인공지능 HW#1

logistic_regression_model

소프트웨어전공 2018044993 임소윤

Model

```
class logistic_regression_model():
    def __init__(self):
        self.w=np.random.normal(size=2)
        self.b=np.random.normal()

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

    def predict(self,x):
        z=np.inner(self.w,x)+self.b
        a=self.sigmoid(z)
        return a
```

```
model = logistic_regression_model()
```

Training

```
def train(X, Y, model, lr):
    dw0=0.0
    dw1=0.0
    db=0.0
    m=len(X)
    cost=0.0
    for x,y in zip(X,Y):
        a=model.predict(x)
        if y==1:
            cost -= log(a)
        else:
            cost -= log(1-a)

        this=np.append(x,1)
        [dw0,dw1,db] = [dw0,dw1,db] + ((a-y)*this)

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

    return cost
```

```
def loss():
    losses = []
    for i in range(4):
        loss = -Y[i]*np.log(model.predict(X[i]))-(1-Y[i])*np.log(1-model.predict(X[i]))
        losses.append(loss)
    return losses
```

```
for epoch in range(10000):
    cost = train(X,Y,model,0.1)
    if epoch%100==0:
        print(epoch, cost)
```

1. AND

```
X=np.array([[0,0],[0,1],[1,0],[1,1]])
Y=np.array([[0],[0],[0],[1]])
```

i. lr=0.01

```
for epoch in range(10000):
    cost = train(X,Y,model,0.01)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.017040694182791853
100 0.017024010770340072
200 0.017007359637806505
300 0.01699074069226062
400 0.016974153841126317
500 0.016957598992180443
600 0.01694107605355093 4
700 0.016924584933715237
800 0.01690812554149873
900 0.01689169778607285
1000 0.016875301576953845
1100 0.016858936824000643
1200 0.01684260343741365
1300 0.016826301327732968
1400 0.01681003040583683
1500 0.01679379058294002
1600 0.01677758177059223
1700 0.01676140388067653
1800 0.016745256825407884
1900 0.01672914051733146
2000 0.016713054869321068
2100 0.0166969997945779
2200 0.01668097520662866
9200 0.01563018791542236
9300 0.015616125496957723
9400 0.01560208810545255 3
9500 0.01558807567459362
9600 0.015574088138300798
9700 0.01556012543072567 1
9800 0.01554618748625096 8
9900 0.015532274239489184
```

```
model.predict((0,0))
```

```
8.77622300234328e-06
```

```
model.predict((0,1))
```

```
0.018058856749258424
```

```
model.predict((1,0))
```

```
0.018058845487079374
```

```
model.predict((1,1))
```

```
0.9747083687687714
```

ii. lr=0.1

```
for epoch in range(10000):
    cost = train(X,Y,model,0.1)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.3507343154158976
100 0.2867564219804135
200 0.24802625095489705
300 0.21922061907452794
400 0.1967010122023596
500 0.1784879050924284
600 0.16339234267116315
700 0.1506462046404768
800 0.13972506382337366
900 0.13025536858869685
1000 0.12196205616562356
1100 0.11463712928223566
1200 0.10811984575048164
1300 0.10228370785416835
1400 0.0970276223817575
1500 0.09226972055806759
1600 0.08794293146772564
1700 0.0839917446470585
1800 0.08036979910533867
1900 0.07703805905598488
2000 0.07396341404514907
2100 0.07111759120013444
2200 0.06847630044217175
9300 0.018294766788274394
9400 0.0181045613825833
9500 0.017918226102169407
9600 0.01773564486542342
9700 0.01755670616314149
9800 0.01738130283648229
9900 0.01720933186769795
```

```
model.predict((0,0))
```

```
1.165395621433059e-05
```

```
model.predict((0,1))
```

```
0.019808758467623207
```

```
model.predict((1,0))
```

```
0.019808738860323433
```

```
model.predict((1,1))
```

```
0.9722561515478174
```

iii. **lr=1.0**

```
for epoch in range(10000):
    cost = train(X,Y,model,1.0)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.015518385625281753
100 0.014243166127203217
200 0.013160122059363108
300 0.01222903986631095
400 0.011420162274306672
500 0.010711004188684894
600 0.010084263515104658
700 0.00952641198401204
800 0.009026721350067197
900 0.008576575902042947
1000 0.008168977772309623
1100 0.007798184851509715
1200 0.0074594416560261525
1300 0.00714877647987925
1400 0.00686284655522605
1500 0.006598818481469403
1600 0.006354274902504857
1700 0.006127140953318087
1800 0.005915625761091782
1900 0.00571817552776168
2000 0.00553343560676764
2100 0.005360219626395339 5
2200 0.005197484179285677
9300 0.0016433605048327186
9400 0.0016276605599412884
9500 0.0016122574407423083
9600 0.0015971428141406353
9700 0.0015823086559140927
9800 0.0015677472365403426
9900 0.0015534511077961272
```

```
model.predict((0,0))
```

```
8.341285294869824e-09
```

```
model.predict((0,1))
```

