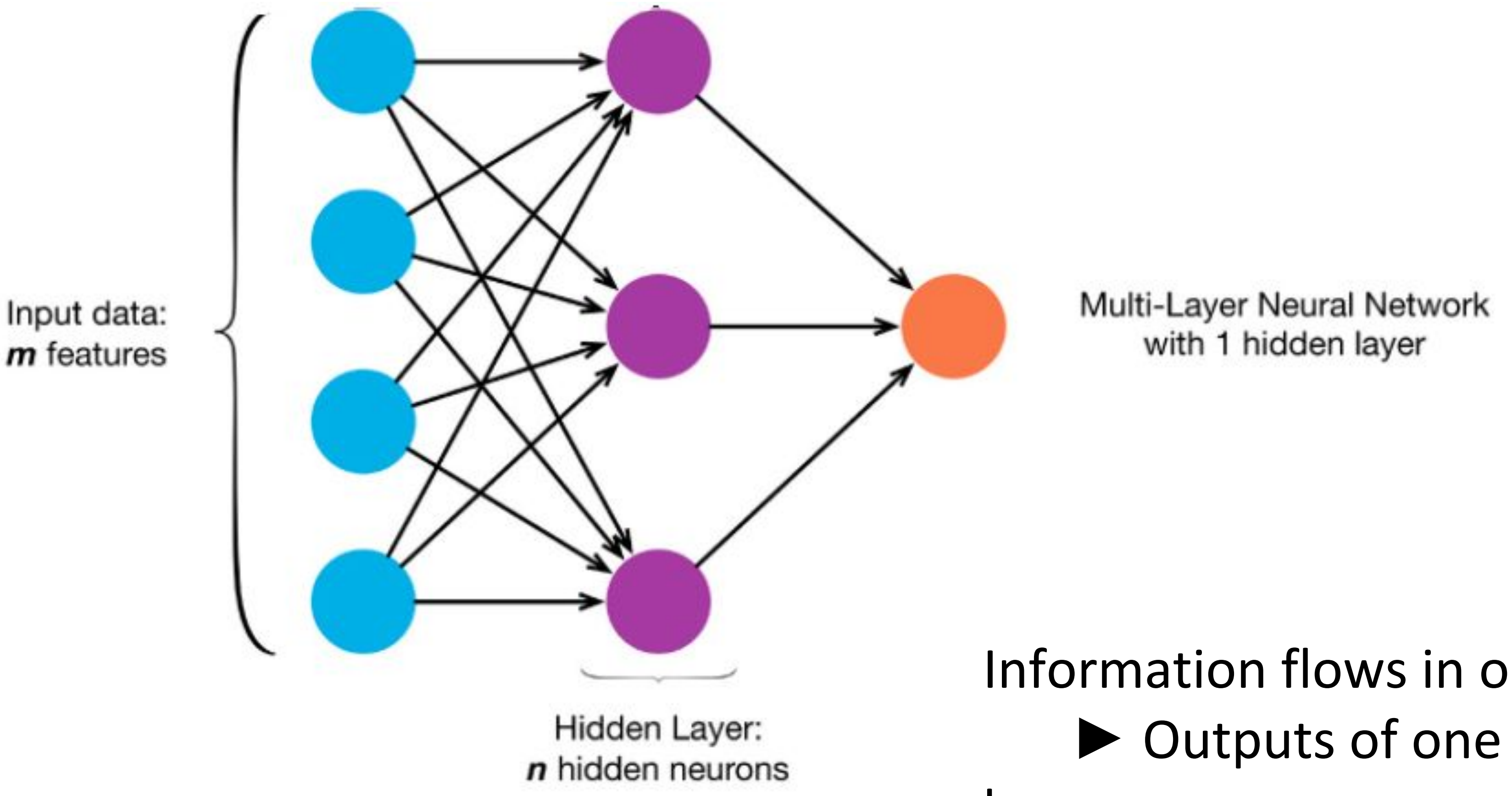
Lecture

- **Multilayer Feedforward Neural Networks**
- **XOR Example- Using Sigmoid Function**
- **Backpropagation**

