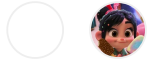


기술블로그



홈 태그 방명록

DL 스터디&프로젝트

[Euron 중급 세션 11주차] 딥러닝 2단계 5. 하이퍼파라미터 튜닝

by 공부하자_ 2023. 11. 20.

딥러닝 2단계: 심층 신경망 성능 향상시키기

5. 하이퍼파라미터 튜닝

📌 튜닝 프로세스(C2W3L01)

핵심어: 하이퍼파라미터(hyperparameter), 튜닝(tuning), 학습률(learning rate)

지금까지 신경망을 학습시킬 때 여러 하이퍼파라미터들이 관여한다는 것을 배웠는데, 그럼 좋은 하이퍼파라미터는 어떻게 찾을 수 있을까? 이번에는 체계적으로 하이퍼파라미터 튜닝을 진행하는 방법에 대해 알아보려고 한다.

분류 전체보기

DL 스터디&프로젝트

Data Science 프로젝트

GitHub 스터디

Data Science 개인 공부

Backend 프로젝트

기타 공부

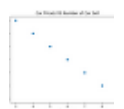
공지사항

최근글 인기글

[Euron 중급 세션 11주차]...
2023.11.20

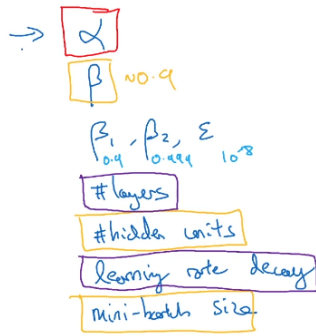


[Euron 중급 세션 10주차]...
2023.11.20



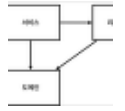
[Euron 중급 세션 10주차] 딥러닝 2단계 ...

Hyperparameters



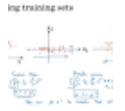
백엔드 프로젝트 8주차 스...

2023.11.11



[Euron 중급 세션 9주차] ...

2023.11.06



Andrew Ng

심층 신경망을 학습시킬 때 가장 어려운 일은 다뤄야 할 하이퍼파라미터가 많다는 것이다. 학습률 α (학습률 감쇠 learning rate decay를 같이 쓸 수 있음)나 모멘텀을 나타내는 β , Adam 최적화 알고리즘의 하이퍼파라미터 $\beta_1, \beta_2, \epsilon$ 도 있다. 층의 수와 층에서 은닉 유닛의 숫자도 정해야 할 수 있다. 또한 미니배치의 크기를 정해야 하는 상황이 생길 수 있다. 그리고 대부분의 학습에서는 일부 파라미터들이 다른 파라미터보다 중요할 수 있는데, 우선 학습률 α 는 튜닝해야 할 가장 중요한 하이퍼파라미터이고 α 이외에 주로 튜닝하는 것들로는 모멘텀이 있다. 이는 기본값을 0.9로 설정할 수 있다. 최적화 알고리즘을 효율적으로 돌리기 위해 미니 배치 크기도 튜닝할 수 있다. 은닉 유닛도 자주 튜닝한다. 즉 오렌지 박스로 표시한 파라미터들(모멘텀, 은닉 유닛, 미니배치 등)은 학습률 다음으로 중요한 요소들이라고 볼 수 있다. 그 다음으로 중요한 것들은 층의 수, 학습률 감쇠이다(보라색 표시/ 한편 Adam 알고리즘에서 $\beta_1, \beta_2, \epsilon$ 는 튜닝하지 않고 0.9, 0.999, 10^{-8} 을 항상 사용함). 이렇게 무엇이 무엇보다 중요한지 그 우선순위에 대해 알아보았다. 학습률이 당연히 가장 중요하고 그 다음이 모멘텀/은닉유닛/미니배치, 그 다음으로 중요한 것이 레이어(층) 수/학습률 감쇠이다. 그러나 이것이 확정적으로 정해진 것은 아니며, 사람마다 의견이 다를 수 있다.

최근댓글

오늘 하루 고생 많으셨습니다.

좋은 글 잘 보고 가요! 감사...

좋은 글 잘 보고 가요! 감사...

잘보고 갑니다!

