# What is this unit about?

This unit is about how the brain works. We don't know how the brain works, but this unit is about what we do know. Typically when you ask how the brain works, you expect one of two sorts of answers. One sort of answer describes the stuff the brain is made of: neurons, synapses, dendrites and axons, and explains what they do and how it might support computation in the brain. The other sort of answer talks about parts of the brain, the hippocampus, the basal ganglia, the cerebellum, and suggests which algorithm might be at work in each.

Of course, neither of these answers really addresses the central mystery we all want answering: *what am I doing when I am thinking, how can a computational device produce the sense that I am 'I'; what is going on?*—nor does any answer you will get now answer the more modest, but still incredibly ambitious scientific programme that neuroscientists have decided is a reasonable substitute for answering these fundamental questions: what computations occur in the brain? How are these made possible by the action of the brain's constituents across different scales: the chemicals that build synapses, the synapses the connect neurons, the neuronal circuits these connections produce, the brain areas that contain the circuits?

This unit, therefore, describes a small part of a science in progress, a science with incredible ambitions and amazing potential that is still in its infancy. We believe that those things we do understand from neuroscience have something useful to tell us as computer scientists, roboticists and technologists.

# Neurons and action potentials

## What is the brain made of?

One answer to this question is that the brain is made of cells; There are other answers: the brain is made of atoms, the brain is made of circuits and so on, but in this introduction we will first concentrate on the cells.

Many of the cells in the brain serve a sort of support role, they hold things in place, help with metabolic processes and maintain the brain as a living organ. These cells, many of which are **glial cells** may well play some role

in neuronal computation; but here we will focus on the **neurons**, the cells which are directly involved in the cell-to-cell signaling we believe is responsible for computation in the brain.

We will be looking in detail at how neurons signal; our aim here, however, is to get an overview of how the story fits together to make it easier to understand the more detailed discussion when we get to it. In Fig. 1 there is a diagram giving the main parts of a neuron. In the center there is the **soma**, the neuron is a cell and the soma is the cell body; here many of the metabolic processes, the life-supporting, functions of the neuron occur, it contains, for example, the nucleus where the genetic material is found and which controls the synthesis of proteins.

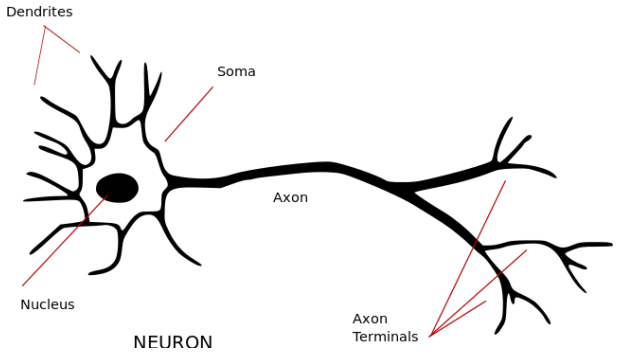



Figure 1: **A diagram of a neuron**; this cartoon shows the main parts of the neuron, very roughly, the dendrites, where the signals come in, the soma where they get added together and the axon, which carries signals on to other neurons. Figure from `wikimedia`

There are two sorts of tubes extending from the soma, the **dendrites** and the **axon**. These really are tubes, though it is tempting to think of them as wires: the neuron has an inside and an outside separated by a membrane. Both inside and outside the neuron there is fluid, basically water with dissolved salts, but the fluid inside and out have different concentrations of the ions that make up the salts. There can be a potential difference, that is a voltage difference, across the membrane and, as we will discuss, the signals we are will be talking about are changes in voltage.

Roughly speaking (and this is a very rough claim) the signals to the neuron come in along the dendrite, and the signals the neuron sends go out along

the axon. The axon of other neurons will transmit signals to this neuron's dendrites at points where they, the other neuron's axon and this neuron's dendrite, nearly touch: the nearly-touching place is called a synapses and synapses contains complicated bio-mechanical machinery to allow the signals to be communicated from axon to dendrite. We will discuss this further soon.

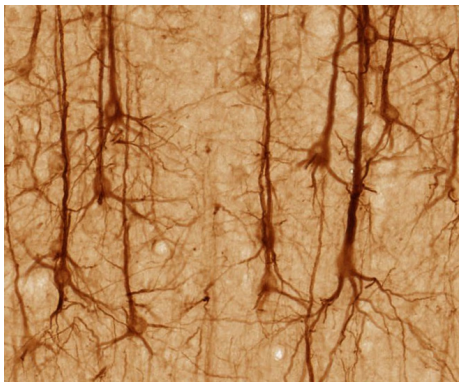


Figure 2: **A picture of some neurons**; these show the pyramidal neurons found in the cortex, part of the brain; they are called pyramidal because of the shape of the soma; you can also see the large number of dendrites. Figure from `wikipedia`

In Fig. 2 is a picture of some neurons; it would be a mistake to think that all neurons are similar to each other. There is a huge diversity of different neurons, they differ in their shape, in their size, in how connected they are and in their voltage dynamics. As an extreme example, the Purkinje cell (see Fig. 3) has a huge number of dendrites and receives connections from as many as 100_000 other neurons, however, many of those other neurons are cerebellum granule cells, very small neurons that receive inputs from only three or four other neurons.

It is tempting, as a computer scientist, to think of neurons as a sort of universal circuit component; this is a mistake. However, there are, nonetheless, aspect to the description of neurons that are reasonably general. The first of these reasonably general aspects we have already covered: signals come in the dendrites, accumulate in the soma and go out the axon. There is a more general aspect we have already alluded to: the signals correspond to changes in voltage.

Figure 3: **Drawing of a Purkinje cell**; this is a drawing of a Purkinje cell, by Santiago Ramón y Cajal, an important neuroanatomist active in the late nineteenth and early twentieth century. Purkinje cells are found in the cerebellum. Figure from `wikipedia`.

## Action potentials

There is a potential difference between the inside and outside of a neuron. For convenience we usually regard the fluid in the brain as being at a zero voltage; Relative to that the fluid inside a neuron has a negative voltage; -70 mV at rest would be a typical value.

You might think this makes no sense, if there is a voltage difference between the inside and outside of the neuron, surely a current would flow between the two equalising the voltage difference?

There are a number of reasons that doesn't happen, firstly the membrane is an insulator, largely preventing the flow of current, secondly, the situation is, as we will see, more subtle. Not only is there a difference in voltage across the membrane, there is a difference in the concentration of ions with, for example, an excess of sodium ions outside the cell relative to inside and an excess of potassium ions inside relative to out.

Along with the voltage differences, these concentration differences also have

the potential to cause ions to flow into or out of the neuron. Finally, there are pumps, minuscule molecular machines, which pump ions in and out of the neuron to help maintain the voltage difference. It is, all-in-all, a complicated story. For now, what we need to know is that there is a voltage difference across the cell membrane. Why is this important? It is important because the signalling dynamics of a neuron is voltage dynamics: signals are carried by what are called **action potentials** or **spikes**: these are spikes in the voltage that travel along the axon.

A picture of an action potential is shown in Fig. 4. During a spike the voltage shoots up by about 80 mV and then falls back to near the resting value, all during 1–2 ms. The dynamics that allow this to happen come from ions travelling through the membrane, in a sense the energy for the spike has been stored up by the all ion pumping that has created the concentration differences across the membrane. The spike will travel along the axon. The axon will usually have many branches and when this happens a spike will travel down each branch: the spike doesn't split in the sense that the spike travelling down each branch will be the same size as the original spike.

Broadly speaking, the spike does not change amplitude or shape as it propagates along the axon. A useful analogy here is a train of dominoes falling over; the energy in that case comes from the energy stored in the domino when it was set upright, the collision from the other domino hitting it is what causes it to fall over, but isn't the source of most of the energy involved in its own fall. When a train of dominoes splits, the wave of falling-over is just as fast along each branch.

# 1 Summary

So far we have begun a broad overview of neurons and what they do; this is all going to be revisited in more detail later. In particular we have described the division of neurons into dendrites, soma and axons and have seen that signals propagate along the axon in the form of voltage spikes.

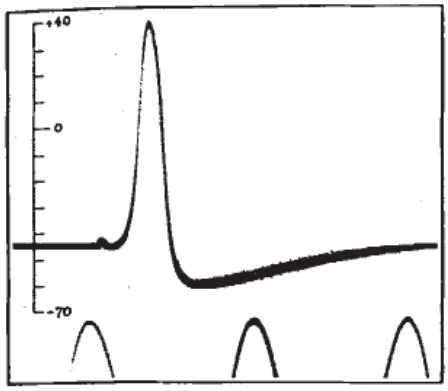



Figure 4: **An action potential**; this is an action potential recorded from an axon, it is a very early recording performed by Hodgkin and Huxley, pioneers in recording and modelling action potentials; this picture is actually a photograph of an oscilloscope trace.

# Synapses

In the previous section we discussed action potentials, the voltage pulses that travel along axons and carry signals from neuron to neuron. What we didn't discuss was how the signals got from the axon of one neuron to the dendrite of the next. The answer is synapses.

In discussing the action potential we mentioned that it is a largely stereotypical signal, the action potentials from a neuron all have roughly the same amplitude and shape. The synapses, in contrast, are diverse. They have different strengths, which can change over time. Beyond these variations in strength, synapses can have different dynamics, with differences in the time course of how the synapse responds to a spike, or in how the synapse responds to spikes coming in quick succession.

The complicated and variable behaviours of synapses are possible because synapses do not simply connect the axon to the dendrite, they are not simply holes or pores through which ions flow. In fact, there are synapses, called **gap junctions** that are like that, these gap junctions are the only synapses in some simple creatures like jellyfish and are found in the mammalian brain. However, the synapses we usually have in mind are **chemical synapses** in which the signal transfer from **pre-synaptic** axon to **post-synaptic** dendrite involves a complex chemical cascade.

## Chemical synapses

In Fig. 1 is a cartoon of a synapse. Note that the dendrite and axon don't actually touch; there is a little gap, called the **synaptic cleft** between the two. Of course, the axon and dendrite are held together by glial cells, cells which also surround the cleft and keep everything in place.

