5. 하이퍼파라미터 튜닝

1. 튜닝 프로세스

- 좋은 하이퍼파라미터는 어떻게 찾을 수 있을까?

1. **Hyperparameters**

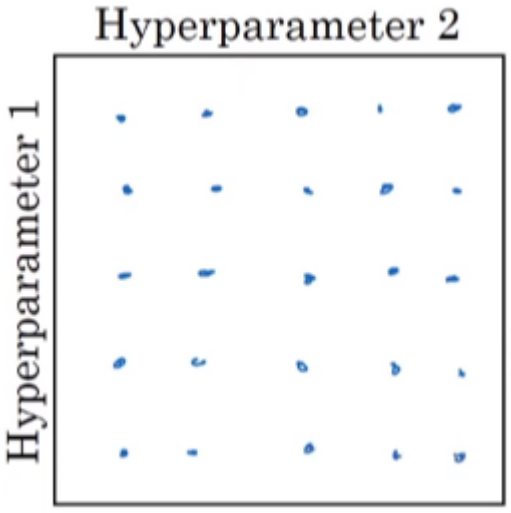
- 심층 신경망을 학습시킬 때 가장 어려운 일: 다뤄야 할 하이퍼파라미터가 많다는 것
  - **학습률 α : 튜닝해야 할 가장 중요한 하이퍼파라미터**
  - **Momentum 알고리즘의 β -> 기본값 0.9 설정**
  - **은닉 유닛의 수**
  - **미니배치의 크기**(최적화 알고리즘을 효율적으로 돌리기 위함)
  - **은닉층의 개수**
  - **학습률 감쇠(learning rate decay) 정도**
  - Adam 알고리즘의 β1,β2,ε -> 0.9, 0.999, 10^(-8)
  - 중요도: 빨 > 파 > 보

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

1. **머신러닝이 만들어진지 얼마 되지 않았을 때**

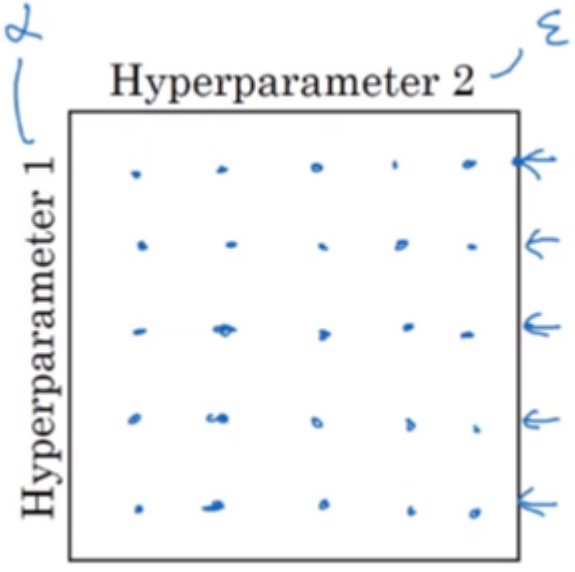
- 두 개의 하이퍼파라미터가 있을 때 각각 Hyperparameter1, Hyperparameter2라고 부른다.

→ **격자점을 탐색**하는 것이 일반적



- 체계적으로 여기 있는 값들을 탐색. (실제로는 더 크거나 작을 수도 있지만 여기서는 5x5 격자의 25개의 점만 생각함)
- 최고의 하이퍼파라미터 정하기.
- 이 예시는 **하이퍼파라미터의 수가 적을 때** 쓸 수 있다.

ex) Hyperparameter1: 학습속도 α, 극단적인 경우로 Hyperparameter2: ε라고 하자. (ε: Adam 알고리즘의 분모에 있는 값)



