DL 스터디&프로젝트

[Euron 중급 세션 12주차] 딥러닝 2단계 6. 배치 정규화

by 공부하자_ 2023. 11. 27.

딥러닝 2단계: 심층 신경망 성능 향상시키기

6. 배치 정규화

📌배치 정규화(C2W3L04)

핵심어: 배치 정규화(Batch Normalization)

딥러닝이 떠오르면서 가장 중요한 아이디어 중 하나로 배치 정규화라는 알고리즘이 꼽힌다. 배치 정규화는 하이퍼파라미터 탐색을 쉽게 만들어줄 뿐만 아니라 신경망과 하이퍼파라미터의 상관관계를 줄여준다. 즉 더 많은 하이퍼파라미터가 잘 작동한다는 것이며, 이는 아주 깊은 심층신경망이라도 아주 쉽게 학습할 수 있도록 도와준다.

분류 전체보기

DL 스터디&프로젝트

Data Science 프로젝트

Github 스터디

Data Science 개인 공부

Backend 프로젝트

기타 공부

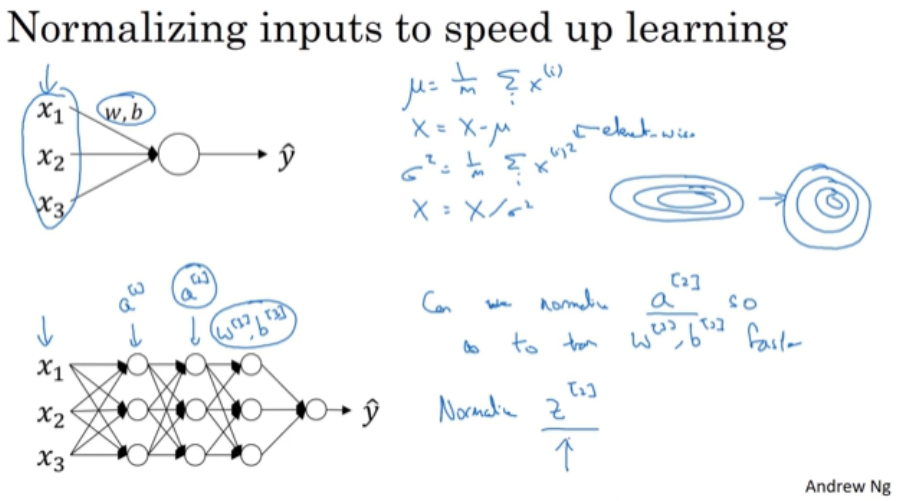
공지사항

최근글 인기글

[Euron 중급 세션 12주차]… 2023.11.27

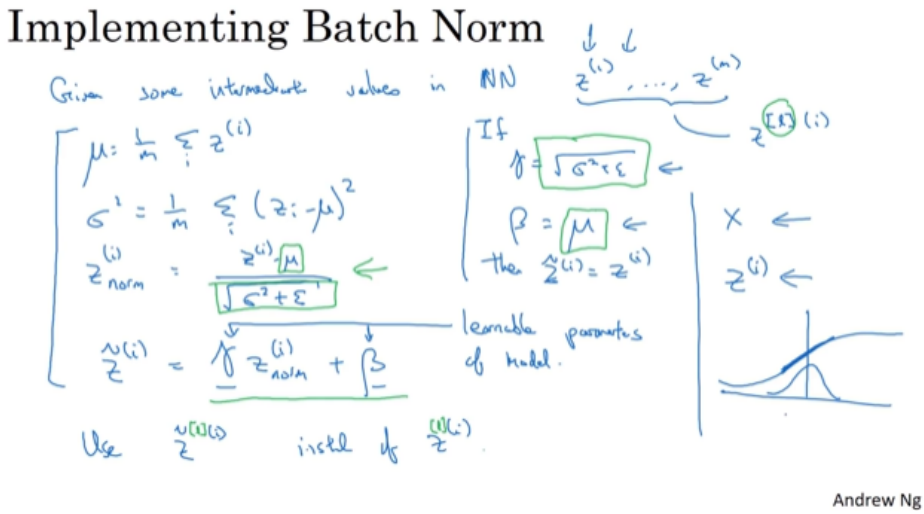
[Euron 중급 세션 11주차]… 2023.11.20

[Euron 중급 세션 10주차] 캐글 필사 과제…

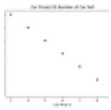


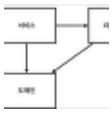
배치 정규화가 어떻게 작동하는지 알아보도록 하자. 로지스틱 회귀 등으로 모델을 학습시킬 때, 입력 변수들을 정규화하면 학습이 빨라졌었다. 평균을 계산할 때는 입력 변수의 평균을 뺐고, 분산을 계산할 때는 x(i)^2를 사용하여 요소별로 하나씩 제곱해주었다. 이 값을 이용하여 정규화해주고, 이 방법이 어떻게 누워있는 학습 등고선을 경사 하강법에 적합하도록 더 둥근 형태로 바꾸는지 학습했었다. 즉 로지스틱 회귀 등 신경망의 입력 변수들을 정규화하면 이렇게 바뀌는 것이다.

심층 신경망의 경우, 여기서는 입력 변수 x뿐 아니라 활성값 a[l](l은 층)이 있으며 또한 만약 w[3], b[3] 파라미터를 학습시킨다면 a[2]의 평균과 분산을 정규화하는 것이 더 효율적이라고 생각할 수 있다. 로지스틱 회귀의 경우 x1, x2, x3를 저규화하는 것이 w, b 학습을 효율적으로 하는지 봤었다. 여기서 짚어봐야 할 부분은 은닉층에 대해, w[3]나 b[3]를 빠르게 학습시킬 수 있도록 a[2]같은 값을 정규화할 수 있냐는 것이다(다음 층의 입력값인 a[2]가 w[3]와 b[3] 학습에 영향을 주기 때문). 이 질문이 배치 정규화하가 하는 일을 나타내고 있다. 사실 a[2]가 아니라, 활성 함수 이전의 값 z[2]를 정규화하는 것이다.



