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

**Homework 9**

Due June 12

학번: 201520908

학과: 소프트웨어

이름: 유성민

학년: 3

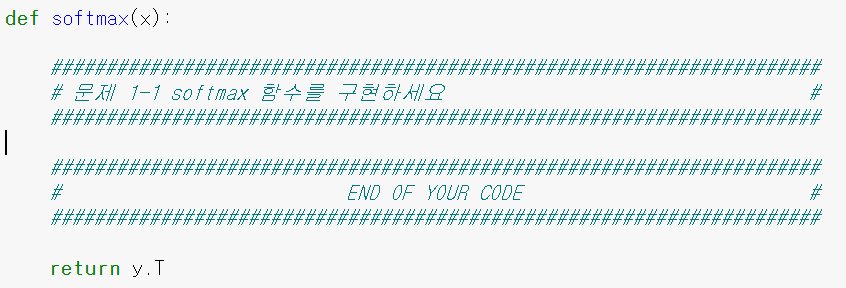
**과제목표**

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

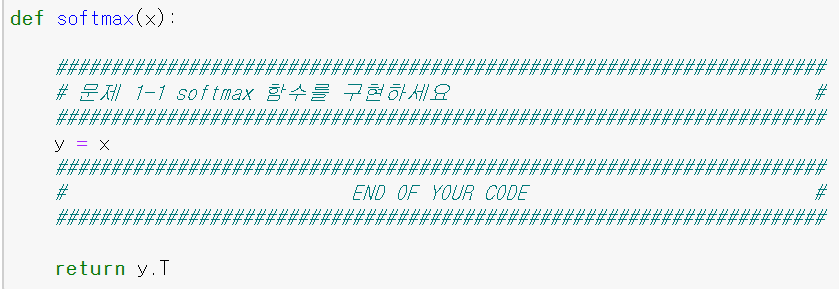
**제출**

* **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**

**보고서 코드 작성 예시**

****

문제 1-1 번의 경우 다음과 같이 문제가 주어집니다. 제가 작성한 주피터 노트북의 최종 학습을 할 수 있도록 코드를 작성한 합니다. 예를 들면 다음과 같이 코드를 작성한 후에,



작성한 부분을 캡처 하시면 됩니다. 특히,

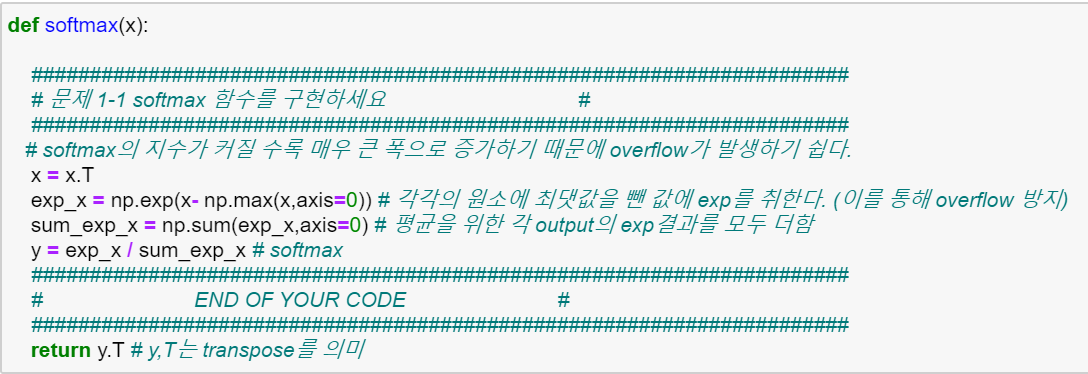
* 워드 글자와 캡처한 그림의 글자 크기가 유사하도록 캡처 하고
* 실제 워드로 코드내용을 적을 경우 **가독성이 떨어지므로** 되도록 캡처합니다.

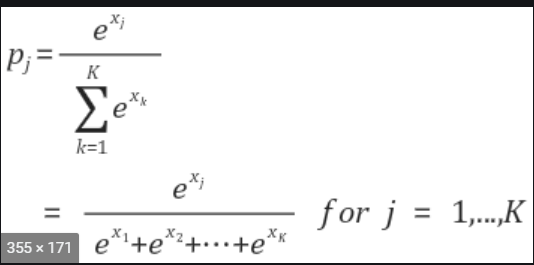
어떠한 방식으로 작성해야 되는지에 대한 힌트는 주피터 노트북의 주석에 달아 놓겠습니다.

