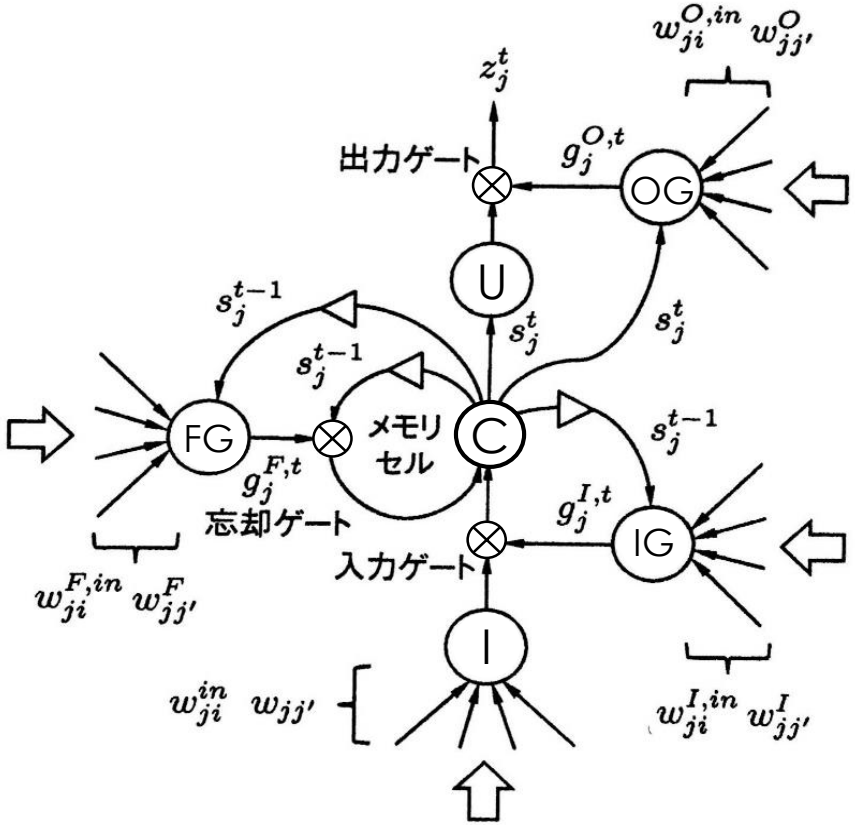
LSTMの逆伝播

セルに関するデルタは

$$\delta_j^{C,t} = \delta_j^{U,t} + g_j^{F,t+1}\delta_j^{C,t+1} + w_j^O\delta_j^{O,t} + w_j^F\delta_j^{I,t+1} + w_j^I\delta_j^{F,t+1}$$

- $\delta_j^{U,t}$：外部出力方面 →Uへ
- $g_j^{F,t+1}\delta_j^{C,t+1}$：セルC自身への帰還
- $w_j^O\delta_j^{O,t}$：出力ゲートOG向けののぞき穴
- $w_j^F\delta_j^{I,t+1}$：入力ゲートIG向けののぞき穴
- $w_j^I\delta_j^{F,t+1}$：出力ゲートFG向けののぞき穴

