

深層学習最前線

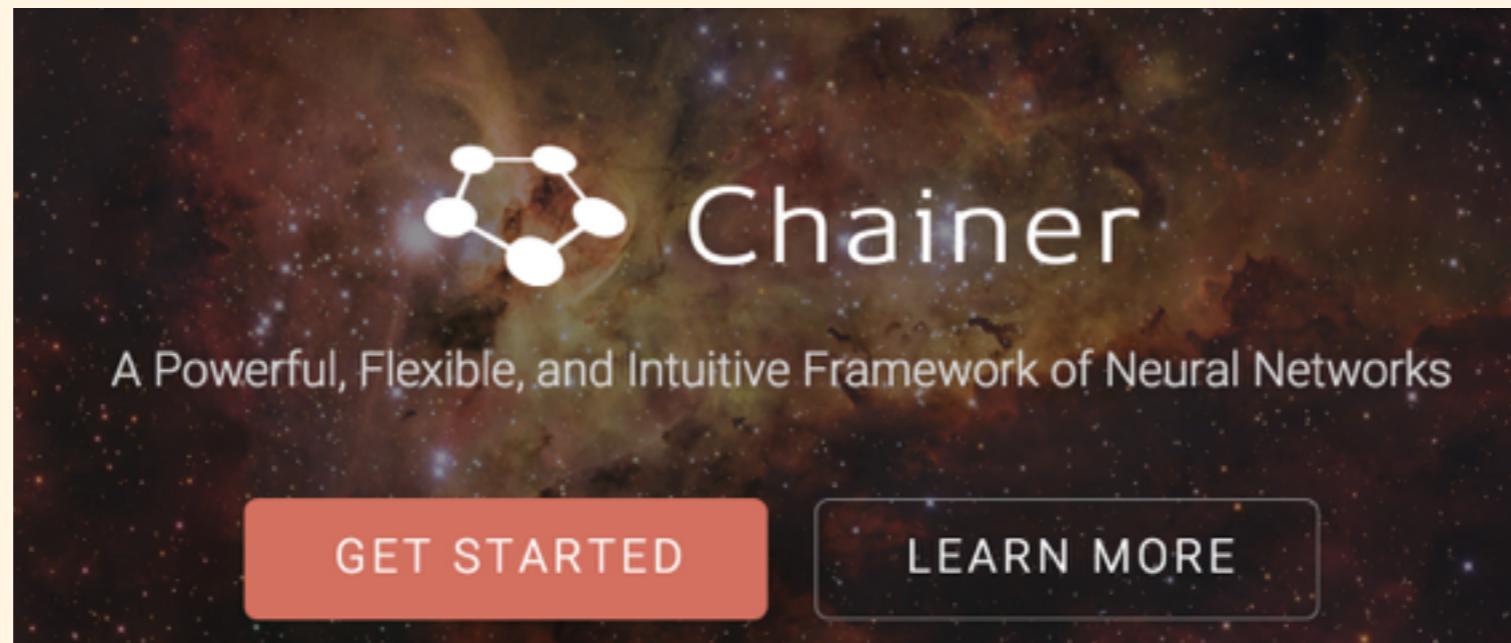
データサイエンス養成読本
機械学習入門編

第1部 特集4 (p.67)

Presenter:
Makoto Otsuka @makotsuka55

このセクションの著者

- 得居 誠也 (Tokui Seiya) @beam2d
- Preferred Networksの中の人
- Chainerを作った人



第1章：準備

問題設定とニューラルネットの基本 (p.68)

第2章：多層パーセプトロンの書き方

構成要素の理解 (p.71)

第3章：ニューラルネットの学習方法

様々な最適化テクニック (p.74)

第4章：画像認識のためのアーキテクチャ

畳込みネット入門 (p.78)

第1章：準備

問題設定とニューラルネットの基本 (p.68)

第2章：多層パーセプトロンの書き方

構成要素の理解 (p.71)

第3章：ニューラルネットの学習方法

様々な最適化テクニック (p.74)

第4章：画像認識のためのアーキテクチャ

畳込みネット入門 (p.78)

分類問題

- なぜ分類問題？
 - 深層学習は機械学習が扱うあらゆる問題に適用可能
 - 基本的な分類問題でも、深層学習のエッセンスは十分に学べる
- 分類問題とは？
 - 1つの入力データに対して複数のラベルから「1つだけ」選んでラベルを振る
 - ラベルが2個なら二値分類問題。それより多いなら多値分類問題。
- 分類問題の目的
 - テストデータセットの入力データに正しいラベルを振ること

全ラベル（知ってる）

二值分類：未・申

多值分類：子・丑・寅・卯・辰・巳・午・未・申・酉・戌・亥

トレーニングデータセット

入力
(画像)



...

出力
(ラベル)

申

申

申

未

未

未

...

テストデータセット

入力
(画像)



...

出力
(ラベル)

?

?

?

?

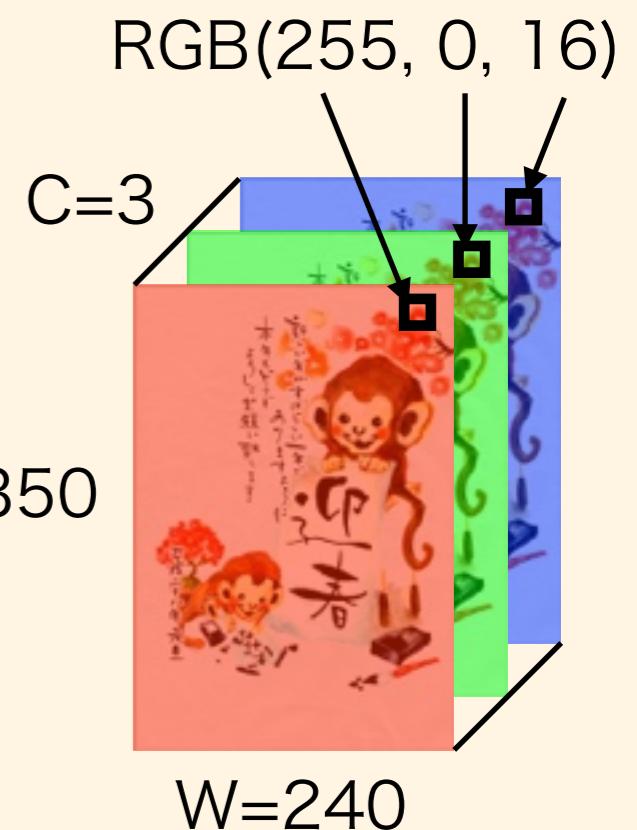
?

?

...

データの数値表現

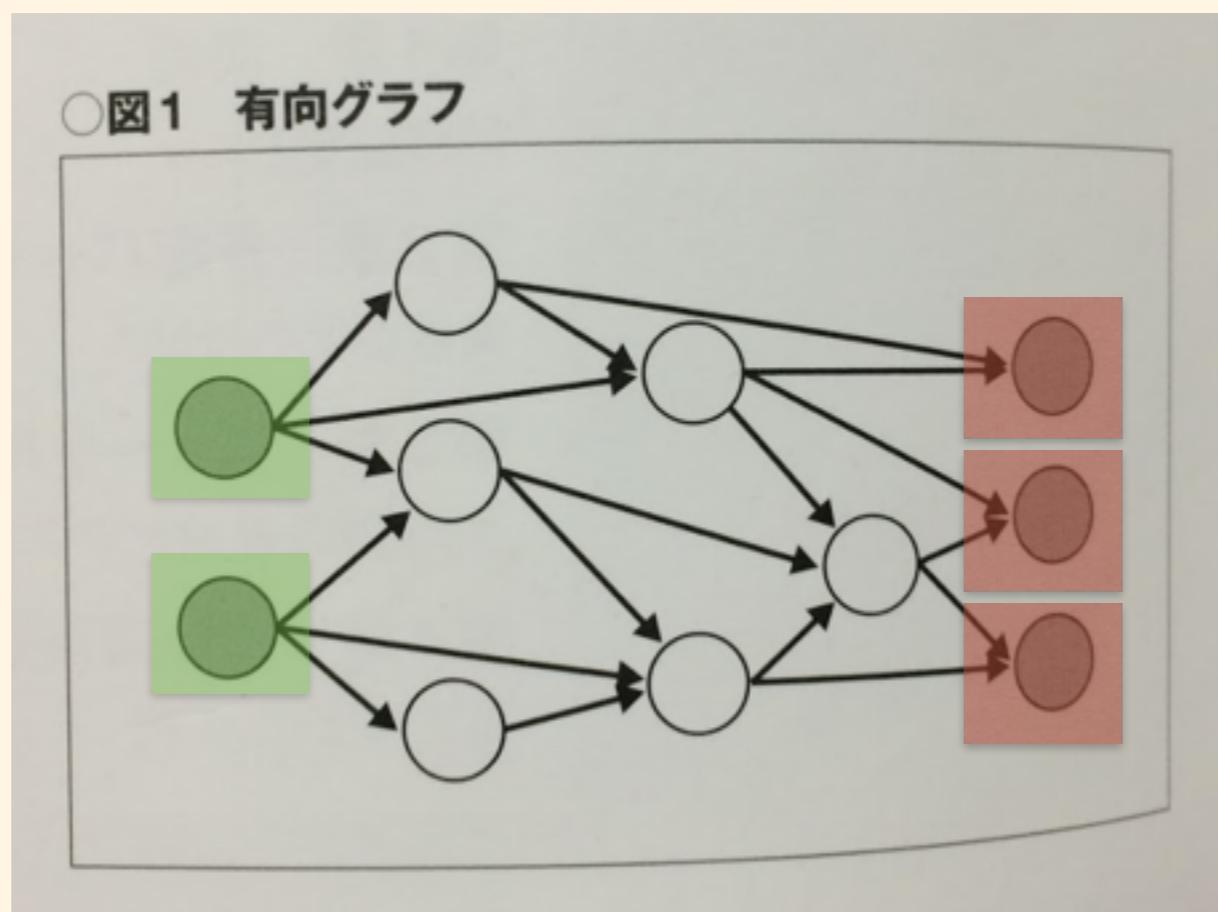
- 画像：画素値の列



- 右上にあるカラーの年賀状 1 枚は
色のチャネル数 (3) × 縦ピクセル数 (350) ×
横ピクセル数 (240) の3次元テンソルで表現できる
- テンソルの要素は [0, 255]
- 音声：フレームごとのスペクトル分布
- 自然言語：Bag-of-Wordsベクトル

ニューラルネット

- 有向グラフで表現される入力と出力の関係性を表すパラメトリックな関数 $y = g(x, \theta)$



入力ユニット

(入力 x の要素を表現)