Importantly, electrical charge does not flow directly from dendrite to axon. When a spike arrives at the synapse it triggers a chain of events. The sudden change in voltage opens channels in the membrane of the synapse that allow calcium to flow into the **terminal bouton**, the part of the synapse on the axon side of the cleft (it is an amazing property of neurons that they contain ion-specific channels). Calcium flows into the synapses and, by way of complicated chemical reactions, this causes some little ($\approx$40 nm) membrane-bound spheres called **vesicles** to fuse with the wall of the cleft and burst, re-

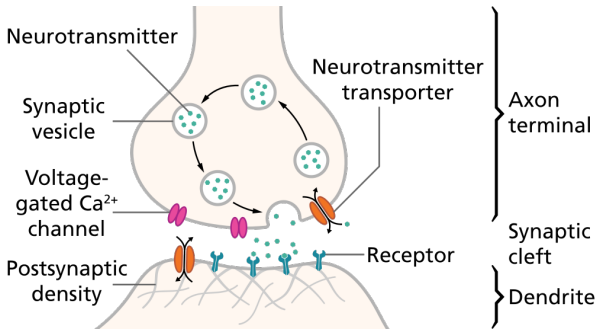



Figure 1: **A diagram of a synapse**; this cartoon shows the main parts of the synapse; the axon of the pre-synaptic neuron is coming in from the top, the dendrite of the post-synaptic neuron is leaving through the bottom. Figure from `wikipedia`

leasing specialised molecules called **neurotransmitters** into the cleft.

These neurotransmitters, in turn, bind with channels ("receptors") on the opposite face of the cleft, that is, with channels in the membrane of the post-synaptic dendrite. These channels then open in response to binding with the neurotransmitter. Depending on the type of synapse, they either allow ions to flow into, or out of, the dendrite, either increasing its voltage, or decreasing it. As a crude first approximation, we might summarise the immediate effect of a synapse as either **excitatory**: increasing the voltage of the post-synaptic neurons, or **inhibitory**: decreasing it.

Each type of synapse has different channels in the dendritic face of the cleft and different neurotransmitters that bind to these channels. In an excitatory synapses these channels allow sodium ions into the cell; Since sodium ions are positive ions this increases the charge inside the cell. Some excitatory channels may also allow influx of positively charged calcium ions. In an inhibitory synapse the channels either allow chlorine ions into the dendrite (chlorine ions are negative so this decreases the voltage), or they allow potassium ions out of the cell (potassium ions are positive, so this decreases the voltage).

Ion channels that open because they have bonded with a neurotransmitter are called **ligand-gated channels**; a ligand is a molecule that binds to things. These channels act as gates, sometimes allowing ions through and

sometimes not and they do so depending on whether or not they are bound to a ligand. This is in contrast with the **voltage-gated channels** that open and close depending on voltage. We will see later that voltage gated channels are important in understanding how spiking happens.

The binding between ligand and ligand-gated channel is quite loose and the molecules are bathed in a warm fluid; random Brownian movements will unbind the ligand allowing the channel to close. The timescale for this unbinding is different for different channels; for typical excitatory synapses it is of the order of tens of milliseconds. The neurotransmitter in the cleft is also cleared away by little pumps; "reuptake pumps", and is repackaged into vesicles ready for the arrival of later spikes.

The final part of this story is the dendrite itself. Consider first the situation with an excitatory neuron: Current flow into a neuron increases its voltage; This ramps up as more and more ligand-gated channels open before decaying away again as they close. The resulting positive pulse is called an **excitatory post-synaptic potential** or EPSP. For an inhibitory synapse the pulse is negative and is called an **inhibitory post-synaptic potential** or IPSP. Example PSPs are showing Fig. 2.

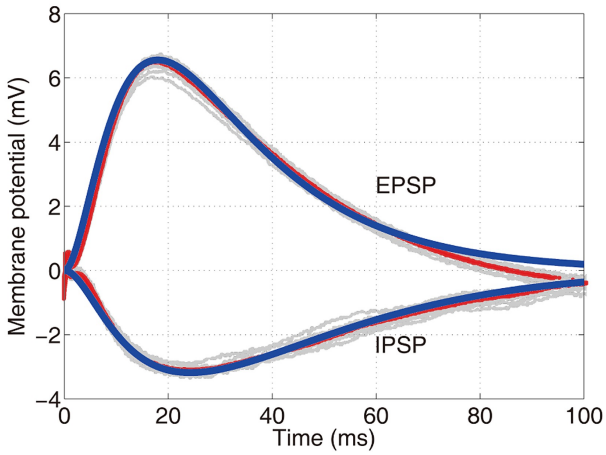



Figure 2: **Some PSPs**; the grey lines are recordings for individual PSPs with the red lines giving averages and blue lines demonstrate a model; we will look at simple models of PSPs later. Figure from [**?**]

In general dendrites lack the voltage-gated channels needed for action po-

tentials and the PSPs in dendrites propagate passively towards the axon. Dendrites are typically shorter than axons and thicker, so this conductance allows the change in voltage at the dendrite to propagate in to the soma. Even in the soma, where there are voltage gates channels an individual EPSP won't change the voltage enough to cause a spike. However, if lots of PSPs arrive at around the same time, the voltage in the soma will increase until it reaches a tipping point, around -55 mV is typical value of where this tipping point is. At the tipping point the opening of voltage-gated channels will cause a spike, usually at the point the axon joins the soma, and this spike will head off down the axon.

# 1 Summary

Here we described synapses; this is a complicated story we will go through again. When the spike arrives at the synapse some of the vesicles, little bags of neurotransmitter, fuse to the membrane of the synapse closest to the dendrite and burst, emptying neurotransmitter into the synaptic cleft. This, in turn, causes channels to open in the dendrite, changing the potential there. This change in potential, a PSP, flows into the soma causing a small change in the potential in the soma. If these small changes add up to push the soma potential to a threshold, the neuron spikes, sending out an action potential along the axon.

# Types of neurons and synapses

This short note emphasises an important fact about the brain: there are many different types of neurons and synapses.

## Dale's principle

In lecture notes 01.2 "Synapses", we noted that synapses come in (roughly) two broad classes: excitatory synapses that make the post-synaptic neuron more likely to fire a spike, and inhibitory, which make it less likely. (Some neurotransmitters have effects that can be described as **modulatory**; We touch upon this in the last subsection of these notes ). Typically, a single neuron will receive a mixture of excitatory, inhibitory, and modulatory inputs from many different sources. Perhaps surprisingly, the picture is usually simpler when it comes to the chemical signals that a neuron *emits* onward to other cells: Most neurons release the same set of chemicals from all terminal boutons on their axon, regardless of the identity of cell receiving these synaptic connections. This is known as **Dale's principle**

According to Dale's principle, the majority of neurons can be roughly classed as **excitatory neurons** whose outgoing synapses are all excitatory, or **inhibitory neurons** whose outgoing synapses are all inhibitory. It is important to understand that the division relates to the outgoing signals, neurons usually receive a mixture of excitatory and inhibitory signals.

In cortex, inhibitory neurons tend to be small and to have only local connectivity; they are also diverse with many different types: basket cells, stellate cells, fast-spiking PV cells, and many others. The excitatory neurons are usually larger, they tend to have local and distal connections and come in varieties of one main type: the pyramidal cell. It is tempting to think of the pyramidal cells as doing 'the work' and the inhibitory cells as helping modulate and sculpt the activity of the pyramidal cells. It remains to be seen if that is a useful way of thinking of what happens. In other parts of the brain there are circuits where the **principal cells**, the ones tasked with signalling outside of the region, are inhibitory.

## More types of synapses

Synapses are also classified by the sort of neurotransmitter they produce. For excitatory synapses this is almost always **glutamate**, a small amino acid. This does not end the classification; there are different receptor types for glutamate, the main ones are called **NMDA** and **AMPA**; the acronyms are short for complicated chemical names and we are, here, straying into more detail than we need. The main point is that NMDA and AMPA receptors have different behaviours, both in short-term in the time course of their binding to the transmitter, and in the longer term, in how they change in number and strength in response to what is happening at the synapse. A glutamate synapse will usually have a mixture of ligand-gated channels with NMDA and AMPA receptors.

For inhibitory synapses the most common neurotransmitter is called **GABA**, again, the acronym is short for a complicated chemical name. There are also two types of receptor here, but these are classes of receptors rather than distinct receptors as in the case of glutamate. The story for inhibitory synapses is more complex even than the story of excitatory synapses: the two classes are **ionotropic** receptors and **metabotropic** receptors. Metabotropic receptors trigger biochemical signals, rather than modulating voltage dynamics via the conductance's of ligand-gated ion channels.

Again, we risk straying into the massive complexity of synapses; the main point of this short subsection is to note that there is a complicated story and to, perhaps, suggest that the complexity of synapses and the huge range of behaviours in terms of the temporal behaviour and the different PSP profiles this produces is an interesting element of any attempt to compare learning in the brain and in computers.

## Neuromodulation

We have concentrated so far on electrical signalling; the part of neuronal computation that involves neurons sending spikes to each other. There is another system in the brain that is important for computation: neuromodulation. Neuromodulators are chemicals that can change the behaviour of a neuron or synapse; there are a lot of different neuromodulators, but the 'big five' are serotonin, dopamine, acetylcoline, histamine, and noradreneline (also called norepinephrine). Many other small molecules and peptides

have been identified as having neuromodulatory functions.

Neuromodulators are released at synapses by specialised cells that are usually found in specific brain regions but with axons that spread over the whole brain. Sometimes the neuromodulator is released to one post-synaptic cell, but often they are released into the extracellular fluid so that they affect a group of cells.

There are a large number of different receptors for the neuromodulators and these can have complicated affects, changing the excitability of a neuron for example, or prompting a synapse to change strength.

It is common to think of neuromodulation as adjusting the computational circuit, like a series of knobs and levers which can up- or down-regulate the computational dynamics and they do so over different timescales. These neuromodulators are very interesting because they seem to link the quite low-level details of how circuits work, they are produced by neurons in response to signals to those neurons, and high-level behaviours. Changes in neuromodulation can be linked to different decision-making strategies and can, it is thought, be experienced as mood.

# Parts of the brain - the cortex

The brain has different parts, different areas made up of different types of neurons arranged in different structures. This might not be so obvious looking at standard pictures of the brain, like the one in Fig. 1, you are mostly looking at the cortex, which forms the outer surface of the mammalian brain. Even the cortex, however, has different areas, as we will discuss in this section.

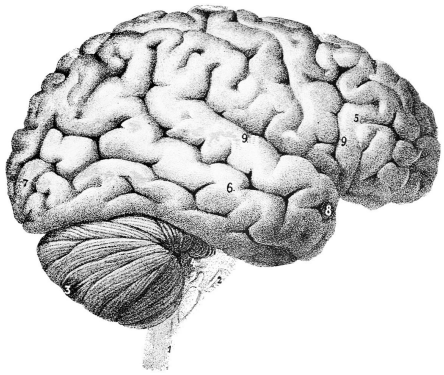


Figure 1: **A picture of the brain**; when we look at a human brain we mostly see cortex, though in this case we can also see the cerebellum, labelled 3. Figure from `wikipedia`.

The different parts of the brain have different functions, but we don't think any of these functions is 'thinking'. 'Thinking' is the function of the whole brain, the product of the cooperative action of all the diverse brain regions.

