전자공학과 2020142001 곽종근

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

제출일: 2024.05.11.

Chap.4 전체 코드

```python
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 1
temp_data = pd.read_csv(fold_dir)
temp_data = temp_data.to_numpy()

# 데이터 분리
x0 = temp_data[:, 0].reshape(-1,1)
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
M = x_with_dummy.shape[1]
```

```python
# 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)


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


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 인덱스를 구합니다.
max_indices = np.argmax(y_hat, axis=0)
y_hat_binary = np.zeros_like(y_hat)
y_hat_binary[max_indices, np.arange(y_hat.shape[1])] = 1


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

Chap.4 One-Hot Encoding

y_target - NumPy object array

	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	1	0	0	0	0	0
4	1	0	0	0	0	0
5	1	0	0	0	0	0
6	1	0	0	0	0	0
7	1	0	0	0	0	0
8	1	0	0	0	0	0
9	1	0	0	0	0	0
10	1	0	0	0	0	0
11	1	0	0	0	0	0
12	1	0	0	0	0	0
13	1	0	0	0	0	0

0~299

y_target - NumPy object array

	0	1	2	3	4	5
299	1	0	0	0	0	0
300	0	1	0	0	0	0
301	0	1	0	0	0	0
302	0	1	0	0	0	0
303	0	1	0	0	0	0
304	0	1	0	0	0	0
305	0	1	0	0	0	0
306	0	1	0	0	0	0
307	0	1	0	0	0	0
308	0	1	0	0	0	0
309	0	1	0	0	0	0
310	0	1	0	0	0	0
311	0	1	0	0	0	0
312	0	1	0	0	0	0

300~599

y_target - NumPy object array

	0	1	2	3	4	5
599	0	1	0	0	0	0
600	0	0	1	0	0	0
601	0	0	1	0	0	0
602	0	0	1	0	0	0
603	0	0	1	0	0	0
604	0	0	1	0	0	0
605	0	0	1	0	0	0
606	0	0	1	0	0	0
607	0	0	1	0	0	0
608	0	0	1	0	0	0
609	0	0	1	0	0	0
610	0	0	1	0	0	0
611	0	0	1	0	0	0
612	0	0	1	0	0	0

600~899

y_target - NumPy object array

	0	1	2	3	4	5
899	0	0	1	0	0	0
900	0	0	0	1	0	0
901	0	0	0	1	0	0
902	0	0	0	1	0	0
903	0	0	0	1	0	0
904	0	0	0	1	0	0
905	0	0	0	1	0	0
906	0	0	0	1	0	0
907	0	0	0	1	0	0
908	0	0	0	1	0	0
909	0	0	0	1	0	0
910	0	0	0	1	0	0
911	0	0	0	1	0	0
912	0	0	0	1	0	0

900~1199

y_target - NumPy object array

	0	1	2	3	4	5
1199	0	0	0	1	0	0
1200	0	0	0	0	1	0
1201	0	0	0	0	1	0
1202	0	0	0	0	1	0
1203	0	0	0	0	1	0
1204	0	0	0	0	1	0
1205	0	0	0	0	1	0
1206	0	0	0	0	1	0
1207	0	0	0	0	1	0
1208	0	0	0	0	1	0
1209	0	0	0	0	1	0
1210	0	0	0	0	1	0
1211	0	0	0	0	1	0
1212	0	0	0	0	1	0

1200~1499

y_target - NumPy object array

	0	1	2	3	4	5
1499	0	0	0	0	1	0
1500	0	0	0	0	0	1
1501	0	0	0	0	0	1
1502	0	0	0	0	0	1
1503	0	0	0	0	0	1
1504	0	0	0	0	0	1
1505	0	0	0	0	0	1
1506	0	0	0	0	0	1
1507	0	0	0	0	0	1
1508	0	0	0	0	0	1
1509	0	0	0	0	0	1
1510	0	0	0	0	0	1
1511	0	0	0	0	0	1
1512	0	0	0	0	0	1

1500~1799

```
# 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
```

Zeros배열로 기본 형태를 만들어 두고
For문을 이용해 각 위치에 해당되는 값을
1로 설정해주는 코드
각 값들이 1로 바뀐 것을 알 수 있다.

Chap.4 Two-Layer Neural Network

M	int	1	4
output_size	int	1	6

M은 input_size = 4, output_size = 6이 된다 / 히든 레이어 = 10일때

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

y_hat - NumPy object array

	0	1	2	3	4	5	6	7	8
0	0.996821	0.996589	0.996598	0.9966	0.996403	0.996278	0.996107	0.99633	0.996623
1	0.994308	0.994112	0.994114	0.994113	0.993925	0.993796	0.993621	0.993898	0.994124
2	0.995591	0.995572	0.995552	0.995522	0.995057	0.995436	0.995433	0.99557	0.995481
3	0.999385	0.999359	0.999359	0.999357	0.999308	0.99931	0.999287	0.999329	0.999358
4	0.987178	0.987051	0.98702	0.98698	0.986393	0.986646	0.986517	0.986939	0.986932
5	0.98818	0.987497	0.987547	0.987584	0.987479	0.986613	0.986044	0.986698	0.987706

y_hat	Array of float64	(6, 1800)

최종적으로 y_hat은 6,1800의 크기가 된다.

```
    # 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

```
# 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)
```

여기서 y_hat은 가중치에 적용된 x를 행렬연산 하여서 시그모이드에 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

y_hat_binary - NumPy object array

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0

이처럼 해당 위치가 1로 바뀜

accuracy	float64	1	0.16666666666666666

최종 정확도는 1/6에 해당되는 값이 나온다.