**배치 정규화**

**딥러닝이 떠오르면서 가장 중요한 아이디어 중 하나로 배치정규화라는 알고리즘**이 꼽힌다. Sergey loffe와 Christian Szegedy가 만들었다. **배치 정규화는 하이퍼파라미터 탐색을 쉽게 만들어줄 뿐만 아니라 신경망과 하이퍼파라미터의 상관관계를 줄여준다.** 즉, 더 많은 하이퍼파라미터가 잘 작동하는 것이다. 이는 아주 깊은 심층신경망이라도 아주 쉽게 학습할 수 있도록 도와준다. 어떻게 작동하는지 살펴보자.

**Nomalizing inputs to speed up learning**

 여러분이 로지스틱 회귀 등으로 모델을 학습시킬 때 입력 변수들을 정규화하면 학습이 빨라졌던 것을 기억해보자. 평균을 계산할 때는 입력변수의 평균을 뺐고 분산을 계산할 때는 x(i)^2를 사용했다. (하나씩 제곱해준다는 것임) 이 값을 정규화해준다.

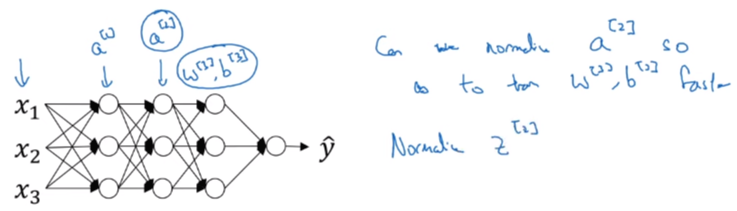
 앞선 영상에서 이 방법이 어떻게 누워있는 학습 등고선을 경사하강법에 접합토록 더 둥근 형태로 바꾸는지 보았다. 즉, 로지스틱 회귀 등 신경망의 입력 변수들을 정규화하면 저렇게 바뀌는 것이다.

친필, 라인, 폰트, 텍스트이(가) 표시된 사진

자동 생성된 설명

 심층 신경망은 어떨까? 여기서는 입력변수 x뿐 아니라 은닉층의 활성값 a^[1], a^[2]와 같은 것들이 있다. 만약 w^[3], b^[3]라는 파라미터를 학습시킨다면 a^[2]의 평균과 분산을 정규화하는 것이 더 효율적이지 않을까? 로지스틱 회귀의 경우 x\_1, x\_2, x\_3를 정규화하는 것이 어떻게 w, b 학습을 효율적으로 하는지 보았다.

여기서 **질문은 은닉층에 대해 w^[3], b^[3]를 빠르게 학습시킬 수 있도록 이 예시의 a^[2]같은 값을 정규화시킬 수 있을까?** 왜냐하면 다음 층 입력값인 a^[2]가 w^[3], b^[3]의 학습에 영향을 주기 때문이다. 이 질문은 배치 정규화가 하고 있는 일을 나타내고 있다.**사실 a^[2]가 아니라 z^[2]를 정규화하는 것**이다. 이에 대해서는 논쟁이 있긴 하다. 활성 함수 이전의 값인 z^[2]를 정규화할 것인지 활성함수 이후의 값인 a^[2]를 정규화할건지 사이에서 말이다.**실제로는 z^[2]를 정규화하는 것이 더 자주 쓰인다.**

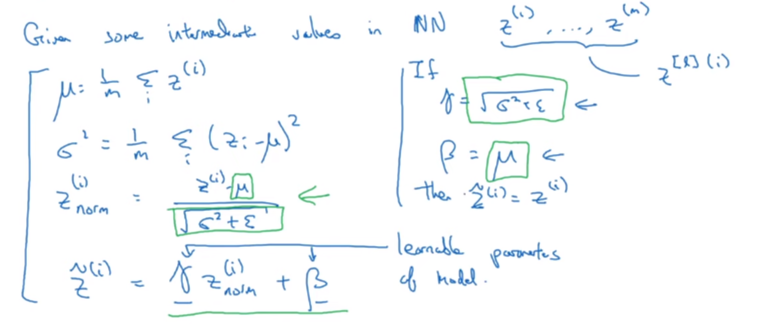


**Implementing Batch Norm**

 이제 **어떻게 배치 정규화를 구현하는지** 보자. 신경망에서 사이값들이 주어졌다고 할 때 은닉 유닛의 값 z^(1)부터 z^(m)까지 있다고 하자. 이 값들은 은닉 유닛의 값들이다. 더 정확하게 쓰자면 z^[l](i)라고 써야한다. i는 1부터 m까지이다. 하지만 간편하게 쓰기 위해서 [l]을 생략한다. 이 값들에 대해서 우리는 평균을 이렇게 계산한다. 그리고 분산을 계산한다. 그리고 각 z^(i)에 대해서 정규화를 하여 z^(i)\_norm을 얻는다. 평균을 뺀 뒤에 표준편차로 나누면 된다. 수학적 안정성을 위해서 분모에 ε를 추가한다.  ε은 표준편차가 0인 경우를 대비한 것이다. 이렇게 z값에 대해서 정규화를 거쳐 모든 z들이 평균이 0이고 표준편차가 1이 되도록 만들었다.