태그

pandas, 데이터분석,
딥러닝스터디, 판다스,
이저스퍼블리싱,
딥러닝교과서, bda,
데이터사이언스, Doit,
판다스입문

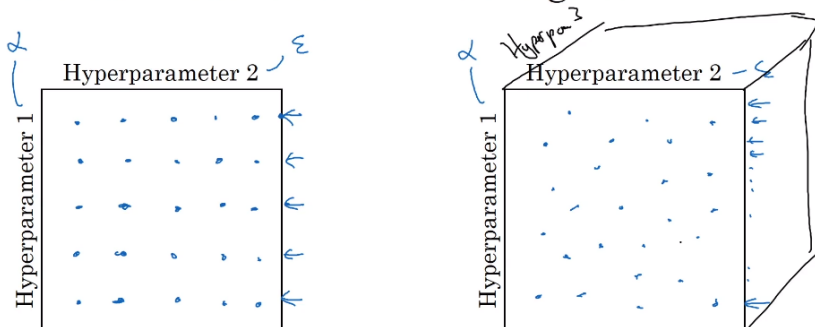
전체 방문자

547

Today : 8

Yesterday : 2

Try random values: Don't use a grid

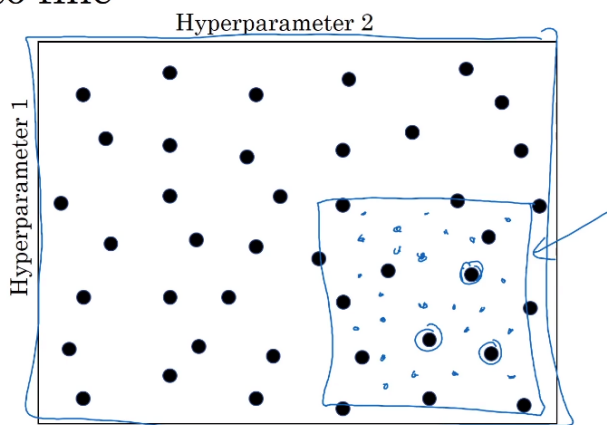


Andrew Ng

만약 하이퍼파라미터를 튜닝한다면 어떤 값을 탐색할지 어떻게 정할 수 있을까? 머신러닝이 만들어진지 얼마 되지 않았을 때는 두 개의 하이퍼파라미터가 있을 때 (Hyperparameter1, Hyperparameter2) 격자점을 탐색하는 것이 일반적인 방법이었다. 그리고 체계적으로 격자점에 있는 값들을 탐색하는 것이다. 예를 들어 슬라이드의 5*5 격자에서 25개의 점만 생각했을 때 최고의 하이퍼파라미터를 정하는 것이다. 그러나 이 방법은 하이퍼파라미터의 수가 적을 때 쓸 수 있으며, 딥러닝 방식에서는 다른 방식을 추천한다. 바로 **무작위로 점을 선택**하는 것이다. 동일하게 25개의 점만을 생각해보았을 때, 그 점들에 대해 하이퍼파라미터를 정하는 것이다. 이렇게 하는 이유는 어떤 하이퍼파라미터가 문제 해결에 더 중요한지 미리 알 수 없기 때문이다. 이전에 보았듯 하이퍼파라미터에는 중요도 순위가 있는데, 예를 들어 Hyperparameter1이 학습률 α (제일 중요)이고 Hyperparameter2를 Adam 최적화 알고리즘의 ϵ (가장 덜 중요)라고 생각하자. 이런 경우 학습률 α 를 고르는 것이 ϵ 를 고르는 것보다 더 중요하다. 왼쪽 격자점의 경우 5개의 α 값(세로)을 확인하게 되는데 이때 ϵ 값이 달라도 결과는 같은 것을 확인할 수 있다. 결과적으로 25개의 모델을 학습시켰지만 가장 중요한 α 에 대해서는 5개만 학습시킨 것과 다를 게 없는 것이다.

반대로 오른쪽처럼 무작위로 모델을 고를 경우, 25개의 서로 다른 학습률 α 값을 이용하여 학습시키게 되고 더 좋은 하이퍼파라미터를 잘 찾게 될 것이다. 위 경우에서는 두 개의 하이퍼파라미터만 사용하여 예시를 들었는데, 실전에서는 훨씬 더 많은 하이퍼파라미터들을 다루게 될 것이다. 만약 하이퍼파라미터가 3개라고 한다면 정사각형 평면을 탐색하는 것이 아니라 정육면체 입체 공간을 탐색하는 것이 된다. 세 번째 차원은 Hyperparameter3를 가리키게 될 거시며, 이 3차원 정육면체 안에서 모델을 고른다면 각 하이퍼파라미터에 대해 훨씬 많은 값을 시험해보게 될 것이다. 실제로는 3보다 더 많은 하이퍼파라미터들을 탐색하게 되며, 어플리케이션에서 어떤 하이퍼파라미터가 가장 중요한지 미리 아는 것은 어려운 일이다. 그리고 격자점보다 무작위로 모델을 정하는 것이 가장 중요한 하이퍼파라미터의 다양한 값을 탐색할 수 있도록 한다(무엇이 중요하건 상관없이).

