[Week 11] 1. Tuning Process

하이퍼파라미터 튜닝 팁.

Searching hyperparameters

중요한 순서

- **learning rate : α**
- **momentum : β (default:0.9)**
- **#of hidden units**
- **mini-batch size**
- **#of layers**
- **learning rate decay**
- **adam opt. param :β_1, β_2, ε (0.9,0.999,10^(-8) 값을 바꾸지 않고 주로 사용)**

하이퍼파라미터의 값이 많은 딥러닝의 경우 어떤 값을 골라야 적절할까?

1. Choose random values

   어떤 하이퍼파라미터가 문제해결에 가장 큰 도움이 되는지, 중요한지 알 수 없기 때문이다. 무작위로 고르게 되면 더 좋은 하이퍼 파라미터 값을 찾을 수 있게 된다.(격자점을 이용하면 반드시 param1의 값이 같을 때 param2를 비교하게 되는데, 이는 param2의 영향력이 적다면 매우 비효율적임. 그러나 무작위로 고르게 되면 모두 다른 param1의 값을 이용해 학습하여 비교할 수 있어 효율적)

2. Coarse to fine (정밀화 접근)

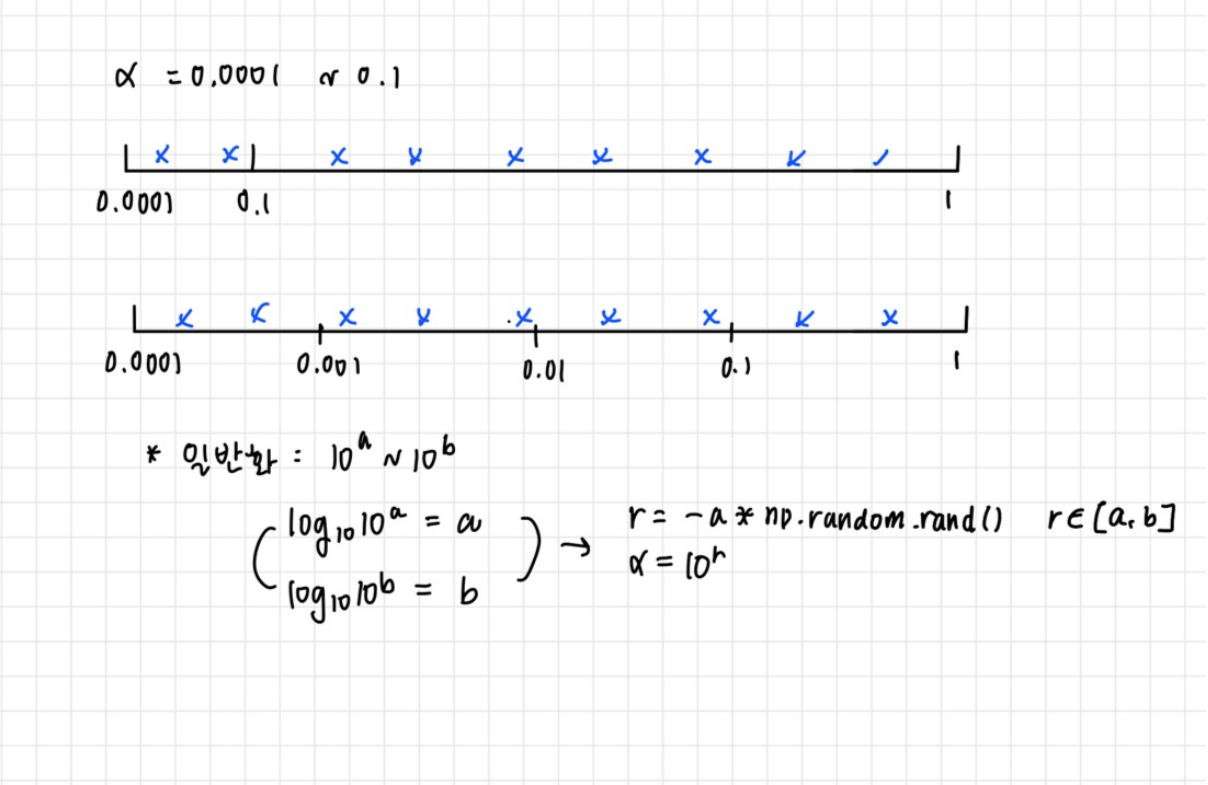
   한 영역에서 높은 정확도를 보이면, 해당 영역을 확대해서 더 조밀하게 점들을 선택한다. 물론 무작위인 것은 동일하나 범위를 좁혀나가는 방식.

Appropriate scale for hyperparameters

무작위라는 것이 공평하게 뽑는다고 볼 수 없다.

Learning rate α

학습률 α를 탐색한다고 할 때, 0.0001~1 까지가 적절한 값이라고 가정해보자.

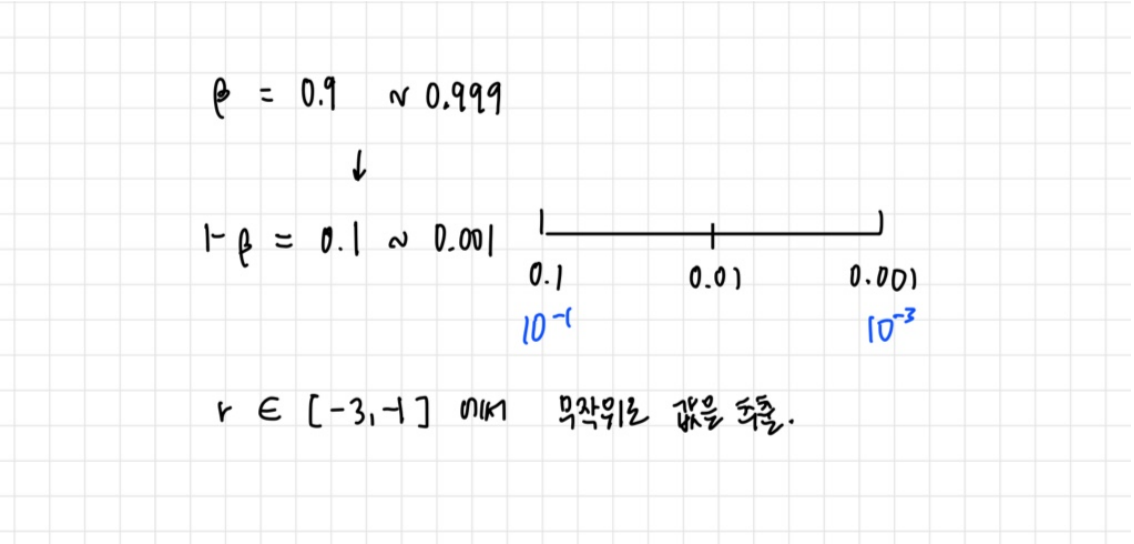


해당 범위에서 균일하게 무작위로 α값을 고른다면, 90%는 0.1~1 사이에 있고 10%는 0.0001~0.1 사이에 있을 것이다. 왜냐하면 확률에 근거하여 0.1~1의 범위가 0.0001~0.1의 범위보다 훨씬 크기 때문이다. 따라서 선형척도 대신 **로그척도**에서 하이퍼파라미터를 찾는 것이 더 합리적이다. 균등한 확률 내에서 랜덤하게 값을 고를 수 있게 되기 때문이다.

implement in log scale

- r = -4 * np.random.rand()
- α = 10^r r[-4,0]

지수가중평균의 하이퍼파라미터 β



β 를 0.9~0.999 사이에서 찾는다고 해보자. 이 값도 scaling 하지 않고 무작위 탐색하는 것은 합리적이지 않다. 1-β 의 값을 찾아보면 더 쉽다. 범위가 0.001~0.1이 되기 때문.

β 가 1에 가까울수록 미세한 변화에도 결과값이 크게 바뀌기 때문에 스케일링을 하는 것이다.

질문)) β 가 0.999→0.9995 는 1000개의 샘플에서 2000개의 값의 평균을 내는 것으로 바뀐다. 왜 2배가 되는거지? 정확한 값이 아닌 비유적인 표현인가?

Hyperparameter tuning in practice : Pandas vs Caviar

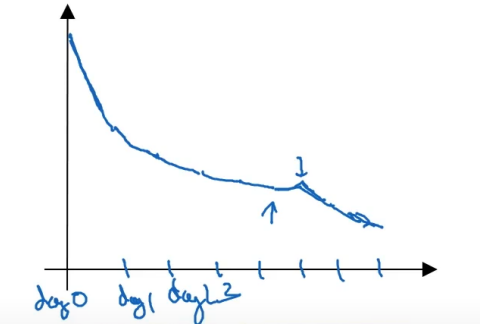
Re-test Hyperparameters occasionally

적절한 하이퍼파라미터를 찾았다고 하더라도, 데이터의 변화나 알고리즘의 변화, 서버의 발전등으로 적절한 하이퍼파라미터의 값이 달라질 수 있다.

