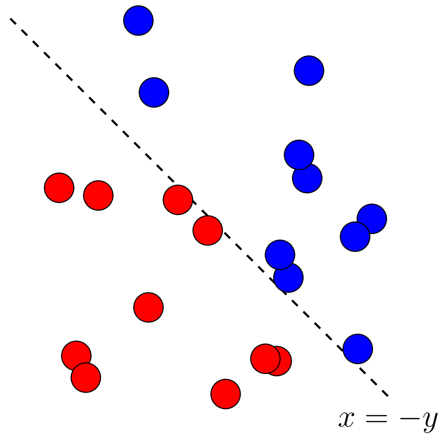
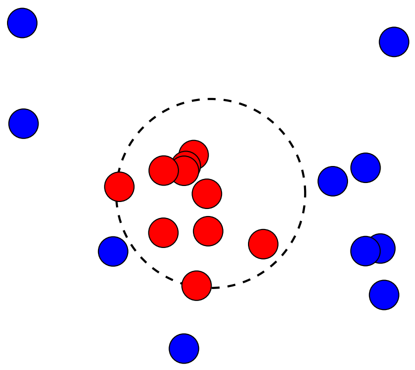
A linear classifier.

In the last section we looked at linear classifiers; this was a reinterpretation of regression as a classifier, effectively using a line, in two-dimensions, or a hyper-plane in more dimensions, to divide the space into two halves, one hopefully mostly ducks, the other mostly rabbits, or whatever it is you are trying to classify. The big problem with this is that it assumes the division is created by a straight line or flat plane or whatever; the linear in linear classifier is a big restriction on the manner you can seperate the data, it can deal with this:



but it can't deal with this:



We will have a look, in a broad way, at the solution to this problem. In short we will use a larger class of seperating lines defined by a neural network; the 'neural network' will be a way of making lots of different potential seperators.

We will see that these neural networks will prove to be a surprisingly good way of doing this; in fact, for a long people didn't expect them to be as good as they are and spent a lot of time thinking of different approaches to this problem that made sense to them mathematical when, as well all now know, the thing that works is of just 'use a big enough neural network'.

However, phrasing neural networks as a surprisingly good way to learn seperation boundaries is a sort of 'modern' way of look the problem, it was actually discovered using a more roundabout analogy with neuroscience and so we will take a short segway though that way of framing neural networks, the 'neural' approach as it were. It is useful to look at this since neural networks are the obvious, mathematical apparent, solution to this problem and, historically, our motivation for studying them came from thinking about the brain!

To see how this works begins by rephrasing the linear classifier as a *perceptron*.

Modelling the brain

The McCulloch-Pitts neuron

The brain is complex in different ways; for example there are lots of different neuron types, pyramidal neurons in the cerebellum are extremely large, have a huge number of inputs and are unusual in producing two different types of output, the granualar cells that signal to them are tiny, extemely numerous and have only a handful of inputs. Signalling in the brain is largely electrical, but there are also chemical signals known as neuromodulation. The list of elaborations and subtle aspects goes on and on. One of the most striking aspect is the basic network complexity, there are a lot of neurons and neurons tend to be connected to a huge number of other neurons; the neurons themselves tend to be more stereotypical, that is more easily grouped into just a few functional types, in the parts of the brain most typical of animals capable of complex inference, the cortex in mammals for example.

A small bit of neuroscience terminology: the neuron has three main parts: soma, dendrites and the axon. The *soma* is sort of the middle part of the cell, it is the cell body where most of the metabolic processes happen; the soma has two sorts of tubes extending from it: the *dendrites* which carry signals into the soma and the *axon* that carries signals away so that they can

