[1주차] ML Stretegy(1)

무엇을 튜닝할 것인지 굉장히 뚜렷한 안목을 가지고 있습니다. 하나의 효과를 얻어내기 위해서 말이죠. 이러한 절차를 orthogonalization (직교화)이라고 부릅니다.

orthogonal (직교)한 특성을 갖게 되면, 직교는 서로 직각으로 구성됨을 뜻합니다. 실제로 제어하려는 것과 이상적으로 정렬된 직교화 컨트롤을 사용하면 조종해야 하는 손잡이를 훨씬 더 쉽게 조종할 수 있습니다. 각도, 악셀 및 브레이킹을 자동차가 원하는 방향으로 조절하기 더 수월해집니다. 이런 것들이 어떻게 머신러닝과 연결될까요?

지도학습이 잘 운영되기 위해선 보통 본인의 시스템 손잡이를 튜닝하여 반드시 4가지가 잘 유지되도록 해야 합니다. 첫 번째로, 적어도 트레이닝 세트에서는 잘 작동하도록 해야 합니다. 따라서, 훈련세트에서의 성능은 어느 정도 수용성 평가를 통과해야 합니다. 특정 어플에서는, 이 성능 레벨이 인간과 준하는 것을 뜻하기도 하는데요, 이것은 어플마다 다를 것입니다.

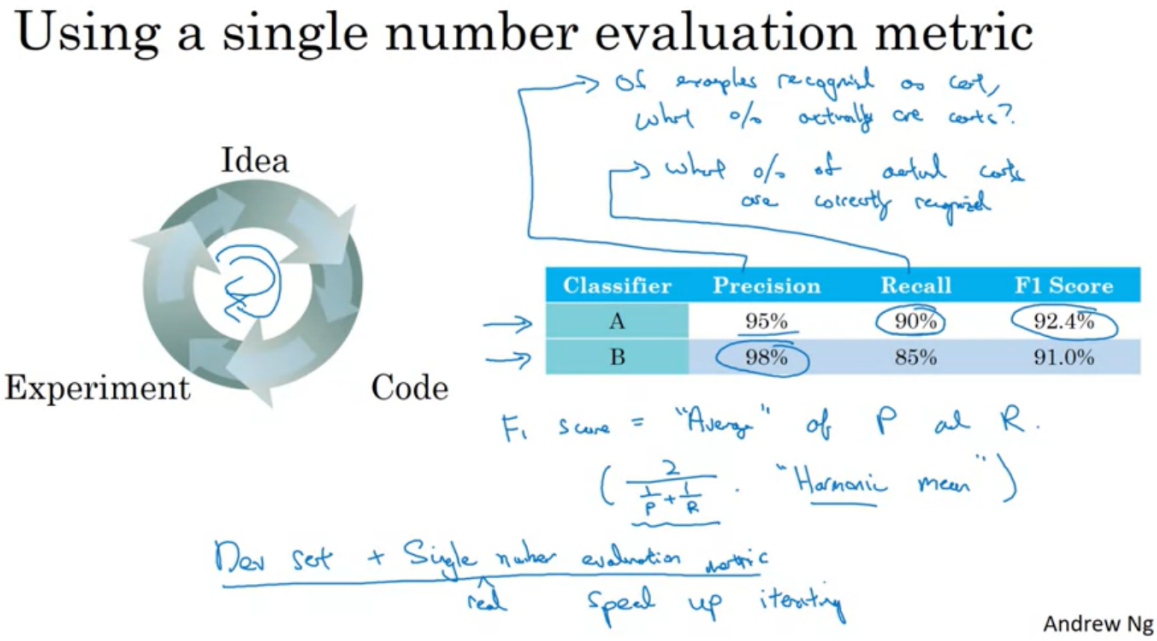
트레이닝 세트에서 잘 구현된 경우, 그 다음으로는 dev set에서도 잘 구현되길 바라셔야 합니다. 그 이후로는, 테스트 세트에서도 잘 되야 하구요. 마지막으로는 비용함수가 적용된 테스트 세트가 잘 되어 실제로도 잘 작동하는 시스템이 되도록 해야 합니다.

만약 classifier A가 95퍼센트의 정확도를 보이면, classifier A는 고양이가 맞는다고 올바르게 판별하는 확률이 95 퍼센트라는 뜻입니다. 재현율은 고양이 이미지에서 실제로 얼마나 classifier를 통해 올바르게 인식이 되었는가? 를 뜻합니다. 실제 몇 퍼센트의 고양이가 올바르게 분류가 되었는가?

:2:4부터 동영상을 재생하고 스크립트를 따라가세요.2:04

그렇다면 만약 classifier A가 90 퍼센트 재현율 이라고 하면 모든 dev set의 이미지에서 실제로 고양이인 이미지가 90 퍼센트 정확도로 인식했다는 뜻입니다.

그래서 classifier를 고르는 과정에서 제가 추천 드리는 것은 두 가지의 숫자, 정밀도, 재현율을 사용하기보다는 정밀도와 재현율을 결합시킨 새로운 평가 지표를 찾는 것입니다.



머신러닝 세계에서는, 정밀도와 재현율을 결합시키는데 있어 F1 score라는 것을 쓰는 게 정석입니다. F1 score의 상세내용은 그리 중요하지 않습니다, 그렇지만 대략적으로 P라는 정밀도 값과 R이라는 재현율값의 평균수치라고 생각하면 됩니다. 공식적으로는, F1 score가 식으로 정도 되는데요. 공식은 2/ 1/P + 1/R입니다. 수학에서는 이 공식을 Harmonic mean of Precision P and recall R이라고 합니다. 비공식적으로는, 정밀도와 재현율의 평균값을 구하는 것이라고 생각하시면 편합니다.