Here we will have a quick tour of different brain parts, their structure and their function. Some of these regions we will revisit in more detail, others not, but the main point here is to give a feeling for the variety across regions of the brain—as well as the types of specialisation they exhibit in their function and in how they perform it. This account will be somewhat mammal, or even human, focused.

## The cortex

The cortex, or more specifically the **cerebral cortex** forms the outer layer of the mammalian brain. In the human brain it contains 14–15 billion neurons

and the extent of the cortex is one of the distinctive feature of the human brain, Fig. 2.

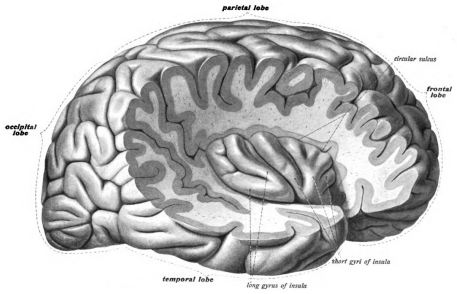



Figure 2: **The cortex**; the cortex, in grey in this picture, is an outer layer; in primate brains, and particularly in human brains, the cortex has become very folded in an effort to fit more in, space is short because we need a lot of cortex to be smart, and our heads are smallish because of constraints on human anatomy related to standing upright limit the size of head that a baby can have at birth. The grooves are called sulci, the ridges are gyri. Figure from `wikipedia`

The cortex is thin, two to five millimetres thick in humans and arranged in layers. Most of the cortex has six layers, Fig. 3, distinguished by different types of neurons and different connectivity patterns.

## Broca's area

Broca's area is perhaps not the obvious place to start when discussing brain regions, it is an area of the cortex responsible for some aspects of speech. However, it does make sense to discuss it because of its historical status in the history of our understanding of how different part of the brain are specialised to different functions.

Much of our early understanding of the brain came from studying patients with brain lesions. Modern neuroscience often seems very focused on animal studies and this has allowed us to make progress in understanding the brain in a precise way, but information from patients still helps guide our subject by linking brain function to our experiences and abilities.

Louis Victor Leborgne was a patient of the nineteenth century neurologist Pierre Paul Broca; he is usually known by the name 'Tan' because at 30 that

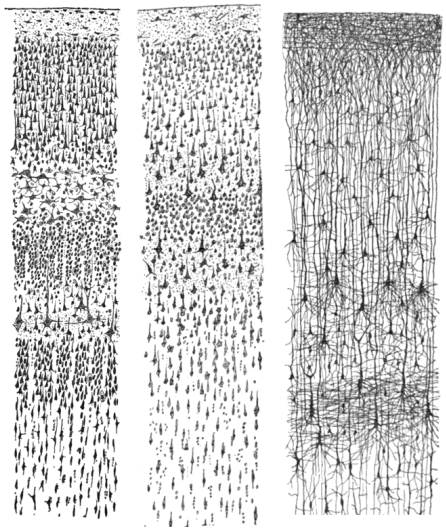


Figure 3: **Layers of the cortex**; in these illustrations by Santiago Ramón y Cajal we see different cortical layers, the middle and left how soma and the right the dendrites. When Cajal did his drawing he would stain the cells to make some visible, only a random subset take up the stain, otherwise they couldn't be easily distinguished by microscope and different staining chemicals showed different parts of the neuron. The left is visual cortex, the middle is motor cortex. The right panel also shows motor cortex and the figure is from `wikipedia`.

was the only word he could say. Interestingly, Leborgne could understand language and could express understanding and emotion with the intonation of how he said 'Tan', that was the only word he could produce.

There are two well-known fictional characters based on Leborgne, Garp, from John Irving's novel of the same name, who differs from Leborgne in his ability to understand speech and Hodor from *The Game of Thrones*, particularly in the HBO TV series; although Hodor's limited speech was ultimately attributed to nonsense magical events, he shows precisely the behaviour described above, he could understand speech and express emotion though intonation, but could not say any word but 'Hodor'.

Sadly Leborgne died, in 1861, soon after he was visited by Broca; an autopsy reveals that he had a very specific lesion in his brain caused by syphilis; a picture is shown in Fig. 4. This lead Broca to suggest that this area was responsible for the production of speech and that lesions to this part of the brain cause **expressive aphasia**, aphasia means a problem with speech and in expressive aphasia the problem is with the production of speech. This was confirmed by subsequent patient, including another studied by Broca, Lelong, who only had five words. Another part of the brain, Wernicke's area, was soon identified with **fluent aphasia**. A patient with fluent aphasia can have a large vocabulary but produces what is called 'word salad', lots of words included many which are not related to the subject of speech.

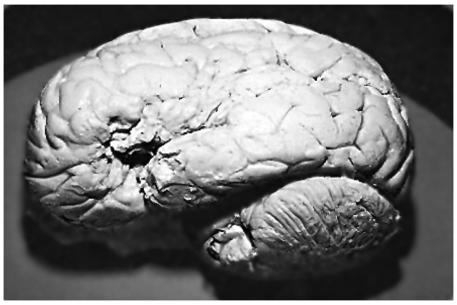


Figure 4: **A picture of the brain of Louis Victor Leborgne ("Tan") as studied by Pierre Paul Broca**. The lesion is clearly visible. Figure from `cambridge.org`.

These days, this neat division of the speech areas into Broca's area and Wernicke's area, and aphasia into expressive and fluent aphasia, is considered

too reductive; the function of these areas in language and their ability to respond to insult is complex. Nonetheless it is amazing to think that the linguistic ability, which we experience as a unitary ability, can be subdivided into different abilities and these abilities are localised in different areas. The neurologist Alexander Luria even discovered that his patient Lev Alexandrovich Zasetsky who received a severe brain injury in the second world war, had problems with nouns to a far greater extent than he had problems with verbs.

## The motor and somatosensory cortices

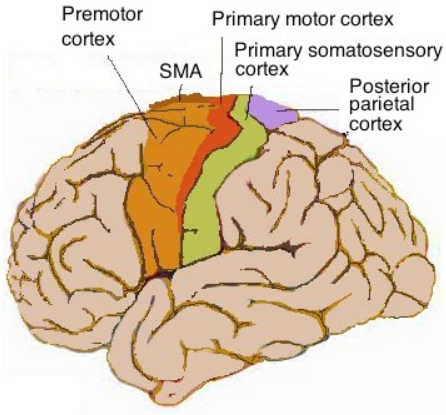



Figure 5: **The motor and somatosensory cortices**; the primary motor cortex is in red, the primary somatosensory cortex in green. Figure from `wikipedia`

Sticking to the cortex, in the first half of the twentieth century the neurosurgeon Wilder Penfield pioneered a technique in neurosurgery, still important today, of waking the patient up during the surgery and stimulating parts of the brain: because the brain itself has no sensory nerve endings this is painless, though it must be alarming. This allowed him to work out which parts of the brain could be operated on, or removed, while causing the least damage to crucial abilities.

As a consequence to this he discovered and mapped out the motor and somatosensory cortices, Fig. 5. Touching parts of the motor cortex cause movement in the patient, touching parts of the somatosensory cortex cause fictive

sensations. Different parts of each correspond to different parts of the body, see Fig. 6 and the amount of space dedicated to each part is proportional to how useful the information is, rather than how big the part is, as illustrated in Fig. 7.

This specialisation of the sensory cortex is not restricted to the somatosensory cortex, there are also parts of the cortex dedicated to vision and hearing; the story with smell and taste is more complicated. These are all primary sensory regions, they do initial processing on the sensory information, but further processing, further along the pathway, often includes the mixing of different pathways. It is easy to see this happens by thinking how much easier it is to hear what someone is saying if you can also see their lips, something we have all become familiar with during the pandemic.

## The cortex and memory

The gradual realisation that different parts of the cortex had different functions lead scientists to assume that memory, particularly long-term memory had a location in the cortex. This lead to a series of studies by Karl Lashley who taught maze tasks to rats and then lesioned parts of their cortex, see for example Lashley (1950) *In search of the engram*. He discovered that cortical lesions did affect rat performance in the maze, but that there was no

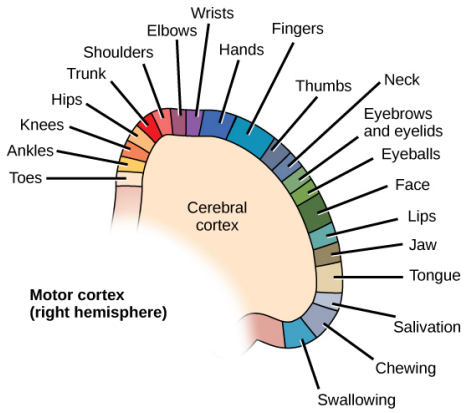



Figure 6: **The motor cortex**; the map from cortex to parts of the body for the motor cortex, a similar map exists for the sensory cortex. Figure from `wikipedia`

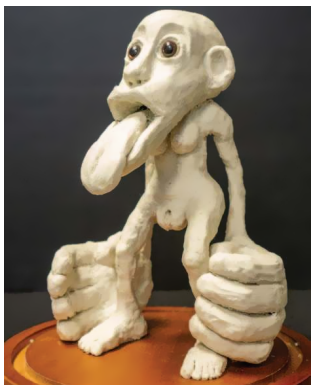


Figure 7: **The sensory homunculus:** A sculpture of a female somatosensory homunculus by artist Haven Wright, depicting the relative contribution of body regions to subjective awareness. In the scientific literature, you typically find male versions with smaller genitals, but that may be a Bowdlerisation. A female somatosensory homunculus based on neuroimaging is still lacking, possibly for the same reason, or possibly because female neuroanatomy is less well studied—a problem common in neurology, neuroscience and psychology. This image is discussed in this context in Wright and Foerder (2021). Sculpture ©Haven Wright. Photo ©Preston Foerder. Image from Wright and Foerder (2021) *The missing female homunculus*

clear relationship between where the lesion was and the consequence for memory, leading him to suppose that memories were randomly deposited across the cortex. We know now that accounts of cortical memory are not so straight-forward, there are areas of the cortex linked to certain types of memory and areas of the cortex more likely to store memories than others, but it does indicate that there is no easy way to describe the localisation of cortical function.

## Summary

The cortex is the outer layer of the brain, although it looks homogenous at first glance different areas have different functions. Based on a patient mostly known as Tan, Broca discovered an area of the brain associated with speech, damage to this area causes expressive aphasia. Conversely, damage to Wernicke's area caused fluent aphasia, a lack of meaning rather than a lack of words. The motor control and somatosensory perception of different parts of the body is also linked to distinct brain areas, one for moving your left pinkie, another for processing feelings from it, for example. In fact there are cortical areas for primary processing related to hearing and vision as well, but the localisation of memory is less clear.

