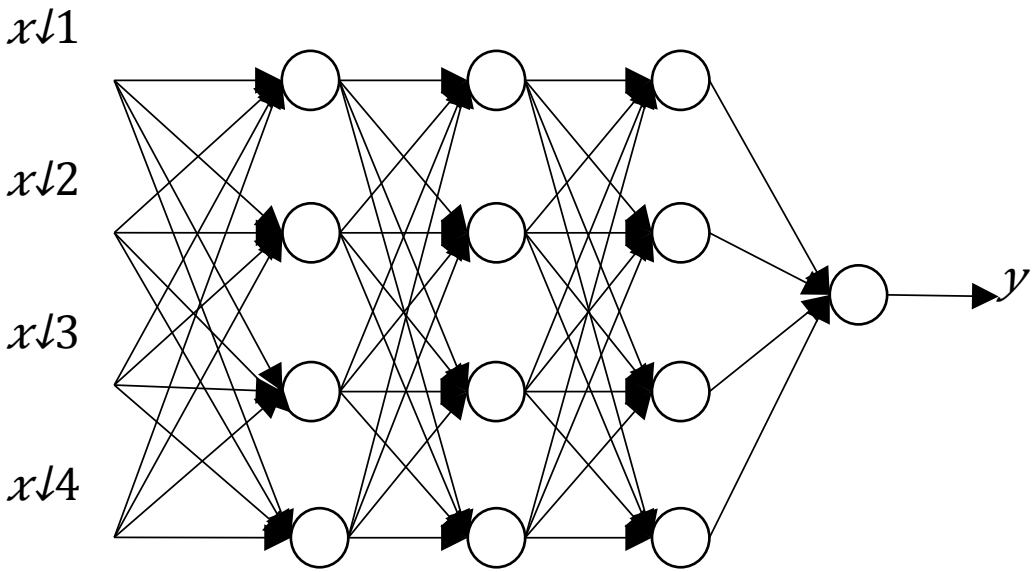
deeplearning.ai

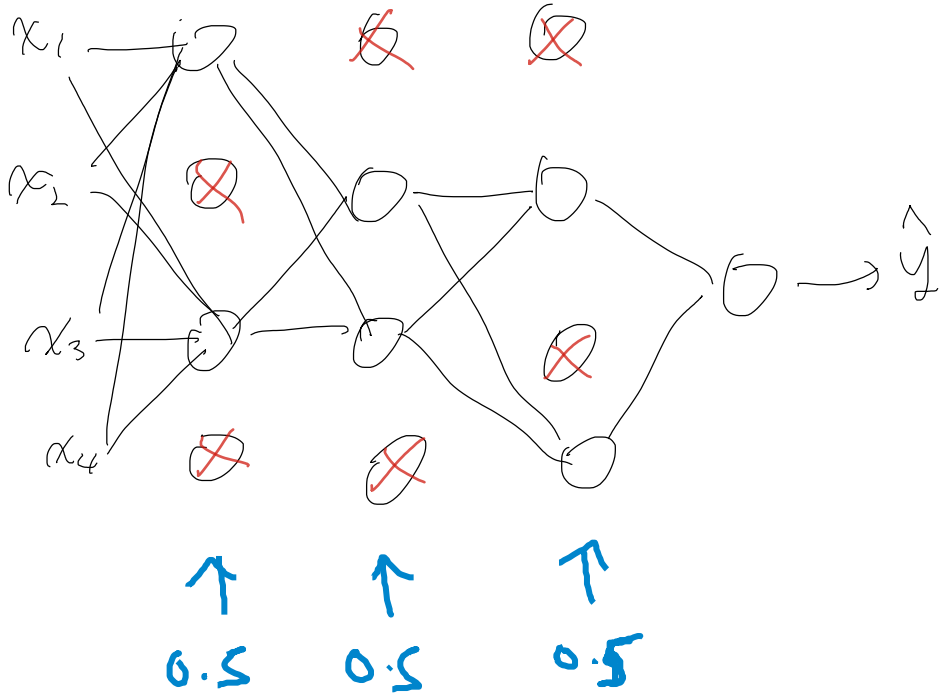
# Regularizing your neural network

# Dropout regularization

# Dropout regularization




특정 확률로 노드들을 제거하고(ingoing/outgoing 에지를 제거) 훨씬 작은 네트워크를 얻는다. 트레이닝에서는 이 작은 네트워크에서 백프로퍼게이션 하는 것이다. 이걸 매 training example마다 수행하면 학습할 때마다 이런 줄어든 NN 중에 하나로 학습하는 셈이 되는 것이다.




