Applied Artificial Intelligence

Session 12: From a Neuron to a Neural Network

Why and How?

Fall 2018

NC State University

Lecturer: Dr. Behnam Kia

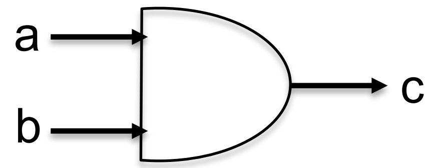
Course Website: https://appliedai.wordpress.ncsu.edu/

Oct 9, 2018

In this session we will:

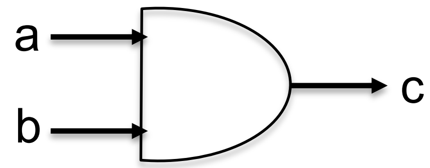
- Look at a single neuron (Logistic Regression Unit)
- And discuss some of its limitations in learning
- And we go from a single neuron to a network of neurons in order to solve the problem

AND Gate



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

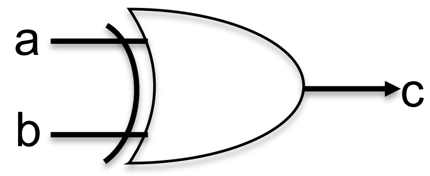
Implement an AND Gate Based Using a Logistic Regression Classifier



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Training Data
x_train= [[0,0], [0,1], [1,0],[1,1]]
y_train=[0 , 0 , 0 , 1]

Implement an XOR Gate Based Using a Logistic Regression Classifier



a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Training Data
x_train= [[0,0], [0,1], [1,0],[1,1]]
y_train=[0 , 1 , 1 , 0]

XOR Problem: What can we do?

XOR Problem: What can we do?

- Apply linear models not to input x, but to a transformed input $\varphi(x)$, where φ is a nonlinear transformation.

XOR Problem: What can we do?

- Apply linear models not to input *x*, but to a transformed input $\varphi(x)$, where φ is a nonlinear transformation.

- But how to choose φ?

XOR Problem: What can we do?

- Start to use a very generic φ.
- For example, start to use all possible nonlinear combinations of features up to some order:

  $x_1 x_2, x_1^2, x_2^2, x_1^3 \ldots$

If we have tens of features, there would be:

- hundreds of quadratic terms
- And thousands of cubic terms
- …

Too many parameters to learn, and not enough data. Overfitting can happen.

XOR Problem: What can we do?

- Manually design the transformations, e.g.:
  - $x_1 \sin x_2,\ x_3\sqrt{x_1}$

- This is an ad hoc, domain specific effort.
- Very hard, requires decades of human effort for each task.

XOR Problem: What can we do?

- Multilayer Neural Network (AKA Deep Learning):
  - The strategy is to learn φ as well

$\hat{y}$= $\varphi(x, w_\varphi). w_C$

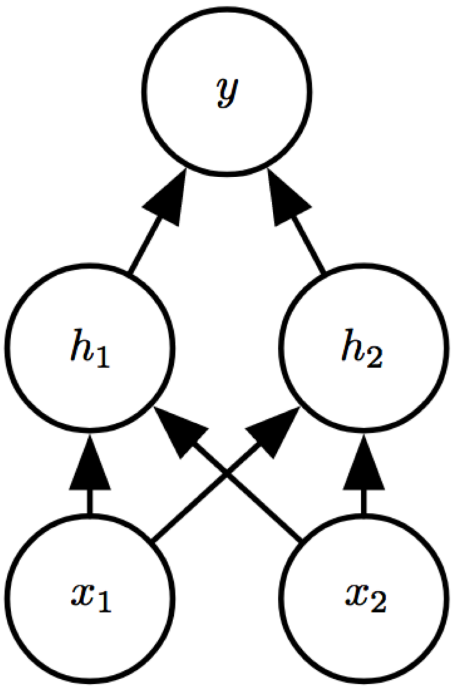
Up until now we had: $\hat{y}$=$x. w$

XOR Problem: What can we do?

