Introduction to Advanced Machine Learning

Data Boot Camp

Lesson 21.1

© 2022 edX Boot Camps LLC. Confidential and Proprietary. All Rights Reserved.

Class Objectives

By the end of this lesson, you will be able to:

Describe the differences between traditional machine learning classification and regression models and neural network models.

Describe the perceptron model and its components.

Implement neural network models using TensorFlow.

Surfing the Neural Net

What is a Neural Network?

What Is a Neural Network?

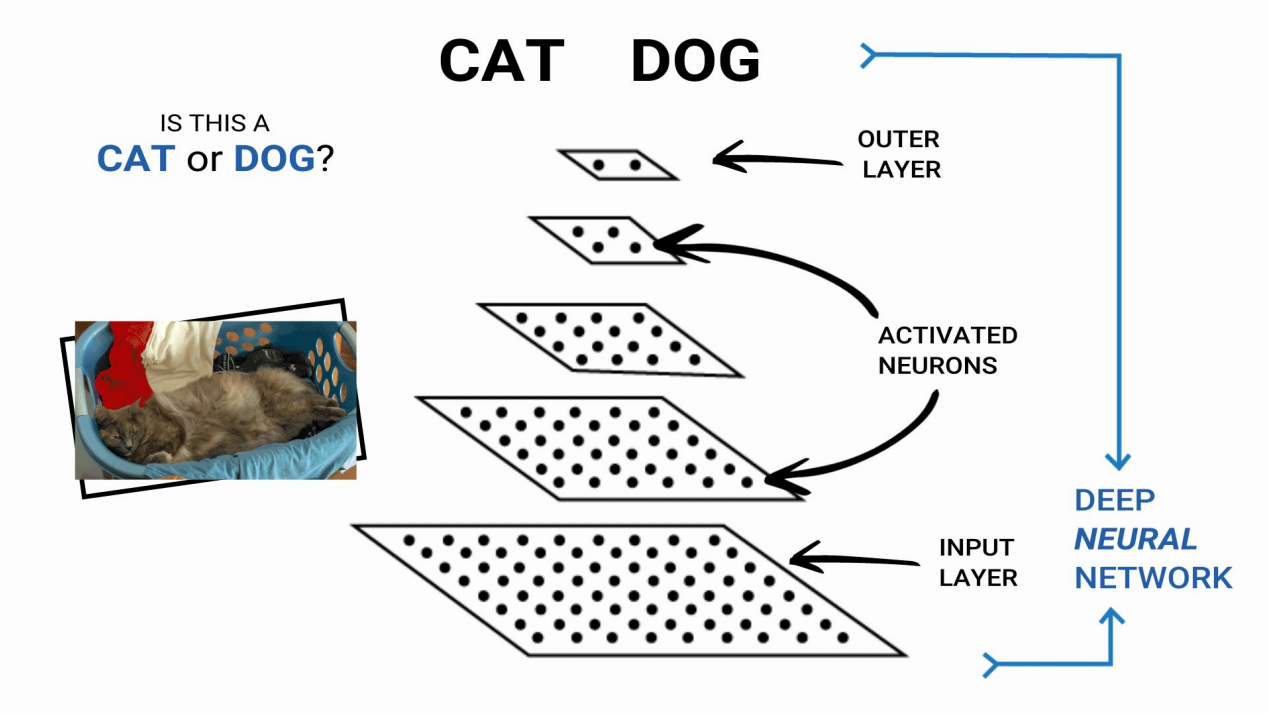
Neural networks are a powerful machine learning technique modeled after neurons in the brain.

Industry leaders such as Google, Facebook, Twitter, and Amazon use neural networks for analyzing complex datasets.

Neural Network is an advanced form of machine learning that contains multiple layers of nodes, which perform individual computations.

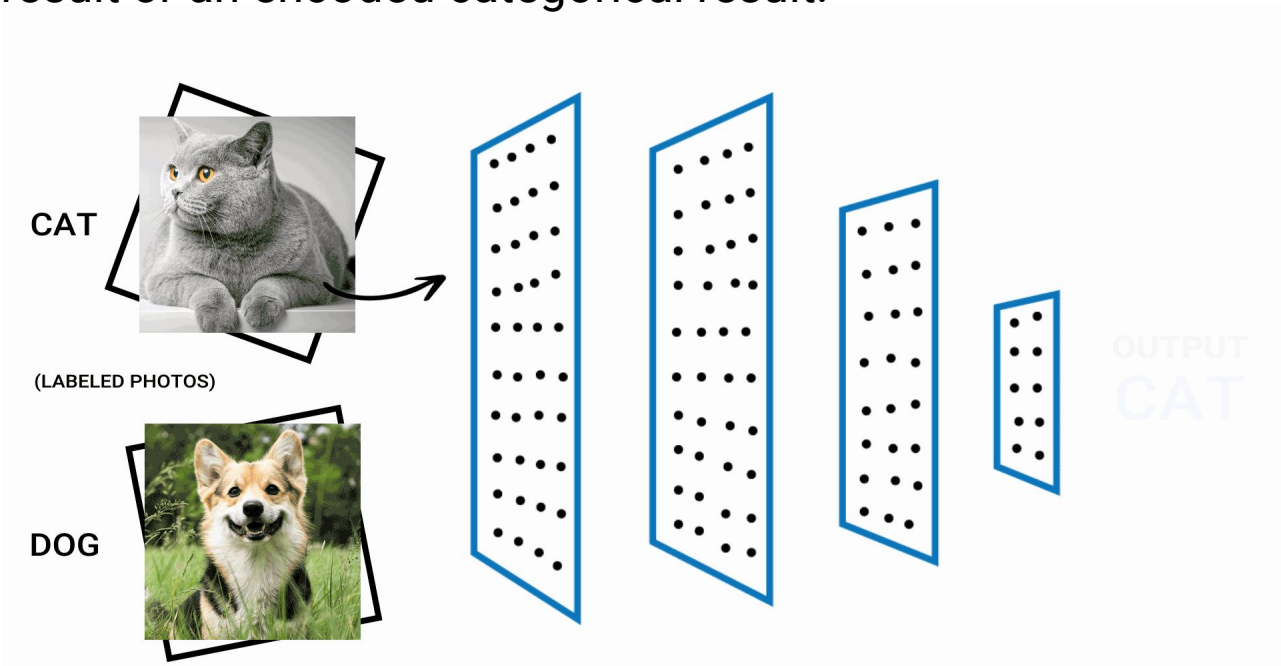
What Is a Neural Network?

In its simplest form, a neural network contains layers of neurons that perform individual computations.



What Is a Neural Network?

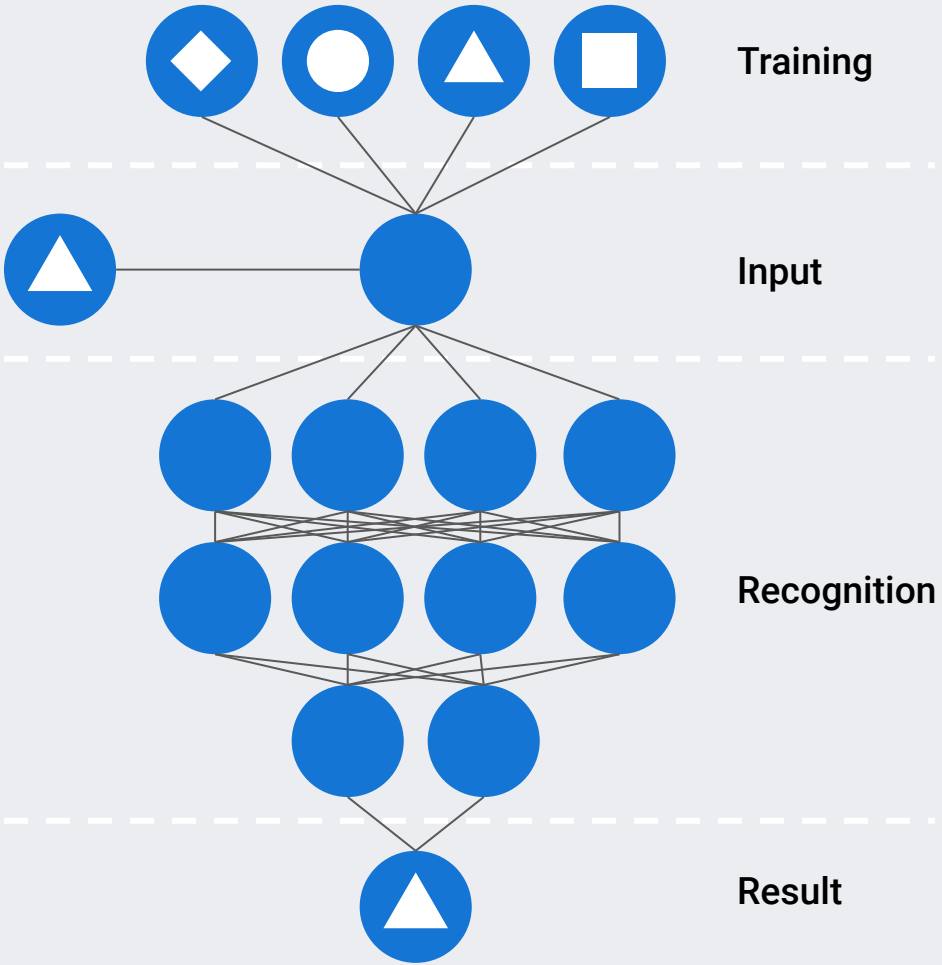
These computations are connected and weighed against one another until the neurons reach the final layer. In the final layer, the neurons return either a numerical result or an encoded categorical result.



What Is a Neural Network?

