2장. 이진 판단 : 천체의 펄서 예측 신경망

부산대학교 정보컴퓨터공학부

15학번 이연걸

1. 개요

균형잡힌 데이터셋과 그렇지 않은 데이터셋간에 차이를 비교해보는 작업을 진행하였다. 기본적으로 천체의 펄서 예측을 위한 신경망 모델의 실험을 진행하였다. 펄서 예측 데이터셋의 데이터들은 90.8% 정도가 일반 별이고 불과 9.2%만 펄서다. 이 때문에 학습이 거의 안 된 신경망도 무조건 펄서가 아니라고 답하는 간단한 방법으로 90% 이상의 정확도를 쉽게 달성할 수 있다.

따라서 데이터의 균형을 맞추어 진행을 한다. 이 경우, 일반 별에 치우쳤던 학습도 균형을 잡으면서 신경망의 추정 성능이 향상되겠지만 그럴 경우 착시 현상이 걷히면서 겉보기 정확도는 오히려 하락할 가능성이 있다. 따라서, 정확도뿐만 아니라 신경망의 성능을 잘 보여줄 수 있는 정밀도, 재현율, F1 등의 지표를 통해서 차이를 알아본다.

한편, 데이터의 균형을 맞출 때 일반 별 데이터 가운데 상당수를 버리면 균형이 맞춰지겠지만 소중한 데이터를 버리는 행위는 좋지 못하기 때문에 펄서 데이터를 중복 사용하고 약간의 잡음 추가를 통해 균형을 맞춘다.

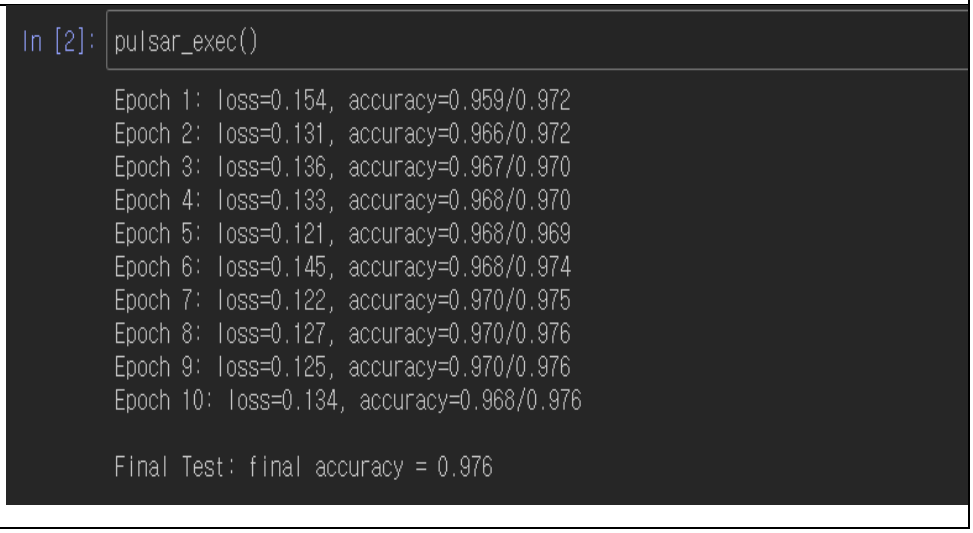
2. 학습결과 및 결론

2.1 균형잡히지 않은 데이터셋의 경우 (pulsar_test)

우선, 균형잡히지 않은 데이터셋으로 pulsar_exec()을 실행시켜본뒤, abalone_exec()실행을 통하여 호환성을 확인해본다.