배치 정규화 구현이 어떻게 이루어지는지 살펴보자. 신경망에서 사잇값들이 주어졌다고 할 때, 은닉 유닛의 값 z(1)부터 z(m)까지 있다고 하자. 이 값들은 은닉 유닛의 값들이다([l]은 생략). 이 값들에 대해 우리는 (어떤 층에서의) 평균을 μ = (1/m) Σ z(i)으로 계산하고 분산은 σ^2 = (1/m) Σ (zi-μ)^2 으로 계산한다. 그리고 각

2023.11.20

[Euron 중급 세션 10주차]…
2023.11.13

백엔드 프로젝트 8주차 스…
2023.11.11


최근댓글

오늘 하루 고생 많으셨습니다.
좋은 글 잘 보고 가요! 감사…
좋은 글 잘 보고 가요! 감사…
잘보고 갑니다!

태그

딥러닝교과서, 판다스입문, 딥러닝스터디, 이지스퍼블리싱, Doit, pandas, 데이터분석, 판다스, bda, 데이터사이언스

z(i)에 대해 정규화를 하여 z(i)_norm을 얻는다. 평균을 뺀 뒤 표준편차로 나누는 것이다(수학적 안정성을 위해 ε추가). 이렇게 z값에 대한 정규화를 거쳐, 모든 z들이 평균이 0이고 표준편차가 1이 되도록 만들었다. 하지만 은닉 유닛이 항상 평균 0에 표준편차 1을 갖도록 하는 것이 좋은 것만은 아니다. 은닉 유닛은 다양한 분포를 가져야 하기 때문인데, 그래서 대신 z~를 계산한다. 이것은 γ*z(i)_norm+β와 같으며 여기서 γ와 β는 모델에서 학습시킬 수 있는 변수이다. 경사하강법(혹은 모멘텀, RMSprop, Adam을 이용한 경사하강법)을 이용하여 γ와 β을 학습시킬 수 있는 것이다(신경망에서 계수를 찾았던 것처럼). γ와 β를 이용하면 z~의 평균을 원하는대로 설정할 수 있다. 예를 들어 γ = sqrt(σ^2 + ε) (표준편차와 동일)이고, β가 μ와 같다면, γ*z(i)_norm+β은 정규화 식을 거꾸로 뒤집은 것과 같은 효과를 낼 것이다. 만약 이 식이 성립한다면, z~i는 zi와 같을 것이다. 이렇게 적절히 γ와 β를 설정해서 네 개의 식으로 이루어진 정규화 과정은, 항등함수를 만드는 것과 똑같 효과를 낸 것이다. 하지만 다른 γ와 β값을 정한다면, 은닉 유닛의 값들이 서로 다른 평균이나 분산을 가지도할 수 있다.

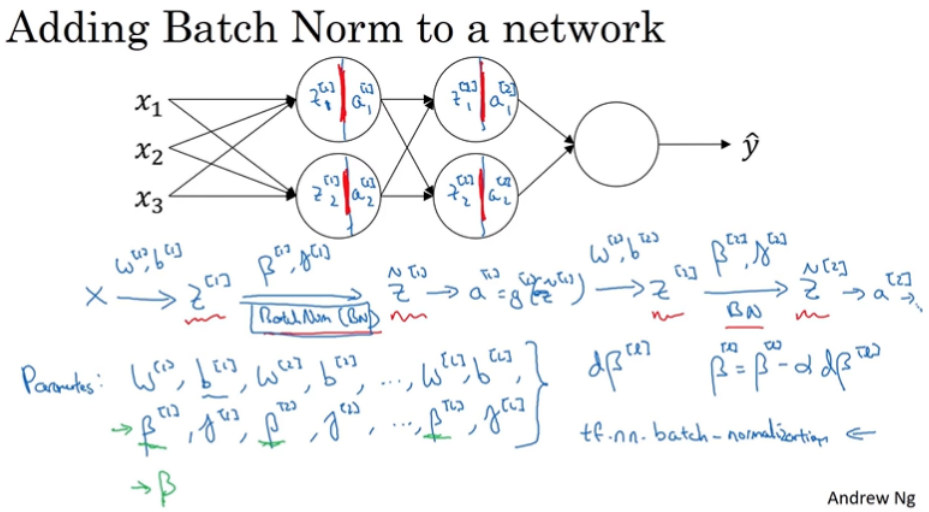
이를 신경망에 적용시켜보자. 이전에는 z(1)등의 값들을 썼다면 이제는 그 대신 z~i를 신경망에 사용할 것이다(또한 어떤 층인지 알 수 있도록 다시 [l]을 써넣자). 여기서 얻을 수 있는 직관은, 우리가 X를 정규화하는 것이 신경망 학습을 도울 수 있는 것을 보았듯 배치 정규화는 입력층에만 정규화를 하는 것이 아니라 신경망 안 깊이 있는 은닉층의 값들까지도 정규화하는 것이다. 이런 정규화를 사용해서 은닉 유닛 z의 평균과 분산을 정규화하는 것이다. 하지만 입력층 X와 은닉 유닛 z(i)를 학습시킬 때의 차이점은, 은닉 유닛 값의 평균과 분산이 0, 1로 고정되기를 원치 않는다는 것이다. 예를 들어 시그모이드 활성 함수가 있을 때 값들이 아래 쪽에 몰려있기보다는, 더 넓은 영역에 걸쳐 퍼져있거나 시그모이드의 비선형성을 살릴 수 있도록 평균이 0이 아닌 다른 값을 갖게 하는 것이 좋다. 이것이 γ와 β를 이용하여 원하는 범위의 값을 만들어내는 이유이다. 다시 말해 은닉 유닛이 표준화된 평균과 분산을 가지되, 평균과 분산은 학습 알고리즘에서 설정할 수 있는 두 변수 γ와 β에 의해 조절되는 것이다. 은닉 유닛 값 z(i)의 평균과 표준편차를 특정한 평균과 분산을 갖도록 정규화하는 것이라고도 할 수 있으며, 이는 0이나 1이 될 수 있고 또다른 값이 될 수도 있다. 이는 γ와 β에 의해 조종되는 것이다.