→ 이런 경우 α를 고르는 것이 ε를 고르는 것 보다 더 중요

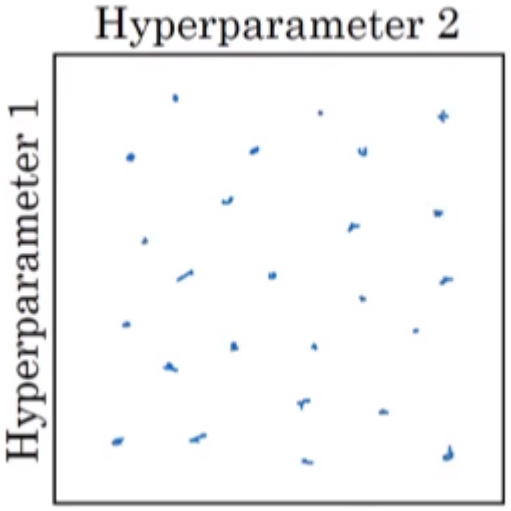
→ 5개의 α값을 확인하게 되는데 이때 **ε값이 달라도 결과는 같은 것**을 확인 가능

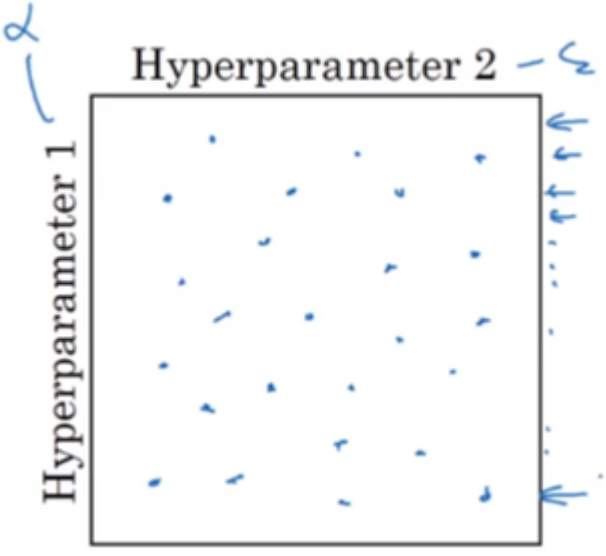
= **25개의 모델을 학습시켰지만 가장 중요한 하이퍼파라미터인 α 5개에 대해서만 학습시킨 것**과 다를게 없다.

