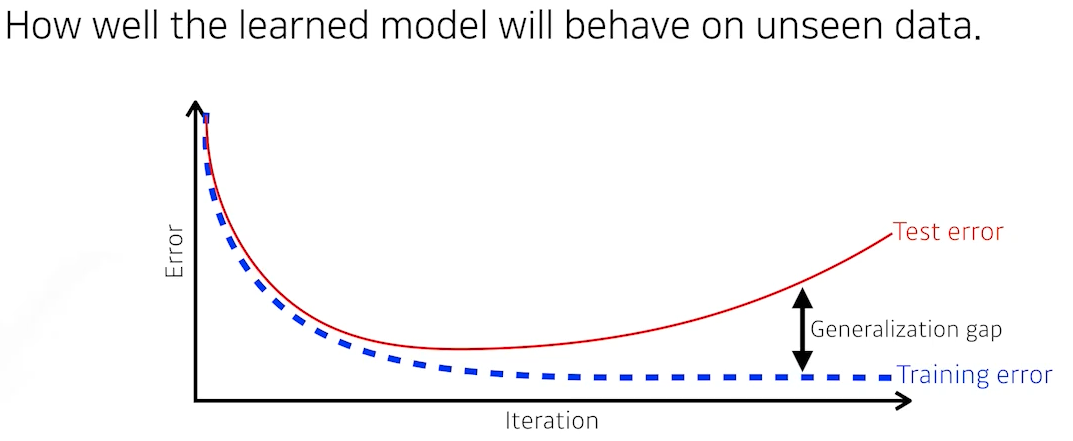
Generalization이 좋다. -> 학습데이터와 테스트용과 갭이 적다.



트레인에 너무 맞춰서 테스트에 잘 동작 안되는게 overfitting

트레인을 너무 적게 시키거나 네트워크가 너무 간단해서 트레인조차도 못맞춰 underfitting

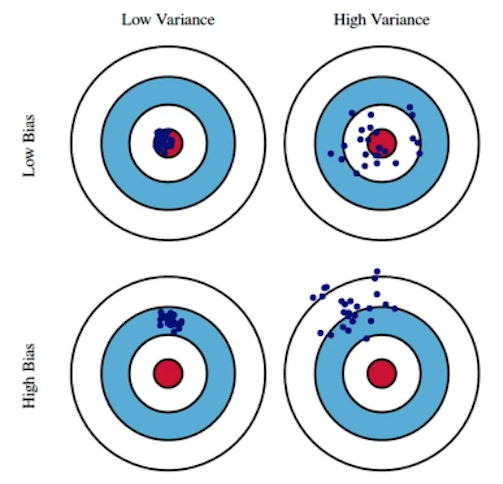
Cross-validation

파라미터 = 내가 학습을 통해 최적의 값을 찾고싶은것 w,b

하이퍼파라미터 = 내가 정하는값들 lr, network를 엄ㄹ마나 크게 가져갈지

크로스밸리데이션을 해서 최적의 하이퍼 파라미터값을 찾고 이를 고정하고 학습할때는 모든 데이터를 다 사용한다. 테스트데이터는 어디에도 쓰이면 안된다.

트레인에는 트레인데이터와 밸리데이션데이터만을 사용해야함



Bias and Variance Tradeoff

텍스트이(가) 표시된 사진

자동 생성된 설명

총량 보존이기때문에 바이어스와 베리언스 다 줄이기는 힘들다.

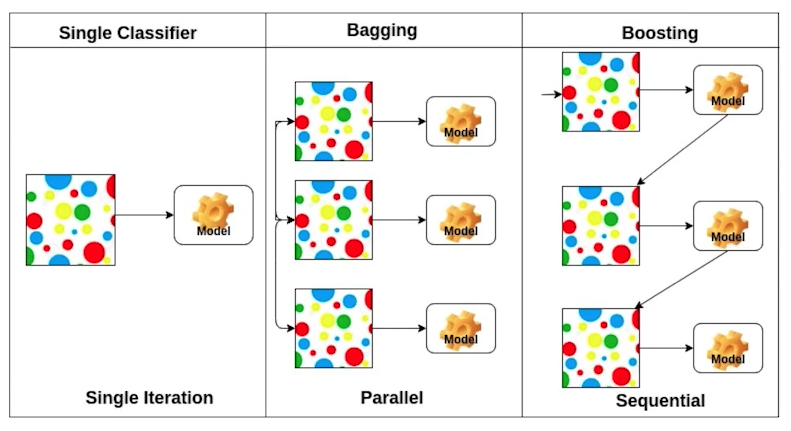
Bootstrapping = Bagging ex)앙상블효과

학습데이터 100개가 있으면 이를 다 쓰려기보단 일부를 활용하여 만든 모델들이 예측하는 값들이 얼마나 일치하는지 보고 전체적인 모델의 uncertainty를 확인할 때 쓴다.

