

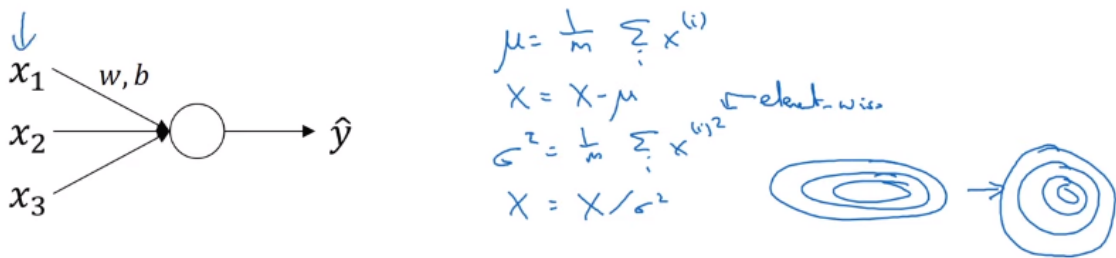
6. 배치 정규화

1. 배치 정규화

: 하이퍼파라미터 탐색을 쉽게 만들어줌 + 신경망과 하이퍼파라미터의 상관관계를 줄여
→ 더 많은 하이퍼파라미터가 잘 작동하도록

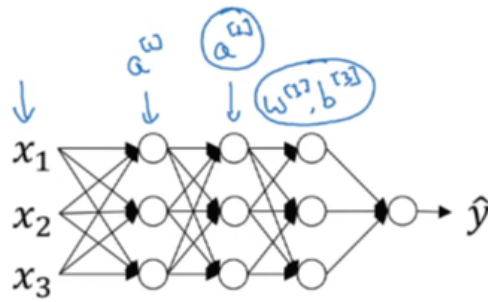
1. Normalizing inputs to speed up learning

- 로지스틱 회귀 등으로 모델을 학습시킬 때 **입력 변수들을 정규화**하면 학습이 빨라짐
 - 평균을 계산할 때: 입력변수의 평균 빼기
 - 분산을 계산할 때: $x(i)^2$ 를 사용



→ 로지스틱 회귀 등 신경망의 입력 변수들을 정규화하면 누워있는 학습 등고선이 더 둥근 형태로 변화

- 심층 신경
 - 입력변수 x 뿐 아니라 은닉층의 활성화값 $a^{[1]}$, $a^{[2]}$ 등 존재
 - 은닉층에 대해 $w^{[3]}$, $b^{[3]}$ 를 빠르게 학습시킬 수 있도록 $a^{[2]}$ 같은 값을 정규화시킬 수 있을까?
- 다음 층 입력값인 $a^{[2]}$ 가 $w^{[3]}$, $b^{[3]}$ 의 학습에 영향을 주기 때문
 - $a^{[2]}$ 가 아니라 $z^{[2]}$ 를 정규화하는 것**



Can we normalize $a^{[2]}$ so as to turn $w^{[2]}, b^{[2]}$ faster
 Normalize $z^{[2]}$

2. Implementing Batch Norm

- 신경망에서 사이값들이 주어졌다고 할 때 은닉 유닛의 값 $z^{(1)}$ 부터 $z^{(m)}$ 까지 있다고 가정
- $z^{[l]}(i)$ 로 표기 (i 는 1부터 m 까지, 간편하게 쓰기 위해 $[l]$ 은 생략)
- 분산 계산
- 각 $z^{(i)}$ 에 대해서 정규화를 하여 $z^{(i)}_{\text{norm}}$ 을 얻음
- 수학적 안정성을 위해서 분모에 ϵ 를 추가(표준편차가 0인 경우를 대비)

→ z 값에 대해서 정규화를 거쳐 모든 z 들이 **평균이 0이고 표준편차가 1**이 되도록 만들기

- 하지만 은닉 유닛이 항상 평균 0, 표준편차 1을 갖는 것이 좋지만은 않다.

→ 은닉 유닛은 다양한 분포를 가져야하기 때문

- $\gamma * z^{(i)}_{\text{norm}} + \beta$ 이용 (γ 와 β 는 모델에서 학습시킬 수 있는 변수)

→ z 의 평균을 원하는 대로 설정 가능

Given some intermediate values in NN $z^{(1)}, \dots, z^{(m)}$

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

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

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

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

learnable parameters of model.

