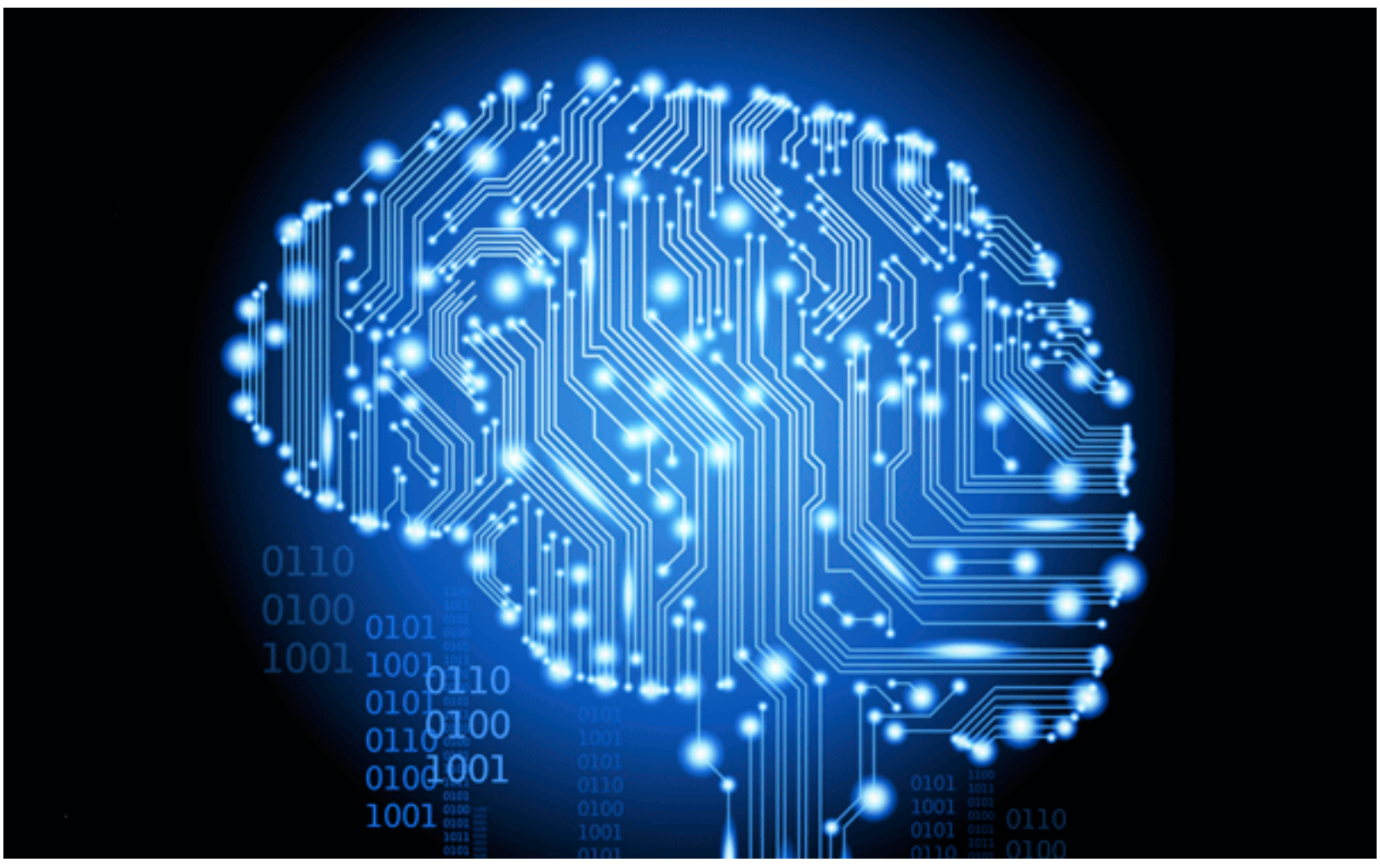
What is a Neural Network?

When people talk about artificial intelligence and machine learning, they most often refer to (artificial) neural networks (ANN or NN). Let's explore some machine learning basics, without excessive math, purely from a programmer's perspective.



