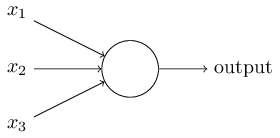
important types of artificial neuron

(the perceptron and the sigmoid neuron)

the standard learning algorithm for neural networks, known as stochastic gradient descent.

Perceptrons

So how do perceptrons work? A perceptron takes several binary inputs, x1,x2,…x1,x2,…, and produces a single binary output:



In the example shown the perceptron has three inputs, x1,x2,x3x1,x2,x3. In general it could have more or fewer inputs. Rosenblatt proposed a simple rule to compute the output. He introduced *weights*, w1,w2,…w1,w2,…, real numbers expressing the importance of the respective inputs to the output. The neuron's output, 00 or 11, is determined by whether the weighted sum ∑jwjxj∑jwjxj is less than or greater than some *threshold value*. Just like the weights, the threshold is a real number which is a parameter of the neuron. To put it in more precise algebraic terms:

Output = {0 if ∑jwjxj≤ threshold

{1 if  ∑jwjxj> threshold

That's all there is to how a perceptron works!

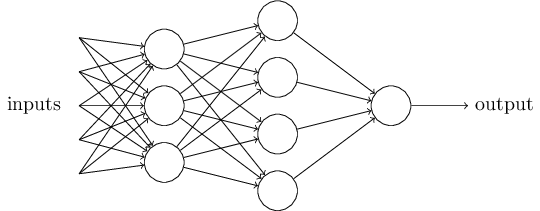
That's the basic mathematical model. A way you can think about the perceptron is that it's a device that makes decisions by weighing up evidence. Let me give an example. It's not a very realistic example, but it's easy to understand, and we'll soon get to more realistic examples. Suppose the weekend is coming up, and you've heard that there's going to be a cheese festival in your city. You like cheese, and are trying to decide whether or not to go to the festival. You might make your decision by weighing up three factors:

1. Is the weather good?
2. Does your boyfriend or girlfriend want to accompany you?
3. Is the festival near public transit? (You don't own a car).

We can represent these three factors by corresponding binary variables x1,x2x1,x2, and x3x3. For instance, we'd have x1=1x1=1 if the weather is good, and x1=0x1=0 if the weather is bad. Similarly, x2=1x2=1if your boyfriend or girlfriend wants to go, and x2=0x2=0 if not. And similarly again for x3x3 and public transit.

Now, suppose you absolutely adore cheese, so much so that you're happy to go to the festival even if your boyfriend or girlfriend is uninterested and the festival is hard to get to. But perhaps you really loathe bad weather, and there's no way you'd go to the festival if the weather is bad. You can use perceptrons to model this kind of decision-making. One way to do this is to choose a weight w1=6w1=6for the weather, and w2=2w2=2 and w3=2w3=2 for the other conditions. The larger value of w1w1 indicates that the weather matters a lot to you, much more than whether your boyfriend or girlfriend joins you, or the nearness of public transit. Finally, suppose you choose a threshold of 55 for the perceptron. With these choices, the perceptron implements the desired decision-making model, outputting 11 whenever the weather is good, and 00 whenever the weather is bad. It makes no difference to the output whether your boyfriend or girlfriend wants to go, or whether public transit is nearby.

By varying the weights and the threshold, we can get different models of decision-making. For example, suppose we instead chose a threshold of 33. Then the perceptron would decide that you should go to the festival whenever the weather was good or when both the festival was near public transit and your boyfriend or girlfriend was willing to join you. In other words, it'd be a different model of decision-making. Dropping the threshold means you're more willing to go to the festival.



In this network, the first column of perceptrons - what we'll call the first layer of perceptrons - is making three very simple decisions, by weighing the input evidence. What about the perceptrons in the second layer? Each of those perceptrons is making a decision by weighing up the results from the first layer of decision-making. In this way a perceptron in the second layer can make a decision at a more complex and more abstract level than perceptrons in the first layer. And even more complex decisions can be made by the perceptron in the third layer. In this way, a many-layer network of perceptrons can engage in sophisticated decision making.

Incidentally, when I defined perceptrons I said that a perceptron has just a single output. In the network above the perceptrons look like they have multiple outputs. In fact, they're still single output. The multiple output arrows are merely a useful way of indicating that the output from a perceptron is being used as the input to several other perceptrons. It's less unwieldy than drawing a single output line which then splits.

Let's simplify the way we describe perceptrons. The condition ∑jwjxj>threshold is cumbersome, and we can make two notational changes to simplify it. The first change is to write ∑jwjxj  as a dot product, w⋅x≡∑jwjxj , where w and x are vectors whose components are the weights and inputs, respectively. The second change is to move the threshold to the other side of the inequality, and to replace it by what's known as the perceptron's bias, b≡−threshold. Using the bias instead of the threshold, the perceptron rule can be rewritten: