Neural Network: Internal Details

Chandranath Adak

Dept. of CSE, IIT Patna

We have covered so far

- What is neural network (NN)
- Architecture of the NN: i/p layer, o/p layer, hidden layer
- NN: forward propagation
- NN: Backpropagation
- Why multiple layers?
- Activation functions: Why do we need non-linearity?

Outline

- Activation functions
- Activation functions: derivatives
- Activation functions: Pros & Cons
- Random initialization
- Cost functions
- Cost functions: Pros & Cons
- Bias vs. Variance

Activation function: Sigmoid

z	$g(z) = \frac{1}{1+e^{-z}}$	$g'(z) = g(z)(1-g(z))$
20	≈ 1	≈ 0
-20	≈ 0	≈ 0
0	0.5	0.25

$g(z)$ = [0,1]

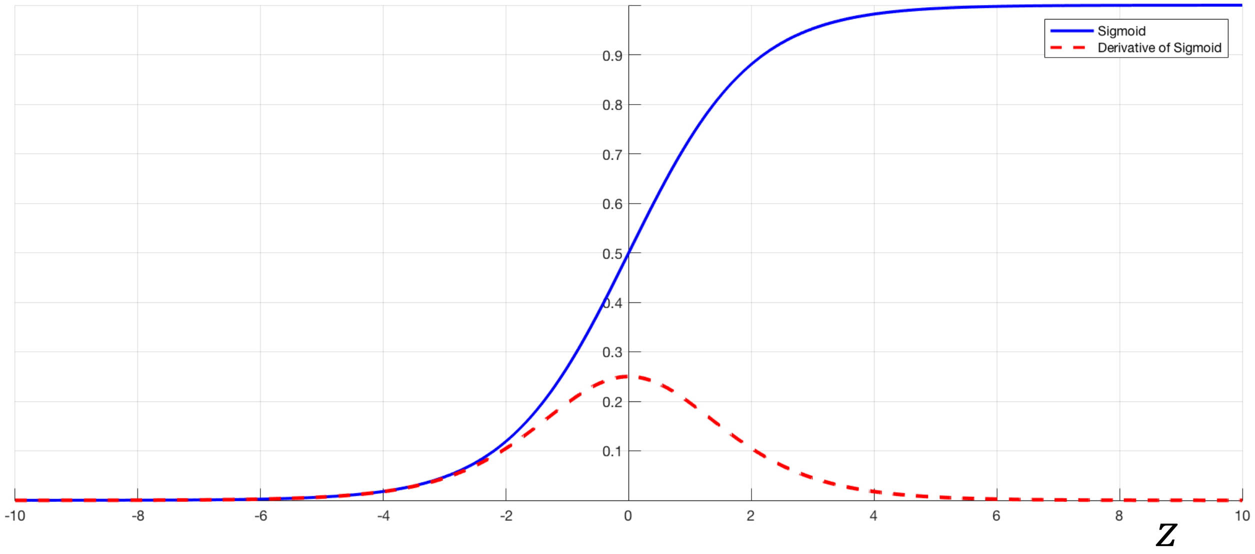
$$\frac{\partial[g(z)]}{\partial z}$$

$$= \frac{\partial[\frac{1}{1+e^{-z}}]}{\partial z}$$

$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2}$$

$$= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2}$$

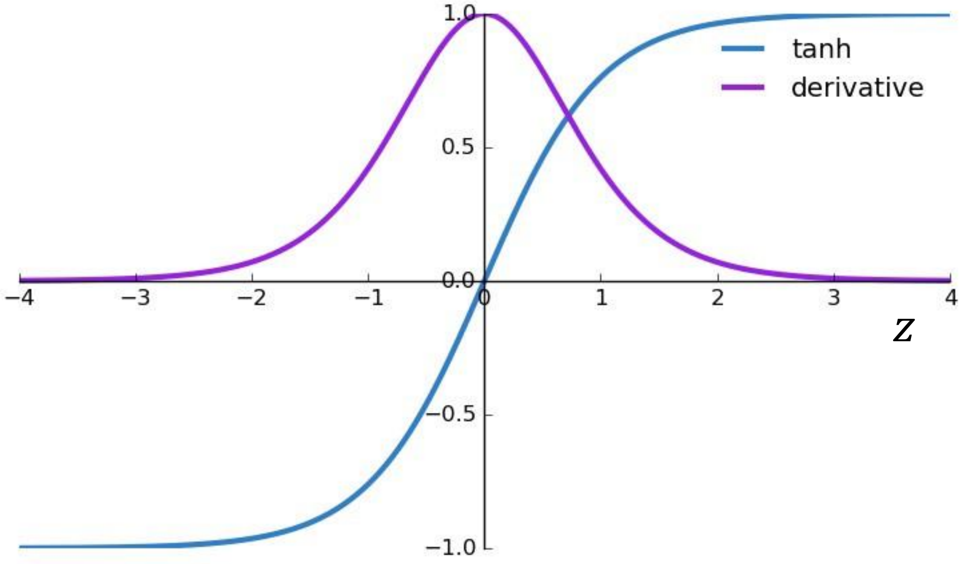
$$= g(z)[1-g(z)]$$



Activation function: tanh

z	$g(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$	$g'(z) = 1 - g^2(z)$
20	≈ 1	≈ 0
-20	≈ -1	≈ 0
0	0	1

