Deep Feedforward Networks

Lecture slides for Chapter 6 of *Deep Learning*

www.deeplearningbook.org

Ian Goodfellow

Last updated 2016-10-04

Roadmap

- Example: Learning XOR
- Gradient-Based Learning
- Hidden Units
- Architecture Design
- Back-Propagation

XOR is not linearly separable

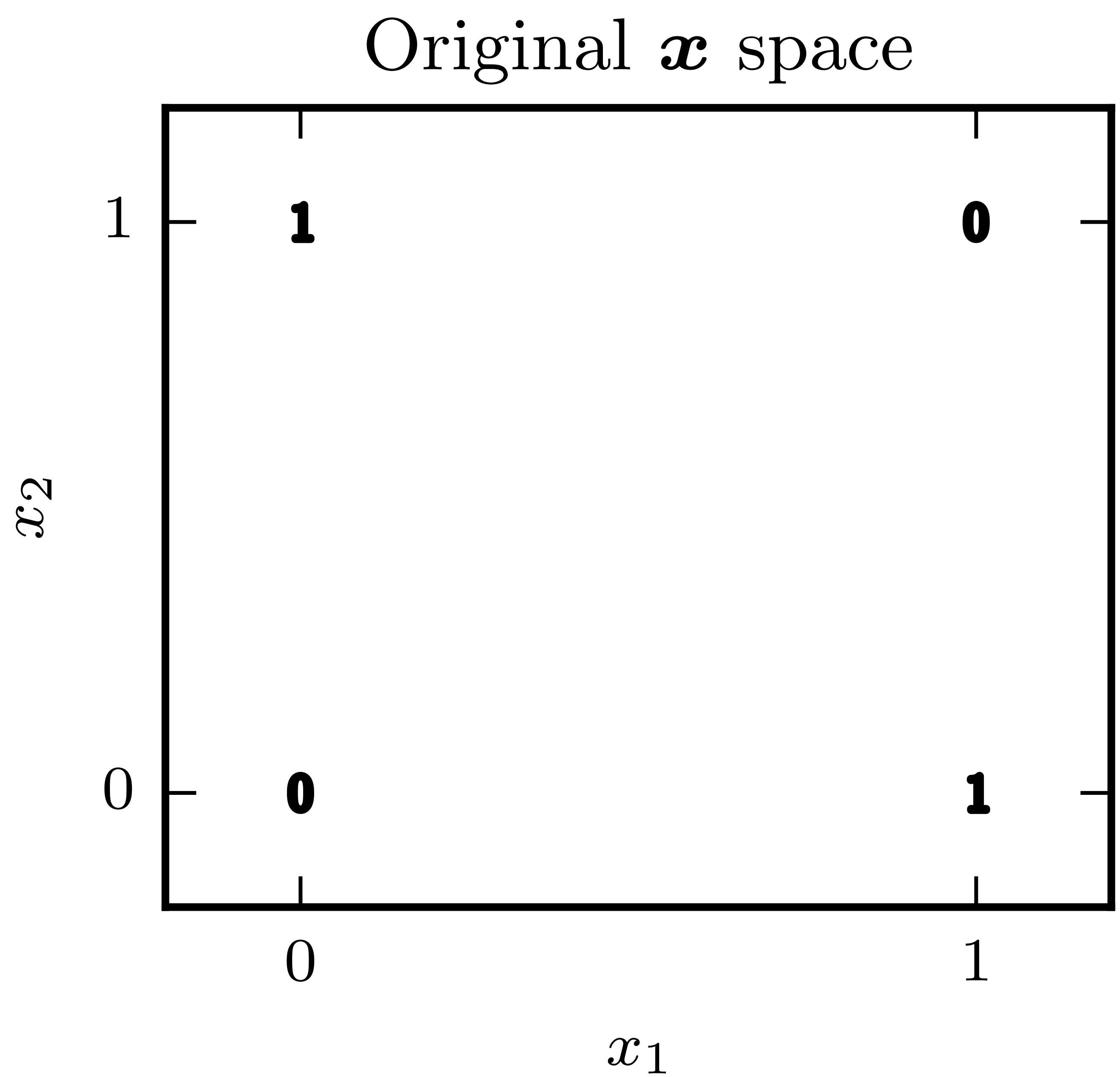


Figure 6.1, left

(Goodfellow 2017)

Rectified Linear Activation

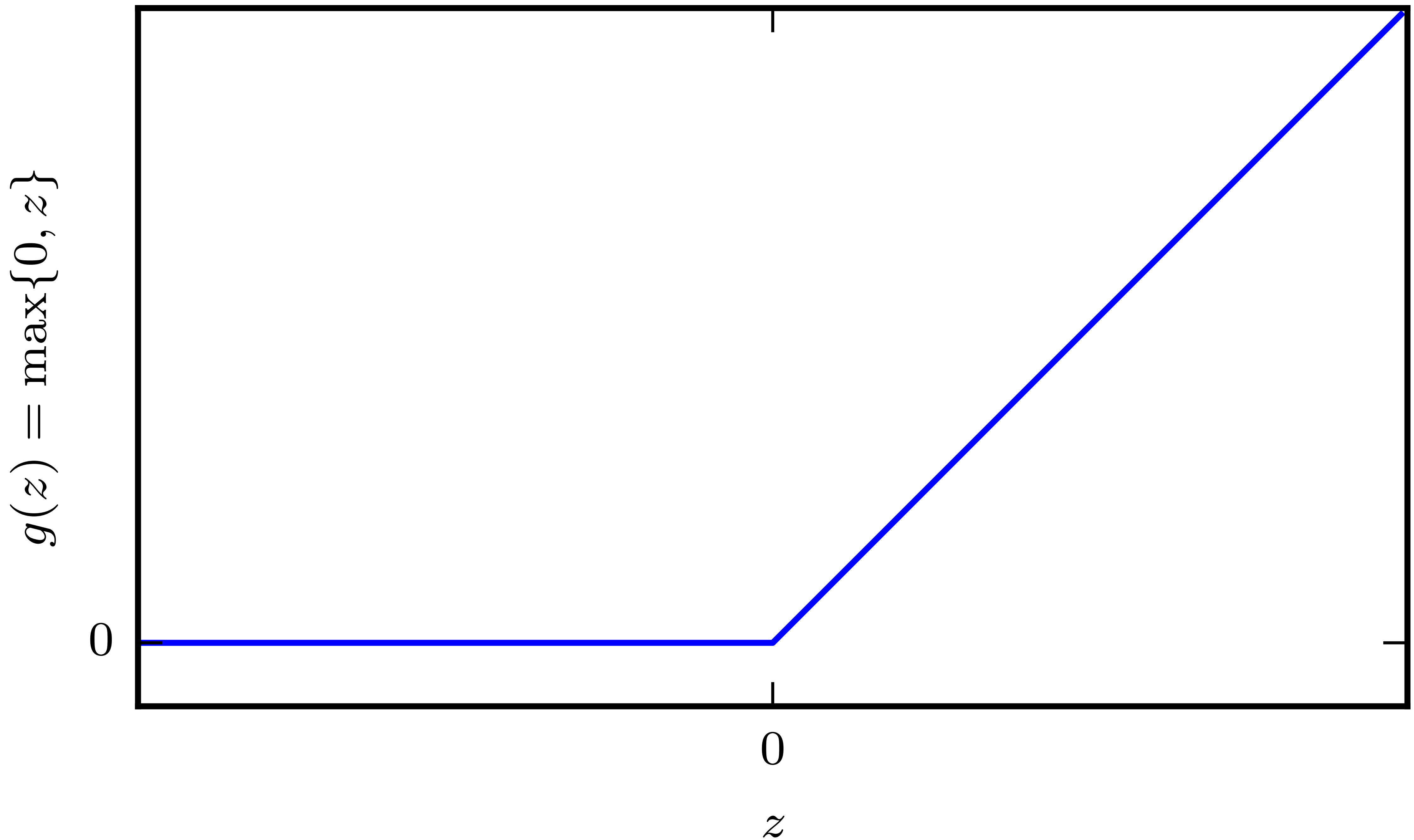


Figure 6.3

(Goodfellow 2017)