If $\gamma = \sqrt{\sigma^2 + \epsilon}$ $\leftarrow z^{[l]}(i)$
 $\beta = \mu$ \leftarrow
 then $\sum_i \tilde{z}^{(i)} = z^{(i)}$

→ 적절히 γ 와 β 를 설정해서 왼쪽의 네개의 식으로 이루어진 정규화 과정은 항등함수를 만드는 것과 똑같은 효과

- 위 과정을 신경망에 적용
 - $z^{(i)}$ 를 $\hat{z}^{(i)}$ 대신 신경망에 사용하는 것

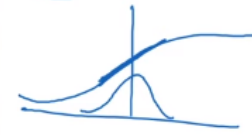
Given some intermediate values in NN $z^{(1)}, \dots, z^{(n)}$

$$\begin{aligned} \mu &= \frac{1}{n} \sum_i z^{(i)} \\ \sigma^2 &= \frac{1}{n} \sum_i (z^{(i)} - \mu)^2 \\ z_{\text{norm}}^{(i)} &= \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ \hat{z}^{(i)} &= \gamma z_{\text{norm}}^{(i)} + \beta \end{aligned}$$

learnable parameters of model.

If $\gamma = \sqrt{\sigma^2 + \epsilon}$ \leftarrow $z^{(i)}$
 $\beta = \mu$ \leftarrow $z^{(i)}$
 then $\hat{z}^{(i)} = z^{(i)}$

Use $\hat{z}^{(i)}$ instad of $z^{(i)}$.

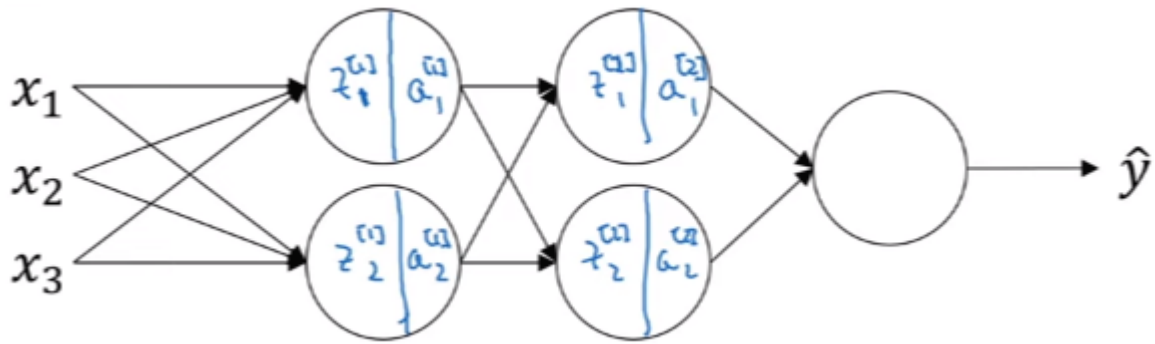


→ 배치 정규화는 입력층에만 정규화를 하는 것이 아니라 신경망 안 깊이 있는 은닉층의 값들까지도 정규화하는 것

- 입력층과 은닉 유닛을 학습시킬 때 차이점: 은닉 유닛 값의 군과 분산이 0, 1로 고정되기를 원치 않는다.
 - ex) 시그모이드 활성화함수가 있을 때 값들이 한 구간에 모여있는 것을 원치 않는다. (더 넓은 영역에 걸쳐 퍼져있거나 시그모이드의 비선형성을 살릴 수 있도록 평균이 0이 아닌 다른 값을 갖게 하는 것이 좋음.)

2. 배치 정규화 적응시키기

1. Adding Batch Norm to a network



• 은닉 유닛을 두가지로 나눌 수 있음.

