인공지능입문 (1반) / 중간고사 4. 23. 2019

1. 가중치 $w_1 = 0.3$, $w_2 = 0.5$와 임계값 $\theta = 0.6$을 가지는 퍼셉트론을 생각하자. 이는 AND 게이트, NAND 게이트, OR 게이트중 어느 것인지 답하고 이유를 설명하시오.

2. (i) 단층 퍼셉트론으로는 XOR 게이트를 구현할수 없음을 기하학적으로 설명하시오.
(ii) AND, NAND, OR 게이트를 조합한 다층 퍼셉트론으로 XOR 게이트를 구현하시오.

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

4. 라벨이 각각 1,2,3,4,5,6,7,8,9인 9개의 손글씨를 신경망에 넣었더니 모두
$$[1 - \sum_{k=1}^{9} e^{-k}, e^{-1}, e^{-2}, e^{-3}, e^{-4}, e^{-5}, e^{-6}, e^{-7}, e^{-8}, e^{-9}]$$
로 동일한 예측값이 나왔다. 교차 엔트로피 (cross entropy)값을 구하시오. (끝까지 계산)

5. 미분의 정의를 이용하여 등식
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$
를 보이시오.

6. 다음 계단 함수의 그래프를 그려주는 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
import numpy as np
import matplotlib.pylab as plt

def sigmoid(x):

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. (6번 문제 코드 계속) `x = np.arange(-5.0, 5.0, 2.0)`과 같이 수정했을 때 출력되는 그래프를 그리시오.

8. 다음은 pickle파일에 저장된 신경망의 정확도를 측정하는 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
import sys, os
import numpy as np
import pickle
from dataset.mnist import load_mnist
from common.functions import sigmoid, softmax

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.load(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()
accuracy_cnt = 0
for i in range(len(x)):
    y = predict(network, x[i])
    p= np.argmax(y)
    if p == t[i]:
        accuracy_cnt += 1
print("Accuracy:" + str(float(accuracy_cnt) / len(x)))
```

9. (8번 문제 코드 계속) 신경망에 엄격한 잣대를 적용하여 정답에 대해 80프로 이상의 확률로 예상할 때만 맞춘걸로 하고 싶다. 이러한 잣대 하에 정확도를 측정하도록 코드를 수정하시오.

10. (8번 문제 코드 계속) 다음 코드를 추가하면 신경망이 틀린 인덱스로 이루어진 리스트가 출력된다. 세개의 빈 칸을 채우시오.

```
error= [    ]
for i in range(len(x)):
    y = predict(network, x[i])
    p= np.argmax(y)
    if p [    ] t[i]:
        error.[    ](i)
```

인공지능입문 (2반) / 중간고사 4. 26. 2019

1. 적절한 가중치 w_1, w_2와 임계값 θ를 잡아 퍼셉트론을 이용하여 조건명제 $p \to q$를 구현하시오.

2. (i) 단층 퍼셉트론으로는 XOR 게이트를 구현할수 없음을 기하학적으로 설명하시오.
(ii) AND, NAND, OR 게이트를 조합한 다층 퍼셉트론으로 XOR 게이트를 구현하시오.

3. 이계 미분을 이용하여 시그모이드(sigmoid) 함수는 $x \leq 0$일 때 아래로 볼록, $x \geq 0$일 때 위로 볼록임을 보이시오.

4. 이층 신경망이 dictionary

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

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

6. 다음은 MNIST 훈련 데이터 앞 25장의 이미지를 출력하는 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
import sys, os
import numpy as np
from dataset.mnist import load_mnist
import matplotlib.pyplot as plt

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False, normalize=False)

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i][0], cmap=plt.cm.binary)
    plt.xlabel(t_train[i])
plt.show()
```

7. (6번 문제 코드 계속) 랜덤하게 25장을 뽑아 이미지를 출력하도록 코드를 수정하시오.

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
import sys, os
sys.path.append(os.pardir)
import numpy as np
import pickle
from dataset.mnist import load_mnist
from common.functions import sigmoid, softmax

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.load(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[i:i+batch_size]
    y_batch = predict(network, x_batch)
    p = np.argmax(y_batch, axis=1)
        accuracy_cnt += np.sum(p == t[i:i+batch_size])
print("Accuracy:" + str(float(accuracy_cnt) / len(x)))
```

9. (8번 문제 코드 계속) 다음 코드를 추가하면 다음과 같은 10×10 행렬이 출력이 된다. 신경망이 4를 9라고 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]]
```

10. 다음은 함수 $f_1(x) = 0.01x^2 + 0.1x$와 $x = 5$에서 접선의 그래프를 출력하는 코드이다. 두개의 빈 칸을 채우시오.

```
import numpy as np
import matplotlib.pylab as plt

def numerical_diff(f, x):
    h = 1e-4     return (f(x+h) - f(x-h)) / [    ]

def function_1(x):
    return 0.01*x**2 + 0.1*x

def tangent_line(f, x):
    d = numerical_diff(f, x)
    y = f(x) - d*x
    return [          ]

x = np.arange(0.0, 20.0, 0.1)
y = function_1(x)
plt.xlabel("x")
plt.ylabel("f(x)")

tf = tangent_line(function_1, 5)
y2 = tf(x)
plt.plot(x, y)
plt.plot(x, y2)
plt.show()
```

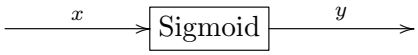
인공지능입문 (1반) / 기말고사 6. 11. 2019

1. 함수

$$f(x, y) = x^2 + y^2 + xy - 6x - 9y$$

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

2. Sigmoid 계층



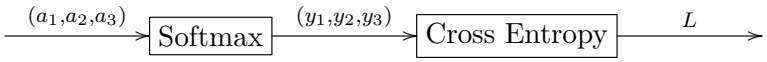
의 역전파는

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

로 주어짐을 미분법을 이용하여 보이시오.

3. 2번 문제를 계산그래프를 이용하여 유도하시오.

4. Softmax-with-Loss 계층



의 역전파는

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

로 주어짐을 다변수 미분법을 이용하여 보이시오. 단, (t_1, t_2, t_3)는 원-핫 인코딩된 라벨이다.

5. 4번 문제를 계산그래프를 이용하여 유도하시오.

