2장. 이진 판단 : 천체의 펄서 여부 판정 신경망

1. 개요

이진 판단 방법을 이용한 천체의 펄서 여부 판정 신경망 실험에서는 균형 잡힌 데이터, 즉 변형이 있는 데이터 사용의 필요성을 4가지 지표를 이용해 확인해보는 작업을 진행하였다. 이 실험에서 사용되는 비펄서와 펄서 데이터는 약 10:1의 비율로 치우쳐진 데이터이다. 따라서 신경망의 학습이 거의 이루어지지 않아도 무조건 펄서가 아니라고 하는 방법을 쓴다면, 쉽게 높은 정확도를 나타낼 수 있는 문제가 있다. 따라서 펄서 데이터 중복 사용, 잡음 추가 등의 방법으로 데이터의 균형을 맞추어 사용한다. 이 실험에서는 착시 없는 평가를 위해 정확도, 정밀도, 재현율, f1의 지표를 통해 데이터 균형의 영향을 알아보고자 한다.

2. 학습결과

우선 기존의 데이터 , 즉 균형 잡히지 않은 데이터의 경우 회귀 분석 방법과 비교해 정확도를 살펴보고 나서 펄서 데이터 중복 사용의 방법으로 균형을 맞춘 데이터의 결과를 살펴보고자 한다.

2.1 pulsar와 abalone 데이터 적용 시 정확도 분석 (pulsar.ipynb)

(1) 모델 1

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10
데이터 정확도	pulsar_exec : 97.6% , abalone_exec : 100.0%

결과 화면	pulsar_exec() Epoch 1: loss=0.154, accuracy=0.959/0.972 Epoch 2: loss=0.131, accuracy=0.966/0.972 Epoch 3: loss=0.136, accuracy=0.967/0.970 Epoch 4: loss=0.133, accuracy=0.968/0.970 Epoch 5: loss=0.121, accuracy=0.968/0.969 Epoch 6: loss=0.145, accuracy=0.968/0.974 Epoch 7: loss=0.122, accuracy=0.970/0.975 Epoch 8: loss=0.127, accuracy=0.970/0.976 Epoch 9: loss=0.125, accuracy=0.970/0.976 Epoch 10: loss=0.134, accuracy=0.968/0.976 Final Test: final accuracy = 0.976 abalone_exec() Epoch 1: loss=-40.173, accuracy=1.000/1.000 Epoch 2: loss=-119.967, accuracy=1.000/1.000 Epoch 3: loss=-199.559, accuracy=1.000/1.000 Epoch 4: loss=-279.150, accuracy=1.000/1.000 Epoch 5: loss=-358.740, accuracy=1.000/1.000 Epoch 6: loss=-438.331, accuracy=1.000/1.000 Epoch 7: loss=-517.923, accuracy=1.000/1.000 Epoch 8: loss=-597.514, accuracy=1.000/1.000 Epoch 9: loss=-677.104, accuracy=1.000/1.000 Epoch 10: loss=-756.694, accuracy=1.000/1.000 Final Test: final accuracy = 1.000

(2) 모델 2

구분	내용
조건	- 에포크 횟수(epoch_count) : 100 - 미니배치 크기(mb_size) : 10
데이터 정확도	pulsar_exec : 97.7% , abalone_exec : 100.0%

결과 화면	

```
pulsar_exec()

Epoch 10:  loss=0.134, accuracy=0.968/0.976
Epoch 20:  loss=0.123, accuracy=0.971/0.969
Epoch 30:  loss=0.126, accuracy=0.969/0.974
Epoch 40:  loss=0.128, accuracy=0.970/0.975
Epoch 50:  loss=0.123, accuracy=0.969/0.975
Epoch 60:  loss=0.130, accuracy=0.970/0.974
Epoch 70:  loss=0.127, accuracy=0.969/0.971
Epoch 80:  loss=0.133, accuracy=0.971/0.975
Epoch 90:  loss=0.124, accuracy=0.970/0.975
Epoch 100:  loss=0.116, accuracy=0.971/0.977

Final Test: final accuracy = 0.977

abalone_exec()

Epoch 10:  loss=-753.259, accuracy=1.000/1.000
Epoch 20:  loss=-1545.559, accuracy=1.000/1.000
Epoch 30:  loss=-2337.859, accuracy=1.000/1.000
Epoch 40:  loss=-3130.161, accuracy=1.000/1.000
Epoch 50:  loss=-3922.461, accuracy=1.000/1.000
Epoch 60:  loss=-4714.762, accuracy=1.000/1.000
Epoch 70:  loss=-5507.063, accuracy=1.000/1.000
Epoch 80:  loss=-6299.362, accuracy=1.000/1.000
Epoch 90:  loss=-7091.663, accuracy=1.000/1.000
Epoch 100:  loss=-7883.964, accuracy=1.000/1.000

Final Test: final accuracy = 1.000
```

2.2 균형 잡힌 데이터와 아닌 데이터 비교 (pulsar_ext.ipynb)

순서대로 정확도, 정밀도, 재현율, F1값(정밀도와 재현율의 조화 평균)을 나타내었다.

