**Machine Learning & Data Mining, Spring 2020**

**Homework 9**

Due June 12

**과제목표**

* **딥 뉴럴 네트워크를 구현할 수 있다.**
* **Backpropagation을 이해하여 구현할 수 있다.**
* **CNN을 이해하여 학습시킬 수 있다.**

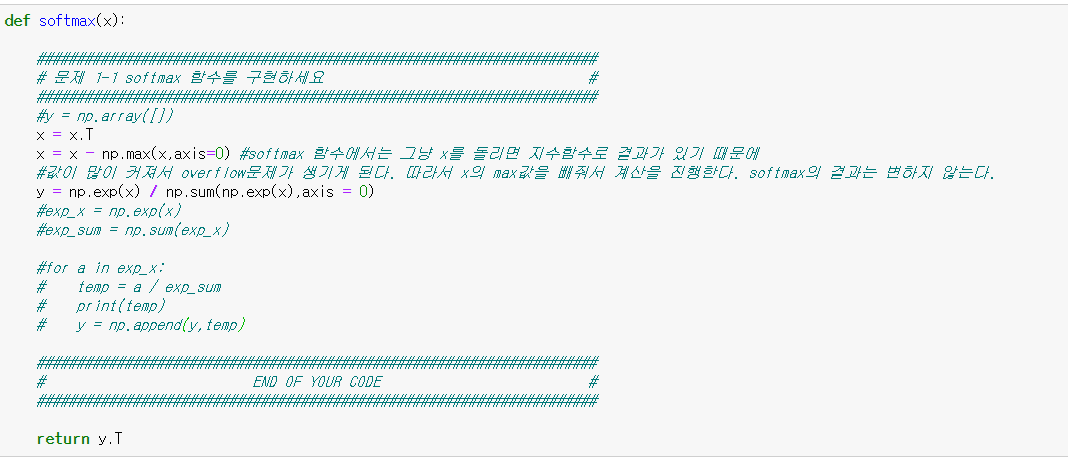
**제출**

* **요구한대로 작성한 보고서와 주피터 노트북**
* **주어진 주피터 노트북을 바탕으로 요구하는 코드를 구현 후 보고서에 해당 부분을 캡처**
  + **요구하는 코드만을 캡처하여 보고서에 제출**
* **요구한 코드 이외에 다른 코드는 수정하지 말것.**
* **최종 점수는 비율적으로 수정해서 들어갈 예정**

**201520949 허윤행**

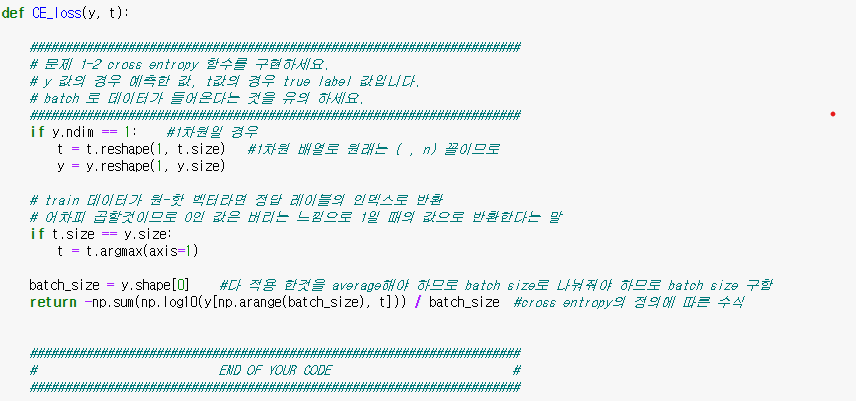
**소프트웨어학과**

**1-1 softmax 함수 구현**

****

Softmax는 그냥 함수의 정의에 따라 구현하게 되면 exponential함수이기 때문에 큰 수는 엄청 큰 수가 되기 때문에 overflow문제가 생기게 된다. 따라서 np.max를 사용해서 표준화를 진행해준다. 0에 가깝게. 그래도 생기는 확률이 계산되므로 상관이 없다. 나머지는 함수의 정의에 따라 진행해주었다. X로 들어오는 input값은 transpose를 하는 이유는 x값은 가로로 긴 행렬이므로 계산하기 편하게 세로로 transpose하고 구현하였다.