6. 다음은 gradient를 구하는 코드이다. 삼변수 함수 $f(x, y, z)$에 대하여 다음 코드를 적용했을 때 gradient가 얻어지는 과정을 자세히 설명하시오.

```
def numerical_gradient(f, x):
    h = 1e-4
    grad = np.zeros_like(x)
    for idx in range(x.size):
        tmp_val = x[idx]
        x[idx] = float(tmp_val) + h
        fxh1 = f(x)
        x[idx] = tmp_val - h
        fxh2 = f(x)
        grad[idx] = (fxh1 - fxh2) / (2*h)
        x[idx] = tmp_val
    return grad
```

7. 다음 Affine 계층 코드이다. 빈 칸을 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(dout, [      ] )
        self.dW = np.dot([      ], dout)
        self.db = np.sum(dout, [      ] )

        return dx
```

8. 다음은 역전파를 이용한 이층 신경망 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
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():
            x = layer.forward(x)
        return x

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

    def accuracy(self, x, t):
        y = self.predict(x)
        y = np.argmax(y, axis=1)
        if t.ndim != 1:
            t = np.argmax(t, axis=1)
        accuracy = np.sum(y == t) / float(x.shape[0])
        return accuracy

    def gradient(self, x, t):
        self.loss(x, t)
        dout = 1
        dout = self.lastLayer.backward(dout)
        layers = list(self.layers.values())
        layers.reverse()
        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
```

9. (8번 문제 코드 계속) 다음은 기계 학습 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)

iters_num = 10000
train_size = x_train.shape[0]
batch_size = 100
learning_rate = 0.1

train_loss_list = []
train_acc_list = []
test_acc_list = []
iter_per_epoch = max(train_size / batch_size, 1)

for i in range(iters_num):
    batch_mask = np.random.choice(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]

loss = network.loss(x_batch, t_batch)
train_loss_list.append(loss)

if i % iter_per_epoch == 0:
    train_acc = network.accuracy(x_train, t_train)
    test_acc = network.accuracy(x_test, t_test)
    train_acc_list.append(train_acc)
    test_acc_list.append(test_acc)
    print(train_acc, test_acc)
```

10. (9번 문제 코드 계속) 다음 코드를 추가하면 다음과 같은 10×10 행렬이 출력이 된다. 신경망이 0을 6이라고 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_test)):
    i = np.argmax(t_test[k])
    y = network.predict(x_test[k].reshape(1,784))
    j = np.argmax(y)
    confusion[i][j] += 1
print(confusion)
```

```
[[ 961    0    1    1    0    7    6    1    1    2]
 [   0 1124    3    1    0    1    2    1    3    0]
 [   2    2 1007    4    5    0    4    5    3    0]
 [   0    0    6  979    0   12    0    7    4    2]
 [   0    0    3    0  957    1    3    3    2   13]
 [   2    1    0    8    0  863    9    2    5    2]
 [   5    3    3    1    2    4  936    0    4    0]
 [   0    9   10    5    1    2    0  993    0    8]
 [   4    3    2   10    4    7    9    5  924    6]
 [   2    6    1    8   11    1    1    6    1  972]]
```

인공지능입문 (2반) / 기말고사 6. 14. 2019

1. Sigmoid 계층

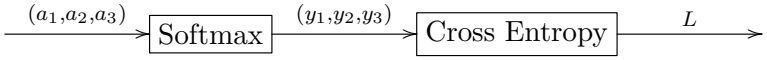
의 역전파는

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

로 주어짐을 미분법을 이용하여 보이시오.

2. 1번 문제를 계산그래프를 이용하여 유도하시오.

3. Softmax-with-Loss 계층



의 역전파는

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

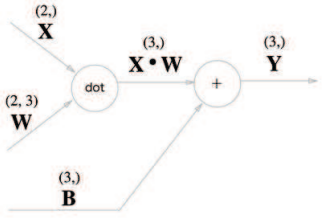
로 주어짐을 다변수 미분법을 이용하여 보이시오. 단, (t_1, t_2, t_3)는 원-핫 인코딩된 라벨이다.

4. 3번 문제를 계산그래프를 이용하여 유도하시오.

5. 입력 데이터 X, 가중치 행렬 W, 편향 벡터B가

$$X = (1,1), \qquad W = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \qquad B = (7,8,9)$$

와 같이 주어져 있다. 흘러들어온 미분 값이 $\frac{\partial L}{\partial Y} = (2,1,-1)$일 때, Affine변환 $Y = XW + B$의 계산그래프



를 이용하여 편미분

$$\frac{\partial L}{\partial X}, \quad \frac{\partial L}{\partial W}, \quad \frac{\partial L}{\partial B}$$

의 값을 각각 구하시오.

6. 다음 경사하강법을 따라 점의 이동경로를 기록하는 코드이다. 빈 칸을 채우시오.

```
def gradient_descent(f, init_x, lr=0.1, step_num=20):
    x = init_x
    x_history = [      ]

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

7. 다음 Relu 계층 코드이다. 빈 칸을 채우시오.

```
class Relu:
    def forward(self, x):
        self.mask = ([      ] )
        out = x.copy()
        out[self.mask] = [      ]
        return out

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

        return dx
```

8. 다음은 역전파를 이용한 이층 신경망 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
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():
            x = layer.forward(x)
        return x

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

```
    def accuracy(self, x, t):
        y = self.predict(x)
        y = np.argmax(y, axis=1)
        if t.ndim != 1:
            t = np.argmax(t, axis=1)
        accuracy = np.sum(y == t) / float(x.shape[0])
        return accuracy

    def gradient(self, x, t):
        self.loss(x, t)
        dout = 1
        dout = self.lastLayer.backward(dout)
        layers = list(self.layers.values())
        layers.reverse()
        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
```

9. (8번 문제 코드 계속) 다음은 기계 학습 코드이다. 밑줄 친 부분을 자세히 설명하시오.

```
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)

iters_num = 10000
train_size = x_train.shape[0]
batch_size = 100
learning_rate = 0.1

train_loss_list = []
train_acc_list = []
test_acc_list = []
iter_per_epoch = max(train_size / batch_size, 1)

for i in range(iters_num):
    batch_mask = np.random.choice(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]

    loss = network.loss(x_batch, t_batch)
    train_loss_list.append(loss)

    if i % iter_per_epoch == 0:
        train_acc = network.accuracy(x_train, t_train)
```

```
        test_acc = network.accuracy(x_test, t_test)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)
        print(train_acc, test_acc)
```

10. (i) (9번 문제 코드 계속) 다음 코드를 추가하여 학습한 weight와 bias를 저장하려고 한다. 빈 칸을 채우시오.

```
with open('neuralnet.pkl', 'wb') as f:
    pickle.[      ] (network.params, f)
```

(ii) (i)에서 저장한 pickle 파일을 불러와 첫번째 weight matrix를 보려고 한다. 빈 칸을 채우시오.

