Chapter 10: Deep Learning Basics

DATA SCIENCE IN SECURITY

Introduction

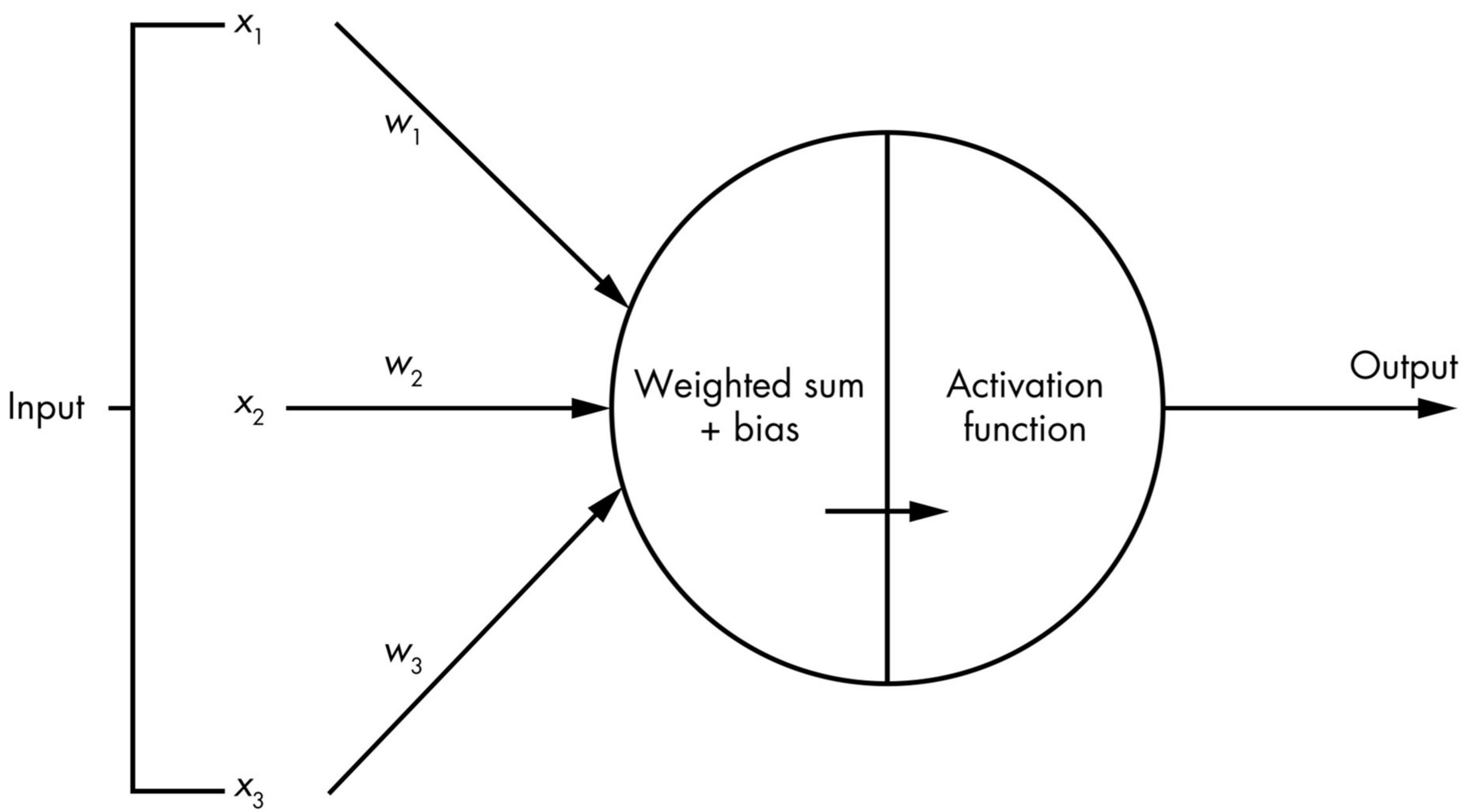
- what is Deep learning?
  - just a type of machine learning
    - But it often leads to models that achieve better accuracy
  - Deep learning models learn to view their training data as a nested hierarchy of concepts
    - automatically combine input features to form new, optimized meta-features
      - which they then combine to form even more features, and so on.
    - allows them to represent incredibly complex patterns

Introduction

- what is Deep learning?
  - “Deep” also refers to the architecture
    - usually consists of multiple layers of processing units
      - each using the previous layer’s outputs as its inputs.
    - Each of these processing units is called a neuron
      - the model architecture as a whole is called
        - a neural network
        - or a deep neural network when there are many layers.