Network Diagrams

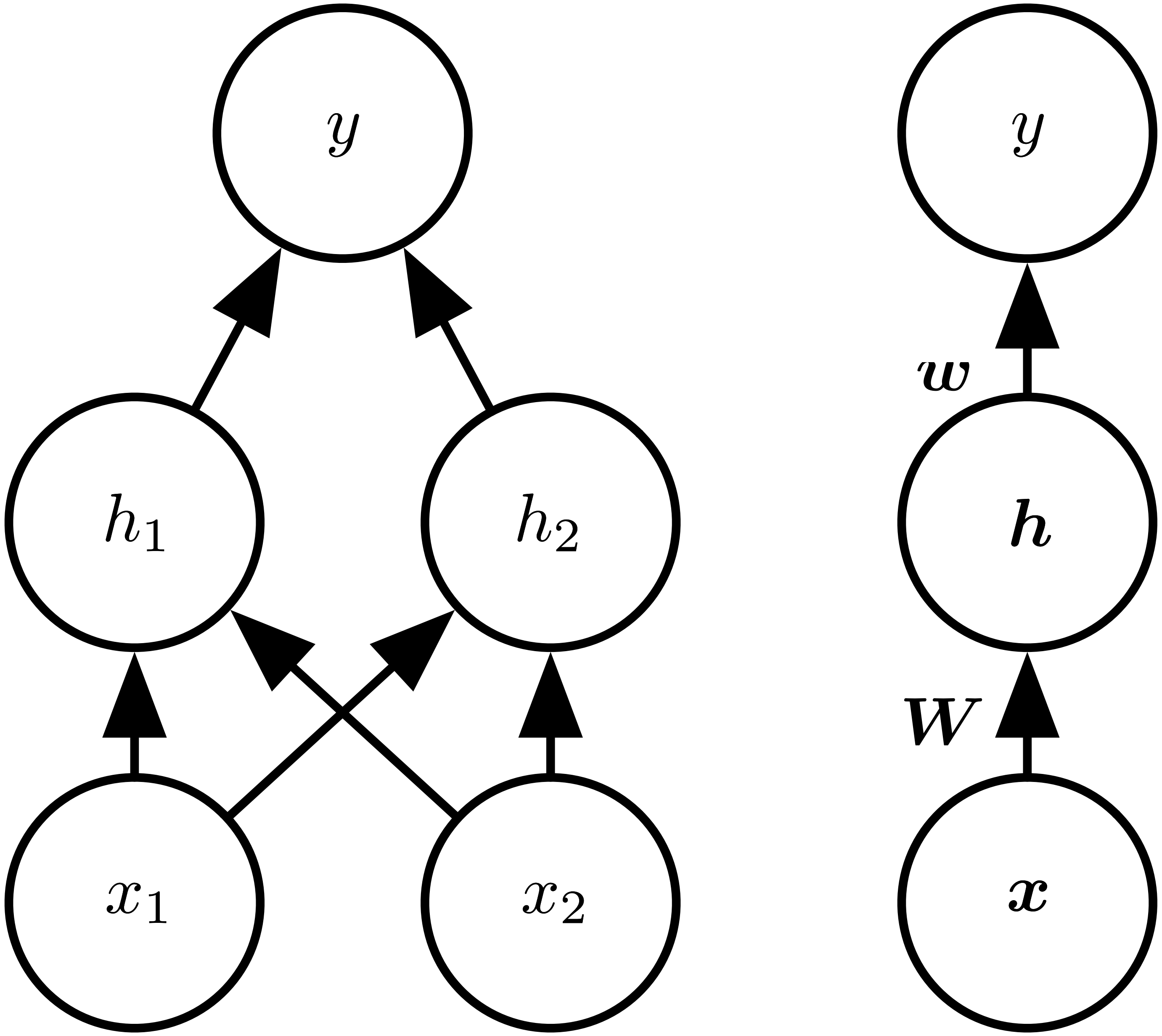


Figure 6.2

Solving XOR

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b. \tag{6.3}$$

$$\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \tag{6.4}$$

$$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \tag{6.5}$$

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \tag{6.6}$$

(Goodfellow 2017)