```
import pickle
with open('neuralnet.pkl', 'rb') as f:
    network = pickle.[      ] (f)
W1, W2 = network['W1'], network['W2']
b1, b2 = network['b1'], network['b2']
print(W1)
```

인공지능입문 (1반) / 중간고사 6. 24. 2020

1. 가중치 w_1, w_2가 1인 퍼셉트론을 생각하자. 어떤 범위의 임계값 θ에 대하여 이 퍼셉트론은 OR게이트를 구현하는가? 그리고, 어떤 범위의 임계값 θ에 대하여 AND 게이트를 구현하는가?

2. (i) 단층 퍼셉트론으로는 동치를 나타내는 조건 명제

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

를 구현할 수 없음을 기하학적으로 설명하시오.

(ii) NAND, OR 게이트를 조합한 다층 퍼셉트론으로 위 동치를 나타내는 조건명제를 구현하시오.

3. 활성화 함수가 sigmoid인 이층 신경망이 dictionary

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

로 주어져 있다. $[\log 4, \log 2]$를 입력했을때 출력되는 소프트맥스(softmax)값을 구하시오.

4. 라벨이 각각 $0,1,2,3,4,5,6,7,8,9$인 10개의 손글씨를 신경망에 넣었다. 입력한 손글씨의 라벨이 k일 경우 k일 확률을 신경망이

$$\frac{k+1}{k+2}$$

로 예측했다. 교차 엔트로피 (cross entropy)값을 구하시오. (끝까지 계산)

5. 확률 변수 X, Y의 확률분포가 다음과 같이 주어져 있다.

Y \ X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

엔트로피(entropy) $H(X)$, 조건부 엔트로피(conditional entropy) $H(X|Y)$, 상호 정보량(mutual information) $I(X;Y)$를 각각 구하시오.

6. 다음 코드를 실행했을때 출력될 값 3개를 순서대로 쓰시오.

```
x=np.arange(-5,5,2)
print(x)
print(x>0)
print(np.array(x>0,dtype=np.int))
```

7. 다음 코드를 실행하면 왼쪽과 같은 이미지가 출력이 된다. 음영이 뒤바뀐 오른쪽 이미지가 출력되도록 코드를 수정하시오.

```
def img_show(img):
    pil_img = Image.fromarray(np.uint8(img))
    pil_img.show()

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=False)
img = x_train[0]
img = img.reshape(28, 28)
img_show(img)
```

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 4개의 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), [    ]):
    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)))
```

9. (8번 문제 코드 계속) 신경망이 4를 보고 9라고 답한 데이터가 몇번째인지 모아놓은 리스트를 출력하도록 코드를 추가하시오.

10. 다음은 함수 $f_1(x) = 0.01x^2 + 0.1x$와 $x = 5$에서 접선의 그래프를 출력하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
def numerical_diff(f, x):
    h = 1e-4
    return (f(x+h) - f(x-h) / [    ]

def function_1(x):
    return 0.01*x[  ]2 + 0.1*x

def tangent_line(f, x):
    d = numerical_diff(f, x)
    y = f(x) - [    ]
    return [    ] t: d*t + y

x = np.[    ](0.0, 20.0, 0.1)
y = function_1(x)
plt.xlabel("x")
plt.ylabel("f(x)")

tf = tangent_line(function_1, 5)
y2 = tf(x)
plt.plot(x, y)
plt.plot(x, y2)
plt.show()
```

인공지능입문 (2반) / 중간고사 6. 25. 2020

1. 참을 거짓으로 거짓을 참으로 바꾸는 NOT 게이트를 생각하자. 적절한 가중치 w와 임계값 θ를 잡아 퍼셉트론을 이용하여 NOT 게이트를 구현하시오. (입력이 하나이므로 가중치도 한 개이다.)

2. (i) 단층 퍼셉트론으로는 XOR 게이트를 구현할수 없음을 기하학적으로 설명하시오.
(ii) AND, NAND, OR 게이트를 조합한 다층 퍼셉트론으로 XOR 게이트를 구현하시오.

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

4. 활성화 함수가 ReLU인 이층 신경망이 dictionary

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

로 주어져 있다. 데이터 $\mathbf{x} = [\log 2, \log 3, \log 4, \log 5, \log 6]$의 라벨이 $[1,0,0,0,0]$이라 하자. 교차 엔트로피(cross entropy) 값을 구하시오.

5. 확률 변수 X, Y의 확률분포가 다음과 같이 주어져 있다.

Y \ X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

엔트로피(entropy) $H(Y)$, 조건부 엔트로피(conditional entropy) $H(Y|X)$, 상호 정보량(mutual information) $I(X;Y)$를 각각 구하시오.

6. 다음 코드를 실행했을때 출력될 5개의 값을 순서대로 쓰시오.

```
x=np.arange(12).reshape(3,4)
print(x)
x=x.T
print(x)
print(x>5)
print(np.sum(x>5,axis=0))
print(np.sum(x>5,axis=1))
```

7. 꼭지점이 $(1,0)$, $(0,1)$, $(-1,0)$인 삼각형을 그리려 한다. 빈 칸을 채우시오.

```
plt.plot(                    )
plt.show()
```

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 4개의 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[          ])


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

9. (8번 문제 코드 계속) 신경망이 틀린 데이터가 몇번째인지 모아놓은 리스트를 출력하도록 코드를 추가하시오.

10. 다음 코드를 실행했을때 출력될 9개의 값을 순서대로 쓰시오.

```
def f(x):
    return np.sum(x**2)

x = np.array([1.0,1.0,1.0])
grad = np.zeros_like(x)
h = 0.1

for idx in range(x.size):
    tmp_val = x[idx]
    x[idx] = float(tmp_val)+h
    print(x)
    fxh1 = f(x)
    x[idx] = float(tmp_val) - h
    print(x)
    fxh2 = f(x)
    grad[idx] = (fxh1 - fxh2) / (2*h)
    print(grad)
    x[idx] = tmp_val
```

인공지능입문 (1반) / 기말고사 6. 24. 2020

1. 함수

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

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

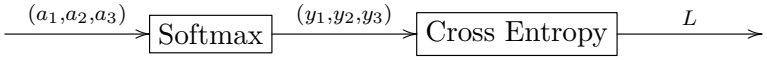
2. Sigmoid 계층

의 역전파는

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

로 주어짐을 미분법을 이용하여 보이시오.

3. Softmax-with-Loss 계층



의 역전파는

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

로 주어짐을 다변수 미분법을 이용하여 보이시오. 단, (t_1, t_2, t_3)는 원-핫 인코딩된 라벨이다.

4. 3번 문제를 계산그래프를 이용하여 유도하시오.

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

$$X = [1, -2, 3, -4], \qquad \frac{\partial L}{\partial Y} = [1, -1, -1, 1]$$

일 때, 출력값 Y와 밑으로 흘려보내는 미분 $\frac{\partial L}{\partial X}$을 각각 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
```

