Structuring ML Projects

[1-1. Introduction to ML Strategy]

<Orthogonalization>

- 머신러닝의 성능을 높이기 위해 할 수 있는 행동들이 많은데, 머신러닝을 효과적으로 구축하는 사람들은 이런 선택에 대한 뛰어난 안목을 가지고 있음
  ⇒ 원하는 하나의 효과를 얻기 위해 변수를 조정하는 것을 직교화(= orthogonalization)라고 함

* Chain of assumption in ML

- Fit training set well on cost function (≒human level performance) → network 조절
- Fit dev set well on cost function → regularization, bigger train set 학습 알고리즘 변경
- Fit test set well on cost function → bigger dev set
- Performs well in real world → change dev set or cost function

[1-2. Setting up your Goal]

<Single Number Evaluation Metric>

- Precision(=정밀도) : 모델이 분류한 정답 중 진짜 정답이 얼마나 있는지 측정
- Recall(=재현율) : 실제 정답 중 모델이 정답을 얼마나 분류했는지 측정

↳ precision과 recall은 trade-off 관계이므로, 둘의 조화평균인 F-1 Score 사용

$$F1\ Score = \frac{2}{1/Precision + 1/Recall}$$

dev set에서 F1 score을 사용하게 되면, 더 빠른 모델 선택이 가능하여 알고리즘 개선의 순환속도 ↑

* Other Example

- 각각의 국가가 오류율(=error)을 나타낸다고 하면

Algorithm	US	China	India	Other	Average
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5%
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

→ average error가 가장 낮은 값을 채택하는 것이 합리적임!!

<Satisficing and Optimizing Metric>

- 성능을 최대로 높이고 싶은 것은 Optimizing Metric으로, 중요한 목표는 satisficing Metric으로 설정

ex)

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

→ Maximizing accuracy,
subject to running time ≤ 100ms

⇒ accuracy : Optimizing metric
running time : satisficing metric

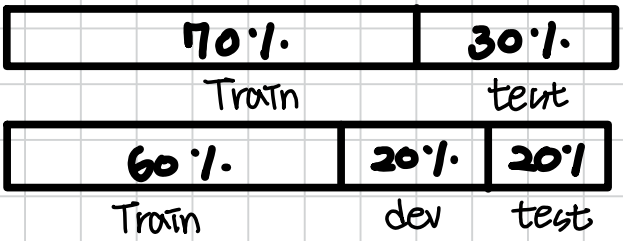
< Train / Dev / Test Distributions >

- dev sets와 test sets의 분포가 같도록 하는 것이 권장됨.
  → randomly shuffled data를 dev / test 로 split
- Guideline : Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on

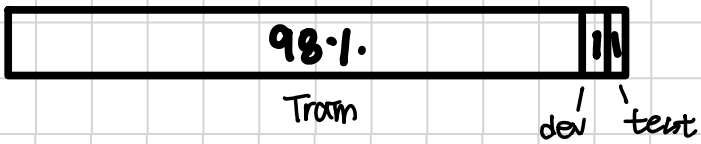
  ※ same distribution ※

< Size of the Dev and Test sets >

* Old way of splitting data (= small data sets)



* Modern way of splitting data (= large data sets)



< When to Change Dev / Test Sets and Metrics? >

* 머신러닝의 단계 ↳ 평가지표가 알고리즘 선호에 도움이 안된다면 새로운 지표를 고려해야 함

① 모델을 평가할 적절한 척도 설정
② 해당 척도를 기준으로 좋은 성능을 이끌어내기

* Cat dataset examples

- Metrics : classification error
- Algorithm A : 3% error + pornographic
- Algorithm B : 5% error

⇒ 알고리즘 입장에서는 A가 더 좋지만, 안좋은 사진을 보여준다는 점에서 B가 더 좋다. 해결방법 ①, ②

해결방법 ① 평가척도 변경

$$\text{Error} : \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} I\{y_{pred}^{(i)} \neq y^{(i)}\}$$

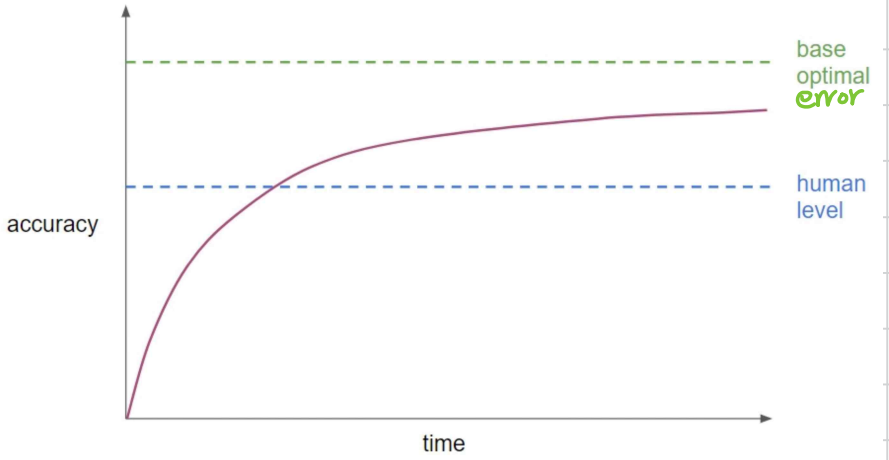
가중치 부여 → $\frac{1}{\sum w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} I\{y_{pred}^{(i)} \neq y^{(i)}\}$, $w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is non-porno} \\ 10 & \text{if } x^{(i)} \text{ is porno} \end{cases}$