- 배치 정규화의 장점은 하이퍼파라미터 탐색을 쉽게 만들어줄 뿐만 아니라, 신경망과 하이퍼파라미터의 상관관계를 줄여줍니다.
- 보통 활성화 함수 이전에 사용되며, 작동원리는 아래와 같습니다.
  - $\mu = \frac{1}{m}\sum_i z^{(i)}$
  - $\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$
  - $z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$
  - $\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$
- γ 와 β 는 학습과정에서 학습하는 파라미터입니다. 정규화 이후 다시 선형변환하는 이유는 항상 같은 분포 값을 갖지 않게 하기 위함입니다.

📌배치 정규화 적용시키기(C2W3L05)

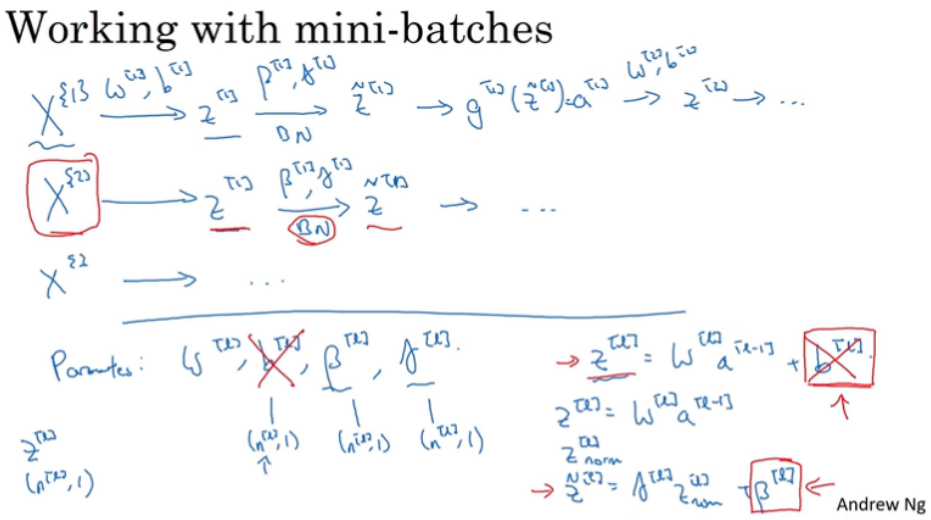
핵심어: 배치 정규화(Batch Normalization)

하나의 은닉층에 배치 정규화를 구현하는 것에 대한 수식을 살펴보았는데, 이번에는 심층 신경망 학습에 어떻게 적용될 수 있는지 살펴보고자 한다.



위와 같은 신경망이 있다고 할 때, 은닉 유닛을 전과 같이 두 영역으로 나눠볼 수 있다. z를 우선 계산하고, 활성화 함수를 이용하여 a를 계산하는 것이다(원 하나가 두 단계의 계산을 나타냄). 다음 층에서도 비슷하게 z[2]_1과 a[2]_1이 될 것. 만약 배치 정규화를 쓰지 않는다면, 입력값 x가 첫 번째 은닉층에 주어질 것이다. 그러면 w[1]과 b[1]에 따라 우선 z[1]을 계산한다. 그리고 활성화 함수에 z[1]이 주어져서 a[1]을 계산하게 된다. 하지만 배치 정규화의 경우 z[1]을 받아서 배치 정규화를 적용하고, 이 과정은 β[1]과 γ[1]의 영향을 받을 것이다. 그럼 새로 정규화된 z[1]을 얻게 되고, 이를 활성화 함수에 줘서 a[1]을 얻는 것이다(g1에 z~[i]를 적용). 이렇게 첫 번째 층에 대한 계산을 마치며, 배치 정규화가 z와 a를 계산하는 사이에 이뤄졌음을 확인할 수 있다. 다음으로 a[1]을 가지고 z[2]를 계산한다. 이 과정 역시 w[2], b[2]의 영향을 받았고 첫 번째 층에서 했던 것처럼 z[2]에서도 대해 배치 정규화를 적용한다. 이 과정은 배치 정규화의 변수인 β[2]과 γ[2]의 영향을 받을 것이다. 이로써 z~[2]을 얻고 활성화 함수에 적용하여 a[2]를 얻게 되는 것이며, 이 과정이 계속된다(마찬가지로 배치 정규화가 z 계산과 a 계산 사이에 이루어짐). 여기서 비정규화된 z값 대신 정규화된 z~값을 사용하며, 이는 첫 번째 층과 두 번째 층에서 도일하다. 비정규화된 z[2] 대신 평균과 분산으로 정규화된 z~[2]를 사용하기도 한다. 즉 신경망에서 변수들 w[1]과 b[1] ~ w[l]과 b[l], 그리고 각 층에서 배치 정규화를 할 때 사용되는 β[1]과 γ[1], β[2]과 γ[2](β는 모멘텀이나 기하급수적 평균을 구할 때 쓰는 하이퍼파라미터와 다른 것)들은 알고리즘에 사용되며, 이젠 이 변수를 찾기 위해 경사 하강법 등 어떤 최적화를 사용할 지 고민해보아야 한다. 예를 들어 어떤 층에서 dβ[l]를 계산하고 β를 β-(학습 속도)*dβ[l]로 수정하면(물론 경사하강법 외에도 Adam, RMSprop, 모멘텀 등을 써서 β나 γ를 업데이트할 수 있음/배치 정규화 또한 평균과 분산으로 굳이