A neural network is a computation model imaging the brain where individual nodes (neurons) form an organism (a network) to process information. In order to understand what this means and how this is special, let's start by looking at how a computer *normally* works.

Conventional Computing

Conventional computers use an algorithmic approach to solving a problem. Provided with a set of instructions (the program) they follow it step by step to reach the desired result.

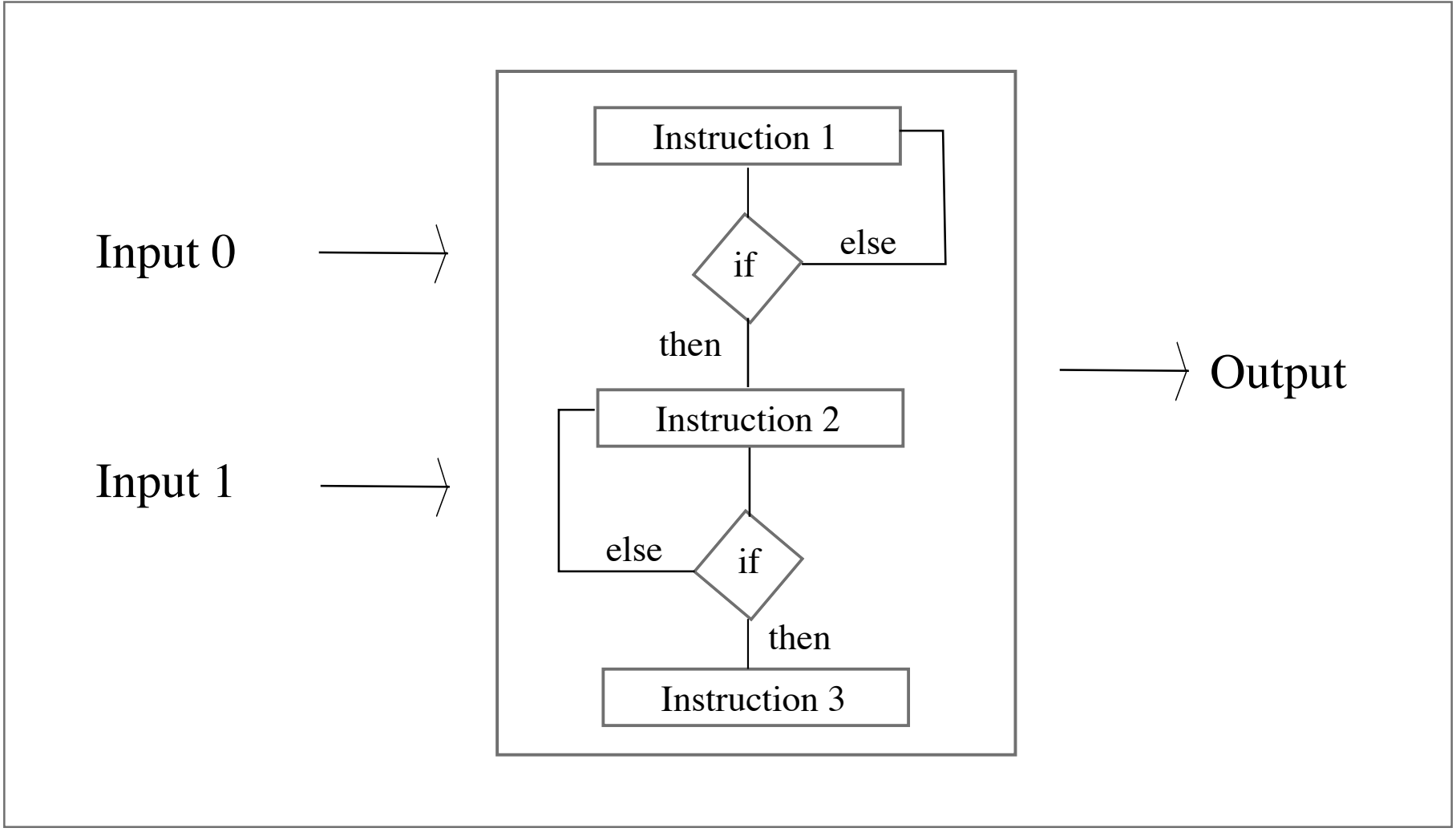


Figure 1: Conventional Computing

Therefore, the conventional computing approach can only solve what we already know. The software developer must first design all the required steps (instructions) that are required for solving the problem. Thus s/he must know already what the result will be.