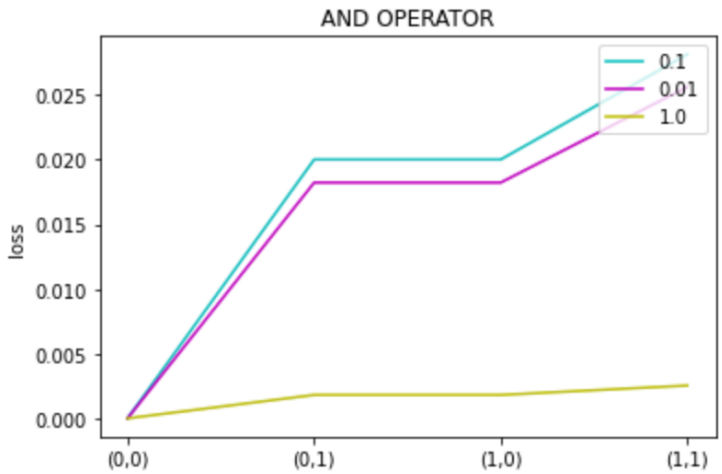
```
0.0018091459883322666
```

```
model.predict((1,0))
```

```
0.0018091459883209367
```

```
model.predict((1,1))
```

```
0.9974671380413457
```



Learning rate 가 0.1, 0.01, 1 중에 1 일 때 loss 가 가장 낮고 0.1 일 때 가장 높다. Learning rate 가 크다고 loss 가 가장 낮거나 작다고 loss 가 가장 큰 것은 아니라는 점을 알 수 있다. 따라서 learning rate 에는 적당한 값을 설정해 주어야 한다.

2. OR

```
X=np.array([[0,0],[0,1],[1,0],[1,1]])
Y=np.array([[0],[1],[1],[1]])
```

i. lr=0.01(or)

```
for epoch in range(10000):
    cost = train(X,Y,model,0.01)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.009362240698320416
100 0.009352657379482146
200 0.009343093513728315
300 0.00933354904218124
400 0.00932402390619982
500 0.009314518047378364
600 0.009305031407545093
700 0.00929556392876128
800 0.009286115553320177
900 0.009276686223745369
1000 0.009267275882790128
1100 0.009257884473435977
1200 0.009248511938891667
1300 0.009239158222592154
1400 0.009229823268196984
1500 0.009220507019590127
1600 0.009211209420877801
1700 0.009201930416388162
1800 0.009192669950669755
1900 0.009183427968490636
2000 0.009174204414837336
2100 0.00916499923491348
9100 0.00856324770569158
9200 0.00855521938092671
9300 0.008547205992953841
9400 0.008539207500316273
9500 0.008531223861709784
9600 0.008523255035982539
9700 0.008515300982133742
9800 0.008507361659313534
9900 0.00849943702682178
```

```
model.predict((0,0))
```

```
0.018750712934196424
```

```
model.predict((0,1))
```

```
0.9925118334900539
```

```
model.predict((1,0))
```

```
0.992508264096933
```

```
model.predict((1,1))
```

```
0.999998911756491
```

ii. lr=0.1(or)

```
for epoch in range(10000):
    cost = train(X,Y,model,0.1)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.7448015869439255
100 0.41157096859951736
200 0.3102044881677973
300 0.24606869761283737
400 0.2027300867986694
500 0.17172658046036077
600 0.14853275332255586
700 0.13057516009271988
800 0.11629248692548026
900 0.10468401981386498
1000 0.09507916717815153
1100 0.08701201790681881
1200 0.08014907565142292
1300 0.07424558013841309
1400 0.06911802284803316
1500 0.06462625556177443
1600 0.060661501300649436
1700 0.057138116981877846
1800 0.053987808754158484
1900 0.051155490618551326
2000 0.04859626814455375
2100 0.046273207483629626
2200 0.044155662064050485
9300 0.010085240301028638
9400 0.00997524703714894
9500 0.009867608718690529
9600 0.009762250899399956
9700 0.009659102226521363
9800 0.009558094282240593
9900 0.009459161434758626
```

```
model.predict((0,0))
```

```
0.02065998581116186
```

```
model.predict((0,1))
```

```
0.9917510613388042
```

```
model.predict((1,0))
```

```
0.9917462865070324
```

```
model.predict((1,1))
```

```
0.999998539706702
```