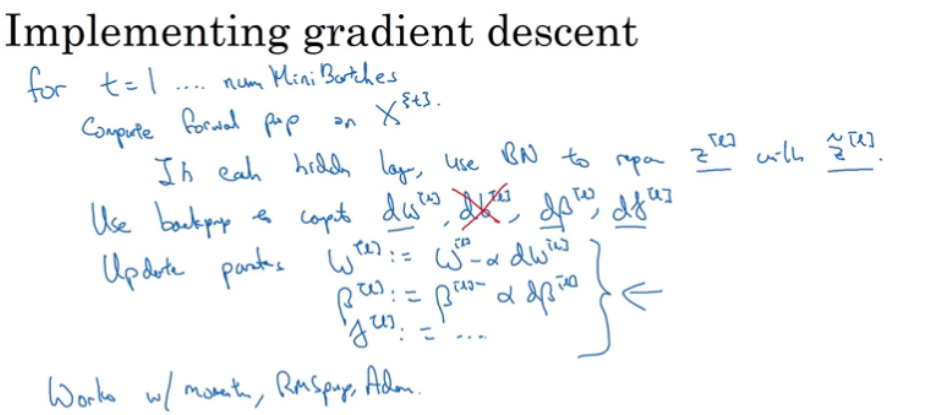
구현할 필요 없고, 텐서플로우의 경우 tf.nn.batch-normalization 함수를 이용해 한 줄로 처리할 수 있다) 이는 경사하강법을 이용해서 배치 정규화로 전체를 학습시킨 경우이다.



하지만 실제로 배치 정규화는 훈련 집합의 미니 배치에 적용된다. 즉 실제로는 첫 번째 미니배치에 대해 이전에 했던 것처럼 w[1]과 b[1]을 이용하여 z[1]을 계산하고, 이 미니배치 안에서 z[1]의 평균과 분산을 계산한 뒤 평균을 빼고 표준편차로 나눠 배치 정규화를 진행한다(+β[1]과 γ[1]을 이용하여 값 조정). 이 과정을 통해 첫 번째 미니 배치에서 z~[1]을 얻게 되고 활성화 함수를 적용해 a[1]을 얻는 것이다. 그리고 z[2]에서도 동일한 과정이 반복된다. 이렇게 첫 번째 미니 배치에 대해 경사하강법을 이용해 과저을 마치면 두 번째 미니배치로 이동한다. 그리고 X{2}에 대해 비슷한 방법으로 z[1]을 계산하고, 배치 정규화를 써서 z~[1]을 계산한다. 참고로 이 정규화에서는 두 번째 미니배치에 있는 데이터만을 이용해서 진행한다. 두 번째 미니배치만 이용해서 평균과 분산을 계산하는 것이다. 그리고 β와 γ로 보정하여 z~을 얻게 된다. 마찬가지로 세 번째 미니배치에 대해서도 동일한 계산을 한다.

이제 매개변수화 과정에서 한 가지 짚고 넘어가야 할 문제가 있다. 각 층에 대해 변수 w[l]과 b[l], β[l]과 γ[l]가 있는데, 여기서 z[l]은 w[l]a[l-1]+b[l]로 계산된다. 배치 정규화는 미니배치를 보고 z[l]이 평균0, 분산1을 갖도록 정규화한 뒤 β와 γ를 이용하여 값을 조정해주는 것이다. 여기서 b[l]은 값이 무엇이든간에 사라지는데, 이는 배치정규화 과정에서 z의 평균을 계산한 뒤 빼주기 때문이다. 즉 미니배치의 모든 예시에 상수 b[l]을 더해도 결국 평균을 빼주면서 사라지기 때문에 아무런 영향을 미치지 않는 것이다. 따라서 배치정규화를 쓸 때 b 변수를 없앨 수 있으며 또는 항상 0으로 둔다고 생각해도 좋다. z[l] = w[l]*a[l-1]이 되는 것과 마찬가지다. 그리고 z~ = γ[l]*z[l]_norm+β[l](여기서는 다음 층에 전달되는 z~의 평균을 정하기 위해 β[l]을 써야함) 으로 z[l]_norm을 계산한다. 다시 말하자면 배치 정규화가 z[l]의 평균을 0으로 만들기 때문에 b[l]이라는 변수가 필요 없으며 그걸 없애는 대신 β[l]이 그 역할을 차지하는 것이다. 결과적으로 편향 변수를 결정하기 때문이다. 끝으로 z[l]의 차원이 (n[l], 1)이었다고 했었는데, 따라서 b[l]은 (n[l], 1)차원(n[l]은 층 l에서의 은닉 유

닛 개수) 따라서 β[l]와 γ[l]의 차원도 (n[1], 1)이 된다. 어떤 신경망이든지 간에 n[l] 개의 유닛을 가지고 있을 때 β[l]와 γ[l]이 각 은닉 유닛의 값을 조정하는 데 쓰이기 때문이다.



Andrew Ng

이제 모든 것을 합쳐서 배치 정규화를 사용하여 경사 하강법을 구현하는 방법을 알아보고자 한다. 미니 배치 경사하강법을 사용한다고 했을 때, t가 1부터 미니배치의 숫자까지 바뀐다고 하자. 이때 미니배치 X{t}에 대해 순방향 전파 사용하며, 이는 각 은닉층에서 z[l]을 z~[l]으로 바꾸기 위해서이다. 이 과정을 거치면 미니배치의 평균과 표준편차를 이용해서 z[l]을 z~[l]으로 바꿔주는 것이 된다. 그리고 역방향 전파를 이용해서 층의 모든 변수에 대해 dw, db, dβ, dγ를 계산한다. 그리고 각 변수들을 업데이트 해준다(방금 전에 b는 없앴으니 생각 x). w는 w-α*dw가 되고, β는 β-α*dβ로, γ는 γ-α*dγ로 업데이트 한다. 이렇게 차이(gradients)를 계산하면 경사하강법을 이용할 수 있는 것이다(물론 모멘텀, RMSprop, Adam을 사용하는 경사하강법에도 쓸 수 있다).

