McCulloch-Pitts neurons

The McCulloch Pitts neuron model, or Threshold Logic Unit, was introduced in 1943 by Warren McCulloch and Walter Pitts as a computational model of a neuronal network [1]. Their thinking was that neurons are joined to each other with connections of variable strength; 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 McCulloch Pitts neuron was the first example of an artificial neuron and had a step function for ϕ:

$$x_i = \begin{cases} 1 & \sum_j w_{ij}x_j > \theta_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

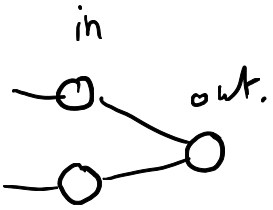
Thus, the neuron has two states, it is in the on state, $x_i = 1$ if the weighted input exceeds a threshold θ_i and an off state, $x_i = -1$ if it doesn't; the picture you might have of how this corresponds to the brain is that 'on' corresponds to rapid spiking and 'off' to spiking at a much lower rate. The w_{ij}, the connection strengths, 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 in silicon 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 connect strength to the original McCulloch-Pitts neuron.

Perceptrons

The perceptron is a machine that does supervised learning, that is, it makes guesses, is told whether or not its guess is 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 $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ 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 described above. 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

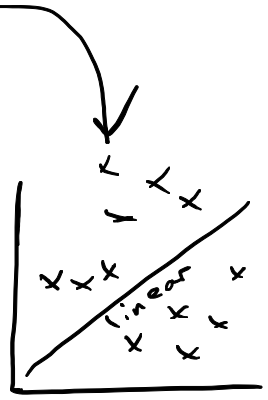
$$\delta\theta = \eta(d - y) \tag{6}$$

where η is some small learning rate and

$$\begin{aligned} w_j &\rightarrow w_j + \delta w_j \\ \theta &\rightarrow \theta + \delta\theta \end{aligned} \tag{7}$$

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 $\sum_i w_i x_i = \theta$ 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. 1. 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. 2, 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 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

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

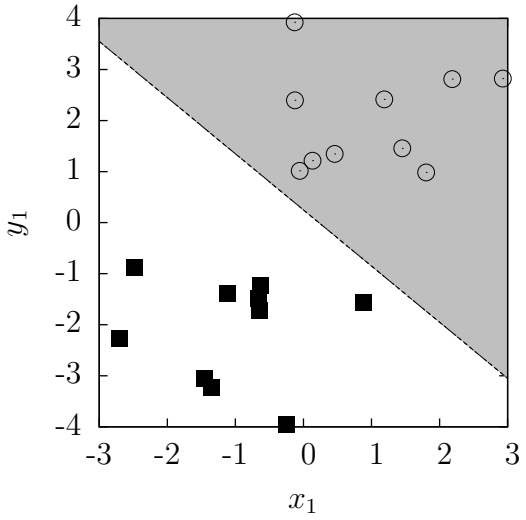


Figure 1: Here the shaded region corresponds to $1.1x_1 + x_2 > 0.25$, so if $w_1 = 1.1$ and $w_2 = 1$ with $\theta = 0.25$ then the corresponding perceptron neuron will be one for all the circle points and -1 for all the square points.

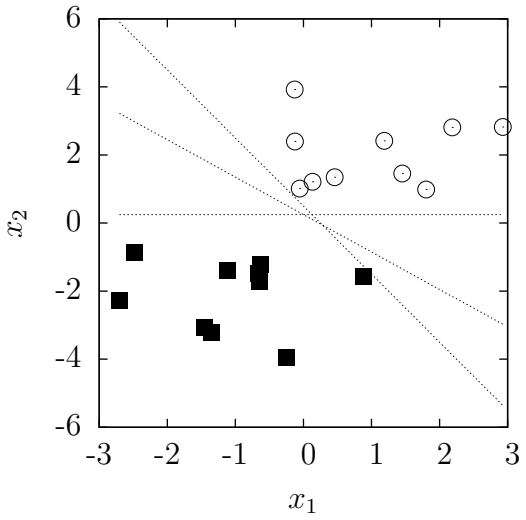


Figure 2: All of these lines separate the points into two classes

`github.com/conorhoughton/COMS30127`

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.

The cerebellum as a perceptron

The cerebellum has a number of striking features; it has a more stereotypical structure than most brain area and this structure is conserved across species. It also has one of the brain's largest cells, the Purkinje cell, and its most numerous, the granule cell.

