Python102

Python for Data Science Bootcamp

(6.3) Machine Learning Basics with Python Part 3

AIAT Academy

Machine Learning Basics

- Machine Learning Basic Part 1
  - Machine Learning with Python using Scikit Learn
  - Linear Regression
  - Logistic Regression
- Machine Learning Basic Part 2
  - Support Vector Machine (SVM)
  - K means Clustering
- **Machine Learning Basic Part 3**
  - **Natural Language Processing (NLP)**
  - **Neural Network and Deep Learning**

Natural Language Processing (NLP)

Natural Language Processing (NLP)

- Imagine you work for Google News and you want to group news articles by topic
- Or you work for a legal film and you need to filter thousands of document pages to find relevant ones
- NLP can help you

Natural Language Processing (NLP)

- To do so, we want to
  - Compile documents
  - Featurize them
  - Compare their features

Natural Language Processing (NLP)

- To do so, we want to
  - Compile documents
  - Extract features from them
  - Compare their features

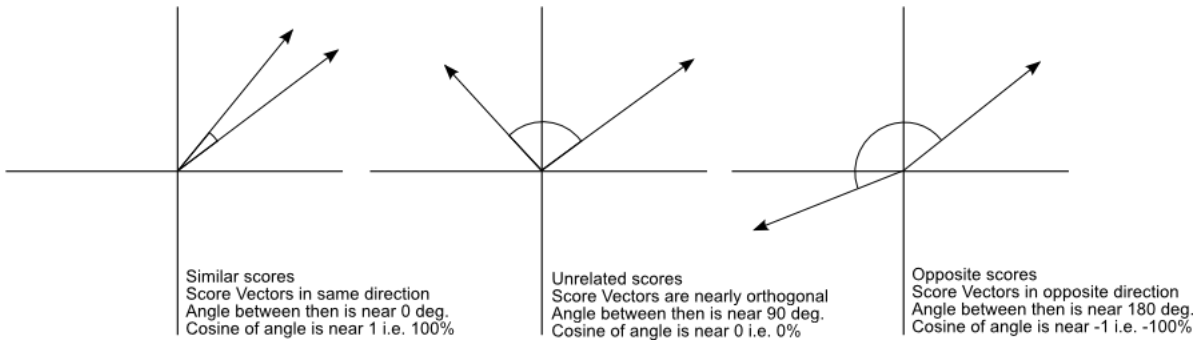
Natural Language Processing (Simple Example)

- You have two short documents
  - "Blue House"
  - "Red House"
- Extract features on word count
  - "Blue House": (Blue, House, Red): (1, 1, 0)
  - "Red House": (Blue, House, Red): (0, 1, 1)

Natural Language Processing (Simple Example)

- The documents represent as a **vector** or word count is called **"Bag of Words"**
  - "Blue House": (Blue, House, Red): (1, 1, 0)
  - "Red House": (Blue, House, Red): (0, 1, 1)
- In principle, you can use cosine similarity on the vectors to calculate similarity between documents

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$



Natural Language Processing (Simple Example)

- We can improve on Bag of Words by adjusting word counts based on their frequency in corpus (the group of all documents)
- We can use TF-IDF (Term Frequency – Inverse Document Frequency)
  - Term Frequency: Importance of the term (word) within that document
    - TF(d, t) = Number of occurrences of term t in document d
  - Inverse Document Frequency: Importance of the term in the corpus
    - IDF(t) = log(D/t)
    - where D = total number of documents and t = number of documents with the term

Natural Language Processing (Simple Example)

- Mathematically , TD-IDF is expressed:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Natural Language Processing with Python

- To start with Natural Language Processing on Python, we will need to download a library
- Go to your terminal or command line and use:

```
conda install nltk
```

