

Two important (and simple) concepts in computational neuroscience

Robert Biegler

The way brains process information is rather different from the way conventional computer programs do so. And although there is no end to how technical you can get when studying real neural networks, there are two basic ideas which can take you some way towards understanding how brains work. These are distributed processing and encoding, and attractors. One important property which comes out of having attractors and distributed coding is content addressable memory.

Let's say you want to know the film in which a hotel owner held his index finger horizontally under his nose and goose-stepped through the dining room. Try to put that into a database of videos, and I doubt you will get very far. Ask in your average British video store and you will be told "It's Fawlty Towers, with John Cleese, the German episode". Humans can retrieve whole memories when given small components of them. That is content addressability. Artificial intelligence researchers have found that it is very hard to make computers do the same thing. In principle, distributed coding in a network with attractors does the job. (I will discuss later why databases don't use distributed coding.)

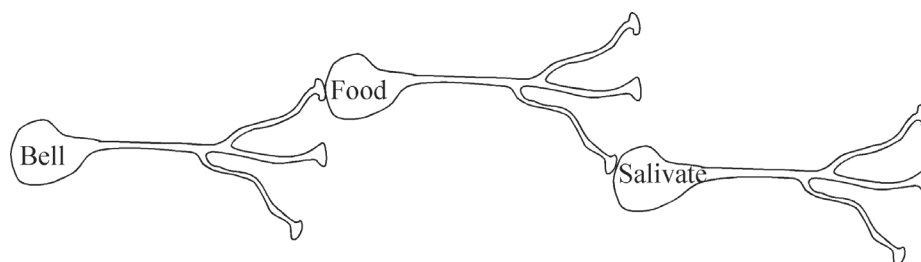
To see how this could work, start off with a local, non-distributed code of conditioning in one of Pavlov's dogs. Whenever a bell rings, the dog soon afterwards gets a piece of food. One simple way to model learning is to assume that there is a neuron in the dog's brain that is a 'bell-detector', that

fires whenever the dog hears a bell. This neuron is very weakly connected to many other neurons, including a 'food-detector'. Neurons have a firing threshold. They only fire if the activation they receive pushes them above this threshold. Initially, the connection between the bell detector and the food detector is so weak that hearing the bell will not make the food neuron fire. Learning consists of strengthening connections whenever activity in the first neuron coincides, or is soon followed by, activity in the neuron it connects to (there are other learning rules, but this one will do for the moment). So if activity in the bell-detector is soon followed by activity in the food-detector, the connection

between them is strengthened. Eventually, it becomes strong enough that the sound of the bell can activate the food-detector. Effectively, the sound of the bell reminds the dog of food.

Note that the learning rule is quite automatic. There is no intelligence built in, and no overall goal that guides learning. The rule just says that if activity in one neuron is soon followed by activity in a second neuron, then the connection from the first to the second should be strengthened. There is no more to it than that.

That sounds all very nice, but you may protest that bells have not featured prominently in the evolutionary history of carnivores, so where does that bell-



