Computational Intelligence

Samaneh Hosseini

Isfahan University of Technology

Outline

- Neural Networks in Practice
  - Regularization
    - Why Regularization Reduces Overfitting?
  - Regularization Techniques
    - L2 Regularization
    - Dropout

Neural Networks in Practice: Regularization

Regularization

What is it?

Technique that constrains our optimization problem to discourage complex models

Regularization

What is it?

Technique that constrains our optimization problem to discourage complex models

Why do we need it?

Improve generalization of our model on unseen data

Regularization in Logistic regression

m : # training data

n_x : ابعاد بردار ورودی

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

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

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

w :

$$\lambda = \text{Regularization Parameter}$$

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

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

Regularization in Neural network

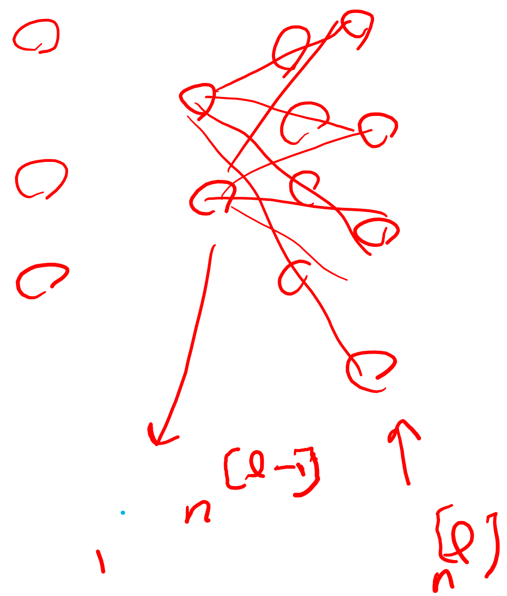
تعداد لایه‌ها

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

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

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

"Frobenius norm"



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

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

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

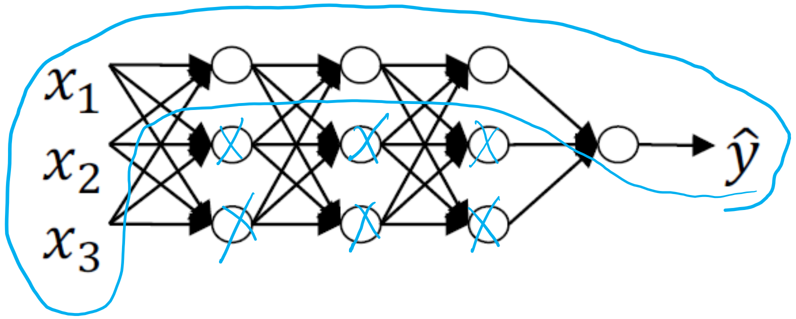
weight decay

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

Regularization:
Why Regularization Reduces Overfitting?

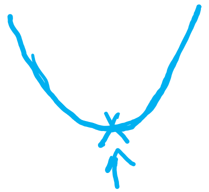
How does regularization prevent overfitting?

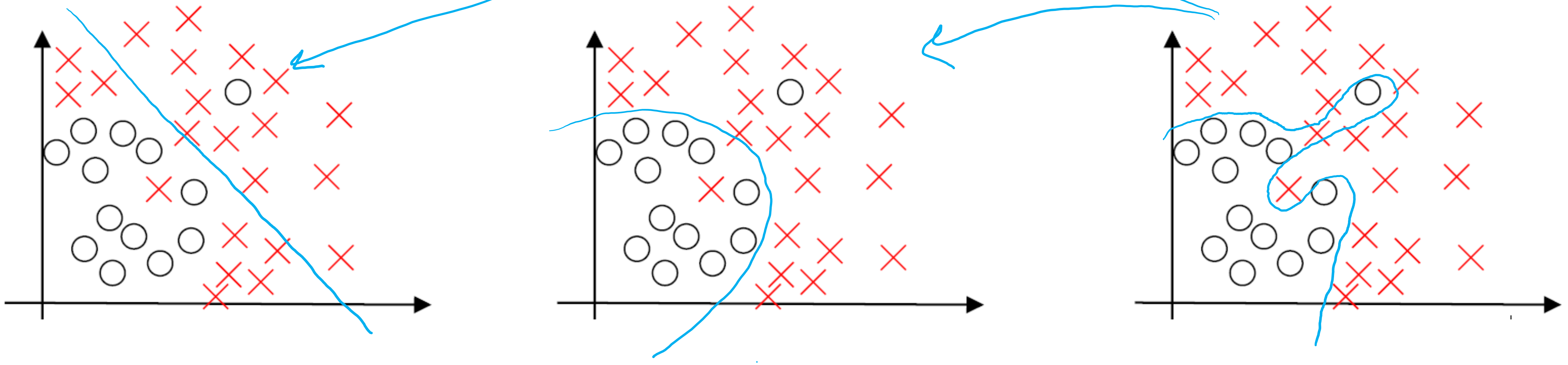
$$J(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} \|w^{[l]}\|_F^2$$



