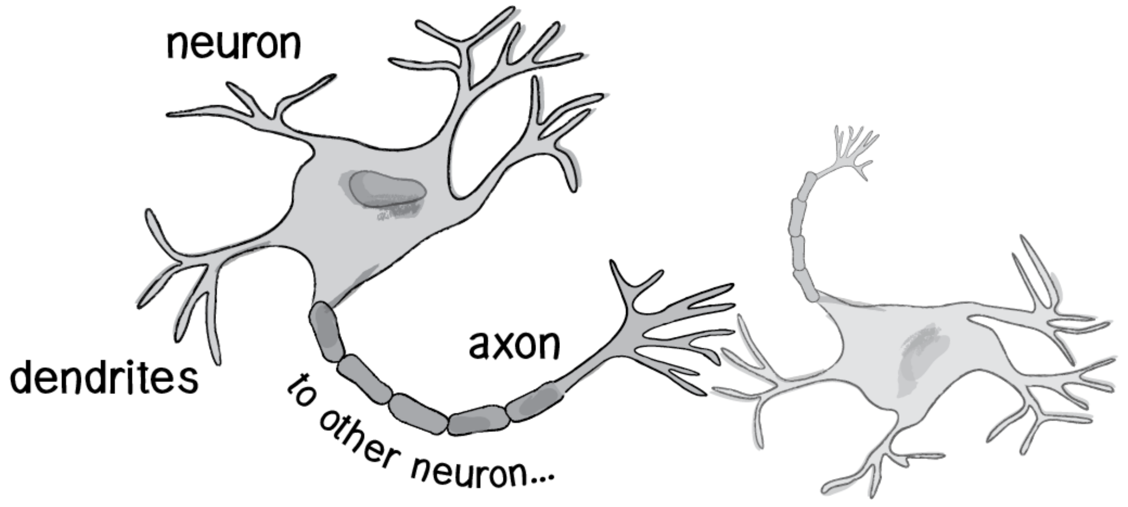
Neural Network

Outline

- What is Neural Network?
- How does it work?
- Pros and Cons associated with neural network

Neural Network

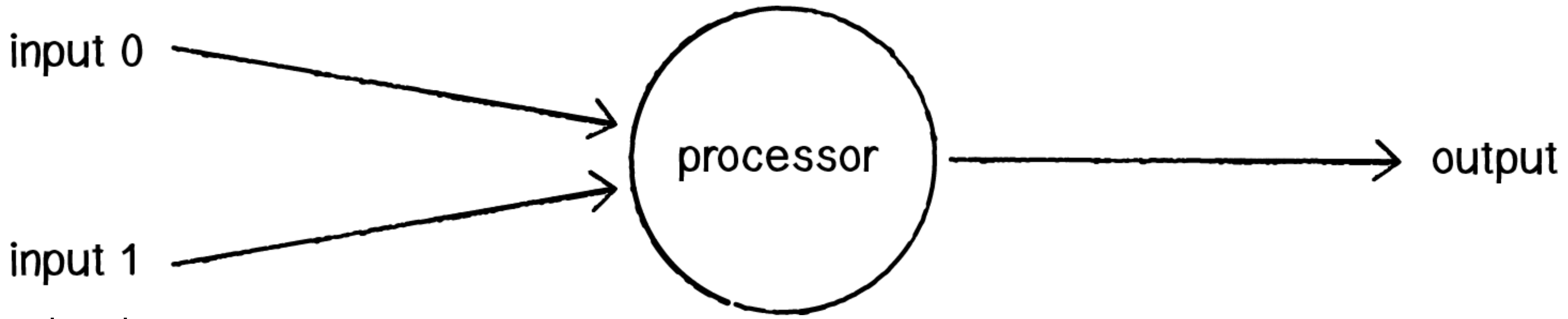


The human brain can be described as a biological neural network—an interconnected web of neurons transmitting elaborate patterns of electrical signals. Dendrites receive input signals and, based on those inputs, fire an output signal via an axon.