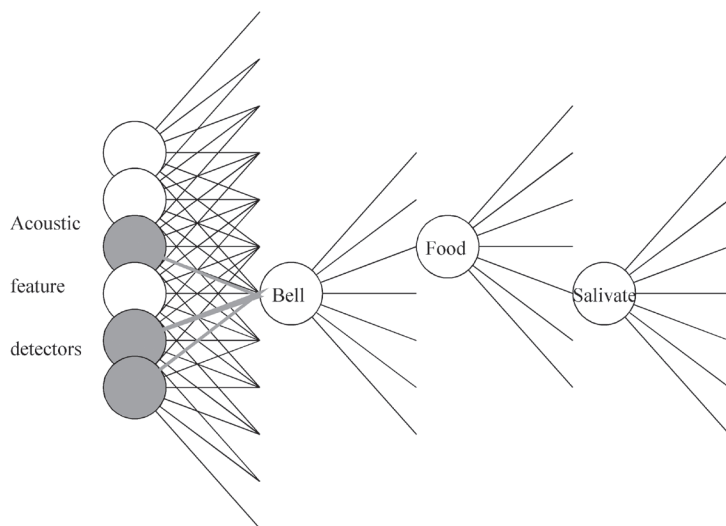


Figure 2

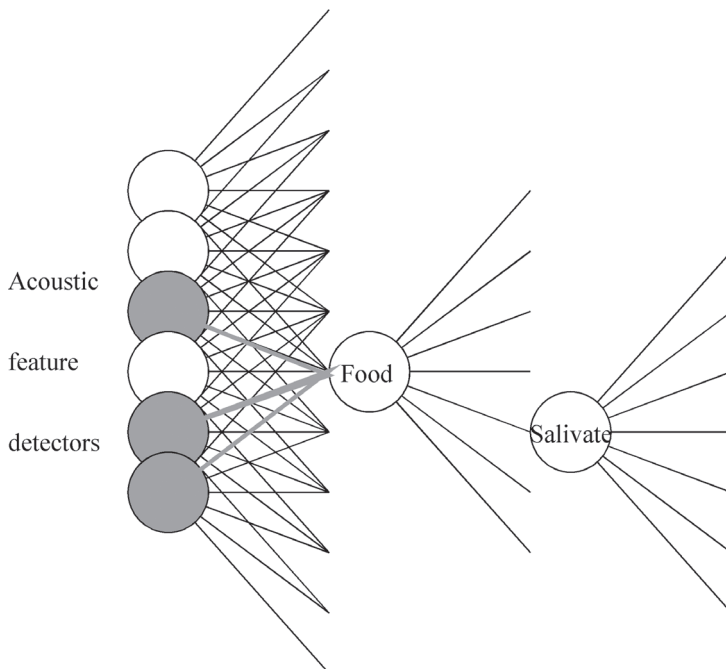


Figure 3

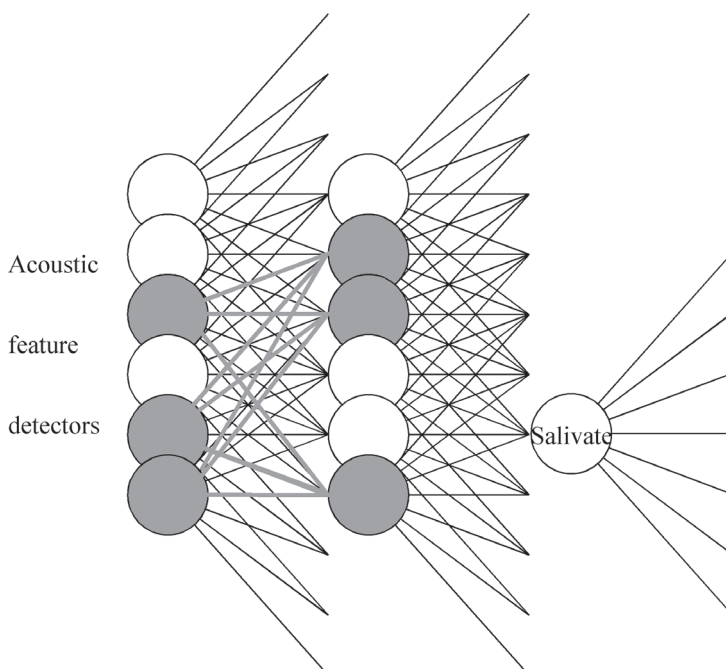


Figure 4

detector come from? One possible answer is that acoustic feature detectors link up with the bell-detector, and that those connection strengths are set up so that all the feature detectors that are active (shown in red) when a bell rings become strongly linked to the bell-detector. (Figure 2)

But how does that happen? The learning rule that relies on activity on both sides of the connection does not seem very useful. After all, why should the neuron that will become a bell-detector fire? In the case of connecting the bell- to the food-detector, the sight and smell of the food will activate the food-detector, and so satisfy the conditions of the learning rule. But there is nothing to make the future bell detector fire before the connections from the acoustic feature detectors are set up.

There are other learning rules which can deal with this difficulty, but it turns out that we don't actually need them. All the important information really is in the connections from the acoustic feature detectors to the bell detector. The bell detector merely passes to the food detector the information that a bell has been heard. If we connected the acoustic feature detectors directly to the food detector, that would fulfil all the conditions necessary for our learning rule to work (Figure 3). Whenever a bell is heard, a specific pattern of feature detectors will be active. If that activity is soon followed by activity in the food detector, the relevant connections will be strengthened, and the ringing of the bell can then activate the food detector. In the firing pattern of the acoustic feature detectors we have a distributed representation of the sound of a bell.