- z 를 먼저 계산하고 활성화 함수를 이용해서 a 를 계산 (즉, 원 하나하나가 두 단계의 계산을 나타냄).
- 다음 층에서도 비슷하게 $z^{[2]}_1$ 와 $a^{[2]}_1$ 이 .

i) 배치 정규화 사용 X

- 입력값 x 가 첫번째 은닉층에 주어짐.
- 그러면 $w^{[l]}$ 과 $b^{[l]}$ 에 따라 $z^{[l]}$ 을 계산.
- 활성화 함수 $z^{[l]}$ 이 주어져서 $a^{[l]}$ 을 계산.

ii) 배치 정규화 사용 O

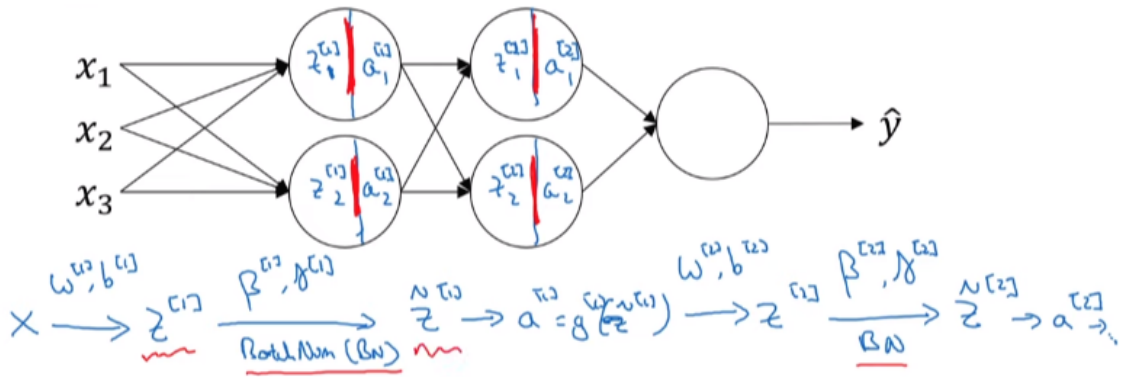
- $z^{[1]}$ 을 받아서 배치정규화(BN)를 시킨다. (BN은 $\beta^{[1]}$ 와 $\gamma^{[1]}$ 의 영향을 받음)
- 새로 정규화된 $z^{[1]}$ 값들을 활성화 함수를 거쳐 활성화 값 $a^{[1]}$ 을 얻는다.

→ 배치 정규화가 z 와 a 를 계산하는 사이에 이루어진다.

- $a^{[1]}$ 을 가지고 $z^{[2]}$ 를 계산. (이 과정 역시 $w^{[2]}$, $b^{[2]}$ 의 영향을 받음)
- 첫번째 층에서 했던 것처럼 $z^{[2]}$ 에 대해 BN을 적용. (배치 정규화 변수인 $\beta^{[2]}$ 와 $\gamma^{[2]}$ 의 영향을 받을 것)

→ $z^{[2]}$ 를 얻게 되고, 활성화 함수를 적용해서 $a^{[2]}$ 를 얻게 됨.

- 이 과정 반복. (마찬가지로 배치 정규화가 z 계산과 a 계산 사이에 이루어진다.)
- 첫 번째 층과 두 번째 층에서도 비정규화된 z 값 대신 정규화된 z 를 사용.



- 신경망에서 변수들이 $w^{[1]}, b^{[1]}$ 부터 $w^{[L]}, b^{[L]}$ 까지 존재. (추가로 $\beta^{[1]}$ 와 $\gamma^{[1]}$, $\beta^{[2]}$ 와 $\gamma^{[2]}$ 등이 있을 것 \rightarrow 각 층에 배치 정규화를 할 때 사용)
- 배치정규화에서 사용되는 $\beta^{[1]}, \beta^{[2]}$ 는 모멘텀이나 단일 RMSprop 알고리즘에 쓰이는 하이퍼파라미터 β 와 다

2. 변수를 찾기 위해 경사하강법 등 어떤 최적화를 사용할지 고민 필요

ex) 어떤 층에서 $d\beta^{[l]}$ 을 계산했다고 하자. $+\beta$ 를 $\beta - (\text{학습 속도}) \cdot d\beta^{[l]}$ 로 수정.

- 이런 경우 Adam, RMSprop, 모멘텀 등을 써서 β 나 γ 를 업데이트 가능.
- 딥러닝 프로그래밍 프레임워크를 사용하면 배치정규화를 따로 구현할 필요가 없다. (단 한줄의 코드로 구현이 가능.)

