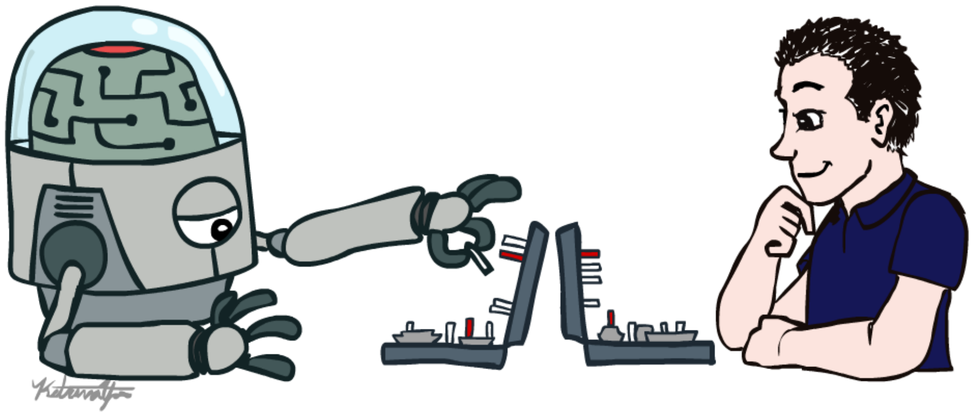
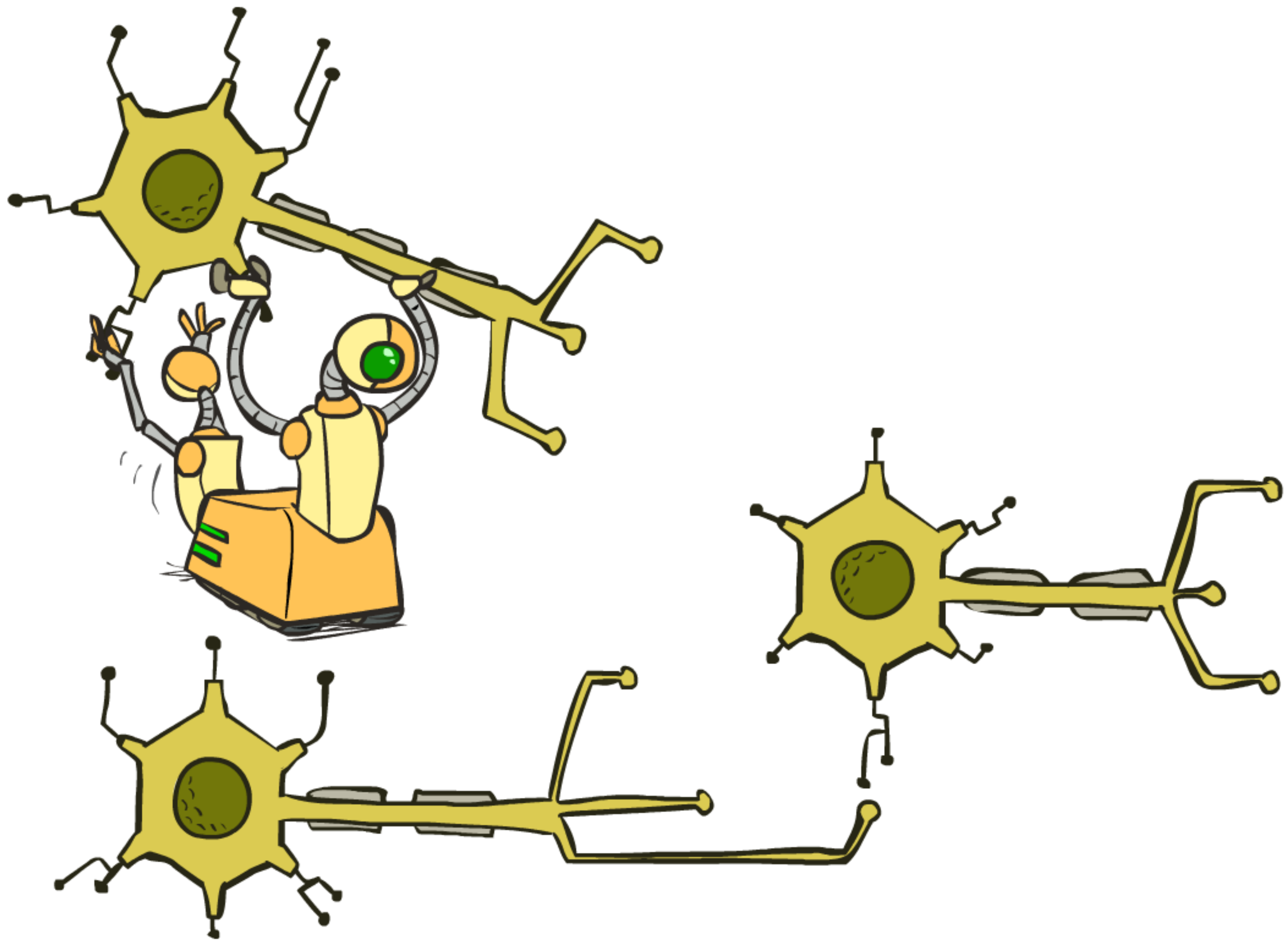
Lecture 12