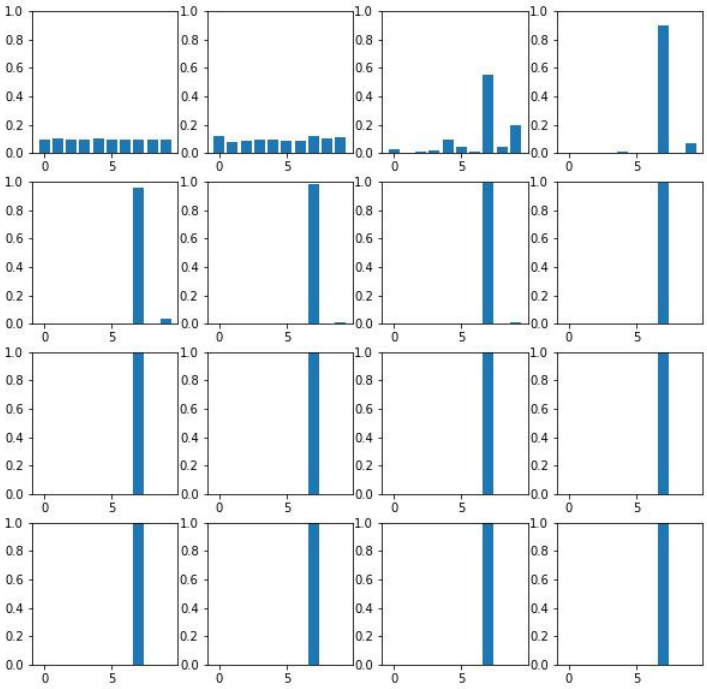
8. 다음은 역전파를 이용한 이층 신경망 코드이다. 4개의 빈 칸을 순서대로 채우시오.

```
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.[      ](x)
        return self.lastLayer.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'], grads['b1'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW, self.layers['Affine2'].db
        return grads
```

9. (8번 문제 코드 계속)
   (i) 입력층, 은닉층, 출력층의 뉴런의 개수가 모두 각각 3개인 얕은 신경망을 만들고 싶다. 본인 이름의 이니셜을 따서 인스턴스를 만드시오.
   (ii) 두 개의 데이터 $[1, 2, 3]$, $[4, 5, 6]$을 배치처리를 통해 점수(score)를 구하는 코드를 쓰시오.
   (iii) 각 라벨이 $[1, 0, 0]$, $[0, 0, 1]$일 때, 배치처리를 통해 손실함수값을 구하는 코드를 쓰시오.
   (iv) 학습율(learning rate)을 0.1로 경사하강법을 적용해서 한걸음 내려간후의 첫번째 가중치 행렬을 출력하는 코드를 쓰시오.

10. (9번 문제 코드 계속) 다음은 첫번째 MNIST 테스트 데이터에 대해 신경망이 예측하는 확률 분포가 학습을 진행해가며 어떻게 변하는지 학습회수가 50의 배수가 될때마다 막대그래프로 표현하는 코드이다. 5개의 빈 칸을 순서대로 채우시오.

```
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()
```



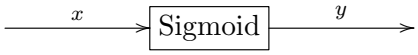
인공지능입문 (2반) / 기말고사 6. 25. 2020

1. 함수

$$f(x,y) = x^2 + 2y^2 + 3xy + 4x + 2y$$

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

2. Sigmoid 계층

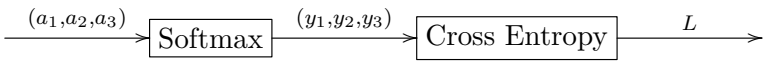


의 역전파는

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

로 주어짐을 미분법을 이용하여 보이시오.

3. Softmax-with-Loss 계층



의 역전파는

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

로 주어짐을 다변수 미분법을 이용하여 보이시오. 단, (t_1, t_2, t_3)는 원-핫 인코딩된 라벨이다.

4. 3번 문제를 계산그래프를 이용하여 유도하시오.

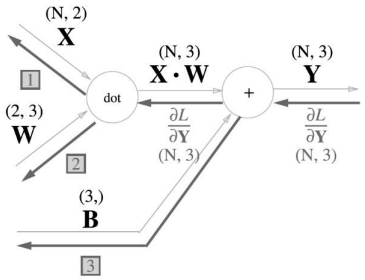
5. 입력 배치 데이터 X, 가중치 행렬 W, 편향 벡터B가

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \qquad W = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \qquad B = (7,8,9)$$

와 같이 주어져 있다. 흘러들어온 미분 값이

$$\frac{\partial L}{\partial Y} = \begin{pmatrix} 2 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

일 때, Affine변환의 계산그래프



를 이용하여 편미분

$$\frac{\partial L}{\partial X}, \quad \frac{\partial L}{\partial W}, \quad \frac{\partial L}{\partial B}$$

의 값을 각각 구하시오.

6. 다음 코드를 실행했을 때 출력될 그림을 그리시오.

```
def f(x):
    return x*(x-1.0)*(x-2.0)*(x-4.0)

def df(x):
    return (x-1.0)*(x-2.0)*(x-4.0)+x*(x-2.0)*(x-4.0)+x*(x-1.0)*(x-4.0)+x*(x-1.0)*(x-2.0)

init_position = [0.0,1.0,2.0,4.0]
lr=0.02
step_num=10

idx = 1
plt.figure(figsize=(10,10))
for init_x in init_position:
    x = init_x
    x_history = []

    for i in range(step_num):
        x_history.append(x)
        x -= lr * df(x)

    x = np.arange(-5.0, 5.0, 0.01)
    y = f(x)
    plt.subplot(2, 2, idx)
    plt.plot(np.array(x_history), f(np.array(x_history)), 'o', color="red")
    plt.plot(x,y)
    plt.ylim(-8, 8)
    plt.xlim(-8, 8)
    idx += 1
plt.show()
```

7. 다음은 Relu 계층 코드이다. 3개의 빈 칸을 순서대로 채우시오.

```
class Relu:
    def forward(self, x):
        self.mask = (      )
        out = x.copy()
        out[self.mask] =
        return out

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

8. 다음은 역전파를 이용한 이층 신경망 코드이다. 5개의 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 [     ].values():
            x = layer.forward(x)
        return x

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

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

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

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

10. (9번 문제 코드 계속) 혼동 행렬(confusion matrix)를 이용하여 불러온 신경망의 정확도를 구하도록 코드를 추가하시오.

인공지능입문 (1반) / 중간고사 4. 27. 2021

1. 가중치 $w_1 = 1, w_2 = 2$인 퍼셉트론을 생각하자. 어떤 범위의 임계값 θ에 대하여 이 퍼셉트론은 OR게이트를 표현하는가? 그리고, 어떤 범위의 임계값 θ에 대하여 AND 게이트를 표현하는가?

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

3. 라벨이 각각 $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$인 10개의 손글씨를 신경망에 넣었다. 입력한 손글씨의 라벨이 k일 경우 k일 확률을 신경망이

$$\frac{k+1}{k+2}$$

로 예측했다. 교차 엔트로피 (cross entropy)값을 구하시오.

4. 활성화 함수가 ReLU인 이층 신경망이 dictionary

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

로 주어져 있다. 3개의 데이터 $[\log 2, \log 3, \log 4, \log 5, \log 6]$와 $[\log 3, \log 4, \log 5, \log 6, \log 7]$와 $[\log 4, \log 5, \log 6, \log 7, \log 8]$의 라벨이 모두 $[1, 0, 0, 0, 0]$이라 하자. 배치처리로 계산하여 교차 엔트로피(cross entropy) 값을 구하시오.

5. 확률 변수 X, Y의 확률분포가 다음과 같이 주어져 있다.

Y \ X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

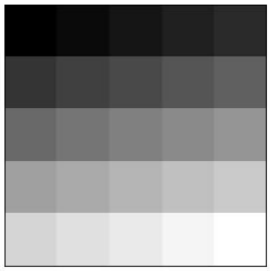
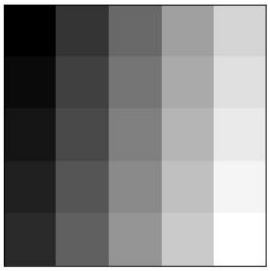
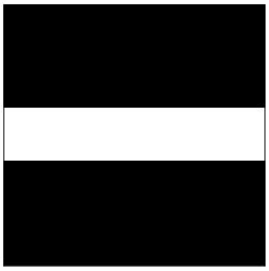
엔트로피(entropy) $H(Y)$, 조건부 엔트로피(conditional entropy) $H(Y|X)$, 상호 정보량(mutual information) $I(X;Y)$를 각각 구하시오.

6. 다음 코드를 실행했을때 출력될 5개의 값을 순서대로 쓰시오.