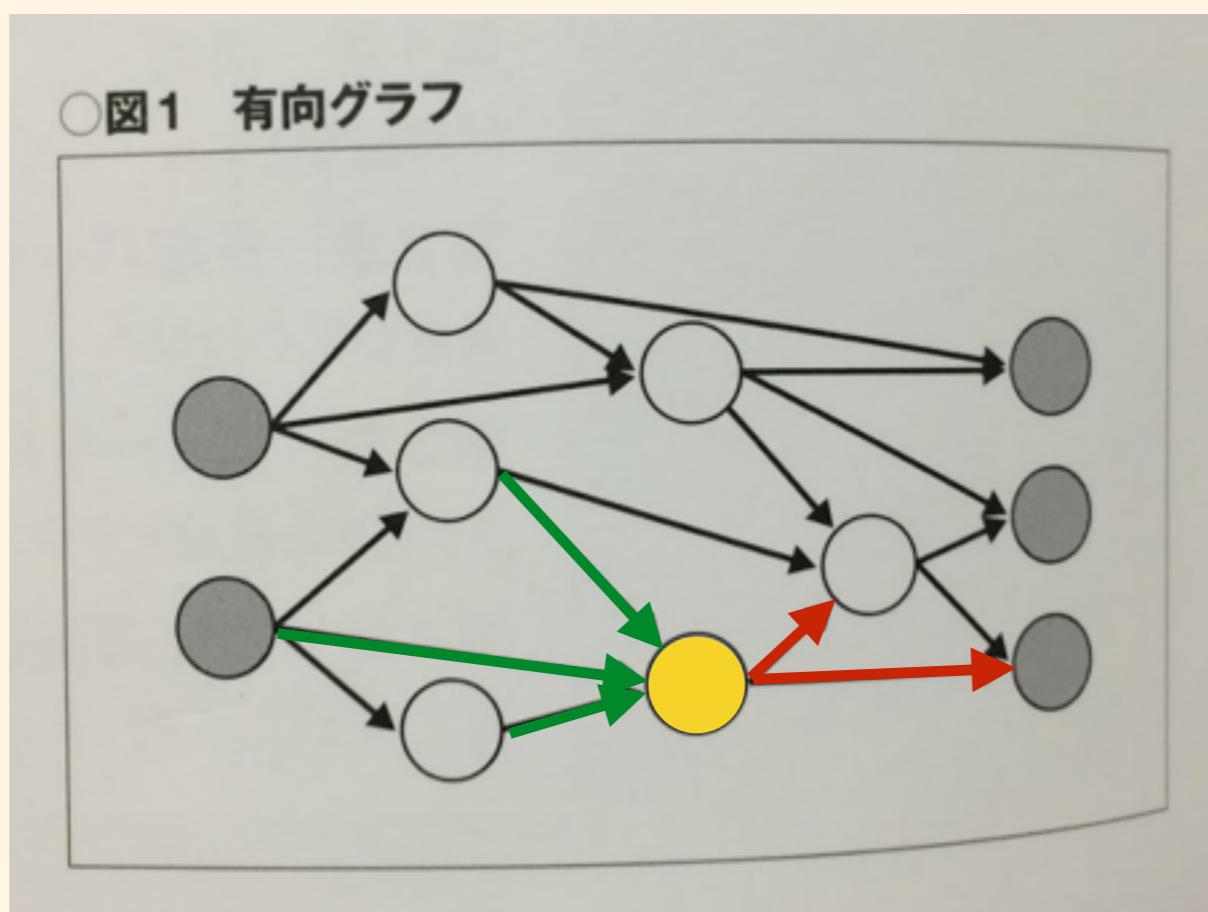
出力ユニット

(入力 y の要素を表現)

○ ユニットの入力



○ ユニットの出力



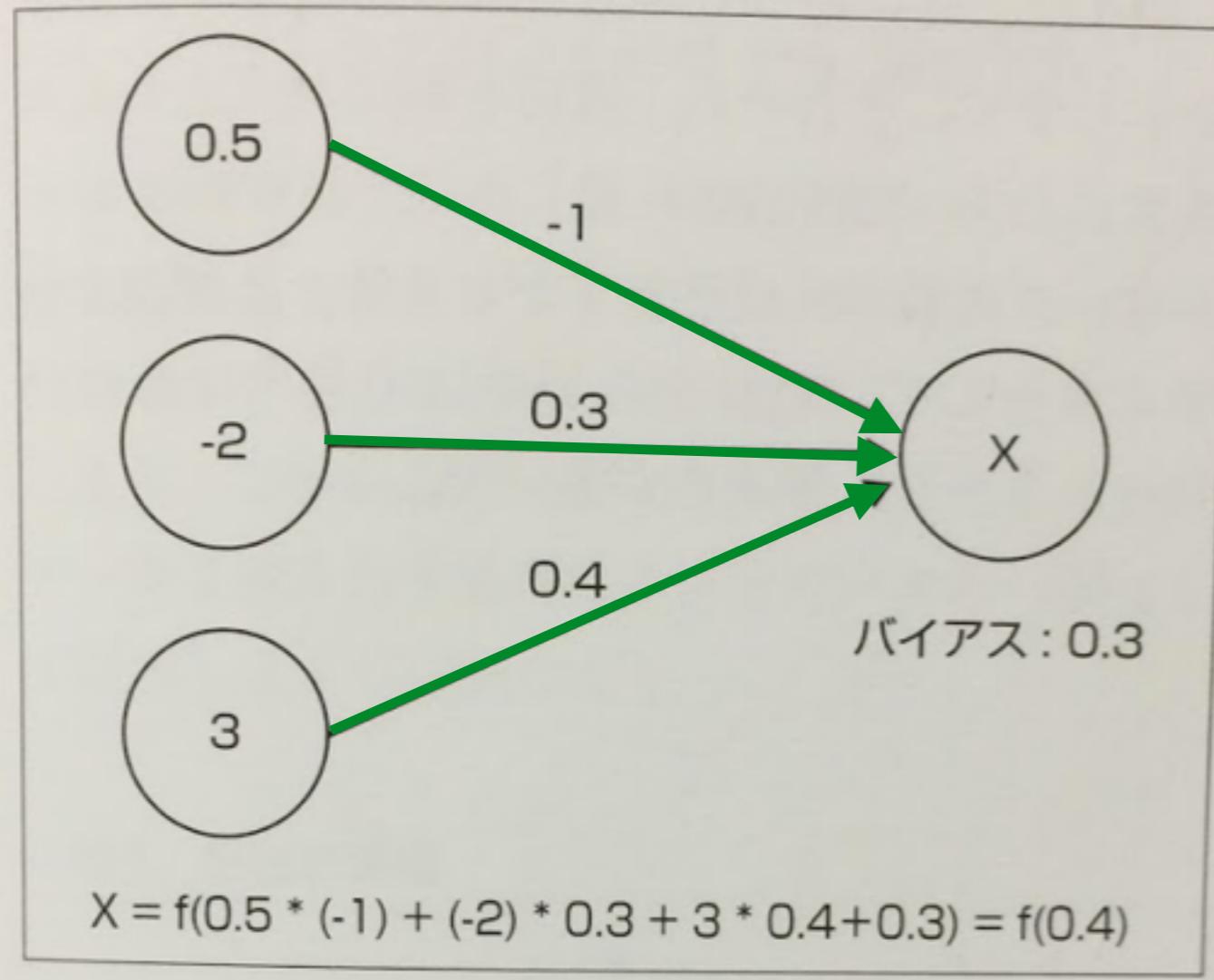
入力ユニット

(入力 x の要素を表現)

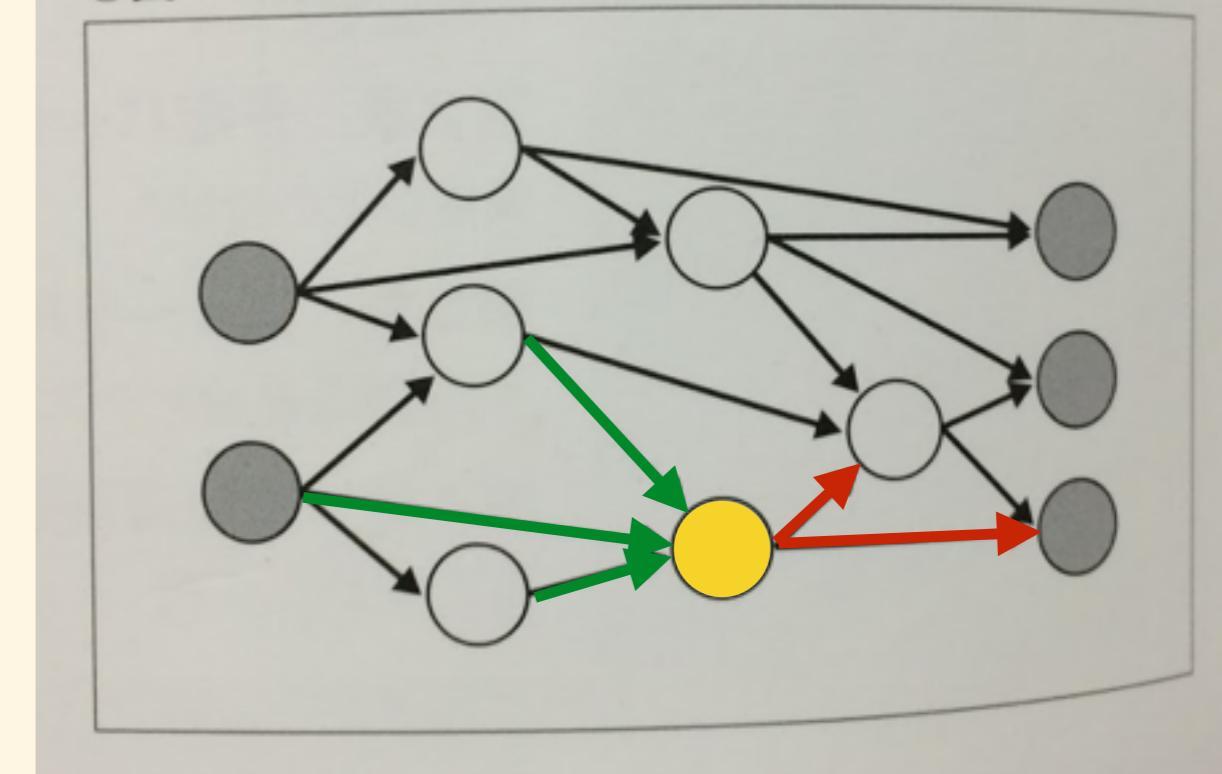
出力ユニット

(入力 y の要素を表現)

