deeplearning.ai

Regularizing your neural network

Regularization

Logistic regression

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

λ = regularization parameter

lambda lambd

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

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

$$\cancel{+ \frac{\lambda}{2m} b^2}$$

omit

L2 regularization $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w \leftarrow$

L1 regularization $\frac{\lambda}{2m} \sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m} \|w\|_1$

w will be sparse

Andrew Ng

Neural network

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

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

"Frobenius norm" $\qquad \|\cdot\|_2^2 \qquad \|\cdot\|_F^2$

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

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

"Weight decay"

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

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

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

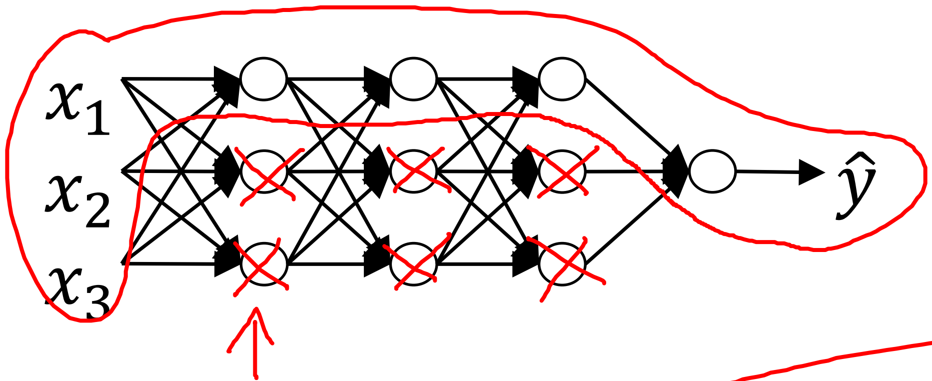
Andrew Ng

deeplearning.ai

Regularizing your neural network

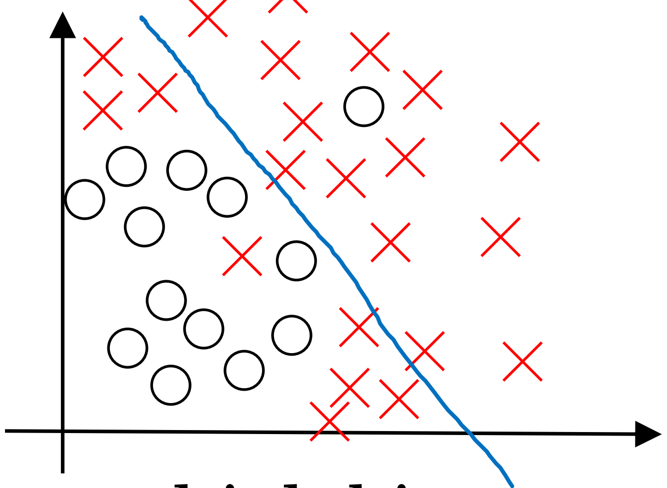
Why regularization reduces overfitting

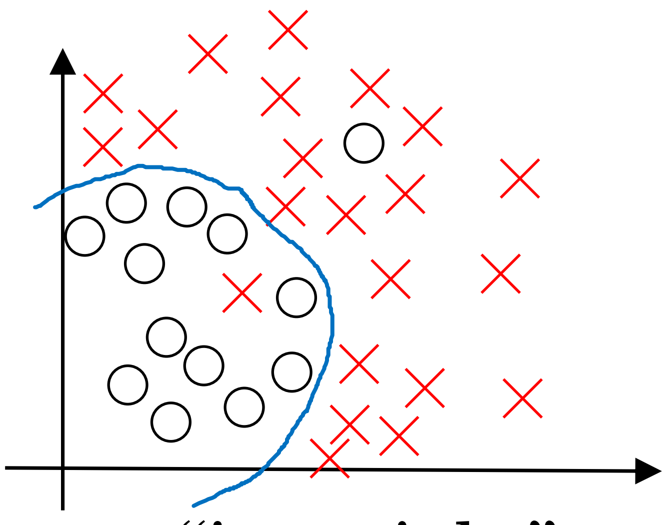
How does regularization prevent overfitting?