# Parts of the brain - not the cortex

In the previous section we discuss the different areas of the cortex and saw that different cortical areas served different functions; away from the cortex the degree of specialisation is more profound though the precise role of different brain areas and how this role relates to its structure is often subtle, or indeed, poorly understood. In this section we will look briefly at a few examples; this is only a very brief tour, touching on only a small number of brain regions and those only briefly as an introduction to the idea that different regions have often precise, but difficult to pin-down, functions.

## The hippocampus

The hippocampus is found at the edge of the cortex, formally it is part of the so called allocortex; little was known about the function of the hippocampus until, sadly, a man called Henry Molaison, Fig 1, known as patient H.M., had his removed in 1953 in an effort to cure his epilepsy.

From the early thirties until the late forties there had been a fashion in psychiatry to perform a brutal surgery called the frontal lobotomy as a treatment for mental health problems; in a frontal lobotomy the frontal lobe of the cortex was severed from the rest of brain, often in a crude fashion using a sharp spike called a leucotome pushed into the brain from the corner of the eye-socket. This surgery was held to alleviate severe mental illness; in fact, it served only to make people with mental illness more passive, and thus easier to care for in a negligent manner, while actually grievously injuring them and bringing them no actual medical benefit. It is disturbing how widespread this surgery was and how casually it was performed; the Nobel Prize was even awarded for its discovery. In this context of this abusive tradition, William Scoville, the surgeon who operated on Molaison was considered very enlightened and, indeed, made other contributions to neurosurgery.

Molaison had intractable epilepsy, severe enough to leave him incapacitated. Scoville believed the fits were starting in the hippocampus. This was actually very prescient, resecting the hippocampus remains a useful approach to some intractable epilepsy, though modern operations are much more targeted than the surgery Scoville performed on Molaison: he removed most of the hippocampus, an area of the brain whose purpose was unknown at the

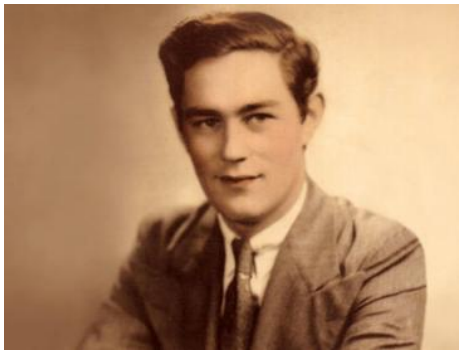


Figure 1: **A picture of Henry Molaison**; a photograph of Henry Molaison taken before his surgery. Photo from `wikipedia`

time. In fact, the tragic consequence was that Molaison was no longer able to form new memories. Surprisingly, Molaison preserved his memories of the past, and his short term memory, the memory of what had happened in a moving window measured in seconds, but he was unable to form new memories.

This has lead to the realisation that the hippocampus stores memories for minutes, days and weeks and is the memory store that supports quick memorisation and recall; it is the system we use to remember where we have left our book. This distinguishes it from longer term memory, memories of our childhood or information we find useful or evocative; these memories are stored in the cortex and are thought to be copied there, or **consolidated**, from the hippocampus. The two memory systems are thought to differ in how they store memories, reflecting the different priorities for each; in the hippocampus it is important not to mix memories up, in the cortex it is useful to link related information.

In fact, the realisation that the hippocampus was responsible for certain types of memory lead to a new type of experimental investigation which involves recording neurons in the brain of awake behaving animals, in this case rats. These experiments led to the discovery of place cells by John O'Keefe and Jonathan Dostrovsky in 1971. Place cells are cells which fire when the animal is in a particular position; they can be thought of as storing a memory for a particular place and their firing as part of the process of remembering that place. This is illustrated in Fig. 2.

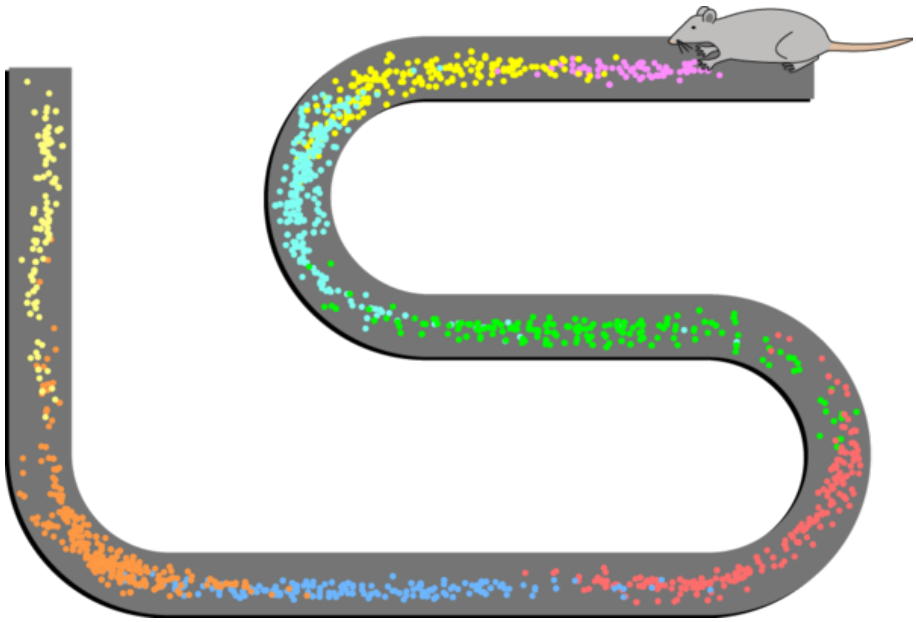


Figure 2: **Place cells**; each dot represents the position of the rat when one of eight different place cells fired a spike, the colours represents a different cell. Figure from `wikipedia`

## The cerebellum

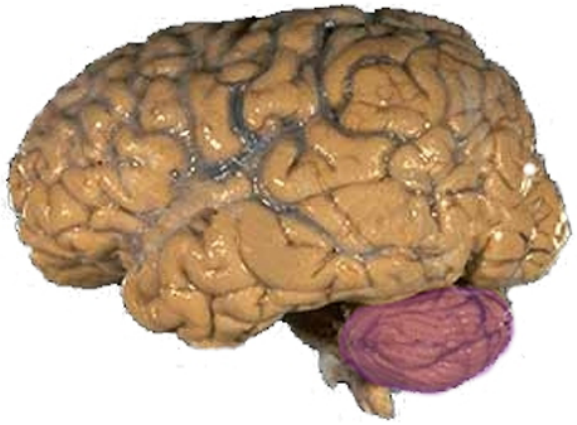


Figure 3: **The cerebellum**; a brain with the cerebellum marked in purple. Figure from the NIH

The cerebellum, sometimes called the hindbrain, is at the back of the head, Fig. 3, and has a very distinctive structure which is largely preserved across species. In contains the numerous cells of the brain, the cerebellar granule cell and one of the largest, the Purkinje cell. In order to decide the degree to which different functions are the preserve of different brain areas, the French neuroantatomist Jean Pierre Flourens performed a series of experiments in the early part of the nineteenth century in which he removed the cerebellum from living animals. He discovered that without a cerebellum animals were still able to move, but without their accustomed grace. This is consistent with what is observed for patients with cerebellar damage, see Fig. 4.

These days it is thought that the cerebellum performs very precise predictions in aid of movement control. A challenge in designing a control system is that the motor controls need to be based on a current estimate of position even though that estimate may not be available straight away because of the latency in sensory processing. A potential solution is to use a *forward model* to predict the current position from the available, past, sensory data and one common idea is that this is the role of the cerebellum.

Figure 4: **A woman with cerebellar damage**; this famous drawing from 1912 shows the distinctive gait associated with cerebellar damage, the patient is able to walk, but her walking seems self-conscious and awkward. Figure from `Wikipedia`.

## The basal ganglia

The basal ganglia are a complicated group of subcortical brain areas thought to act as a gate to decision making and as a centre for processing rewards and their association with actions. One important aspect of the basal ganglia is how they relate to dopamine, one of the neuromodulators; the basal ganglia include the substantia nigra one of the two main areas of the brain where **dopaminergic**, that is dopamine producing, neurons are found. The other area, the ventral tegmental area, or VTA, produces dopamine in reaction to rewarding events, or perhaps unexpectedly rewarding events; the prompt for release of dopamine by the substantia nigra is harder to summarise, but again, seems to be related to reward and reward-cued behaviours.

Parkinson's disease, a neurodegenerative disorder associated with stiff, even frozen, movements is associates with the loss of cells in the basal ganglia. It is believed that the basal ganglia provides a final 'go' signal allowing motor commands to travel to their muscle target and that with cell loss in the basal ganglia this 'go' signal does not occur. Providing L-dopa, a dopamine precursor, can alleviate this difficulty, though, of course, this broad uplift in dopamine is not a complete substitute for the more exquisitely modu-

lated provision of dopamine we can assume is provided by the substantia nigra

## Summary

After Henry Moliason had a surgery to remove his hippocampus, it was realised that the hippocampus has an important role in the sort of memory we use to recall where we have left our book or when we agreed to come home; this is also required for forming long-term memory. The hippocampus includes place cells, cells that fire in a particular location, presumably this place memory is a part of and possibly a precursor to a more complicated and abstract memory system. The cerebellum is found at the back of the head and is thought to help the brain control movements, possibly by predicting the consequence of motor commands. The basal ganglia play a role in decision making and Parkinson's disease is associated with loss of neurons in the basal ganglia.

# Recording from the brain

In the earliest days of philosophy and science, there was a belief that we could make progress in understanding the brain by thinking about thought, by a complex process of structured introspection. It is likely that this still has a role in our broad subject, but it is equally clear that the fact of science, the facts we can discover by observing the dynamics of neural matter, are vital to understanding the brain. The tools of neuroscience were for a long time very crude; as we saw, the main approach was to study patients while they were alive and then dissect them after they died. Now, though, we have a diversity of approaches to recording the dynamics of neural matter.

The ideal, obviously, would be to record all the voltage changes for all the neurons in the brain and, probably, the different levels of different chemical. That isn't possible and any approach to recording from the brain is a compromise between spatial and temporal resolution and between the degree of intervention: at one extreme ***in vitro*** experiments are done on slices of brain that have been removed from an animal, typically one that has been killed as part of the experiment, these experiments can record the voltage inside the neuron an sub-millisecond resolutions. At the other extreme *electroencephalography* is completely noninvasive, you might end up with conductive gel on your hair but beyond that, there is no ill-effect or annoyance to the subject, but the data are noisy and reflect a blurry average of the activity of many synapses.