Having got rid of the bell-detector, do we actually need the food detector? Dogs can distinguish quite well between different kinds of food, so there can't be just one generic food detector. Some of the food dogs can identify is no more natural than the sound of a bell. And if we have something like a dog biscuit detector, the same argument applies as to the bell detector: there must be pattern of activity in feature detectors that is unique to this specific food, and

all the important information is in this pattern. Is it possible to eliminate all the localised representations and link up two patterns like this (Figure 4)?

To look into that, it would be helpful to have a simplified representation of the connection patterns. The picture above is just too confusing. So imagine rotating the row of neurons in the middle 90° to the right (Figure 5). Their axons then go down, but because we are interested in the connections from acoustic features to a food pattern, we don't actually need to look at the axons going out from there. We also put dendritic trees on, the kind of branching structure that neurons use to receive input. Then the axons from the acoustic feature detectors can just project horizontally to the right, making links to the dendritic trees sticking up vertically from the horizontal row of food feature detectors below. The little circles are the connections, or synapses. The coloured circles show strong connections, the white circles show connections that are very weak. This is the same pattern of connections as shown above.

Putting one pattern in and getting one pattern out is all very well, but it seems rather inefficient to devote a whole network of neurons to that single purpose. Can we put several different associations into the same network? It turns out that is possible. Figure 6 shows a representation of the network that has been simplified yet more. The vertical row on the left of the line is an input pattern or *input vector*. The first such vector is the same one we have seen before, which stands for the sound of a bell. A 0 means there is no activity in that feature detector, a 1 means the detector is active. On the horizontal row below the line is the output vector. That is what the network is supposed to produce when given either the whole input vector or a fragment of the input vector. The numbers in between are the *connection matrix*. 0 stands for an inactive connection, equivalent to the empty circles in the last figure. 1 stands for an active connection, equivalent to the coloured circles in the last figure. In this very simple network, we have only

those two values; connections are either totally off or totally on, at full strength. The learning rule is simply that if there is simultaneous activity on both input and output side, set the connection to full strength, otherwise leave it as it is. This is called the *Hebbian learning rule*, after its inventor, Donald Hebb.

The first pairing is that of bell with food, as before. Let's say the second pairing is learning to associate the sight of a leash with going for a walk (we need more than just acoustic feature detectors on the input side for that). The third pairing might be the association between chasing after a swan and getting a good whack across the nose.

We now have overlaid the patterns of connections from several associations. Can we get the originals out again? Can we perhaps even get an output vector when we only have a fragment of the input vector?

We can, provided we add two more things. One is that we need an *activation function*, which is a specification of what a neuron is supposed to do with its inputs. I will assume a simple threshold function. If the sum of the inputs reaches or exceeds the threshold, the output of the neuron will be 1, otherwise it will be 0. Looking at the connection matrix after the first pairing, we can see that if we put in the same input vector

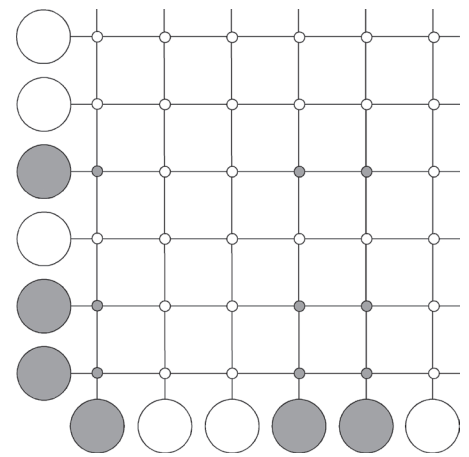


Figure 5

as before, and we set the threshold of each output neuron to 3, then we can get the same output vector as before. If we put in any other vector, nothing will happen at all (please don't just take my word for it, do check).

If we just leave it at that, we won't have pattern completion. If you put in only part of the original input vector, none of the output neurons reaches threshold. What is needed is either a way to normalise, to compensate for the number of active inputs, or a variable threshold. A variable threshold is not biologically plausible, so normalisation is it. It can be achieved by having an inhibitory interneuron. The more inputs are active, the more inhibition (blue synapses) arrives at the output neurons (Figure 7).

