deeplearning.ai

Regularizing your neural network

Regularization

Logistic regression

Why Regularization?
↳ to ↓ overfitting

- Getting more data can also ↓ overfitting

$$\min_{w,b} J(w,b) \rightarrow w \in \mathbb{R}^{n_x},\ b \in \mathbb{R}$$

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

L2 Regularization

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

Q Why not regularize "b" as well? $\rightarrow \frac{\lambda}{2m} b^2$

well you're doing this to ↓ overfitting of your params
params are w, b
$\dim(w) = (n_x \times 1)$
$\dim(b) = (1 \times 1)$
⇒ the overfitting would likely be happening due to w & not b [more params to tweak in w]

L1 Regularization

This is when we add $\frac{\lambda}{2m}\|w\|_1$

ie, $\frac{\lambda}{m}\sum_{j=1}^{n_x}|w_j|$ to the eqn

If L_1 is used, w ends up being sparse ⇒ many 0s in w
⇒ helps in storage of params as most values are 0s, so you need to store only non-zeros & their positions

$\boxed{L_2 \text{ is used more often than } L_1}$

λ = Regularization param
↳ find value via dev set

Neural network

$$J(w^{[1]}, b^{[1]}, \dots, \underbrace{w^{[L]}}_{\text{\# layers in NN}}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{L} \underline{\underline{\|w^{[j]}\|_2^2}}$$

for each jth layer → $$\underline{\underline{\|W^{[j]}\|^2}} = \sum_{i=1}^{n^{[L-1]}} \sum_{k=1}^{n^{[L]}} (w_{ik}^{[j]})^2$$ } For each w at layer j, find Σ of s_q, then do for all layers

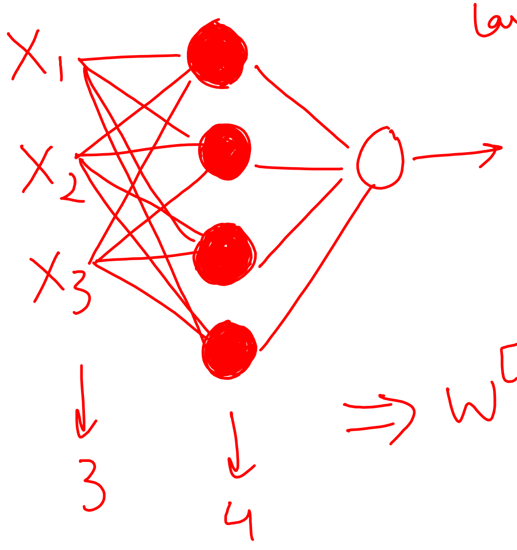
aka "Forbenious Norm"

↳ why are the limits $n^{[L]}$ & $n^{[L-1]}$

Remember dim(w) from before : $(n^{[L]}, n^{[L-1]})$

$n^{[L]}$ → # hidden units in current layer

$n^{[L-1]}$ → # hidden units in prev layer



$\Rightarrow W^{[1]} = (4 \times 3)$

So now how does Back prop change w/ the Addition of Regularization ?

$$\frac{\partial J}{\partial w^{[L]}} = ?$$

Now, $\underline{\underline{dw^{[L]}}} =$ old dw $+ \frac{\lambda}{m} w^{[L]}$

update

$$w^{[L]} = w^{[L]} - \alpha \cdot \underline{\underline{dw^{[L]}}}$$

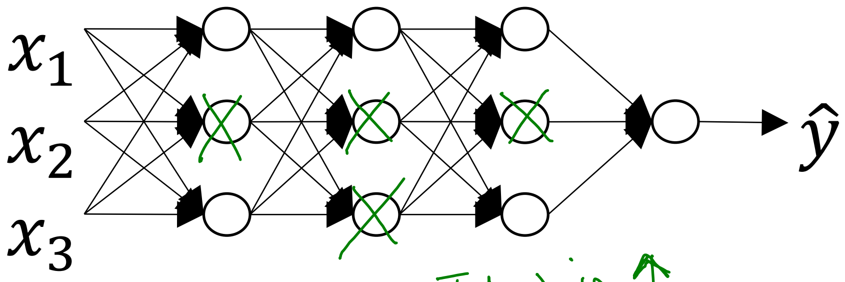
The new $dw^{[L]}$ makes the update further ↓ by $\frac{\alpha\lambda}{m} w^{[L]}$

∴ we call gradient descent w/ Regularization "WEIGHT DECAY"

Andrew Ng

How does regularization prevent overfitting?

$$J\underbrace{(w^{[l]}, b^{[l]})}_{\text{All layers}} = \frac{1}{m}\sum L(y^{(i)}, \hat{y}^{(i)}) + \underbrace{\frac{\lambda}{2m}\sum_{i=1}^{L}\|w^{[l]}\|_F^2}_{\text{why does this} \downarrow \text{ overfitting?}}$$

