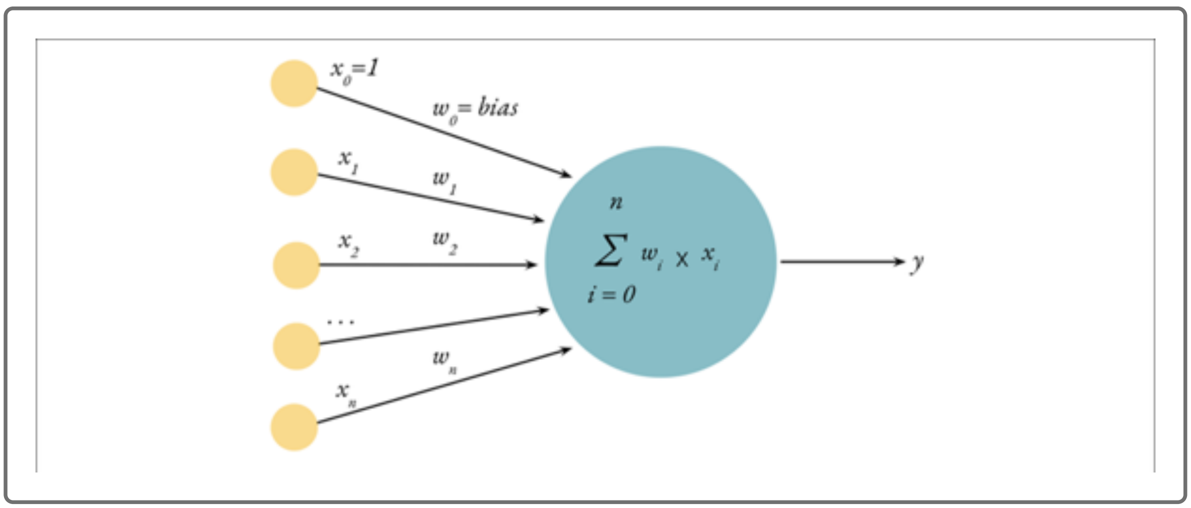
19.1.2 Perceptron, the Computational Neuron

Brain metaphors, easy to use, and cost-effective? Excellent at detecting complex, nonlinear relationships? Neural networks are starting to sound like a great fit for the model. Beks sends Andy a quick Slack message to let him know the research phase of the project is well under way. Her next step will be to dig into the math a bit: What exactly goes into a neural network? To explore this, she'll start with the perceptron model.

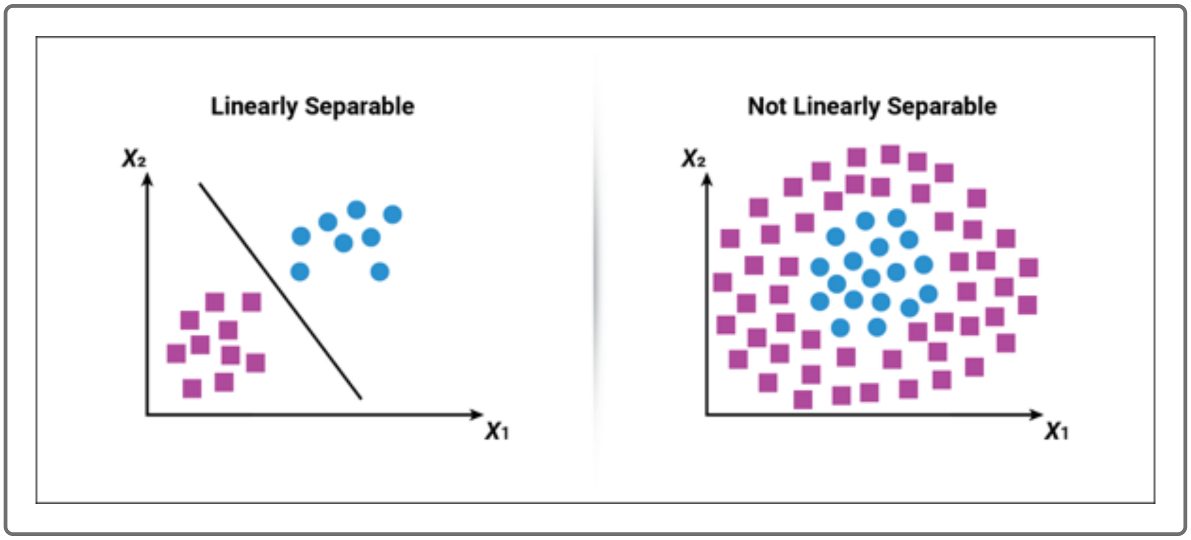
Although artificial neural networks have become popular in recent years, the original design for computational neurons (and, subsequently, the neural network) dates as far back as the late 1950s, when Frank Rosenblatt, a pioneer in the field of artificial intelligence, created the perceptron, a machine for training the first neural network. The **perceptron model** is a single neural network unit, and it mimics a biological neuron by receiving input data, weighing the information, and producing a clear output.