- Pattern Recognition —facial recognition, optical character recognition, etc.
- Time Series Prediction — Will the stock rise or fall tomorrow? Will it rain or be sunny?
- Signal Processing —Cochlear implants and hearing aids need to filter out unnecessary noise and amplify the important sounds. Neural networks can be trained to process an audio signal and filter it appropriately.
- Control —You may have read about recent research advances in self-driving cars. Neural networks are often used to manage steering decisions of physical vehicles (or simulated ones).
- Soft Sensors —A soft sensor refers to the process of analyzing a collection of many measurements. A thermometer can tell you the temperature of the air, but what if you also knew the humidity, barometric pressure, dewpoint, air quality, air density, etc.? Neural networks can be employed to process the input data from many individual sensors and evaluate them as a whole.
- Anomaly Detection —Because neural networks are so good at recognizing patterns, they can also be trained to generate an output when something occurs that doesn't fit the pattern. Think of a neural network monitoring your daily routine over a long period of time. After learning the patterns of your behavior, it could alert you when something is amiss.

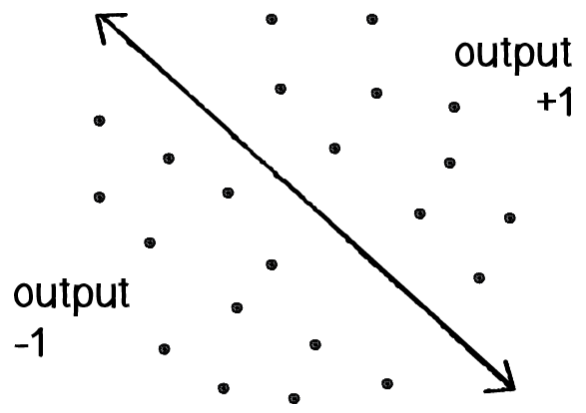
The Perceptron

a perceptron is the simplest neural network possible: a computational model of a single neuron

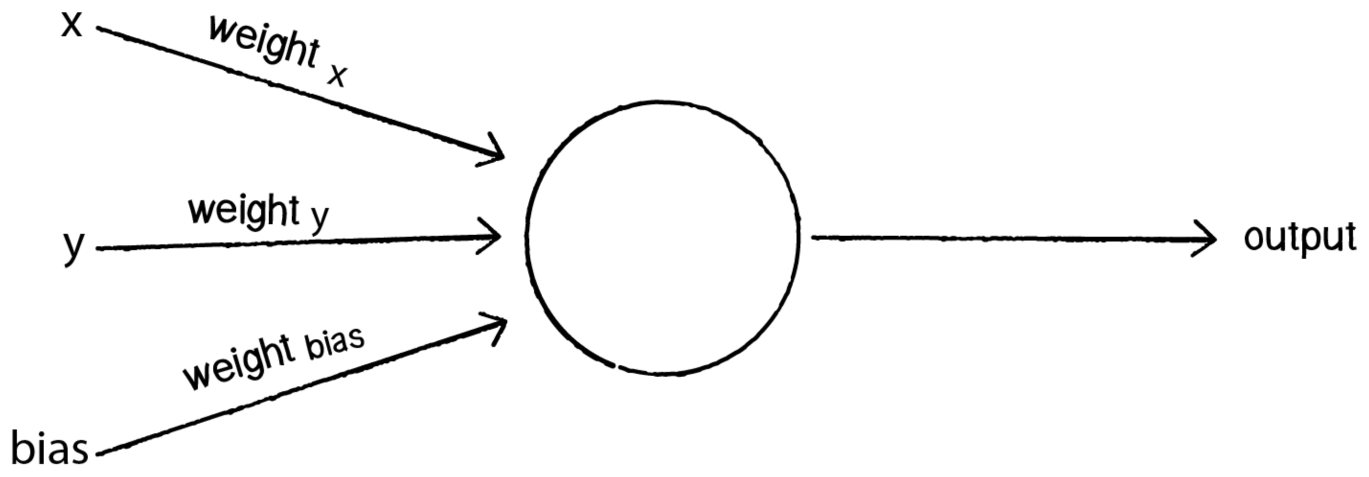


- Receive inputs
  - Input 0: x1 = 12
  - Input 1: x2 = 4
- Weight inputs.
  - Weight 0: 0.5
  - Weight 1: -1
- Sum inputs.
  - Sum = 6 + -4 = 2
- Generate output.
  - if the sum is a positive number, the output is 1; if it is negative, the output is -1.
  - Output = sign(sum) ⇒ sign(2) ⇒ +1