하지만 은닉 유닛이 항상 평균 0, 표준편차 1을 갖는 것이 좋지만은 않다. 말이 될지 모르지만 은닉 유닛은 다양한 분포를 가져야하기 때문이다. 그래서 대신 z~(틸드?)를 계산한다. 이는 γ\*z^(i)\_norm+β와 같다. 여기서 γ와 β는모델에서 학습시킬 수 있는 변수이다. 즉 여기에서 경사하강법을 이용해서 아니면 모멘텀, RMSprop, Adam을 이용한 경사하강법 등 신경망에서 계수들을 찾았던 것처럼 말이다. y와 β를 이용하면 z~의 평균을 원하는 대로 설정할 수 있다. 예를 들어 γ=√σ^2+ε로써 분모와 같고 β가 μ와 같다면 γ\*z^(i)\_norm+β는 이 식(왼쪽 수식의 세번째, 초록색 화살표)을 거꾸로 뒤집은 것과 같은 효과를 낼 것이다. 만약 이 식이 성립한다면 z~^(i)는 z^(i)와 같게 된다.

이렇게 적절히 γ와 β를설정해서 **왼쪽의 네개의 식으로 이루어진 정규화 과정은 항등함수를 만드는 것과 똑같은 효과**를 낸다.  하지만 다른 γ와 β값을 정한다면 은닉 유닛의 값들이 서로 다른 평균이나 분산 값을 만들게 할 수 있다.

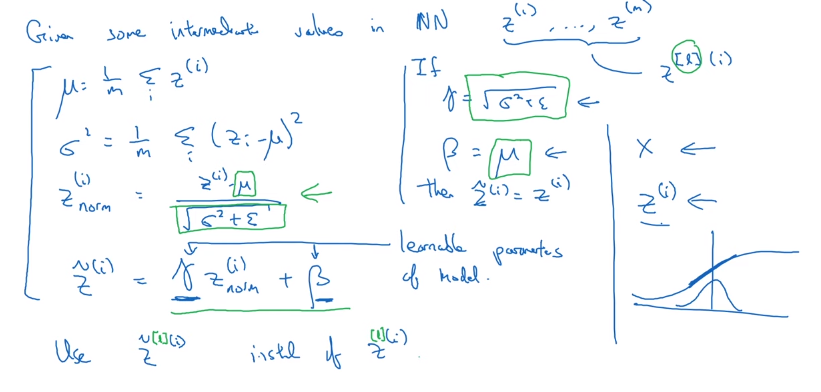
 폰트, 친필, 화이트, 텍스트이(가) 표시된 사진

자동 생성된 설명

 이제 이걸 신경망에 적용시켜 보자. 이전에는 여기 z^(1) 등의 값을 썼다면 이제는 z~^(i)를 z^(i) 대신 신경망에 사용하는 것이다. 그리고 어떤 층인지 알 수 있도록 다시 [l]을 써넣을 수 있다. 여기서 얻었으면 하는 직관은 우리가 X를 정규화하는 것이 신경망 학습을 도울 수 있는 것을 봤듯이 **배치 정규화는 입력층에만 정규화를 하는 것이 아니라 신경망 안 깊이 있는 은닉층의 값들까지도 정규화하는 것**이다.

이런 정규화를 사용해서 은닉 유닛 z의 평균과 분산을 정규화한다. 하지만 입력층과 은닉 유닛을 학습시킬 때 차이점은 은닉 유닛 값의 펴균과 분산이 0, 1로 고정되기를 원치 않는다. 예를 들어 시그모이드 활성함수가 있을 때 값들이 한 구간에 모여있는 것을 원치는 않는다. 더 넓은 영역에 걸쳐 펴져있거나 시그모이드의 비선형성을 살릴 수 있도록 평균이 0이 아닌 다른 값을 갖게 하는 것이 좋다. 이것이 γ와 β를 이용해 원하는 범위의 값을 만들어내는 이유이다.

즉 은닉 유닛이 표준화된 평균과 분산을 갖되 평균과 분산은 학습 알고리즘에서 설정할 수 있는 두 변수 γ와 β에 의해 조절되는 것이다. 다시 말하자면 은닉 유닛값 z^(i)의 평균과 표준편차를 특정한 평균과 분산을 갖도록 정규화하는 것이다. 이것은 0이나 1이 될 수도 있고 또 다른 값이 될 수도 있다. γ와 β의 조절에 따라서 결정된다.



**배치 정규화 적응시키기**

**Adding Batch Norm to a network**

 아래와 같은 신경망이 있다고 하자.**은닉 유닛을 두가지로 나눌 수 있다**. **z를 우선 계산**하고 **활성화 함수를 이용해서 a를 계산**한다. 즉, 원 하나하나가 **두 단계의 계산**을 나타낸다. 다음 층에서도 비슷하게 z^[2]\_1와 a^[2]\_1이 된다.