`github.com/conorhoughton/COMS30127`

Figure 3: A drawing by Santiago Ramón y Cajal of a Purkinje cell. [Picture taken from `http://en.wikipedia.org/wiki/Golgi's_method`]

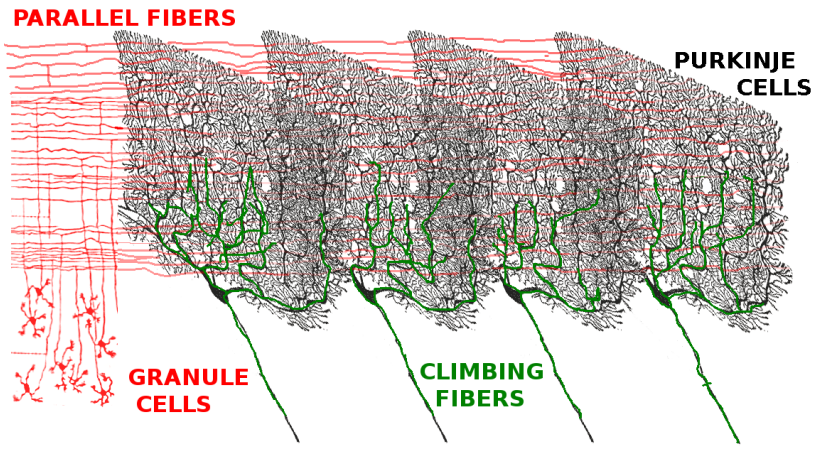


Figure 4: A cartoon of the cerebellar circuitry. A vertical axon rises from each granule cells, splits once and then extends horizontally in two directions making connections with multiple Purkinje cells. Each Purkinje cell has its own climbing fiber which winds up around it.

Purkinje cells have a distinctive structure with a huge, highly branched, but flat dendritic arbor, see Fig. 3; this allows an extensive connectivity with each Purkinje cell receiving inputs from around 100,000 other cells. In the cerebellum the Purkinje cell are lined up like pages in a book, with their arbors lying in parallel planes. They receive two excitatory inputs, weak inputs from parallel fibres, axons that run perpendicular to the planes of the Purkinje cell dendritic arbors, and a strong input from a climbing fibre, a single axon which winds around the Purkinje cell and makes multiple contacts with it, see Fig. 4.

Another peculiarity is that the Purkinje cell has different responses to different inputs; in response to multiple weak inputs from the parallel fibers it fires a normal sort of spike, called in this context a *simple spike*; in response to single spike from the climbing fiber is fires a special spike, called a *complex spike*, with a leading spike, a number of small 'spikelets' and a sustained after-period of depolarization; this is illustrated in Fig. 6.

It is still unclear exactly what the cerebellum does; what is known is that it is important for actions, fine motor control and proprioception; problems with the cerebellum are associated with ataxia, loss of fine motor control,

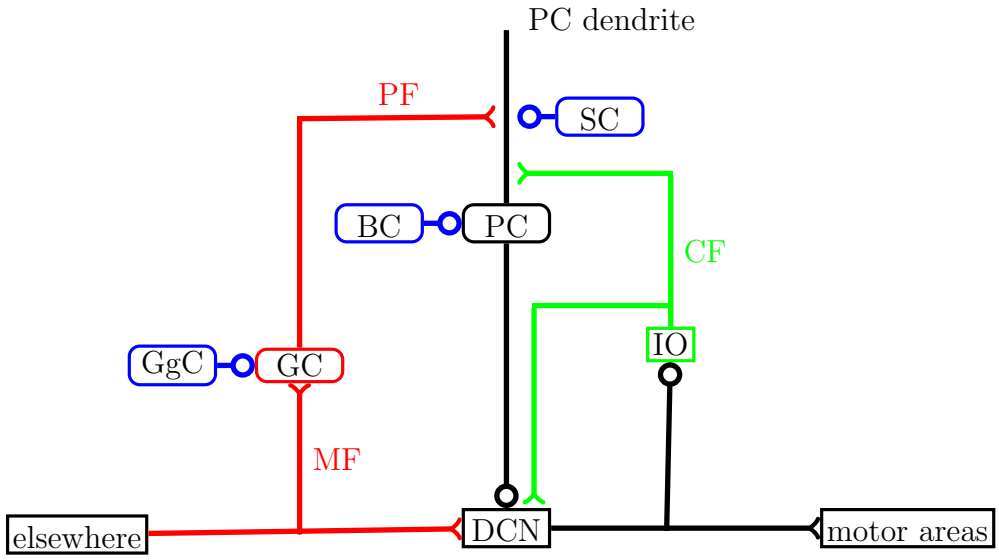


Figure 5: A schematic of the cerebellar circuit. The granule cells (GC) receive input from a diverse range of other parts of the brain along the mossy fibers (MF). Each granule cell will combine input from just three or four mossy fibers and do this in lots of different combinations. The parallel fiber (PF) carries spikes from the GC to the Purkinje cell (PC) whose large dendrite is drawn as a line. The PC also receives input from a climbing fiber (CF) coming from Inferior Olivary Nucleus (IO). In turn it sends an inhibitory signal to the Deep Cerebellar Nucleus (DCN); the DCN has inhibitory neurons which act on IO and excitatory neurons which act on the motor system. The basket cells (BC), the Golgi cells (GgC) and the stellate cells (SC) are all local inhibitory cells.

