deeplearning.ai

Regularizing your neural network

Regularization

오버피팅을 줄이는데 사용할 수 있는 방법에는 데이터셋을 늘리는 게 있지만 어떤 경우에는 이 옵션이 비용이 매우 비싸거나 불가능할 수도 있는 반면 regularization의 경우에는 오버피팅을 줄이는데(또는 variance를 줄이는데) 일반적으로 도움이 된다.

Logistic regression

우선 LR에서 regularization을 살펴보자

$w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

$\min_{w,b} J(w,b)$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\|w\|_2^2$$

트레이닝 데이터에서 잘 자동하는 것과 파라미터의 norm을 작게하는것 사이의 트레이드오프를 조절하는 하이퍼파라미터로 작용한다.

λ = regularization parameter

lambda lambd

high variance 문제는 파라미터가 너무 많을 때 발생하고 w는 매우 high dimensional parameter vector이다. 고로 w에 비해서 b는 파라미터의 수가 상대적으로 많이 적기 때문에 굳이 이 term을 넣지

L_2 regularization : $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$ ←

L_1 regularization : $\frac{\lambda}{2m}\sum_{j=1}^{n_x}|w_j| = \frac{\lambda}{2m}\|w\|_1$

L1 regularization 사용하면 w vector에 0 값이 많아질 것이다

w will be sparse

L1/L2 regularization 차이점 :
- http://playdata.io/question/질문드립니다-5/
- https://brunch.co.kr/@itschloe1/11

Andrew Ng

Neural network

그럼 NN에서는 어떨까?

$$J(w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}} \sum_{j=1}^{n^{[l-1]}} (w_{ij}^{[l]})^2$$

$w^{[l]}$: $(n^{[l]}, n^{[l-1]})$

"Frobenius norm" of the matrix

$\|\cdot\|_2^2 \rightarrow \|\cdot\|_F^2$

개념은 L2 norm이랑 똑같은데 엄밀하게는 L2 norm이라고 안부르고 Frobenius norm이라고 부른다.

$$dw^{[l]} = (\text{from backprop}) + \frac{\lambda}{m} w^{[l]}$$

미분

$$\frac{\partial J}{\partial w^{[l]}} = dw^{[l]}$$

$$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$$

L2 norm에 대한 term을 추가해도 일반적인 GD 식 그대로 사용 가능

요기다 삽입하면

$$w^{[l]} := w^{[l]} - \alpha\left[(\text{from backprop}) + \frac{\lambda}{m} w^{[l]}\right]$$

$$= w^{[l]} - \frac{\alpha\lambda}{m} w^{[l]} - \alpha\,(\text{from backprop})$$

$$= \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha\,(\text{from backprop})$$

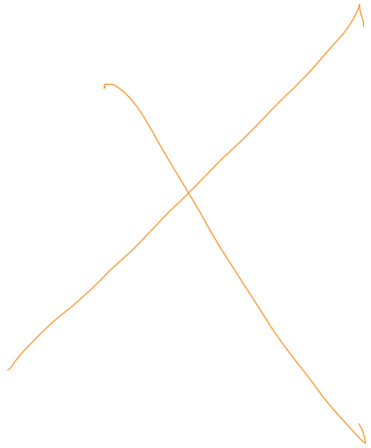
< 1

"Weight decay"

Andrew Ng

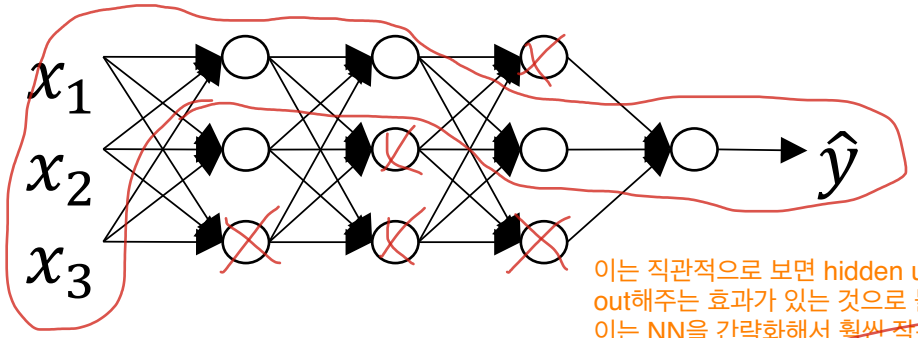
Neural network

$$J(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^{L} \|w^{[l]}\|^2$$



즉, 왜 variance problem을 줄이는데 도움이 되냐?

