13강: Debugging ML Models and Error Analysis

Diagnostics for debugging learning algorithms

제약조건이 있는 로지스틱 회귀를 이용한 스팸 분류 모형을 만든다고 가정해보자.

(cf. Bayesian Logistic Regression)

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda \|\theta\|^2$$

위의 수식을 만족하게끔 gradient ascent를 이용해 모형을 얻었는데,
20% 정도 되는 test error를 얻었으면 우린 어떡하면 좋을까?
~~노트북 닫고 때려쳐야지~~

우리는 다음과 같은 방법들을 이용해 우리의 알고리즘을 개선시키고자 할 것이다.

- training examples 더 가져오기
- feature 개수 줄이기
- feature 개수 늘리기
- feature 바꿔보기 : 이메일 제목 vs. 이메일 본문
- 경사하강법의 반복횟수(# of iterations) 늘려보기
- Newton's Method 시도해보기
- λ값을 바꿔보기
- SVM 써보기

Variance vs. Bias

> Bias-variance is the single most powerful tool for analyzing the performance of a learning algorithm.

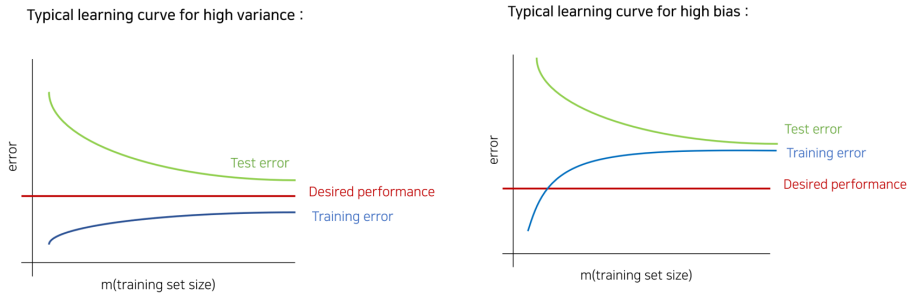
가장 기본적이면서 효과적인 진단 방법은 variance/bias를 살펴보는 것이다.

주로 모델의 성능이 좋지 않으면 오버피팅/ 언더피팅일 가능성이 높다.

- 오버피팅 : variance가 높은 경우 (새로운 데이터를 잘 예측하지 못하는 경우)
  - 훈련에서는 잘 맞추는데 테스트에서는 못 맞추는 경우
- 언더피팅 : bias가 높은 경우 (기존의 데이터도 잘 설명하지 못하는 경우)
  - 예) 너무 작은 수의 피처가 사용되면 bias가 클 확률이 높아짐
  - training error 와 test error 모두 높은 경우

알고리즘을 개선시키는 제일 기본적이고 확실한 방법은 데이터의 개수를 늘리는 방법일것이다.
(Data is everywhere but very far away from me ^^)

데이터의 개수 m에 따른 모델 error의 변화(learning curve)를 살펴보자



왼쪽의 learning curve는 모델의 variance가 높은 경우이다.

- train set을 이용하여 모델을 학습시킬 때에는 원하는 수준의 학습에 도달
- 테스트에서는 원하는 수준보다 에러 값이 높은 상태이다.
- 데이터의 개수가 늘어남에 따라 테스트 에러 값이 낮아지고 있음 → 데이터 개수를 늘리는 것이 도움이 됨

오른쪽의 learning curve는 모델의 bias가 높은 경우이다.

- train error / test error 둘 다 원하는 수준에 도달하지 않음

→ 많은 데이터 사례를 수집해서 훈련시키는 것은 도움이 되지 않음

💡 새로운 모델을 만들었어요! 뭘해봐야할까요? 🤔

→ 지저분한 코드라도 빠르게 만들어서 돌려본다 (quick and dirty code)
→ Bias /Variance 진단을 해보자

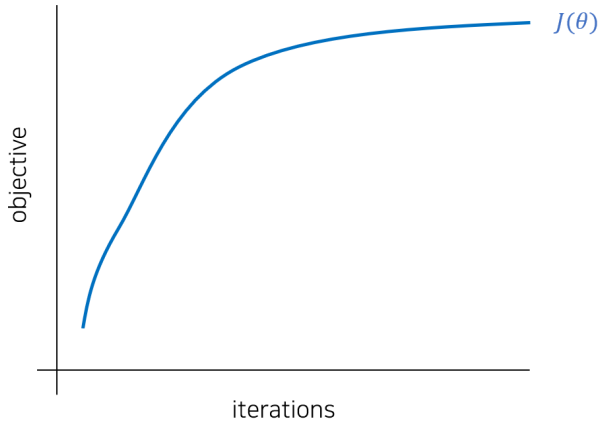
이러한 Bias / Variance 문제가 가장 흔하고 중요한 진단 방법이고,
그 외의 문제에 대한 진단은 알고리즘을 만든 자의 창의성에 달려있다.