or

```
pip install nltk
```

Natural Language Processing (NLP)
Colab

AiAT ACADEMY

Neural Networks

Neural Networks

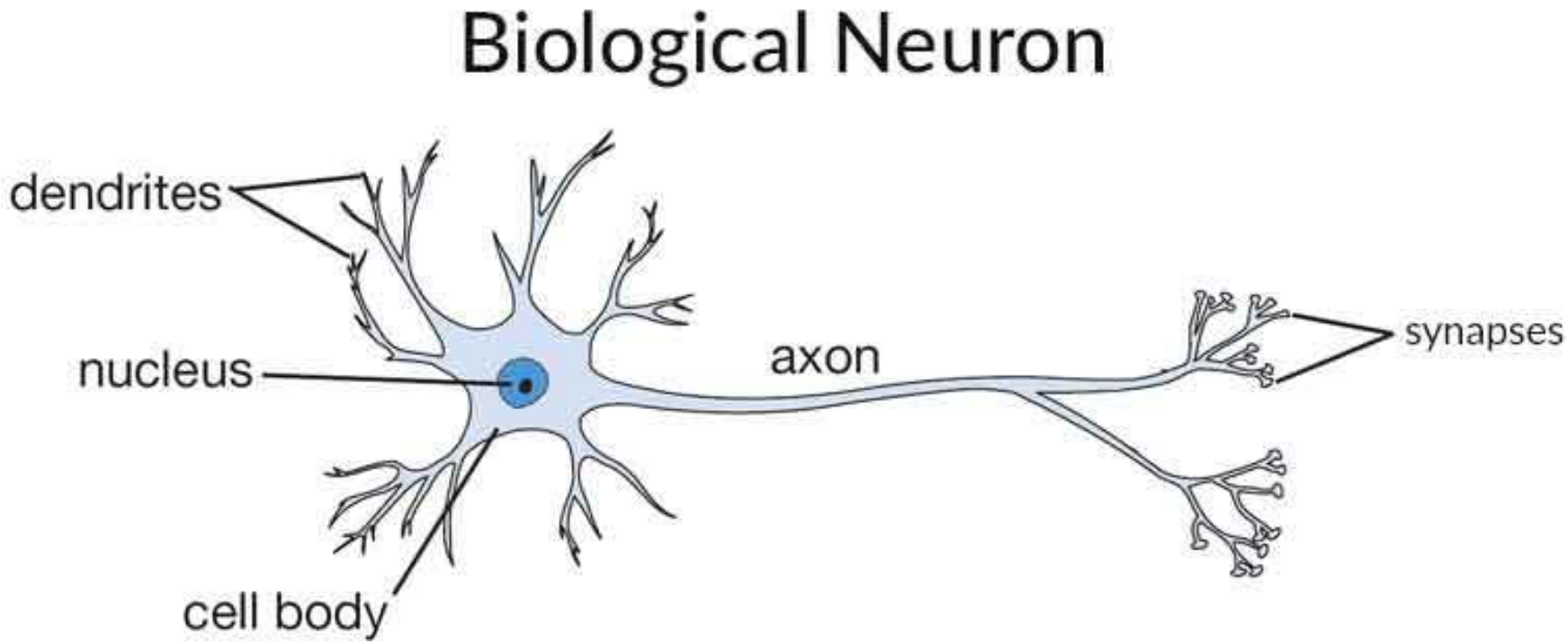
- Neural networks are modelled after biological neural networks and attempt to allow computers to learn in similar manner to humans (**reinforcement learning**)
- Use cases
  - Pattern Recognition
  - Time Series Prediction
  - Signal Processing
  - Anomaly Detection
  - Control

Neural Networks

- Neural networks are modelled after biological neural networks and attempt to allow computers to learn in similar manner to humans (**reinforcement learning**)
- Use cases
  - Pattern Recognition
  - Time Series Prediction
  - Signal Processing
  - Anomaly Detection
  - Control

Neural Networks

- The human brain has interconnected neurons with dendrites that receive inputs, and then based on those inputs, produce an electronical signal output through the axon