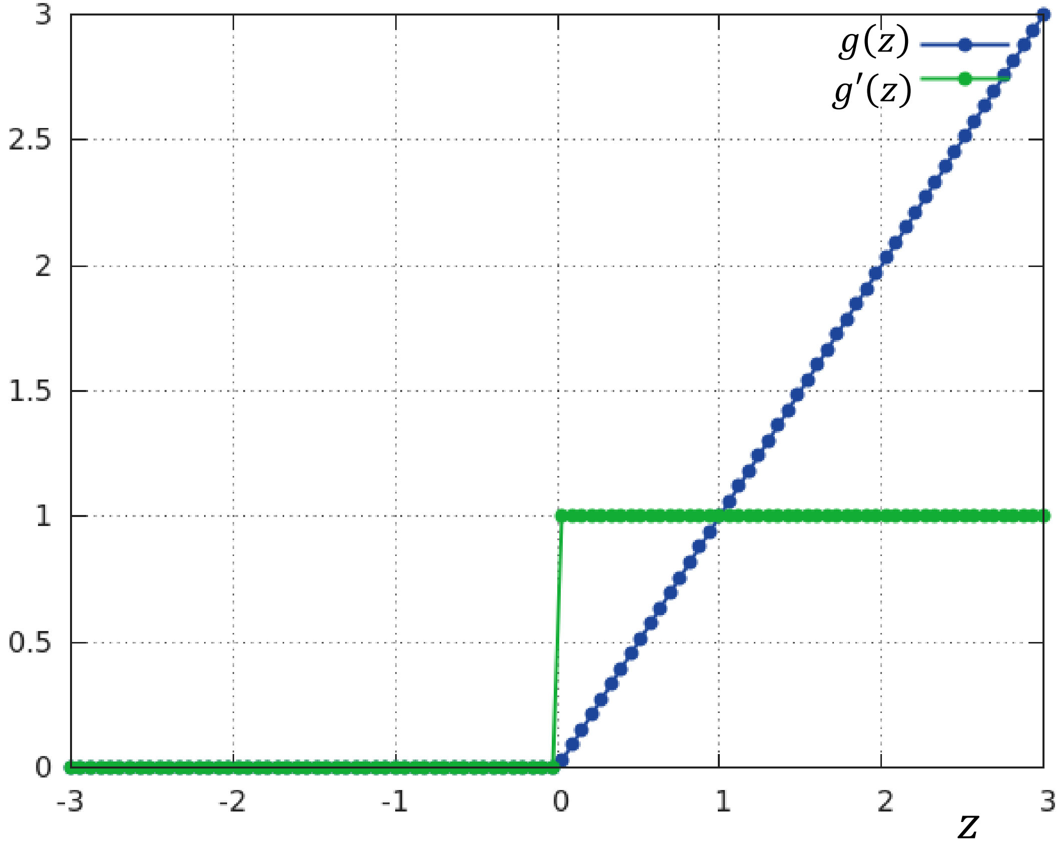
$g(z)$ = [-1,1]



Activation function: ReLU (Rectified Linear Unit)

z	$g(z) = \max(0, z)$	$g'(z) = \begin{cases} 0\,; & z < 0 \\ 1\,; & z > 0 \\ undefined; & z = 0 \end{cases}$