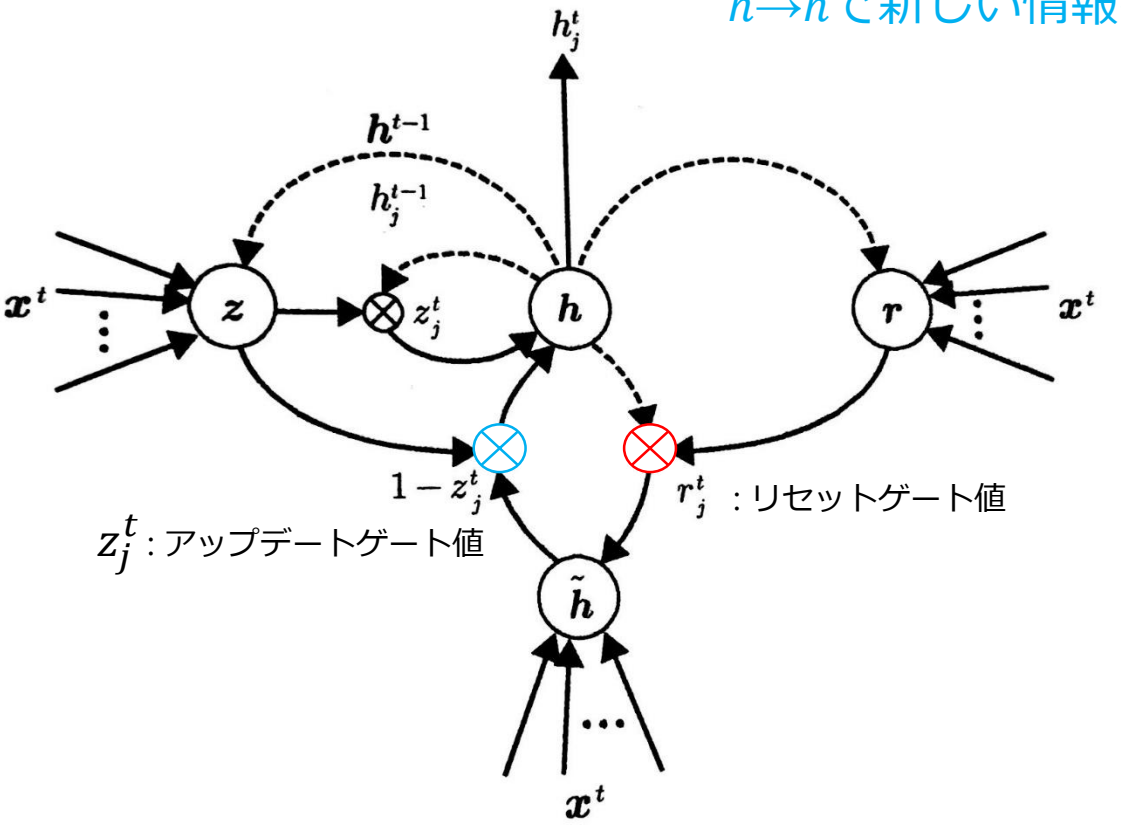
(連鎖則より求まる)



ゲート付き再帰的ユニット (Gated Recurrent Unit)

GRU：LSTMとは違ってメモリセルを持たず，更新・リセットゲートを用いることで入出力情報の保持・忘却を調整する　⇒シンプル，計算量低減

$h \rightarrow \tilde{h}$で過去の情報を消去
$\tilde{h} \rightarrow h$で新しい情報を取り込む



リセットゲート値は
$$\boldsymbol{r}^t = f(\boldsymbol{W}_r \boldsymbol{x}^t + \boldsymbol{U}_r \boldsymbol{h}^{t-1})$$

アップデートゲート値は
$$\boldsymbol{z}^t = f(\boldsymbol{W}_z \boldsymbol{x}^t + \boldsymbol{U}_z \boldsymbol{h}^{t-1})$$

ユニットの活性は
$$\boldsymbol{h}^t = \boldsymbol{z}^t \boldsymbol{h}^{t-1} + (\boldsymbol{1} - \boldsymbol{z}^t)\tilde{\boldsymbol{h}}^t$$

$$\tilde{\boldsymbol{h}}^t = f(\boldsymbol{W}\boldsymbol{x}^t + \boldsymbol{U}\boldsymbol{r}^t \boldsymbol{h}^{t-1})$$

$\boldsymbol{r}^t$の値により
$\tilde{\boldsymbol{h}}^t$を減衰させる

$\boldsymbol{W}$，$\boldsymbol{U}$：GRUの重みパラメータ

RNNと自然言語処理

入力文と翻訳先の出力文の長さが異なる場合

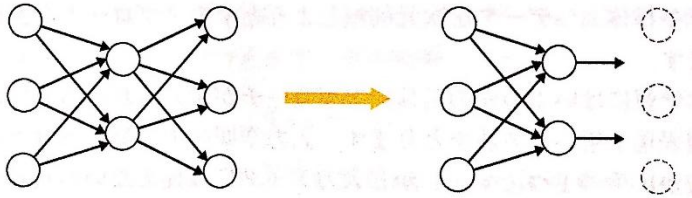
符号化器：RNNを用いて入力文全体$x^1, x^2, \ldots, x^T$を文脈c (context) に変換

$$\boldsymbol{c} = q(\boldsymbol{z}^1, \ldots, \boldsymbol{z}^T) \ , \quad \boldsymbol{z}^t = f(\boldsymbol{W}^{in}\boldsymbol{x}^t + \boldsymbol{W}\boldsymbol{z}^{t-1})$$