2. **딥러닝에서**

- **무작위로 점들을 선택**하는 것 추천

→ 어떤 하이퍼파라미터가 문제 해결에 더 중요한지 미리 알 수 없기 때문

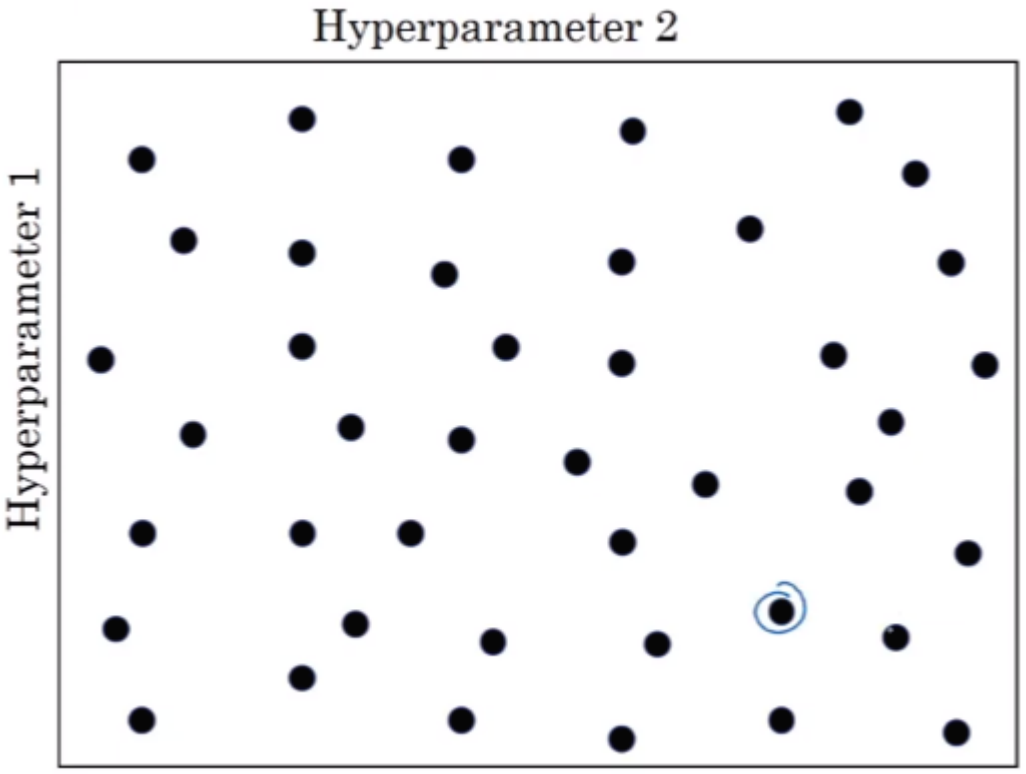




→ **25개의 서로 다른 학습속도 α값을 이용하여 학습**시키게 되고 **더 좋은 하이퍼파라미터를 잘 찾게 될 것**

3. **Coarse to fine**

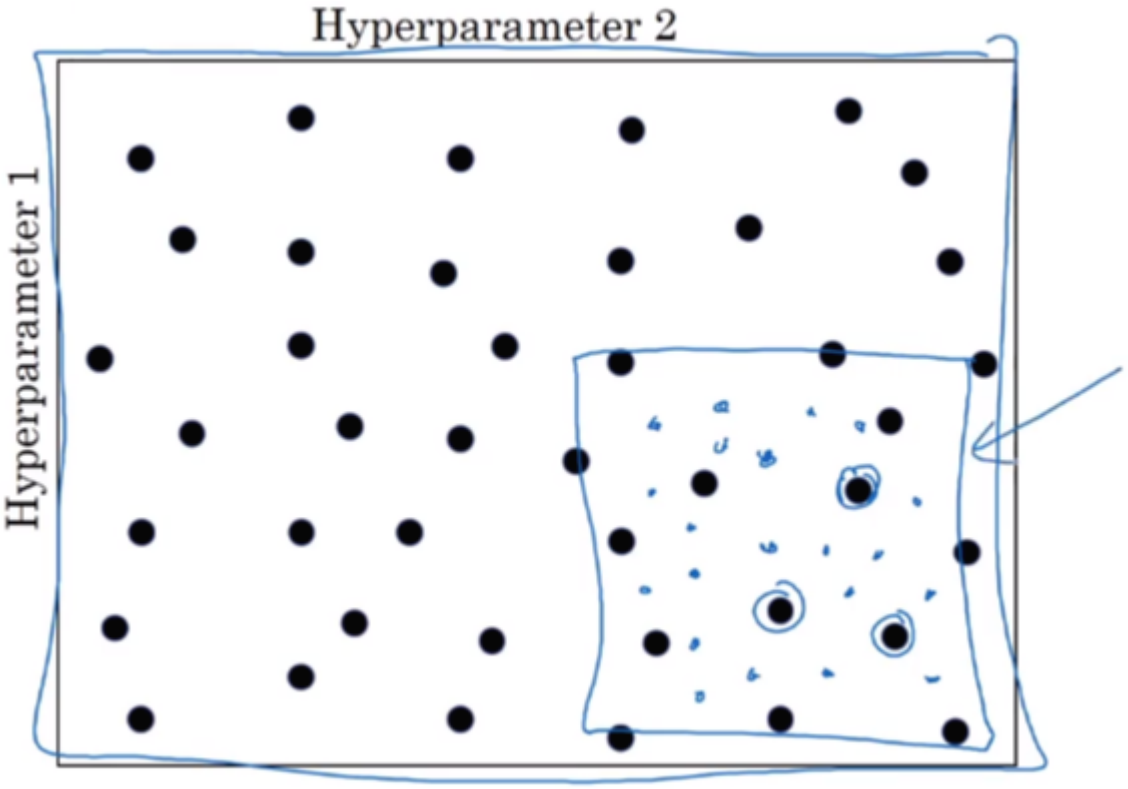
- **다른 일반적 방법 중 하나는 정밀화 접근**



→ 파란색 동그라미의 점이 최고라는 것을 찾았다. 그렇다면 아마도 그 주변에 있는 점들도 좋은 성능을 보일 것

- 정밀화 접근에서는 더 작은 영역으로 확대해서 더 조밀하게 점들을 선택

→ 무작위인 것은 그대로지만 최고의 하이퍼파라미터들이 이 영역에 있으리라는 믿음 하에 **파란색 사각형(더 작은 범위의 사각형) 안에 초점을 두고 탐색하는 것**



→ 전체 사각형에서 탐색한 뒤에 더 작은 사각형으로 범위를 좁혀 나가는 것

반드시 알아야할 두가지

1. **격자점이 아닌 무작위이다.**
2. **원한다면 정밀화 접근을 이용할 수 있다.**

2. 적절한 척도 선택하기

1. **Checking hyperparameters at random**

1) 어떤 레이어 l에 대해서 **은닉 유닛의 수 n^l**을 정한다고 하자. 값의 범위는 50부터 100

- 50부터 100까지의 수직선에서 무작위하게 값들을 고른다고 가정

→ 하이퍼파라미터를 고르는 꽤 합리적인 방법이다.

2) 신경망에서 레이어의 숫자 L을 정한다고 했을 때 층의 숫자가 2에서 4 사이라고 생각할 수 있다.

- 2에서 4까지의 숫자를 선택할 때 무작위하게 뽑거나 격자점을 사용해도 문제 X

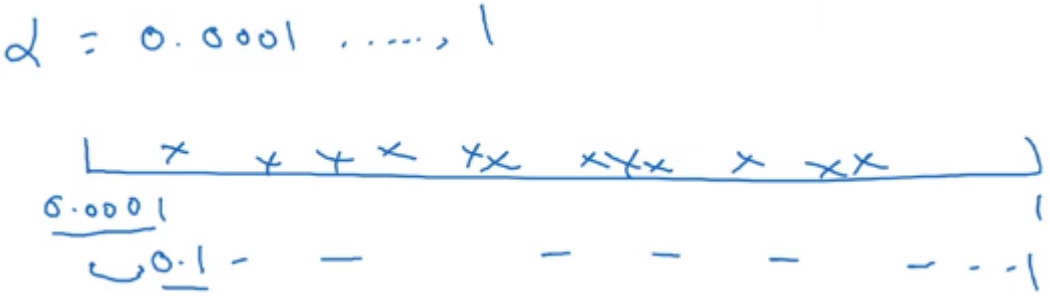
→ 1), 2) 와 같은 예시들은 가능한 값 중 무작위하게 뽑는 것이 합리적인 경우

하지만 모든 하이퍼파라미터가 그렇지는 않다.

3) 학습률 α를 탐색하는데 범위로 0.0001부터 1까지라고 가정

- 0.0001에서 1까지의 수직선 상에서 균일하게 **무작위로 값을 고르게 되면, 90%의 값이 1과 0.1사이에 존재**할 것이다. 즉,
- 즉, **90%의 자원을 0.1과 1사이를 탐색**하는 데 쓰고 단 10%만을 0.0001과 0.1사이를 탐색하는 데 쓰는 것이다.

→ **공평하다고 할 수 없다.**



- **선형척도 대신 로그척도에서 하이퍼파라미터를 찾는 것이 더 합리적**

→ 수직선 위에 0.0001부터 0.001, 0.01, 0.1, 1까지 값들이 있을 것이다. 이런 로그 척도에서 균일하게 무작위로 값을 뽑는 것

- 파이썬 구현

```
r=-4*np.random.rand() # -4~0사이의 무작위 값
a=10^r # 10^(-4)~10^0 사이의 값
```

