

1. 삼변수 함수

$$f(x, y, z) = xyz$$

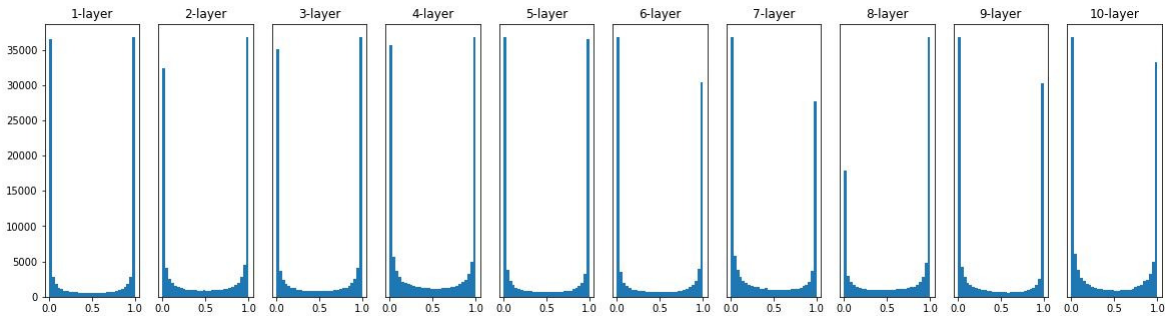
에 대하여 learning rate $\eta = 1$ 과 관성 계수 $\alpha = 1$ 로 Momentum을 적용하려 한다. 초기 위치 $\mathbf{x}_0 = (1, 2, 1)$ 에서 출발하여 세 발자국 걸어갈 때, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ 를 구하시오.

2. 이변수 함수

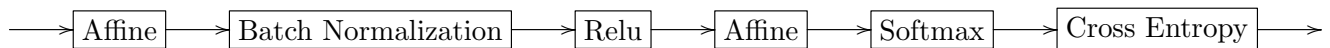
$$f(x, y) = xy$$

에 대하여 learning rate $\eta = 1$ 과 forgetting factor $\gamma = 3/4$ 으로 RMSProp을 적용하려 한다. 초기 위치 $\mathbf{x}_0 = (1, 2)$ 에서 출발하여 두 발자국 걸어갈때, $\mathbf{x}_1, \mathbf{x}_2$ 를 구하시오.

3. 활성화 함수를 시그모이드 함수로 사용하는 10층 신경망에서 가중치를 표준정규분포를 따라 초기화하였다. 다음 그림은 신경망에 데이터를 입력했을때 활성화층 출력값들의 히스토그램이다. 이후 다량의 데이터를 입력하여 여러번 학습을 시도하여도 낮은 층으로 내려갈수로 가중치와 편향이 제대로 학습이 되지 않았다. 그 이유를 수학적으로 설명하시오.



4. 신경망이



와 같이 주어져 있다. 첫번째 Affine층의 가중치 행렬 W_1 , 두번째 Affine층의 가중치 행렬 W_2 가 각각

$$W_1 = \begin{pmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad W_2 = \begin{pmatrix} \log 2 & \log 2 & 0 \\ -\log 2 & 0 & \log 2 \\ 0 & 0 & \log 2 \end{pmatrix}$$

이고 편향은 없다. 배치정규화층(batch normalization)의 확대(scale)는 $\gamma = [3, 2, 1]$, 이동(shift)은 $\beta = [-1, -1, 0]$ 이다. 입력 데이터 $[1, 2, 3], [3, 0, 1]$ 이 배치 묶음으로 입력되고 라벨이 각각 $[0, 1, 0], [0, 0, 1]$ 일 때 손실함수값을 구하시오.

5. 활성화 함수가 Relu인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}, b_1 : [0, 0, 0, 0, 0], W_2 : \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, b_2 : [0, 0, 0, 0, 0]\}$$

로 주어져 있다. 데이터 $\mathbf{x} = [1, 2, 1, 2, 1]$ 의 라벨이 $[1, 0, 0, 0, 0]$ 이라 하자. 손실함수는 교차 엔트로피(cross entropy)에 L^2 -규제 (L^2 -regularization)가 가해져 있다. 가중치 감소의 세기가 $\lambda = \frac{1}{17}$ 일 때, 데이터 \mathbf{x} 의 손실함수 값을 구하시오.

