보편적인 머신 러닝 작업 흐름

1. 문제 정의와 데이터셋 수집

먼저 주어진 문제를 정의해야 한다.

- 입력 데이터는 무엇인가? 어떤 것을 예측하려 하나?
- 당면한 문제가 어떤 종류인가? 이진 분류인가? 다중 분류인가? 스칼라 회귀인가? 문제의 유형을 식별하면 모델의 구조와 손실 함수 등을 선택하는 데 도움이 된다.

이 단계에서 가설을 세워야 한다.

- 주어진 입력으로 출력을 예측할 수 있다고 가정을 세우자.
- 가용된 데이터에 입력과 출력 사이의 관계를 학습하는 데 충분한 정보가 있다고 가설을 세우자.

입력 X와 타깃 Y의 샘플을 수집했다고 X에 Y를 예측하기에 충분한 정보가 있는 것은 아니다. 모델이 작동하기 전까지 가설에 불가하다. 검증될지 아닐지 지켜봐야 한다.

머신 러닝은 훈련 데이터에 있는 패턴을 기억하기 위해서만 사용한다는 것을 유념하자. 이미 보았던 것만 인식할 수 있다. 미래를 예측하기 위해 과거 데이터에서 훈련한 머신 러닝을 사용하는 것은 미래가 과거처럼 움직인다고 가정한 것이다.

2. 성공 지표 선택

성공하기 위해서는 성공은 무엇인가를 정의해야 한다. 정확도인가? 정밀도나 재현율인가? 고객 재방문인가? 성공의 지표가 모델이 최적화할 손실 함수를 선택하는 기준이 된다. 머신 러닝의 다양한 성공 지표가 여러 가지 종류의 문제에 어떻게 관련되어 있는지 알고 싶다면 캐글의 데이터 과학 경연 대회를 살펴보자.

3. 평가 방법 선택

목표를 정했다면 현재 진척 상황을 평가할 방법을 정해야 한다.

- 홀드아웃 검증 세트 분리 : 데이터가 풍부할 때 사용한다.

- K-겹 교차 검증 : 홀드아웃 검증을 사용하기에 샘플의 수가 너무 적을 때 사용한다.
- 반복 K-겹 교차 검증 : 데이터가 적고 매우 정확한 모델 평가가 필요할 때 사용한다.

---

4. 데이터 준비

먼저 머신 러닝 모델에 주입할 데이터를 구성해야 한다. 여기에서는 이 머신 러닝 모델을 심층 신경망이라고 가정한다.

- 데이터는 텐서로 구성된다.
- 텐서에 있는 값을 일반적으로 작은 값으로 스케일 조정 되어 있다. 예를 들어 [-1, 1]이나 [0, 1] 범위이다.
- 특성마다 범위가 다르면 정규화해야 한다.
- 특성 공학을 수행할 수 있다. 특히 데이터가 적을 때이다.

입력 데이터와 타깃 데이터의 텐서가 준비되면 모델을 훈련할 수 있다.

---

5. 기본보다 나은 모델 훈련하기

이 단계의 목표는 통계적 검정력을 달성하는 것이다. 아주 단순한 모델보다 나은 수준의 작은 모델을 개발한다.

통계적 검정력을 달성하는 것이 항상 가능하지 않다. 여러 개의 타당성 있는 네트워크 구조를 시도해 보고 무작위로 예측하는 모델보다 낫지 않다면 입력 데이터에 존재하지 않는 것을 얻으려고 하는 신호이다. 2개의 가설이 있다.

- 주어진 입력으로 출력을 예측할 수 있다.
- 가용한 데이터에 입력과 출력 사이의 관계를 학습하는 데 충분한 정보가 있다.

이 가설이 잘못된 것일 수 있지만, 그때는 기획부터 다시 해야 한다. 하지만 일이 잘 진행된다고 가정하면 첫 번째 모델을 만들기 위해 세 가지의 중요한 선택을 해야 한다.

- **마지막 층의 활성화 함수** : 네트워크의 출력에 필요한 제한을 가한다. 네트워크에 따라 마지막 층에 사용하는 함수가 다르다.
- **손실 함수** : 풀려고 하는 문제의 종류에 적합해야 한다.
- **최적화 설정** : 어떤 옵티마이저를 사용하나? 학습률은 얼마인가? 대부분의 경우 rmsprop과 기본 학습률을 사용하는 것이 무난하다.