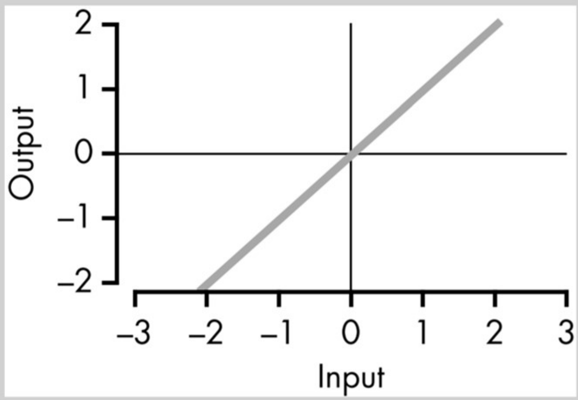
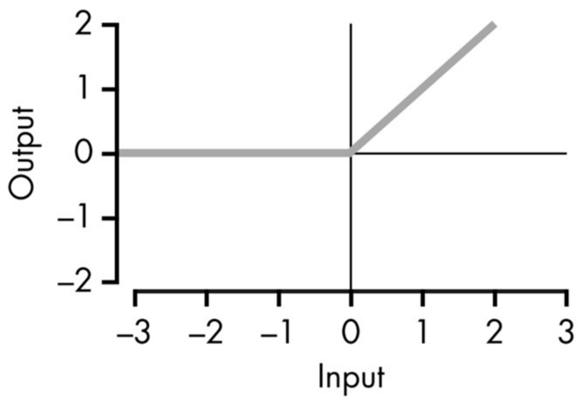
how neural networks work

- Anatomy of a Neuron



how neural networks work

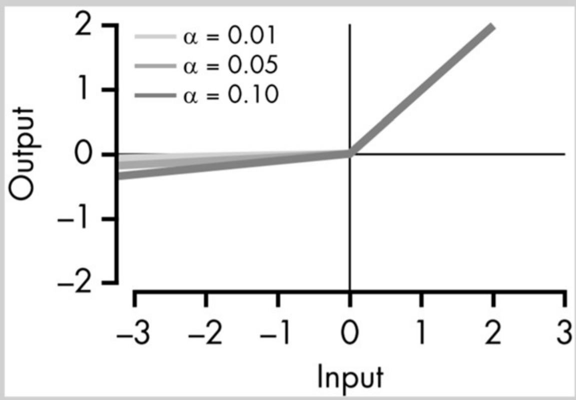
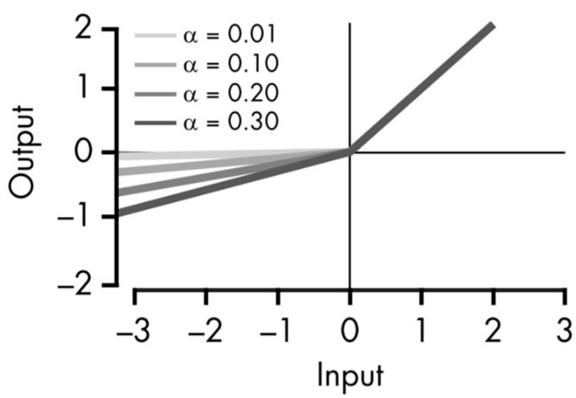
- Common activation functions

<table>
<tr><th>Name</th><th>Plot</th><th>Equation</th><th>Description</th></tr>
<tr><td>Identity</td><td>
</td><td>$f(x) = x$</td><td>Basically: no activation function!</td></tr>
<tr><td>ReLU</td><td>
</td><td>$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$</td><td>Just max(0, x).

ReLUs enable fast learning and are more resilient to the vanishing gradient problem (explained later in this chapter) compared to other functions, like the sigmoid.</td></tr>
</table>

how neural networks work

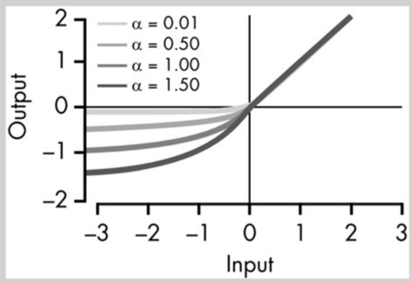
- Common activation functions

Name	Plot	Equation	Description
Leaky ReLU		$f(x) = \begin{cases} \alpha x \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	Like normal ReLU, but instead of 0, a small constant fraction of *x* is returned. Generally you choose α to be very small, like 0.01. Also, α stays fixed during training.
PReLU		$f(x) = \begin{cases} \alpha x \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	This is just like leaky ReLU, but in PReLU, α is a parameter whose value is optimized during the training process, along with the standard weight and bias parameters.

how neural networks work

- Common activation functions

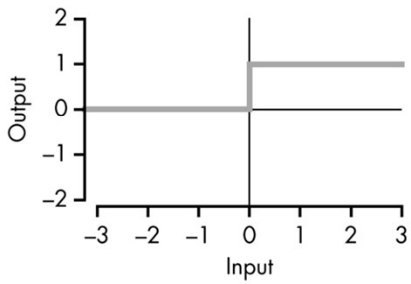
ELU



$$f(x) = \begin{cases} \alpha\left(e^{x} - 1\right) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Like PReLU in that α is a parameter, but instead of going down infinitely with a slope of α when $x < 0$, the curve is bounded by α, because e^{x} will always be between 0 and 1 when $x < 0$.