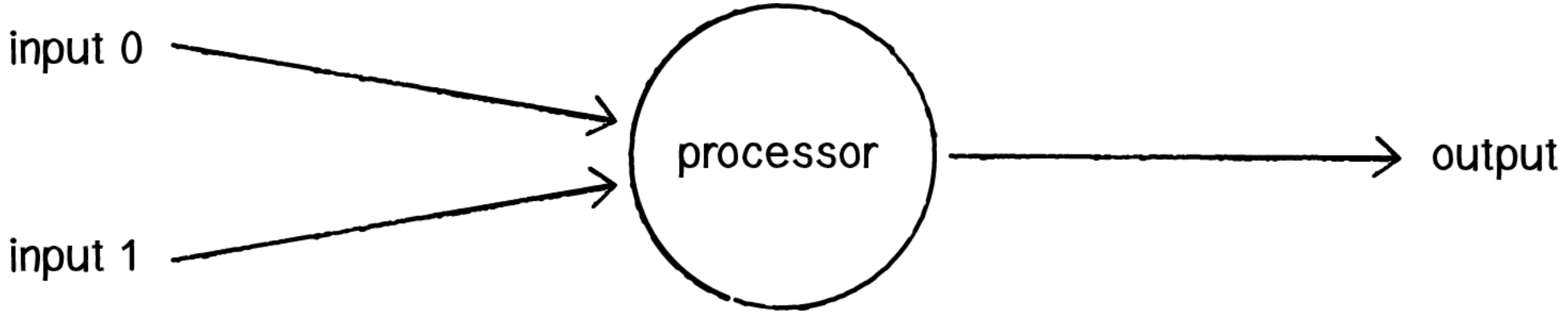


Neural Networks

- There are problems that are **difficult for humans but easy for computers**
  - Calculating large arithmetic problems
- Then there are problems **easy for humans but difficult for computers**
  - Recognizing a picture of a person from the side
- **Neural Networks** attempt to solve problems that would normally be easy for humans but hard for computers

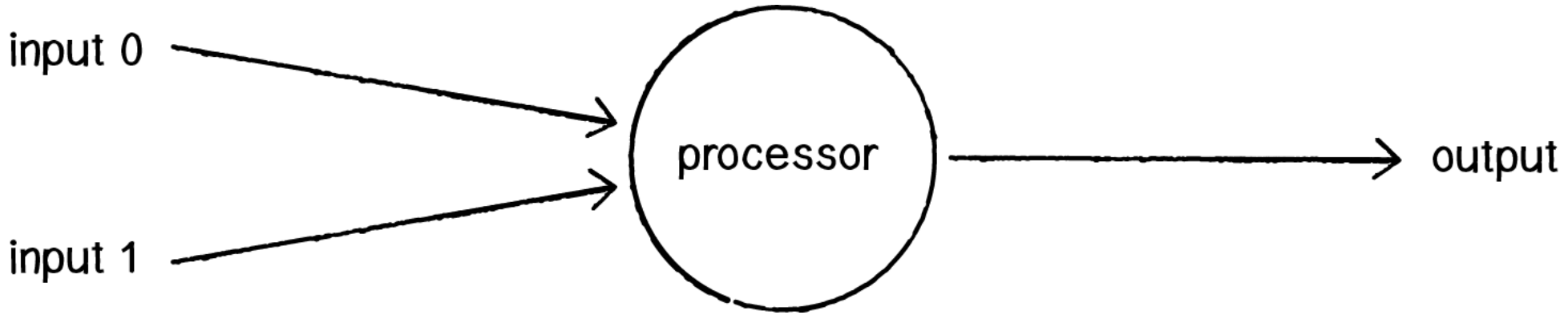
Neural Networks (Perceptron)

- **Perceptron** in Simplest Neural Network
  - A perceptron consists of one or more input, a processor, and a single output
  - A perceptron follows the "feed-forward" model, meaning inputs are sent into the neuron, are processed, and result in an output



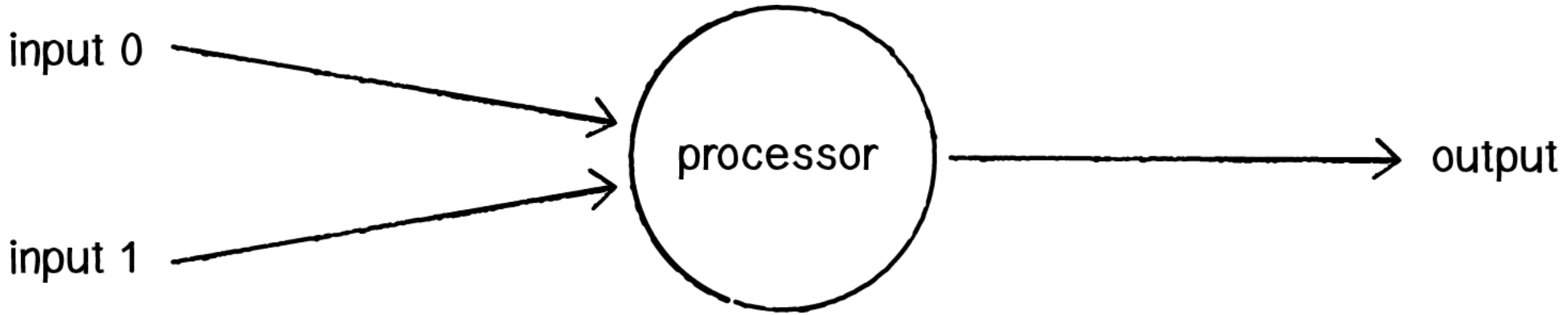
Neural Networks (Perceptron)

