BRAIN and TRAIN

We are working our way towards an understanding of artificial neural networks and how they can be trained using the technique of gradient descent (a calculus tool) and so-called backpropagation (which speeds up a key part of the computation).

Since the human brain is the only intelligent system that we are aware of – and since it is the motivating example for artificial neural networks – it makes some sense to consider what is known about it. Alas, while we know a truly vast amount, yet it is far far too little. What is known can (and does) fill many very thick books; and what is unknown is essentially all the most interesting and important things, such as how the brain manages to make us who we are (entities with minds that can have experiences, feel, think, plan, remember, learn, care, decide, etc). Later on we may take a quick look at some philosophical aspects of mind. Here we content ourselves with a very quick overview of some so-called functional neuroanatomy.

A weighted directed graph (WDG) is a triple <V,E,f> where V is a nonempty set, E is a subset of VxV, and f is a function from E to the real numbers. One way to think about the brain is as an enormous WDG. How enormous?

Among the cells in the brain are those called neurons, which are communication cells that can signal one another via connections (synapses). Moreover, each synapse has chemical characteristics that affect the sort of influence a signal coming from one cell has on the receiving cell. The neurons and the synapses between them thus form an enormously large directed and weighted graph. How large? Well, there are roughly 10^11 (best current count is 86 billion) neurons (these are the vertices) in the human brain, and on the order of 10^14 synapses (edges). That is, |V| ~~ 10^11 and |E| ~~ 10^14. That's big!!!

[By comparison, the World Wide Web apparently has (in April 2016) between 5 and 50 billion pages. So, similar to one brain – in sheer vertex-count. I did not manage to find data on the number of active links, but an old rule of thumb is to keep it to less than 100 per page. If we take 100 as a typical number (it is probably far less) then the brain wins just in sheer numbers (a typical neuron has roughly 5000 synapses), let alone the fact that it is a self-sustaining processor rather than a passive engine for other agents to use. Of course, that is a bit unfair – there are processes running on the web all on their own, once started by humans. So, maybe – just maybe – the web is almost comparable to a single human brain, at least by some crude measures.]

Aside from it's incredible size, why is such a graph as the brain interesting? (i) it's our brain, and supposedly the basis for who we are; (ii) it changes over time: old neurons die, new ones grow, new connections form, weights change with experience; (iii) it is an active structure, sending information around inside itself,

rather than a more typical graph that is simply a passive data storage that a separate algorithm acts upon; (iv) it is massively parallel: activity goes on in billions of neurons at the same time, unlike a typical graph algorithm that travels along only one edge (between two vertices) at a time.

Thinking about the brain this way has stimulated research in computer science, to pursue computational models of massive parallel processing. There are two rather distinct areas that have come about: (i) artificial neural networks, and (ii) abstract parallel processing machines. The former (ANNs) are more closely patterned on actual neuronal behaviors, and often are used as models of the brain, as well as in artificial intelligence, with a special focus on learning or training based on experience. The latter provide designs for parallel computer architectures aimed at achieving much greater processing speed than conventional computers have. We will examine ANNs in another lecture; in the current lecture, we will take a (brief) look at the brain.

The (human) brain – some major components:

Most of our neurons and synapses are contained in the ``neocortex'' (present in mammals); it wraps around a smaller and evolutionarily older part of the brain sometimes called the "smell brain", "crocodile brain", or "reptile brain".

A rough outline of the human brain seen from the left is given in Fig 1:

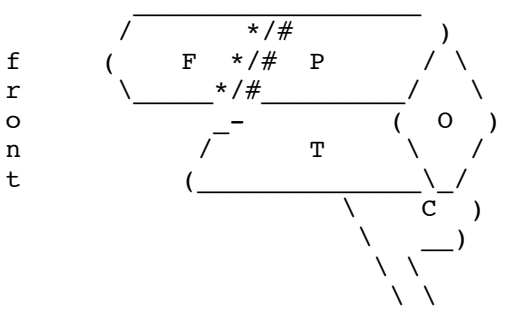


Fig 1

where F = frontal lobe, P = parietal lobe, T = temporal lobe, and O = occipital lobe. The lower tip of the smell brain is just visible below the temporal lobe, where it joins the spinal cord; also visible is a bit of the cerebellum, C. The four lobes shown form the neocortex; inside, at the top of the spinal cord (not shown) are the older parts of the brain (hidden behind the temporal lobe).