6. 밑줄 친 부분을 수식으로 바꾸고 이로부터 Adam의 점화식을 이끌어 내시오.

```
class Adam:
    def __init__(self, lr=0.001, beta1=0.9, beta2=0.999):
        self.lr = lr
        self.beta1 = beta1
        self.beta2 = beta2
        self.iter = 0
        self.m = None
        self.v = None
    def update(self, params, grads):
        if self.m is None:
            self.m, self.v = {}, {}
            for key, val in params.items():
                self.m[key] = np.zeros_like(val)
                self.v[key] = np.zeros_like(val)
            self.iter += 1
            lr_t = self.lr * np.sqrt(1.0 - self.beta2**self.iter)
                    / (1.0 - self.beta1**self.iter)
            for key in params.keys():
                self.m[key] += (1 - self.beta1) * (grads[key] - self.m[key])
                self.v[key] += (1 - self.beta2) * (grads[key]**2 - self.v[key])
                params[key] -= lr_t * self.m[key] / (np.sqrt(self.v[key]) + 1e-7)
```

7. 다음은 배치 정규화(batch normalization)의 순전파와 역전파 코드이다. 빈 칸을 채우시오.

```
def forward(self, x, train_flg):
    if self.running_mean is None:
        N, D = x.shape
        self.running_mean = np.zeros(D)
        self.running_var = np.zeros(D)
    if train_flg:
        mu = x.mean(axis=0)
        xc = x - mu
        var = 
        std = np.sqrt(var + 10e-7)
        xn = 
```

```

        self.batch_size = x.shape[0]
        self.xc = xc
        self.xn = xn
        self.std = std
        self.running_mean = self.momentum * self.running_mean + (1-self.momentum) * mu
        self.running_var = self.momentum * self.running_var + (1-self.momentum) * var
    else:
        xc = self.running_mean
        xn = xc / ((self.running_var + 10e-7))
    out = self.gamma * xn + self.beta
    return out

```

```

def backward(self, dout):
    dbeta = 
    dgamma = np.sum(self.xn * dout, axis=0)
    dxn =  * dout
    dxc = dxn / self.std
    dstd = -np.sum((dxn * self.xc) / (self.std * self.std), axis=0)
    dvar = 0.5 * dstd / 
    dxc += (2.0 / self.batch_size) * self.xc * dvar
    dmua = np.sum(dxc, axis=0)
    dx = dxc - dmua / self.batch_size
    self.dgamma = dgamma
    self.dbeta = dbeta
    return dx

```

8. 다음 코드를 실행했을 때 출력될 값들을 대략적으로 쓰시오.

```

dropout = Dropout(0.3)
x = np.random.rand(100,100)
y = dropout.forward(x, train_flg=True)
print(np.sum(np.abs(y)<1e-7) / np.size(y))
print(np.mean(y))
z = dropout.forward(x, train_flg=False)
print(np.sum(np.abs(z)<1e-7) / np.size(z))
print(np.mean(z))

```

9. 다음은 확장된 심층 신경망 코드의 앞부분이다. 인스턴스

```
Net = MultiLayerNetExtend(100,[100,50,25],10,use_dropout=True,use_batchnorm=True)
```

를 통해 만들어진 심층 신경망의 각층을 순서대로 쓰고 화살표로 연결하십시오. (4번 문제처럼)

```

class MultiLayerNetExtend:
    def __init__(self, input_size, hidden_size_list, output_size,
                  activation='relu', weight_init_std='relu', weight_decay_lambda=0,
                  use_dropout = False, dropout_ratio = 0.5, use_batchnorm=False):
        self.input_size = input_size
        self.output_size = output_size

```

```

self.hidden_size_list = hidden_size_list
self.hidden_layer_num = len(hidden_size_list)
self.use_dropout = use_dropout
self.weight_decay_lambda = weight_decay_lambda
self.use_batchnorm = use_batchnorm
self.params = {}
self.__init_weight(weight_init_std)
activation_layer = {'sigmoid': Sigmoid, 'relu': Relu}
self.layers = OrderedDict()
for idx in range(1, self.hidden_layer_num+1):
    self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)],
                                              self.params['b' + str(idx)])

    if self.use_batchnorm:
        self.params['gamma' + str(idx)] = np.ones(hidden_size_list[idx-1])
        self.params['beta' + str(idx)] = np.zeros(hidden_size_list[idx-1])
        self.layers['BatchNorm' + str(idx)] = BatchNormalization(self.params['gamma'
                                                                    + str(idx)], self.params['beta' + str(idx)])

    self.layers['Activation_function' + str(idx)] = activation_layer[activation]()
    if self.use_dropout:
        self.layers['Dropout' + str(idx)] = Dropout(dropout_ratio)
idx = self.hidden_layer_num + 1
self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)],
                                          self.params['b' + str(idx)])

self.last_layer = SoftmaxWithLoss()
def __init_weight(self, weight_init_std):
    all_size_list = [self.input_size] + self.hidden_size_list + [self.output_size]
    for idx in range(1, len(all_size_list)):
        scale = weight_init_std
        if str(weight_init_std).lower() in ('relu', 'he'):
            scale = np.sqrt(2.0 / all_size_list[idx - 1])
        elif str(weight_init_std).lower() in ('sigmoid', 'xavier'):
            scale = np.sqrt(1.0 / all_size_list[idx - 1])
        self.params['W' + str(idx)] = scale * np.random.randn(all_size_list[idx-1],
                                                                all_size_list[idx])
        self.params['b' + str(idx)] = np.zeros(all_size_list[idx])