○図2 ユニットの計算



○図1 有向グラフ



ユニット出力

ロジット

ウェイト

ユニット入力

バイアス

$$X = f(z)$$

$$z = \mathbf{w}^T \mathbf{x} + b$$

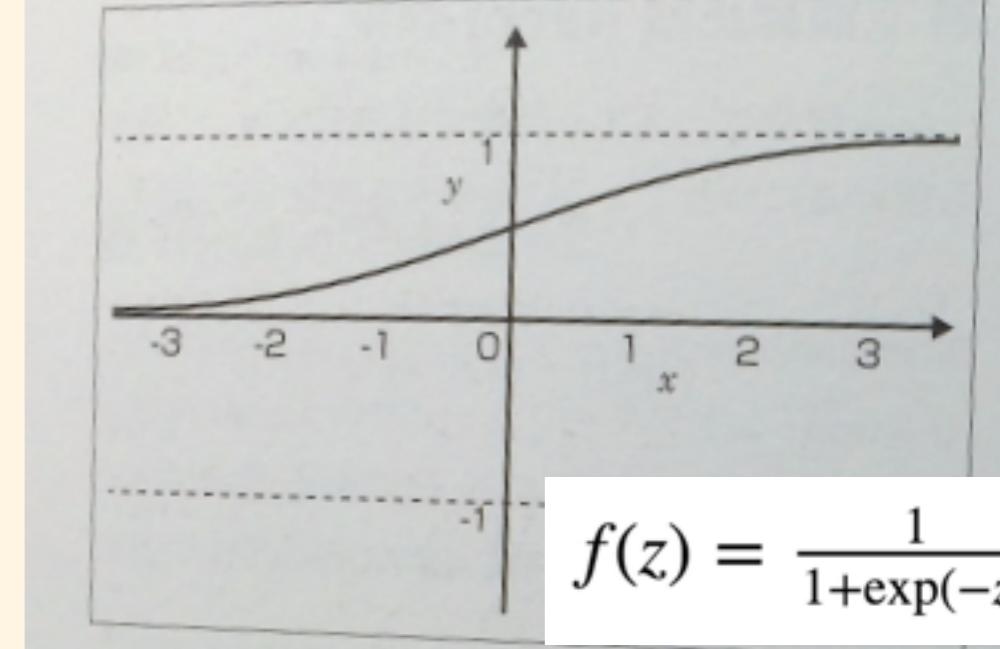
$$\mathbf{w} = [-1, 0.3, 0.4]^T$$

$$\mathbf{x} = [0.5, -2, 3]^T$$

$$b = 0.3$$

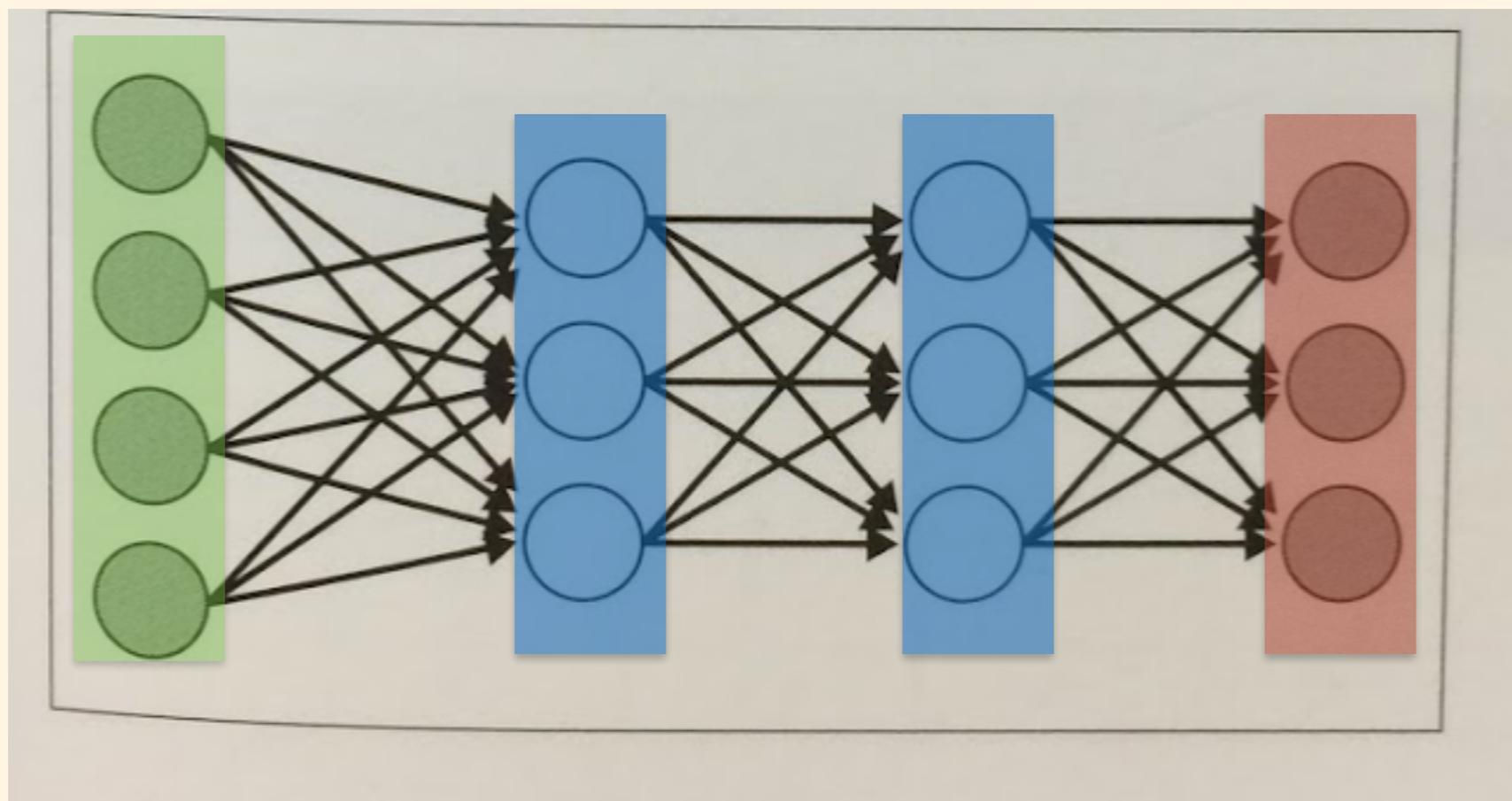
活性化関数

○図2 シグモイド



多層パーセプトロン

$$\mathbf{y} = g(\mathbf{x}; \theta)$$



入力層

中間層

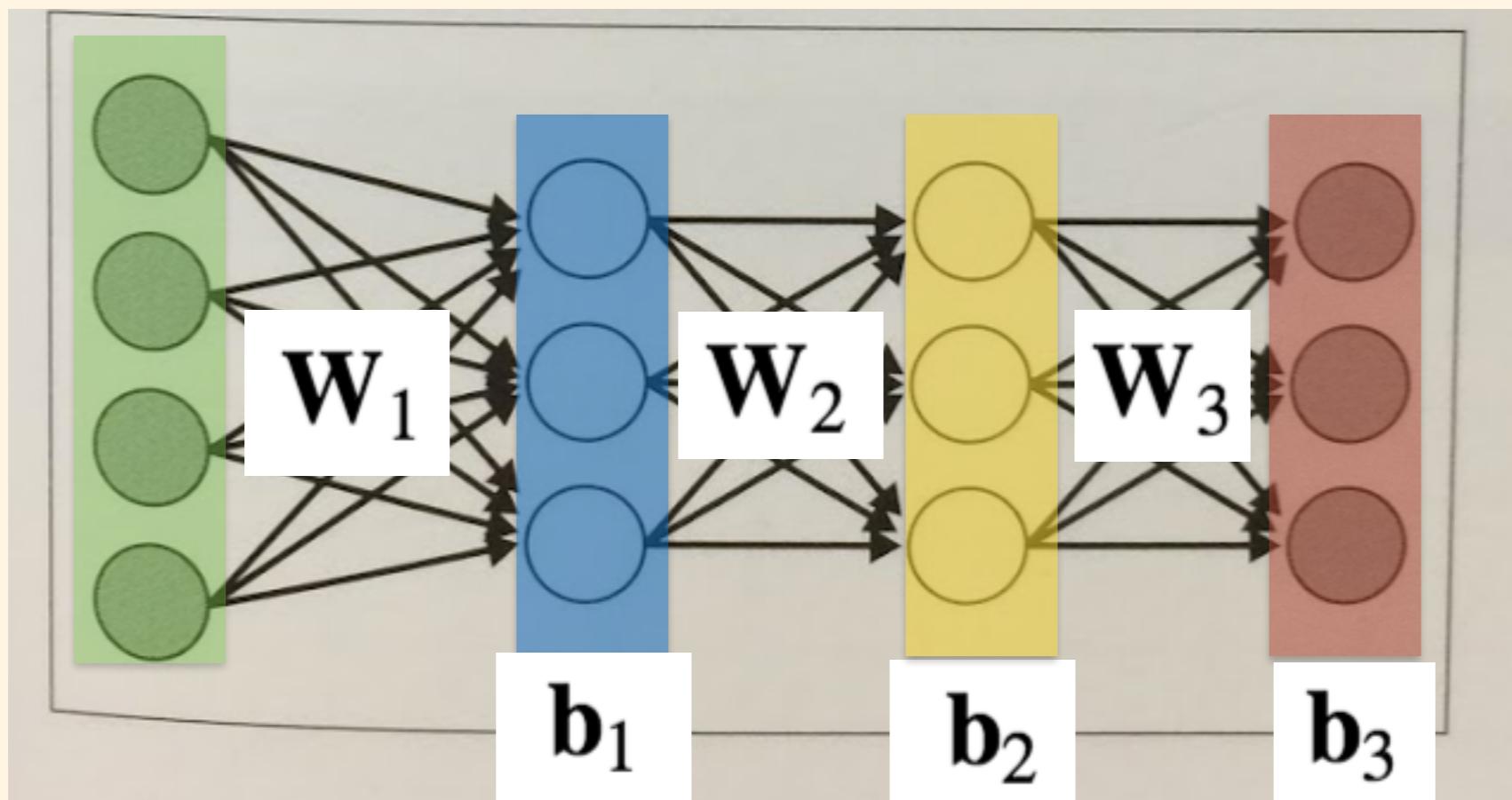
出力層

(入力データ x を表現) (出力データ y を表現)

多層パーセプトロン

$$\mathbf{y} = g(\mathbf{x}; \theta)$$

$$\theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3\}$$



入力層

中間層

出力層

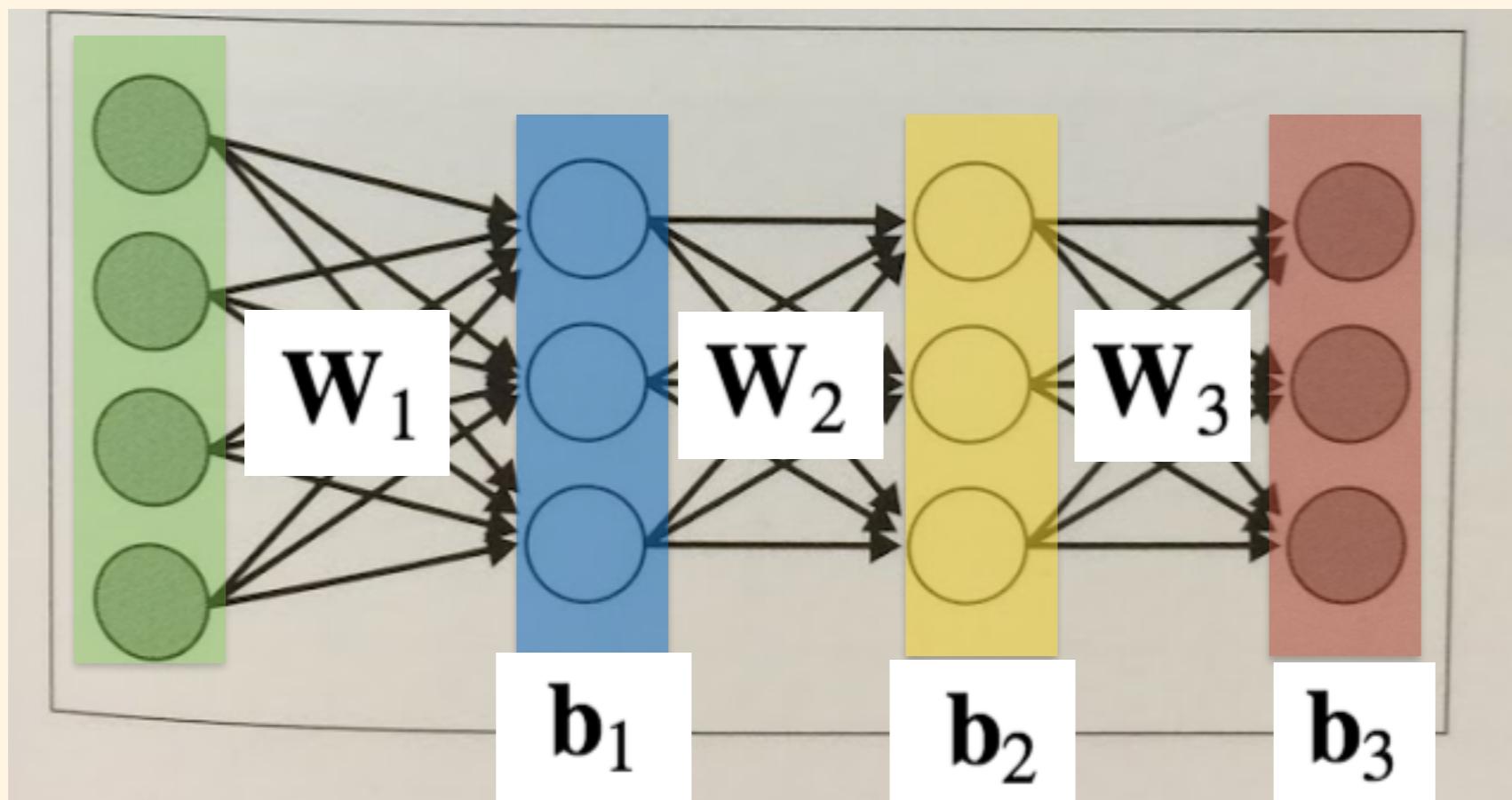
(入力データ x を表現) (出力データ y を表現)