Boosting

100개중 모델을 간단하게 만들고 100개를 예측시켜 -> 80성공 20실패

두번째 모델은 20개에서만 잘 동작하는 모델을 만들자



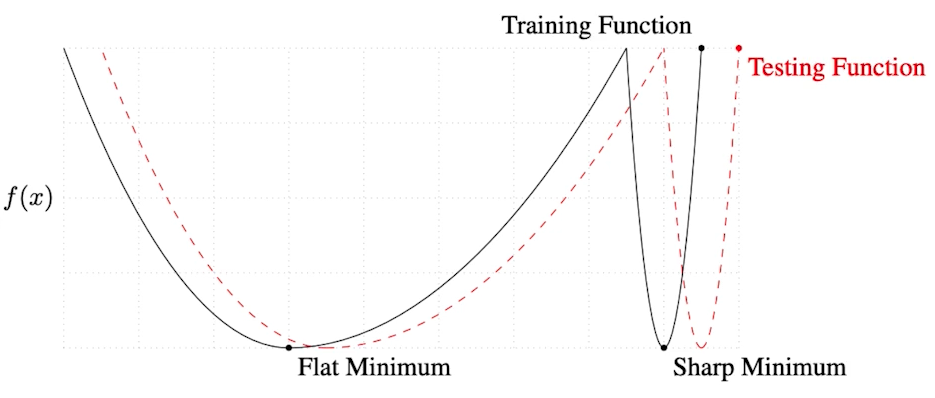
GD 종류

Stochastic/ Mini-batch/ Batch gradient

미니배치가 메인

배치사이즈를 작게 쓰는게 일반적으로 좋다. Generalization이 좋다.

Flat minimum은 트레인과 테스트에서 모두 성능이 좋을것

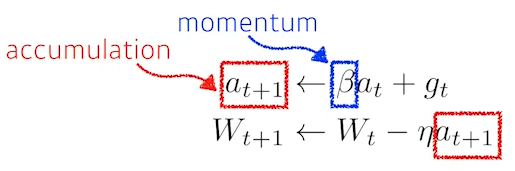


GD lr 잡기가 너무 힘들다는 문제점이 있다.(딱 맞아야 잘 됨)

텍스트이(가) 표시된 사진

자동 생성된 설명

Momentum(관성) 한번 이쪽으로 가면 다음번에 방향이 조금 달라져도 원래 가던거를 좀 이어가자



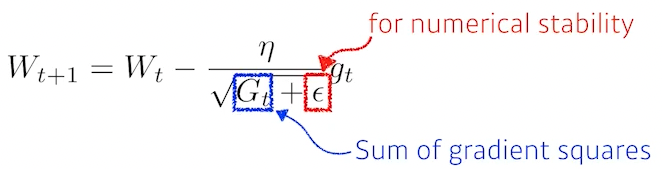
Nesterov Accelerate(NAG)

Momentum과 유사하나 minimum에 조금 더 빨리 수렴한다.

Adagrad

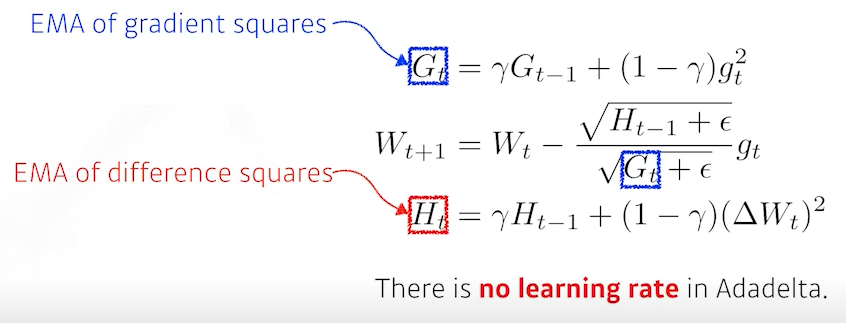
학습을 하면서 적게 변화한 파라미터를 많이 변화시키는 걸 목표로

G가 점점 커지면서 lr이 점점 작아져서 학습이 많이 진행되면 변화가 거의 없다는 단점

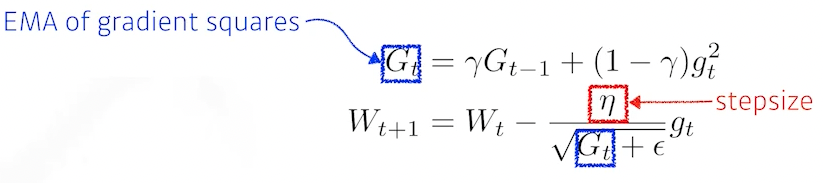


Adadelta

Adagrad에서 윈도우 사이즈를 줄여 메모리부족을 해결



RMSprop



Adam

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

ES = 일반적으로 epoch이 쌓이면 overfitting(validation error 증가)되기때문에 ES진행

Parameter Norm Penalty

부드러운 함수이면 generalization이 높을것이다.

