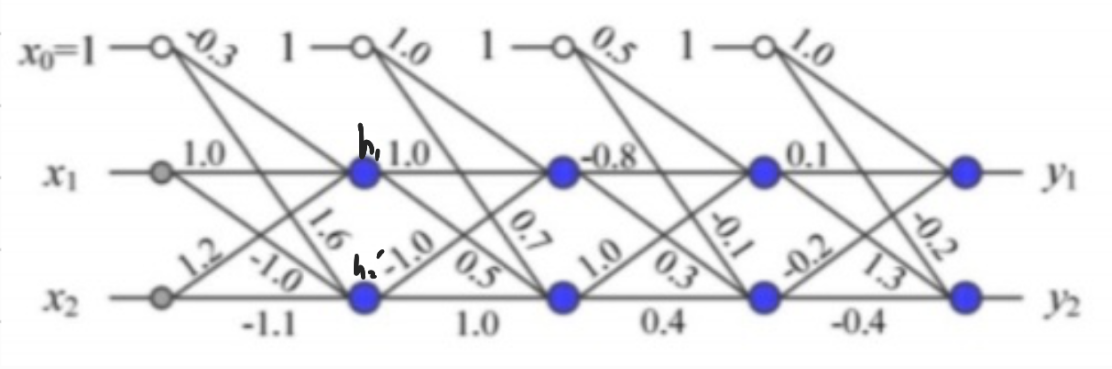
5.



1) $U^1 = \begin{bmatrix} u_{10} & u_{20} \\ u_{11} & u_{21} \\ u_{12} & u_{22} \end{bmatrix} = \begin{bmatrix} -0.3 & 1.6 \\ 1.0 & -1.0 \\ 1.2 & -1.1 \end{bmatrix}$

$U^2 = \begin{bmatrix} 1.0 & 0.7 \\ 1.0 & 0.5 \\ -1.0 & 1.0 \end{bmatrix}$ $U^3 = \begin{bmatrix} 0.5 & -0.1 \\ -0.8 & 0.3 \\ 1.0 & 0.4 \end{bmatrix}$ $U^4 = \begin{bmatrix} 1.0 & -0.2 \\ 0.1 & 1.3 \\ -0.2 & -0.4 \end{bmatrix}$

2)

```
# i->h1
input_x = np.array([1,0])
input_w=np.array([[1.0,-1.0],[1.2,-1.1]])
bias = np.array([1])
bias_weights = np.array([-0.3,1.6])
hidden_w = np.dot(input_x,input_w)
hidden_o = sigmoid(hidden_w+bias*bias_weights)
# h1->h2
hidden_w2 = np.array([[1,0.5],[-1,1]])
bias2 = np.array([1])
bias_weights2 = np.array([1,-0.7])
hidden_o2 = np.dot(hidden_o,hidden_w2)
hidden_o2 = sigmoid(hidden_o2+bias2*bias_weights2)
# h2->h3
hidden_w3 = np.array([[-0.8,0.3],[1.0,0.4]])
bias3 = np.array([1])
bias_weights3 = np.array([0.5,-0.1])
hidden_o3 = np.dot(hidden_o2,hidden_w3)
hidden_o3 = sigmoid(hidden_o3+bias3*bias_weights3)
# h3->h4
hidden_w4 = np.array([[0.1,1.3],[-0.2,-0.4]])
bias4 = np.array([1])
bias_weights4 = np.array([1.0,-0.2])
hidden_o4 = np.dot(hidden_o3,hidden_w4)
hidden_o4 = sigmoid(hidden_o4+bias4*bias_weights4)
print(hidden_o4)
```

```
[0.72001853 0.59119767]
```

$h_{11} = 1\times1 + 1.2\times0 + (-0.3) = 0.7 \quad \sigma(h_{11}) = 0.668$

$h_{12} = 1\times(-1) + (-1.1)\times0 + 1.6 = 0.6 \quad \sigma(h_{12}) = 0.645$

$\Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & -1.0 \\ 1.2 & -1.1 \end{bmatrix} + \begin{bmatrix} -0.3 \\ 1.6 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.6 \end{bmatrix}$

$\text{sigmoid}\left(\begin{bmatrix} 0.7 \\ 0.6 \end{bmatrix}\right) = \begin{bmatrix} 0.668 \\ 0.645 \end{bmatrix}$