```

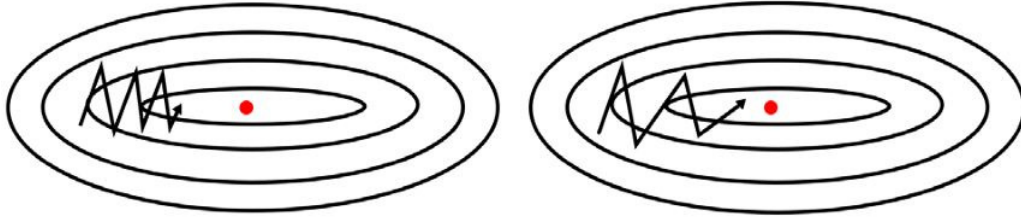
10. (9번 문제 계속) 다음 코드를 실행했을 때 출력될 값은 대략적으로 얼마가 되겠는가?

```

x=0
for i in range(1,5):
    x+=np.mean(Net.params['W'+str(i)]**2)
print(x)

```

1. (i) 왼쪽은 경사하강법으로 오른쪽은 Momentum으로 이동한 점의 경로를 나타낸 그림이다. 수학(등위선, 접선, gradient, 벡터, 벡터의 덧셈)을 사용하여 점의 이동 경로를 각각 설명하시오.



- (ii) AdaGrad를 사용하여도 오른쪽과 비슷한 이동 경로를 얻을 수 있다. 수학(편미분)을 사용하여 점의 이동 경로를 설명하시오.

2. 이변수 함수

$$f(x, y) = xy$$

에 대하여 learning rate $\eta = 1$ 과 계수 $\beta_1 = \beta_2 = 1/2$ 로 Adam을 적용하려 한다. 초기 위치 $\mathbf{x}_0 = (1, 2)$ 에서 출발하여 두 발자국 걸어갈때, $\mathbf{x}_1, \mathbf{x}_2$ 를 구하시오.

3. 배치 정규화 (batch normalization)층으로 들어온 데이터 배치 묶음과 흘러들러온 미분이 각각

$$\mathbf{x} = \begin{pmatrix} 5 & 0 & 4 \\ 4 & 1 & 2 \\ 3 & 2 & 2 \\ 2 & 3 & 6 \\ 1 & 4 & 1 \end{pmatrix}, \quad \frac{\partial L}{\partial \mathbf{y}} = \begin{pmatrix} \sqrt{2} & -\sqrt{2} & -4\sqrt{5} \\ \sqrt{2} & -\sqrt{2} & 4\sqrt{5} \\ 1 & 1 & 4\sqrt{5} \\ -\sqrt{2} & \sqrt{2} & 4\sqrt{5} \\ -\sqrt{2} & \sqrt{2} & -8\sqrt{5} \end{pmatrix}$$

이다. 배치 정규화층의 확대(scale)는 $\gamma = [\sqrt{2}, \sqrt{2}, 4\sqrt{5}]$ 이고 이동(shift)은 $\beta = [2, 2, 10]$ 이다. 출력값 \mathbf{y} 와 미분 $\frac{\partial L}{\partial \beta}, \frac{\partial L}{\partial \gamma}$ 을 구하시오.

4. 활성화 함수가 Relu인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}, b_1 : [0, 0, 0, 0, 0], W_2 : \begin{bmatrix} -1 & 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix}, b_2 : [0, 0, 0, 0, 0]\}$$

로 주어져 있다. 데이터 $\mathbf{x} = [1, 2, 3, 4, 5]$ 의 라벨이 $[1, 0, 0, 0, 0]$ 이라 하자. 손실함수는 교차 엔트로피(cross entropy)에 L^2 -규제 (L^2 -regularization)가 가해져 있다. 가중치 감소의 세기가 $\lambda = \frac{1}{11}$ 일 때, 데이터 \mathbf{x} 의 손실함수 값을 구하시오.

5. 교차 엔트로피(cross entropy)값을 L_c 라 하고 L^2 -규제 (L^2 -regularization)까지 고려한 손실함수 값을 L 이라 하자. k 번째 Affine층의 가중치 행렬 W_k , 편향벡터 b_k 그리고 가중치 감소 계수 λ 에 대하여 등식

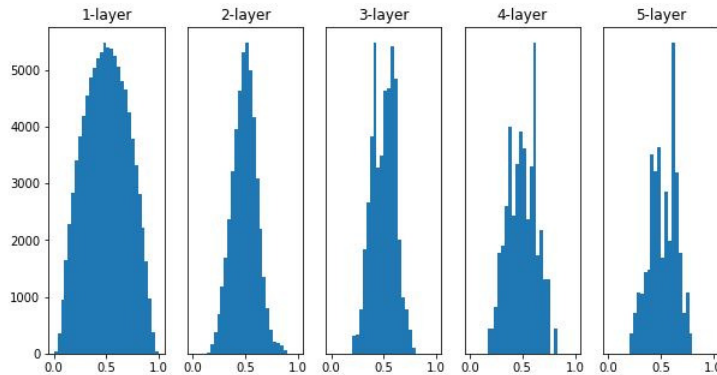
$$\frac{\partial L}{\partial W_k} = \frac{\partial L_c}{\partial W_k} + \lambda W_k, \quad \frac{\partial L}{\partial b_k} = \frac{\partial L_c}{\partial b_k}$$

을 설명하시오. (편의를 위해 W_k 를 2×2 행렬, b_k 를 2차원 벡터라 가정해도 좋다.)