iii. **lr=1.0(or)**

```
for epoch in range(10000):
    cost = train(X,Y,model,1.0)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.008491527044107843
100 0.007767859977314262
200 0.007157272455817322
300 0.006635255474503983
400 0.006183891976295131
500 0.005789783390972625
600 0.0054427116420209904
700 0.005134749517169608
800 0.004859653531012153
900 0.004612439970741886
1000 0.004389082781086963
1100 0.004186294335183254
1200 0.004001363740605919
1300 0.0038320358147778916
1400 0.003676419283907372
1500 0.003532916296717233 8
1600 0.003400167697946950 6
1700 0.003277010100900447
1800 0.003162441895908701 6
1900 0.003055596098587220 7
2000 0.002955718485209839 7
2100 0.002862149852496290 6
2200 0.002774311522296582 5
9300 0.0008714472161877511
9400 0.0008631014150795696
9500 0.0008549138467657812
9600 0.0008468800554167494
9700 0.0008389957508946262
9800 0.0008312568011246722
9900 0.0008236592248839543
```

```
model.predict((0,0))
```

```
0.0018126677638707898
```

```
model.predict((0,1))
```

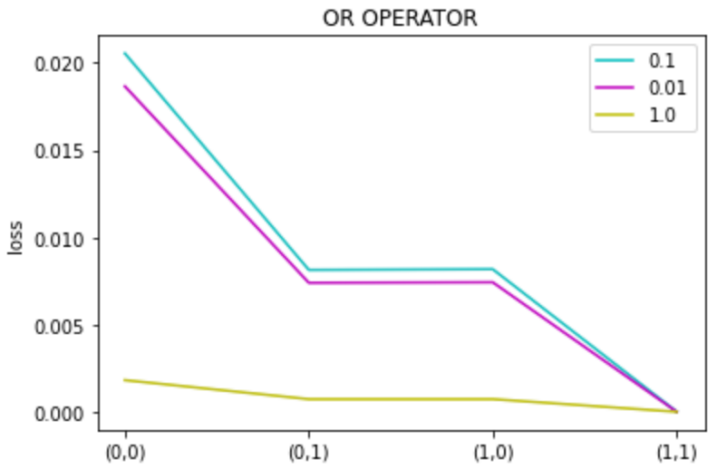
```
0.9992750229574504
```

```
model.predict((1,0))
```

```
0.9992750197407952
```

```
model.predict((1,1))
```

```
0.999999999044157
```



AND OPERATOR 와 마찬가지로 Learning rate 가 0.1, 0.01, 1 중에 1 일 때 loss 가 가장 낮고 0.1 일 때 가장 높다. Learning rate 가 크다고 loss 가 가장 낮거나 작다고 loss 가 가장 큰 것은 아니라는 점을 알 수 있다. 따라서 learning rate 에는 적당한 값을 설정해 주어야 한다.

3. XOR

```
X=np.array([[0,0],[0,1],[1,0],[1,1]])
Y=np.array([[0],[1],[1],[0]])
```

i. lr=0.01(xor)

```
for epoch in range(10000):
    cost = train(X,Y,model,0.01)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.6931471805599453
100 0.6931471805599453
200 0.6931471805599453
300 0.6931471805599453
400 0.6931471805599453
500 0.6931471805599453
600 0.6931471805599453
700 0.6931471805599453
800 0.6931471805599453
900 0.6931471805599453
1000 0.6931471805599453
1100 0.6931471805599453
1200 0.6931471805599453
1300 0.6931471805599453
1400 0.6931471805599453
1500 0.6931471805599453
1600 0.6931471805599453
1700 0.6931471805599453
1800 0.6931471805599453
1900 0.6931471805599453
2000 0.6931471805599453
2100 0.6931471805599453
2200 0.6931471805599453
9300 0.6931471805599453
9400 0.6931471805599453
9500 0.6931471805599453
9600 0.6931471805599453
9700 0.6931471805599453
9800 0.6931471805599453
9900 0.6931471805599453
```

```
model.predict((0,0))
```

```
0.5
```

```
model.predict((0,1))
```

```
0.5
```

```
model.predict((1,0))
```

```
0.5
```

```
model.predict((1,1))
```

```
0.5
```

ii. lr=0.1(xor)

```
for epoch in range(10000):
    cost = train(X,Y,model,0.1)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.7977451039694735
100 0.7214345839361004
200 0.7051235513334968
300 0.6983195999363069
400 0.6954177665190089
500 0.6941561569390167
600 0.693599392494443
700 0.6933510313947956
800 0.6932394239546243
900 0.6931890241028495
1000 0.6931661916747708
1100 0.6931558267176995
1200 0.6931511152909773
1300 0.6931489719135472
1400 0.6931479963105126
1500 0.693147552097287
1600 0.6931473497952234
1700 0.6931472576514511
1800 0.6931472156787089
1900 0.6931471965585684
2000 0.6931471878483536
2100 0.6931471838803192
2200 0.6931471820726144
9300 0.6931471805599453
9400 0.6931471805599453
9500 0.6931471805599453
9600 0.6931471805599453
9700 0.6931471805599453
9800 0.6931471805599453
9900 0.6931471805599453
```

```
model.predict((0,0))
```

```
0.5
```

```
model.predict((0,1))
```

```
0.5
```

```
model.predict((1,0))
```

```
0.5
```

```
model.predict((1,1))
```

```
0.5
```