```
x=np.arange(12).reshape(3,4)
print(x)
x=x.T
print(x)
print(x>5)
```

```
print(np.sum(x>5,axis=0))
print(np.sum(x>5,axis=1))
```

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

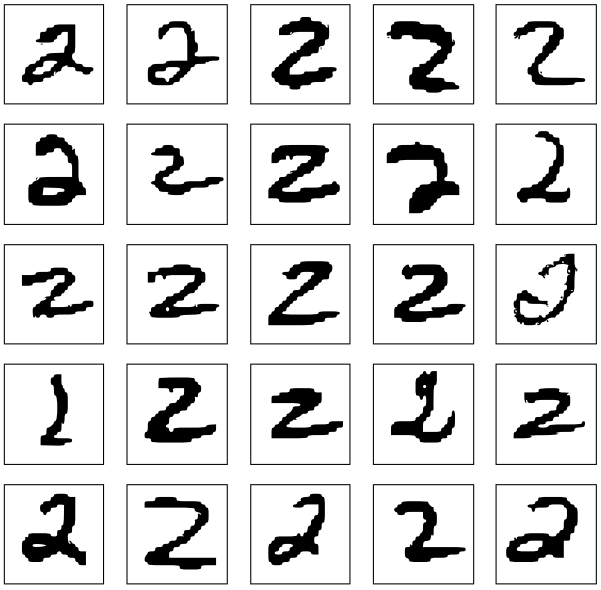
```
x=np.arange(25).[      ]
plt.imshow(x,cmap=plt.cm.gray)
plt.show()

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

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

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 4개의 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.      (y_batch, axis=1)
    accuracy_cnt += np.sum(p == t[      ])

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

9. 다음은 훈련용 MNIST데이터중 라벨이 2인 데이터 첫 25개를 다음과 같이 출력하는 코드이다. 3개의 빈 칸을 순서대로 채우시오.



```
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False, normalize=False)
N=2
N_index=[]
for k in range(len(x_train)):
    if t_train[k]==N:
        N_index.      
plt.figure(figsize=(10,10))
for i in range(25):
    plt.      (5,5,i+1)
    plt.xticks([])
    plt.yticks([])
```

```
    plt.imshow(x_train[      ][0], cmap=plt.cm.binary)
plt.show()
```

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

```
y = np.array([[np.e**(-1),0,0,0,0,0,0,0,0,1-np.e**(-1)],
    [0,np.e**(-2),0,0,0,0,0,0,0,1-np.e**(-2)],
    [0,0,np.e**(-3),0,0,0,0,0,0,1-np.e**(-3)]])
t = np.array([[1,0,0,0,0,0,0,0,0,0],[0,1,0,0,0,0,0,0,0,0],[0,0,1,0,0,0,0,0,0,0]])

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)
```

인공지능입문 (2반) / 중간고사 4. 22. 2021

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. $a_2 - a_1 = \log 2$, $a_3 - a_1 = \log 3$일 때, (a_1, a_2, a_3)의 소프트맥스(softmax)값을 구하시오.

3. 활성화 함수가 ReLU인 이층 신경망이 dictionary

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

로 주어져 있다. 3개의 데이터 $[1,2,3,4,5]$, $[2,3,4,5,6]$, $[3,4,5,6,7]$에 대하여 한번에 배치처리로 계산하여 소프트맥스(softmax)값을 구하시오.

4. 라벨이 각각 $1,2,3,4,5,6,7,8,9$인 9개의 손글씨를 신경망에 넣었다. 입력한 손글씨의 라벨이 k일 경우 k일 확률을 신경망이

$$\frac{1}{e^k}$$

로 예측했다. 교차 엔트로피 (cross entropy)값을 구하시오.

5. 확률 변수 X, Y의 확률분포가 다음과 같이 주어져 있다.

Y \ X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

엔트로피(entropy) $H(X)$, 조건부 엔트로피(conditional entropy) $H(X|Y)$, 상호 정보량(mutual information) $I(X;Y)$를 각각 구하시오.

6. 다음 코드를 실행했을 때 출력될 값 5개를 순서대로 쓰시오.

```
x=np.arange(9).reshape(3,3)
print(x)
x+=1
print(x)
x+=np.array([1,2,3])
print(x)
x+=np.array([[1,2,3]]).T
print(x)
print(np.sum(x>10,axis=1))
```

7. 다음 코드를 실행하면 왼쪽과 같은 이미지가 출력이 된다. 음영이 뒤바뀐 오른쪽 이미지가 출력되도록 코드를 수정하시오.

```
def img_show(img):
    pil_img = Image.fromarray(np.uint8(img))
    pil_img.show()

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=False)
img = x_train[0]
img = img.reshape(28, 28)
img_show(img)
```

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 4개의 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()
```

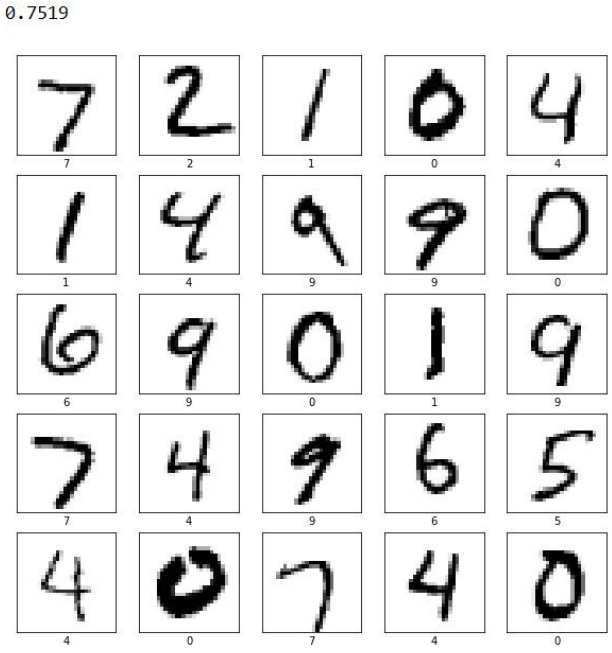
```
network = init_network()

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.argmax(y_batch, axis=1)
    accuracy_cnt += np.sum(p [     ] t[i:i+batch_size])

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

9. (i) 가중치 행렬은 표준정규분포를 따라 랜덤하게 생성하고 편향 벡터는 제로 백터로 잡아서 정확도를 측정하도록 8번 문제 코드에서 `init_network` 함수를 다음과 같이 수정하려 한다. 3개의 빈칸을 순서대로 채우시오.

```
def init_network():
    network = [     ]
    network['W1'] = [     ](784,50)
    network['b1'] = [     ](50)
            ⋮
    return network
```

(ii) 정확도가 대략 몇 프로 정도 될지 예상하고 그 이유를 설명하시오.