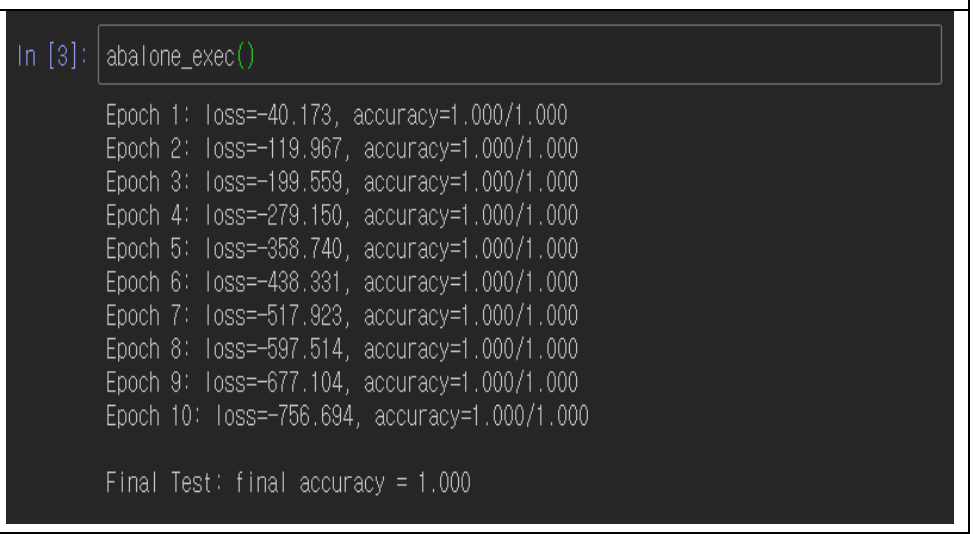
2.1.1 pulsar_exec()

Table 1 pulsar_exec() (균형잡히지 않은 데이터셋)

구분	내용
데이터 정확도	97.6%
결과화면	

2.1.2 abalone_exec()

Table 2 abalone_exec() (균형잡히지 않은 데이터셋)

구분	내용
데이터 정확도	100.0%
결과화면	

2.1.3 결론

2.1.1에서 실행결과 첫 에포크에서 얻어진 97.2%의 정확도가 추가 학습을 진행해 97.6%로 조금 높아졌다. 퍼셉트론이 하나인 단층 퍼셉트론 신경망에서 이 정도 정확도가 얻어지는 것으로 보아 특징값 8개를 이용해 펄서 여부를 판정하는 방법은 상당히 적절해보인다.

2.1.2에서 실행결과 abalone_exec() 함수는 잘 동작하였지만, 터무니없는 손실 함숫값들이 출력되면서 정확도가 1.000으로 나타났다. 이는 pulsar.ipynb 파일에서 abalone.ipynb 파일을 불러들인 후에 forward_postproc(), backward_postproc(), eval_accuracy() 함수를 회귀 분석이 아닌 이진 판단에 알맞은 형태로 수정했기 때문이다. 전복의 고리 수를 추정하는 회귀 분석 문제의 성격에 전혀 맞지 않은 방법이어서 올바르게 동작하지 않았다.

2.2 균형 잡힌 데이터셋의 경우 (pulsar_ext_test)

에포크를 10으로 설정하고 에포크마다 정확도, 정밀도, 재현율, F1 순으로 변화를 확인하였다. 여기서 정밀도란 신경망이 참으로 추정한 것 가운데 정답이 참인 것의 비율, 재현율이란 거꾸로 정답이 참인 것들 가운데 신경망이 참으로 추정한 것의 비율을 말한다. 정밀도와 재현율은 하나만 살펴서는 큰 의미가 없고 정확도가 높아지려면 정밀도와 재현율 가운데 적어도 한 가지는 높아야 하는데, 정확도만 가지고서는 두 값 가운데 어느 쪽이 높고 어느 쪽이 낮은지 쉽게 알아차릴 수 없다.

