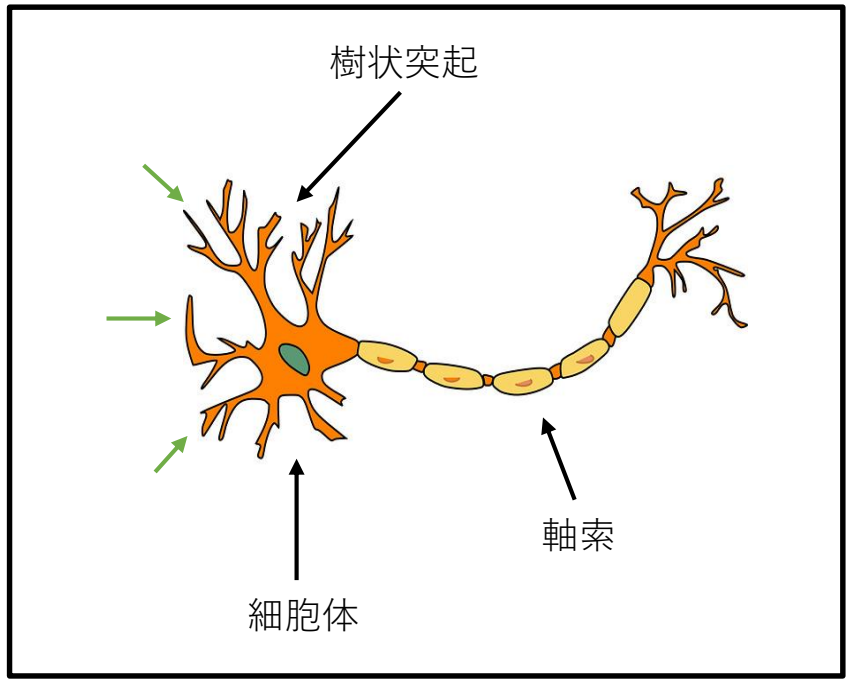
はじめに

- CNN(コンボリューショナル ニューラル ネットワーク)のアーキテクチャ
- MATLAB演習：〇×判定のCNN

NN(Neural Network)

■ 人間の脳神経回路の仕組み



図：ニューロン

ニューラルネットワークは人間の脳神経回路を数学的モデルで表現したもの.

ニューロンの知見
〇神経細胞はニューラルネットワークを形作っている.
〇他の複数の神経細胞から伝えられる信号に対して
ある一定の大きさ(しきい値)を超えると初めて神経細胞は反応し(これを発火という),別の神経細胞に一定の強さの信号を伝える.
〇複数の神経細胞から伝えられる信号の和は信号ごとにその重みが異なる.

この知見を数学的に抽象化し,それを単位(ユニット)として人工的にネットワーク化したものがニューラルネットワークである.

NN(Neural Network)とは

ニューラルネットワークとは

単純構造である人工ニューロン(ユニット)を,実際の脳内のようにそれぞれをつなげてネットワーク化して処理を行うもの.

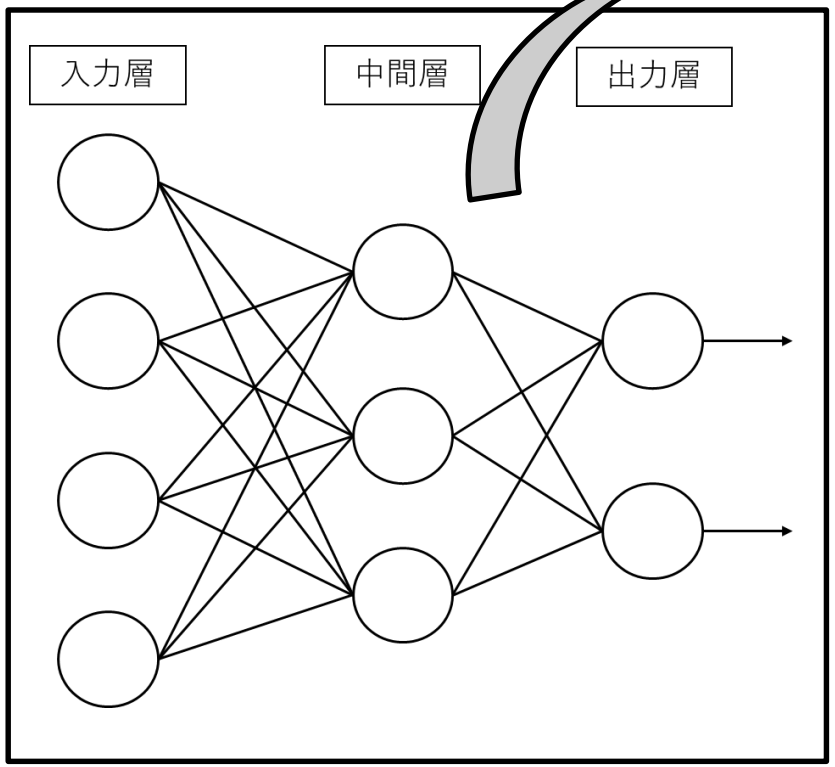
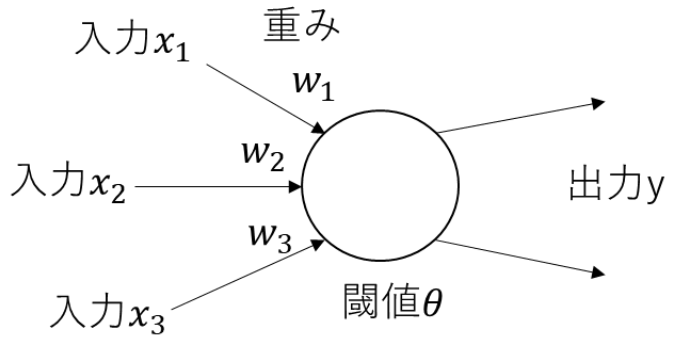


図:ニューラルネットワーク



人工ニューロン(ユニット)

入力信号$x_1, x_2, \ldots, x_n$(nは自然数)の時，各信号には重み$w_1, w_2, \ldots, w_n$が与えられるとする．閾値をθとするとき,ニューロンの出力yは

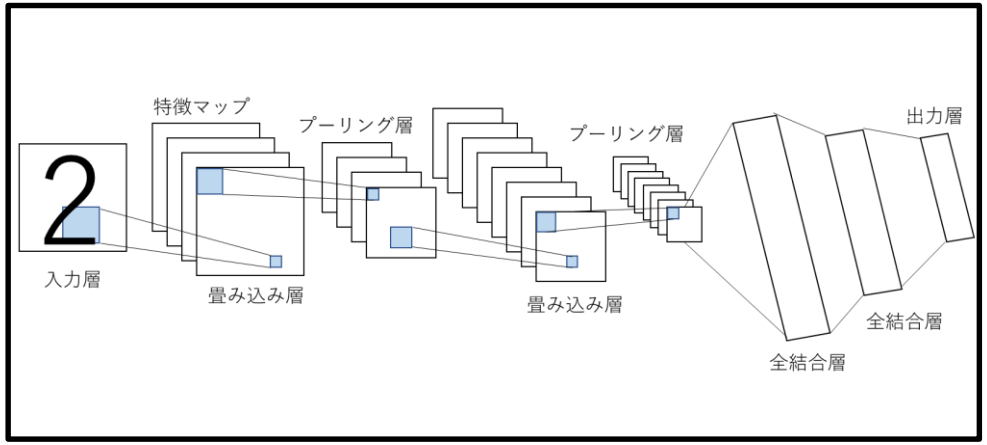
$$y = a(s)$$

ここで，aは「活性化関数(activation function)」であり，シグモイド関数や，thanh関数，ReLUなどがある．,sは「入力の線形和」と呼ばれ，次のように定義される．

$$s = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n - \theta$$

CNN(Convolutional Neural Network)

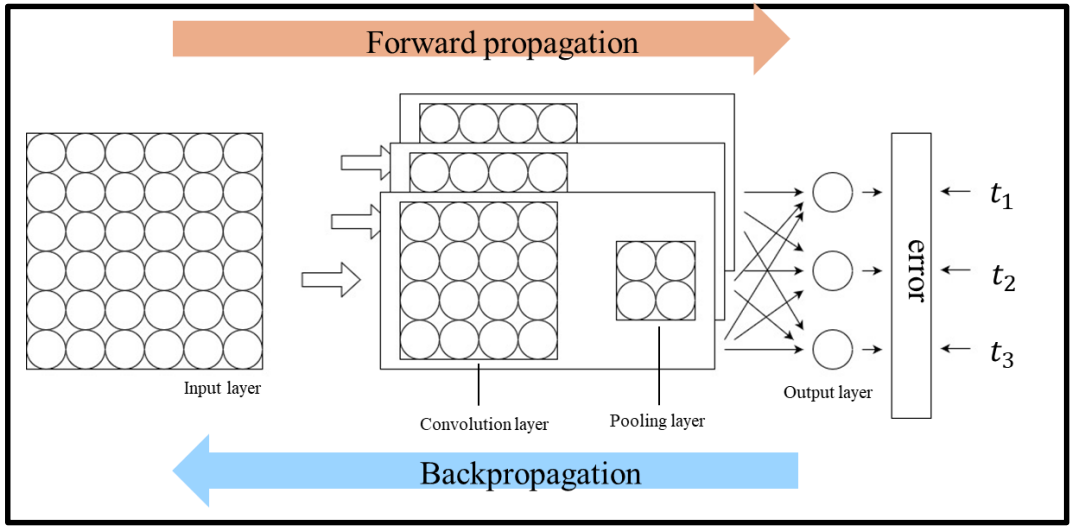
CNN(Convolutional Neural Network)とは



図：CNN(Convolutional Neural Network)

畳み込みニューラルネットワーク(Convolutional Neural Network)：何段もの深い層を持つニューラルネットワークで，特に画像認識の分野で優れた性能を発揮しているニューラルネットワーク．「畳み込み層」や「プーリング層」などの幾つかの特徴的な機能を持った層を積み上げることで構成されている．

学習



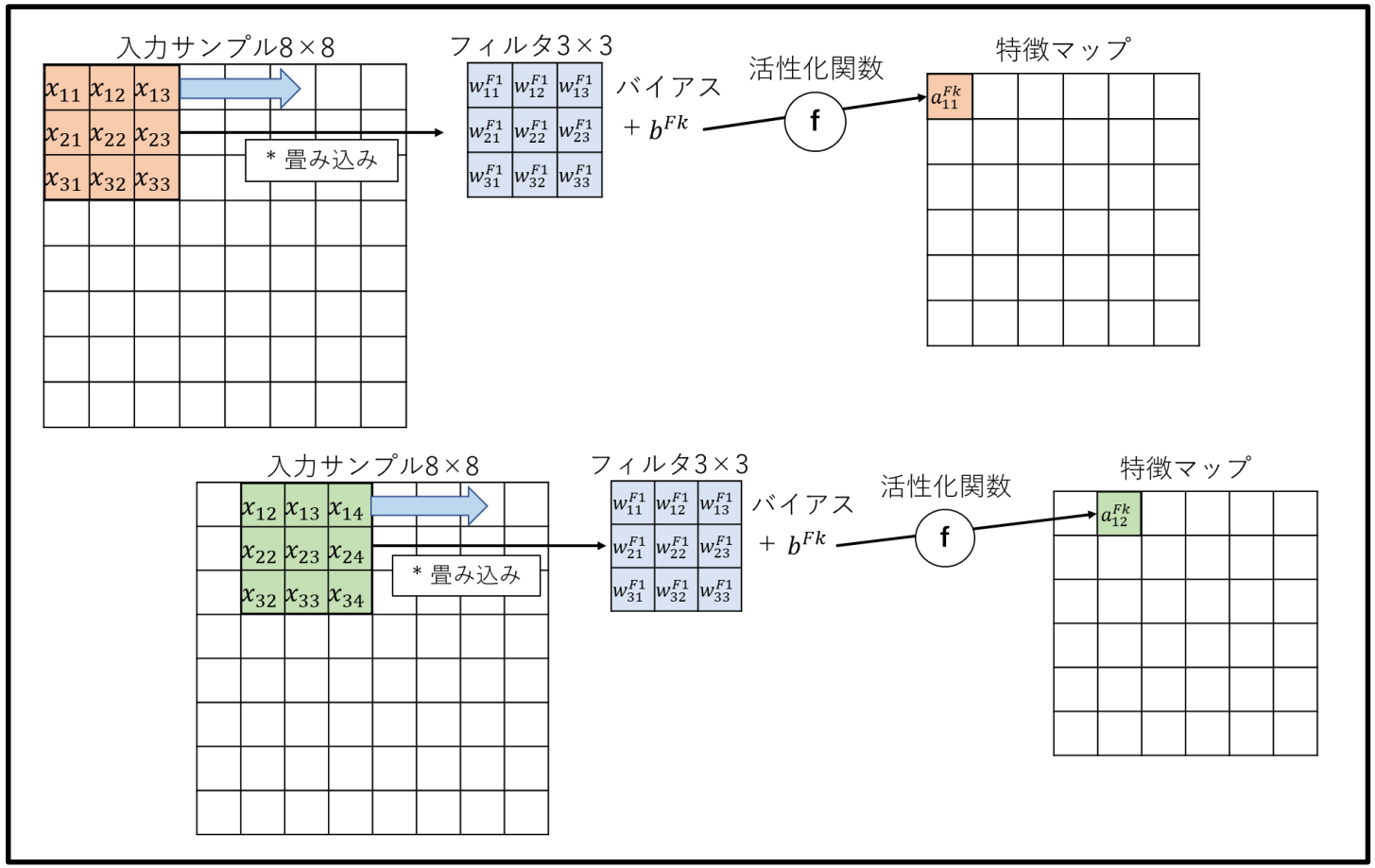
図：学習

「教師あり学習」と「教師なし学習」がある．「教師あり学習」は，学習時にあらかじめ正解の出力データが与えられ(これを学習データという)，その学習データから，重みとバイアスを決定する学習をいう．学習は誤差逆伝播法を用いて行われれる．「教師なし学習」は，正解の出力データは与えられておらず，類似性でグループ分けする学習をいう．

Convolutional Layer

x_{ij}: ユニットに入力される画像の中の画素(i行j列)の値
w_{ij}^{Fk}: k枚目の特徴マップを作るためのフィルターのi行j列の値
Z_{ij}^{Fk}: k枚目i行j列にあるユニットの重み付き入力
b^{Fk}: k枚目i行j列にあるユニットのバイアス
a_{ij}^{Fk}: k枚目i行j列にあるユニットの出力(活性化関数の値)

畳み込み層



畳み込み処理

畳み込み層では，入力サンプルと重みフィルタの畳み込み演算が行われる．フィルタを一定の間隔でスライドさせていき，全体に対して畳み込み処理を行う．

畳み込んだ後に，バイアス(共通の値)を加え，活性化関数に通した値が畳み込み層の出力値となる．入力画像にフィルタをかけ，バイアスを加えたものをZ_{ij}^{Fk}(重み付き入力)とすると，

$$Z_{ij}^{Fk} = w_{11}^{Fk}x_{ij} + w_{11}^{Fk}x_{ij+1} + w_{11}^{Fk}x_{ij+2} + \ldots w_{11}^{Fk}x_{i+2j+2} + b^{Fk}$$

