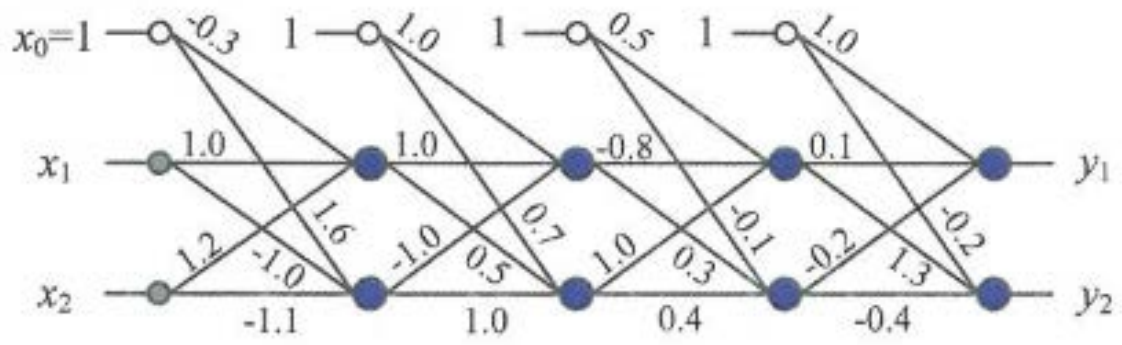
머신러닝 과제04

제출: 각 문제에 대한 source code 및 report (word format)을 zip으로 압축, KLAS에 제출
제출시 파일이름: 학번_이름.zip (예: 2014200154_홍길동.zip)

(이론)_____

1 다음은 은닉층이 3개인 DMLP이다.

Hint 계산은 Matlab 또는 Python을 사용하시오.



(1) 가중치 행렬 $\mathbf{U}^1, \mathbf{U}^2, \mathbf{U}^3, \mathbf{U}^4$를 식 (4.1)처럼 쓰시오.

$$\text{가중치 행렬: } \mathbf{U}^l = \begin{pmatrix} u_{10}^l & u_{11}^l & \cdots & u_{1n_{l-1}}^l \\ u_{20}^l & u_{21}^l & & u_{2n_{l-1}}^l \\ \vdots & & \ddots & \vdots \\ u_{n_l 0}^l & u_{n_l 1}^l & \cdots & u_{n_l n_{l-1}}^l \end{pmatrix}, l = 1,2,\cdots,L \quad (4.1)$$

(2) $\mathbf{x}=(\mathbf{1},\mathbf{0})^{\mathbf{T}}$가 입력되었을 때 출력 $\mathbf{y}$를 구하시오. 활성함수로 로지스틱 시그모이드를 사용하시오.

(3) $\mathbf{x}=(\mathbf{1},\mathbf{0})^{\mathbf{T}}$가 입력되었을 때 출력 $\mathbf{y}$를 구하시오. 활성함수로 ReLU를 사용하시오.

(4) $\mathbf{x}=(\mathbf{1},\mathbf{0})^{\mathbf{T}}$의 기대 출력이 $\mathbf{y}=(\mathbf{0},\mathbf{1})^{\mathbf{T}}$일 때, 현재 1.0인 u_{12}^3 가중치를 0.9로 줄이면 오류에 어떤 영향을 미치는지 설명하시오.

2 [그림 4-14]에서 나머지 8개 화소의 값을 계산하시오.

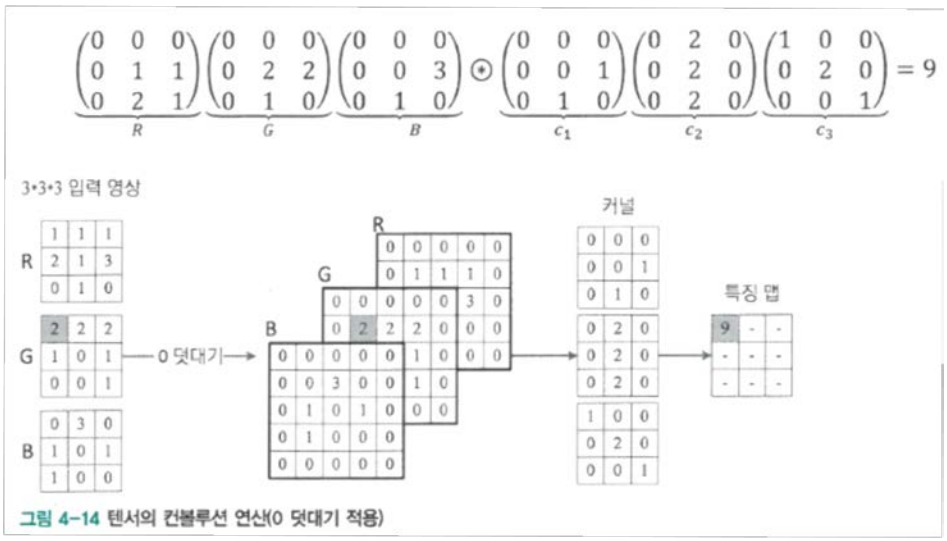


그림 4-14 텐서의 컨볼루션 연산(0 덧대기 적용)

3 [그림 4-8(b)]에서 커널 $\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$을 적용한 결과를 쓰시오. 이때 0 덧대기를 하고 바이어스로 0.5를 사용하시오.

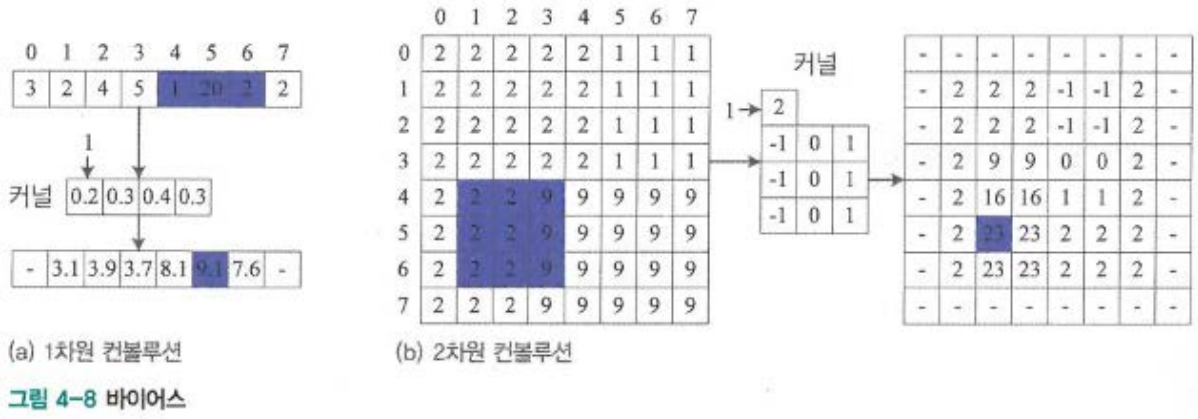


그림 4-8 바이어스

4 문제 3의 결과에 최대 풀링과 평균 풀링을 적용한 결과를 각각 쓰시오. 보폭으로 1을 사용하시오.

5 프로젝트를 어떻게 진행할 것인지에 대한 간단한 아이디어를 모델 diagram(입력/출력, 모델 기능 포함)과 함께 제시하시오. (반 page 내로 간략히 서술)

(실습)_____

1. 아래의 참고비디오를 시청 후, 아래의 소스코드를 실행하시오. 각 line 별로 주석을 작성하시오.

참고 비디오 (Lecture 05):
https://www.youtube.com/watch?v=Mf8jna42p2M&list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o_2m&index=6

1) 그래프(x 축: epoch, y 축: loss)를 도시하고, 해석하시오.

2) torch.nn 과 torch.optim 모듈은 어떤 기능을 제공하는지 조사하시오.

```
import torch
from torch.autograd import Variable

x_data = Variable(torch.Tensor([[1.0], [2.0], [3.0]]))
y_data = Variable(torch.Tensor([[2.0], [4.0], [6.0]]))


class Model(torch.nn.Module):

    def __init__(self):
        """
        In the constructor we instantiate two nn.Linear module
        """
        super(Model, self).__init__()
        self.linear = torch.nn.Linear(1, 1)  # One in and one out

    def forward(self, x):
        """
        In the forward function we accept a Variable of input data and we must return
        a Variable of output data. We can use Modules defined in the constructor as
        well as arbitrary operators on Variables.
        """
        y_pred = self.linear(x)
        return y_pred

# our model
model = Model()


# Construct our loss function and an Optimizer. The call to model.parameters()
# in the SGD constructor will contain the learnable parameters of the two
# nn.Linear modules which are members of the model.
criterion = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# Training loop
for epoch in range(500):
        # Forward pass: Compute predicted y by passing x to the model
    y_pred = model(x_data)

    # Compute and print loss
    loss = criterion(y_pred, y_data)
```

```
    print(epoch, loss.data[0])

    # Zero gradients, perform a backward pass, and update the weights.
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()


# After training
hour_var = Variable(torch.Tensor([[4.0]]))
y_pred = model(hour_var)
print("predict (after training)",  4, model(hour_var).data[0][0])
```

2. 아래의 참고비디오를 시청 후, 아래의 소스코드를 실행하시오. 각 line 별로 주석을 작성하시오.

참고 비디오 (Lecture 06):
https://www.youtube.com/watch?v=113b7O3mabY&list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o_2m&index=7

1) Regression과 Classification의 차이는 무엇인가(loss함수의 차이도 적을 것)

2) Classification에서 logit의 의미는 무엇인가.

3) Softmax layer는 어떤 역할을 수행하는가.

4) BCELoss란 무엇인가? 정보이론/확률이론 으로부터 수학적으로 유도하시오.

```
import torch
from torch.autograd import Variable
import torch.nn.functional as F

x_data = Variable(torch.Tensor([[1.0], [2.0], [3.0], [4.0]]))
y_data = Variable(torch.Tensor([[0.], [0.], [1.], [1.]]))


class Model(torch.nn.Module):

    def __init__(self):
        """
        In the constructor we instantiate nn.Linear module
        """
        super(Model, self).__init__()
        self.linear = torch.nn.Linear(1, 1)  # One in and one out

    def forward(self, x):
        """
        In the forward function we accept a Variable of input data and we must return
        a Variable of output data.
        """
```

```
        y_pred = F.sigmoid(self.linear(x))
        return y_pred

# our model
model = Model()


# Construct our loss function and an Optimizer. The call to model.parameters()
# in the SGD constructor will contain the learnable parameters of the two
# nn.Linear modules which are members of the model.
criterion = torch.nn.BCELoss(size_average=True)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# Training loop
for epoch in range(1000):
        # Forward pass: Compute predicted y by passing x to the model
    y_pred = model(x_data)

    # Compute and print loss
    loss = criterion(y_pred, y_data)
    print(epoch, loss.data[0])

    # Zero gradients, perform a backward pass, and update the weights.
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

# After training
hour_var = Variable(torch.Tensor([[1.0]]))
print("predict 1 hour ", 1.0, model(hour_var).data[0][0] > 0.5)
hour_var = Variable(torch.Tensor([[7.0]]))
print("predict 7 hours", 7.0, model(hour_var).data[0][0] > 0.5)
```