2022 Deep Learning Study

Deep Learning Ch3-4

D조 Ch3-4 김도희 류하영 설현기 이지수 최동석 하정원

CH4 머신 러닝의 기본 요소

Machine -Learning

경험을 통해 자동으로 개선하는 컴퓨터 알고리즘의 연구

훈련 데이터의 성능에 비해 처음 본 데이터에 대한 성능이 좋아지지 않음

but, 훈련 데이터의 성능은 훈련 진행될 수록 항상 증가

머신 러닝 목표: 처음 본 데이터에서 잘 작동하는 일반화된 모델을 얻는 것

=> 따라서, 과대적합은 주요 장애물

관측할 수 있는 것만 제어할 수 있으므로, 모델 일반화 성능에 대해 **신뢰 가능한 측정 방법이 아주 중요**

이번 절에선 일반화, 즉 머신 러닝 모델의 성능을 어떻게 측정하는지 집중

4.2.1
훈련, 검증, 테스트 세트

A. 모델 평가 핵심 : 가용 데이터를 항상 훈련/검증/테스트 3개의 세트로 나누는 것

**훈련/테스트 세트 2개를 사용하지 않는 이유 : 모델 개발할 때 항상 모델 설정을 튜닝하기 때문

ex).

*① **층 수/층의 유닛 수**를 선택 (이런 파라미터를 네트워크 가중치와 구분하기 위해 하이퍼파라미터라 함)*

*② **검증 세트에서 모델 성능 평가**하여 이런 튜닝 수행, 이때 이 튜닝도 어떤 파라미터 공간에서 좋은 설정을 찾는 학습*

=> 검증 세트 성능을 기반으로 모델 설정을 튜닝하면 검증 세트로 모델을 직접 훈련하지 않더라도 빠르게 검증 세트에 과대적합됨, 핵심은 정보 누설 개념에 존재

B. 정보 누설 개념 : 검증 세트의 모델 성능 기반해 모델 하이퍼파라미터를 조정할 때마다 검증 데이터 관한 정보가 모델로 새는 것

① 하나의 파라미터에 대해 단 한 번만 튜닝한다면 아주 적은 정보가 누설

② 검증 세트로 모델을 평가할 만하지만 한 번 튜닝하고 난 이후, 검증 세트 평가한 결과로 다시 모델 조정 과정 여러 번 반복하면 검증 세트 관한 정보 모델에 多 노출

=> 검증 데이터 맞춰 최적화 했기 때문에 검증 데이터에 의도적으로 잘 수행되는 모델 만들어짐

검증 데이터 아니고 완전히 새로운 데이터에 대한 성능이 관심 대상이라면 모델 평가하기 위해 이전에 본 적 없는 **완전히 다른 데이터셋**을 사용해야 함

=> 여기서 새로운 데이터셋 = 테스트 세트

모델은 간접적으로라도 테스트 세트에 대한 어떤 정보도 얻어서는 X, 테스트 세트 성능에 기초해 튜닝한 모델의 모든 설정은 **일반화 성능을 왜곡**

데이터 3세트로 나누는 것은 간단해 보일 수 있지만 데이터 적을 땐 몇 가지 고급 기법 사용하면 도움됨

=> 대표적 3가지 평가 방법 : 단순 홀드아웃 검증, K-겹 교차 검증, 셔플링사용한 반복 K-겹 교차 검증

4.2.1

A. 단순 홀드아웃 검증

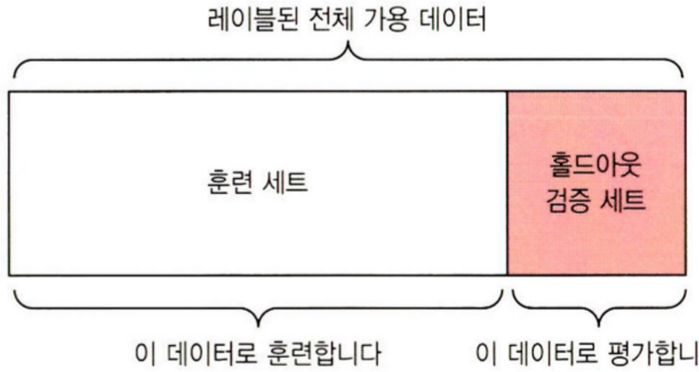
---

① 데이터 일정량을 테스트 결과로 떼어놓음
② 남은 데이터에서 훈련하고 테스트 세트로 평가

정보 누설을 막기 위해 테스트 세트 사용하여 모델 튜닝해선 X
=> 이와 같은 이유로 검증 세트도 따로 떼어 놓아야 함

단점 : 데이터 적을 땐, 검증/테스트 세트의 샘플도 너무 적어 주어진 전체 데이터를 통계적으로 대표 X
=> 다른 난수 초기값으로 셔플링해 데이터 나눴을 때, 모델 성능 매우 달라지는 점에서 확인 가능

그림 4-1 단순 홀드아웃 검증 분할



코드 4-1 홀드아웃 검증 구현 예

```
num_validation_samples = 10000

np.random.shuffle(data) -------- 데이터를 섞는 것(셔플링)이 일반적으로 좋습니다.

validation_data = data[:num_validation_samples] -------- 검증 세트를 만듭니다.
data = data[num_validation_samples:]

training_data = data[:] -------- 훈련 세트를 만듭니다.

model = get_model()
model.train(training_data)
validation_score = model.evaluate(validation_data)  ---- 훈련 세트에서 모델을 훈련하고 검증 세트로 평가합니다.

# 여기에서 모델을 튜닝하고,
# 다시 훈련하고, 평가하고, 또 다시 튜닝하고 ...

model = get_model()
model.train(np.concatenate([training_data,
                            validation_data]))  ---- 하이퍼파라미터 튜닝이 끝나면 테스트 데이터를 제외한 모든 데이터를 사용하여 모델을 다시 훈련시킵니다.[4]
test_score = model.evaluate(test_data)
```

