3. Structuring Machine Learning Projects

Week	Week2

Error Analysys 오류 분석

Carrying Out Error Analysis

오류 분석을 위해서는 먼저 개발 세트에서 잘못 표기된 예시를 찾고 이렇게 잘못 표기된 예시에서 거짓 양성과 거짓 음성을 찾습니다.

그래서 다양한 카테고리에 대한 오류의 총 개수를 각각 카테고리별로 찾아냅니다.

이 과정에서 새 오류 카테고리를 생성해야겠다는 생각이 들 수 있다.

Cleaning Up Incorrectly Labeled Data

오류가 10%인 분류기가 있을때 잘못된 라벨링으로 인한 오류가 0.6% 이고 나머지 이유로 9.4%의 오류가 발생하면 라벨링을 고치는 것보다 다른 요인을 고쳐서 오류를 줄이는게 더 합리적이다.

Correcting incorrect det,test set exmaples 라벨링을 고칠때의 가이드라인과 원리

- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got right as well as ones it got wrong
- Train and dev,test data may now come from slightly different distributions

Build your First System Quickly, then Iterate

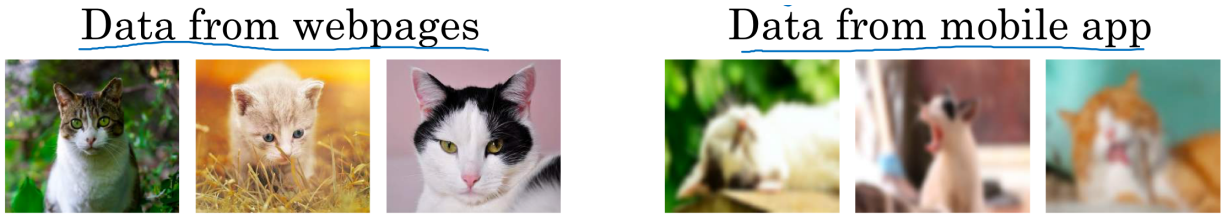
- Set up dev/test set and metric
- Build initial system quickly

- Use Bias/Variance analysis & Error analysis to prioritize next step

첫번째 시스템을 빨리 만들고 그 다음에 계속 반복해라. 지저분해도 일단 만들고 우선순위 정해서 하는게 좋다.

Mismatched Training and Dev/Test Set

Training and Testing on Different Distributions



웹패이지에서 얻은 고해상도 20만장과 모바일앱에서 얻은 저해상도 만장의 이미지가 있다고 해보자.

우리가 원하는건 모바일 앱에서 잘 작동하는 것이다.

이럴때 데이터 분포를 어떻게 나눠서 학습을 해야 할까?

Option 1) 전체 이미지 21만장을 나눠서 205000 : 2500 : 2500 으로 나눈다.

장점 : train, dev, test 세트가 모두 같은 분포에서 와서 관리가 쉽다.

단점 : 2500개의 dev 세트에서 상당수가 웹페이지 이미지 분포에서 온다. → 우리가 원하는 건 모바일에서 잘 작동하는건데 → 이게 치명적인 단점임.

우리가 원하는 바를 이루기 위한 목표 설정

Option 2) training 을 웹페이지에서 온 20만장의 사진으로만 설정. 모바일 앱의 5천장정도는 training data에 넣어도 ㄱㅊ

이제 남은 5000장의 모바일 앱 이미지를 dev와 test에 2500장씩 나눠서 모델링.

→ 우리가 진짜로 원하는 목적인 모바일에서 잘 분류하는 모델을 만들 수 있다.

단점은 각 데이터의 분포가 다르다는 건데 장기적으로 더 좋은 성능을 낼 수 있다.

Bias and Variance with Mismatched Data Distributions

train data가 dev/test data와 다른 분포에서 오는 경우 편향과 분산을 분석하는 방법이 달라진다.

위의 예시에서 고해상도 고양이 이미지를 train data로 학습하고 저해상도 고양이 이미지를 dev set로 학습했을 때 dev set에서 오류가 더 컸다면 이게 문제처럼 보일 수 있는데 사실 데이터 분포가 다르기 때문에 10%의 오류가 어쩌면 dev set에서 잘 하는 걸 수도 있다.

→ 분산 문제가 없는 걸 수도 있음.

train set를 무작위로 섞은 뒤 일부 train set를 추출해 이를 train-dev set라고 하자.

