Deep learning 강의자료 후반부

이민수

0. 들어가며

이번 강의에서는 기본적인 딥러닝 모델들을 적용할 때 어떻게 하면 효과적으로 학습할 수 있을지에 대해서 설명하고자 합니다. 요즘에는 딥러닝 프레임워크들이 워낙 잘 구현되어 있어 원리를 잘 몰라도 코드를 돌릴 수 있지만 학습을 잘 하기위해서 어떤 방법을 기본적으로 사용하는지 이해한다면 모델을 튜닝 하는 데에 도움이 될 수 있다고 생각합니다.

미니배치, 가중치 매개변수 초기값, 배치정규화, Over fitting 방지법(가중치감소, 드랍아웃)을 설명하겠습니다.

1. 학습을 위한 기술

1.1 미니배치 학습

기계학습 문제는 훈련 데이터를 사용해 학습합니다. 더 구체적으로 말하면 훈련 데이터에 대한 손실 함수의 값을 구하고, 그 값을 최대한 줄여주는 매개변수를 찾아냅니다. 이렇게 하려면 모든 훈련 데이터를 대상으로 손실 함수 값을 구해야 합니다. 즉, 훈련 데이터가 100개 있으면 그로부터 계산한 100개의 손실 함수 값들의 합을 지표로 삼는 것입니다.

$$E = -\frac{1}{N}\sum_{n}\sum_{k} t_{nk}\log y_{nk}$$

데이터가 N개라면 t_{nk}는 n번째 데이터의 k번째의 값을 의미합니다. (y_{nk}는 신경망의 출력, t_{nk}는 정답 레이블입니다.) 수식이 좀 복잡해 보이지만 데이터 하나에 대한 손실 함수를 단순히 N개로 확장한 후 N으로 나누어 정규화 하고 있습니다. N으로 나눔으로써 '평균 손실 함수'를 구하는 것입니다. 이렇게 평균을 구해 사용하면 훈련 데이터 개수와 관계없이 언제든 통일된 지표를 얻을 수 있습니다.

그런데 딥러닝이 학습하는 빅데이터의 수준은 수백만에서 수천만이 넘는 거대한 값이 되기도 합니다. 이 많은 데상으로 일일히 손실 함수를 계산하여 합하는 것은 현실적이지 않습니다. 이런 경우 데이터 일부를 추려 전체의 '근사치'로 이용할 수 있습니다. 이 일부를 미니배치라고 합니다.

가령 60,000장의 훈련 데이터 중에서 100장을 무작위로 뽑아 그 100장을 사용하여 학습하는 것입니다. 이런 학습 방법을 미니배치 학습이라고 합니다.

1.2 가중치의 초기값

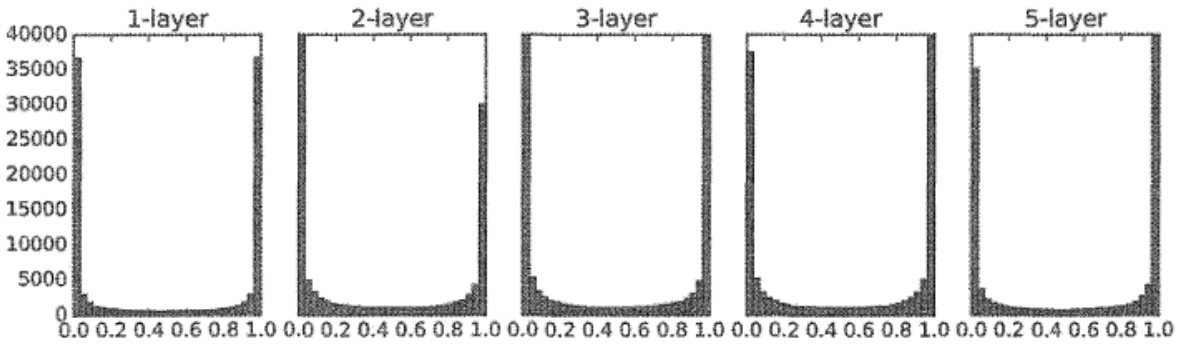
신경망 학습에서 특히 중요한 것이 가중치의 초기값입니다. 가중치의 초기값을 무엇으로 설정하느냐가 신경망 학습의 성패를 가르는 일이 실제로 자주 있습니다. 권장하는 초기값에 대해서 설명하고 실제로 신경망 학습이 신속하게 이루어 지는 모습을 확인하겠습니다.

