W20: Deep Learning with Python

Saturday January 12, 2019, 1:00 - 5:00 pm

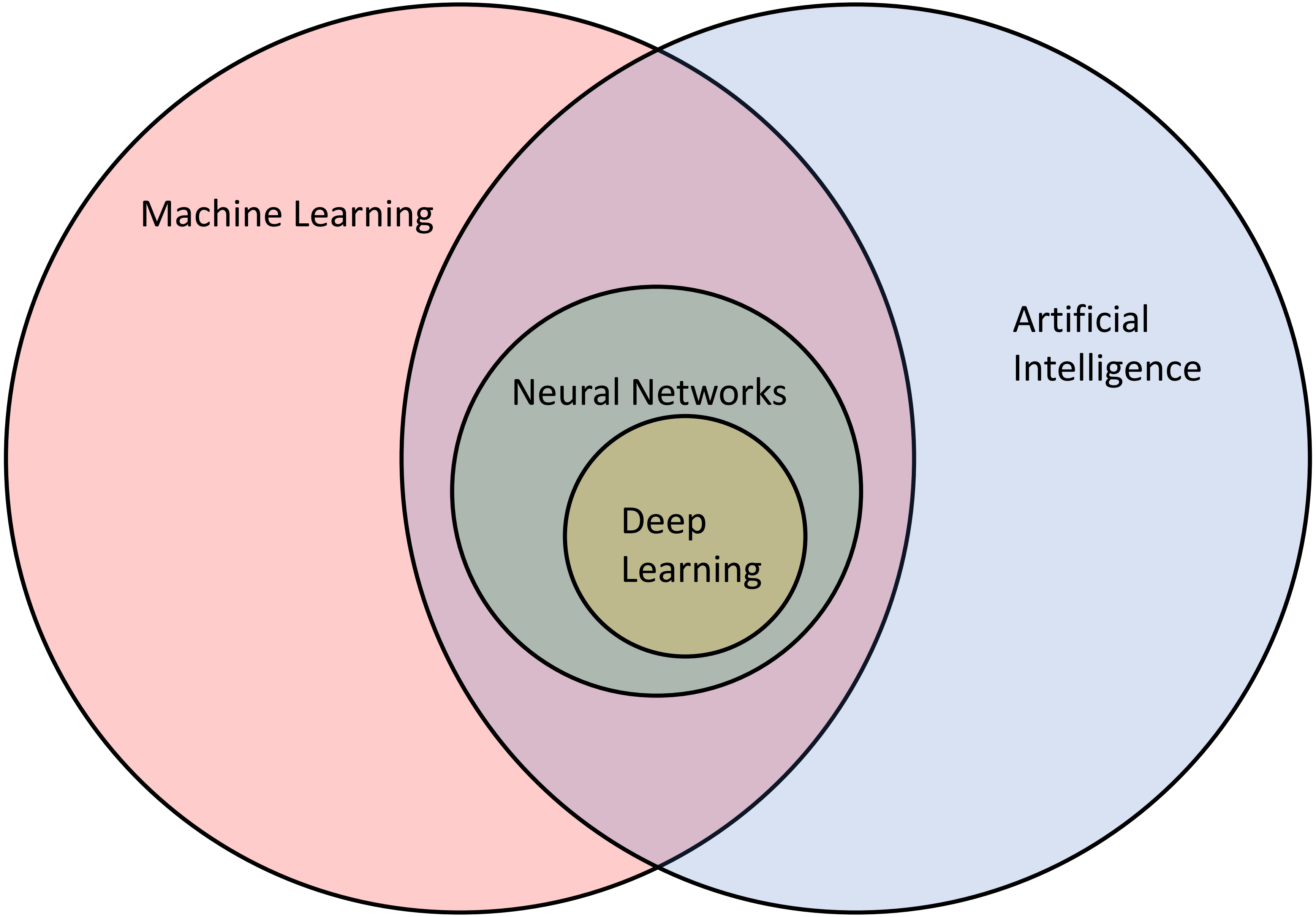
AAPT Winter Meeting, Houston, Texas

Jeff Groff, Shepherd University