となる．

活性化関数を$\alpha(Z)$とすると，Z_{ij}^{Fk}に対するユニットの出力a_{ij}^{Fk}は，

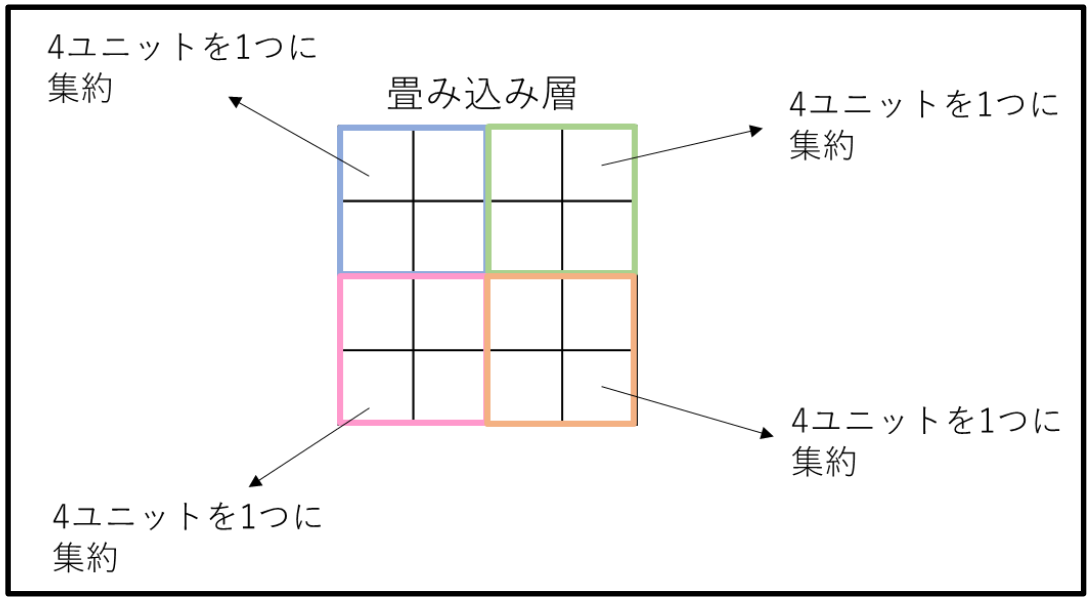
$$a_{ij}^{Fk} = \alpha(Z_{ij}^{Fk})$$

となる．

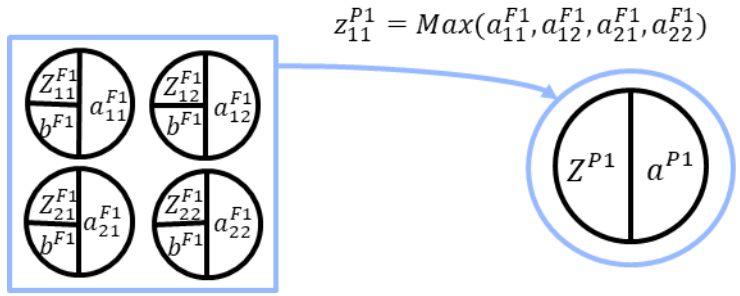
Pooling Layer

Z_{ij}^{Pk}:k枚目i行j列にあるユニットの入力
a_{ij}^{Fk}:k枚目i行j列にあるユニットの出力(入力値Z_{ij}^{Pk}と一致する)
p: p = 1,2,3 …
$P_{i,j}$: i行j列にあるユニットのインデックス集合

Pooling層



プーリング層



畳み込み層の情報を縮約するプーリング層を設ける.
左図では，特徴マップの2×2ユニットを一つのユニットに縮約している.
プーリング層の縮約の仕方には様々な種類がある．対象領域を$a_{2i-12j-1}^{Pk}, a_{2i-12j}^{Pk}, a_{2i2j-1}^{Pk}, a_{2i2j}^{Pk}$の$P_{i,j}$とすると

☆最大プーリング：$Z_{ij}^{Pk} = \mathrm{Max}(a_{2i-12j-1}^{Pk}, a_{2i-12j}^{Pk}, a_{2i2j-1}^{Pk}, a_{2i2j}^{Pk})$

・平均プーリング：$Z_{ij}^{Pk} = \frac{1}{|P_{i,j}|}(a_{2i-12j-1}^{Pk} + a_{2i-12j}^{Pk} + a_{2i2j-1}^{Pk} + a_{2i2j}^{Pk})$

・LPプーリング：

$$Z_{ij}^{Pk} = \left[\frac{1}{|P_{i,j}|}\sum_{(m,n)\in P_{i,j}} {a_{mn}^{Pk}}^{p}\right]^{\frac{1}{p}}$$

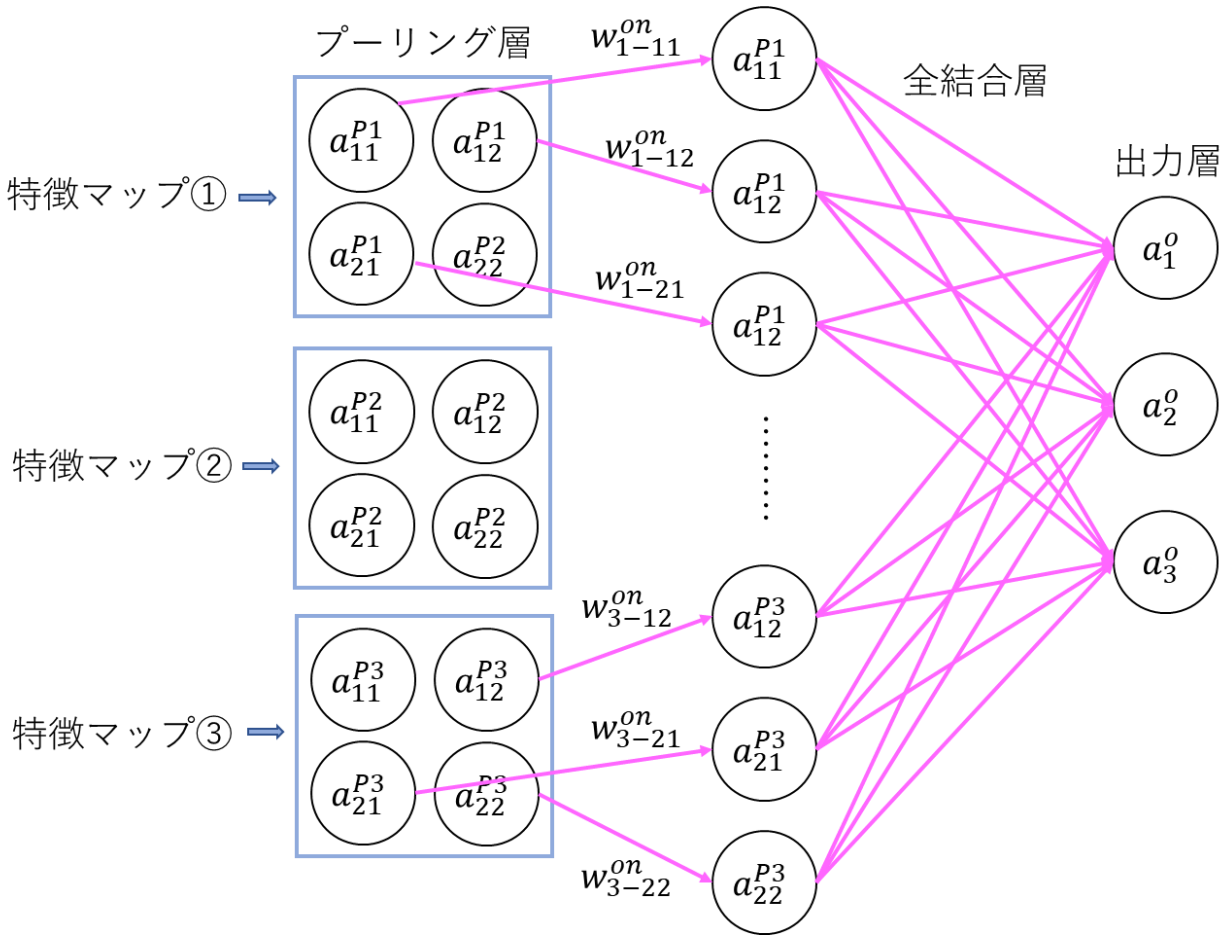
プーリング層のユニットはその重みやバイアスの概念がない．すなわちモデルを定めるパラメーターがない．加えて活性化関数という概念もない．最大値プーリングした時の一般式を以下に示す．

$$\begin{cases} Z_{ij}^{Pk} = \mathrm{Max}(a_{2i-12j-1}^{Pk}, a_{2i-12j}^{Pk}, a_{2i2j-1}^{Pk}, a_{2i2j}^{Pk}) \\ a_{ij}^{Pk} = Z_{ij}^{Pk} \end{cases}$$

Fully connected Layer ・ Output Layer

w_{k-ij}^{on}:k枚目のプーリング層のi行j列にあるユニットから出力層n番目のユニットに向けられた矢の重み
Z_n^o:出力層n番目にあるユニットの重み付き入力
b_n^o:出力層n番目にあるユニットのバイアス
a_n^o:出力層n番目にあるユニットの出力(活性化関数の値)

■ 全結合層・出力層



全結合層では，プーリング層の全ユニットから矢を受ける(全結合)．こうすることによって，プーリング層のユニット情報を総合的に調べることが出来る．

出力層のn番目のユニット($n = 1,2,3 \ldots$)に対して，Z_n^o(重み付き入力)は次のように表される．

$$Z_n^o = w_{1-11}^{on} a_{11}^{P1} + \cdots + w_{2-11}^{on} a_{11}^{P2} + \cdots + w_{3-11}^{on} a_{11}^{P3} + \cdots + b_n^o$$

出力層にあるユニットの出力を考える．CNN全体の出力となる．
出力層のn番目のユニットの出力値をa_n^oとし，活性化関数を$\alpha(Z)$とすると出力は，

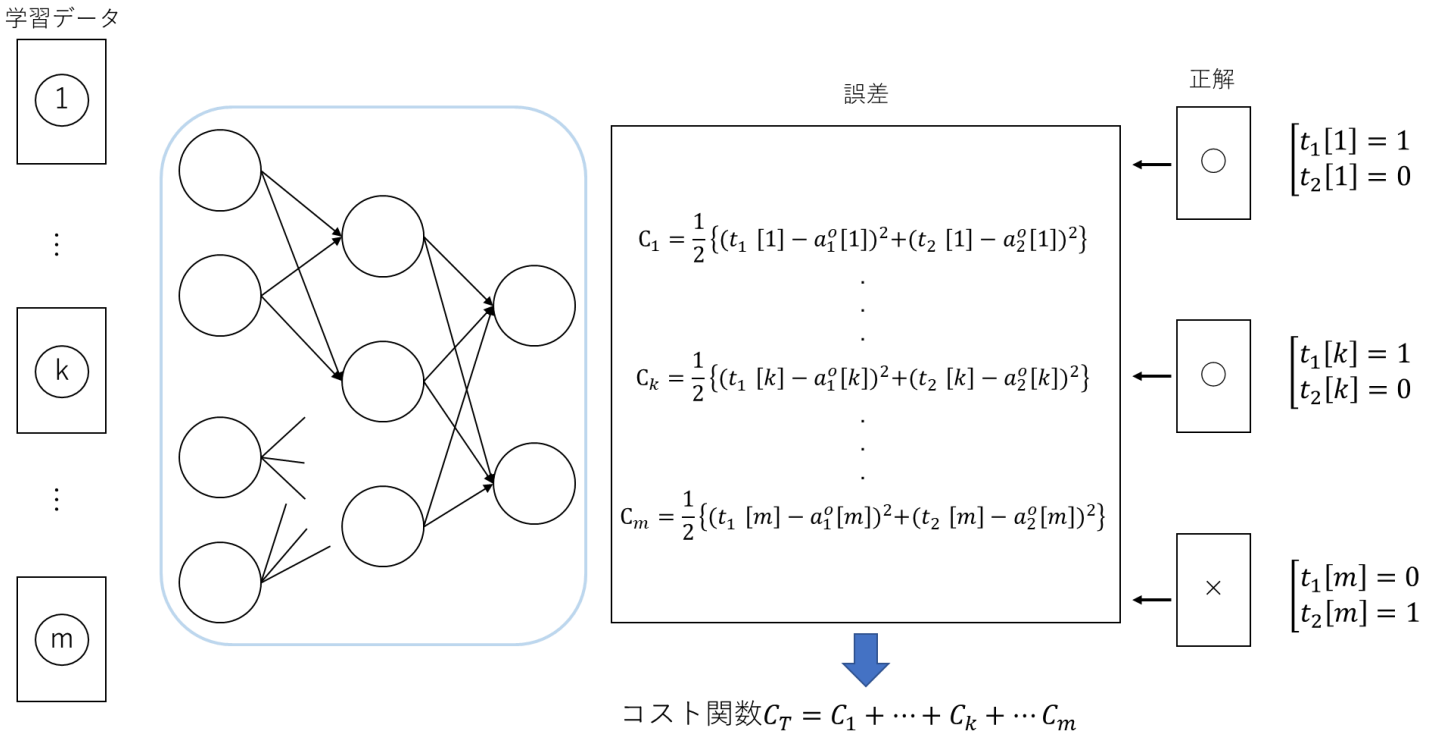
$$a_n^o = \alpha(Z_n^o)$$

となる．

Cost function

a_n^o: 出力層n番目にあるユニットの出力(活性化関数の値)
t_n:出力変数に対する正解

コスト関数



コスト関数(損失関数,誤差関数):モデルのパラメータを用いて表された誤差全体の関数
ニューラルネットワークの重みとバイアスはコスト関数を最小化することによって決まる.

ニューラルネットワークが出す予測値と正解との2乗誤差Cの値をC_kとすると次のように表現される.

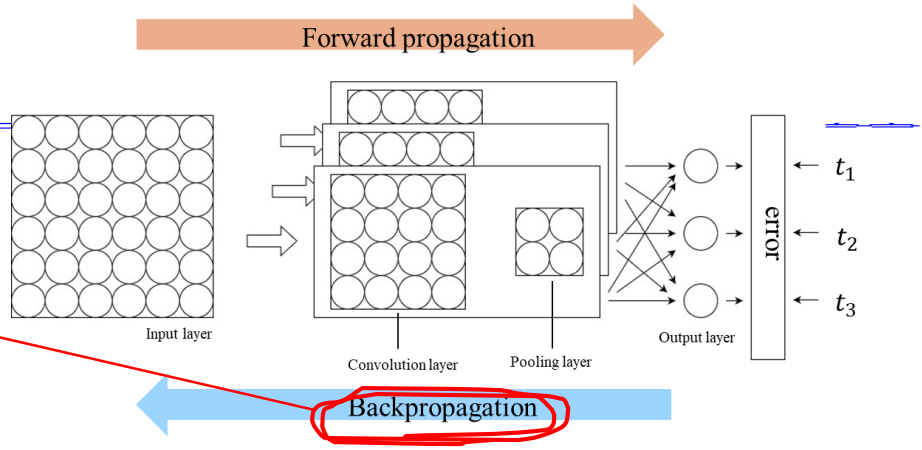
$$C_k = \frac{1}{2}\{(t_1[k] - a_1^o[k])^2 + (t_2[k] - a_2^o[k])^2\}$$

学習データ全体について2乗誤差したものを合わせたものがコスト関数である.

$$C_T = C_1 + \cdots + C_k + C_m$$

このコスト関数を最小化するために勾配均等法を用いるが, コスト関数が複雑なために勾配成分を具体的な形で求めるのは困難
→誤差逆伝播法を用いる

Back-propagation



誤差逆伝播法(Back-propagation)

誤差逆伝播法は誤差が小さくなるように全体の結合重みを調整する.