여러 머신러닝 팀을 보면서 느낀 것은 많은 팀들이 잘 정의된 dev set를 갖추었는데, 즉, 정밀도와 재현율을 잘 측정하도록 구축한 경우를 말하는데요, single number 평가 지표와 함께 사용하는 것을 보았습니다. 가끔씩 single row number라고 표현하기도 합니다.

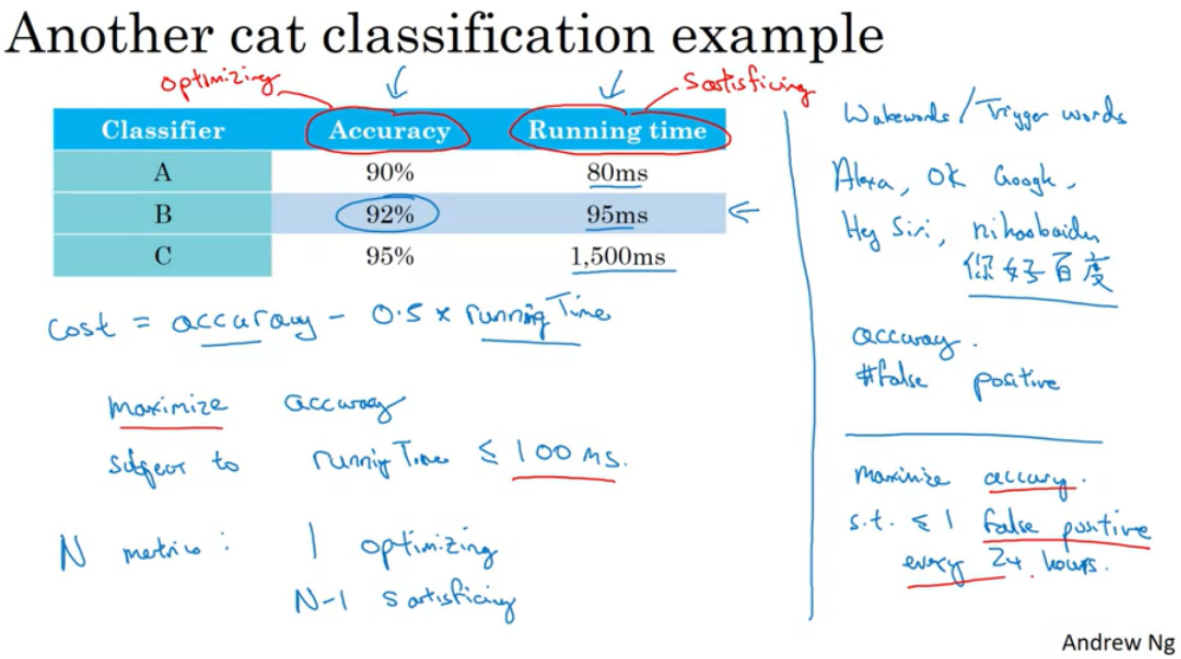
평가 지표는 classifier A 또는 classifier B가 더 좋은지 빠르게 판단할 수 있게 해주고, 결과적으로 dev set와 single number 평가 지표를 구축하여 반복하는데 속도를 높일 수 있습니다.

Another example

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%

Andrew Ng

이러한 많은 숫자들을 보면서 한가지를 고르는 것은 너무 어렵습니다. 그래서 제가 이번 예시를 통해 추천 드리는 것은, 대륙 별 성과 기록을 유지하는 것 외에, 평균을 계산하는 것입니다. 평균 성과가 합리적인 single real number 평가 지표라고 하면 평균값을 산출함으로써, 알고리즘 C가 가장 낮은 평균 에러 값을 가지고 있다는 것을 빨리 확인할 수 있습니다.



여러분이 중요시 여기는 부분을 모두 감안하여 single row 평가 지표로 결합시키는 것은 쉬운 일이 아닙니다. 개인적으로 이런 경우에는 **satisficing (최소한의 충족)과 optimizing (최적화) 매트릭스가 간혹 유용한 경우**가 있었습니다.

러닝 타임

이미지 분별 시 classifier A 는 80 밀리세컨드, B는 95 밀리세컨드 , C는 1,500 밀리세컨드 소요 (= 이미지를 판별하는데 총 1.5초가 소요된다는 말)

ex) 종합비용= 정확도 - 0.5 * 러닝타임

정확도가 optimizing metric

러닝타임이 satisficing metric

일반적으로, 신중히 생각하는 N개의 매트릭스가 있으면 한가지를 최적화된 것으로 고르는 것이 합리적입니다. 그렇게 선택한 것에 대해선 가능한 한 잘 해야 합니다. 그 이후, N-1을 satisficing으로 선택합니다. 즉, 러닝타임이 100 밀리세컨드 한계치 이내인 경우를 선별하는 것입니다. 그 범위 안에 속하면, 얼마나 그 이상 빠른지 여부는 그다 신경 쓰지 않고 물론 그 한계치 이내이긴 하지만, 그 이후로는 동일 히 여깁니다.