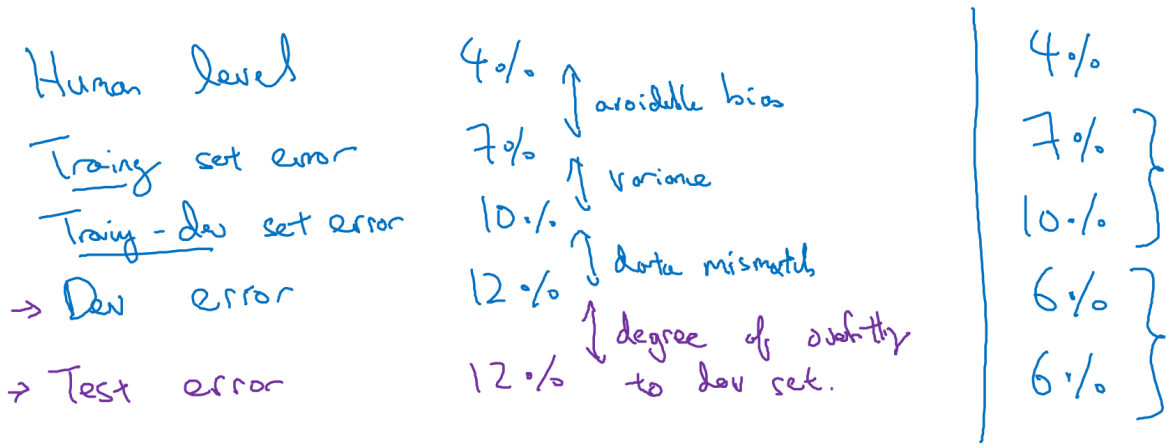
= train 과 train-dev 가 같은 분포로 되어있을 것이다.

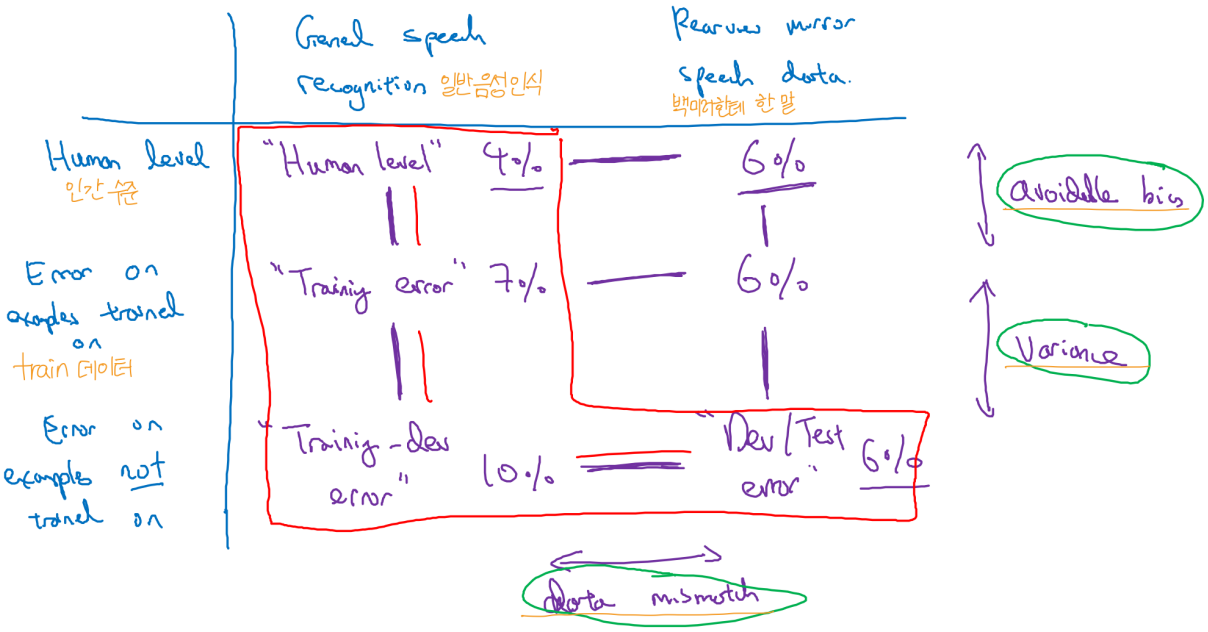
학습을 train에서만 진행하고 train-dev 에서는 진행하지 않는다.

이제 분포가 비슷한 상황에서 train, train-dev set의 오류 차이를 본다.

여기서도 오류가 크다면 이제서야 variance 문제가 있음을 알 수 있따.