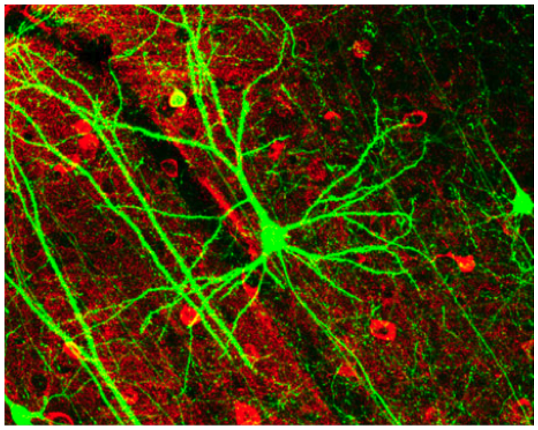


Figure 1: A picture of a neuron from Wikipedia. This is a pyramidal neuron, one of the main neuron types in the cortex, clever experimental techniques have been used to make the neuron glow green, otherwise they are hard to pick out because the brain is just so full of stuff! The soma is the green triangle in the middle, most of the lines are dendrites, the axon is the very thin line sloping down from the base of the triangle.

be transmitted to other neurons. The joining point is the *synapse*, it allows the signal arriving along the axon of one neuron, the *pre-synaptic neuron* to be converted into a change in voltage in the dendrite of the *post-synaptic neuron*. The synapse is obviously important; whereas the electrical signal, the *spike* or *action potential* that travels down the axon is in a way simple, it is a spike in voltage that does not seem to carry information in its amplitude or shape, the synapse is complex with an elaborate biomechanical machinery to allow it to change its strength, the degree to which a pre-synaptic spike changes the post synaptic voltage.

The McCulloch Pitts neuron model, or Threshold Logic Unit, was introduced in 1943 by Warren McCulloch and Walter Pitts[1] as a computational model of a neuronal network [1]. This model focusses on the network complexity of the brain, the neurons in the model are very simple allowing interesting behaviour to emerge through the network. In the model neurons are joined to each other with connections of variable strength, reflecting the abil-

[1]Walter Pitts was an interesting and odd man, a genius in the old-fashioned self-destructive and brilliant sense.

ity of synapses, in the brain to change strength. The neurons are almost like adding machines: in the soma the inputs from other neurons are added up and they determine the activity of the neuron in a non-linear way; they also knew that neurons tend to ignore input up to some threshold value before responding strongly. These properties they tried to include in their model neurons.

Artificial neurons, of the sort used in artificial intelligence, are described by a single dynamical variable, x_i say for a neuron labelled i; the value of x_i is determined by the weighted input from the other neurons:

$$x_i = \phi\left(\sum_j w_{ij}x_j - \theta_i\right) \tag{1}$$

ϕ is an activation function, θ is a threshold and the w_{ij} are the connection strengths weighting the inputs from the other neurons; the analogue of the synapse strengths. The McCulloch Pitts neuron was the first example of an artificial neuron and had a step function for ϕ:

$$x_i = \begin{cases} 1 & \sum_j w_{ij}x_j > \theta_i \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

This step function is often called the *Heaviside* function.

Thus, the neuron has two states, it is in the on state, $x_i = 1$ if the weighted input exceeds a threshold θ_i and an off state, $x_i = -1$ if it doesn't; the picture you might have of how this corresponds to the brain is that 'on' corresponds to rapid spiking and 'off' to spiking at a much lower rate. For McCulloch and Pitts this was the simplest way they could model the properties that neurons had been observed to have of being silent below some threshold. Now neurons above their threshold do increase their activity as the input increases and this is one of the many ways that the McCulloch-Pitts neuron was a simplification. It was useful at the time because McCulloch and Pitts wanted to model these neurons using simple electrical circuits at a time when computing was in its infancy. They hoped, in fact, that these threshold units, as they called them, would form the basis of computer architectures, as we know this didn't happen, instead computers were built out of small logical units. These days the simple on-off-ness of these units is inconvenient and different activation functions are used, such as ReLU, we will return to this.

