전자공학과 2020142001 곽종근



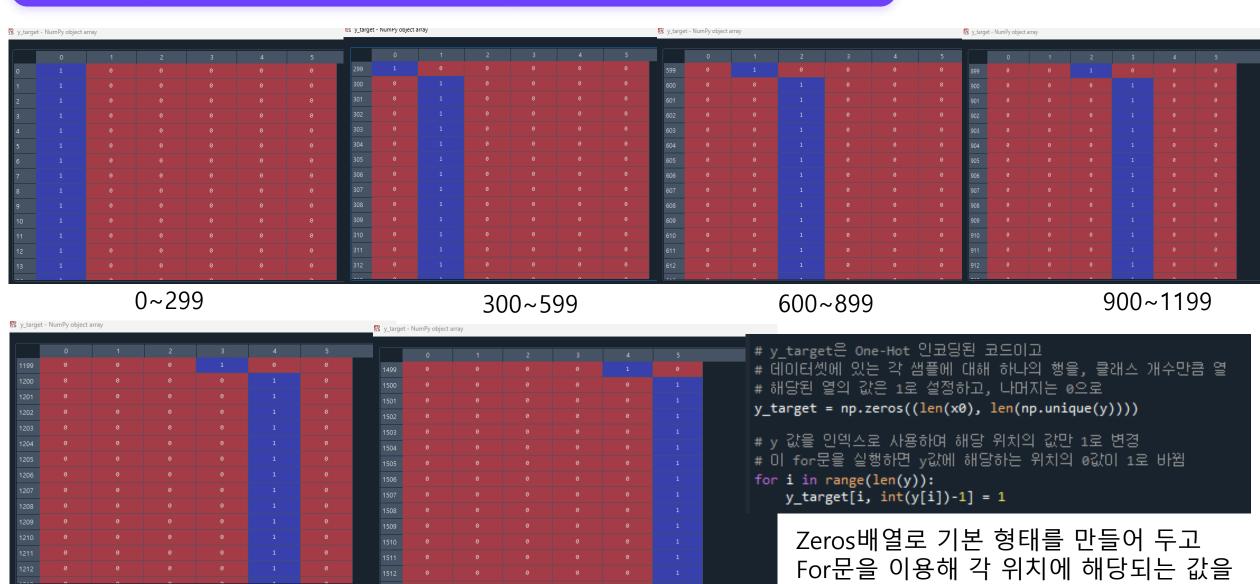
8주차 인공신경망 Two-Layer Neural Network Chap.4 실습

제출일: 2024.05.11.

## Chap.4 전체 코드

```
import numpy as np
                                                                       # 입력 속성 수 추출
import pandas as pd
                                                                      M = x with dummy.shape[1]
import matplotlib.pyplot as plt
                                                                       # 출력 클래스 수 추출
# 데이터 불러오기
                                                                       output size = y target.shape[1]
fold dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3확년 15
temp data = pd.read csv(fold dir)
                                                                       # hidden layer의 노드 수를 원하는 값으로 바꾸는 부분
temp_data = temp_data.to_numpy()
                                                                       hidden size = 10
# 데이터 분리
                                                                       # weight 초기화 (np.random.rand를 사용하며 표현)
x0 = temp_data[:, 0].reshape(-1,1)
x1 = temp_data[:, 1].reshape(-1,1) # temp_data 1열을 저장
                                                                       v = np.random.rand(hidden size,M)
x2 = temp_data[:, 2].reshape(-1,1) # temp_data 2열을 저장
                                                                       w = np.random.rand(output size, hidden size+1)
y = temp data[:, 3].reshape(-1,1) # temp data 3열을 y로 저장
                                                                       # bias 초기화 (모든 요소가 1로 설정)
# 시그모이드 함수를 선언
                                                                       bias input hidden = np.ones((1, hidden size))
def Sigmoid(x):
                                                                       bias hidden output = np.ones((1, output size))
   return 1 / (1 + np.exp(-x))
xtotal data= np.hstack((x0,x1,x2))
dummy data = np.ones((len(xtotal data), 1))
                                                                       A=v@x with dummy.T
x with dummy = np.hstack((xtotal data, dummy data))
                                                                       b=Sigmoid(A)
                                                                       b with dummy = np.vstack([b,np.ones([1,len(xtotal_data)])])
# y target은 One-Hot 인코딩된 코드이고
                                                                       B=w@b with dummy
# 데이터셋에 있는 각 샘플에 대해 하나의 행을, 클래스 개수만큼 열
                                                                      y hat = Sigmoid(B)
# 해당된 열의 값은 1로 설정하고, 나머지는 0으로
y_target = np.zeros((len(x0), len(np.unique(y))))
                                                                       # 각 열에서 최대값의 인덱스를 구합니다.
                                                                       max indices = np.argmax(y hat, axis=0)
# v 값을 인덱스로 사용하여 해당 위치의 값만 1로 변경
                                                                       y hat binary = np.zeros like(y hat)
# 이 for문을 실행하면 y값에 해당하는 위치의 0값이 1로 바뀜
                                                                       y hat binary[max indices, np.arange(y hat.shape[1])] = 1
for i in range(len(y)):
   y_target[i, int(y[i])-1] = 1
                                                                       # 정확도 계산
# 입력 속성 수 추출
                                                                       match count = np.sum(np.all(y hat binary == y target.T, axis=0))
M = x_with_dummy.shape[1]
                                                                       accuracy = match count / y target.shape[0]
```