이를테면, 다음과 같은 문제 예시가 있다고 해보자.

> ***“ SVM 쓰기 싫어 ! 그래도… Logistic Regression!”***
>
> - 로지스틱 회귀 : 스팸 메일에 대해 2%의 에러율 / 정상메일에 대해 2%의 에러율
>
> ~~(안 돼 돌아가 채용 제안 메일을 스팸 분류하면 어쩔려고??)~~
>
> - SVM : 스팸 메일에 대해 10%의 에러율 / 정상메일에 대해 0.01%의 에러율
>
> ~~너는 합격쓰 ☆~~
>
> 위의 두 모델 성능 평가는 정상메일 분류에 더 가중치를 둔다. (weighted accuracy)
> 이를 수식으로 표현하면 다음과 같다.
>
> $$a(\theta) = \max_{\theta} \sum_{i} w^{(i)} I\{h_\theta(x^{(i)}) = y^{(i)}\}$$
> $$a(\theta_{SVM}) > a(\theta_{BLR})$$
> • BLR=Bayesian Logistic Regression
>
> - 하지만 계산의 효율성을 위해 로지스틱 회귀를 쓰고 싶은 경우에는 어떡하면 좋을까? 🤔

Optimization algorithm diagnostics

다른 일반적인 질문은 ‘알고리즘이 수렴하고 있는가?’이다.



일반적으로 훈련의 반복 횟수를 늘리다보면 , 대개의 objective function $J(\theta)$은 수렴하는데,
이를 위해 얼마나 반복해야하는걸까? 반복을 많이하는 것은 효과적인걸까?

알고리즘의 최적화 문제를 진단하기 위해서, 우리는 다음과 같은 질문을 할 수 있다.

- 알고리즘이 수렴하는가 ?
- 올바른 목적함수를 이용해 최적화를 하고 있는가?
- 하이퍼파라미터가 적절한가?
  - $J(\theta) = \max_\theta \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}, \theta) - \lambda\|\theta\|^2$에서 λ
  - $J(\theta) = \min_{w,b} \|w\|^2 + C\sum_{i=1}^m \xi_i$ 에서 C

즉, 최적화 알고리즘의 문제를 진단하기 위해서는

1. 최적화 알고리즘 그 자체가 잘못돼서 수렴하지 못하는 것인지
2. cost function
$J(\theta)$를 잘못 만든 것인지

확인할 필요가 있다.

이러한 진단은 앞서 언급했던 정확도를 나타내는 함수 $a(\theta)$와 cost function $J(\theta)$를 가지고 할 수 있다.

CASE 1.

$$a(\theta_{SVM}) > a(\theta_{BLR})$$
$$J(\theta_{SVM}) > J(\theta_{BLR})$$

$J(\theta_{BLR}) = \max_\theta \sum_{i=1}^m \log p(y^{(i)}|x^{(i)}, \theta) - \lambda\|\theta\|^2$라는 것, 즉 **최대화**의 문제임을 고려할 때 ,
이는
θ_{BLR}이 목적함수 J를 최대화하지 못했음을 의미한다.

이 경우, 모델의 문제는 알고리즘의 수렴 그 자체에 있다.

즉, **최적화 알고리즘 자체에 문제가 있는 것**이다.

CASE 2.

$$a(\theta_{SVM}) > a(\theta_{BLR})$$
$$J(\theta_{SVM}) \leq J(\theta_{BLR})$$

위의 부등식에서 , θ_{BLR}이 목적함수 J를 최대화했음을 알 수 있다.

하지만 여전히 SVM의 정확도 값이 더 높다.

이는 $J(\theta)$가 최대화해야하는 함수가 아닌 잘못된 함수임을 의미한다.

즉, **최적화 문제를 위한 목적함수에 문제가 있는 것**이다.

우리는 모델 진단을 variance/bias, 그리고 optimization의 관점으로 살펴보았다.

그러한 관점으로 모델의 개선 방안을 다시 정리해보자.

- training examples 더 가져오기 — **Fixes high variance**
- feature 개수 줄이기 — **Fixes high variance**
- feature 개수 늘리기 — **Fixes high bias**
- 이메일 제목을 feature로 써보기 — **Fixes high bias**
- 경사하강법의 반복회수(# of iterations) 늘려보기 — **Fixes optimization algorithm**
- Newton's Method 시도해보기 — **Fixes optimization algorithm**
- λ값을 바꿔보기 — **Fixes optimization objective**
- SVM 써보기 — **Fixes optimization objective**

Debugging an RL algorithm

RL = reinforcement learning

하늘을 나는 멋진 자율비행 ~~바둑기~~ 헬리콥터를 생각해보자.