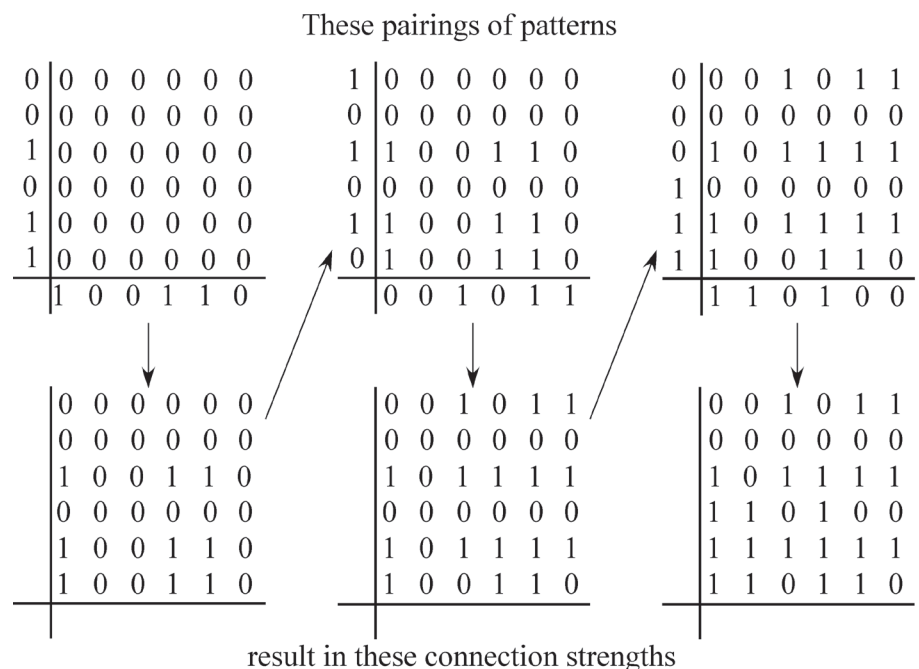


Figure 6

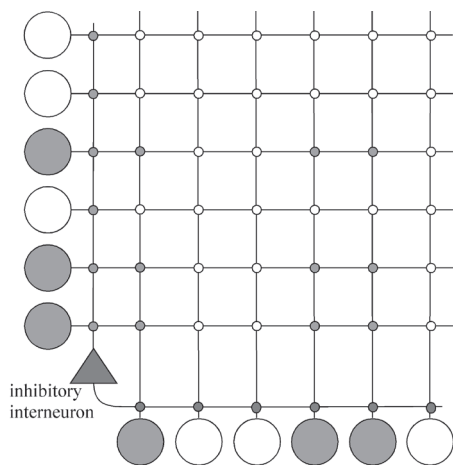


Figure 7

By tuning the inhibitory interneuron appropriately, this can have the same effect as dividing the incoming activation by the number of active input neurons. If we use numbers for the activations and connection strengths again, we can see whether we can retrieve an association after the network has learned several more associations. The diagram on the next page (Figure 8) shows the connection matrix after learning the three pairings of input and output vectors above.

The 3 or 2 in the leftmost column shows the activation of the inhibitory interneuron, which is simply the number of active input units. The activation at the output neurons is the sum of inputs multiplied by connection strengths, then the whole lot divided by the inhibition. If the final value reaches 1, the output neuron fires, if the input remains below that threshold,

Complete input vector	Fragment of input vector with unique combination
0	0
0	0
1	1
0	0
1	1
1	1
1	1
1	1
1	1
3	2
1	1

Fragment of input vector with all shared components	Error correction
0	0
0	0
1	1
0	1
1	1
1	1
1	1
0	1
2	3
1	1

Figure 8

the output neuron remains silent.

We can get the original association back, even if we put in only a fragment of the input vector, so long as that fragment contains a unique combination of components. Only the first input vector had both the third and the sixth input neuron active, so putting just those two in still retrieves the complete association. Both the first and the second input vector had the third and fifth input neurons active. Give the network that retrieval cue, and it returns a combination of both output vectors.