$\lambda \rightarrow$ زیاد

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



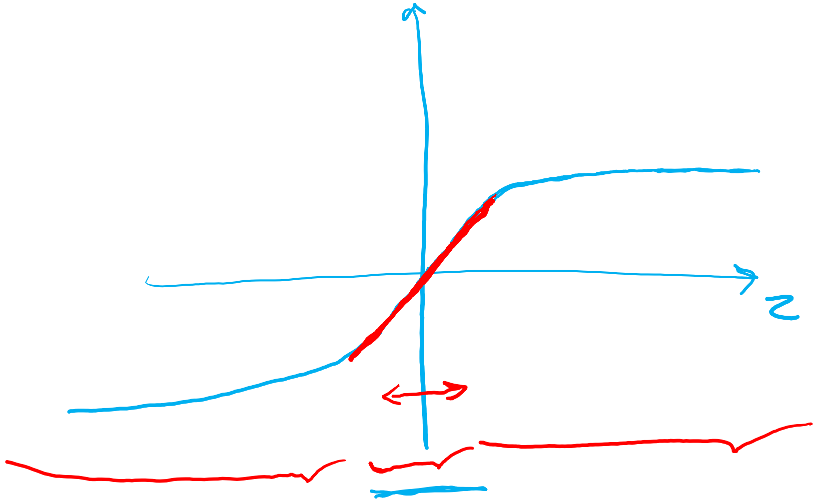


high bias

just right

high variance

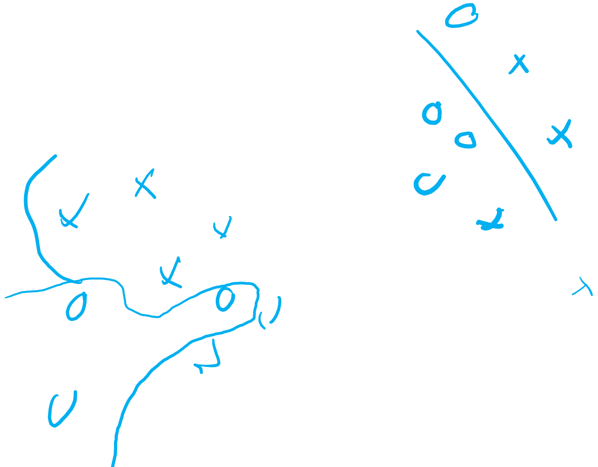
How does regularization prevent overfitting?