헬리콥터의 알고리즘은 다음과 같다.

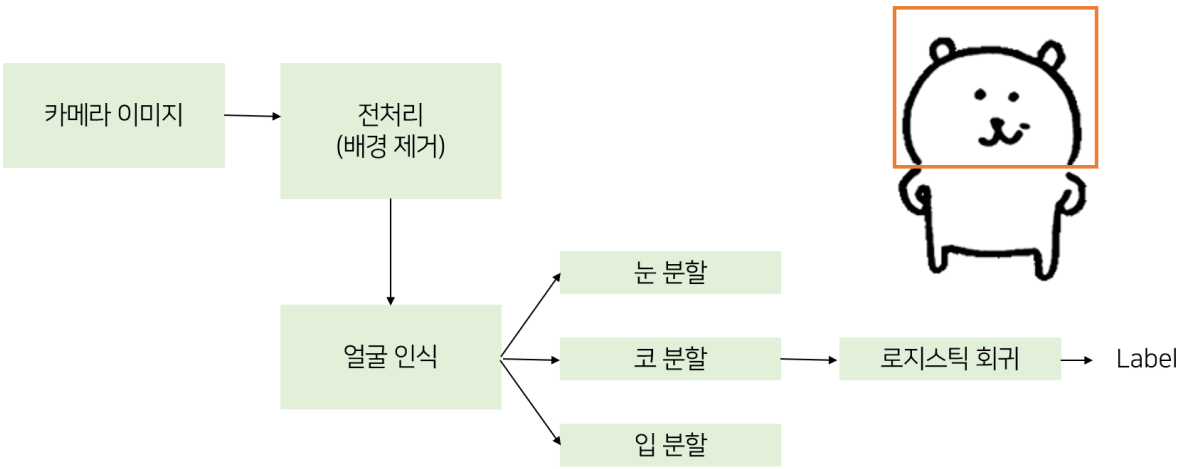
1. 헬리콥터의 시뮬레이터를 구축한다.
2. cost function을 설정한다. $J(\theta) = \|x - x_{desired}\|^2$, x는 헬리콥터의 위치
3. 시뮬레이션에서 RL 알고리즘을 수행해 $\theta_{RL} = \arg\min_\theta J(\theta)$를 찾는다.

이런! θ_{RL} 값에 따라 작동하는 헬리콥터가 형편 없다! 🥲
위의 3단계에서 시뮬레이터, $J(\theta)$, θ_{RL}에 문제가 없다면 헬리콥터는 잘 작동해야한다.

1. 시뮬레이션에서는 문제없이 잘 작동했는데 현실에서는 그렇지 않다 → 시뮬레이터의 문제
2. θ_{human}=인간이 움직이는 방식, $J(\theta_{human}) < J(\theta_{RL})$→ 알고리즘이 수렴에 실패했음을 의미한다.
3. $J(\theta_{human}) \geq J(\theta_{RL})$ → cost function의 설정이 잘못되었음

Error analysis

얼굴인식 시스템을 생각해보자.
많은 application에서는 다양한 learning algorithm을 결합해 pipeline을 만든다.



Error analysis

Component	Accuracy
Overall	85%
Preprocess (remove background)	85.1%
Face detection	91%
Eyes segementation	95%
Nose segementation	96%
Mouth segementation	97%
Logistic Regression	100%

- 파이프라인의 단계별로 각 단계가 완벽하게 수행됐을때의 정확도의 증분을 살펴본다.

  예. 배경제거의 경우 포토샵으로 완벽하게 제거

- 얼굴 인식 모델의 경우 얼굴 인식→ 눈 분할에서 제일 모델 개선의 여지가 큰 것을 분석 가능

Ablative analysis

파이프라인의 구성 요소가 최종 성능에 얼마나 기여했는지를 측정하는 방법

Component	Accuracy
Overall system	99.9%
Spelling correction	99%
Sender host features	98.9%
Email header features	98.9%
Email text parser features	95%
Javascript parser	94.5%
Features from images	94.0%

- 단순 선형 회귀 모델을 통해서 94%의 성능이 나옴
  → 99.9%로 모델 성능이 모델이 향상
- 각 구성 요소를 순서대로 제거
  → 얼마나 정확도가 감소하는지 분석
- Email text parser features가 성능향상에 제일 많이 기여함을 알 수 있음