- Multilayer Neural Network (AKA Deep Learning):
  - The strategy is to learn φ as well

$\hat{y} = \varphi(x, w_{\varphi}).w_C$

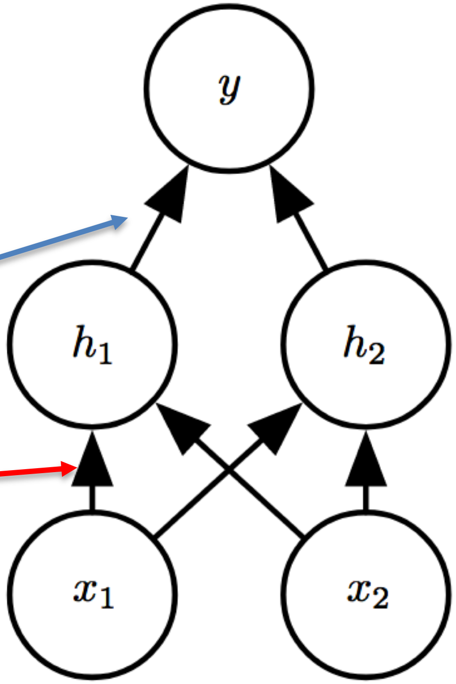
Up until now we had: $\hat{y} = x.w$



XOR Problem: What can we do?

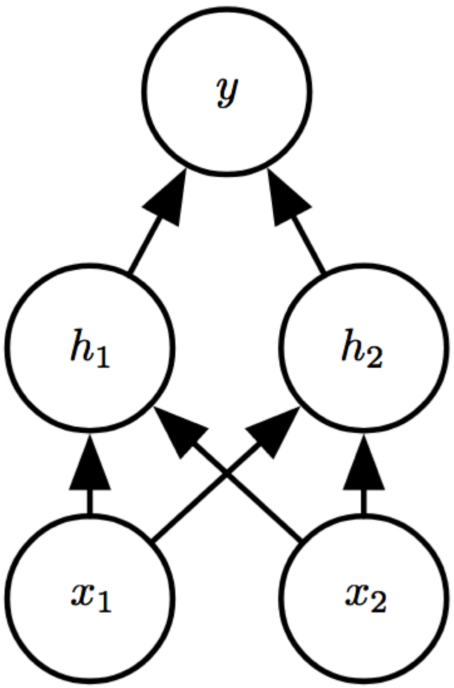
- Multilayer Neural Network (AKA Deep Learning):
  - The strategy is to learn φ as well

$\hat{y}= \varphi(x, w_{\varphi}).w_C$



Up until now we had: $\hat{y}=x.w$

Multilayer Neural Network (AKA Deep Learning)



- Logistic unit ⇨ Neuron Model
- Parameters ⇨ weights
- Output function ⇨ Activation function