多層パーセプトロン

$$y = g(\mathbf{x}; \theta)$$

$$\mathbf{x}_1 = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3\}$$



入力層

中間層

出力層

(入力データ x を表現) (出力データ y を表現)

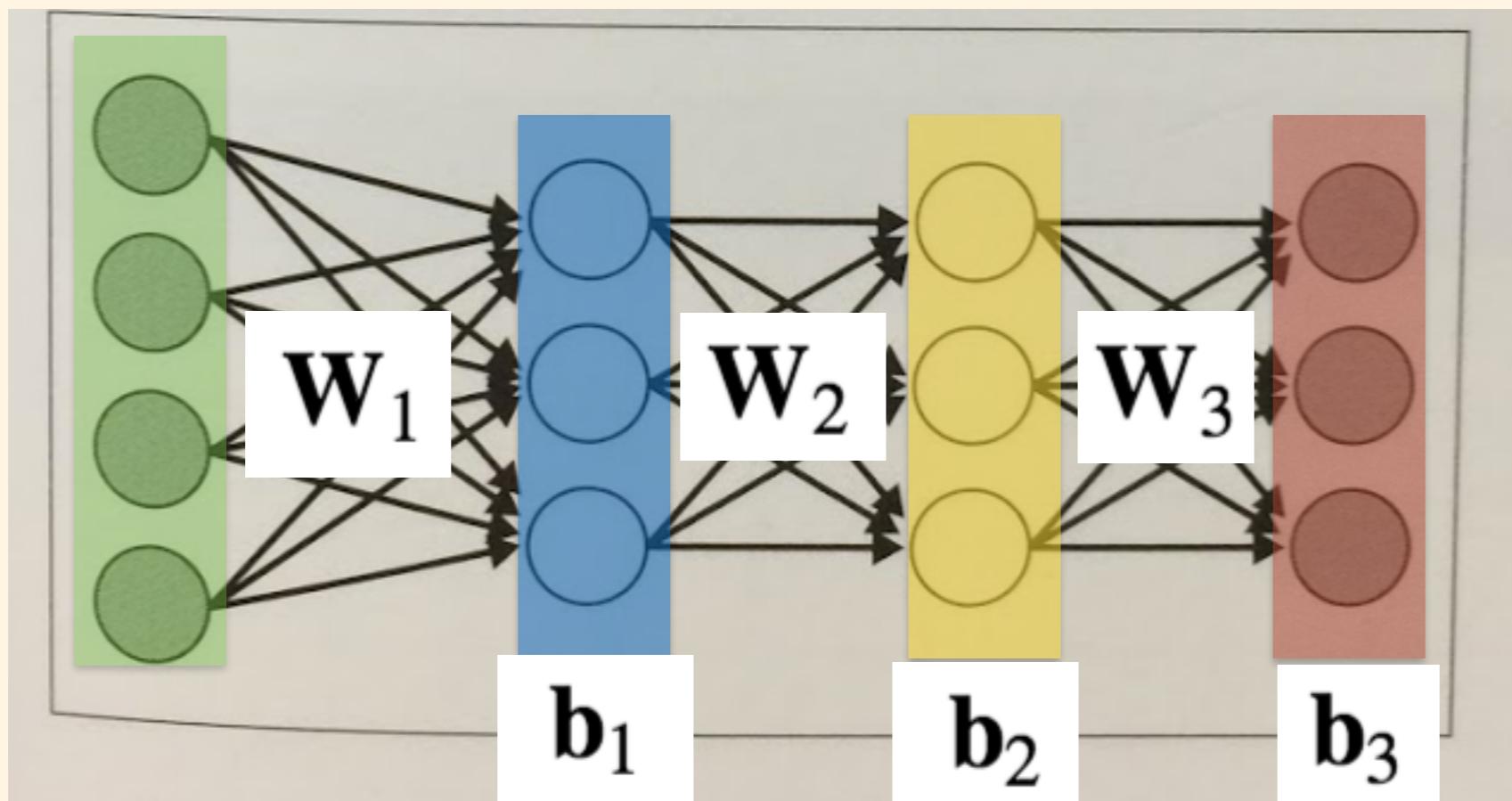
多層パーセプトロン

$$y = g(\mathbf{x}; \theta)$$

$$\mathbf{x}_1 = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3\}$$

$$\mathbf{x}_2 = f(\mathbf{W}_2 \mathbf{x}_1 + \mathbf{b}_2)$$



入力層

中間層

出力層

(入力データ x を表現) (出力データ y を表現)

多層パーセプトロン

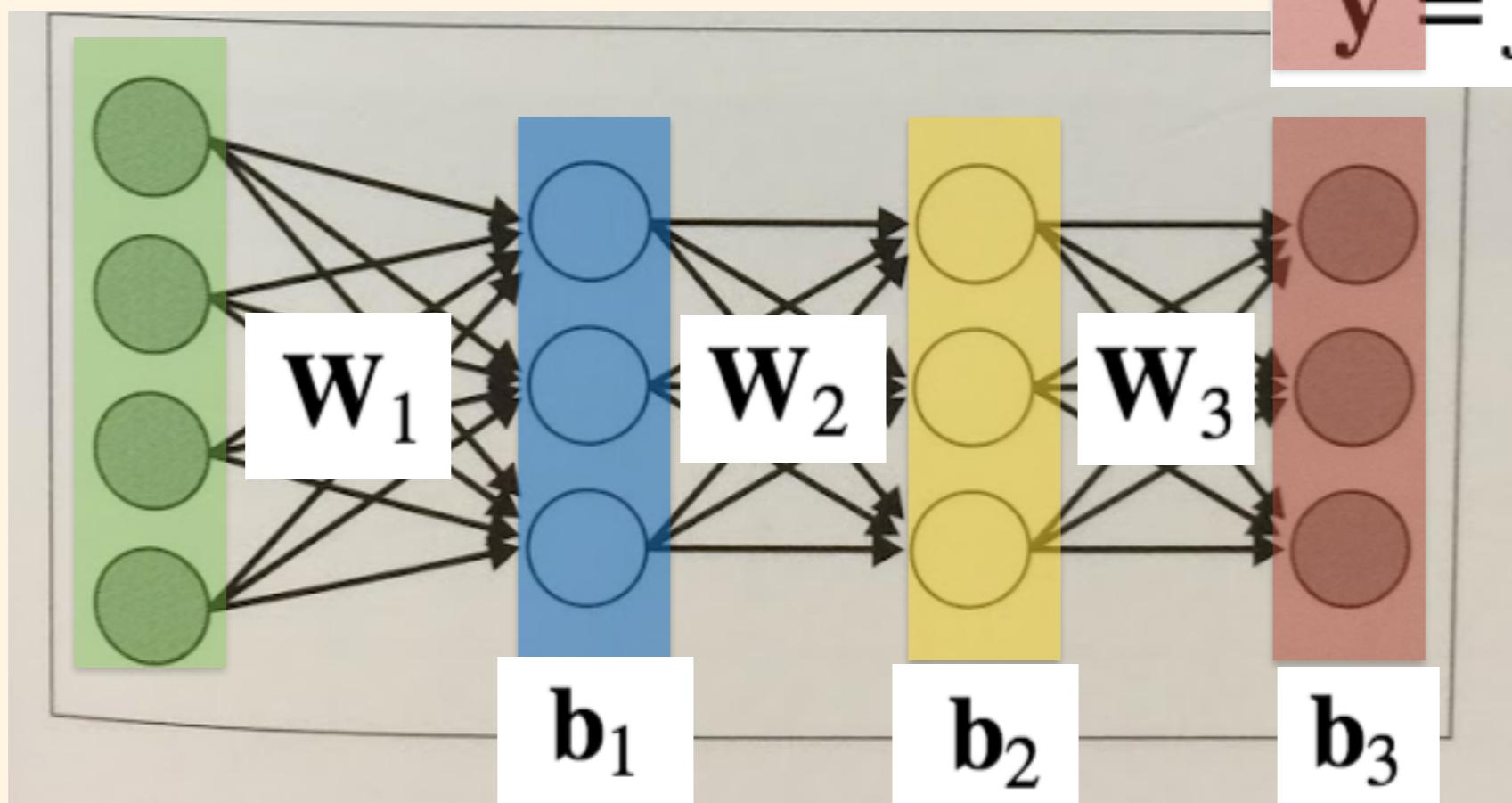
$$\mathbf{y} = g(\mathbf{x}; \theta)$$

$$\mathbf{x}_1 = f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\theta = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3\}$$

$$\mathbf{x}_2 = f(\mathbf{W}_2 \mathbf{x}_1 + \mathbf{b}_2)$$

$$\hat{\mathbf{y}} = f(\mathbf{W}_3 \mathbf{x}_2 + \mathbf{b}_3)$$



$$\hat{\mathbf{y}} \longleftrightarrow \mathbf{y}$$

予測誤差

入力層

中間層

出力層

(入力データ x を表現) (出力データ y を表現)

一般的の深層学習

- ・ いろいろな問題設定に使える
 - ・ 教師あり学習（分類問題・回帰問題）
 - ・ 教師なし学習
 - ・ 強化学習
- ・ 多段の非線形変換を学習
 - ・ 1層あたりのユニット数を増やすのではなく、層の数を増やすことにより効率的に非線形関数を学習

深層学習のフレームワーク

- ・ 深層学習のモデルは自由度が高い
- ・ 深層学習はGPUとの親和性が高い
- ・ コーディングにはオープンソースソフトウェアを使うのが一般的
 - ・ Caffe
 - ・ Torch7
 - ・ Theano/Pylearn2
 - ・ 注：Pylearn2は開発終了宣言してるので避けるべし
 - ・ Keras, Lasagne, Blocks, Theano
 - ・ Chainer
 - ・ TensorFlow

フレームワークの比較

- まずはひとつのフレームワークに慣れてから他のフレームワークに移るべし

○表1 フレームワークの比較

フレームワーク名	言語	ユーザ言語	速度	柔軟さ	開発 コミュニティ	ドキュメント	サンプル	日本語での サポート
Caffe	C++、Python	Protocol Buffer (text)、Python	◎	△	◎	△	◎	×
Torch7	Lua、C	Lua	○	○	○	○	△	×
Theano/Pylearn2	Python、C++	YAML(Pylearn2)、 Python	○	○	○	○	◎	×
Chainer	Python	Python	○	◎	△	○	○	○

第1章：準備

問題設定とニューラルネットの基本 (p.68)

第2章：多層パーセプトロンの書き方

構成要素の理解 (p.71)

第3章：ニューラルネットの学習方法

様々な最適化テクニック (p.74)

第4章：画像認識のためのアーキテクチャ

畳込みネット入門 (p.78)