그리고 구체적으로, 여기 있는 dev set와 테스트세트는 모두 같은 분포도에서 만들어져야 합니다. 그러므로 미래에 어떤 유형의 데이터를 다루게 되더라도, 어떤 데이터가 좋은 것이 될지

여부와는 별도로, 이렇게 생긴 데이터를 수집할 수 있도록 하십시오. **이 데이터가 어떤 유형의 데이터이단간에, dev set와 테스트세트에 모두 넣으세요. 이렇게 해서 여러분이 원하는 목표를 원하는 곳에 설정할 수 있는 것입니다.**

size 선택법

<선진유형>

: 트레이닝 dev와 테스트 세트일 설정해야 하는 경우, 트레이닝에 60%, dev에 20%, 그리고 테스트에 20% 사용하는 것입니다.

<현재>

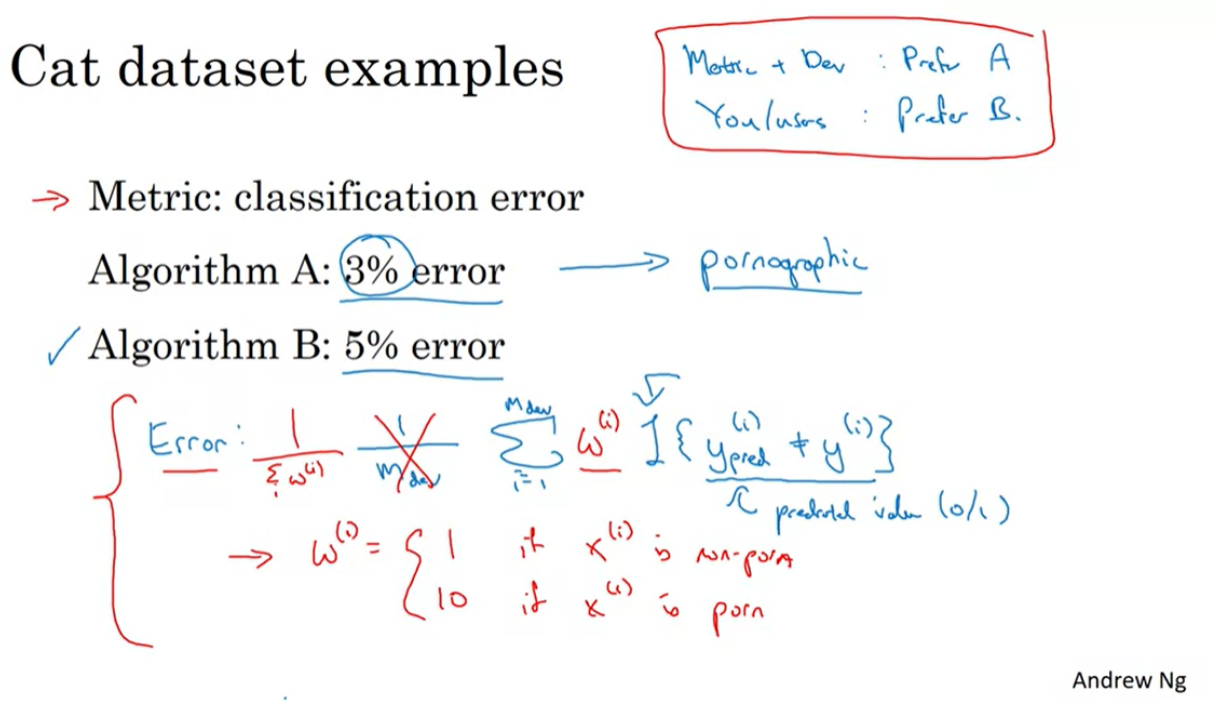
: 백만 개의 트레이닝 샘플이 있다고 가정해봅시다. 이런 경우에는 98%를 트레이닝 세트로 설정하고 1%는 dev, 나머지 1%는 테스트로 분배하는 것이 합리적일 수 있습니다. **그리고 dev와 테스트 세트를 줄이기 위해 DNT를 쓸 때도 말이죠.** 만약 백만 개의 사례가 있다고 하면 그것의 1%는 10,000개의 사례입니다, 이것은 dev와 테스트 세트로는 **충분**한 양입니다. 그래서, 최근 데이터세트의 양이 훨씬 더 큰 딥러닝 세대에서는 데이터나 dev set, 그리고 테스트 세트에 20% 또는 30% 미만으로 구성하는 것이 합리적인 편이라고 볼 수 있습니다. 딥러닝의 알고리즘 또한 데이터를 필요로 하기 때문에, 큰 데이터 세트를 보유하고 있는 것에 대한 문제점이 상당부분 트레이닝 세트로 이전됩니다.

그렇다면 테스트 세트는 어떨까요? 테스트 세트의 목적은 시스템 개발을 마친 이후로는, **테스트 세트가 마지막 시스템이 얼마나 좋은지 평가하는데 도움**을 줍니다. **가이드라인은 테스트 세트를 적당히 크게 하여 전반적인 시스템의 성능의 컨피던스 레벨이 높을 수 있도록 하는 것**입니다.

보유하고 있는 데이터에 따라 그 수치는 약간씩 다를 수 있습니다. 특정 어플에서는, 마지막 시스템의 전반적인 성능에 대한 높은 신뢰수준이 필요하지 않을 수 있습니다. 필요한 것은 트레이닝과 dev set가 전부일수도 있습니다. 제가 생각하기엔 테스트 세트가 없는 것도 괜찮을 수 있습니다.