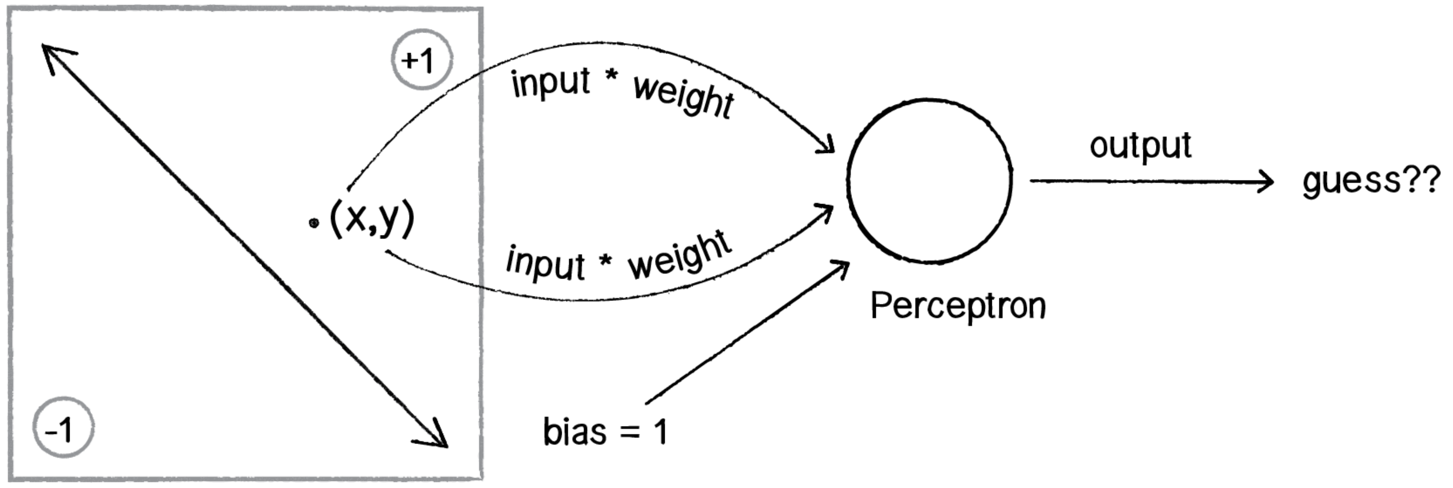
Simple Pattern Recognition Using a Perceptron



Let's consider the point (0,0). What if we send this point into the perceptron as its input: x = 0 and y = 0? No matter what the weights are, the sum will always be 0! But this can't be right—after all, the point (0,0) could certainly be above or below various lines in our two-dimensional world.
To avoid this dilemma, our perceptron will require a third input, typically referred to as a bias input. A bias input always has the value of 1 and is also weighted.