Solving XOR

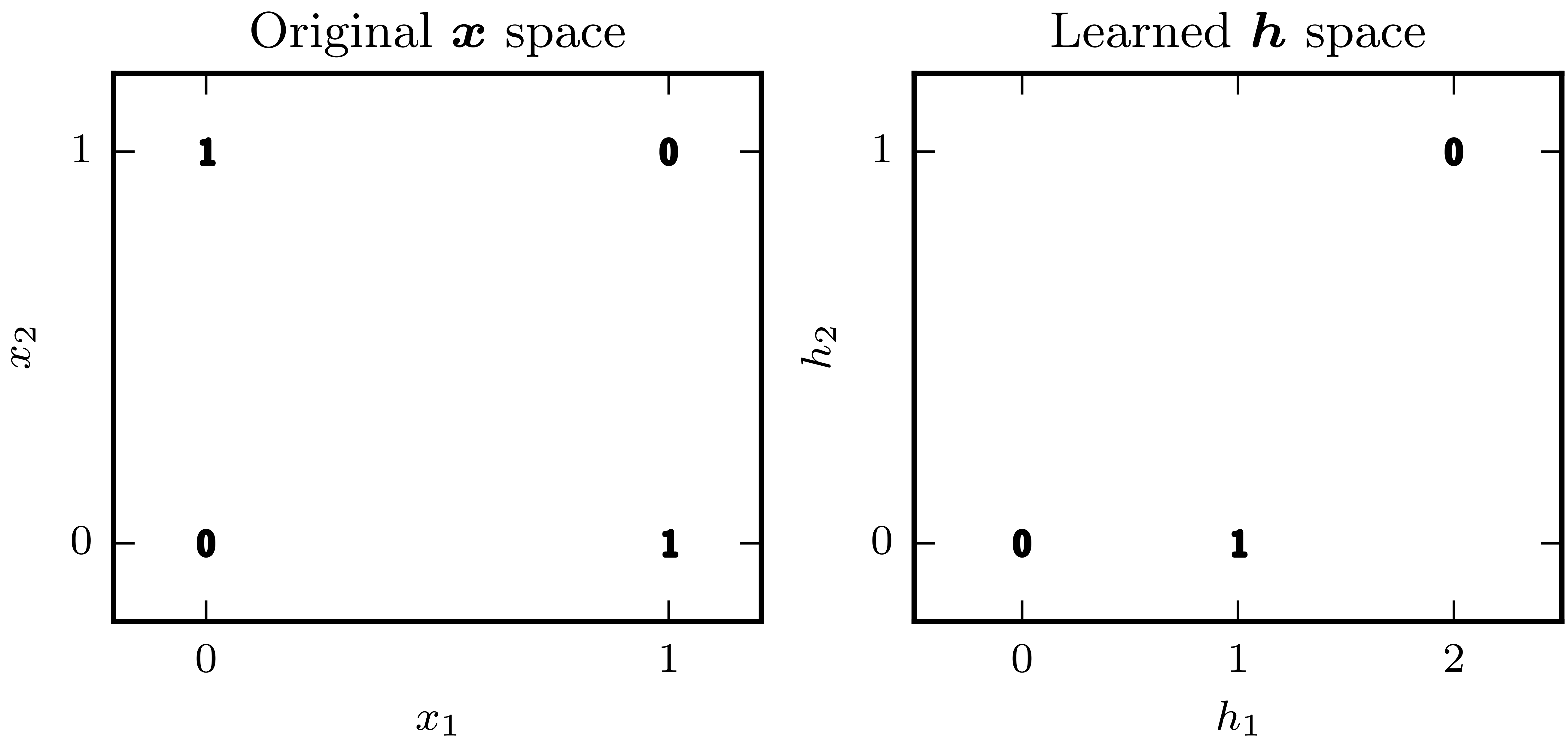


Figure 6.1

(Goodfellow 2017)

Roadmap

- Example: Learning XOR
- Gradient-Based Learning
- Hidden Units
- Architecture Design
- Back-Propagation

Gradient-Based Learning

- Specify
    - Model
    - Cost
- Design model and cost so cost is smooth
- Minimize cost using gradient descent or related techniques

Conditional Distributions and Cross-Entropy

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{y} \mid \boldsymbol{x}). \tag{6.12}$$

Output Types

Output Type	Output Distribution	Output Layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary cross-entropy
Discrete	Multinoulli	Softmax	Discrete cross-entropy
Continuous	Gaussian	Linear	Gaussian cross-entropy (MSE)
Continuous	Mixture of Gaussian	Mixture Density	Cross-entropy
Continuous	Arbitrary	See part III: GAN, VAE, FVBN	Various

(Goodfellow 2017)

Mixture Density Outputs

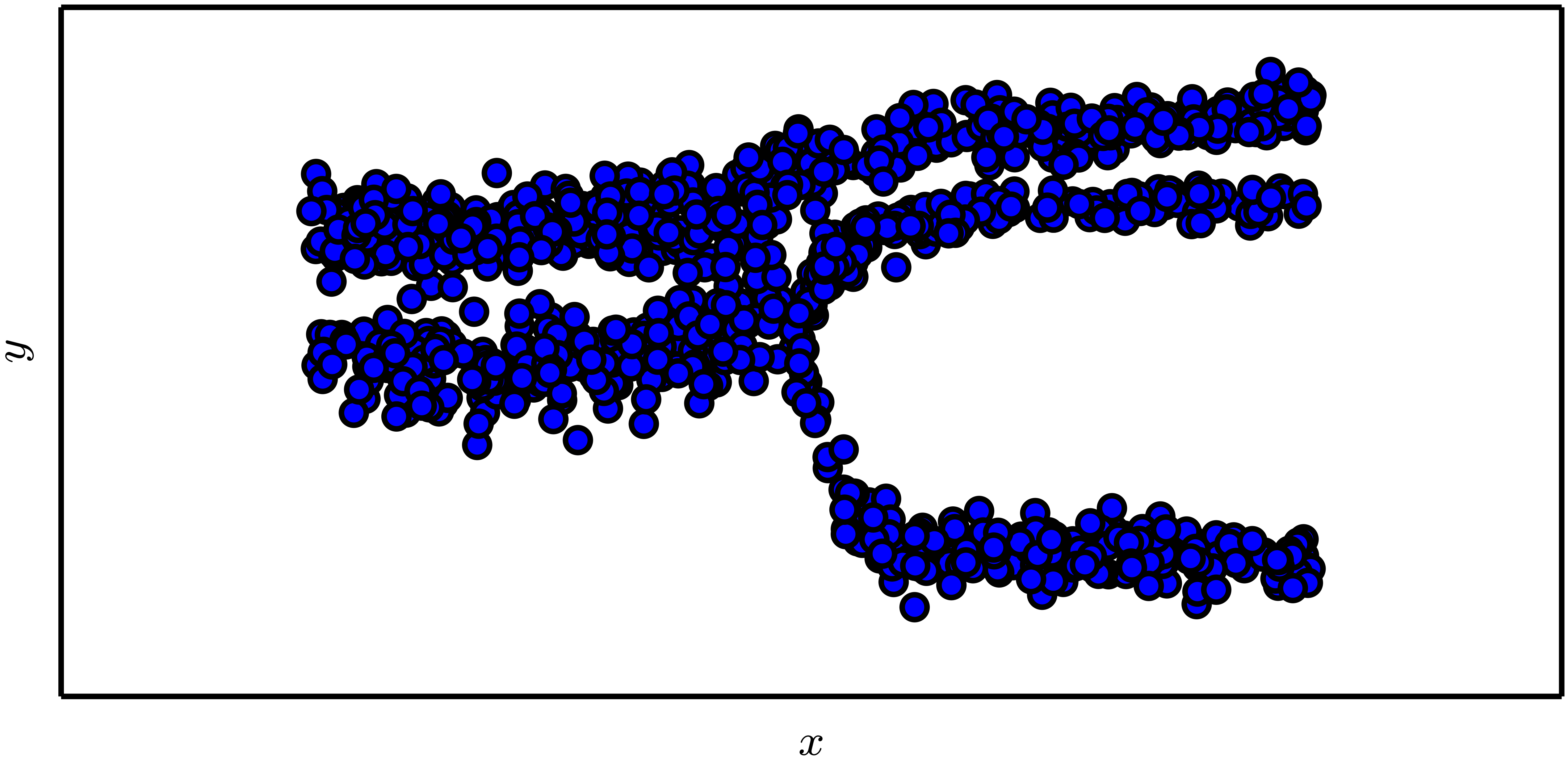
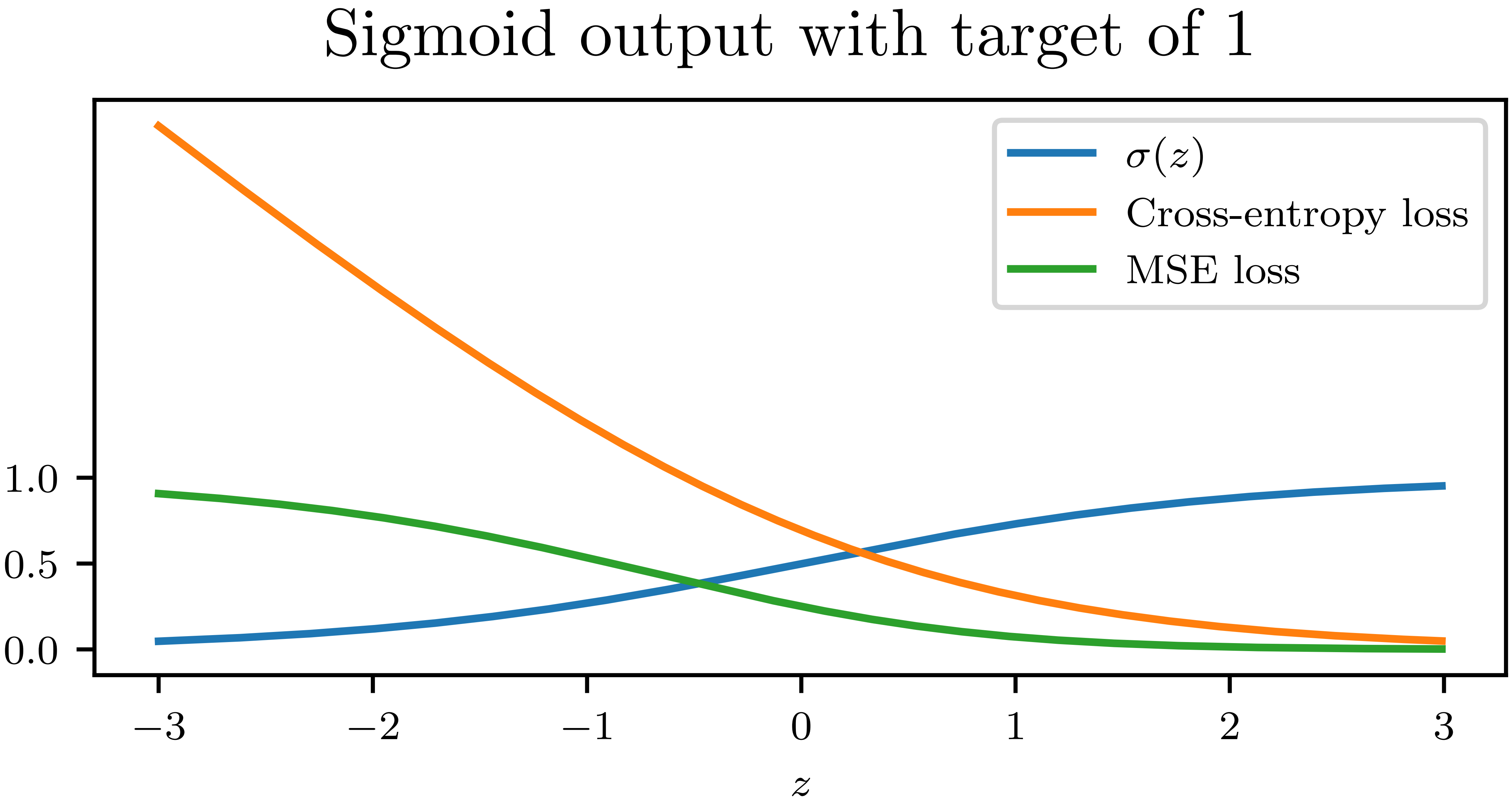