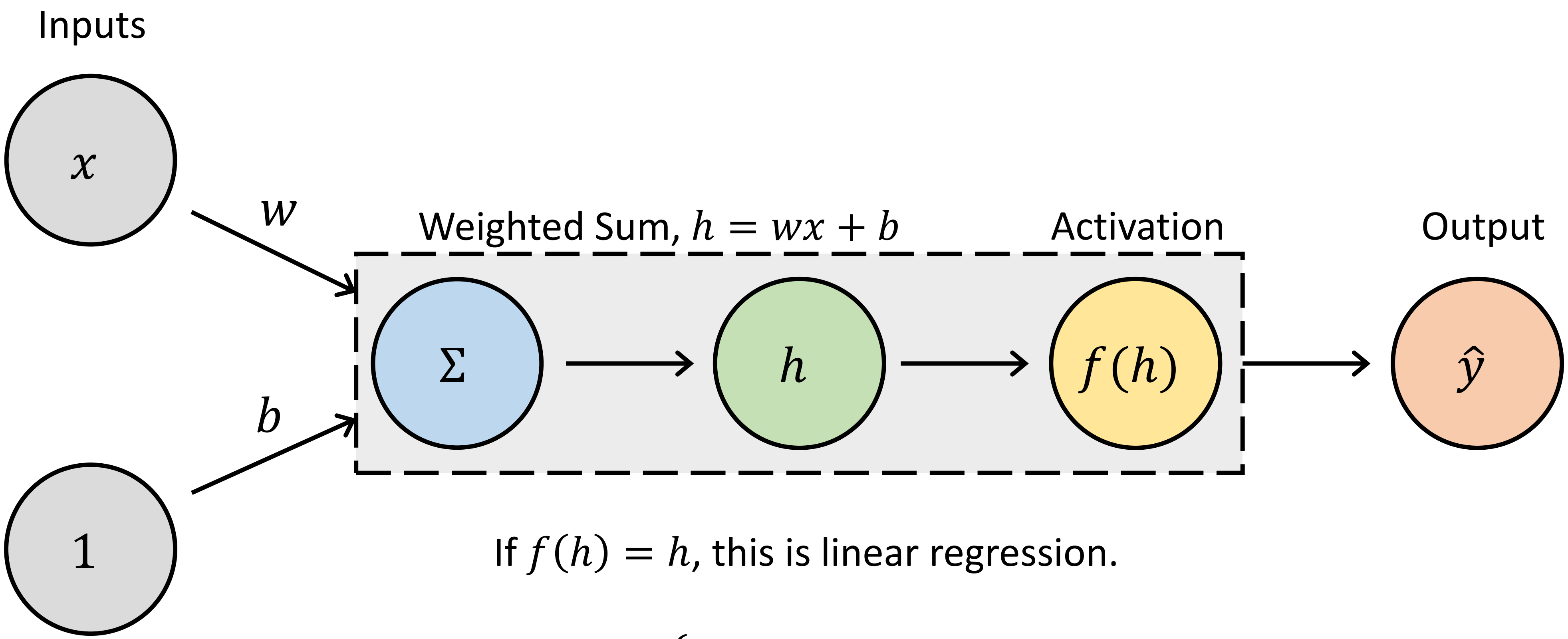
Workshop Materials:

https://github.com/prof-groff/deep-learning

Deep Learning: A Venn Diagram



Single-Layer, Single-Node Neural Network



If $f(h) = h$, this is linear regression.