이와 같은 과정 반복

$\therefore O_1 = 0.72 \quad O_2 = 0.591$

3)

```
# i->h1
input_x = np.array([1,0])
input_w=np.array([[1.0,-1.0],[1.2,-1.1]])
bias = np.array([1])
bias_weights = np.array([-0.3,1.6])
hidden_w = np.dot(input_x,input_w)
hidden_o = relu(hidden_w+bias*bias_weights)
# h1->h2
hidden_w2 = np.array([[1,0.5],[-1,1]])
bias2 = np.array([1])
bias_weights2 = np.array([1,-0.7])
hidden_o2 = np.dot(hidden_o,hidden_w2)
hidden_o2 = relu(hidden_o2+bias2*bias_weights2)
# h2->h3
hidden_w3 = np.array([[-0.8,0.3],[1.0,0.4]])
bias3 = np.array([1])
bias_weights3 = np.array([0.5,-0.1])
hidden_o3 = np.dot(hidden_o2,hidden_w3)
hidden_o3 = relu(hidden_o3+bias3*bias_weights3)
# h3->h4
hidden_w4 = np.array([[0.1,1.3],[-0.2,-0.4]])
bias4 = np.array([1])
bias_weights4 = np.array([1.0,-0.2])
hidden_o4 = np.dot(hidden_o3,hidden_w4)
hidden_o4 = relu(hidden_o4+bias4*bias_weights4)
print(hidden_o4)
```

```
[0.934 0.   ]
```

$\therefore O_1 = 0.934, \quad O_2 = 0$

4) $u_{i,9}^3$ 이 1일 때의 오류값은 [0.72 , -0.408] 이다.

$u_{i,9}^3$ 이 0.9일 땐 오류값은 [0.719 , -0.413] 이다.

따라서 weight 값이 줄어들면서 오류도 줄어들었다.

7.

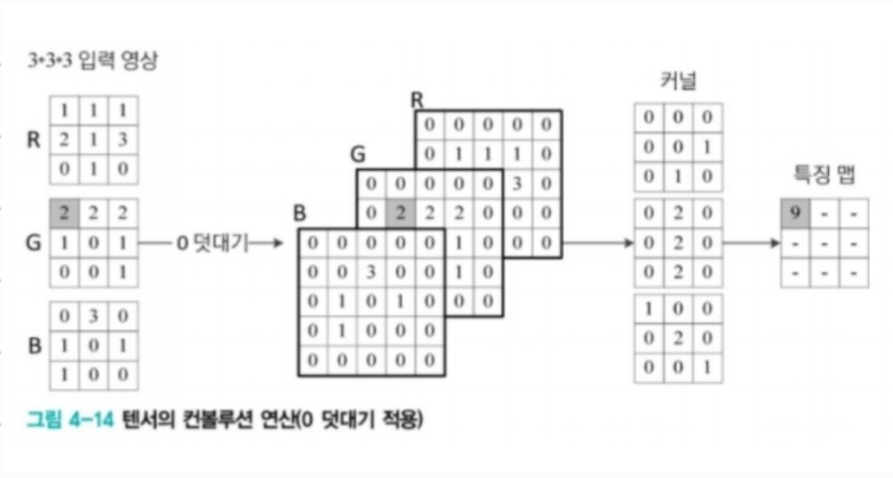
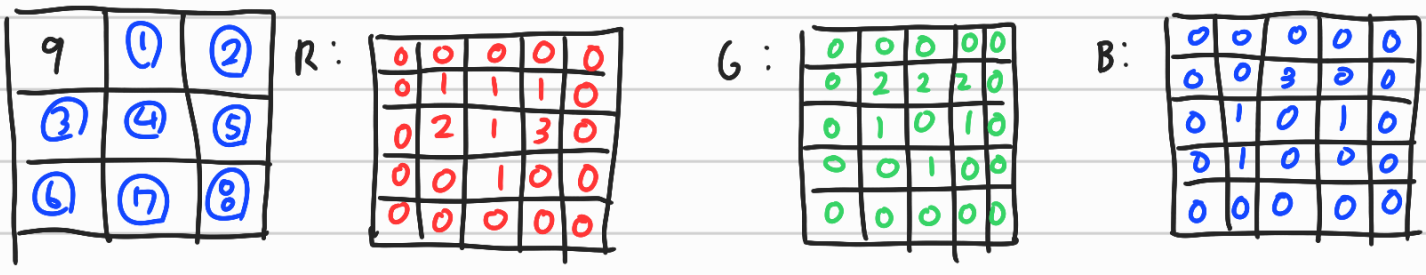