Coarse to fine



Andrew Ng

다른 일반적인 방법 중 하나는 **정밀화 접근**이다. 예를 들어 2차원에서 점들을 사용한다고 하자. 그리고 특정 점(파란색 표시)이 가장 좋다는 것을 찾았으며, 그렇다면 그 주변에 있는 점들도 좋은 성능을 보일 것이다. 그렇다면 정밀화 접근에서는, 더 작은 영역으로 확대해서 더 조밀하게 점들을 선택한다. 무작위인 것은 그대로지만 최고의 하이퍼파라미터들이 해당 영역에 있으리라는 믿음 아래 파란색 사각형 안에 초점을 두고 탐색하는 것이다. 즉 전체 사각형에서 탐색한 뒤, 더 작은 사각형으로 범위를 좁혀나가는 것이다. 그러면 파란 사각형 안에서 더 조밀하게 시험해볼 수 있게 되는 것이다. 이런 정밀화 접근도 자주 쓰이는 방식이다. 그리고 이렇게 하이퍼파라미터의 여러 값들을 시험해보며 학습의 목표나 개발 목표 등에 있어서 최고의 파라미터를 고르는 것이다.

이렇게 해서 하이퍼파라미터를 찾는 방법에 대해 정리해보았는데, 반드시 알아두어야 할 것은 첫째, 격자점이 아니라 무작위라는 점, 둘째, 원한다면 정밀화 접근을 이용할 수 있다는 점이다.

- 딥러닝에는 다양한 하이퍼파라미터가 존재합니다. 상황에 따라 다를 수도 있지만, 보통 우선 조정하는 순서로 나열했습니다.
 - 학습률(α)
 - 모멘텀(Momentum) 알고리즘의 β
 - 은닉 유닛의 수
 - 미니배치 크기
 - 은닉층의 갯수
 - 학습률 감쇠(learning rate decay) 정도
 - 아담(Adam) 알고리즘의 $\beta_1, \beta_2, \epsilon$
- 딥러닝의 하이퍼파라미터 탐색은 무작위 접근 방식이 좋습니다. 이유는 어떤 하이퍼파라미터가 문제 해결에 더 중요한지 미리 알 수 없기 때문입니다.
- 다른 일반적 접근 방식중 하나는 "정밀화 접근" 입니다. 우선 전체 하이퍼파라미터 공간에서 탐색하여 좋은 점을 찾은 후, 그 근방에서 더 정밀하게 탐색하는 과정입니다.

🔍 그리드 서치(Grid Search)

Grid search(격자 탐색)은 모델의 하이퍼 파라미터에 넣을 수 있는 값들을 순차적으로 입력한뒤에 가장 높은 성능을 보이는 하이퍼 파라미터들을 찾는 탐색 방법
그리드 서치를 하는 이유는 가장 우수한 성능을 보이는 모델의 하이퍼 파라미터를 찾기 위해서이며, 모든 경우의 수를 넣어보고 가장 성능이 좋게 만드는 모델의 하이퍼 파라미터를 찾는 것.

(출처: <https://huidea.tistory.com/32>)

🔍 랜덤 탐색(Random Search)

주어진 구간 안에서 랜덤으로 숫자를 뽑아서 실험을 함. Grid Search에서 실험할 하이퍼파라미터들을 명시적으로 정해줘야 한다면, Random Search는 하이퍼파라미터로 시도할 숫자의 구간과, 횟수를 정해줌.

예를 들어 0과 1사이에서 랜덤으로 숫자를 뽑아서 그 숫자를 정규화 계수로 하는 모델을 만들되, 그런 실험을 50번 반복해라 같은 식. Scikit-learn의 RandomSearchCV라는 함수를 활용하면 Random Search를 구현할 수 있음.