(1) 모델 3 (균형 잡히지 않은 데이터의 경우)

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - adjust_ratio : false
데이터 결과	정확도: 96.7%, 정밀도: 97.6%, 재현율: 64.9%, F1값: 78.0%

결과 화면	pulsar_exec() Epoch 1: loss=0.139, result=0.965,0.975,0.624,0.761 Epoch 2: loss=0.128, result=0.971,0.969,0.696,0.810 Epoch 3: loss=0.130, result=0.973,0.951,0.734,0.828 Epoch 4: loss=0.131, result=0.973,0.917,0.762,0.832 Epoch 5: loss=0.130, result=0.975,0.953,0.755,0.843 Epoch 6: loss=0.127, result=0.974,0.893,0.809,0.849 Epoch 7: loss=0.130, result=0.972,0.950,0.718,0.818 Epoch 8: loss=0.119, result=0.974,0.960,0.743,0.837 Epoch 9: loss=0.116, result=0.974,0.971,0.727,0.832 Epoch 10: loss=0.120, result=0.967,0.976,0.649,0.780 Final Test: final result = 0.967,0.976,0.649,0.780

(2) 모델 4 (균형 잡힌 데이터 - 펄서 데이터의 비율을 늘린 경우)

구분	내용
조건	- 에포크 횟수(epoch_count) : 10 - 미니배치 크기(mb_size) : 10 - adjust_ratio : true
데이터 결과	정확도: 91.5%, 정밀도: 90.9%, 재현율: 91.9%, F1값: 91.4%
결과 화면	pulsar_exec(adjust_ratio= True) Epoch 1: loss=0.405, result=0.923,0.939,0.902,0.920 Epoch 2: loss=0.376, result=0.918,0.983,0.848,0.910 Epoch 3: loss=0.377, result=0.904,0.879,0.934,0.906 Epoch 4: loss=0.364, result=0.915,0.987,0.839,0.907 Epoch 5: loss=0.378, result=0.925,0.941,0.905,0.923 Epoch 6: loss=0.367, result=0.923,0.982,0.859,0.916 Epoch 7: loss=0.369, result=0.917,0.986,0.844,0.909 Epoch 8: loss=0.363, result=0.927,0.940,0.910,0.925 Epoch 9: loss=0.349, result=0.926,0.981,0.867,0.920 Epoch 10: loss=0.378, result=0.915,0.909,0.919,0.914 Final Test: final result = 0.915,0.909,0.919,0.914

3.결론

2.1 (pulsar와 abalone데이터 적용 시 정확도 분석) 의 학습 결과를 살펴보면 다음 표와 같다.

에포크 횟수(= 미니배치 크기)		10	100
정확도	pulsar_exec	97.6%	97.7%
	abalone_exec	100%	100%

pulsar_exec의 경우 약 97%로 아주 높은 정확성을 보여주었다. 정확도가 높아서 좋은 평가 방법이라 생각할 수 있지만, 균형 잡히지 않은 데이터셋에서는 90% 이상의 정확도가 도달하기 쉬운 수치임에 주의해야 한다. 모든 문제에 펄서가 아니라고 답하게 되면 정확도가 아주 높아질 것이다. 즉, 정확도 결과에 착시가 포함될 수 있음에 주의해야 한다는 것이다. 또한, 에포크 횟수를 높여보았음에도 비슷한 수치의 정확도가 나오는 모습에서 착시가 포함되어있을 확률이 높음을 유추해 볼 수도 있다.

abalone_exec의 경우, 엉뚱한 loss값과 함께, 정확도가 100%에 달하는 모습을 보였다. 이는 pulsar.ipynb 파일에서 eval_accuracy(), forward_postproc(), backward_postproc()를 시그모이드 함수를 이용하여 이진 판단에 맞는 형태로 수정했기 때문이다. 회귀 분석의 성격에 맞지 않아 터무니 없는 정확도를 보였음을 유추해볼 수 있다.

위 결과에서 pulsar_exec의 결과 또한 너무 높은 정확도를 보여 온전한 성능 검사가 불가능하기에 정밀도, 재현율, F1 값의 확장된 척도를 사용해 이후 실험을 진행하였다.

cf) 정확도 $= \frac{TP + FN}{TP + TN + FP + FN}$, 정밀도 $= \frac{TP}{TP + FP}$, 재현율 $= \frac{TP}{TP + TN}$,

F1 $= \frac{2 * TP}{2 * TP + FP + TN}$

2.2 (균형 잡힌 데이터와 아닌 데이터 비교 실험) 의 학습 결과를 살펴보면 다음 표와 같다.

데이터셋 변형	adjust_ratio	정확도	정밀도	재현율	f1값
X	false	96.7%	97.6%	64.9%	78.0%
O	true	91.5%	90.9%	91.9%	91.4%

cf) adjust_ratio가 false일 때는 데이터셋의 변형이 없다. (즉, 균형 잡히지 않은 데이터를 사용하였다.) 하지만, adjust_ratio가 true일 때는 펄서의 비율을 늘려, 별과 펄서의 양이 같도록 변경하였다.

이 결과에서 보면, 데이터셋 변형이 없을 때에 비해 변형이 있을 때 정확도 및 정밀도가 소폭 하락하는 모습을 볼 수 있다. 하지만 반대로, 재현율과 f1값은 크게 증가하는 것을 확인 할 수 있다.

2.1 실험의 결과에서도 보았듯이 균형 잡히지 않은 데이터셋에서는 높은 정확도에 쉽게 도달 가능하므로 (착시 현상이 있다.) , 다른 지표의 결과 또한 좋은 모델이 성능이 더 뛰어나다라고 이야기 할 수 있을 것이다. 그런 관점에서 본다면 , 별과 펄서 비율의 균형을 맞춘 모델이 재현율과 f1의 부분에서 큰 증가를 보여주었기에, 데이터셋의 변형이 없는 모델보다 성능이 개선되었다고 볼 수 있다.

위 실험을 통해 이진 판단을 이용한 신경망에서 데이터 비율 변경(데이터 중복 사용) , 잡음 추가 등의 방법으로 데이터의 균형을 맞추어 사용하는 것이 균형 잡히지 않은 데이터를 사용할 때보다

여러 면에서 성능 개선의 결과를 가져올 수 있음을 확인할 수 있었다.