시스템을 구축할 때 테스트 세트를 설치하지 않는 것은 절대 추천하지 않습니다. 저는 개인적으로 테스트 세트를 따로 두는 것이 안심이 된다고 생각하는데요. 변형 이전에 어떻게 했는지 공정한 평가 수치를 받을 수 있는데요 만약 아주 큰 dev set를 가지고 있어서 dev set를 심하게 overfit하지 않을 것이라 생각되면 train dev set를 갖는 것이 합리적인 방법이 아니라고 하기 어렵습니다. 물론 제가 일반적으로 추천하는 것은 아니지만요.

평가 metric 지표 셋팅



평가 매트릭 플러스 dev set는 A 알고리즘을 선호합니다. 그 이유는 A 알고리즘이 오류 수치가 너 낮고 지금 현재 사용하고 있는 매트릭이기 때문이죠. 그리고 유저들은 B 알고리즘을 선호할 것입니다. 포르노 사진을 걸러낼 수 있기 때문이죠. 이런 경우가 발생할 경우, 평가 매트릭이 더 이상 올바르게 알고리즘의 선호도 순서를 랭킹할 수 없을 때, A 알고리즘이 더 좋은 것이고 틀리게 평가를 하는 경우, 여러분이 아마도 평가 매트릭을 바꿔야 하거나 dev set나 테스트세트를 바꿔야 하는 신호일 수 있습니다.

이런 평가 매트릭을 바꾸는 한가지 방법은, **weight 변수를 넣는 것**입니다. 이것을 w(i)라고 하고, x(i)가 일반 이미지인 경우는 1이고 x(i) 값이 포르노 이미지인 경우 10 아니면 더 큰 숫자인 100으로 식을 만드는 것입니다. 이와 같이, 포르노 이미지인 경우에는 더 큰 weight를 부여해서 오류 값이 올라가도록 하는 것입니다. 이렇게 하면 포르노 이미지를 고양이 이미지로 잘못 분류하는 경우 오류 값이 훨씬 더 많이 올라가게 지정하는 것입니다. 이번 사례에서는, 10배의 높은 weight를 지정하여 포르노 이미지를 올바르게 판별하는 것에 대해 더 큰 비중을 부여합니다. 이 normalization을 constant로 만들고 싶으면, 엄밀히 이야기해서 이것은 1 나누기 w(i)의 합이 됩니다. 이렇게 하면 이 오류 값은 0에서 1사이의 값을 가지게 될 것이고요. 이 weighting의 상세 내용은 그리 중요하지 않습니다. 이 weighting을 도입하기 위해선, dev set와 test set를 거쳐야 합니다. 포르노 이미지를 dev와 테스트세트에 레이블 하여 이 weighting 함수를 도입시킬 수 있습니다.

> 가장 중요한 것은, 평가 매트릭이 알고리즘 관련하여 원하는 순서로 선호도를 알려주지 않는다고 하면 새로운 평가 매트릭 도입을 고려할 시점입니다.

앞서 말한 내용은, 평가 매트릭을 정의할 수 있는 한가지 방법입니다. 평가 매트릭의 목적은 2개의 classifier 중, 본인 어플에서 어떤 게 더 정확히 적용될 수 있는지 평가하는 것입니다. 이 비디오에서 알려드리고자 하는 내용만 따져보면, 새로운 오류 매트릭을 어떻게 정의하는지까지는 상세히 알 필요가 없습니다. 중요한 부분은 바로 이전 오류 매트릭이 만족스럽지 못한 경우, 불만족스러운 매트릭을 계속 유지하지 말고, 어떤 것이 더 뛰어난 알고리즘인지 그 선호도를 잘 캡처하는 새로운 매트릭을 정의하려고 시도해 보십시오. 눈치를 채셨겠지만 이제까지는, classifier를 평가하는 방식에 대해서만 정의하는 방법에 대해서 이야기 했었습니다. 즉, 우리는 음란물이 스트리밍되는 것과 관련해 classifier들을 잘 선호도 별로 랭킹하고 분류하는 것을 잘 평가할 수 있도록 관련 평가 매트릭을 정의 해했었습니다. 이런 사례가 바로 직교화의 실제 사례인데요, 머신러닝의 문제를 쪼개서 특정 단계로 그 절차는 나누어 보는 것입니다. 첫번째 단계는 수행하고 싶은 것을 캡처하는 매트릭을 정의하는 방법을 파악하는 것입니다. 그리고 해당 매트릭에서 잘 구현하는 것에 대해선 따로 고민해볼 것 같습니다.