This document is intended only as a quick tour of some recording techniques: the speed at which the field is advancing and the vast diversity of approaches means any description is only very partial and slightly out-of-date. I will only mention ways of recording brain activity, but this has been paralleled by technologies to distinguish different cells, for example, using dyes that transport backwards through a synapse allowing researches to see what cell is connected to which and technologies, such as optogenetics and Designer Receptors Exclusively Activated by Designer Drugs (DREADDs), which can switch on or off specific cell populations, allowing researchers to work out what different cells are contributing to network activity.

## In vitro electrophysiology

In **in vitro** electrophysiology a small piece of the brain is removed and placed in a dish where it is kept alive by careful control of chemical composition and oxygen level of the fluid the slice is bathed in. An electrode is then used to record from the cell; this electrode is often a thin glass tube filled with salty water. Some approaches puncture a neuron's cell membrane with this electrode, and measure the voltage *inside* the cell (*intracellular* recording). Intracellular recordings can be used, for example, to measure the size of an PSP, the dendritic voltage change at a synapse. Another class of approaches, **patch clamp**, does not puncture the cell. Instead, the tip of a glass electrode is suctioned onto the cell's exterior, making a seal (Fig. 1). A patch-clamp recording can measure the ionic currents flowing through membrane-bound ion channels and receptors inside the patched portion of membrane.

Slices are usually made in a way that preserves as much of the neurites (the axons and dendrites) as possible; this is easier in some parts of the brain than others, the hippocampus for example lends itself to *in vitro* approaches because it has a very laminar structure and a lot of what we know about synapses was discovered in hippocampus. However, the key point is that these neurons are not in their natural environment, they will have damaged neurites, the chemical and temperature environment is not normal and their input is nothing like what it would be in the brain.

One nice approach, which has been used a lot in studying the retina, is to lay the slice directly only an array of electrodes. This does not give **intracellular data**: the electrodes record from outside the cell, giving **extracellular data**, so the data are spike times rather than voltages, but the density of electrodes that are possible does give a complete picture of the network activity, albeit in this weird context.

## In vivo electrophysiology

The *vitro* in *in vitro* means glass, referring to the glass dish the brain slice is placed in. In contrast, the *vivo* in *in vivo* means living and *in vivo* electrophysiology is performed using living animals, either anaesthetised or awake and behaving. To do this an electrode is placed in the animals brain; in the anesthetized preparation a hole is cut in the brain, *trepanation*, and the electrode is inserted through that, or, in the awake behaving preparation, a *head stage*

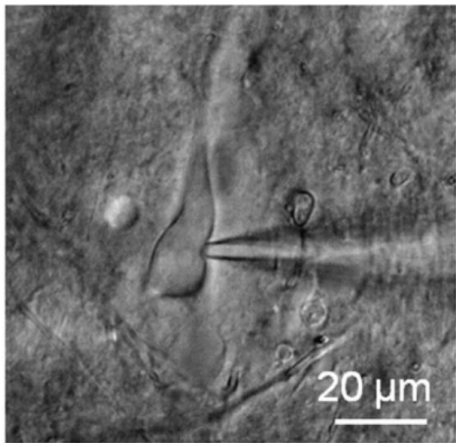



Figure 1: **Patched cell**: here an electrode has been *patched clamped* onto a cell to record its internal voltage. [figure from Ding, S., Mattam, S. G. & Zhou F. M. (2011)]

is mounted on the head covering and sealing up a hole in the skull. In the past the electrode was moved until the activity it recorded showed it was near a neuron; these days though silicon probes can be used. Silicon probes are made using processes similar to those used to manufacture silicon integrated circuits and have hundreds of recording sites; in this case the experimenter will rely on at least some of the recording sites being near the neurons.

In an **in vivo** experiment the electrode is outside the neuron so only the spikes are recorded and typically, spikes from more than one neuron are mixed together. A set of algorithms called **spike sorting** are used to assign spikes to different neurons. The accuracy of spike sorting is often debated. High-density silicon probes can help improve accuracy, since spikes are often detected at different amplitudes by electrodes at multiple locations. This provides a richer set of features that spike sorting can use to distinguish spikes from nearby neurons.

The advantage of **in vivo** electrophysiology is that you can record from the brain in a more natural state; however, the need to tether the electrode to the recording equipment in the awake behaving preparation limits the experiments that can be performed. Furthermore, the data recorded gives spikes, not voltages and spiking activity rather than synaptic activity. Although the

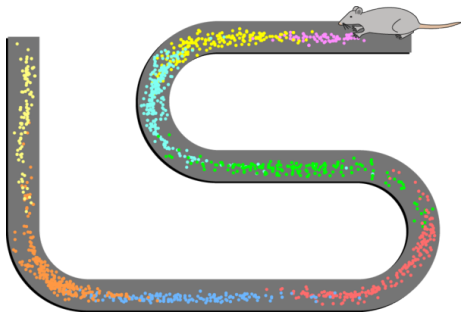


Figure 2: **Place cells**. Here each color of dot corresponds to a different neuron and the dots are placed on the maze in the location the rat was in when a spike in one of the eight neurons was fired. It shows clear evidence of place cells, with different cells firing in different locations. [figure from wikipedia]

number of neurons that can be recorded has increased a huge amount over the last few years, it is still a long way from recording whole brain activity. It can also be difficult to decide exactly what sort of neuron you are recording from, although different cells have different firing properties they are not always distinct enough to be distinguishable from their spike trains. It is an invasive procedure, there are animal welfare concerns and, except in very limited circumstances related to medical treatment, it is not possible to record from humans. Nonetheless *in vivo* electrophysiology has been one of the main tools of neuroscience over the last 50 years and has been the source a lot of the progress that has been made, see as just one example Fig. 2

## Calcium imaging

Although, for simplicity and for historical reasons, the story we tell about neuronal dynamics focuses on the movement of sodium and potassium ions, in fact, in mammalian brains calcium ions play a big role in spiking, synaptic signalling, and plasticity.

Over the last decade, it has become possible to measure calcium ion concentrations in neurons—using either genetic tools to make neurons produce calcium-sensitive fluorescent proteins, or special dyes to introduce into neurons. These chemicals bond with calcium and emit photons. This really is incredible; it is also very useful, using two-photon microscopy it is possi-

ble to measure neuronal activity. This allows a large number of neurons to be imaged and recorded and, often, allows the same cells to be recorded from over a long period or even across successive days; since the recording is linked to an image, it is often possible to identify cell type and to study development. Huge numbers of neurons can be recorded.

There are many constraints on calcium fluorescence imaging. For a start, calcium concentrations inside neurons change more slowly than the dynamics that produce spikes, so complicated **deconvolution** algorithms are needed to deduce spike times from luminescence. In addition, one must be able to record the emitted fluorescence. Experiments that record large numbers of cells (hundreds, thousands) therefore typically use a *head-fixed* preparation. For a rodent, the head of animal stands on a wheel or ball, free to move, but with its head fixed under the microscope and with a monitor in front of it: of this is rather grandly called a virtual reality experiment, but that amounts to little more than linking the picture on the monitor to the movement of the wheel or ball. Another, very impressive, example, involves the developing zebrafish, hatchling zebrafish respond to optical flow, the movement, or apparent movement, of the river bed. Since hatching zebrafish are transparent, a huge fraction of its neurons can be recorded at once. New methods of long-term calcium imaging using implanted microendoscopes and fibre optics are also under development.

The glory of calcium imaging is that you can see the activity of neurons spread across the piece of the brain that is being imaged, so rather than including an image here I urge you to go and look at videos on youtube.

## Electroencephalography

In electroencephalography (EEG) electrodes are placed on the scalp, usually of human participants and these are used to record the electrically activity in the brain. The story behind the discovery of EEG is amusing, you can read about it by looking up Hans Berger on wikipedia; in short, he wanted to explain what he believed was the real phenomena of telepathy. It is surprising that EEG works, the biggest electrically signals in the brain come from the activity in synapses, if all the synapses were in random directions these signals would all cancel out; it turns out that, particularly in cortex, there is some bias in the orientation in synapses, leading to an overall electrical field that can be measured at the scalp.

Needless to say, EEG signals are incredibly noisy, they mix together a plethora of neural activity and average a huge number of tiny electrical fields. The spatial resolution is terrible. However, the key advantage of EEG is two fold, it has good temporal resolution and it is easy to do, it is non-invasive, portable and relatively inexpensive. It was, for a long time, an important diagnostic tool; when I had childhood migraine for example, I was given an EEG; these days I would probably have an MRI. It has been of considerable help in studying epilepsy and lead to the discovery of sleep stages, the distinction between slow wave and REM sleep. I use EEG data in my own work on language since linguistic processing happens at short temporal scales and the only useful experimental subjects are humans. There is an example EEG trace, from a medical case report, in Fig. 3.

Magnetoencephalography (MEG), which detects magnetic fields rather than electrical ones produces better data, basically the magnetic fields aren't screened in the same way by the brain fluid; but MEG is much more expensive, the instruments used to measure magnetic field require superconductors and, thus, need to be cooled. There is optimism that a new technology, optically-pumped magnetometry, will lead to a cheaper and less cumbersome version of MEG.

## Magnetic resonance imaging

In magnetic resonance imaging (MRI) an oscillating magnetic field is used to shift atoms between different energy states, this causes them to release radio-frequency electromagnetic radiation, which can in turn be detected. By changing the oscillations, different sorts of things can be imaged; a structural scan can be used to get a very detailed image of the brain. This is very important in diagnostics, the brain can't be x-rayed; since neural matter has the same density as the fluid around it, by design of course to protect the brain, it is invisible in x-rays. As recently as the 1970 the only way to image the brain if a tumour was suspected was to introduce an air bubble into the skull and then x-ray the brain while trying by head movements to get the bubble in the right place. Another type of scan, Diffusion-Tensor Imaging (DTI), can find tracts of neuronal axons, showing how the brain is connected, see Fig. 4.

For our purposes, the most important type of MRI is functional MRI. In fMRI the radio frequency pulses are tuned to distinguish hemoglobin that is bound

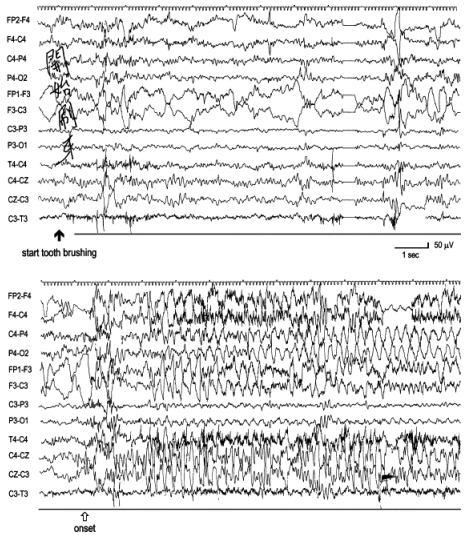



Figure 3: **EEG.** This shows the EEG traces from a woman who has epileptic fits, accompanied by orgasm, when she bushes her teeth; the moment she starts toothbrushing are marked by arrows and the over-synchonized activity typical of epilepsy is seen soon afterwards. [figure from Chuang, Yao-Chung, et al. «Tooth-brushing epilepsy with ictal orgasms." Seizure 13.3 (2004): 179-182.]