Ashis Kumar Chanda
chanda@rowan.edu

Rowan University

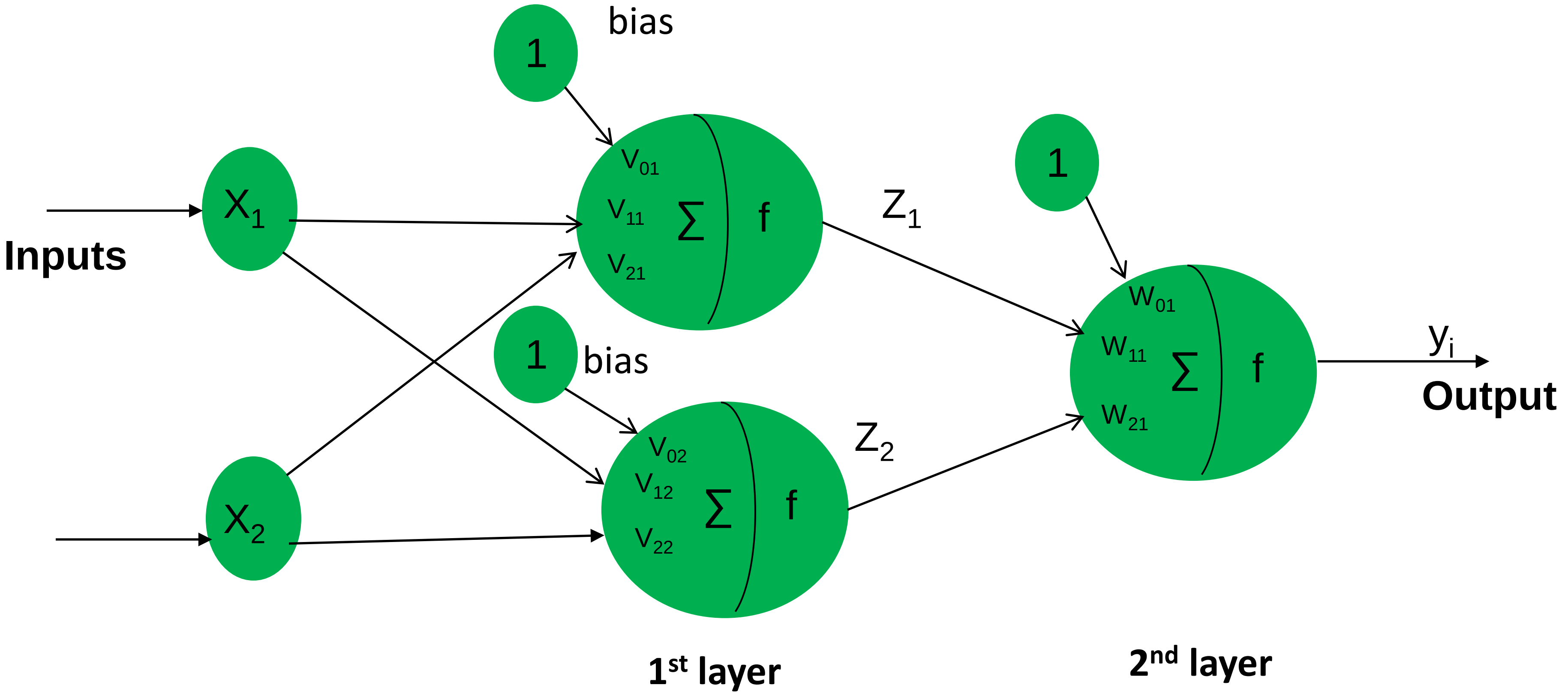
Neural Network Example



XOR Example

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

XOR Architecture



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

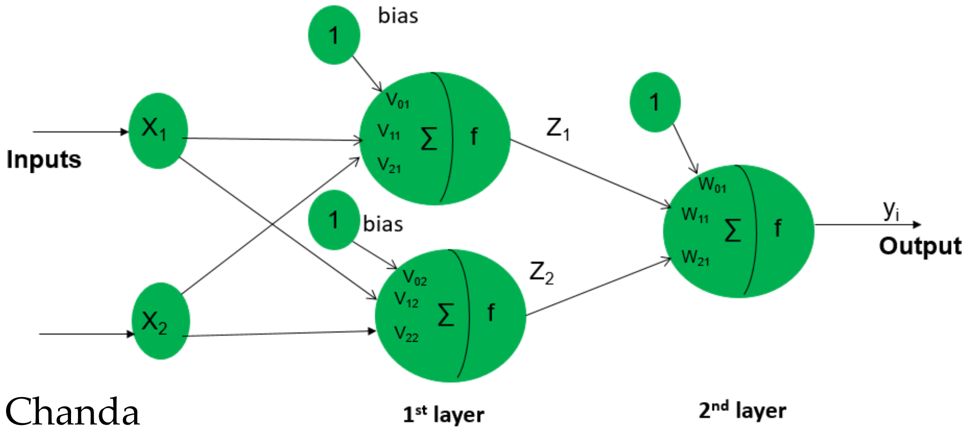
Sigmoid function is used as activation function

Example (XOR)

- Suppose the first input is (0,0).
- We randomly initialize the weights with some small values.
- The weights are in green color.
- We need to calculate the activation function values (red color) for 1st layer, and then, for 2nd layer.

1st layer

X_1	X_2	V_{01}	V_{11}	V_{21}	V_{02}	V_{12}	V_{22}	Z_1	Z_2
0	0	-0.3	0.21	0.15	0.25	-0.40	0.1		
0	1								
1	0								
1	1								



2nd layer

Z_1	Z_2	W_{01}	W_{11}	W_{21}	y_1
		-0.4	-0.2	0.3	

Example (XOR)

1st layer

$z_{in1} = -.3(1) + .21(0) + .15(0) = -.3$

$z_1 = f(z_{in1}) = .43$ $\longrightarrow$ $g(z) = \dfrac{1}{1 + e^{-z}}$

