딥러닝 I (1반) / 중간고사 4. 25. 2022

1. (i) 단층 퍼셉트론으로는 동치를 나타내는 조건 명제

$$(p \to q) \quad \text{and} \quad (q \to p)$$

를 구현할 수 없음을 기하학적으로 설명하시오.

(ii) 다음 XOR게이트 코드를 수정하여 (i)을 구현하시오.

```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

2. (i) 시그모이드 함수는 등식

$$y' = y(1-y)$$

을 만족함을 보이시오.

(ii) (i)을 이용하여 시그모이드 함수는 증가 함수임을 보이시오.

(iii) (i)과 (ii)를 이용하여 시그모이드 함수의 그래프는 $x > 0$일때 위로 볼록하고 $x < 0$일때 아래로 볼록함을 보이시오.

3. 활성화 함수가 시그모이드인 3층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, b_1 : [0,0,0], W_2 : \begin{bmatrix} 3\log 2 & 9\log 2 \\ -5\log 2 & 0 \\ 0 & -9\log 2 \end{bmatrix}, b_2 : [0,0],\ W_3 : \begin{bmatrix} 10\log 2 & 1 \\ -5\log 2 & -1 \end{bmatrix}, b_3 : [2022, 2022]\}$$

로 주어져 있다. $[\log 2, 0]$을 입력했을때 출력되는 소프트맥스(softmax)값을 구하시오.

4. 활성화 함수가 ReLU인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}, b_1 : [0,0,0,0,0],\ W_2 : \begin{bmatrix} \log 2 & 0 & 0 & 0 & 0 \\ 0 & \log 3 & 0 & 0 & 0 \\ 0 & 0 & \log 4 & 0 & 0 \\ 0 & 0 & 0 & \log 5 & 0 \\ 0 & 0 & 0 & 0 & \log 6 \end{bmatrix}, b_2 : [0,0,0,0,0]\}$$

로 주어져 있다. 3개의 데이터 $[0,1,1,1,1]$, $[1,0,1,1,1]$, $[1,1,0,1,1]$의 라벨이 각각 $[1,0,0,0,0]$, $[0,0,1,0,0]$, $[0,0,0,0,1]$이라 하자. 배치처리로 계산하여 교차 엔트로피(cross entropy) 값을 구하시오.

5. 라벨이 각각 $1,2,3,4,5,6,7,8,9$인 9개의 손글씨를 신경망에 넣었다. 라벨이 k인 손글씨를 보고 k일것 같다고 예측한 확률이

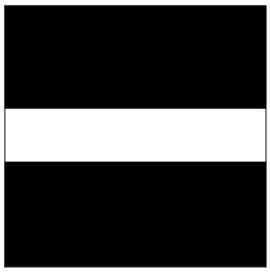
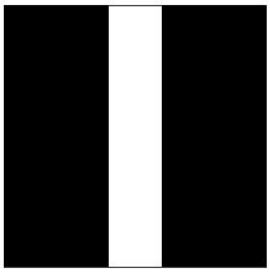
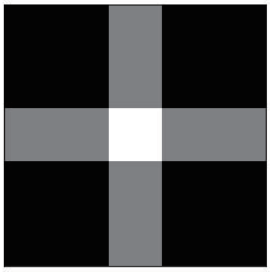
$$\frac{k}{k+1}$$

이었을 때, 교차 엔트로피 (cross entropy)값을 구하시오.

6. 다음 코드를 실행했을 때 출력될 값 5개를 순서대로 쓰시오.

```
x=np.arange(12).reshape(3,4)
print(x)
x-=7
print(x)
x+=np.array([1,2,3,4])
print(x)
x = (x>0)
print(x)
x = np.sum(x,axis=0)
print(x)
```

7. 적절한 5×5 행렬에 plt.imshow를 적용하여 다음 세 이미지를 출력하려 한다. 3개의 빈 칸을 순서대로 채우시오.

```
x1=np.zeros((5,5))
x1[      ]=1
plt.imshow(x1,cmap=plt.cm.gray)
plt.show()

x2=np.zeros((5,5))
x2[      ]=1
plt.imshow(x2,cmap=plt.cm.gray)
plt.show()

x=
plt.imshow(x,cmap=plt.cm.gray)
plt.show()
```

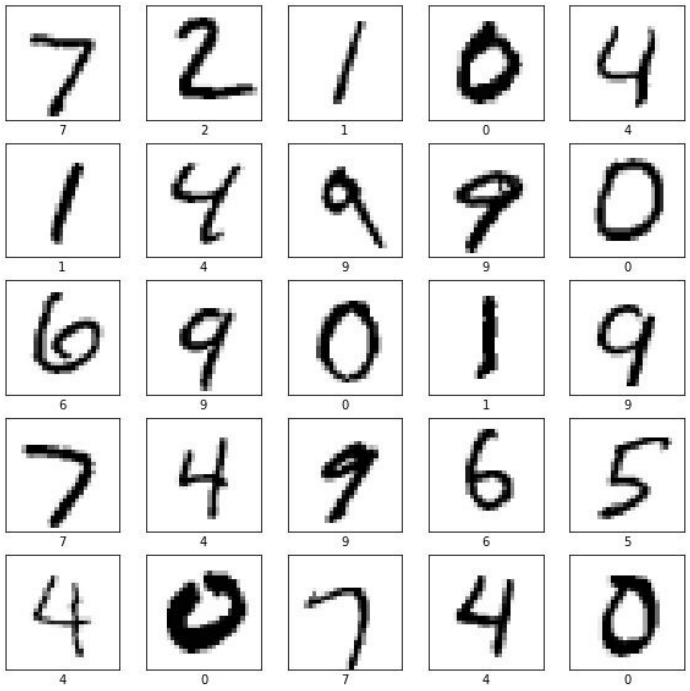
8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist(normalize=True,
    flatten=True, one_hot_label=False)
    return x_test, t_test

def init_network():
    with open("sample_weight.pkl", 'rb') as f:
    network = pickle.      (f)
    return network
```

```
def predict(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = softmax(a3)
    return y
x, t = get_data()
network = init_network()

batch_size = 100
accuracy_cnt = 0

for i in range(0, len(x), batch_size):
    x_batch = x[      ]
    y_batch = predict(network, x_batch)
    p = np.argmax(y_batch,      )
    accuracy_cnt      np.sum(p == t[i:i+batch_size])

print("Accuracy:" + str(float(accuracy_cnt) /      ))
```

9. 8번 문제 코드에 추가하여 신경망이 90프로 이상의 확신을 가지고 맞춘 이미지가 몇프로 인지 구하고 첫 25장의 이미지를 5×5 테이블로 출력하려 한다. 5개의 빈 칸을 순서대로 채우시오.

```
0.7519
```



```
confident=[]
for i in range(len(x)):
    y = predict(network, x[i])
    p= np.argmax(y)
    if (      ) & (      ):
        confident.append(i)
print(len(confident)/len(x))
plt.figure(figsize=(10,10))
for i in range(25):
    plt.      (5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(      .reshape(28,28), cmap=plt.cm.binary)
    plt.xlabel(str(      ))
plt.show()
```

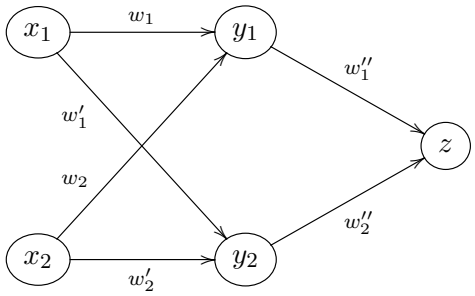
10. 8번 문제 코드에 다음 코드를 추가하면 다음과 같은 10×10 행렬이 출력이 된다. 신경망이 6을 0이라고 답한 횟수를 답하고 그 이유를 설명하시오. 대각원이 큰 숫자가 나오는 이유도 설명하시오.

```
confusion = np.zeros((10,10), dtype=int)

for k in range(len(x)):
    i = int(t[k])
    y = predict(network, x[k])
    j = np.argmax(y)
    confusion[i][j] += 1

print(confusion)
```

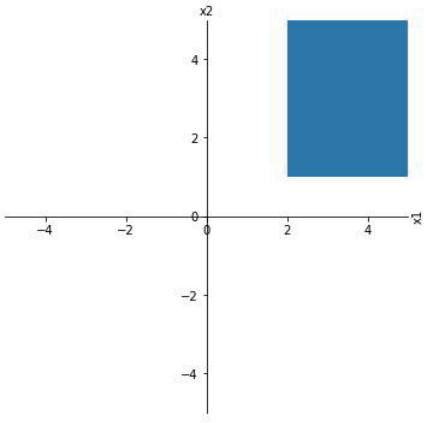
```
[[ 962    0    3    2    1    3    5    1    3    0]
 [   0 1109    2    2    0    2    5    2   13    0]
 [  13    2  952    7    7    1   15   13   19    3]
 [   1    1   24  937    0   20    1   11   11    4]
 [   1    2    6    0  921    0   12    2    3   35]
 [  10    1    4   34    5  793   14    5   18    8]
 [  16    3    5    1   10    9  910    1    3    0]
 [   4    8   24    5    6    0    0  958    1   22]
 [   5    4    6   19   10   13   14   11  888    4]
 [  12    6    1    9   31    7    1   16    4  922]]
```

딥러닝 I (2반) / 중간고사 4. 22. 2022

1. 다음 이층 퍼셉트론을 생각하자.