## Chap.4 One-Hot Encoding

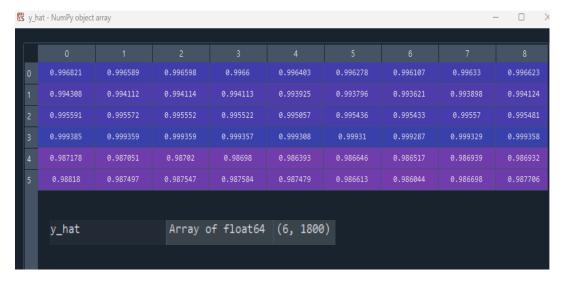


 For문을 이용해 각 위치에 해당되는 값을 1로 설정해주는 코드 각 값들이 1로 바뀐 것을 알 수 있다.

## Chap.4 Two-Layer Neural Network

М	int	1	4
output_size	int	1	6

M은 input\_size = 4, output\_size = 6이 된다 / 히든 레이어 = 10일때



최종적으로 y\_hat은 6,1800의 크기가 된다.

```
# 최종적으로 가중치를 곱한 값을 시그모이드 함수에 2번 적용하여
# 최종 y_hat출력
A=v@x_with_dummy.T
b=Sigmoid(A)
b_with_dummy = np.vstack([b,np.ones([1,len(xtotal_data)])])
B=w@b_with_dummy
y_hat = Sigmoid(B)
```

```
v Array of float64 (10, 4)
w Array of float64 (6, 11)
```

Hidden layer로 가는 weight = v로 (10,4)사이즈 최종 Output layer로 가는 weight = w로 (6,11) 사이즈

```
# 입력 속성 수 추출

M = x_with_dummy.shape[1]

# 출력 클래스 수 추출

output_size = y_target.shape[1]

# hidden layer의 노드 수를 원하는 값으로 바꾸는 부분

hidden_size = 10

# weight 초기화 (np.random.rand를 사용하여 표현)
v = np.random.rand(hidden_size,M)
w = np.random.rand(output_size,hidden_size+1)
```

데이터에서 입출력 수를 구하고 그 크기만큼 weight를 그 크기만큼 초기화해주는 코드 여기서 히든레이어 사이즈는 10으로 설정

여기서 y\_hat은 가중치에 적용된 x를 행렬연산 하여서 시 그모이드에 적용시키는 과정을 2번 해준다.

## Chap.4 Accuracy 함수 구현

```
# 각 열에서 최대값의 인덱스를 구한다.

max_indices = np.argmax(y_hat, axis=0)
# y_hat의 크기만큼 제로배열을 만들어준다.

y_hat_binary = np.zeros_like(y_hat)
# 위에서 만들어준 제로배열에 대해 위에서 구한 최대값 인덱스 부분을 1로 변환.

y_hat_binary[max_indices, np.arange(y_hat.shape[1])] = 1

# 정확도 계산, np.all로 전체가 다똑같은 개수를 구해서 이 값을 전체 개수로 나눠줌

match_count = np.sum(np.all(y_hat_binary == y_target.T, axis=0))
accuracy = match_count / y_target.shape[0]
```

각 열에서 최대값의 위치를 구해주고 제로스 배열에 위에서 구한 위치의 값을 1로 바꿔주고 이 행렬의 각 6개의 열과 완전히 일치하는 개수를 세서 전체 개수와 나눠준다.

y_hat_binary	Array	of	float64	(6, 1800)
max_indices	Array	of	int64	(1800,)

최대값을 1로 바꾼 배열의 크기는 (6,1800) 최대값을 구한 것의 크기는 1800개가 나온다



accuracy	float64	1	0.1666666666666666
----------	---------	---	--------------------

최종 정확도는 1/6에 해당되는 값이 나온다.