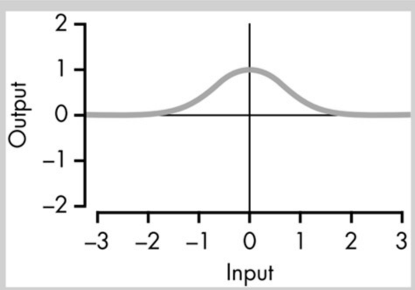
Step



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Just a step function: the function returns 0 unless $x \geq 0$, in which case the function returns 1.

Gaussian

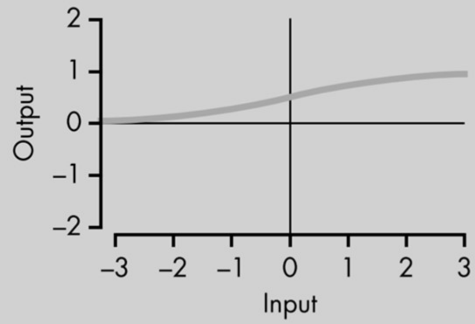


$$f(x) = e^{-x^{2}}$$

A bell-shaped curve whose maximum value tops out at 1 when $x = 0$.

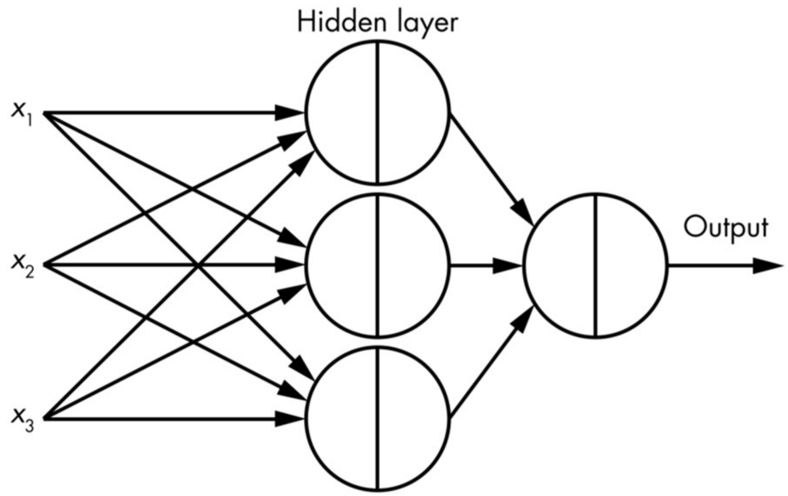
how neural networks work

- Common activation functions

Name	Plot	Equation	Description
Sigmoid		$f(x) = \frac{e^x}{e^x + 1}$	Because of the vanishing gradient problem (explained later in this chapter), sigmoid activation functions are often only used in the final layer of a neural network. Because the output is continuous and bounded between 0 and 1, sigmoid neurons are a good proxy for output probabilities.
Softmax	(multi-output)	$f(x) = \frac{e^{x_j}}{\sum_{k=1}^{k=K} e^{x_k}}$ for $j = 1,\ 2,\ \ldots, K$	Outputs multiple values that sum to 1. Softmax activation functions are often used in the final layer of a network to represent classification probabilities, because Softmax forces all outputs from a neuron to sum to 1.

