Week5 학습정리

Week5-1

1. Rest API

API : 데이터를 주고 받는 수단

- API
  - "An application programming interface (API) is a connection between computers or between computer programs."
  - **Resource(데이터)를 web으로 주고 받는 수단**
- Client-Server Architecture
  - Client: Request / Server: Response

REST API 란?

- WHAT → **URL** : URL을 정의할 수 있어야 한다.
  - Represent Resource : URL을 통해 데이터 대표 가능
  - Noun type : 정보 명사형 지정
  - ex. GET /username/jin
- HOW → **HTTP Methods**
  - 대표 **HTTP Methods :** GET, POST, PUT, DELETE

2. FastAPI

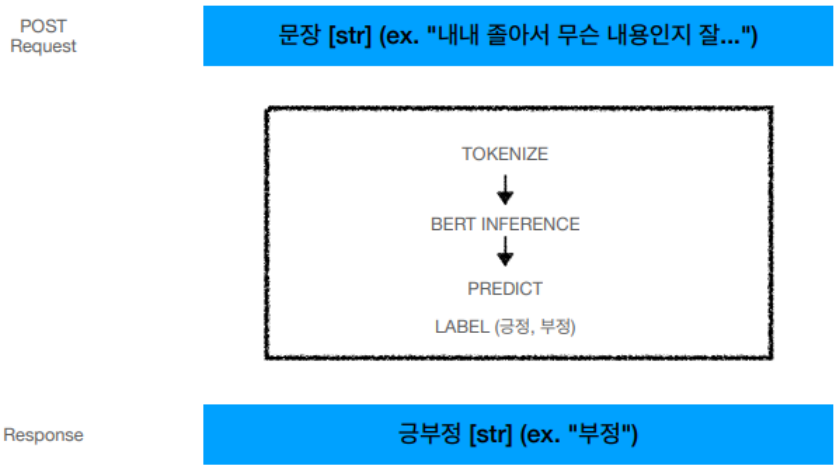
- micro web framework
  - ex. Flask(동기만 지원), FastAPI(비동기 지원)
- FaskAPI 특징
  - Pydantic
    - Automatic documentation & data validation
  - SwaggerUI
    - GUI
  - ASGI (Asynchronous Server Gateway Interface)
    - Use "async def", and get Faster Speed

3. Installation

튜토리얼 파일

- https://github.com/ChristinaROK/PreOnboarding_AI

4. Tutorial



5. Reference

- REST API 한글 설명 블로그
- HTTP Methods 설명 영문 블로그
- FAST API

Week5-3

Tuning hyper-parameters

하이퍼 파라미터는 파라미터와 다르다. 파라미터는 모델이 학습을 하면서 최적의 값을 찾는 반면 하이퍼 파라미터는 학습 전에 고정된 값을 제공해야 한다. 머신 러닝 모델은 파라미터 수가 딥러닝 모델보다 적기 때문에 학습을 여러 번 수행하며 자동으로 최적의 하이퍼 파라미터를 찾아주는 auto hyper-parameter tuning 기법이 많지만, 한 번 학습할 때 시간과 자원이 많이 소요되는 딥러닝 모델은 auto hyper-parameter tuning을 수행하기 쉽지 않다. 따라서 딥러닝 모델은 어떻게 최적의 하이퍼 파라미터를 찾을 수 있는지 알아보자.