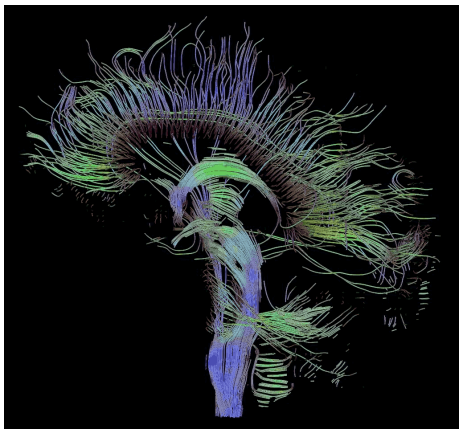


Figure 4: **Diffusion tensor imaging.** DTI allows us to image the connectivity of the brain. [figure from wikipedia, rendered by Thomas Schultz from data from Gordon Kindlmann and Andrew Alexander.]

Figure 5: **MRI machine.** A patient about to enter an MRI machine; the experience is a little claustrophic and very noisy, the magnets make a noise, but not unpleasant. [figure from radiology.ucsf.edu.]

to oxygen from hemoglobin that is not. This gives what is called the BOLD signal, a measure of the level of oxygenated blood in a part of the brain. This is believed to track neuronal activity, so fMRI can show which parts of the brain are active while the participant performs a task: the task of course being constrained by the MRI machine itself, the MRI machine is a huge cylinder with the subject, or patient, slotted into a hole at its centre; if you've never had an MRI the machine will be familiar to you from TV where it is often used as a short hand for 'serious medical stuff', see Fig. 6. The fMRI images have poor temporal resolution, basically the bold signal is giving the average activity of the brain over about three seconds; the spatial resolution, however, is pretty good; each voxel, the area resolved by the machine, is a cubic millimetre.

## Summary

All approaches to recording from the brain are a trade-off between how much of the brain is recorded, how good the resolution is, how invasive it is, how much it constrains any behaviour and how much it costs. *in vitro* physiology measures in the voltage inside the cell, but only for brain slices, *in vivo* electrophysiology measures spikes in the living brain, often in awake and behaving animals, the number of cells that can be recorded from is limited but increasing. Calcium imaging records from a huge number of cells, but the experiments are difficult and the behaviour constrained. EEG and fMRI experiments can be used with human participants, the first has poor spatial resolution, the second poor temporal resolution and both are a long long

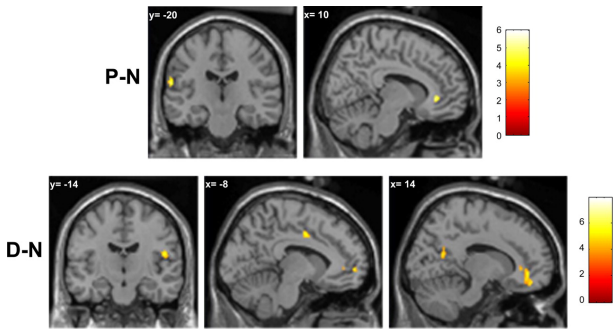



Figure 6: **fMRI images**. An fMRI study to distinguish the parts of the brian responding to disgusting or painful looking images [figure from Benuzzi, Francesca et al. Does It Look Painful or Disgusting? Ask Your Parietal and Cingulate Cortex, Journal of Neuroscience, 28 923-931]

way from recording from individual neurons.

# Computing

These are getting started notes on Python, the name of the corresponding example program is given in brackets in a fixed width font (`fixed width font`) just before the code listing.

## Computing in Computational Neuroscience

People in computational neuroscience usually use either MATLAB or Python or they use a specialised simulation tool such as GENESIS or NEURON or NEST. These simulation tools are optimised to efficiently simulate large complicated neurons, in the case of GENESIS or NEURON, or large and complicated networks of neurons, in the case of NEST. The simulation tools have been written over many years and can be quite complicated and idiosyncratic, however, NEURON at least, now has a Python interface.

A few years ago MATLAB had some advantages, until recently it had a very large user base across many parts of applied mathematics, a huge collection of libraries and package. Matlab uses a matrix paradigm which is easy to use and quick when you get used to it. Python is a proper programming language, with a nicer language structure than Matlab, it is free and open and has many libraries, though perhaps not as many as Matlab for some specialist applications, like wrangling EEG data, but more for machine learning applications. For this reason, MATLAB is quickly disappearing and the field is moving to the jupyter languages: Python, Julia, and R.

Other commonly used tools include Brian, a Python based package which supplies efficient specification and integration of differential equations used in neuronal simulations, XPP which is useful for analysing the dynamics of the differential equations used in neuroscience, NeuroML which is a metalanguage for describing neuronal models and R which is used for statistics.

These notes are only the roughest of introductions to Python; Google and StackOverflow are your friends. Conor Houghton has a website with some hopefully fun coding puzzles if you'd like to try it:

- `www.choicetask.com/python`

If you do try it, please forward any feedback and suggestions for new pages to

conor.houghton@gmail.com.

## Python - getting started

Python either runs as a script or on an interpreter. To get the interpreter you type `python` or `python3` on the command line and you get something that looks like this:

```
Python 3.10.12 (main, Jun 11 2023, 05:26:28) . . .      1
Type "help", "copyright", "credits" or "license" . . .  2
>>>                                                     3
>>>                                                     4
```

where the numbers on the right are just for these notes. The '3.10.12' is the version number. Up until April 20, 2020, two incompatible versions of Python: Python 2 and Python 3, were still in use. Since April 20, 2020, Python 2 has ceased development and all code you will use in this course will be in Python 3.

You can use python on the interpreter as a glorified calculator

```
>>> 5+6                                 1
11                                      2
>>> 12/9                                3
1.3333333333333333                      4
>>>                                     5
```

It has big numbers, which can be useful sometimes

```
>>> 2**100                              1
1267650600228229401496703205376         2
```

To get special functions and so forth you need to `import` the math package

```
>>> import math                         1
>>> math.tan(math.pi/4.0)               2
0.9999999999999999                      3
```

The syntax for the python math package is more or less the same as for `math.h` in C or `cmath` in C++. Notice there is a namespace, you need to write `math.tan(math.pi/4.0)`.

If you want to import the functions and so on into your namespace you use `from`, eg

```
>>> from math import tan
>>> from math import pi
>>> tan(pi/4.0)
0.9999999999999999
```

and, if you want to import everything from `math` into your current namespace it is

```
>>> from math import *
>>> sin(pi)
1.2246467991473532e-16
```

Of course, we don't want to use it as an interpreter for anything very complicated, so we write scripts. However, as an interpreted language you don't compile it, so if you have a program called `foo.py` you write

```
> python foo.py
```

to run it, here the '>' denotes the command prompt. Alternatively, you can add a `#!` (pronounced "shebang") to `foo.py`. In my case I would add `#!/usr/bin/env python3` as the first line of `foo.py` and change `foo.py` to executable by typing `chmod u+x foo.py` and then I can run `foo.py` directly

```
> ./foo.py
```

The initial `#!/usr/bin/env` uses the `env` command to find the appropriate python executable on your system, and this command should exist on POSIX compliant environments (MacOS and most Linux distributions). You can view the location found by `env` with the shell command `which python3`. You may need to change these paths to point to the python executable on your machine using this output of `which python` or `which python3`. If you use the command `python`, verify that it points to `python3` rather than an older installation of`python2`. If you are using a different setup, you may need to specify the full path to your python3 interpreter executable in the `#!` line.

### Python - some basics

There are lots of introductions to Python on the web and that's probably the best place to look. Variables are not declared (`hello_world.py`)

```
hello = "hello world"
print hello
```

prints hello. It has a nice slice notation (`hello_world_slice.py`)

```
hello = "hello world"
print(hello[0])
print(hello[1:4])
print(hello[-1])
```

prints `h`, `ell` and `d`, `hello[-1]` gives the last character.

One of the most distinctive features is that blocks are created by indenting rather than brackets. In (`hello_world_for.py`)

```
hello = "hello world"

for a in hello:
    print(a,end='')
print()
```

the block for the for loop is `print(a)`, because of the indent, note the colon after the `for` statement too. This sort of for loop, looping over parts of an object, is considered more pythonic than the indexed loops in C++ and so on, this is possible though (`hello_world_indexed.py`)

```
hello = "hello world"

for i in range(0,len(hello)):
    print(hello[i],end='')
print()
```

If you do want the index and the object it refers to you should use an enumeration
(`hello_world_enumeration.py`):

```
hello = "hello world"

for i,a in enumerate(hello):
    print(i,a)
print()
```

The if statement is also indented, well all blocks are, but for completeness (`if.py`)

```
import random

a=random.randint(1,3)

if a==1:
    print('one')
elif a==2:
    print('two')
else:
    print('three')
```

For some reason what are called arrays or vectors in other languages are called lists in python (`if.py`)

```
hello = "hello world"

letters = []

for a in hello:
    letters.append(a)

print(letters)
```

Lists need not be heterogenous (`hetro_list.py`)

```
list=[72,79,86,96,103,"Cathedral Parkway"]
print list
```

Python also has a tuple type, like a list but immutable. The `enumerate` we saw above makes an `enumerate` object, but you can cast it to a list of tuples (`enumerate_object.py`):

```
hello = "hello world"
print(enumerate(hello))
print(list(enumerate(hello)))
```

or

```
print([*enumerate(hello)])
```

One thing about Python that is hard to get used to if you are used to C or C++ is that what I keep calling variables sometimes behave as *references* or *pointers*. They name memory locations rather than pieces of data. This often catches me out, to see the difference look at (`labels.py`)

```
a = [0,1,2]
b = a

print(a)
print(b)

b[1]=-1

print(a)
print(b)
```

and note that changing `b[1]` has changed the value of `a[1]`.

To complicate matters, Python allows *shadowing* of variable names. Ra-assigning an existing symbol binds the symbol to a new memory location, leaving the old one untouched. For example, the following code

```
a=b=[1]
print(a,b)
a[0]=2
print(a,b)
a=[3]
print(a,b)
```

outputs

```
[1] [1]
[2] [2]
[3] [2]
```

In the above code, the symbols `a` and `b` are initially references to *the same* list, which contains a cell with the value 1. Changing the contents of this cell by assigning to `a[0]` also changes `b[0]`. However, when we create and bind the

new list `[3]` to `a`, this new list now behaves separately from the one bound to `b`.