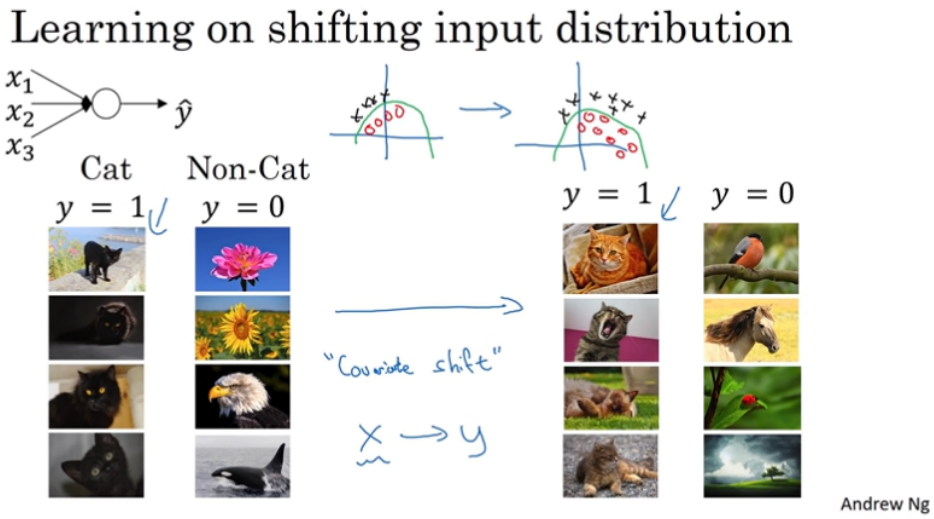
- 은닉층에서 두 단계로 나뉩니다.
  - 첫째, 선형결합인 z 를 계산하고, 이를 배치 정규화 시킵니다.
  - 둘째, 정규화 된 값들을 활성화 함수를 거쳐 활성화 값 a 를 얻습니다.
- 선형결합 단계에서 상수항 b 는 없어집니다. 그 이유는 배치 정규화 과정에서 z의 평균을 빼주면 사라지기 때문입니다.

📌배치 정규화가 잘 작동하는 이유는 무엇일까요? (C2W3L06)

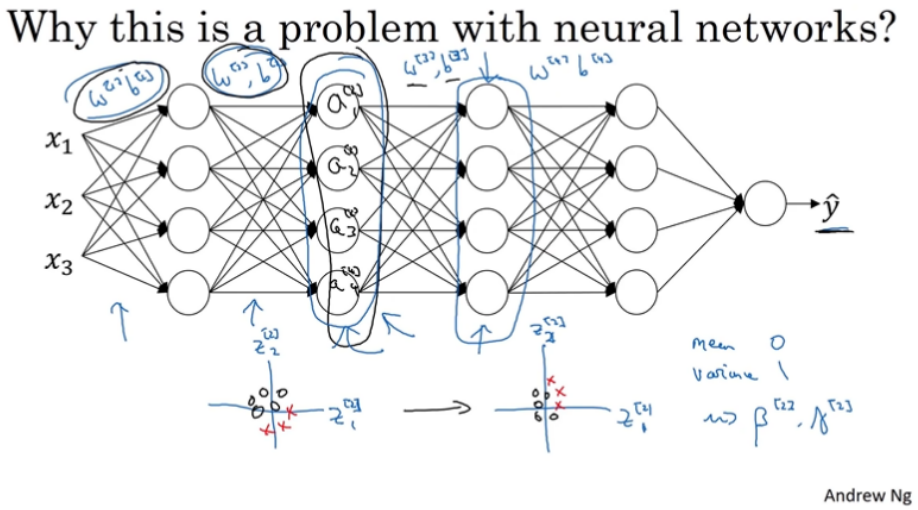
핵심어: 배치 정규화(Batch Normalization), 정규화(regularization), 드롭아웃(dropout)

배치 정규화가 잘 작동하는 이유는 첫 번째, 입력 특성 x를 평균 0, 분산 1로 정규화하는 것이 학습 속도를 올리는 것과 비슷한 동작을 하기 때문이다(X의 스케일을 맞추면 학습속도 상승). 은닉 유닛과 입력층 모두에서 말이다. 하지만 이는 부분적인 그림

에 불과하며, 이것 외에도 배치 정규화가 무엇을 하는 것인지 더욱 깊이 이해할 수 있는 몇 가지 직관이 더 있다. 이번에는 그 내용에 대해 살펴보고자 한다.



배치 정규화가 작동하는 두 번째 이유는 깊은 신경망에서, (예를 들어 10번 층의 가중치가 1번 층처럼) 앞쪽 층의 가중치의 변화에 영향을 덜 받는다는 것이다. 고양이 분류를 위해 신경망을 학습한다고 하자. 로지스틱 회귀처럼 얕거나 깊을 수 있다. 왼쪽에는 검은 고양이 이미지만 써서 학습시킨 반면 오른쪽에는 다양한 색깔의 고양이로 학습시켰다고 했을 때, 오른쪽에 있는 색이 있는 고양이가 정답일 경우 왼쪽 모델은 좋은 성능을 내지 못할 것이다. 정답(o)과 오답(x)이 그래프와 같이 분포했을 때 (이를 일반화한 것이 오른쪽 그래프), 왼쪽 데이터로 학습시킨 모델이 오른쪽 데이터에서 좋은 성능을 못 낼 것으로 추측할 수 있다. 사실 두 함수는 동일하지만, 왼쪽 데이터에만 학습 알고리즘을 적용해서 얻은 초록색 결정 기준에 좋은 성능을 기대하기는 어렵다. 이처럼 데이터 분포가 변화하는 것을 좀 어려운 말로 공변량 변화 (covariate shift)라고 하며 이 아이디어는 X, Y 간의 대응을 학습시킬 때 X의 분포가 바뀐다면 학습 알고리즘을 다시 학습시켜야 한다는 것이다. 이 예시의 고양이 구분하기처럼 X에서 Y로 대응하는 관측 함수가 바뀌지 않더라도 말이다. 우리가 새롭게 함수를 학습시킨다면 더 정확해지거나 관측 함수가 함께 움직여서 더 나빠질 수도 있다.