**주피터 노트북의 설명에 따라서 해당 문제의 코드들을 캡처하세요.**

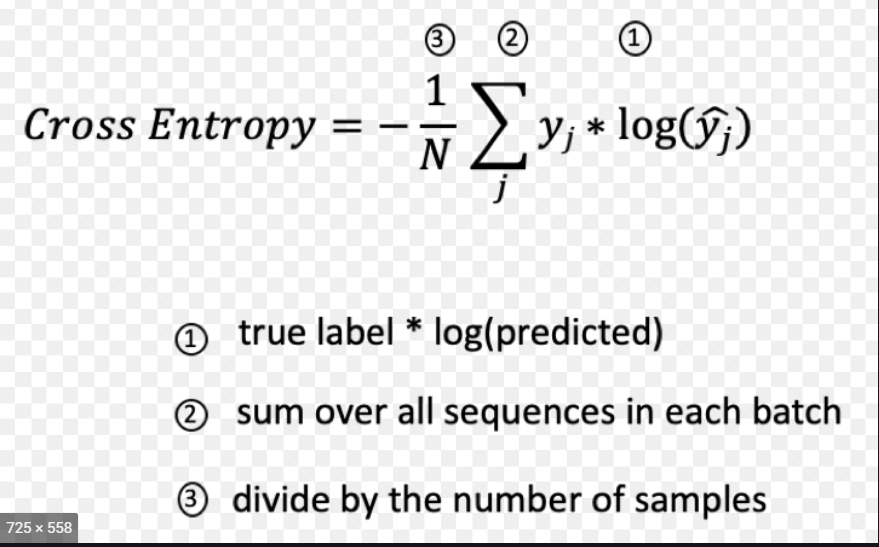
**[1] DNN 과 네트워크에 사용되는 함수 구현**

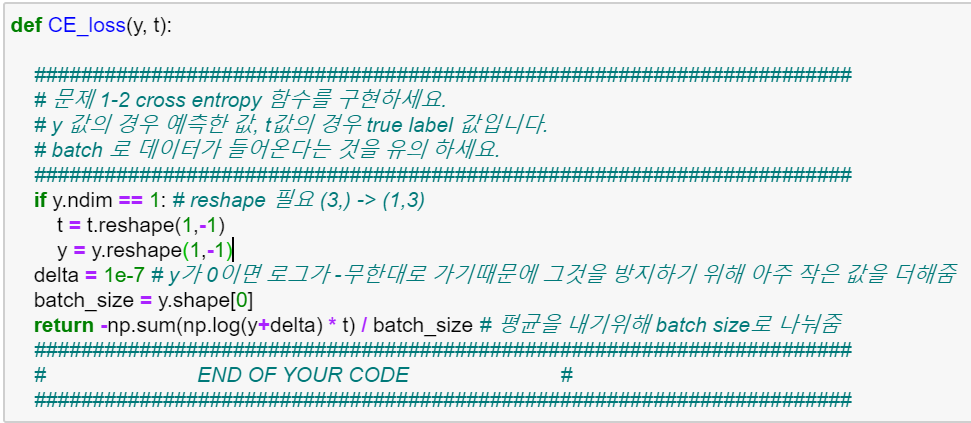