- A perceptron process follows 4 mains steps
  - Receive inputs
  - Weight inputs
  - Sum inputs
  - Generate output



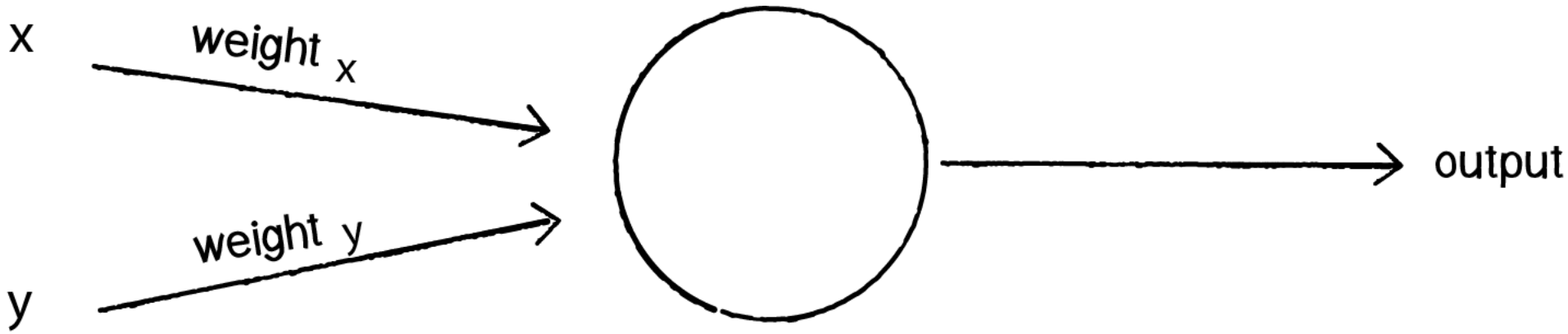
Neural Networks (Perceptron)

- Let's say we have a perceptron with two inputs
  - **Input 0: x1 = 12**
  - **Input 1: x2 = 4**



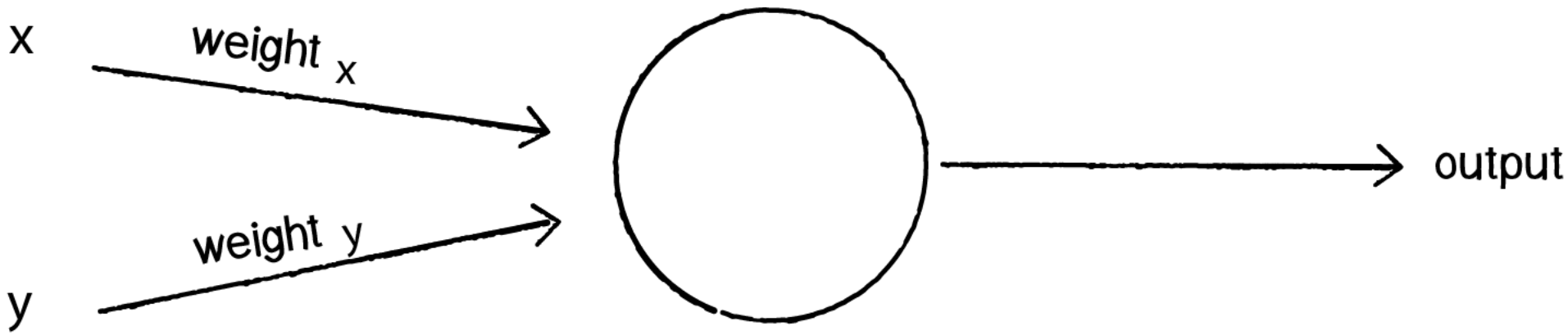
Neural Networks (Perceptron)

- Each input that is sent into the neuron must first be **weighted**
  - Multiplied by some value (often a number between -1 and 1)