따라서 F1 값을 정확도 대신 평가지표로 사용하기도 한다. F1값은 정밀도와 재현율의 조화 평균으로 정의된다. 조화 평균의 성질상 F1값은 정확도와 달리 정밀도와 재현율의 두 값을 경계로 하는 구간 안에서만 값을 가질 수 있다. 특히 두 값 중 작은쪽에 가까워지는 성질이 있다. 따라서 정밀도와 재현율 모두 높다면 F1값이 커지고 더욱 균형 잡힌 성능 지표의 역할을 할 수 있게 된다. 네 가지 평가지표 정확도, 정밀도, 재현율, F1 값을 수식으로 정의해본다. 미니배치 데이터들을 정답이 참인지 거짓인지에 따라 T(True), F(False)의 두 집단으로 나누고 추정이 참인지 거짓인지에 따라 P(Positive), N(Negative)의 두 집단으로 나눈다. 이 두 가지 기준에 따라 데이터들을 TP, TN, FP, FN의 네 집단으로 나눌 수 있으며 네 가지 평가지표는 각각 다음과 같이 정의된다.

$$\text{정확도} = \frac{TP + FN}{TP + TN + FP + FN} \quad \text{정밀도} = \frac{TP}{TP + FP}$$

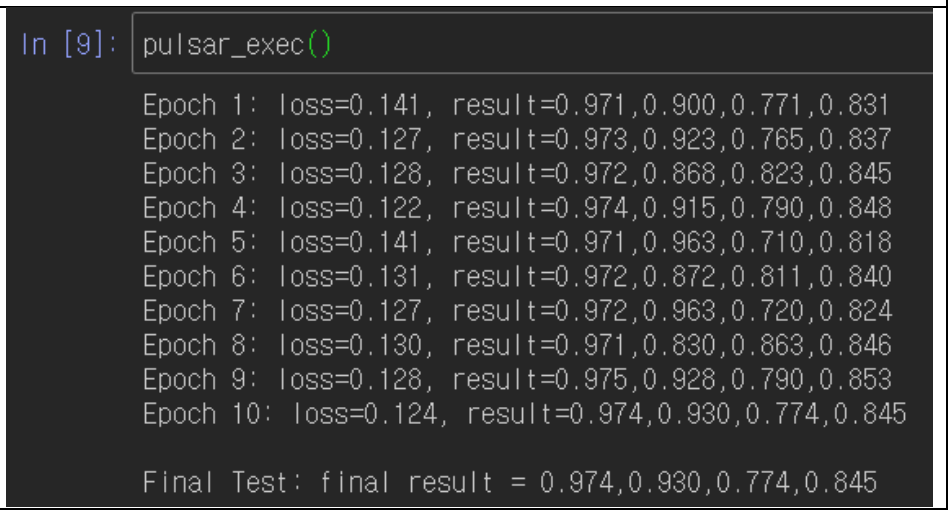
$$\text{재현율} = \frac{TP}{TP + TN}$$

$$\text{F1} = \frac{2 * \text{정밀도} * \text{재현율}}{\text{정밀도} + \text{재현율}} = \frac{2}{\frac{1}{\text{정밀도}} + \frac{1}{\text{재현율}}} = \frac{2}{\frac{TP+FP}{TP} + \frac{TP+TN}{TP}} = \frac{2*\text{TP}}{2*TP+FP+TN}$$

한편, 2.2.1에서는 adjust_ratio 매개변숫값이 디폴트값인 False이므로 펄서 데이터보다 별 데이터가 훨씬 많은 기존 데이터셋을 그대로 이용하고, 2.2.2에서 True로 하여 펄서 데이터를 필요한 만큼 중복 사용하는 방법으로 두 가지 데이터 수를 같게 만든 균형 잡힌 데이터를 이용할 수 있도록 한다. 이를 통해 2.2.1과 2.2.2를 비교하고 균형잡힌 데이터셋과 그렇지 않은 데이터셋의 평가지표들의 차이를 살펴본다.