Multilayer Feedforward Networks

- Most common neural network
- An extension of the perceptron
  - ► Multiple layers
    - ■ The addition of one or more “hidden” layers in between the input and output layers
  - ► Activation function is not simply a threshold
    - ■ Usually a sigmoid function
  - ► A general function approximator
    - ■ Not limited to linear problems

Multilayer Feedforward Networks

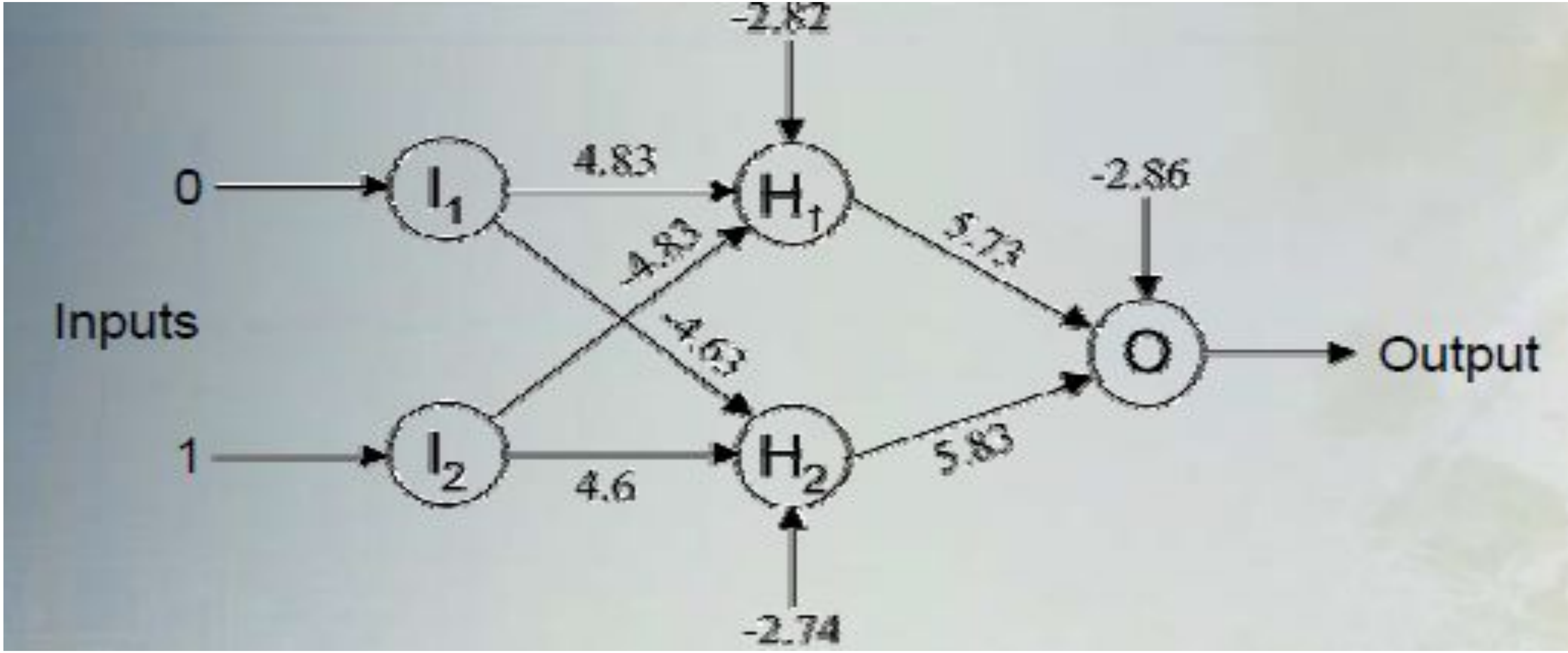