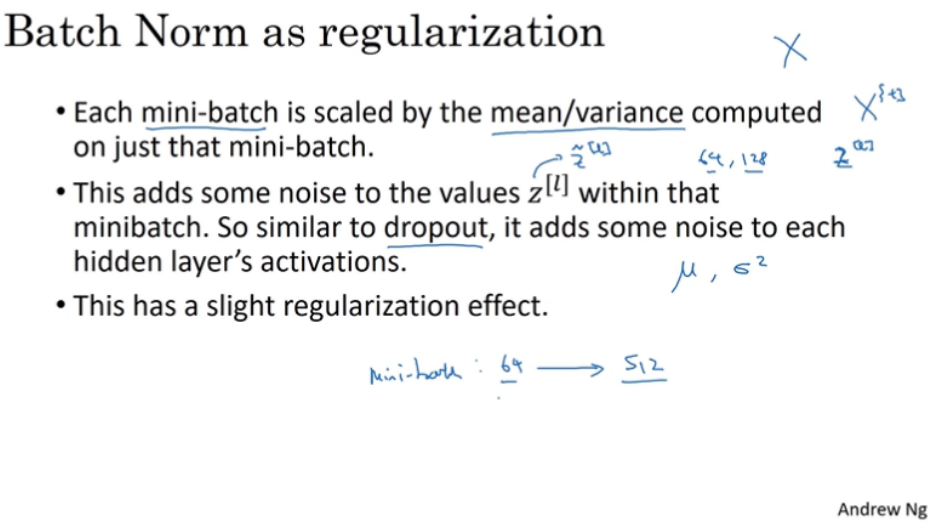


그럼 공변량 변화가 신경망에 어떻게 적용될 수 있을까? 이 세 번째 은닉층의 관점에서 학습 과정을 살펴보자. 이 신경망은 w[3]와 b[3]을 학습시키고 있다. 세 번째 은닉층은 앞선 층에서 값들을 받아오고 관측값인 Y와 가까운 yhat과 관련 있는 일을 할 것이다. 잠시 왼쪽 부분을 가려보면, 세 번째 층이 값들을 받아오고, 그것들을 a[2]_1, a[2]_2, a[2]_3, a[2]_4라고 하자. 특성 값인 x_1, x_2, x_3, x_4가 될 수도 있다. 이 세 번째 층이 할 일은 이 값들을 받아와서 yhat으로 대응시키는 것이다. 즉 경사하강법을 써서 w[3]와 b[3]나 나아가 w[4]와 b[3], w[5]와 b[5]를 신경망이 좋은 성능을 내도록 학습시킴으로써 왼쪽에 가려져 있는 값들로부터 출력값인 yhat을 만들어낸다고 볼 수 있다.

이제 신경망의 왼쪽을 다시 드러내고, 신경망은 매개변수 w[2]와 b[2], w[1]와 b[1]도 학습시키고 있다. 이 매개변수의 값이 바뀌면 여기 a[2]의 값들도 바뀐다.따라서 세 번째 은닉층의 관점에서 이 은닉층(3~4번째 층)의 값들이 계속 바뀌고 있는 것이다. 앞서 살펴봤던 공변량 변화의 문제를 계속 겪게되는 것이다. 여기서 배치 정규화는 은닉층 값들의 분포가 변화하는 양을 줄여주며, 만약 이 은닉층들의 값의 분포를 그린다면(여기서는 재정규화된 z인 z[2]_1와 z[2]_2를 쓴다), 값 네 개 대신 두 개를 사용해서 2차원에 나타낼 수 있다. 이때 배치 정규화가 말하고자 하는 것은 z[2]_1과 z[2]_2가 바뀔 수 있을 것이고(신경망이 앞선 층의 매개변수를 새로 고치면서 아마 바뀔 것이다), 배치 정규화는 얼마나 바뀌든지 간에 z[2]_1과 z[2]_2의 평균과 분산이 동일하게 유지될 것이라고 말한다. 즉 z[2]_1과 z[2]_의 값이 바뀌더라도 적어도 평균과 분산은 0과 1처럼 유지될 것이다. 굳이 0과 1이 아니더라도 β[2]나 γ[2]와 같은 값도 가능하다. 다시말해 배치정규화가 하는 일은 앞선 층에서의 매개변수가 바뀌었을 때 세 번째 층의 값이 받아들여서 학습하게 될 값의 분포를 제한하는 것이며, 입력값이 바뀌어 발생하는 문제를 더욱 안정화시키는 것이라고 볼 수 있다. 뒤쪽 층은 당연하게 더 쉽게 학습할 수 있다. 입력 값의 분포가 조금 바뀌더라도 그에 대한 영향이 거의 없고, 앞쪽 층이 계속 학습하면서 값을 바꾸더라도 뒤쪽 층이 그것 때문에 겪는 부담을 줄이는 것이 된다. 그리고 이것은 또 앞쪽 층의 매개변수와 뒤쪽 층의 매개변수 사이의 관계를 약화시키며, 신경망의 각 층이 다른 층과 상관 없이 스스로 배

울 수 있도록 하는 것이다. 이것을 통해서 전체 신경망의 학습 속도를 상승시킬 수 있다.

반드시 알아두어야 할 것은 배치정규화에 의해 신경망에서 뒤쪽에 있는 층의 관점에서 앞선 층이 너무 많이 변화하지 않는다는 것이다. 평균과 분산이 일정하도록 제한받았기 때문이다. 이를 통해 뒤쪽 층의 학습이 더욱 용이해진다.



