8주차 실습과제 코드

2020142001 곽종근
제출일: 2024.05.12.

실습과제 Chap.4 - 1주차

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 데이터 불러오기
fold_dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년
1학기\\머신러닝실습\\Machine-Learning\\NN_data.csv"
temp_data = pd.read_csv(fold_dir)
temp_data = temp_data.to_numpy()

# 데이터 분리
x0 = temp_data[:, 0].reshape(-1,1)  # temp_data 0열을 저장
x1 = temp_data[:, 1].reshape(-1,1)  # temp_data 1열을 저장
x2 = temp_data[:, 2].reshape(-1,1)  # temp_data 2열을 저장
y = temp_data[:, 3].reshape(-1,1)   # temp_data 3열을 y로 저장

# 시그모이드 함수를 선언
def Sigmoid(x):
    return 1 / (1 + np.exp(-x))

xtotal_data= np.hstack((x0,x1,x2))
dummy_data = np.ones((len(xtotal_data), 1))
x_with_dummy = np.hstack((xtotal_data, dummy_data))

# y_target은 One-Hot 인코딩된 코드이고
# 데이터셋에 있는 각 샘플에 대해 하나의 행을, 클래스 개수만큼 열
# 해당된 열의 값은 1로 설정하고, 나머지는 0으로
y_target = np.zeros((len(x0), len(np.unique(y))))

# y 값을 인덱스로 사용하여 해당 위치의 값만 1로 변경
# 이 for문을 실행하면 y값에 해당하는 위치의 0값이 1로 바뀜
for i in range(len(y)):
    y_target[i, int(y[i])-1] = 1

# 입력 속성 수 추출
# .shape[1]은 y_target의 열의 개수를 구해주는 것으로 여기서 속성 수를 나타냄
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)


# bias 초기화 (모든 요소가 1로 설정)
bias_input_hidden = np.ones((1, hidden_size))
bias_hidden_output = np.ones((1, output_size))

# 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)

#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.
# 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]
```