6. 다음은 옵티마이저 RMSProp을 구현하는 코드이다. 빈 칸을 채우시오.

```
class RMSprop:
    def __init__(self, lr=0.01, decay_rate = 0.99):
        self.lr = lr
        self.decay_rate = decay_rate
        self.h = None
    def update(self, params, grads):
        if self.h is None:
            self.h = {}
        for key, val in params.items():
            self.h[key] = np.zeros_like(val)
        for key in params.keys():
            self.h[key]  self.decay_rate
            self.h[key] +=  * grads[key] * grads[key]
            params[key] -= self.lr * grads[key] / ( + 1e-7)
```

7. Xavier 초기값을 따라 가중치를 초기화 했을 때 다음과 같은 활성화층의 히스토그램을 얻는다. 빈 칸을 채우시오.



```
input_data = np.random.randn(1000, 100)
node_num = 100
hidden_layer_size = 5
activations = {}
x = input_data
for i in range(hidden_layer_size):
    if i != 0:
        x = activations[i-1]
```

```

w = np.random.randn(node_num, node_num) * 
a = np.dot(x, w)
z = sigmoid(a)
activations[i] = z
for i, a in activations.items():
    plt.(1, len(activations), i+1)
    plt.title(str(i+1) + "-layer")
    if : plt.yticks([], [])
    plt.(a.flatten(), 30, range=(0,1))
plt.show()

```

8. 다음 코드를 실행했을 때 출력될 값들을 대략적으로 쓰시오.

```

gamma = np.array([1,2,3])
beta = np.array([4,5,6])
BN = BatchNormalization(gamma, beta)
x = 2*np.random.randn(1000,3)+3
y = BN.forward(x)
print(np.mean(y,axis=0))
print(np.mean((y - np.mean(y,axis=0))*2, axis=0))

```

9. 다음은 Drop Out 층의 순전파와 역전파를 구현하는 코드이다.

```

class Dropout:
    def __init__(self, dropout_ratio=0.5):
        self.dropout_ratio = dropout_ratio
        self.mask = None
    def forward(self, x, train_flg=True):
        if train_flg:
            self.mask = np.random.rand(*x.shape) > self.dropout_ratio
            return x * self.mask
        else:
            return x * (1.0 - self.dropout_ratio)
    def backward(self, dout):
        return dout * self.mask

```

입력 데이터와 생성된 random 행렬이 각각

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad \begin{pmatrix} 0.4 & 0.9 & 0.1 \\ 0.2 & 0.2 & 0.6 \\ 0.9 & 0.4 & 0.1 \end{pmatrix}$$

이고 dropout ratio는 0.3라 하자. 훈련할 때와 테스트할 때 출력값이 각각 어떻게 되겠는가?

10. 다음은 심층 신경망 코드의 앞부분이다. 빈칸을 채우시오.

```

class MultiLayerNet:
    def __init__(self, input_size, hidden_size_list, output_size, activation='relu',

```

```

        weight_init_std='relu', weight_decay_lambda=0):
self.input_size = input_size
self.output_size = output_size
self.hidden_size_list = hidden_size_list
self.hidden_layer_num = len(hidden_size_list)
self.weight_decay_lambda = weight_decay_lambda
self.params = {}
self.__init_weight(weight_init_std)
activation_layer = {'sigmoid': Sigmoid, 'relu': Relu}
self.layers = OrderedDict()
for idx in range(1, self.hidden_layer_num+1):
    self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)],
                                              self.params['b' + str(idx)])
    self.layers['Activation_function' + str(idx)] = activation_layer[activation]()
idx = self.hidden_layer_num + 1
self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)],
                                          self.params['b' + str(idx)])

self.last_layer = SoftmaxWithLoss()
def __init_weight(self, weight_init_std):
all_size_list = [self.input_size] + self.hidden_size_list + [self.output_size]
for idx in range(1, len(all_size_list)):
    scale = weight_init_std
    if str(weight_init_std).lower() in ('relu', 'he'):
        scale = 
    elif str(weight_init_std).lower() in ('sigmoid', 'xavier'):
        scale = 
    self.params['W' + str(idx)] = 
    self.params['b' + str(idx)] = np.zeros(all_size_list[idx])
def predict(self, x):
for layer in self.layers.values():
    x = layer.forward(x)
return x
def loss(self, x, t):
y = self.predict(x)
weight_decay = 0
for idx in range(1, self.hidden_layer_num + 2):
    W = self.params['W' + str(idx)]
    weight_decay += 0.5 * self.weight_decay_lambda * 
return self.last_layer.forward(y, t) + 

```


1. 데이터 X 가 $N \times C \times H \times W = 2 \times 2 \times 3 \times 3$ 텐서

$$X = \left(\begin{pmatrix} 0 & 0 & 3 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \\ 3 & 0 & 0 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \right)$$

로, 필터 W 가 $FN \times C \times FH \times FW = 2 \times 2 \times 2 \times 2$ 텐서

$$W = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix} \right)$$

로 주어져 있다. padding=0와 stride=1로 합성곱한 값을 구하시오.