**1-1 softmax 함수 구현**

****

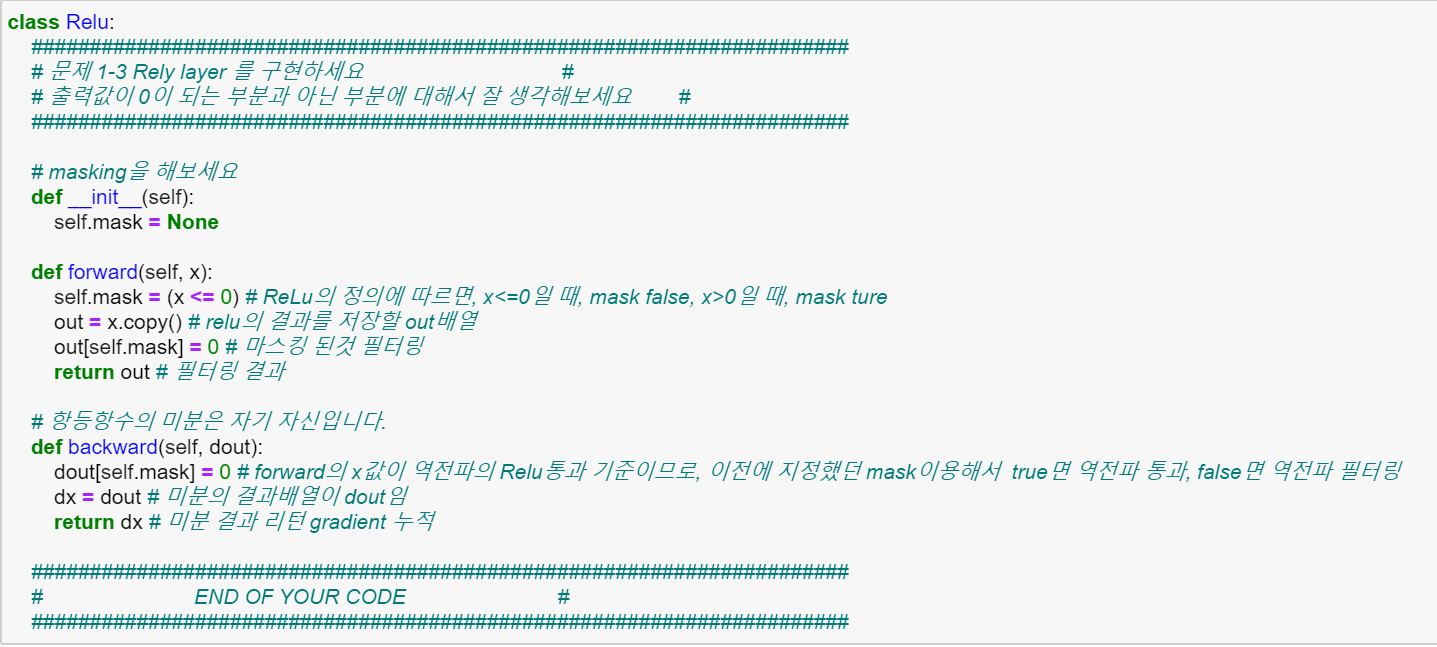
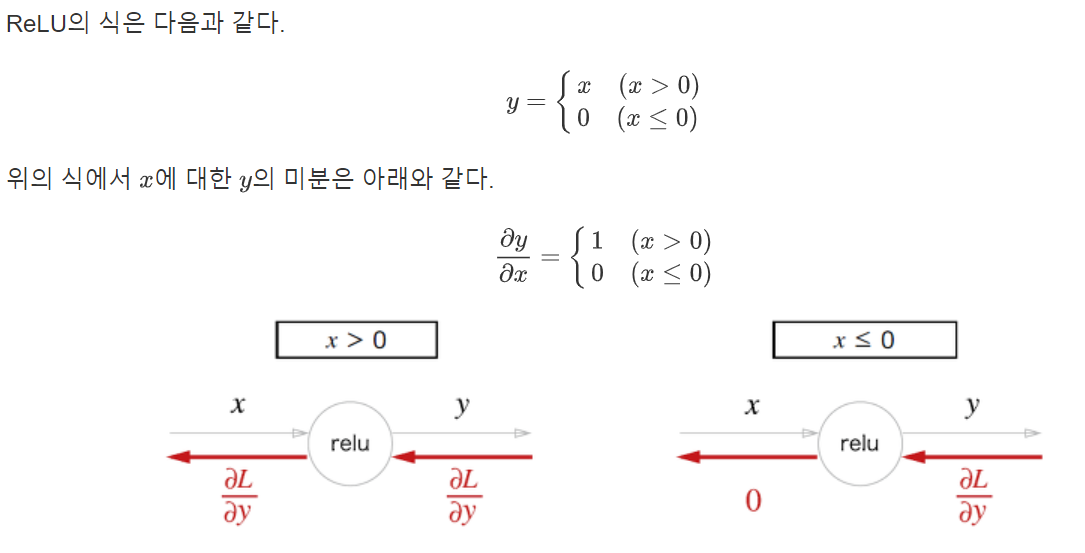
****