Information flows in one direction

► Outputs of one layer act as inputs to next layer

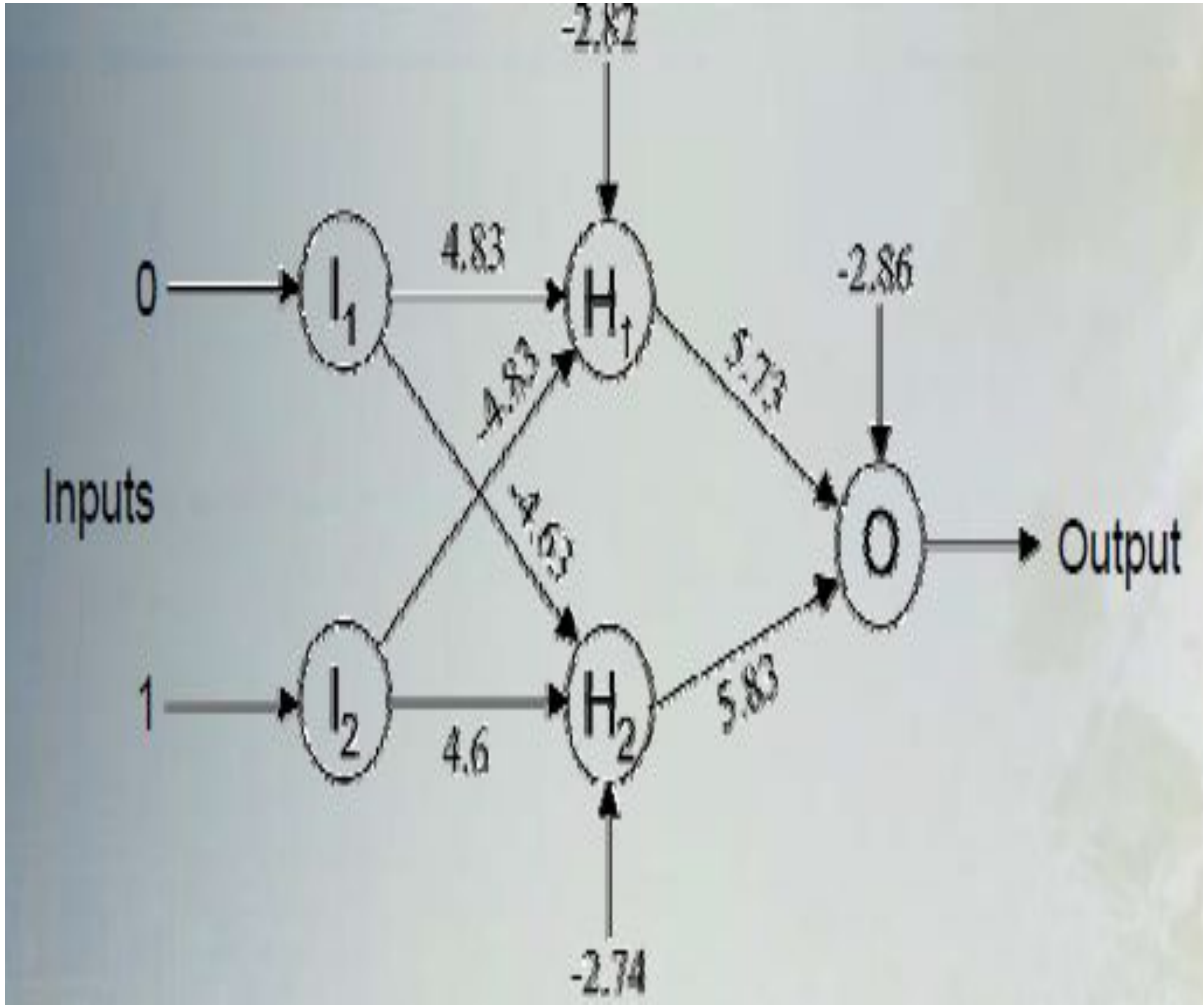
https://help.imsl.com/c/6.0/stat/default.htm?turl=multilayerfeedforwardneuralnetworks.htm

XOR Example



https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

XOR Example- Using Sigmoid Function



https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Inputs: 0, 1

H_1:

Net = ?

Output = ?

H_2:

Net = ?