$z_{in2} = .25(1) - .4(0) + .1(0) = 0.25$

$z_2 = f(z_{in2}) = .56$

$y_{in1} = -.4(1) - .2(.43) + .3(.56) = -.318$

$y_1 = f(y_{in1}) = .42$ $\longrightarrow$ $g(z) = \dfrac{1}{1 + e^{-z}}$

Example (XOR)

- We finished the calculation for the first example.
- Our model prediction is 0.42
- Now, we need to update weights (Backpropagation).

x_1	x_2	v_{01}	v_{11}	v_{21}	v_{02}	v_{12}	v_{22}	z_1	z_2
0	0	-0.3	0.21	0.15	0.25	-0.40	0.1	0.43	0.56
0	1								
1	0								
1	1								

z_1	z_2	w_{01}	w_{11}	w_{21}	y_1
0.43	0.56	-0.4	-0.2	0.3	0.42

Weight update (**backpropagation**)

$\delta_1 = (t_1 - y_1)f'(y_{in1})$

$=(t_1 - y_1)\{ f(y_{in1})[1- f(y_{in1})] \}$

$\delta_1 = (0 - .42).42[1-.42] = -0.102$

t_1 = real output

y_1 = predicted output

f' = derivation of sigmoid

$\Delta W_{01} = -0.102 \text{x } 1 = -0.102$

$\Delta W_{11} = -0.102 \text{x } 0.43 = -0.04386$

$\Delta W_{21} = -0.102 \text{x } 0.56 = -0.05712$

Weight update (**backpropagation**)