3. Working with mini batches

- 실제로는 배치 정규화가 훈련 집합의 미니배치에 적용
 - 첫번째 미니배치에 대해서 앞선 설명과 마찬가지로 $w^{[1]}$ 과 $b^{[1]}$ 을 이용해 $z^{[1]}$ 을 계산
 - 미니배치 안에서 $z^{[1]}$ 의 평균과 분산을 계산한 뒤에 평균을 빼고 표준편차로 나눠 배치 정규화를 진행
 - $\beta^{[1]}$ 와 $\gamma^{[1]}$ 을이용해서 값 조정 필요
 - 첫 번째 미니 배치에서 $z^{[1]}$ 을 얻게 되고 활성화 함수를 적용해 $a^{[1]}$ 을 얻는 것. 그리고 $z^{[2]}$ 를 $w^{[2]}$ 와 $b^{[2]}$ 를 이용해 계산.

$$x^{(1)} \xrightarrow{w^{(1,1)}, b^{(1,1)}} z^{(1)} \xrightarrow{\frac{\gamma^{(1,1)}}{\sigma^{(1,1)}}} \tilde{z}^{(1)} \rightarrow g^{(1)}(\tilde{z}^{(1)}) = a^{(1)} \xrightarrow{w^{(1,2)}, b^{(1,2)}} z^{(2)} \rightarrow \dots$$

- 첫번째 미니배치에 대해 경사하강법을 이용해서 과정을 마쳤으면, 두번째 미니배치 $x^{[2]}$ 로 이동.
- $x^{[2]}$ 에 대해 비슷한 방법으로 $z^{[1]}$ 을 계산.
- 배치 정규화를 써서 $z^{[1]}$ 을 계산. (이 정규화에서는 두번째 미니배치에 있는 데이터만을 이용해서 진행)
- β 와 γ 로 보정해서 z 를 얻는다.
- 이 과정 반복

- 각 층에 대해 $w^{[l]}$ 과 $b^{[l]}$ 그리고 $\beta^{[l]}$ 와 $\gamma^{[l]}$ 변수가 존재.

→ 여기서 $z^{[l]} = w^{[l]} * a^{[l-1]} + b^{[l]}$ 로 계산

- 배치 정규화는 미니배치를 보고 $z^{[l]}$ 이 평균 0, 분산 1을 갖도록 정규화한 뒤 β 와 γ 를 이용하여 값을 조정해주는 것
- 여기서 $b^{[l]}$ 이 무엇이든지 간에 사라짐.

→ 배치정규화의 정규화 과정에서 z 의 평균을 계산한 뒤에 빼주기 때문

- 미니배치의 모든 예시에 상수를 더해줘도 결국 평균을 빼주면서 사라지기 때문에 아무런 영향을 끼치지 않는다.

→ 배치 정규화를 쓴다면 이 변수를 없앨 수 있다.

→ $z^{[l]} = w^{[l]} * a^{[l-1]}$ 이 되는 것

- $z = \gamma^{[l]} * z^{[l]}_{norm} + \beta^{[l]}$ (여기서 다음 층에 전달되는 z 의 평균을 정하기 위해 $\beta^{[l]}$ 은 써줘야 함)

→ 배치 정규화가 $z^{[l]}$ 의 평균을 0으로 만들기 때문에 $b^{[l]}$ 이라는 변수는 필요가 없는 것

($\beta^{[l]}$ 이 그 역할을 차지하는 것)

Parameters: $w^{[l]}$, $b^{[l]}$, $\beta^{[l]}$, $\gamma^{[l]}$

$z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$

$z^{[l]}_{norm} = \frac{z^{[l]} - \beta^{[l]}}{\sqrt{\gamma^{[l]}}}$

$z^{[l]} = \gamma^{[l]} z^{[l]}_{norm} + \beta^{[l]}$

Andrew Ng

- $z^{[l]}$ 의 차원: $(n^{[l]}, 1)$

- $b^{[l]}$ 차원: $(n^{[l]}, 1)$ (여기서 $n^{[l]}$ 은 층 l에서의 은닉 유닛 숫자)