가중치와 임계값 (w_1, w_2, θ), (w_1', w_2', θ'), (w_1'', w_2'', θ'')을 적절히 선택하여 출력값이 1이 되는 실수쌍 (x_1, x_2)이 다음 영역이 되도록 하시오.



2. (a_1, a_2, a_3)과 $(a_1 + C, a_2 + C, a_3 + C)$의 소프트맥스(softmax) 값은 동일함을 보이시오.

3. 활성화 함수가 시그모이드인 3층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} -1 & -2 & -3 \\ 4 & 5 & 6 \end{bmatrix}, b_1 : [0,0,0], W_2 : \begin{bmatrix} -3\log 2 & -6\log 2 \\ -5\log 2 & 0 \\ -9\log 2 & 0 \end{bmatrix}, b_2 : [0,0],\ W_3 : \begin{bmatrix} 9\log 2 & 9\log 2 \\ 5\log 2 & -5\log 2 \end{bmatrix}, b_3 : [2022, 2022]\}$$

로 주어져 있다. $[\log 2, 0]$을 입력했을때 출력되는 소프트맥스(softmax)값을 구하시오.

4. 활성화 함수가 ReLU인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},\ b_1 : [0,0,0,0,0],\ W_2 : \begin{bmatrix} 0 & 0 & 0 & 0 & \log 5 \\ \log 2 & 0 & 0 & 0 & 0 \\ 0 & \log 3 & 0 & 0 & 0 \\ 0 & 0 & \log 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ b_2 : [0,0,0,0,0]\}$$

로 주어져 있다. 3개의 데이터 $[1,2,3,4,5]$와 $[2,3,4,5,6]$와 $[3,4,5,6,7]$의 라벨이 모두 $[1,0,0,0,0]$이라 하자. 배치처리로 계산하여 교차 엔트로피(cross entropy) 값을 구하시오.

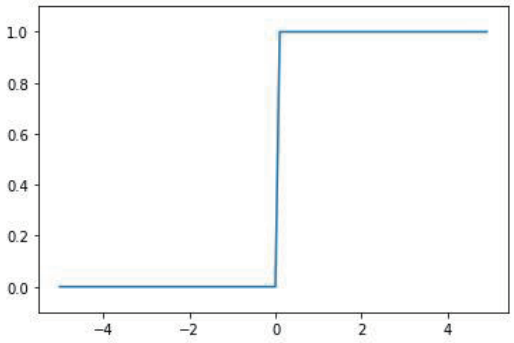
5. 라벨이 각각 0,1,2,3,4,5,6,7,8,9인 10개의 손글씨를 신경망에 넣었더니 모두

$$[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, 0, 0, 0, 0]$$

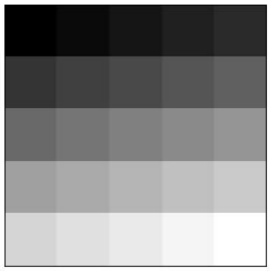
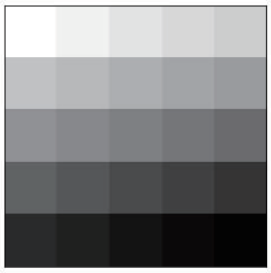
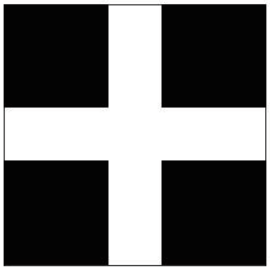
로 동일한 예측값이 나왔다. 평균제곱오차 (mean squared error)값을 구하시오.

6. 계단 함수 그래프를 그리기 위해 다음 코드를 실행하면 다음 그림처럼 불필요한 세로선이 표시된다. 그 이유를 설명하시오.

```
def step_function(x):
    return np.array(x > 0, dtype=np.int)

X = np.arange(-5.0, 5.0, 0.1)
Y = step_function(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)
plt.show()
```



7. 적절한 5×5 행렬에 `plt.imshow`를 적용하여 다음 세 이미지를 출력하려 한다. 4개의 빈 칸을 순서대로 채우시오.

```
x=np.arange(25).[      ]
plt.imshow(x,cmap=plt.cm.gray)
plt.show()

x=[      ]-x
plt.imshow(x,cmap=plt.cm.gray)
plt.show()

x=np.zeros((5,5))
x[      ]=1
x[      ]=1
plt.imshow(x,cmap=plt.cm.gray)
plt.show()
```

8. 다음은 cross_entropy_error 함수를 수정한 코드이다. 출력될 3개의 값을 순서대로 쓰시오.