構成要素

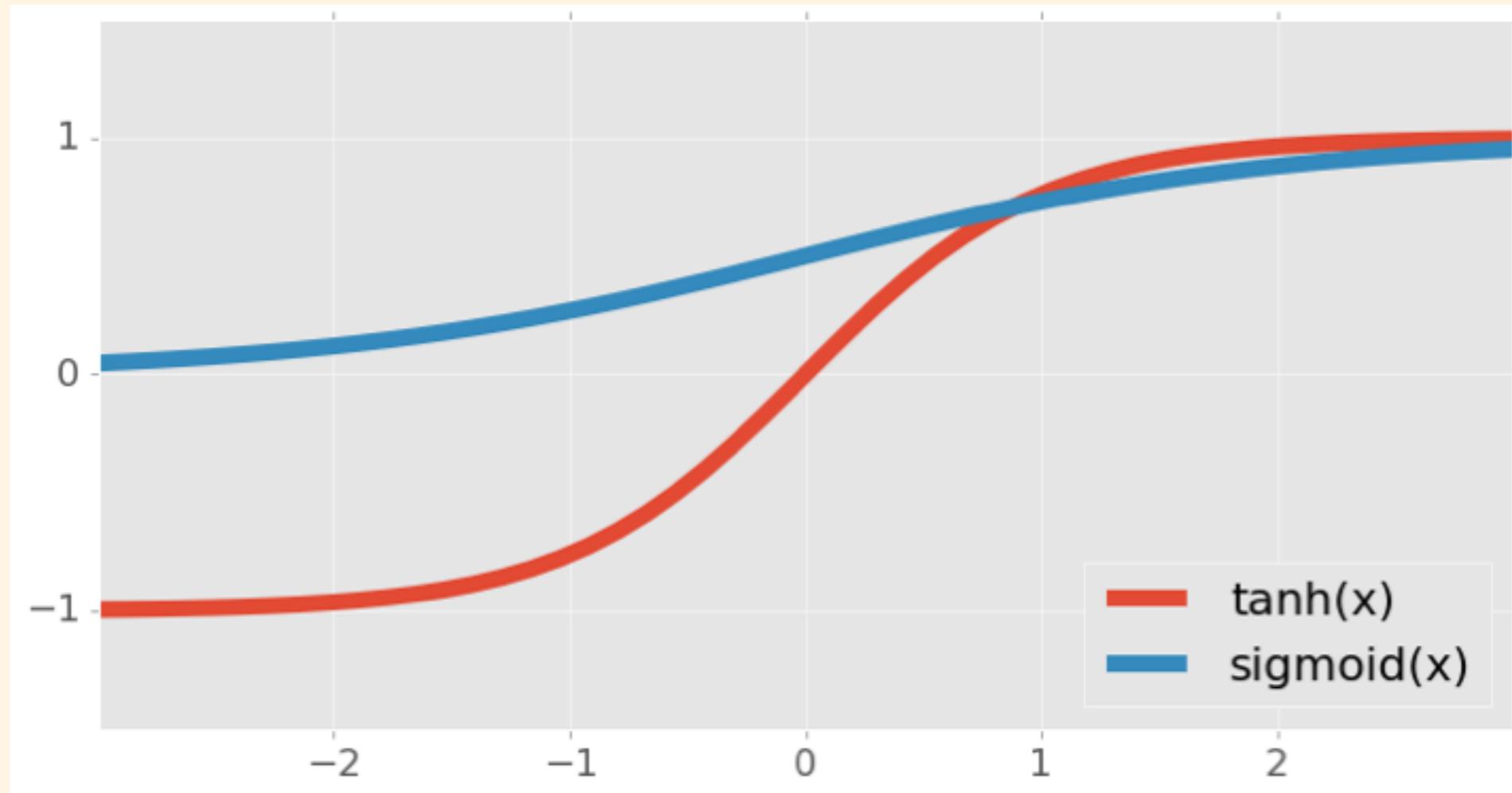
- ・ 重み行列 (w) をかけてから、バイアスペクトル (b) を足す (全結合層)
- ・ 非線形関数 (f) を適用する (活性化関数)
- ・ ネットワークの出力を評価する (損失関数)

活性化関数 f

- 非線形性はここで生み出される！
- これがないと、ニューラルネットもただのアファイン変換（線形変換+シフト）学習器になっちゃう
- 非線形変換学習器だから嬉しい
- \tanh 、 sigmoid 、 ReLU (Rectified Linear Unit)などなど色々ある

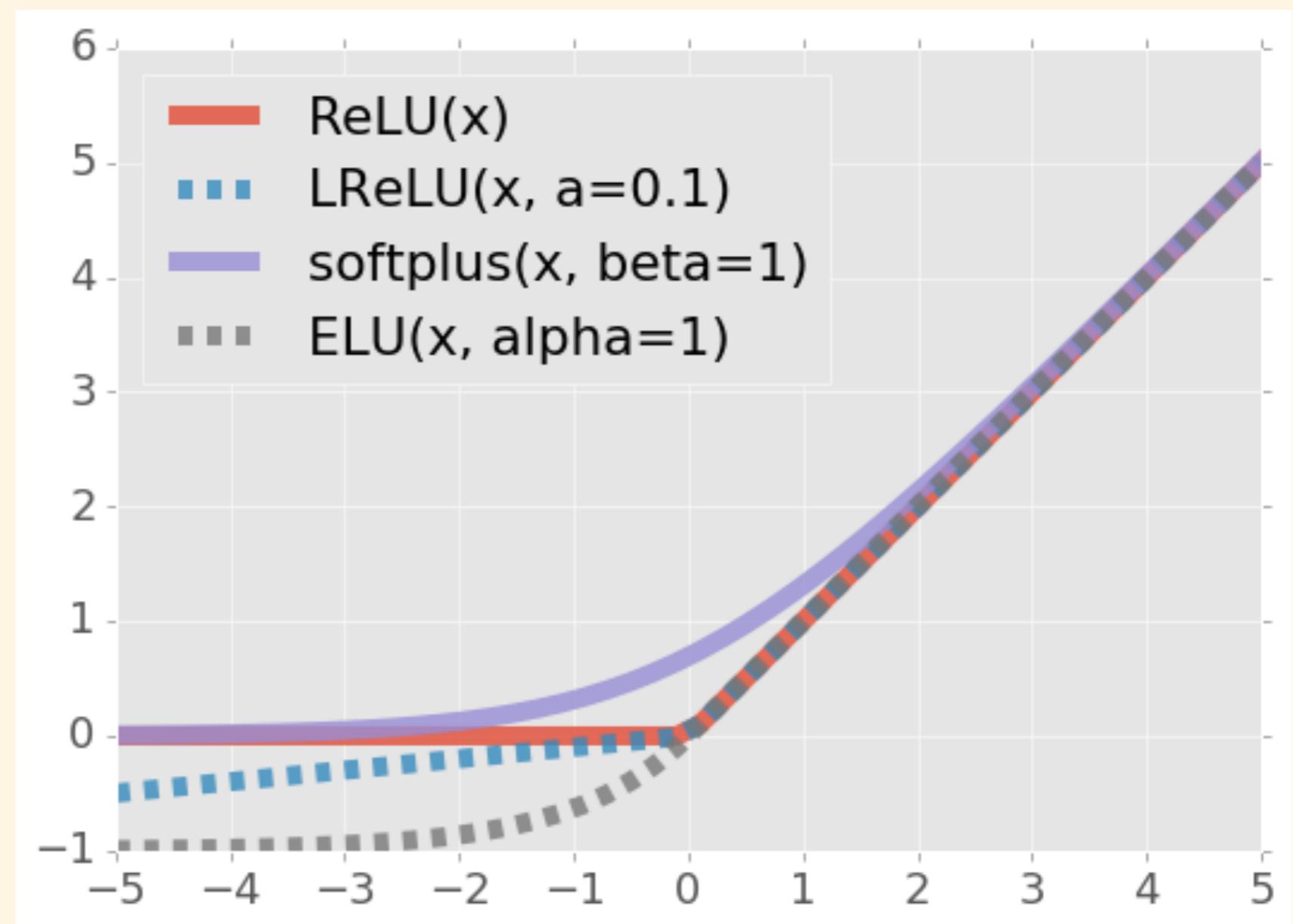
活性化関数 (1/3)

- 双曲線正接 (tanh) 関数
- シグモイド関数



活性化関数 (2/3)

- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Parametric ReLU
- Softplus
- ELU

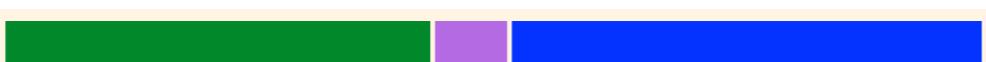


ソフトマックス関数 (3/3)

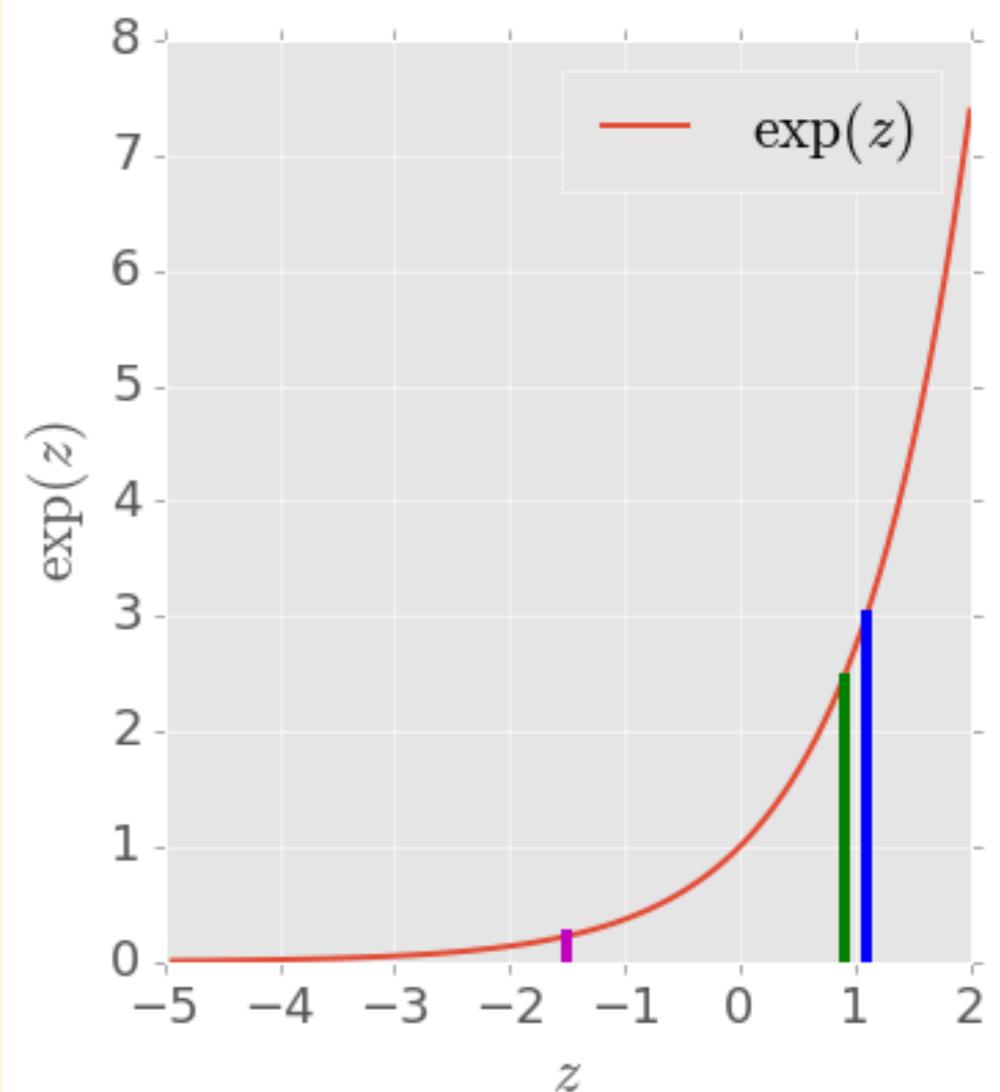
$$p(y_k = 1; z_{1:K}) = \frac{\exp(z_k)}{\sum_{k'=1}^K \exp(z_{k'})}$$

- シグモイド関数をK個のクラスに拡張したもの
- 数値ベクトルを確率ベクトルにマッピング
 - [0.04, 0.43, 0.53] = softmax([-1.5, 0.9, 1.1])
 - z が大きいと、そのクラスが 1 となる確率が高くなる
- K=3の場合

$$p(y_k = 1; z_1, z_2, z_3) = \frac{\exp(z_k)}{\exp(z_1) + \exp(z_2) + \exp(z_3)}$$