훈련-평가-튜닝 반복할 때,
계속 새로운 모델 만듦

최적 하이퍼파라미터 구한 후,
마지막 모델 훈련시킬 때, 훈련/검증 데이터 모두 사용하는 게 중요

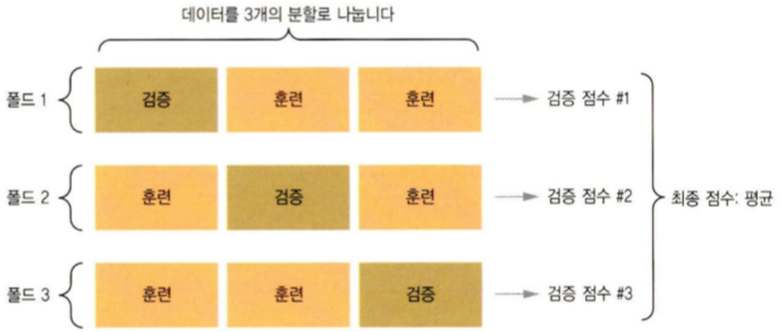
4.2.1

B. K-겹 교차 검증

① 데이터를 동일 크기 가진 K개 분할로 나눔
② 각 분할 i에 대해 남은 K-1개 분할로 모델 훈련 후 분할 i에서 모델 평가
③ 최종 점수 : 이렇게 얻은 K개 점수를 평균화

=> 모델 성능이 데이터 분할 따라 편차 클 때 도움,
홀드 아웃처럼 모델 튜닝에 별개 검증 세트 사용

그림 4-2 3-겹 교차 검증



코드 4-2 K-겹 교차 검증 구현 예[5]

```
k = 4
num_validation_samples = len(data) // k

np.random.shuffle(data)

validation_scores = []
for fold in range(k):
    validation_data = data[num_validation_samples * fold:
     num_validation_samples * (fold + 1)]
    training_data = data[:num_validation_samples * fold] +
     data[num_validation_samples * (fold + 1):]

    model = get_model() ------- 훈련되지 않은 새로운 모델을 만듭니다.
    model.train(training_data)
    validation_score = model.evaluate(validation_data)
    validation_scores.append(validation_score)

validation_score = np.average(validation_scores) ------- 검증 점수: K개 폴드의 검증 점수 평균

model = get_model()
model.train(data)
test_score = model.evaluate(test_data)
```

검증 데이터 부분을 선택합니다.

남은 데이터를 훈련 데이터로 사용합니다. 리스트에서 + 연산자는 두 리스트를 더하는 것이 아니고 연결합니다.

테스트 데이터를 제외한 전체 데이터로 최종 모델을 훈련합니다.

K-겹 교차 검증은 사이킷런의 cross_validate() 함수를 사용하여 쉽게 구현 가능

해당 함수 사용하려면 케라스 모델을 사이킷런과 호환되도록 KerasClassifier/KerasRegressor 클래스로 모델을 감싸야 함

4.2.1

C. 셔플링 사용한 반복 K-겹 교차 검증

비교적 가용 데이터 적고 정확하게 모델 평가하고자 할 때 사용

*** 셔플링 사용한 반복 K-겹 교차 검증 방법**

① K-겹 교차 검증 여러 번 적용하되, K개의 분할로 나누기 전에 매번 데이터를 무작위로 섞음

② 최종 점수 : 모든 K-겹 교차 검증을 실행해 얻은 점수의 평균

=> P * K개 (P=반복 횟수) 모델 훈련+평가하므로 비용 大

참고) 사이킷런 0.19버전에 추가된 RepeatedKFold(회귀) + RepeatedStratified(분류) 클래스를 cross_validate() 함수 적용하여 구현 가능

Summary

평가 방식을 선택할 때 다음 사항 유념 필수

① **대표성 있는 데이터** : 훈련/테스트 세트가 주어진 데이터에 대한 대표성 필요

ex). 숫자 이미지를 분류하는 문제에서 샘플 배열이 클래스 순서대로(0~9) 나열되어 있다고 가정

이 배열의 처음 80% 훈련/20% 테스트로 만들면 훈련 0~7/테스트 8~9가 담김

=> 훈련/테스트 세트로 나누기 전에 데이터를 무작위로 섞는 것이 일반적

② **시간의 방향** : 과거로부터 미래를 예측하려고 할 때, 데이터 분할 전 무작위로 섞는 건 절대 X

=> 미래의 정보가 누설되기 때문 *모델이 사실상 미래 데이터에서 훈련될 수 있어 훈련 데이터보다*

테스트 데이터가 미래의 것이어야 함

③ **데이터 중복** : 데이터를 섞고 훈련/검증 세트로 나누었을 때 데이터 포인트가 중복

=> 이는 훈련 데이터 일부로 테스트하는 최악의 경우

THANK YOU

D조 Ch3-4 김도희 류하영 설현기 이지수 최동석 하정원