**튜닝 프로세스**

 지금까지 신경망을 학습시킬 때 여러 하이퍼파라미터들이 관여한다는 걸 배웠다. 그렇다면**좋은 하이퍼파라미터는 어떻게 찾을 수 있을까?** 체계적으로 하이퍼파라미터를 튜닝할 수 있는 팁을 알아보자.

**Hyperparameters**

 심층 신경망을 학습시킬 때 가장 어려운 일은 다뤄야 할**하이퍼파라미터**가 많다는 것이다. (우선 조정하는 순으로 나열)

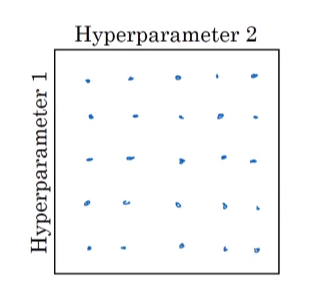
* 학습률 α
* Momentum 알고리즘의 β -> 기본값 0.9 설정
* 은닉 유닛의 수
* 미니배치의 크기
* 은닉층의 개수
* 학습률 감쇠(learning rate decay) 정도
* Adam 알고리즘의 β​1​​,β​2​​,ϵ -> 0.9, 0.999, 10^(-8)

 대부분의 학습에서는 일부 파라미터들이 다른 파라미터보다 중요하다.**우선 학습률 α는 튜닝해야 할 가장 중요한 하이퍼파라미터이다.** 학습률 α 이외에 주로 튜닝하는 것들로는 모멘텀이 있다. 기본값 0.9정도로 설정할 수 있다. 최적화 알고리즘을 효율적으로 돌리기 위해 미니 배치 크기도 튜닝할 수 있다. 은닉 유닛도 자주 튜닝한다. 오렌지색 글씨는 빨간색 글씨(학습률 α) 다음으로 중요한 것들이다. 보라색 글씨는 다음으로 중요한 것들이다. 그리고 Adam 알고리즘에서는 β​1​​,β​2​​,ϵ는 튜닝하지 않고 0.9, 0.999, 10^(-8)을 항상 사용한다. 물론 원한다면 튜닝을 사용해도 좋다.

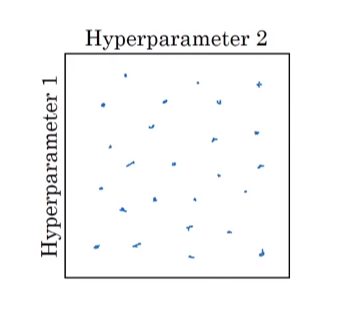
**Try random values : Don't use a grid**

 만약 하이퍼파라미터를 튜닝한다면**어떤 값을 탐색할지 어떻게 정할 수 있을까?**

i) **머신러닝이 만들어진지 얼마 되지 않았을 때**는 두 개의 하이퍼파라미터가 있을 때 각각 Hyperparameter1, Hyperparameter2라고 부르는데 **격자점을 탐색**하는 것이 일반적이었다. (아래 그림 참고) 그리고 체계적으로 여기 있는 값들을 탐색하는 것이다. (실제로는 더 크거나 작을 수도 있지만 여기서는 5x5 격자의 25개의 점만 생각함) 그리고 최고의 하이퍼파라미터를 정하는 것이다. 이 예시는 하이퍼파라미터의 수가 적을 때 쓸 수 있다.

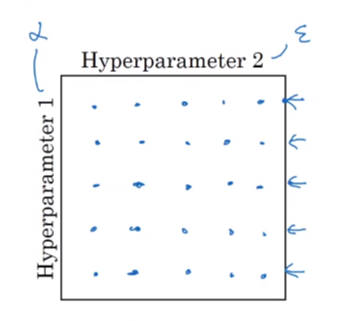


ii) 하지만**딥러닝**에서는 다음과 같은 방식을 추천한다. **무작위로 점들을 선택**하는 것이다. 위 상황과 동일하게 25개의 점만 생각해보자. 그 점들에 대해서 하이퍼파라미터를 정하는 것이다. 이렇게 하는 이유는 어떤 하이퍼파라미터가 문제 해결에 더 중요한지 미리 알 수 없기 때문이다.