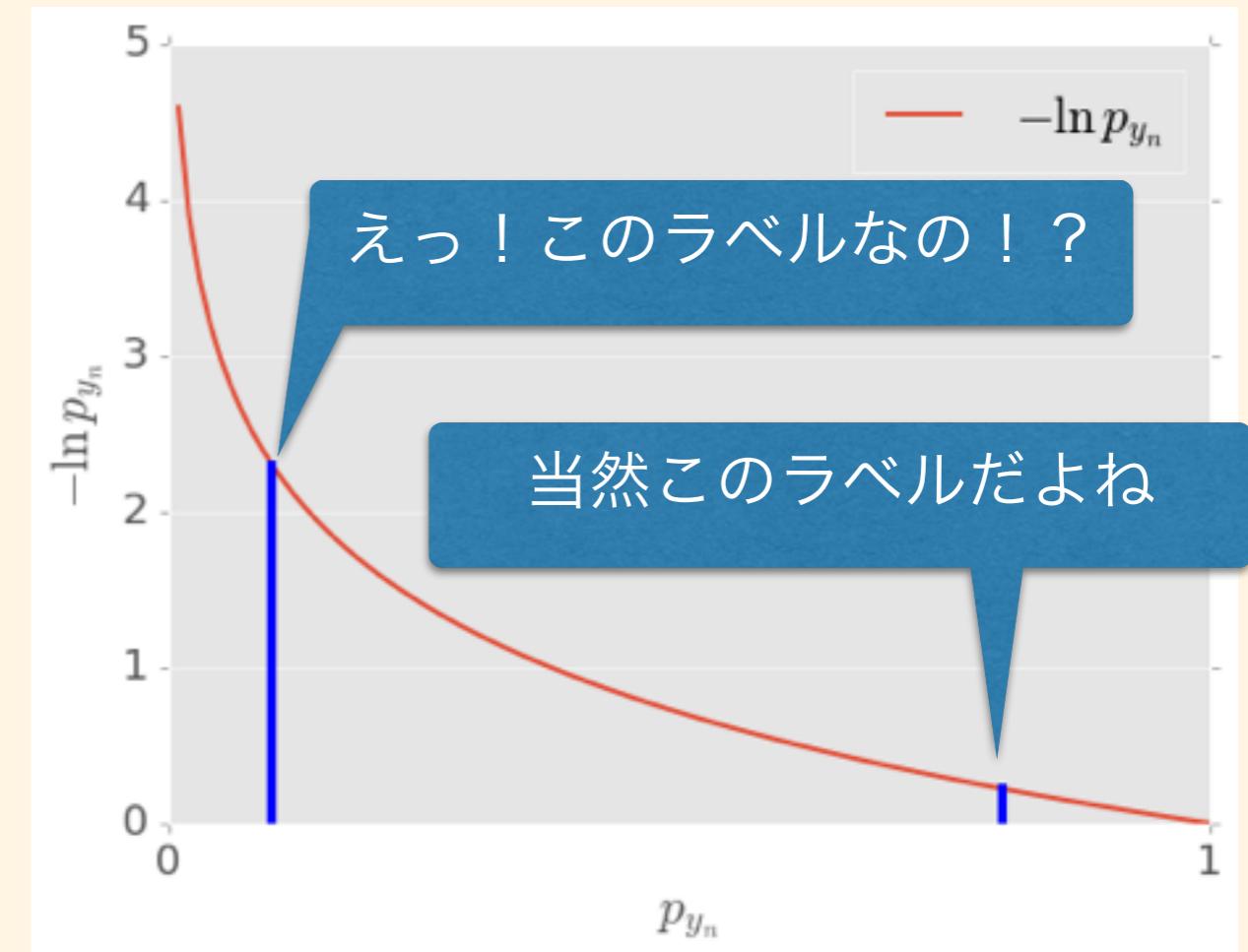
$$\begin{aligned} p(y_1 = 1; z_1, z_0) &= \frac{1}{1 + \exp(-\{z_1 - z_0\})} \\ &= \text{sigmoid}(z_1 - z_0) \end{aligned}$$



損失関数

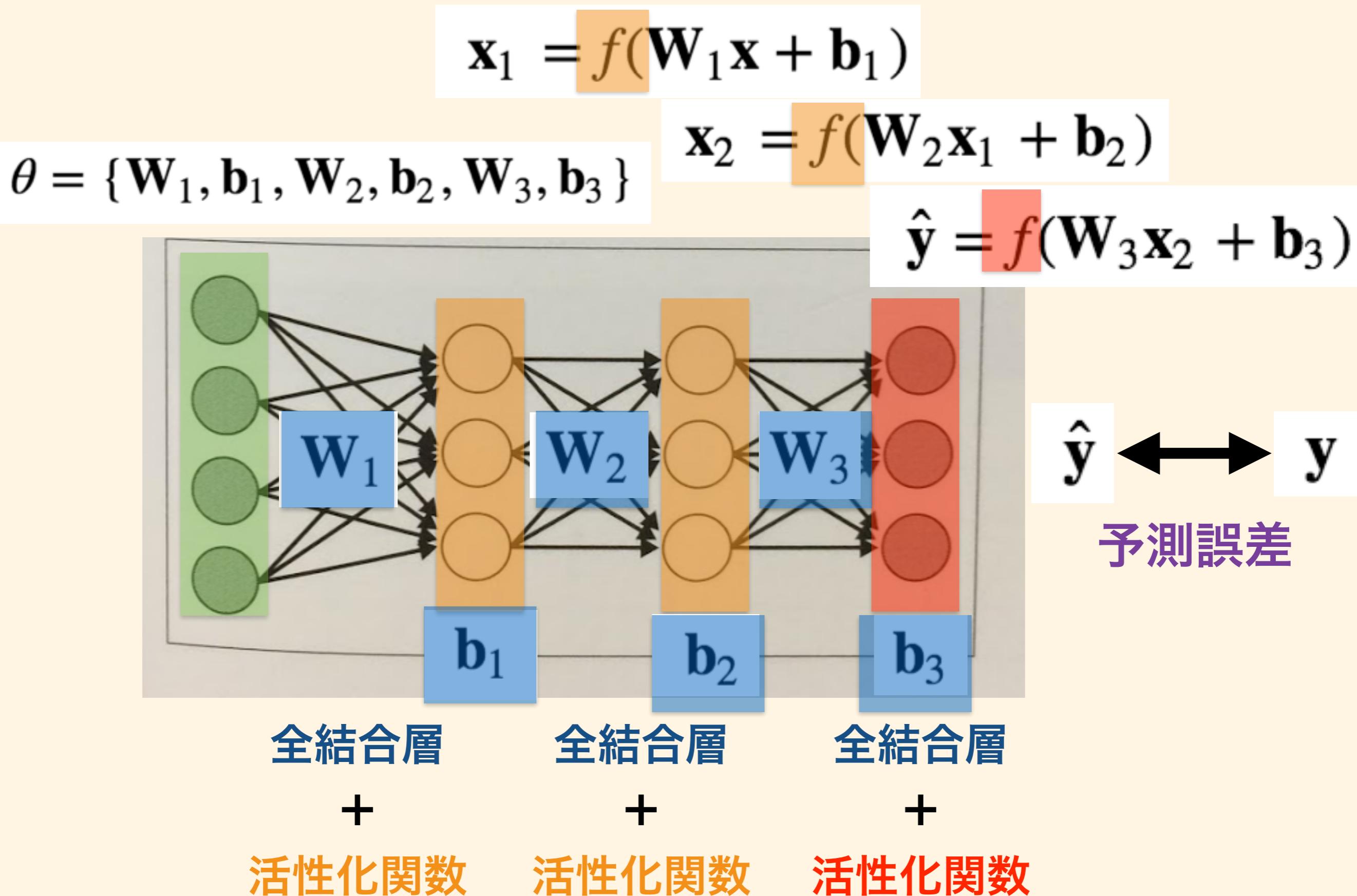
- 予測の外れ度合いを表す関数

- 多値分類問題



- 出力層に対する非線形変換にはソフトマックス関数を用いる
- 損失関数には交差エントロピー誤差関数 $-\ln p_{y_n}$ を用いる
- p_{y_n} : n個目のデータで、実際に出たラベル y_n に対するモデルの予測値
- そのラベルが実際に出了ことに対するビックリ度

多層パーセプトロンの書き方



○リスト2 Torch7での多層パーセptron記述(lua)

```
require 'nn'

mlp = nn.Sequential()
: add(nn.Linear(784, 1000))
: add(nn.ReLU())
: add(nn.Linear(1000, 10))

cri = nn.CrossEntropyCriterion()

function forwardBackward(x, t)
    local output = mlp:forward(x)
    local loss = cri:forward(output, t)
    local g_output = cri:backward(output, t)
    mlp:backward(x, g_output)
end
```

○リスト3 Chainerでの多層パーセプトロン記述(python)

```
import chainer
import chainer.functions as F

model = chainer.FunctionSet(
    layer1=F.Linear(784, 1000),
    layer2=F.Linear(1000, 10),
)

def forward_backward(x, t):
    h = F.relu(model.layer1(x))
    y = model.layer2(h)
    loss = F.softmax_cross_entropy(y, t)
    loss.backward()
```

現在のバージョン v1.5 では書き方がちょっと違う

第1章：準備

問題設定とニューラルネットの基本 (p.68)

第2章：多層パーセプトロンの書き方

構成要素の理解 (p.71)

第3章：ニューラルネットの学習方法

様々な最適化テクニック (p.74)

第4章：画像認識のためのアーキテクチャ

畳込みネット入門 (p.78)

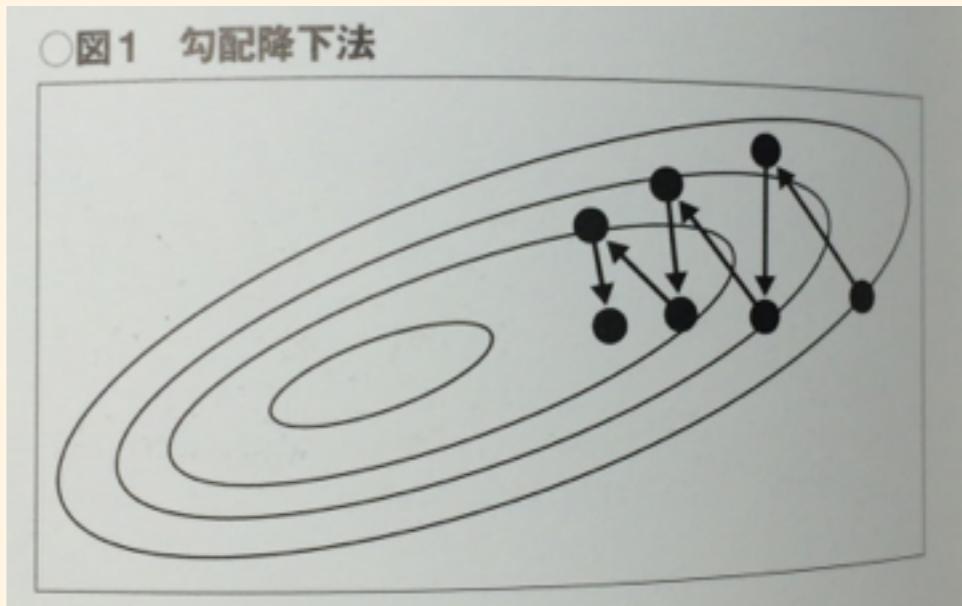
最適化と目的関数

- 分類問題でやりたいこと
 - トレーニングデータを使って、正しいラベルを予測できるようにパラメータを学習する
- (数理) 最適化
 - なんらかの関数の値を小さくするパラメータを求めること
- 目的関数
 - 小さくしたい関数
 - ここでは損失関数の総和

$$-\sum_{n=1}^N \ln p_{y_n}$$

勾配法

- 勾配降下法 (Gradient Descent)