Neural networks are an advanced form of machine learning that:

- Recognizes patterns and features of input data
- Provides a clear quantitative output

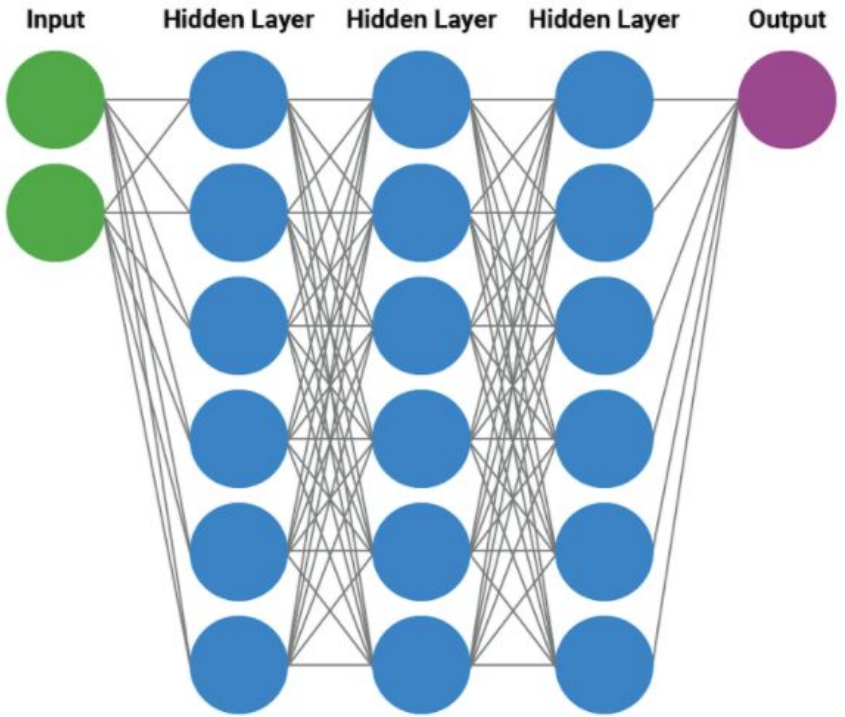


Neural Net

Example diagram of a neural network:

A neural network can train on each pixel of each image as a scaled value from 0 (completely white) to 1 (completely black).

With enough data points, a trained neural network model can classify handwritten numbers with a high degree of accuracy.



The MNIST Dataset

The MNIST (Modified National Institute of Standards and Technology) dataset contains black and white images of handwritten numbers.

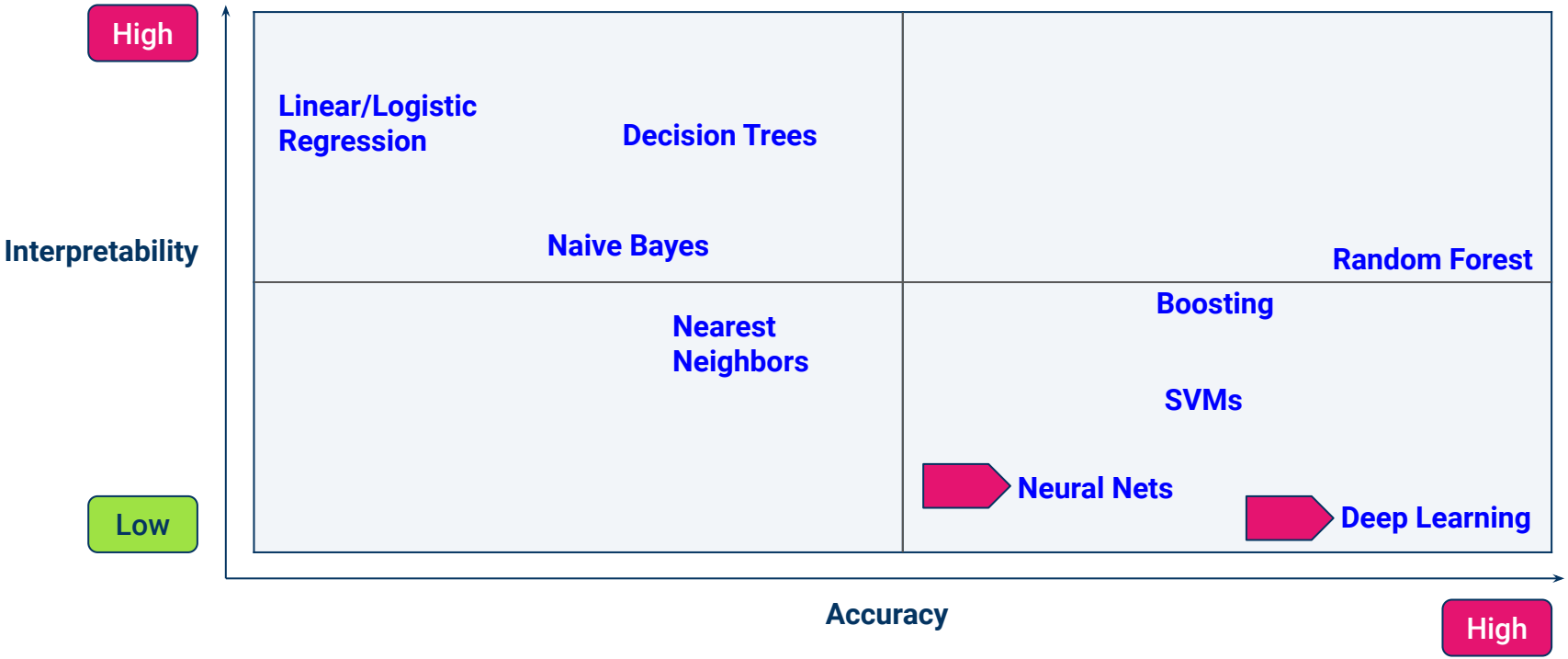
A neural network can train on each pixel of each image as a scaled value from zero (completely white) to one (completely black).

With enough data points, a trained neural network model can classify handwritten numbers with a high degree of accuracy.

Sample images from MNIST test dataset

Neural Net

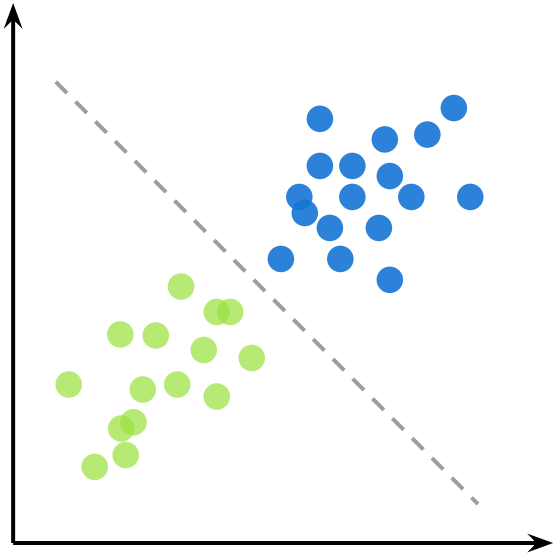
The following diagram plots the different types of machine learning models:



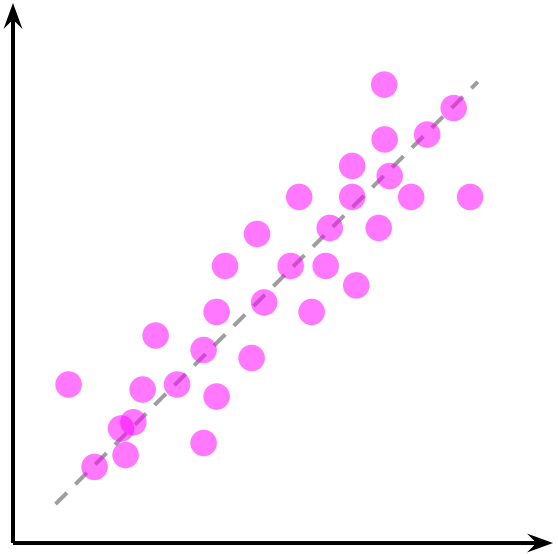
Neural Net

Neural networks are very useful in data science because they serve multiple purposes. Two popular uses include:

Classification algorithms that determine how to categorize an input.



Regression models that predict a dependent output from independent input variables.



Activity: Working Through the Logistics

In this activity, you'll use a logistic regression to categorize data from the `make_blobs` function from scikit-learn to create data.

Suggested Time:

15 minutes

Activity: Working Through the Logistics

Instructions	Use the starter code provided to create your `make_blobs` dataset from scikit-learn.
	Split your dataset into training and testing sets using scikit-learn's `train_test_split` module.
	Create a LogisticRegression instance from scikit-learn's `LogisticRegression` model.
Hint	If you need a reminder on how to create a `LogisticRegression` model, review the Scikit-learn documentation.
	Train your `LogisticRegression` model on the training dataset.
	Evaluate your trained `LogisticRegression` model using the `accuracy_score` metric from scikit-learn.

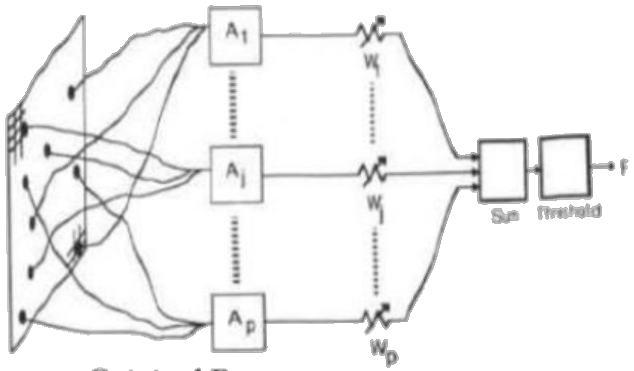
Time’s Up! Let’s Review.

Perceptron, the Computational Neuron

Perceptron

Artificial neural networks have become popular in recent years. But, the original design for a computational neuron, known as a perceptron, dates back to the late 1950s.

Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.

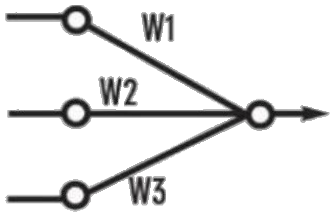


Original Perceptron

(From Perceptrons by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press.