그래서 직관적으로 생각해보면, 각 example에 대해 매번 작은 네트워크로 학습시키니까 오버피팅이 줄어드는 것이라고 볼 수 있다.

Andrew Ng

가장 일반적인 구현법을 소개한다.

# Implementing dropout ("Inverted dropout")

Illustrate with layer $l = 3$. keep-prob = 0.8 (0.2)

dropout vector

$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\_prob$

20%를 제거하겠다는 의미

$a3 = np.multiply(a3, d3)$ # $a3 *= d3$.

$a3 /= keep\_prob$

이게 무슨 의미인지 설명:

해당 레이어에 50개의 뉴런이 있다고 가정해보자

50 units ⇝ 10 units shut off (왜냐하면 20% 제거하니까)

$z^{[4]} = w^{[4]} \cdot a^{[3]} + b^{[4]}$

z는 이렇게 계산되는데 여기에 사용되는 a가 20% 줄어든 상태인 셈이다.

$a^{[3]}$ reduced by 20%.

그래서 z에 대해 기대하는 값이 줄어들지 않게 하기위해 0.8로 나눠주는 것이다.

reduced by

$/= 0.8$

Test

결론은... different training example에 대해 다른 히든 유닛을 zero out하므로, 같은 트레이닝셋에 대해 여러번 학습을 시키면 랜덤하게 다른 히든 유닛을 제거하게 된다.

Andrew Ng

# Making predictions at test time

$a^{[0]} = X$

No drop out

드랍아웃 사용하지 않음 드랍아웃 사용하면 prediction에 노이즈만 추가하는 셈이다.

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

$a^{[1]} = g^{[1]}(z^{[1]})$

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

$a^{[3]} = \ldots$

$\hat{y}$

$/= keep\text{-}prob$

Andrew Ng

deeplearning.ai

# Regularizing your neural network

# Understanding dropout

드랍아웃이 대체 왜 regularizer로 잘 동작하는걸까?

# Why does drop-out work?

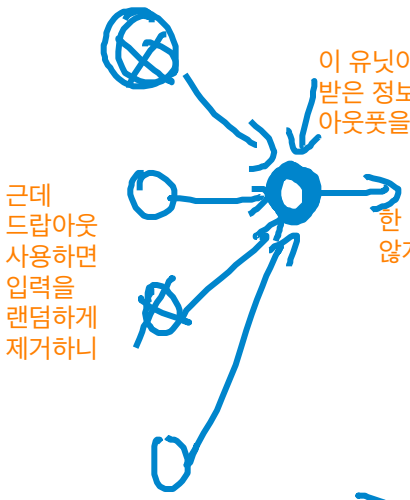
앞에서 봤던 것처럼 첫번째 intuition은 매번 작은 네트워크로 학습시키니까 regularizing effect를 갖게 되는 것이고
두번째 intuition은... 아래에서...

## Intuition: Can't rely on any one feature, so have to spread out weights.

→ Shrink weights. $L_2$

고로 L2 랑 비슷한 효과를 낼 수 있다.

weights를 spread하는 것은 weight를 shrink하는 효과가 있고



이 유닛이 할 일은 입력으로 받은 정보를 가지고 의미있는 아웃풋을 내보내는 것이다.

근데 드랍아웃 사용하면 입력을 랜덤하게 제거하니

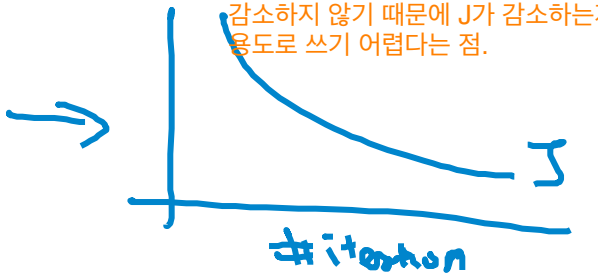
→ Computer Vision

컴퓨터비전 쪽에서 초기에 드랍아웃으로 효과 많이 봤는데 입력이 워낙 커서이다. 이미지 데이터의 경우 충분한 데이터를 갖기 힘들고 거의 항상 오버피팅 되기 때문에 드랍아웃 유용하다.