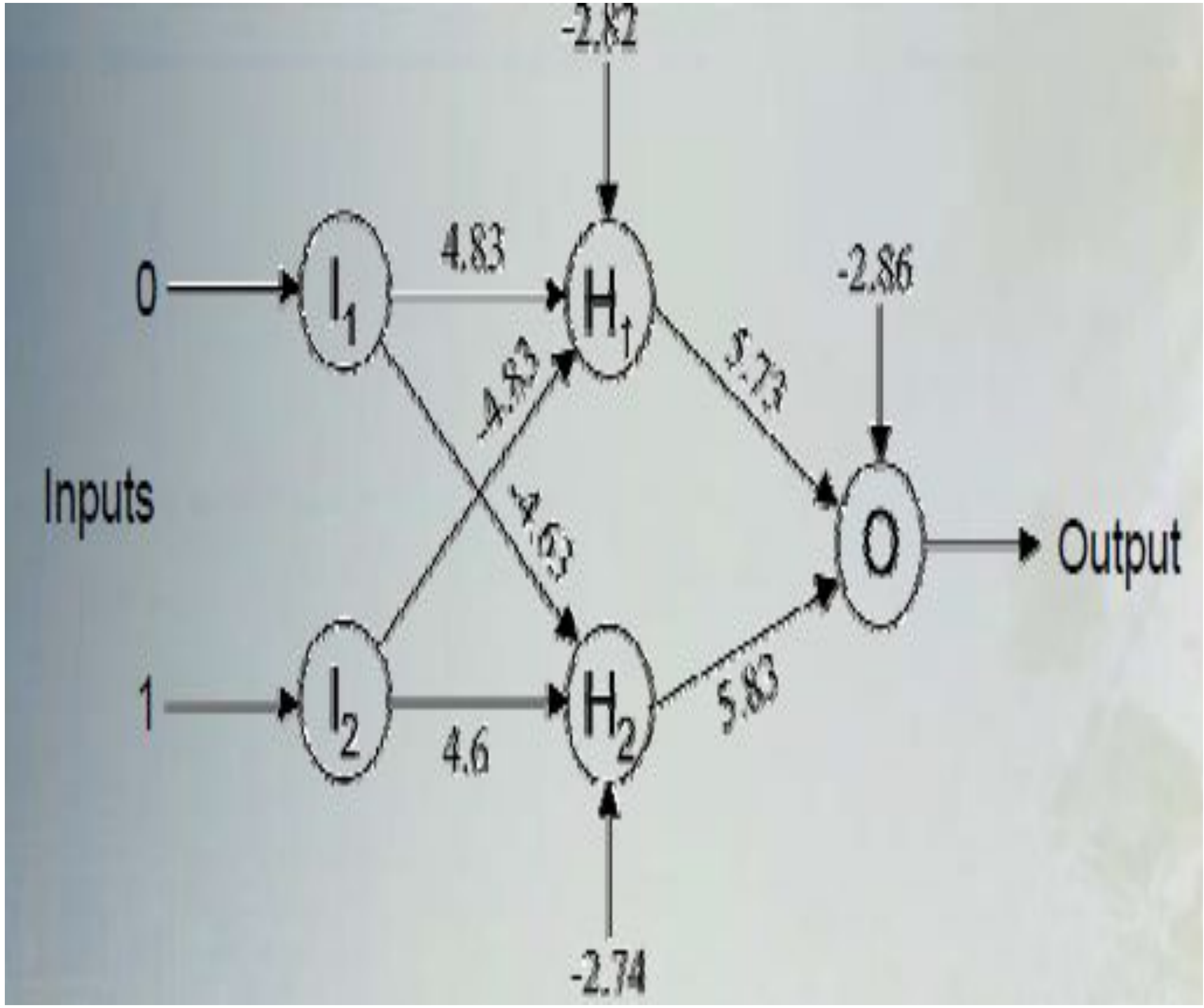
Output = ?

O:

Net = ?

Output = ?

XOR Example- Using Sigmoid Function



https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Inputs: 0, 1

H_1:

Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65

Output = $1 / (1 + e^{7.65}) = 4.758 * 10^{-4}$

H_2:

Net = ?