2. (1번 문제 계속) 다음은 합성곱 클래스의 순전파 코드이다. 편향이 $b_1 = -1, b_2 = -1$ 로 주어져 있다. 밑줄친 4개의 값을 구하시오.

```
def forward(self, x):
    FN, C, FH, FW = self.W.shape
    N, C, H, W = x.shape
    out_h = 1 + int((H + 2*self.pad - FH) / self.stride)
    out_w = 1 + int((W + 2*self.pad - FW) / self.stride)
    col = im2col(x, FH, FW, self.stride, self.pad)
    col_W = self.W.reshape(FN, -1).T
    out = np.dot(col, col_W) + self.b
    out = out.reshape(N, out_h, out_w, -1).transpose(0, 3, 1, 2)
    self.x = x
    self.col = col
    self.col_W = col_W
    return out
```

3. (1,2번 문제 계속) 주어진 합성곱층을 Affine층으로 바꾸시오. 즉, 입력 노드, 출력 노드, 에지를 그리고 에지위에 가중치를 표시하고 가중치 행렬과 편향 벡터를 구하고 데이터를 나열하시오.

4. (1,2번 문제 계속) 다음은 합성곱 클래스의 역전파 코드이다. 흘러 들어온 미분이

$$\frac{\partial L}{\partial Y} = \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix}$$

일 때, 밑줄친 6개의 값을 구하시오.

```
def backward(self, dout):
    FN, C, FH, FW = self.W.shape
    dout = dout.transpose(0,2,3,1).reshape(-1, FN)
    self.db = np.sum(dout, axis=0)
    self.dW = np.dot(self.col.T, dout)
    self.dW = self.dW.transpose(1, 0).reshape(FN, C, FH, FW)
    dcol = np.dot(dout, self.col.W.T)
    dx = col2im(dcol, self.x.shape, FH, FW, self.stride, self.pad)
    return dx
```

5. (i) 데이터 X 가 $N \times C \times H \times W = 2 \times 2 \times 3 \times 3$ 텐서

$$X = \begin{pmatrix} \begin{pmatrix} 4 & 0 & 5 \\ 1 & 2 & 3 \\ 7 & 6 & 8 \\ 2 & 0 & 1 \\ 8 & 7 & 3 \\ 4 & 5 & 6 \end{pmatrix} & \begin{pmatrix} 5 & 6 & 7 \\ 3 & 8 & 4 \\ 0 & 1 & 2 \\ 0 & 1 & 8 \\ 7 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \end{pmatrix}$$

로 주어져 있다. $PH=2, PW=2$, no padding, stride=1로 Max Pooling한 값을 구하시오.

- (ii) Max Pooling층으로 흘러들어온 미분이

$$\frac{\partial L}{\partial Y} = \begin{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 4 & 3 \\ 2 & 1 \end{pmatrix} & \begin{pmatrix} -1 & -2 \\ 3 & 4 \\ 4 & 3 \\ -2 & -1 \end{pmatrix} \end{pmatrix}$$