Figure 6.4

Don’t mix and match



(Goodfellow 2017)

Roadmap

- Example: Learning XOR
- Gradient-Based Learning
- Hidden Units
- Architecture Design
- Back-Propagation

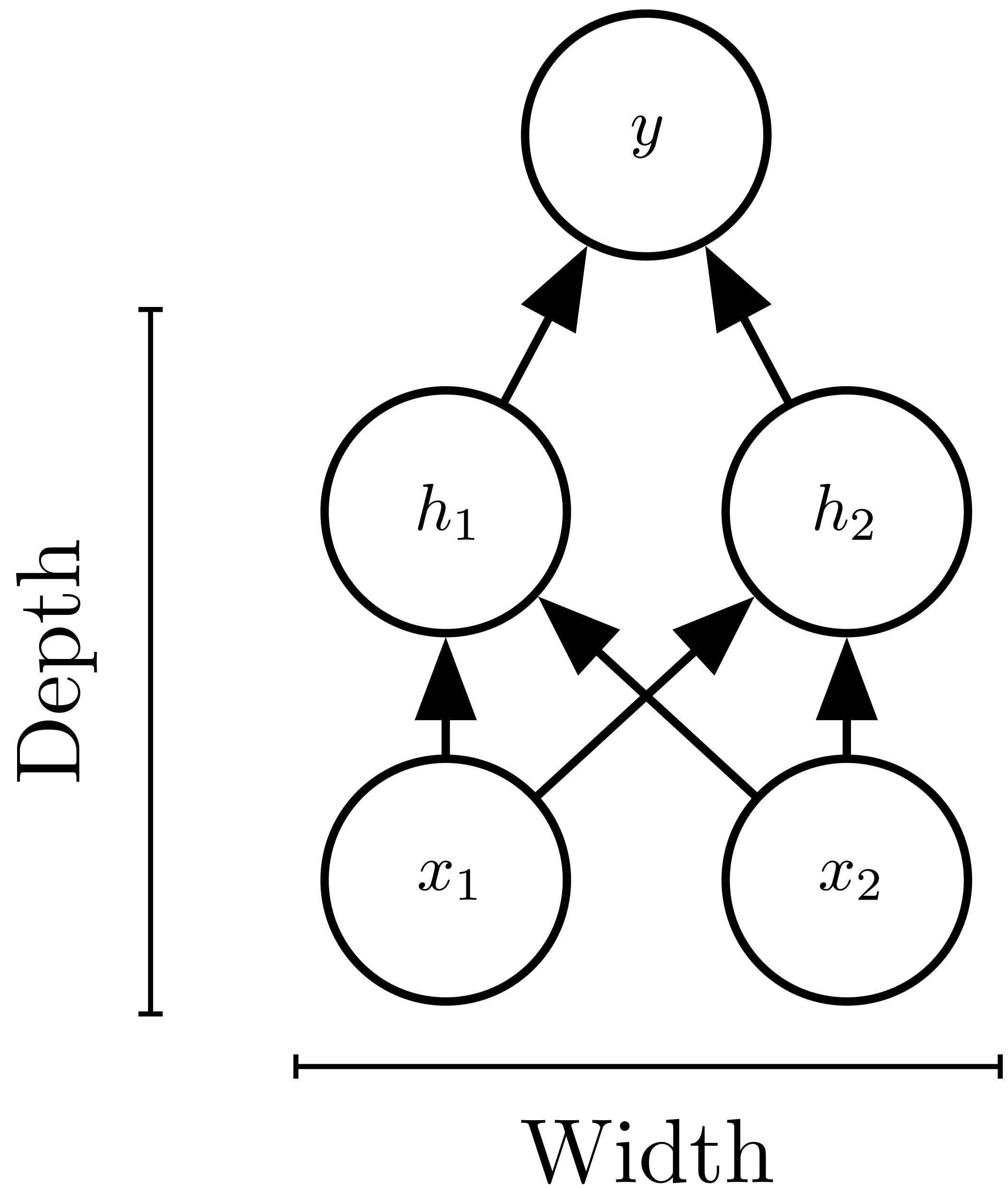
Hidden units

- Use ReLUs, 90% of the time
- For RNNs, see Chapter 10
- For some research projects, get creative
- Many hidden units perform comparably to ReLUs. New hidden units that perform comparably are rarely interesting.