### Python - functional features

Python has some stylish functional programming feactures for working with lists, for example `map` does a function on all the elements in a sequence so (`map.py`)

```
def cube(x):
    return x*x*x

numbers = [1,2,3,4,5]

cubes = map (cube,numbers)

print(cubes)
```

where you should also note the syntax for defining functions, there are functions and classes in the usual way, though classes have no private members. In fact this code can be made more streamlined, there is a construction for avoiding giving names to things that don't need to be named (`map_lambda.py`):

```
numbers = [1,2,3,4,5]

cubes = map (lambda x:x*x*x,numbers)

print(cubes)
```

In fact, there is also a list comprehension type construction that also does the same thing

```
numbers = [1,2,3,4,5]

cubes = [x*x*x for x in numbers]

print(cubes)
```

Python contains a number of these functional commands in addition to `map` and a number of different data structures, part of the skill of writing 'pyhtonic' code is to make these work for you.

The class construction has a few annoyances; for example each class can only have one constructor, which is defined using the `__init__` function (`class_example.py`).

```
class Counter:

    def __init__(self,value):
        self.value=value

    def add(self,increment)
        self.value+=increment

counter=Counter(5)
print(counter.value)
counter.add(7)
print(counter.value)
counter.value+=3
print(counter.value)
```

Member variables have the `self` prefix and notice that if a class function uses a member variable then `self` has to be one of its arguments. Finally, as mentioned before the member variables are not private.

### Scientific calculation packages

Python has a huge number of packages. For an introduction to some of them in the context of a very annoying / fun puzzle try `http://www.pythonchallenge.com/`. Here we will quickly introduce packages commonly used in scientific computing

- *PyTorch* This is a machine learning library, but it also adds matrices and vectors. It is quickly taking over form Numpy.
- *Numpy* This adds matrix and vector datatypes and enables fast vectorized calculations, it also includes methods for finding eigenvectors and eigenvalues.
- *Scipy* This adds various numerical routines for working out things like integrals and the solution to differential equations.
- *matplotlib* Plots stuff.

Here `numpy` is imported with the name `np` (`np_array.py`)

```
import numpy as np
d1=np.array([1,-1,1])
d2=np.array([1,2,1])
print(d1)
print(d2)
print(d1*d2)
print(np.dot(d1,d2))
```

so we define two numpy arrays, we see that * gives element-by-element multiplication, whereas `np.dot(d1,d2)` gives the dot product. We can do matrix multiplication as well (`np_matrix.py`)

```
import numpy as np
d1=np.array([(1,-1,1),(1,2,1),(1,-1,1)])
print(d1)
d2=np.array([1,3,1])
print(d2)
print(np.dot(d1,d2))
```

and lots of linear algebra (`np_det.py`)

```
import numpy as np
d1=np.array([(1,-1,3),(1,2,1),(3,1,1)])
print(np.linalg.det(d1))
```

As for `matplotlib` here is a simple example (`plot.py`)

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10)
plt.plot(x, np.sin(x), linewidth=2)
plt.savefig("example.png")
plt.show()
```

It both saves the plot as `example.png` and shows it on the screen.

Matplotlib also provides a module called *pylab* that is designed to resemble Matlab's API. You may see code that uses `from pylab import *` to import

all functions from matplotlib.pyplot, numpy, numpy.fft, numpy.linalg, and numpy.random, and some other things. This shadows some builtin functions like `sum`, which can lead to strange bugs. Although usually discouraged, pylab can provide a simpler interface for interactive data exploration if you are aware of these issues.

## Julia

Julia is a compiled programming language that runs much faster than Python; it is possible to write fast code in MATLAB where everything is written as linear algebra or using carefully optimized numpy or fancy just-in-time compilation in Python. The idea of Julia is that it runs at C-like speeds for natural, modern-looking code. It does this by allowing, while not requiring, type declaration and by not having classes, instead it has *types*, a bit like *structs* in C, and multiple dispatch.

It has other features to make it useful for scientific computing, little things like being able to `2v` when you mean `2*v`, along with big things, like a sophisticated multi-dimensional array datatype that can be used for efficient matrix operations. Presumably to help persuade people to finally abandon MATLAB it has a MATLAB-like syntax, blocks are denoated using a `end` keyword, the first element of an array `a` is `a[1]` and `1:10` means one to ten, not one to nine.

If you are used to Python, Julia can seem frustrating to debug, mostly because the typing can be hard to get used to, but debugging Julia as a Python programmer really reminds you how often you cast variables without even noticing; this is part of why Julia is much faster.

This only outlines the simplest parts of the language, the wikibook

> https://en.wikibooks.org/wiki/Introducing_Julia/

is a good place to look for a longer introduction. There is online Julia at

> https://juliabox.com/

### A simple example

Here is a programme to add powers of two (`add.jl`):

```
highest_power=10

value=1.0::Float64
current=0.5::Float64

for i in 1:highest_power
   value+=current
   current*=0.5
end

println(value)
```

**Line 1** defined `highest_power`; this is dynamically typed, as in Python, but `value` and `current` are given a type, `Float64`; as an indication of how seriously it takes typing, is **line 3** was `value=1::Float64` it would return an error since `1` isn't a `Float64`. You can find a full list of types in the wikibook, it has lots of different int and float types, along with rational numbers using `//` to seperate numerator and divisor (`rational.jl`):

```
a=2//3
b=1//2
println(a-b)
```

## Arrays

Arrays are what Python calls lists, python is the odd one out here, array is a more common name. Julia has the same slicing functionality as Python, although as mentioned above indexing is different (`slice.jl`)

```
a=[1,2,3,4,5]
println(a[1:3])

for i in a
   println(i)
end
```

prints `[1,2,3]` from **line 2**, **line 4** to **line 6** demonstrates a for loop. The last element in an array is indexed `end` so in the programme above `a[end]` is 5.

Arrays can store mixed items, but the array can be typed, so `a` in (`typed_list.jl`)

```
a=Int64[1,2,3,4,5]
push!(a,6)
println(a)
```

can only store items of type `Int64`. `push!` pushes an item onto the list, like `append` in Python, again, this is the more common notation. The `!` is part of an convention where all commands that change an array have a `!`.

As mentioned above, Julia arrays can be multidimensional and have matrix like operations, but this won't be explored in this brief overview. There is also a tuple type which is immutable.

### Functions

Here is a programme with some functions (`functions.jl`)

```
function add_to_int(a::Integer,b::Integer)
         println("int version")
         a+b
end

function add_to_int(a::Real,b::Real)
         println("float version")
         convert(Int64,a+b)
end

function add_to_int(a,b)
         println("what are these things")
         0
end

println(add_to_int(12,6))
println(add_to_int(12.0,6.0))
```

Obviously this is a very artificial example, but it shows some of the features of functions, first, their return value is the most recently evaluated expression and second, and more importantly, they support multiple dispach; the function is chosen to match the type of the arguments, here there

is one function for `Integers`, this is a supertype which includes, for example, `Int64`, there is one for `Real`, the supertype that includes various floats, and one with no type; the correct function is used for each. If there is no correct fuction there will be an error.

There is also a terse 'mathematical functions' style function syntax that is useful, well, for functions that do the sort of things mathematical functions do (`math_fxn.jl`)

```
f(x,y)=2x+y

println(f(1,3))
```

You can also return more than one value (`multiple_return.jl`)

```
function powers(x)
         x,x^2,x^3
end

multiples(x::Float64) = x,2x,3x

a,b,c=powers(2)

println(a,' ',b,' ',c)

a,b,c=multiples(2)

println(a,' ',b,' ',c)
```

### Composite Types

Julia doesn't have classes, this comes as a surprise at first, but it does have composite data types that work like structs, combining this with multiple dispach captures important parts of the functionality of classes, in a way that supports fast code (`struct_example.jl`

```
mutable struct Cow
        name::String
        age::Int64
```

```
end

mutable struct Poem
        name::String
end

function move(cow::Cow)
         print(cow.name," walks forward")
         println("showing the weight of her ",cow.age," years")
end

function move(poem::Poem)
         println(poem.name, " moves us to tears with its beauty")
end

poem = Poem("The Red Wheelbarrow")
cow = Cow("Hellcow",42)

move(cow)
move(poem)
```

You can see that although the structs have no methods, the function `move` can have different meaning for the two different data types. The default constructor defines the variables in the order they appear, it is possible to define other constructors, but that won't be considered here.

You can write constructors for these structs as ordinary functions; you can also use the key word `new` to write internal constructors, these serve, typically, to impose constraints on the input, see `constructor.jl`

```
struct Joke

    question::String
    answer::String

    function Joke(question::String,answer::String)
        if question[end]!="?"
            question=string(question,"?")
        end
```

```
        new(question,answer)
    end

end

function make_joke()
    Joke("what weapon does a fat jedi use","a heavy sabre")
end

joke=make_joke()

println(joke.question)
println(joke.answer)
```

### Making functions

Functions are objects just like any other, so they can be returned like other objects (`make_function_1.jl`):

```
function make_adder(a::Int64)
    function adder(b::Int64)
        a+b
    end
end

three_adder=make_adder(3)
two_adder=make_adder(2)

println(two_adder(5)," ",three_adder(5))
```

or even (`make_function_2.jl`)::