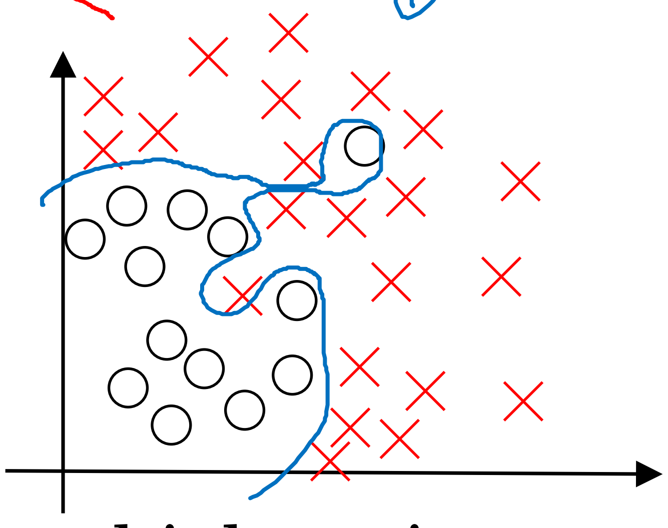


$$J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \underbrace{\|w^{[l]}\|_F^2}$$

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

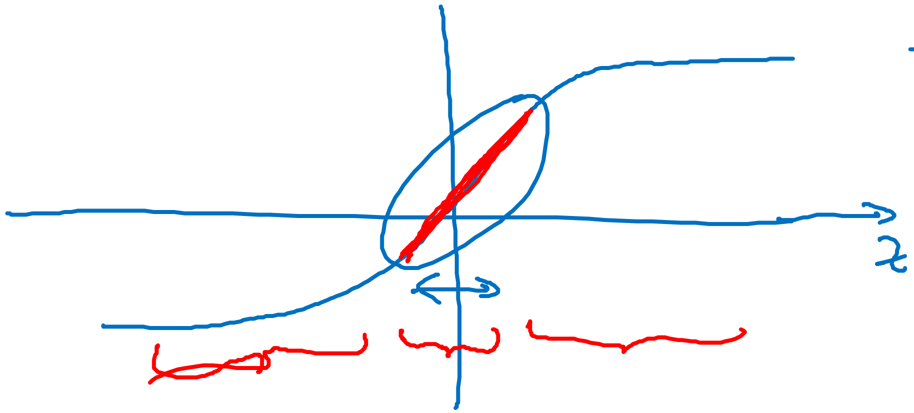

high bias


"just right"


high variance

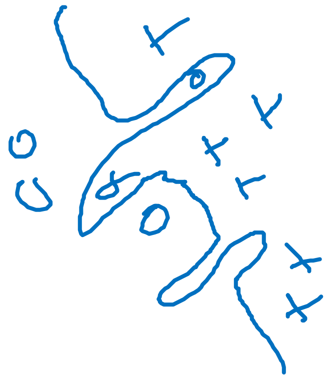
Andrew Ng

How does regularization prevent overfitting?



tanh

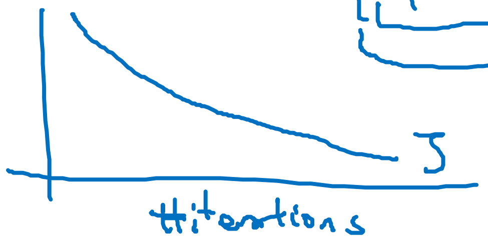
$g(z) = \tanh(z)$



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

Every layer $\approx$ linear.

$$J(\ldots) = \underbrace{\boxed{\sum_i \mathcal{L}(\hat{y}^{(i)}, y^{(i)})} + \frac{\lambda}{2m} \sum_l \|W^{[l]}\|_F^2}$$