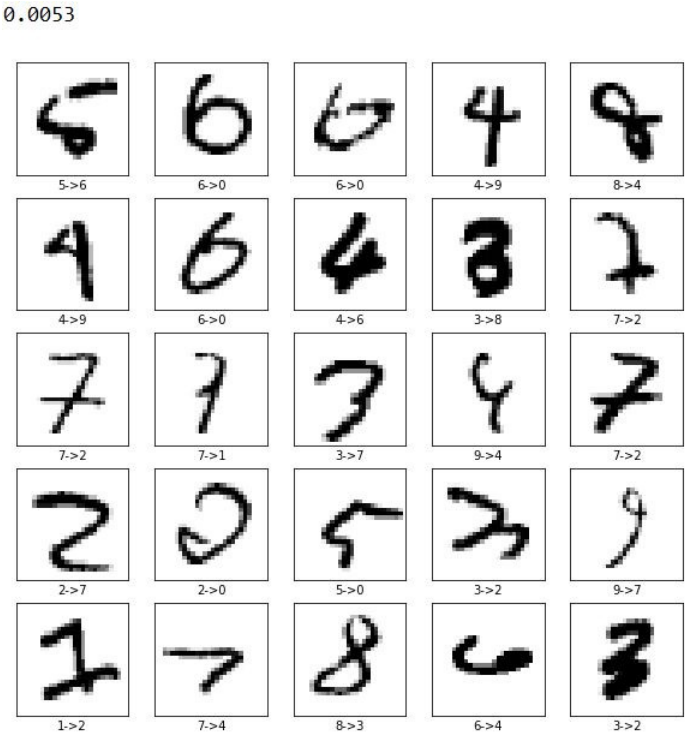
```
y = np.zeros((5,6))
for i in range(5):
    y[i,i]=np.exp(-(i+1))
    y[i,-1]=1-np.exp(-(i+1))
t = np.zeros((5,6))
for i in range(5):
    t[i,i]=1

if t.size == y.size:
    t = t.argmax(axis=1)
print(t)

batch_size = y.shape[0]
print(batch_size)

loss = -np.sum(np.log(y[np.arange(batch_size), t])) / batch_size
print(loss)
```

9. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist(normalize=True,
    flatten=True, one_hot_label=False)
    return x_test, t_test

def init_network():
    with open("sample_weight.pkl", 'rb') as f:
    network = pickle.[      ](f)
    return network

def predict(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = softmax(a3)
    return y

x, t = get_data()
network = init_network()

batch_size = 100
accuracy_cnt = [      ]

for i in range(0, len(x), [      ]):
    x_batch = x[i:i+batch_size]
    y_batch = predict(network, x_batch)
    p = np.[      ](y_batch, axis=1)
```

```
    accuracy_cnt += np.sum(p [    ] t[i:i+batch_size])

print("Accuracy:" + str(float(accuracy_cnt) / len(x)))
```

10. 9번 문제 코드에 추가하여 신경망이 90프로 이상의 확신을 가지고 대답했으나 틀린 이미지가 몇 프로인지 구하려고 한다. 또한, 다음과 같이 첫 25장의 이미지를 5×5 테이블로 출력하고 이미지 밑 라벨 옆에 신경망이 답한 숫자도 표시하시하려 한다. 5개의 빈 칸을 순서대로 채우시오.




```
error=[]
answer=[]

for i in range(len(x)):
    y = predict(network, x[i])
    p= np.argmax(y)
    if ([    ]) & ([    ]):
        error.append(i)
        answer.append(p)
print(len(error)/len(x))