XOR Problem: Multilayer Neural Network Solution

XOR is not linearly separable

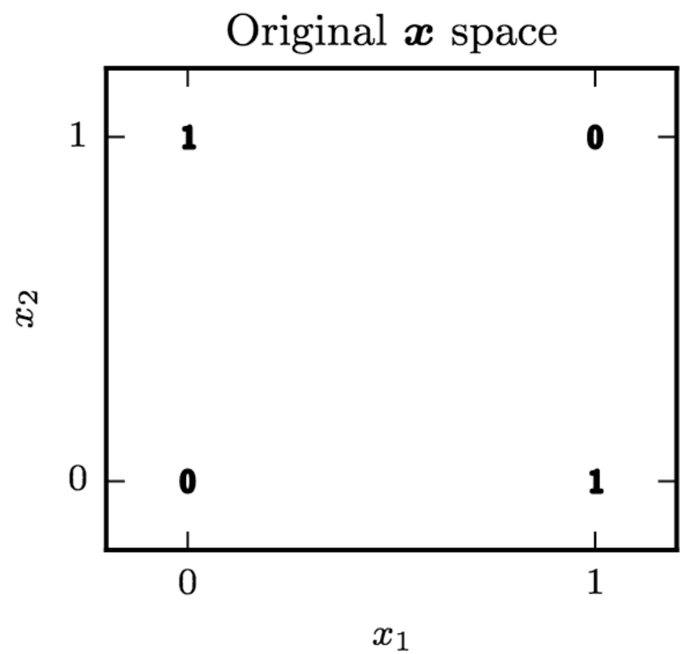


Figure 6.1, left

(Goodfellow 2017)

Slides are from Goodfellow, et.al for "Deep Learning" Book

XOR Problem: Multilayer Neural Network Solution

XOR is not linearly separable

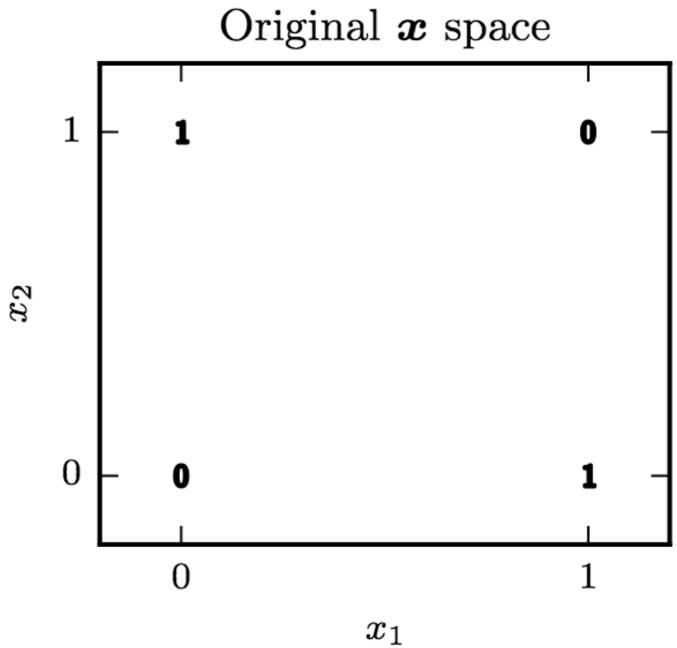
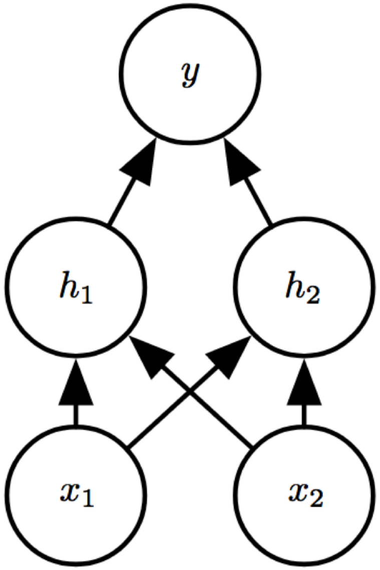


Figure 6.1, left



(Goodfellow 2017)