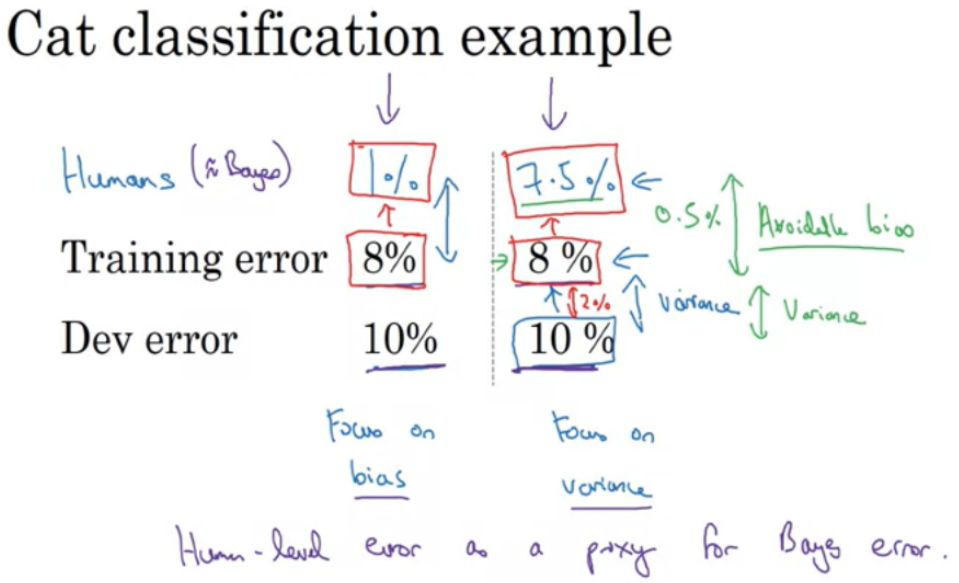
즉, 머신러닝 수행업무에 대해서 2가지의 단계로 보십시오. 목표를 결정하는 접근을 하면

1.매트릭을 정의하는 것이 첫 번째 단계

2.나머지는 두 번째 단계입니다.

매트릭과 dev set가 망가지는 예

고양이 Classifier A와 B가 각각 있다고 해봅시다. 각각 3퍼센트 오류, 5퍼센트 오류가 dev set에서 나왔습니다. 아니면 다운받은 이미지들을 바탕으로 한 테스트세트에서 나온 것일 수도 있죠. 고품질의 이미지들 말입니다. 그런데 알고리즘 제품을 투입시키면 B 알고리즘이 더 뛰어난 성능을 발휘하는 것처럼 보이는 것입니다. dev set에서는 더 좋은데도 말이죠. 그리고, 다운로드 받은 고품질의 이미지들을 통해 트레이닝을 진행해왔는데도 모바일 어플로 출시하면 유저들이 수많은 다양한 사진을, 잘 프레임되지 않은 사진을 올리는 것입니다. 고양이 사진에 국한됨에도 불구하고 고양이들이 다양한 표정들이 있고, 이미지들이 흐릿하고, 결국에는 B 알고리즘이 더 잘 작동되는 것입니다. 이런 사례가 바로 매트릭과 dev set가 망가지는 예입니다. **문제는 dev set와 테스트세트에서 평가를 한다는 것입니다. 아주 잘나오고 고화질의 고프레임 사진을 바탕으로 말이죠. 하지만 유저가 신경 쓰는 것은 본인들이 올린 이미지가 잘 작동되는 것입니다.** 이런 사진들은 프로페셔널하지 않거나, 흐릿하게 나오거나 프레임이 낮은 사진일 수도 있습니다. 그러므로 가이드라인은, 매트릭에서 잘 하는 것이고, 현재 dev set 또는 dev와 테스트세트 분포 데서 잘하는 것입니다. 이런 것이 어플에서 잘 작동하는 것으로 이어지지 않는다면, **매트릭과 dev test set를 바꾸십시오.**



Andrew Ng

우리는 여러분이 어떻게 러닝 알고리즘이 트레이닝세트에서 잘 되기를 바라는지에 대해 이야기 했었는데요. 가끔씩은 너무 잘하는 것이 별로 좋지 않을 수도 있습니다. 인간레벨성능을 알면 그것에 따라 얼마나 성능이 발휘되야 하는지 너무 잘하면 좋지 않을 수도 있으니 적당한 레벨로 알고리즘이 트레이닝세트에서 발휘할 수 있도록 가능케 할 것입니다. 어떤 말인지 자세히 설명해드리도록 하겠습니다. 고양이 인식 프로그램을 봤었는데요, 이 사진을 보면 인간은 거의 완벽에 가까운 정확도를 가지고 있고 인간오류는 1퍼센트라고 해봅시다. 이 경우, 러닝 알고리즘이 8퍼센트 트레이닝 오류, 10퍼센트 dev 오류 그리소 트레이닝세트에서는 조금 잘 하고 싶으셨을 수도 있죠. 그러므로 트레이닝 세트에서의 알고리즘의 성능과 인간의 차이가 매우 크기 때문에 이것은 해당 알고리즘이 트레이닝세트에 잘 피팅되지 않고 있음을 보여줍니다. 그러므로, 바이어스 와 편차를 줄이는 방면에서, 이번 경우에는 바이어스를 줄이는데 중점을 둘 것 같습니다. 더 큰 신경망을 트레이닝 한다거나 더 오랜 시간 동안 트레이닝세트를 운영하는 방법이 있겠습니다. 트레이닝세트에서 더 잘하기 위함이죠. 이번에는 동일한 트레이닝 오류와 dev error를 봅시다. 그리고 인간레벨성능이 1퍼센트가 아니라고 가정해봅시다. 이것을 이쪽으로 복사해보겠습니다, 다른 어플 또는 다른 데이터세트에서는 인간 오류가 7.5퍼센트라고 해봅시다. 데이터세트에 있는 이미지가 너무 흐려서 사람이 판단하기에도 고양이인지 여부를 알 수 없는 경우가 있을 수 있겠죠. 이 사례는 약간 부자연스러운 사례일 수 있긴