라인, 도표, 원, 그래프이(가) 표시된 사진

자동 생성된 설명

 i) **만약 배치 정규화를 쓰지 않는다면** 입력값 x가 첫번째 은닉층에 주어지게 된다. 그러면 w^[l]과 b^[l]에 따라 우선 z^[l]을 계산한다. 그리고 활성화 함수 z^[l]이 주어져서 a^[l]을 계산한다.

 ii) 하지만 **배치 정규화에서는**

**(1) z^[1]을 받아서 배치정규화(BN)를 시킨다.** (BN은 β^[1]와 y^[1]의 영향을 받음)

**(2) 새로 정규화된 z^[1] 값들을 활성화 함수를 거쳐 활성화 값 a^[1]을 얻는다.**

텍스트, 라인, 친필, 도표이(가) 표시된 사진

자동 생성된 설명

 이렇게 첫번째 층에 대한 계산을 마친다. **배치 정규화가 z와 a를 계산하는 사이에 이루어진다**. 다음으로 a^[1]을 가지고 z^[2]를 계산한다. 이 과정 역시 w^[2], b^[2]의 영향을 받는다. 첫번째 층에서 했던 것처럼 z^[2]에 대해 BN을 적용한다.

이 과정은 배치 정규화 변수인 β^[2]와 γ^[2]의 영향을 받을 것이다. 그러면 z~^[2]를 얻게 되고 이제 활성화 함수를 적용해서 a^[2]를 얻게 된다. 이 과정이 계속 된다.

마찬가지로 배치 정규화가 z계산과 a계산 사이에 이루어진다. 첫번째 층과 두번째 층에서도 비정규화된 z값 대신 정규화된 z~를 사용한다. 비정규화된 z^[2] 대신 평균과 분산으로 정규화된 z~^[2]를 사용한다.

친필, 라인, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

 즉 신경망에서 변수들이 w^[1], b^[1]부터 w^[l], b^[l]까지 있다. 추가로 β^[1]와 γ^[1], β^[2]와 γ^[2]등이 있을 것이다. (이는 각 층에 배치 정규화를 할 때 사용됨) 여기 β는 모멘텀이나 기하급수적 평균을 구할 때 쓰는 하이퍼파리미터와는 다르다. Adam논문 에서도 하이퍼파라미터를 나타내기 위해 β를 썼고 배치정규화 논문에서도 β를 썼지만 두 β는 완전히 다르다. 배치정규화에서 사용되는 β^[1], β^[2]는 모멘텀이나 단일 RMSprop 알고리즘에 쓰이는 하이퍼파라미터 β와 다르다.

친필, 폰트, 서예, 필기된이(가) 표시된 사진

자동 생성된 설명

 위 변수들이 알고리즘에 쓰인다. **이제 변수를 찾기 위해 경사하강법 등 어떤 최적화를 사용할지 고민할 차례**이다. 예컨대 어떤 층에서 dβ^[l[을 계산했다고 하자. 그리고 β를 β-(학습 속도)\*dβ^[l]로 수정했다. 이런 경우 Adam, RMSprop, 모멘텀 등을 써서 β나 γ를 업데이트 할 수 있다. (경사하강법만 있는 것은 아님)

또 이전 강의에서 배치 정규화를 위해 평균과 분산을 뺴고 나누는 방식을 설명했지만 딥러닝 프로그래밍 프레임워크를 사용하면 배치정규화를 따로 구현할 필요가 없다. 단 한줄의 코드로 구현이 가능하다. 예를 들어 텐서플로 프레임워크에서는 이 함수를 이용해서 배치 정규화를 구현할 수 있다. (물론 프로그래밍 프레임워크에 대해서는 나중에도 다루겠지만 실제로 모든 사항을 하나하나 구현할 필요는 없다. 하지만 코드가 뭘 하는지 알기 위해 이 과정이 어떻게 작동하는지 알아두는 것이 좋다.) 아무튼 **딥러닝 프레임워크에서 배치 정규화는 코드 한줄로 구현이 가능하다**. 지금까지 경사하강법을 이용해서 배치 정규화로 전체를 학습시켰다.

**Working with mini batches**

 하지만 실제로는 **배치 정규화가 훈련 집합의 미니배치에 적용**된다. 즉 실제로는 첫번째 미니배치에 대해서 앞선 설명과 마찬가지로 w^[1]과 b^[1]을 이용해 z^[1]을 계산한다. 그리고 이 미니배치 안에서 z^[1]의 평균과 분산을 계산한 뒤에 평균을 빼고 표준편차로 나눠 배치 정규화를 진행한다. 물론 β^[1]와 γ^[1]을이용해서 값을 조정해야 한다. 이 과정을 통해 첫번째 미니 배치에서 z~^[1]을 얻게 되고 활성화 함수를 적용해 a^[1]을 얻는 것이다. 그리고 z^[2]를 w^[2]와 b^[2]를 이용해 계산해낸다. 이과정을 반복한다.

친필, 폰트, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

이렇게 첫번째 미니배치에 대해 경사하강법을 이용해서 과정을 마쳤으면 두번째 미니배치 X^{2}로 이동한다. 그리고 X^{2}에 대해 비슷한 방법으로 z^[1]을 계산한다. 그리고 배치 정규화를 써서 z~^[1]을 계산한다. 이 정규화에서는 두번째 미니배치에 있는 데이터만을 이용해서 진행한다. 두번째 미니배치만 이용해서 평균과 분산을 계산하는 것이다. 그리고 β와 γ로 보정해서 z~를 얻는다. 이 과정의 반복이다. 마찬가지로 세번째 미니배치에 대해서도 해준다.