Frank Rosenblatt
(1928-1971)

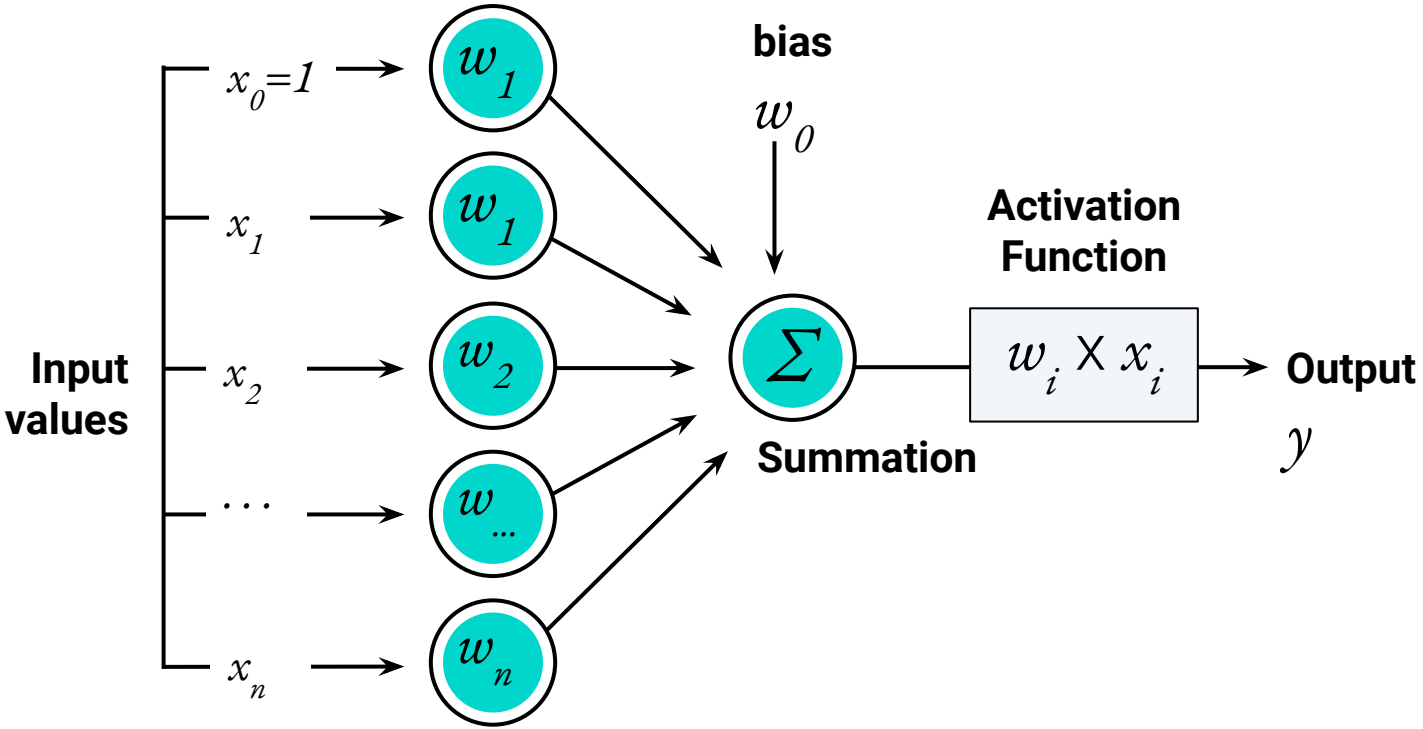
Simplified model:



A **Perceptron** is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

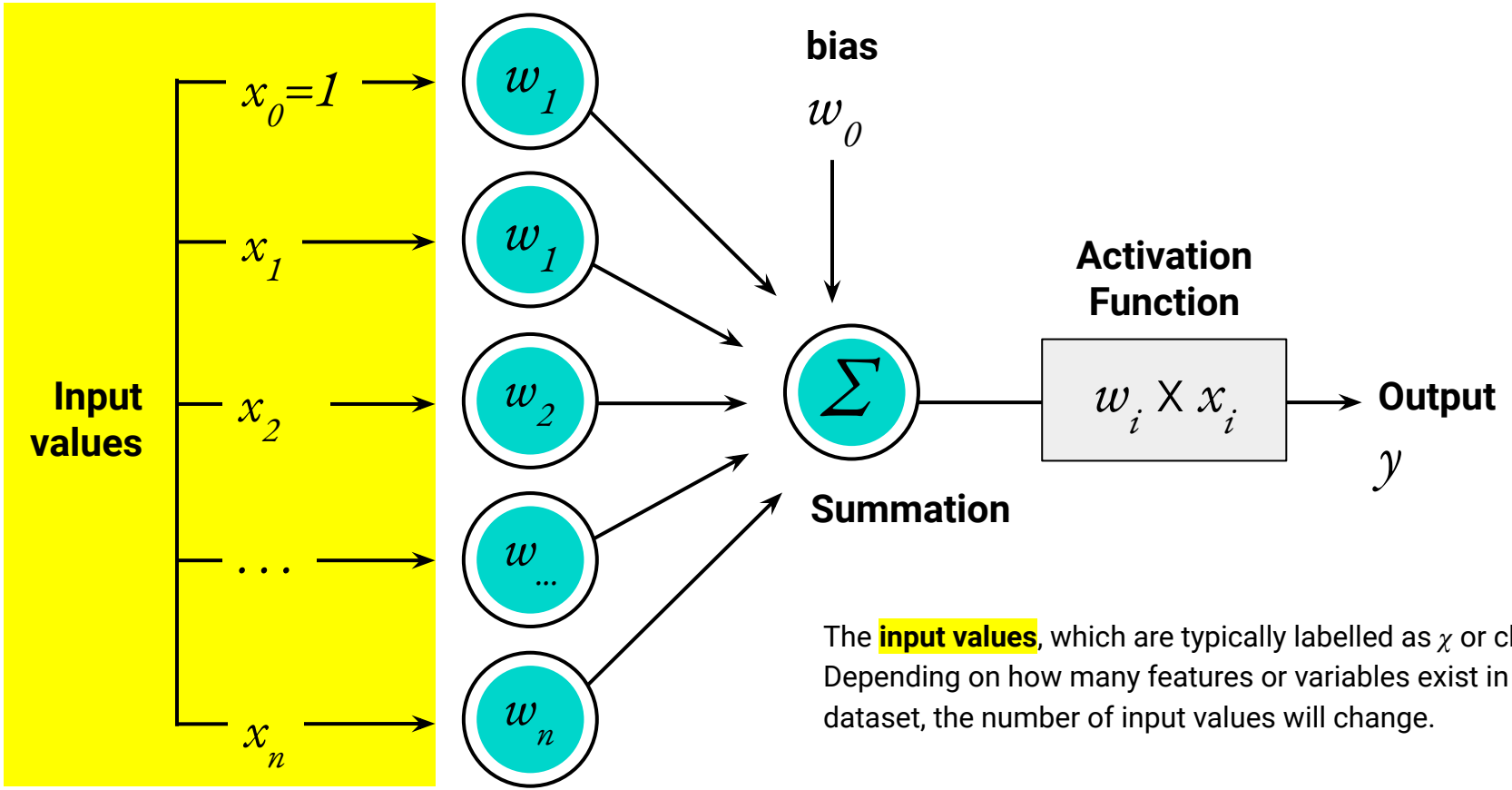
Meet the Perceptron

The **perceptron model** mimics a biological neuron by receiving input data, weighting the information, and producing a clear output.



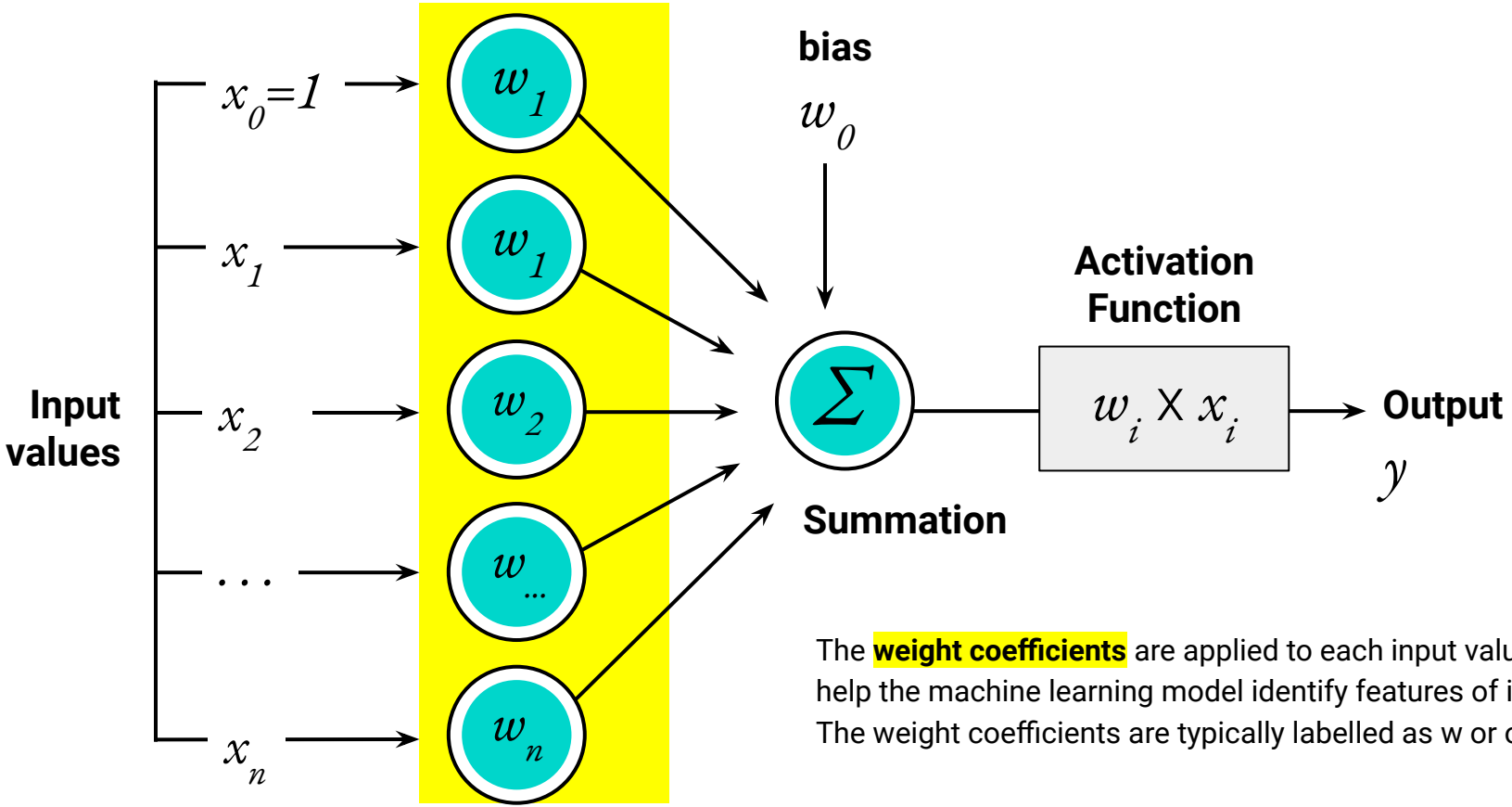
The perceptron model has four major components.