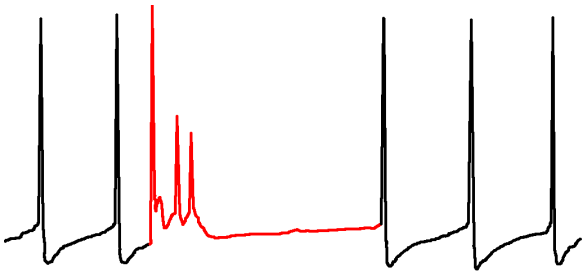


Figure 6: A complex spike. This drawing shows simple spikes in black and a complex spike in red. The complex spike is followed by a long refractory period during which spiking is not possible. This is a sketch, not an actual recording, but a typical time scale would have this refractory period 50 ms long.

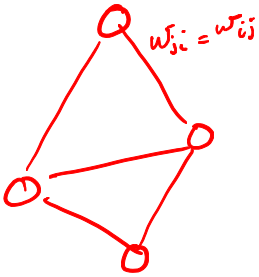
poor motor learning and poor balance. There is a specific gait associated with cerebellar damage, one that exhibits a certain self-consciousness or vigilance is required for movement. According to most ideas about cerebellar function it is required for the calculation of fine motor signals [3], or for predicting the sensory or proprioceptive consequences of motor actions [4]; it is believed it encodes forward and backwards models of movement so that it predicts the consequences of motor commands, or calculates the motor command that will a particular movement.

Whatever exactly it does, it is widely believed, in accordance with the Marr-Albus model [5, 3], that the connections from parallel fibers to Purkinje cells acts as a perceptron. Thus, if y is the output of the Purkinje cell and, in this simple model, taking into account the fact Purkinje cells are inhibitory

$$y = -\sum w_i x_i \tag{15}$$

where the x_is are the activities in the parallel fibers and w_i is the strength of the synapse from the ith parallel fiber to the Purkinje cell. The idea in the Marr-Albus model is that the climbing fiber carries the error signal $d - y$ making the cerebellum an example of supervised learning.

Hopfield network



In contrast to a perceptron, a Hopfield network is a recurrently connected network; it is intended to perform pattern completion and was proposed by John Hopfield in 1982 [6], though other people had had the idea before in different contexts. The idea behind a Hopfield network is that you evolve the network according to the McCulloch-Pitts relation, so, in the synchronous update version, from one iteration to the next

$$\hat{x}_i = \phi\left(\sum_j w_{ij} x_j\right) \tag{16}$$

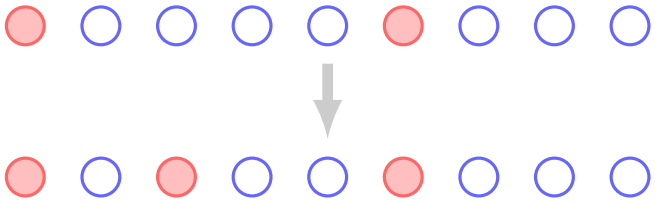
and then $x_i \to \hat{x}$; that is all the nodes update using the old values. In the most common version of a Hopfield network, the w_{ij} are symmetric, that is

$$w_{ij} = w_{ji} \tag{17}$$

The threshold values θ_i have been set to zero, this is something you can do in a Hopfield network if you want because the learning rule doesn't change to threshold. In the asynchronous scheme, the you update the nodes one-by-one, for example, after choosing a random node.

The idea is that this is a model for *auto-associative* memory. Auto-associative memories are patterns representing memories along with some dynamics that complete partial patters. Imagine a sequence of McCulloch-Pitts neurons

where the filled circles correspond to on. Recall occurs when the network is presented with a partial pattern and evolves into the complete patterns.



The Hopfield network is intended to model the recall, and learning, of memories in an autoassociative network.