how neural networks work

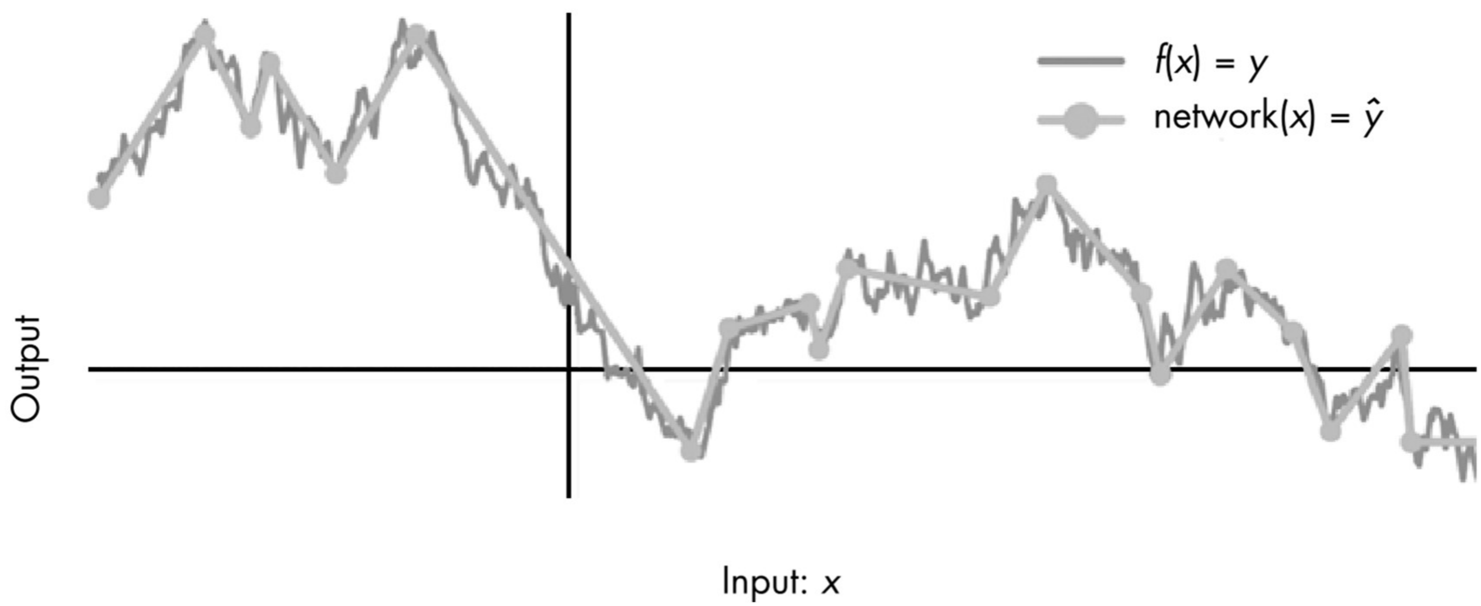
- A Network of Neurons



  - the total number of optimizable parameters is
    - number of edges connecting an input to a neuron, plus the number of neurons.

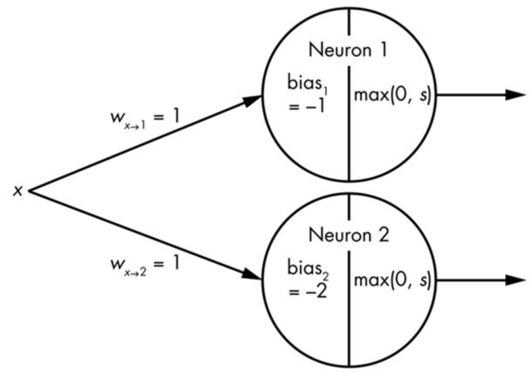
how neural networks work

- Universal Approximation Theorem
  - a feed-forward network with a single hidden layer of neurons with nonlinear activation functions can approximate (with an arbitrarily small error) any continuous function on a compact subset of R^n

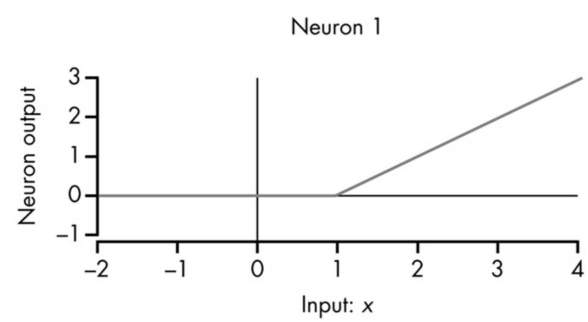


Building Your Own Neural Network

- Starting with two ReLU neurons

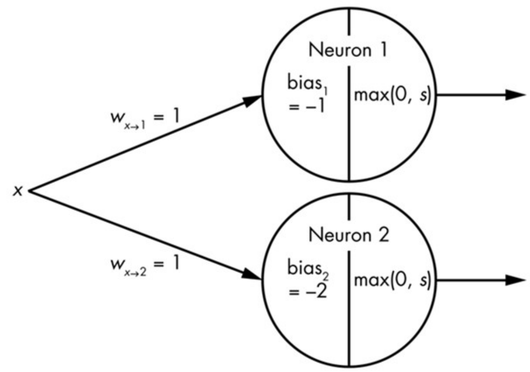


Input	Weighted sum	Weighted sum + bias	Output
x	$x * w_{x\to 1}$	$x * w_{x\to 1}$ + bias$_1$	max(0, $x * w_{x\to 1}$ + bias$_1$)
0	0 * 1 = 0	0 + −1 = −1	max(0, −1) = 0
1	1 * 1 = 1	1 + −1 = 0	max(0, 0) = 0
2	2 * 1 = 2	2 + −1 = 1	max(0, 1) = 1
3	3 * 1 = 3	3 + −1 = 2	max(0, 2) = 2
4	4 * 1 = 4	4 + −1 = 3	max(0, 3) = 3
5	5 * 1 = 5	5 + −1 = 4	max(0, 4) = 4