**공식에 따라 구현만 해주면 되는 간단한 코딩입니다. 지수함수의 특성상 값이 굉장히 커지기 때문에 overflow위험이 있어서 feature값 중 최대 값을 뽑고 각각의 feature값에 대해 최대값을 빼주고, 이 값에 exponential을 취합니다. 이 값 전체를 더하고 나누면 각각의 feature값으로 분류될 확률 값이 나오게 됩니다. 계산의 편의성을 위해 행벡터('ㅡ')형태로 transpose를 취해서 진행했습니다.**

**1-2 cross entropy 함수 구현**

****

**softmax와 마찬가지로 정의대로 구현만 하면 되는 간단한 코딩입니다. 입력형태를 1차원 벡터형태로 만들고, 원 핫 인코딩 형태의 t벡터와 확률 값으로 이루어진 y벡터의 각 요소에 log값을 취하고 곱하여 실제 레이블로 예측되었을 확률들을 더하고, 배치 사이즈로 나눠 평균을 구합니다. 다만 log함수의 특성상 0이 들어오면 음의 무한대로 발산하고, y값은 확률 값이기에 0이 가능하기에 그 것을 방지하기위해 보정값으로 아주 작은값인 delta를 더했습니다.**

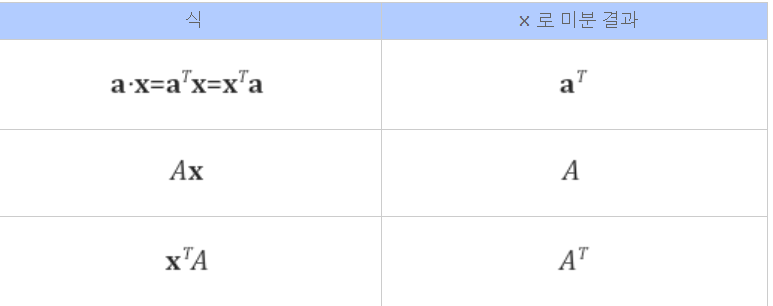
**1-3 Relu layer 구현**

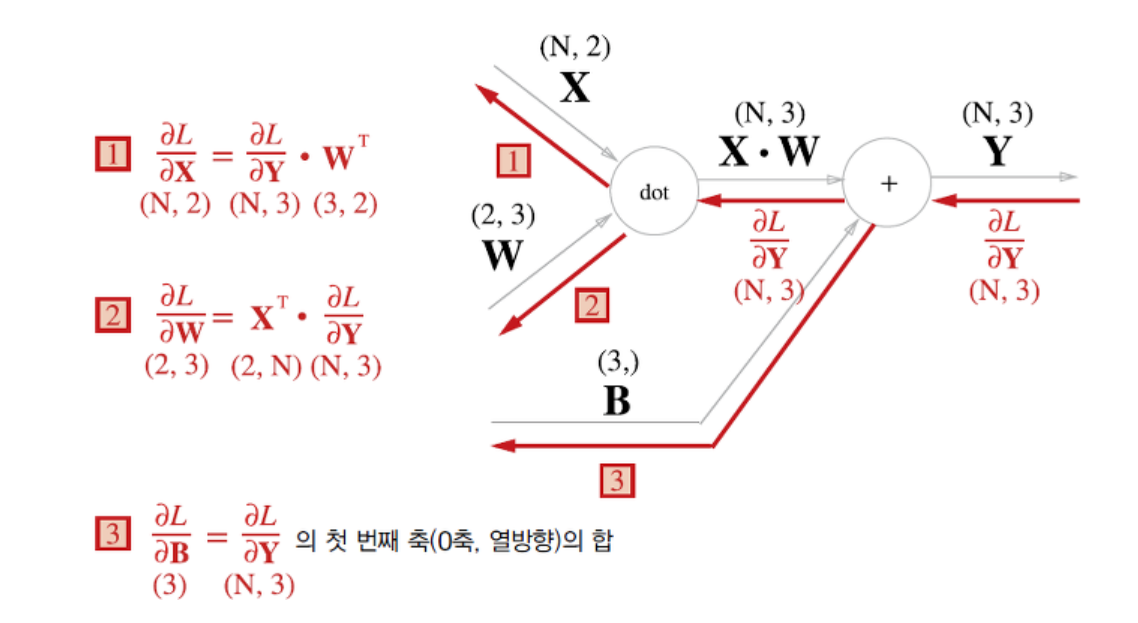
**Relu함수의 정의는 0보다 작거나 같으면 0이 되고, 크면 항등함수가 되는 함수인데, forward 방향은 이 정의대로 구현하면 됩니다. masking(0 또는 1)을 만들어주기위해 mask벡터에 0또는 1의 값을 저장하고, x벡터를 그대로 out벡터로 복사하고 mask가 0이었던 것을 0으로 만들어서 자기자신, 0으로 구성된 out벡터를 만듭니다, backward방향도 비슷합니다. forward의 마스킹 기준을 가지고, 0보다 크면 미분값을 그대로 통과시키고, 0보다 작거나 같다면 0으로 만듭니다.**