Roadmap

- Example: Learning XOR
- Gradient-Based Learning
- Hidden Units
- Architecture Design
- Back-Propagation

Architecture Basics



(Goodfellow 2017)

Universal Approximator Theorem

- One hidden layer is enough to *represent* (not *learn*) an approximation of any function to an arbitrary degree of accuracy

- So why deeper?

    - Shallow net may need (exponentially) more width

    - Shallow net may overfit more

(Goodfellow 2017)

Exponential Representation Advantage of Depth

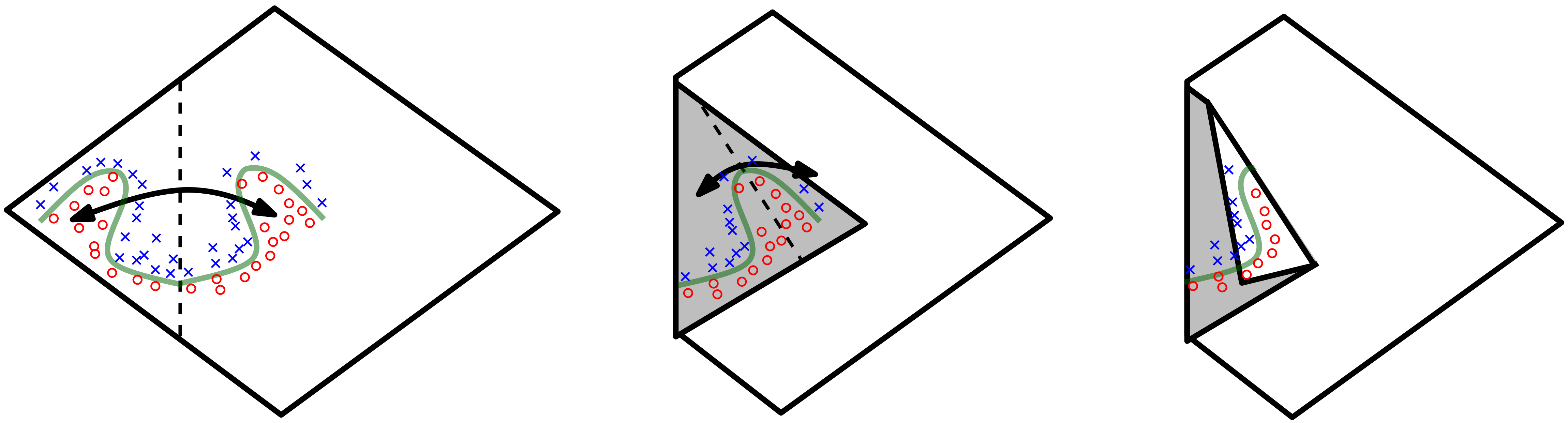


Figure 6.5

(Goodfellow 2017)

Better Generalization with Greater Depth

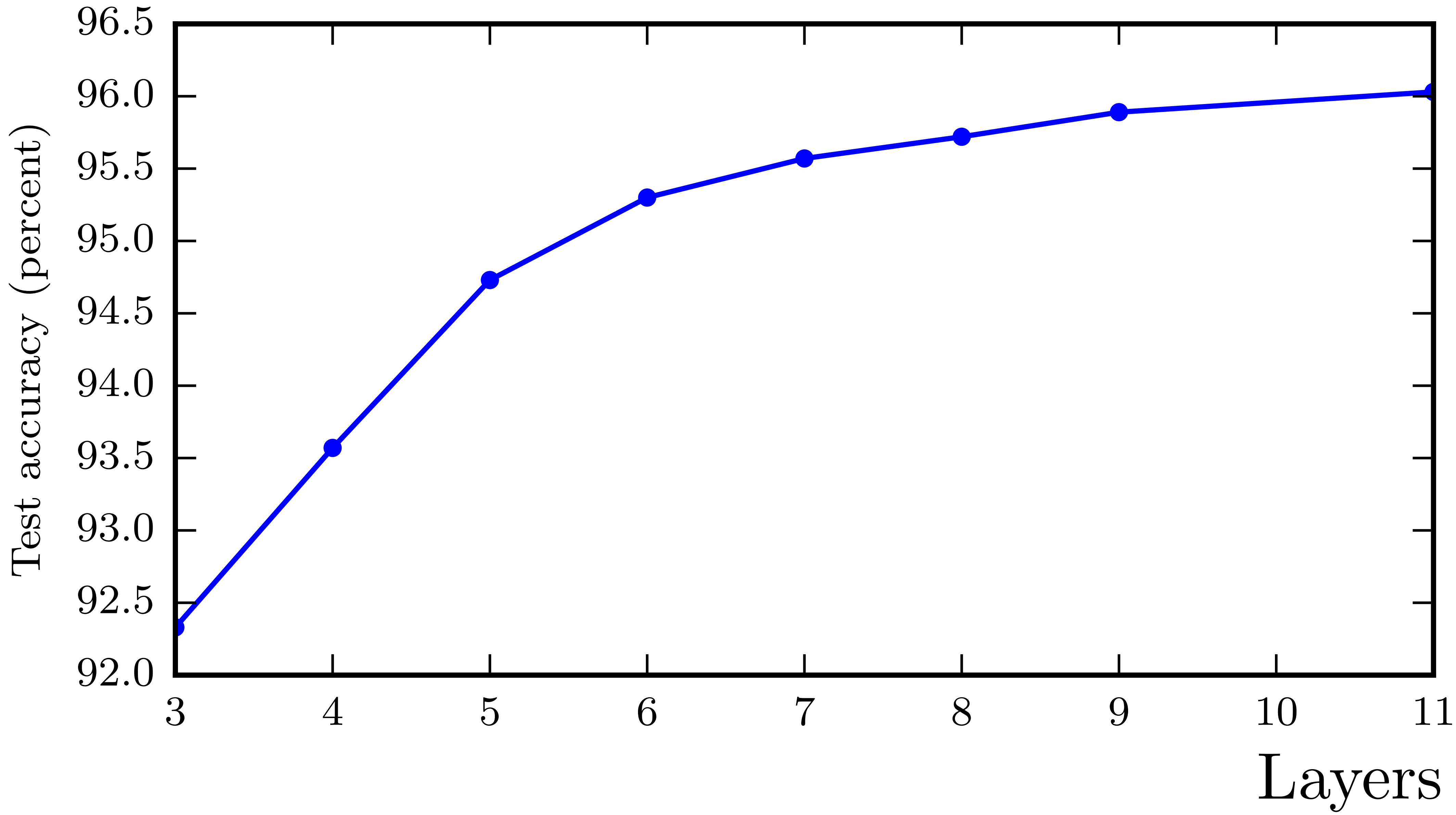


Figure 6.6

(Goodfellow 2017)