Slides are from Goodfellow, et.al for "Deep Learning" Book

XOR Problem: Multilayer Neural Network Solution

XOR is not linearly separable

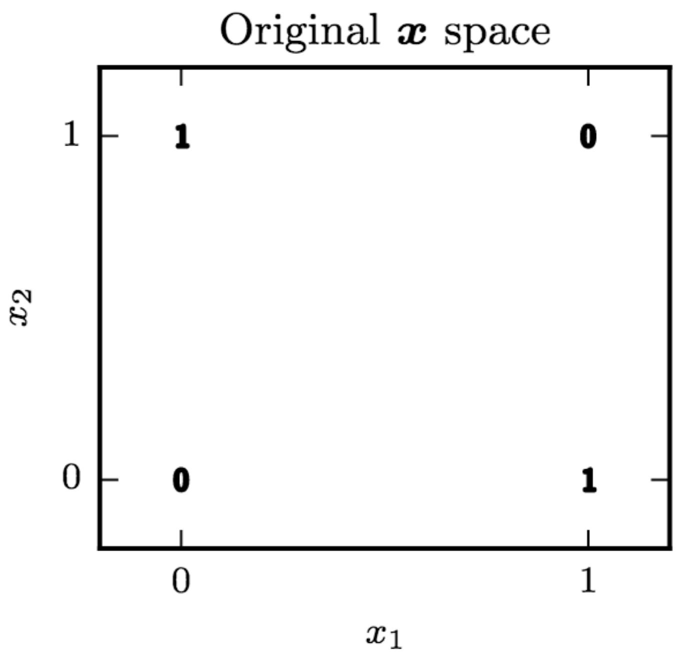
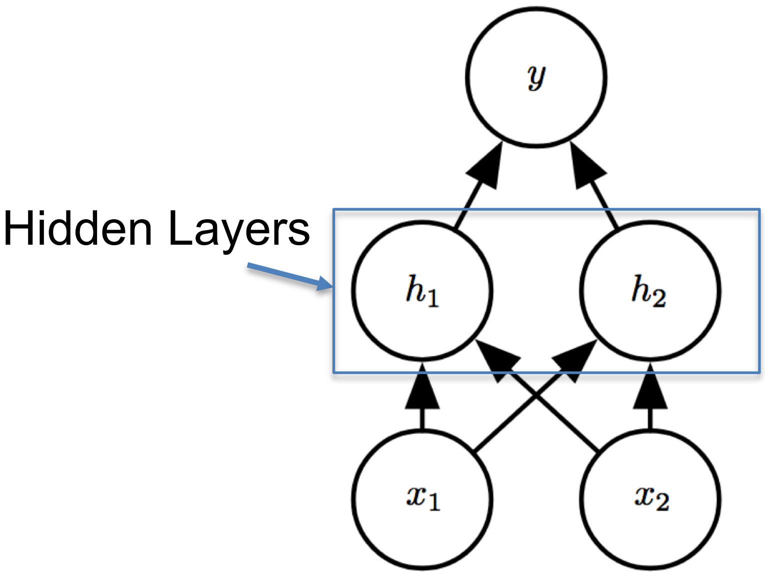