複号化器：contextを受け取り，訳文を決めるための確率分布を与える

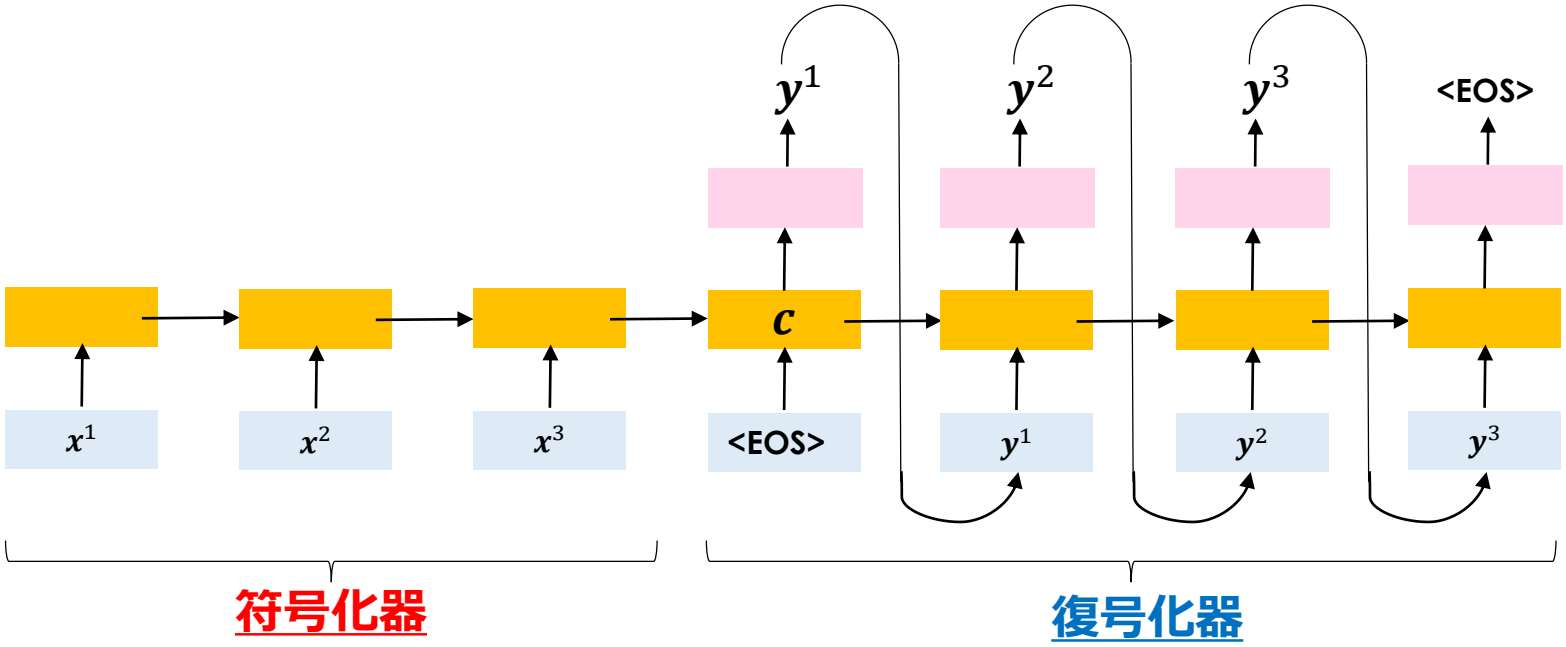
出力：　　$P(\boldsymbol{y}^t = \boldsymbol{y} | \boldsymbol{y}^1 \ldots \boldsymbol{y}^{t-1}, \boldsymbol{c})$

※注意：7章；砂時計型ニューラルネットにおける自己符号化器(下図)とは別物



Seq2Seq (sequence to sequence) 学習

RNNを用いてcontextをつくり，翻訳文の単語を一文字ずつ生成する．



最尤法

$$P(y^1 \dots y^{T'} | x^1 \dots x^T) = \prod_{t=1}^{T'} P(y^t | y^1 \dots y^{t-1}, c)$$

⇒対数尤度関数(今回の誤差関数)の最小化により翻訳推定を行う

参考文献

●講談社「これならわかる深層学習入門」

●講談社「深層学習」

●MathWorks「たたみ込みニューラルネットワークの層の指定」

https://jp.mathworks.com/help/nnet/ug/layers-of-a-convolutional-neural-network.html