Large, Shallow Models Overfit More

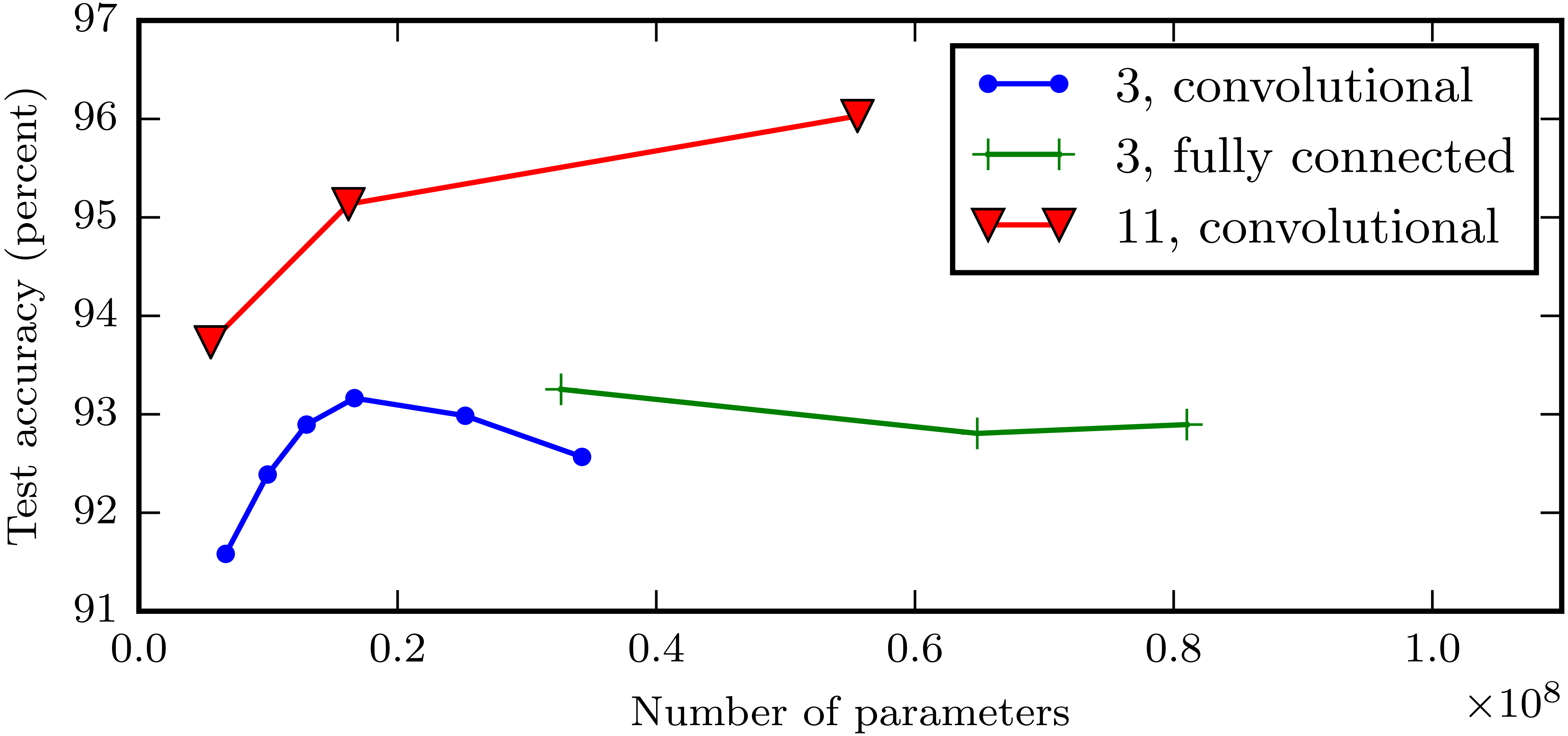


Figure 6.7

(Goodfellow 2017)

Roadmap

- Example: Learning XOR
- Gradient-Based Learning
- Hidden Units
- Architecture Design
- Back-Propagation

(Goodfellow 2017)

Back-Propagation

- Back-propagation is "just the chain rule" of calculus

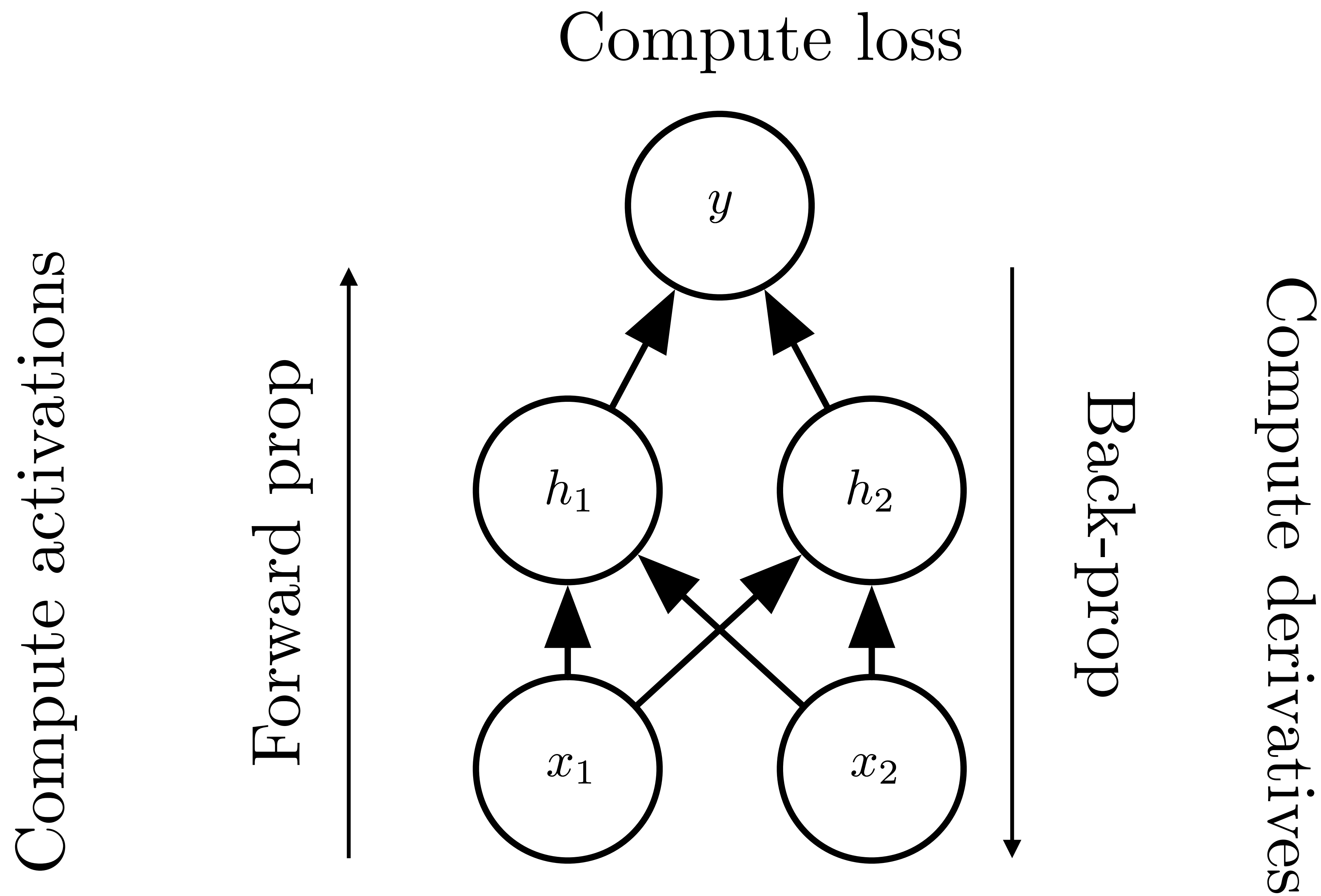
$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}. \tag{6.44}$$