Conventional computers/programs are **deterministic** (always lead to the same result) and **predictable** (the result is known in advance).

Neural Networks

A neural network represents a different computing approach to solving a problem. Instead of *calculating* the result, it's *guessing* it! *What!?* Yes, you read correctly, *guessing*. Or more scientifically, the network is *approximating* the solution. I.e. starting from a wild guess, using random numbers, the network is continuously revising its guessing method until its very close to the desired outcome. Here's how it works.

Instead of receiving an exact method how to calculate the output, the computer receives (lots of) reference data and starts *guessing*. Every time it guesses the output, we're letting it know whether its prediction (guess) was correct or not. If it was incorrect, we're telling it know by how much the prediction was off the desired result

(target), so the network can adjust accordingly and try a slightly different guess next time. This process is repeated a largue number of times. With every *guess* the networks's predicted result will move closer to the actual desired one.

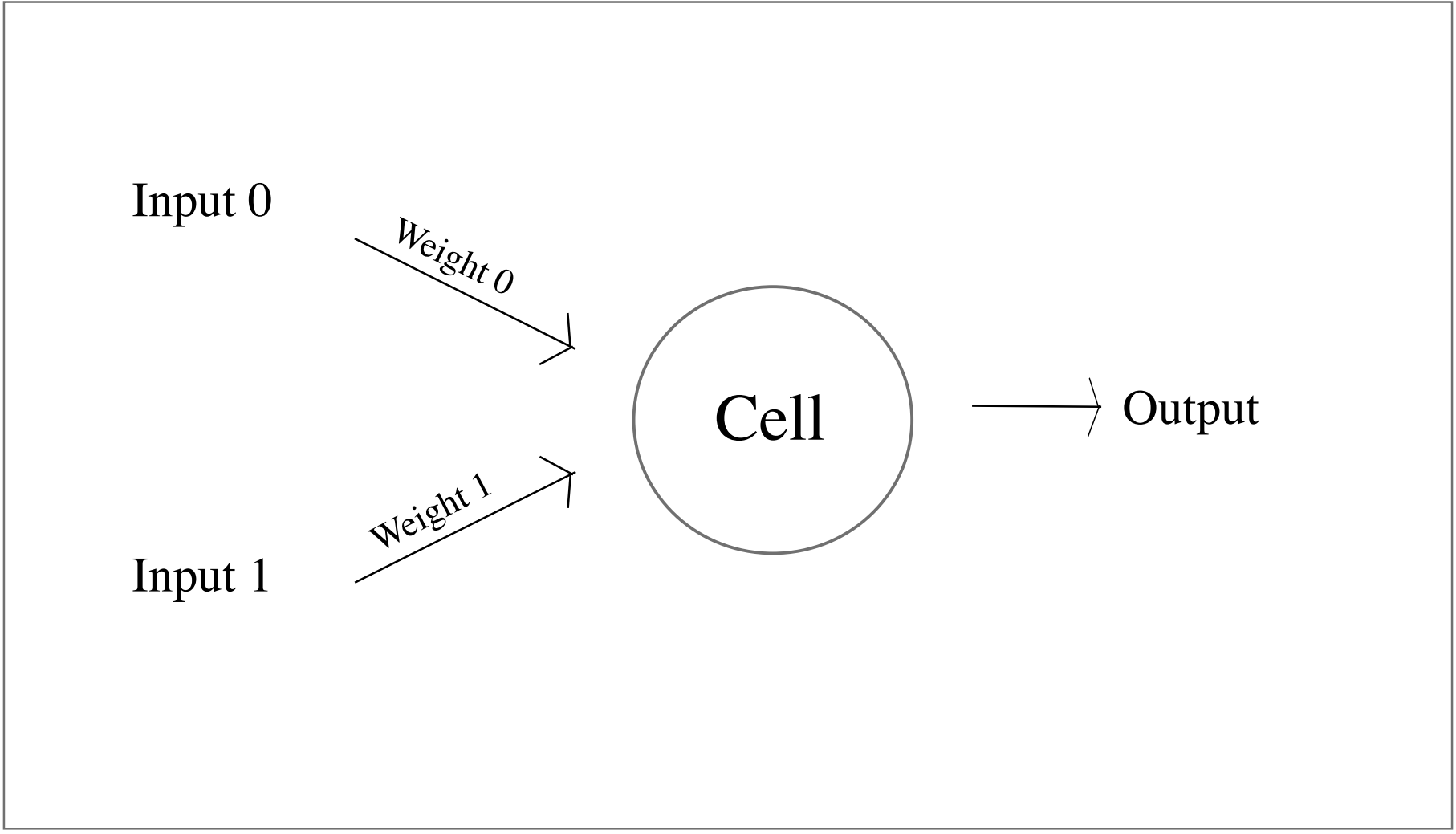
Please note that this is a **very simplistic** explanation and only describes one type of neural network and machine learning. But in principle this is how it works. You noticed that using this method we still need to know the correct answer (that's why this is called **supervised learning**), at least for a (large) number of sample cases. The network is trained, under our supervision, on known data. Eventually it figures out an underlying *rule* for how to get from input to output and will be able to predict the correct results even for unknown data and without receiving any feedback.