- $\beta^{[l]}$ 와 $y^{[l]}$ 의 차원: $(n^{[l]}, 1)$

→ 왜냐하면 어떤 신경망이든지 간에 $n^{[l]}$ 개의 은닉 유닛을 갖고 있을 때 $\beta^{[l]}$ 와 $y^{[l]}$ 이 각 은닉 유닛의 값을 조정하는데 쓰이기 때문

4. Implementing gradient descent

- 미니배치 경사 하강법을 사용한다고 가정.
 - t가 1부터 미니배치의 숫자까지 바뀐다고 하면, 미니배치 $X^{(t)}$ 에 대해서 **순방향 전파**를 사용.
- 각 은닉 층에서 $z^{[l]}$ 을 $\tilde{z}^{[l]}$ 으로 바꾸기 위함.
 - 미니배치의 평균과 표준편차를 이용해서 $z^{[l]}$ 을 $\tilde{z}^{[l]}$ 으로 바꾸어 주는 것.
 - **역방향 전파**를 이용해서 층의 모든 변수에 대해 dw , db , $d\beta$, dy 를 계산. (b는 위의 설명에서 없었으므로 생각하지 않도록 하자)
 - **각 변수들을 업데이트** 한다.
 - $w^{[l]} := w - \alpha dw^{[l]}$
 - $\beta^{[l]} := \beta - \alpha d\beta^{[l]}$
 - $y^{[l]} := y - \alpha dy^{[l]}$
 - 차이를 계산하면 경사하강법을 이용할 수 있다. (물론 모멘텀, RMSprop, Adam을 사용하는 경사하강법에서도 사용 가능)
 - 세 가지의 경사하강법을 이용해서 업데이트 하지 않고 위의 업데이트된 식의 알고리즘을 이용해서 업데이트를 사용할 수도 있는 것

for $t = 1 \dots \text{num Mini Batches}$
 Compute forward pass on $X^{(t)}$.
 In each hidden layer, use BN to replace $z^{[l]}$ with $\tilde{z}^{[l]}$.
 Use backprop to compute $\frac{dw^{[l]}}{dz^{[l]}}$, $\frac{d\beta^{[l]}}{dz^{[l]}}$, $\frac{dy^{[l]}}{dz^{[l]}}$
 Update params $\left. \begin{aligned} w^{[l]} &:= w^{[l]} - \alpha \frac{dw^{[l]}}{dz^{[l]}} \\ \beta^{[l]} &:= \beta^{[l]} - \alpha \frac{d\beta^{[l]}}{dz^{[l]}} \\ y^{[l]} &:= y^{[l]} - \alpha \frac{dy^{[l]}}{dz^{[l]}} \end{aligned} \right\} \leftarrow$
 Works w/ momentum, RMSprop, Adam.

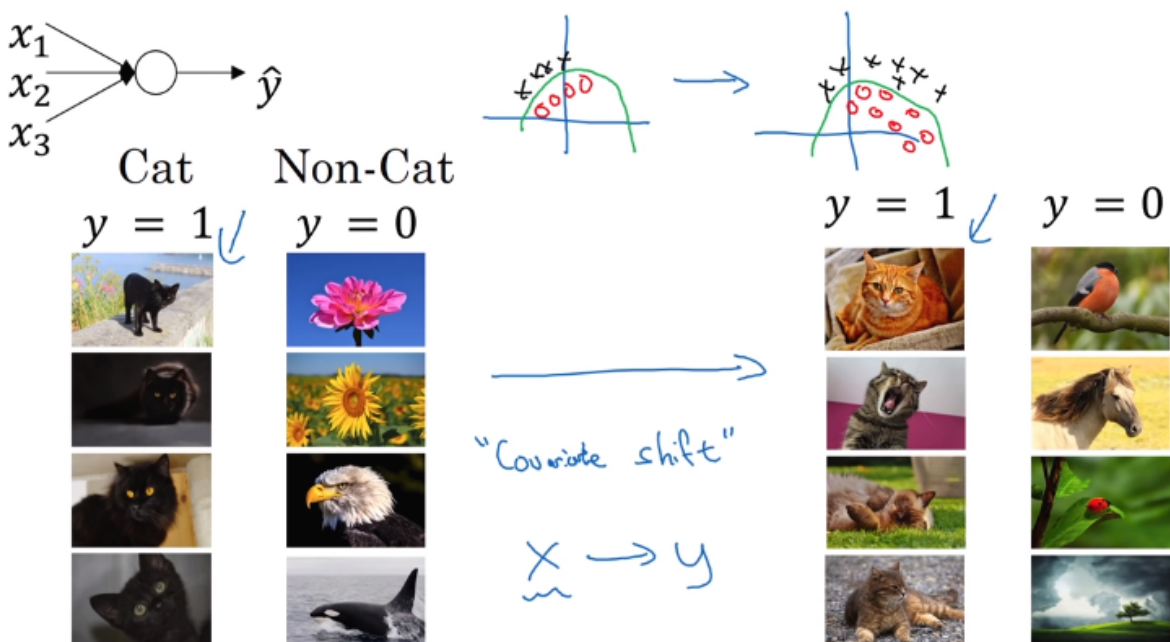
3. 배치 정규화가 잘 작동하는 이유는 무엇일까요?