From the top:

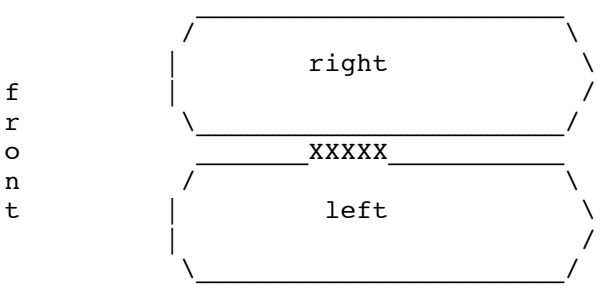


Fig 2

The top view shows that the brain (neocortex) comes in two very similar "hemispheres", left and right, each of which look like Fig 1 above when seen from the side. Thus there are two frontal lobes, two parietal lobes, etc. The two hemispheres are connected by a thick band of axon fibers XXXXX called the corpus callosum.

The strip *** along the back edge of F, and the strip ### along the front edge of P, are called the motor strip and the somatosensory strip, resp. The motor strip consists of neurons whose signals proceed down the spinal cord to muscles, causing them to contract; the frontal lobes (left and right) are thought to be involved in planning, and thus it seems not unreasonable that motor commands (enacting a plan) would emanate from there. The somatosensory strip receives sensory signals traveling from the body up through the spinal cord and smell brain and other inner structures (especially the thalamus) and from there to various places including the parietal lobe.

A striking feature is that of the "crossed" brain: all of the motor and sensory signals that travel to and from the brain via the spinal cord cross over to the opposite side. That is, the left half of the body (below the neck) is connected (both for sensory and for motor signals) to the right side of the brain, and vice versa.

The two eyeballs are situated just under the left and right frontal lobes. The "crossing" situation for vision is more complicated than for the motor and somatosensory strips just mentioned. Each eye (at the back, or retina) sends via its light-sensitive neurons (rods and cones) signals to either the right or left occipital lobe, depending not on which eye but rather on whether the light comes from the right or left of center of gaze. That is, half of the retina in each eye projects to the left occipital lobe, and half to the right occipital lobe. Specifically, light from the right of center falls on the left half of the retina in each eye, and then signals are sent from there to the left side of the brain, and similarly for the right. (However, the corpus callosum transmits this information from each side also to the other side.) Destruction of, say, the left occipital lobe, results in complete loss of visual awareness of the right visual field, and vice versa.

It is known that the occipital lobe performs only the "early" visual processing, such as edge orientation and color and motion, and that this information in turn is passed to other areas such as the temporal and parietal lobes and elsewhere, where a presumed (but ill-understood) process of integration occurs, affording high-level determination of what is seen (eg, a house or a face). The temporal lobe is also involved in memory and in processing of auditory information. The parietal lobe is thought to process highly abstract ideas such as mathematics.

On the other hand, recent evidence suggests that many parts of the brain are not particularly specialized and can step in to do whatever is needed at the moment. For instance, the occipital lobe can (and does) process Braille – not only in blind people but also in sighted people when their eyes are closed (and if they have taken the trouble to learn Braille); and yet when sighted people open their eyes the occipital lobe reverts to early-vision processing and the Braille processing goes on elsewhere.

This is truly amazing! And it (and much other evidence) calls into question what is sometimes called (unfairly) the Luria hypothesis: that the brain is an I/O device where sense data comes in and passes (mostly) through the thalamus and from there to auditory, visual, tactile and other specialized areas, then on to "association" areas (e.g., parietal lobe) and finally to the frontal lobe where actions are planned and initiated. While this is almost certainly *partly* true, it appears to be at best only that.

