07-1

data(x) preprocessing for gradient descent : normalization(노말라이제이션의 한 종류가 standardization이다)

내가 사용한 배치노말라이제이션도 여기에 들어갈듯??

overfitting을 피하기위한 방법!

문제를 적절히 가르는 선은 1차선(직선)으로 표현되는 모델일수도있고

구불구불한 2차이상의 선일수도있는데

보통 복잡한 모델일수록 선이 구불해야하는건 사실이고, 그걸 표현하기위해서 레이어의 수가 많아져야하는거다

근데 그 레이어의수(웨이트수)가 많아지면 구불구불해지는것도 엄청 가능해져서 결국 오버피팅처럼 존나 딱맞는모양으로 구불구불해질수도있다.

그래서 그 구불구불해질수잇는 정도를 제한하기위해서 사용하는 기법이 바로 Regularization이다.

이 선이 구불구불해지는 부분이 생기는 이유는 어떤 특정한 웨이트의 값이 존나 클때 발생한다.

그래서 이 구불구불한걸 펴기위해서(제한하기위해서) 비용함수에 웨이트의 값이 너무 커질수없도록 조치를 취하는걸 레귤러라이제이션이라할수있다.

람다 x 시그마 w^2 을 코스트함수에 더하면되는데

이 람다(레귤러라이제이션strength)의 값이 0이면 레귤러라이제이션을 안쓰는것이고

1이면 레귤러라이제이션을 꽤하겠다는거고

0.01은 꽤뭐..쓰겟다

0.1이면 오 중요하게생각한다

0.001~ 등의 수이면 뭐 어느정도 하겠다는거다.

로우피팅 = bias가 높다고 표현하고(얼마나적절하지않은가)

오버피팅 = variance가 높다고 표현한다(얼마나평균가가까운가)

07-2

validation set

08-1

레이어 한개를 가지고 xor형 데이터를 학습시켜봤더니 아무리 학습시켜도 50% 정확도밖에는 안나왔다.

컨볼루셔널 동작 구조를 한번 제대로 뜯어서 살펴보기

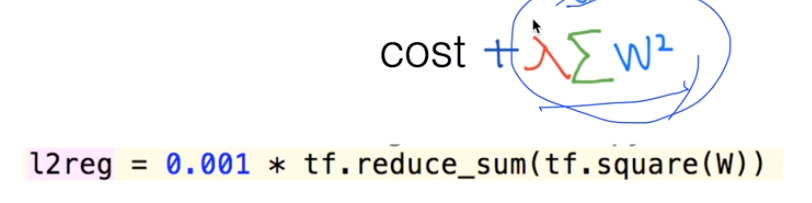
10-3

solutions for overfitting

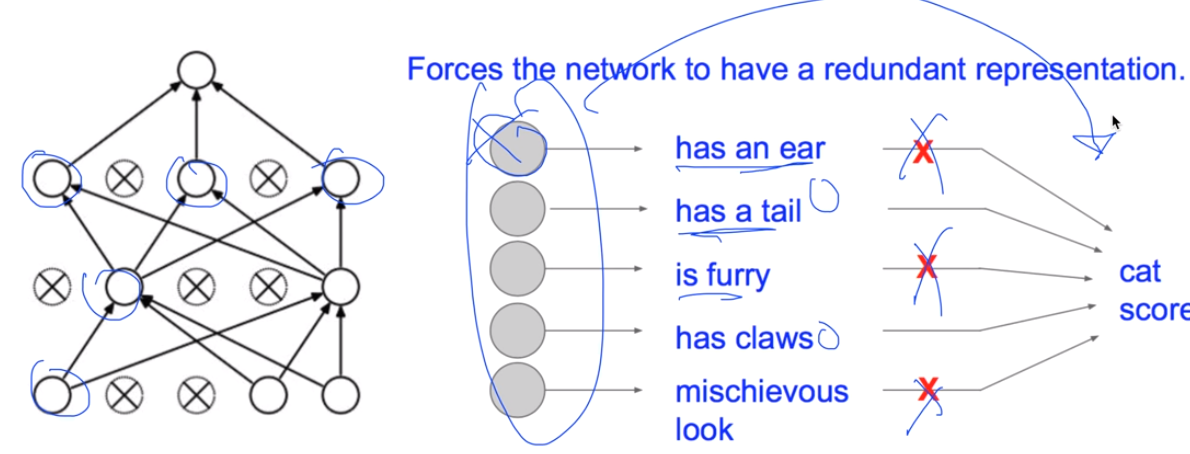
1. 데이터양을 늘리자

2. Regularization

12reg = 0.001 \* tf.reduce\_mean(tf.square(W))