The perceptron model has four major components:

- **Input values**, typically labelled as **x** or χ (chi, pronounced kaai, as in eye)
- A **weight coefficient** for each input value, typically labelled as **w** or ω (omega)
- **Bias** is a constant value added to the input to influence the final decision, typically labelled as $\mathbf{w_0}$. In other words, no matter how many inputs we have, there will always be an additional value to "stir the pot."
- A **net summary function** that aggregates all weighted inputs, in this case a weighted summation:



Perceptrons are capable of classifying datasets with many dimensions; however, the perceptron model is most commonly used to separate data into two groups (also known as a **linear binary classifier**). In other words, the perceptron algorithm works to classify two groups that can be separated using a linear equation (also known as **linearly separable**). For example, in the image below, we have a purple group and blue group in both figures. In the left figure, we can draw a line down the middle of the figure to separate the groups entirely. While in the right figure, there is no single (straight) line that can be drawn to separate the two groups entirely. Therefore, the left image is considered linearly separable while the right is not:



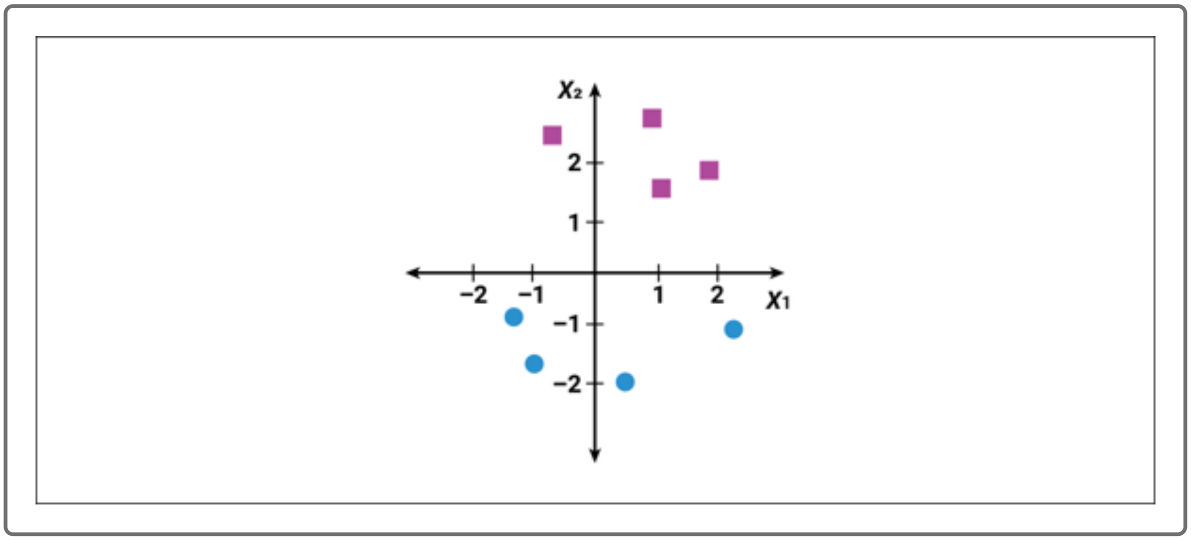
Refer to the perceptron model with labeled input data, above. Does it represent supervised or unsupervised machine learning?