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)



① : R: 1×1 + 1×1 = 2 , G: 2×2 = 4 B: 3×2 + 1×1 = 7 ∴ 2+4+7 = 13

② : R: 1×3 = 3 , G: 2×2 + 1×2 = 6 B: 0 ∴ 3+6+0 = 9

③ : R: 1×1 = 1 , G: 1×2 = 2 B: 1×2 = 2 ∴ 1+2+2 = 5

④ : R: 3×1 + 1×1 = 4 G: 2×2 + 1×2 = 6 B: 0 ∴ 4+6+0 = 10

⑤ : R: 0 G: 2×2 + 2×1 = 6 B: 3×1 + 2×1 = 5 ∴ 6+5 = 11

⑥ : R: 1×1 = 1 G: 1×2 = 2 B: 1×2 = 2 ∴ 1+2+2 = 5

⑦ R: 0 G: 1×2 = 2 B: 1×1 = 1 ∴ 2+1 = 3

⑧ R: 0 G: 2×1 = 2 B: 0 ∴ 2 = 2

9	13	9
5	10	11
5	3	2

8.

0	0	0	0	0	0	0	0	0	0
0	2	2	2	2	2	1	1	1	0
0	2	2	2	2	2	1	1	1	0
0	2	2	2	2	2	1	1	1	0
0	2	2	2	2	2	1	1	1	0
0	2	2	2	9	9	9	9	9	0
0	2	2	2	9	9	9	9	9	0
0	2	2	2	9	9	9	9	9	0
0	2	2	2	9	9	9	9	9	0
0	0	0	0	0	0	0	0	0	0

⊛

-1	-1	-1
0	0	0
1	1	1

output shape

$8-3+2+1=8$

8×8

4.5	6.5	6.5	6.5	5.5	4.5	3.5	2.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	7.5	14.5	22.5	23.5	24.5	16.5
0.5	0.5	7.5	14.5	22.5	23.5	24.5	16.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
-3.5	-5.5	-12.5	-19.5	-26.5	-26.5	-26.5	-17.5

9. $\text{floor}\left(\frac{8-3+2}{2}+1\right)-4$

$\therefore$ 4 x 4 shape

4.5	6.5	5.5	3.5
0.5	0.5	0.5	0.5
0.5	7.5	22.5	24.5
0.5	0.5	0.5	0.5

원래문제 8번의 결과로 풀림

10. 문제: 8번의 zero padding 된 것에 max pooling, average pooling 적용

max pooling

2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	1	1
2	2	2	2	2	2	2	1	1
2	2	9	9	9	9	9	9	9
2	2	9	9	9	9	9	9	9
2	2	9	9	9	9	9	9	9
2	2	9	9	9	9	9	9	9
2	2	9	9	9	9	9	9	9

avg Pooling

$\frac{8}{9}$	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{10}{9}$	$\frac{8}{9}$	$\frac{2}{3}$	$\frac{4}{9}$
$\frac{4}{3}$	2	2	2	$\frac{5}{3}$	$\frac{4}{3}$	1	$\frac{2}{3}$
$\frac{4}{3}$	2	2	2	$\frac{5}{3}$	$\frac{4}{3}$	1	$\frac{2}{3}$
$\frac{4}{3}$	2	$\frac{25}{9}$	$\frac{32}{9}$	$\frac{37}{9}$	$\frac{35}{9}$	$\frac{33}{9}$	$\frac{22}{9}$
$\frac{4}{3}$	2	$\frac{32}{9}$	$\frac{46}{9}$	$\frac{59}{9}$	$\frac{58}{9}$	$\frac{57}{9}$	$\frac{38}{9}$
$\frac{4}{3}$	2	$\frac{13}{3}$	$\frac{20}{3}$	9	9	9	6
$\frac{4}{3}$	2	$\frac{39}{9}$	$\frac{20}{3}$	9	9	9	6
$\frac{8}{9}$	$\frac{4}{3}$	$\frac{26}{9}$	$\frac{40}{9}$	6	6	6	4