For one thing, once signals pass from the thalamus to the cortex (e.g., to the occipital lobe), signals are then sent from those cortical areas *back* to the thalamus, in so-called thalamo-cortical loops. It is not clear what is going on here – but indeed that can be said of a great deal of brain activity. One speculation is that such loops may be related to *deep-learning* that is the current hot item in AI (and that we will be examining shortly).

Overall, then, how the brain manages to do what it does is still in many respects uncertain. Even vision, the most heavily studied brain function, remains cloudy in at least one fundamental respect: at what point, and as part of what process, does visual awareness (actual subjective experience of seeing) occur? Not only is this not known, but no one has even managed to formulate a clear guess as to what it could be. The easiest notion to form (and then reject!) is that some sort of image is created in the brain, a bit like a TV set (but then "who" is looking at it?).

<u>What is a neuron</u>?

Now let us return to individual neurons. A neuron is a highly specialized cell, with the usual cell body containing a nucleus, as well as a long "axon" that acts a bit like a current-conducting wire. The axon usually splits into many "tips" that can come close to other neurons.

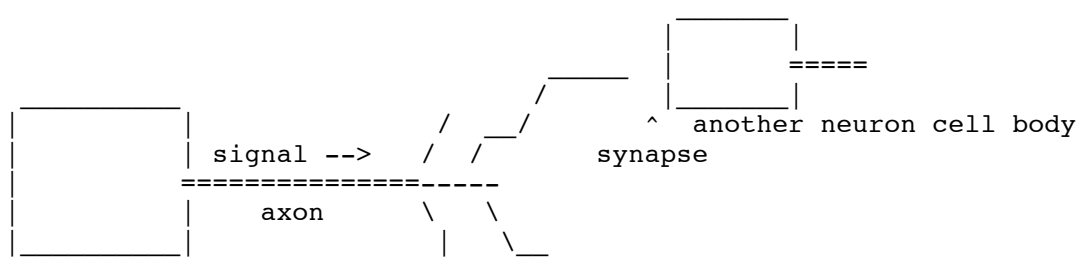


Fig 3

When the proper electro-chemical conditions occur in the cell body, an electrical current ("action potential") is initiated there, which travels from the cell body all the way along the axon to the axonal tips where it causes molecules known as neurotransmitters to be released. If there is another neuron close enough, some of the neurotransmitters will come into contact with it. If in addition to this close proximity there is a suitable chemical relationship between the neurotransmitters and the receiving site on the contacted neuron, that latter cell becomes either more likely to fire (excitatory synapse) or less likely (inhibitory synapse).

The cerebellum (C, in Fig 1) consists mostly of inhibitory synapses, which seems not unreasonable given that its apparent function is to provide fine motor control (as in piano-playing or reaching to pick up something); damage to the cerebellum leads to loss of this control so that motions tend to be exaggerated as in overreaching.

A number of Nobel prizes have been awarded to advances in neuroscience. One such was to Hodgkin and Huxley for their careful empirical and theoretical explanation of how – in terms of the mathematics of certain chemical reactions – a neuron's axon manages to transmit electrical impulses (the action potential). But neurons are tiny – how could they do that? Well, it turns out that certain species of squid have a so-called giant axon (not the giant squid, though), thick enough to see with the unaided eye (about 1mm, the thickness of a pencil-lead). Nature has been kind to neuroscientists now and then.

Another example is the stomatogastric ganglion (STG) in lobsters. This consists of a few dozen neurons tightly interconnected, that govern a wide variety of behaviors; even though the entire STG network has been fully mapped out neuron by neuron and synapse by synapse, its overall behavior is still not fully understood.

And this brings us to another complexity of studying the brain. It is more than just a graph. The brain also has neuromodulators – chemicals that when released can change the way synapses work. These often are also neurotransmitter chemicals,

except that when they circulate in the cerebrospinal fluid they can influence a large number of neurons. Some common types are: serotonin, dopamine, histamine, epinephrine (aka adrenalin), melatonin, tryptamine, and norepinephrine. This phenomenon appears to be behind the complex behavior-switching of STG, and of vastly more than that as well.