Of course, even after training any prediction is still, strictly speaking, *guessing* or *approximating* the solution. Therefore one cannot be 100% sure that its prediction is correct. There is always a certain error rate (those cases where the neural network *guessed* wrong) and machine learning experts use this measure to compare the performance of each other's networks.

So, now let's look how this works.

The Perceptron

I mentioned above that a neural network consists of individual nodes (in the brain these units are called *neurons*). In the computer and machine learning world we refer to such a node as a *perceptron*, a computational model of a neuron.



The *perceptron* consists of a core which is connected to at least two inputs an one output. It receives inputs, does some processing and spits out an output.

Processing? Let's look at what's going on inside the perceptron. It's very simple.

The input connections of the perceptron are assigned different weights. These weights are values between 0 to 1 and can be thought of as an expression of *priority* or *importance* of the respective input. In the beginning, i.e. for our first *guess*, we use random numbers for these weights.

The perceptron's output is simply the sum of the weighted inputs.

```
output = (input0 * weight0) + (input1 * weight1)
```

This perceptron's `output` is the neural network's *guess* for the desired result. In the scenario of *supervised learning*, as described above, this output is then compared to the desired output or `target` and the difference between both (the `error`) is calculated.

```
error = target - output
```

Now the network not only knows that its *guess* was wrong but also by how much (`error`). We now want the network to try again, but of course not using the same but new values. Since the `input` to the system is a given and fix, the only values we can change are the `weights` of these inputs. And that's exactly what we do.

```
weight0 += input0 * error * learning_rate
weight1 += input1 * error * learning_rate
```

The `learning_rate` defines the interval for changing the weights, i.e. a high rate mean faster but likely less accurate change, a low rate mean more accurate but slower change.

Above 3 steps form the basic algorithm of the most simple neural network one can think of. There is no hidden layer, no sigmoid function, no back propagation, or any other mechanism neural networks usually have. Yet, it still works.

To put this to a test, let's start coding. For the versed programmer, a few lines of code make things clearer than pages of detailed verbal explanations.

In my next post we'll apply the above algorithm to the problem of automatic image recognition of handwritten digits. But before you move on, I strongly suggest you take a look at below readings which explain the above in much more length and depth.

Further Readings

I found below sources extremely helpful when writing this post and strongly recommend these readings to get a much more in-depth understanding of what neural networks are and how they work.

- Neural Networks, by Christos Stergiou and Dimitrios Siganos, 1996
- Neural Networks in Plain English, by Mat Buckland, 2002
- Nature of Code: 10 Neural Networks, by Daniel Shiffman, 2014

After you've fully digested these you're ready to move on.

Written on July 9, 2015