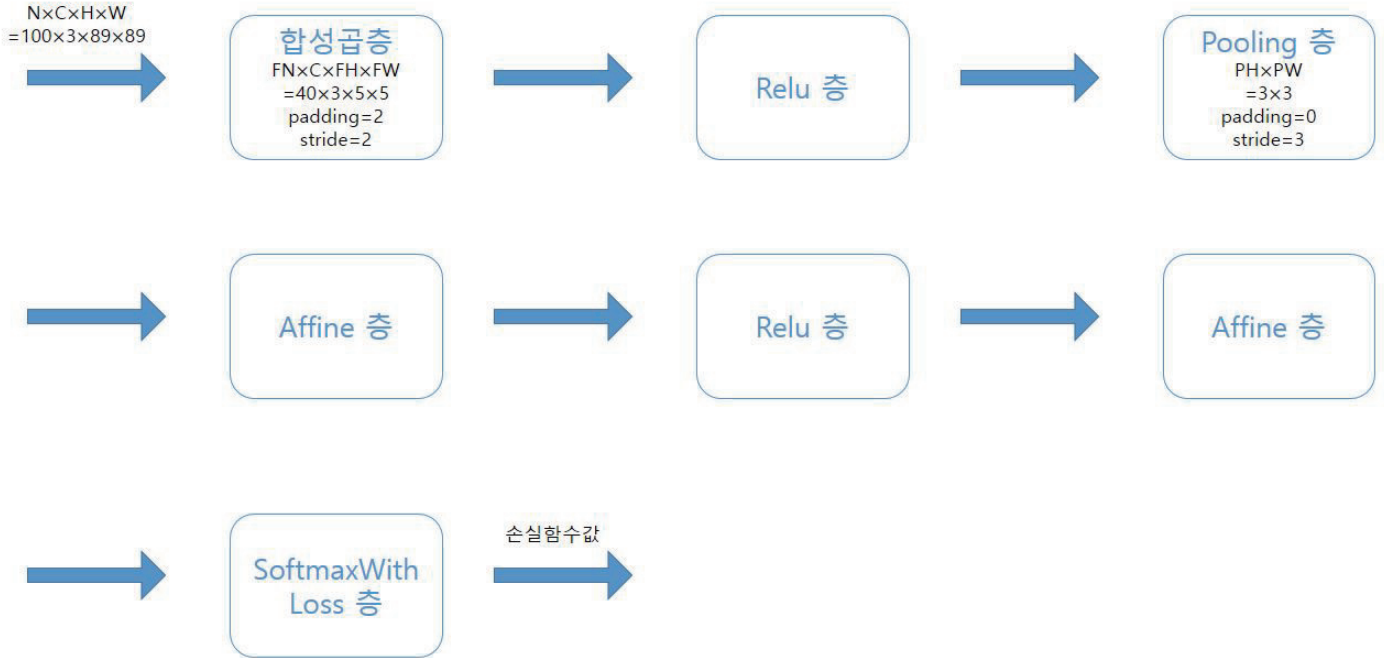
일 때, 데이터에 대한 미분

$$\frac{\partial L}{\partial X}$$

을 구하시오.

6. 합성곱층의 필터는 통상적으로 3×3 나 5×5 를 쓰는 경우가 많다. 합성곱 신경망에서 각 합성곱층의 필터의 해상도를 모두 동일하게 잡더라도 낮은 층의 필터는 좁은 영역의 로컬한 특징을 찾아내지만 높은 층으로 갈수록 점점 더 넓은 영역의 글로벌한 특징을 찾아낸다. 이게 가능한 이유를 설명하시오.

7. 합성곱 신경망이 다음과 같은 층으로 이루어져 있다. 은닉층의 뉴런수는 50개이고 출력층의 뉴런수는 10개이다. 89×89 해상도와 RGB 채널을 가지는 컬러사진 100장을 넣었을때 흘러가는 데이터의 shape을 화살표마다 순서대로 쓰시오. 첫번째 Affine층의 가중치 행렬의 shape을 구하시오.



8. 합성곱 신경망이



와 같이 주어져 있다. 합성곱층의 필터, 첫번째 Affine층의 가중치 행렬, 두번째 Affine층의 가중치 행렬이 각각

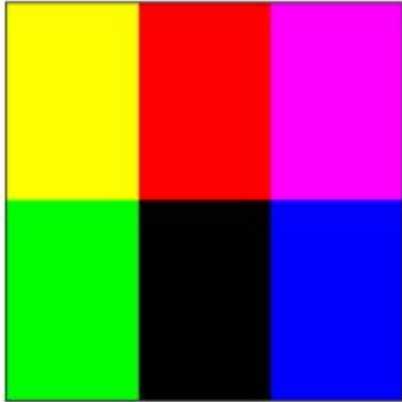
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} \log 2 & 0 & 0 & 0 \\ 0 & \log 3 & 0 & 0 \\ 0 & 0 & \log 4 & 0 \\ 0 & 0 & 0 & \log 5 \end{pmatrix}$$

으로 주어져 있다. 편향은 모두 없다. 합성곱층에서는 stride=1, padding=0이고 Max Pooling층에서는 $PH=PW=2$, stride=2, padding=0이다. 데이터

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

의 라벨이 $[0, 1, 0, 0]$ 일 때, 손실함수 값을 구하시오.

9. 오른쪽은 RGB 색상표이다. 빈 칸을 채워서 왼쪽 이미지가 출력되도록 하시오.



Color Fields	Color	Red	Green	Blue
White		255	255	255
Silver		192	192	192
Gray		128	128	128
Black		0	0	0
Red		255	0	0
Maroon		128	0	0
Yellow		255	255	0
Olive		128	128	0
Lime		0	255	0
Green		0	128	0
Aqua		0	255	255
Teal		0	128	128
Blue		0	0	255
Navy		0	0	128
Pink		255	102	255
Fuchsia		255	0	255
Purple		128	0	128

```
img = np.zeros((30,30,3))
img[ ] = 1
img[ ] = 1
img[ ] = 1
plt.xticks( )
plt.yticks( )
plt.imshow(img)
plt.show()
```

10. (9번 문제 계속) 다음 코드를 실행했을 때 출력될 이미지를 그리시오.

```
W = np.array([[[[1,0,-1],[2,0,-2],[1,0,-1]], [[1,0,-1],[2,0,-2],[1,0,-1]],
               [[1,2,1],[0,0,0],[-1,-2,-1]]]])
b=0
conv_layer = Convolution(W, b)
img = img.reshape(1,30,30,3)
img = img.transpose(0,3,1,2)
out = conv_layer.forward(img)
out = out[0].transpose(1,2,0)
plt.xticks( )
plt.yticks( )
plt.imshow(out, "gray")
plt.show()
```

1. 데이터 X 가 $N \times C \times H \times W = 2 \times 2 \times 3 \times 3$ 텐서

$$X = \left(\begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right)$$

로, 필터 W 가 $FN \times C \times FH \times FW = 2 \times 2 \times 2 \times 2$ 텐서

$$W = \left(\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \right)$$

로 주어져 있다. no padding과 stride=1로 합성곱한 값을 구하시오.