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

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

λ large $\simeq$ linear



Regularization Techniques:
II: Dropout

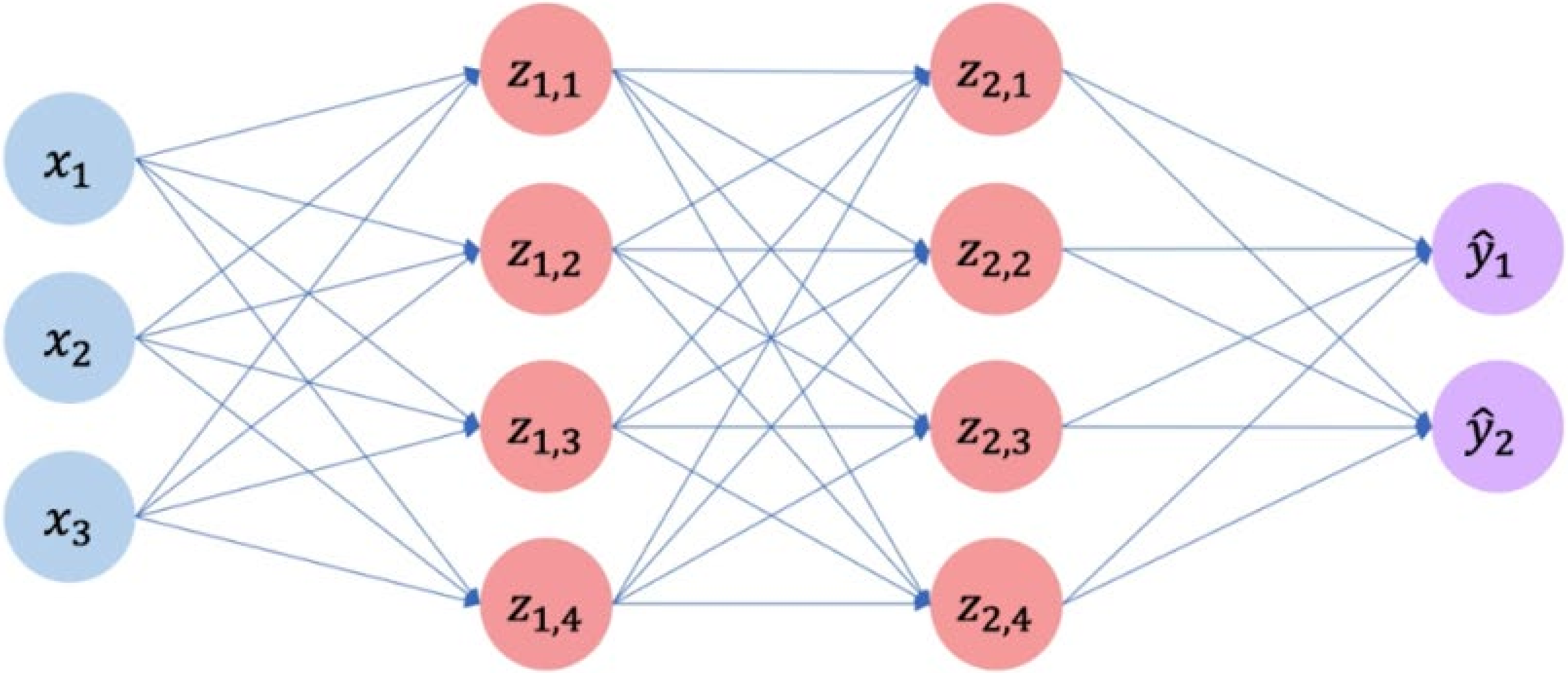
I : L_2 Regularization

$$\frac{\lambda}{2m}\sum_{i=1}^{l}\|w^{[l]}\|_F^2$$

II : Dropout

Regularization II: Dropout

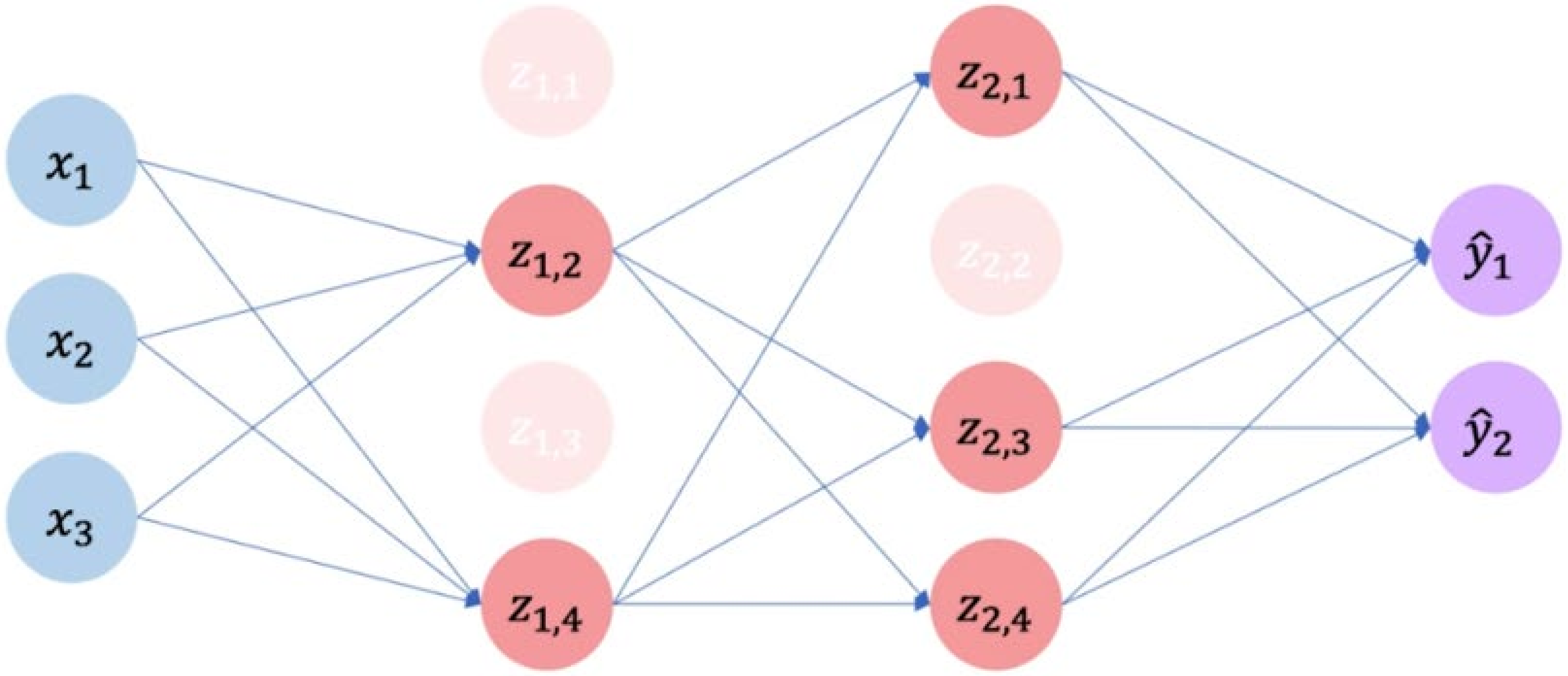
- During training, randomly set some activations to 0