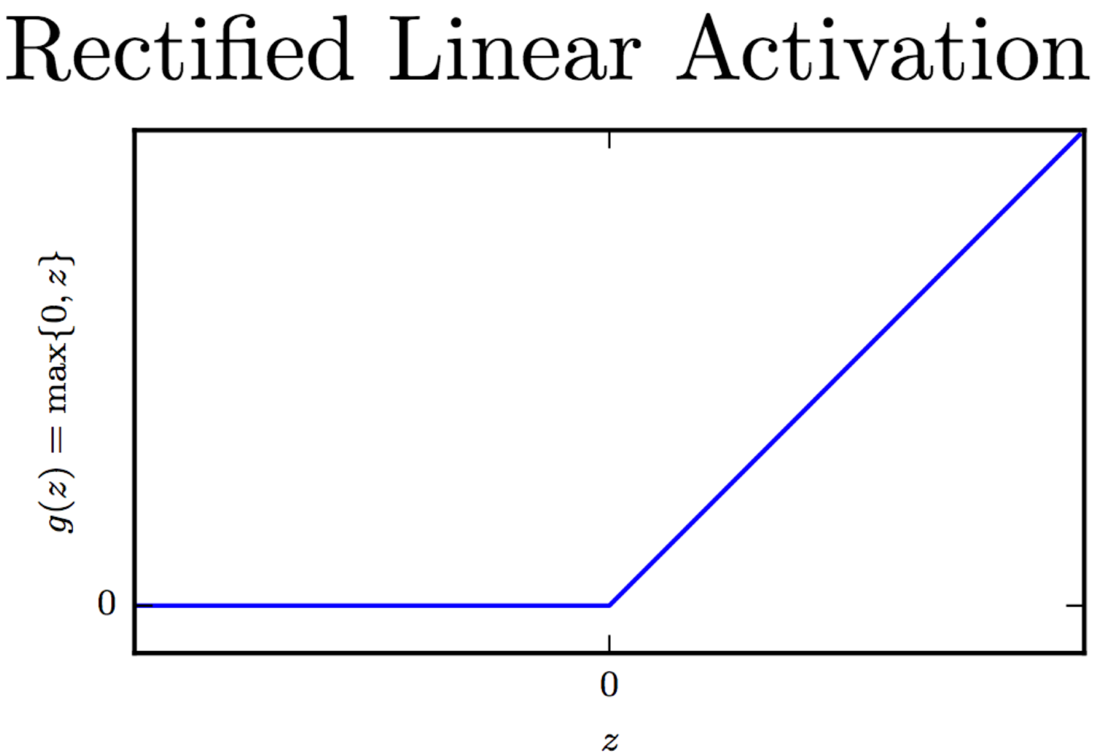
Figure 6.1, left

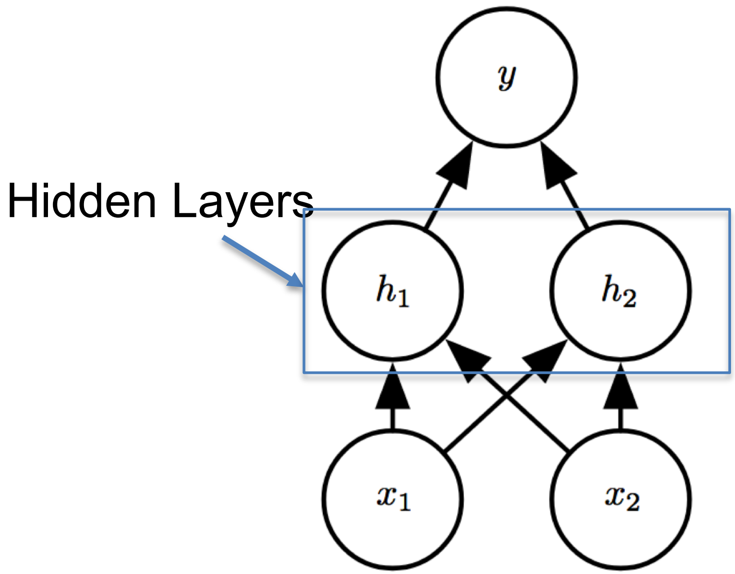


(Goodfellow 2017)

Slides are from Goodfellow, et.al for “Deep Learning” Book

XOR Problem: Multilayer Neural Network Solution





Slides are from Goodfellow, et.al for “Deep Learning” Book

XOR Problem: Multilayer Neural Network Solution

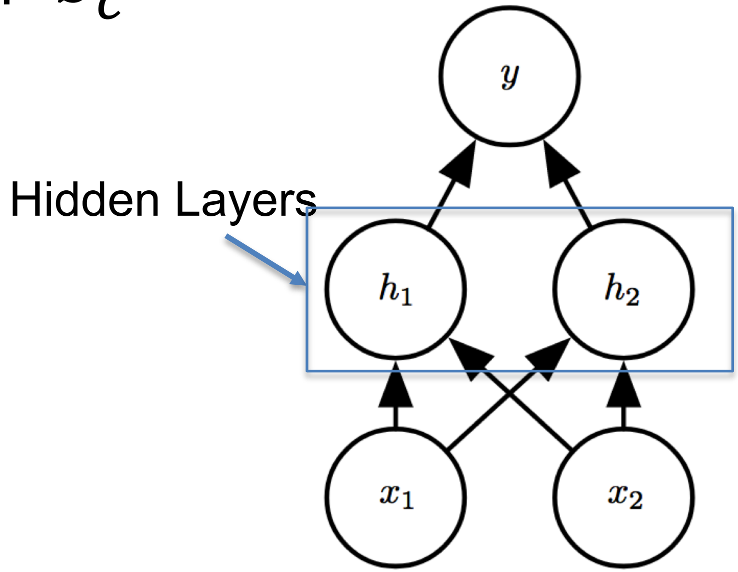
$$\hat{y} = \max\{0, x.w_\varphi + b_\varphi\}.w_c + b_c$$

$$w_\varphi = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b_\varphi = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$w_c = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$b_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Slides are from Goodfellow, et.al for “Deep Learning” Book