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z, \tag{6.46}$$

- But it's a particular implementation of the chain rule
  - Uses dynamic programming (table filling)
  - Avoids recomputing repeated subexpressions
  - Speed vs memory tradeoff

(Goodfellow 2017)

Simple Back-Prop Example



(Goodfellow 2017)

Computation Graphs

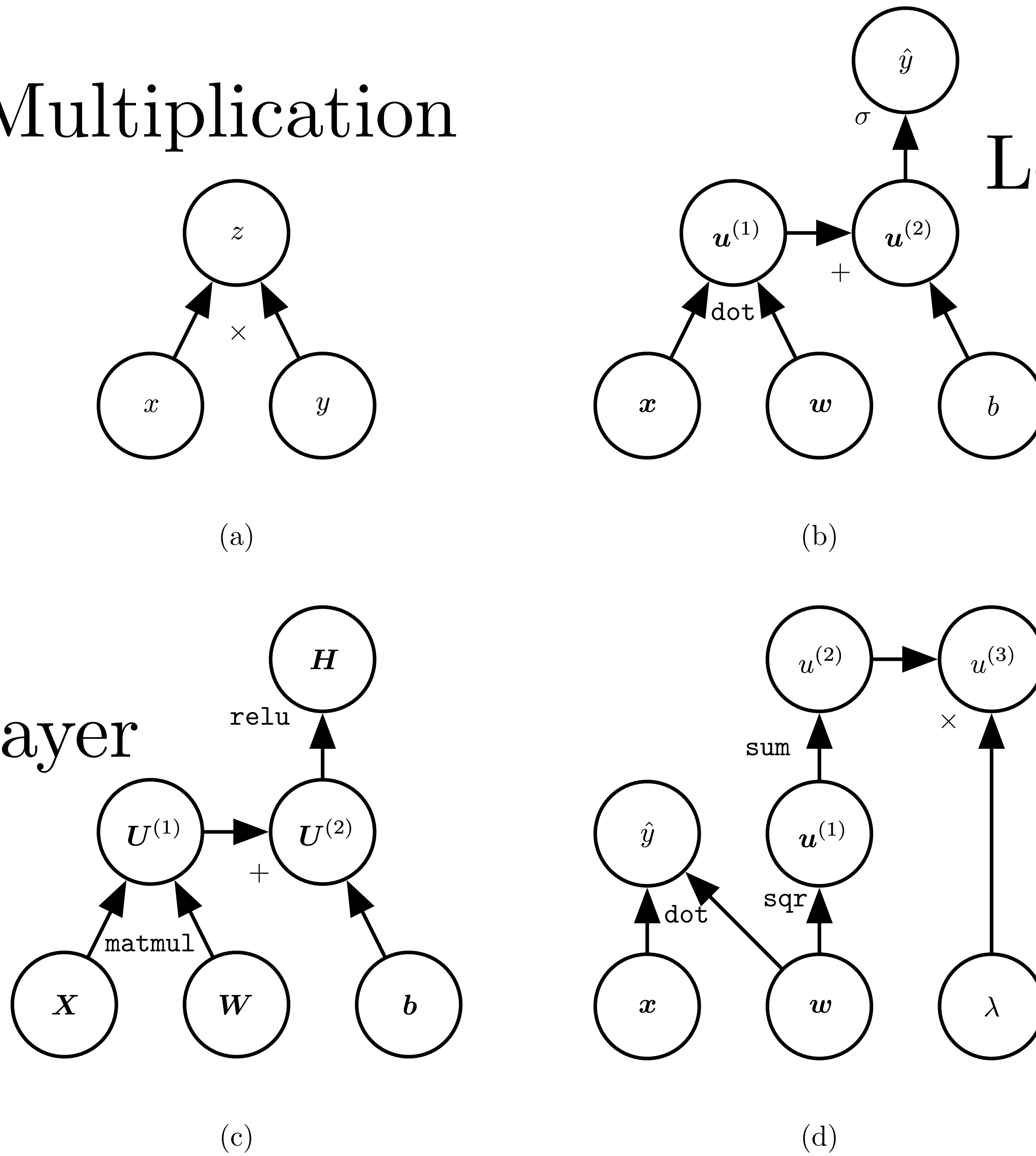
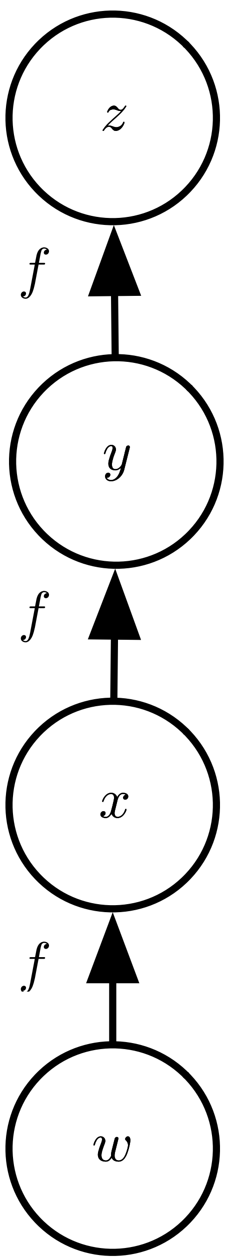


Figure 6.8

(Goodfellow 2017)

Repeated Subexpressions



$$\frac{\partial z}{\partial w} \tag{6.50}$$

$$= \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w} \tag{6.51}$$

$$= f'(y) f'(x) f'(w) \tag{6.52}$$

$$= f'(f(f(w))) f'(f(w)) f'(w) \tag{6.53}$$

Back-prop avoids computing this twice

Figure 6.9

(Goodfellow 2017)