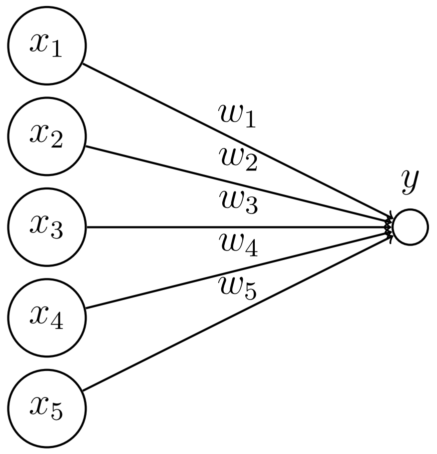
The w_{ij}, the connection strengths. For those of you who are interested these are like the synapse strengths, a positive w_{ij} is an excitatory synapse and negative, an inhibitory; a given neurons have both negative and positive out-going synapses, that is there is no restriction that says w_{ij} always has the same sign for a given j, this is different from real neurons where all the outgoing synapses from a given neuron are either excitatory or inhibitory.

While it should be clear that this network has some of the properties, very abstracted, of a neuronal network, it might not be so clear what can be done with the neurons. When they were working, at the dawn of the age of electronic computers, McCulloch and Pitts believed that their neurons might form the natural unit in computer circuits. In other words, they thought they might perform the role actually played by logical circuits. In fact, it is still not clear if the artificial neuron is or isn't the natural unit of computation since they are a component in, for example, deep learning networks. In fact, there are two major applications of McCulloch-Pitts neurons: the perceptron and the Hopfield network. These two applications add a rule for changing the connection strength to the original McCulloch-Pitts neuron. We won't discuss the Hopfield networks here but they are worth learning about!

Perceptrons

The perceptron is a machine that does supervised learning, that is, it makes guesses, is told whether or not its guess is correct, adjusts itself so that it would've been more correct and then makes another guess. They were first discovered in 1957 by Frank Rosenblatt [2] and introduced to the world with great fanfare, it was claimed that they would solve problems from object recognition to consciousness: if you consider the perceptron as the forbearer of the deep learning network, then perhaps we don't know if these claims will be fulfilled, but we do know that the original perceptron proved quite limited in artificial intelligence. It does, however, appear to describe some neuronal processes, if we ignore the implementation details.

Anyway, a perceptron is made of two layers of neurons, an input layer and an output layer of McCulloch-Pitts neurons. For simplicity let's assume the output layer has a single neuron.



Now, if the input is given by $\mathbf{x} = (x_1, x_2, \dots, x_n)$ the output, y, is

$$y = \phi(r) \tag{3}$$

where

$$r = \sum_j w_j x_j - \theta \tag{4}$$

and ϕ here is the simple Heaviside-like activation function. Now for a given input, if the actual value of the output should be d the error is $d-y$. The perceptron learning rule is to change the w_j weight by an amount proportional to the error and how much x_j was 'to blame' for the error:

$$\delta w_j = \eta(d-y)x_j \tag{5}$$

and

$$\delta\theta = \eta(d-y) \tag{6}$$

where η is some small learning rate and

$$\begin{array}{rcl} w_j & \to & w_j + \delta w_j \\ \theta & \to & \theta + \delta\theta \end{array} \tag{7}$$

You can see how this might work, if x_j was positive and y was too big, this would make w_j smaller so in future y would be smaller when it had the same input.

In fact, the perceptron can only solve problems with a linear classifier: if we think of the x_i's as parametrizing an n-dimensional space then $\sum_i w_i x_i =$