1.2.1 초기값을 동일하게 하면

가중치 W를 모두 같은 값으로 초기화 한다면 어떻게 될까요? 순전파때는 두번째 층의 뉴런에 모두 같은 값이 전달됩니다. 미분의 연쇄법칙을 떠올려보면 두번째 층의 모든 뉴런에 같은 값이 입력된다는 것은 역전파 때 두번째 층의 가중치가 모두 똑같이 갱신된다는 말이 됩니다. 다시 말해 뉴런이 100개가 됐든 1000개가 됐든 같은 값을 출력하게 돼 네트워크의 표현력을 제한하게 된다는 얘기입니다.

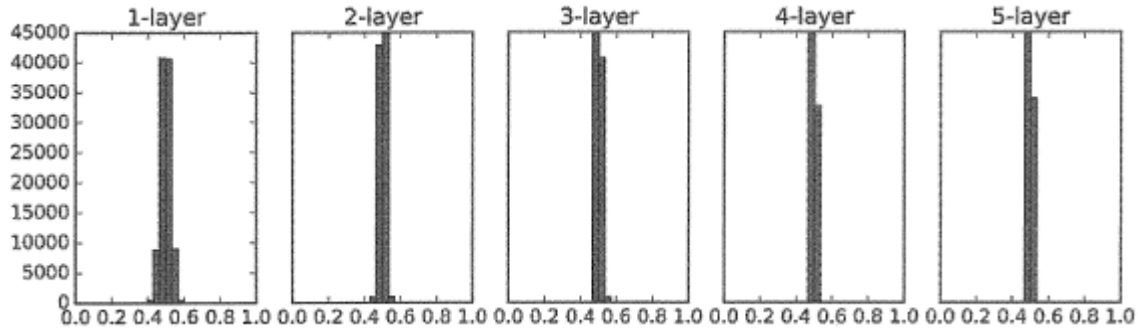
1.2.2 은닉층의 활성화값 분포와 문제점

가중치의 초기값에 따라 은닉층의 활성화값들이 어떻게 변화는지 알아봅시다. 구체적으로는 활성화 함수로 시그모이드 함수를 사용하는 5층 신경망에 무작위로 생성한 데이터를 흘리며 각 층의 활성화값 분포를 히스토그램으로 그려보겠습니다. 층이 5개가 있으며, 각 층의 뉴런은 100개로 설정하였습니다. 입력 데이터로서 1000개의 데이터를 정규분포로 무작위로 생성하여 이 5층 신경망에 흘립니다. 활성화 함수로는 시그모이드 함수를 이용했습니다. 가중치의 분포는 표준편차가 1인 정규분포를 사용했습니다. 이때 활성화값들의 분포가 어떻게 되는지 관찰하여 봅시다.



각 층의 활성화값들이 0과 1에 치우쳐 분포됨을 알 수 있습니다. 여기에서 사용한 시그모이드 함수는 그 출력이 0이나 1에 가까워지면 미분이 0에 다가가기 때문에 데이터가 0과 1에 치우쳐 분포하게 되면 역전파의 기울기 값이 점점 작아지다 사라지게 됩니다. 이것이 기울기 소실이라 알려진 문제입니다. 층을 깊게 하는 딥러닝에서 기울기 소실은 심각한 문제가 될 수도 있습니다.

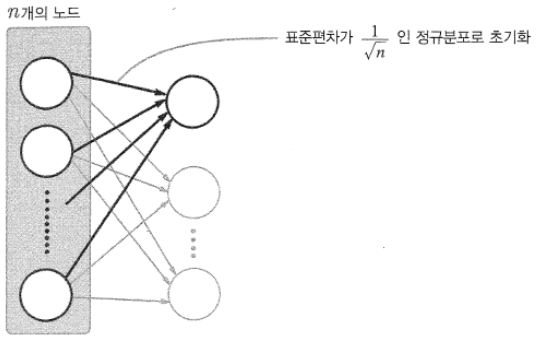
표준편차를 0.01로 한 정규분포의 경우 각 층의 활성화값 분포는 다음과 같이 됩니다.