**이 과정을 위해 forward에서 생성하였던, mask를 재사용합니다. layer를 거슬러오며 미분값이 담긴 dout벡터에서 mask값에 따라 0을 만들거나 그대로 통과시켜주면 됩니다.**

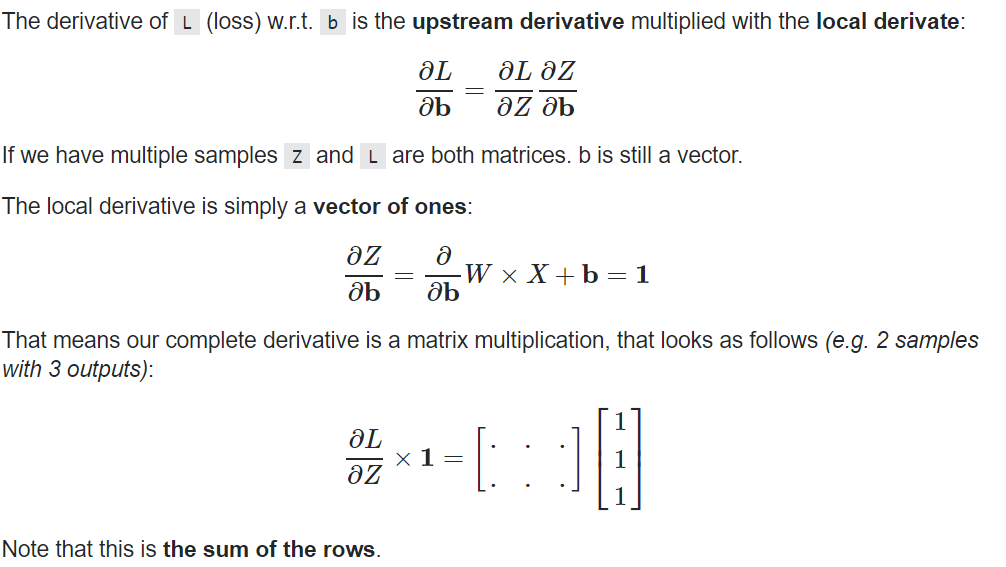
**1-4 FC layer 구현**

**선형대수에서 행렬의 미분은 아래와 같이 합니다.**

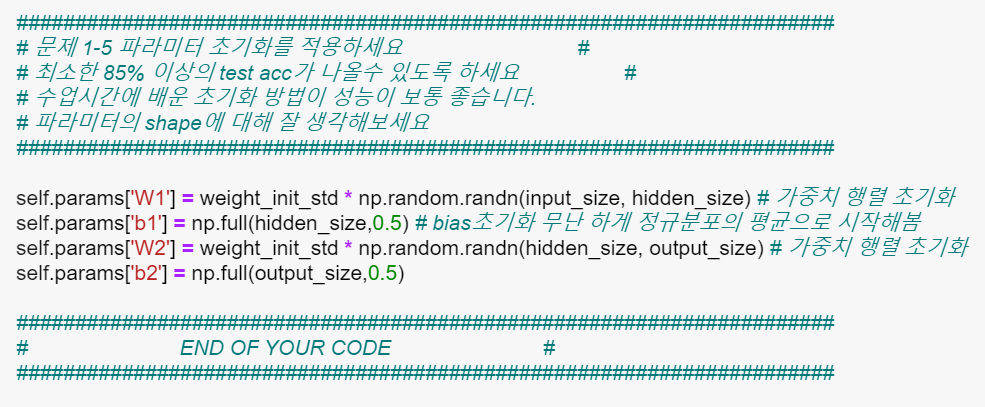
****

****

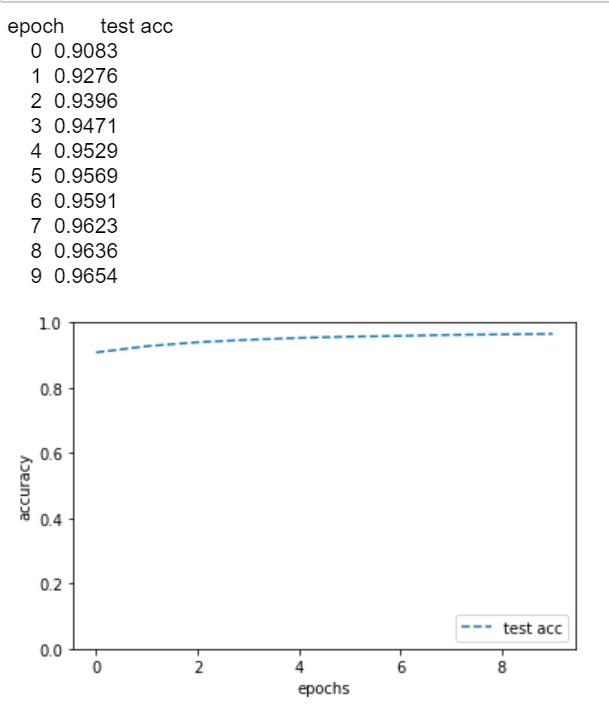
**forward 방향에서 를 하게 됩니다. backward를 할 때, 을 편미분하면 w에 대해서 미분하면 이 되고, x에 대해 미분하면 가 됩니다. 여기에 chain rule에 의해 이전의 미분값 가 곱해집니다. bias는 열의 합이 되는데 chain rule에 의해 아래와 같이 됩니다.**

****

**이 미분학적 이론이 어려울 뿐이지, 프로그래밍은 간단합니다. forward 방향에서 수업시간에는** 으로 배워서 잠깐 혼란이 왔었는데, x의 shape와 w의 shape가 각각 n x m와 m x o 형태로 되어있어서 딱히 transpose를 할 필요가 없었습니다. 이 부분은 디버깅 중 보니 프로그래머가 사이즈 정하는 나름인 것 같습니다.