Regularization II: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

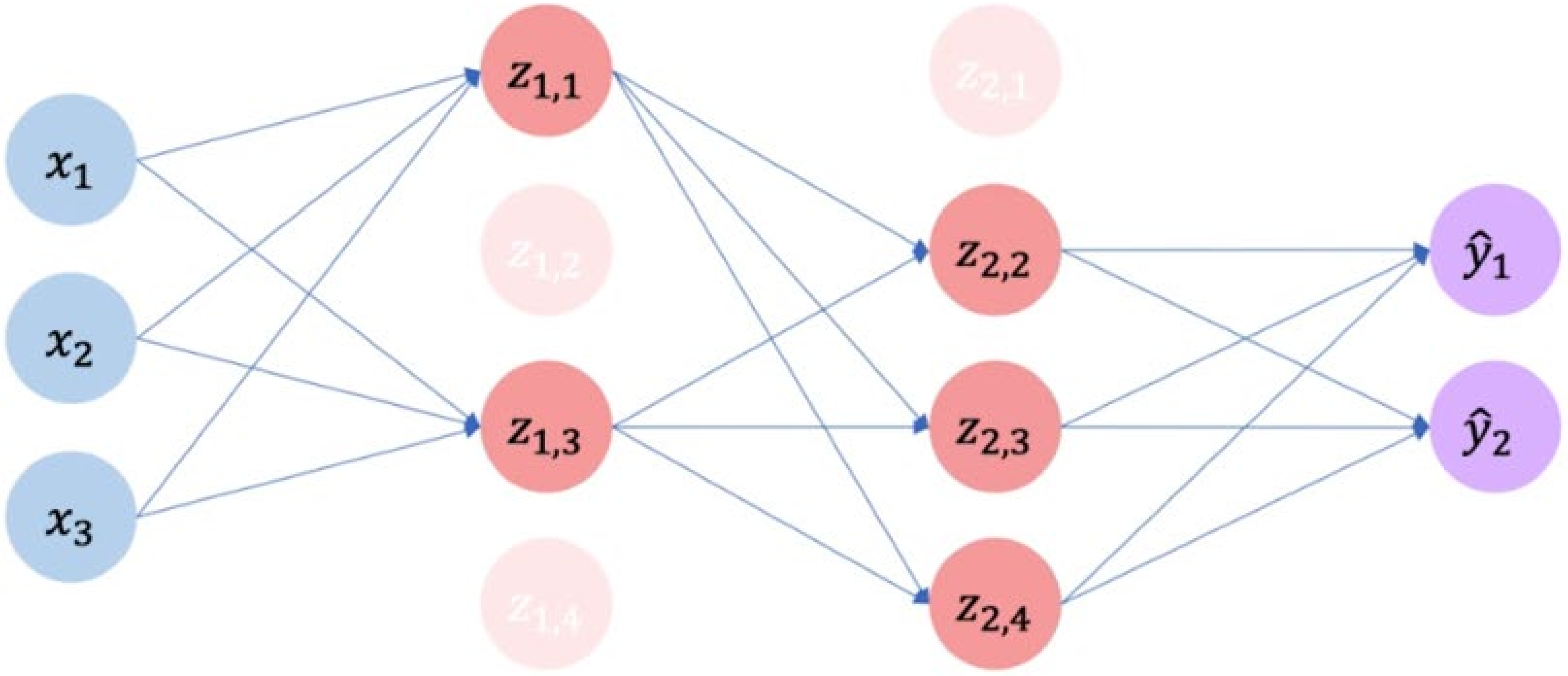
```
tf.keras.layers.Dropout(p=0.5)
```



Regularization II: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

```
tf.keras.layers.Dropout(p=0.5)
```