친필, 폰트, 화이트보드, 텍스트이(가) 표시된 사진

자동 생성된 설명

 이제 매개변수화 과정에서 한가지 말씀드리고 싶은 것이 있다. 방금 각 층에 대해 w^[l]과 b^[l] 그리고 β^[l]와 γ^[l] 변수가 있다고 했다. 여기서 **z^[l]=w^[l]\*a^[l-1]+b^[l]로 계산**된다. 여기서 배치 정규화는 미니배치를 보고 z^[l]이 평균 0, 분산 1을 갖도록 정규화한 뒤 β와 y를 이용하여 값을 조정해주는 것이다. **여기서 b^[l]이 무엇이든지 간에 없어진다. 왜냐하면 배치정규화의 정규화 과정에서 z의 평균을 계산한 뒤에 빼주기 때문이다.** **미니배치의 모든 예시에 상수를 더해줘도 결국 평균을 빼주면서 사라지기 때문에 아무런 영향을 끼치지 않는다**.

즉 배치 정규화를 쓴다면 이 변수를 없앨 수 있다. 아니면 b^[l]변수를 항상 0으로 둔다고 생각해도 좋다. **z^[l]=w^[l]\*a^[l-1]**이 되는 것이다. **z~=y^[l]\*z^[l]\_norm+β^[l]**여기서 다음 층에 전달되는 z~의 평균을 정하기 위해 β^[l]은 써줘야한다.

다시 말하자면 배치 정규화가 z^[l]의 평균을 0으로 만들기 때문에 b^[l]이라는 변수는 필요가 없는 것이다. **이를 없애는 대신 β^[l]이 그 역할을 차지하는 것**이다. 결과적으로 편향 변수(b)를 결정하기 때문이다.

텍스트, 친필, 폰트, 화이트보드이(가) 표시된 사진

자동 생성된 설명

 끝으로 z^[l]의 차원은 (n^[l], 1)이었다. 즉 b^[l]은 (n^[l], 1) 차원일 것이다. 여기서 n^[l]은 층 l에서의 은닉 유닛 숫자이다. 따라서 β^[l]와 y^[l]의 차원도 (n^[l], 1)이 된다. 왜냐하면 어떤 신경망이든지 간에 n^[l]개의 은닉 유닛을 갖고 있을 때 β^[l]와 y^[l]이 각 은닉 유닛의 값을 조정하는데 쓰이기 때문이다.

**Implementing gradient descent**

 이제 모든 걸 합쳐서**배치 정규화를 사용하여 경사하강법을 구현하는 법**을 살펴보자. 미니배치 경사 하강법을 사용한다고 해보자. t가 1부터 미니배치의 숫자까지 바뀐다고 가정하자.

**미니배치 X^{t}에 대해서 순방향 전파를 사용한다. 각 은닉 층에서 z^[l]을 z~^[l]으로 바꾸기 위해서이다. 이 과정을 거치면 미니배치의 평균과 표준편차를 이용해서 z^[l]을 z~^[l]으로 바꾸어 주는 것**이다. 그리고**역방향 전파를 이용해서 층의 모든 변수에 대해 dw, db, dβ, dy를 계산**한다. (b는 위의 설명에서 없앴으므로 생각하지 않도록 하자) **그리고 각 변수들을 업데이트** 한다. w^[l]:=w-αdw^[l]로 β^[l]:=β-αdβ^[l]로 γ^[l]:=γ-αdγ^[l]로 각각 업데이트 된다.

이렇게 **차이를 계산하면 적은 것처럼 경사하강법을 이용할 수 있다.** 물론 모멘텀, RMSprop, Adam을 사용하는 경사하강법에서도 쓸 수 있다. **세가지의 경사하강법을 이용해서 업데이트 하지 않고 위의 업데이트된 식의 알고리즘을 이용해서 업데이트를 사용할 수도 있는 것**이다. (지지난 포스팅 참고) 이런 다양한 알고리즘을 이용해서 β나 γ같이 배치정규화에 쓰이는 변수를 업데이트 할 수 있다.

친필, 텍스트, 폰트, 서예이(가) 표시된 사진

자동 생성된 설명

**배치 정규화가 잘 작동하는 이유는 무엇일까요?**

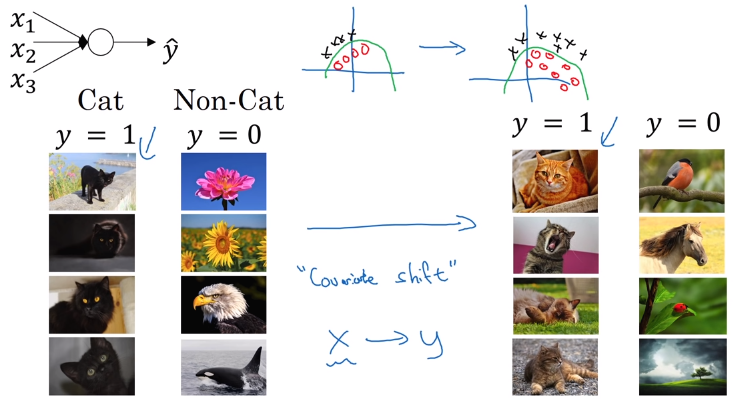
**배치 정규화는 어떻게 작동하는 걸까?**

**첫번째 이유**는 **입력특성 X를 평균 0, 분산 1로 정규화하는 것이 학습속도를 올리는 것**을 보았다. 어떤 특성은 0에서 1까지 어떤 특성은 1에서 1,000까지인 것을 입력특성 X에 대해 비슷한 범위를 갖도록 정규화하여 학습 속도를 높이는 것이다.

