10장 인공신경망 소개

은닉층의 수: 사실 은닉층이 하나인 다층 퍼셉트론이더라도 뉴런의 수가 충분하면 아주 복잡한 함수도 모델링 할 수 있습니다. 하지만 심층 신경망은 얕은 신경망보다 파라미터 효율성이 훨씬 좋습니다. 데이터는 계층적 구조를 갖고 있는 경우가 많습니다. 이런 계층 구조는 심층 신경망이 좋은 솔루션으로 빨리 수렴하게끔 도와줄 뿐만 아니라 새로운 데이터에 일반화되는 능력도 향상시켜줍니다.

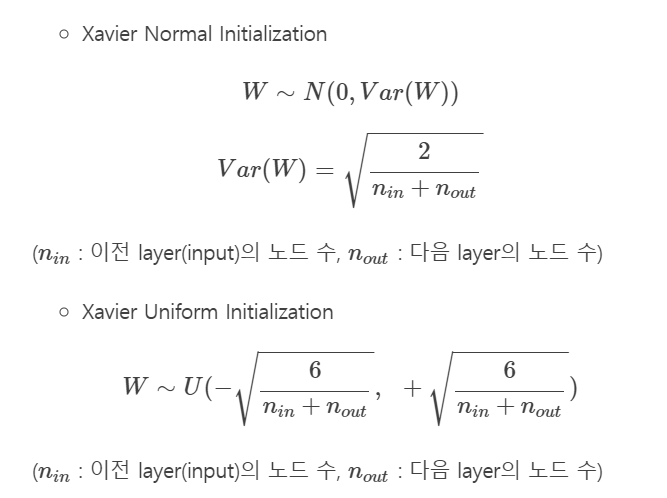
11장 심층 신경망 훈련

심층 신경만 훈련의 문제점:

1. Vanishing gradient(or exploding gradient)
2. 훈련이 극단적으로 느려진다.
3. 파라미터가 많아 과대적합이 발생할 수 있다.

Vanishing gradient(or exploding gradient)

세이비어 초기화: 적절한 신호가 흐르기 위해서는 각 층의 출력에 대한 분산이 입력에 대한 분산과 같아야 한다. 그리고 역방향에서 층을 통과하기 전과 후의 그래디언트 분산이 동일해야한다. 이 두가지는 층의 입력과 출력 연결 개수가 같지 않다면 보장할 수 없다. 이들은 실전에서 매우 잘 작동한다고 입증된 대안을 제시했다.



다음의 분표에서 무작위로 난수를 추출하여 초기화하는 것을 세이비어 초기화 혹은 글로럿 초기화라고 부른다.

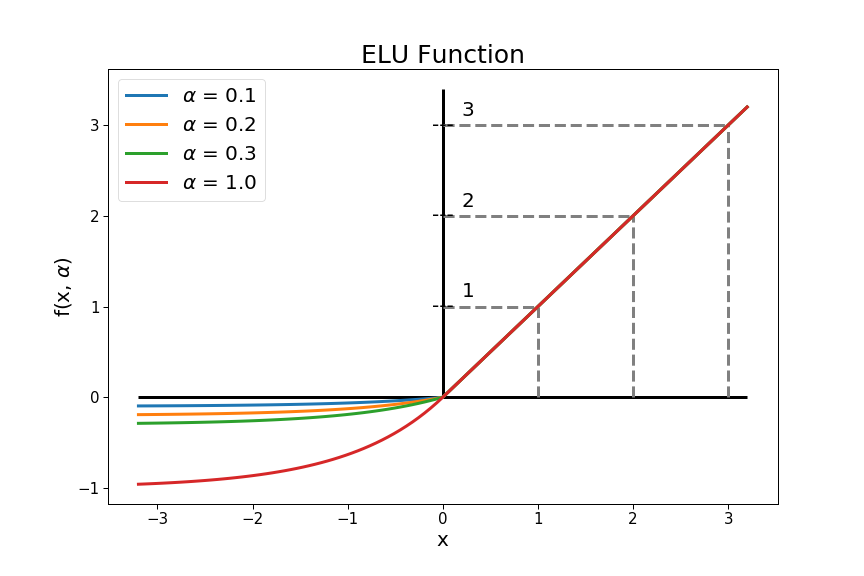
He 초기화: ReLU 활성화 함수를 위한 초기화 전략

수렴하지 않는 활성화 함수

ReLU: 시그모이드에 비해 계산도 빠르고 특정 양수값에 수렴하지 않는다는 장점이 있습니다. 하지만 죽은 ReLU(dying ReLU)로 알려진 문제가 있습니다. 이는 훈련하는 동안 일부 뉴런이 0 이외의 값을 출력하지 않는다는 의미이다. ReLU 함수는 입력이 음수이면 그래디언트가 0이 되기 때문에 이러한 뉴런이 다시 살아나기는 어렵다. 이 문제를 해결한 것이 LeakyReLU.