1. 입력특성 X 를 평균 0, 분산 1로 정규화 하는 것이 학습 속도를 올리는 것

- 배치 정규화가 은닉유닛과 입력층 모두에서 작동하는 이유: 비슷한 일을 하기 때문

2. Learning or shifting input distribution

- 신경망에서 깊은 10번 층의 가중치가 1번 층처럼 앞쪽 층의 가중치의 변화에 영향을 덜 받는다는 것
- 예시: 고양이 분류를 위 신경망 학습

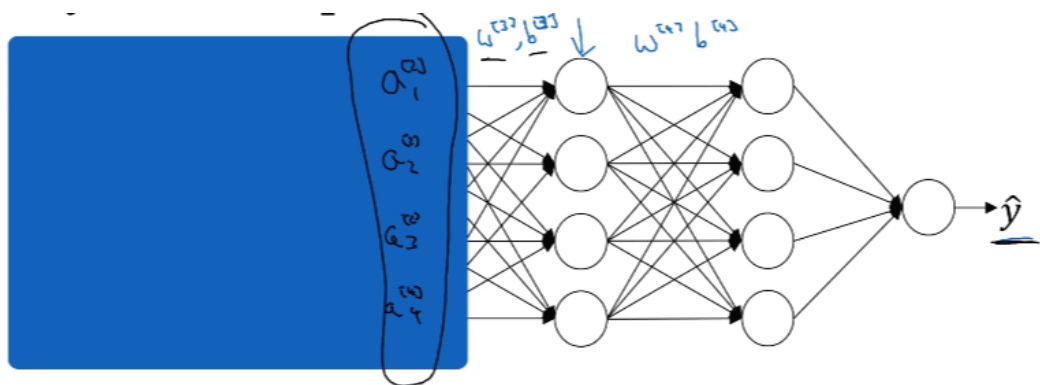


→ 검정 고양이의 이미지만 써서 학습시켰다고 가정

- 신경망을 다양한 색깔의 고양이에 적용해보면, 유색 고양이가 정답인 경우 신경망이 좋은 성능을 못 낼 것
- 데이터 분포가 변화하는 것: 공변량 변화량
 - X, Y 간의 대응을 학습시킬 때 X 의 분포가 바뀐다면 학습 알고리즘을 다시 학습해야 한다는 것
 - 공변량 변화량이 신경망에 어떻게 적용?

ex) 세 번째 은닉층 학습시키고 있는 경우

- $w^{[3]}$ 와 $b^{[3]}$ 을 학습시키는 중
- 세번째 은닉층은 앞선 층에서 값들을 받아옴.
 - 받아오는 값: $a^{[2]}_1, a^{[2]}_2, a^{[2]}_3, a^{[2]}_4$ (특성값인 x_1, x_2, x_3, x_4 가 될 수도 있음)
- 세번째 은닉층이 할 일: 값을 받아와서 y^{\wedge} 으로 대응시키는 것
 - 경사하강법을 사용하여, $w^{[3]}$ 와 $b^{[3]}$ 이나 $w^{[4]}$ 와 $b^{[4]}$ 나 $w^{[5]}$ 와 $b^{[5]}$ 를 신경망이 좋은 성능을 내도록 학습시키는 것