문제 유형	마지막 층의 활성화 함수	손실 함수
이진 분류	시그모이드	binary_crossentropy

문제 유형	마지막 층의 활성화 함수	손실 함수
단일 레이블 다중 분류	소프트맥스	categorial_crossentropy
다중 레이블 다중 분류	시그모이드	binary_crossentropy
임의 값에 대한 회귀	없음	mse
0과 1사이 값에 대한 회귀	시그모이드	mse 또는 binary_crossentropy

6. 몸집 키우기 : 과대적합 모델 구축

이제 모델이 충분히 성능을 내는지 질문해보아야 한다. 주어진 문제를 적절히 모델링하기에 충분한 층과 파라미터가 있는가? 예를 들어 2개의 유닛을 가진 하나의 은닉 층으로 구성된 네트워크가 있다고 가정하자. 이 네트워크가 MNIST 데이터셋에서 통계적 검정력을 가질 수 있지만 문제를 잘 해결하기에는 충분하지 않다. 머신 러닝은 최적화와 일반화 사이의 줄다리기이다. 과소용량과 과대용량의 경계에 적절히 위치한 모델이 이상적이다.

얼마나 큰 모델을 만들어야 하는지 알기 위해서 **과대적합된 모델을 만들어야 한다.**

1. 층을 추가한다.
2. 층의 크기를 키운다.
3. 더 많은 에포크 동안 훈련한다.

관심대상인 훈련과 검증 지표는 물론 항상 훈련 손실과 검증 손실을 모니터링 하자. **검증 데이터에서 모델 성능이 감소하기 시작했을 때 과대적합에 도달한 것이다.**

다음 단계에서 규제와 모델 튜닝을 시작하여 과소적합도 아니고 과대적합도 아닌 이상적인 모델에 가능한 가깝도록 만든다.

7. 모델 규제와 하이퍼파라미터 튜닝

이 단계가 대부분의 시간을 차지한다. 반복적으로 모델을 수정하고 훈련하고 검증 데이터에서 평가한다. 그리고 다시 수정하고 가능한 좋은 모델을 얻을 때까지 반복한다.

- 드롭아웃을 추가한다.
- 층을 추가하거나 제거해서 다른 구조를 시도해본다.
- L1이나 L2 또는 두 가지 모두 추가한다.
- 최적의 설정을 찾기 위해 하이퍼파라미터를 바꾸어 시도해본다.
- 선택적으로 특성 공학을 시도해본다. 새로운 특성을 추가하거나 유용하지 않을 것 같은 특성을 제거한다.

다음 사항을 유의하자 ! 검증 과정에서 얻은 피드백을 사용하여 모델을 튜닝할 때마다 검증 과정에 대한 정보를 모델에 누설하고 있다는 것이다. 몇 번만 반복하는 것은 큰 문제가 되지 않는다. 하지만 많이 반복하게 되면 결국 모델이 검증 과정에 과대적합될 것이다.

만족할 만한 모델 설정을 얻었다면 가용한 모든 데이터를 사용해서 제품에 투입할 최종 모델을 훈련시킨다. 그리고 마지막에 딱 한 번 테스트 세트에서 평가한다. 테스트 세트의 성능이 검증 데이터에서 측정한 것보다 많이 나쁘다면 검증과정에 전혀 신뢰성이 없거나 모델의 하이퍼파라미터를 튜닝하는 동안 검증 데이터에 과대적합된 것이다. 이 경우에는 좀 더 신뢰할 만한 평가 방법을 바꾸는 것이 좋다.

요약

- 주어진 문제와 훈련할 데이터를 정의한다. 이 데이터를 수집하고 필요하면 레이블을 태깅한다.
- 성공을 어떻게 측정할 지 선택하자. 검증 데이터에서 모니터링할 자료는 무엇인가?
- 평가 방법을 결정하자. 홀드아웃 검증이나 K-겹 교차 검증인가? 검증에 사용해야 할 데이터의 양은 얼마나 되는가?
- 단순한 랜덤 선택 모델보다 나은 통계적 검정력이 있는 첫 번째 모델을 만든다.
- 과대적합된 모델을 만든다.
- 검증 데이터의 성능에 기초하여 모델에 규제를 적용하고 하이퍼파라미터를 튜닝한다. 머신 러닝 연구의 대부분은 이 단계에 집중된다. 하지만 큰 그림을 놓치지 말자.