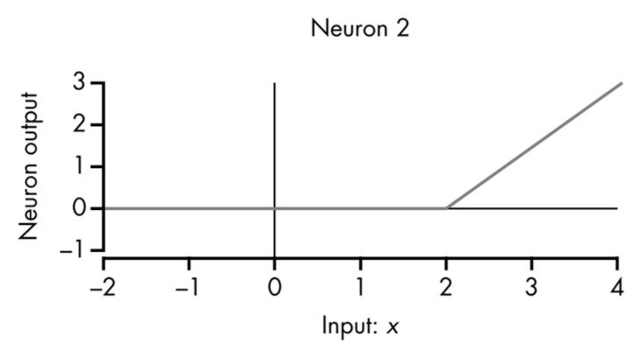


Building Your Own Neural Network

- Starting with two ReLU neurons

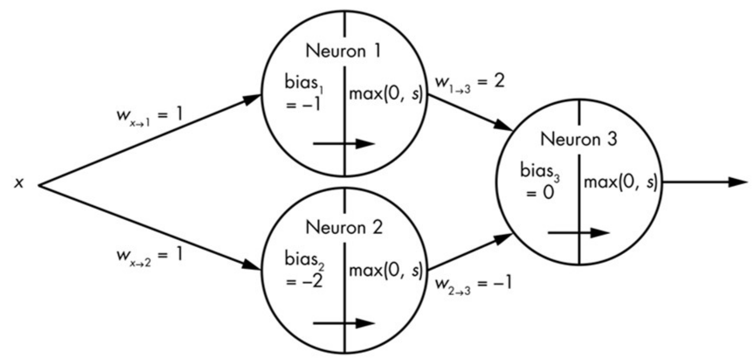


Input	Weighted sum	Weighted sum + bias	Output
x	$x * w_{x \to 2}$	$(x * w_{x \to 2}) + \text{bias}_2$	$\max(0, (x * w_{x \to 2}) + \text{bias}_2)$
0	0 * 1 = 0	0 + −2 = −2	max(0, −2) = 0
1	1 * 1 = 1	1 + −2 = −1	max(0, −1) = 0
2	2 * 1 = 2	2 + −2 = 0	max(0, 0) = 0
3	3 * 1 = 3	3 + −2 = 1	max(0, 1) = 1
4	4 * 1 = 4	4 + −2 = 2	max(0, 2) = 2
5	5 * 1 = 5	5 + −2 = 3	max(0, 3) = 3