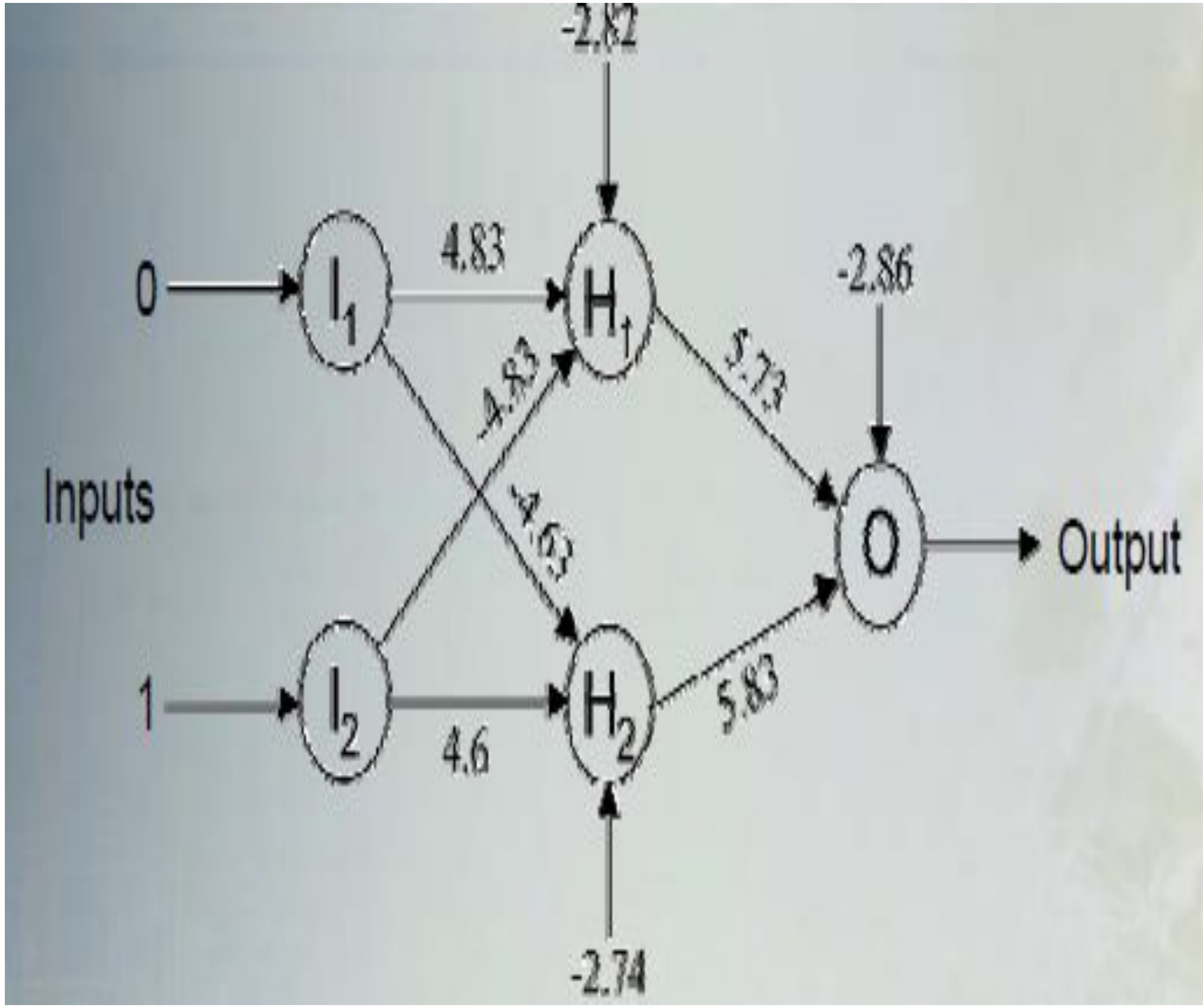
Output = ?

O:

Net = ?

Output = ?