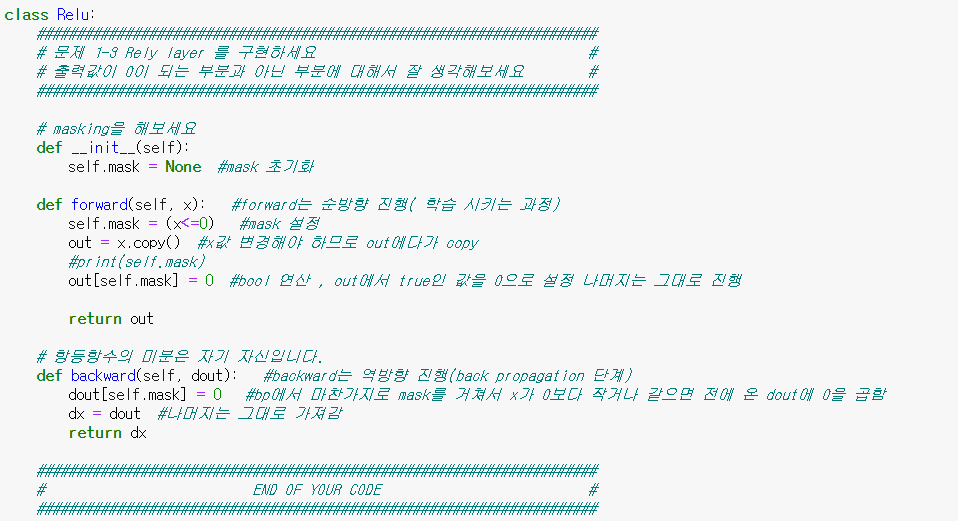
**1-2 cross entropy 함수 구현**



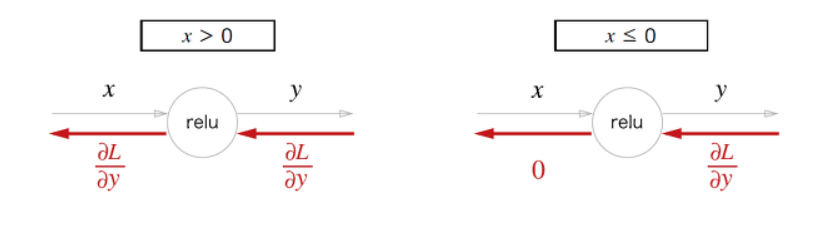
Cross entropy 구현에 있어서 첫 줄은 1차원의 경우는 shape을 찍어보면 배열 연산에 있어서 reshape을 해줄 필요가 있다. 주석에서 설명한 것처럼 train data가 one-hot encoding 된 데이터이므로 정답 레이블의 인덱스로 반환을 하게 된다. 따라서 0아니면 1이기 때문에 0인 값은 곱해도 0이므로 1일 때의 값으로 반환하는 것, 따라서 argmax으로 진행하였다.

지금까지 진행한 것은 데이터 하나에 해당하는 연산이므로 batch size를 통해 average를 구해준다. 그 다음은 cross entropy의 정의에 따라 return을 해준다.