- 신경망은 매개변수 $w^{[2]}, b^{[2]}$ 와 $w^{[1]}, b^{[1]}$ 도 학습시키고 있다. 이 매개변수의 값이 바뀌면 $a^{[2]}$ 값들도 변화.
 - 따라서 세번째 은닉층의 관점에서 은닉층의 값들이 계속 바뀌고 있는 것으로, 앞서 살펴봤던 공변량 변화의 문제를 계속 겪게 되는 것
- 배치 정규화는 은닉층 값들의 분포가 변화하는 양을 줄여준다.
 - 여기에서는 재정규화된 z 인 $z^{[2]}_1$ 과 $z^{[2]}_2$ 를 사용
 - 배치 정규화는 얼마나 바뀌든지 간에 $z^{[2]}_1$ 과 $z^{[2]}_2$ 의 평균과 분산이 동일하게 유지될 것이라는 것을 의미
 - $z^{[2]}_1$ 과 $z^{[2]}_2$ 의 값이 바뀌더라도 적어도 평균과 분산은 0과 1처럼 유지될 것. 굳이 0과 1이 아니더라도 $\beta^{[2]}$ 나 $y^{[2]}$ 와 같은 값들도 가능. 신경망이 평균과 분산으로 0과 1을 가질 수도 있지만 충분히 다른 값도 가능하다는 것
- 배치정규화가 하는 일: 앞선 층에서 매개 변수가 바뀌었을 때 세번째 층의 값이 받아 들어서 학습하게 될 값의 분포를 제한하는 것. 입력값이 바뀌어서 발생하는 문제를

더욱 안정화시켜 뒤쪽 층은 당연히 더 쉽게 학습할 수 있는 것을 뜻한다.

- 앞쪽 층이 계속 학습하면서 값을 바꿔더라도 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.

3. Batch Norm as regularization

- 배치 정규화의 두번째 효과: **규제 효과**

(1) 각각의 미니배치 $X^{(t)}$ 가 가진 $z^{[l]}$ 에 대해서 그 미니배치의 평균과 분산에 따라 값을 조정할 것

- 미니배치에서 계산한 평균과 분산은 전체 데이터로부터 계산한 것에 비해 다소 잡음을 갖고 있다.

→ 64, 128, 256 내지는 더 큰 훈련의 예시를 지닌 미니배치에 대해 상대적으로 작은 데이터에 대해서 추정한 것이기 때문

(2) $z^{[l]}$ 에서 $\tilde{z}^{[l]}$ 으로 조정하는 과정에도 잡음이 끼어있다.

- 잡음이 끼어 있는 평균과 분산으로 계산하기 때문이다. (드롭아웃처럼 은닉층의 활성화수에 잡음이 끼어있다.)

◦ 드롭아웃에서는 은닉층을 가져와서 확률에 따라 0을 곱하거나 확률에 따라 1을 곱한다.

→ 따라서 드롭아웃은 곱셈 잡음을 갖고 있다. (0이나 1을 곱하기 때문)

- 배치 정규화는 **표준편차로 나누기 때문에 곱셈 잡음도 있고 평균을 빼기 때문에 덧셈 잡음도 있다.** 여기서 평균과 표준편차의 추정치에는 잡음이 다소 끼어있다. 따라서 드롭아웃처럼 배치 정규화는 약간의 일반화 효과를 가지고 있다.

→ 은닉층에 잡음을 추가하는 것은 이후 은닉층이 하나의 은닉층에 너무 의존하지 않도록 한다. 따라서 드롭아웃처럼 은닉층에 잡음을 추가해서 아주 약간의 일반화 효과를 보여준다. 하지만, 잡음이 아주 작다보니 일반화 효과가 그렇게 크지는 않다.

→ 더욱 강력한 일반화를 원한다면 **배치정규화와 드롭아웃을 함께** 사용할 수도 있다.