XOR Example- Using Sigmoid Function



https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Inputs: 0, 1

H_1:

Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65

Output = 1 / (1 + $e^{7.65}$) = 4.758 * 10^{-4}

H_2:

Net = 0(-4.63) + 1(4.6) -2.74 = 1.86

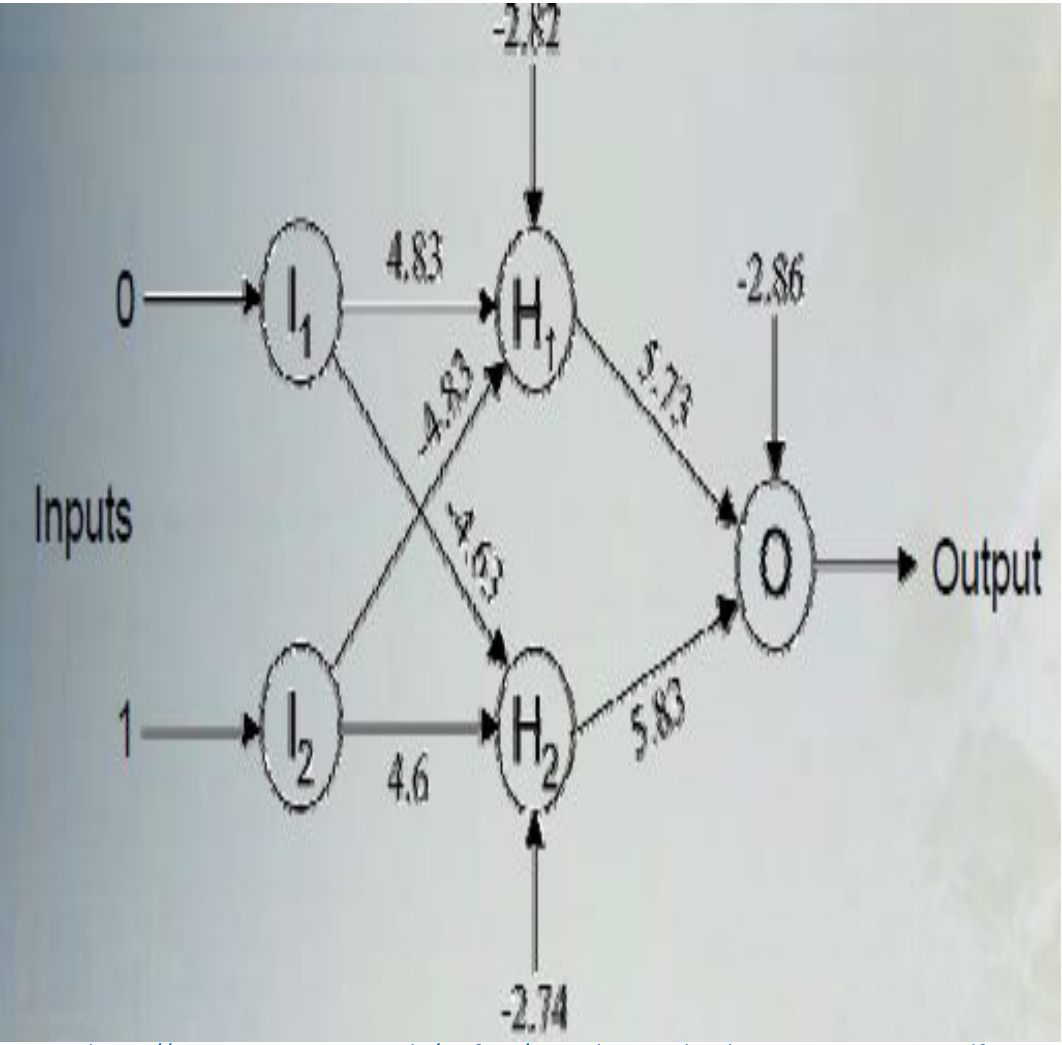
Output = 1 / (1 + $e^{-1.88}$) = 0.8652

O:

Net = ?

Output = ?