하이퍼 파라미터 종류

1. Optimizer

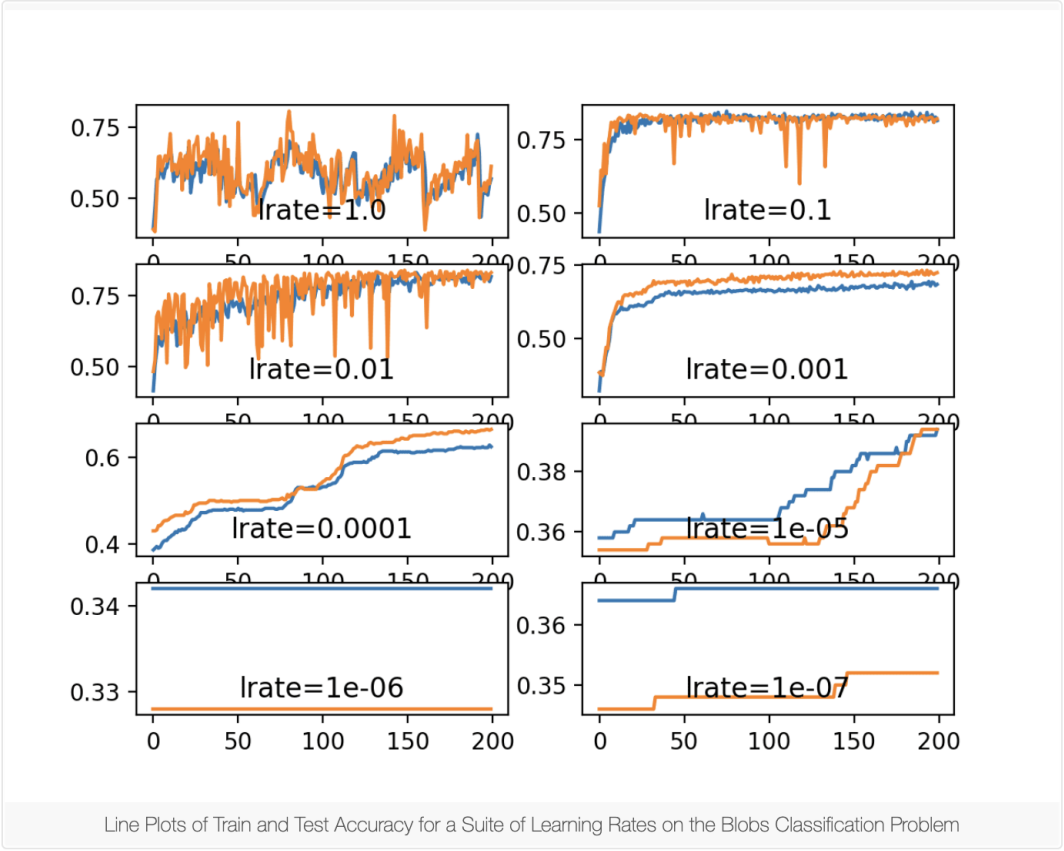
   Optimizer는 모델의 파라미터를 업데이트한다. Optimizer 종류마다 서로 다른 방식으로 파라미터가 이동할 **방향**과 **크기**를 정의한다. 논문에서 자주 등장하는 AdamW optimizer는 4개의 하이퍼 파라미터를 매개변수로 갖는다. lr(learning rate), betas, eps, 그리고 weight decay가 있다. 보통 하이퍼 파라미터를 튜닝할 때는 lr를 바꿔가며 성능 변화를 측정한다. 나머지 하이퍼 파라미터(betas, eps, weight decay)는 AdamW 저자가 찾은 최적의 값을 사용한다.

   1. learning rate

      learning rate는 파라미터가 이동할 **크기**를 조정한다. 만약 learning rate가 너무 작으면 수렴하기까지 더 많은 iteration(학습 횟수)가 필요하다. 따라서 학습이 비효율적이다. 또한 많은 iteration으로 학습을 했다고 해도 학습 데이터 셋에 과적합되어 robust한 모델을 만들 수 없다는 단점이 있다. 반대로 learning rate가 너무 크면 모델이 global minimum에 수렴하는 것이 아니라 발산할 수 있다. 즉, 학습할 수록 학습 loss가 점점 커지는 상황이 생긴다. [시각화 링크]

      따라서 적절한 learning rate를 찾는 것은 모델이 효율적으로 global minimum을 찾는데 핵심적인 역할을 한다.

      [아래 표 참고] lr가 너무 작을 때는 (1e-07<=lr<=1e-3) 학습 데이터의 수렴 속도가 느리고 또한 테스트 데이터의 성능이 다소 낮은 것으로 보아 과적합이 발생했다. 반대로 lr가 너무 클 땐 (lr=1.0) 모델이 학습을 하지 못하는 것을 볼 수 있다.



Line Plots of Train and Test Accuracy for a Suite of Learning Rates on the Blobs Classification Problem

2. weight decay

- SGD optimizer를 사용하는 경우

  weight decay는 loss 함수에 parameter(weight)의 l2 norm을 더해줌으로써 큰 값을 갖는 parameter에 추가적인 패널티를 부여하는 regularization 기법이다. parameter의 값이 크다는 말은 모델이 학습 데이터에 과적합되어 있다는 것을 의미한다. 따라서 너무 큰 값을 갖는 parameter는 패널티를 부과해 과적합을 방지할 수 있다.

$$C = C + \frac{\lambda}{2n}\sum_{w} w^2$$

  (C는 cost(loss) 함수, w는 parameter(weight), λ(lambda)는 weight decay coefficient). 여기서 regularization의 정도를 결정하는 가 하이퍼 파라미터에 해당한다.

- AdamW optimizer를 사용하는 경우

  Adam과 같이 adaptive gradient 방식을 따르는 optimizer는 loss 함수에 l2 regularization을 줄 경우 SGD optimizer보다 그 영향력이 적어진다. AdamW는 loss 함수 뿐만 아니라 업데이트 식에 직접 weight decay 식을 추가해 regularization 효과를 줬다. 역시 weight decay coefficient 가 하이퍼 파라미터에 해당한다. [참고 한글 블로그]

3. batch size

   학습 데이터 셋을 순회하면서 데이터 하나의 미분값을 계산해 파라미터를 업데이트하는(=Stochastic Gradient Descent, SGD) 방법은 robust한 모델을 만들지 못한다는 단점이 있다. 따라서 딥러닝을 학습할 때는 주로 학습 데이터 셋을 batch size 만큼 묶어 batch마다 파라미터를 업데이트하는 Mini-batch Gradient Descent 방법을 사용한다. Mini-batch Gradient Descent는 SGD보다는 robust한 모델을 만들 수 있으며 Batch Gradient Descent (epoch마다 파라미터를 업데이트 하는 방법)보다는 효율적으로 모델이 global minimum에 수렴한다는 장점이 있다.

4. epochs

   학습 데이터 셋을 순회하는 총 횟수를 epoch라고 한다.

하이퍼 파라미터 튜닝 방법

1. learning rate

   해당 논문에서 제안한 learning rate의 최솟값과 최댓값 범위 설정 방법은 다음과 같다.

   1. 몇 epoch를 돌면서 lr 값을 0에서 시작해서 max_lr(여기서는 0.02)에 도달할 때까지 선형적으로 증가한다.
   2. learning rate를 x축으로 accuracy를 y축으로 갖는 plot를 작성한다. [아래 표 참고]
   3. accuracy가 증가하기 시작하는 지점을 (lr=0.001) base_lr로 설정. accuracy가 다시 감소하는 구간 직전을 (lr=0.006) max_lr로 설정.

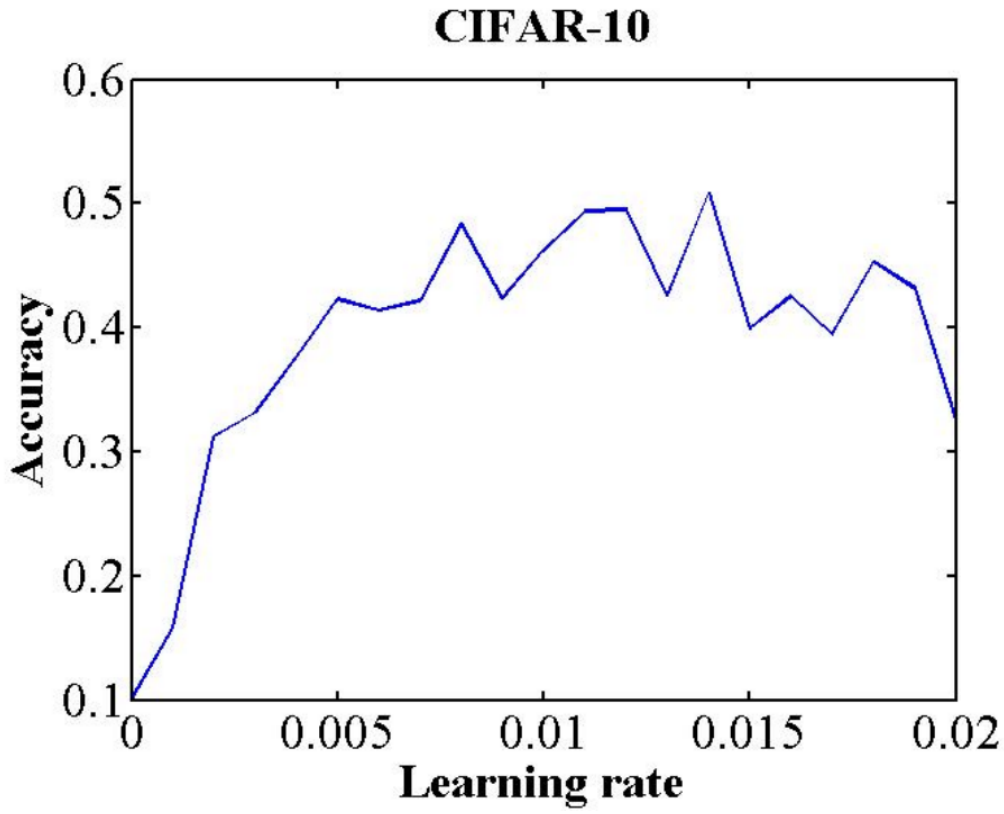


Figure 3. Classification accuracy as a function of increasing learning rate for 8 epochs (LR range test).

   매 epoch마다 초기 설정한 lr 값을 유지할 수도 있지만 각종 scheduler 함수를 사용하면 step마다 또는 epoch마다 lr 값을 변경할 수 있다.

   - torch.optim.lr_scheduler: epoch 값에 따라 learning rate 값을 조정하는 스케줄러 클래스 제공
     - Cyclical learning rate : lr가 최솟값과 최댓값 범위를 사이를 반복적으로 이동하며 값을 조정하는 방법. (논문 참고)
   - transformers package schedulers: 파이토치 패키지의 torch.optim.lr_scheduler.LambdaLR 클래스 파라미터 중 하나인 lr_lambda 함수를 1) step 단위로 2) 스케줄러 종류에 맞게 구현해 여러 스케줄러 함수를 제공
     - 가장 많이 사용되는 스케줄러 함수 : get_linear_schedule_with_warmup

2. batch size

   batch size가 클 수록 학습 속도가 빨라지지만 하드웨어 큰 메모리 사이즈가 요구된다. 따라서 하드웨어 메모리 사이즈가 허용할 수 있는 최대 크기의 batch size를 선택하되 batch size가 증가하면 learning rate 크기도 함께 증가시켜야 한다는 점을 주의하자. [시각화 자료 링크]

3. epoch

epoch를 크게 설정하면 모델이 학습 데이터에 과적합될 수 있다. 따라서 validation score가 감소하기 전까지만 학습 데이터를 반복해서 학습해야한다.

하이퍼 파라미터 튜닝 패키지 (Pytorch lightning package version)

1. Pytorch Lightning Package - LR finder

   [주의] lightning package를 사용한 모델만 사용 가능

   보통 논문에서 공개한 모델을 사용할 때는 논문에서 제공한 최적의 하이퍼 파라미터 값을 사용한다. 하지만 내가 새로운 모델을 구현했거나 새로운 데이터 셋으로 모델을 학습할 경우 최적의 learning rate 값을 찾아야 한다. 그 때 pytorch lightening에서 제공하는 LR finder를 사용해 learning rate 시작점을 찾을 수 있다.

2. Pytorch Lightning Package - early stopping

   특정 metric (주로 validation loss)이 n번 이상 이전 값보다 개선되지 않을 때 학습을 종료한다. 함수의 인자로 early stopping을 판단하는 기준 metric의 종류와 학습 종료까지 전까지 몇 번의 카운트를 할지 입력할 수 있다.

레퍼런스

1. How to Tune Hyper-Parameters in Deep Learning | by Neil Zhang | Medium
2. Hyper-parameter Tuning Techniques in Deep Learning | by Javaid Nabi | Towards Data Science

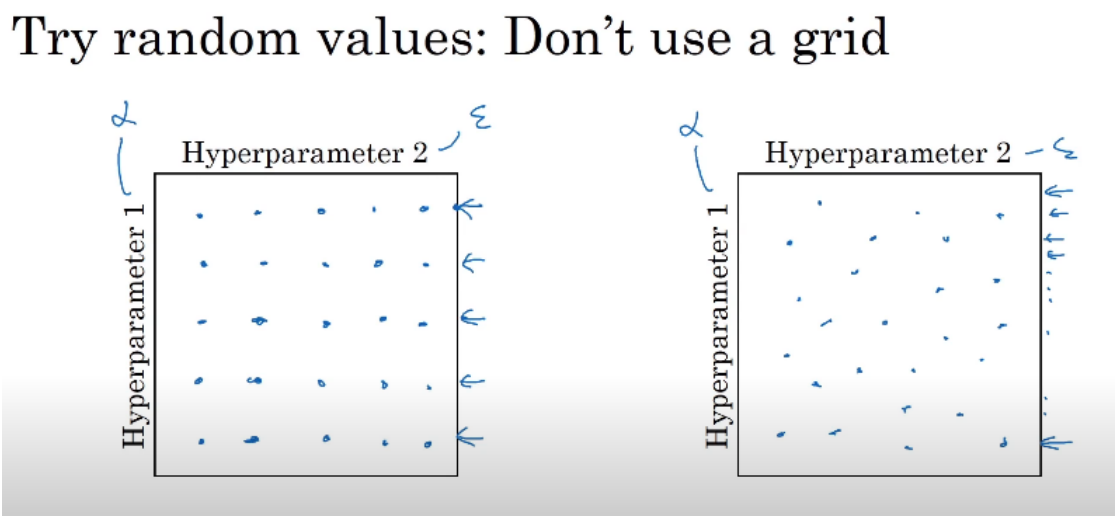
Tuning Process (C2W3L01)

Adam 옵티마이저의 주요 튜닝 대상 by 앤드류 응

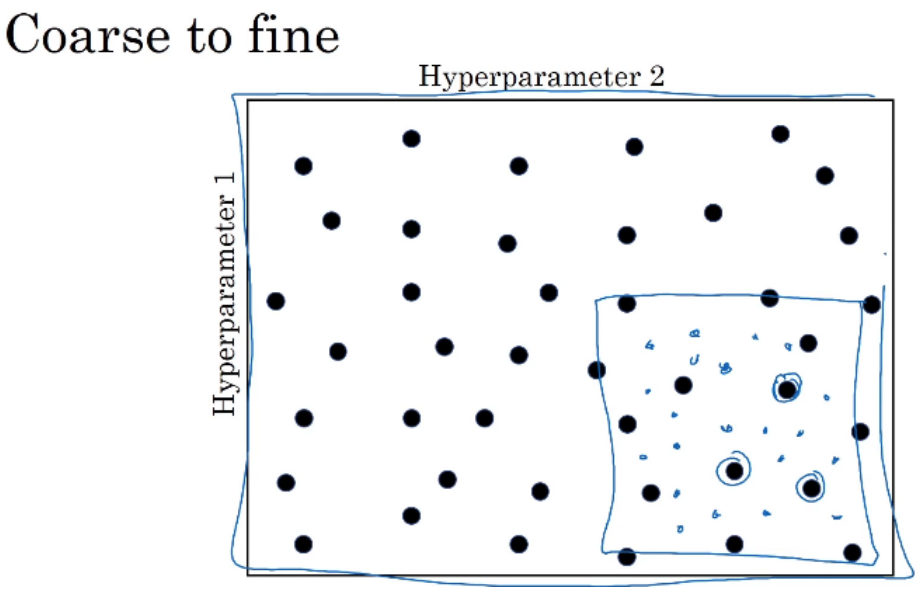
1. 학습률 : 가장 중요
2. 모멘텀 : ~ 0.9 (기본)
3. hidden units
4. mini-batch size
5. 레이어 개수
6. β1 (0.9), β2 (0.999), ε(10^-8)
7. 학습률 감쇠율