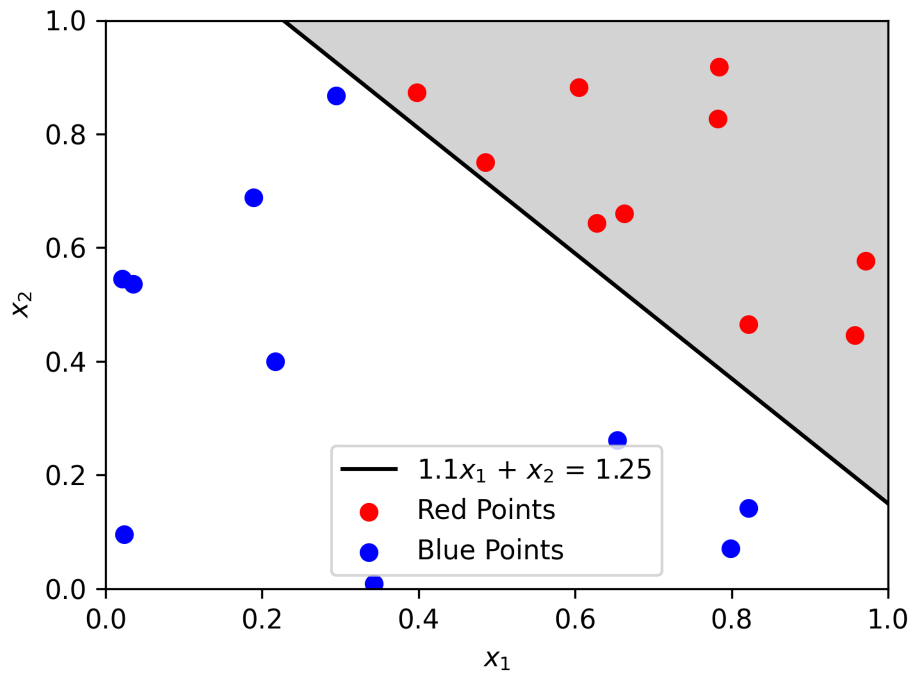


Figure 2: Here the shaded region corresponds to $1.1x_1 + x_2 > 0.25$, so if $w_1 = 1.1$ and $w_2 = 1$ with $\theta = 0.25$ then the corresponding perceptron neuron will be one for all the circle points and -1 for all the square points.

θ is a hyperplane in that space, so a pattern $\mathbf{x}$ is classified one way or the other according to which side of that hyperplane it lies, see for example Fig. 2. Thus, the perceptron works only if there is a hyperplane dividing the data into two, with one class of data on one side and one on the other. In fact, if the data is linearly separable the perceptron is guaranteed to converge to a solution which manages this separation. Now, as illustrated in Fig. 3, there will not be a unique hyper-plane separating the two classes and the perceptron won't, typically, find what you might regard as the 'best' hyperplane, where by best you might mean the line that is, in some sense, as far from the individual data points as possible; finding that best hyperplane is the idea behind the support vector machine.