그래서 우리는 **배치 정규화가 은닉유닛과 입력층 모두에서 작동하는 이유가 비슷한 일을 하기 때문이라는 직관**을 얻었다. 하지만 이는 부분적인 그림에 불과하다. 이외에도 배치 정규화가 무엇을 하는 것인지 더욱 깊이 이해할 수 있는 직관이 몇 가지 더 있다.

**Leraning or shifting input distribution**

 배치정규화가 작동하는**두번째 이유**는 신경망에서 깊은, 예컨대 **10번 층의 가중치가 1번 층처럼 앞쪽 층의 가중치의 변화에 영향을 덜 받는다는 것**이다. 이를 쉽게 이해할 수 있도록 다음 예시를 보자.

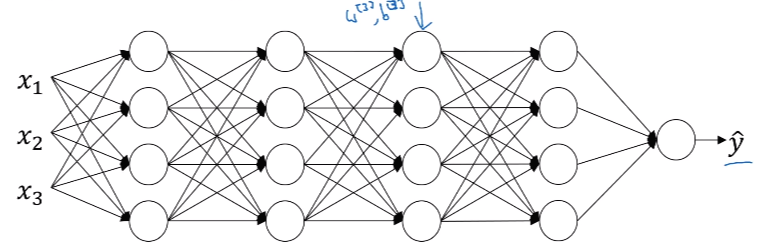


 고양이 분류를 위해 신경망을 학습한다고 하자. 로지스틱 회귀처럼 얕거나 깊을 수도 있다. 그리고 검정 고양이의 이미지만 써서 학습시켰다고 하자.

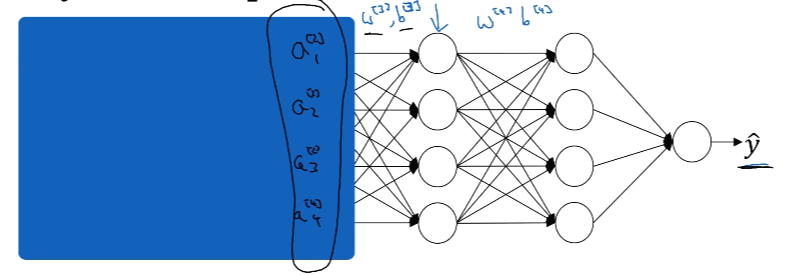
그리고 이 신경망을 다양한 색깔의 고양이에 적용해보자. 왼쪽처럼 검정 고양이가 아니라 오른쪽에 있는 유색 고양이가 정답인 경우 신경망이 좋은 성능을 못 낼 것이다. 아래 이미지의 그래프와 같이 정답과 오답이 다음 파란색 그래프처럼 분포되어있다고 하자. 그리고 이를 일반화 하여 오른쪽 그래프의 정답, 오답 분포에도 적용하는 것을 생각해보면 왼쪽 데이터로 학습시킨 모델이 오른쪽 데이터에서 좋은 성능을 못 낼 것이라고 추측할 수 있다.

사실 두 함수는 이렇게 동일하지만 왼쪽 데이터에만 학습 알고리즘을 적용해서 얻은 초록색 결정기준에 좋은 성능을 기대하기는 어렵다. 이처럼 **데이터 분포가 변화하는 것**을 **"공변량 변화량"**이라고 부른다. 공변량 변화량은**X, Y간의 대응을 학습시킬 때 X의 분포가 바뀐다면 학습 알고리즘을 다시 학습해야한다는 것**이다. 예시의 고양이 구분하기 처럼 X에서 Y로 대응하는 관측함수가 바뀌지 않더라도 말이다. 여러분이 새롭게 함수를 학습시킨 다면 더 정확해지거나 관측함수가 함께 움직여서 더 나빠질 수도 있다.

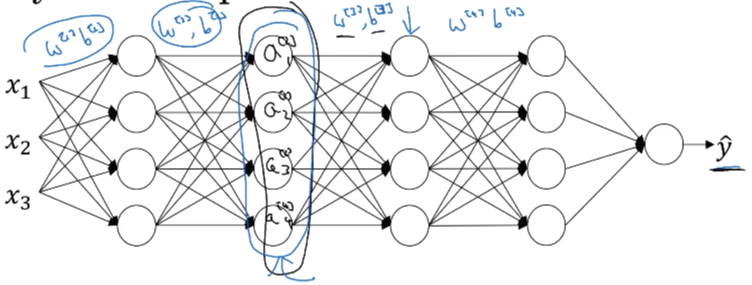
 그럼 공변량 변화가 신경망에 어떻게 적용될 수 있을까? 다음과 같은 심층 신경망을 생각해보자. 그리고 세번째 은닉층의 관점에서 학습과정을 살펴보자. 이 신경망은 w^[3]와 b^[3]을 학습시키고 있다. 세번째 은닉층은 앞선 층에서 값들을 받아온다. 그리고 관측값인 Y와 가까운 y^와 관련된 일을 할 것이다.



