철판 분류 신경망

정보컴퓨터공학부 강민진

jupyter notbook을 사용하였으며, 사이킷런의 MSE를 이용하였다.

철판 분류

27개의 특징 벡터, 7가지의 one-hot vector 형태의 불량 유형 정보.

-> 선택 분류를 통하여 결과가 7가지 중 하나로 나와야 한다.

선택 분류 문제에서는 softmax 함수를 사용할 수 있다.

각 후보 항목에 대한 **로짓값**(로그 척도의 상대적 추천 강도)를 추정하도록 구성된다.

또한, **확률 분포와 교차 엔트로피** 개념을 이용하여 교차 엔트로피를 손실 함수로 하여 학습을 수행하면 선택을 정답에 근접시킬 수 있다.

데이터 벡터 -> **퍼셉트론** -> **로짓값 벡터** -> **확률값 벡터** (softmax 함수를 이용하여 로짓값 벡터에서 확률값 벡터로 변환)

소프트맥스 일반식을 이용하여 계산하게 되면 x_i가 매우 큰 값이면 **오버플로**가 발생할 수 있으며, x_i가 매우 작은 값이면 **분모가 0**이 되어 오류가 발생할 수 있으므로 소프트맥스 변형식을 사용하는 것이 권장된다. 소프트맥스 변형식은 소프트맥스 일반식의 분모와 분자에 각각 e의 x_k 제곱승을 나눠주면 된다. (x_k는 x_i의 최댓값이다.)

- sigmoid 함수

sigmoid 함수는 softmax 함수에서 입출력 벡터 크기를 각각 2에서 1로 줄인 것이며, 거짓인 경우에 대한 로짓값을 0으로 전제한다.

선택 분류 문제에 softmax 함수 대신 sigmoid 함수를 사용하게 되면 전체 확률 값 1이라는 제한 없이 후보 항목 별로 확률을 계산할 수 있다.(softmax 함수는 후보 항목을 전부 합치면 확률 값 1)

sigmoid 함수와 softmax 함수는 대체로 비슷한 결과를 보이는 경우가 많다.

- Entropy

Entropy 란 확률적으로 발생하는 사건에 대한 정보량의 평균을 의미한다. 즉, 정보량에 대한 기댓값이며 동시에 사건을 표현하기 위해 요구되는 평균 자원이라고도 할 수 있다.

- Cross Entropy

두 개의 확률 분포 p와 q에 대해 하나의 사건 X가 갖는 정보량으로 정의된다. 즉, 서로 다른 두 확률분포에 대해 같은 사건이 가지는 정보량을 계산한 것이다. q에 대한 정보량을 p에 대해서 평균 낸 것으로 볼 수 있다.

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} L_{ic}log(P_{ic})$$

n = 데이터 갯수
C = 범주 갯수
L = 실제 값 (주로 0 또는 1)
P = 실제 값에 대한 확률 값 (0~1)

```
%run steel.ipynb

steel_exec()

Epoch 1: loss=42340319470364608.000, accuracy=0.283/0.317
Epoch 2: loss=33259026415803072O.000, accuracy=0.320/0.197
Epoch 3: loss=375496977437094208.000, accuracy=0.319/0.366
Epoch 4: loss=324020125985020864.000, accuracy=0.312/0.197
Epoch 5: loss=351108410488109120.000, accuracy=0.324/0.445
Epoch 6: loss=337380780169668288.000, accuracy=0.317/0.202
Epoch 7: loss=327085401357975104.000, accuracy=0.314/0.458
Epoch 8: loss=455603437806234304.000, accuracy=0.323/0.368
Epoch 9: loss=358882792467167424.000, accuracy=0.330/0.199
Epoch 10: loss=372600441821722048.000, accuracy=0.328/0.176

Final Test: final accuracy = 0.176

LEARNING_RATE = 0.001
steel_exec()

Epoch 1: loss=374595399826049472.000, accuracy=0.270/0.151
Epoch 2: loss=327585861879742784.000, accuracy=0.315/0.422
Epoch 3: loss=343250087061236480.000, accuracy=0.330/0.430
Epoch 4: loss=456570767342007424.000, accuracy=0.315/0.402
Epoch 5: loss=321088835962357568.000, accuracy=0.330/0.192
Epoch 6: loss=287941591008536896.000, accuracy=0.334/0.182
Epoch 7: loss=343088727095030912.000, accuracy=0.335/0.366
Epoch 8: loss=379529148627230848.000, accuracy=0.327/0.182
Epoch 9: loss=399492718719796928.000, accuracy=0.314/0.340
Epoch 10: loss=343490614176474304.000, accuracy=0.337/0.217

Final Test: final accuracy = 0.217
```

[그림 1] sigmoid, MSE(평균 제곱 오차)

```
%run steel.ipynb #-Copy1

steel_exec()

Epoch 1: loss=15.984, accuracy=0.306/0.320
Epoch 2: loss=15.509, accuracy=0.326/0.197
Epoch 3: loss=15.984, accuracy=0.306/0.348
Epoch 4: loss=15.004, accuracy=0.348/0.197
Epoch 5: loss=15.286, accuracy=0.336/0.202
Epoch 6: loss=15.390, accuracy=0.332/0.440
Epoch 7: loss=15.509, accuracy=0.326/0.442
Epoch 8: loss=15.628, accuracy=0.321/0.455
Epoch 9: loss=15.360, accuracy=0.333/0.322
Epoch 10: loss=15.316, accuracy=0.335/0.455

Final Test: final accuracy = 0.455

LEARNING_RATE = 0.1
steel_exec()

Epoch 1: loss=16.876, accuracy=0.267/0.297
Epoch 2: loss=15.346, accuracy=0.334/0.238
Epoch 3: loss=15.286, accuracy=0.336/0.238
Epoch 4: loss=15.568, accuracy=0.324/0.440
Epoch 5: loss=15.687, accuracy=0.319/0.248
Epoch 6: loss=15.687, accuracy=0.319/0.235
Epoch 7: loss=15.509, accuracy=0.326/0.361
Epoch 8: loss=15.658, accuracy=0.320/0.317
Epoch 9: loss=15.108, accuracy=0.344/0.437
Epoch 10: loss=15.271, accuracy=0.337/0.496

Final Test: final accuracy = 0.496
```

[그림 2] softmax, cross entropy

k-fold cross validation

학습이 ovefitting 될 수 있으므로 5개로 나눠서 5-fold cross validation을 하여 편향된 학습을 방지하였다.

함수 안의 구조에는 큰 변화가 없었으나, 중간과정과 결과에서는 큰 차이를 보였다. loss function이 높게 나타난 것은 데이터의 개수가 적다는 이유를 들 수 있다.

결론 :

softmax, cross entropy를 활용하는 경우에 정확도는 20 ~ 50% 대에서 왔다갔다하는 것을 확인할 수 있다. 단층 퍼셉트론에서 hyperparameter를 여러 값으로 바꿔도 정확도는 50%를 넘기기 힘들다.

원인은 데이터 수의 부족, 단층 퍼셉트론의 단순한 구조를 꼽을 수 있다.

tuning보다는 data 의 preprocessing, loss function을 어떻게 설정하느냐가 중요하며, 데이터의 개수와 학습에 따라 tuning의 의미와 영향이 달라진다는 것을 배울 수 있었다.