XOR Problem: Multilayer Neural Network Solution

$$\hat{y} = \boxed{\max\{0, x.w_\varphi + b_\varphi\}}.w_c + b_c$$

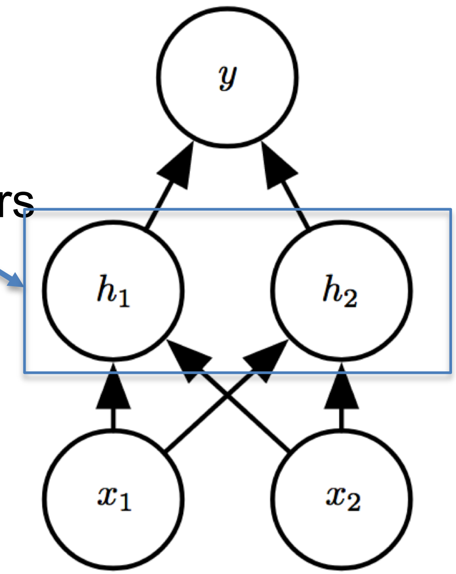
Hidden Layers

$$w_\varphi = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b_\varphi = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

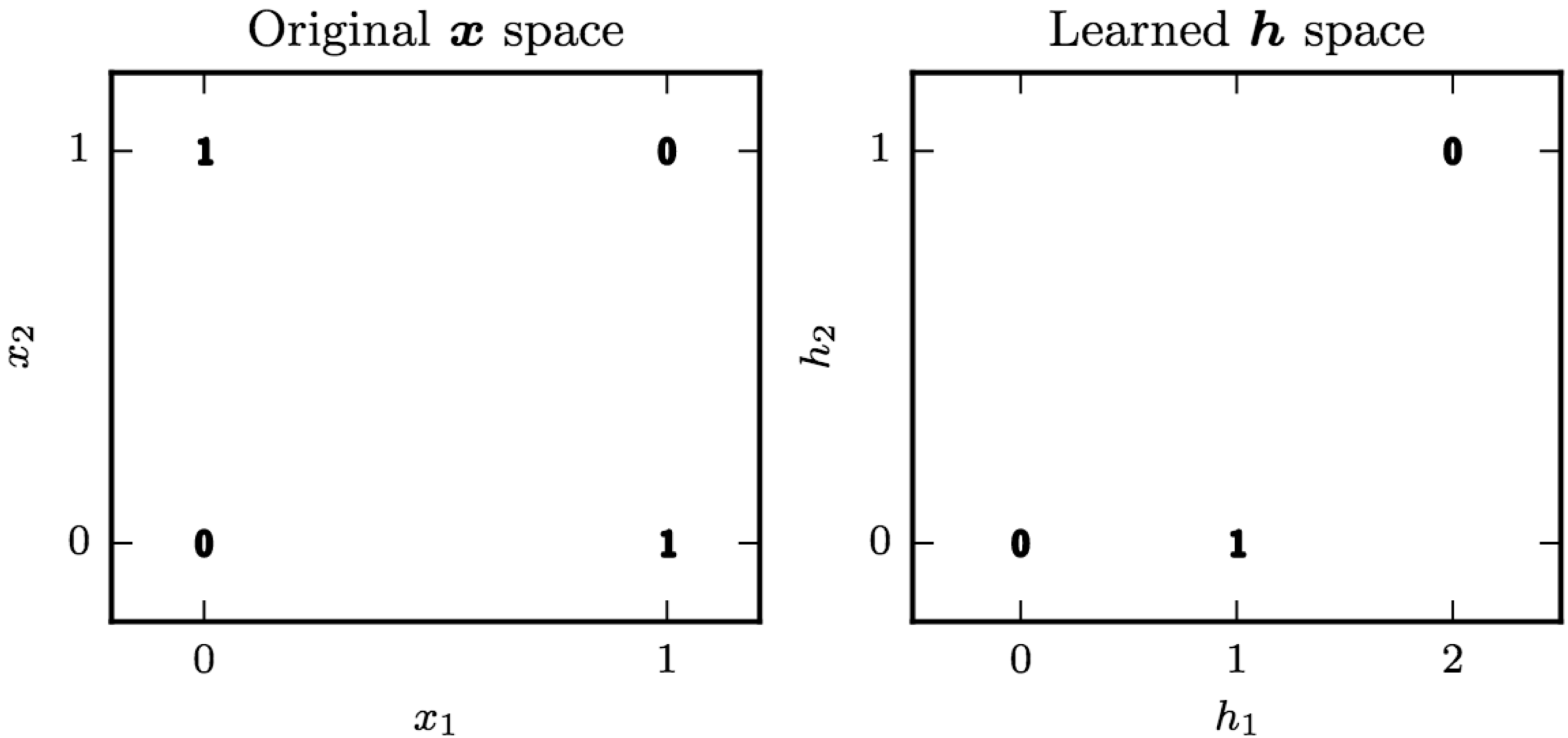
$$w_c = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$b_c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Slides are from Goodfellow, et.al for “Deep Learning” Book