Efferent and Afferent Signals:

When a motor-neuron (i.e., one in the motor cortex of the frontal lobe) emits an action potential, it travels along the spinal cord to the appropriate muscle which then contracts. This is called an *efferent* signal – it goes to the body's effectors. And when sensory signals from the body (e.g., skin) are sent to the brain, this happens because of neurons in the skin and elsewhere, sending their action potentials back up the spinal cord to the brain; these are called *afferent* signals, as in *affect* (feeling). So, afferent signals tell us about things happening to us (pain, temperature, touch, etc).

But we need to know more than what is happening to us. We need to know about the larger world (and for instance vision can tell us much), and we also need to know what we are doing. This latter may seem odd: of course we know what we are doing – we are doing it, after all! (Psychologists call this – and related processes – *working memory*.) And yet how would the brain "know" that it is sending a signal to lift your arm, or to say "hello"? After all, your car's pistons do not know that they are sending kinetic energy to the wheels; a car has no KB!

If the brain has a KB (and it presumably does, but we don't really know anything significant about it), then somehow, when an efferent signal is sent to our muscles, a copy must be kept in the brain, so we can know we are saying hello or lifting our arm. And to some extent this is understood: when an efferent signal is sent, a so-called *efference copy* is produced and kept in the brain.

This was first hypothesized by Helmholtz around 1850 or so, to explain how we can see objects as stable when we turn our heads. Somehow – he argued – the brain has to factor the head-turning signals into the visual processing to produce a stable grasp of the world. Many studies over many years have confirmed this sort of thing. One region that appears heavily involved is the hippocampus, an older (non-neocortical) part of the brain that adjoins the temporal lobe and that appears to do a great many things including spatio-temporal processing and transfer of short-term to long-term memory.

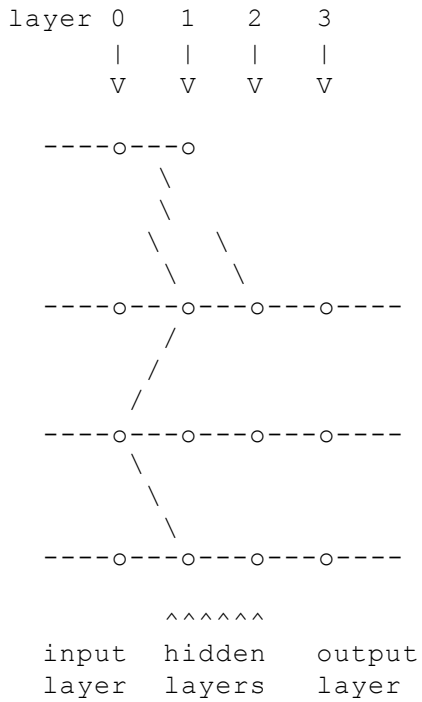
Now, you may be thinking – ok, well and good, but so what? How will any of this help anyone build better, smarter robots?

Well, efference copy is the very trick that one of our humanoid robots – Alice – needed in order to know when she was talking, and thereby to avoid responding to

her own words as if they were commands to her. I had known about efference copy for 15 years or more, and never imagined I would need to build it into my robots almost at the start!

Note that this is not at all the same thing as recognizing ones own voice. I can hear a recording of my voice and think "oh, that's me talking" but I would not mean I was actually in the process of talking as I was thinking that thought. I'd mean just that it sounds like me, and maybe in fact my voice had been recorded some time in the past. I know when I am *actually* talking – not only do my mouth and tongue move and my throat and nasal passages vibrate, but beyond all that *I am aware of involvement in the production of the actvity*. This is unlike, say, ones heartbeat, which we can at times become aware of but that mostly goes on without voluntary control.

So not only are we pushing up against things very relevant to AI and robotics, but also to age-old questions about mind and self and will.