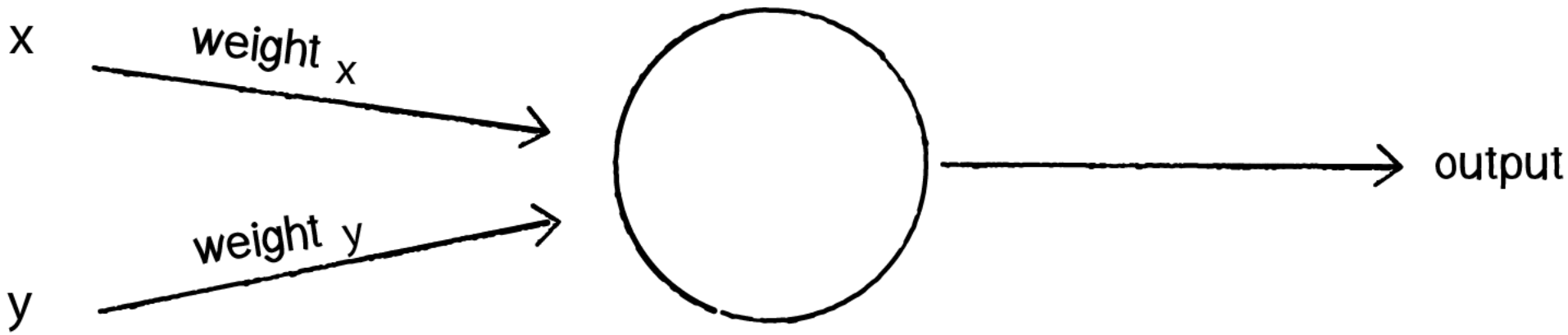
Neural Networks (Perceptron)

- When creating a perceptron, we will typically begin by assigning random weights
  - **Weight 0: 0.5**
  - **Weight 1: -1**



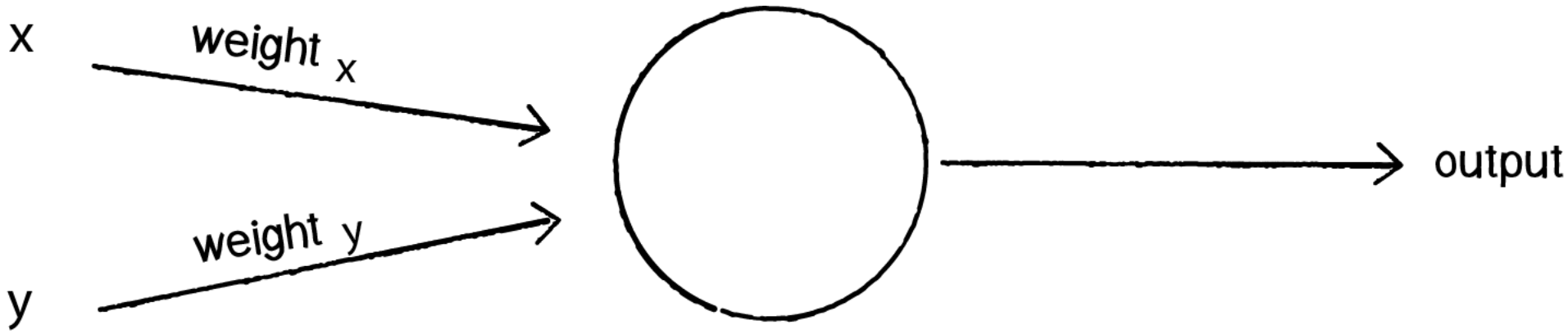
Neural Networks (Perceptron)

- We take each input and multiply it by its weight
  - **Input 0 * Weight 0: 0.5 => 12 * 0.5 = 6**
  - **Input 1 * Weight 1: -1 => 4 * -1 = -4**



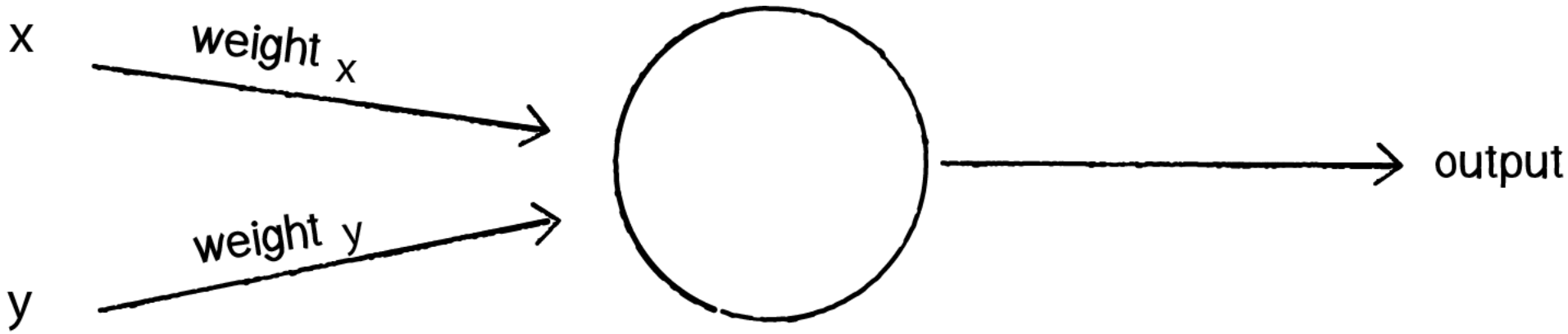
Neural Networks (Perceptron)