3. drop-out



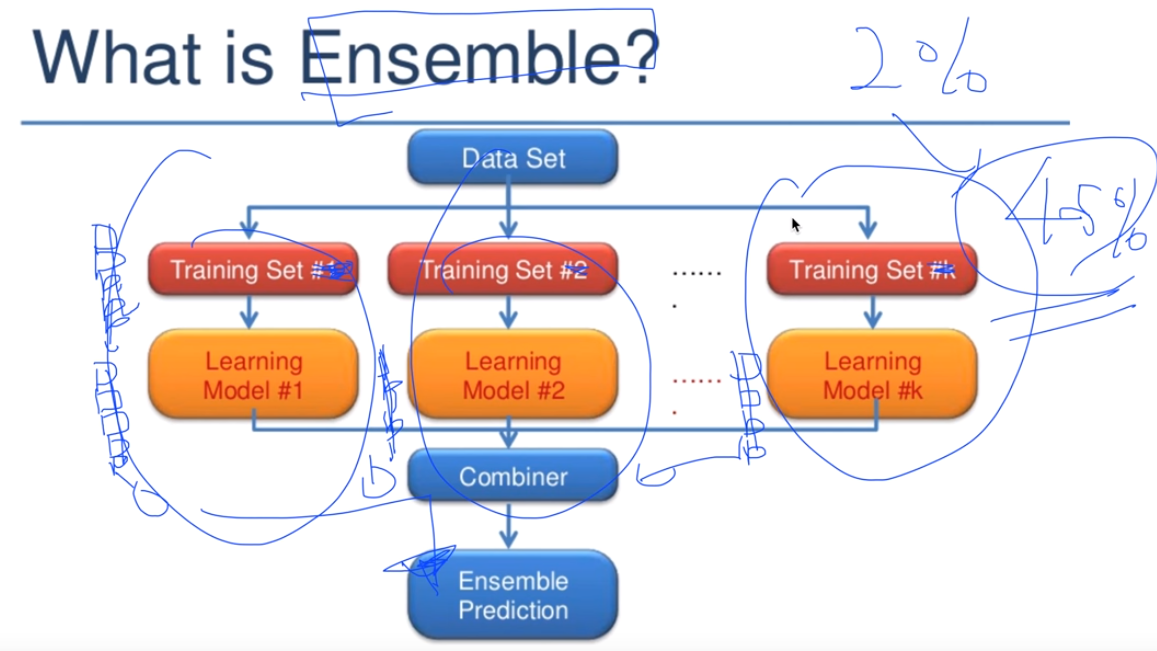
<Ensemble(앙상블)>

일반적으로 데이터셋이 클 때 사용한다.

각각의 모델을 만들어놓고 각각 학습시키고 나중에 합쳐서 결과를 낸다.

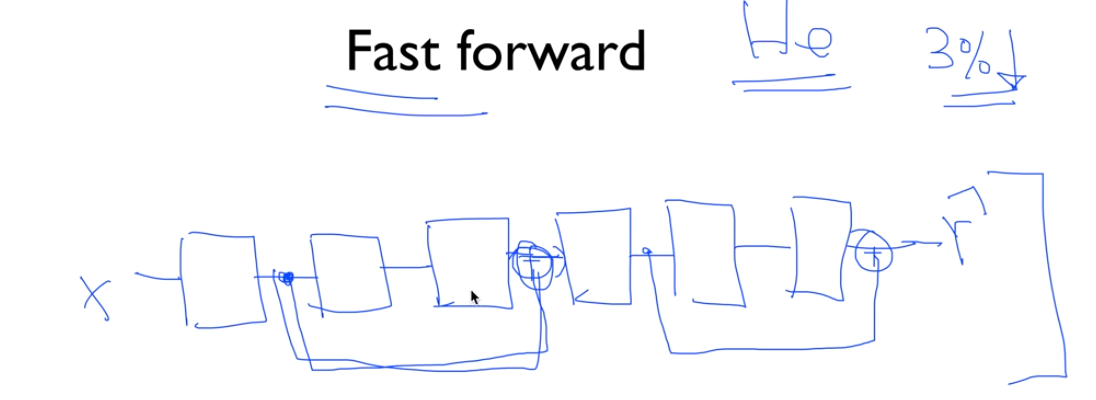
마치 여러명의 독립된 전문가에게 물어서 정답을 찾아가는 방식이라 생각하면 된다.