그러면 사람들은 어떻게 하이퍼파라미터의 값을 찾을까?

2가지 접근이 있는데, 컴퓨터 자원의 양에 따라 선택하면 된다.

1. Babysitting one model

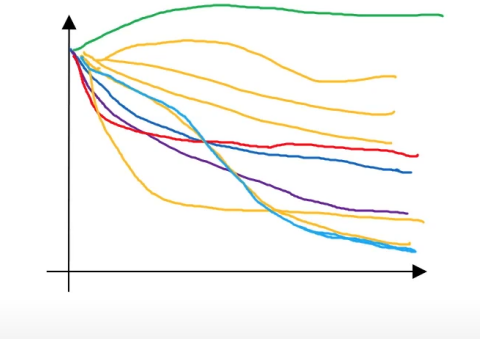


데이터가 방대하여(온라인 광고, CV application등) 모델 자체가 너무 커서 여러 모델을 동시에 학습시킬 컴퓨터자원이 충분하지 않을 때 사용.

심지어 학습과정에서 사용하기도 함. 매일매일 값을 조정하면서 cost function J의 값을 관찰함.

하나의 모델의 성능을 잘 지켜보다가 학습속도를 조금씩 바꿔가는 것이다.

2. Training many models in parallel



우리가 갖고 있는 하이퍼파라미터를 며칠에 걸쳐 스스로 학습하게 하는 것. 동시에 다른 모델의 다른 하이퍼파라미터에 변화를 주어 학습시키면서 여러 모델을 한번에 훈련하고 해당 모델들을 한눈에 비교하여 가장 최적의 모델을 선택.

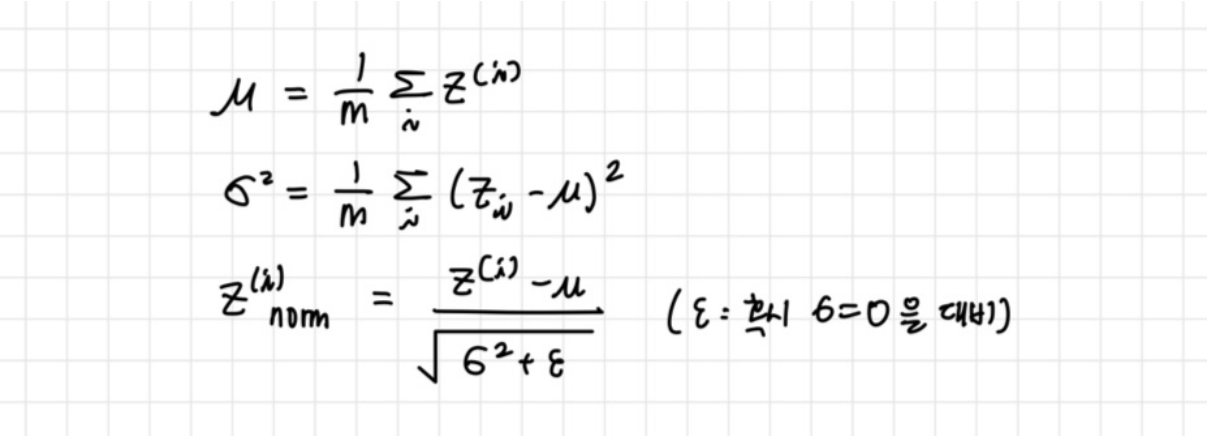
[Week 11] 2. Batch Normalization

로지스틱 회귀에서 입력 변수 X 정규화하면 학습이 빨라졌었다. 같은 원리로 은닉층이 있는 신경망에서 우리가 W^[3],b^[3]의 파라미터를 학습시킨다면, a^[2]의 평균과 분산을 정규화하는 것이 더 효율적이지 않을까? 입력값인 a^[2]가 W^[3],b^[3]의 학습에 영향을 주기 때문이다.

배치 정규화는 입력층에만 정규화를 하는 것이 아니라 은닉층의 값들도 정규화를 하는 것이다. 은닉층 또한 표준화된 평균과 분산을 갖되, 조절될 수 있도록 한다.

사실 a^[2]가 아닌 활성함수를 거치기 이전인 z^[2]를 정규화함.(더 효율적이라고 한다)

신경망에서 사잇값들이 주어졌다고 할 때,

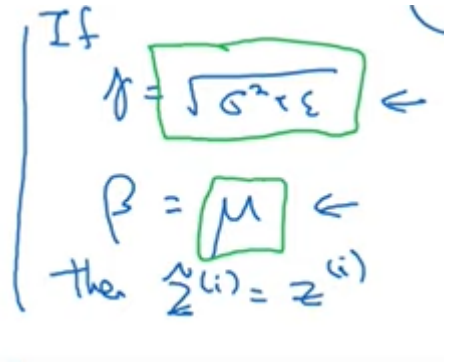


layer l에서 은닉 유닛 값이 z^[1]~z^[m] 이라고 하면, 그동안 우리가 배웠던 정규화를 거쳐 해당 레이어의 유닛들의 z값의 평균이 0이고 표준편차가 1이 되게 만들 수 있다. 하지만 모든 hidden unit이 평균이 0이고 표준편차가 1인것은 좋지 않다. hidden unit 다양한 분포를 가져야 하기 때문이다.

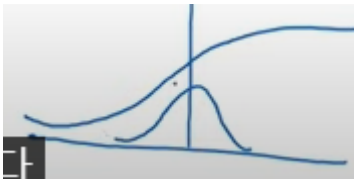
따라서 우리는 z~를 계산한다.

$$\tilde{z}^{(i)} = \gamma z_{norm}^{(i)} + \beta$$

여기서 감마와 베타는 모델에서 학습시킬 수 있는 변수이다. 감마와 베타를 이용하면 z~의 평균을 원하는대로 설정할 수 있다.



위의 정규화 과정은 항등함수를 만드는 것과 똑같은 효과를 낸 것이다. 그러나 다른 감마와 베타 값을 정한다면, 히든 유닛의 값들이 서로 다른 평균과 분산을 가지게 할 수 있다. z^[l](i) 가 아닌 z~^[l](i)를 이제 사용할 것.

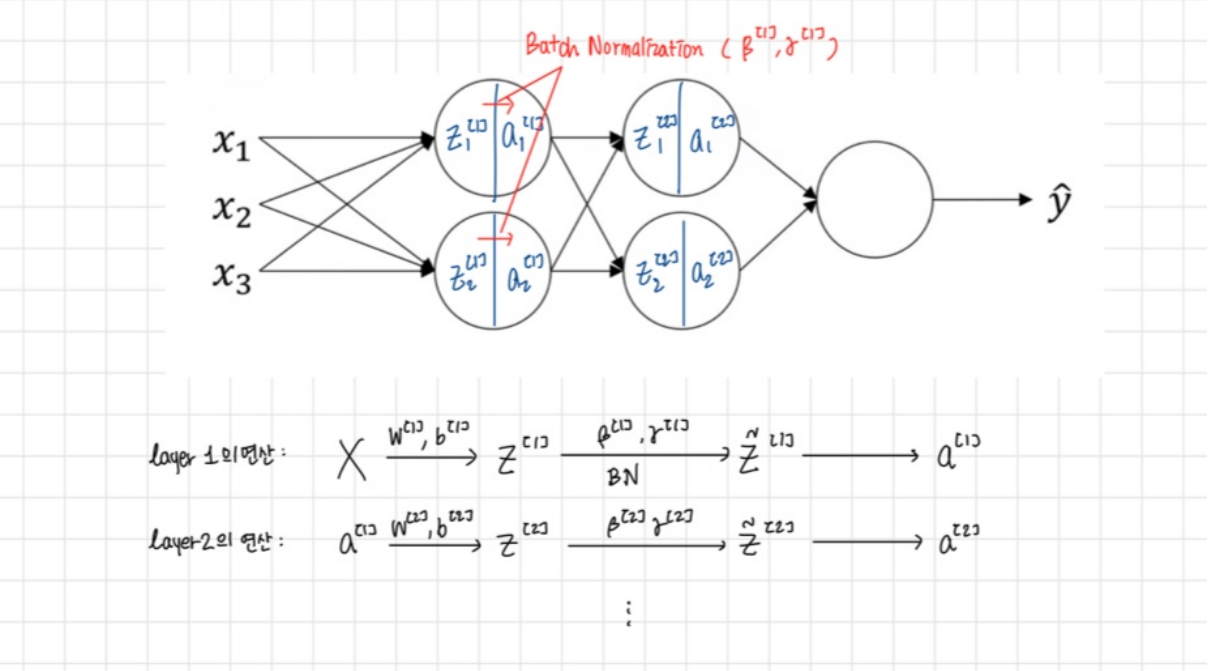


입력층의 정규화와 다른 점은 hidden unit의 값의 평균과 분산이 0,1로 고정되기를 원치 않는다는 점이다. 예를 들어 sigmoid 함수가 있을 때, 시그모이드의 비 선형성을 살릴 수 있록 평균이 0이 아닌 다른 값들을 가지게 하는 것이 좋다.

[Week 11] 3. Fitting Batch Norm into a Neural Network

Batch-norm Process

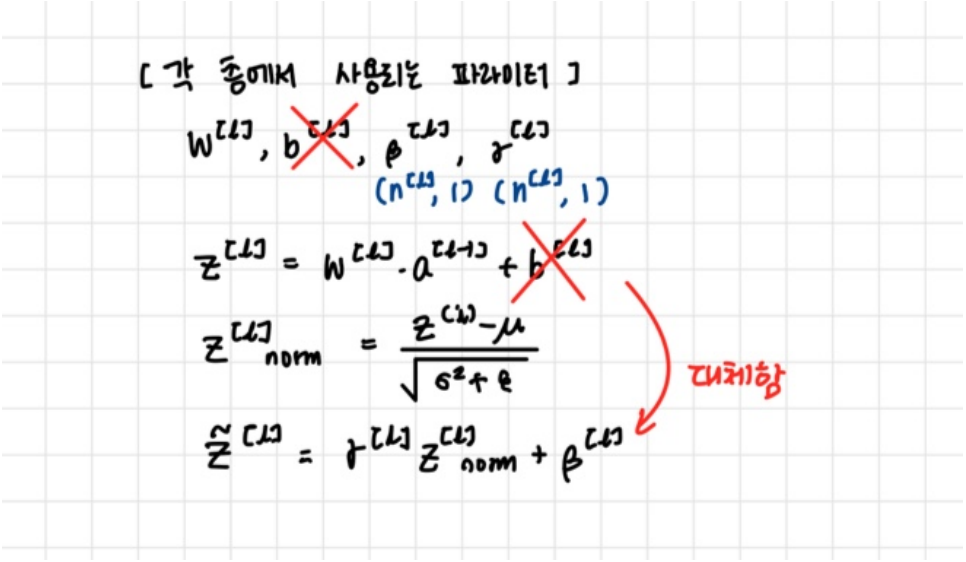
배치 정규화 과정을 끝내고 나면, 어떤 최적화 방식을 사용하여 업데이트를 할지 정해야 한다. 딥러닝 프레임워크를 사용하면 하나하나 구현할 필요없이 코드한줄로 구현이 가능하다.



배치 정규화는 z값 이후, a값 계산 이전에 이루어진다.비정규화된 z값을 베타와 감마를 이용해 정규화된 값 z~로 바꾸어 사용하는 것이다.

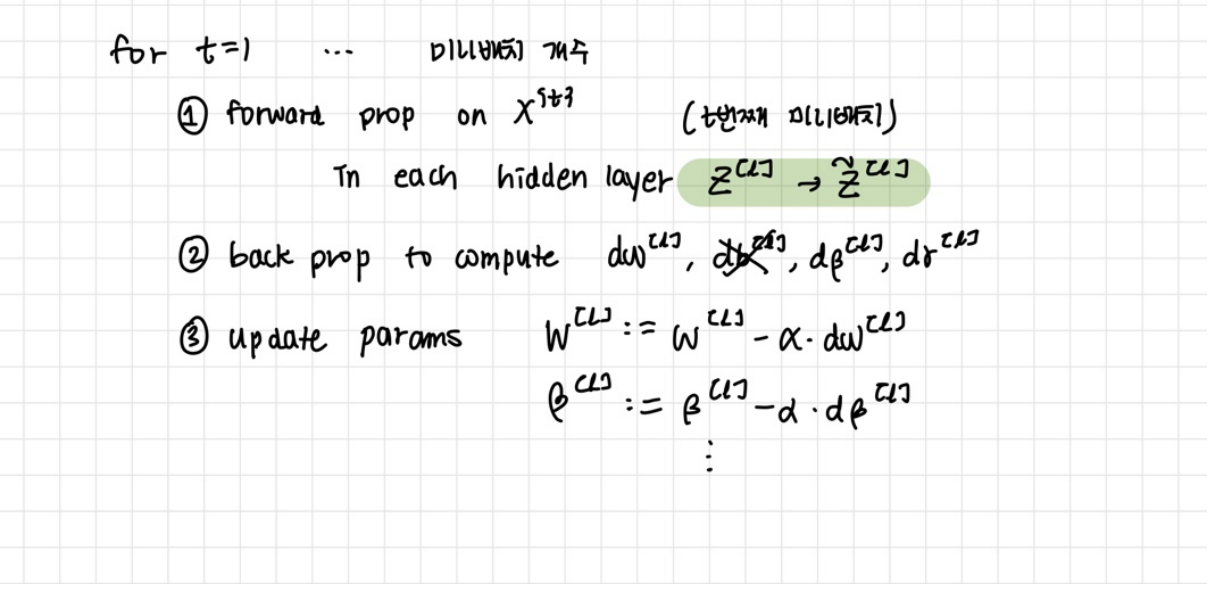
*모멘텀이나 최적화 알고리즘에 쓰이는 베타와 다름!

실제 배치 정규화는 training set의 mini-batch에 적용된다. 미니배치 안에서의 평균과 분산을 계산하여 정규화가 진행된다는 것이다. 이 값을 베타와 감마를 통해 z~로 값을 조정해 준다.



학습과정에서 사용되는 파라미터를 위와 같다고 배웠다. 그러나 b^[l]은 값이 무엇이든지 간에 사라진다. 왜냐하면 배치 정규화의 과정에서 평균을 연산한 뒤에 빼주기 때문이다. 따라서 배치 정규화를 쓴다면 b라는 변수값은 필요가 없어짐.

Implementing GD with Batch-normalization



[Week 11] 4. Why does Batch Norm work?

1. **Covariate Shift 완화**

X→Y를 학습하는 모델을 만들때, X의 분포가 변하면 다시 학습해야 함을 의미.

앞의 은닉층의 값들이 학습하면서 계속 변화하기 때문에, 뒤따르는 유닛들은 공변량 변화의 문제를 겪게 된다. 배치 정규화는 **은닉층 값들의 분포가 변화하는 양을 줄여준다. 물론 w,b값이 바뀌어도, z의 평균과 분산은 그대로 유지되기 때문이다.** 배치 정규화는 앞선 층에서 매개변수가 바뀌었을 때 다음 층에서 받아들일 값의 분포를 제한한다는 것이다.

또한 앞쪽 층의 매개변수와 뒤쪽 층의 매개변수간의 관계를 약화시킨다. 즉, 각각의 유닛이 독립적으로 학습할 수 있게 하는 것이다. 이는 전체 신경망의 학습 속도를 향상시킬 수 있다.

2. **Regularization**

미니배치는 전체 데이터셋에 비해 상대적으로 작은 데이터이기 때문에 variance가 클 수 밖에 없다. 따라서 z~을 구할 때에도 잡음이 생길 수 밖에 없고, 이는 약간의 일반화 효과를 가지고 있다.

*하지만 배치 정규화를 일반화의 목적으로 사용하지는 말 것. 학습속도를 올리는데 사용해라.

Batch norm at Test time

학습과정에서는 미니배치만큼을 한번에 처리하지만, test에서는 한번에 샘플 하나씩을 처리해야 한다.

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$$

위는 우리가 학습과정에서 사용했던 식이다. 그러나 test에서는 샘플을 하나씩 처리하므로 평균과 분산을 처리할 다른 방법이 필요하다.

각각 독립된 평균과 분산의 추정치를 사용하면 된다. 여러 미니배치 과정 중에 구한 지수가중평균을 추정치로 사용한다. 즉, training 과정에서 여러 미니배치를 통해 누적된 평균값과 분산값 각각의 지수가중평균을 구하여 사용하는 것.