The perceptron learning rule can be motivated by thinking about error

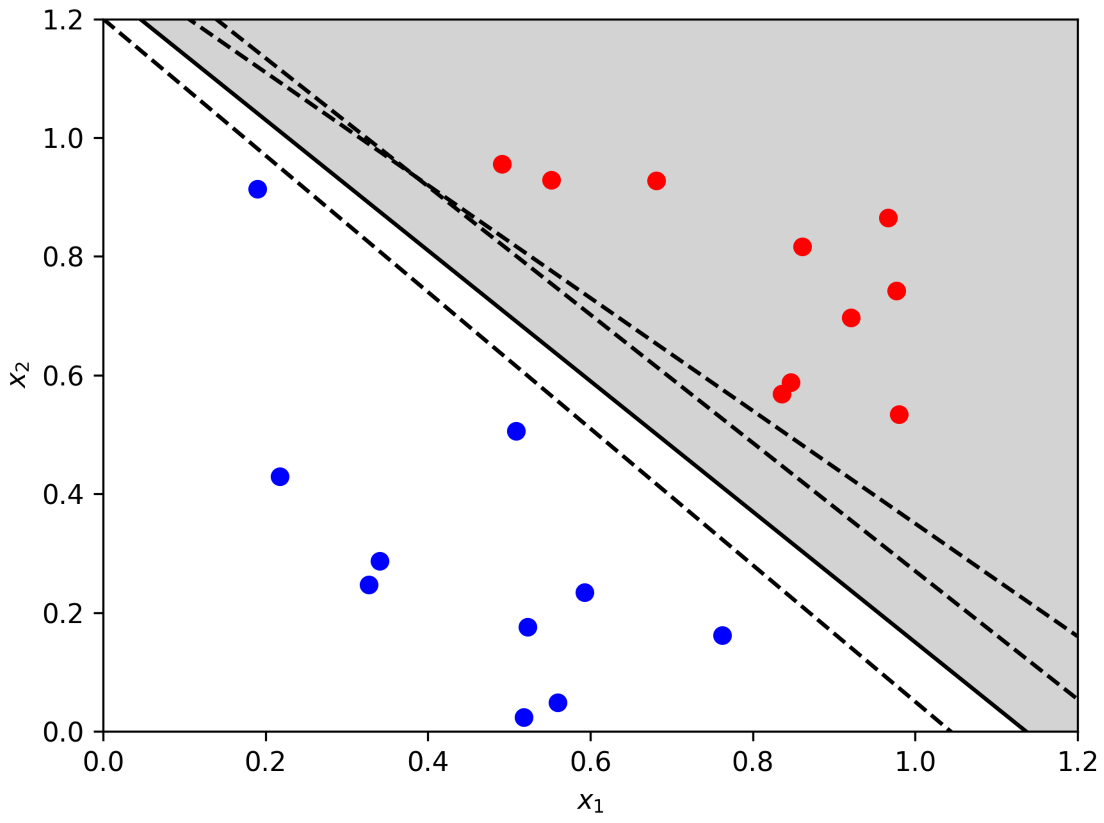


Figure 3: All of these lines separate the points into two classes

minimization. Consider an error function

$$E = \langle (d^a - y^a)^2 \rangle_a \tag{8}$$

If a labels lots of input, $\mathbf{x}^a$, desired output d^a pairs, with y^a the actual output; the angle brackets are an average over these pairs. Now consider the gradient of E with w_i:

$$\frac{\partial E}{\partial w_i} = -2 \left\langle (d^a - y_a) \frac{\partial y^a}{\partial w_i} \right\rangle \tag{9}$$

Now if we ignore for the moment the fact that the activation function for the McCulloch-Pitt neuron isn't differentiable and write

$$y^= \phi(r) \tag{10}$$

with

$$r = \sum_i w_i x_i - \theta \tag{11}$$

we have

$$\frac{dy^a}{dw_i} = \frac{d\phi}{dr} \frac{\partial r}{\partial w_i} \tag{12}$$

so using

$$\frac{\partial r}{\partial w_i} = x_i \tag{13}$$

we get

$$\frac{\partial E}{\partial w_i} = -2 \left\langle (d^a - y^a) x_i^a (\text{stuff involving the derivative of } \phi) \right\rangle \tag{14}$$

Of course, in the McCulloch-Pitts case the 'stuff involving the derivative of ϕ' is either zero or undefined, but we can that this gives a similar learning rule: one way to reduce the error is to move a tiny amount in the opposite direction to the gradient, the gradient is the direction along which the value increases quickest. The actual perceptron rule we gave updates a small bit after every presentation rather than averaging first over a collection of presentations, both approaches make sense. Here we have dealt with the weights, the same approach can be applied to the threshold θ.

This is the way to a more modern approach to the perceptron rule and these days smooth, or mostly-smooth activation functions are used in artificial neurons for this reason. The basic limitation of the perceptron is that it has only one layer and so only learns a linear classification; these days artificial neural networks have more than one layer; this complicates the idea of adjusting the w_i in a way that is weighted by x_i, this, in a sense, changing the weight according to how much it is 'to blame' for the error. Back propagation resolves this, it can be thought of as propagating the error backwards from layer to layer; although another way to think of it is as doing gradient descent on all the weights. At the moment the back propagation algorithm is not considered very biological, though it is very possible that when we understand why deep learning networks are so effective it will be clear that the important aspects of the algorithm can be recognized in neuronal dynamics.

References

[1] McCulloch, W and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5:115–133.

[2] Rosenblatt, F. (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory. Psychological Review, 65:386–408.