(출처: <https://datarian.io/blog/grid-search-random-search>)

Random Search는 주어진 범위 내에서 임의의 조합을 추출하여 최적의 조합을 탐색하는 방법

(출처: <https://dacon.io/en/codeshare/4646>)

🔍 그리드 서치 VS 랜덤 탐색

- 랜덤 탐색은 하이퍼파라미터 조합을 무작위로 선택하여 탐색하는 방법이며, 그리드 서치는 모든 조합을 체계적으로 탐색함
- 랜덤 탐색은 다양한 조합을 탐색할 수 있으며, 최적의 조합을 더 빠르게 찾을 수 있음. 그리드 서치는 모든 조합을 탐색하기 때문에, 탐색 공간이 크면 계산 비용이 매우 높아질 수 있음(효율 ▼)
- 랜덤 탐색은 불규칙적인 탐색 과정을 가지고 있어 최적의 조합을 찾는 시간이 불규칙적일 수 있으나 그리드 서치의 경우 체계적인 탐색 방식이기 때문에 최적의 조합을 찾는 시간이 일정함.

▶ 랜덤 탐색과 그리드 서치는 각각의 장단점을 가지고 있으며, 상황에 따라 적절한 방법을 선택하여 하이퍼파라미터 튜닝을 수행할 수 있음.

(참고: <https://dacon.io/competitions/open/235698/talkboard/403915>)

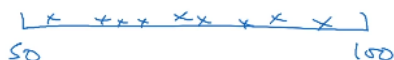
📌 적절한 척도 선택하기(C2W3L02)

핵심어: 하이퍼파라미터(hyperparameter), 학습률(learning rate)

지금까지 무작위로 하이퍼파라미터를 찾는 것이 더 효율적인 탐색이라고 학습했는데, 여기서 무작위라는 것이 가능한 값들 중 공평하게 뽑는 것이라고는 할 수 없다. 대신 적절한 척도를 정하는 것이 더욱 중요하다. 이번에는 어떻게 척도를 정하는지 알아보려고 한다.

Picking hyperparameters at random

→ $n^{trial} = 50, \dots, 100$

 50 100

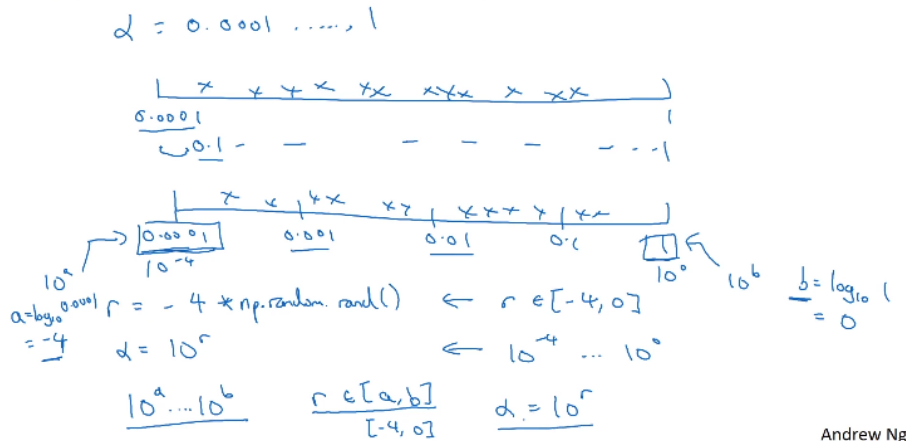
→ #layers L: 2 - 4

2, 3, 4

Andrew Ng

어떤 레이어 l 에 대해 은닉 유닛의 수 n_l 을 정하고, 값의 범위로 50부터 100을 생각하자. 이 경우 50부터 100까지의 수직선에서 무작위하게 값들을 고른다고 가정한다. 이는 하이퍼파라미터를 고르는 꽤 합리적인 방법이다. 또는 신경망에서 레이어의 수 L 을 정한다고 했을 때 층의 숫자가 2에서 4 사이라고 생각할 수 있다. 2에서 4까지의 숫자를 선택할 때(층의 수 하이퍼파라미터를 선택할 때)는 무작위하게 뽑는 것은 물론, 격자점을 사용해도 문제 없다(이 예시들은 가능한 값 중 무작위하게 뽑는 것이 합리적인 경우들이며, 모든 하이퍼파라미터가 다 그렇지 않다).