2. (1번 문제 계속) 다음은 합성곱 클래스의 순전파 코드이다. 편향이 $b_1 = -1$, $b_2 = -1$ 로 주어져 있다. 밑줄친 4개의 값을 구하시오.

```
def forward(self, x):
    FN, C, FH, FW = self.W.shape
    N, C, H, W = x.shape
    out_h = 1 + int((H + 2*self.pad - FH) / self.stride)
    out_w = 1 + int((W + 2*self.pad - FW) / self.stride)
    col = im2col(x, FH, FW, self.stride, self.pad)
    col_W = self.W.reshape(FN, -1).T
    out = np.dot(col, col_W) + self.b
    out = out.reshape(N, out_h, out_w, -1).transpose(0, 3, 1, 2)
    self.x = x
    self.col = col
    self.col_W = col_W
    return out
```

3. (1,2번 문제 계속) 주어진 합성곱층을 Affine층으로 바꾸시오. 즉, 입력 노드, 출력 노드, 에지를 그리고 에지위에 가중치를 표시하고 가중치 행렬과 편향 벡터를 구하고 데이터를 나열하시오.

4. (1,2번 문제 계속) 다음은 합성곱 클래스의 역전파 코드이다. 흘러 들어온 미분이

$$\frac{\partial L}{\partial Y} = \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix}$$

일 때, 밑줄친 6개의 값을 구하시오.

```
def backward(self, dout):
    FN, C, FH, FW = self.W.shape
    dout = dout.transpose(0,2,3,1).reshape(-1, FN)
    self.db = np.sum(dout, axis=0)
    self.dW = np.dot(self.col.T, dout)
    self.dW = self.dW.transpose(1, 0).reshape(FN, C, FH, FW)
    dcol = np.dot(dout, self.col_W.T)
    dx = col2im(dcol, self.x.shape, FH, FW, self.stride, self.pad)
    return dx
```

5. (i) 데이터 X 가 $N \times C \times H \times W = 2 \times 2 \times 3 \times 3$ 텐서

$$X = \begin{pmatrix} \begin{pmatrix} 2 & 0 & 6 \\ 8 & 7 & 5 \\ 4 & 3 & 1 \end{pmatrix} & \begin{pmatrix} 7 & 6 & 5 \\ 3 & 8 & 4 \\ 0 & 2 & 1 \end{pmatrix} \\ \begin{pmatrix} 7 & 0 & 3 \\ 5 & 2 & 1 \\ 6 & 4 & 8 \end{pmatrix} & \begin{pmatrix} 0 & 6 & 3 \\ 1 & 2 & 5 \\ 4 & 8 & 7 \end{pmatrix} \end{pmatrix}$$

로 주어져 있다. $PH=2, PW=2$, no padding, stride=1로 Max Pooling한 값을 구하시오.