Implementing dropout ("Inverted dropout")

Illustrate with layer $l=3$ keep_prob $= 0.8$ 0.2

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

$a3 = np.multply(a3, d3)$

$\boxed{a3 \mathrel{/}= keep\_prob}$

50 unit $\longrightarrow$ 10 unit shut off

$$Z^{[4]} = w^{[4]} a^{[3]} + b^{[4]}$$

reduced 20%

$/= 0.8$

Making predictions at test time

$a^{[0]} = X$

No drop out

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

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

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

$a^{[2]} = g^{[2]}(z^{[2]})$

$\vdots$

$\downarrow$

$\hat{y}$

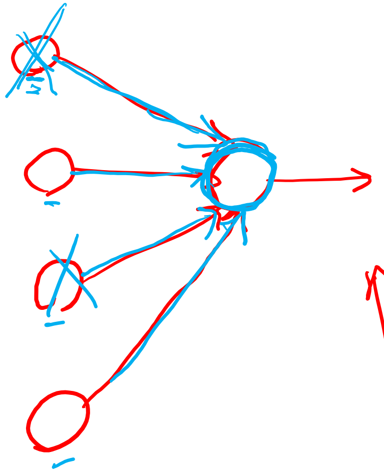
$/=$ keep_prob

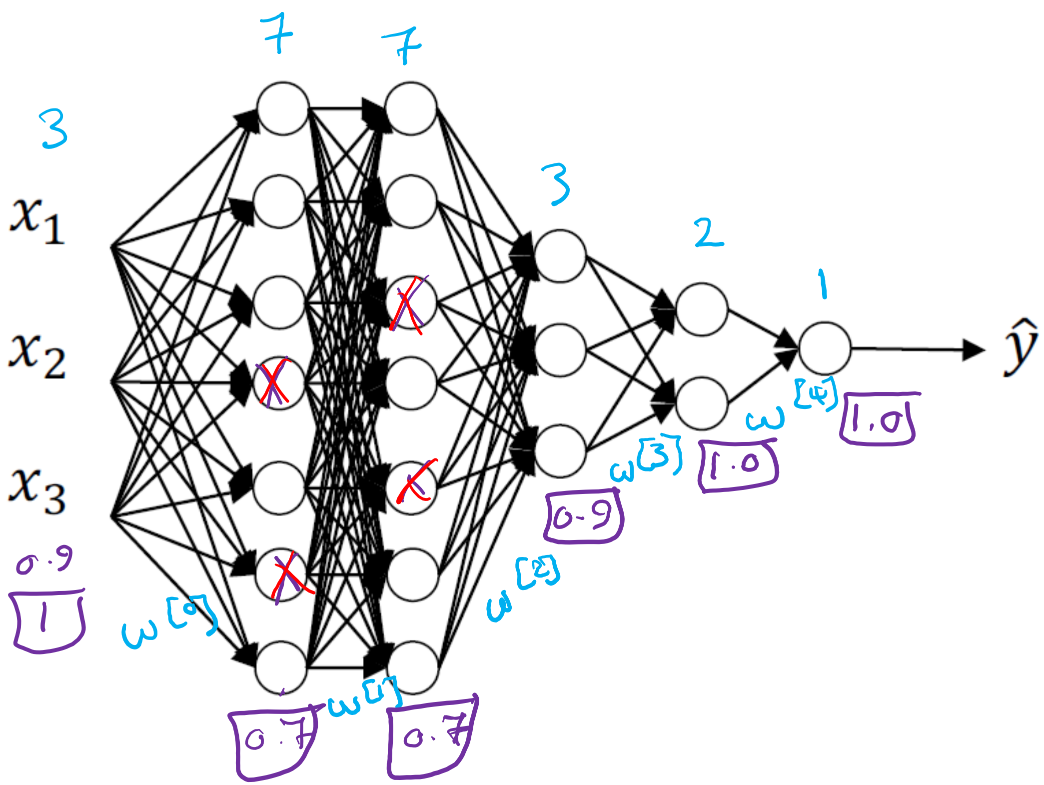
Understanding Dropout

Why does drop-out work?

Intuition: Can't rely on any one feature, so have to spread out weights   Shrink weight → L2

→ Computer vision





keep_prob

Core Foundation Review

Regularization

How does regularization prevent overfitting

Regularization techniques: L2, Dropout

Technique that constrains our optimization problem to discourage complex models