●Qiita「誤差逆伝播法のノート」

https://qiita.com/eijian/items/c947fb6b5e7a49858fb4

●WANTEDLY「Fully Convolutional Networks を用いた自動線画生成に関する研究の紹介」by 株式会社クロスコンパス

https://www.wantedly.com/companies/xcompass/post_articles/72450

補足：L^Pプーリング でP→∞のとき 最大プーリングと等価になる証明

- **L^Pプーリング**：最大プーリングと平均プーリングの拡張

$$u_{ijk}^{(l)} = \left(\frac{1}{H^2} \sum_{(p,q)\in Q_{ij}} \left(z_{pqk}^{(l-1)} \right)^P \right)^{\frac{1}{P}}$$

P=1　：平均プーリング
P→∞：最大プーリング

$z_{p^*q^*} = max_{(p,q)} z_{pq}$　とすると，

$$\left(\sum_{(p,q)\in Q_{ij}} \left(z_{pq} \right)^P \right)^{\frac{1}{P}} = z_{p^*q^*} \left(1 + \sum_{(p,q)\neq(p^*,q^*)} \left(\frac{z_{pq}}{z_{p^*q^*}} \right)^P \right)^{\frac{1}{P}}$$

において $\frac{z_{pq}}{z_{p^*q^*}} < 1$より，P→∞ のとき右辺は $z_{p^*q^*} = max_{(p,q)} z_{pq}$ に収束

CNNの補足：正規化層

●局所コントラスト正規化層(Local Contrast Normalization : LCN Layer)
データ集合の中から1枚の画像に対してのみ正規化
…各チャネルごとに行う．最近はあまり要らない

平均的な画素値 $\overline{x_{ij}} = \sum_{p,q} w_{pq} x_{i+p,j+q}$

画素値の分散 $\sigma_{ij}^2 = \sum_{p,q} w_{pq} \left(x_{i+p,j+q} - \overline{x_{ij}}\right)^2$

減算正規化 $z_{ij} = x_{ij} - \overline{x_{ij}}$

除算正規化 $z_{ij} = \dfrac{x_{ij} - \overline{x_{ij}}}{\sigma_{ij}}$

●局所応答正規化層(Local Response Normalization : LRN Layer)
一定数の隣接し合うチャネルからの要素を使用して正規化　(2012，Alexnet)

$$z_{ijk}^{(l)} = \frac{z_{ijk}^{(l=1)}}{\left(c + \alpha \sum_{m=\max\left(0,k-\frac{N}{2}\right)}^{\max\left(K-1,k+\frac{N}{2}\right)} \left(z_{ijm}^{(l-1)}\right)^2\right)^{\beta}}$$

チャネルごとの
出力データの平方和

⇒輝度だけに対する規格化になっている

c, α, β, N: ハイパーパラメータ

RTRLの補足：$p_{rs}(t)$の導出方法

$$\frac{\partial E^t(\boldsymbol{w})}{\partial w_{rs}} = \sum_k \frac{\partial E^t(\boldsymbol{w})}{\partial y^t(\boldsymbol{w})} \frac{\partial y^t(\boldsymbol{w})}{\partial w_{rs}}$$

$$p_{rs}^k(t) = \frac{\partial y^t(\boldsymbol{w})}{\partial w_{rs}} = f^{out\prime}(v_k^t) \frac{\partial}{\partial w_{rs}} \sum_j w_{kj}^{out} z_j^t$$

出力層の活性化関数の微分

中間層の伝播の合計

$$= f^{out\prime}(v_k^t) \left(\delta_{r,k} z_s^t + \sum_j w_{kj}^{out} p_{rs}^j(t) \right)$$

クロネッカーのデルタ $\delta_{r,k} = \begin{cases} 0 \\ 1 \end{cases}$

同様にして，

$$p_{rs}^j(t) = \frac{\partial z_j^t(\boldsymbol{w})}{\partial w_{rs}}$$

$$= f'(u_j^t) \frac{\partial}{\partial w_{rs}} \left(\sum_i w_{ji}^{in} x_i^t + \sum_{j\prime} w_{jj\prime} z_{j\prime}^{t-1} \right)$$

$$= f'(u_j^t) \frac{\partial}{\partial w_{rs}} \left(\delta_{r,j} x_s^t + \delta_{r,j} z_s^{t-1} + \sum_j w_{jj}' p_{rs}^{j'}(\mathrm{t}-1) \right)$$

※簡単のために以下のユニットの組み合わせを表す
i：⇔入力層
j：⇔中間層
k: ⇔出力層
r：⇔中間層or出力層
s：⇔全ての層

※ j'：他の中間層を指す

参考文献

＊「関係データ学習」 石黒勝彦・林浩平　講談社

＊「続・わかりやすい　パターン認識」 石井健一郎・上田修功　オーム社