Symbol-to-Symbol Differentiation

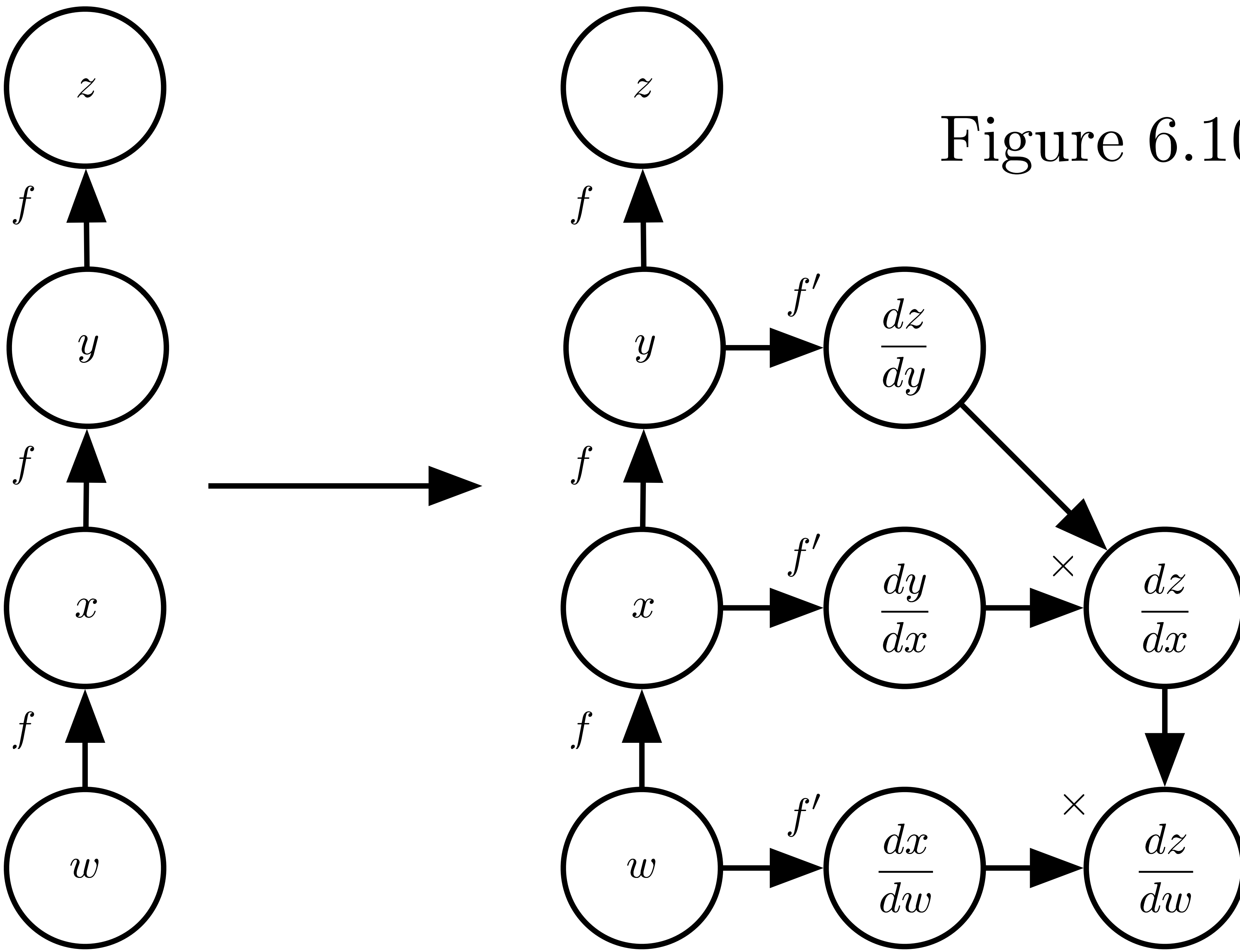


Figure 6.10

(Goodfellow 2017)

Neural Network Loss Function

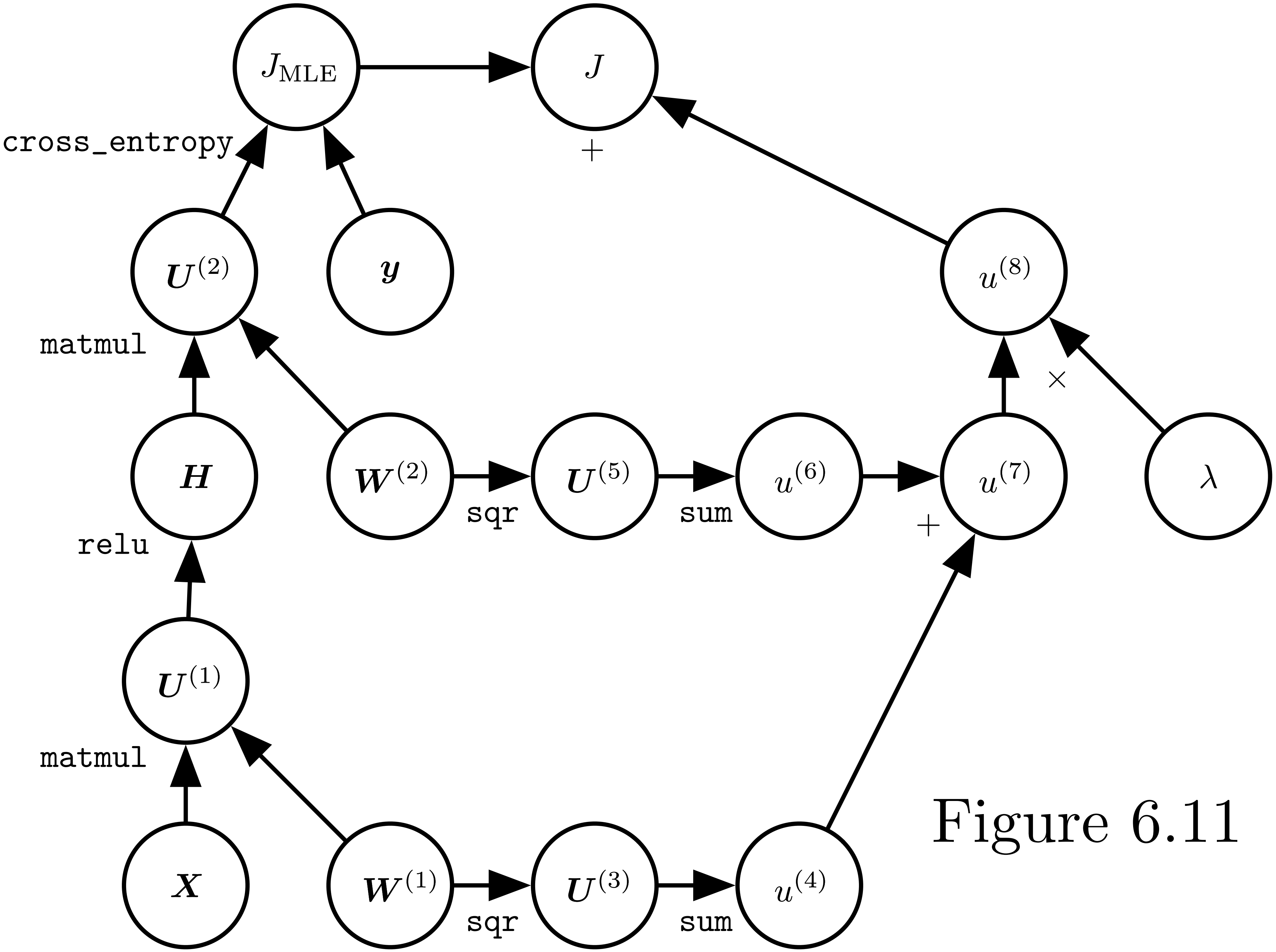


Figure 6.11

(Goodfellow 2017)

Hessian-vector Products

$$\boldsymbol{H}\boldsymbol{v} = \nabla_{\boldsymbol{x}} \left[\left(\nabla_{\boldsymbol{x}} f(x) \right)^{\top} \boldsymbol{v} \right]. \tag{6.59}$$

(Goodfellow 2017)

Questions