XOR Problem: Multilayer Neural Network Solution



Multilayer Feedforward Neural Network (Deep Feedforward Networks)

Good News

&

Bad News

Multilayer Feedforward Neural Network (Deep Feedforward Networks)

Good News

A Feedforward Neural Network with one hidden layer can represent and approximate any function to an arbitrary degree of accuracy.

This is called Universal Approximator Theorem.

Multilayer Feedforward Neural Network (Deep Feedforward Networks) Good News

- Deeper networks are much more powerful than shallow networks.
- Shallow network may need exponentially more width (neurons in a layer) to implement the same function.

Eldan, Ronen, and Ohad Shamir. "**The power of depth for feedforward neural networks**." *Conference on Learning Theory*. 2016.

J. Hastad. **Almost optimal lower bounds for small depth circuits**. ACM symposium on Theory of computing,. ACM, 1986

Inception (2010)

Multilayer Feedforward Neural Network (Deep Feedforward Networks)

Bad News

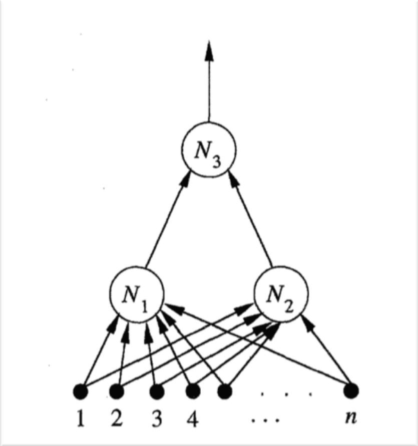
Training a 3-Node Neural Network is NP-Complete

*Avrim L. Blum and Ronald L. Rivest**

MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139
avrim@theory.lcs.mit.edu
rivest@theory.lcs.mit.edu

4.1 Introduction

One reason for the recent surge in interest in feed-forward neural networks is the development of the "back-propagation" training algorithm [14]. The ability to train large multi-layer networks is essential for utilizing neural networks in prac-



Multilayer Feedforward Neural Network (Deep Feedforward Networks)
Bad News

Universal Approximator Theorem says that a large enough network can approximate any function.

But it doesn't say how to train the network or learn those parameters.