- 더 일반적인 경우

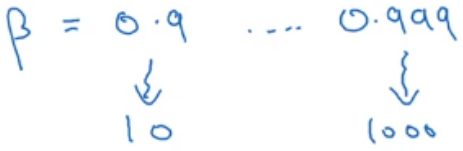
1. 10^a에서 10^b까지 로그 척도로 탐색한다면, 수직선이 시작값이 10^a가 될 것.
2. 따라서 a는 10을 밑으로 하는 log를 0.0001에 대해 취하면 얻을 수 있다.
3. 그 결과 a = -4가 될 것.
4. 마찬가지로 수직선의 끝 값은 10^b이다. (여기서 b는 10이 밑인 log를 1에 취하면 0이라는 것을 알 수 있다.)
5. **r은 a와 b사이서 균일하게 무작위로 뽑힌다.** 이 경우 **r은 -4와 0사이다**.
6. **무작위 하이퍼파라미터 α는 10^r**이 된다.

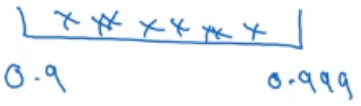
2. **Hyperparameters for exponentially weighted averages**

- 지수가중평균을 계산할 때 사용되는 하이퍼파라미터 β에 관한 것

ex) β를 0.9와 0.999 범위에서 찾는다고 하면,

- 0.9의 경우] 지수가중평균이 최근 10일의 평균기온처럼 마지막 10개 값의 평균과 비슷
- 0.999의 경우] 마지막 1000개 값의 평균과 비슷
- 0.**9와 0.999 사이를 탐색한다면 그 사이를 무작위 탐색하는 것은 합리적이지 않다.**





→ **더 나은 방법은 1-β 에 대해서 값을 탐색**하는 것

= β를 0.1에서 0.01을 거쳐 0.001사이에서 탐색하는 것 (10^(-1) ~ 10^(-3))

→ -3과 -1사이에서 균일하게 무작위로 값을 뽑기

- **1-β = 10^r 이므로, β = 1-10^r**
- 이 방법을 이용하면 0.9부터 0.99를 탐색할 때와 0.99부터 0.999를 탐색할 때 동일한 양의 자원을 사용할 수 있다.

- 왜 선형 척도에서 샘플을 뽑는 것은 좋지 않을까?
  - 만약 β가 1에 가깝다면 β가 아주 조금만 바뀌어도 결과가 아주 많이 바뀌게 된다.
  - **1/(1-β)라는 식이 β가 1에 가까워질수록 작은 변화에도 민감하게 반응하기 때문.**

  → **따라서 β가 1보다 가까운 곳에서 더 조밀하게 샘플을 뽑는다.**

3. 하이퍼파라미터 튜닝 실전

1. **Re-test hyperparameters occasionally**

- 서로 다른 어플리케이션 영역 간에 공유되는 것들
  - 컴퓨터 비전 커뮤니티에서 발전된 컨브넷이나 레스넷 등 → 음성에 잘 적용됨
  - 음성에서 발전된 아이디어들이 자연어 처리에서도 잘 적용되고 있음

- 하이퍼파라미터들이 만족할만한 결과를 내는지 몇 달마다 재평가하기를 권장

2. **Babysitting one model**

- 모델 돌보기: 데이터는 방대하지만 CPU나 GPU 등 컴퓨터 자원이 많이 필요하지 않아서 적은 숫자의 모델을 한번에 학습시킬 수 있을 때 사용
  - ex) 0일 차에 무작위하게 매개변수를 설정하고 학습을 시작.

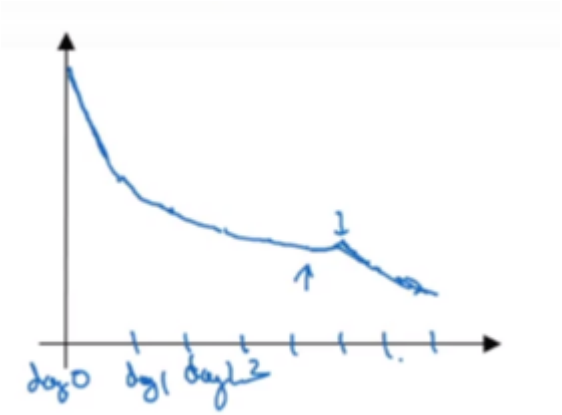
  → 학습 곡선에서 비용함수 J나 개발 세트의 오차가 하루가 다르게 점진적으로 감소할 것.
  - 1일 차 끝 무렵에 학습이 꽤 잘 된다.