These examples were all of associating different input and output vectors, but they can also be the same, by having recurrent connection which feed the output back into the input. What is the point? Then you can exploit the pattern completion abilities of the network for complete recall from a retrieval cue consisting of only a small part of the original information. That is the *content addressability* that computers have so much trouble with. Recurrent connections appear, for example, in the hippocampus, and are also part of models of top-down processing in perception.

The key concept here is that of an attractor. To see how that works, consider the three neuron network in part A of the next figure (Figure 9). Each neuron inhibits both others (the lines with small empty circles represent axons with inhibitory synapses) and has an excitatory recurrent connection. Assume all inhibitory connections are equally strong, and that all excitatory connections are equally strong. For a first approximation, assume that the activation of any neuron is simply a weighted sum of all inputs, with an upper and lower limit to activity. This architecture creates a winner take all network. (To keep things simple for the moment, this network does not have a distributed representation.) If you provide external input to, for example, neuron z, it will inhibit neurons x and y, provide excitation to itself through the recurrent connection (the line with the filled in circle represents an excitatory synapse). Therefore, z will

remain active after you take the external input away, until you reset the network through external inhibitory connections (none of the external inputs are shown here). Now provide excitatory input to all neurons, giving neuron x a little more. It will then gain more through its recurrent connection than the other neurons do, and it will send stronger inhibition to the other neurons. Under these conditions, x will remain active when you take the external input away, while y and z will fall silent. Whichever neuron has the strongest input will remain active, and the others will stop firing.

If you represent the activity pattern of the whole network as a vector in a three dimensional space (one dimension for each neuron; see part B of the figure), and you ask where in that space will the network activity end up once you remove the external input, there are three stable states, with either x, y or z maximally active. All other initial activation state will converge onto, or in other words be attracted to one of these stable points. The network has three point attractors. The volume of space from which the network approaches one specific attractor is the basin of that attractor.

Where the borders between the attractor basins are depends on the relative strengths of connections. Imagine that the inhibition from z is stronger (part C). Then even a slightly weaker input to z will still allow it to suppress x and y, meaning the borders in that space have shifted away from z, so that z has a larger attractor basin (part D). The number of attractors also depends on the pattern and strengths of connections. Make the connections between x and y mutually excitatory (E), and they will always be jointly active. The separate attractor basins for x and y have joined into a single basin. Now neuron z will need strong input to suppress x and y, so the volume of the attractor basin around maximal activation of z will shrink (F).

In this three neuron network, approach to an attractor involves changing the firing rates of neurons, while in the quite abstract and simplified