- The outputs of a perceptron is generated by passing that sum through an activation function
- In the case of a simple binary output, the activity function is what tells the perceptron whether to "fire" or not



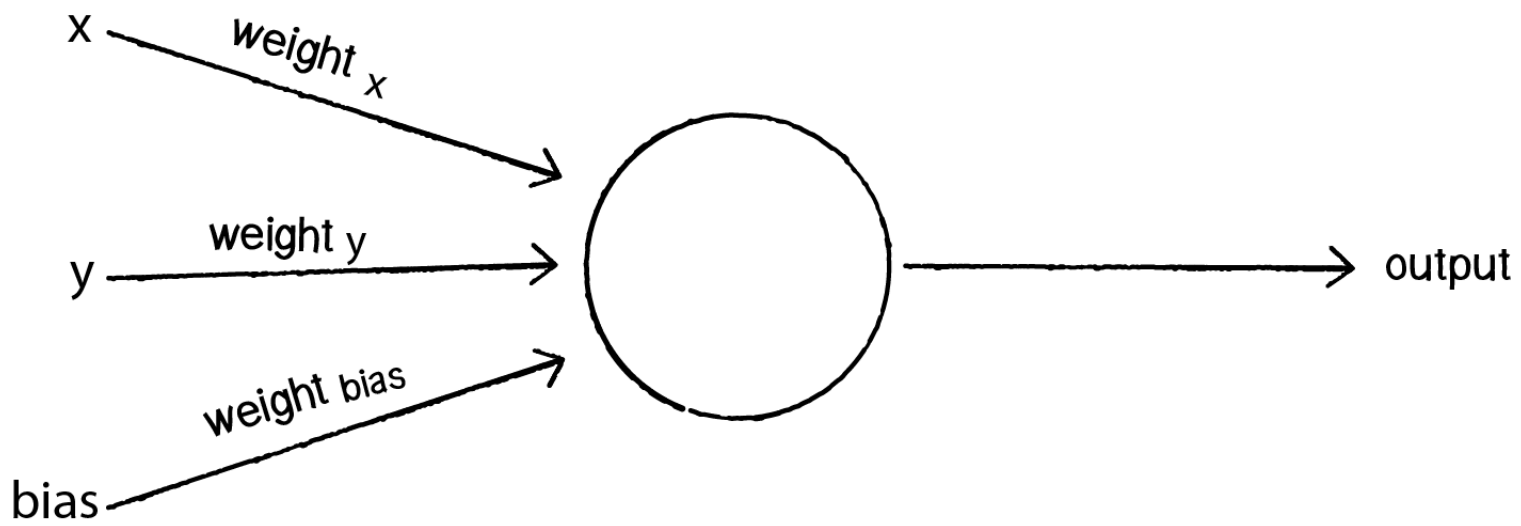
Neural Networks (Perceptron)

- Many activation functions to choose from (Logistic, Trigonometric, Step, etc.)
- Let's make the activation function the sign of the sum
- In other words, if the sum gives a positive number
  - the outputs is 1;
  - if it is negative, the outputs is -1



Neural Networks (Perceptron)

- However, one more thing to consider is **Bias**
- If both inputs were equal to zero, then any sum no matter what multiplicative weight would also be zero (no generated output)
- To avoid this problems
  - Add third input known as a **bias** input with a value of 1 (this avoids the zero issues)



Neural Networks (Perceptron)