**1-3 Relu layer 구현**

****

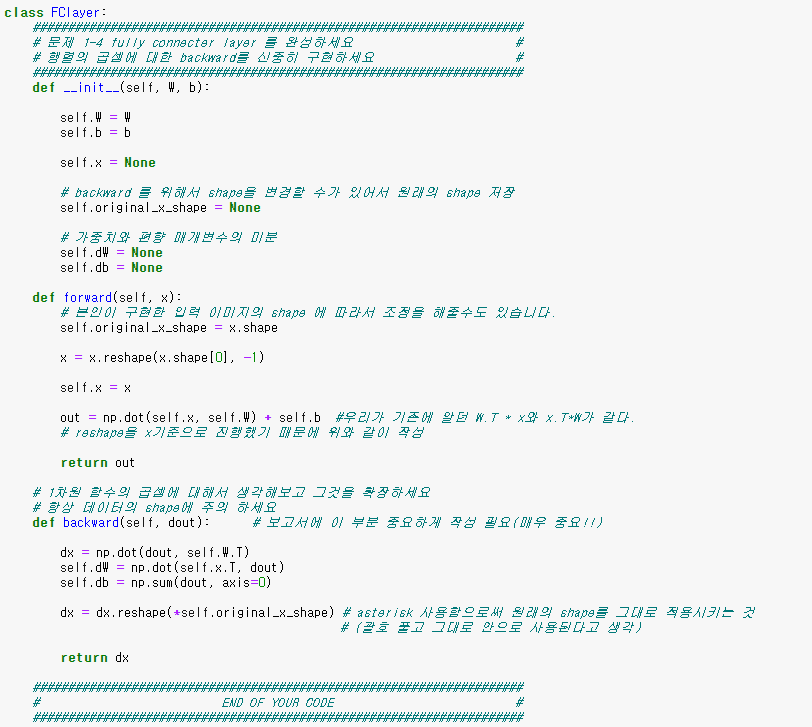
Relu부터는 forward 와 backward 부분이 있는데 forward는 학습하는 방향, backward는 역방향, 즉 backpropagation단계다. 우선 forward에서는 mask를 설정을 해주도록 한다. Mask는 실제 relu함수에서는 0보다 작거나 같은 구간에서는 0으로 masking을 하고 그 이외구간에서는 자기 자신을 출력하도록 하므로 mask를 설정해준다.

****

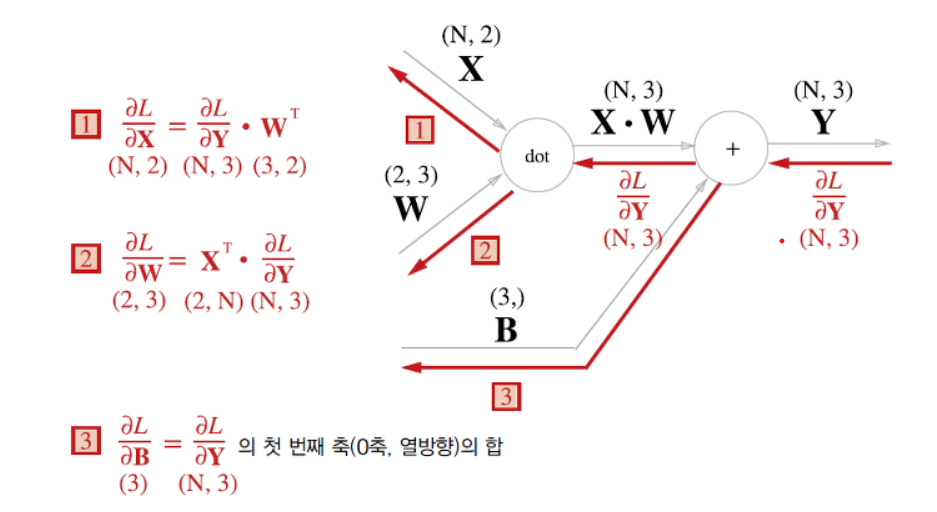
**<그림> relu 참고자료**

forward에서는 masking을 통해서 구현해주고, backward에서는 마찬가지로 bp에서 mask를 거쳐서 x가 0보다 작거나 같으면 전에 온 dout에 0을 곱해 준다. 나머지는 그대로 가져가도록 한다.

**1-4 FC layer 구현**

****

Weight 와 bias가 같이 과정에 진행되는 FC layer에서는 각각 matrix를 초기화 해주고 진행한다. Forward 과정에서는 단순히 우리가 알던 W.T와 X를 곱해주고 bias를 더해주는 식으로 계산을 진행한다. Reshape은 -1로 하는 과정은 원래 shape은 (n, )로 찍히므로 이를 하나의 배열로 만들어 주기 위함이다.