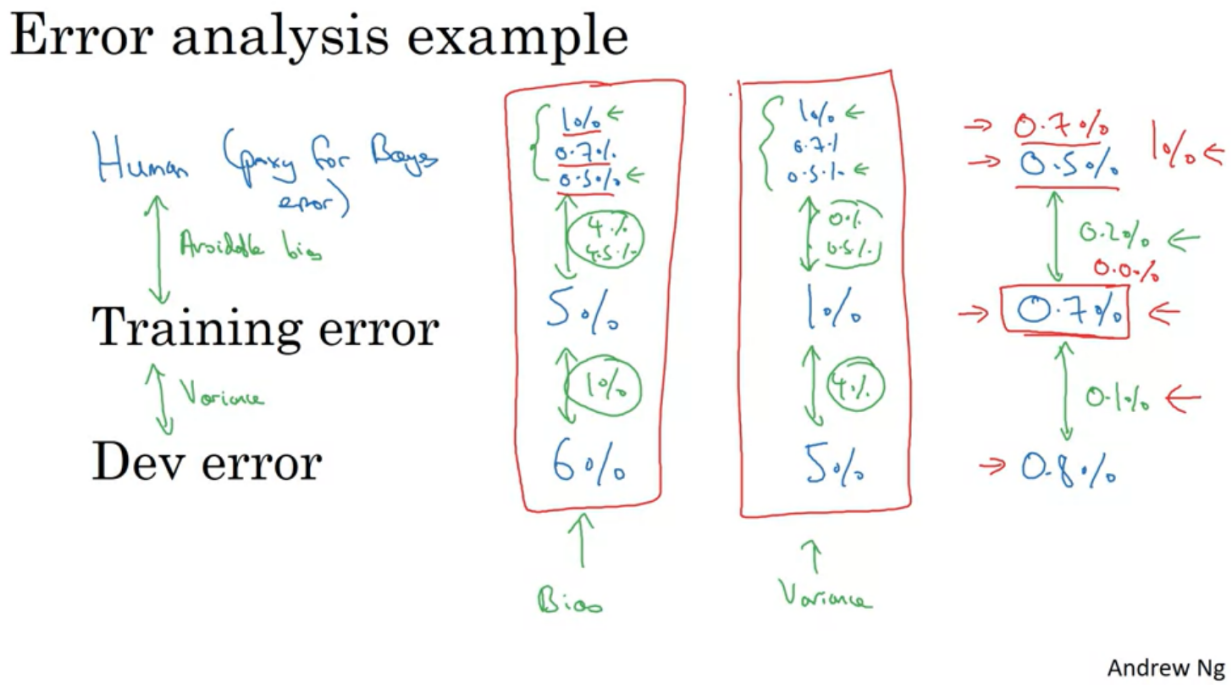
합니다. 왜냐면, 인간은 사진을 보고 고양이인지 이미지를 분간하는 게 굉장히 뛰어나기 때문입니다. 하지만 예시인 만큼, 한번 보도록 하겠습니다. 그럼 데이터세트에 있는 이미지가 너무 흐리거나 화질이 너무 좋지 않아서 인간도 7.5퍼센트의 오류를 범한다고 가정해봅시다. 이런 경우, 트레이닝 오류와 dev error가 다른 예제들과 동일하긴 하지만, 트레이닝세트에서는 잘 하고 있는 것을 보실 수 있습니다. 인간레벨성능에서 조금 못 미치는 수준을 보이고 있습니다. 두 번째는, 이 부분을 조금 줄이고 싶을 수 있겠죠. 러닝 알고리즘의 편차를 줄이는 겁니다. 그렇게 하기 위해 Regularization (일반화)를 시도하여 dev error를 트레이닝 오류와 비슷한 수준으로 만드는 것이죠. 이전 코스에서 다루었던 바이어스와 편차에 대한 내용에서는, 주로 Bayes error가 0에 가까운 업무를 다루었습니다. 그러므로 여기서 일어난 것을 설명 드리자면, 고양이인식 프로그램 예로 들자면, 인간레벨 오류를 Bayes error의 추정치로 생각하거나 Bayes optimal error의 추정 수치로 생각합니다. 컴퓨터의 인식 영역에서는, 꽤 합리적인 프록시 값입니다. 이유인 즉 슨, 사람은 computer vision영역에서 굉장히 능숙하고 사람이 할 수 있는 부분이 Bayes error와 크게 다르지 않기 때문입니다. 인간레벨 오류는 정의로만 봤을 때, Bayes error보다 못합니다. 그 이유는 그 어떤 것도 Bayes error보다 더 나을 순 없기 때문이죠. 하지만 인간레벨 오류는 Bayes error에서 크게 떨어지지 않을 수 있습니다. 여기서 놀라운 사실을 발견했는데요, **인간레벨오류가 얼마인지에 따라 또는, 이 값은 사실 Bayes error와 동일합니다**만, 어떤 것을 이룰 수 있느냐에 따라 다르지만, 동일한 트레이닝 오류와 dev error인 경우, 이 2가지의 사례와 같이, 바이어스를 줄이는 기술이나 편차를 줄이는 방법에 중점을 두기로 했습니다. 왼쪽 예제에서는, 8퍼센트의 트레이닝은 1퍼센트까지 줄일 수 있다고 하면 굉장히 큰 값입니다. 바이어스를 줄이는 기술이 가능케 하죠. 오른쪽 예시는, Bayes error가 7.5퍼센트입니다. 여기서는 인간레벨오류를 Bayes error를 프로시로 사용합니다. 만약 Bayes error가 7.5퍼세트와 가깝다고 생각되는 경우, 트레이닝 오류를 더 이상 줄일 수 있는 공간이 많지 않습니다. 7.5퍼센트보다 더 많이 뛰어나길 바라지 않습니다. 어차피 그렇게 되기 위해선 교육 량을 늘려야 할 것이기 때문이죠. 대신에, 여기 2퍼센트 갭을 줄이는 방안을 생각해보면, 일반화 과 같은 또는 트레이닝 데이터를 더 수집하는 방안으로 편차를 줄이는 기술을 생각해 볼 것입니다. 몇 가지 이름을 부여하자면, 자주 쓰이는 용어는 아니지만, 개인적으로 유용한 용어였는데요, 생각하는 사고방식이 유용합니다. Bayes error 또는 Bayes error의 근사치와 트레이닝 오류의 차이를 avoidable bias라고 할 것입니다. 여러분이 할 수 잇는 것은 트레이닝 성능을 Bayes error 값으로 내려갈 때까지 계속 개선시키는 것입니다. 하지만 Bayes error보다 잘하면 안되겠죠. 사실 overfitting을 하지 않는 이상 Bayes error보다 잘할 수는 없습니다. 그리고 이것, 트레이닝 오류와 dev error의 차이는 알고리즘 편차 문제의 수치입니다. ‘avoidable bias’라는 용어는 최소의 오류 값이나 특정 오류가 있다는 것을 인정하는 셈이고, Bayes error가 7.5퍼센트인 경우 이 이하로 내려갈 수 없다는 뜻도 내포하고 있습니다. 이 오류 수치 밑으로는 내려가면 좋지 않습니다. 그러므로, 트레이닝 오류가 8퍼센트라고 이야기하기 보다는, 8퍼센트는 바이어스의 측정수치이며, 해당 예시에서는 avoidable bias가 0.5퍼센트이거나, 2퍼센트는 편차 수치입니다. 그러므로 2퍼센