Appropriate scale for hyperparameters



Andrew Ng

다른 예시를 살펴보자. 학습률 α 를 탐색하는데 범위로 0.0001부터 1까지를 생각하고 있다고 했을 때, 0.0001부터 1까지의 수직선 상에서 균일하게 무작위로 값을 고를 것이다. 여기서 90%의 샘플이 0.1과 1 사이에 있을 것이며, 이는 즉 90%의 자원을 0.1과 1 사이를 탐색하는 데 쓰고, 단 10%만을 0.0001과 0.1 사이를 탐색하는 데 쓰는 것이 된다. 이는 비합리적으로 보일 수 있다. 이런 경우, 이와 같은 선형 척도 대신 로그 척도에서 하이퍼파라미터를 찾는 것이 더 합리적이다.

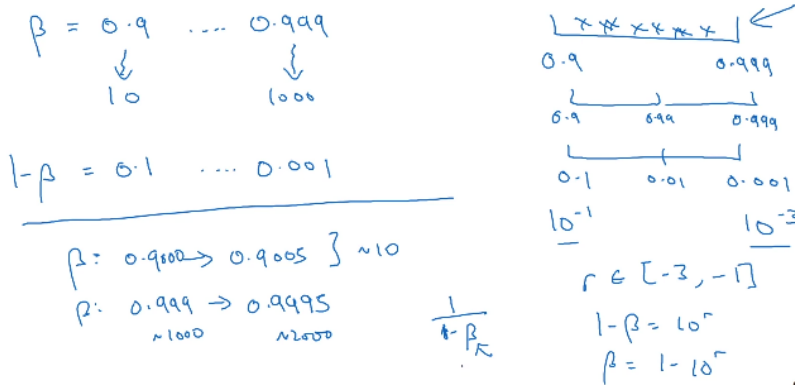
똑같이 수직선 위에 0.0001부터 0.001, 0.01, 0.1, 그리고 1까지 값들이 있다고 했을 때, 이와 같은 로그 척도에서 균일하게 무작위로 값을 뽑는 것이다. 그러면 0.0001과 0.001 사이, 0.001과 0.01 사이를 탐색할 때 더 많은 자원을 쓸 수 있는 것이다. 파이썬에서는 이를 구현하기 위해 " $r = -4 * \text{np.random.rand}()$ "를 쓰고, 그럼 무작위로 선택된 α 값은 10^r 이 되는 것이다. 구체적으로 보자면 $r = -4 *$

$\text{np.random.rand}()$ 에서 r 은 -4와 0 사이의 무작위 값일테고, 따라서 α 값은 10^{-4} 와 10^0 사이, 즉 10^{-4} 와 1 사이가 되는 것이다. 더 일반적인 경우, 10^a 에서 10^b 까지를 로그 척도로 탐색한다면, 위 예시에서는 0.0001이 10^a 가 되며 a 는

$\log_{10}(0.0001)$ 에 대해 계산했을 때 얻을 수 있는 것이다. 마찬가지로 오른쪽 값 1에서 b 는 $\log_{10}(1)=0$ 으로 0이라는 점을 확인할 수 있다. r 은 a 와 b 사이에서 균일하게 무작위로 뽑히며(위 예시에서는 r 이 -4와 0 사이) 그리고 무작위의 하이퍼파라미터 α 는 10^r 이 되는 것이다. 다시 정리하자면 낮은 값에서 log를 취해서 a 를 갖고, 높은 값에서 log를 취해 b 를 찾는 다음 10^a 에서 10^b 까지를 로그 척도로 탐색하는

것이다. r 을 a 와 b 사이에서 균일하게 무작위로 뽑으면 하이퍼파라미터 10^r 이 되는 것이며 이런 방식으로 로그 척도에서 샘플링을 할 수 있는 것이다.