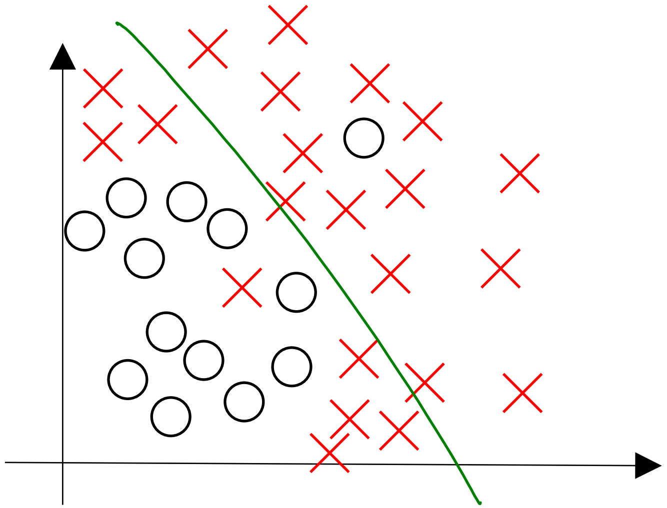


If λ is $\uparrow$

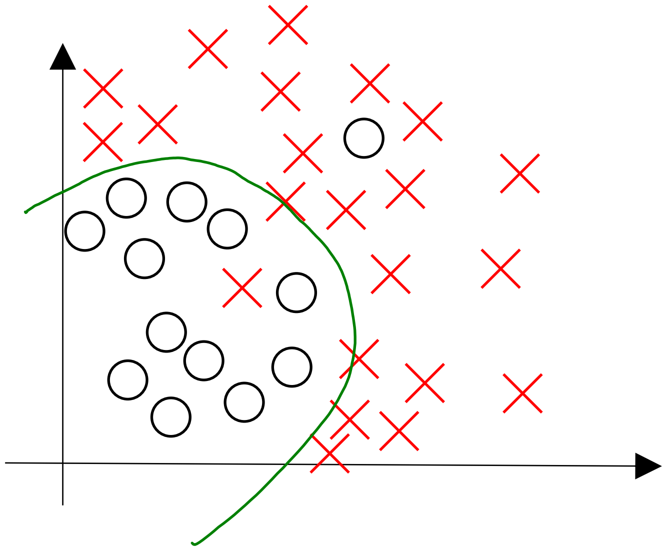
If λ is high, then many weights w in diff layers become close to 0 to $\downarrow$ cost func $\Rightarrow$ simpler model

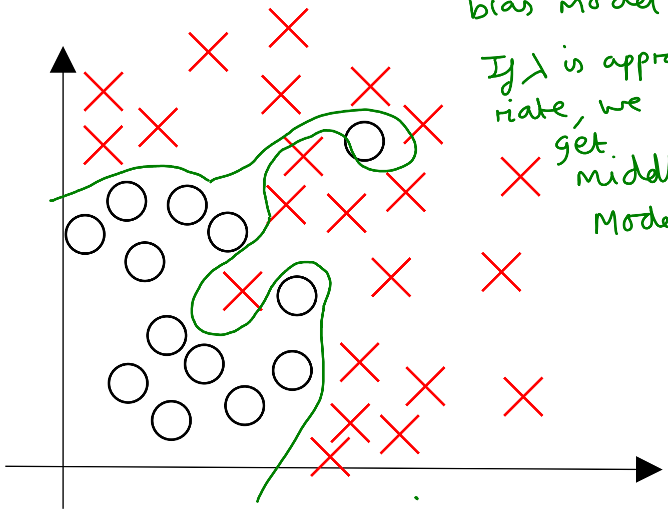
If λ too high, we get high bias model

If λ is appropriate, we get middle model



High bias



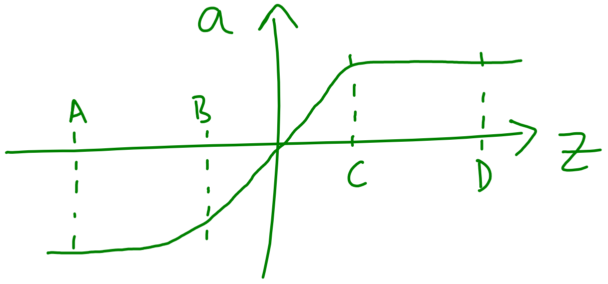


High variance

Andrew Ng

How does regularization prevent overfitting?

Assume, we use tanh Activation func



$g(z) = a$
$g(z) = \tanh(z)$

Now If we $\uparrow \lambda$, $w^{[l]} \downarrow$

If $w^{[l]} \downarrow$ $z^{[l]} \downarrow$, why? $z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$

$\Rightarrow z \propto w$

If $z \downarrow$ and lets assume z goes from being in Range $C \to D$ or Range $A \to B$ to Range $B \to C$

then we can see derivative of z "dz" is linear, $\Rightarrow$ every layer is computing a linear eqn $\Rightarrow$ model has become simpler

$g'(z)$

$\Rightarrow$ If $\lambda \uparrow$, we get simpler model

Andrew Ng