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow([    ].reshape(28,28), cmap=plt.cm.binary)
    plt.xlabel(str([    ])+'->'+str([    ]))
plt.show()
```

딥러닝 I (1반) / 기말고사 6. 20. 2022

1. 함수

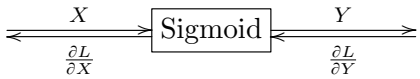
$$f(x,y) = 2x^2 + y^3 - 3xy - 9x + 3y$$

에 대하여 초기위치 $(0,0)$에서 출발하여 학습률(learning rate) 1/3로 경사하강법을 적용하려 한다. 두 걸음 갔을 때 위치를 구하시오.

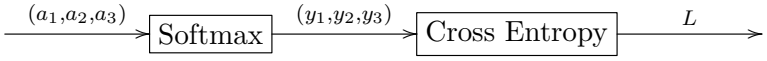
2. Sigmoid 층으로 들어오는 데이터 셋 X와 위에서 흘러들어오는 미분 $\frac{\partial L}{\partial Y}$이 각각

$$X = \begin{pmatrix} \log 2 & \log 3 & \log 4 & \log 5 \\ \log 6 & \log 7 & \log 8 & \log 9 \end{pmatrix}, \qquad \frac{\partial L}{\partial Y} = \begin{pmatrix} 3^2 & 4^2 & 5^2 & 6^2 \\ 7^2 & 8^2 & 9^2 & 10^2 \end{pmatrix}$$

일 때, 밑으로 흘려보내는 미분 $\frac{\partial L}{\partial X}$을 구하시오.



3. Softmax-with-Loss 계층



의 역전파는

$$(y_1 - t_1, y_2 - t_2, y_3 - t_3)$$

로 주어짐을 미분법 또는 계산그래프를 이용하여 유도하시오.

4. 선형변환

$$S(x_1, x_2, x_3, x_4) = (x_2, x_1, x_4, x_3)$$

을 생각하자. 이 선형변환으로 흘러들어오는 미분이 (d_1, d_2, d_3, d_4)일 때, 역전파를 거쳐 밑으로 흘러가는 미분은

$$(d_2, d_1, d_4, d_3)$$

임을 보이시오.

5. 활성화 함수가 ReLU인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},\ b_1 : [0,0,0,0,0],\ W_2 : \begin{bmatrix} 5 & 0 & 0 & 0 & 5 \\ 5 & 0 & 0 & 0 & 5 \\ 5 & 0 & 0 & 0 & 5 \\ 5 & 0 & 0 & 0 & 5 \\ -5 & 5 & 5 & 5 & -5 \end{bmatrix},\ b_2 : [0,0,0,0,0]\}$$

로 주어져 있다. 입력된 데이터는 $[1,2,3,4,5]$이고 라벨이 $[1,0,0,0,0]$일 때, 미분

$$\frac{\partial L}{\partial W_1}, \qquad \frac{\partial L}{\partial b_1}, \qquad \frac{\partial L}{\partial W_2}, \qquad \frac{\partial L}{\partial b_2}$$

을 구하시오.

6. (i) 다음은 경사하강법을 따라 이동하는 점의 경로를 표시하는 코드이다. 출력될 그림을 그리시오.

```
def gradient_descent(f, init_x, lr=0.01, step_num=100):
    x = init_x
    x_history = []

    for i in range(step_num):
        x_history.append(x.copy())
        grad = numerical_gradient(f,x)
        x -= lr * grad
    return x, np.array(x_history)

def function_2(x):
    return x[0]**2 + x[1]**2

init_x = np.array([-3.0, 4.0])

lr = 0.1
step_num = 20
x, x_history = gradient_descent(function_2, init_x, lr=lr, step_num=step_num)

plt.plot( [-5, 5], [0,0], '--b')
plt.plot( [0,0], [-5, 5], '--b')
plt.plot(x_history[:,0], x_history[:,1], 'o')
plt.xlim(-3.5, 3.5)
plt.ylim(-4.5, 4.5)
plt.xlabel("X0")
plt.ylabel("X1")
plt.show()
```

(ii) lr = 0.01로 수정했을때 출력될 그림을 그리시오.
(iii) lr = 1.01로 수정했을때 출력될 그림을 그리시오.
(iv) .copy()를 삭제했을 때 출력될 그림을 그리시오.

7. 다음은 Affine 계층 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
class Affine:
    def __init__(self, W, b):
        self.W = W
        self.b = b

    def forward(self, x):
        self.x = x
        out = np.dot(self.x, self.W) + self.b
        return out

    def backward(self, dout):
        dx = np.dot(      ,       )
        self.dW = np.dot(      ,       )
        self.db = np.sum(dout,       )
        return dx
```

8. 다음 코드를 실행하면 다음과 같은 출력값이 나온다.

```
layers=[]
```

```
b = np.zeros(100)
for k in range(11):
    W = 0.1*np.random.randn(100,100)
    layers.append(Affine(W,b))
    if k != 10:
        layers.append(Sigmoid())
x = np.random.rand(1,100)
for k in range(21):
    x = layers[k].forward(x)
dout = np.random.rand(1,100)
for k in range(21):
    dout = layers[20-k].backward(dout)
    if k % 2 == 0:
        print(np.sum((layers[20-k].dW)**2))
```

```
902.4785606506806
51.597695851281784
4.188285226425244
0.17557297587332535
0.0090041816780866
0.0005788259219041987
3.169636643830432e-05
1.512800199810902e-06
8.709281946181118e-08
5.9768406738885675e-09
3.538439536682618e-10
```

(i) 출력된 값들이 무엇인지 쓰시오.
(ii) 위와 같이 가파르게 작아지는 수학적 이유를 설명하시오.
(iii) 신경망을 학습 시킬때 어떤 문제가 생기는지 쓰시오.
(iv) 해결 방법을 쓰시오.

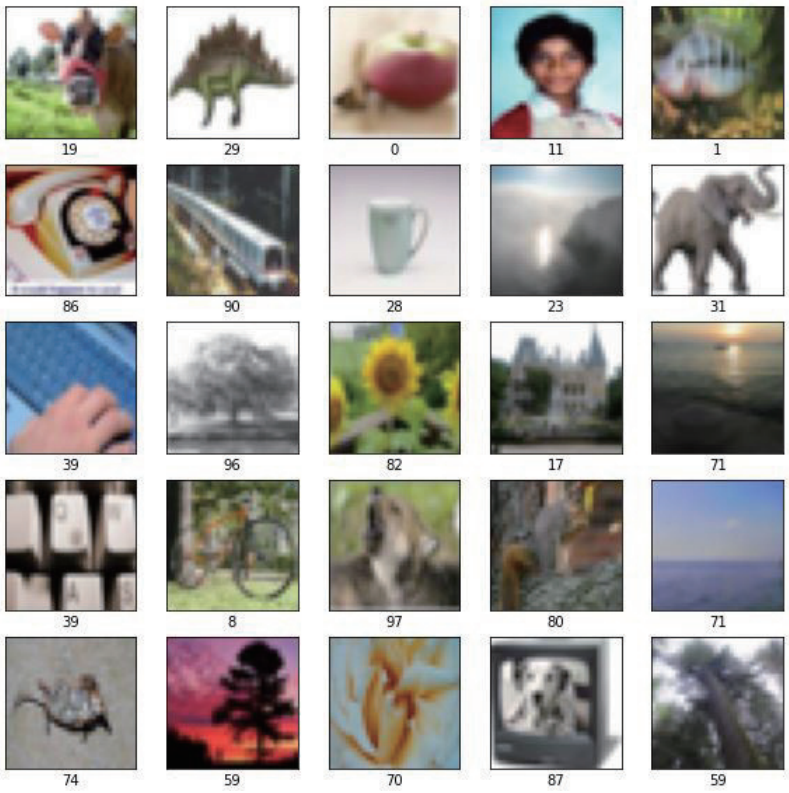
9. 다음은 역전파를 이용한 이층 신경망 코드이다. 4개의 빈 칸을 순서대로 채우시오.

```
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size, init_std = 0.01):
        self.params = {}
        self.params['W1'] = init_std * np.random.randn(input_size, hidden_size)
        self.params['b1'] = np.zeros(hidden_size)
        self.params['W2'] = init_std * np.random.randn(hidden_size, output_size)
        self.params['b2'] = np.zeros(output_size)
        self.layers = OrderedDict()
        self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])
        self.layers['Relu1'] = Relu()
        self.layers['Affine2'] = Affine(self.params['W2'], self.params['b2'])
        self.lastLayer = SoftmaxWithLoss()

    def predict(self, x):
        for layer in self.layers.[      ]():
            x = layer.forward(x)
        return x

    def loss(self, x, t):
        y = self.predict(x)
        return [      ].forward(y, t)
```

```
def gradient(self, x, t):
    self.loss(x, t)
    dout = [      ]
    dout = self.lastLayer.backward(dout)
    layers = list(self.layers.values())
    layers.[      ]()
    for layer in layers:
        dout = layer.backward(dout)

    grads = {}
    grads['W1'] = self.layers['Affine1'].dW
    grads['b1'] = self.layers['Affine1'].db
    grads['W2'] = self.layers['Affine2'].dW
    grads['b2'] = self.layers['Affine2'].db
    return grads
```

10. CIFAR100 데이터의 키, 벨류들의 shape, 훈련데이터 첫 25장의 이미지를 출력하면 다음과 같다. 라벨은 0부터 99까지의 정수로 이루어져 있다.

```
dict_keys(['x_train', 't_train', 'x_test', 't_test'])
(50000, 32, 32, 3)
(50000,)
(10000, 32, 32, 3)
(10000,)
```



(i) 다음과 같이 신경망 인스턴스를 만든후 CIFAR100 데이터셋으로 훈련을 시키려 한다. 빈 칸을 채우시오.

```
network = TwoLayerNet(input_size=[      ], hidden_size=50, output_size=[      ])
```

(ii) 학습시작전 정확도는 대략 몇 프로가 나오겠는가?

딥러닝 I (2반) / 기말고사 6. 17. 2022

1. 함수

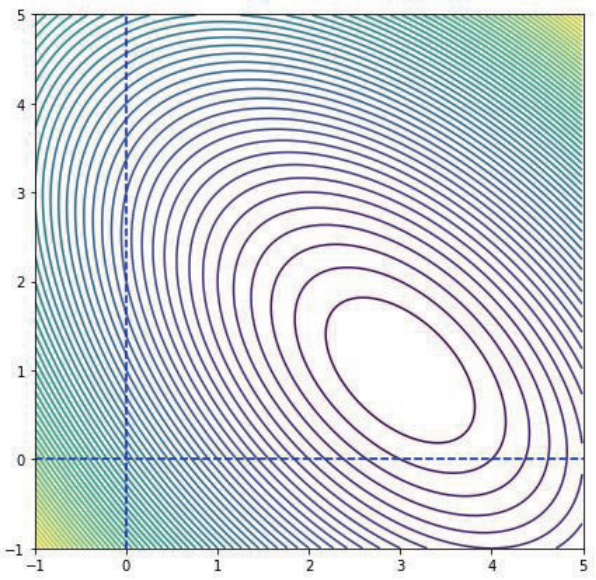
$$f(x,y) = x^2 + y^2 + xy - 7x - 5y$$

를 생각하자.