- Supervised machine learning
- Unsupervised machine learning

Check Answer

Finish ▸

The perceptron model is designed to produce a discrete classification model and to learn from the input data to improve classifications as more data is analyzed. To better understand how the perceptron model and algorithm works, let's consider the following dataset:



In this example, we want to generate a perceptron classification model that can distinguish between values that are purple squares versus values that are blue circles. Since this perceptron model will try to classify values in a two-dimensional space, our input values would be:

- χ_2 - the y value
- χ_1 - the x value
- ω_0 - the constant variable (which becomes the bias constant)

IMPORTANT

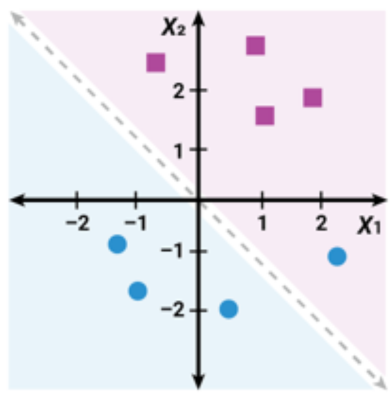
> There will always be one more input variable than the number of dimensions to ensure there is a bias constant within the model.

As for the weight and bias coefficients, these values are arbitrary when the perceptron model first looks at the data. As a result, the two-dimensional perceptron's net sum function would be:

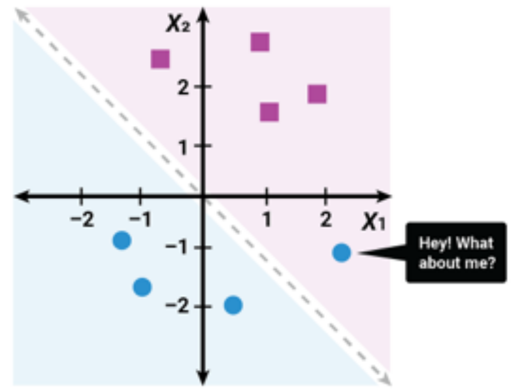
$\omega_0 + \chi_1\omega_1 + \chi_2\omega_2$

Where ω_0 is the bias term, and $\chi_1\omega_1$ and $\chi_2\omega_2$ are the weighted x and y values for each data point. If the net sum of the data point is greater than zero, it classifies the data point as a purple square, otherwise the data point is classified as a blue circle.

Due to the initial weight coefficients being arbitrary, it is very likely that the first iteration of the perceptron model will classify values incorrectly. Let's say that the first iteration of our perceptron model looks like the following image:

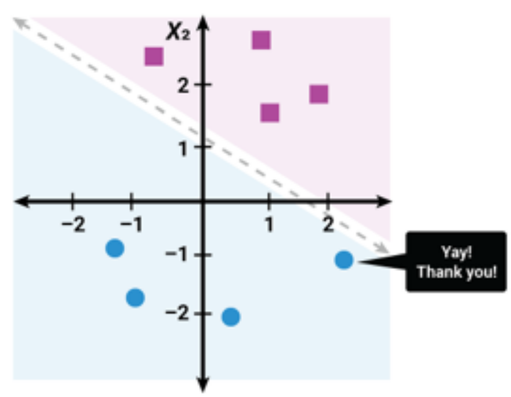


In this image, the perceptron's linear classifier is represented with the dashed line. According to the perceptron model, values above the dashed line would be considered to be purple squares, and values below the dashed line would be considered to be blue circles. Although the perceptron model correctly classified all of the purple square data points, it misclassified one of the blue circle data points:



The next step in the perceptron algorithm is to check each data point and determine if we need to update the weight coefficients to better classify all data points. When the perceptron model evaluates all of the input data, the correctly classified data points will not change the weight coefficients; however, the incorrectly classified data points will adjust the weight coefficients to move toward the data point.

After adjusting the weights, the perceptron algorithm will reevaluate each data point using the new model:



As with other machine learning algorithms, this process of perceptron **model training** continues again and again until one of three conditions are met:

1. The perceptron model exceeds a predetermined performance threshold, determined by the designer before training. In machine learning this is quantified by minimizing the **loss** metric.
2. The perceptron model training performs a set number of iterations, determined by the designer before training.
3. The perceptron model is stopped or encounters an error during training.

At first glance, the perceptron model is very similar to other classification and regression models; however, the power of the perceptron model comes from its ability to handle multidimensional data and interactivity with other perceptron models. As more multidimensional perceptrons are meshed together and layered, a new, more powerful classification and regression algorithm emerges—the neural network.