해결방법 ② : 실제 data로 dev / test set 변경

ex) dev / test 에서는 고화질 데이터를 사용하고, 실제 앱에서는 흐릿한 사진을 사용한다면
→ dev / test 에서도 흐릿한 사진으로 학습하는게 더 좋음!

[1-3. Comparing to Human-level Performance]

<Why Human-level Performance?>



: 이론상 가능한 최적의 오차

- ML은 human-level 까지는 학습이 빠르게 잘 되다가, 그 이후로는 속도가 느려진다.
  - → human-level performance와 base optimal error가 크게 차이가 없기 때문.
  - → human-level에 도달하기 이전에는 많은 기법을 사용하여 성능을 높일 수 있지만, 이상이 되면 이런 기법을 사용하기 어려워진다.

<Avoidable Bias>

ex) Cat classification example

Human error(≈base)	1%	7.5%
Training error	8% (7%)	8% (0.5%)
Dev error	10% (2%)	10% (2%)
	↓ focus on bias	↓ focus on variance

→ human-level error ≈ base optimal error

- Avoidable Bias : Base error 와 training error의 차
  값이 클수록 아직 훈련이 덜 된 것으로, 더 깊은 학습 실행
- Variance : traing error 와 dev error의 차

<Understanding Human-level Performance>

Medical image classification example:

Suppose:

(a) Typical human ..... 3 % error

(b) Typical doctor ..... 1 % error

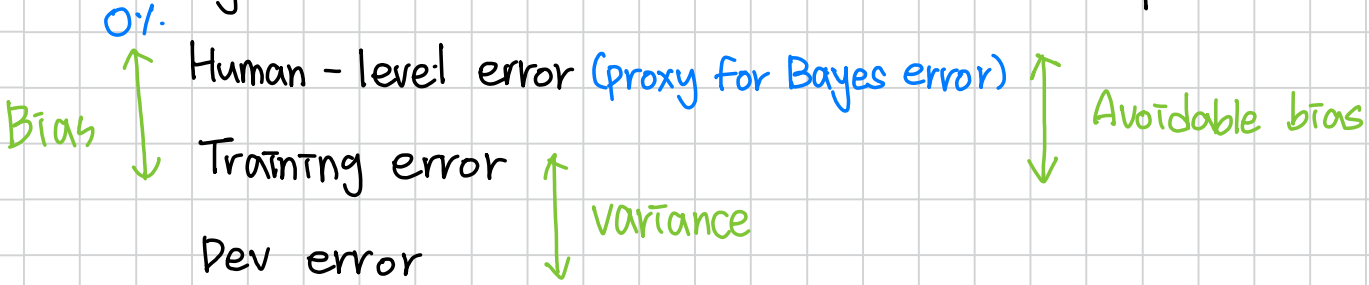
(c) Experienced doctor ..... 0.7 % error

(d) Team of experienced doctors .. 0.5 % error

→ TF) 어느 분야에서 특정 인간의 역량을 능가하는 시스템을 도입할 때 적합

→ a proxy for Bayes error

*Summary of bias / variance with human-level performance



<Surpassing Human-level Performance>

Problems where ML significantly surpasses human-level performance

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

<공통점>
- structured data. Not natured perception
- Lots of data

<Improving your Model Performance>

- 지도 학습 알고리즘이 잘 작동할 수 있도록 한다는 것은 아래의 두 과정을 거칩니다.
  - 첫째, 훈련세트에 잘 들어 맞아야합니다. 즉, 회피 가능 편향을 줄이는 것입니다.
  - 둘째, 개발 및 시험 세트에서도 좋은 성능을 내도록 일반화합니다. 즉, 분산이 낮아야합니다.
- 직교화를 떠올려 보시면, 몇가지 방법으로 회피 가능 편향과 분산을 각각 줄일 수 있습니다.
  - 회피 가능 편향:
    - 더 큰 모델로 훈련시킵니다.
    - 훈련을 더 오래 시키거나 더 나은 알고리즘으로 최적화를 합니다. ex) momentum, RMSProp, Adam
    - 다른 신경망 구조를 만들거나 최적의 하이퍼파라미터를 찾습니다. ex) RNN, CNN
  - 분산
    - 더 많은 데이터를 사용합니다.
    - 정규화를 진행합니다.
    - 다른 신경망 구조를 만들거나 최적의 하이퍼파라미터를 찾습니다.