딥러닝 2단계: 심층 신경망 성능 향상시키기

<머신러닝 어플리케이션 설정하기>

train/dev/test 세트

<신경망을 만들 때 결정해야하는 요인>

- 신경망 훈련 결정
- 몇 개의 층
- 은닉 유닛 몇 개
- 학습률
- 활성화 함수

⇨ 반복적 과정으로 파라미터를 고쳐가면서 특정 데이터에 맞는 신경망 만든다!

·○ nlp, 구조화된 데이터, 웹검색, 컴퓨터 보안, 물건배송 에 딥러닝이 적용된다.

·○ 가지고 있는 데이터 양, 입력특성 개수, 컴퓨터 설정 등의 요인으로 결정한다.

⇨ 데이터세트 잘 설정하는 것 중요하다!

train data를 다 가져와서

·○ train set / hold-out cross validation set(development set) / test set

어떤 모델이 가장 좋은 성능을 내는지 확인하여 최종 모델에 적용한다.

·○ 일반적으로 70 train /30 test or 60 test / 20 development setv/ 20 test set

★ **dev set:** 서로 다른 알고리즘을 시험하고 어떤 알고리즘이 성능이 좋은지 확인할 때 쓰인다.

★ **test set :** 최종 분류기가 어느 정도 성능인지 신뢰있는 추정치를 제공한다.

·○ 큰데이터 세트면 개발 테스트를 20 or 10퍼센트보다 작게 설정해도 ok

<mismatched train/test distribution>

- training set : 인터넷에서 가져온 고양이 사진
- test set: 사용자에 의해 업로드 된 고양이 사진

⇨ 두 분포가 다를 수 있음

★ 개발과 테스트 세트가 같은 분포에서 와야한다!

(개발 세트를 이용해 다양한 모델을 평가해야하기 때문이다.)

★ 테스트 데이터는 없어도 된다. (비편향 추정에 대한 정보가 필요없는 경우)

★ 모든 테스트 세트를 훈련세트에서 훈련시키고, 다른 모델 아키텍트 시도 후 개발세트에서 평가한다.

=> 이 과정 반복해서 좋은 모델을 찾는다.

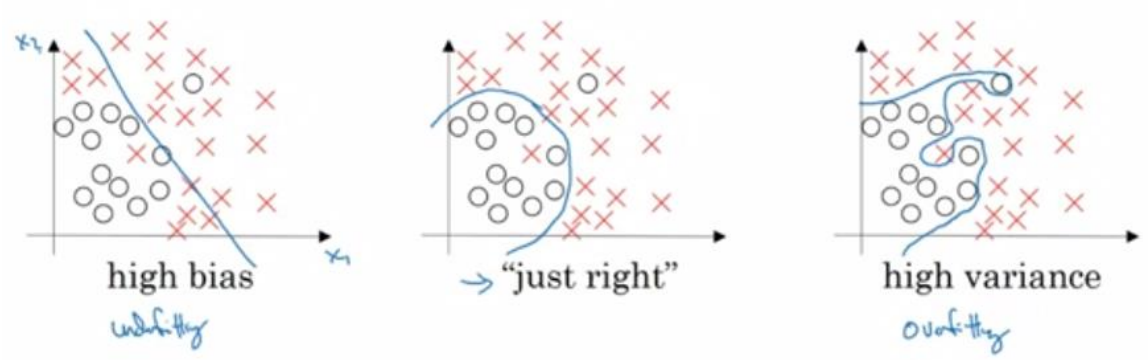
★ 훈련과 개발세트만 있는 경우 개발 세트를 테스트 세트로 한다.

테스트 세트를 교차검증 세트로 이용하는 것 but 테스트 세트에 과적합이라서 좋은 용어는 X

<왜 데이터를 나눌까?>

1. 반복을 더 빠르게 할 수 있다.
2. 편향과 분산을 효율적을 측정할 수 있다.
3. 개선하는 효율적 방법을 더 알기 쉽다.