誤差逆伝播法の基本的な考え方

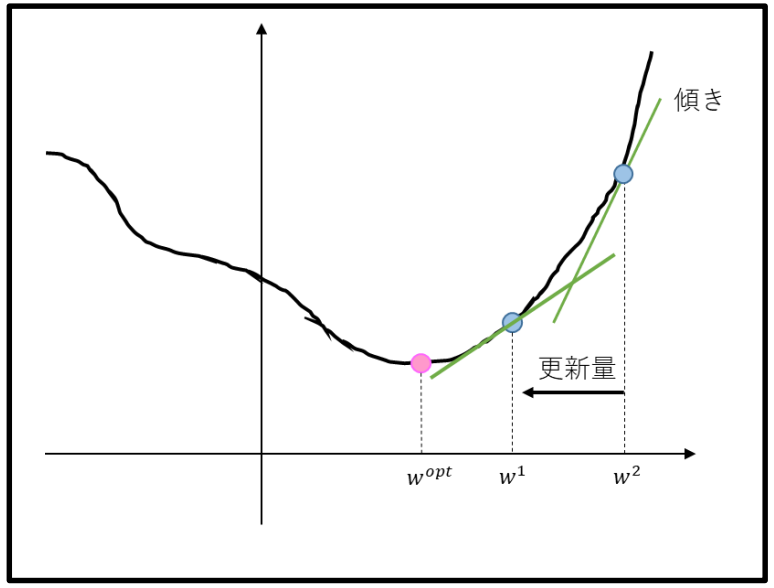


図:勾配降下法

勾配降下法:方程式から直接求めるのではなく,グラフ上の点を少しずつ動かしながら,手探りで関数値の最小点を探し出す手法.
多変数関数の最小値を探す問題には勾配降下法が有効.

勾配降下法をニューラルネットワークに適用する

滑らかな関数$f(x_1, x_2, \dots, x_n)$において,変数を順に

$$x_1 + \Delta x_1, x_2 + \Delta x_2, x_n + \Delta x_n$$

と微小にそれぞれ変化させたとき,関数fが最も減少するのは次の関数が成り立つときである.(この時のηは正の小さな定数とする.)

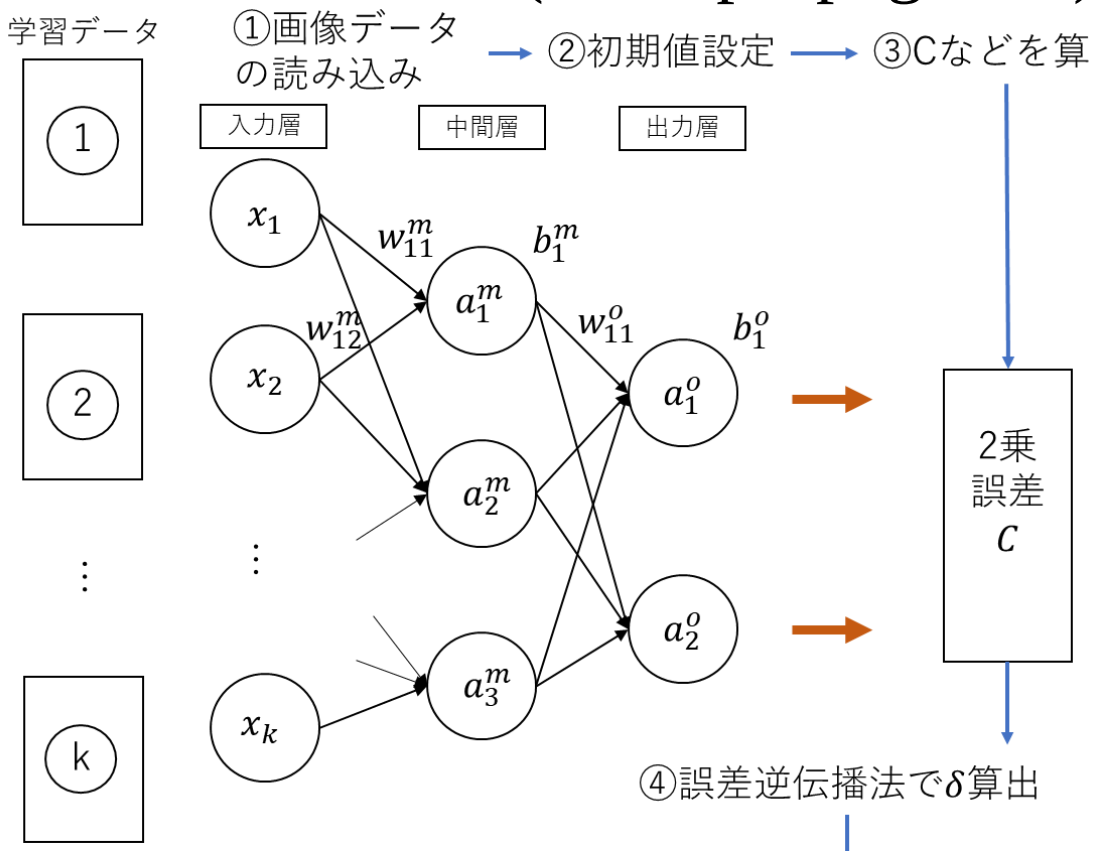
$$(\Delta x_1, \Delta x_2, \dots, \Delta x_n) = -\eta \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

↑これを勾配という.

Back-propagation

w_{k-ij}^{n}:k枚目のプーリング層のi行j列にあるユニットから出力層n番目のユニットに向けられた矢の重み
w_{ij}^{m}:i行j列にあるユニットから，中間層に向けられた矢の重み
b_n^m: 中間層n番目にあるユニットのバイアス　　δ_j^m: 中間層のユニットの誤差
a_n^m: 中間層n番目にあるユニットの出力　　δ_j^o: 出力層のユニットの誤差
Z_n^m : 中間層n番目にあるユニットの重み付き入力
b_n^o: 出力層n番目にあるユニットのバイアス
a_n^o: 出力層n番目にあるユニットの出力
z_n^o : 出力層n番目にあるユニットの重み付き入力

■ 誤差逆伝播法(Back propagation)



③ユニットの出力値，2乗誤差Cの値C_kを出力する

<中間層>

$$Z_k^m = \sum_{j=0}^{k} w_{ij}^m x_j + b_j^m$$

$$a_i^m = \alpha(Z_i^m)$$

<出力層>

$$Z_k^o = \sum_{j=0}^{k} w_{ij}^o x_j + b_j^o$$

$$a_i^o = \alpha(Z_i^o)$$

2乗誤差

$$C_k = \frac{1}{2}\{(t_1\ [k] - a_1^o[k])^2 + (t_2\ [k] - a_2^o[k])^2\}$$

④誤差逆伝播法から各層のユニット誤差δを求める

出力層 : $\delta_j^o = \frac{\partial C}{\partial a_j^o} a'(Z_j^o)$

中間層 : $\delta_i^m = \{\sum_{k=0}^{n} \delta_k^{m+1} w_{ki}^{m+1}\} \alpha'(Z_i^m)$

⑤ユニットの誤差から2乗誤差Cの偏微分を求める

$$\frac{\partial C}{\partial w_{ji}^l} = \delta_j^l a_i^{l-1}, \qquad \frac{\partial C}{\partial b_j^l} = \delta_j^l \ (l = \text{層番号})$$

⑥コスト関数C_Tと勾配ΔC_T

$$C_T = C_1 + \cdots + C_k + C_m$$

$$\Delta C_T = \Delta C_1 + \cdots + \Delta C_k + \Delta C_m$$

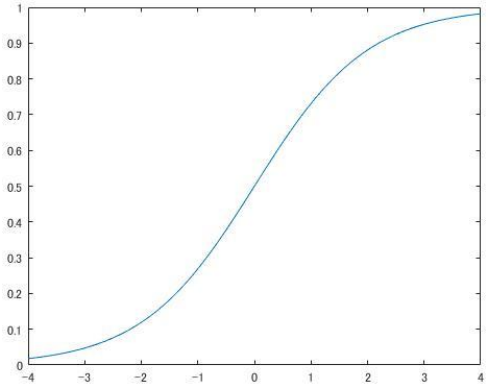
Activation function

■ 活性化関数

ニューラルネットワークにおいて入力信号の総和を出力信号に変換する関数のこと.
勾配の発散，消失問題を防いだり，学習効率を上げるために，様々な活性化関数が利用されている.
活性化関数は中間層と出力層で区別して設計されている.

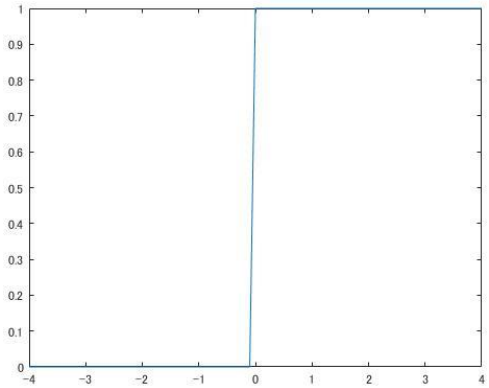
■ 中間層

☆シグモイド関数



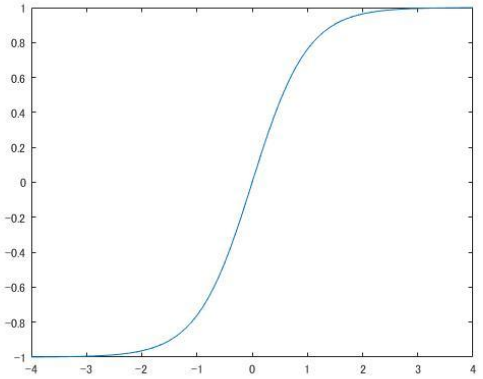
$$y = \frac{1}{1 + \exp(-x)}$$

・ステップ関数



$$y = \begin{matrix} 0\ (x \leqq 0) \\ 1\ (x > 0) \end{matrix}$$

・tanh関数



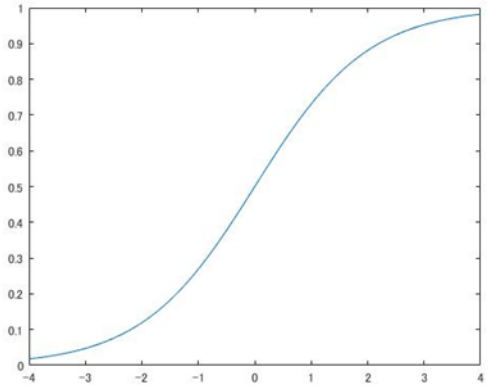
$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Activation function

活性化関数

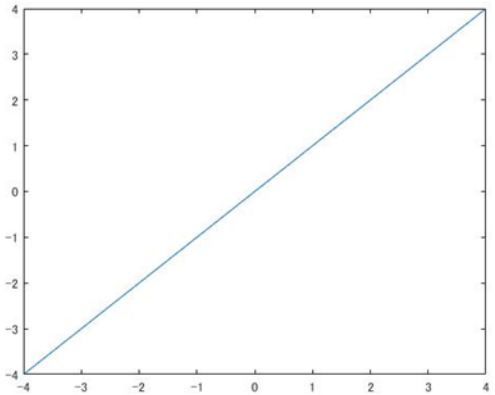
出力層

☆シグモイド関数



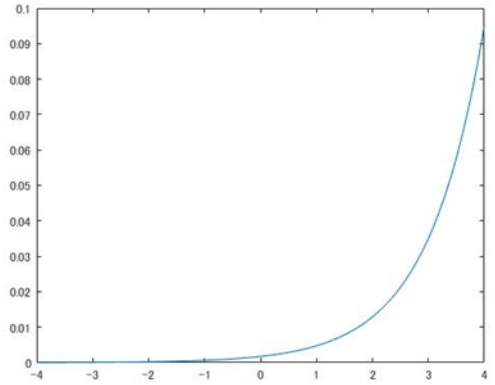
$$y = \frac{1}{1 + \exp(-x)}$$

・恒等関数



$$y = x$$

・ソフトマックス関数



$$y = \frac{\exp(x_k)}{\sum_{i=1}^{K} \exp(x_i)}$$

Padding

Zero padding

0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	1	1	0	0	1	0
0	0	0	0	1	0	1	0
0	1	0	1	0	1	1	0
0	0	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

ゼロパディング：左図のように，入力値の周りを0で埋めること．
畳み込むと画像のサイズが小さくなってしまうので，周りに0を埋めてサイズが変わらないようにする．

ゼロパディングで得られるメリット

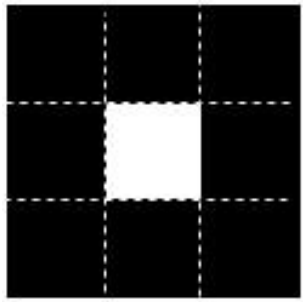
- 端のデータに対する畳み込み回数が増えるので端の特徴も得られる
- 畳み込み演算の回数が増えるのでパラメーターの更新が多く実行される
- カーネルのサイズや，層の数を調整できる

畳み込み層とプーリング層によりサイズは次第に小さくなるので，ゼロパディングすることでサイズを増やすと，層の数を増やすことが出来る．

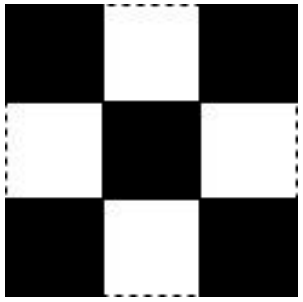
MATLAB演習：〇×判定のCNN

■ 〇×判定

以下のような3×3画素からできている〇と×を判定するCNNを作成する.

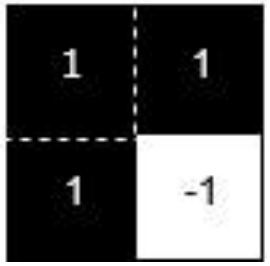


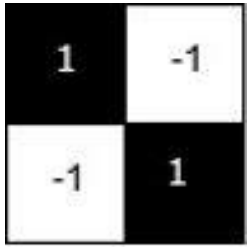
〇



×

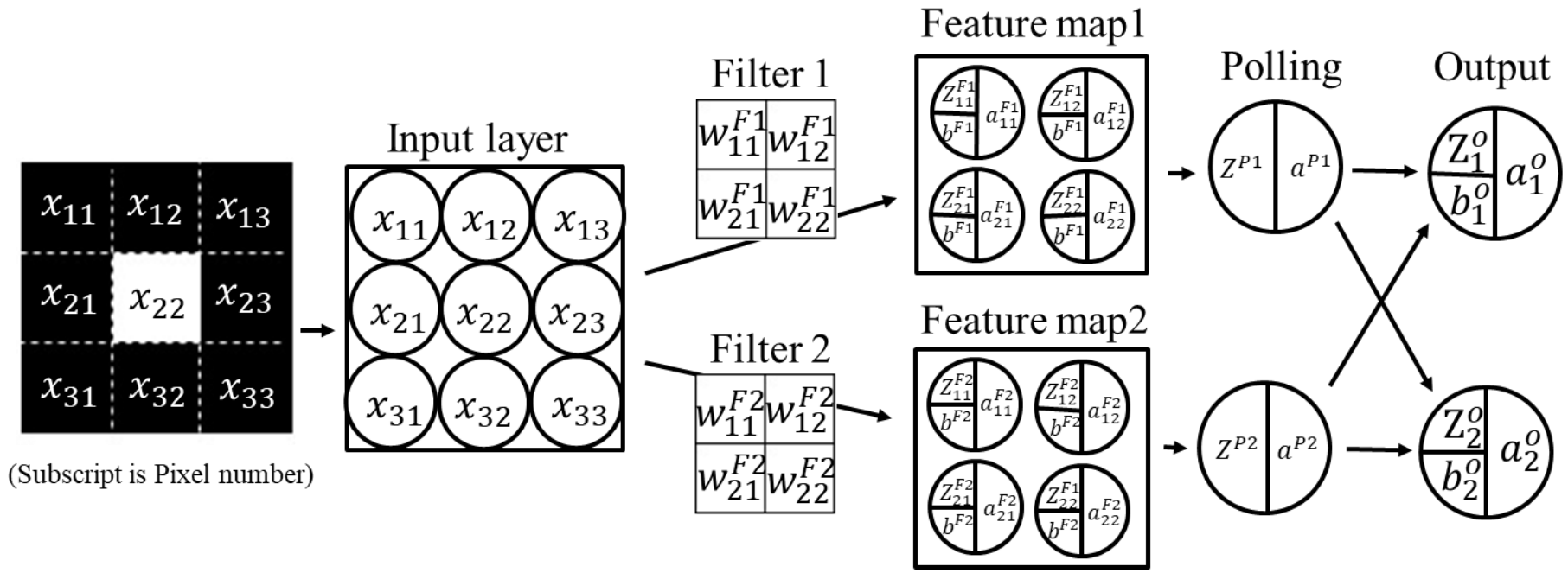
畳み込む際に用いるフィルターを以下に示す.