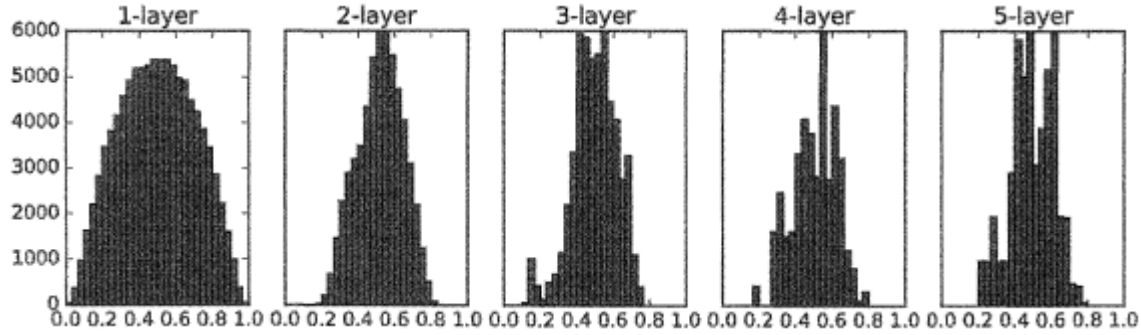


이번에는 0.5 부근에 집중되었습니다. 앞의 예처럼 0과 1로 치우치진 않았으니 기울기 소실 문제는 일어나지 않습니다만, 활성화값들이 치우쳤다는 것은 표현력 관점에서는 큰 문제가 있는것입니다. 이 상황에서는 다수의 뉴런이 거의 같은 값을 출력하고 있으니 뉴런을 여러 개 둔 의미가 없어진다는 뜻입니다. 예를 들어 뉴런 100개가 거의 같은 값을 출력한다면 뉴런 1개짜리와 별반 다를게 없는 것이죠. 그래서 활성화값들이 치우치면 표현력을 제한다는 관점에서 문제가 됩니다.

1.2.3 Xavier 초기값

이러한 기울기 소실 문제와 표현력의 제한 문제를 덜 이르킨다 알려져있는 초기값을 알아보겠습니다. 사비에르(Xavier)라는 사람이 논문에서 권장하는 가중치 초기값인 Xavier 초기값을 써보겠습니다. 현재 Xavier 초기값은 일반적인 딥러닝 프레임워크들이 표준적으로 이용하고 있습니다. 이 논문에서 각 층의 활성화값들을 광범위하게 분포시킬 목적으로 가중치의 적절한 분포를 찾고자 했습니다. 그리고 앞 계층의 노드가 n개라면 표준편차가 1/n^1/2 인 분포로 사용하면 된다는 결론을 이끌었습니다.

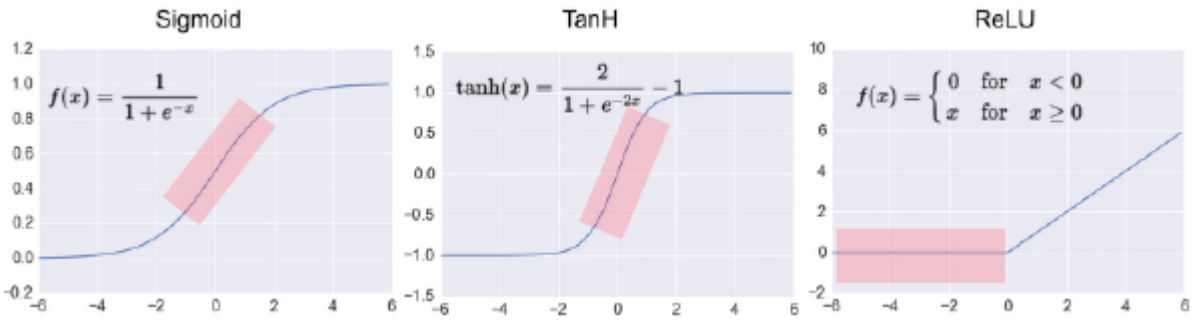




Xavier 초기값을 사용한 결과는 위와 같습니다. 이결과를 보면 층이 깊어지면서 형태가 다소 일그러지지만, 앞에서 본 방식보다는 확실히 넓게 분포됨을 알 수 있습니다. 각 층에 흐르는 데이터는 적당히 퍼져 있으므로, 시그모이드 함수의 표현력도 제한받지 않고 학습이 효율적으로 이뤄질 것으로 기대됩니다.