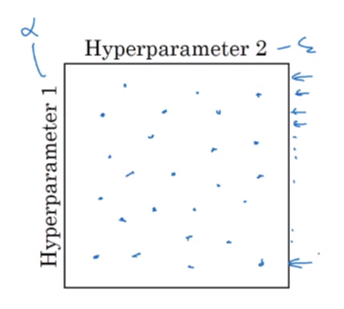


 위에서 설명했던 것처럼 하이퍼파라미터에는 중요도 순위가 있다. 예를 들어 Hyperparameter1이 학습속도 α이고 극단적인 경우로 Hyperparameter2를 ϵ라고 하자. (ϵ: Adam 알고리즘의 분모에 있는 값) 이런 경우 α를 고르는 것이 ϵ를 고르는 것 보다 더 중요하다.

 그래서**격자점(i의 상황)으로 다시 돌아가면** 5개의 α값을 확인하게 되는데 이때**ϵ값이 달라도 결과는 같은 것**을 확인할 수 있을 것이다.**25개의 모델을 학습시켰지만 가장 중요한 하이퍼파라미터인 α 5개에 대해서만 학습시킨 것**과 다를게 없다.



  반대로**무작위로 모델(ii의 상황)을 고르면** 어떨까? 그렇게 되면 **25개의 서로 다른 학습속도 α값을 이용하여 학습**시키게 되고 **더 좋은 하이퍼파라미터를 잘 찾게 될 것이다.**



 여기서는 두 개의 하이퍼파라미터만 써서 예를 들었지만 실제로는 훨씬 많은 하이퍼파라미터를 다루게 될 것이다. 예컨대 세 개의 하이퍼파라미터를 다룬다고 하면 정사각형이 아닌 정육면체를 탐색하게 되는 것이다. 그리고 세번째 차원은 Hyperparameter3를 가르키게 된다. 그 3차원 정육면체 안에서 모델을 고른다면 각 하이퍼파라미터에 대해서 훨씬 많은 값을 시험해보게 될 것이다. 실제로는 3보다 더 많은 하이퍼파라미터를 탐색하곤 한다. 그리고 어플리케이션에서 어떤 하이퍼파라미터가 가장 중요한지 미리 알기 어렵다.

=> 격자점보다 **무작위로 모델을 정하는 것이 가장 중요한 하이퍼파라미터의 다양한 값을 탐색할 수 있다.** 무엇이 중요하든 상관없이 말이다.

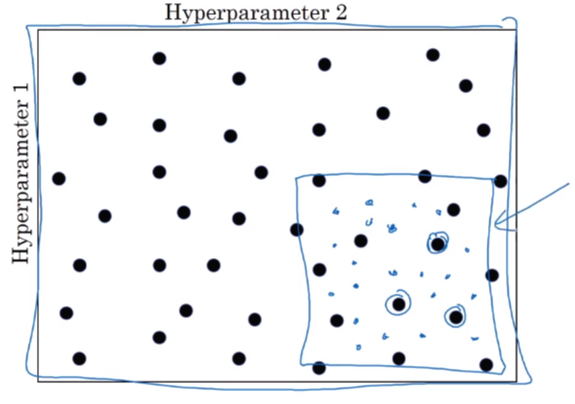
**Coarse to fine**

**다른 일반적 방법 중 하나는 정밀화 접근**이다. 아래 2차원 예시에서 이 점들을 사용한다고 해보자. 그리고 파란색 동그라미의 점이 최고라는 것을 찾았다. 그렇다면 아마도 그 주변에 있는 점들도 좋은 성능을 보일 것이다.

스크린샷, 도표, 패턴이(가) 표시된 사진

자동 생성된 설명

 그러면**정밀화 접근에서는 더 작은 영역으로 확대해서 더 조밀하게 점들을 선택**한다. 무작위인 것은 그대로지만 최고의 하이퍼파라미터들이 이 영역에 있으리라는 믿음 하에 **파란색 사각형(더 작은 범위의 사각형) 안에 초점을 두고 탐색하는 것**이다.