How does regularization prevent overfitting?

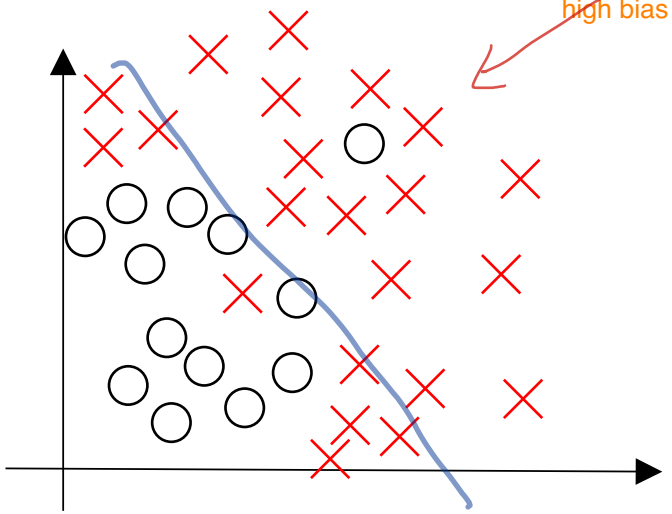


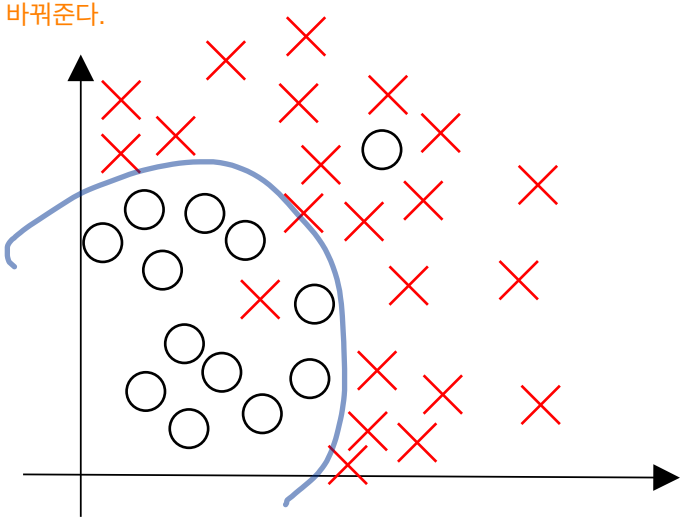
$$J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i}^{m} L(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$

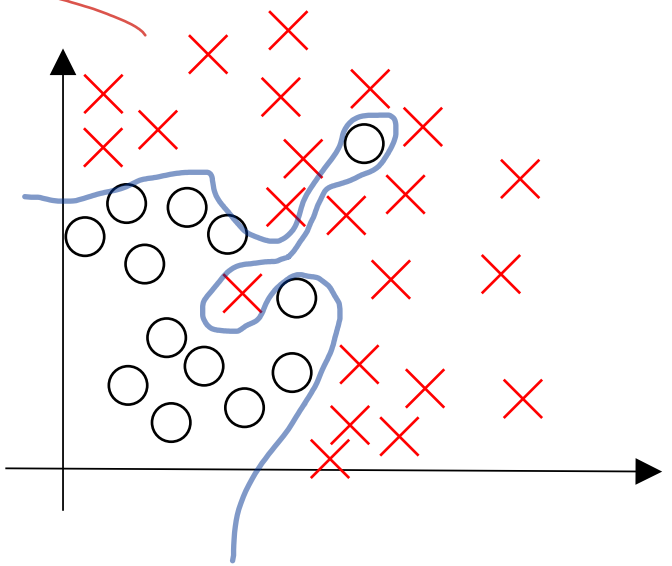
$$w^{[l]} \approx 0$$

이 term이 왜 오버피팅을 막아주냐면 lambda를 엄청 크게하면 w를 0에 가깝게 해주는 효과가 있다.

이는 직관적으로 보면 hidden unit을 zero out해주는 효과가 있는 것으로 볼 수 있다. 이는 NN을 간략화해서 훨씬 작은 NN, 마치 logistic regression unit을 쌓아놓은 것처럼 되어 버리고 이는 overfitting case를 왼쪽의 high bias case에 가깝게 바꿔준다.