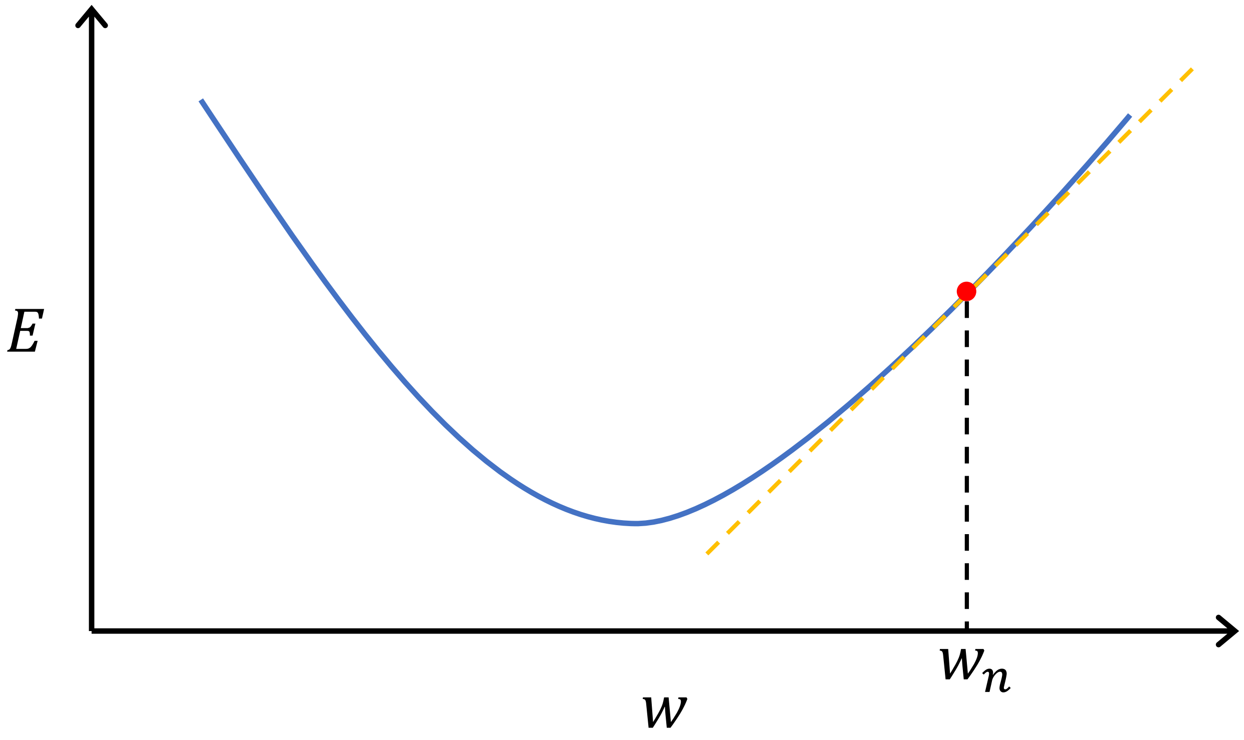
If $f(h) = \begin{cases} 1 \text{ if } h > 0 \\ 0 \text{ otherwise} \end{cases}$, this is a perceptron.

If $f(h) = \frac{1}{1 + e^{-h}}$, this is logistic regression.

Gradient Descent

Goal: Find network weights that minimize the error between the model outputs (predictions) and the actual data (targets).

To minimize $E(w,b) = \frac{1}{2}(y - \hat{y}(w,b))^2$ we can iterate each weight at each time step by a value proportional to the partial derivative of the error with respect to the weight.