 즉, **전체 사각형에서 탐색한 뒤에 더 작은 사각형으로 범위를 좁혀 나가는 것**이다. 그렇다면 작은 사각형안에서 더 조밀하게 시험해볼 수 있다. 이러한 정밀화 접근도 자주 쓰이는 방식이다. 그리고 이렇게 하이퍼파라미터의 여러 값들을 시험해보며 학습의 목표나 개발 목표 등에 있어서 최고의 파라미터를 고르는 것이다.

 이번 강의에서는 하이퍼파라미터를 찾는 방법을 체계적으로 정리했다.  이보다 더 많은 종류의 탐색이 있지만 우리가 **반드시 알아야할 두가지**는...

**첫째, 격자점이 아닌 무작위이다.**

**둘째, 원한다면 정밀화 접근을 이용할 수 있다.**

**적절한 척도 선택하기**

 지난 강의에서 **무작위로 하이퍼파라미터를 찾는 것이 더 효율적인 탐색**이라는 것을 배웠다. 하지만 여기서 무작위라는 것이 가능한 값들 중 공평하게 뽑는 것이라고 할 수는 없다. 대신 적절한 척도를 정하는 것이 더욱 중요하다. 어떻게 척도를 정하는지 알아보자.

**Checking hyperparameters at random**

i) 어떤 레이어 l에 대해서 은닉 유닛의 수 n^l을 정한다고 하자. 값의 범위는 50부터 100이다. 이런 경우 50부터 100까지의 수직선에서 무작위하게 값들을 고른다고 하자. 이는 하이퍼파라미터를 고르는 꽤 합리적인 방법이다.

ii) 또는 신경망에서 레이어의 숫자 L을 정한다고 했을 때 층의 숫자가 2에서 4사이라고 생각할 수 있다. 2에서 4까지의 숫자를 선택할 때 무작위하게 뽑는 것을 물론이거니와 격자점을 사용해도 문제가 없다.

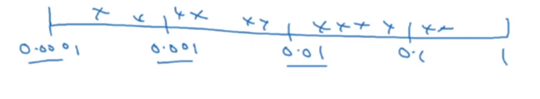
다음 i), ii)과 같은 예시들은 가능한 값 중 무작위하게 뽑는 것이 합리적인 경우이다. **하지만 모든 하이퍼파라미터가 그렇지는 않다.**다른 예시들을 살펴보자.

 iii) **학습률 α를 탐색**하는데 **범위로 0.0001부터 1까지**를 생각하고 있다고 하자. 0.0001에서 1까지의 수직선 상에서 균일하게 **무작위로 값을 고르게 되면, 90%의 값이 1과 0.1사이에 존재**할 것이다. 즉, **90%의 자원을 0.1과 1사이를 탐색**하는 데 쓰고 단 10%만을 0.0001과 0.1사이를 탐색하는 데 쓰는 것이다. 이는 **공평하다고 할 수 없다.**

친필, 라인, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

**따라서 선형척도 대신 로그척도에서 하이퍼파라미터를 찾는 것이 더 합리적**이다. 수직선위에 0.0001부터 0.001, 0.01, 0.1, 1까지 값들이 있을 것이다. 이런 로그 척도에서 균일하게 무작위로 값을 뽑는 것이다. 그러면 0.0001과 0.001사이, 0.001과 0.01 사이를 탐색할 때 더 많은 자원을 쓸 수 있는 것이다.



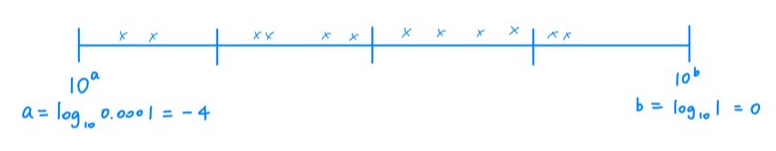
**파이썬에서는 어떻게 구현**할 수 있을까?

r=-4\*np.random.rand() # -4~0사이의 무작위 값

a=10^r # 10^(-4)~10^0 사이의 값

 r=-4\*np.random.rand()를 쓰면 무작위로 선택된 α값은 10^r이 된다. 첫번째 줄에서 r은 -4와 0사이의 무작위 값이고 α는 10^(-4)와 10^0(== 1)사이가 된다.