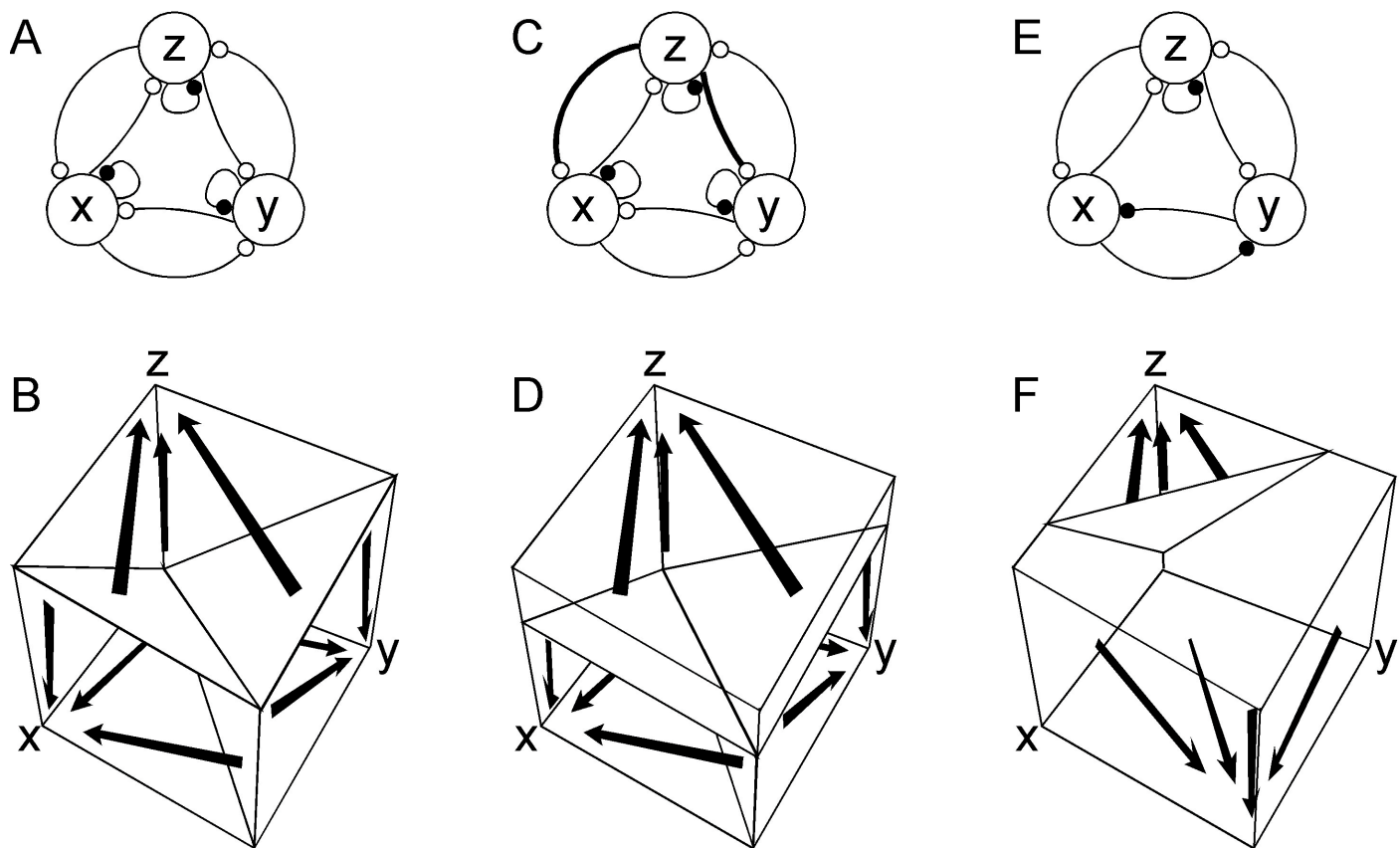


Figure 9

associative network above, neurons were either on or off, and approaching an attractor state involved changing the pattern of firing. In a real biological network, you must expect both these things. In any case, the basic principle is the same. The pattern of connections sets up a limited number of stable states or attractors, and within some limits, deviations from these attractors are corrected. In the associative network, each distinct memory is one point attractor, and the approach to that attractor provides pattern completion, and therefore a content addressable memory.

The idea of retrieval as approaching an attractor state can reconcile two notions of forgetting, that of memory being overwritten like a file on the hard disc of a computer, and that of temporary retrieval failure which can be overcome by providing a better retrieval cue. In a simple associative network, forgetting would involve random changes to synapses, which would change the volumes of attractor basins. If the attractor basin of one memory item shrinks, then that attractor can only be reached from starting points quite close to the attractor, which means

you need a very good retrieval cue. Eventually, the basin may disappear entirely, but that is merely the endpoint of a continuous process.

Models of top down processing in perception also typically involve a network with recurrent connection which perform pattern completion through having discrete attractor states.

The networks described here are still quite limited. For example, retrieval does not change the associative network described above. That is in marked contrast to the phenomenon of retrieval practice: imagine I tell you that the Spanish word *cebolla* means onion, and that *murcielago* means bat. Later I again tell you the meaning of one of these words, but I *ask* you the meaning of the other. Assuming you still can retrieve that information, you will later remember the retrieved word better than the one I presented again. But retrieval does nothing to the simple model network, and so it cannot provide any insight into retrieval practice. Perhaps the network's most fundamental shortcoming regarding the psychology of memory is that it *always* gives you some result, whether

it makes sense or not, and is incapable of providing an output that means "I don't remember". The network also is enormously simplified when it comes to the biology. Real neurons are much more complex than the on-off switches with a threshold in this simple model. It is already clear that at least some of these subtleties are important for at least some computations. Models at the level of this one are only the beginning. Still, understanding these basic ideas should give a reasonable start towards understanding brains.



Robert Biegler was born and hasn't died yet. He is interested in ravens, chicory, cephalopods and the evolution of cognition, and he organises the Trondheim Nerd Pride March. He firmly believes the world should be a little bit more surreal.