$$\Delta w = w_{n+1} - w_n = -\eta \frac{\partial E}{\partial w}$$

$$\Delta b = b_{n+1} - b_n = -\eta \frac{\partial E}{\partial b}$$

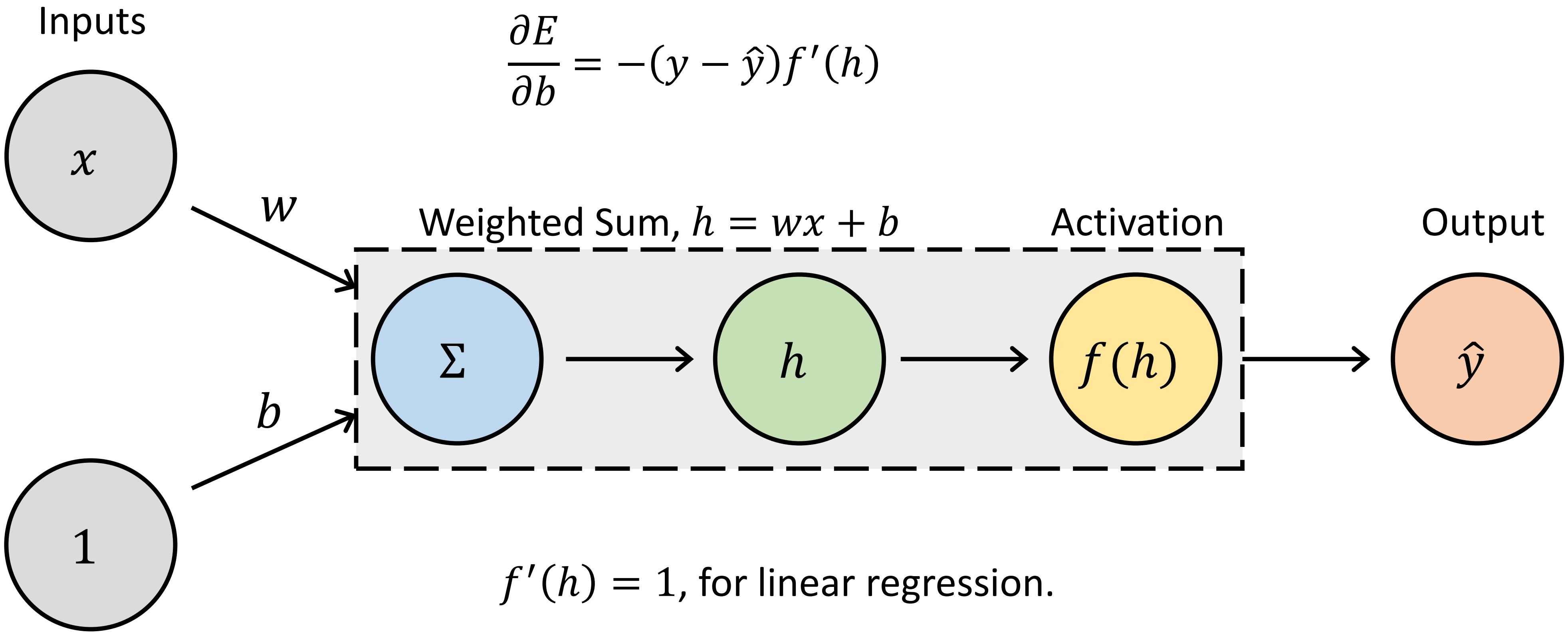
$$\Delta w = \frac{-\eta \sum_{\text{all data}} \frac{\partial E}{\partial w}}{\text{number of data records}}$$

Gradient Descent

$$E = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial E}{\partial w} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial w} = -(y - \hat{y})f'(h)\frac{\partial h}{\partial w} = -(y - \hat{y})f'(h)x$$