 더 일반적인 경우를 살펴보자. 10^a에서 10^b까지 로그 척도로 탐색한다면 수직선이 시작값이 10^a가 될 것이다. 따라서 a는 10을 밑으로 하는 log를 0.0001에 대해 취하면 얻을 수 있다. 그 결과 a는 -4가 될 것이다. 마찬가지로 수직선의 끝 값은 10^b이다. 여기서 b는 10이 밑인 log를 1에 취하면 0이라는 것을 알 수 있다. 그리고 어떻게 할까? **r은 a와 b사이서 균일하게 무작위로 뽑힌다.** 이 경우**r은 -4와 0사이다**. 그리고**무작위 하이퍼파라미터 α는 10^r**이 된다.



  다시 살펴보면 낮은 값에서 log를 취해서 a를 찾고 높은 값에서 log를 취해 b를 찾는다. 이렇게 10^a에서 10^b까지를 로그 척도로 탐색하는 것이다. r은 a와 b사이에서 균일하게 무작위로 뽑으면 하이퍼파라미터가 10^r이 되는 것이다. 이렇게 로그 척도에서 샘플링 하는 법을 배워보았다.

**Hyperparameters for exponentially weighted averages**

 또 다른 예시는**지수가중평균을 계산할 때 사용되는 하이퍼파라미터 β에 관한 것**이다. β​를 0.9와 0​dp r를 를 0.9와 0.999 범위에서 찾는다고 하자. (바로 직전 포스팅 참고하길 바람)

0.9의 경우에는 지수가중평균이 최근 10일의 평균기온처럼 마지막 10개 값의 평균과 비슷하고 0.999의 경우에는 마지막 1000개 값의 평균과 비슷했던 것을 기억하는가? **바로 위에서 설명했던 것처럼 만약 0.9와 0.999 사이를 탐색한다면 그 사이를 무작위 탐색하는 것은 합리적이지 않다.**

친필, 텍스트, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**더 나은 방법은 1-β​dp eogotj dp 에 대해서 값을 탐색**하는 것이다. **그러면 0.1부터 0.0001이 된다**. 그러면 β​fmf 를 0.1에서 0.01을 거쳐 0.001사이에서 탐색하는 것이다. 그렇다면 10^(-1)에서 10^(-3)이다. (위의 설명에서는 작은 값이 왼쪽에 큰 값이 오른쪽에 있었지만 여기서는 반대임) 즉, 여기서 해야할 일은 -3과 -1사이에서 균일하게 무작위로 값을 뽑는 것이다.**1-β를 10^r로 생각하면 되니 β가 1-10^r이 되는 것이다.**

텍스트, 친필, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

 이렇게 적절한 척도 위에서 무작위로 하이퍼파라미터 샘플은 추출한다. 이 방법을 이용하면 0.9부터 0.99를 탐색할 때와 0.99부터 0.999를 탐색할 때 동일한 양의 자원을 사용할 수 있다.

 이쯤 되면 왜 이러한 방법이 필요한지 수학적 증명이 궁금할 것이다. **왜 선형 척도에서 샘플을 뽑는 것은 좋지 않을까?** **만약 β가 1에 가깝다면 β가 아주 조금만 바뀌어도 결과가 아주 많이 바뀌게 된다.** 예를들어 β가 0.9에서 0.9005로 바뀌었다면 결과에 거의 영향을 주지 않는다.(1번 상황) 하지만 β가 0.999에서 0.9995로 바뀌었다면 알고리즘 결과에 큰 영향을 줄 것이다.(2번 상황) 1번의 경우, 대략 10개의 값을 평균내는 것이지만 2번의 경우에는 마지막 1000개의 값의 지수가중평균을 내는 것에서 마지막 2000개 값의 평균을 내는 것으로 바뀌었기 때문이다.**왜냐하면 1/(1-β)라는 식이 β가 1에 가까워질수록 작은 변화에도 민감하게 반응하기 때문이다.** **따라서 β가 1보다 가까운 곳에서 더 조밀하게 샘플을 뽑는다.** 반대로 1-β는 0이 가까운 곳이 된다. 따라서 가능한 결과 공간을 탐색할 때 더 효율적으로 샘플을 추출할 수 있는 것이다.