MATLAB演習：○×判定のCNN

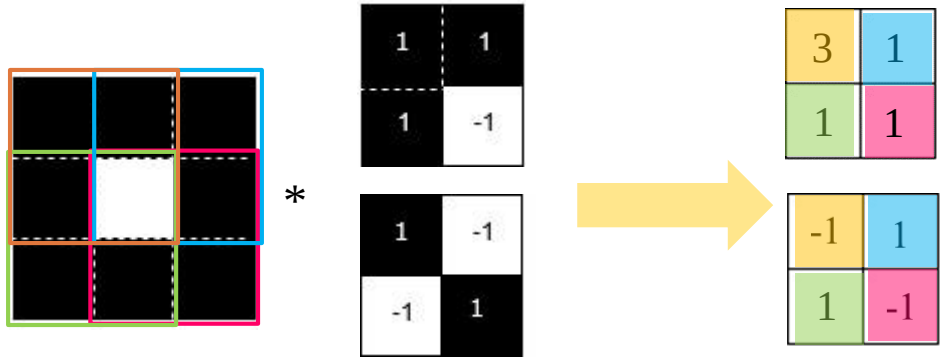
■ 判定詳細



Convolution (filter) Layer

畳み込み（フィルター）層

Filter used for convolution

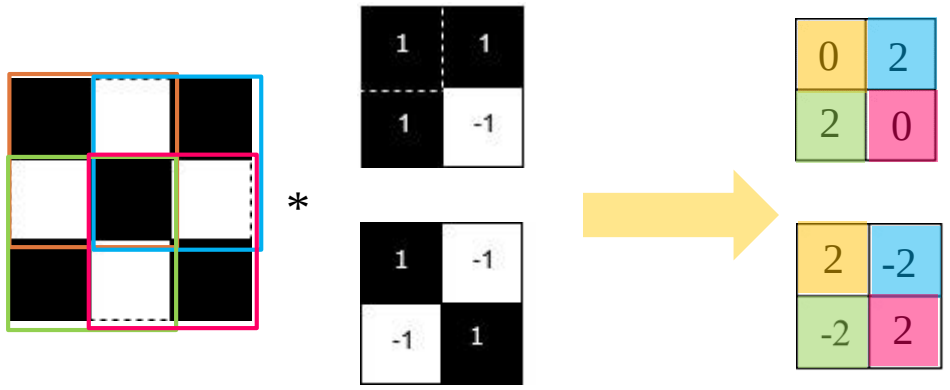


Convolution result:

$$c_{ij}^{Fk} = w_{11}^{Fk} x_{ij} + w_{12}^{Fk} x_{ij+1} + \cdots + w_{22}^{Fk} x_{i+1j+1}$$

Weighted input:

$$Z_{ij}^{Fk} = w_{11}^{Fk} x_{ij} + w_{12}^{Fk} x_{ij+1} + \cdots + w_{22}^{Fk} x_{i+1j+1} + b^{Fk}$$



Polling layer

Case:○

3	1
1	1

→ 3

1	-1
-1	1

→ 1

Case：×

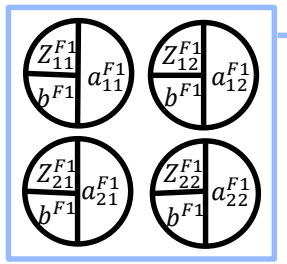
2	-2
-2	2

→ 2

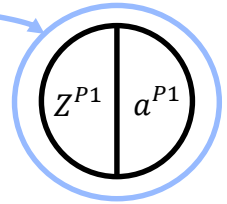
0	2
2	0

→ 2

Feature map1



$Z_{11}^{P1} = Max(a_{11}^{F1}, a_{12}^{F1}, a_{21}^{F1}, a_{22}^{F1})$



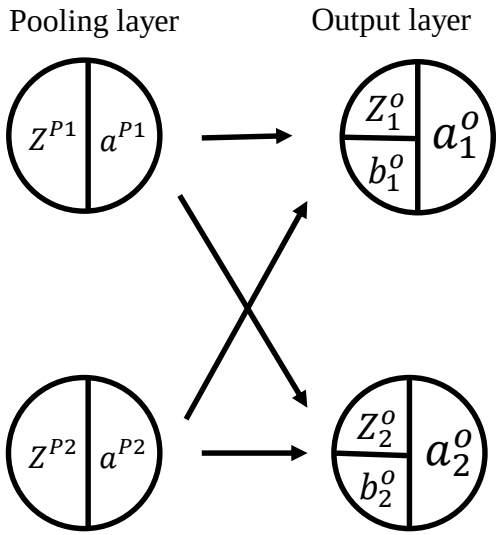
Pooling layer

Activation function : $a(x) = x$

プーリング層における入力には重み,バイアス,活性化の概念が無いため,入力と出力の値は同じになる.
※プログラムでは直接a^{P1}を算出

Output layer

出力層では,〇と×を区別する
ために2つのユニットを用意.



出力層のn番目のユニットの重み付き入力は以下の式で表される.

$$Z_n^o = w_{1-11}^{on} a_{11}^{P1} + w_{1-12}^{on} a_{12}^{P1} + w_{2-11}^{on} a_{11}^{P2} + w_{2-12}^{on} a_{12}^{P2} + b_n^o$$

係数 w_{k-ij}^{on} は,n番目のユニットによるプーリング層
$a_{ij}^{Pk}(k = 1,2: i = 1,2: j = 1,2)$ に与えられる重み.
b_n^o は出力層のn番目のユニットのバイアス.

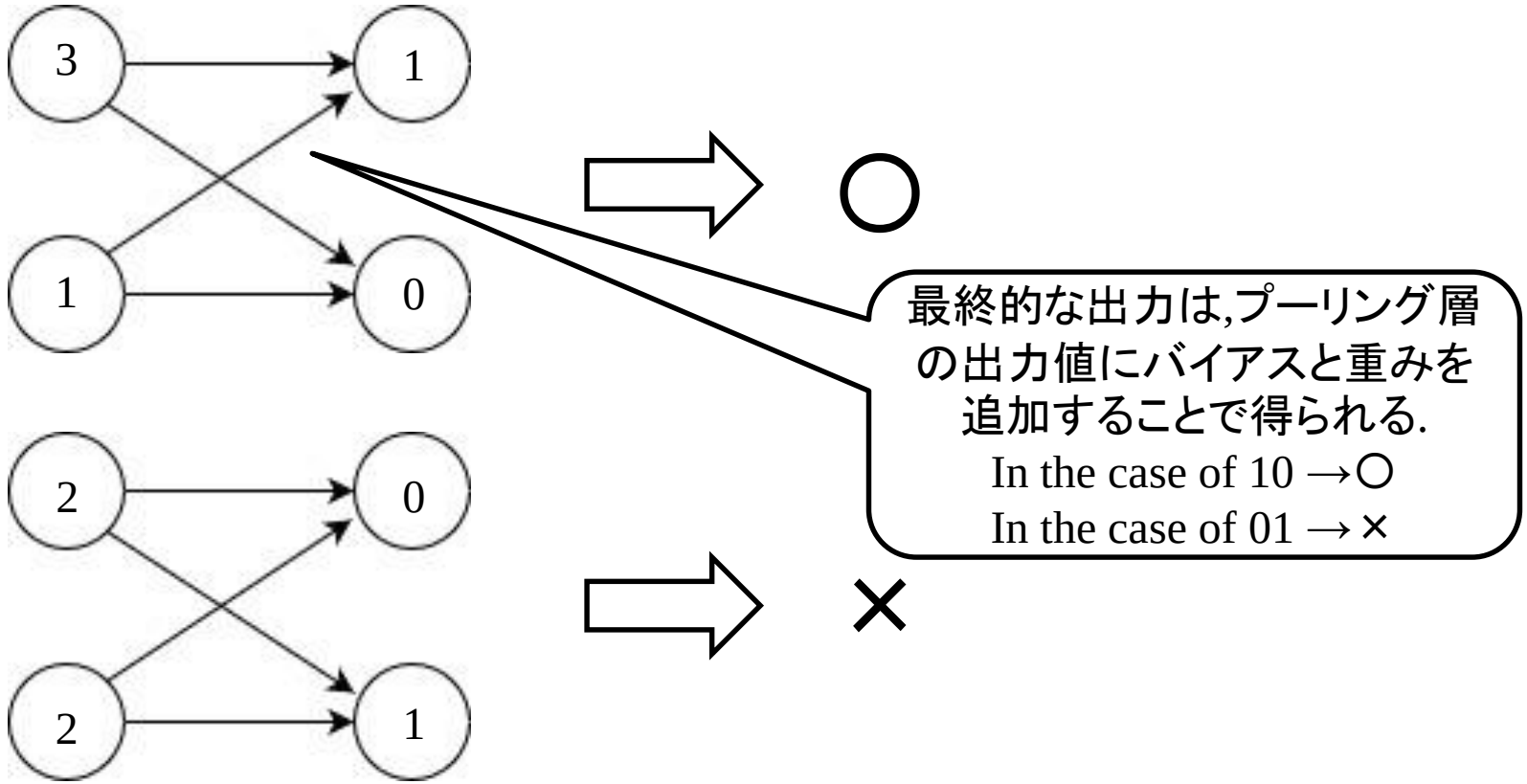
出力層のn番目のユニットの出力値が a_n^o ,
その活性化関数が $a(Z)$ の場合,

$$a_n^o = a(Z_n^o)$$

a_n^o における最大のnは判定番号

Fully connected layer & Output layer

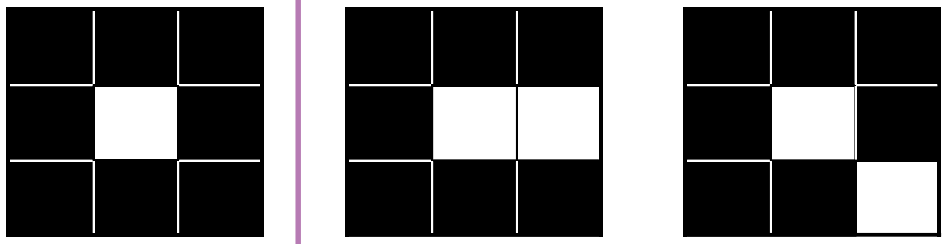
- Fully connected layer and output layer



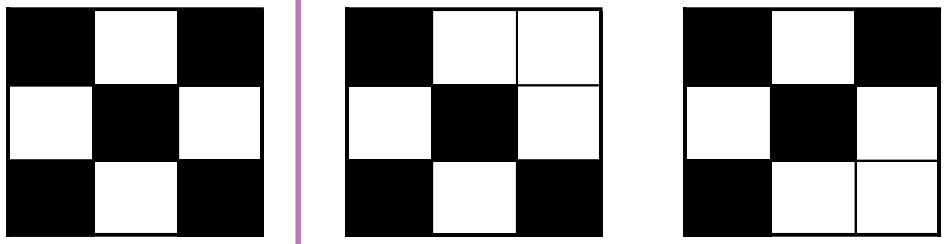
今回のプログラムで用いた教師データ

以下のスライドでは,この部分の
教師データを元に説明していく.