iii. lr=1.0(xor)

```
for epoch in range(10000):
    cost = train(X,Y,model,1.0)
    if epoch%100==0:
        print(epoch, cost)
```

```
0 0.6931471805599453
100 0.6931471805599453
200 0.6931471805599453
300 0.6931471805599453
400 0.6931471805599453
500 0.6931471805599453
600 0.6931471805599453
700 0.6931471805599453
800 0.6931471805599453
900 0.6931471805599453
1000 0.6931471805599453
1100 0.6931471805599453
1200 0.6931471805599453
1300 0.6931471805599453
1400 0.6931471805599453
1500 0.6931471805599453
1600 0.6931471805599453
1700 0.6931471805599453
1800 0.6931471805599453
1900 0.6931471805599453
2000 0.6931471805599453
2100 0.6931471805599453
2200 0.6931471805599453
9300 0.6931471805599453
9400 0.6931471805599453
9500 0.6931471805599453
9600 0.6931471805599453
9700 0.6931471805599453
9800 0.6931471805599453
9900 0.6931471805599453
```

```
model.predict((0,0))
```

```
0.5
```

```
model.predict((0,1))
```

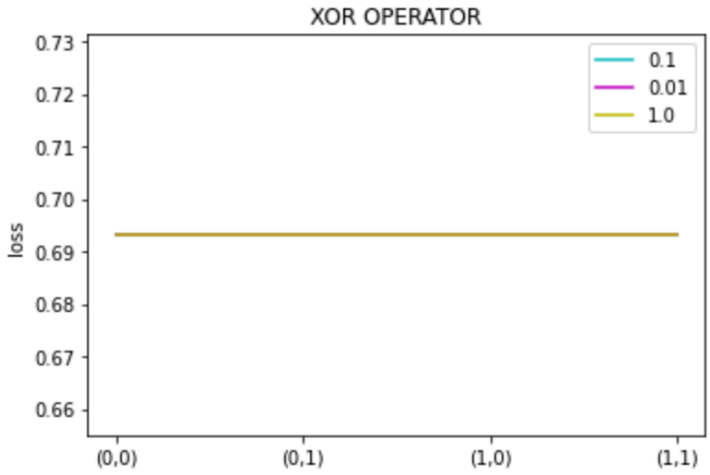
```
0.5
```

```
model.predict((1,0))
```

```
0.5
```

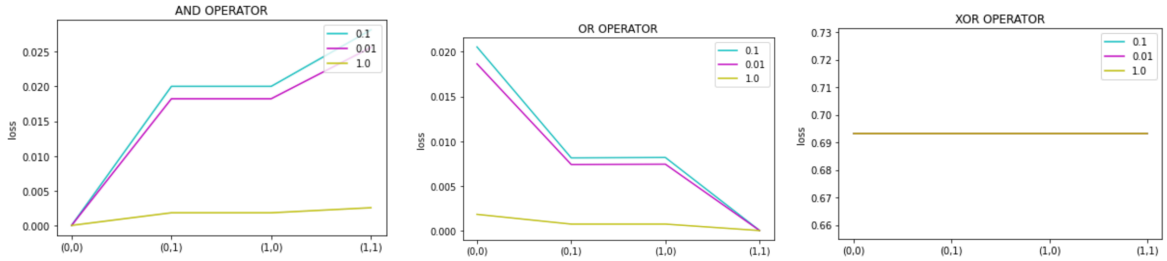
```
model.predict((1,1))
```

```
0.5
```



AND, OR OPERATOR 와 달리 XOR OPERATOR 는 learning rate 가 0.1, 0.01, 1 일 때 각각 모두 loss 가 같은 값으로 나온다. Model.predict 또한 모두 0.5 라는 잘못된 결과가 도출되는 것을 볼 수 있다.

결론



AND와 OR을 보면 알 수 있듯이 learning rate가 1일 때 loss가 가장 낮고 0.01이 아닌 0.1일 때 가장 높은 걸 볼 수 있다.

Learning rate의 값이 크다고 loss가 가장 낮은 것은 아니며, 반대로 learning rate의 값이 작다고 loss가 가장 높은 것 또한 아니다.

이는 적당한 learning rate의 값을 찾아 설정해주어야 함을 알려준다고 볼 수 있다.

다만 XOR의 경우에는 model.predict에서도 모두 0.5라는 값이 나오는 오류가 있었으며 loss plot에서도 모두 같은 값이 나오는 결과가 도출됨을 볼 수 있다.