1. Training 과정 코드분석

-train\_step() 분석

Forward -> calculate loss -> backward -> update step -> assert 점검

매개변수를 함수 내 input, output 변수에 할당

**Forward**

deque모듈을 이용해서 input변수 양방향 큐 데이터로 저장(z\_stack) -> 반복문을 이용해 레이어 순서대로 mini batch input을 forward한 값을 z\_stack에 추가하고 activation에 저장해서 다음 레이어에 할당, 반복

**calcultate loss**

할당받은 input과 output 데이터간에 차이로 loss를 구해 저장

**Backward**

Z\_stack에 저장된 각 레이어들의 forward 값들을 하나씩 빼서 lz변수에 저장(=각 레이어의 Local gradient), Upstream gradient 선언, backward를 하며 경사를 저장할 빈 데크 선언 -> 반복문을 이용해 레이어 역순대로 Local graident

**-train\_step() 분석**

매개변수로 받은 배치 입력, 출력데이터를 mini\_batch\_inputs, mini\_batch\_outputs 지역변수에 저장

**#forward**

입력 받은 데이터를 deque모델을 이용해 스택 자료구조형으로 z\_stack 변수에 저장

입력 받은 데이터를 activation 초기값으로 지정

반복문을 사용해서 Layer의 각각의 레이어에 대해 연산을 한다. 반복문 내 변수 z에는 레이어의 forward연산 한 결과를 저장하고, 변수 activation에는 forward연산 결과인 z를 레이어의 activation연산한 결과를 저장한다. 여기서 activation은 레이어들의 forward연산을 진행하기 위해 사용하고 z는 backward연산을 진행하기 위해 사용된다.

**#calculate loss**

forward연산 이후 네트워크를 통과한 결과값인 activation과 레이블 outputs와의 loss값을 객체가 가지고 있는 지정된 loss연산을 통해 변수 loss\_err에 저장한다.

**#backward pass**

backward연산을 위해서 z\_stack에 마지막에 저장해 놓은 데이터를 변수 lz의 초기값으로 지정

변수 upstream\_gradient에 초기값으로 전체 loss 값을 지정

빈 스택을 grads로 지정

반복문을 사용해서 Layer의 역순으로 경사를 구한다. 변수 layer\_err에 해당 layer의 결과값 lz와 upstream\_gradient값으로

초기 upstream\_gradient loss와 activation 미분값

변수 layer\_err에 upstream\_gradient와 lz에 저장해놓은 레이어의 연산값의 미분값을 곱해 저장한다. 이것은 레이어의 activation함수에 대한 경사(backward한 결과)이다.