〇の教師データ ：



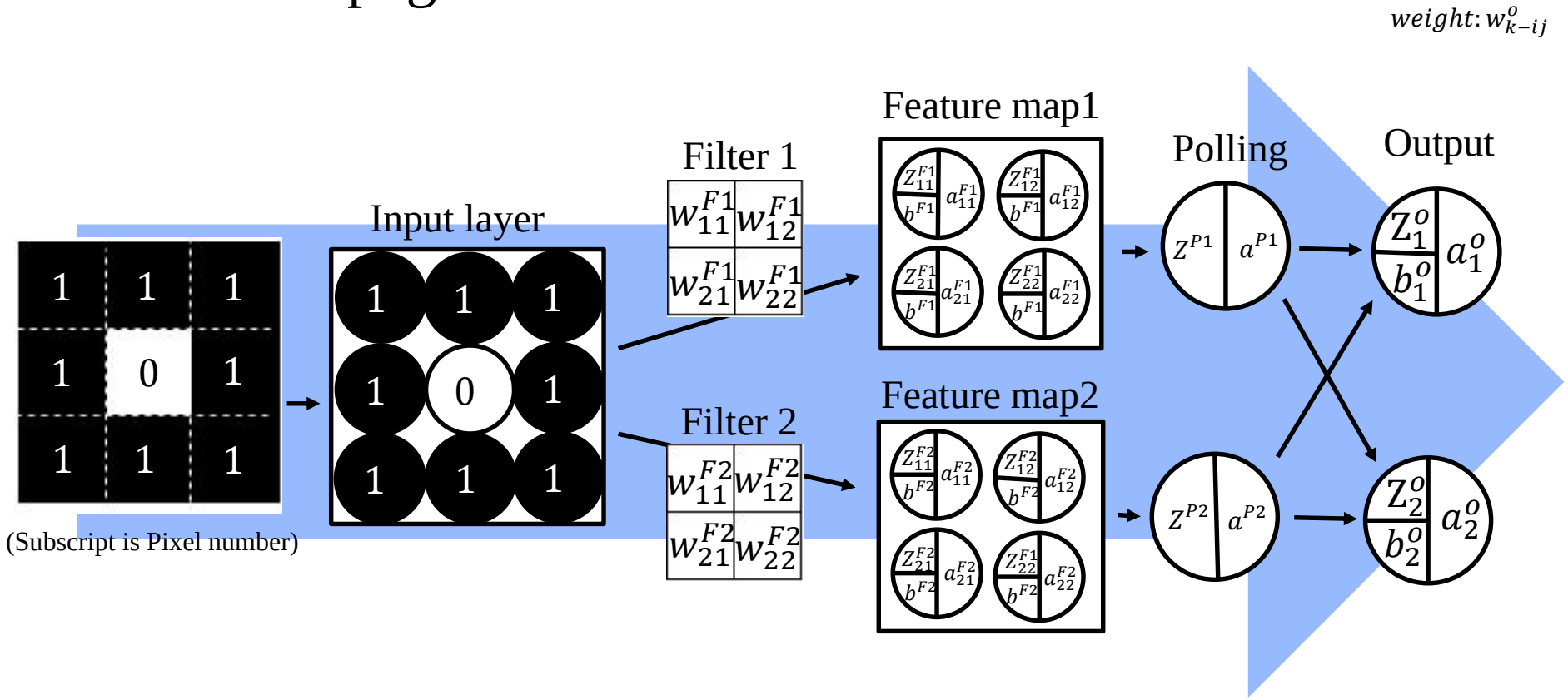
×の教師データ ：



プログラム内では３×３の行列として
　黒 → １
　白 → ０
で入力を行った。

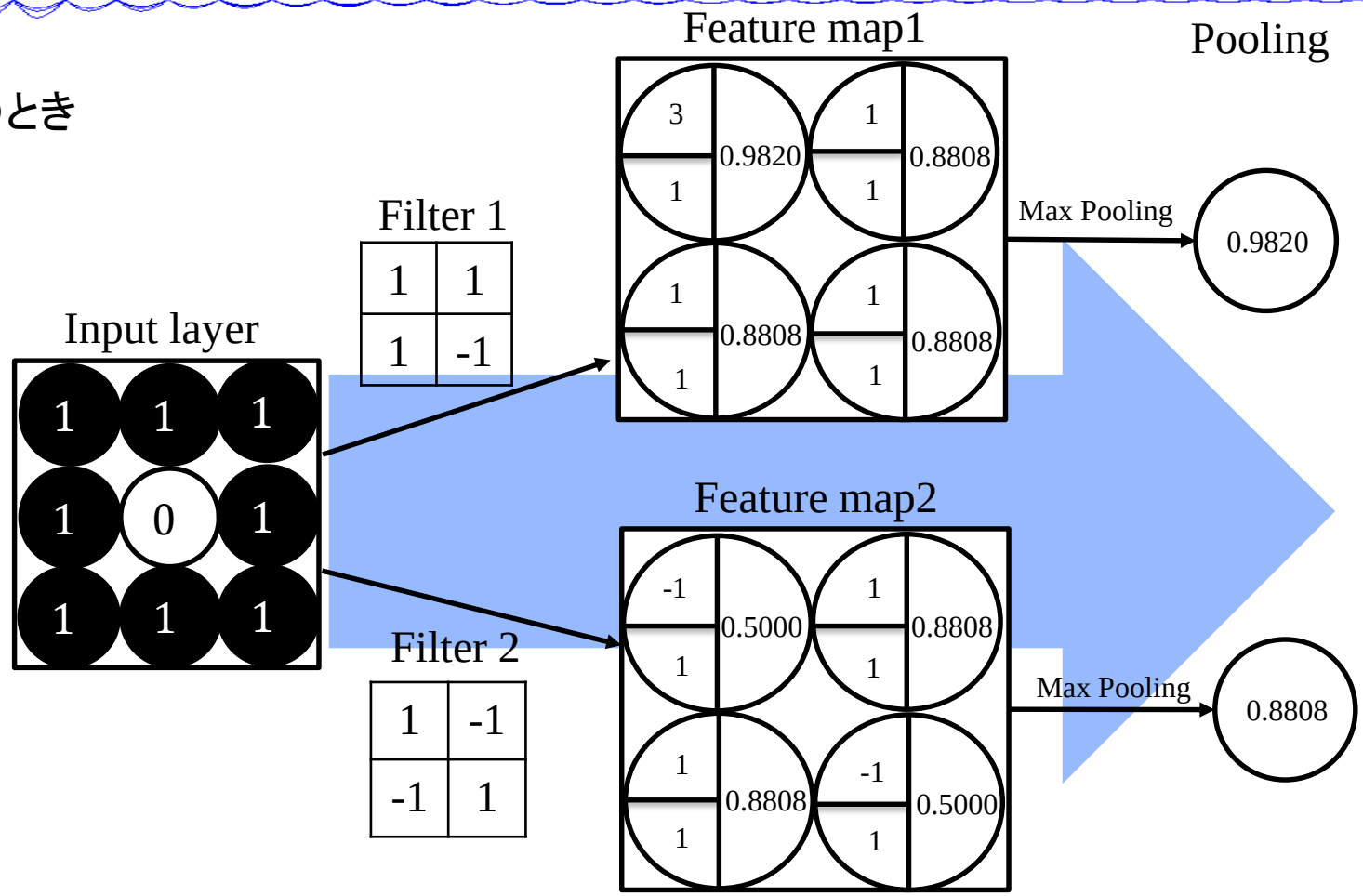
Forward Propagation

Forward Propagation

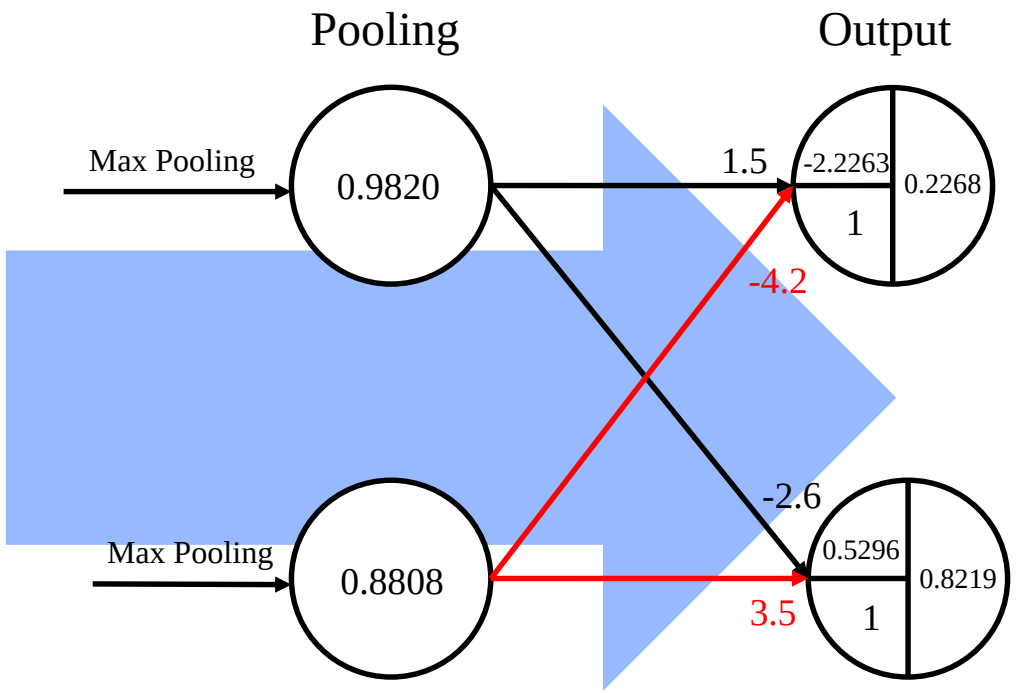


Convolutional Layer & Pooling Layer

〇が教師のとき



Pooling & Output Layer



出力値	正解値
0.2268	1
0.8219	0

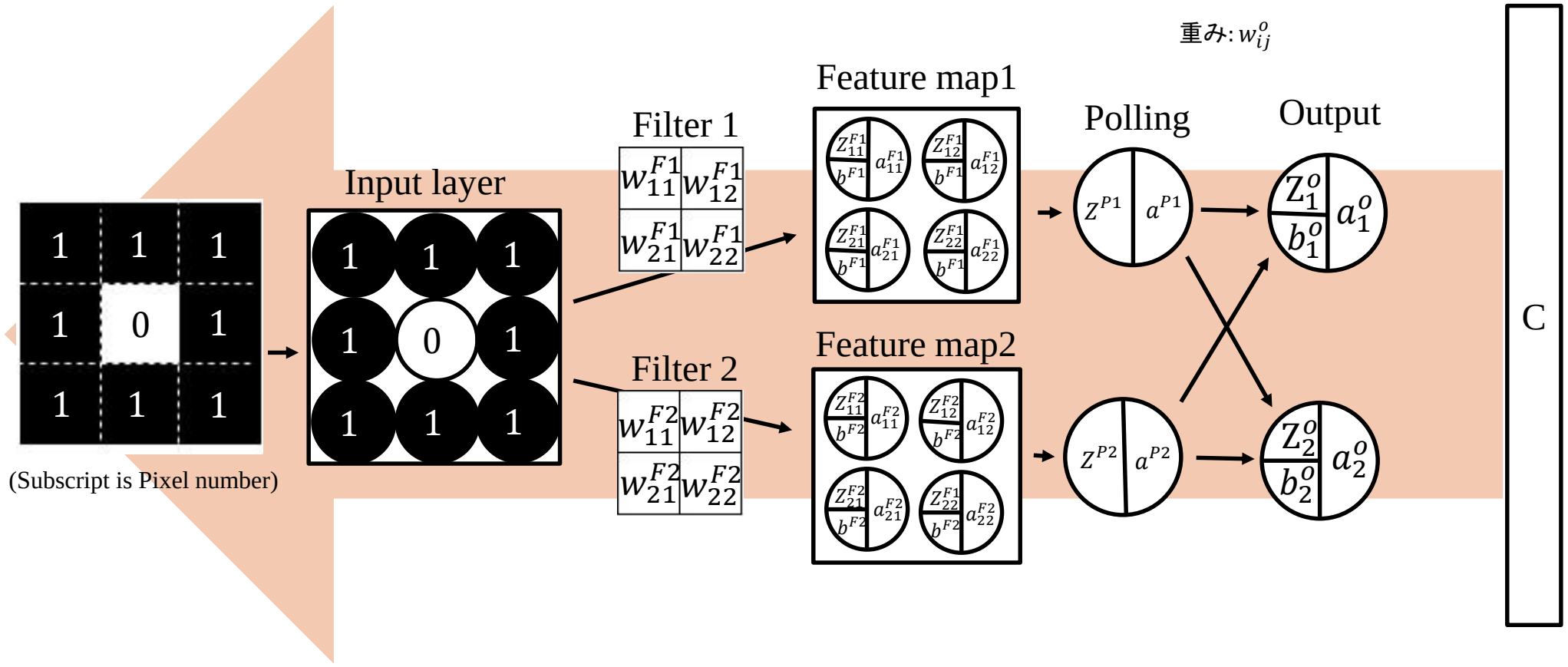
出力値が正解値に近づくように
重みとバイアスの更新が必要.

Back Propagation

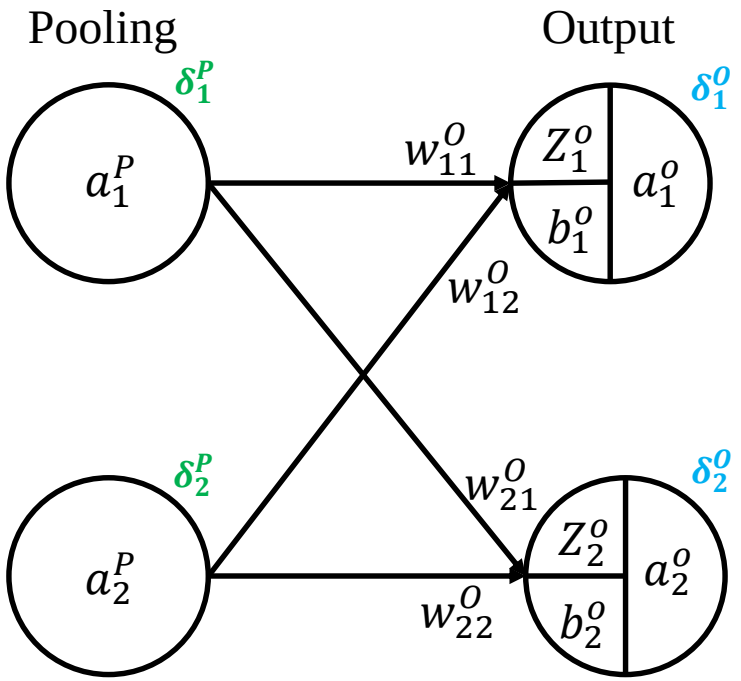
Back Propagation

C_k：出力値と正解値との2乗誤差

$$C_k = \frac{1}{2}\{(t_1\ [k] - a_1^o[k])^2 + (t_2\ [k] - a_2^o[k])^2\}$$



Back Propagation （Output → Pooling）



$$\delta_i^O = (a_i^3 - t_i)a'(Z_i^O) \quad \cdots (1)$$

$$a'(Z_i^O) = a(Z_i^O)(1 - a(Z_i^O))$$

$$\frac{\partial C}{\partial w_{j\,i}^O} = \delta_i^O a_j^p \quad \cdots (2)$$

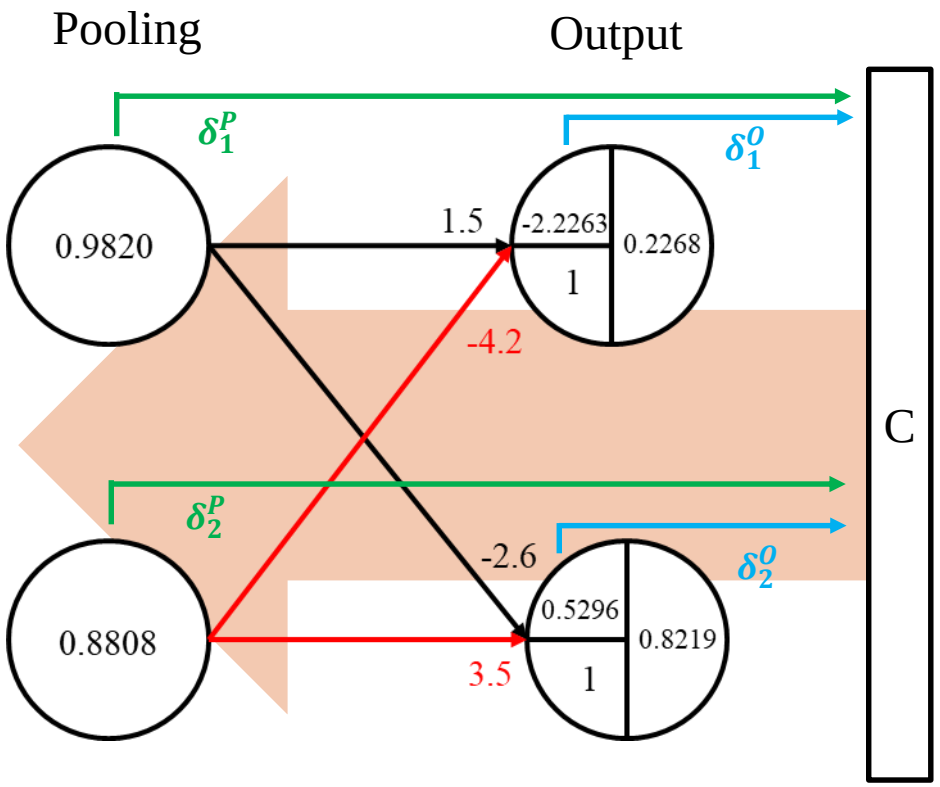
$$\frac{\partial C}{\partial b_i^3} = \delta_i^O \quad \cdots (3)$$

$$\boldsymbol{\delta_i^P} = (\delta_i^O w_{j\,i}^O + \delta_2^O w_{j\,i}^O)a'(Z_i^P) \quad \cdots (4)$$

$$w_{j\,i}^O = w_{j\,i}^O - \eta \frac{\partial C}{\partial w_{j\,i}^O} \quad \cdots (5)$$