Input Values (shown as *x*'s)



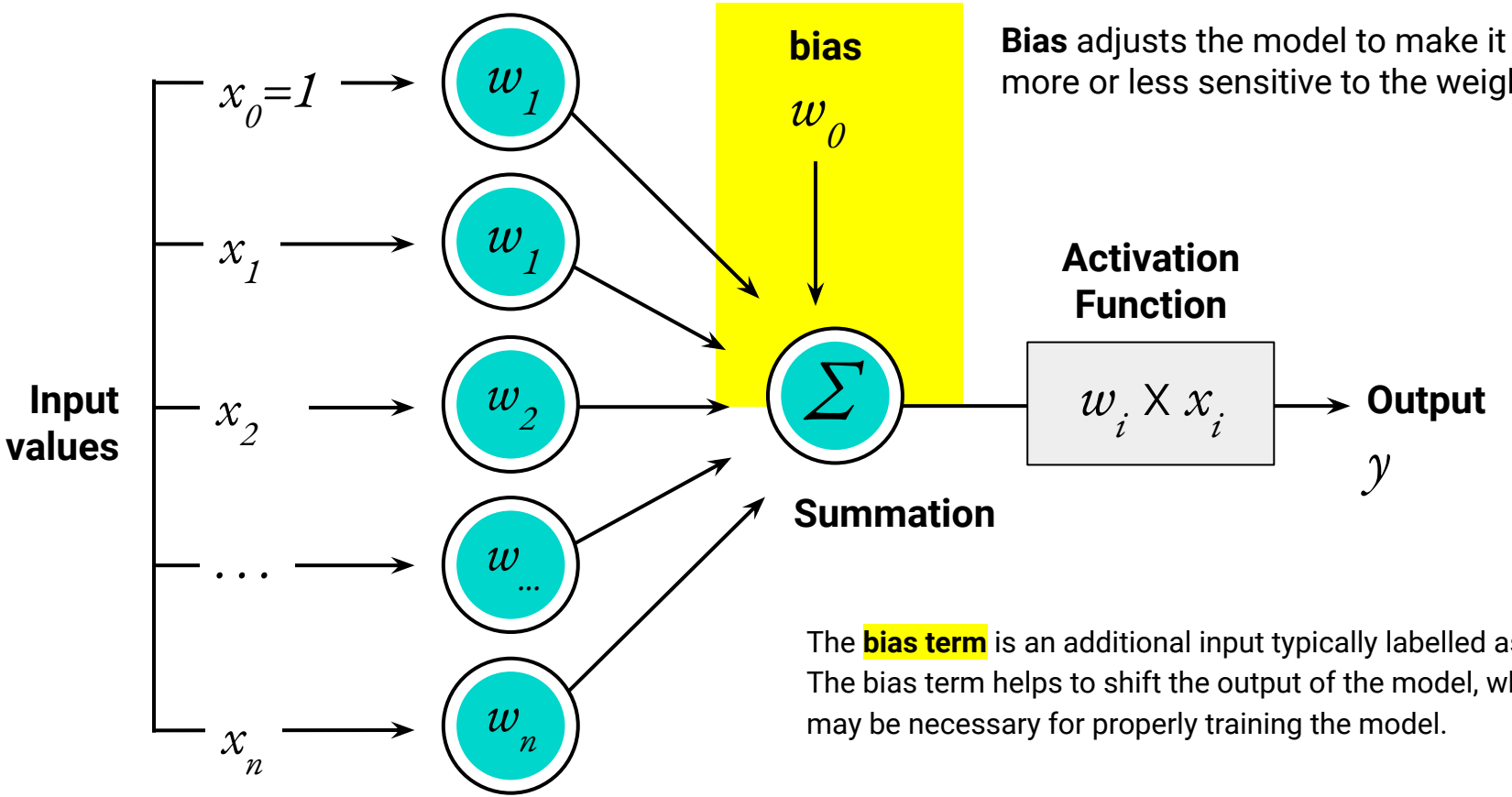
The **input values**, which are typically labelled as χ or chi. Depending on how many features or variables exist in the dataset, the number of input values will change.

Weight Coefficients (denoted as *w*'s)



The **weight coefficients** are applied to each input value to help the machine learning model identify features of interest. The weight coefficients are typically labelled as w or omega.

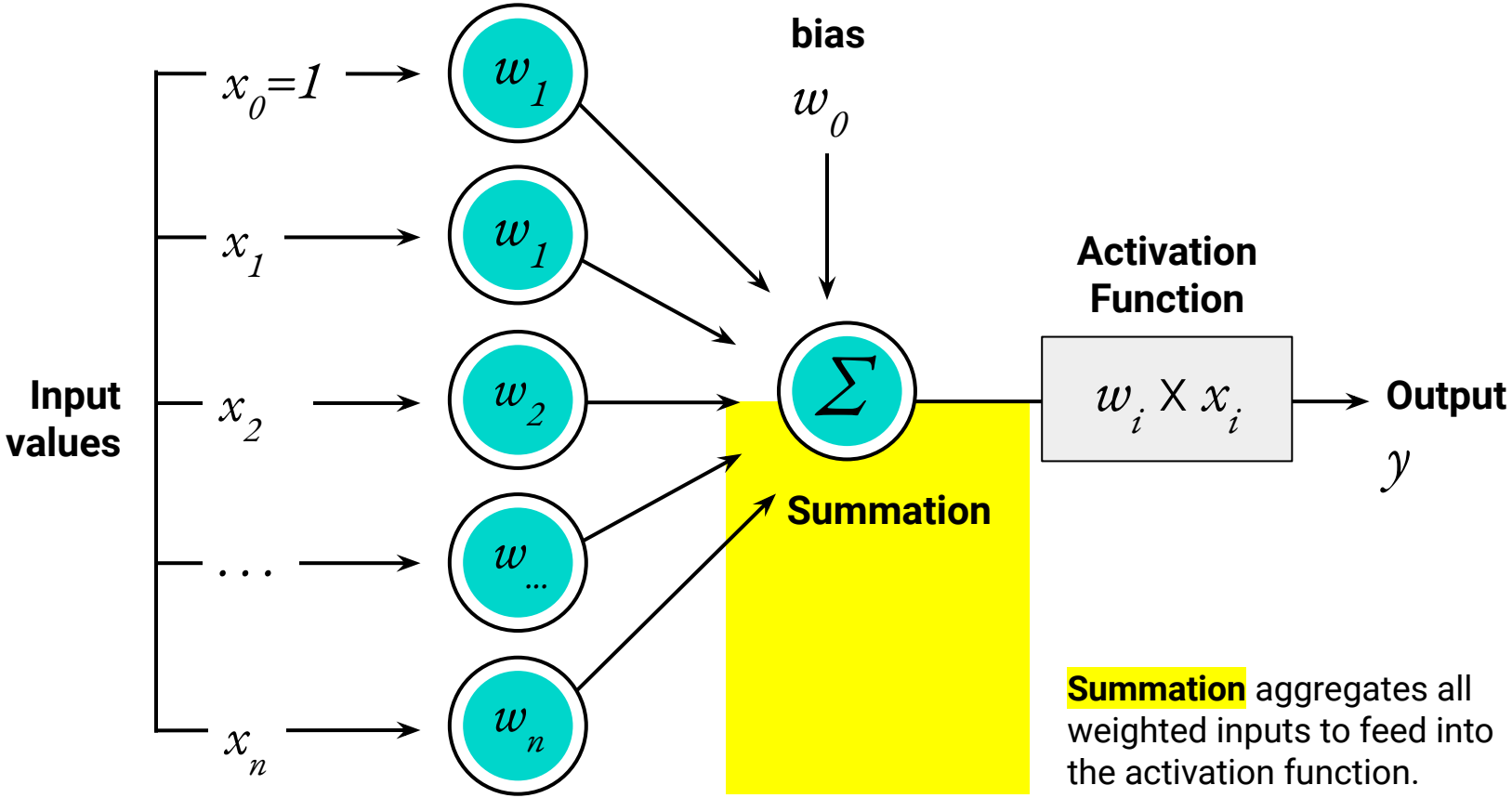
A Constant Value Called Bias



Bias adjusts the model to make it either more or less sensitive to the weights.

The **bias term** is an additional input typically labelled as ω. The bias term helps to shift the output of the model, which may be necessary for properly training the model.

Net Summary Function (the Summation)

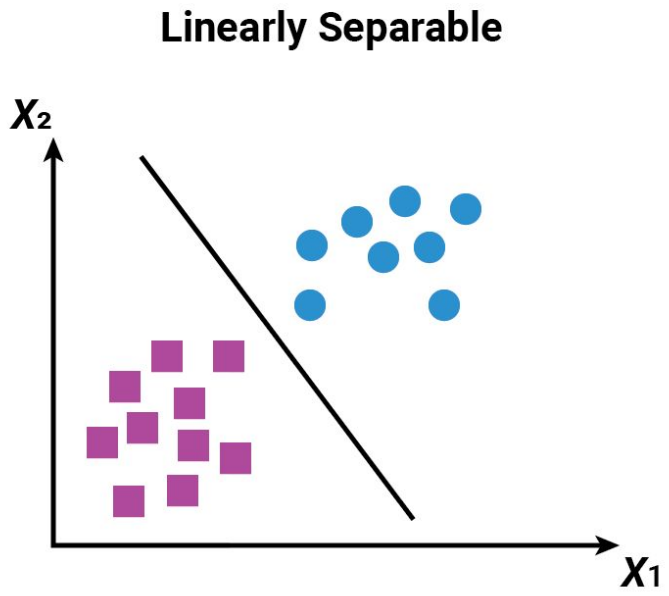


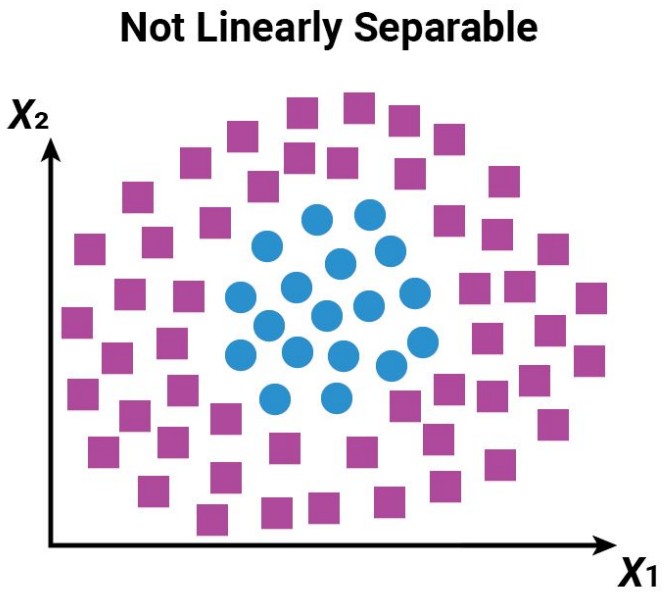
Summation aggregates all weighted inputs to feed into the activation function.

A perceptron model—also known as a **linear binary classifier**—contains a linear equation that separates data into two groups. This is the most common use.

Linearly Separable

Consider linearly separable, the perceptron algorithm separate and classify the data into two groups using linear equation.



