# 인공지능입문 (1반) / 중간고사 6. 24. 2020

1. 가중치 $w_1, w_2$가 1인 퍼셉트론을 생각하자. 어떤 범위의 임계값 $\theta$에 대하여 이 퍼셉트론은 OR게이트를 구현하는가? 그리고, 어떤 범위의 임계값 $\theta$에 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 $k$일 경우 $k$일 확률을 신경망이

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

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

5. 확률 변수 $X, Y$의 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 $w$와 임계값 $\theta$를 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 $X, Y$의 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.

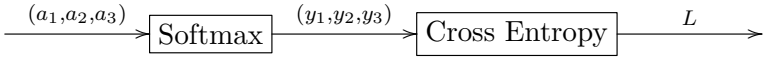
2. Sigmoid 계층


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 $X$와 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 $Y$와 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
```

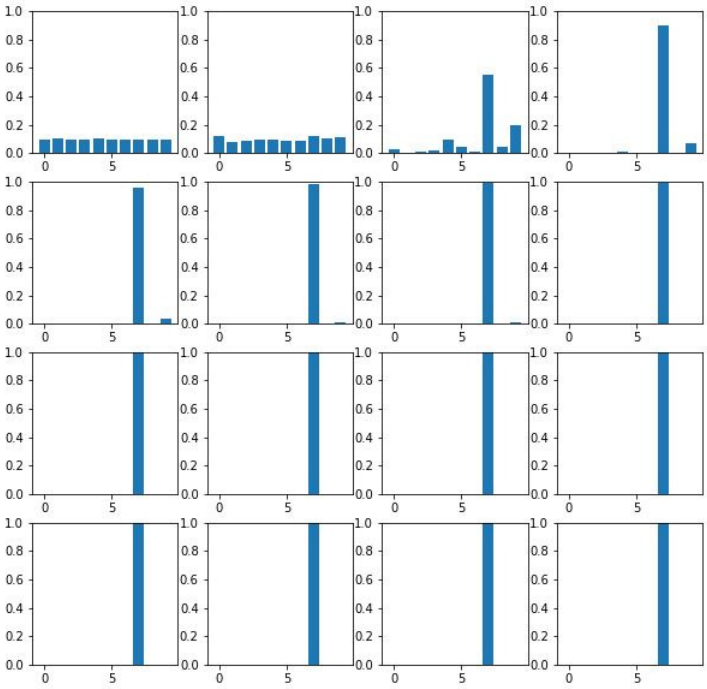
8. 다음은 역전파를 이용한 이층 신경망 코드이다. 4개의 빈 칸을 순서대로 채우시오.

```
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. (8번 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.

```
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()
```



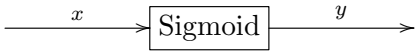
# 인공지능입문 (2반) / 기말고사 6. 25. 2020

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

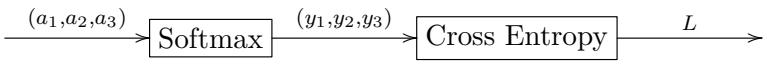



의 역전파는

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

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.

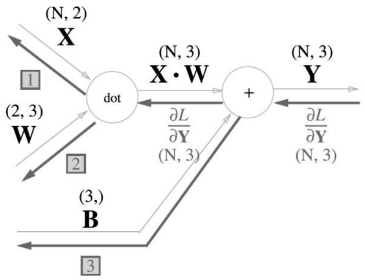
5. 입력 배치 데이터 $X$, 가중치 행렬 $W$, 편향 벡터$B$가

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

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.