5.5 活性化関数レイヤの実装

この節では，計算グラフの考え方をニューラルネットワークに適用する．ここでは，ニューラルネットワークを構成する「層（レイヤ)」をひとつのクラスとして実装する．まずは，活性化関数である ReLU と Sigmoid レイヤを実装する．

5.5.1 ReLU レイヤ

活性化関数として使われる ReLU(Rectified Linear Unit) は，次の式 (5.7) で表された．

$$y = \begin{cases} x & (x > 0) \\ 0 & (x \leqq 0) \end{cases} \tag{5.7}$$

式 (5.7) から，x に関する y の微分は式 (5.8) のように求められる．計算グラフで表すと，図 5-18 のように書くことができる．

$$\frac{\partial y}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \leqq 0) \end{cases} \tag{5.8}$$

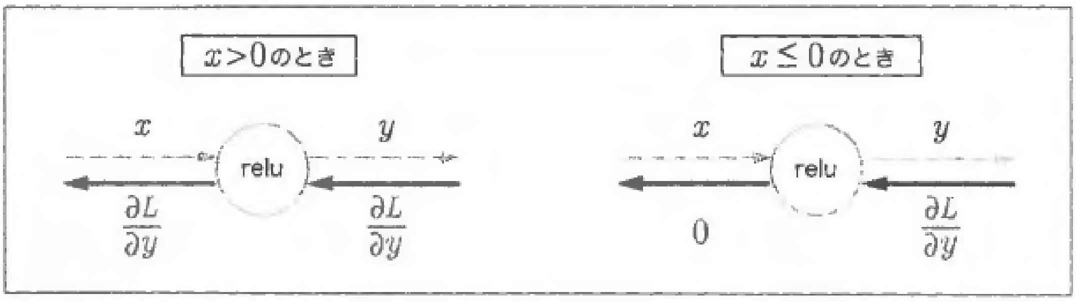

図 5-18　ReLU レイヤの計算グラフ

では，この ReLU レイヤの実装を行う．実装は，以下のソースコードのようになる．

ソースコード 1　ReLU

```
class Relu:
    def __init__(self):
        self.mask = None

    def forward(self, x):
        self.mask = (x <= 0)
        out = x.copy()
        out(self.mask) = 0

        return out
```

```
    def backward(self, dout):
        dout[self.mask] = 0
        dx = dout

        return dx
```

Relu クラスは，mask というインスタンス変数を持つ．mask は，True/False からなる Numpy 配列で，順伝播の入力である x の要素で 0 以下の場所を True，0 より大きい要素を False として保持する．

ソースコード 2　mask の例

```
>>> x = np.array([[1.0, -0.5], [-2.0, 3.0]])
>>> print(x)
[[ 1. -0.5]
 [-2. 3. ]]
>>> mask = (x <= 0)
>>> print(mask)
[[False True]
 [True False]]
```

図 5-18 に示すように，順伝播の入力が 0 以下ならば，逆伝播の値は 0 になるため，逆伝播では，順伝播時に保持した mask を用いて，上流から伝播された dout に対して，mask の要素が True の場所を 0 に設定する．