$$b_i^O = b_i^O - \eta \frac{\partial C}{\partial b_i^O} \quad \cdots (6)$$

Back Propagation （Output → Pooling）



C_k = 0.6367

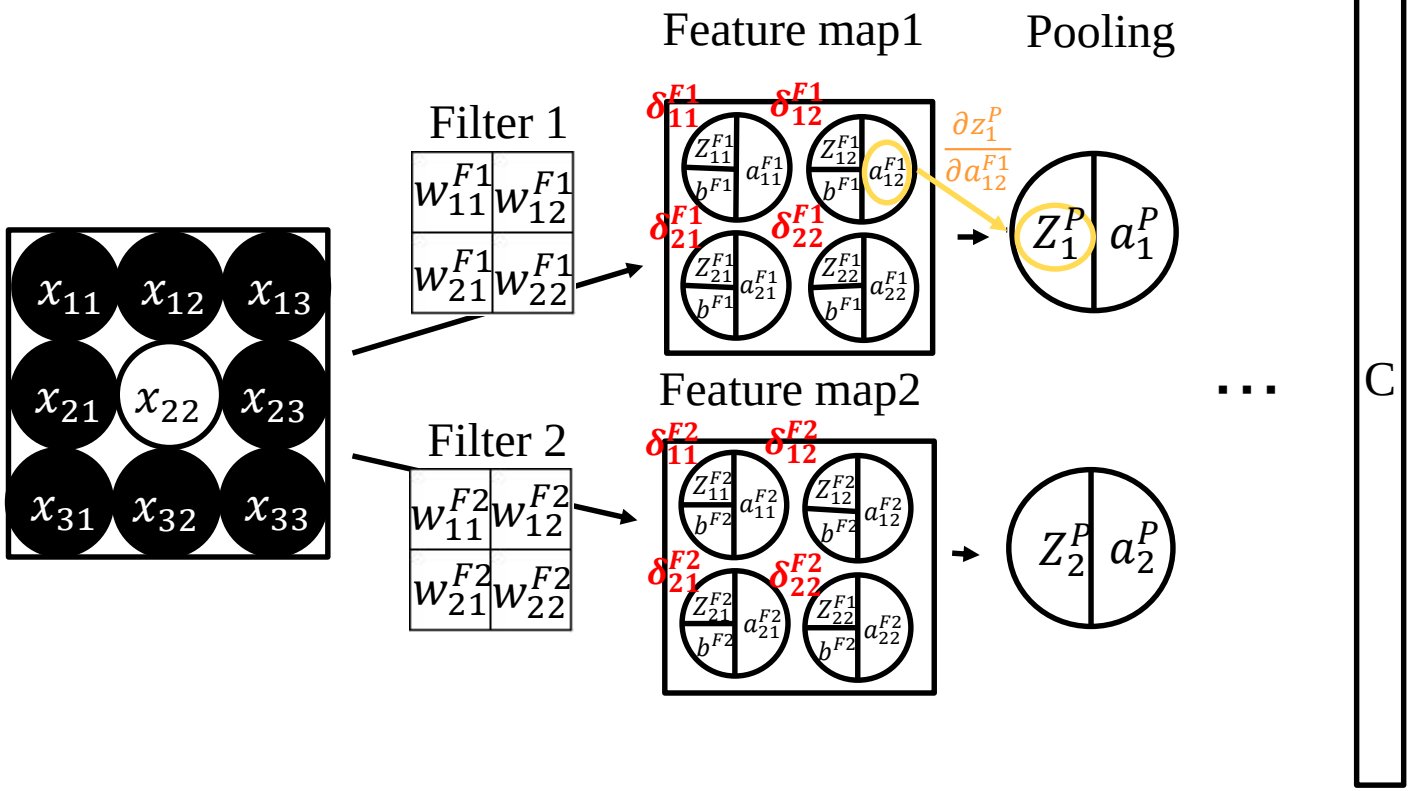
$$\delta_i^O = (a_i^3 - t_i)a'(Z_i^O) \quad \cdots (1)$$
$$a'(Z_i^O) = a(Z_i^O)(1 - a(Z_i^O))$$

$$\delta_i^P = (\delta_i^O w_{ji}^O + \delta_2^O w_{ji}^O)a'(Z_i^P) \quad \cdots (4)$$

$\delta_1^O = -0.1356$ $\delta_1^P = -0.0091$

$\delta_2^O = 0.1203$ $\delta_2^P = 0.1040$

Back Propagation （Pooling → Convolutional）



$$\frac{\partial Z_i^P}{\partial a_{ij}^{Fk}} = \begin{cases} 1 \ (a_{ij}^{Fk} \text{ is Max}) \\ 0 \ (a_{ij}^{Fk} \text{ is not Max}) \end{cases} \cdots (7)$$

$$\boldsymbol{\delta_{ij}^{Fk}} = \left(\boldsymbol{\delta_1^O} w_{k1}^O + \boldsymbol{\delta_2^O} w_{k2}^O \right) \times \frac{\partial Z_i^P}{\partial a_{ij}^{Fk}} \times a'\left(\mathrm{Z_{ij}^{Fk}}\right) \cdots (8)$$

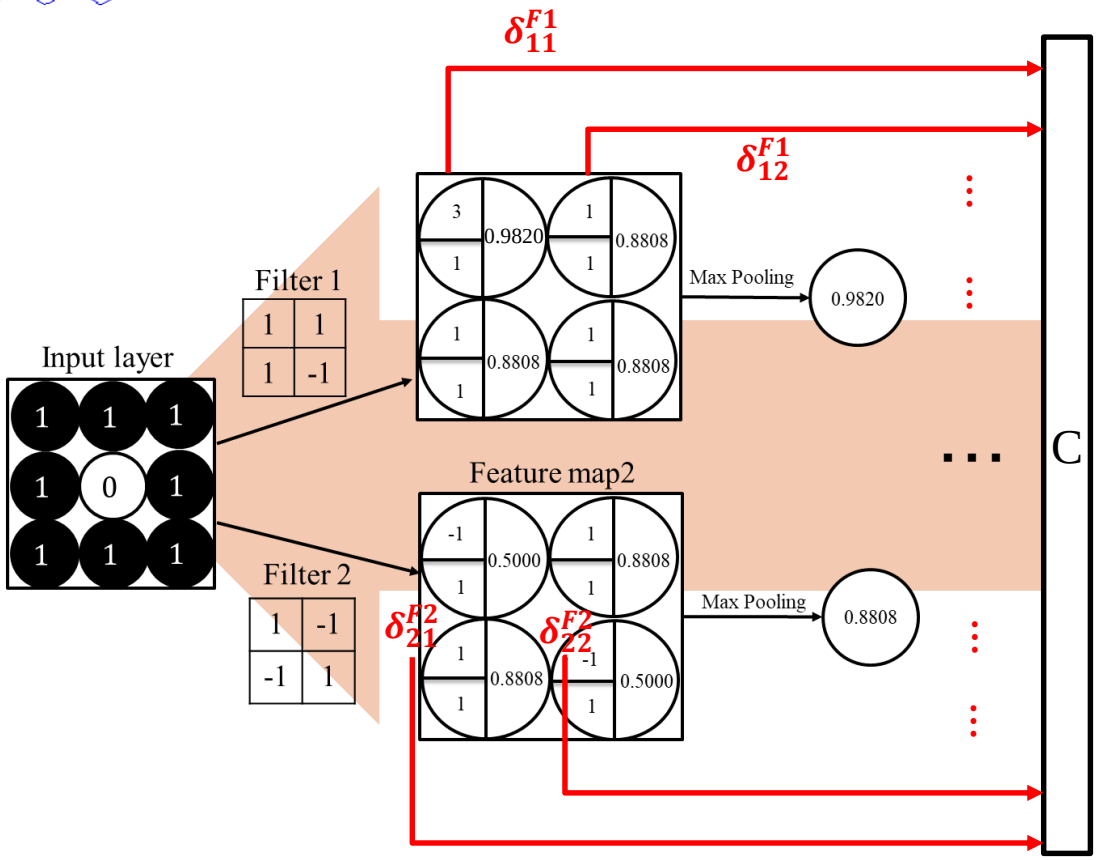
$$\frac{\partial \mathrm{C}}{\partial w_{j\,i}^{Fk}} = \boldsymbol{\delta_{11}^{Fk}} x_{\boldsymbol{ji}} + \boldsymbol{\delta_{12}^{Fk}} x_{\boldsymbol{ji+1}} + \boldsymbol{\delta_{21}^{Fk}} x_{\boldsymbol{j+1\,i}} + \boldsymbol{\delta_{22}^{Fk}} x_{\boldsymbol{j+1\,i+1}} \cdots (9)$$

$$\frac{\partial \mathrm{C}}{\partial b^{Fk}} = \boldsymbol{\delta_{11}^{Fk}} + \boldsymbol{\delta_{12}^{Fk}} + \boldsymbol{\delta_{21}^{Fk}} + \boldsymbol{\delta_{22}^{Fk}} \cdots (10)$$

$$w_{j\,i}^{Fk} = w_{j\,i}^{Fk} - \eta \frac{\partial C}{\partial w_{j\,i}^{Fk}} \cdots (11)$$

$$b^{Fk} = b^{Fk} - \eta \frac{\partial C}{\partial b^{Fk}} \cdots (12)$$

Back Propagation （Pooling → Convolutional）



$$\delta_1^O = -0.1356 \quad \delta_2^O = 0.1203$$

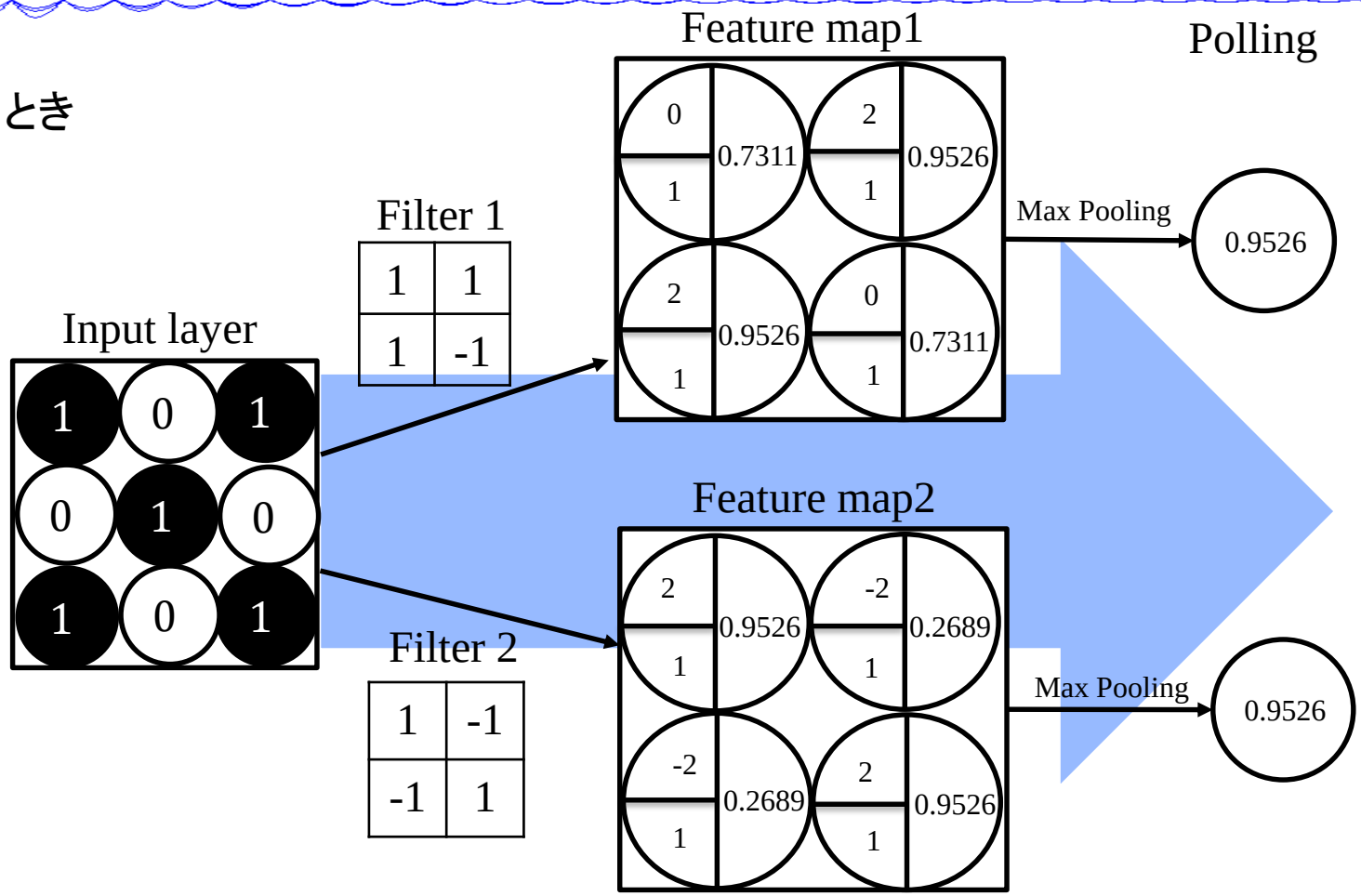
$$\delta_{ij}^{Fk} = \left(\delta_1^O w_{k1}^O + \delta_2^O w_{k2}^O\right) \times \frac{\partial Z_i^P}{\partial a_{ij}^{Fk}} \times a'\left(Z_{ij}^{Fk}\right) \cdots (8)$$

$$\delta_{11}^{F1} = (1.0e-0.3) \times -0.1610 \quad \delta_{12}^{F1} = 0$$
$$\delta_{21}^{F1} = 0 \quad \delta_{22}^{F1} = 0$$

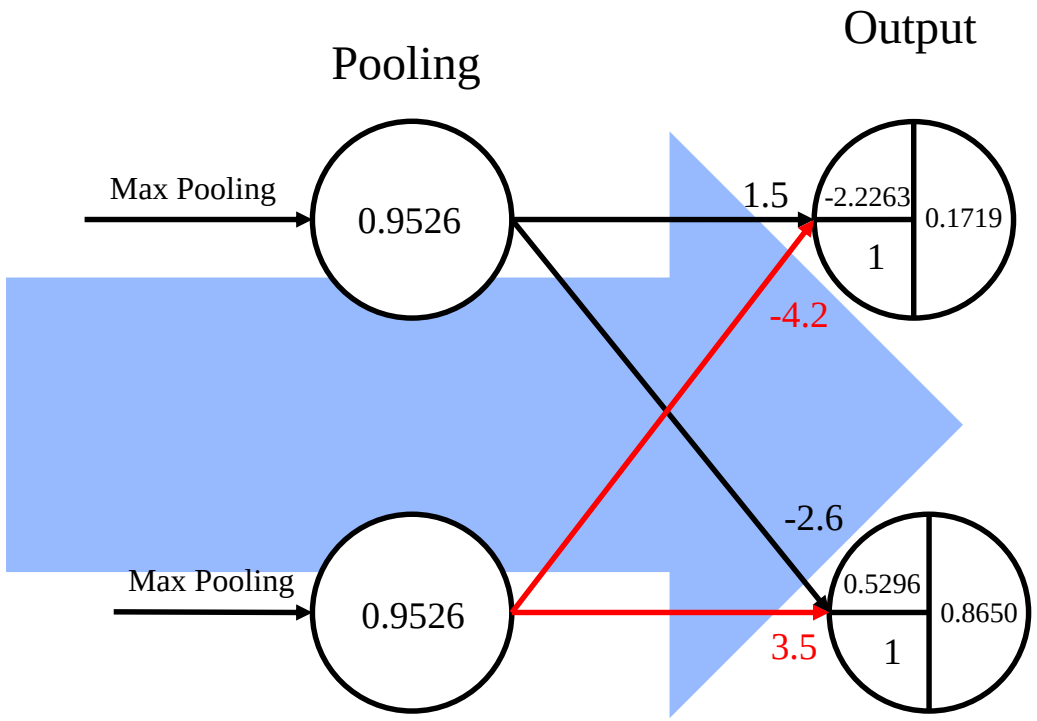
$$\delta_{11}^{F2} = 0 \quad \delta_{12}^{F2} = 0.0109$$
$$\delta_{21}^{F2} = 0.0109 \quad \delta_{22}^{F1} = 0$$