실제로 2~5% 정도까지 성능향상이 되기 때문에 실전에서 사용하면 굉장히 좋다고 함.

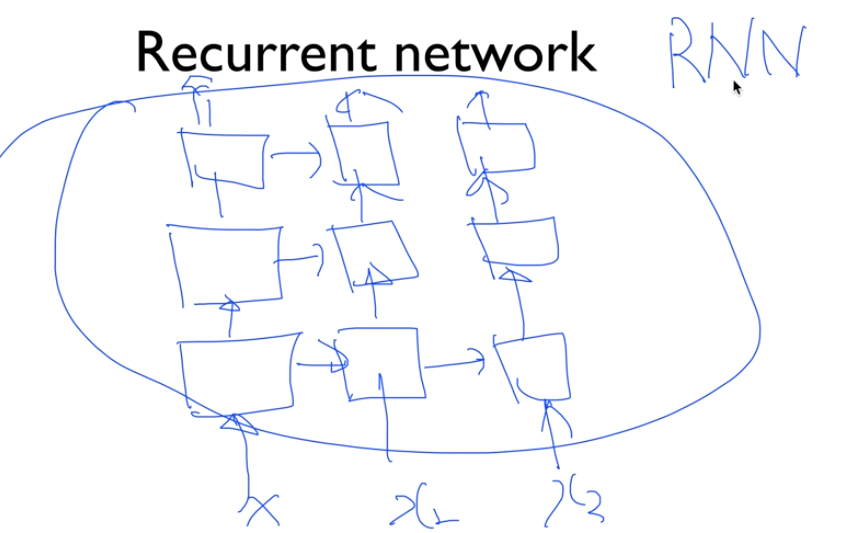


10-4

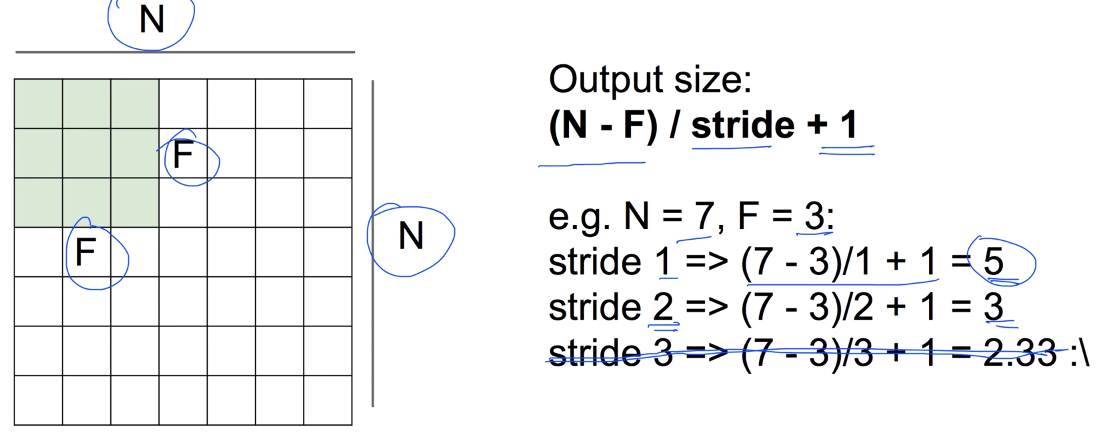
Fast forward? Lesnet!