다른 예시) train error는 1% train-dev error는 1.5% dev error는 10%라고 한다면 low variance문제. → 전형적인 데이터 불일치 문제





Addressing Data Mismatch

- Carry out manual error analysis to try to understand difference between training and dev/test sets
- Make training data more similar; or collect more data similar to dev/test sets
    - artificial data synthesis

데이터 불일치 문제가 있다고 생각될 경우 오류 분석을 할 것을 권합니다

훈련 세트나 개발 세트를 보고 이 데이터 분포가 어떻게 다를 수 있는지에 대한 통찰력을 얻어 보세요

그런 다음 개발 세트와 더 비슷해 보이는 훈련 데이터를 더 많이 얻을 수 있는 방법을 찾아봅니다

우리가 논의한 방법 중 하나는 인공 데이터 합성인데요 인공 데이터 합성은 실제로 효과가 있습니다 저는 음성 인식에서 인공 데이터 합성이 이미 아주 우수한 음성 인식 시스템의 성능을 크게 향상시키는 것을 보았습니다 이렇게 인공 데이터 합성은 아주 효과적일 수 있습니다 한편 인공 데이터 합성을 사용할 때에는 데이터를 전체 예시 집합의 극히 일부분에서 시뮬레이션하고 있지는 않은지 주의하세요

Learning from Multiple Tasks

Transfer learning

pre-training

fine-tuning : pre-training에서 학습한 모델의 가중치 변경하는거

전이학습이 필요한 경우 → 출발지점에 문제관련데이터가 많고, 도착 지점에는 데이터가 없을 때

반대의 경우에는 필요없음.

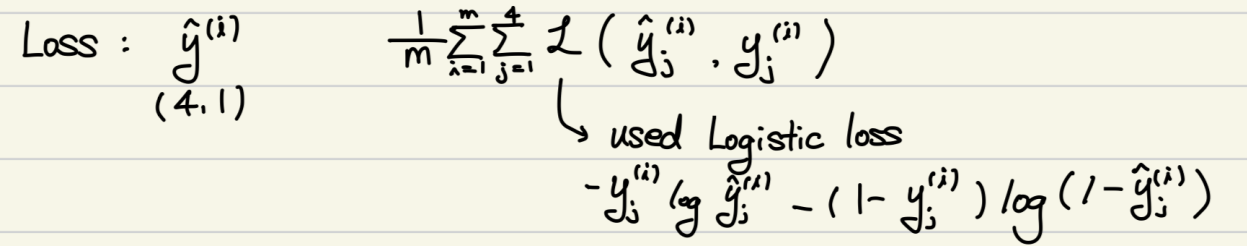
A에서의 작업을 B로 옮기는 경우

- Task A and B have the same input X
    - same input x는 데이터 분포가 같다는 의미일까?
- You have a lot more data for Task A than Task B
- Low level features from A could be helpful for learning B

Multi-task Learning

신경망이 여러가지 일을 동시에 할 수 있게 만드는 것.

자율주행차는 보행자, 신호등, 다른 자동차, 신호표시판 등 여러가지 처리해야함.



소프트맥스는 단일 라벨을 부여하는데 여러 라벨을 가질 수 있음.

multi-task learning이 쓰이는 경우

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually : Amount of data you have for each task is quite similar
- Can train a big enough neural network to do well on all the tasks.

multi-task learning보다 transfer learning이 더 자주 쓰임

예외적인 분야는 컴퓨터 비전 객체 탐지인데요 서로 다른 많은 객체를 감지하기 위해 신경망을 훈련하는 경우입니다 이 경우 개별 신경망 훈련보다 더 잘 작동합니다 하지만 전이 학습과 다중 작업학습이 비슷하게 보이는 경우가 많다고 해도 실제로는 다중작업학습보다 전이 학습의 적용 사례를 훨씬 더 많이 봤습니다 제 생각에는 각각의 다른 작업들을 일일히 세팅하거나 찾는 일이 굉장히 어렵기 때문에 그런 것 같습니다

End-to-end Deep Learning

What is End-to-end Deep Learning?

input x 에서 output y를 뽑는거

데이터가 많이 필요

Whether to use End-to-end Deep Learning

장점

- Let the data speak 데이터 순수하게 반영 가능
- Less hand-designing of components needed

단점

- May need large amount of data
- Excludes potentially useful hand-designed components 직접만든 유용한 요소들을 배제함

Applying ene-to-end deep learning

Key question : Do you have sufficient data to learn a function of the complexity needed to map x to y?