Convolutional & Pooling Layer

×が教師のとき

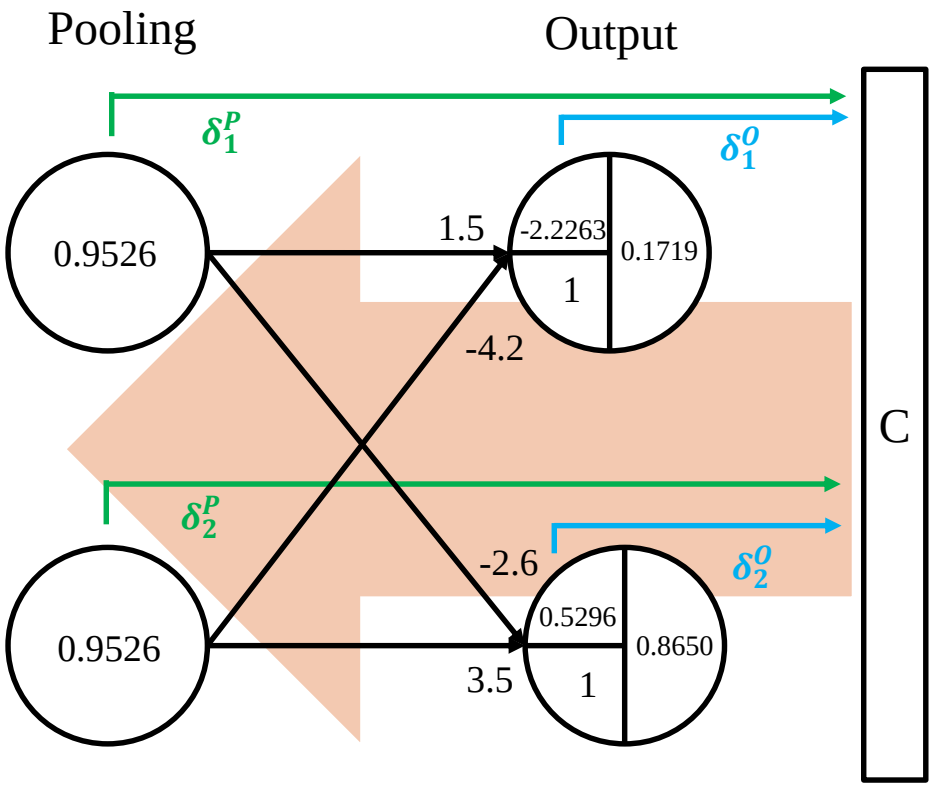


Pooling & Output Layer



出力値	正解値
0.1719	0
0.8650	1

Back Propagation （Output → Pooling）



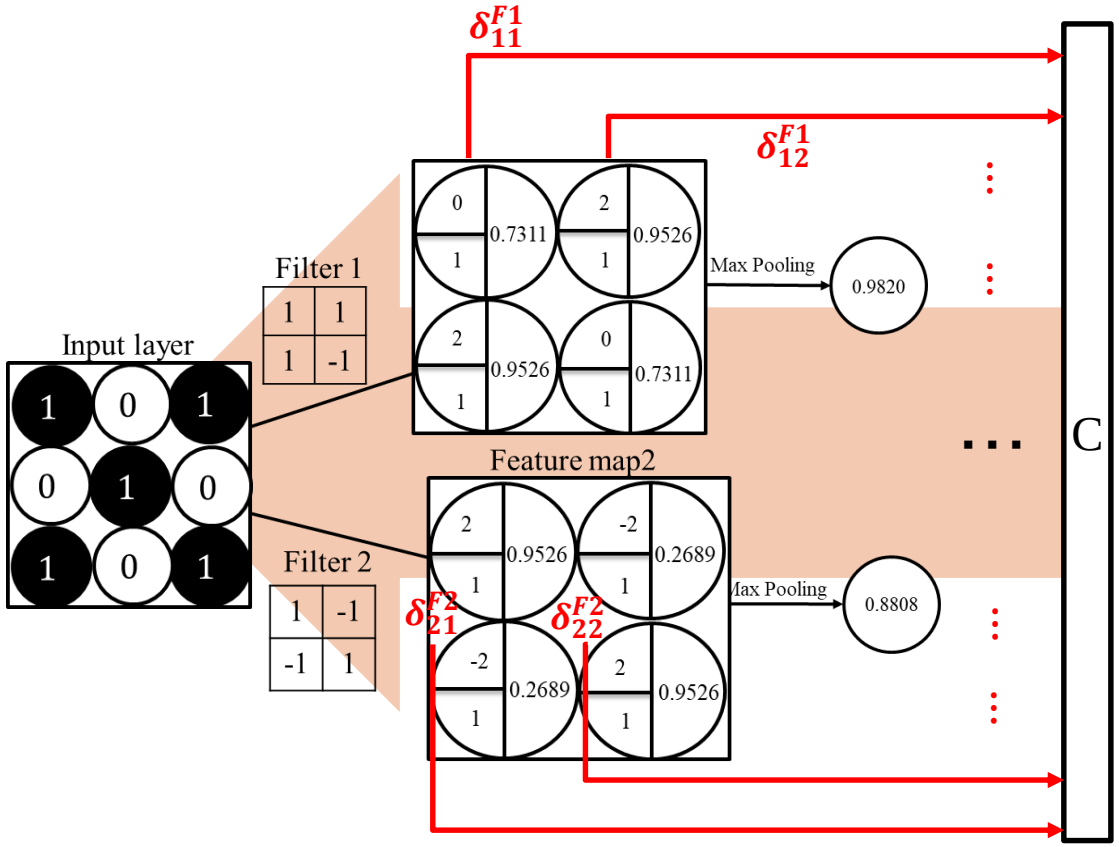
C_k = 0.0239

$$\delta_i^O = (a_i^3 - t_i)a'(Z_i^O) \quad \cdots (1)$$
$$a'(Z_i^O) = a(Z_i^O)(1 - a(Z_i^O))$$

$$\delta_i^P = (\delta_i^O w_{ji}^O + \delta_2^O w_{ji}^O)a'(Z_i^P) \quad \cdots (4)$$

$\delta_1^O = 0.0245$	$\delta_1^P = 0.0035$
$\delta_2^O = -0.0158$	$\delta_2^P = -0.0071$

Back Propagation （Pooling → Convolutional）



$\delta_1^O = -0.1356 \quad \delta_2^O = 0.1203$

$$\delta_{ij}^{Fk} = \left(\delta_1^O w_{k1}^O + \delta_2^O w_{k2}^O \right) \times \frac{\partial Z_i^P}{\partial a_{ij}^{Fk}} \times a'\left(\mathrm{Z_{ij}^{Fk}}\right) \cdots (8)$$

$\delta_{11}^{F1} = 0 \quad \delta_{12}^{F1} = (1.0e-0.3) \times 0.1568$

$\delta_{21}^{F1} = (1.0e-0.3) \times 0.1568 \quad \delta_{22}^{F1} = 0$

$\delta_{11}^{F2} = (1.0e-0.3) \times -0.3225 \quad \delta_{12}^{F2} = 0$

$\delta_{21}^{F2} = 0 \quad \delta_{22}^{F1} = (1.0e-0.3) \times -0.3225$

wとbの更新方法 (example 1/2)

ここまでで計算したパラメータを用いてwとbを更新する

～出力層における重みとバイアスの更新～

$$\Delta w_{1\,1}^{o} = -\eta \frac{\partial C}{\partial w_{1\,1}^{o}}$$

$$\frac{\partial C}{\partial w_{1\,1}^{o}} = \left(\frac{\partial C_1}{\partial w_{1\,1}^{o}} + \frac{\partial C_2}{\partial w_{1\,1}^{o}}\right)$$

$$= \left(\delta_1^o[1] a_1^2[1] + \delta_1^o[2] a_1^2[2]\right)$$

〇が教師のときの誤差　×が教師のときの誤差

…

$$\Delta b_1^o = -\eta \frac{\partial C}{\partial b_1^o}$$

$$\frac{\partial C}{\partial b_1^o} = \left(\frac{\partial C_1}{\partial b_1^o} + \frac{\partial C_2}{\partial b_1^o}\right)$$

$$= \left(\delta_1^o[1] + \delta_1^o[2]\right)$$

[1] → 教師が〇
[2] → 教師が×

wとbの更新方法 (example 2/2)

～畳み込み層における重みとバイアスの更新～

$$\Delta w_{1\,1}^{Fk} = -\eta \frac{\partial C}{\partial w_{1\,1}^{Fk}}$$

$$\frac{\partial C}{\partial w_{1\,1}^{Fk}} = \left(\frac{\partial C_1}{\partial w_{1\,1}^{Fk}} + \frac{\partial C_2}{\partial w_{1\,1}^{Fk}} \right)$$

$$= \left(\delta_1^O[1] a_1^2[1] + \delta_1^O[2] a_1^2[2] \right)$$

○が教師のときの誤差　×が教師のときの誤差

…

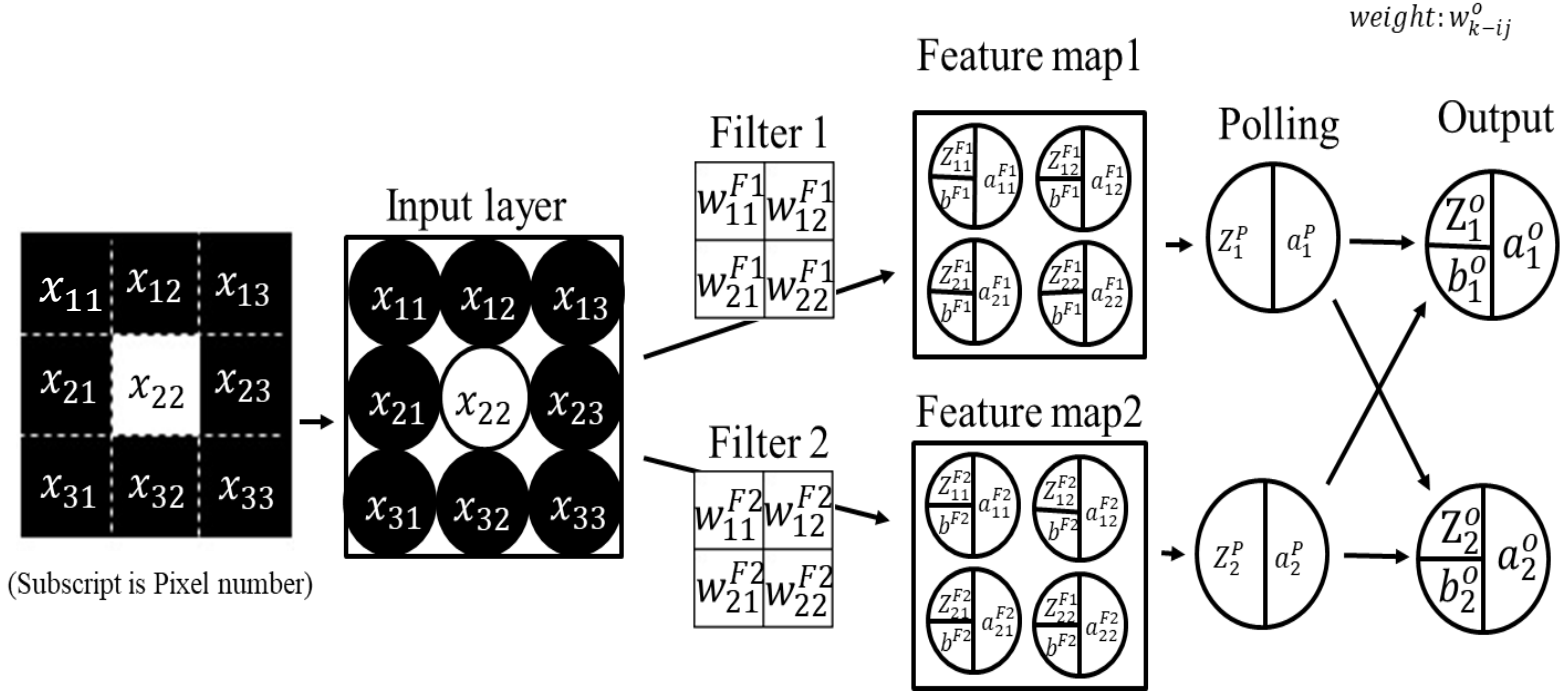
$$\Delta b^{Fk} = -\eta \frac{\partial C}{\partial b^{Fk}}$$

$$\frac{\partial C}{\partial b^{Fk}} = \left(\frac{\partial C_1}{\partial b^{Fk}} + \frac{\partial C_2}{\partial b^{Fk}} \right)$$