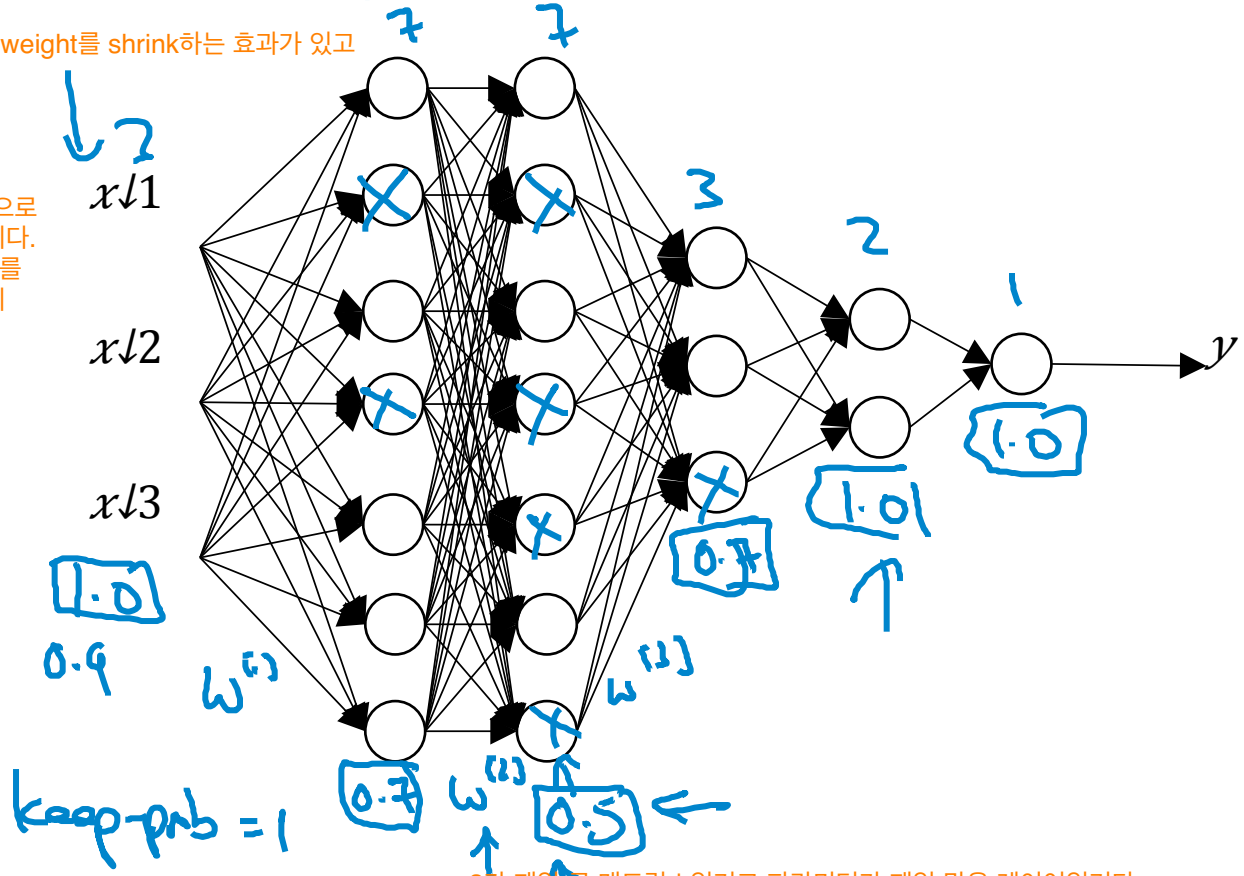
한 입력 피쳐에 의존적이지 않게 되는 효과가 있다.

Downside: J

드랍아웃 단점은 cost function J가 잘 정의되지 않는다는 점이다. 그래서 그래프가 아래처럼 예쁘게 감소하지 않기 때문에 J가 감소하는지 디버깅하는 용도로 쓰기 어렵다는 점.




응형은 그냥 keep_prob 1로 돌려보고 J가 잘 감소하면 그때 드랍아웃 적용하는 식으로 해결한다고 함.




w2가 제일 큰 매트릭스일거고 파라미터가 제일 많은 레이어일거다. 고로 이 레이어의 keep.prob을 상대적으로 작게 줘야한다. (0.5)

Andrew Ng