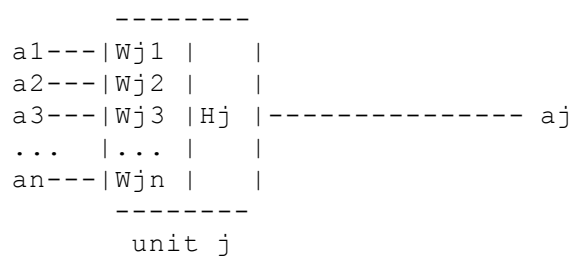


Training

As noted, so-called ML is really just one way – by training – that an agent can learn. ML is the subject of the current section. In later chapters, we will look at some other ways (e.g., reasoning and communicating). ML has become rather math-intensive in recent decades, so we will provide a summary of some of the relevant calculus and linear algebra as needed.

Artificial Neural Networks (ANNs)

Neural networks are abstract mathematical models of interconnected neuronal behavior. Perhaps the first such model was that of the McCullough-Pitts artificial neuron in 1949. This postulates a unit of processing (a cell body and an axon) with characteristics indicated (Note: the Hj in the figure below should be Theta_j; and in lecture I used the subscripts Wij instead of Wji.)



Here the ai are incoming signals (0 or 1) that arrive at the processing unit j, where each is multiplied by a weight Wij and the result summed: SUM = aiWj1 + a2Wj2 + ... + anWjn. This sum is then compared to the threshhold value Theta_j for the unit. Finally, the output signal aj (which is sometimes called the activation level of the unit) is defined as

$$a_j = \begin{cases} 1 \text{ if SUM} \geq \text{Theta\_j} \\ 0 \text{ otherwise} \end{cases}$$

More precisely,

$$a_j(t+1) = \begin{cases} 1 \text{ if } SUM(t) \geq H_j \\ 0 \text{ otherwise} \end{cases}$$

where SUM(t) = ai(t)Wj1 + a2(t)Wj2 + ... +

an(t)Wjn. A neat mathematical notation for

this is

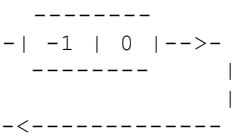
$$a_j(t+1) = H(SUM(t) - Theta_j)$$

where H(x) = 1 if x >= 0, and 0 otherwise (the so-called Heaviside function). Using the vector dot-product we can write this even more compactly as aj(t+1) = H[a(t).Wj - Theta_j].

That is, the incoming signals ai vary dynamically at each time step, and the output signal aj is recomputed and sent one time-step after the incoming signals arrive. The weights Wij can be any real numbers; a negative weight can reduce the overall incoming contribution to SUM, and corresponds to an inhibitory synapse.

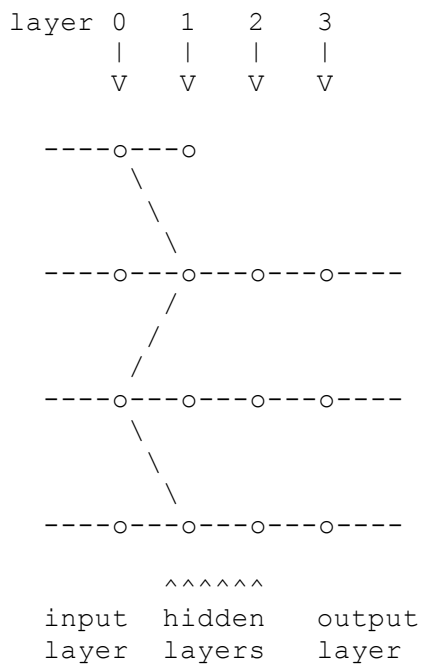
By a (neural) network is meant a collection of such units that are connected to one another via signal wires. An output wire such as aj above can split and connect to many units. Note that in principle aj(t) can be an input to unit j and thus can influence aj(t+1). A network that allows this is called "recurrent". One prominent and important example is the "complete" (aka Hopfield) network in which *every* pair of units is connected by a wire.