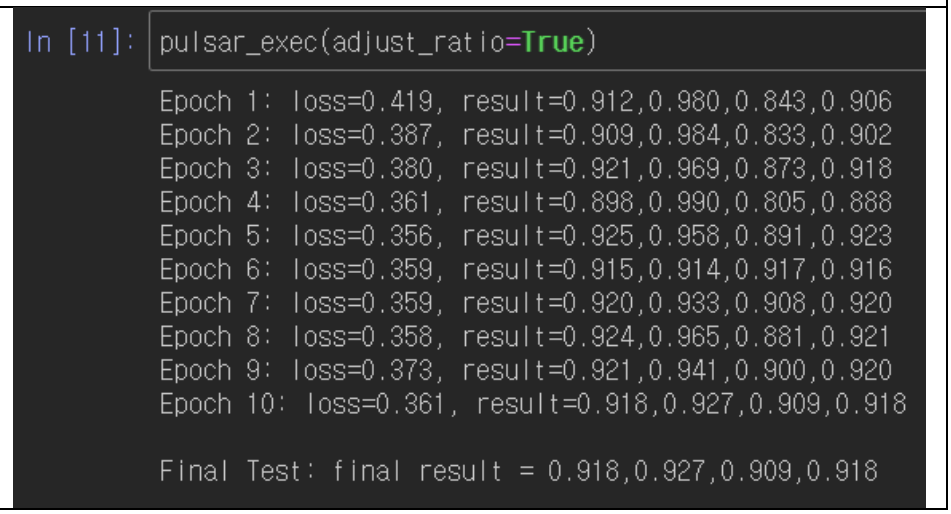
2.2.1 pulsar_exec() (adjust_ration가 디폴트인 False이므로 균형잡히지 않은 데이터셋 사용)

Table 3 pulsar_exec()

구분	내용
데이터 정확도	정확도 - 97.4%, 정밀도 – 93.0%, 재현율 – 77.4%, F1 – 84.5%
결과화면	

2.2.2 pulsar_exec(adjust_ratio=True) (균형잡힌 데이터 사용)

Table 4 pulsar_exec(adjust_ratio=Ture)

구분	내용
데이터 정확도	정확도 – 91.8%, 정밀도 – 92.7%, 재현율 – 90.9%, F1 – 91.5%
결과화면	

2.2.3 결론

우선 2.2.1을 살펴본다. 네 개 숫자는 순서대로 정확도, 정밀도, 재현율, F1 값에 해당한다. 전체적으로 정밀도가 93.0%이지만, 재현율이 77.4%로 매우 낮다. 정확도는 97.4%로서 상당히 높은 값이 얻어졌다. 별 데이터가 많다 보니 학습이 별 판정을 선호하는 쪽으로 몰리면서 별을 별로 답하는 경우가 많이 반영되다 보니 낮은 재현율에도 착시 현상이 생기면서 정확도가 높다. 이에 반해 F1값은 84.5%로 정밀도와 재현율을 잘 대표한다.

2.2.2를 살펴본다. 정확도가 97.4%에서 91.8%로 떨어져 성능이 많이 나빠진 것처럼 보인다. 하지만, 정밀도는 92.7%로 비슷했으며 재현율이 77.4%에서 90.9%로 크게 좋아졌다. 재현율의 급격한 상승에 힘입어 F1값 또한 84.5%에서 91.8%로 높아졌다. 펄서 데이터와 별 데이터가 균형을 잡으면서 정확도가 하락한 것 같지만 실제로는 성능이 크게 좋아진 것이다.

한편 부족한 펄서 데이터들을 복사해 별 데이터와 같은 개수로 채우고 나면 같은 내용의 데이터가 학습용 데이터와 평가용 데이터 양쪽 모두 이용될 수 있어서 오버피팅 현상이 일어날 수 있다. 이를 줄이기 위해서는 조금씩 다른 케이스를 조작적으로 추가시켜서 다양한 케이스를 확보할 수 있도록 Data Augmentation을 활용한다. 이는 특히 이미지 분류에서 많이 사용되는데, 멀쩡한 이미지를 회전시키거나 약간 찌그러뜨린 수정본들도 함께 학습시킨다고 생각하면 되겠다.