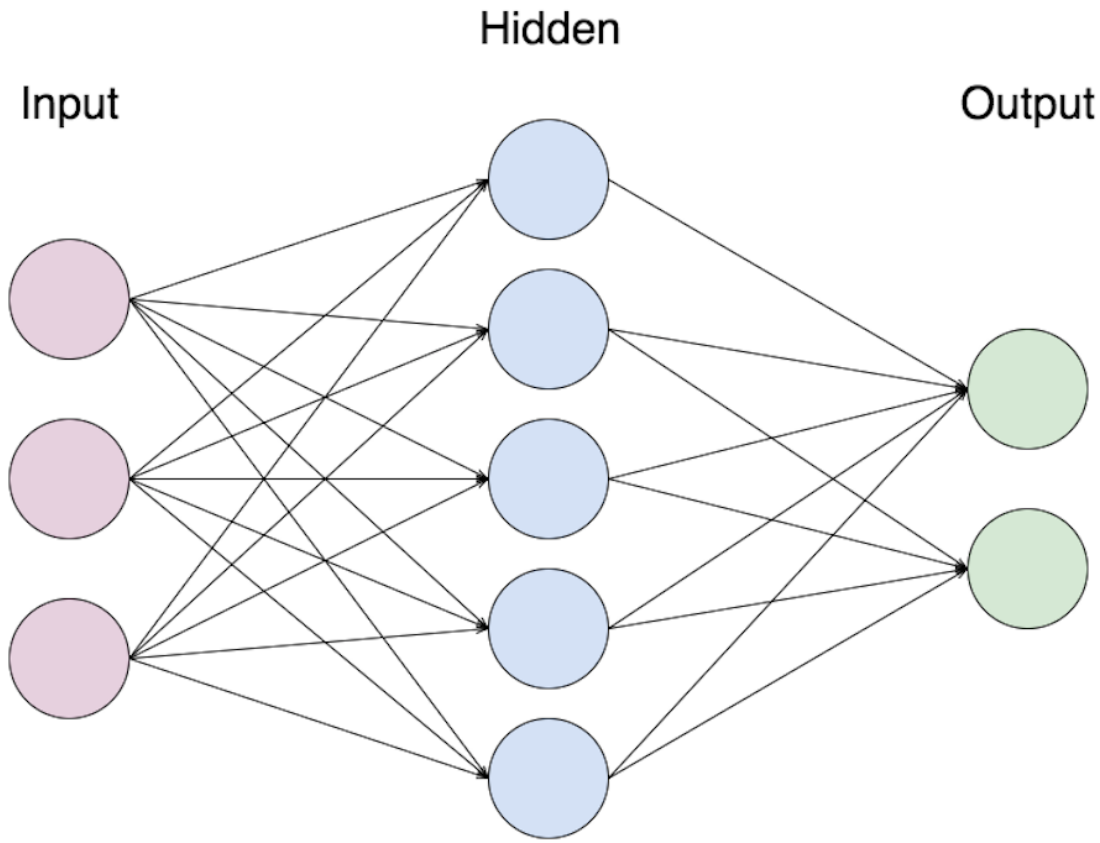
- To train the perceptron we use the following steps
  - Provide the perceptron with inputs for which there is a known answer
  - Ask the perceptron to guess an answer
  - Compute the error (How far off from the correct answer?)
  - Adjust all the weights according to the error
  - Return to step 1 and repeat

Neural Networks (Perceptron)

- We repeat this until we reach an error we are satisfied (we set this before hand)
- That is how a single perceptron would work, now to create neural network all you have to do is linking many perceptrons together in layers