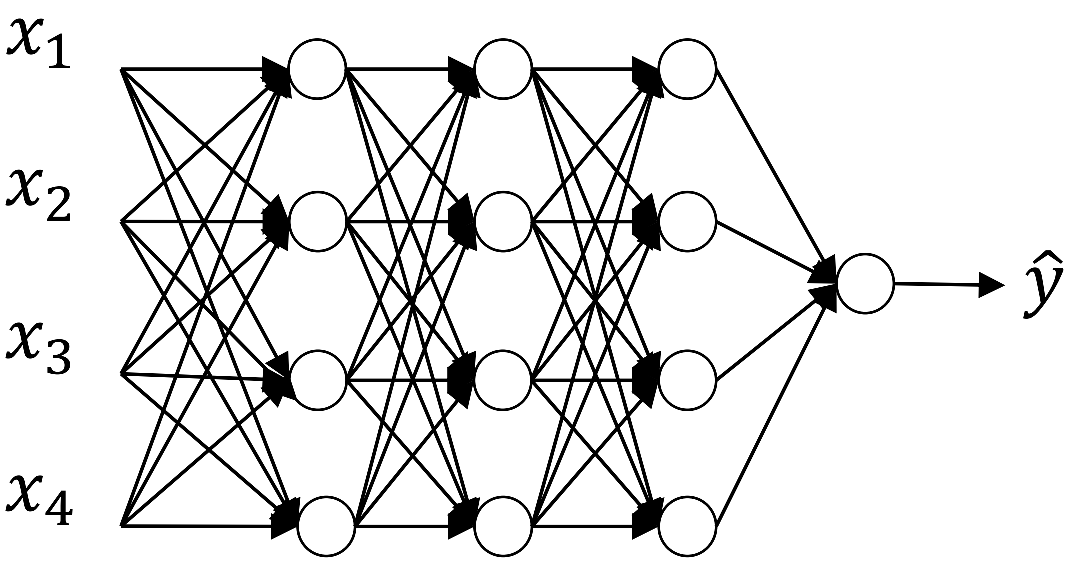
Andrew Ng

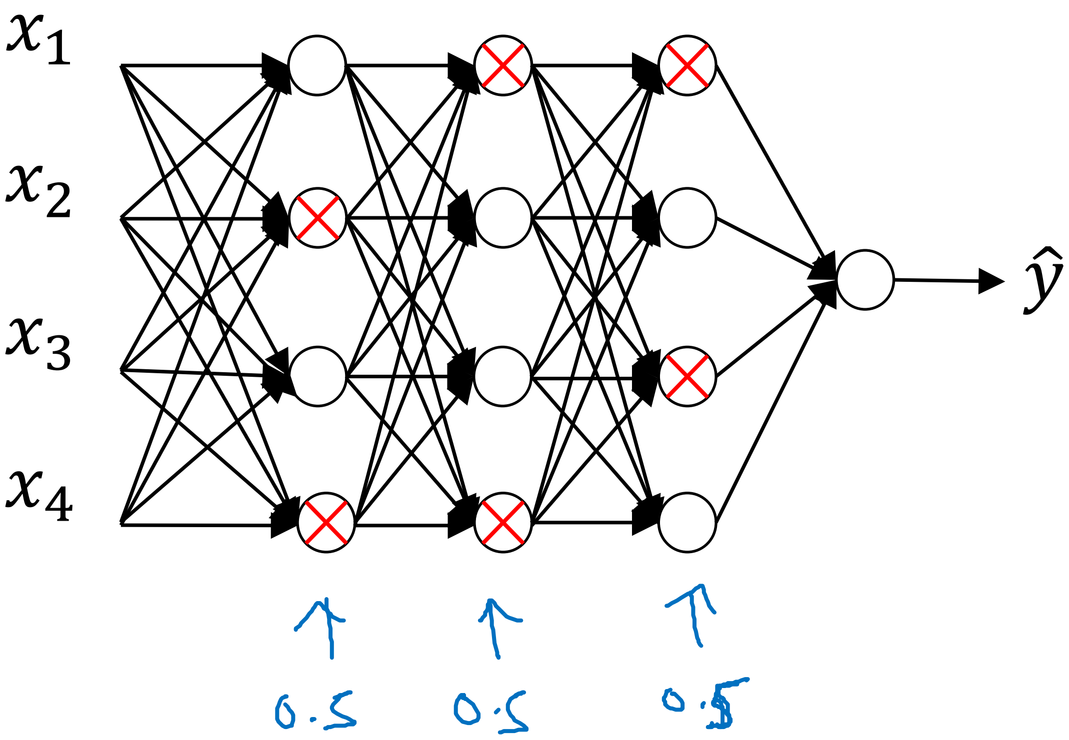
deeplearning.ai

Regularizing your neural network

Dropout regularization

Dropout regularization





Andrew Ng

Implementing dropout ("Inverted dropout")

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

→ $\boxed{d3}$ = np.random.rand(a3.shape[0], a3.shape[1]) < keep-prob

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

→ $\boxed{\text{a3 /= } \cancel{0.8} \text{ keep-prob}}$ ←

50 units. ⇝ 10 units shut off

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

reduced by 20%. Test

/= 0.8

Andrew Ng

Making predictions at test time

$a^{[0]} = X$

No drop out.