트를 줄일 수 있는 공간은 0.5퍼센트 줄이는 공간보다 훨씬 더 많습니다. 반대로 왼쪽 예시는, 7퍼센트는 avoidable bias 이고, 2퍼센트가 편차입니다. 왼쪽 사례에서는, avoidable bias 를 줄이면 더욱 큰 효과를 얻을 수 있겠죠. 이번 예시에서는, 인간레벨 오류에 대해 다뤘는데요, Bayes error에 대한 추정치를 이해하면 시나리오마다 각각 다른 전술로 접근하여, bias avoidance 기술을 이용하거나, variance avoidance 기술을 이용할 수 있습니다. 본인이 어느 곳에 중점을 둘지 여부와 관련하여 의사결정을 내리는데 있어, 인간레벨성능이 미묘하게 관여하는 부분이 있는데요, 다음 비디오에선, 인간레벨성능이 구체적으로 어떤 것을 이야기하는지 자세히 알아보겠습니다.

인간오류를 생각하기 가장 좋은 부분은 프록시로 또는, Bayes error로 접근을 하는 방식입니다.

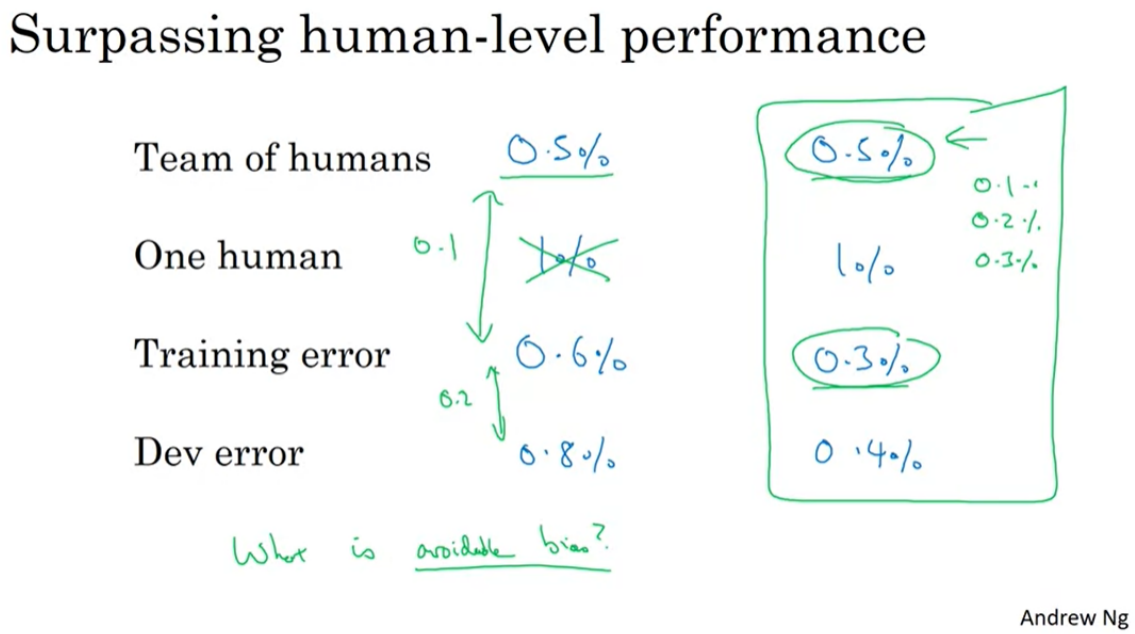
프록시나 Bayes error의 추정 값을 원하는 경우, 경험이 많은 의사 팀이 토론에 거쳐 0.5퍼센트의 오류를 범할 수 있다고 하면 Bayes error는 0.5퍼센트이거나 그 이하일 것입니다.