- (ii) Max Pooling층으로 흘러들어온 미분이

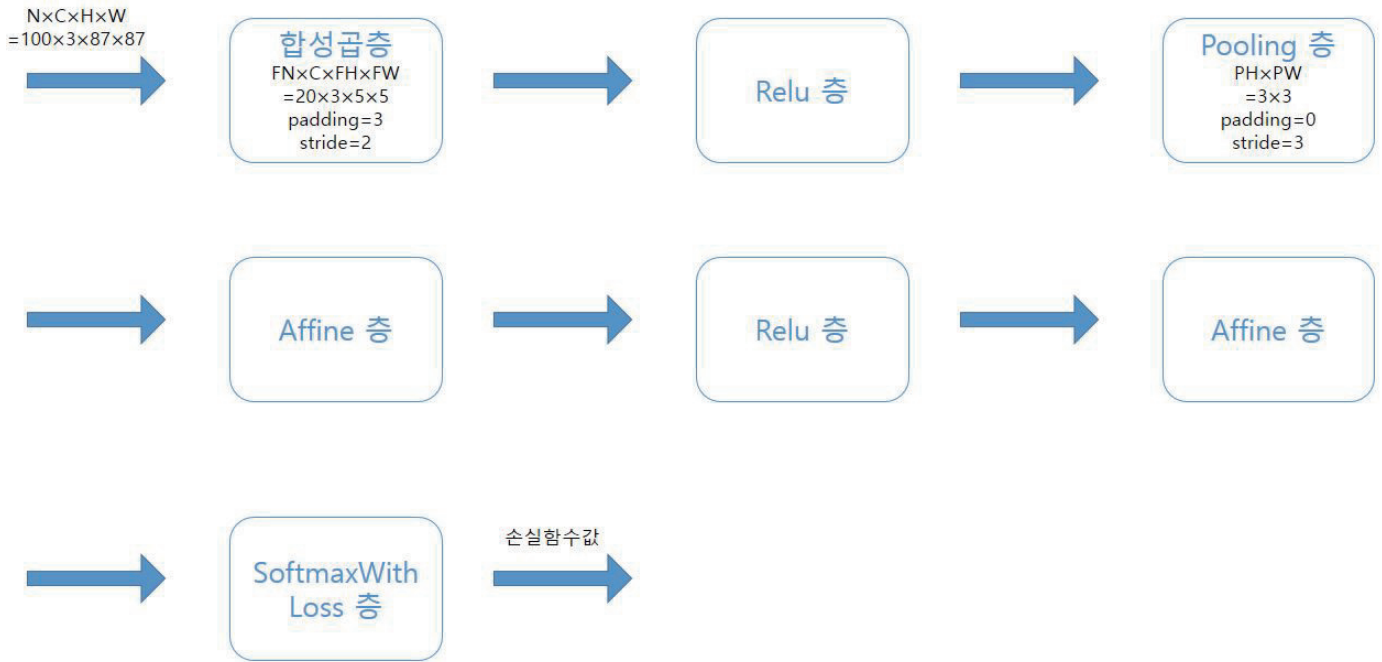
$$\frac{\partial L}{\partial Y} = \begin{pmatrix} \begin{pmatrix} -1 & -2 \\ 3 & 4 \end{pmatrix} & \begin{pmatrix} 1 & -2 \\ -3 & 4 \end{pmatrix} \\ \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix} & \begin{pmatrix} 4 & -3 \\ 2 & -1 \end{pmatrix} \end{pmatrix}$$

일 때, 데이터에 대한 미분

$$\frac{\partial L}{\partial X}$$

을 구하시오.

6. 합성곱 신경망이 다음과 같은 층으로 이루어져 있다. 은닉층의 뉴런수는 80개이고 출력층의 뉴런수는 10개이다. 87×87 해상도와 RGB 채널을 가지는 컬러사진 100장을 넣었을때 흘러가는 데이터의 shape을 화살표마다 순서대로 쓰시오. 첫번째 Affine층의 가중치 행렬의 shape을 구하시오.



7. (6번 문제 계속) He 초기값을 따라 필터와 가중치 행렬을 초기화하려 한다. 합성곱층의 필터 W_1 , 첫번째 Affine층의 가중치 행렬 W_2 , 두번째 Affine층의 가중치 행렬 W_3 를 각각 어떤 분포를 따라 랜덤하게 초기화해야 하는가?

8. 합성곱 신경망이



와 같이 주어져 있다. 합성곱층의 필터, 첫번째 Affine층의 가중치 행렬, 두번째 Affine층의 가중치 행렬이 각각

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} \log 5 & 0 & 0 & 0 \\ 0 & \log 4 & 0 & 0 \\ 0 & 0 & \log 3 & 0 \\ 0 & 0 & 0 & \log 2 \end{pmatrix}$$

으로 주어져 있다. 편향은 모두 없다. 합성곱층에서는 stride=1, padding=0이고 Max Pooling층에서는 $PH=PW=2$, stride=2, padding=0이다. 데이터

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

의 라벨이 $[0, 1, 0, 0]$ 일 때, 손실함수 값을 구하시오.

9. 필터가 파이토치에서는 $FN \times C \times FH \times FW$ 포맷을 따르고 텐서플로우에서는 $FH \times FW \times C \times FN$ 포맷을 따른다.

- (i) 파이토치 코드를 텐서플로우 코드로 바꾸려 한다. 필터에 어떤 transpose를 해야 하는가?
- (ii) 텐서플로우 코드를 파이토치 코드로 바꾸려 한다. 필터에 어떤 transpose를 해야 하는가?

10. 다음은 교재의 im2col로 구현한 합성곱이 정의대로 구현한 합성곱에 비해 몇배나 빠른지 측정하는 코드이다. 빈 칸을 채우시오.

```
import time
from common.layers import Convolution

input = np.random.rand(100,3,100,100)
filters = np.random.rand(20,3,5,5)

pad = 0
stride = 1
N, C, H, W = input.shape
FN, C, FH, FW = filters.shape
out_h = 1 + int((H + 2*pad - FH) / stride)
out_w = 1 + int((W + 2*pad - FW) / stride)
out = np.zeros((N, FN, out_h, out_w))

start_naive = time.time()
for i in range(N):
    for j in range(FN):
        for k in range(out_h):
            h_start = k * stride
            h_end = h_start + FH
            for l in range(out_w):
                w_start = l * stride
                w_end = w_start + FW
                out[i,j,k,l] = 
end_naive = time.time()

conv_layer= Convolution(filters,0)

start_im2col = time.time()
conv_layer.forward(input)
end_im2col = time.time()

print()
```