Hyperparameters for exponentially weighted averages



Andrew Ng

또 다른 예시로는 지수가중평균을 계산할 때 사용되는 하이퍼파라미터 β 에 관한 것이다. β 를 0.9와 0.999 사이에서 찾다고 할 때, 0.9의 경우에는 지수가중평균이 최근 10일간의 평균 기온처럼 마지막 10개 값의 평균과 비슷하고, 0.999의 경우에는 마지막 1000개 값의 평균과 비슷했던 점을 학습했었다. 이전 슬라이드에서 보았던 것처럼 0.9와 0.999 사이를 탐색한다면 0.9와 0.999 사이를 균일하게 무작위로 탐색하는 것은 합리적이지 않다. 더 나은 방법은 $1-\beta$ 에 대해 값을 탐색하는 것인데, 이 값은 0.1부터 0.001 사이가 되며 그러면 β 를 0.1에서 0.01을 거쳐, 0.001 사이에서 탐색하는 것이다. 이전 슬라이드에서 배웠던 방법을 써보자면, 0.1은 10^{-1} 이고 0.001은 10^{-3} 이다(이전 슬라이드에서는 작은 값이 왼쪽, 큰 값이 오른쪽에 있었지만 여기서는 반대로 작성하여 큰 값이 왼쪽, 작은 값이 오른쪽에 있도록 한다). 여기서 해야 할 일은 -3과 -1 사이에서 균일하게 무작위로 값을 뽑는 것이다. 이제 $1-\beta$ 를 10^r 로 생각하면 되니 β 가 $1-10^r$ 이 되는 것이다. 이렇게 적절한 척도 위에서 무작위로 하이퍼파라미터 샘플을 추출하며, 이 방법을 이용하면 0.9부터 0.99를 탐색할 때와 0.99부터 0.999를 탐색할 때 동일한 양의 자원을 사용할 수 있다.

이쯤 되면 왜 이 방법이 필요한지 수학적 증명이 궁금할 수 있는데, 왜 선형 척도에서 샘플을 뽑는 것은 안 좋을까? 만약 β 가 1에 가깝다면, β 가 아주 조금만 바뀌어도 결과가 아주 많이 바뀌게 된다. 예를 들어 β 가 0.9에서 0.90005로 바뀌었다면 결과에 거의 영향을 주지 않는다. 그러나 β 가 0.999에서 0.9995로 바뀌었다면, 알고리즘의 결과에 큰 영향을 줄 것이다. 전자는 대략 10개의 값을 평균내는 것이지만, 후자의 경우 마지막 1000개의 값의 지수가중평균을 내는 것에서 마지막 2000개 값의 평균을 내는 것으로 바뀌었기 때문이다($1/(1-\beta)$ 라는 식은 β 가 1에 가까워질수록 작은 변화에도 민감하게 반응함). 따라서 β 가 1보다 가까운 곳에서 더 조밀하게 샘플을 뽑는다. 반대로 $1-\beta$ 는 0이 가까운 곳이 되는 거시다. 따라서 가능한 결과 공간을 탐색할 때 더 효율적으로 샘플을 추출할 수 있는 것이다.

만약 하이퍼파라미터를 고를 때 적절한 척도를 사용하지 않더라도 크게 걱정할 필요는 없다. 다른 척도가 우선하는 상황에서 균일한 척도에서 샘플링을 하더라도 정밀화 접근을 사용하면 괜찮은 결과를 얻을 수 있기 때문이며, 그래서 반복할수록 더 유의미한 하이퍼파라미터 범위를 탐색하게 되는 것이다.

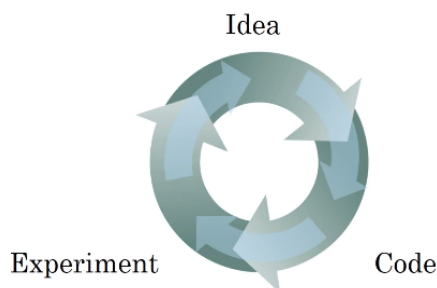
- 무작위로 뽑는 것이 합리적인 하이퍼파라미터들이 있습니다. 예를 들어, 은닉 유닛의 수, 은닉층의 수
- 하지만, 학습률의 경우 다릅니다.
 - 1 과 0.0001 사이의 값중에 균일하게 무작위 값을 고르게 되면, 90%의 값이 1 과 0.1 사이에 존재하기 때문에, 공평하다고 할 수 없습니다.
 - 따라서 선형척도대신 로그척도에서 하이퍼파라미터를 찾는 것이 합리적입니다.
 - 위 예시에 따르면 0과 -4 사이에 균일하게 무작위로 고르고 10의 지수로 바꿔주는 것입니다.
- 다른 예시로, 지수 가중 이동 평균에서 사용되는 β 입니다. 마찬가지로 0.9 와 0.999 사이의 값을 탐색하는 것은 비합리적이기 때문에 $1-\beta$ 를 취해준 후, 위의 예시와 마찬가지로 로그척도에서 무작위 값을 선택하여 탐색합니다.
- 왜 선형척도에서 샘플을 뽑은 것이 안좋은 까요?
 - 그 이유는 위의 값들은 1에 가까울 수록 알고리즘 결과에 더 큰 영향을 끼치기 때문입니다.