Weight update of 2nd layer:

W_{01} (new) = -0.4 + (-0.102) = -0.502
W_{11} (new) = -.2 + (-0.04386) = -0.243
W_{21} (new) = -.3 + (-0.05712) = 0.243

[Here, we don't use learning rate]

Weight update (**backpropagation**)

$\delta_{in1} = \delta_1 \, \mathbf{w_{11}} = \mathbf{-.102(-.2) = .02}$

$\delta_1 = \delta_{in1} \, \mathbf{f'(z_{in1}) = .02(.43)(1-.43) = .005}$

$\delta_{in2} = \delta_1 \, \mathbf{w_{21}} = \mathbf{-.102(.3) = -.03}$

$\delta_2 = \delta_{in2} \, \mathbf{f'(z_{in2}) = -.03(.56)(1-.56) = -.007}$

$\Delta v_{01} = 0.005 \times 1 = 0.005$ $\quad\quad$ $\Delta v_{02} = -0.007 \times 1 = -0.007$

$\Delta v_{11} = 0.005 \times 0.0 = 0.0$ $\quad\quad$ $\Delta v_{12} = -0.007 \times 0.0 = 0.0$

$\Delta v_{21} = 0.005 \times 0.0 = 0.0$ $\quad\quad$ $\Delta v_{22} = -0.007 \times 0.0 = 0.0$

Weight update (**backpropagation**)

Weight update of 1st layer:

V_{01} (new) = -0.3 + (0.005) = -0.295
V_{11} (new) = 0.21 + (0.0) = 0.21
V_{21} (new) = 0.15 + (0.0) = 0.15

V_{02} (new) = 0.25 + (-0.007) = 0.243
V_{12} (new) = -0.4 + (0.0) = -0.4
V_{22} (new) = 0.1 + (0.0) = 0.1

Example (XOR)

X_1	X_2	V_{01}	V_{11}	V_{21}	V_{02}	V_{12}	V_{22}	Z_1	Z_2
0	0	-0.3	0.21	0.15	0.25	-0.40	0.1	0.43	0.56
0	1	-.295	0.21	0.15	0.243	-0.40	0.1		
1	0								
1	1								

Z_1	Z_2	W_{01}	W_{11}	W_{21}	y_1
0.43	0.56	-0.4	-0.2	0.3	0.42
		-0.502	-0.243	0.243	

Example (XOR)

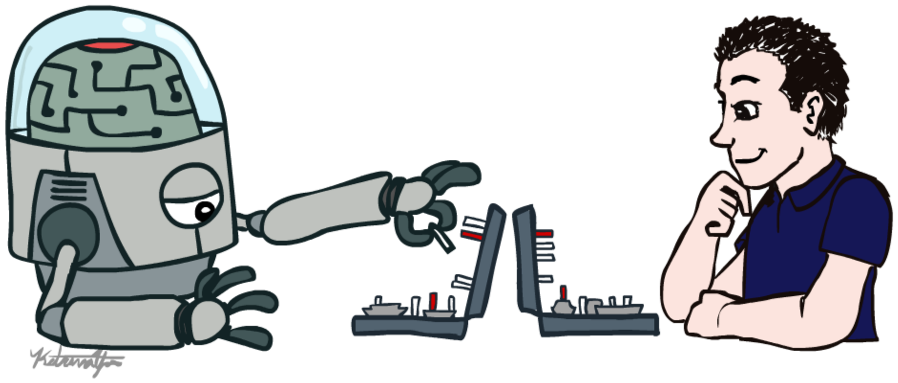
- After around 100 iteration the program reach termination condition.

Vanishing gradient

- The fact that in a feedforward network (FFN), the backpropagated **error signal** typically decreases (or increases) exponentially as a function of the distance from the final layer.
- The result is the general **inability** of models with **many layers** to learn on a given dataset.
- The use of **Relu** as an activation function can help to reduce the problem.

Fun Neural Net Demo Site

- Demo-site:
  - http://playground.tensorflow.org/



Thanks!