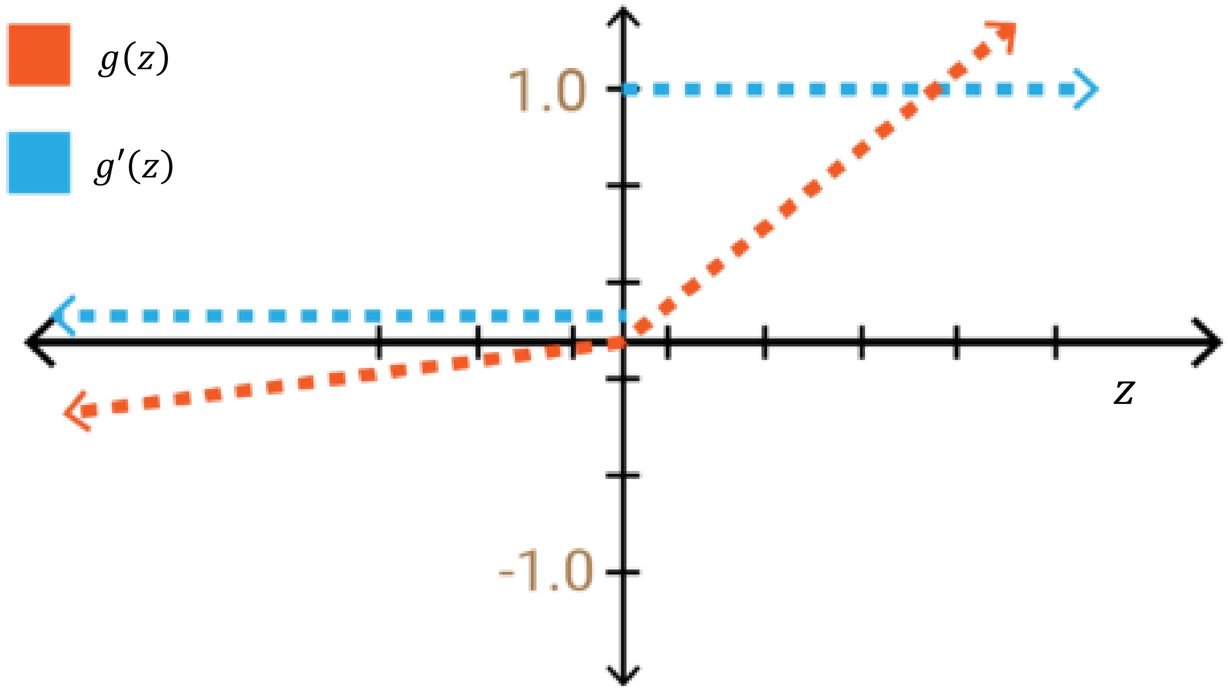
$g(z) = [0, \infty]$



Activation function: Leaky ReLU

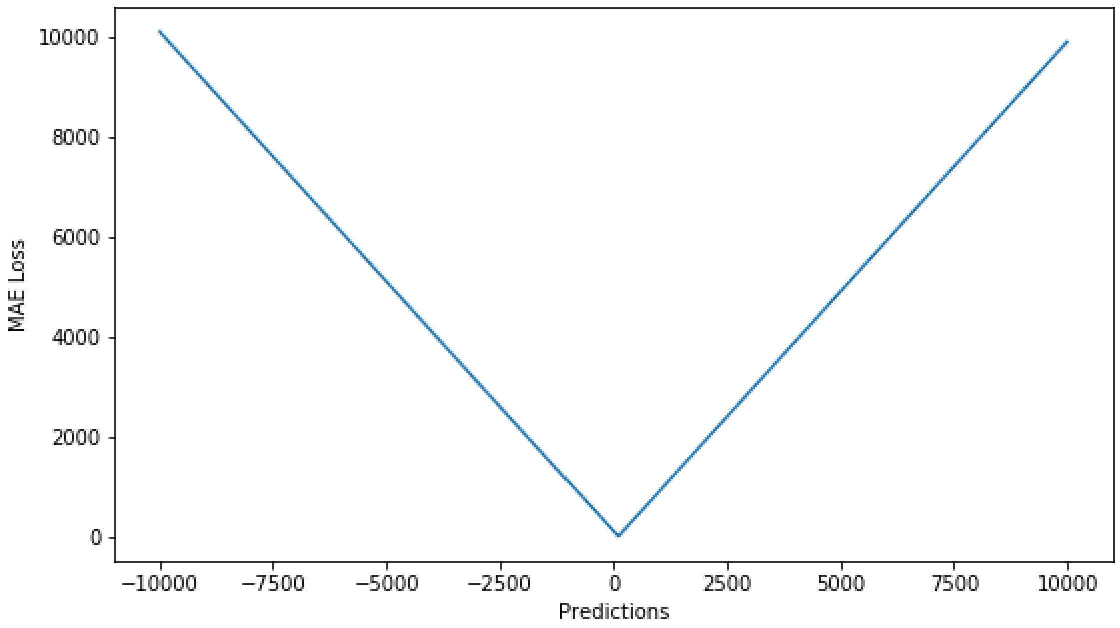
z	$g(z)$ $= \max(0.01z, z)$	$g'(z) = \begin{cases} 0.01\,; \quad z < 0 \\ 1\,; \quad z > 0 \\ undefined; \quad z = 0 \end{cases}$

$$g(z) = [-\infty, \infty]$$



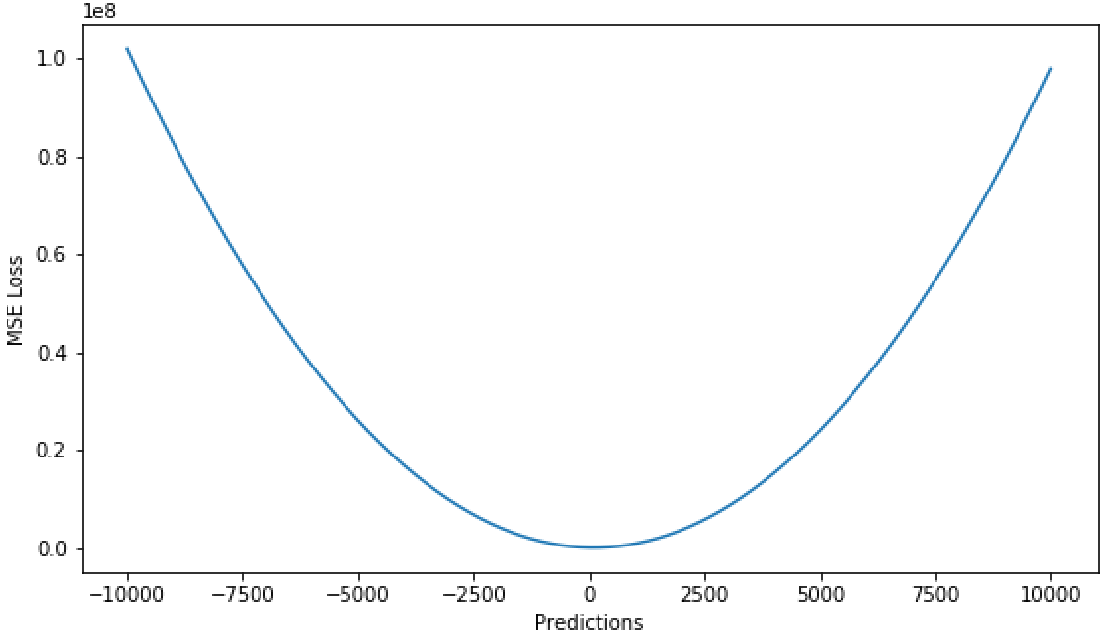
Loss function: MAE (L1 loss)