 잠시 왼쪽 부분을 가려보겠다. 세번째 층은 값들을 받아온다. 그것들을 a^[2]\_1, a^[2]\_2, a^[2]\_3, a^[2]\_4라고 하자. (특성값인 x\_1, x\_2, x\_3, x\_4가 될 수도 있음) 세번째 은닉층이 할 일은 값을 받아와서 y^으로 대응시키는 것이다. 경사하강법을 써서 w^[3]와 b^[3]이나 w^[4]와 b^[4] 나다가 w^[5]와 b^[5]를 신경망이 좋은 성능을 내도록 학습시키는 것이다. 왼쪽에 가려져 있는 값들로부터 출력값인 y^을 만들어 낸다.



 이제 신경망의 왼쪽을 다시 벗기겠다. **신경망은 매개변수 w^[2], b^[2]와 w^[1], b^[1]도 학습시키고 있다. 이 매개변수의 값이 바뀌면 a^[2] 값들도 바뀐다.** 따라서 **세번째 은닉층의 관점에서 은닉층의 값들이 계속 바뀌고 있다. 따라서 앞서 살펴봤던 공변량 변화의 문제를 계속 겪게 되는 것**이다.



 여기에서 **배치 정규화는 은닉층 값들의 분포가 변화하는 양을 줄여준다**. 만약 이 은닉층들의 값의 분포를 그린다면 여기서는 재정규화된 z인 z^[2]\_1과 z^[2]\_2를 쓴다. 값 4개 대신 2개를 사용해서 2차원에 나타낼 수 있다. 여기서 배치 정규화가 말하고자 하는 것은 z^[1]과 z^[2]가 바뀔 수 있을 것이고 신경망이 앞선 층의 매개변수를 새로 고치면서 아마 바뀔 것이다. 여기서 배치 정규화는 얼마나 바뀌든지 간에 ^[2]\_1과 z^[2]\_2의 평균과 분산이 동일하게 유지될 것이라는 것을 말한다. **즉 ^[2]\_1과 z^[2]\_2의값이 바뀌더라도 적어도 평균과 분산은 0과 1처럼 유지될 것이다. 굳이 0과 1이 아니더라도 β^[2]나 y^[2]와 같은 값들도 가능하다. 신경망이 평균과 분산으로 0과 1을 가질 수도 있지만 충분히 다른 값도 가능하다는 것**이다.

친필, 라인, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

 즉 배치정규화가 하는 일은 앞선 층에서 매개 변수가 바뀌었을 때 세번째 층의 값이 받아들여서 학습하게 될 값의 분포를 제한하는 것이다. 입력값이 바뀌어서 발생하는 문제를 더욱 안정화시켜 뒤쪽 층은 당연히 더 쉽게 학습할 수 있는 것이다. 입력값의 분포가 조금 바뀌더라도 조금만 바뀌는 것이다.**앞쪽 층이 계속 학습하면서 값을 바뀌더라도 뒤쪽 층이 그것 때문에 겪는 부담을 줄인다. 그리고 이것을 또 앞쪽 층의 매개변수와 뒤쪽 층의 매개변수간의 관계를 약화시킨다. 따라서 신경망의 각 층이 다른 층과 상관없이 스스로 배울 수 있게 된다. 이를 통해 전체 신경망의 학습속도를 상승시킨다.**

 반드시 알아두어야 할 것은 **배치정규화에 의해 신경망에서 뒤쪽에 있는 층의 관점에서 앞선층이 너무 많이 변화하지 않는다는 것이다. 평균과 분산이 일정하도록 제한 받았기 때문에 뒤쪽 층의 학습이 더욱 용이해진다.**

**Batch Norm as regularzation**

 배치 정규화의 **두번째 효과**가 있다. 바로 **규제효과**다. 배치정규화에서 다소 비직관적이다.

(1) 각각의 미니배치 X^{t}가 가진 z^[l]에 대해서 그 미니배치의 평균과 분산에 따라 값을 조정할 것이다. 여**기서 미니배치에서 계산한 평균과 분산은 전체 데이터로부터 계산한 것에 비해 다소 잡음을 갖고 있다.** 왜냐하면 64, 128, 256 내지는 더 큰 훈련의 예시를 지닌 미니배치에 대해 상대적으로 작은 데이터에 대해서 추정한 것이기 때문이다.