LeakyReLU: 하이퍼파라미터 알파가 leaky한 정도를 정한다. 주로 0.01로 한다. 이 작은 기울기가LeakyReLU를 죽지 않게 만들어준다. LeakyReLU는 일반적인 ReLU보다 거의 항상 성능이 좋다. ReLU의 변종으로는 RReLU(훈련하는동안 알파를 무작위로 선택하고 테스트 시에는 평균을 사용), PReLU(알파가 훈련하는동안 학습) 등이 있다.

ELU(Exponential linear unit): ReLU의 변종.

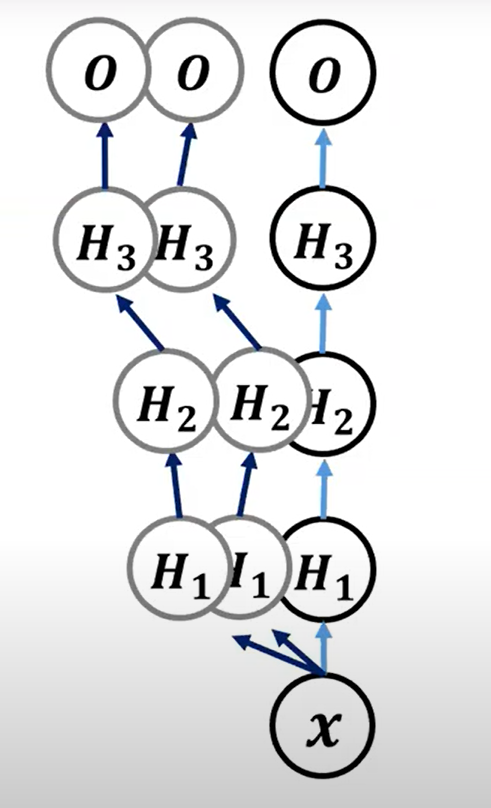
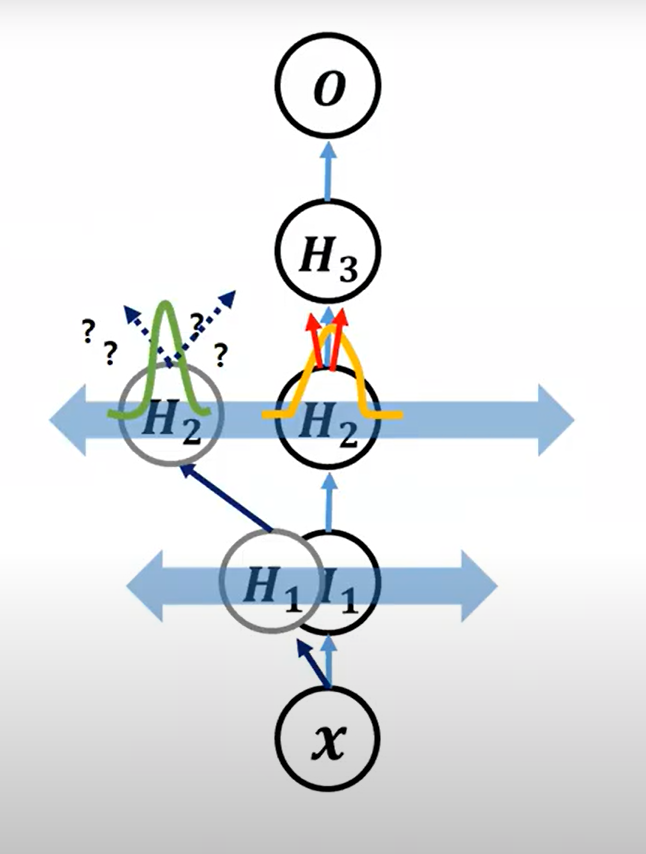


* Z<0 일때, 음수값이 들어오므로 활성화 함수의 평균 출력이 0에 가까워집니다. 이는 그래디언트 소실문제를 완화합니다.
* Z<0이어도 그래디언트가 0이 아니므로 dying ReLU를 만들지 않습니다.
* 알파=1 일때, 이 함수는 모든 구간에서 부드러워 경사 하강법의 속도를 높여줍니다.