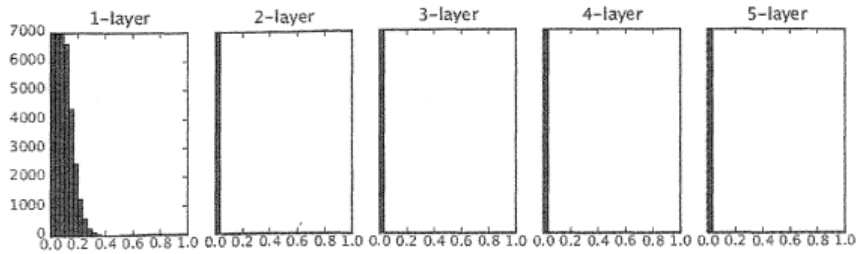
1.2.4 He 초기값

Xavier 초기값은 sigmoid 함수와 tanh함수에 알맞게 설정된 가중치 초기값입니다. 반면 ReLU를 이용할 때는 ReLU에 특화된 초기값을 이용하라고 권장합니다. 이 특화된 초기값을 찾아낸 사람의 이름을 따 He(히) 초기값이라 합니다.

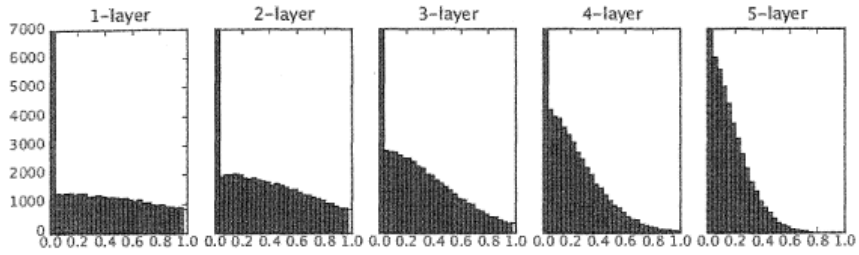


He초기값은 ReLU함수에 맞게 Xavier 초기값을 변형한 것입니다. Xavier 초기값은 활성화 함수가 선형인 것을 전제로 하는데 Sigmoid와 tanh는 중앙 부근이 거의 선형인 함수이므로 Xavier 초기

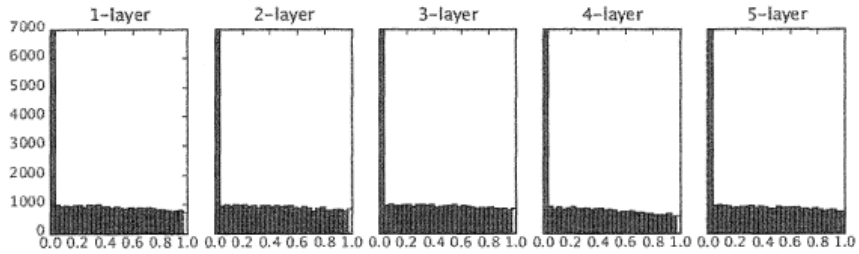
값이 적당합니다. 반면 ReLU는 음의영역이 값이 0이라서 더 넓게 분포시키기 위해 계수가 클 필요가 있습니다. 따라서 Xavier초기값의 표준편차에 2를 곱하여 더 넓은 정규 분포를 만들어 주었습니다.



표준편차가 0.01인 정규분포를 가중치 초깃값으로 사용한 경우



Xavier 초깃값을 사용한 경우



He 초깃값을 사용한 경우

결과를 보면 std= 0.01일 때의 각 층의 활성화값들은 아주 작은 값들입니다. 신경망에 아주 작은 값이 흐른다는 것은 역전파 때 가중치의 기울기 역시 작아진다는 뜻입니다. 이는 중대한 문제이며 실제로도 학습이 거의 이뤄지지 않을 것입니다.

이어서 Xavier 초기값 결과를 보면 층이 깊어지면서 치우침이 조금씩 커집니다. 실제로 층이 깊어지면 활성화값들의 치우침도 커지고, 학습할 때 '기울기 소실'문제를 일으킵니다.

마지막으로 He 초기값은 모든 층에서 균일하게 분포되었습니다. 층이 깊어져도 분포가 균일하게 유지되기에 역전파 때도 적절한 값이 나올 것으로 기대할 수 있습니다.

즉, 활성화 함수로 ReLu를 사용할 때는 He 초기값을, sigmoid나 tanh 등의 S자 모양 곡선일때는 Xavier 초기값을 쓰겠습니다.