RNN – 옆으로도 뻗어나가는…

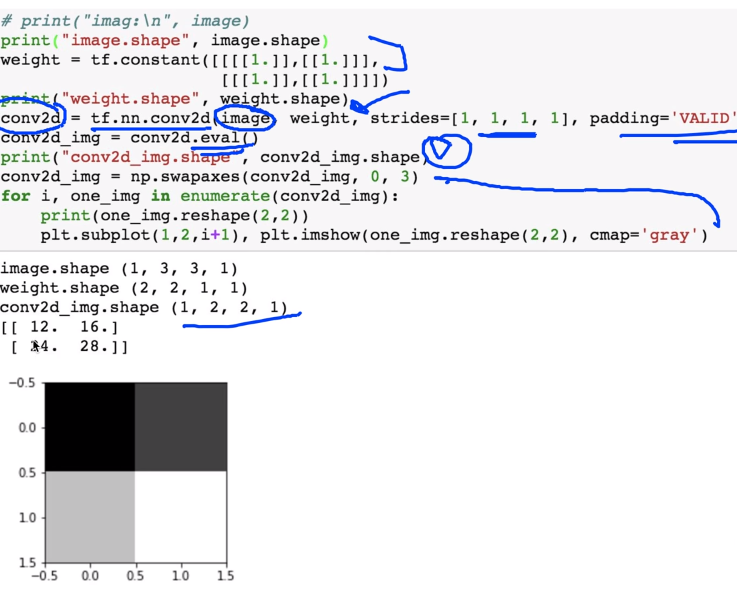


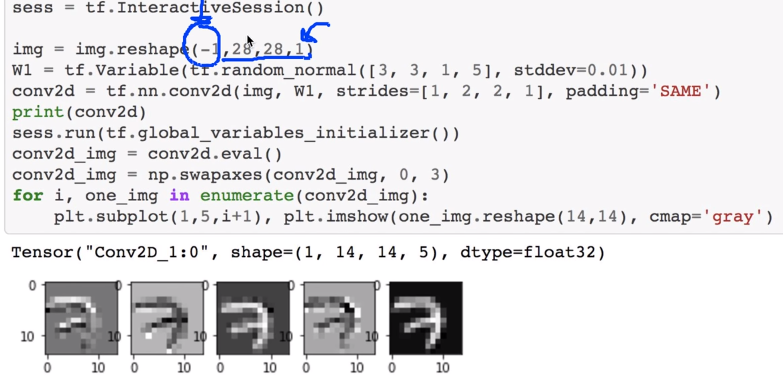
11-1



LAB 11-1

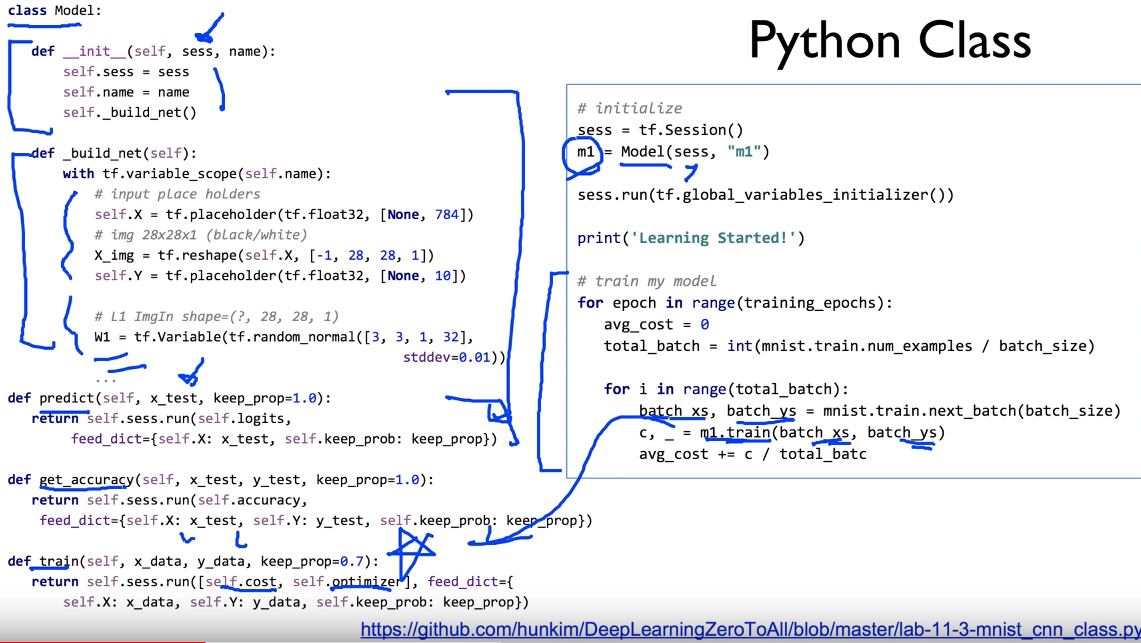
# Visualize, 그리고 tf. InteractiveSession



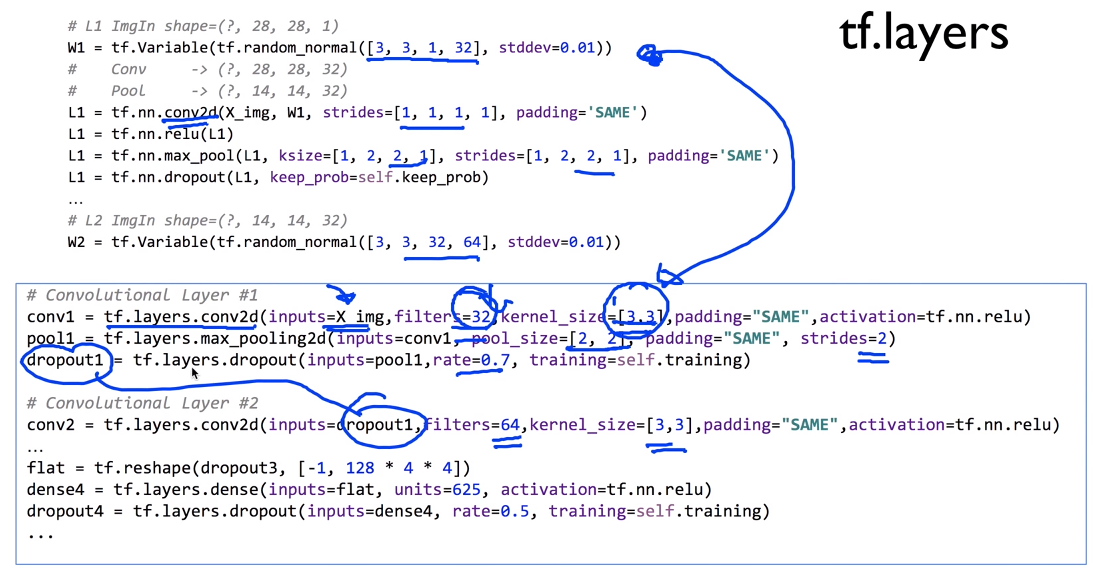