5.5.2 Sigmoid レイヤ

続いて，シグモイド関数を実装する．シグモイド関数を計算グラフで表すと，次の図 5-19 のようになる．

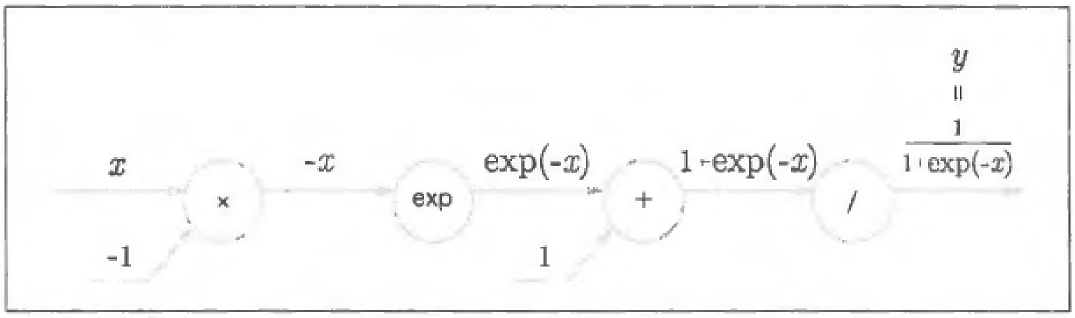

図 5-19　Sigmoid レイヤの計算グラフ（順伝播のみ）

図 5-19 では「×」と「+」ノードの他に，「exp」と「/」ノードが新しく登場している．「exp」ノードは $y = \exp(x)$ の計算を行い，「/」ノードは，$y = \frac{1}{x}$ の計算をする．Sigmoid レイヤの逆伝播の計算の流れをまとめると，図 5-20 のような計算グラフとなる．

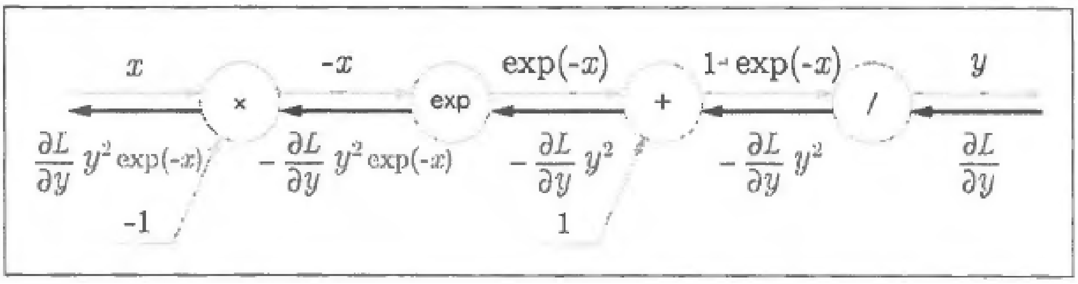

図 5-20　Sigmoid レイヤの計算グラフ

図 5-20 の結果から逆伝播の出力は，$\frac{\partial L}{\partial y} y^2 \exp(-x)$ となり，この値が下流にあるノードに伝播していく．ここで $\frac{\partial L}{\partial y} y^2 \exp(-x)$ という値が順伝播の入力 x と出力 y だけから計算できる点に注目すると，次の図 5-21 のようなグループ化した「sigmoid」ノードとして書くことができる．

そして，$\frac{\partial L}{\partial y} y^2 \exp(-x)$ は，さらに次のように整理して書くことができる．

$$
\begin{aligned}
\frac{\partial L}{\partial y} y^2 \exp(-x) &= \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x) \\
&= \frac{\partial L}{\partial y} \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)} \\
&= \frac{\partial L}{\partial y} y(1-y)
\end{aligned}
\tag{5.12}
$$

そのため，図 5-21 で表される Sigmoid レイヤの逆伝播は，順伝播の出力だけから計算することが出来る．

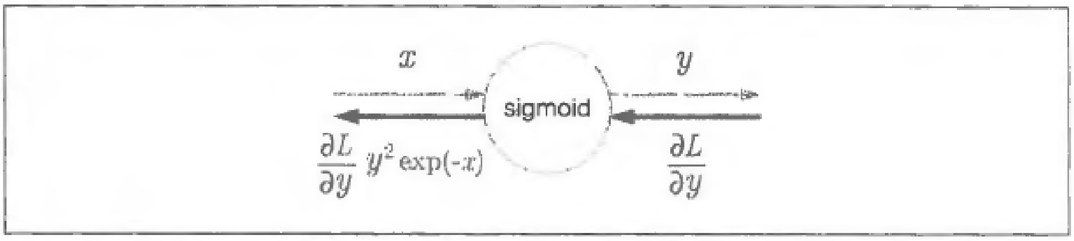

図 5-21　Sigmoid レイヤの計算グラフ：順伝播の出力 y によって，逆伝播の計算を行うことができる

ソースコード 3　Sigmoid

```
class Sigmoid:
    def __init__(self):
        self.out = None
    def forward(self, x):
        out = 1 / (1 + np.exp(-x))
        self.out = out

        return out

    def backward(self, dout):
```

```
        dx = dout * (1.0 - self.out) * self.out

        return dx
```

この実装では，順伝播時に出力をインスタンス変数の out に保持している．そして逆伝播時に，その out 変数を使って計算を行う．