****

**<그림> 참고자료**

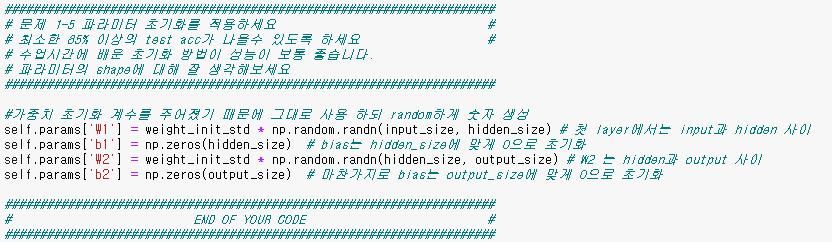
가장 중요한 backward 과정이다. 이는 추가로 자료들을 찾아보고 참고하였다. 우선 backward에서 backward를 부르기 전 단계에서 넘어오는 값들이 중요하다. Chain rule에 따라서 각각이 어떻게 되는가에 대한 참고자료이다. 우선 검은색 글자로 된 것이 forward 빨간색 글자가 backward이다.

backward에서 1번을 보면 결과가 나오는 이유는 chain rule에 따라 X에 대해서 L을 편미분한 값은 Y에 대해서 L을 편미분한 값에 X에 대해서 Y를 편미분한 값이다. X에 대해서 Y를 편미분한 값만 모르는 상황인데 이는 X와 W를 dot product한 값과 B를 더한 값이다. 이를 식으로 표현하면 X.T \* W + B인데 이를 X에 대해서 편미분을 하게 되면 B는 상수취급 되면서 사라지게 되고 X.T \*W는 W.T\*X와 같은 값이므로 이를 다시 미분하면 W.T가 된다. 따라서 dout은 Y에 대해서 L을 편미분한 값이므로 dout과 W.T를 dot product를 해준다.

dW값도 마찬가지다. 이는 위에서 설명한 그대로이다. 차이점은 chain rule에서 순서가 다르다는 점이다. W에 대해서 L을 편미분한 것을 구하려고 하니까 우리가 알고있는 것은 Y에 대해서 L를 미분한 것을 알고 있으므로 chain rule에 따라 W에 대해서 Y를 미분한 것을 구해야 하므로 이는 X.T이다. 따라서 dot product는 순서가 중요하므로 X.T 와 dout을 해주도록 한다.

dx값들이 바뀌었으므로 모양도 다시 잡아줘야 하므로 처음에 저장했던 original shape를 통해 적용을 해준다. 여기서 사용한 문법은 asterisk로 쉽게 말해 괄호를 푼다고 생각하면 된다. \*를 사용한 연산은 사용 방법이 다양한데 여기서는 괄호를 푸는 방법으로 사용되었다.

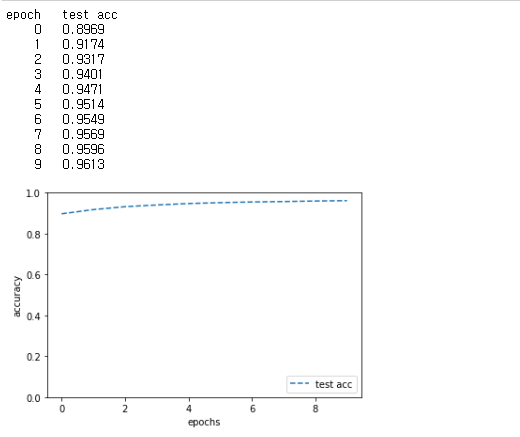
**1-5 DNN 파라미터 초기화 하기**

****

파라미터로 주어진 weight\_init\_std를 통해서 W와 bias값들을 설정해주는 과정인데 이는 random하게 숫자를 생성해준다. 대신 중요한 점은 size를 잘 맞춰야 한다는 점인데 계층이 두 개이므로 가운데에는 hidden layer가 있기 때문에 각각의 size를 맞춰주도록 한다.

Bias는 0으로 ,W는 랜덤하게 생성하고 표준화 계수를 곱해서 숫자를 생성해준다.

**1-6 최종 학습결과 제출**



수업시간에 배운 대로 epochs(계층을 얼마나 지났는지)가 늘어날수록 학습 정확도가 올라가는 것을 확인할 수 있었다.