- バッチアルゴリズム (全データを使って1回パラメータを更新)
- 学習率を徐々に小さくしながら、収束させる

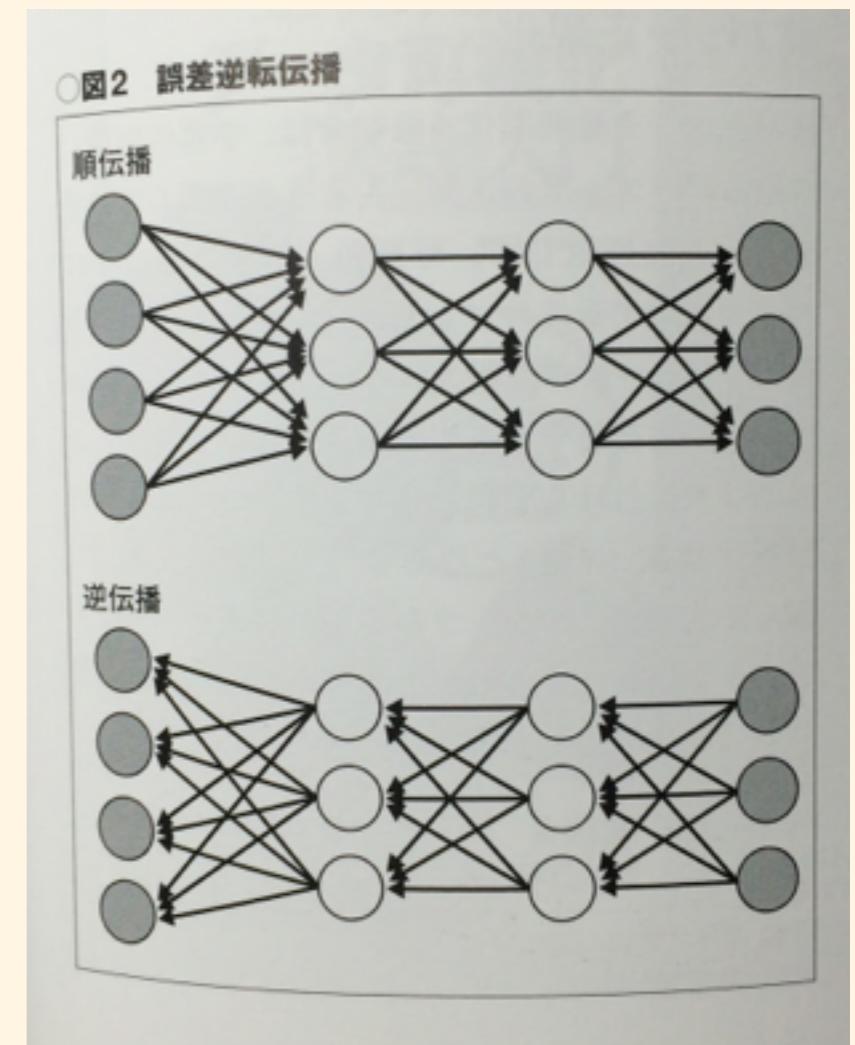
- 確率的勾配降下法 (Stochastic Gradient Descent, SGD)

- ミニバッチ (いくつかのデータの集まり) を用いたアルゴリズム
- 1つのデータ (ベクトル) → いくつかのデータ (行列)

誤差逆伝播法

Error Backpropagation (BP)

- 勾配を計算するためのアルゴリズム
- 連鎖律を使う : $df(g(x))/x = df/dg * dg/dx$
- 予測誤差は出力から入力方向に流して使う
- BP + SGD でニューラルネットの学習を行う



効率的な最適化手法

- モーメンタム法
 - 慣性項を入れて、前に求めた勾配方向を引きずるようにする
- Adam
 - パラメータごとに学習率を自動調整
- 他にもいろいろ (Nesterov's momentum, RMSProp, AdaGrad, AdaDelta, and etc.)

最適化のためのテクニック

- 荷重減衰：大きなウエイトにペナルティをかける
- Max norm：入力ノルムを一定値以下に保つ
- 勾配クリッピング：勾配の大きさに上限を設定
- パラメータの賢い初期化：Glorot initializationなど
- バッチ正規化：各層のユニットの値をミニバッチに対して正則化

過学習と正則化

- ・ 「過学習」とは?
 - ・ 訓練データに適合しすぎて、テストデータに対する性能が低下
 - ・ 自由度が高いニューラルネットにとって大きな問題
- ・ そこで「正則化」
 - ・ 荷重減衰、Max norm、バッチ正則化など
 - ・ Dropout (学習時にユニットをランダムに抜く)

第1章：準備

問題設定とニューラルネットの基本 (p.68)

第2章：多層パーセプトロンの書き方

構成要素の理解 (p.71)

第3章：ニューラルネットの学習方法

様々な最適化テクニック (p.74)

第4章：画像認識のためのアーキテクチャ

畳込みネット入門 (p.78)