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \widehat{y}_i|}{n}$$



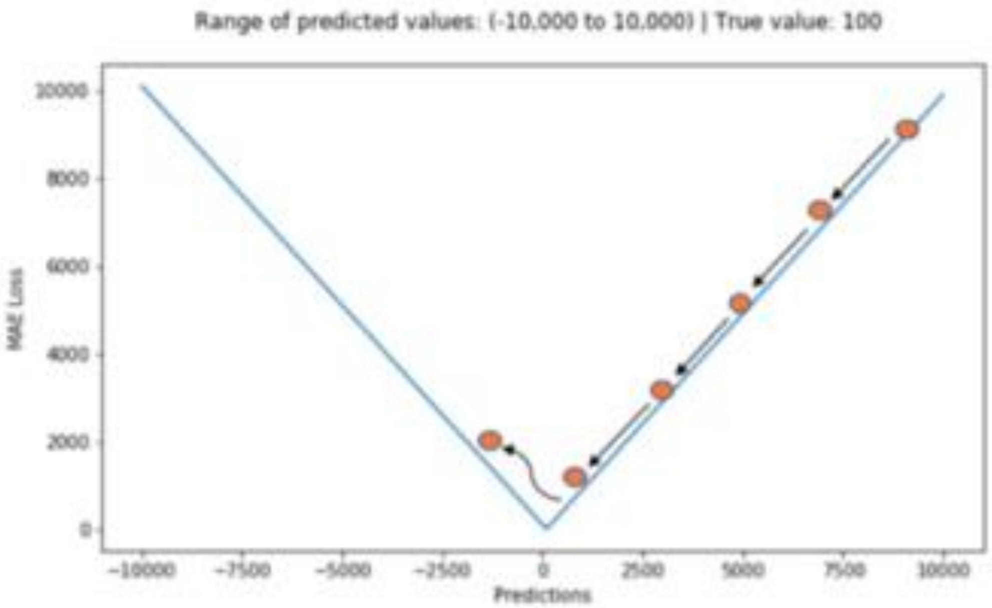
Loss function: MSE (L2 loss)

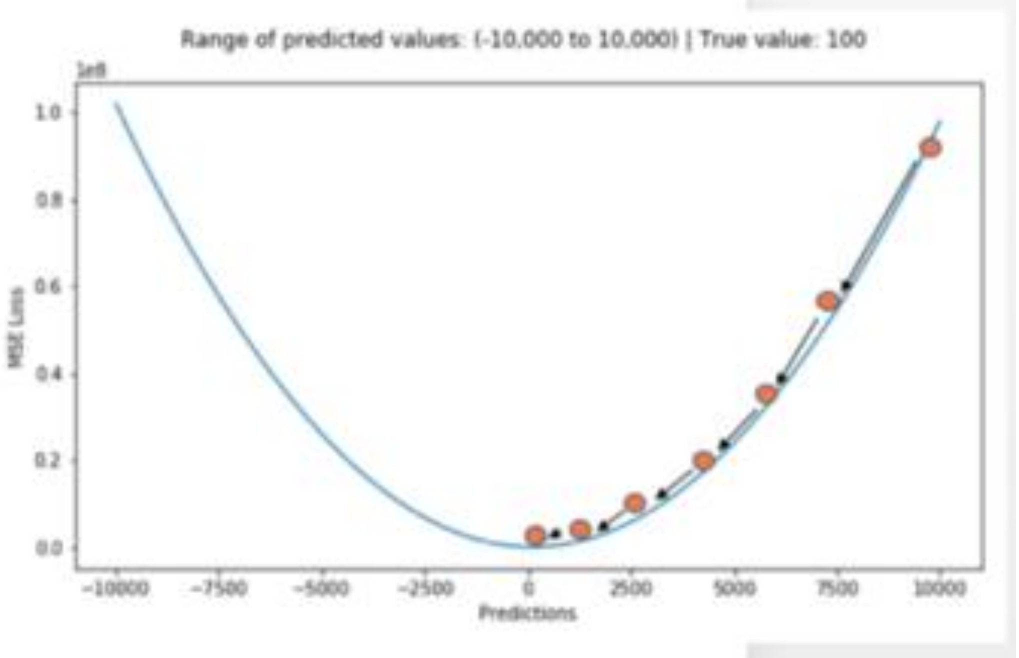
$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n}$$



- MAE is more robust to outliers than MSE.
- MAE loss is useful if the training data is corrupted with outliers.