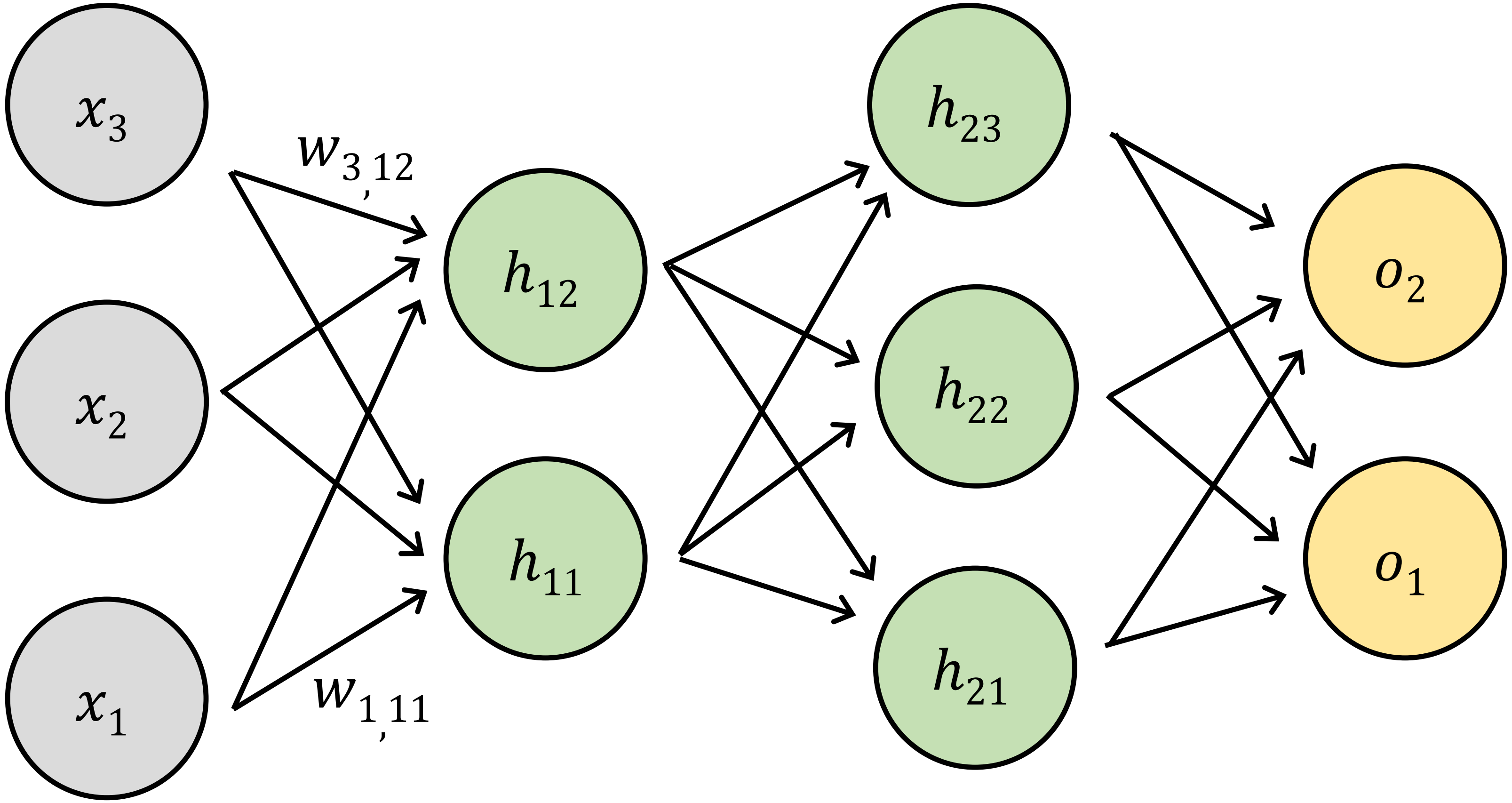
$$\frac{\partial E}{\partial b} = -(y - \hat{y})f'(h)$$



$f'(h) = 1$, for linear regression.

$f'(h) = f(h)(1 - f(h))$, for logistic regression.

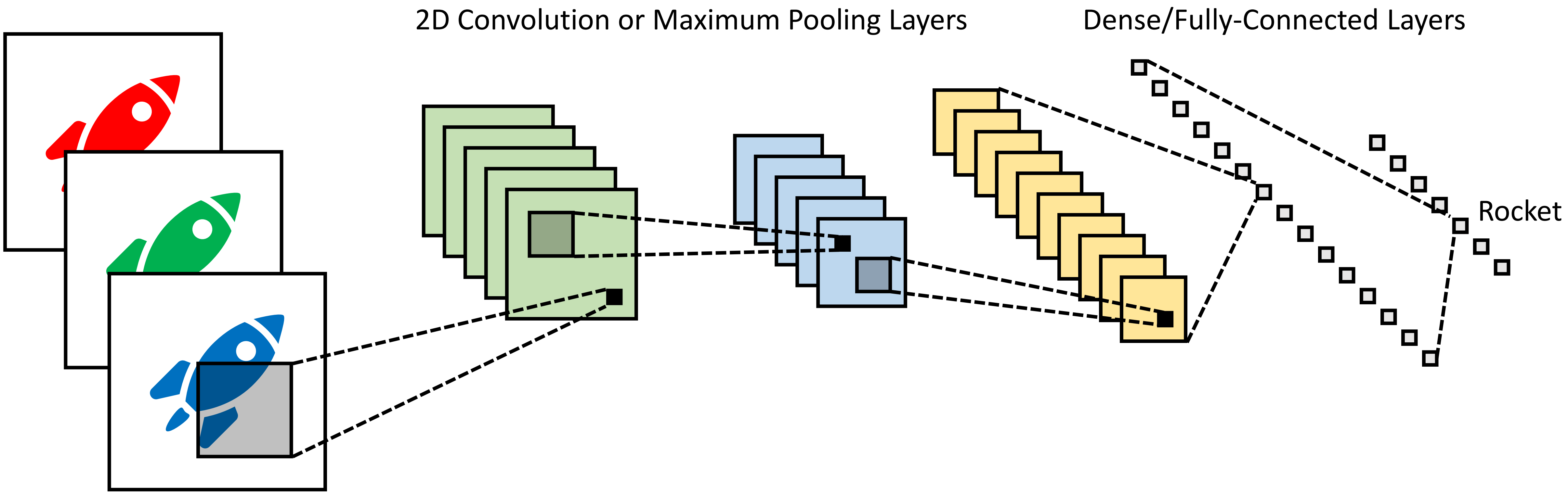
Multi-Node, Multi-Layer Neural Network
Deep Learning Models Have Many Layers