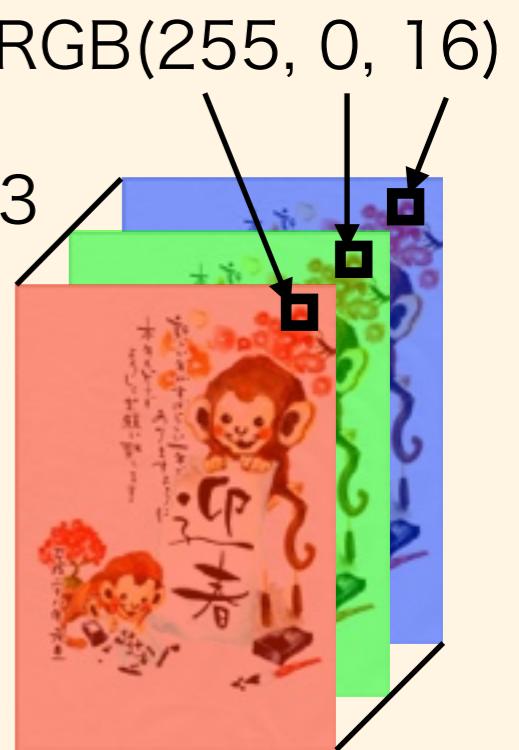
画像の構造

- グレースケール画像



H=350

- カラー画像

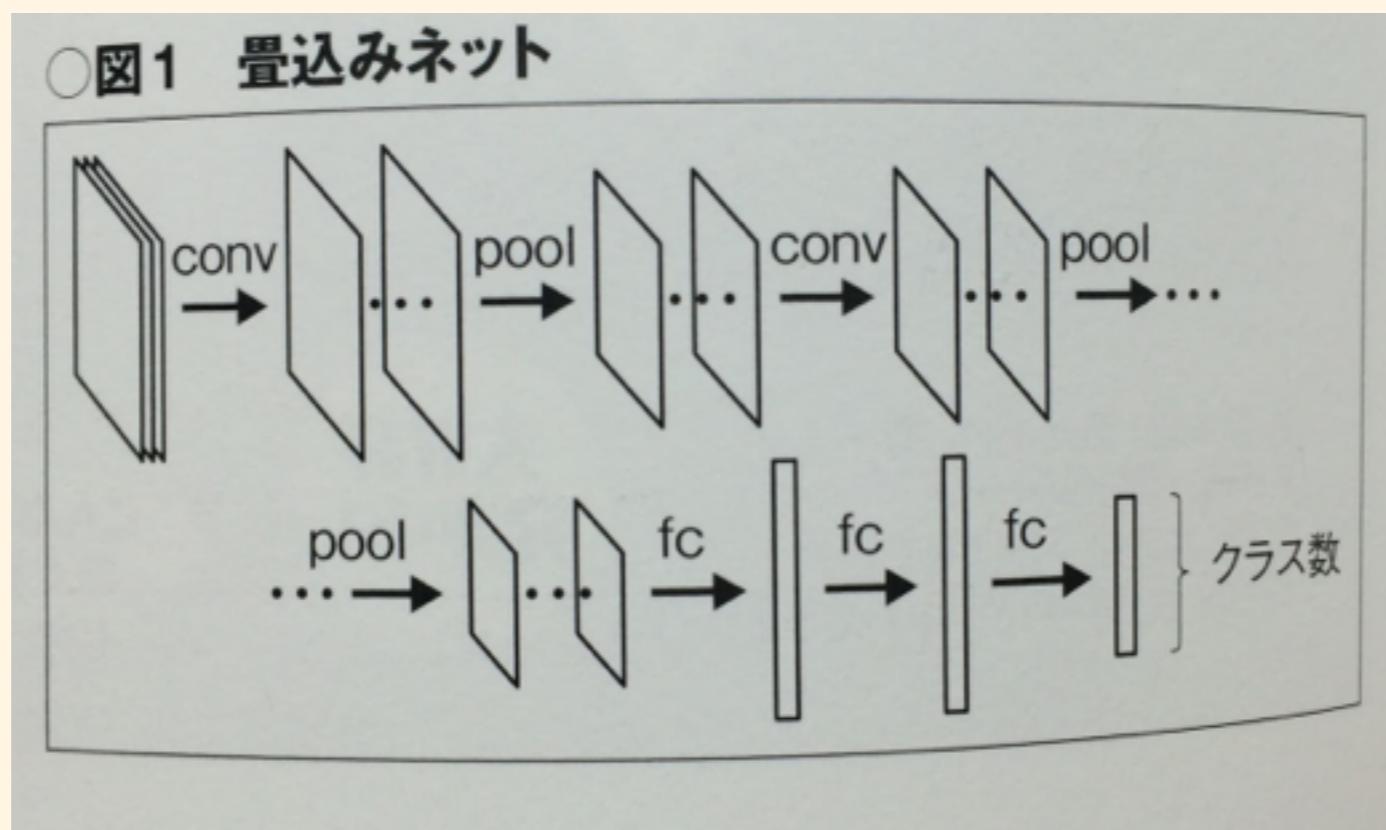


- (Minibatch) x Channel x Height x Width
- 局所性：近傍ピクセル同士に強い相関
- 平行移動普遍性

畳込みネット

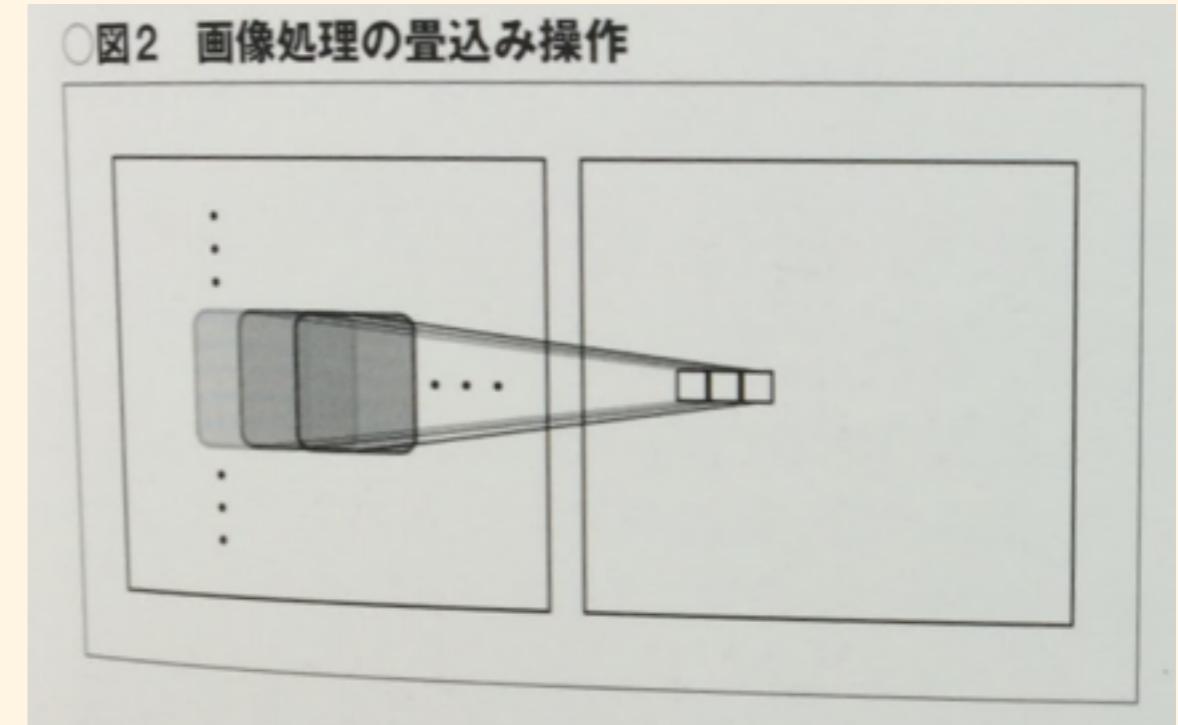
Convolutional Neural Network

- ・局所性と平行移動普遍性を利用
- ・畳込み層(conv)、プーリング層(pool)、全結合層(fc)を組み合わせた多層パーセプトロンの亜種



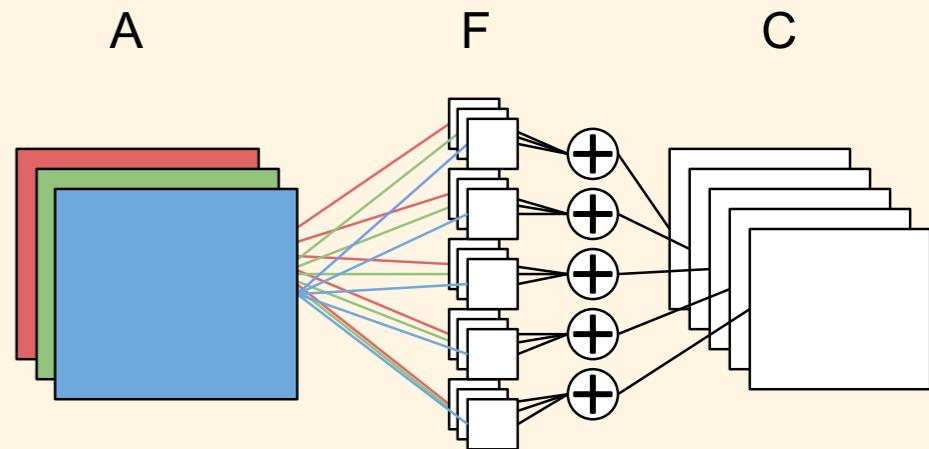
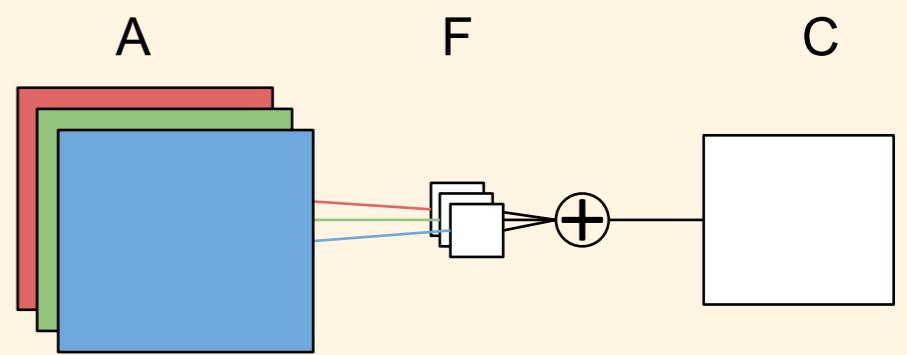
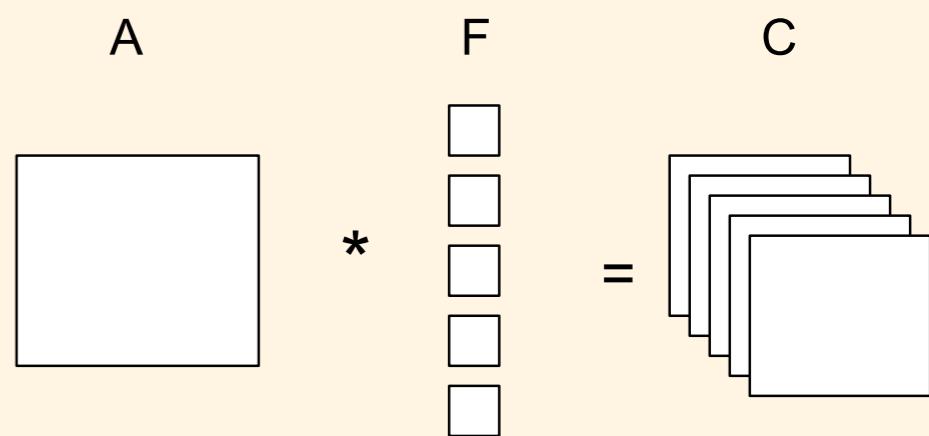
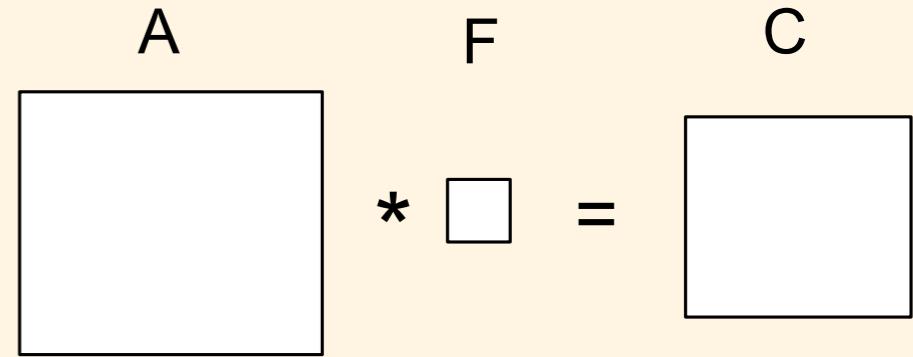
畠込み層

- 画像処理の畠込み
 - ガウシアンフィルタやラプラシアンフィルタなど
 - パラメータ固定
- 出力チャネル数は通常 1 つ
- ニューラルネットの畠込み
 - パラメータの学習が可能
 - 出力チャネル数が多い



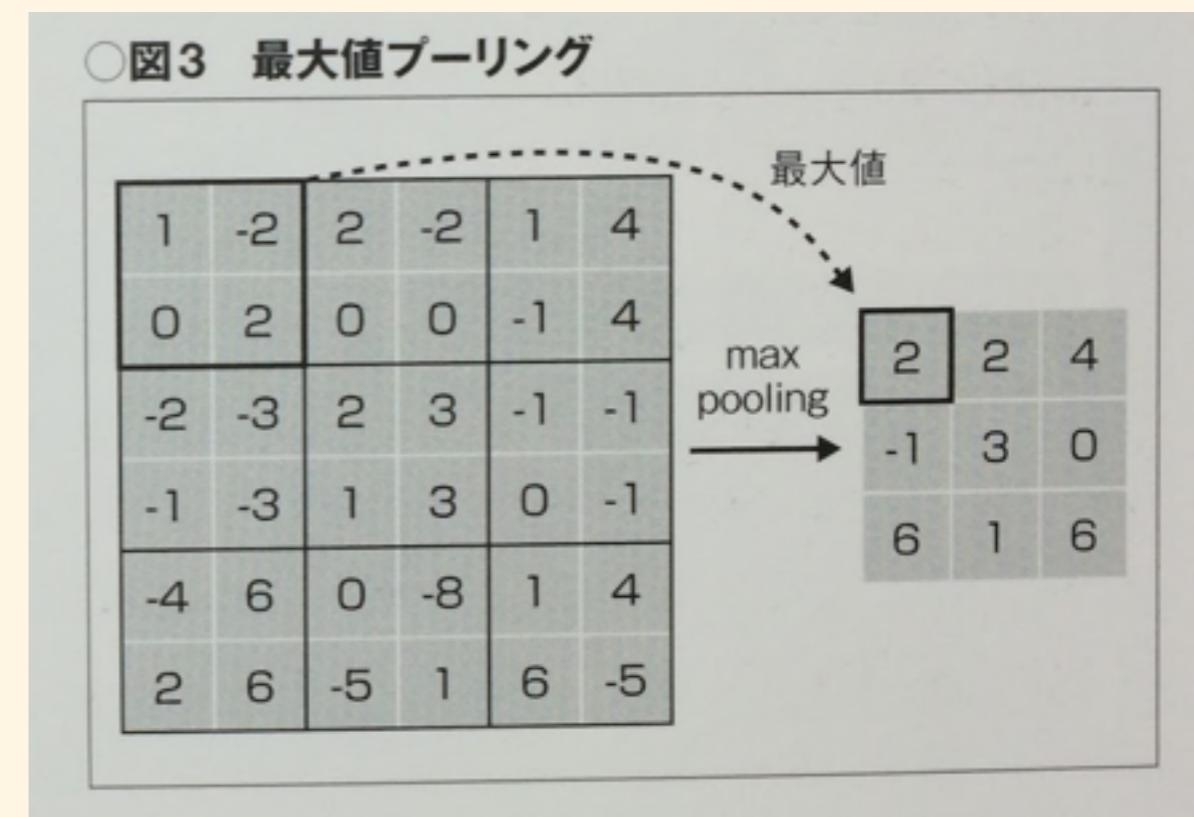
畠込み層

- 畠込み層の表現： $M \times K \times K \times N$ のテンソル
 - M : 入力チャネル数
 - $K \times K$: フィルタサイズ
 - N : 出力チャネル数 (フィルタ数ともいう)
- $M \times K \times K$ のフィルタから1枚のフィルタ画像ができる
- フィルタをかける際のステップ幅が大きいとフィルタ画像が小さくなる



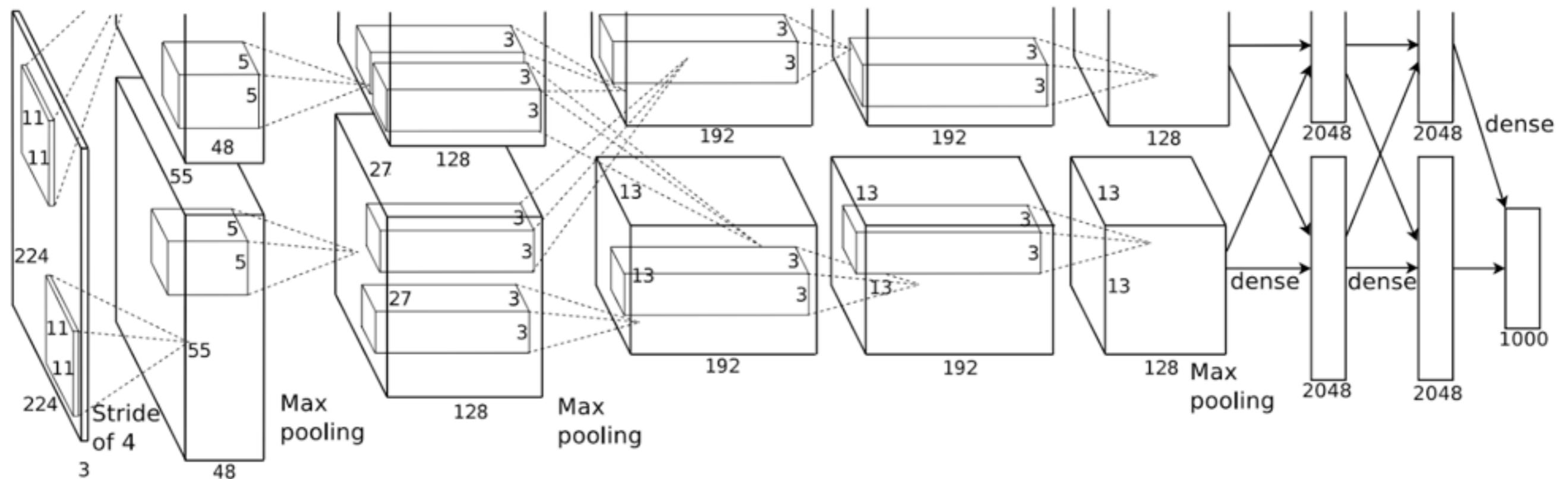
プーリング層

- ・サブサンプリングをして画像の解像度を下げる操作
- ・最大値プーリング
(Max Pooling)
- ・近傍での最大値を出力



設計の勘所

- ・ 置込みネットの設計にはまだ決定打がない
- ・ 経験的に上手くいくという設計はある



Alex Krizhevsky et al. 2012

Thanks ;)