(2) **여기서 z^[l[에서 z~^[l]으로 조정하는 과정 역시 잡음이 끼어있다. 왜냐하면 잡음이 끼어 있는 평균과 분산으로 계산하기 때문이다.** 즉 드롭아웃처럼 은닉층의 활성함수에 잡음이 끼어있다. 드롭아웃에서는 은닉층을 가져와서 확률에 따라 0을 곱하거나 확률에 따라 1을 곱한다. 따라서 드롭아웃은 곱셈 잡음을 갖고 있다. (0이나 1을 곱하기 때문)

 반면에 **배치 정규화는 표준편차로 나누니 곱셈 잡음도 있고 평균을 빼니 덧셉 잡음도 있다. 여기서 평균과 표준편차의 추정치에는 잡음이 다소 끼어있다. 따라서 드롭아웃처럼 배치 정규화는 약간의 일반화 효과를 가지고 있다.** 왜냐하면 은닉층에 잡음을 추가하는 것은 이후 은닉층이 하나의 은닉층에 너무 의존하지 않도록 한다. 따라서 드롭아웃처럼 은닉층에 잡음을 추가해서 아주 약간의 일반화 효과를 보여준다. 여기서 잡음이 아주 작다보니 일반화 효과가 그리 크지는 않다. 그래서 더욱 강력한 일반화를 원한다면 배치정규화와 드롭아웃을 함께 사용할 수도 있다.

(3) 다른 비관적 효과를 보자. 여러분이**큰 미니배치를 사용한다고 하자**. 64대신 512크기를 사용한다고 하자. 큰 미니배치를 사용하면 잡음이 줄어들고 따라서 일반화 효과도 줄어들 것이다. 드롭아웃의 이상한 특징으로 큰 미니배치를 사용하면 일반화 효과가 줄어드는 것이다. 그런 이유로 강사님은 배치정규화를 일반화 목적으로 사용하지 않는다. 배치 정규화는 그를 목적으로 만들어 진것이 아니다. 하지만 의도하지 않고나, 의도한 것보다 더 큰 효과를 학습 알고리즘에 가져올 수도 있다. **하지만 배치 정규화를 일반화 목적으로 사용하지 않길 바란다. 은닉층의 활성함수를 정규화해서 학습속도를 올리는 용도로 사용**해라. 일반화는 여러분이 의도치 않은 부수 효과에 가깝다.

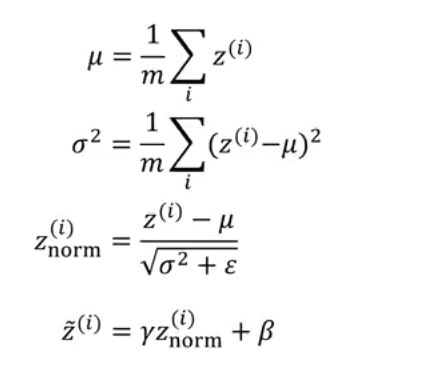
베치정규화에 대한 논의를 맺기 전에 한가지 더 알아두어야 할 것이 있다. **배치 정규화는 한 번에 미니배치 하나의 데이터를 다뤄 미니배치의 평균과 분산을 계산한다.** 테스트 과정에서는 예측을 해서 신경망을 평가한다. 예시의 미니배치 없이 한 번에 하나씩 처리한다. 따라서 테스트 과정에서는 예측이 잘 맞도록 조금 다른 접근을 해야한다. 다음 강의이자 배치 정규화에 대한 마지막 영상에서 예측을 할 수 있는 배치 정규화를 이용해 신경망을 학습시키는 방법을 다루도록 하자.

**테스트시의 배치 정규화**

**배치 정규화는 한번에 하나의 미니배치 데이터를 처리한다. 하지만 테스트에서는 한 번에 샘플을 하나씩 처리**해야 한다. 이를 위해 신경망을 학습시키는 법을 살펴보자.

**Batch Norm at test time**

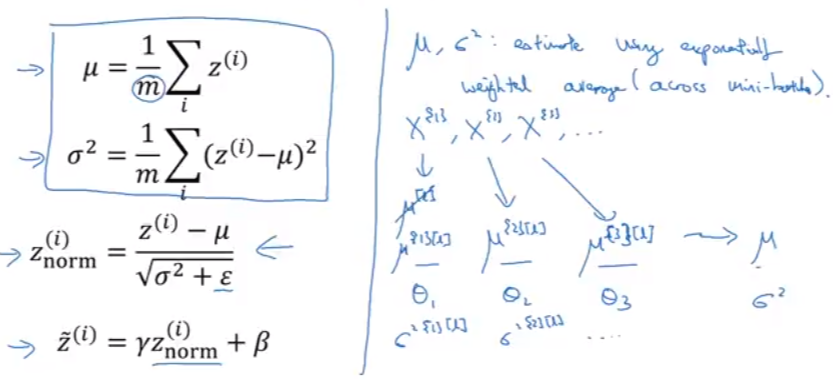
 여기에 나와있는 식들은 학습 중 배치 정규화를 위해 사용했던 식들이다.