하이퍼 파라미터 탐색 방법

1. 그리드 서치
2. 랜덤 서치 : 무작위로 조합 선택 ⇒ 딥러닝은 어떤 하이퍼 파라미터가 문제해결에 도움이 되는 지 알 수 없기에 추천

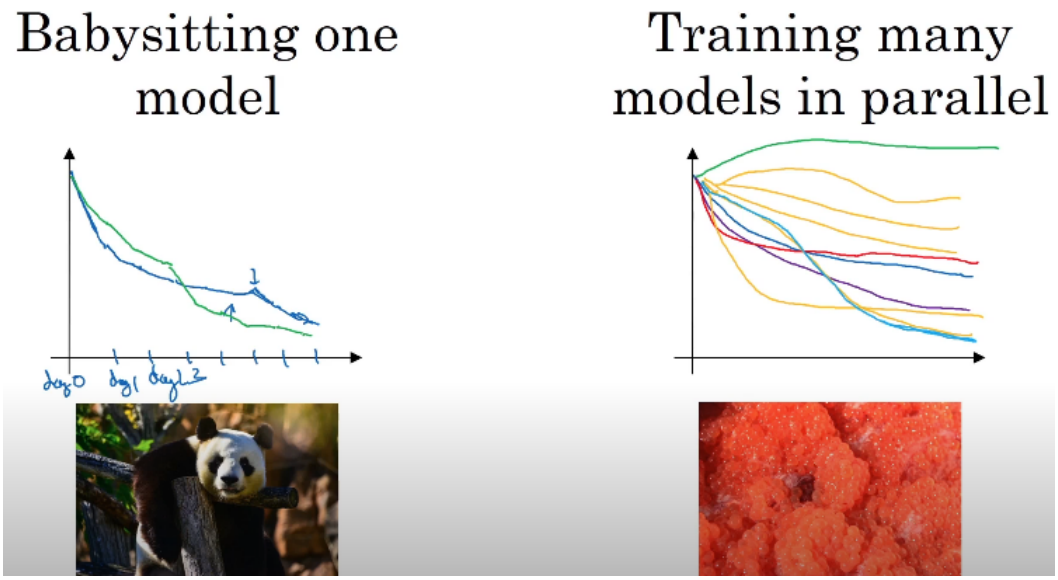


3. 랜덤 → 정밀화 접근 : 최고로 성능이 좋은 하이퍼 파라미터 영역 근처에서 재탐색



Hyperparameter Tuning in Practice (C2W3L03)

하이퍼 파라미터 튜닝 방법은 도메인 마다 다르다.



하이퍼 파라미터 튜닝 접근 방법

1. 모델 돌봄 학습(판다 전급) : 데이터가 많고 여러 모델을 동시에 학습시킬 컴퓨터 자원이 충분하지 않을 때 사용
   - 예 : 온라인 광고, 컴퓨터 비전 : 하나의 모델에 집중해서 매개변수를 조금씩 조절
2. 모델 병렬 학습(캐비어 전급) : 컴퓨팅 자원이 충분하다면 여러 모델을 다양한 하이퍼 파라미터로 학습해서 가장 좋은 모델 선정