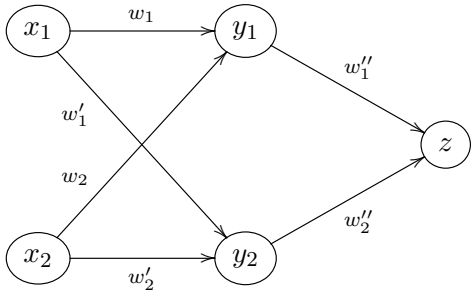
10. 8번 문제 코드에 추가하여 신경망이 90프로 이상의 확신을 가지고 맞춘 이미지가 몇프로 인지 구하고 첫 25장의 이미지를 5×5 테이블로 출력하려 한다. 5개의 빈 칸을 순서대로 채우시오.



```
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()
```

인공지능입문 (3반) / 중간고사 4. 23. 2021

1. (i) 단층 퍼셉트론으로는 XOR 게이트를 구현할수 없음을 기하학적으로 설명하시오.
(ii) 다음 이층 퍼셉트론에서 가중치와 임계값 (w_1, w_2, θ), (w_1', w_2', θ'), (w_1'', w_2'', θ'')를 찾아 XOR 게이트를 표현하시오.



2. 시그모이드 함수는 등식

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

을 만족함을 보이시오.

3. 활성화 함수가 ReLU인 이층 신경망이 dictionary

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

로 주어져 있다. 데이터 $\mathbf{x} = [\log 2, \log 3, \log 4, \log 5, \log 6]$의 라벨이 $[1,0,0,0,0]$이라 하자. 교차 엔트로피(cross entropy) 값을 구하시오.

4. 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.

5. 확률 변수 X, Y의 확률분포가 다음과 같이 주어져 있다.

Y \ X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

엔트로피(entropy) $H(Y)$, 조건부 엔트로피(conditional entropy) $H(Y|X)$, 상호 정보량(mutual information) $I(X;Y)$를 각각 구하시오.

6. 정삼각형을 그리려 한다. 빈 칸을 채우시오.

```
plt.plot(                    )
plt.show()
```

7. 다음 코드를 실행하면 왼쪽 사진이 출력된다. 오른쪽 사진이 출력되도록 코드를 수정하시오.

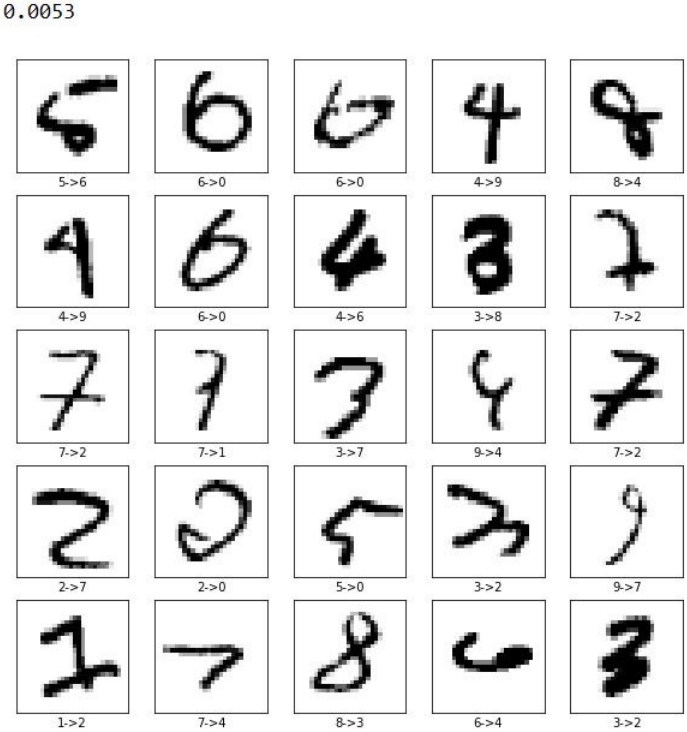
```
img = imread('../dataset/lena_gray.png')
plt.imshow(img, cmap=plt.cm.gray)
plt.axis('off')
plt.show()
```

8. 다음은 pickle파일에 저장된 신경망의 정확도를 배치처리를 통해 측정하는 코드이다. 4개의 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.load(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), batch_size):
    x_batch = x[      ]
    y_batch = predict(network, x_batch)
    p = np.argmax(y_batch, [      ])
    accuracy_cnt += np.sum(p == t[      ])

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

9. 8번 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()
```

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

```
y = np.array([[np.e**(-1),1-np.e**(-1),0,0,0,0,0,0,0,0],
    [1-np.e**(-3),np.e**(-3),0,0,0,0,0,0,0,0]])
t = np.array([[1,0,0,0,0,0,0,0,0,0],[0,1,0,0,0,0,0,0,0,0]])

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)
```

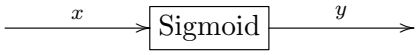
인공지능입문 (1반) / 기말고사 6. 15. 2021

1. 함수

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

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

2. Sigmoid 계층

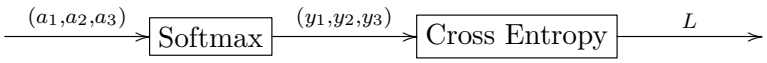


의 역전파는

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

로 주어짐을 미분법을 이용하여 보이시오.

3. Softmax-with-Loss 계층



의 역전파는

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

로 주어짐을 계산그래프를 이용하여 유도하시오.

4. 좌표평면상의 점을 원점을 중심으로 θ만큼 회전시키는 선형변환을 생각하자. 이 선형변환의 역전파는 흘러들어온 미분을 $-\theta$만큼 회전하여 밑으로 흘려보냄을 보이시오.

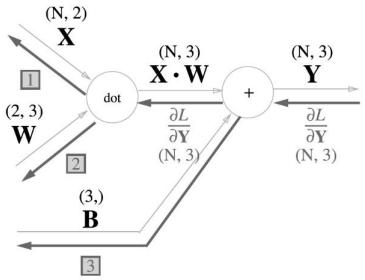
5. 입력 배치 데이터 X, 가중치 행렬 W, 편향 벡터B가

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad W = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 3 & 0 \end{pmatrix}, \qquad B = (1,2,3)$$

와 같이 주어져 있다. 흘러들어온 미분 값이

$$\frac{\partial L}{\partial Y} = \begin{pmatrix} 1 & 2 & 3 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \end{pmatrix}$$

일 때, Affine변환의 계산그래프



를 이용하여 미분

$$\frac{\partial L}{\partial X}, \quad \frac{\partial L}{\partial W}, \quad \frac{\partial L}{\partial B}$$

의 값을 각각 구하시오.

6. 다음 코드를 실행했을때 출력될 9개의 값을 순서대로 쓰시오.