Here is a simple example of a (recurrent) network:



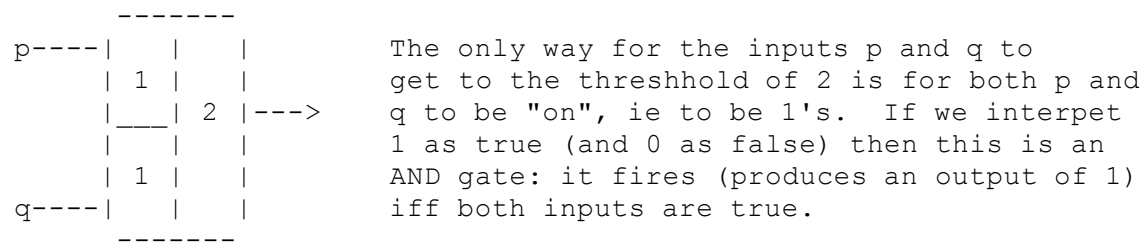
This has just one unit which feeds back to itself. It "blinks" since whatever the incoming signal is on the left (say on, or 1), it produces the opposite (eg off, or 0) on the right one step later, but then that (e.g., 0) becomes the new incoming signal, etc. So the output signal perpetually changes back and forth: 0,1,0,1,0,...

In addition to recurrent networks, there are so-called feedforward networks, in which each neuron is used only once in a given computation, as signals pass through it (from left to right) and then on to another "layer" of neurons:



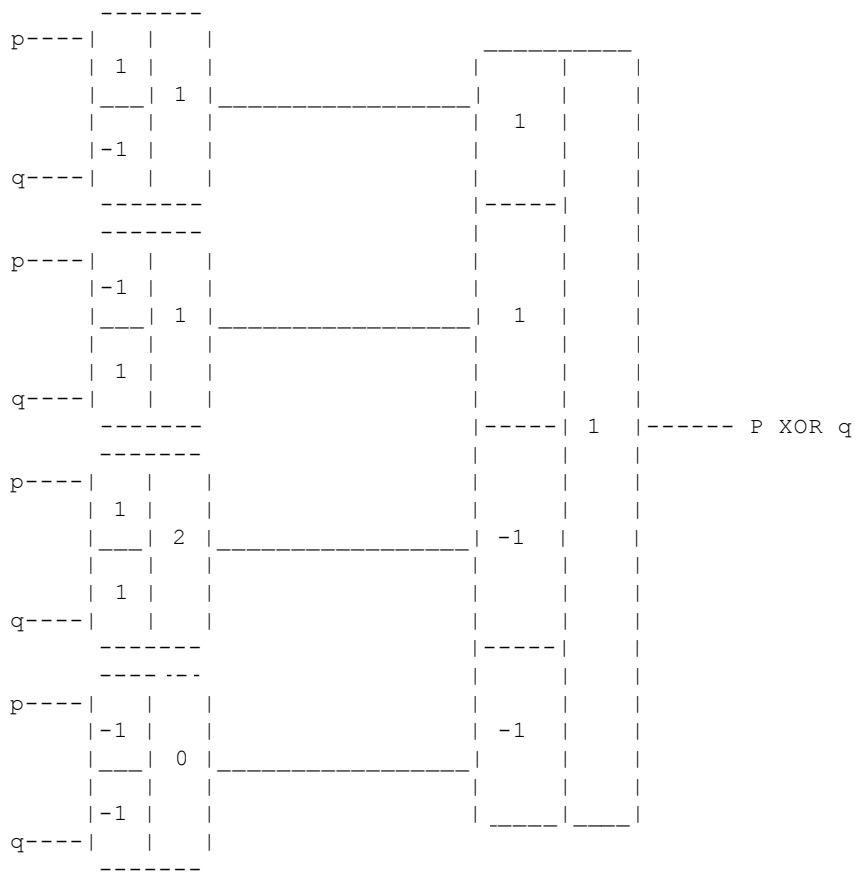
In the diagram, layer 0 is the input layer, layers 1 and 2 are "hidden" layers, and layer 3 is the output layer. (Only a few “axons” connecting units in one layer to the next layer to the right are shown, to keep the diagram uncluttered.) The input-layer units simply pass on the incoming data to the next layer; they do not perform any computation.