친필, 폰트, 텍스트, 서예이(가) 표시된 사진

자동 생성된 설명

 이 강의를 통해 하이퍼파라미터 샘플을 고를 때 적절한 척도를 고를 수 있게 되길 바란다. **만약 하이퍼파라미터를 고를 때 적절한 척도를 사용하지 않더라도 크게 걱정하지 않아도 된다.** 다른 척도가 우선하는 상황에서 균일한 척도에서 샘플링을 하더라도**정밀화 접근을 사용**하면 괜찮을 결과를 얻을 수 있다. 그래서 반복할수록 더 유의미한 하이퍼파라미터 범위를 탐색하게 되는 것이다.

**하이퍼파라미터 튜닝 실전**

 이전까지는 하이퍼파라미터를 탐색하는 방법을 설명하지 않았다. 하이퍼파라미터 탐색을 마무리하기에 앞서 탐색을 어떻게 할 수 있는지 몇 가지 팁을 알려주고자 한다.

**Re-test hyperparameters occasionally**

 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 있다. 예를 들면 컴퓨터 비전 커뮤니티에서 발전된 컨브넷이나 레스넷 등이 있다. (이는 강의 뒷부분에서 다시 다룰 것임) 이것들은 음성에 잘 적용되고 있다. 그리고 이 음성에서 발전된 아이디어 들이 자연어 처리에서도 잘 적용되고 있는 것을 보았다.

텍스트, 스크린샷, 화이트, 폰트이(가) 표시된 사진

자동 생성된 설명

 즉, 딥러닝 분야의 사람들이 다른 영역에서 영감을 얻기 위해 그 분야의 논문을 점점 많이 찾아 읽고 있다. 하지만 하이퍼파라미터를 찾는 과정은 그렇지 못하다는 직관을 얻었다.

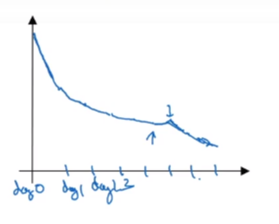
 로지스틱 문제 하나만 보더라도 여러분이 좋은 하이퍼파라미터를 찾았다고 할 때 알고리즘을 계속 발전시키거나 몇 달에 걸쳐 데이터가 바뀔 수도 있다. 데이터 센터의 서버를 업그레이드 시킬 수도 있다. 이러한 변화들 때문에 여러분이 찾았던 하이퍼파라미터가 녹스는 것이다. 그래서 다시 시험해보고나 하이퍼파라미터들이 아직도 만족할만한 결과를 내는지 몇 달마다 재평가하기를 권한다.

 결국 사람들이**하이퍼파라미터를 찾을 때 크게 두 가지 서로 다른 방법**을 사용한다.

**Babysitting one model**

 첫번째 방법은 **모델 돌보기**이다. **데이터는 방대하지만 CPU나 GPU 등 컴퓨터 자원이 많이 필요하지 않아서 적은 숫자의 모델을 한번에 학습시킬 수 있을 때 사용**한다. 이런 경우에 학습과정에서 모델 돌보기를 사용한다.

 예를 들어 0일차에 무작위하게 매개변수를 설정하고 학습을 시작했다. 그러면 학습곡선에서 비용함수 J나 개발 세트의 오차가 하루가 다르게 점진적으로 감소할 것이다. 1일차 끝 무렵에 학습이 꽤나 잘 된다. 그러고 나면 학습 속도를 조금 올려서 조금 더 나은지 보자고 말할 수 있다. (이런 식으로 성능을 올려나가는 것임) 그리고 2일차에도 꽤 좋은 성과를 내고 있는 것을 볼 수 있다. 여기서도 모멘텀을 약간 올리거나 학습속도를 약간 낮출 수 있다. 그리고 3일차에 들어선다... 그렇게 하이퍼파라미터를 계속 조절하다보면 어떤 날엔 학습속도가 너무 커서 몇 일 전으로 돌아가기도 한다. 이렇게 며칠, 몇 주에 걸쳐 매일 모델을 돌보며 학습시키는 것이다.