- One big problem in using MAE loss is that its gradient is the same throughout, which means the gradient will be large even for small loss values.
- MSE behaves nicely in this case and will converge even with a fixed learning rate.



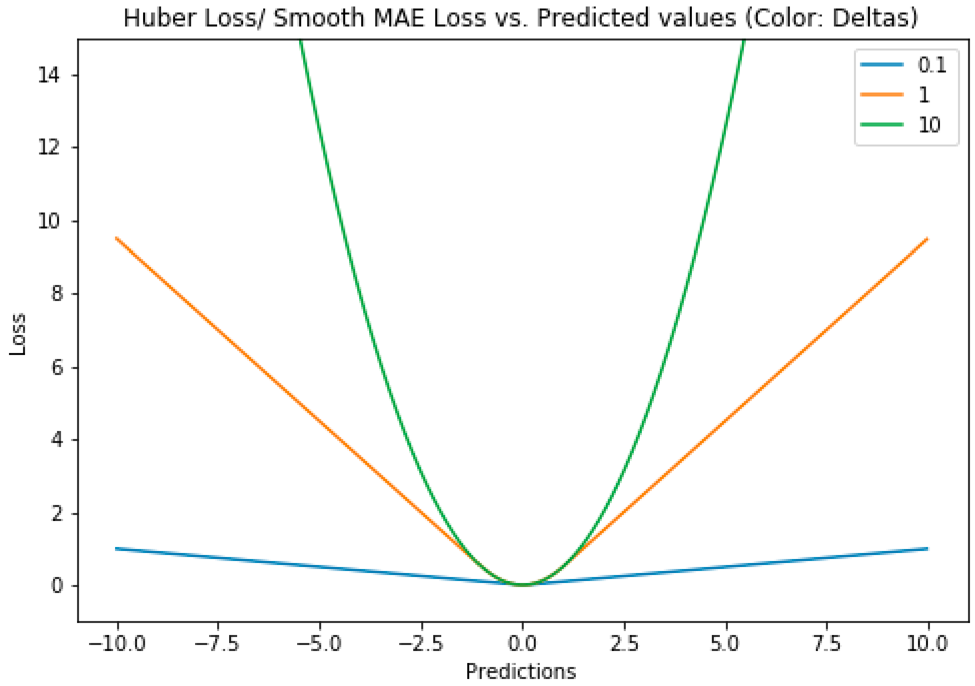


MAE is more robust to outliers, but its derivatives are not continuous, making it inefficient to find the solution.
MSE is sensitive to outliers, but gives a more stable and closed form solution

Loss function: Huber loss

- Huber loss is less sensitive to outliers in data than the MSE.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta \, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