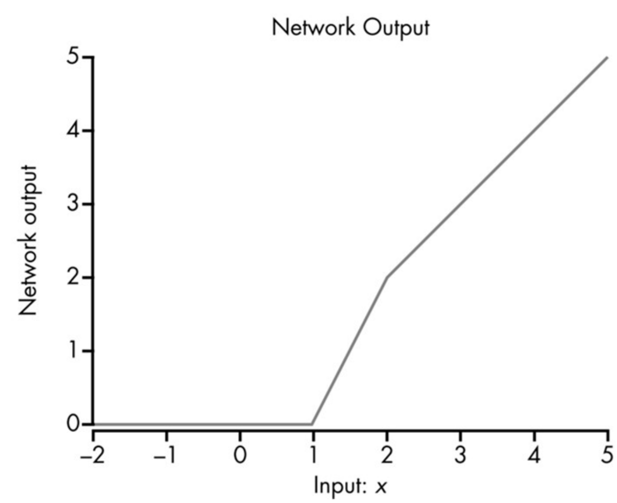


Building Your Own Neural Network

- Adding another ReLU neurons

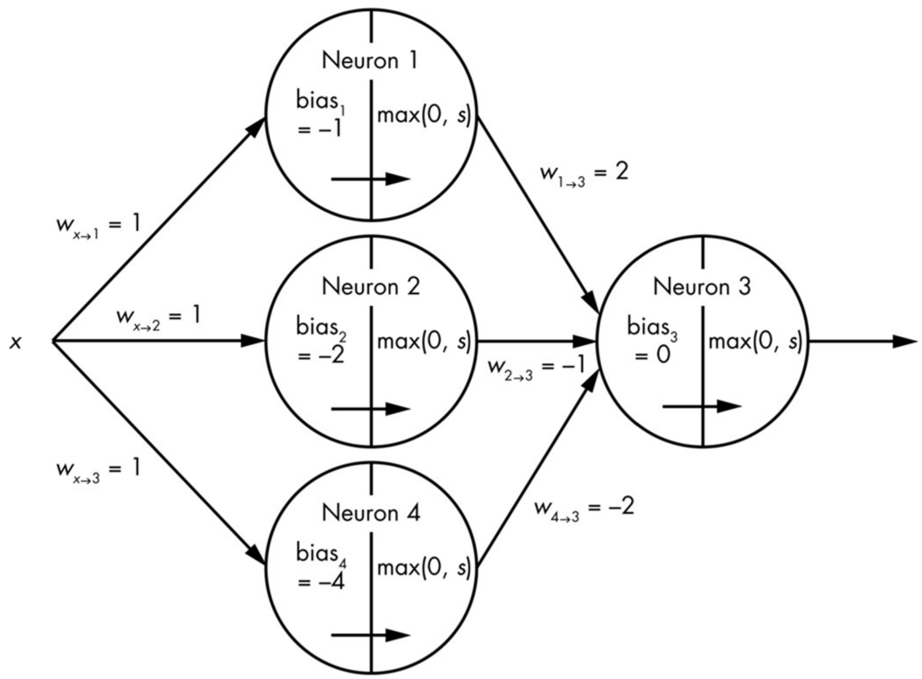


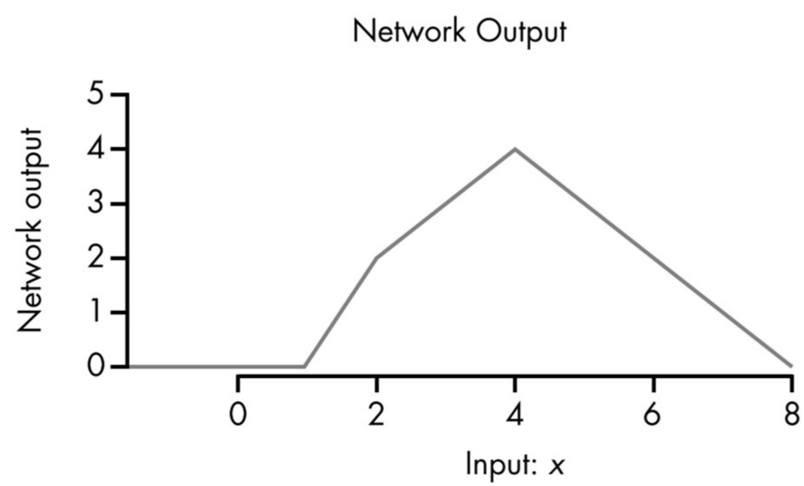
Original network input	Inputs to neuron$_3$		Weighted sum	Weighted sum + bias	Final network output
x	neuron$_1$	neuron$_2$	(neuron$_1$ * $w_{1\to3}$) + (neuron$_2$ * $w_{2\to3}$)	(neuron$_1$ * $w_{1\to3}$) + (neuron$_2$ * $w_{2\to3}$) + bias$_3$	max(0, (neuron$_1$ * $w_{1\to3}$) + (neuron$_2$ * $w_{2\to3}$) + bias$_3$)
0	0	0	(0 * 2) + (0 * −1) = 0	0 + 0 + 0 = 0	max(0, 0) = 0
1	0	0	(0 * 2) + (0 * −1) = 0	0 + 0 + 0 = 0	max(0, 0) = 0
2	1	0	(1 * 2) + (0 * −1) = 2	2 + 0 + 0 = 2	max(0, 2) = 2
3	2	1	(2 * 2) + (1 * −1) = 3	4 + −1 + 0 = 3	max(0, 3) = 3
4	3	2	(3 * 2) + (2 * −1) = 4	6 + −2 + 0 = 4	max(0, 4) = 4
5	4	3	(4 * 2) + (3 * −1) = 5	8 + −3 + 0 = 5	max(0, 5) = 5



Building Your Own Neural Network

- Adding Another Neuron to the Network





Automatic Feature Generation

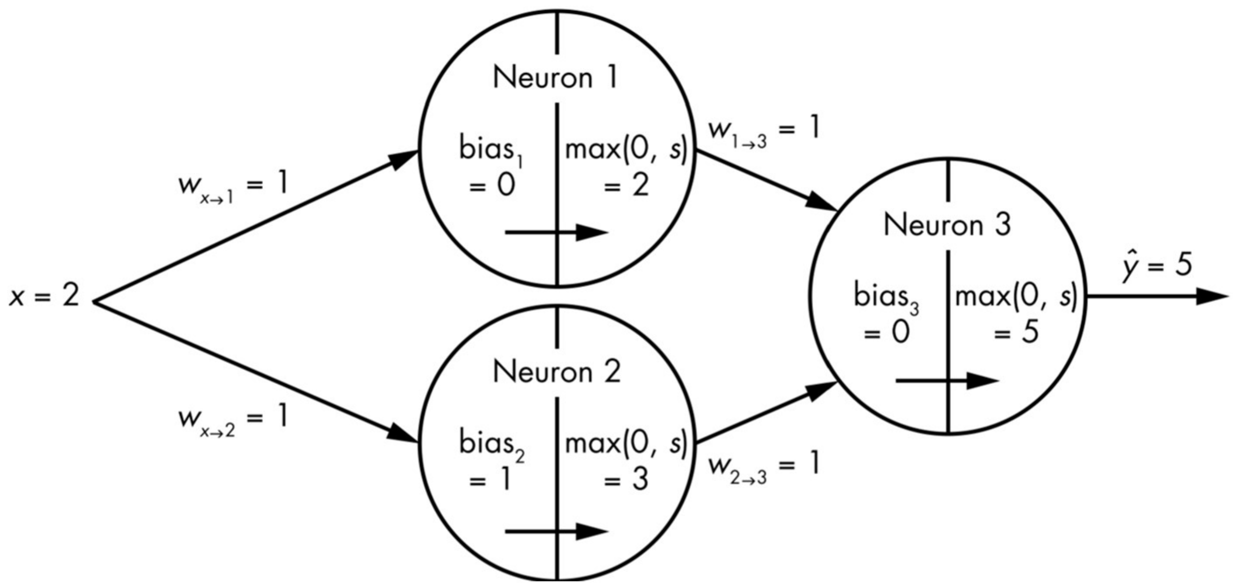
- what happens when we have multiple hidden layers of neurons?
  - you give raw features to neural network
  - each layer represent those raw features in ways that work well as inputs to later layers.