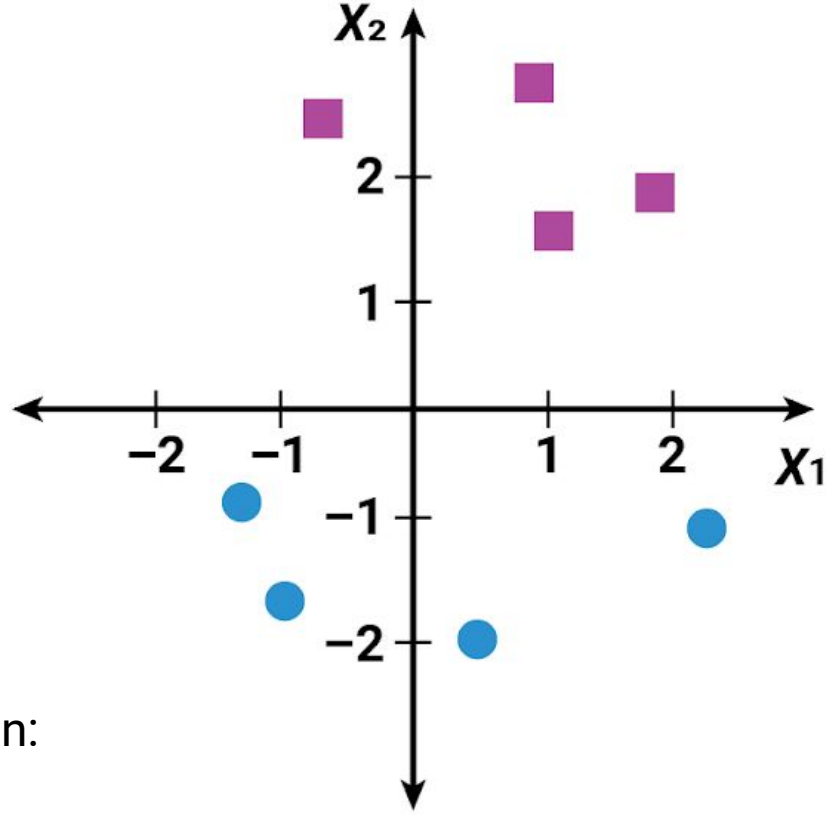


Since we provide the model our input feature and parameters the perceptron model is a form of supervised machine learning.

Perceptron Example

Our model will try to classify values in a two-dimensional space; our perceptron model will use three inputs:

$\chi 1$	the *x* value
$\chi 2$	the *y* value
$\omega 0$	the bias constant



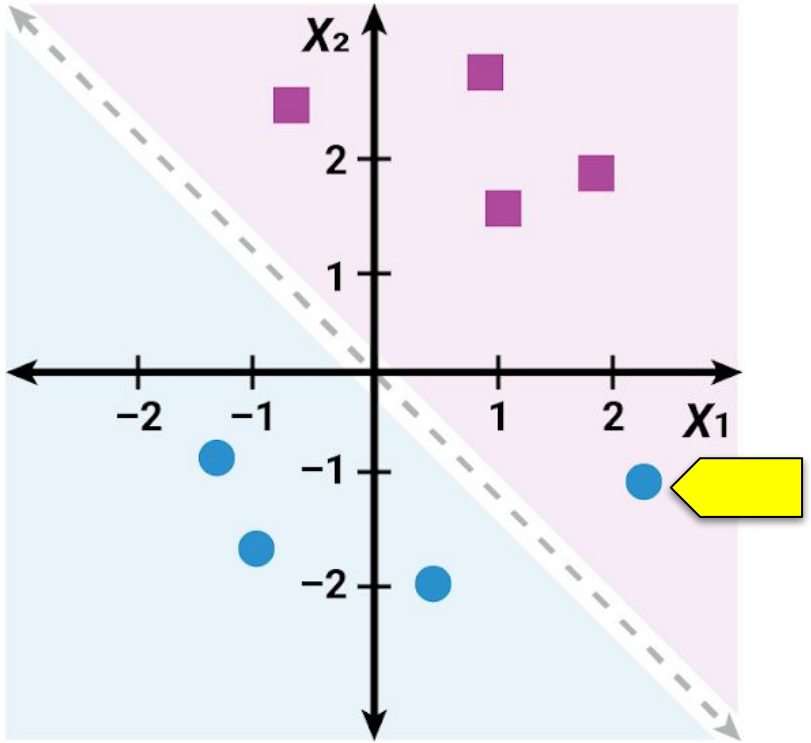
The end result of our two-dimensional perceptron model is the net sum function:
$\omega 0 + \chi 1 \omega 1 + \chi 2 \omega 2$.

If a data point is misclassified, the weights will move the model closer to the missed data point.

Untrained Perceptron Model on Our Dataset

An untrained model almost classified the two groups perfectly.

It misclassified one blue circle.

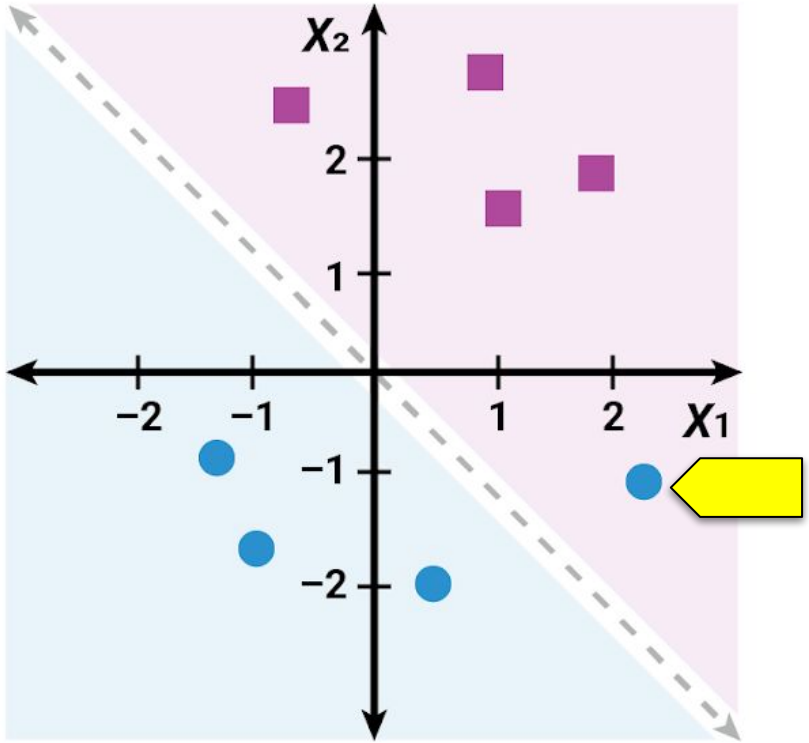


Untrained Perceptron Model on our Dataset

The perceptron model will evaluate each data point and determine if the input weights should change.

If a data point is classified correctly, the weights will not change.

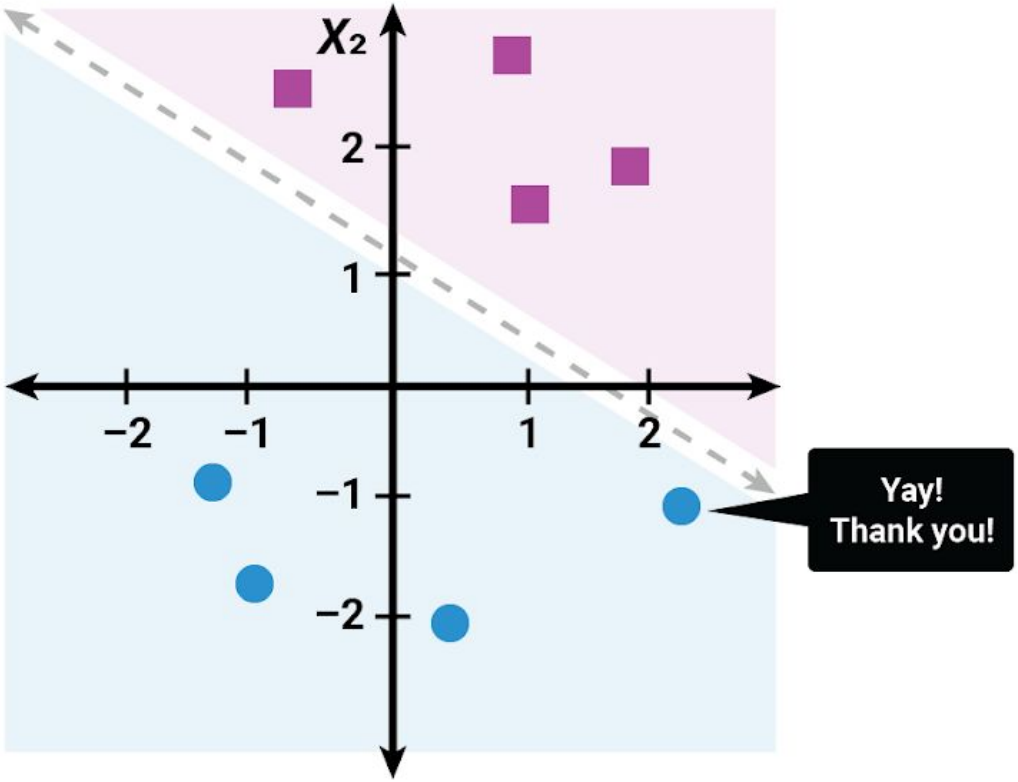
If a data point is misclassified, the weights will move the model closer to the missed data point.



Trained Perceptron Model on our Dataset

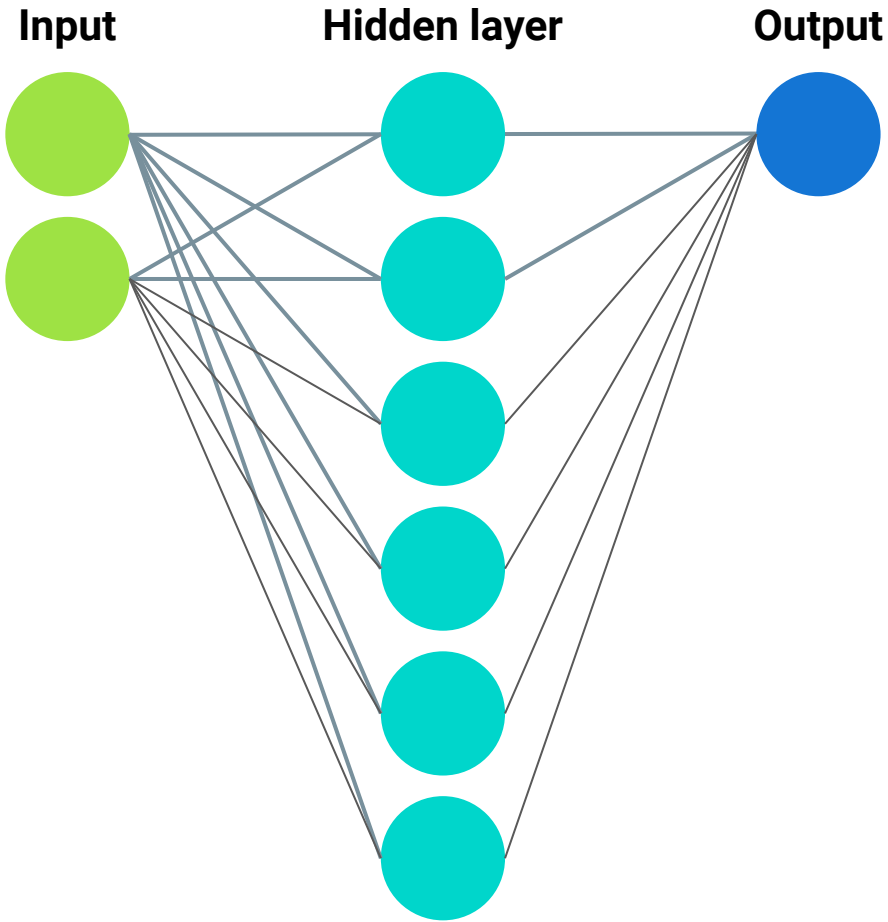
Each data point is evaluated to determine if the input weights should change.