배치 정규화(Batch Normalization)

ELU와 He 초기화를 사용하면 훈련 초기 단계에서 그래디언트 소실이나 폭주문제를 크게 완화시킬 수 있지만 훈련하는 동안 다시 발생하지 않을 거라는 보장이 없습니다. 일반적으로 훈련하는 동안 이전 층의 파라미터가 변함에 따라 각 층에 들어오는 입력의 분포가 변환됩니다. 이를 내부 공변량 변화(internal covariate shift) 문제라고 부릅니다.

내부 공변량 변화(internal covariate shift)

위의 그림과 같이 가중치가 변함에 따라 hidden layer 안의 노드변화가 심한 경우 학습이 어렵다. 이처럼 hidden layer의 분포가 크게 달라지는 것을 내부 공변량 변화라고 부른다. 이러한 내부 공변량 변화를 줄여주기 위해 BN을 각 layer마다 추가하게 된다.

그래디언트 클리핑

역전파될 때 일정 임계값을 넘어서지 못하게 그래디언트를 그냥 단순히 잘라내면 그래디언트 폭주문제를 해결할 수 있다. 이를 그래디언트 클리핑이라고 한다.

미리 훈련된 층 재사용하기

아주 큰 규모의 DNN을 훈련시킬 경우 비슷한 유형의 문제를 처리한 신경망이 이미 있는지 찾아보고 그런 다음 그 신경망의 하위층을 재사용하는 것이 좋습니다. 이를 전이학습(transfer learning)이라고 합니다.

훈련의 속도를 높이는 방법

1. 연결 가중치에 좋은 초기화 전략 적용하기
2. 좋은 활성화 함수 사용하기
3. 배치 정규화 사용하기
4. 미리 훈련된 신경망의 일부 재사용하기
5. Fast optimizer 사용

Fast optimizer의 종류: momentum optimization, Nesterov Accceleranted gradient, AdaGrad, RMSProp, Adam

Adam(adaptive moment esrimation) = AdaGrad + RMSProp

AdaGrad:

RMSProp:

* Skip
* <https://dalpo0814.tistory.com/29>
* <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

학습률 스케줄링

높은 학습률로 시작하고 학습 속도가 느려질 때 학습률을 낮춘다면 좋은 솔루션에 빨리 도달할 수 있습니다. 학습 스케줄에는 다음의 것들이 보편적입니다.

* 미리 정의된 개별적인 고정 학습률
* 성능 기반 스케줄링(스텝마다 검증 오차 측정하여 오차가 줄어들지 않으면 학습률 감소)
* 지수 기반 스케줄링(반복 횟수 t의 함수로 학습률을 설정)
* 거듭제곱 기반 스케줄링(지수 기반과 유사)

과대적합을 피하기 위한 규제 방법

1. 조기 종료
2. L1, L2 규제
3. 드롭 아웃
4. 맥스-노름 규제
5. 데이터 증식

맥스-노름 규제: 가중치 벡터의 크기를 미리 정해 높은 상한 값을 넘지 못하도록 제한하는 방법.

데이터 증식: 밝기, 명암, 색조 등을 변경하고 이동, 회전, 크기 조절, 뒤집기, 자르기 등의 기법을 이용한다.

12장 텐서플로우를 사용한 사용자 정의 모델과 훈련

Pass

13장 텐서플로우에서 데이터 적재와 전처리하기

메모리에 올릴 수 없는 큰 규모의 데이터 셋으로 딥러닝을 해야하는 경우 데이터 API를 활용할 수 있다. 또한 데이터를 불러온 후 전처리도 진행해야하는데 사용자 전처리 층을 만들거나 Keras에서 제공하는 표준 전처리 층을 사용할 수 있다.

TFRecond 포맷

대용량 데이터를 저장하고 효율적으로 읽기 위해 텐서플로우가 선호하는 포맷이다. 이는 크기가 다른 연속된 이진 레코드를 저장하는 단순한 이진 포맷이다. 100MB가 넘는 데이터의 경우 이 포맷을 통해 학습속도의 큰 향상을 기대할 수 있다.