배치 정규화의 두 번째 효과는 규제 효과인데, 배치 정규화에서 다소 비직관적일 수 있다. 각각의 미니 배치 X{t}가 가진 z[l]에 대해 그 미니배치의 평균과 분산에 따라 값을 조정할 것이다. 이때 미니배치에서 계산한 평균과 분산은 전체 데이터로부터 계산한 것에 비해 다소 잡음을 갖고 있다. 64,128, 256 내지는 더 큰 크기의 훈련 예시를 지닌 미니 배치에 대해 상대적으로 작은 데이터에 대해 추정한 것이기 때문이다. 여기서 z[l]에서 z~[l]으로 조정하는 과정 역시 잡음이 끼어있는데, 잡음이 끼어있는 평균과 분산으로 계산하는 것이기 때문이다. 드롭아웃과 유사하게 은닉층의 활성 함수에 잡음이 끼는 것인데, 드롭아웃에서는 은닉층을 가져와서 확률에 따라 0을 곱하거나 1을 곱했으므로 곱셈 잡음을 가지는 것이다. 반면에 배채 정규화는 표준편차로 나누니 곱셈 잡음도 있고 평균을 빼니 덧셈 잡음도 있다. 드롭아웃과 마찬가지로 배치 정규화는 약간의 일반화 효과를 가지는 것이다. 은닉층에 잡음을 추가하는 것은, 이후의 은닉층이 하나의 은닉층에 너무 의존하지 않도록 하는 것이기 때문이다. 잡음이 아주 작다보니 일반화 효과가 그리 크지 않은데, 이때 더욱 강력한 일반화를 위해 배치 정규화와 드롭아웃을 함께 사용하는 방법이 있다.

또 다른 비직관적인 효과를 살펴보자. 큰 미니배치를 사용한다고 했을 때(64대신 512) 잡음이 줄어들고, 따라서 일반화 효과도 줄 것이다. 이는 큰 미니배치를 사용하면 일반화 효과가 줄어든다는 드롭아웃의 이상한 특성이다. 그래서 배치 정규화를 일반화의 목적으로 사용하기는 어렵다. 배치 정규화는 일반화를 목적으로 만든 것이 아니다. 하지만 의도하지 않거나, 의도한 것보다 더 큰 효과를 학습 알고리즘에 가져오는 경우도 종종 있다. 그래도 배치 정규화를 일반화 목적으로 사용하는 것은 권장하지 않으며, 은닉층의 활성 함수를 정규화해서 학습 속도를 올리는 용도로 사용하는 것이 좋다. 일반화는 의도하지 않은 부수적인 효과에 가깝다.

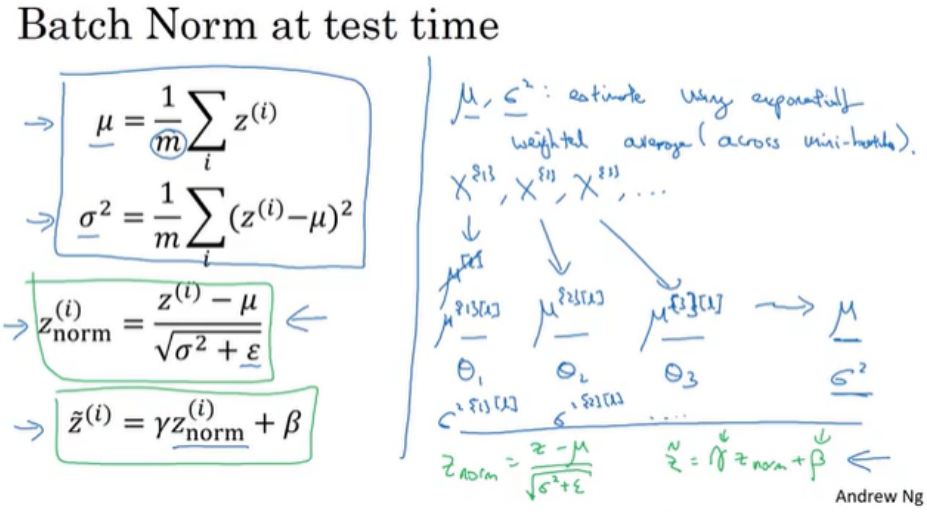
배치 정규화에 대해 논의할 때 한 가지 더 알아두었으면 하는 것은, 배치 정규화는 한 번에 미니배치 하나의 데 | 터를 다루며 미니배치의 평균과 분산을 계산한다는 것이다. 테스트 과정에서는 예측을 해서 신경망을 평가하는데, 예시에는 미니배치가 없으므로 한 번에 예시 하나씩을 처리한다. 따라서 테스트 과정에서는 예측이 잘 맞을 수 있도록 조금 다르게 접근해야 한다.

- 배치 정규화는 입력특성 X 의 평균을 0, 분산을 1로 만듦으로써 학습 속도를 빠르게 합니다.
- 배치 정규화가 잘 되는 이유중 하나는 이전 층의 가중치 영향을 덜 받게 하는데 있습니다. 은닉층 값의 분포 변화를 줄여줘서, 입력 값의 분포를 제한하기 때문입니다. 즉, 배치 정규화는 입력값이 바뀌어서 발생하는 문제를 안정화 시킵니다. 앞층과 뒷층의 매개변수의 상관 관계를 줄여주기 때문에, 학습속도를 향상시킬 수 있습니다.
- 배치 정규화의 또 다른 효과는 파라미터의 정규화(regularization)입니다. 미니배치로 계산한 평균과 분산은 전체 데이터의 일부으로 추정한 것이기 때문에 잡음이 끼어있습니다.
- 드롭아웃의 경우 은닉유닛에 확률에 따라 0 혹은 1을 곱하기 때문에 곱셈 잡음이 있습니다. 배치 정규화의 경우 곱셈잡음($\times \frac{1}{\sigma}$)과 덧셈잡음($+(-\mu)$)이 동시에 있습니다. 따라서 약간의 정규화 효과가 있습니다.
- 은닉층에 잡음을 추가한다는 것은 이후 은닉층이 하나의 은닉 유닛에 너무 의존하지 않도록 만듭니다.
- 큰 미니배치를 사용시 이 정규화 효과는 상대적으로 약해집니다.