XOR Example- Using Sigmoid Function



https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Inputs: 0, 1

H_1:

Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65

Output = 1 / (1 + $e^{7.65}$) = 4.758 * 10^{-4}

H_2:

Net = 0(-4.63) + 1(4.6) -2.74 = 1.86

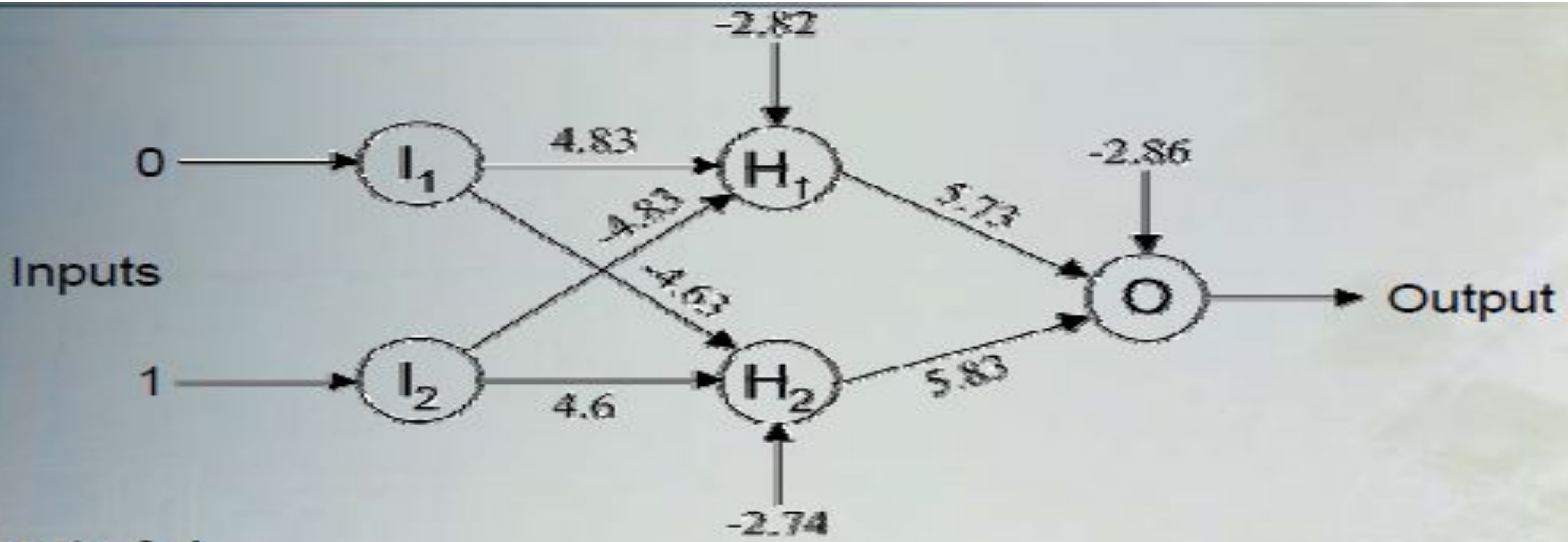
Output = 1 / (1 + $e^{-1.88}$) = 0.8652

O:

Net = 4.758 * 10^{-4}(5.73) + 0.8652(5.83) – 2.86 = 2.187

Output = 1 / (1+$e^{-2.187}$) = 0.8991 (approx. 1)

XOR Example



Inputs: 0, 1

H_1: Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65
Output = $1 / (1 + e^{7.65}) = 4.758 \times 10^{-4}$

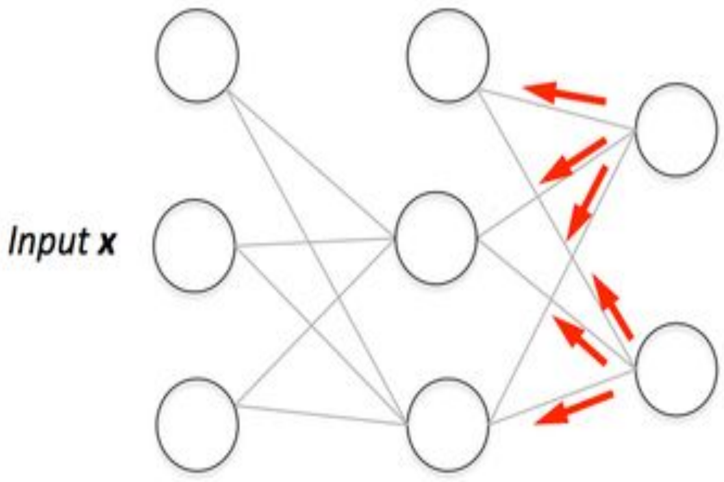
H_2: Net = 0(-4.63) + 1(4.6) – 2.74 = 1.86
Output = $1 / (1 + e^{-1.86}) = 0.8652$