(3) 큰 미니배치를 사용한다고 가정.

ex) 64대신 512크기를 사용할 때, **큰 미니배치를 사용**하면 잡음이 줄어들고 따라서 **일반화 효과도 줄어들** 것이다. 드롭아웃의 이상한 특징으로 큰 미니배치를 사용하면 일반화 효과가 줄어드는 것이다. → 배치정규화를 일반화 목적으로 사용X

- 배치 정규화는 일반화를 목적으로 만들어진 것이 아니다. 하지만 의도한 것보다 더 큰 효과를 학습 알고리즘에 가져올 수도 있다. **은닉층의 활성화함수를 정규화해서 학습속도를 올리는 용도로 사용하는 것이 좋음.**

4. 테스트시의 배치 정규화

- 배치 정규화는 한번에 하나의 미니배치 데이터를 처리. 하지만 테스트에서는 **한 번에 샘플을 하나씩 처리**해야 함.

1. Batch Norm at test time

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

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

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

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

- 첫번째 식: 단일 미니배치에서 **평균을 구하기 위해** $z^{(i)}$ 의 값을 합했다. 미니배치 안의 샘플들을 모두 합하는 것. 여기에서 **m은 미니배치 안의 샘플 수**이지 전체 훈련 세트에서의 개수가 아니다.
- 두번째 식: **분산을 계산.**

- 세번째 식: 평균과 표준편차로 크기를 조정해서 z_norm 도 계산. ϵ 은 수학적 안정성을 위해 추가됨.
- 네번째 식: $z\sim$ 는 z_norm 을 y 와 β 를 써서 조정한 것
- 첫번째와 두번째 식을 계산하는데 필요한 μ 와 σ^2 는 미니 배치 안에서 계산되지만 테스트 과정에서는 64, 128, 256개 등의 샘플을 포함하는 미니배치가 없으므로 동시에 처리 불가.

→ μ 와 σ^2 를 처리할 다른 방법이 필요

2. 테스트 과정에서 신경망을 어떻게 적용할 수 있을까?

- 각각 독립된 μ 와 σ^2 의 추정치를 사용하면 됨.
 - 전형적인 배치 정규화를 구현할 때는 여러 미니배치에 걸쳐서 구한 지수 가중 평균을 추정치로 사용
 - 어떤 층 L 을 선택하고, 미니배치 $X^{\{1\}}$, $X^{\{2\}}$ 등과 대응하는 값 y 가 있다고 하자.
 - L 층에 대해서 $X^{\{1\}}$ 을 학습시킨다면 $\mu^{\{l\}}$ 을 얻게 된다.
- $\mu^{\{1\}}[i]$, 그 층의 첫번째 미니배치라고 하겠다.
- 두 번째 미니배치에 대해서도 학습시키면 두 번째 μ 값을 얻을 수 있다.
 - 이 은닉층의 세번째 미니배치에 대해서 세번째 μ 의 값을 얻을 수도 있다.
 - 지수가중평균을 이용해서 $\theta_1, \theta_2, \theta_3$ 의 평균을 계산한다고 하자. 현재 온도의 기하급수적 평균을 계산했을 때 이 평균 벡터에서 가장 최근의 평균값이 무엇이였는지 기록해야했다. 그러면 지수가중평균이 그 은닉층의 z 값 평균의 추정치가 되는 것이다. 비슷하게 지수가중평균을 이용해서 σ^2 의 값을 추적할 수도 있다. 그 층의 첫번째 미니 배치에서 σ^2 를 구하고 두번째 미니배치에서 반복한다. 이렇게 **신경망에서 서로 다른 미니배치로 학습시킨 각 층에 대해서 μ 와 σ^2 의 이동 평균을 구할 수 있다.**

→ $\mu = \frac{1}{m} \sum_i z^{(i)}$

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

→ $z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$ ←

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

μ, σ^2 : estimate using exponentially weighted average (across mini-batches).

$\chi^{(1)}, \chi^{(2)}, \chi^{(3)}, \dots$

↓ ↓ ↓

$\mu^{(1)}[1]$ $\mu^{(2)}[1]$ $\mu^{(3)}[1]$ → μ

θ_1 θ_2 θ_3

$\sigma^{(1)}[1]$ $\sigma^{(2)}[1]$...

σ^2

- 테스트 과정에서 가지고 있는 z 값과 μ 와 σ^2 의 지수가중평균을 이용해서 z_{norm} 을 계산만 하면 된다.
- 학습과정 중에 μ 와 σ^2 는 64나 256등의 미니배치로 계산하지만 테스트할 때는 한 번에 샘플 하나를 처리해야 한다