(i) $\nabla f(x,y) = (0,0)$을 풀어 $(3,1)$이 유일한 극점임을 보이시오.

(ii) 헤세 판정법으로 $(3,1)$이 최소점임을 보이시오.

(iii) 다음은 함수 f의 등위선들이다. $(0,0)$부터 출발하여 경사하강법을 적용했을때 이동하는 괘적을 곡선으로 그리고 수학적인 이유를 설명하시오.

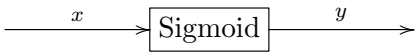


2. 함수

$$f(x,y) = x^2 + 2y^2 - xy - 8x - 4y$$

에 대하여 초기위치 $(0,0)$에서 출발하여 학습률(learning rate) 1/2로 경사하강법을 적용하려 한다. 두 걸음 갔을 때 위치를 구하시오.

3. Sigmoid 계층



의 역전파는

$$\frac{\partial L}{\partial y}\, y(1-y)$$

로 주어짐을 미분법을 이용하여 보이시오.

4. ReLU층으로 들어오는 데이터 배치 묶음 X와 위에서 흘러들어오는 미분 $\frac{\partial L}{\partial Y}$이 각각

$$X = \begin{pmatrix} 1 & -2 & 3 & -4 \\ -5 & 6 & -7 & 8 \end{pmatrix}, \qquad \frac{\partial L}{\partial Y} = \begin{pmatrix} 1 & -2 & -3 & 4 \\ -1 & 2 & 3 & -4 \end{pmatrix}$$

일 때, 밑으로 흘려보내는 미분 $\frac{\partial L}{\partial X}$을 구하시오.

5. 활성화 함수가 sigmoid인 이층 신경망이 dictionary

$$\{W_1 : \begin{bmatrix} \log 2 & \log 2 & 0 \\ \log 2 & 0 & \log 2 \end{bmatrix},\ b_1 : [0,0,0],\ W_2 : \begin{bmatrix} 9 & 0 \\ 0 & 5 \\ 0 & 6 \end{bmatrix},\ b_2 : [0,0]\}$$

로 주어져 있다. 입력된 데이터는 $[2,1]$이고 라벨이 $[1,0]$일 때, 미분

$$\frac{\partial L}{\partial W_1}, \qquad \frac{\partial L}{\partial b_1}, \qquad \frac{\partial L}{\partial W_2}, \qquad \frac{\partial L}{\partial b_2}$$

을 구하시오.

6. 다음은 역전파를 이용한 이층 신경망 코드이다. 4개의 빈 칸을 순서대로 채우시오.

```
class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size, init_std = 0.01):
        self.params = {}
        self.params['W1'] = init_std * np.random.randn(input_size, hidden_size)
        self.params['b1'] = np.zeros(hidden_size)
        self.params['W2'] = init_std * np.random.randn(hidden_size, output_size)
        self.params['b2'] = np.zeros(output_size)
        self.layers = OrderedDict()
        self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])
        self.layers['Relu1'] = Relu()
        self.layers['Affine2'] = Affine(self.params['W2'], self.params['b2'])
        self.lastLayer = SoftmaxWithLoss()

    def predict(self, x):
        for layer in self.layers.values():
            [      ] = layer.forward(x)
        return x

    def loss(self, x, t):
        y = [      ](x)
        return self.lastLayer.forward(y, t)

    def gradient(self, x, t):
        self.loss(x, t)
        dout = 1
        dout = self.lastLayer.backward(dout)
        layers = list(self.layers.values())
        layers.[      ]()
        for layer in layers:
            dout = layer.backward(dout)
        grads = [      ]
        grads['W1'], grads['b1'] = self.layers['Affine1'].dW,
                                   self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW,
                                   self.layers['Affine2'].db
        return grads
```