1.3 배치 정규화

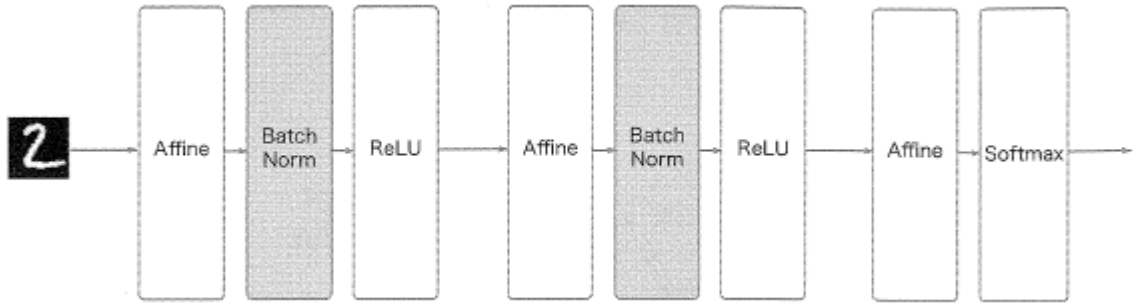
앞 절에서 가중치의 초기값을 적절히 설정하면 각 층의 활성화값 분포가 적당히 퍼지면서 학습이 원활하게 수행된다는 것을 배웠습니다. 그렇다면 각 층이 활성화값을 강제로 퍼트리게하면 어떨까요? "배치 정규화"가 그런 아이디어에서 출발한 방법입니다.

배치정규화가 주목받는 이뉴는 다음과 같습니다.

- 학습을 빨리 진행할 수 있다.
- 초기값에 크게 의존하지 않는다
- 오버피팅을 억제한다

딥러닝의 학습시간이 길다는 걸 생각해보면 첫번째 이점은 아주 반가운 일입니다. 초기값에 크게 신경 쓸 필요가 없고, 오버피팅 억제 효과가 있다는 점도 딥러닝 학습의 두통거리를 덜어줍니다.

배치 정규화의 기본 아이디어는 앞에서 말했듯이 각 층에서의 활성화값이 적당히 분포되도록 조정하는 것입니다. 그래서 다음 그림과 같이 데이터 분포를 정규화하는 '배치 정규화 계층'을 신경망에 삽입합니다.



배치 정규화는 그 이름과 같이 학습 시 미니배치를 단위로 정규화합니다. 구체적으로는 데이터 분포가 평균이 0, 분산이 1이 되도록 정규화 합니다.

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

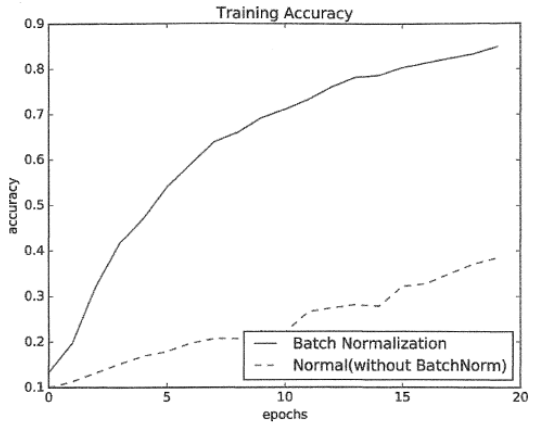
$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

미니배치 B = $\{x_1, x_2, ..., x_m\}$이라는 m개의 입력 데이터의 집합에 대해 평균 u_B와 분산 σ_b^2 을 구합니다. 그리고 입력 데이터를 평균이 0, 분산이 1이 되게 정규화 합니다.

단순히 미니배치 입력 데이터 $\{x_1, x_2, ..., x_m\}$을 평균 0, 분산 1인 데이터 x-hat으로 변환하는 일을 활성화 함수의 앞에 삽입함으로써 데이터 분포가 덜 치우치게 할 수 있습니다.

1.3.1 배치 정규화의 효과



MNIST 데이터셋을 사용하여 배치정규화 계층을 사용할 때와 사용하지 않을 때의 학습 진도가 어떻게 달라지는지를 나타낸 그래프입니다. 그래프와 같이 배치 정규화가 학습을 빨리 진전시키고 있습니다.

1.4 오버피팅 해결책

오버피팅이란 신경망이 훈련 데이터에만 지나치게 적응되어 그 외의 데이터에는 제대로 대응하지 못하는 상태를 말합니다. 훈련 데이터에는 포함되지 않은, 아직 보지 못한 데이터가 주어져도 바르게 식별해내는 모델이 바람직합니다. 딥너링 기법을 이용할 시에 복잡하고 표현력이 높은 모델

을 만들 수 있는 만큼 오버피팅을 억제하는 기술이 중요해지는 것입니다.