Since a data point is misclassified, the weights will move the model closer to the missed data point.



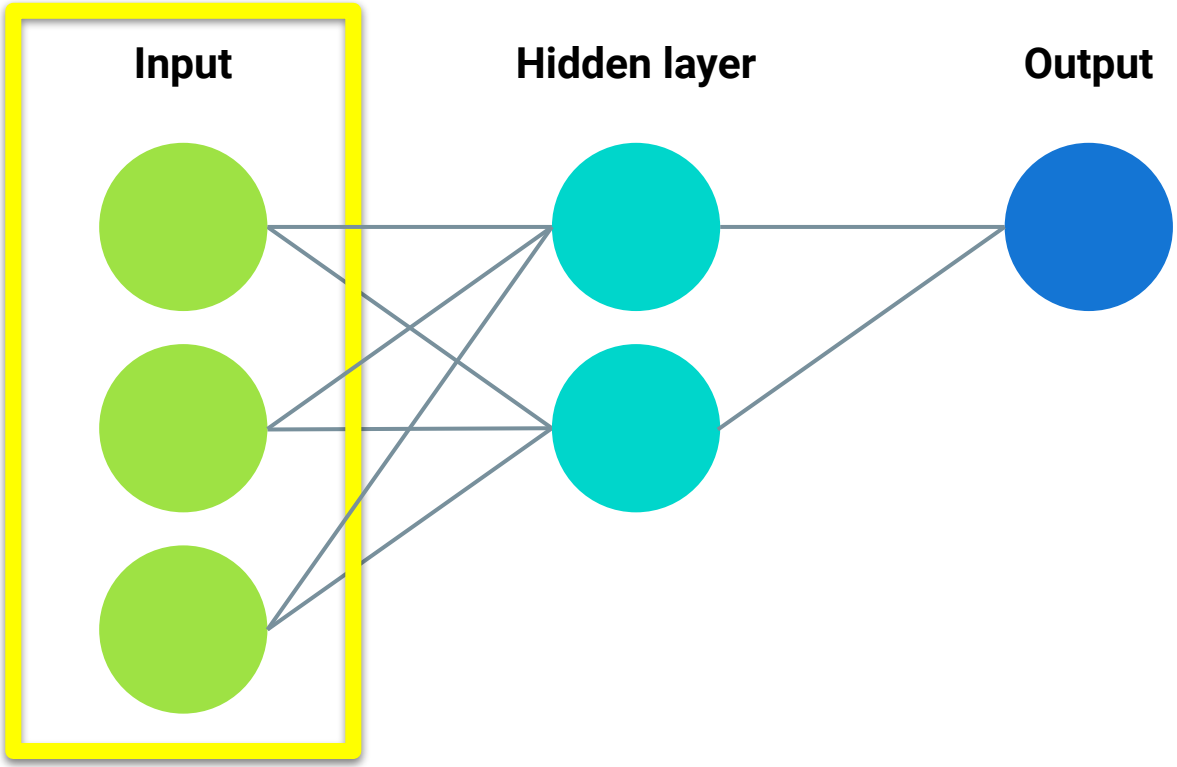
The Neural Network

A modern neural network model is a structure composed of several connected perceptrons that learn from input data to produce an output.



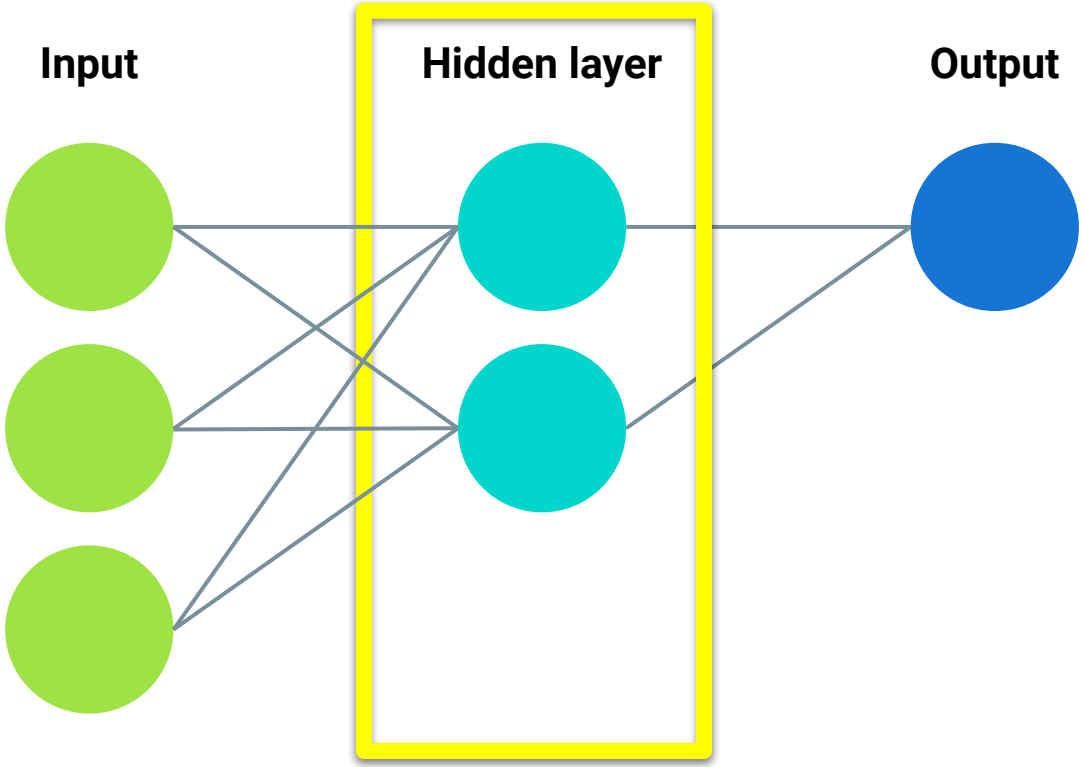
The Structure of a Neural Network

An **input layer** of input values transformed by weight coefficients



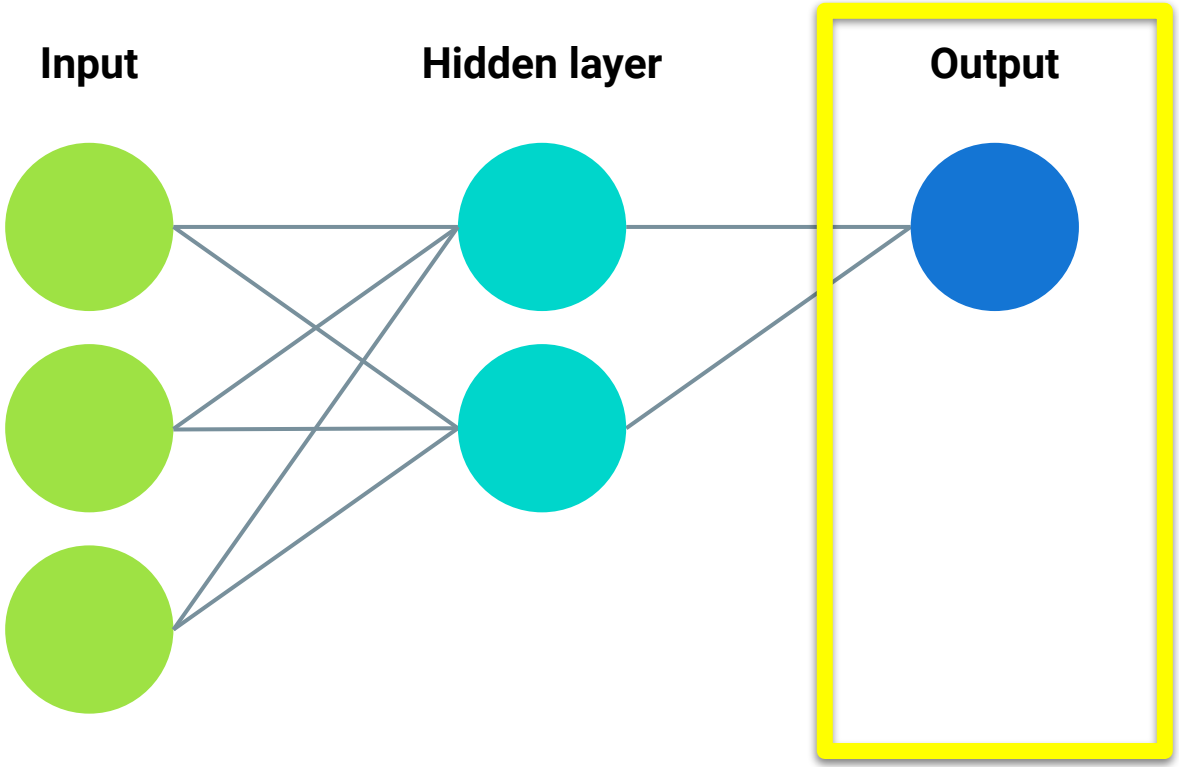
The Structure of a Neural Network

A single **hidden layer** that can contain a single neuron or multiple neurons



The Structure of a Neural Network

An **output layer** that reports the outcome of the value



If each neuron has its own output, how does the neural network combine each neuron's output into the model's classification or regression output?

Activation Functions

Activation Functions

Neural networks link neurons that process input together to produce a clear, quantitative output.