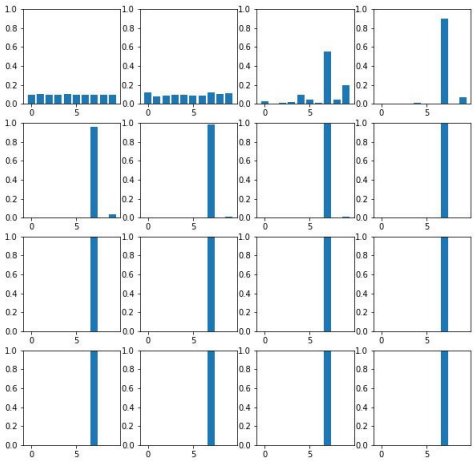
7. (6번 문제 코드 계속)
   (i) 입력층, 은닉층, 출력층의 뉴런의 개수가 각각 3개, 5개, 3개인 얕은 신경망을 만들고 싶다. 본인 이름의 이니셜을 따서 인스턴스를 만드시오.
   (ii) 세 개의 데이터 $[1,2,3]$, $[4,5,6]$, $[7,8,9]$를 배치처리를 통해 점수(score)를 구하는 코드를 쓰시오.

(iii) 각 라벨이 $[1,0,0]$, $[0,1,0]$, $[0,0,1]$일 때, 배치처리를 통해 손실함수값을 구하는 코드를 쓰시오.

(iv) 학습율(learning rate)을 0.2로 경사하강법을 적용해서 한걸음 내려간후의 첫번째 가중치 행렬을 출력하는 코드를 쓰시오.

8. (6번 문제 코드 계속) 다음은 첫번째 MNIST 테스트 데이터에 대해 신경망이 예측하는 확률분포가 학습을 진행해가며 어떻게 변하는지 학습회수가 50의 배수가 될때마다 막대그래프로 표현하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
sample=x_test[0]
iters_num = 1000
eval_interval=50
train_size = x_train.shape[0]
batch_size = 100
learning_rate = 0.1

plt.figure(figsize=(10,10))
for i in range(iters_num):
    batch_mask = np.random.[ ](train_size, batch_size)
    x_batch = x_train[batch_mask]
    t_batch = t_train[batch_mask]
    grad = network.gradient(x_batch, t_batch)
    for key in ('W1', 'b1', 'W2', 'b2'):
        network.params[key] [ ] learning_rate * grad[key]
    if (i [ ] eval_interval == 0) & ((i//eval_interval)<16):
        probability=softmax(network.predict(sample.reshape(1,784)))
        plt.[ ](4,4,int((i//eval_interval)+1))
        plt.[ ](range(len(probability[0])),probability[0])
        plt.ylim(0, 1.0)
    plt.show()
```



9. 학습이 끝난 신경망의 파라미터을 불러와서 혼동 행렬(confusion matrix)을 구하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

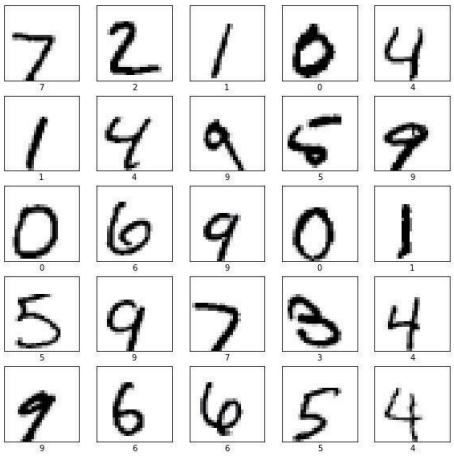
```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist(normalize=True)
    return x_test, t_test

def init_network():
    with open("neuralnet.pkl", 'rb') as f:
    network = pickle.[ ](f)
```

```
    return network

def predict(network, x):
    W1, W2 = network['W1'], network['W2']
    b1, b2 = network['b1'], network['b2']
    a1 = np.dot(x, W1) + b1
    z1 = relu(a1)
    a2 = np.dot(z1, W2) + b2
    return a2

x_test, t_test = get_data()
network = init_network()
confusion = np.[     ]((10,10), dtype=int)

for k in range([     ](x_test)):
    i=int(t_test[k])
    y = predict(network, x_test[k])
    j= np.[     ](y)
    confusion[i][j] [     ] 1
print(confusion)
```

10. (9번 문제 코드 계속)

(i) 다음 코드를 추가하면 다음과 같이 왼쪽으로 3 픽셀만큼, 아래로 3 픽셀만큼 이동시킨 이미지가 출력이 된다. 2개의 빈 칸을 순서대로 채우시오.

```
k=3
x=np.zeros((10000,1,28,28))
x[     ] = x_test[     ]
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x[i][0], cmap=plt.cm.binary)
    plt.xlabel(t_test[i])
plt.show()
```



(ii) 다음 코드를 추가해서 평행이동한 데이터의 정확도를 측정해 보면 25프로밖에 나오지 않는다. 그 이유를 설명하시오.

```
y = predict(network,x.reshape(10000,784))
p = np.argmax(y,axis=1)
accuracy = np.mean(p == t_test)
print(accuracy)
```