One way to think about this is to note that there is an 'energy' associated with a Hopfield network:

$$E = -\frac{1}{2}\sum_{ij} w_{ij} x_i x_j \tag{18}$$

and you can show than if you update a node you will reduce the energy. Roughly speaking if you update a node x_i then it is more likely to have the same sign as a connected node x_j if the connection between them is large and positive since this means $w_{ij}x_j$ will have a big effect on the activation of x_i and vice versa; x_i and x_j are more likely to have an opposite sign if the connection is large and negative. This means that updating the neurons will tend to reduce the energy. In fact, this can be proved and that the system will evolve to a local minimum.

Now the question is how to create the local minima? Here are two similar rules to achieve this:

$$w_{ij} = \frac{1}{N}\sum_a x_i^a x_j^a \tag{19}$$

or

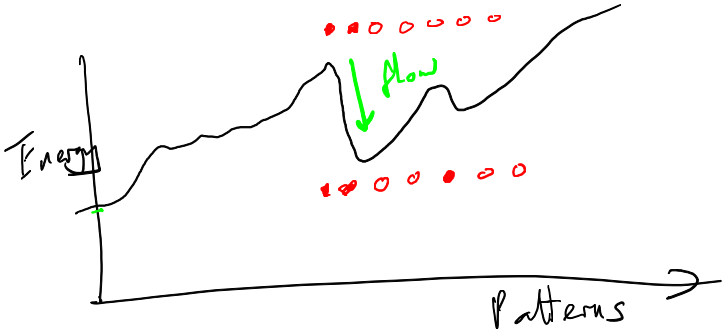
$$\delta w_{ij} = \eta(x_i - \alpha)(x_j - \alpha) \tag{20}$$

where N is the number of patterns to be stored, and a indexes the patterns. There are other rules, in fact there are rules that can store more patterns, but these two rules, the first a sort of 'top down' rule and the second a more on-line rule, are inspired by Hebbian plasticity.

We will discuss synapses in much more detail later on; synapses are the connections from one neuron to another; an *excitatory synapse* means that if the pre-synaptic neuron spikes, it makes the post-synaptic neuron more likely to spike, an *inhibitory synapse* is the opposite, if the pre-synaptic neuron spikes, it makes the post-synaptic neuron less likely to spike. Crucially, synapses can change their strength, both in the short term and in the long term.

Synaptic plasticity usually refers to the long-term changes in synapse strength, a long term increase in synaptic strength is called *long term potentiation* of LTP, a decrease is called *long term depression* or LTD. It is believed that synapses respond to their pre- and post-synaptic activity, so that the changes depend on the behavior of the pre- and post-synaptic neurons. It is not known in detail what rules govern this plasticity, it seems different neurons have different plasticity rules.

`github.com/conorhoughton/COMS30127`



The closest thing to an overall rule was formulated by Hebb in 1949 when he said [7]:

> Let us assume that the persistence or repetition of a reverberatory activity (or 'trace') tends to induce lasting cellular changes that add to its stability. [...] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

In other words, if one neurons tends to cause another to fire, the synapse from the first to the second will get stronger. In artificial neurons or rate-based neurons, lack spiking dynamics and instead have a continuous state or rate variable; since *Hebbian plasticity* often plays a role in artificial neural networks it is often applied to a rule that strengthens synapses between neurons that are active at the same time, that is, the explicit causal structure is ignored in favor of

> Neurons that fire together wire together.

This leads to a plasticity rule

$$\delta w_{ij} = \eta x_i x_j \tag{21}$$

where w_{ij} is the strength of the synapse from neuron i to neuron j, x_i and x_j are the states of the two neurons and η is a learning rate. Another version is

$$\delta w_{ij} = \eta(x_i - \alpha)(x_j - \alpha) \tag{22}$$

where having α allows negative changes, when $x_i > \alpha$ and $x_j < \alpha$, or vice versa.

This plasticity rule clearly matches the on-line Hopfield learning rule; in fact, in the Hopfield network α has to be set equal to the average density of the patterns, that is, the average number of ones, for convergence. So, to recap, during learning the patterns are activated and plastic changes are made to the synapse strength according to a simple correlation based Hebbian plasticity rule.

$$\delta w_{ij} = \eta(x_i - a)(x_j - a) \tag{23}$$

On capacity
↳ Only so many patterns can be memorised by neurons in a network → too many and it becomes impossible to distinguish.
↓ See *

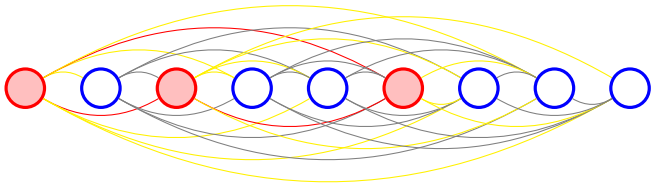


Figure 7: Learning in the associate network. The pattern has been imposed and connection strengths are changed. The red links increase by $\eta(1-a)^2$ and the gray by $\eta(-a)^2$, the yellow links decrease by $\eta a(1-a)$.