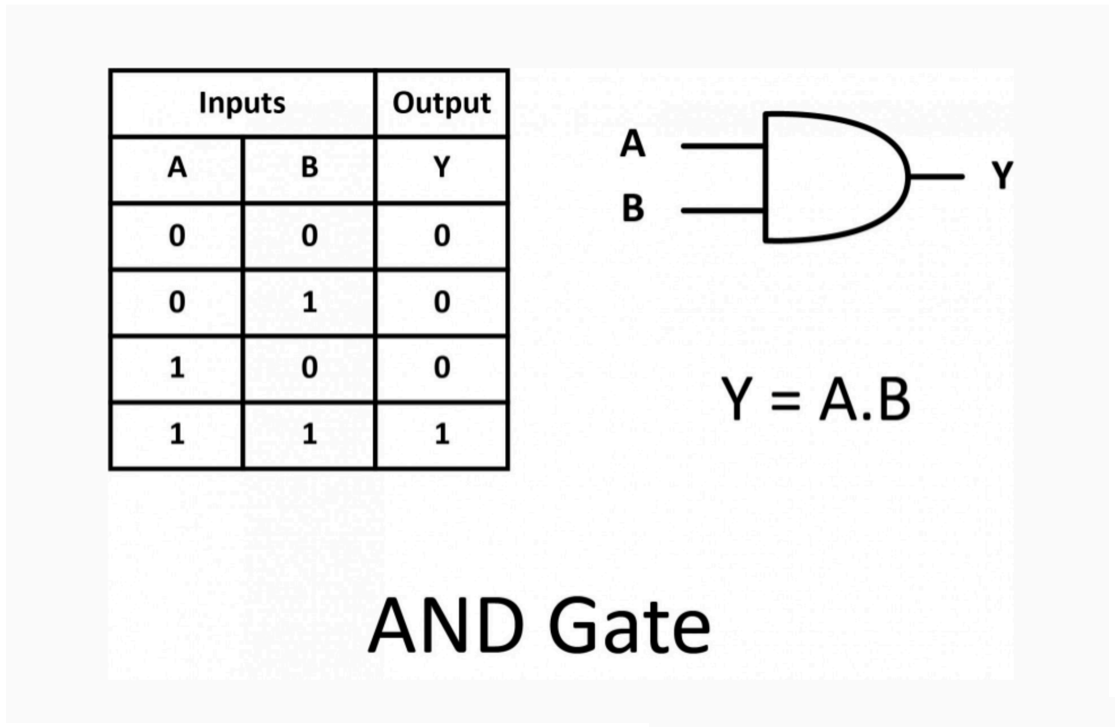


- Provide the perceptron with inputs for which there is a known answer.
- Ask the perceptron to guess an answer.
- Compute the error. (Did it get the answer right or wrong?)
- Adjust all the weights according to the error.
- Return to Step 1 and repeat!



Logical Gates

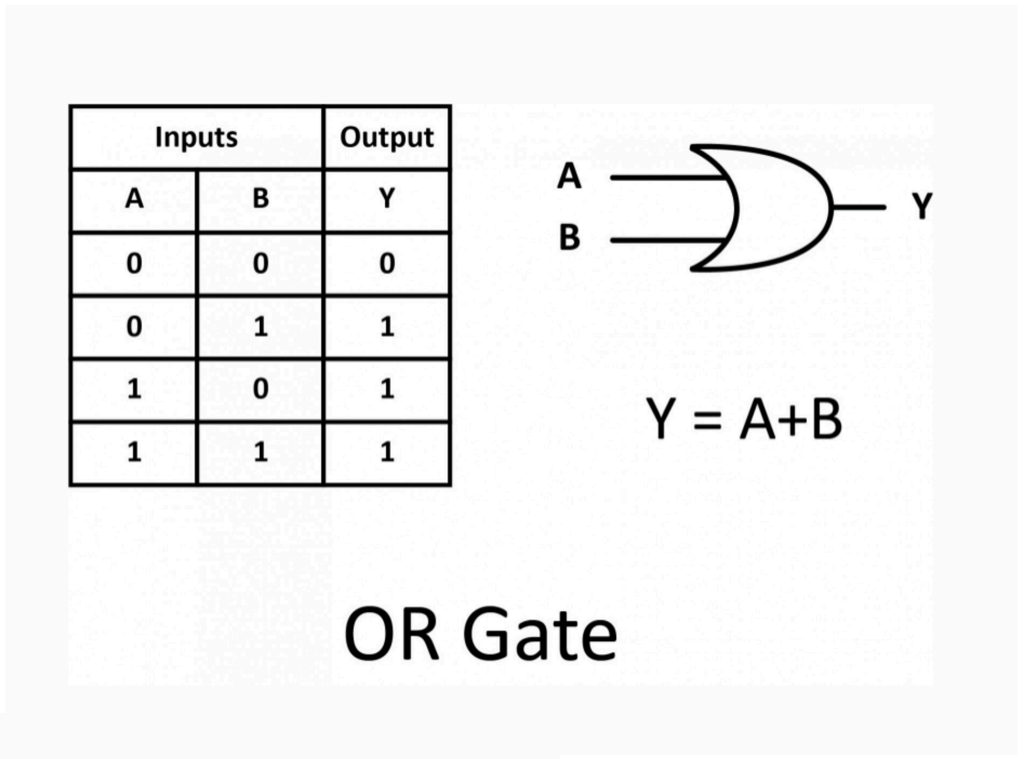
A single perceptron can model the logical AND/OR/NOT gate



Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = A.B$

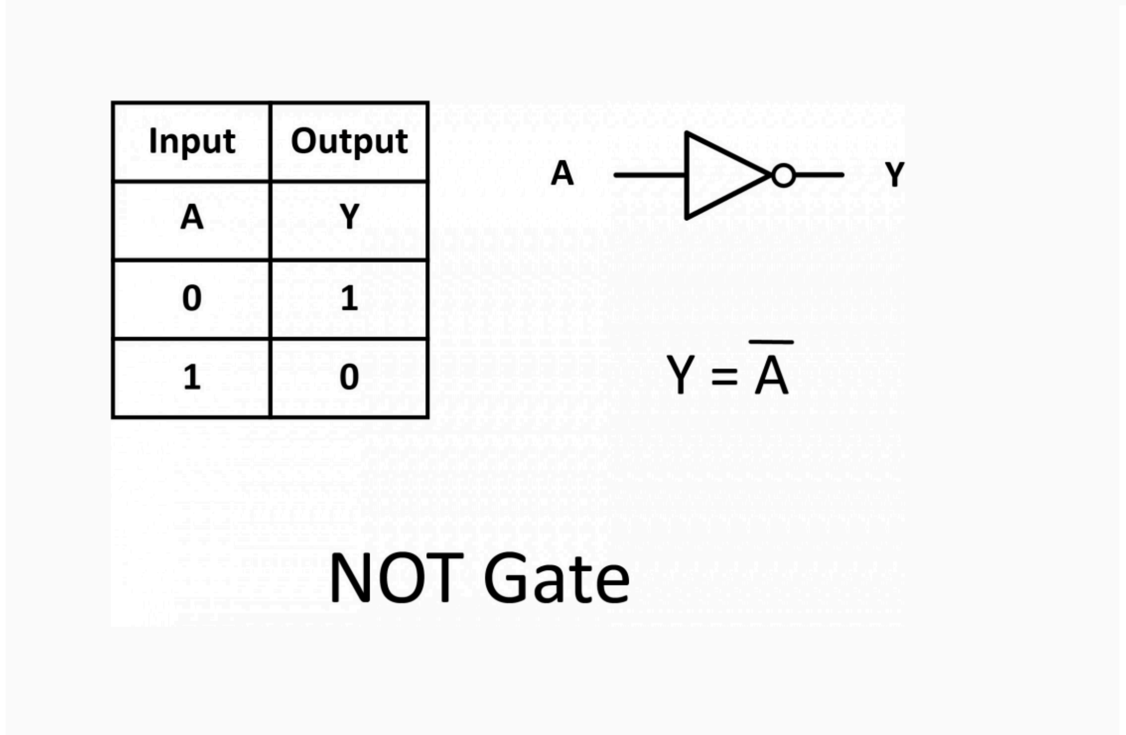
AND Gate



Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$Y = A+B$

OR Gate



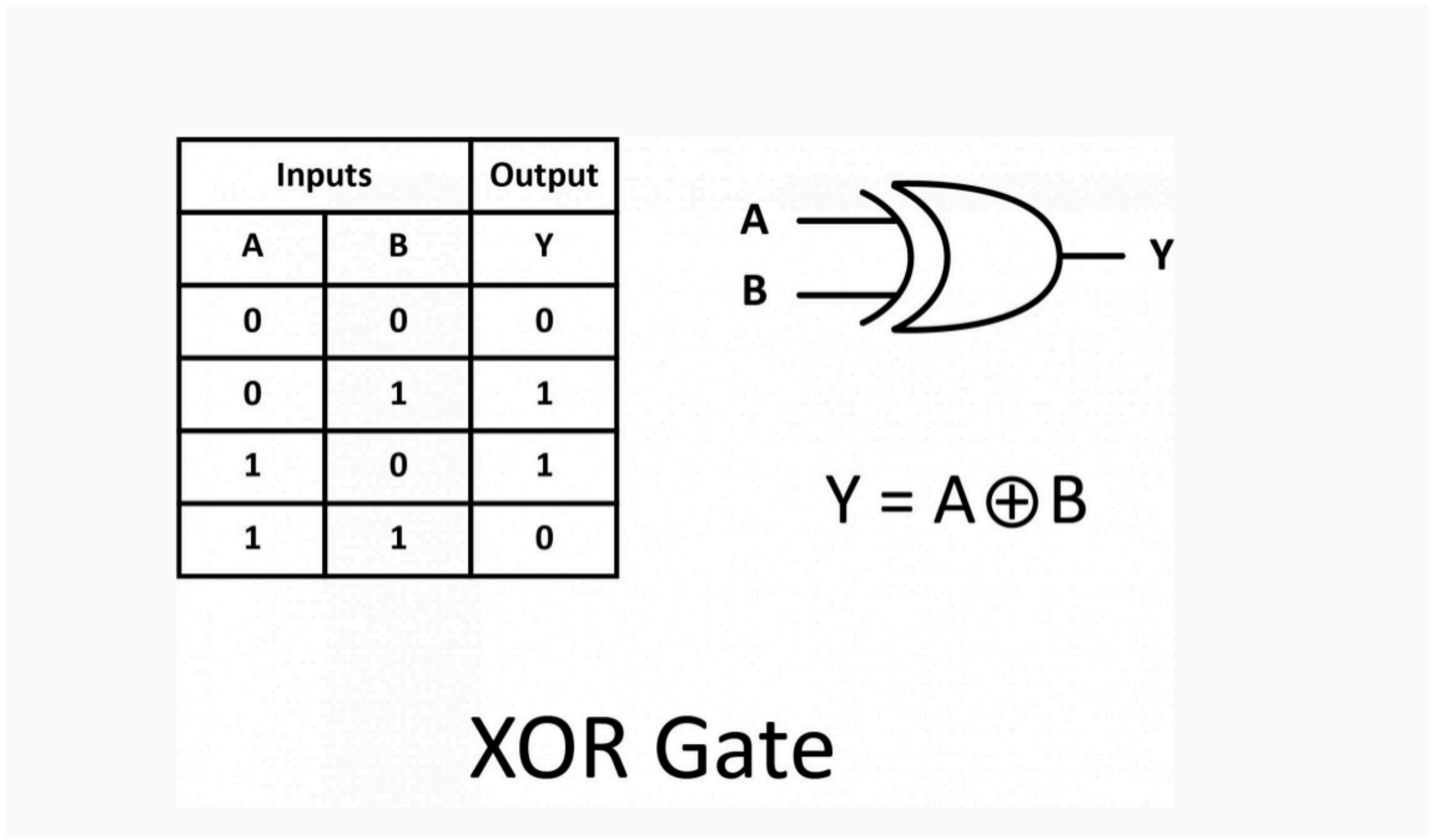
Input	Output
A	Y
0	1
1	0

$Y = \overline{A}$

NOT Gate

Logical Gates

One perceptron can't solve XOR. But what if we made a network out of two perceptrons? If one perceptron can solve OR and one perceptron can solve NOT AND, then two perceptrons combined can solve XOR.