 이와 같이 **모델 돌보기는 성능을 잘 지켜보다가 학습 속도를 조금씩 바꾸는 방식**이다. **이 방식은 여러 모델을 동시에 학습 시킬 컴퓨터의 자원이 충분치 않을 때 사용**한다.

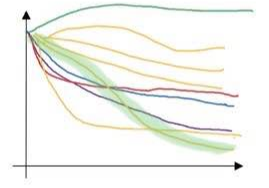
**Training many models in parallel**

 또 다른 접근은 **여러 모델을 함께 학습**시키는 것이다. 여러분이 갖고 있는 하이퍼파라미터를 며칠에 걸쳐 스스로 학습하게 한다. 그럼 아래와 같은 함수(파란색)를 그리게 된다. 비용함수 J를 그린 것 일수도 있다. 학습 오차나 개발 세트의 오차 등 어떤 수치를 나타내고 있을 것이다. **그리고 동시에 다른 모델의 다른 하이퍼파라미터 설정을 다루기 시작**한다. 두번째 모델(보라색)은 다른 학습 곡선을 그리게 된다. 아래 보이는 것처럼 말이다. (두번째 모델이 더 나은 그래프를 그리고 있음) 그리고 동시에 세번째 모델(빨간색)도 학습시킨다. 학습 곡선이 아래와 같이 그려진다. 또 다른 것들(주황색, 초록색)은 발산하는 그래프 모양을 그린다. 아래아 같이 서로 다른 모델을 동시에 학습시키는 것이다. 아래 주황색 선들도 서로 다른 모델을 나타낸다.

라인이(가) 표시된 사진

자동 생성된 설명

 이 방법을 쓰면 여러 하이퍼파라미터 설정을 시험해볼 수 있다. 그리고 마지막에 최고 성능을 보이는 것을 고르는 것이다. 이 예시에서는 아마 다음 곡선이 최고의 곡선일 것이다.



|  |  |
| --- | --- |
| **Babysittine one model** | **Training many models in parallel** |
| **팬더**와 같다. (팬더는 한 번에 한 마리씩만 아이를 갖는다) 그리고 아기 팬더가 살아남을 수 있도록 정말 많은 노력을 기울인다. 말 그대로 모델이나 아기 팬더를 '돌보기'하는 것이다.  - 포유류의 번식 과정 | **캐비어 전략**이라고도 한다. (물고기랑 비슷하기 때문) 한 철에 1억개의 알을 품는 물고기가 있다. 물고기가 번식하는 과정은 하나에 많은 집중을 쏟기보다 하나 또는 그 이상이 더 잘 살아남기를 그지 지켜본다.  - 어류, 파충류의 번식 과정 |

 이 두 접근중에 무엇을 선택할지는 컴퓨터 자원의 양과 함수에 관계가 있다. 만약 **여러 모델을 동시에 학습시키기에 충분한 컴퓨터를 갖고 있다면 물론 캐비어 접근을 사용**해서 서로 다른 하이퍼파라미터를 시험해 볼 수 있다. **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 **주로 팬더 접근을 사용**한다. 하나의 모델에 집중해 매개변수를 조금씩 조절하며 그 모델이 잘 작동하게끔 만드는 것이다. 하지만 **팬더 접근에서도 한 모델이 잘 작동하는지 확인한 뒤에 2-3주 뒤에 다른 모델을 초기화해서 다시 돌보기를 할 수 있다.** 팬더가 일생에 여러 마리의 새끼를 돌보는 것처럼 말이다. 한 번에는 하나 혹은 아주 적은 숫자의 새끼만 돌보지만 말이다.