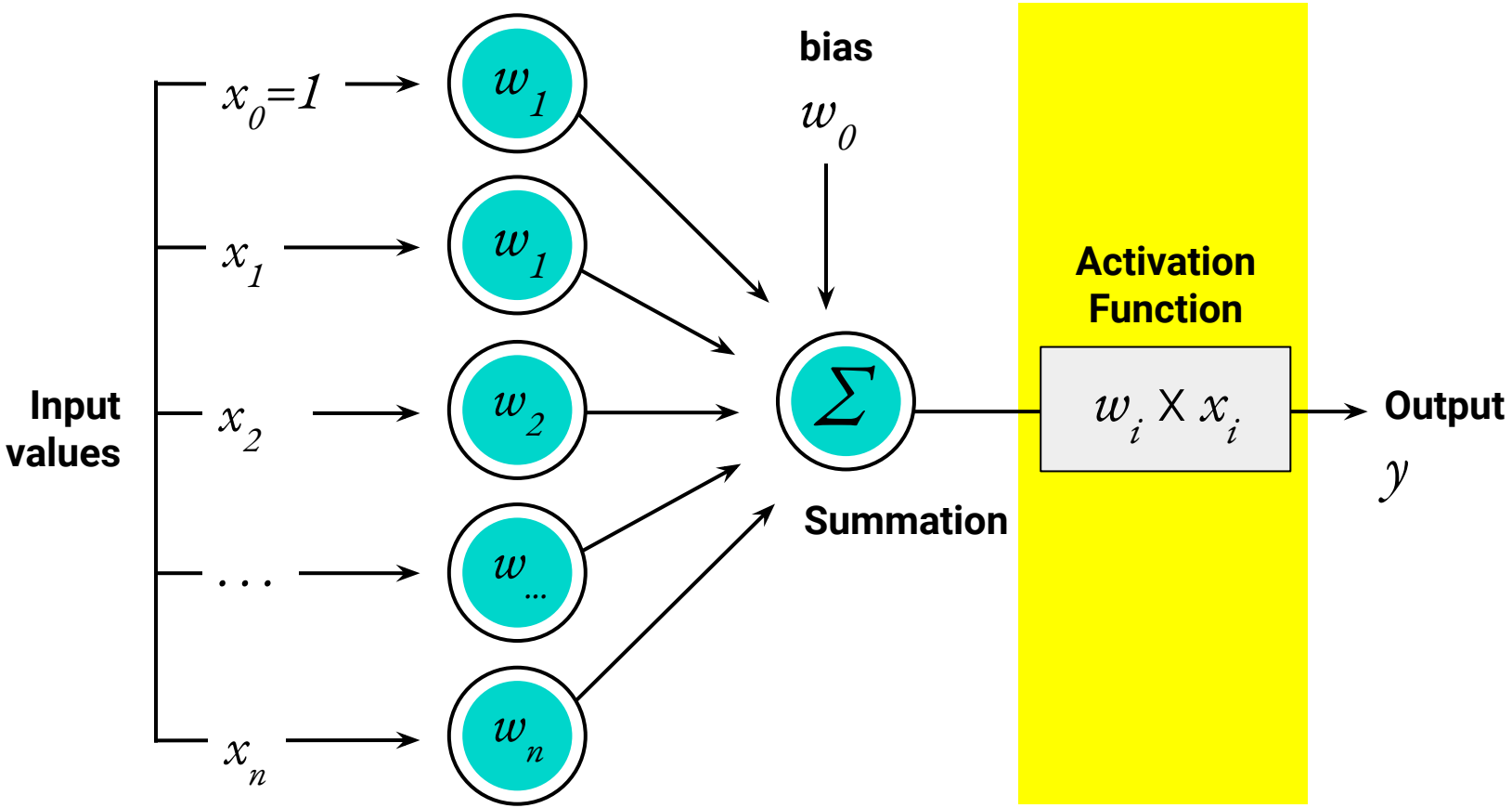
An **activation function** combines all these outputs into a single classifier or regression model.

When building a model, we apply the activation function to the end of each neuron (each individual perceptron model).

This mathematical function transforms each neuron's output into a quantitative value.

The quantitative output value becomes the input value for the next layer in the neural network model.

Activation Function (Transformation After Summation)

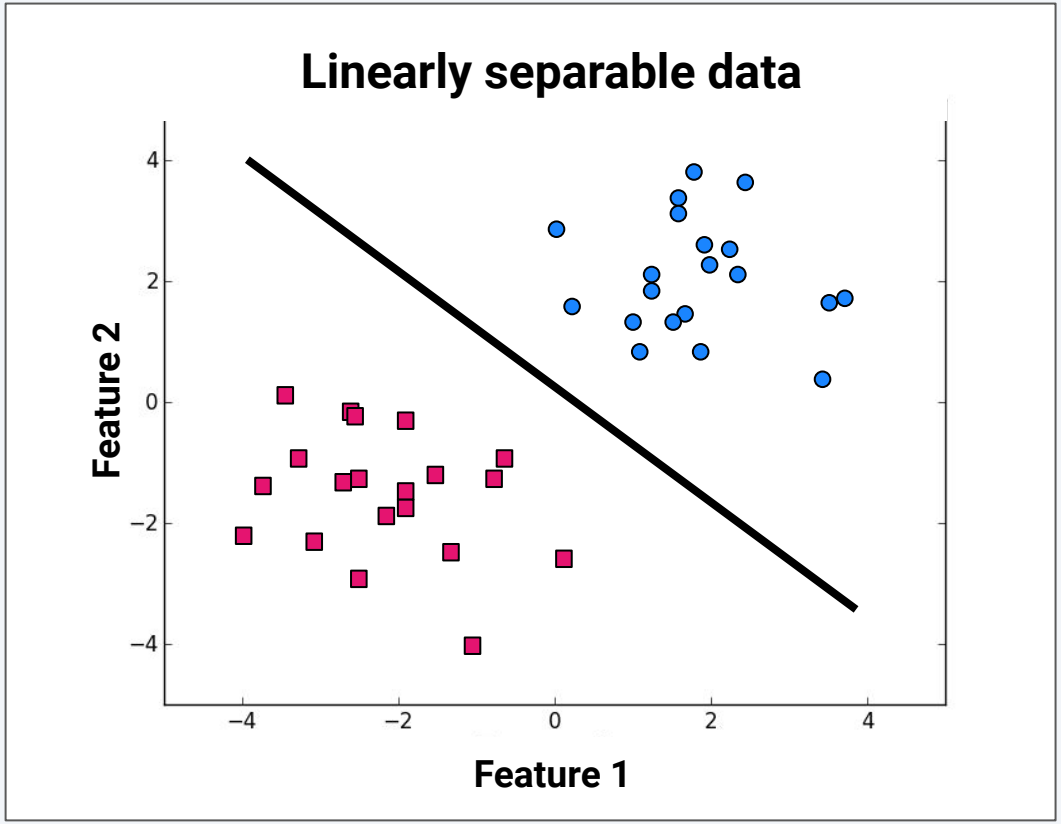


A wide variety of activation functions exist, and each has a specific purpose.

However, most neural networks will use one of the following activation functions.

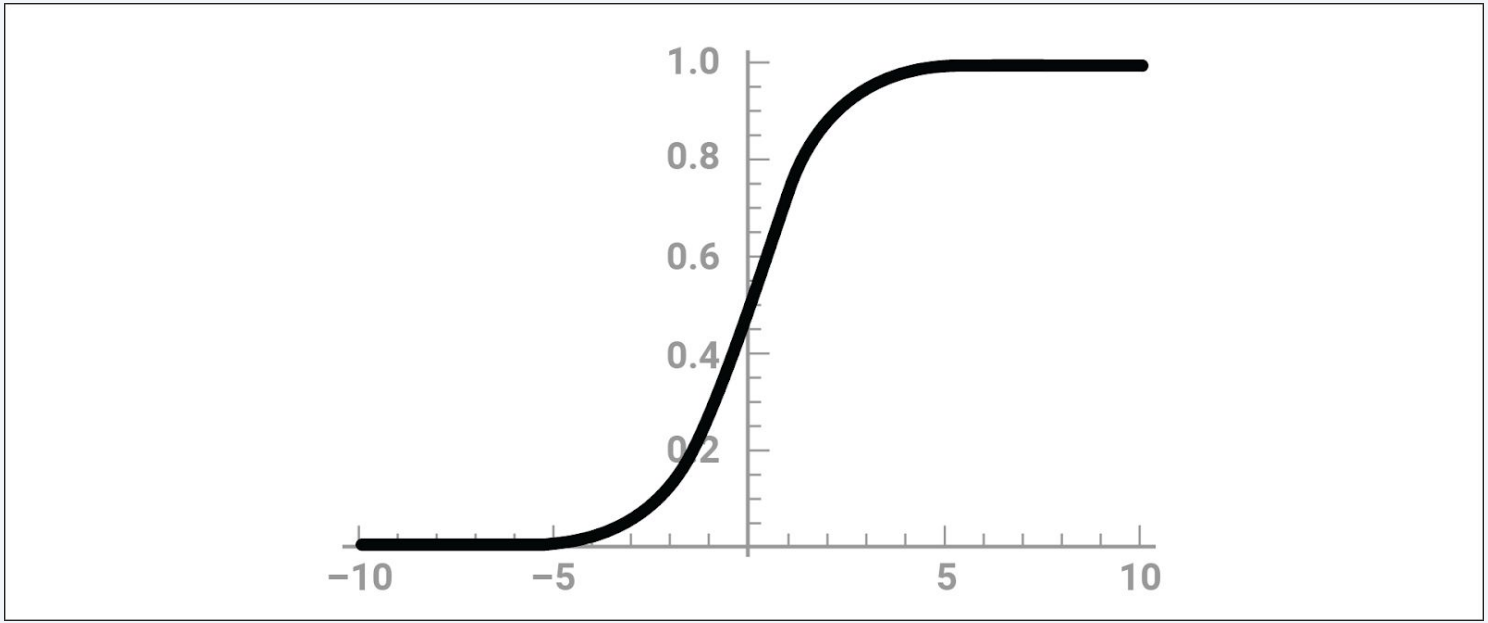
Linear Function

The **linear function** transforms the output into the coefficients of a linear model (the equation of a line).