1.4.1 가중치 감소

오버피팅 억제용으로 예로부터 많이 이용해온 방법 중 가중치 감소(weight decay)라는 것이 있습니다. 이는 학습 과정에서 큰 가중치에 대해서는 그에 상응하는 큰 페널티를 부과하여 오버피팅을 억제하는 방법입니다. 원래 오버피팅은 가중치 매개변수의 값이 커서 발생하는 경우가 많기 때문입니다.

가장 일반적인 regulization 기법인 L2 Regularization을 소개하겠습니다. 기존 손실함수(L_{old})에 모든 학습파라메터의 제곱을 더한 식을 새로운 손실함수(L_{new})로 씁니다. 아래 식과 같습니다. 여기에서 1/2이 붙은 것은 미분편의성을 고려한 것이고, λ는 패널티의 세기를 결정하는 사용자 지정 하이퍼파라메터입니다. 이 기법은 큰 값이 많이 존재하는 가중치에 제약을 주고 가중치 값을 가능한 널리 퍼지도록 하는 효과를 내어 오버피팅을 억제합니다.

$$W = [\, w_1 \quad w_2 \quad \dots \quad w_n \,]$$

$$L_{new} = L_{old} + \frac{\lambda}{2}(w_1^2 + w_2^2 + \dots + w_n^2)$$

1.4.1 드롭아웃(Dropout)

신경망 모델이 복잡해지면 가중치 감소만으로 오버피팅에 대응하기 어려워집니다. 이럴 때 흔히 드롭아웃이라는 기법을 이용합니다.

드롭아웃은 뉴런을 임의로 삭제하면서 학습하는 방법입니다. 훈련 때 은닉층의 뉴런을 무작위로 골라 삭제합니다. 그림 (a)와 같이 모든 layer에 대해 학습을 수행하는 것이 아니라 그림 (b)와 같이 망에 있는 입력 layer나 hidden layer의 일부 뉴런을 생략(dropout)하고 줄어든 신경망을 통해 학습을 수행합니다. 일정한 미니배치 구간 동안 생략된 망에 대한 학습을 끝내면, 다시 무작위로 다른 뉴런들을 생략하면서 반복적으로 학습을 수행합니다.

dropout이 오버피팅에 효과적인 이유는 다음과 같이 알려져 있습니다.

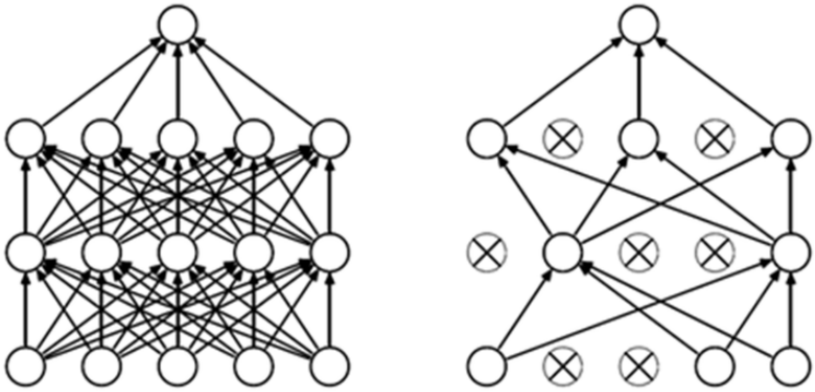
1) Voting효과

일정한 mini-batch 구간 동안 줄어든 망을 이용해 학습을 하게 되면, 그 망은 그 망 나름대로 overfitting이 되며, 다른 mini-batch 구간 동안 다른 망에 대해 학습을 하게 되면 그 망에 대해 다시 일정 정도 overfitting이 됩니다.

이런 과정을 무작위로 반복을 하게 되면, voting에 의한 평균 효과를 얻을 수 있기 때문에, 결과적으로 앙상블 학습의 효과를 내게 됩니다.

2) Co-adaptation(상호동조)을 피하는 효과

Dropout은 연결을 계속 끊어 뉴런들이 비슷한 특징에 집중해 버리는 상호동조를 예방하는 의미도 있습니다. 특정 뉴런의 바이어스나 가중치가 큰 값을 갖게 되면 그것의 영향이 커지면서 다른 뉴런들의 학습 속도가 느려지거나 학습이 제대로 진행이 되지 못하게 되는 경우가 발생합니다. 하지만 dropout을 하면서 학습을 하게 되면, 해당 뉴런이 켜지지 않은 상태에서도 학습이 진행되기 때문에 특정 뉴런의 영향을 줄일 수 있어 뉴런들이 서로 동조화 되는 것을 피할 수 있습니다.



(a) Standard Neural Net (b) After applying dropout.