Here is a (rather trivial) feedforward network that acts like an AND-gate:

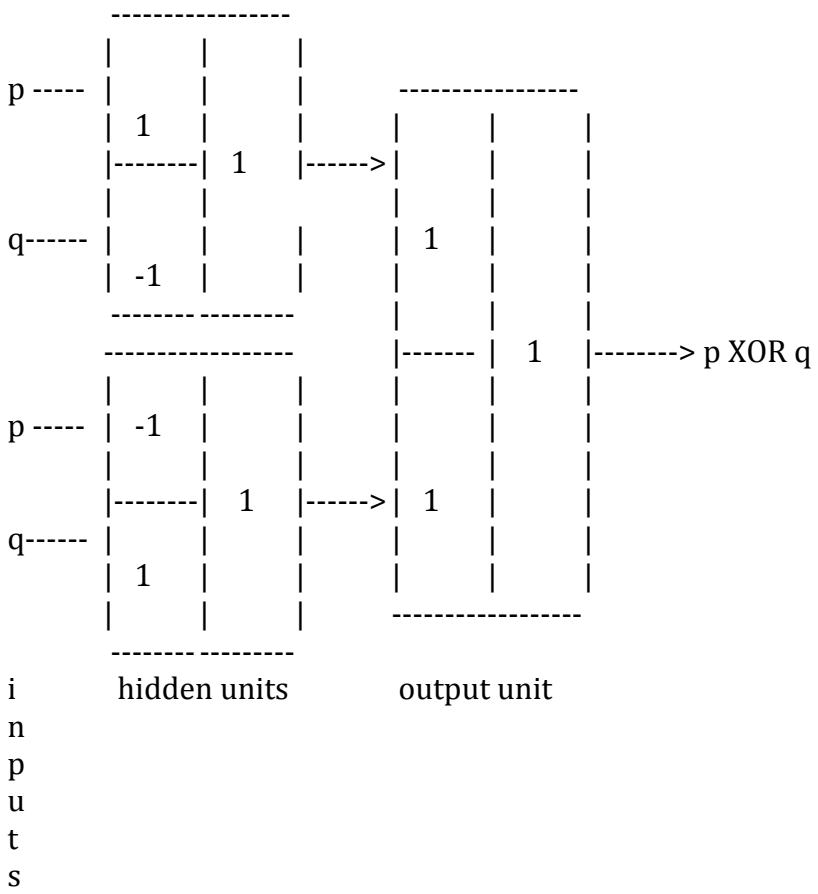


Notice that there are no hidden layers here: just an input layer (p and q) and an output layer, for a total of three units. Also one can easily and similarly create OR-gates, NOT-gates, and many other logic gates.

But it turns out that to make an XOR-gate (true iff exactly one of its two inputs is true) one must have at least one hidden layer; here is one way to do it that makes explicit use of all four possible input-pair values for p and q:



Here there are two input-layer units, p and q, as before; for ease of drawing I have shown each four times. But although this is very clear and easy to understand, it is unnecessarily complex and we can simply remove the bottom half, leaving this:



The two hidden units fire if either p is true and q false (top unit) or vice versa (second unit). Thus at most one of the two hidden units can fire for any given input values; and any such firing corresponds to p and q having opposite values (which corresponds exactly to XOR).

With enough hidden units in a feedforward network, it is possible to compute any computable function. But building a network by hand to compute a function is not where the excitement lies. Rather, it is in having the network “train” automatically (i.e., via an algorithm) to learn to compute a function.

For feedforward networks there is a famous and much-used algorithm, "gradient descent with error backpropagation" (often inaccurately just called backprop), that allows the network’s synaptic weights Wij to be adjusted to fit with ``training data" so that the outputs are the desired ones for given inputs; and after that the network often tends to exhibit the desired input-ouput relationship even for new data. For instance, suppose a feedforward network is trained on 100 instances of the handwritten letter "E" and another 100 that are not E's. Let us further suppose that there are, say, 625 input units (corresponding to a 25x25 grid of pixels) and two output units (corresponding to "E and "not-E"). Once its weights are adjusted by repeated runs of the learning algorithm so that it correctly categorizes the 200 inputs, it could then be used with no further adjustment to distinguish new

handwritten letters (as E or not-E); of course it surely would make errors, but if the percentage of errors is very small we might be content. A more sophisticated network might be trained to distinguish all 26 letters, and upper and lower case as well. The rough idea is to compare the actual output with the desired output (for a given input) and calculate a set of weight-alterations that will bring the ouput closer to what is wanted. The actual algorithm uses derivatives and requires a change in the basic formula for aj(t+1) so that instead of the Heaviside function, a differentiable substitute is used.