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

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

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

$a^{[2]} = \dots$

$\downarrow$

$\hat{y}$

/= keep-prob

Andrew Ng

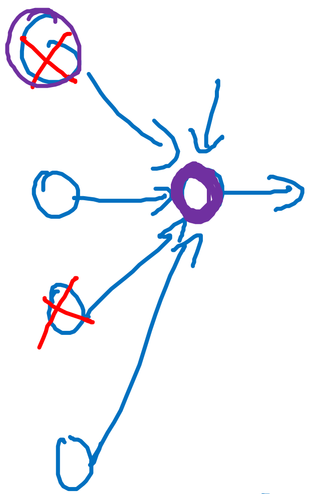
deeplearning.ai

Regularizing your neural network

Understanding dropout

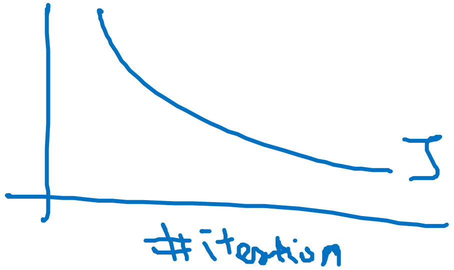
Why does drop-out work?

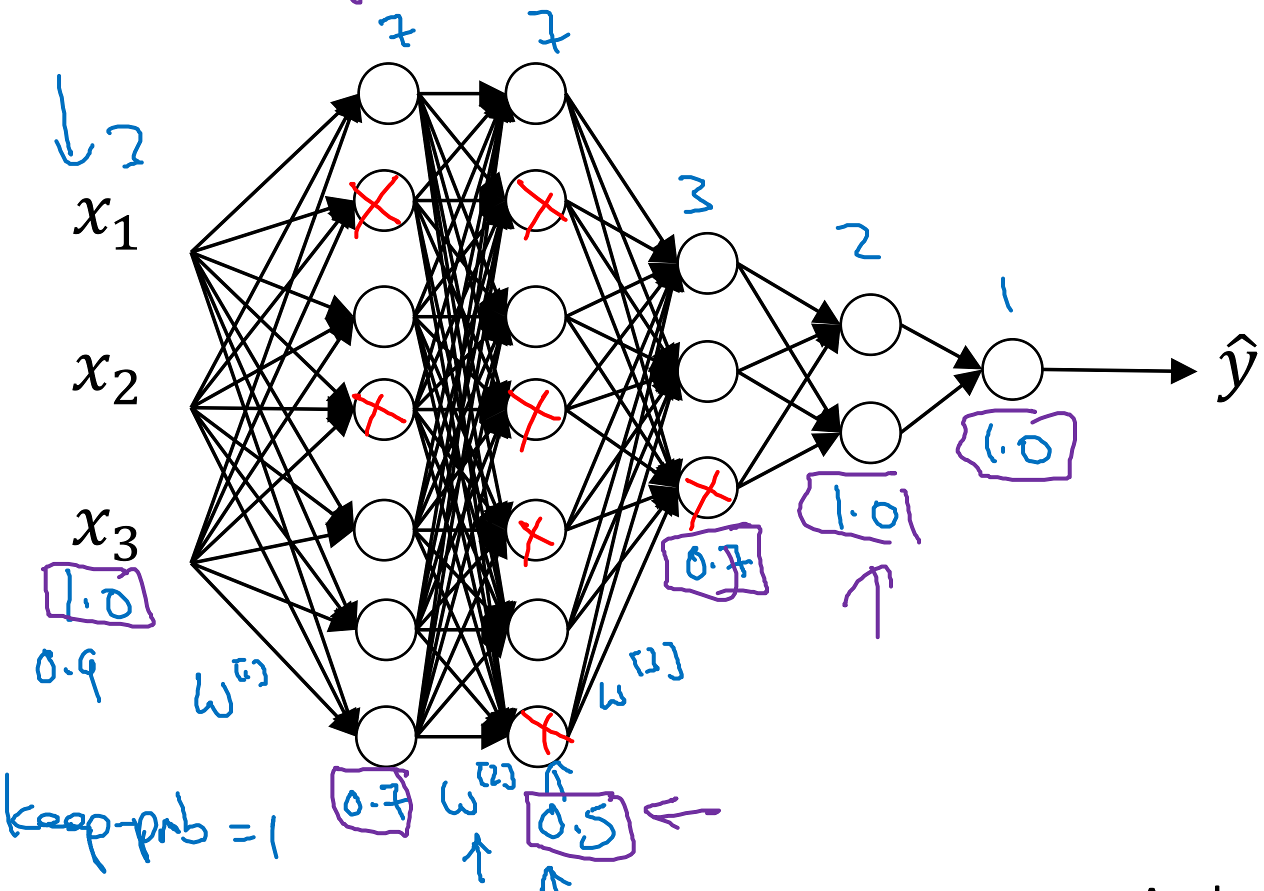
Intuition: Can't rely on any one feature, so have to spread out weights. ⤳ Shrink weights. L_2