training neural networks

- we start with a training dataset and a network with randomly initialized parameters.
  - feed a network an observation, x, from your training dataset
  - receive some output, ˆy (forward propagation)
  - figure out how changing your parameters will shift ˆy closer to your goal, y.
  - Parameters all throughout the network are then nudged just a tiny bit in a direction that causes ˆy to shift a little closer to y
    - If $\partial\hat{y}/\partial w$ is positive
      - you should increase w by a small amount
- The process of iteratively calculating partial derivatives, updating parameters, and then repeating is called *gradient descent*

training neural networks

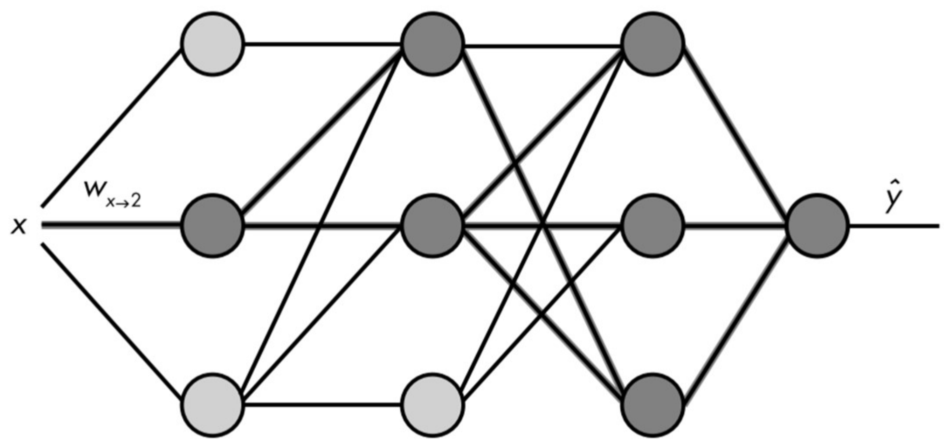
- Using Backpropagation to Optimize a Neural Network



$$\frac{\partial \hat{y}}{\partial w_{x\to1}} = \frac{\partial \hat{y}}{\partial \text{neuron}_1} * \frac{\partial \text{neuron}_1}{\partial w_{x\to1}} \qquad \frac{\partial \hat{y}}{\partial w_{x\to1}} = 1 * 2 = 2$$

training neural networks

- Using Backpropagation to Optimize a Neural Network
  - Path explosion



$$\frac{\partial \hat{y}}{\partial \mathrm{neuron}_{i+1}} * \frac{\partial \mathrm{neuron}_{i+1}}{\partial \mathrm{neuron}_{i}} * \frac{\partial \mathrm{neuron}_{i}}{\partial w}$$

training neural networks

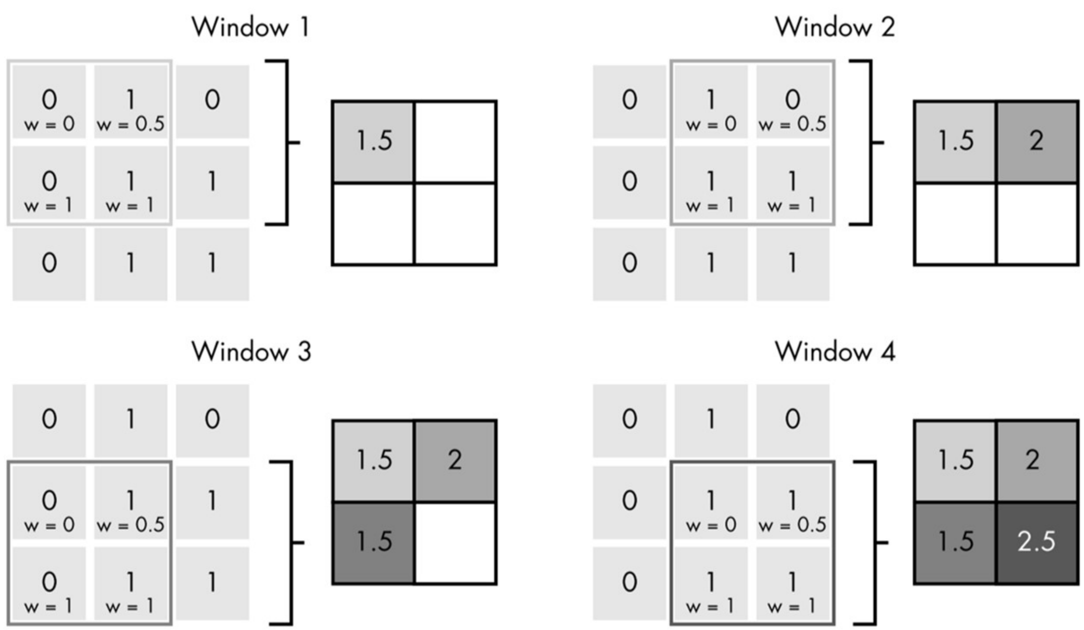
- Using Backpropagation to Optimize a Neural Network
  - Vanishing Gradient
    - *Consider a weight parameter in the first layer of a neural network that has ten layers*
      - *Its parameters are updated based on*
        - *the summation of a massive very tiny numbers*
        - *many of which end up canceling one another out*
    - *it can be difficult for a network to coordinate sending a strong signal down to parameters in lower layers*
    - *certain network designs try to get around this problem*

types of neural networks

- Feed-Forward Neural Network
  - simplest kind of neural network
  - consists of stacks of layers of neurons
  - Each layer of neurons is connected to some or all neurons in the next layer
    - Each neuron doesn’t necessarily have to connect to every neuron in the next layer
  - connections never go backward or form cycles

types of neural networks

- Convolutional Neural Network
  - contains convolutional layers where
    - the input that feeds into each neuron is defined by a window that slides over the input space