🔴 하이퍼파라미터 튜닝 실전(C2W3L03)

핵심어: 하이퍼파라미터(hyperparameter)

이번에는 하이퍼파라미터 탐색을 어떻게 할 수 있는지 몇 가지 팀에 대해 알아보려고 한다.

Re-test hyperparameters occasionally



- NLP, Vision, Speech,
Ads, logistics,

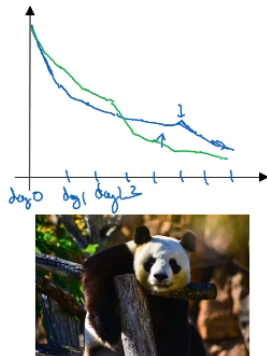
- Intuitions do get stale.
Re-evaluate occasionally.

Andrew Ng

오늘날 딥러닝은 여러 분야에 적용되고 있다. 한 어플리케이션에서 얻은 하이퍼파라미터에 대한 직관이 다른 영역에서 쓰일 수도, 아닐 수도 있다. 서로 다른 어플리케이션 영역 간에 공유되는 것들이 있는데, 예를 들면 컴퓨터 비전 커뮤니티에서 발전된 컨브넷이나 레스넷 등이 있는데, 이는 추후 다룰 것이다. 이것들은 음성에 잘 적용되며, 이 음성에서 발전된 아이디어들이 자연어 처리(NLP)에서도 잘 적용되고 있는 모습을 볼 수 있다. 즉 딥러닝 분야의 사람들이 다른 영역에서 영감을 얻기 위해 그 분야의 논문을 점점 많이 찾아 읽고 있는 것이다.

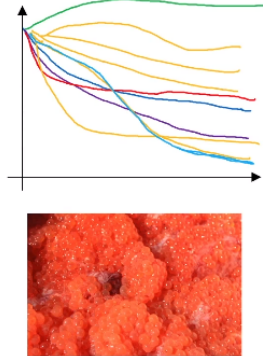
하지만 하이퍼파라미터를 찾는 과정은 그렇지 못하다는 직관을 얻었다. 로지스틱 문제 하나만 보더라도 좋은 하이퍼파라미터를 찾았다고 했을 때 알고리즘을 계속 발전시키거나 몇 달에 걸쳐 데이터가 바뀔 수도 있고, 데이터 센터의 서버를 업그레이드 시킬 수도 있는 것이다. 이러한 변화들 때문에 우리가 찾았던 하이퍼파라미터들이 녹스는 것이다. 그래서 다시 시험해보거나, 하이퍼파라미터들이 아직도 만족할만한 결과를 내는지 몇 달마다 재평가하기를 권한다.

Babysitting one model



Panda ←

Training many models in parallel



Caviar ←

Andrew Ng

결국 사람들이 하이퍼파라미터를 찾을 때 크게 두 가지 서로 다른 방법을 사용하는 것을 볼 수 있다. 하나는 모델 돌보기인데, 데이터는 방대하지만 CPU나 GPU 등 컴퓨터 자원이 많이 필요하지 않아서 적은 숫자의 모델을 한 번에 학습시킬 수 있을 때 사용하는 것이다. 이런 경우 학습 과정에서 모델 돌보기를 하는데, 예를 들어 0일차에 무작위하게 매개변수를 설정하고 학습을 시작하면 학습곡선에서 비용함수 J 나 개발 세트의 오차가 하루가 다르게 점진적으로 감소할 것이다. 이때 1일차 끝 무렵에 학습이 꽤나 잘 되었다면 학습 속도를 조금 올려서 조금 더 나은지 보자고 할 수 있다. 이런 식으로 성능을 올리는 것이다. 그리고 2일차에도 꽤 좋은 성과를 내고 있는 것 같다면 여기서도 모멘텀을 약간 올리거나 학습 속도를 약간 낮출 수 있을 것이다. 또 3일차에 들어가서도 다시 결과를 살펴본다. 그렇게 하이퍼파라미터를 계속 조절하다 보면 어떤 특정한 날에 학습 속도가 너무 커서 며칠 전의 모델로 돌아가기도 할 수 있다. 이렇게 며칠, 몇 주에 걸쳐 매일 모델을 돌보며 학습시키는 것이다. 이것이 모델 돌보기라는 한 가지 접근 방법이며, 이는 성능을 잘 지켜보다가 학습 속도를 조금씩 바꾸는 방식인 것이다. 이 방식은 여러 모델을 동시에 학습시킬 컴퓨터 자원이 충분하지 않을 때 사용한다.