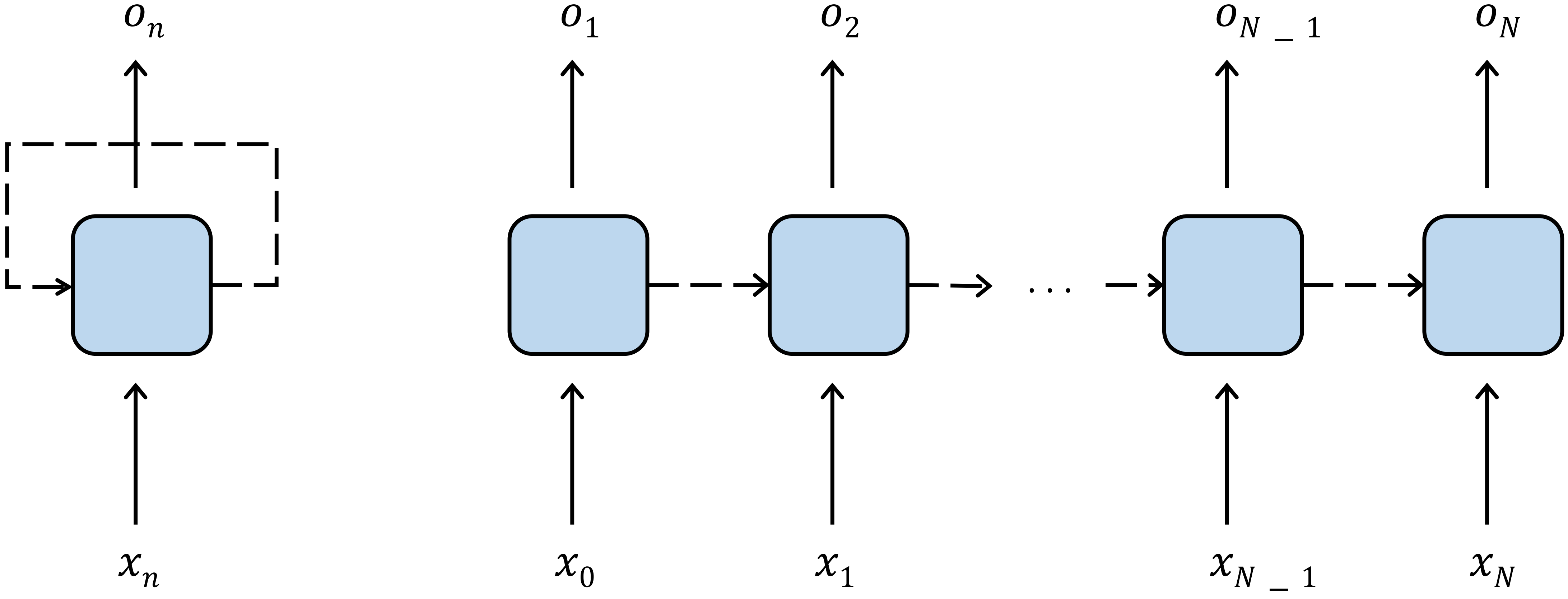
Convolutional Neural Network

Input Layer:
3-Channel
(RGB) Image

Output Layer:
One Node for
Each Class



Recurrent Neural Network



Unrolled