```
def f(x):
    return np.sum(x**2)

x = np.array([1.0,2.0,3.0])
grad = np.zeros_like(x)
h = 0.1

for idx in range(x.size):
    tmp_val = x[idx]
    x[idx] = float(tmp_val) + h
    print(x)
    fxh1 = f(x)
    x[idx] = float(tmp_val) - h
    print(x)
    fxh2 = f(x)
    grad[idx] = (fxh1 - fxh2) / (2*h)
    print(grad)
    x[idx] = tmp_val
```

7. (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.

8. 다음은 Affine층 코드이다.

```
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(dout, self.W.T)
        self.dW = np.dot(self.x.T, dout)
        self.db = np.sum(dout, axis=0)
        return dx
```

(i) 본인 이름의 이니셜을 따서 인스턴스를 만드시오.
(ii) 5번 답을 출력하는 코드를 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.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
```

10. 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))
```

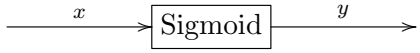
```
902.4785606506806
51.597695851281784
4.188285226425244
0.17557297587332535
0.0090041816780866
0.0005788259219041987
3.169636643830432e-05
1.512800199810902e-06
8.709281946181118e-08
5.9768406738885675e-09
3.538439536682618e-10
```

(i) 위와 같은 값이 나오는 수학적 이유를 설명하시오.
(ii) 신경망을 학습 시킬때 어떤 문제가 생기는지 쓰시오.
(iii) 해결 방법을 쓰시오.

인공지능입문 (2반) / 기말고사 6. 10. 2021

1. 함수
$$f(x,y) = 2x^2 + y^4 - xy - 6x + 3y$$
에 대하여 초기위치 $(0,0)$에서 출발하여 학습률(learning rate) 1/3로 경사하강법을 적용하려 한다. 두 걸음 갔을 때 위치를 구하시오.

2. Sigmoid 계층



의 역전파는
$$\frac{\partial L}{\partial y}\, y(1-y)$$
로 주어짐을 계산그래프를 이용하여 유도하시오.

3. 선형변환
$$S(x_1, x_2, \cdots, x_n) = (x_n, x_1, x_2, \cdots, x_{n-1})$$
을 생각하자. 이 선형변환으로 흘러들어오는 미분이 $(d_1, d_2, \cdots, d_n)$일 때, 역전파를 거쳐 밑으로 흘러가는 미분은
$$(d_2, d_3, \cdots, d_n, d_1)$$
임을 보이시오.

4. 활성화 함수가 ReLU인 이층 신경망이 dictionary
$$\{W_1 : \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},\ b_1 : [5,4,3,2,1],\ W_2 : \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix},\ b_2 : [-2+\log 2, -3, -1, 0, 0]\}$$
로 주어져 있다. 입력된 데이터는 $[1,2,3,4,5]$이고 라벨이 $[1,0,0,0,0]$일 때, 교차 엔트로피(cross entropy)값을 구하시오.

5. (4번 문제 계속) 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. 다음 코드를 실행했을 때 출력될 그림을 그리시오.
```
def f(x):
    return x*(x-1.0)*(x-2.0)*(x-4.0)

def df(x):
```

```
    return (x-1.0)*(x-2.0)*(x-4.0)+x*(x-2.0)*(x-4.0)+x*(x-1.0)*(x-4.0)+x*(x-1.0)*(x-2.0)

init_position = [0.0,1.0,2.0,4.0]
lr=0.02
step_num=10

idx = 1
plt.figure(figsize=(10,10))
for init_x in init_position:
    x = init_x
    x_history = []

    for i in range(step_num):
        x_history.append(x)
        x -= lr * df(x)

    x = np.arange(-5.0, 5.0, 0.01)
    y = f(x)
    plt.subplot(2, 2, idx)
    plt.plot(np.array(x_history), f(np.array(x_history)), 'o', color="red")
    plt.plot(x,y)
    plt.ylim(-8, 8)
    plt.xlim(-8, 8)
    idx += 1
plt.show()
```

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. 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)
```

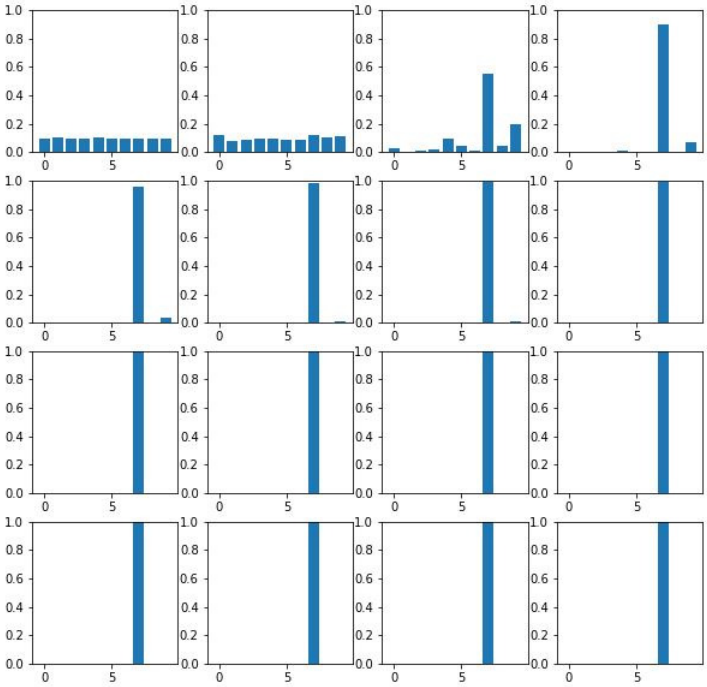
```
        self.params['b1'] = np.zeros(hidden_size)
        self.params['W2'] = init_std * np.random.randn(hidden_size, output_size)
        self.params['b2'] = np.zeros(output_size)
        self.layers = OrderedDict()
        self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])
        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'], grads['b1'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW, self.layers['Affine2'].db
        return grads
```

9. (8번 문제 코드 계속)
   (i) 입력층, 은닉층, 출력층의 뉴런의 개수가 각각 3개, 5개, 2개인 얕은 신경망을 만들고 싶다. 본인 이름의 이니셜을 따서 인스턴스를 만드시오.
   (ii) 두 개의 데이터 $[1, 2, 3]$, $[4, 5, 6]$을 배치처리를 통해 점수(score)를 구하는 코드를 쓰시오.
   (iii) 각 라벨이 $[1, 0]$, $[0, 1]$일 때, 배치처리를 통해 손실함수값을 구하는 코드를 쓰시오.
   (iv) 학습율(learning rate)을 0.1로 경사하강법을 적용해서 한걸음 내려간후의 첫번째 편향 벡터를 출력하는 코드를 쓰시오.