→ Computer Vision

Downside: J



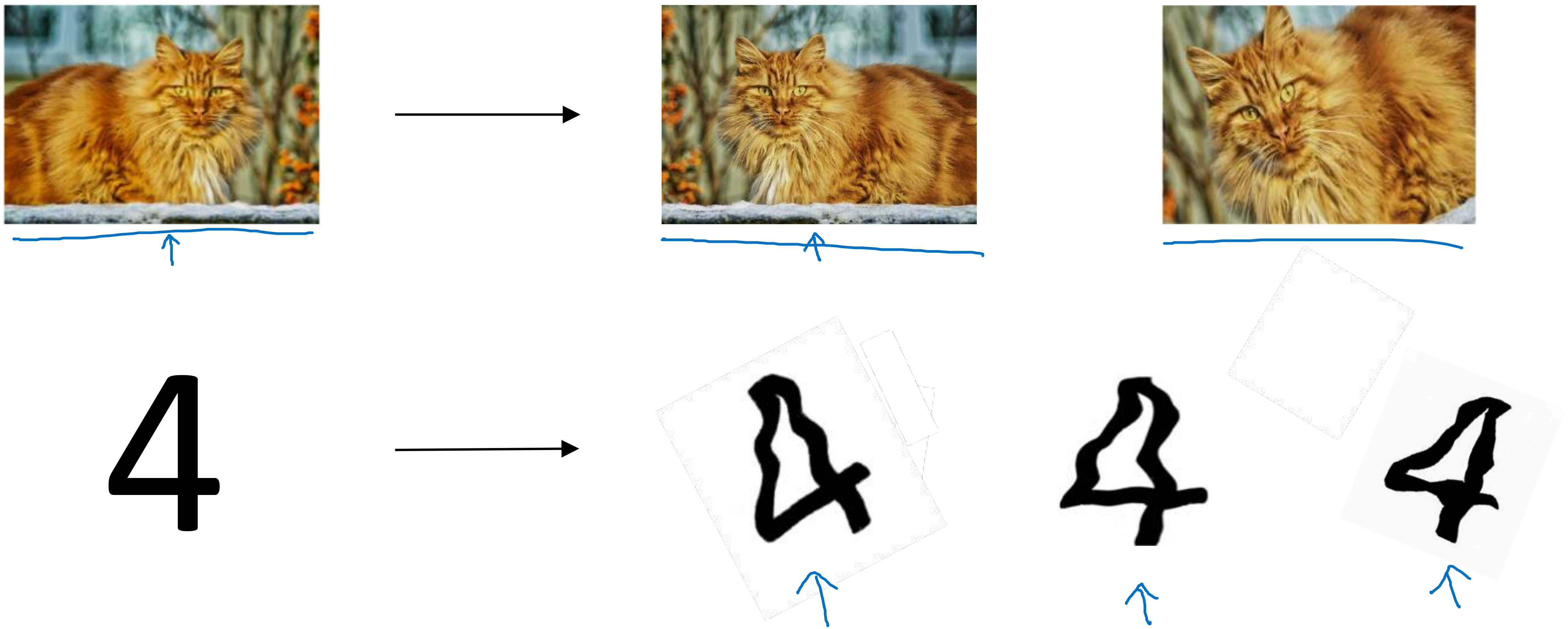


deeplearning.ai

Regularizing your neural network

Other regularization methods

Data augmentation



Andrew Ng

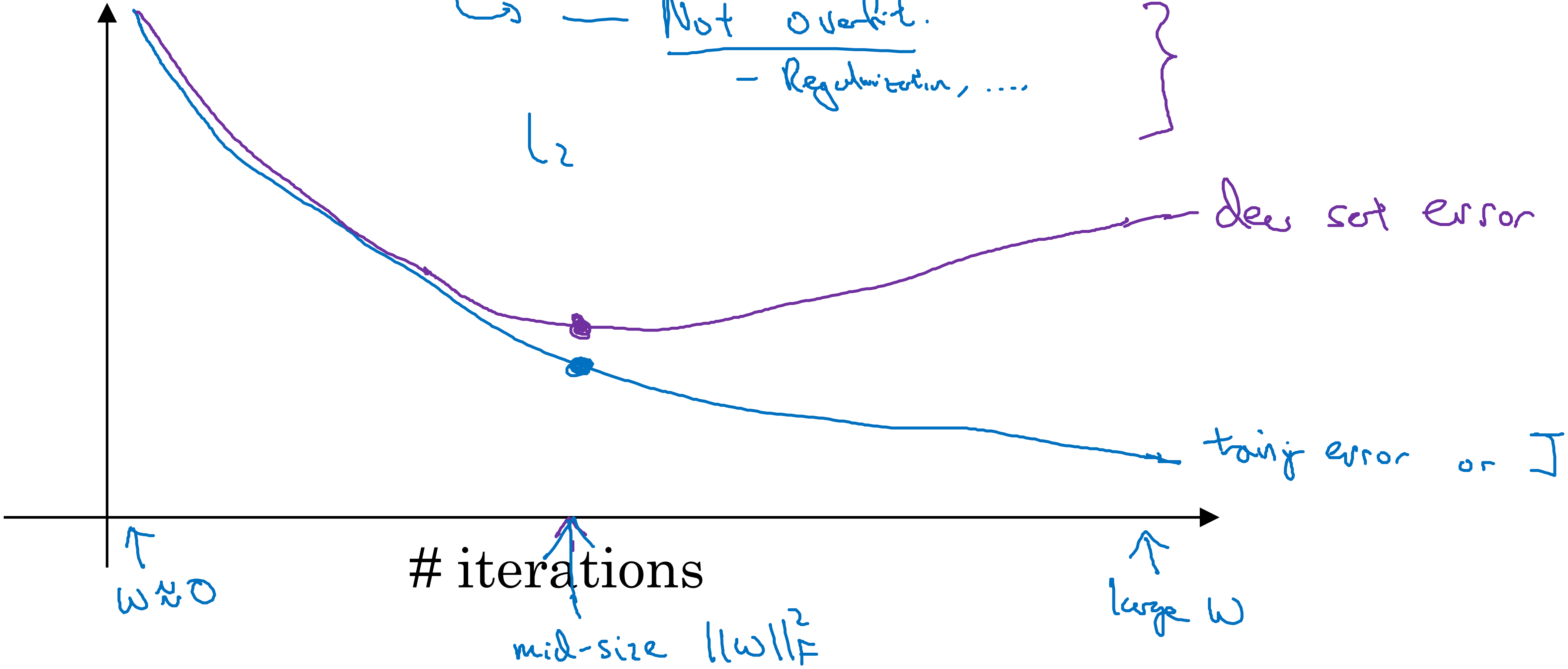
Early stopping

Orthogonalization.

- Optimize cost function J
  - Gradient, ...
- Not overfit.
  - Regularization, ...

L_2

$J(w,b)$



Andrew Ng