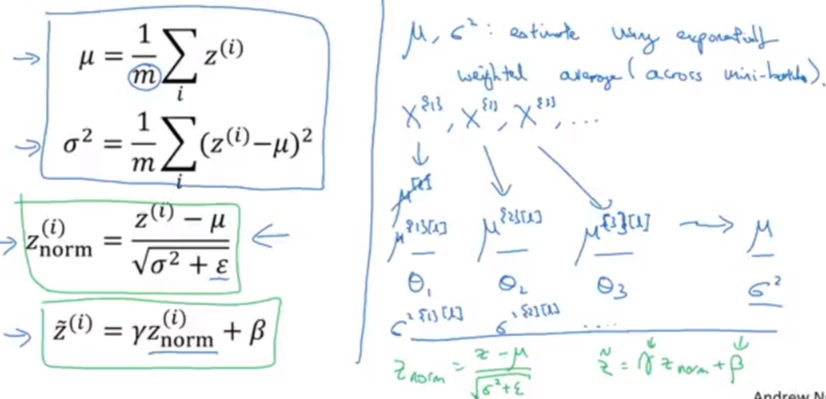
 첫번째 식은 단일 미니배치에서 평균을 구하기 위해 z^(i)의 값을 합했다. 미니배치 안의 샘플들을 모두 합하는 것이다. 여기에서 m은 미니배치 안의 샘플 수이지 전체 훈련 세트에서의 개수가 아니다. 그리고 두번째 식은 분산을 계산한다. 세번째 식에서 평균과 표준편차로 크기를 조정해서 z\_norm도 계산한다. ε은 수학적 안정성을 위해 추가되었다. 네번째 식의 z~는 z\_norm을 y와 β를 써서 조정한 것이다.

 첫번째와 두번째 식을 계산하는데 필요한 μ와 σ^2는 미니 배치 안에서 계산되지만 테스트 과정에서는 64, 128, 256개 등의 샘플을 포함하는 미니배치가 없으므로 동시에 처리할 수 없다. 따라서 μ와 σ^2를 처리할 다른 방법이 필요하다. 만약 하나의 샘플만 있을 때 그 샘프의 평균과 분산을 구하는 것은 말이 안되기 때문이다.

**그렇다면 테스트 과정에서 신경망을 어떻게 적용할 수 있을까? 각각 독립된 μ와 σ^2의 추정치를 사용**하면 된다. **전형적인 배치 정규화를 구현할 때는 여러 미니배치에 걸쳐서 구한 지수 가중 평균을 추정치로 사용**한다. 다시 깔끔하게 설명하자면 어떤 층 L을 고른다. 그리고 미니배치 X^{1}, X^{2} 등과 대응하는 값 γ가 있다고 하자. L층에 대해서 X^{1}을 학습시킨다면 μ^[l]을 얻게 된다. 여기서는 μ^{1}[l], 그 층의 첫번째 미니배치라고 하겠다. 그리고 두번째 미니배치에 대해서도 학습시키면 두번째 μ값을 얻을 수 있다. 그리고 이 은닉층의 세번째 미니배치에 대해서 세번째 μ의 값을 얻을 수도 있다. 우리가 봤듯이 지수가중평균을 이용해서 θ\_1, θ\_2, θ\_3의 평균을 계산한다고 하자. 현재 온도의 기하급수적 평균을 계산했을 때 이 평균 벡터에서 가장 최근의 평균값이 무엇이였는지 기록해야했다. 그러면 지수가중평균이 그 은닉층의 z값 평균의 추정치가 되는 것이다. 비슷하게 지수가중평균을 이용해서 σ^2의 값을 추적할 수도 있다. 그 층의 첫번째 미니 배치에서 σ^2를 구하고 두번째 미니배치에서 반복한다. 이렇게 신경망에서 서로 다른 미니배치로 학습시킨 각 층에 대해서 μ와 σ^2의 이동 평균을 구할 수 있다.



 그러면 테스트 과정에서 여러분은 갖고 있는 z값과 μ와 σ^2의 지수가중평균을 이용해서 z\_norm을 계산만 하면 된다. 가장 최근의 값이 뭐였든 간에 네번째 식에서 값을 조정하는데 쓰인다. 그리고 테스트 샘플에 대해서 방금 계산한 z\_norm과 신경망 학습 과정에서 학습시킨 β, γ매개변수를 이용해 z~를 계산할 수도 있다.



 여기서 알아두어야 할 것은 **학습과정 중에 μ와 σ^2는 64나 256등의 미니배치로 계산**하지만 **테스트할 때는 한 번에 샘플 하나를 처리해**야 한다. μ와 σ^2를 훈련세트로부터 추정해야 한다. 이에는 여러가지 방법이 있다.

 이론적으로는 훈련세트를 이용해 최종적으로 학습한 신경망의 μ와 σ^2를 얻을 수도 있지만 실제로는 μ와 σ^2가 학습하면서 가진 값을 추적하면서 지수가중평균을 사용한다. 지수가중평균을 이용하는 것을 달리 이동 평균이라고도 한다. μ와 σ^2의 추정치를 대충 결정한 뒤 테스트 과정에서 사용하는 것이다. 은닉층의 z값을 조정할 때 필요하기 때문이다. 실제로도 이 방법은 μ와 σ^2를 추정하는데 꽤나 안정적이다.

 만약 딥러닝 프레임워크를 사용한다면 기본적으로 μ와 σ^2를 추정할 수 있는 방법이 있다. 실제로는 은닉층의 값 z에 대해 평균과 분산을 추정하는 것은 테스트 과정에서도 합리적으로 잘 동작한다.

 이렇게 배치 정규화에 대한 내용이 끝났다. 이를 이용하면 더 깊은 신경망을 학습시키고 학습 알고리즘을 더 빨리 실행시킬 수 있을 것이다.