→ 학습 속도를 조금 올려서 조금 더 나은지 보자고 말할 수 있다. (이런 식으로 성능을 올려나가는 것)

- 2일 차에도 꽤 좋은 성과를 내고 있는 것을 볼 수 있다.

→ 여기서도 모멘텀을 약간 올리거나 학습속도를 약간 낮출 수 있다.

- 하이퍼파라미터를 계속 조절하다보면 어느 날엔 학습 속도가 너무 커서 몇 일 전으로 돌아가기도 한다.
- 이렇게 며칠, 몇 주에 걸쳐 매일 모델을 돌보며 학습 시키는 것

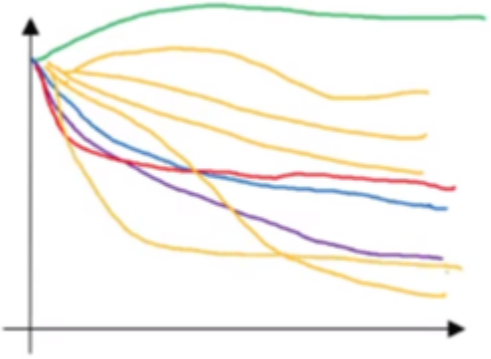


→ **모델 돌보기: 성능을 잘 지켜보다가 학습 속도를 조금씩 바꾸는 방식**이다. **이 방식은 여러 모델을 동시에 학습 시킬 컴퓨터의 자원이 충분치 않을 때 사용**

→ 팬더와 같다. (팬더는 한 번에 한 마리씩만 아이를 갖는다) 모델이나 아기 팬더를 '돌보기'하는 것이다.

3. **Training many models in parallel**

- 여러 모델을 함께 학습시키는 것



- 하이퍼파라미터를 며칠에 걸쳐 스스로 학습하게 한다. → 함수(파란색)를 그리게 된다. (비용함수 J를 그린 것 일수도 있음.)
    - 학습 오차나 개발 세트의 오차 등 어떤 수치를 나타내고 있을 것
    - **동시에 다른 모델의 다른 하이퍼파라미터 설정을 다루기 시작**한다.
- 두번째 모델(보라색)은 다른 학습 곡선을 그리게 된다. (두번째 모델이 더 나은 그래프를 그리고 있음)
    - 동시에 세번째 모델(빨간색)도 학습시킨다.
- 또 다른 것들(주황색, 초록색)은 발산하는 그래프 모양을 보인다.

- **여러 하이퍼파라미터 설정을 시험해볼 수 있다.** 마지막에 최고 성능을 보이는 것을 고르는 것

→ **캐비어 전략**이라고도 한다. (물고기랑 비슷하기 때문) 물고기가 번식하는 과정은 하나에 많은 집중을 쏟기보다 하나 또는 그 이상이 더 잘 살아남기를 그저 지켜본다.

4. **두 가지 방법 중 어떤 걸 선택?**

- 컴퓨터 자원의 양과 함수 관계가 있다.
- 만약 **여러 모델을 동시에 학습시키기에 충분한 컴퓨터를 갖고 있다면**

→ **캐비어 접근을 사용**해서 서로 다른 하이퍼파라미터를 시험해 볼 수 있다.

- **온라인 광고나 컴퓨터 비전 어플리케이션 등 많은 데이터가 쓰이는 곳에서는 학습시키고자 하는 모델이 너무 커서 한 번에 여러 모델을 학습시키기 어렵다.**

→ 어플리케이션에 따라 큰 차이가 있지만 **주로 팬더 접근을 사용**한다. 하나의 모델에 집중해 매개변수를 조금씩 조절하며 그 모델이 잘 작동하게 만드는 것이다. 하지만 **팬더 접근에서도 한 모델이 잘 작동하는지 확인한 뒤에 2-3주 뒤에 다른 모델을 초기화해서 다시 돌보기를 할 수 있다.**