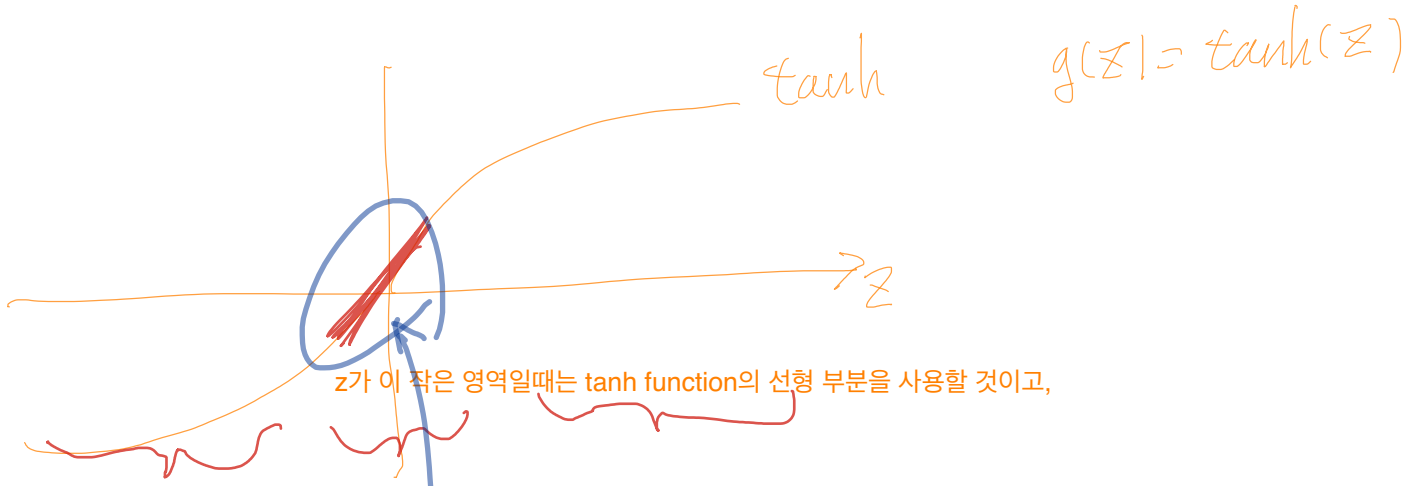


이 얘기는 직관적으로 그렇단 얘기고 실제로는 모든 히든 유닛을 사용하긴 하지만 그 효과를 훨씬 줄여서 smaller network처럼 만든다.

Andrew Ng

어디셔널 인튜이션을 소개해줄게

How does regularization prevent overfitting?



z가 이 작은 영역일때는 tanh function의 선형 부분을 사용할 것이고,

z가 많이 작거나 클때는 activation이 점점 덜 linear해진다.

$\lambda \uparrow$ 커지면 $W^{[l]} \downarrow$ 작아지고 $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$

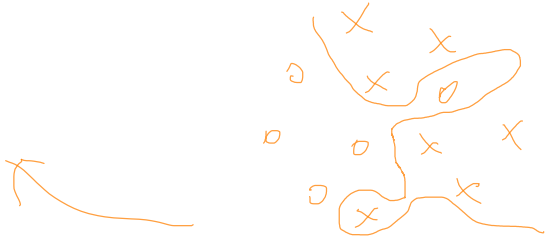
z도 작아지니까 g(z)가 저 영역이 되어서 roughly linear해질것이다.

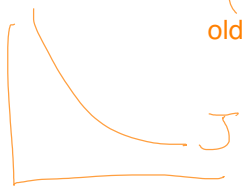
$\Rightarrow$ Every layer will be roughly linear ($\approx$ linear regression)

$$J(\dots) = \boxed{\sum \mathcal{L}(\hat{y}, y)} + \frac{\lambda}{2m} \sum \|w\|_F^2$$

old term new term

예전에 배운것처럼 linear activation을 쓰면 아무리 레이어를 많이 쌓아도 전체적으로 보면 커다란 linear function처럼 동작하게 되기 때문에 high variance case를 선형에 가까운 방향으로 해소시켜준다.





구현시 중요한 점은, J의 definition이 바뀌었으니 옛날처럼 old term을 그려보면 monotonically 감소하지 않을 것이다. new term을 그려야 monotonically 감소한다.

Andrew Ng