**1-5 DNN 파라미터 초기화 하기**

**W행렬은 0근처의 값들로 초기화 하는 것으로 수업시간에 배웠습니다. 0근처의 값들의 모임은 가우시안 정규분포의 값들이 적당합니다. 때문에 표준 정규 분포를 따르는 값들을 생성하는 랜덤함수 randn()를 이용하여 초기화 하였고, 히든레이어의 사이즈와 input 벡터사이즈를 맞추기 위해서 크기는 input벡터 크기 x hidden layer 노드 수로 설정해야 합니다. 1개의 layer만 가정하므로 w2를 거친 후에 곧바로 output이 나오므로, w2의 크기는 hidden layer 노드 수 x output 벡터 크기로 설정해야 합니다. bias는 각 layer의 결과와 크기가 같아야 하므로 첫 layer의 bias는 hidden layer의 노드 수와 크기가 같아야 하며, 두 번째 bias는 output vector와 크기가 같아야 합니다. W행렬을 표준정규분포를 따르는 값들로 채웠기에 bias도 초기화 시, 무난하게 표준정규분포에서 평균인 0.5로 시작했습니다.**

**1-6 최종 학습결과 제출**

**학습을 거듭할수록(epochs 증가) 성능이 좋아지는 것을 볼 수 있었습니다. w행렬을 1회에서도0.9083이라는 엄청난 성능을 보여줬습니다.**

**이상입니다 감사합니다**