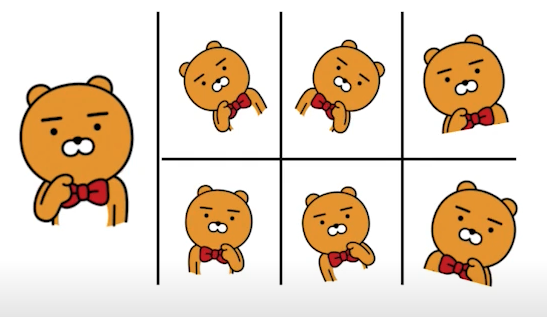
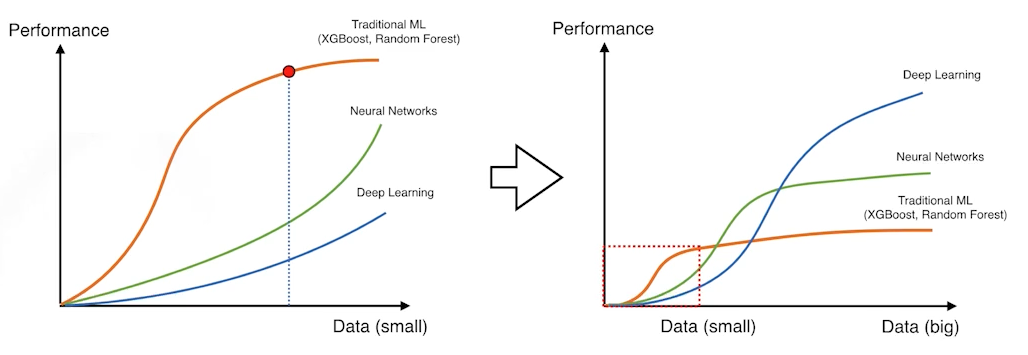
텍스트이(가) 표시된 사진

자동 생성된 설명

Data Augmentation

DL과 NN은 데이터가 많으면 많을수록 좋다! 다만 데이터를 모으기 힘드니

여러가지 변형을 줘서 양을 늘리자!

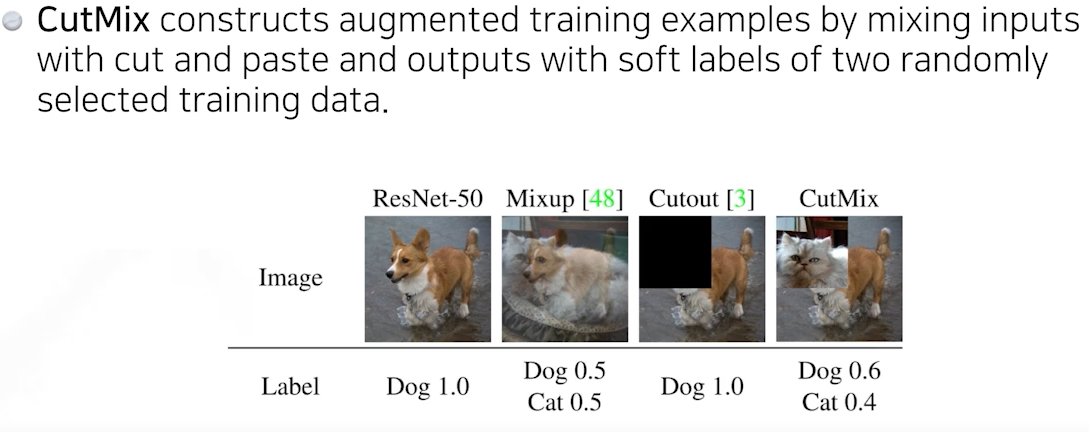


텍스트이(가) 표시된 사진

자동 생성된 설명

Label Smoothing

입력데이터 두개를 섞어서 학습



믹스업이나 컷믹스 활용해보자 성능 많이 올라간다

Dropout 연결을 임의적으로 끊어줘

Batch Normalization(이해 더 필요함!!)