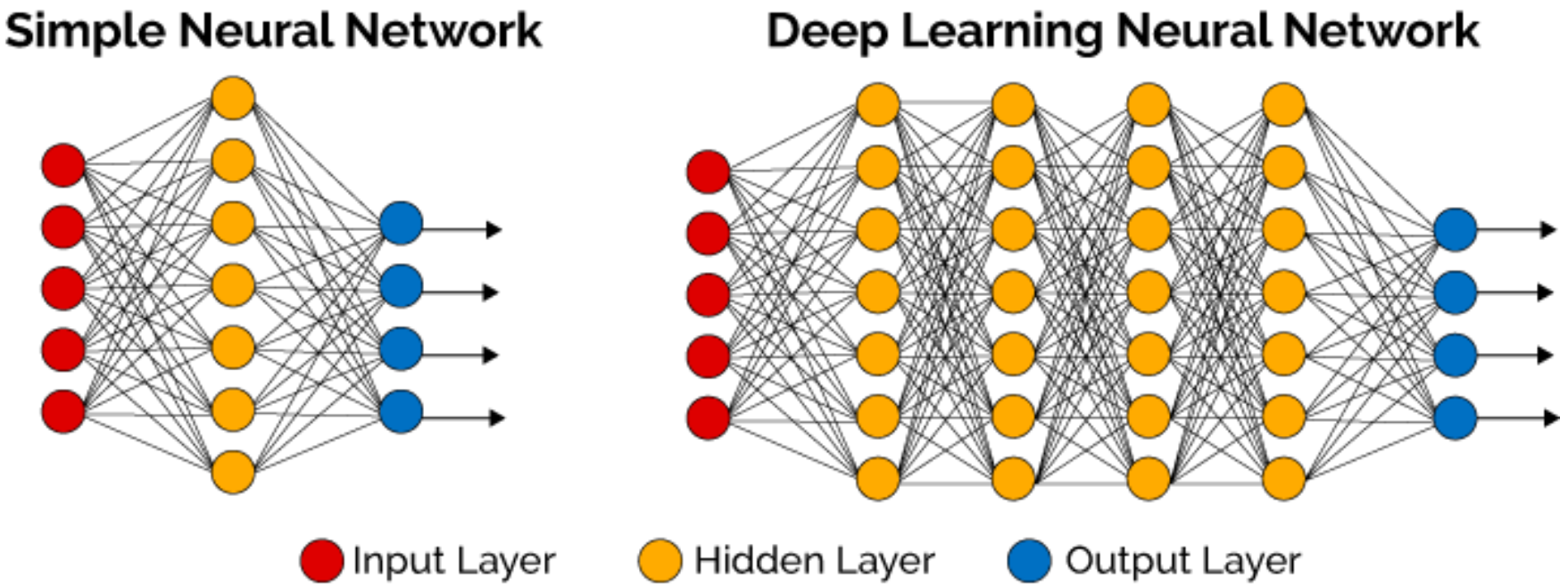
Neural Networks (Perceptrons)

- You will have an **input layer and an output layer**
- Any layers in between are known as **hidden layers**
  - Because you do not directly "see" anything but the input or output



Neural Networks (Deep Learning)

- You may have heard of the term "Deep Learning"
- That is just a **Neural Network with many hidden layers**, causing it to be "deep"
- For example, Microsoft's state of the art vision recognition uses 152 layers



AiAT ACADEMY

TensorFlow

TensorFow (Overview)

- TensorFlow is an open source software library developed by Google (https://www.tensorflow.org)
- Is has quickly become the most popular Deep Learning library in the field
- It can run on either CPU or GPU
  - Typically, Deep Neural Networks run much faster on GPU

TensorFlow

TensorFow (Overview)

- The basic idea of TensorFlow is to be able to create data flow graphs
- These graphs have nodes and edges, just as we saw in the lecture for Neural Networks
- The arrays (data) passed along from layer of nodes to layer to layer of nodes is known as a Tensor

AiAT ACADEMY

TensorFlow Basics

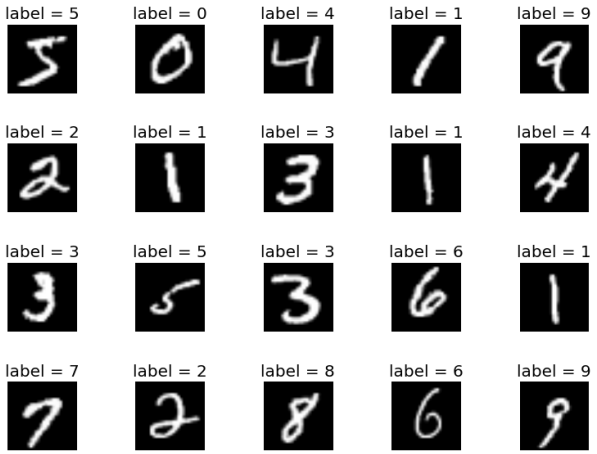
Colab

AiAT ACADEMY

MINIST with Multi-Layer Perceptron

TensorFow (MNIST)

- The MNIST data set is a collection of arrays representing **hand writing digit (0-9)** using pixels
- Let's Explore how we can use TensorFlow to help classify what number is written simply by training on the array values



MINIST with Multi-Layer Perceptron ***Colab***

AiAT ACADEMY

TensorFlow Estimators

TensorFow (Estimators)

- TensorFlow has an Estimator object you can use to quickly create models without needing to manually define the Graph

TensorFow (Estimators)

- Estimator Steps
  - Read in data (normalize if necessary)
  - Train/Test split the data
  - Create Estimator Feature Columns
  - Create input Estimator Function
  - Train Estimator Model
  - Predict with new test input function

AiAT ACADEMY

TensorFlow Estimators
Colab