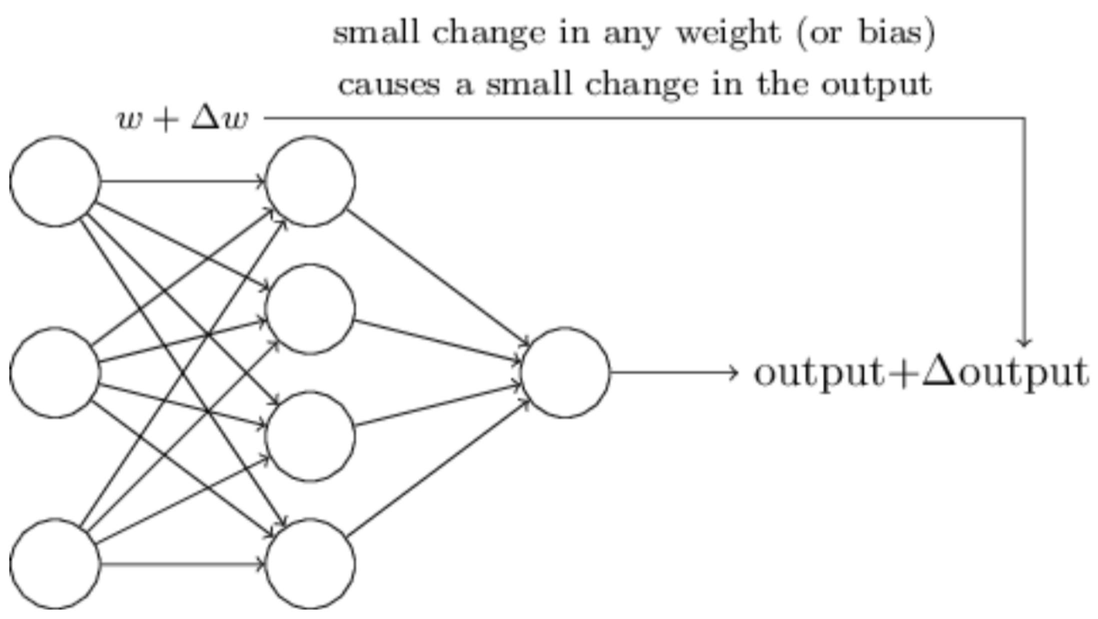


Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = A \oplus B$

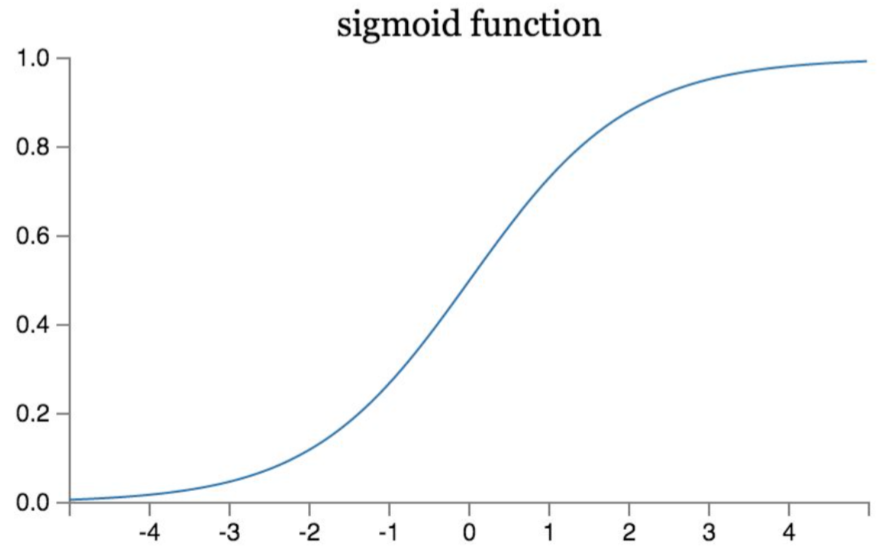
XOR Gate

Sigmoid Neutrons for Classification


small change in any weight (or bias)
causes a small change in the output
$w + \Delta w$
output+Δoutput

Do not know the weights and biases
To have neural network learn,

- Change a weight or bias
- Observe the change in the output
  - If the change makes prediction better, keep it and now manipulate a different weight
  - If worse, forget it and try again
- Repeat steps
  - An accumulation of small changes lead to big change in output and greater accuracy



Output:

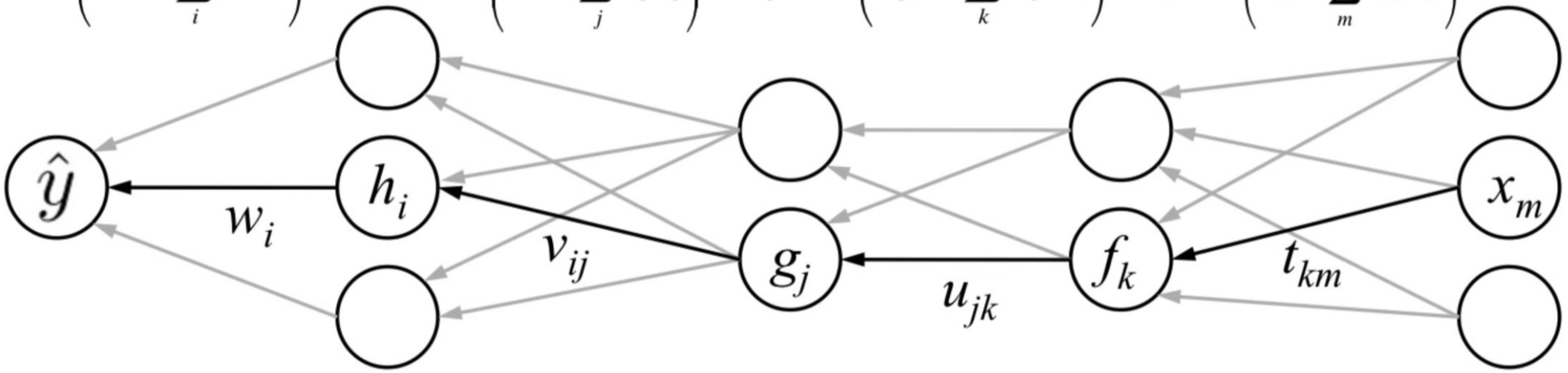
$$\sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

z=wx+b
if z is largely positive, output 1
if z is largely negative, output 0

Backpropagation Learning

The solution to optimizing weights of a multi-layered network is known as backpropagation. The output of the network is generated in the same manner as a perceptron. The inputs multiplied by the weights are summed and fed forward through the network. Training the network (i.e. adjusting the weights) involves taking the error (desired result - guess). The error, however, must be fed backwards through the network. The final error ultimately adjusts the weights of all the connections.