Hinge loss

$$Hinge\ loss = \max(0, 1 - y.\hat{y})$$

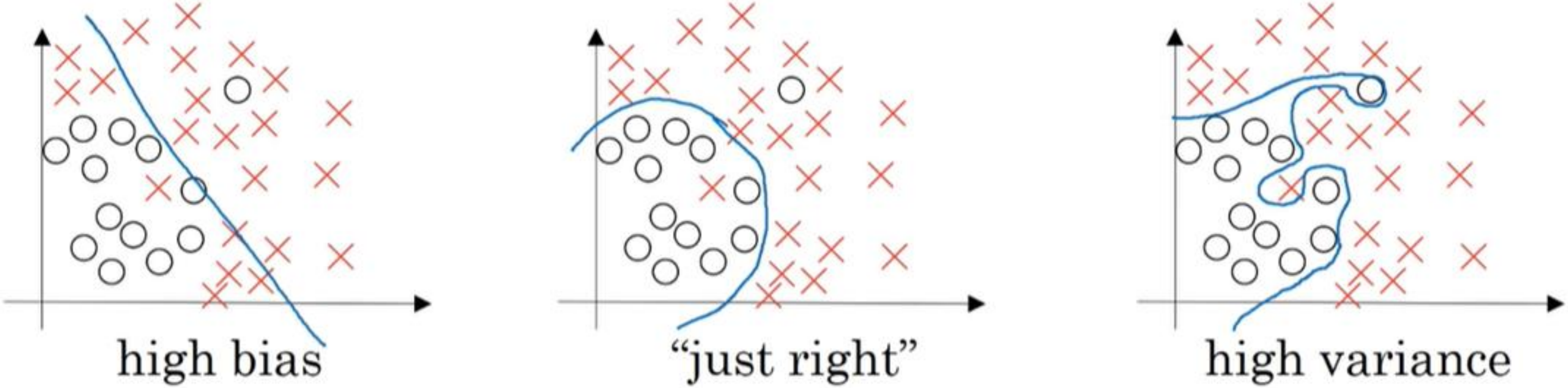
Mostly used in SVM

Cross Entropy loss

$$CE = -\sum_{i=1}^{c} y_i \log \widehat{y_i}$$

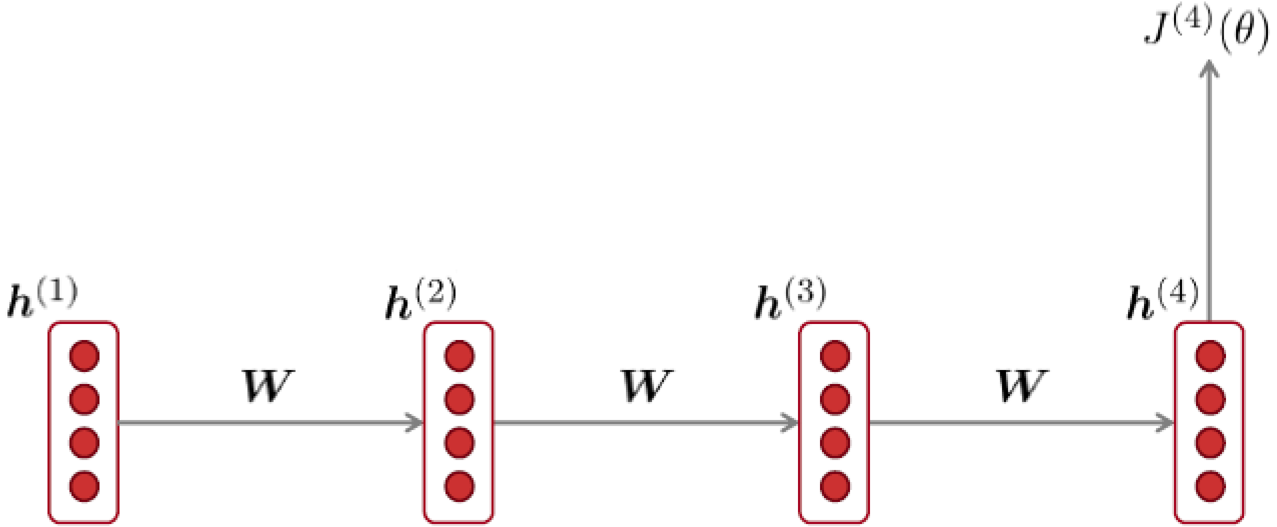
CE is mostly used for multi-class classification