Typically a is very small for real networks so there will be a large increase for the connection between two neurons that are active at the same time, a tiny increase for pairs neurons that are inactive at the same time and a medium size decrease for pairs of neurons where one is active and one inactive. See Fig. 7.

During recall some of the neurons are held in the active state and the rest of the network evolves according to a threshold input rule. That means each neuron has an input given by

$$r_i = \sum w_{ij} x_j \tag{24}$$

and is set in the active state if $r_i > 0$. The idea is that after learning the pattern $\{0, 2, 5\}$

the connections between these nodes will be strong, so if the network has nodes $\{0, 5\}$ activated

the value $r_2 = w_{12} + w_{52}$ will be larger than the threshold and the subsequent dynamics will switch neuron 2 on. However, in this network, if a different initial set of neurons are activated, the activity will die away because the r_i will all be sub-threshold.

When many patterns are stored it is likely that there will be interference between them. This is illustrated in Fig. **??**. Although the figure shows

how a single neuron fails to participate in two patterns, for larger networks some overlap is possible, but too much overlap prevents retrieval. In fact the capacity is proportional to the number of neurons, N. A hand-waving argument goes like this: the number of connections is roughly N^2 and the amount of information in a pattern is N so the number of patterns that can be stored is $N^2/N = N$ [8].

The capacity is also larger if there is sparseness; one way to think of this is to observe the weight decrease between an active and inactive connection is $\delta w_{ij} = -\eta\alpha(1-\alpha)$ so the smaller a is the smaller the amount these links are decreased. Links are decreased if, in the pattern, one neuron is active and one inactive, they are strengthened if both neurons are active, the increase is $\eta(1-\alpha)^2$. Hence, it takes $1/\alpha$ patterns where a connect is weakened to wipe out the strengthening that results if the connect is part of a pattern. In fact, it is estimated that the capacity of a network is

$$P = \frac{k}{\alpha}N \tag{25}$$

where k is constant which has been found to be about $k \approx 0.035$, this is reduced to

$$P = c\frac{k}{\alpha}N \tag{26}$$

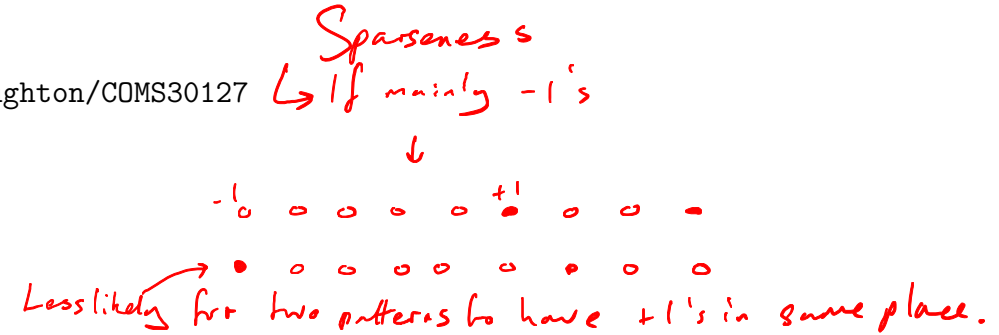
if there are missing connections, where c stands for the fraction of pairs that are connected.

The actual sparseness of the brain needs to balance this advantage, the increased capacity, along with a metabolic advantage and more abstract computational advantage which says that a sparse coding for information involves object recognition or segmentation against the disadvantages, most obviously the vulnerability of the pattern to the loss of neurons or connections and, perhaps more importantly, a sparse code involves fewer elements and so may be less useful for retrieval. It is hard to actually estimate sparseness in practice since neurons are not, in reality, on-off units.

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.

github.com/conorhoughton/COMS30127



[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.

[3] Albus, JS. (1971) A theory of cerebellar function. Mathematical Biosciences 10: 25–61.

[4] Gao J-H, Parsons LM, Bower JM, Xiong J, Li J and Fox PT (1996) Cerebellum implicated in sensory acquisition and discrimination rather than motor control. Science 272: 545–7.

[5] Marr, D (1969) A theory of cerebellar cortex. Journal of Physiology 202: 437–70.

[6] Hopfield, JJ. (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA, 79:2554–2558.

[7] Hebb DO. (1949) The Organization of Behavior. New York: Wiley & Sons.

[8] Amit D. (1992) Modeling Brain Function: The World of Attractor Neural Networks. Cambridge University Press, Cambridge England.