$$\hat{y} = \sigma\left(w_0 + \sum_i w_i h_i\right) \qquad h_i = \sigma\left(v_{i0} + \sum_j v_{ij} g_j\right) \qquad g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right) \qquad f_k = \sigma\left(t_{k0} + \sum_m t_{km} x_m\right)$$



Backpropagation Learning with Gradient Decent

Error Term:

$$E = \frac{1}{2}(\hat{y} - y)^2$$

Last hidden layer:

$$\begin{aligned}\frac{\partial E}{\partial h_i} &= (\hat{y} - y)\frac{\partial \hat{y}}{\partial h_i} \\ &= (\hat{y} - y)\hat{y}(1 - \hat{y})w_i\end{aligned}$$

Last second hidden layer:

$$\begin{aligned}\frac{\partial E}{\partial g_j} &= (\hat{y} - y)\frac{\partial \hat{y}}{\partial g_j} \\ &= (\hat{y} - y)\hat{y}(1 - \hat{y})\sum_i w_i \frac{\partial h_i}{\partial g_j} \\ &= (\hat{y} - y)\hat{y}(1 - \hat{y})\sum_i w_i h_i(1 - h_i)v_{ij}\end{aligned}$$

$$\frac{\partial E}{\partial g_j} = \sum_i h_i(1 - h_i)v_{ij}\frac{\partial E}{\partial h_i}$$

Backpropagation Learning with Gradient Decent

Weights update:

$$w_i' = w_i - \eta \frac{\partial E}{\partial w_i}$$

If the gradient positive, shift wi from the increasing tendency
If the gradient negative, shift wi towards the decreasing tendency

Pros of NN

Advantages of Neural Networks:

Neural Networks have the ability to learn by themselves and produce the output that is not limited to the input provided to them.

- The input is stored in its own networks instead of a database, hence the loss of data does not affect its working.
- These networks can learn from examples and apply them when a similar event arises, making them able to work through real-time events.
- Even if a neuron is not responding or a piece of information is missing, the network can detect the fault and still produce the output.
- They can perform multiple tasks in parallel without affecting the system performance.

Cons of NN

Neural networks almost always underperform approaches that have stronger statistical foundations, and tend to be more difficult to work with, for the following reasons.

- Neural networks are not magic hammers. Neural networks are still viewed by many as being magic hammers which can solve any machine learning problem, and as a result, people tend to apply them indiscriminately to problems for which they are not well suited. Although neural networks do have a proven track record of success for certain specific problem domains, as a consumer of machine learning technology, you're almost always better off using approaches that have stronger theoretical underpinnings, rather than just throwing a general-purpose neural network at your problem and hoping for the best. Try to understand and simplify your problem as well as you possibly can first, and then look for techniques that have a structure which is a good match for your problem.

- Neural networks are too much of a black box. This has several consequences. It makes them difficult to train: the training outcome can be nondeterministic and depend crucially on the choice of initial parameters, e.g. the starting point for gradient descent when training back propagation networks. It makes them hard to determine how they are solving a problem, because they are opaque. It makes them difficult to troubleshoot when they don't work as you expect, and when they do work, you will never really feel confident that they will generalize well to data not included in your training set because, fundamentally, you don't understand how your network is solving the problem, or what idiosyncrasies of the training data the network has overfit to.

Cons of NN

- Neural networks are not a substitute for understanding the problem deeply. Instead of investing your time throwing a neural net at the problem, you're almost always better off investing a little extra time studying, analyzing and dissecting your data first, then using your better understanding to choose a technique with stronger theoretical foundations, a technique which you will now have more confidence will work well for your problem. For example, if you are building a classifier, rather than just throwing a back-prop neural network at your data and hoping for the best, spend time visualizing the data and selecting or creating the best input features using whatever domain-specific knowledge and expertise you have available to you. You might discover that you can clearly differentiate between your training classes using just three out of your original ten features for example, or that you need to develop a more sophisticated nonlinear derived feature to better separate your classes, and those discoveries will often be guided by your knowledge of the natural structure in the data.