[다양한 Learning Rate Scheduler 중 적어도 한 가지를 소개]

LamdaLR이란, Lambda 표현식으로 작성한 함수를 통해 Learning rate를 조절하는 기법이다. 즉, 초기 learning rate에 lambda 함수에서 나온 값을 곱해줘서 learning rate를 계산하는 방식인데, 이때, learning rate는 backpropagation 학습 과정에서 모델의 weight인 gradient의 변화/업데이트 보폭을 말한다. 구현에 대해 이야기하면, LambdaLR의 경우에는 Pytorch에서 제공하는 클래스를 활용해서 사용할 수 있고, 우선 optimizer와 scheduler를 정의하고, 학습할 때 batch마다 optimizer.step()을 진행하고, 각 epoch마다 scheduler.step()을 진행해 학습률을 업데이트 한다.

이외에도 다른 Learning Rate Scheduler로는 MultiplicativeLR이 있다. 이 방식은 Lambda 표현식으로 작성한 함수를 통해 learning rate를 조절하는 방식으로, 초기 learning rate에 lambda 함수에서 나온 값을 누적곱해서 learning rate를 계산한다.

StepLR은 step size마다 gamma 비율로 learning rate를 감소시키고, ReduceLROnPlateau은 성능이 향상이 없을 때 learning rate를 감소시킨다. 그렇기 때문에 validation loss나 metric(평가지표)을 learning rate step함수의 input으로 넣어주어야 한다. 그래서 metric이 향상되지 않을 때, patience횟수(epoch)만큼 참고 그 이후에는 learning rate를 줄인다. optimizer에 momentum을 설정해야 사용할 수 있다.

[Training Error와 Generalization Error 사이 간극을 줄이는 방안]

우선 Training Error와 Generalization Error 간의 차이를 줄이는 방법으로는 Dataset Augmentation이 있다. 이 방법은 학습데이터의 크기를 늘리는 것으로 단순히 새로운 데이터 추가하는 것을 넘어서 이미지 반전/밝기 조절/노이즈 등을 통한 데이터 부풀리기로 볼 수 있다.

또한, Early Stopping이 있다. Epoch 수(학습 반복 횟수)가 늘어날수록 학습셋에 대한 오차는 줄어들지만, 검증셋의 오차가 증가하며 오버피팅이 발생할 수 있기 떄문에, 이전 에폭과 비교해서 오차가 증가하면 오버피팅이 발생하기 전에 학습을 멈추는 것을 'Early Stopping'이라고 한다.

Dropout은 학습할 때 뉴런의 일부를 네트워크 상에 존재하지 않는 것처럼 랜덤으로 '0'으로 만드는 방식으로 드롭아웃의 확률은 일반적으로 p=0.5를 사용하며, 학습할 때만 드롭아웃을 하며 테스트할 때는 모든 뉴런을 사용한다. 보통, 하나의 모델로 학습하면 오버피팅이 발생할 수 있지만, 드롭아웃을 통해 앙상블 학습(ensemble learning)처럼 마치 여러 모델을 학습시킨 것과 같은 효과를 주어 오버피팅 문제를 해결할 수 있다.