Homework 1 : Logistic Regression

2021050300
김재민

1. 모델 및 학습 코드

- 다음은 Logistic Regression 모델과 이에 대한 학습 함수들의 코드입니다.
- 이러한 코드는 AND, OR, XOR Gate에 대해 각각 모델 학습 및 테스트를 수행하며, 학습률을 조정함으로써 이에 대한 결과를 확인할 수 있습니다.

1-1. 모델 정의 코드

```
import random
import math
import numpy as np

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

class LogisticRegressionModel:
    def __init__(self):
        self.w = [random.random(), random.random()]
        self.b = random.random()
    def sigmoid(self, z):
        return 1 / (1 + math.exp(-z))
    def predict(self, x):
        z = np.dot(self.w, x) +self.b
        a =self.sigmoid(z)
        return a

model = LogisticRegressionModel()

def test(model):
    test_cases = [(0,0), (0,1), (1,0), (1,1)]
    for case in test_cases:
        print(f"model.predict{case} = {model.predict(case)}")
```

1-2. 학습 함수 코드 (3가지의 Learning Rate)

1-2-1. Learning Rate = 0.01

```
def train1(X, Y, model, lr =0.01):
    dw0, dw1, db =0.0, 0.0, 0.0
    m =len(X)
    cost =0.0

    for x, y in zip(X, Y):
        a = model.predict(x)
        cost +=-y * math.log(a) - (1 - y) * math.log(1 - a)
        dw0 += (a - y) * x[0]
        dw1 += (a - y) * x[1]
        db += (a - y)
```

```
    cost /= m
    model.w[0] -= lr * dw0 / m
    model.w[1] -= lr * dw1 / m
    model.b -= lr * db / m
    return cost
```

1-2-2. Learning Rate = 0.1

```
def train2(X, Y, model, lr =0.1):
    dw0, dw1, db =0.0, 0.0, 0.0
    m =len(X)
    cost =0.0

    for x, y in zip(X, Y):
        a = model.predict(x)
        cost +=-y * math.log(a) - (1 - y) * math.log(1 - a)
        dw0 += (a - y) * x[0]
        dw1 += (a - y) * x[1]
        db += (a - y)

    cost /= m
    model.w[0] -= lr * dw0 / m
    model.w[1] -= lr * dw1 / m
    model.b -= lr * db / m
    return cost
```

1-2-2. Learning Rate = 0.2

```
def train3(X, Y, model, lr =0.2):
    dw0, dw1, db =0.0, 0.0, 0.0
    m =len(X)
    cost =0.0

    for x, y in zip(X, Y):
        a = model.predict(x)
        cost +=-y * math.log(a) - (1 - y) * math.log(1 - a)
        dw0 += (a - y) * x[0]
        dw1 += (a - y) * x[1]
        db += (a - y)

    cost /= m
    model.w[0] -= lr * dw0 / m
    model.w[1] -= lr * dw1 / m
    model.b -= lr * db / m
    return cost
```

1-3. 각 논리 연산자 코드

1-3-1. AND

```
# AND gate
print('Here is AND gate')
X_and = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y_and = np.array([0, 0, 0, 1])
and_model = LogisticRegressionModel()

print('Learning Rate : 0.01')
for epoch in range(10000) :
  cost = train1(X_and, Y_and, and_model, 0.1)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(and_model)

print('Learning Rate : 0.1')
for epoch in range(10000) :
  cost = train2(X_and, Y_and, and_model, 0.2)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(and_model)

print('Learning Rate : 0.2')
for epoch in range(10000) :
  cost = train3(X_and, Y_and, and_model, 0.5)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(and_model)
```

1-3-2. OR

```
# OR gate
print('Here is OR gate')
X_or = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y_or = np.array([0, 1, 1, 1])
or_model = LogisticRegressionModel()

print('Learning Rate : 0.01')
for epoch in range(10000) :
  cost = train1(X_or, Y_or, or_model, 0.1)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(or_model)

print('Learning Rate : 0.1')
for epoch in range(10000) :
  cost = train2(X_or, Y_or, or_model, 0.2)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(or_model)

print('Learning Rate : 0.2')
for epoch in range(10000) :
  cost = train3(X_or, Y_or, or_model, 0.5)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(or_model)
```

1-3-3. XOR.

```
# XOR gate
print('Here is XOR gate')
X_xor = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y_xor = np.array([0, 1, 1, 0])
xor_model = LogisticRegressionModel()

print('Learning Rate : 0.01')
for epoch in range(10000) :
  cost = train1(X_xor, Y_xor, xor_model, 0.1)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(xor_model)

print('Learning Rate : 0.1')
for epoch in range(10000) :
  cost = train2(X_or, Y_or, or_model, 0.2)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(xor_model)

print('Learning Rate : 0.2')
for epoch in range(10000) :
  cost = train3(X_xor, Y_xor, xor_model, 0.5)
  if epoch % 1000 ==0:
    print(epoch, cost)
test(xor_model)
```

2. 각 논리 연산자의 결과

2-1. AND

입력값	예측 결과 (lr = 0.01)	예측 결과 (lr = 0.1)	예측 결과 (lr = 0.2)
(0,0)	1.2337e - 05	4.3946e - 07	2.2543e - 08
(0,1)	0.0201	0.0067	0.0025
(1,0)	0.0201	0.0067	0.0025
(1,1)	0.9717	0.9905	0.9964

- AND는 예측결과가 lr이 0.01 ~ 0.2으로 변화하면서, (0,0), (0,1), (1,0)은 0에 수렴하고 (1,1)은 1에 수렴하는 변화를 통해 모델의 학습이 성공하였음을 알 수 있다.

2-2. OR

입력값	예측 결과 (lr = 0.01)	예측 결과 (lr = 0.1)	예측 결과 (lr = 0.2)
(0,0)	0.0204	0.0067	0.0025
(0,1)	0.9918	0.9973	0.9989
(1,0)	0.9918	0.9973	0.9989
(1,1)	0.9999	0.9999	0.9999

- OR은 예측결과가 lr이 0.01 ~ 0.2으로 변화하면서, (0,0)은 0에 수렴하고 (0,1), (1,0),(1,1)은 1에 수렴하는 변화를 통해 모델의 학습이 성공하였음을 알 수 있다.

2-3. XOR

입력값	예측 결과 (lr = 0.01)	예측 결과 (lr = 0.1)	예측 결과 (lr = 0.2)
(0,0)	0.5	0.5	0.5
(0,1)	0.5	0.5	0.5
(1,0)	0.5	0.5	0.5
(1,1)	0.5	0.5	0.5

- XOR은 예측결과가 lr이 0.01 ~ 0.2으로 변화하지만, 결과값은 변하지 않음을 통해 모델의 학습이 성공하지 못함을 알 수 있다.

3. 결론

- AND와 OR Gate는 Logistic Regression을 통해 성공적으로 학습되었다.
- XOR은 Logistic Regression으로 해결할 수 없는 비선형적 문제가 발생한다.
- 이로 인해 Multi - Layer Perceptron이 필요하다.