O: Net = $4.758 \times 10^{-4}(5.73) + 0.8652(5.83) - 2.86 = 2.187$
Output = $1 / (1 + e^{-2.187}) = 0.8991 \equiv$ "1"

https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf

Backpropagation

- Most common method of obtaining the many weights in the network
- A form of supervised training
- The basic backpropagation algorithm is based on minimizing the error of the network using the derivatives of the error function
  - ► Simple
  - ► Slow
  - ► Prone to local minima issues



https://www.bing.com/images/search?view=detailV2&ccid=5Wr9UUgM&id=66DD02B757CF921CC8DE7526C2A8B0AA49FE6273&thid=OIP.5Wr9UUgMLWVFkXmZ_tV1UQHaFb&mediaurl=https%3a%2f%2fcdn-images-1.medium.com%2fmax%2f1600%2f0*KOoCMnuItbHKq5Xx.png&exph=356&expw=485&q=backpropagation+in+neural+network&simid=607996820675366888&ck=399E6463AC0B3362FACE33667E1DA5A4&selectedIndex=19&FORM=IRPRST&ajaxhist=0

Backpropagation

- Most common measure of error is the mean square error:

$$E = (\text{target} - \text{output})^2$$

- Partial derivatives of the error w.r.to the weights:

► Output Neurons:

let: $\delta_j = f'(net_j)\,(target_j - output_j)$ j = output neuron

$\partial E/\partial w_{ji} = -output_i\,\delta_j$ i = neuron in last hidden

► Hidden Neurons:

let: $\delta_j = f'(net_j)\,\Sigma(\delta_k w_{kj})$ j = hidden neuron

$\partial E/\partial w_{ji} = -output_i\,\delta_j$ i = neuron in previous layer

k = neuron in next layer

Backpropagation

- Calculation of the derivatives flows backwards through the network, hence the name, backpropagation
- These derivatives point in the direction of the maximum increase of the error function
- A small step (learning rate) in the opposite direction will result in the maximum decrease of the (local) error function:

$$\mathbf{w_{new} = w_{old} - \alpha\, \partial E/\partial w_{old}}$$

where α is the learning rate

Backpropagation

- The learning rate is important:
  - ► Too small: Convergence extremely slow
  - ► Too large: May not converge
- Momentum
  - ► Tends to aid convergence
  - ► Applies smoothed averaging to the change in weights:

$$\Delta_{new} = \beta\Delta_{old} - \alpha\ \partial E/\partial w_{old}, \quad \beta \text{ is the momentum coefficient}$$

$$w_{new} = w_{old} + \Delta_{new}$$

  - ► Acts as a low-pass filter by reducing rapid fluctuations

REFERENCES

- Tom M. Mitchell, *Machine Learning*, McGrawHill Publications, Indian Edition, 2017

Tutorial:

- https://www2.econ.iastate.edu/tesfatsi/NeuralNetworks.CheungCannonNotes.pdf
- https://help.imsl.com/c/6.0/stat/default.htm?turl=multilayerfeedforwardneuralnetworks.htm
- https://www.bing.com/images/search?view=detailV2&ccid=5Wr9UUgM&id=66DD02B757CF921CC8DE7526C2A8B0AA49FE6273&thid=OIP.5Wr9UUgMLWVFkXmZ_tV1UQHaFb&mediaurl=https%3a%2f%2fcdn-images-1.medium.com%2fmax%2f1600%2f0*KOoCMnultbHKq5Xx.png&exph=356&expw=485&q=backpropagation+in+neural+network&simid=607996820675366888&ck=399E6463AC0B3362FACE33667E1DA5A4&selectedIndex=19&FORM=IRPRST&ajaxhist=0