다른 접근은 여러 모델을 함께 학습시키는 것인데, 우리가 갖고 있는 하이퍼파라미터를 며칠에 걸쳐 스스로 학습하게 하는 것이다. (그래프는 비용 함수 J 를 그리거나 학습 오차, 개발세트 어떤 수치를 나타내고 있을 것) 그리고 동시에 다른 모델의 다른 하이퍼파라미터 설정을 다루기 시작하는데, 그렇다면 두 번째 모델은 다른 학습 곡선을 그릴 것이다(보라색 곡선). 이에 더하여 동시에 세 번째 모델도 학습시킨다(빨간색

학습곡선). 또 다른 것은 초록색 곡선과 같이 발산한다고 하자. 이렇게 서로 다른 모델을 도시에 학습시키는 것이다(주황색 선들도 서로 다른 모델을 나타냄). 이 방법을 쓰면 여러 하이퍼파라미터 설정을 시험해볼 수 있다. 그리고 마지막에는 최고 성능을 보이는 것을 고르면 되는 것이다.

비유를 하자면, 왼쪽 접근은 판다와 같다. 판다는 한 번에 한 마리 씩만 아이를 가지며 아기 판다가 살아남을 수 있도록 많은 노력을 기울인다. 말 그대로 모델이나 아기 팬더를 '돌보기'하는 것이다. 한편 오른쪽 접근은 캐비어 전략이라고 할 수 있는데 이는 물고기와 비슷하기 때문이다. 한 철에 1억개의 알을 품는 물고기가 있는데, 물고기가 번식을 하는 과정은 하나에 많은 집중을 쏟기보다는 그 이상이 더 잘 살아남기를 바라며 그저 지켜보는 것이다. 이는 포유류의 번식과 어류, 파충류의 번식 차이와 비슷하다. 이 두 접근 중 무엇을 선택할 것인지는 컴퓨터 자원과 양의 함수 관계에 있다. 만약 여러 모델을 동시에 학습시키기에 충분한 컴퓨터를 가지고 있다면 캐비어 접근법으로 서로 다른 하이퍼파라미터들을 시험해볼 수 있을 것이다. 하지만 온라인 광고나 컴퓨터 비전 어플리케이션 등 많은 데이터가 쓰이는 곳에서는 학습시키고자 하는 모델이 너무 커서 한 번에 여러 모델을 학습시키기 어렵다. 물론 어플리케이션에 따라 큰 차이가 있지만, 이러한 경우 판다 접근을 주로 사용하는 것으로 보인다. 하나의 모델에 집중해 매개변수를 조금씩 조절하며 그 모델이 잘 작동하게끔 만드는 것이다. 하지만 판다 접근에서도 한 모델이 잘 작동하는지 확인한 뒤에 2주, 3주 후 다른 모델을 초기화해서 다시 돌보기를 할 수 있다. 이는 판다가 일생에 여러 마리의 새끼를 돌보는 것과 유사하다.

- 하이퍼파라미터 튜닝 방법은 두 가지가 있습니다.
- 모델 돌보기(baby sitting one model) = 판다 접근
 - 컴퓨터의 자원이 많이 필요하지 않거나, 적은 숫자의 모델을 한번에 학습시킬 수 있을 때 사용합니다.
 - 하나의 모델로 매일 성능을 지켜보면서, 학습 속도를 조금씩 바꾸는 방식입니다.
- 동시에 여러 모델 훈련(Training many models in parallel) = 캐비어 접근
 - 컴퓨터의 자원이 충분히 많아 여러 모델을 한번에 학습시킬 수 있을 때 사용합니다.

공감

'DL 스터디&프로젝트' 카테고리의 다른 글

[Euron 중급 세션 10주차] 캐글 필사 과제 - Pytorch Tutorial for Deep Learning Lovers (2)	2023.11.20
[Euron 중급 세션 10주차] 딥러닝 2단계 4. 최적화 알고리즘 (1)	2023.11.13
[Euron 중급 세션 9주차] 딥러닝 2단계 3. 최적화 문제 설정 (2)	2023.11.06
[Euron 중급 세션 8주차] 딥러닝 2단계 2. 신경망 네트워크의 정규화 (1)	2023.10.30