최적 오류는 0.5퍼센트보다 높을 수 없습니다. 그래서 이런 설정에서, 저 같은 경우엔 0.5퍼센트를 Bayes error의 추정치로 지정할 것입니다. 그렇게 해서 0.5퍼센트를 인간레벨성능으로 정의하는 것이죠.



이번 토론 내용과 이전 코스에서 다뤘던 내용의 큰 차이는, 트레이닝 오류를 0퍼센트와 비교하기 보다는, 그것을 바이어스의 추정 그 자체로 보는 것입니다.

이전 코스에서, 트레이닝 오류를 계산하고, 0보다 얼마나 더 큰지 비교해보았는데요, 이렇게 해서 바이어스가 얼마나 큰지 알아봤습니다. Bayes error가 0에 가까운 경우엔 잘 된다는 사실을 알았는데요, 고양이 인식 프로그램 같은 경우 말이죠. 이 경우, 인간이 거의 완벽하기 때문에, Bayes error도 거의 완벽하다고 볼 수 있습니다. 그러므로, Bayes error가 0에 가까울 경우 잘 작동하는데요, 데이터가 음성인식기기처럼 혼잡하거나, 오디오가 시끄러운 경우 여러분이 말한 것을 인식하기 어려운 경우가 있습니다. 이런 경우, 표기화가 불가능하죠. 이런 문제의 경우, Bayes error에 대한 추정치를 더 정확하게 알면, avoidable bias와 편차를 더 정확히 추정할 수 있습니다. 그러므로 결과적으로 바이어스를 줄이는데 중점을 둘 것인지, 편차를 줄이는데 초점을 맞출 것인지 결정을 내릴 수 있습니다.



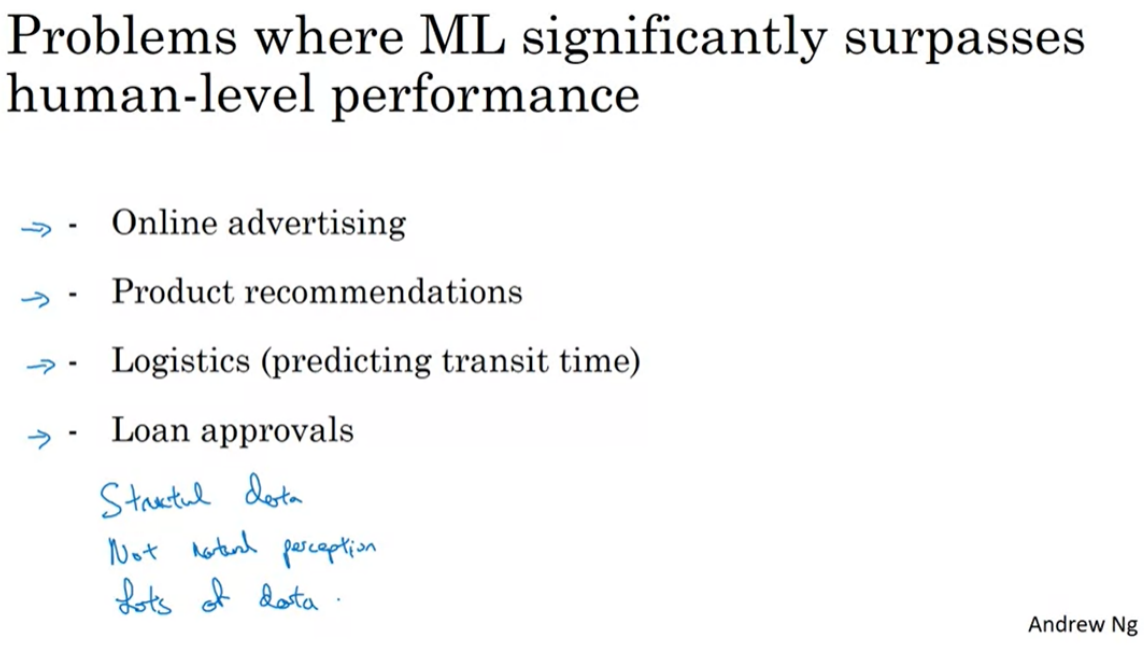
1. 왼쪽 예시

0.5퍼센트가 base error의 추정치

적어도 0.1퍼센트로 avoidable bias를 추정하고, 편차를 0.2퍼센트 정도로 추정

2. 오른쪽 예시

사람으로 구성된 팀과, 개인 인간 능력이, 이전과 동일하고 알고리즘이 0.3퍼센트의 트레이닝 에러를 갖고, 0.4퍼센트의 dev error를 갖는다고 해봅시다. avoidable bias는 어떤 것일까요? 이제는 답하기 어려워졌습니다. 트레이닝에러가 0.3퍼센트라는 사실이, 곧 0.2퍼센트 오버피팅 했다는 것일까요? 아니면 베이스에러가 0.1퍼센트이거나, 또는 0.2퍼센트 베이스에러일 수도 있고, 0.3퍼센트일 수도 있겠죠. 정확히 알 수가 없습니다.



Reducing (avoidable) bias and variance