LAB 11-3

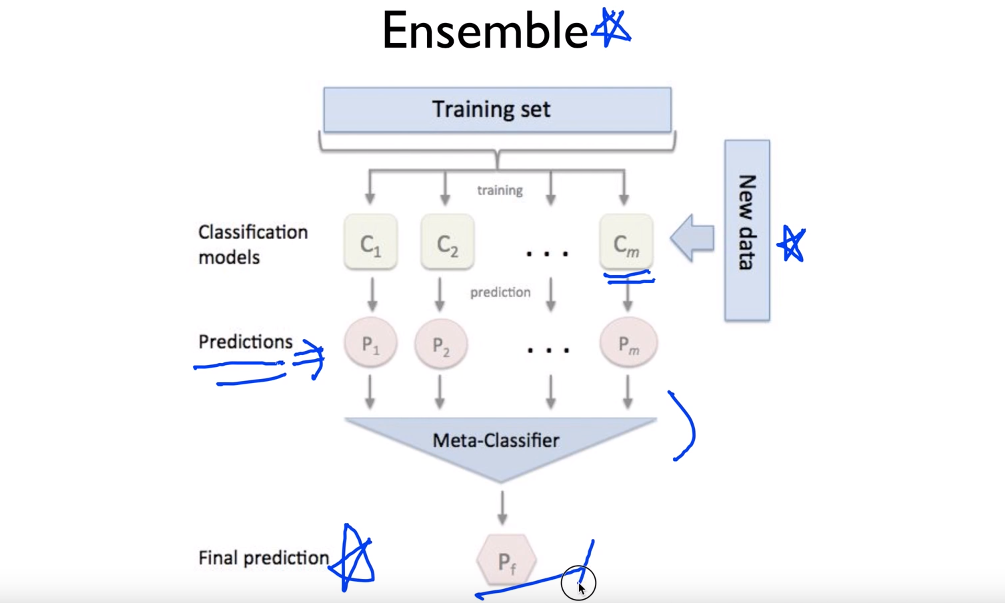
네트워크를 CLASS를 사용해서 간결하게 정리하기

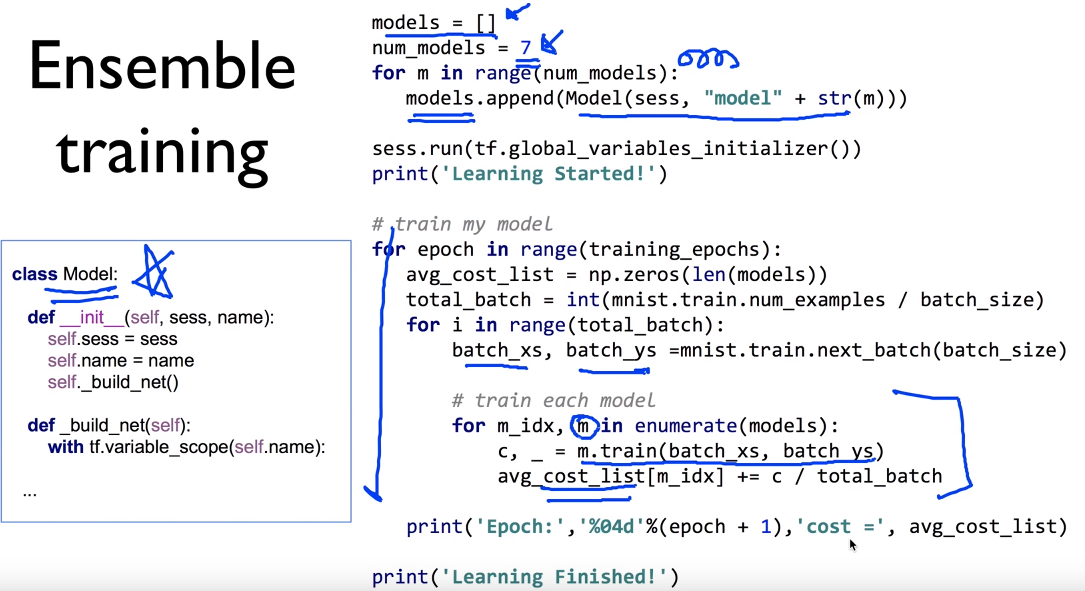


tf.layers



Ensemble (앙상블)

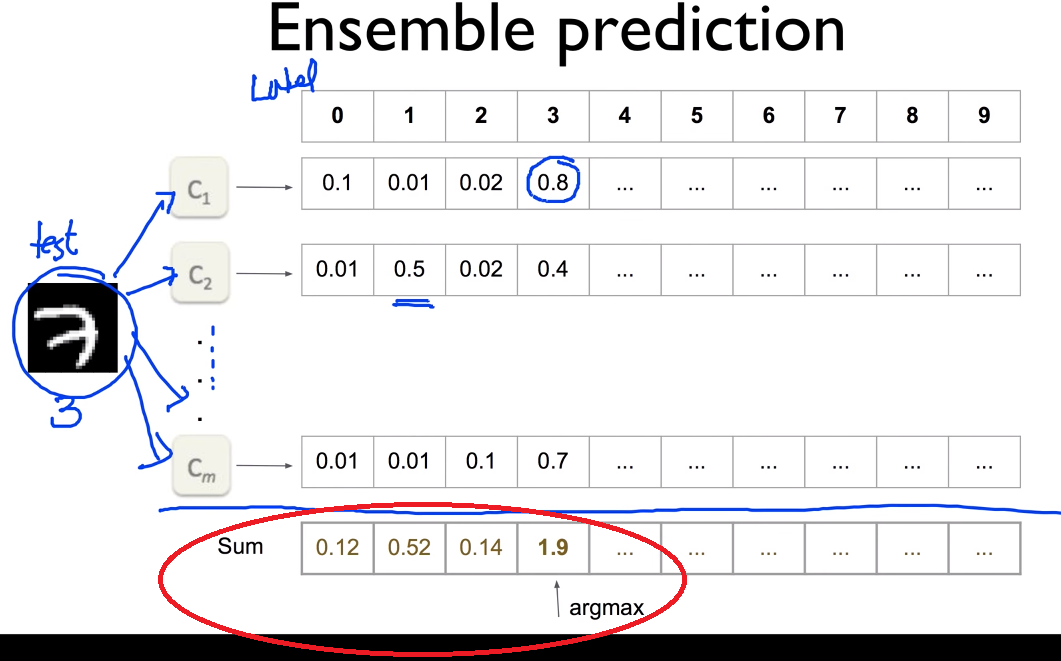




이게 앙상블하는 방식인데, 같은 데이터로 모델만 여러 개 만들어서 학습을 따로 시킨다.

그러면 조금씩 웨이트값이 다르게 학습이될것이고, 결과적으로 새로운 데이터에 대한 test를 진행할때는 생성한 모든 네트워크에 똑같이 집어넣어서 결과를 각각 받아내서 어떠한 알고리즘으로 최종 결과를 정해낸다.

이렇게하면 분명 결과는 더좋아질거같긴한데. 그렇지만 시간이 몇배로 더 걸리는 일 아닌가?



가장 간단한 방식의 앙상블(합방식)이다. 모든 네트워크가 각각 class에 대한 예측값을 내면 그걸 다 더해서 가장 높은 class가 정답이 되는 방식.