편향 / 분산



- 데이터의 과소적합: 높은 편향값의 클래스
- 데이터의 과대적합: 높은 분산의 클래스

인간은 대략 0%의 오차를 낸다고 가정

<훈련세트 오차 1%, 개발세트 오차 11%일 때>

훈련세트에 과대적합되어서 교차검증세트에서 일반화 X

⇨ 높은 분산

<훈련세트 오차 15%, 개발세트 오차 16%일 때>

훈련데이터에 대해서도 잘 맞지 않으면 데이터의 과소적합

⇨ 높은 편향

<훈련세트 오차 15%, 개발세트 오차 30%일 때>

⇨ 높은 편향 & 높은 분산

<훈련세트 오차 0.5%, 개발세트 오차 1%일 때>

⇨ 낮은 편향 & 낮은 분산

★ 최적의 오차(베이지안 오차)가 거의 0이라는 가정하에 성립한다!

★ 최적의 오차가 높을 경우(15%)에는 훈련세트 오차가 15% 인 것은 합당하다.

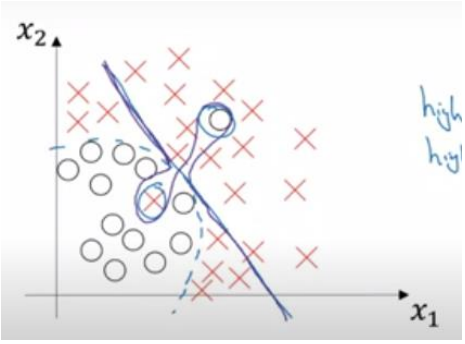
⇨ 어떤 분류기도 잘 작동하지 않을 경우 편향과 분산을 분석해야한다.

⇨ 훈련세트 오차를 확인함으로서 편향문제를 알 수 있다.

⇨ 훈련세트에서 개발세트로 갈 때 오차의 크기로 분산 문제 알 수 있다.

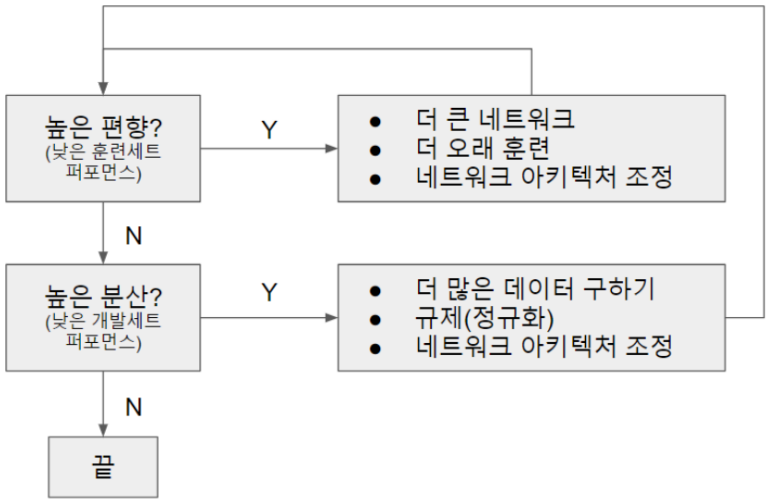
훈련세트, 테스트세트가 같은 확률분포이고, 최적 오차가 0에 가깝다는 가정 전제

<높은 분산 높은 편향일 때>



- 거의 선형이고 데이터에 과소적합한 선
- 일부데이터에는 과도적합

머신러닝을 위한 기본 레시피



1. 알고리즘이 높은 편향을 가지는지, 훈련세트 훈련데이터의 성능을 본다.
2. **높은 편향**을 가져서 훈련세트에 잘 맞지 않으면:

⇨ 더 많은 은닉층 은닉 유닛 or 오랜시간 훈련시키기 or (다양한 신경망 아키텍처 찾기)

베이즈 오차가 높지 않다면 더 크게 훈련하는 경우 최소한 훈련세트에 대해서는 잘 맞을 것이다.

네트워크 크기는 클수록 좋다 but 계산 시간만 단점이다.

3. 편향을 수용가능한 크기로 줄이고 나면 분산 문제를 확인하기 위해 개발세트 성능을 본다.

4. 높을 분산일 때:

⇨ 데이터 더 가져오기 or 정규화 시도 or (다른 아키텍처 찾기)

5. 낮은 편향과 낮은 분산 찾을 때까지 반복

<편향분산 트레이드 오프>

편향 증가 분산 감소 or 편향 감소 분산 증가

- 위에서 언급한 방법들은 편향만을 감소 or 분산만을 감소하여 서로 영향 주지 않는다.
- 지도학습에 딥러닝이 유용한 이유: trade off가 적다!