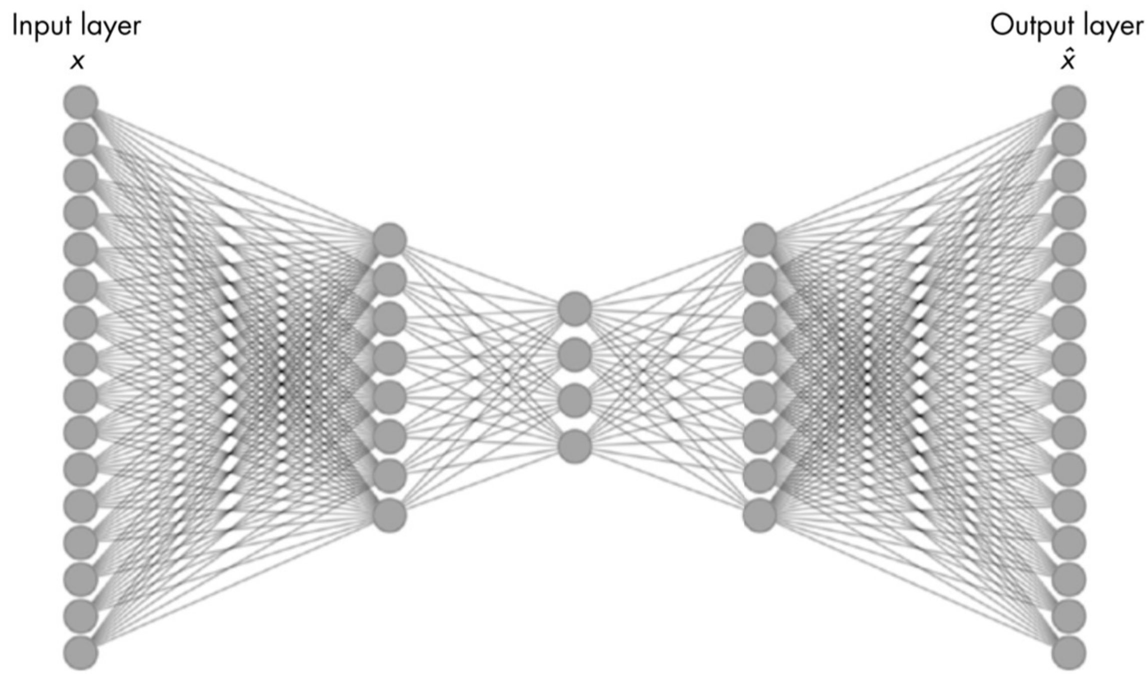


types of neural networks

- Convolutional Neural Network
  - Also contain pooling layer
    - “zoom out” on the data
      - reducing the size of the features for faster computation
      - while retaining the most important information
  - The structure of these networks encourages localized feature learning
    - extremely effective at image recognition and classification

types of neural networks

- Autoencoder Neural Network



types of neural networks

- Generative Adversarial Network
  - a system of two neural networks
    - competing with each other to improve themselves at their respective tasks.
      - the generative network tries to create fake samples from random noise
      - the discriminator network attempts differentiate between real and fake samples

types of neural networks

- Generative Adversarial Network
  - Both neural networks in a GAN are optimized with backpropagation
    - their loss functions are direct opposites of one another
      - generator network optimizes its parameters based on how well it fooled the discriminator network in a given round
      - discriminator network optimizes its parameters based on how accurately it could discriminate between generated and real samples.
  - GANs can be been used to generate real-looking data or enhance low-quality or corrupted data

types of neural networks

- Recurrent Neural Network
  - connections between neurons form directed cycles
  - activation functions are dependent on time-steps
    - allows the network to develop a memory
    - helps it learn patterns in sequences of data
  - the inputs, the outputs, or both are some sort of time series

types of neural networks

- Recurrent Neural Network
  - are great for tasks where data order matters
    - connected handwriting recognition
    - speech recognition
    - language translation
    - and time series analysis
  - In the context of cybersecurity
    - network traffic analysis
    - behavioral detection
    - static file analysis
      - Because program code is similar to natural language
        - order matters
        - it can be treated as a time series

types of neural networks

- Recurrent Neural Network
  - vanishing gradient is an issue
    - each time-step in an RNN is similar to an entire extra layer
    - Backpropagation causes signals in earlier time-steps to become incredibly faint
  - long short-term memory (LSTM) network
    - a special type of RNN
    - designed to address vanishing gradient problem.
      - contain memory cells and special neurons that try to decide
        - what information to remember
        - and what information to forget.

types of neural networks

- ResNet (residual network)
  - creates skip connections between neurons in early layers of the network to deeper layers
  - pass numerical information directly between layers
    - without to pass through the kinds of activation functions
    - helps greatly reduce the vanishing gradient
    - enables ResNets to be incredibly deep