📌 테스트시의 배치 정규화?(C2W3L07)

핵심어: 배치 정규화(Batch Normalization), 지수 가중 이동 평균 (Exponentially Weighted Average)

배치 정규화는 한 번에 하나의 미니배치 데이터를 처리한다. 하지만 테스트에서는 한 번에 샘플 하나씩을 처리해야 한다. 이를 위해 신경망을 학습시키는 법을 알아보고자 한다.



여기에 나와있는 식들은 학습 중 배치 정규화를 위해 사용했던 식들이다. 단일 미니배치에서 평균을 구하기 위해 z(i) 값을 합산했고, 미니 배치 안의 샘플들을 모두 합한 것이다. 여기서 m은 미니 배치 안의 샘플 수이지, 전체 훈련 세트에서의 개수가 아니다. 그리고 분산을 계산할 수 있는데, 평균과 표준편차로 크기를 조정해서 z_norm도 계산할 수 있다(ε은 수학적 안정성을 위해 추가). z~는 z_norm을 γ와 β을 써서 조정한 것이다. 여기서 위 2개의 식을 계산하는 데 필요한 μ와 σ^2은 미니 배치 안에서

계산되지만, 테스트 과정에서는 64,128, 256개 등의 샘플을 포함하는 미니 배치가 없으므로 동시에 처리할 수 없다. 따라서 μ와 σ^2을 처리할 다른 방법이 필요하다. 만약 하나의 샘플만 있을 때 그 샘플의 평균과 분산을 구하는 것은 거의 발생하기 힘든 상황이기 때문이다.

실제로 동작할 때는, 각각 독립된 μ와 σ^2의 추정치를 사용하면 된다. 전형적인 배치 정규화를 구현할 때는 여러 미니 배치에 걸쳐서 구한 지수가중평균을 추정치로 사용한다. 다시 정리하자면, 어떤 층 L을 고르고 미니배치 X{1}, X{2} 등과 대응하는 값 Y가 있다고 하자. L 층에 대해 X{1}을 학습시킨다면 μ[l]을 얻을 것이다. 여기서는 μ{1}[l], 그 층의 첫 번째 미니 배치라고 표기한다. 그리고 두 번째 미니배치에 대해서도 학습시키면 두 번째 μ 값을 을 수 있다. 마찬가지로 이 은닉층의 세 번째 미니 배치에 대해서 세 번째 μ 값을 얻을 수도 있다. 지금까지 봤듯 지수가중평균을 이용해서 θ_1, θ_2, θ_3의 평균을 계산한다고 하자. 예를 들어 현재 온도의 기하급수적 평균을 계산했을 때, 이 평균 벡터에서 가장 최근의 평균값이 뭐였는지 기록해야 했다. 그럼 지수가중평균이 그 은닉층의 z값 평균의 추정치가 되는 것이다. 비슷하게 지수가중평균을 이용해서 σ^2의 값을 추적할 수도 있다. 해당 층의 첫 번째 미니배치에서 σ^2을 구하고 두 번째 미니배치에서 반복하는 것이다. 이렇게 μ와 σ^의 이동 평균을 구할 수 있는 것이다. 신경망에서 서로 다른 미니 배치로 학습시킨 각 층에 대해서 말이다.

최종적으로 테스트 과정에서는, 가지고 있는 z값과 μ와 σ^2의 지수가중평균을 이용하여 z_norm을 계산만 하면 된다. 가장 최근의 값이 뭐였든 간에 여기서 값을 조정하는 데 사용되는 것이다. 그리고 테스트 샘플에 대해 z~를 계산할 수도 있을 것이다. 방금 왼쪽에서 계산한 z_norm과 신경망 학습 과정에서 학습시킨 γ, β 매개변수를 이용하는 것이다. 여기서 알아두어야 할 것은, 학습 과정 중에 μ와 σ^2는 64나 256개 등의 미니 배치로 계산하지만, 테스트할 때는 한 번에 샘플 하나를 처리해야 한다. 즉 μ와 σ^2를 훈련세트로부터 추정해야 하는 것인데, 여기에는 여러 가지 방법이 있다. 이론적으로는 훈련 세트를 이용해 최종적으로 학습한 신경망의 μ와 σ^2를 얻을 수도 있고, 실제로는 μ와 σ^2가 학습하면서 가진 값을 추적하며 지수가중평균을 사용한다. 지수가중평균을 이용하는 것은 이동 평균이라고 말하기도 한다. μ와 σ^2의 추정치를 대충 결정한 뒤 테스트 과정에서 사용하는 것이다. 은닉층의 z값으 조정할 때 필요하기 때문이다. 실제로 이 방법은 μ와 σ^2를 추정하는 데 꽤 안정적이다. 딥러닝 프레임워크에는 기본적으로 μ와 σ^2를 추정할 수 있는 방법이 있을 것이며, 실제로 은닉층의 값 z에 대해 평균과 분산을 추정하는 것도 테스트 과정에서도 합리적으로 잘 동작할 것이다.

- 이전 강의에서 사용된 배치 정규화의 수식입니다.
  - $\mu = \frac{1}{m}\sum_i z^{(i)}$
  - $\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$
  - $z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$
  - $\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$
- 테스트 시에는 배치가 하나이기 때문에 평균과 분산을 계산할 수 없습니다. 따라서, 학습시에 사용된 미니배치들의 지수 가중 이동 평균을 추정치로 사용합니다.