Recurrent networks can also be trained, and have found useful application when matching a new item with a set of stored ones to see which is the closest match. In this paradigm, there are no layers, and the entire network can be used for input, as well as for output.

Input values can be specified as activation values (0 or 1) on the connecting wires; and output values are also so specified, after the ensuing computations settle into a stable (unchanging) state.

An example is face-recognition: a complete (aka Hopfield) network is trained on, say, 100 faces (photographs) so that, given any one of them as input (eg pixel data) it it simply remains in that state. Then a new photograph is presented, and the network goes through stages of processing as activation levels change until eventually each wire reaches a final activation (0 or 1) that no longer changes. It turns out that (if the training was done properly) the final state is exactly one of the original 100 stored images, and (typically) the closest match to the new image.

A famous example involved a Hopfield net trained on one photo of JFK among many of other people, and then (after training was over) it was presented with a photo of JFK from a very different angle (such as profile instead of face-forward). It immediately picked out the old JFK training image as the best fit.

While feedforward nets can also be used for this purpose, Hopfield networks can store far more information due to the very large number of connections. Viewed as storage devices, complete networks are said to implement a kind of "associative memory."

Now, if you are not impressed with the above example, let's return to ML's newest trick: deep learning.

Imagine a "stack" of two layers of units, with an input layer having, say, nine units in a 3x3 array-pattern, where each unit's input is either light or dark (or 1 or 0); and a second layer with just one unit, with threshold value 0.5, and with connections from all nine below plus having an output wire. Also assume the synaptic weights are required to lie between 0 and 1/9, and initially set randomly in that range.

If we "run" this network with some input values, we'll see some output value (0 or 1) at the top. Let's try using that output as a predictor of the input: whatever the output is, pretend all nine inputs were that same value. Of course, most of the time this will probably be false. But some predictions will be better than others, in the sense that they predict more units correctly over a wider assortment of input values. Let's use the number N of correctly-predicted input values (0<=N<=9) as a measure of the goodness of the prediction.

Now suppose we allow the synaptic weights to vary randomly, while watching how good the predictions are, and then we choose the "best" weight selection, i.e., the set of values that gives the best average N overall for general inputs. (This would of course be incredibly inefficient of there were many weights to vary; but consider the idea as a theoretical exercise for the moment, in our example with only 9 weights.)

What would those "best" weights be? Answer: all equal to 1/9. And then the output simply tells us whether there are more light input values than dark; i.e., it picks out the "majority" feature. When only 0-4 units are 1, the output is 0; and when 5-9 are 1, it is 1.

In effect, we are using the single output wire (one number) to represent a lot of data (nine numbers), and to give us a sort of summary of those numbers. That can't be very accurate in all details, of course; but it still tells us something interesting about them.

Now, we did this ourselves, by watching and selecting. But all this could have been automated, so that the network would be comparing its own output to the input. And then it would "discover" that its best weights are such-and-such – namely all ones in this case – on its own.

Alas, automating this process gets very computationally expensive when applied to cases with more than just a small number of units. But the good news is that mathematical/statistical tricks can reduce the cost enormously. Still, the basic idea is simple: use the compressed output layer to predict the more expansive input layer, varying the weights in the process. When the predictions become good, this means the network has found some summary-feature of the inputs that is being captured in (some of) the output units. And that – in essence – is the underlying idea behind deep learning.

In the pretend case above, one of the learned/discovered features was lightness/darkness. In the famous Google case, a kind of "catiness" was learned; there were many layers and many units. And the varying of the weights was not random but rather used gradient-descent with backpropagation. We will turn to these shortly, after introducing more linear algebra.