$$Binary\ CE = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

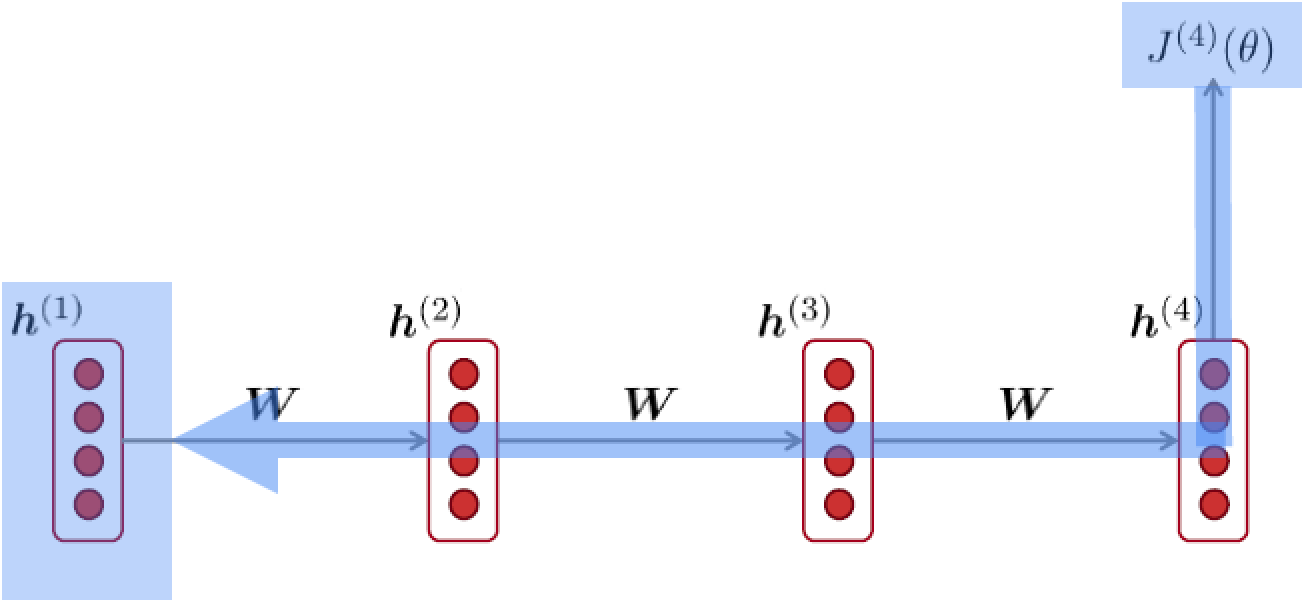


Underfitting

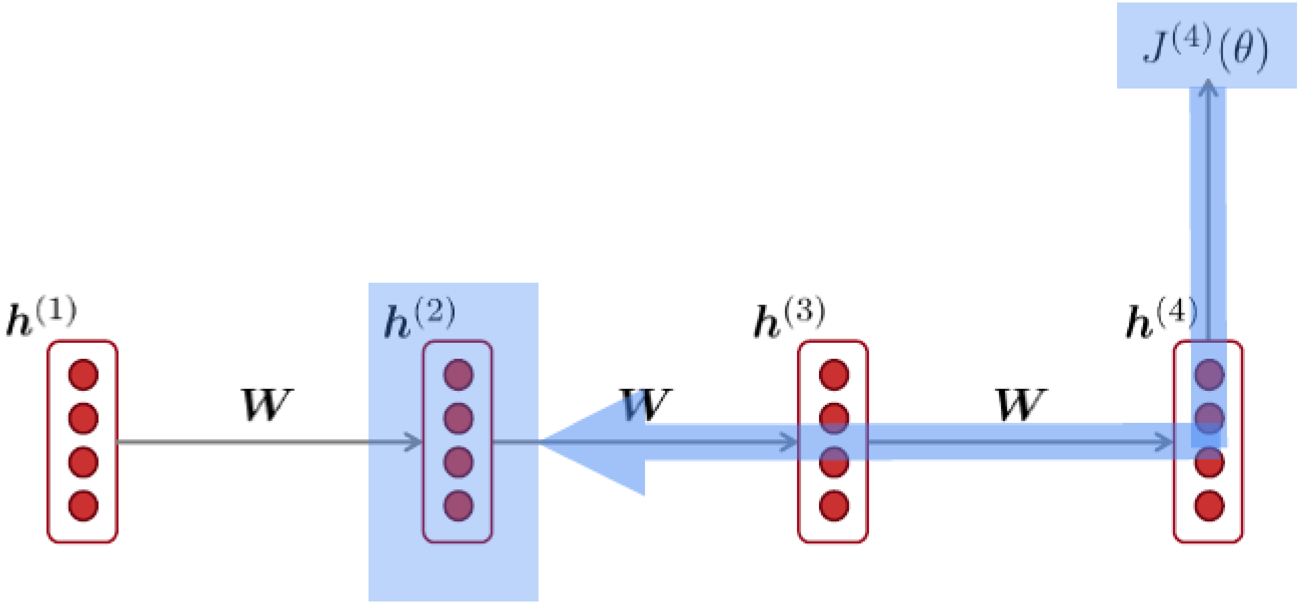
Overfitting



Courtesy: Abigail See, NLP, Stanford University

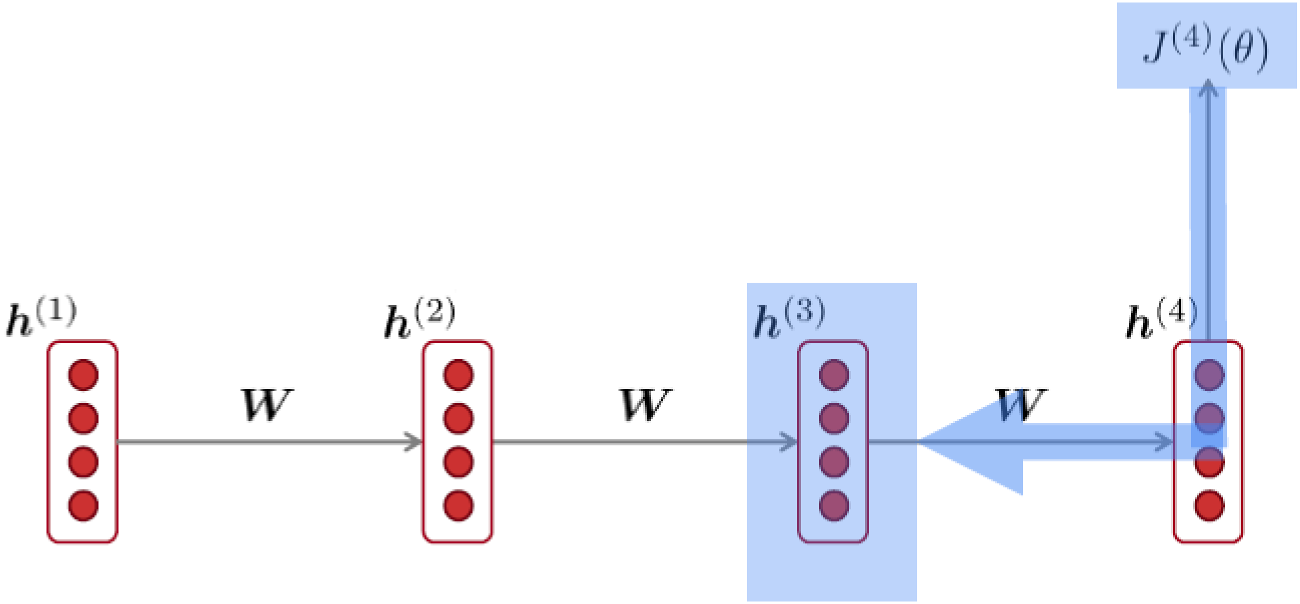


$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \ ?$$



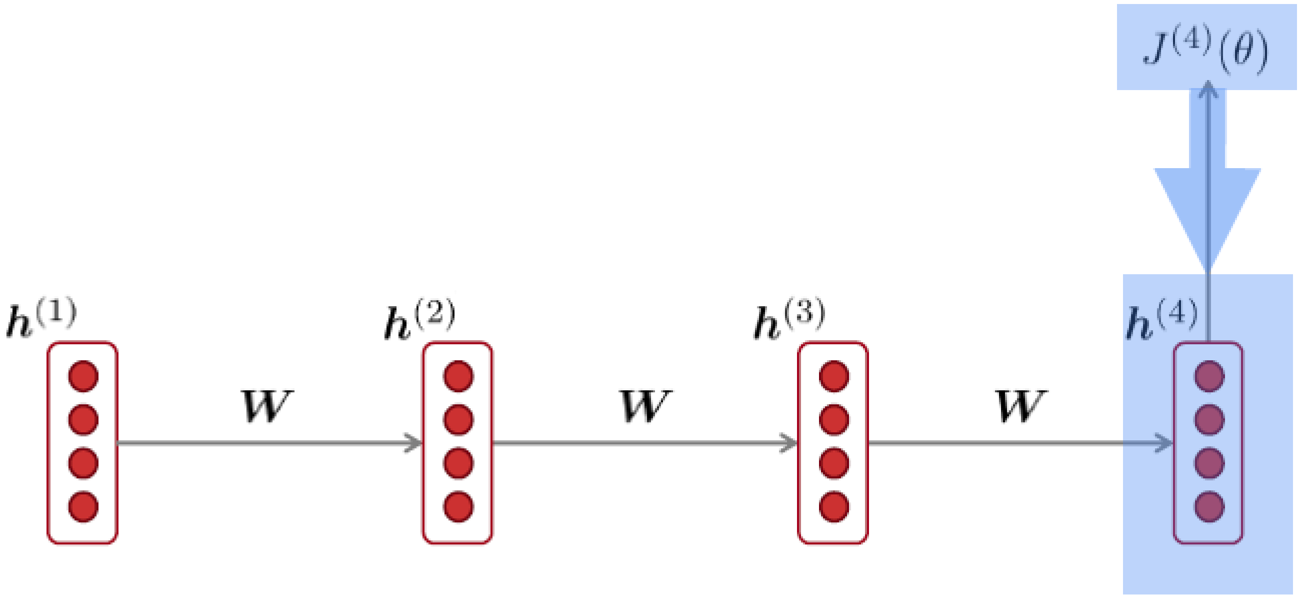
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