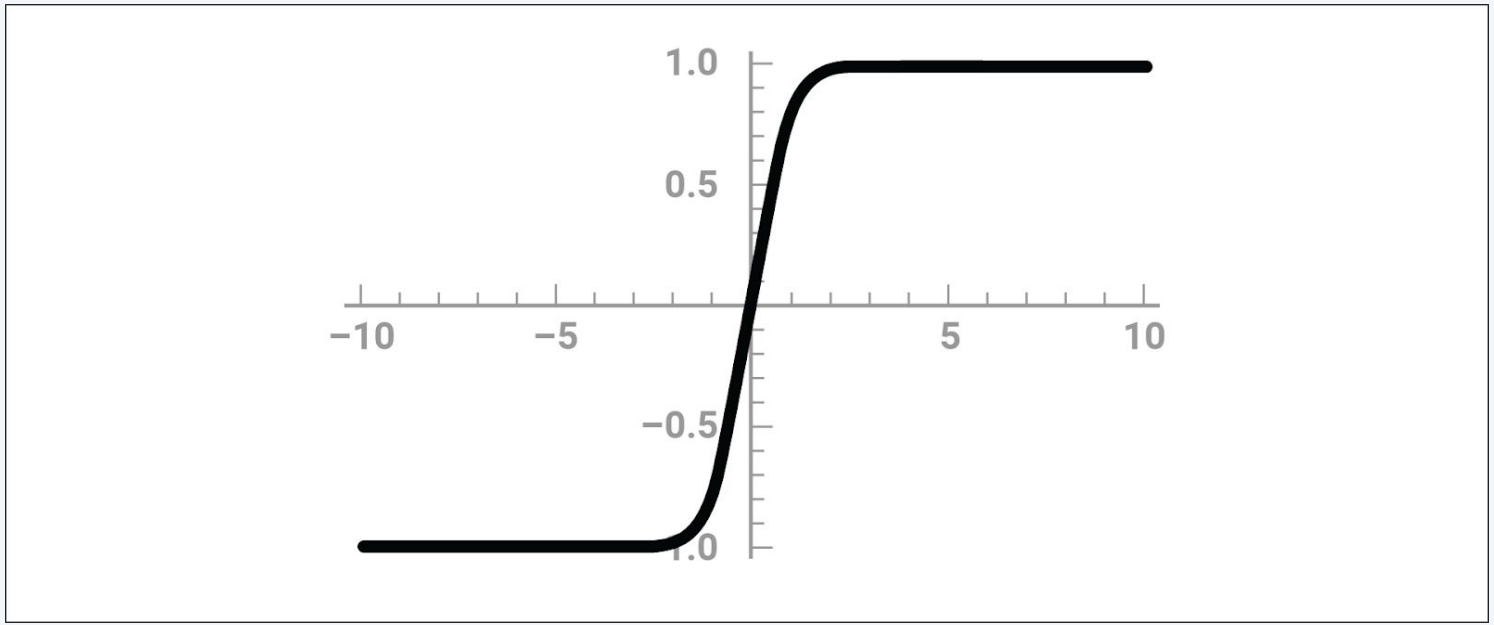
Sigmoid Function

The **sigmoid function** transforms the neuron's output to a range between 0 and 1, which is especially useful for predicting probabilities. A neural network that uses the sigmoid function will output a model with a characteristic S-curve.



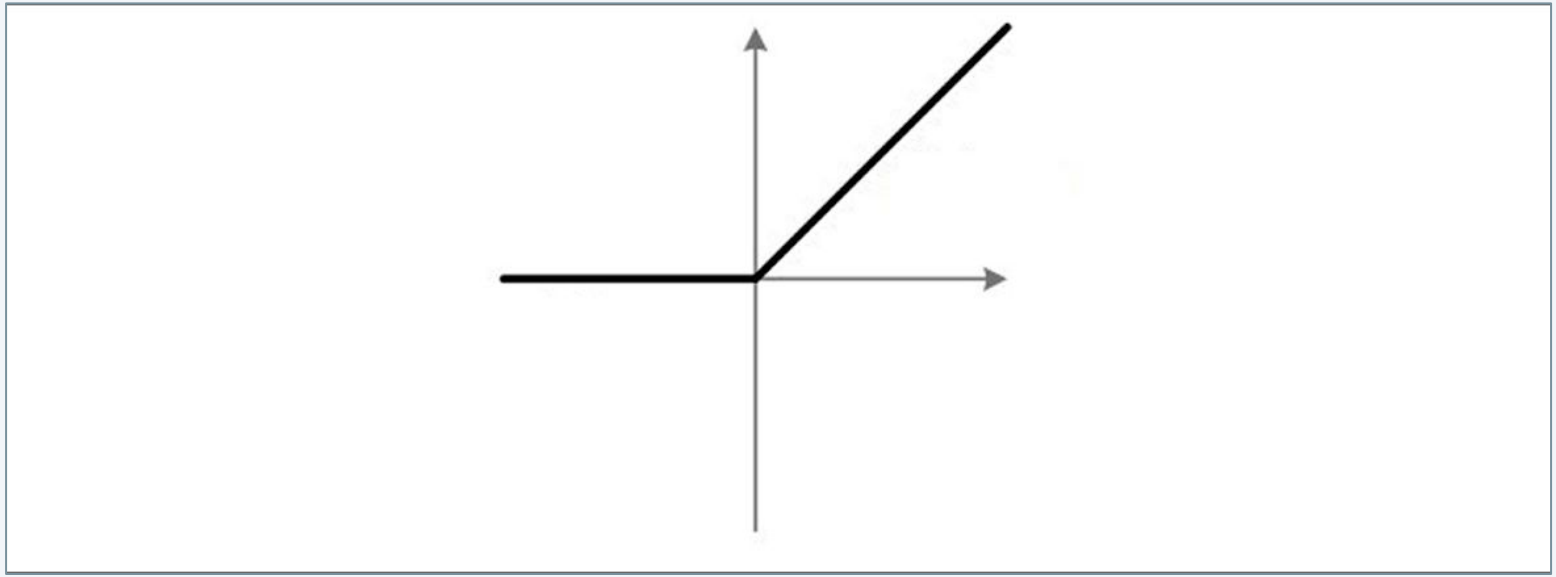
Tanh Function

The **tanh function** transforms the output to a range between −1 and 1.
The output for a model using a tanh function also forms a characteristic S-curve.
It's primary use is classifying data into one of two classes.



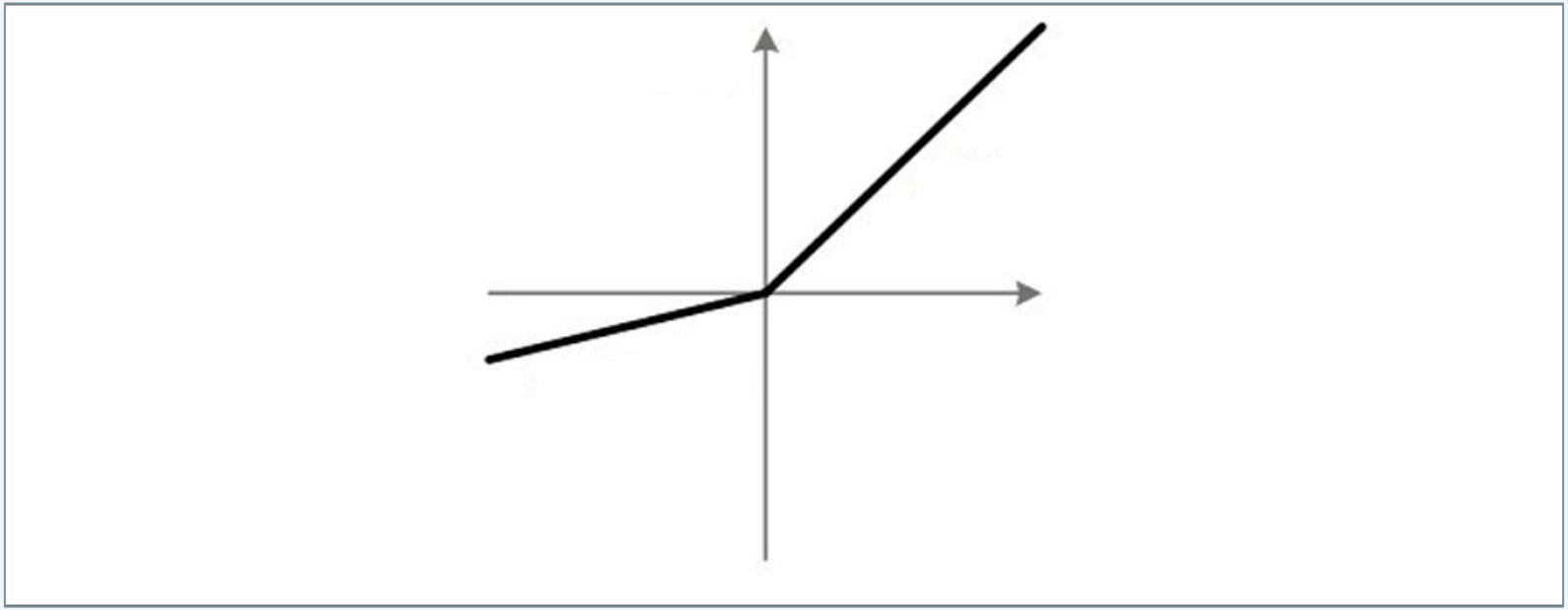
Rectified Linear Unit (ReLU)

The **rectified linear unit (ReLU)** function returns a value from 0 to infinity. This activation function transforms any negative input to 0. It is the most commonly used activation function in neural networks due to its faster learning and simplified output. However, it is not always appropriate for simpler models.



Leaky ReLU Function

The **leaky ReLU function** is an alternative to the ReLU function, which can sometimes work better. Instead of transforming negative input values to 0, it transforms them into much smaller negative values.



Questions?

TensorFlow Playground

TensorFlow is a neural network and machine learning library for Python that has become an industry standard for developing robust neural network models.

Creating a Neural Network

Why TensorFlow?

- Machine learning library
- Used to build neural networks
- Runs on multiple platforms
- Supports distributed computing
- Allows detailed fine-tuning

Modeling Workflow

Regardless of what machine learning model or technology we use, we follow the same general modeling workflow across all of data science:

01 Decide on a model and create a model instance.

02 Split the dataset into training and testing sets and preprocess the data.

03 Train/fit the training data to the model.

04 Evaluate the model for predictions and transformations.

Time to Code

TensorFlow Playground

Suggested Time:

15 minutes

Questions?

Instructor Demonstration

Setup Google Colab

Questions?

Instructor Demonstration

Understanding the TensorFlow Neural Network Structure

Break

Time to Code

Work Through a Neural Network Workflow

Suggested Time:

20 minutes

Activity: BYONNM—Build Your Own Neural Network Model

In this activity, you'll use a neural network model to classify the data from `make_blobs`.

Suggested Time:

20 minutes

Activity: Working Through the Logistics

Instructions	Using the starter code provided, visualize the blobs dummy dataset using a Pandas scatter plot.
	Randomly split the dummy data into training and test datasets using scikit-learn's `train_test_split` method.
	Normalize both datasets using scikit-learn's `StandardScaler` class.
	Using the Keras module, create a basic neural network with five neurons in the hidden layer. **Note:** Your neural network should use two inputs and produce one classification output.
	Compile your basic neural network model.
	Train the neural network model over 50 epochs.
	Evaluate the performance of your model by printing your test loss metric and determining the predictive accuracy of the model on the test dataset.
Bonus	Create a new neural network with a different number of neurons.
	Train the new neural network model on the same training data and test the performance on the same testing dataset.
	Create a line plot that visualizes the neural network predictive accuracy over each epoch.

Time's Up! Let's Review.

Recap