10. (9번 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.

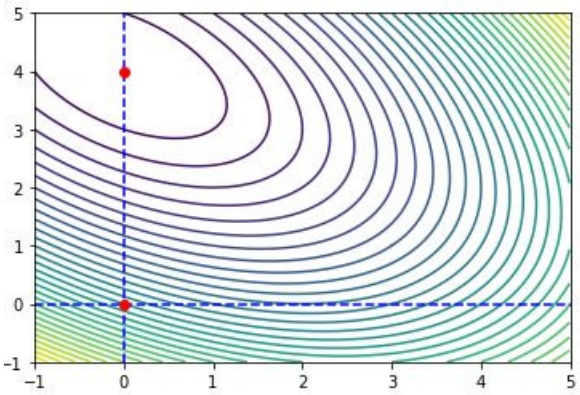
```
sample=x_test[0]
iters_num = 1000
eval_interval=50
train_size = x_train.shape[0]
```

```
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()
```



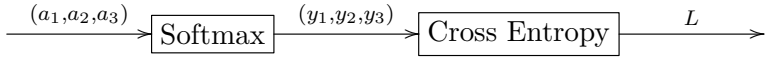
인공지능입문 (3반) / 기말고사 6. 11. 2021

1. 함수
$$f(x,y) = x^2 + y^2 + xy - 4x - 8y$$
를 생각하자.
(i) $\nabla f(x,y) = (0,0)$을 풀어 $(0,4)$가 유일한 극점임을 보이시오.
(ii) 헤세 판정법으로 $(0,4)$가 최소점임을 보이시오.
(iii) 다음은 함수 f의 등위선들이다. $(0,0)$부터 출발하여 경사하강법을 적용했을때 이동하는 괘적을 곡선으로 그리고 수학적인 이유를 설명하시오.



2. 함수
$$f(x,y) = 2x^2 + y^4 - xy - 6x + 3y$$
에 대하여 초기위치 $(0,0)$에서 출발하여 학습률(learning rate) 1/3로 경사하강법을 적용하려 한다. 두 걸음 갔을 때 위치를 구하시오.

3. Softmax-with-Loss 계층



의 역전파는
$$(y_1 - t_1, y_2 - t_2, y_3 - t_3)$$
로 주어짐을 다변수 미분법을 이용하여 보이시오. 단, (t_1,t_2,t_3)는 원-핫 인코딩된 라벨이다.

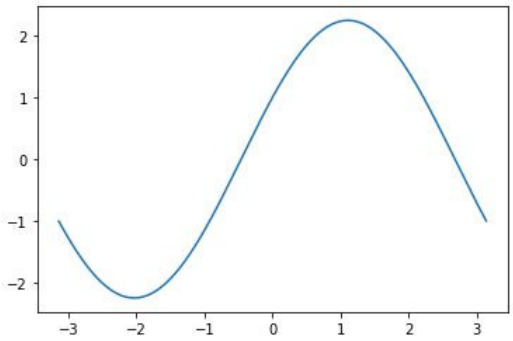
4. 활성화 함수가 sigmoid인 이층 신경망이 dictionary
$$\{W_1 : \begin{bmatrix} 1 & 2 & 3 \\ -4 & -5 & -6 \end{bmatrix},\ b_1 : [0,0,0],\ W_2 : \begin{bmatrix} 5\log 2 & 0 \\ 0 & 3\log 2 \\ 4\log 2 & 0 \end{bmatrix},\ b_2 : [0,0]\}$$
로 주어져 있다. 입력된 데이터는 $[\log 4, \log 2]$이고 라벨이 $[1,0]$일 때, 교차 엔트로피(cross entropy)값을 구하시오.

5. (4번 문제 계속) 미분
$$\frac{\partial L}{\partial W_1},\quad \frac{\partial L}{\partial b_1},\quad \frac{\partial L}{\partial W_2},\quad \frac{\partial L}{\partial b_2}$$

을 구하시오.

6. 다음 코드를 실행하면 다음과 같은 그래프가 나온다. 다음 그래프의 최대가 되는 지점과 최소가 되는 정확한 지점을 다변수 미분법을 사용하여 구하시오.

```
def f(x,y):
    return 2*(x**3) + x*(y**2) + 5*x**2 + y**2
def g(theta):
    h = 1e-4
    return (f(h*np.cos(theta),1+h*np.sin(theta))
        - f(-h*np.cos(theta),1-h*np.sin(theta))) / (2*h)
theta = np.arange(-np.pi, np.pi, 0.01)
y = g(theta)
plt.plot(theta, y)
```



7. 다음은 Relu 계층 코드이다. 3개의 빈 칸을 순서대로 채우시오.

```
class Relu:
    def forward(self, x):
        self.mask = (          )
        out = x.copy()
        out[self.mask] =       
        return out

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

8. 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 [     ].values():
            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'], grads['b1'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW, self.layers['Affine2'].db
        return grads
```

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

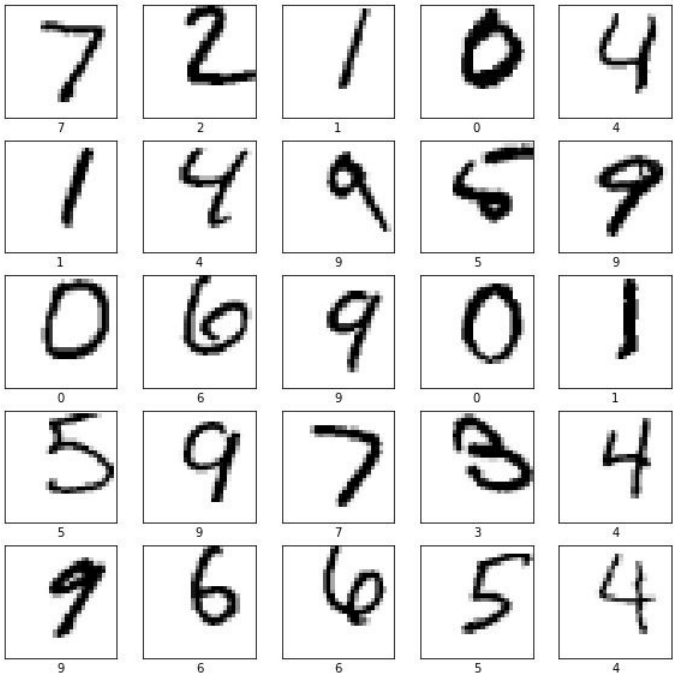
```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist(normalize=True)
    return x_test, t_test

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 오른쪽으로 3 픽셀만큼, 위로 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)
```