chain rule!



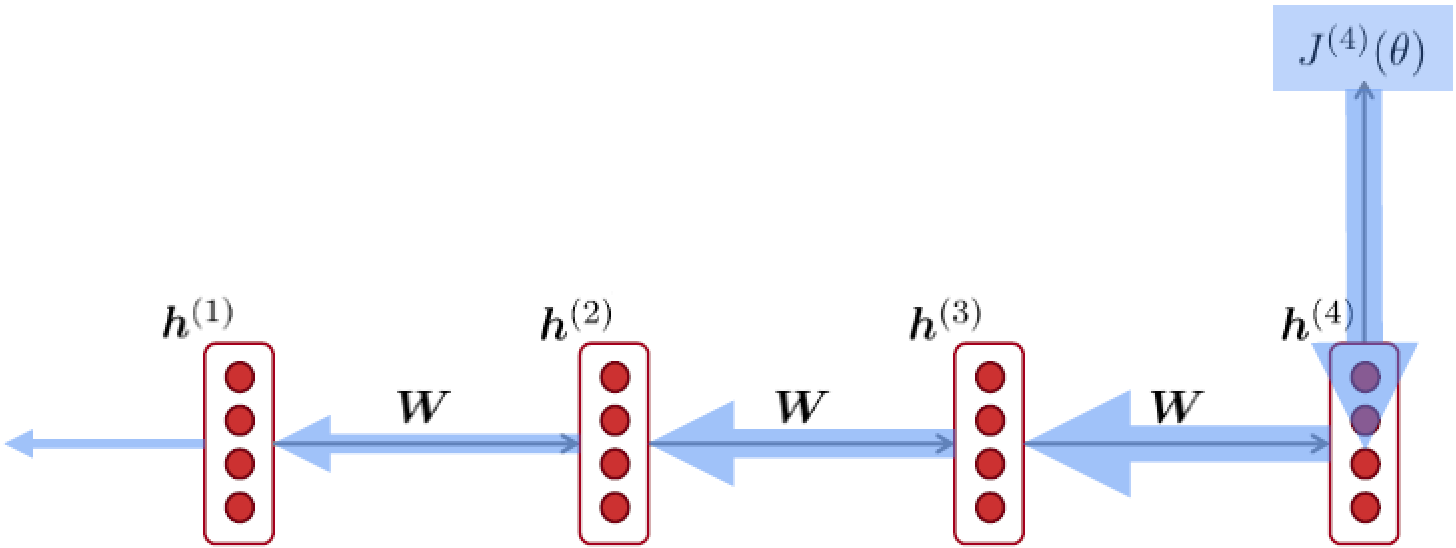
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \qquad \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(3)}}$$

chain rule!



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Thank You!

Activation function

- The activation function is used for transformation
- Performed after the input/ previous layer before sending it to the next layer/final output layer.