$$= \left(\delta_1^O[1] + \delta_1^O[2] \right)$$

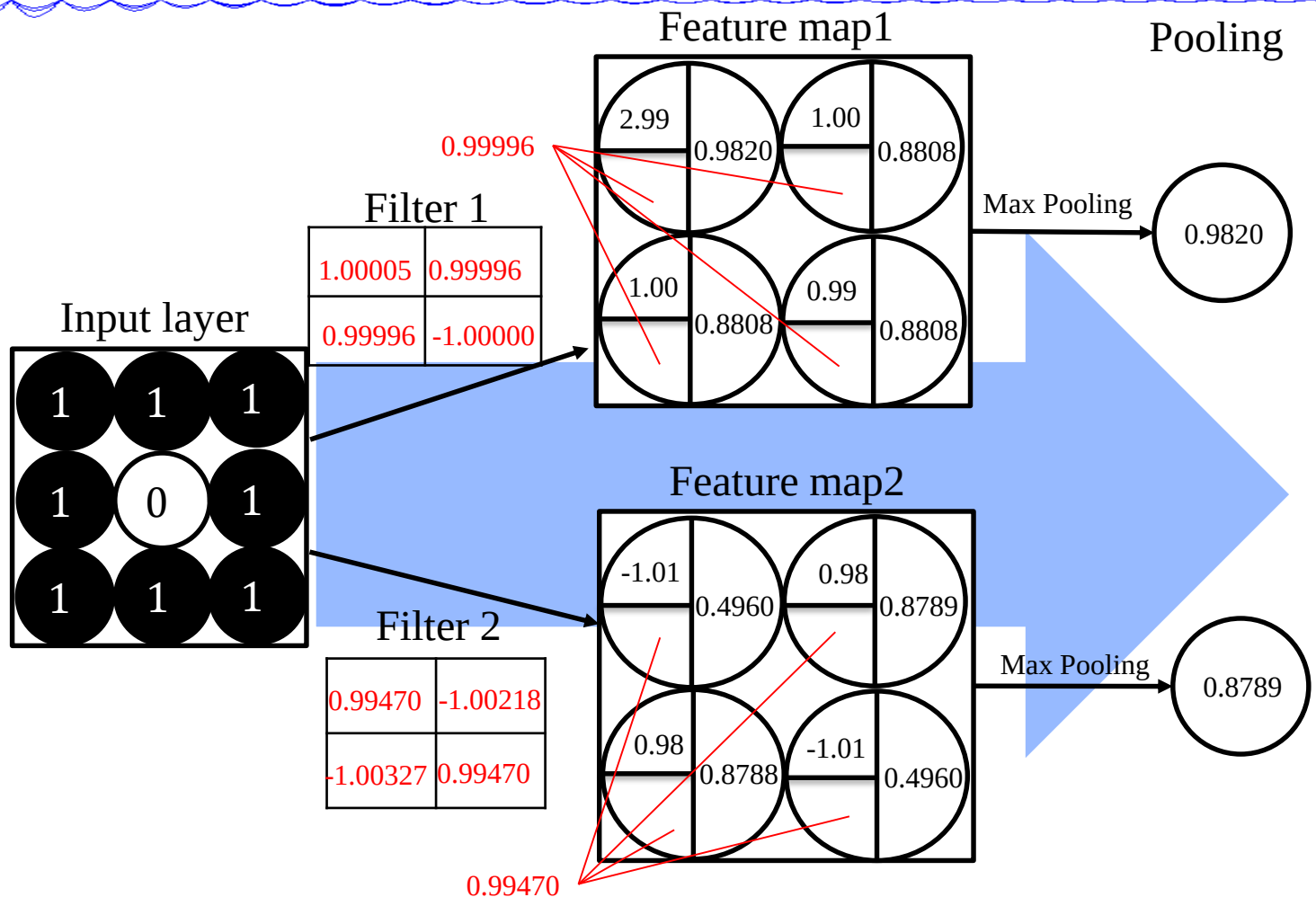
[1] → 教師が○
[2] → 教師が×

wとbの更新

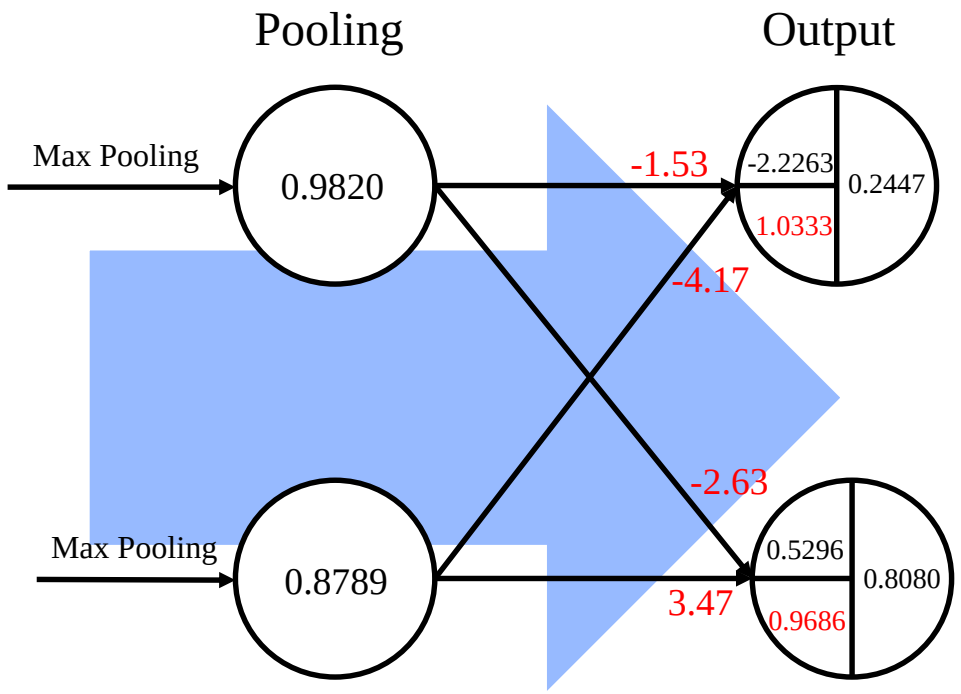


- $(new)w_{i\,j}^{Fk} = (old)w_{i\,j}^{Fk} + \Delta w_{i\,j}^{Fk}$
- $(new)w_{i\,j}^{2} = (old)w_{i\,j}^{O} + \Delta w_{i\,j}^{O}$
- $(new)b_i^{Fk} = (old)b_i^{Fk} + \Delta b_i^{Fk}$
- $(new)b_i^{O} = (old)b_i^{O} + \Delta b_i^{O}$

1回更新後(〇が教師)



1回更新後(〇が教師)

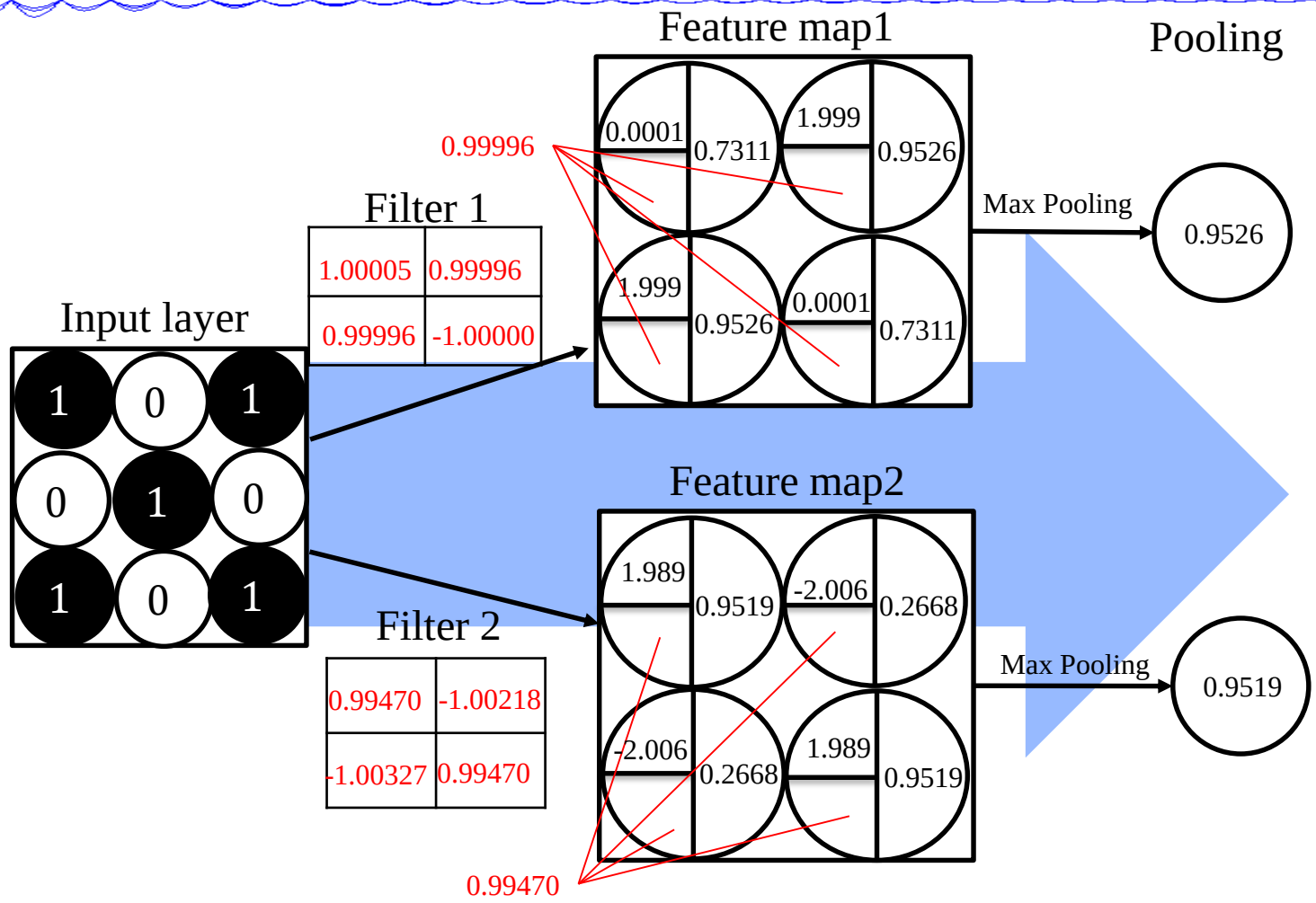


出力値 (更新前)	出力値 (更新後)	教師値
0.2268	0.2447	1
0.8219	0.8080	0

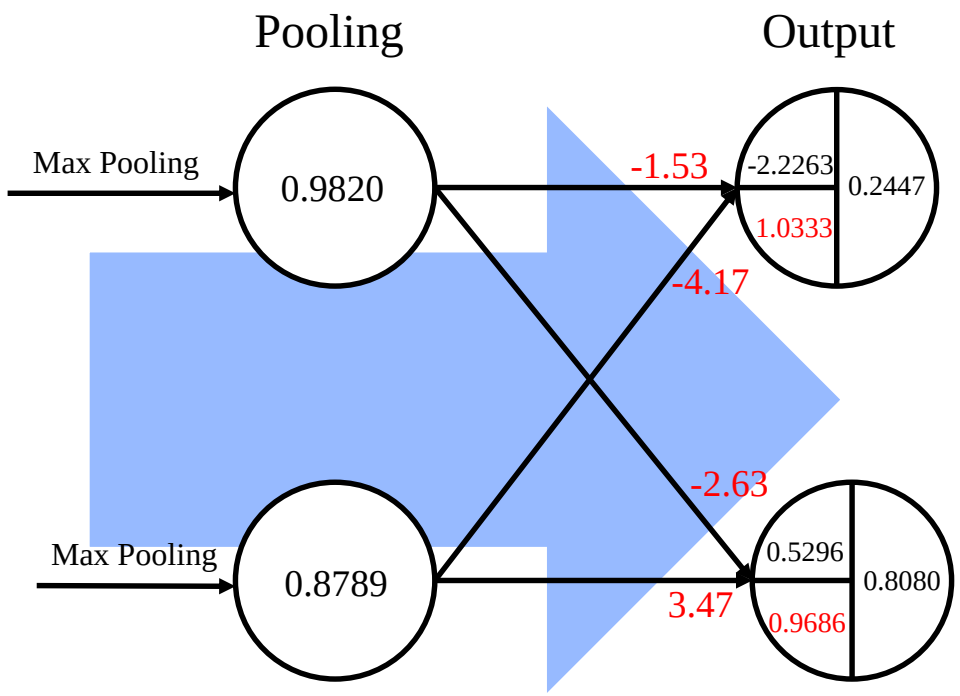
$C_k = 0.6267$

更新前と比べて,更新後の出力値の方が教師値に近いことが分かる.

1回更新後(×が教師)



1回更新後(×が教師)



出力値 (更新前)	出力値 (更新後)	教師値
0.1719	0.1859	0
0.8650	0.8542	1

$C_k = 0.0254$

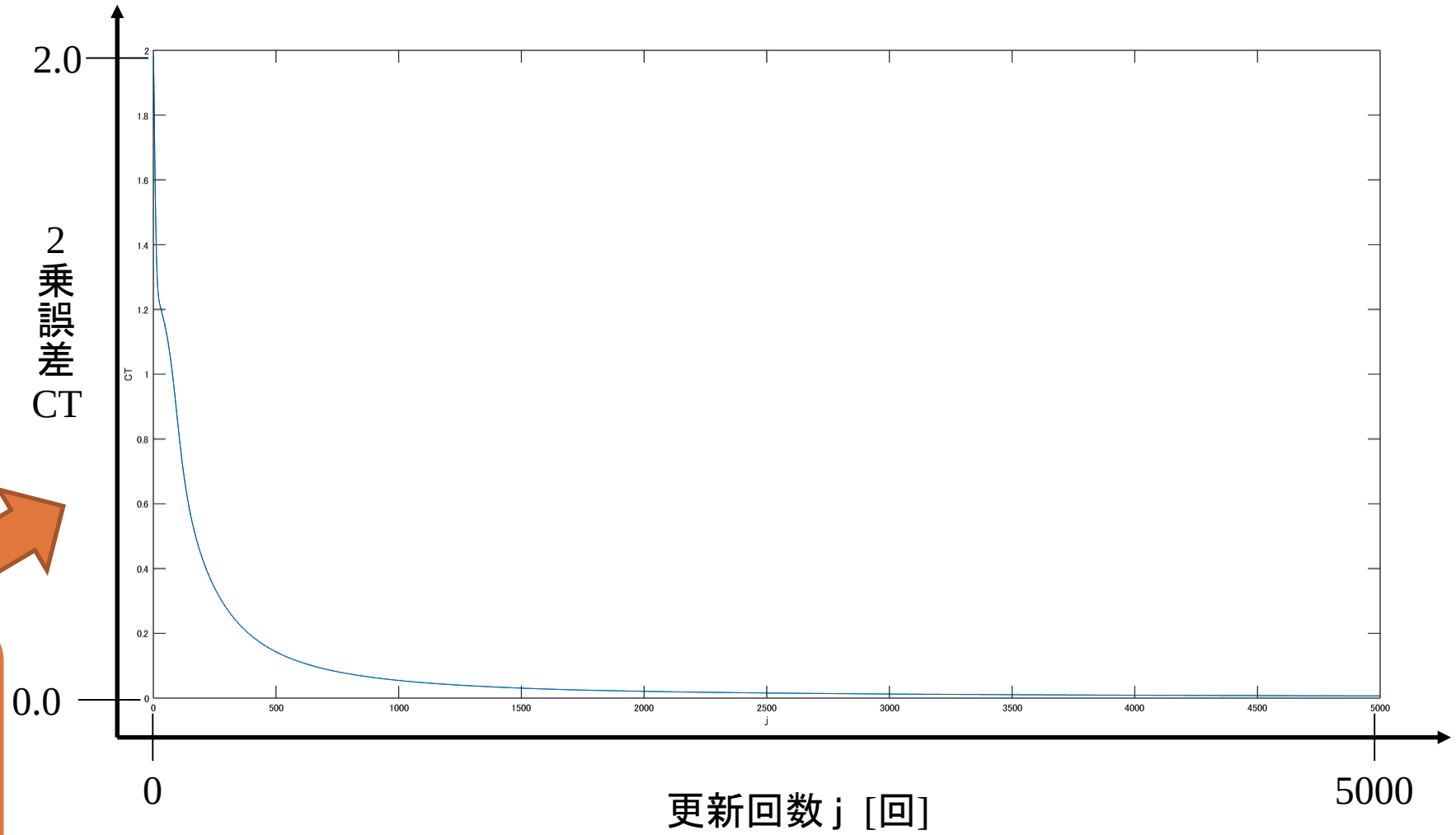
5000回更新後の出力

〇が教師	×が教師
0.9254	0.0692
0.0008	0.9895

In the case of 10 →〇
In the case of 01 →×
に近い出力結果となった.

2乗誤差の変化

j	0	1	5000
CT	1.9818	1.9560	0.0188



更新を繰り返すたびに
誤差の総和が減少
⇒CTが限りなく0に近い値に収束

結論

- NNやCNNにおける学習は,教師値と出力値の誤差が少なくなるように重みwとバイアスbの変化量を求め,十分な回数だけ更新を重ねる必要がある.

- ２乗誤差の和 CT のグラフより,学習を重ねるごとに出力値が正解に近づいていることを確認した.