```

function make_adder(a::Int64)
    b::Int64->a+b
end

three_adder=make_adder(3)
two_adder=make_adder(2)
```

```
8
println(two_adder(5)," ",three_adder(5))
9
```

Note: As of 2024 some application-specific machine-learning libraries for Python (e.g. Google's *Jax*) can just-in-time compile subsets of Python to run in parallel. Optimisations in these libraries for certain "tensor" computations (parallel operations over large multi-dimensional arrays) allow them to outperform Julia on certain tasks resembling e.g. deep learning, even when executed on the CPU rather than GPU. These libraries, however, come with their own restrictions and particularities. This landscape is constantly in flux and you will find that that various benchmarks change as hardware and software evolves.

# Some mathematics

## A first order differential equation

Many of the differential equations we come across in neuroscience are inhomogeneous first order differential equations

$$\tau \frac{\mathrm{d}x}{\mathrm{d}t} = -x + u(t) \tag{1}$$

where $x(t)$ is a time-varying quantity we are interested in and $u(t)$ is some outside **driving function**.

One example we will examine is the integrate and fire equation:

$$\tau_m \frac{\mathrm{d}v}{\mathrm{d}t} = E_l - v(t) + g_l \cdot i(t) \tag{2}$$

where $v(t)$ is the voltage inside a neuron, and the driving function is made up of two parts $E_l$, the so called reversal potential and $g_l \cdot i(t)$ which represents current coming into the cell, for example, through an electrode if the cell being modelled is being experimented on, or because of signals from other neurons.

Another example is the gating equation which represents the opening and closing of ion-channels, little gates in the membrane of the cells and a third example is the equation for synaptic conductance. In each of these examples the underlying equation is the same, or nearly the same, and so we will look at it now in the abstract without considering the application.

These are called first order linear differential equations, first order because the highest derivative is $dx/\,\mathrm{d}t$, there is no $d^2x/\,\mathrm{d}t^2$ term for example, and linear because there are no non-linear $x$ terms, no $x^2$ or anything like that.

First order linear differential equations can be solved, or, at least the solution can be written as an integral.

## Solution using an integrating factor

In these notes we will review the use of an **integrating factor** to integrate Equation (1). Other approaches include using Laplace transforms or remembering that first-order linear ODEs tend to have solutions of the form $c \cdot \exp(at)$ and using this form as an ansatz.

Start by re-writing Equation (1) in the form

$$\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{1}{\tau}x = \frac{1}{\tau}u(t) \tag{3}$$

The integration factor simplifies (3) by multiplying it across by $e^{t/\tau}$; This gives

$$e^{t/\tau}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{1}{\tau}e^{t/\tau}x = \frac{1}{\tau}e^{t/\tau}u(t) \tag{4}$$

The clever bit, now, is to notice that the two terms on the left hand side can be rewritten in product form

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{t/\tau}x(t)\right) = \frac{1}{\tau}e^{t/\tau}u(t) \tag{5}$$

Moving $\tau$ over to the left-hand side and integrating both sides, we obtain

$$\tau e^{t/\tau}x(t) = \int_0^t e^{\tilde{t}/\tau}u(\tilde{t})\,\mathrm{d}\tilde{t} + c \tag{6}$$

where $c$ is some integration constant. Setting $t = 0$ shows that $c = \tau x(0)$ and, hence, dividing across by the stuff in front of $x(t)$

$$\begin{aligned} x(t) &= e^{-t/\tau}\left[\frac{1}{\tau}\int_0^t e^{\tilde{t}/\tau}u(\tilde{t})\,\mathrm{d}\tilde{t} + x(0)\right] \\ &= x(0) + \frac{1}{\tau}\int_0^t e^{-(t-\tilde{t})/\tau}u(\tilde{t})\,\mathrm{d}\tilde{t} \end{aligned} \tag{7}$$

Hence, $x(t)$ is written as an integral, if we can do the integral we have solved the equation.

The easiest case is obviously $u(t) = \bar{u}$ a constant:

$$\begin{aligned} x(t) &= \bar{u}\left[\frac{1}{\tau}e^{-t/\tau}\int_0^t e^{\tilde{t}/\tau}\,\mathrm{d}\tilde{t}\right] + x(0)\,e^{-t/\tau} \\ &= e^{-t/\tau}x(0) + (1 - e^{-t/\tau})\bar{u} \\ &= \bar{u} + [x(0) - \bar{u}]e^{-t/\tau} \end{aligned} \tag{8}$$

It is easy to understand this solution:

$$\tau\frac{\mathrm{d}x}{\mathrm{d}t} = \bar{u} - x(t) \tag{9}$$

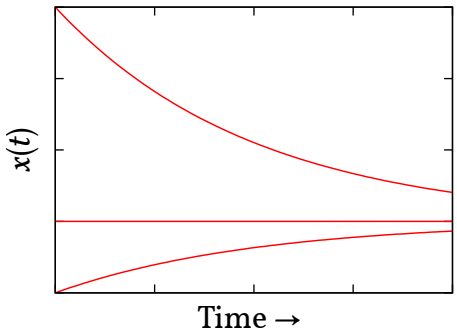



Figure 1: Two examples of $u(t)$ decaying towards $\bar{u}$. Here $\bar{u} = 1$ and $\tau = 10$; in the top curve $u(0) = 2.5$ whereas in the other $u(0) = 0.5$.

has $\mathrm{d}x/\,\mathrm{d}t = 0$ only if $x(t) = \bar{u}$ furthermore if $x(t) > \bar{u}$ then $\mathrm{d}x/\,\mathrm{d}t < 0$ and $x(t)$ decreases towards $\bar{u}$; if $x(t) < \bar{u}$ so $\mathrm{d}x/\,\mathrm{d}t > 0$ then $x(t)$ increases. Since $\mathrm{d}x/\,\mathrm{d}t$ is zero for $x(t) = \bar{u}$ we say this is the equilibrium; since the sign of $\mathrm{d}x/\,\mathrm{d}t$ means that $x(t)$ always approaches $\bar{u}$ we say this solution is stable.

We can see in the solution that $x(t)$ decays exponentially towards $\bar{u}$, where and it does so with a time scale that depends on $\tau$. An example is shown in Fig. 1.

What happens if $u(t)$ isn't constant? Well, some of the same logic applies: the sign of $\mathrm{d}x/\,\mathrm{d}t$ means that the solution is trying to get to $u(t)$ but the difference is now that $u(t)$ might change before it gets there. Actually looking at

$$x(t) = e^{-t/\tau}\left[\frac{1}{\tau}\int_0^t e^{\tilde{t}/\tau}u(\tilde{t})\,\mathrm{d}\tilde{t} + u(0)\right] \tag{10}$$

let us consider the function

$$h(t) = \frac{1}{\tau}e^{-t/\tau}\int_0^t e^{\tilde{t}/\tau}u(\tilde{t})\,\mathrm{d}\tilde{t} \tag{11}$$

and do a change of variable $\tilde{t} = t - s$ so

$$h(t) = \frac{1}{\tau}\int_0^t e^{-s/\tau}u(t-s)\mathrm{d}s \tag{12}$$

then the role of the integral is to filter or smooth out $u(t)$ by averaging it over its past with a exponential window. This is illustrated in Fig. 2.

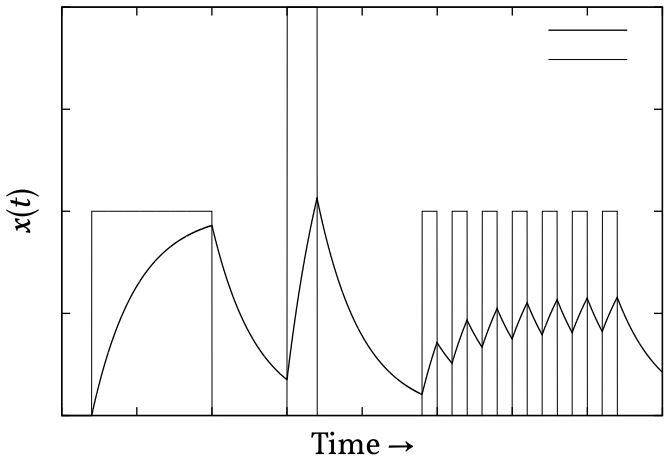



Figure 2: This shows how the response $x(t)$ filters the input $u(t)$; the differential equation has $\tau = 3$ and an input $u(t)$ which varies as shown above; the code used to calculate $x(t)$ is `filtered_input.py`.

Imagine that $u(t)$ varies slowly compared to the time scale $\tau$, so by the time $u(t-s)$ changes much compared to $u(t)$ then $\exp(-s/\tau)$ is more-or-less zero, then

$$h(t) \approx \frac{1}{\tau} u(t) \int_0^t e^{-s/\tau} \mathrm{d}s = u(t) \left(1 - e^{-t/\tau}\right) \tag{13}$$

and substituting back in we get

$$x(t) \approx u(t) + [u(0) - u(t)] e^{-t/\tau} \tag{14}$$

# Numerical integration of differential equations

This chapter is about the Taylor series and about the numerical solution of differential equations; two methods are considered, the Euler method and one of the Runge Kutta methods. These are the most straight-forward way to solve differential equations on a computer, there are other methods that are commonly used in neuroscience, like backwards Runge Kutta and adaptive time step methods, but these are easy enough to understand in the future once you know about Euler and Runge Kutta.

## The Taylor series

As an introduction it is often useful to use a series description of functions. This can help with numerical calculations of the values and it can be useful in studying properties of the function. Here, it will be used to work out how to efficiently calculate the solutions to differential equations. The Taylor series is one commonly applicable approach to representing a function as a series.

Imagine you have a function $f(t)$ that can be represented as a series

$$f(t) = \sum_{n=0}^{\infty} a_n t^n = a_0 + a_1 t + a_2 t^2 + \dots \tag{1}$$

Now, putting $t = 0$ we get

$$f(0) = a_0 \tag{2}$$

Next, differentiate

$$\frac{df(t)}{dt} = a_1 + 2a_2 t + 3a_3 t^2 + \dots = \sum_{n=1}^{\infty} a_n n t^{n-1} \tag{3}$$

so, putting $t = 0$ we get

$$\frac{df}{dt}|_{t=0} = a_1 \tag{4}$$

Differentiating again gives

$$\frac{d^2 f(t)}{dt^2} = 2a_2 + 6a_3 t + 12a_4 t^2 \dots = \sum_{n=2}^{\infty} a_n n(n-1) t^{n-2} \tag{5}$$

so

$$\frac{1}{2}\frac{d^2f}{dt^2}|_{t=0} = a_2 \tag{6}$$

and so on.

In fact, by this sort of calculation we see that

$$a_n = \frac{1}{n!}\left.\frac{d^n f}{dt^n}\right|_{t=0} \tag{7}$$

or, put another way,

$$f(t) = \sum_{n=0}^{\infty} \frac{1}{n!}\left.\frac{d^n f}{dt^n}\right|_{t=0} t^n \tag{8}$$

This is the Taylor series. We haven't proven that $f(t)$ has a series of the form $\sum_{n=0}^{\infty} a_n t^n$ at all and not all functions do, in particular, if the function is badly behaved at $t = 0$ it may not. We also haven't proved that the series converges. If we write

$$f(t) = \sum_{n=0}^{N-1} \frac{1}{n!}\left.\frac{d^n f}{dt^n}\right|_{t=0} t^n + E_N(t) \tag{9}$$

where $E_N(t)$ represents the error from stopping at after $N$ terms, we might expect $E_N(t)$ vanishes as $N$ goes to infinity. In fact, this doesn't always happen and sometimes, even when the series does converge, it does so very slowly, so $E_N(t)$ remains large even for very large values of $N$. Frequently the series converges for some values of $t$ but not for others. Nonetheless, the Taylor series is often useful.

You might already know the series expansion of $\exp t$, but lets calculate it as a Taylor series. Since

$$\frac{d}{dt}\exp t = \exp t \tag{10}$$

we know

$$\frac{d^n}{dt^n}\exp t = \exp t \tag{11}$$

or

$$\left.\frac{d^n}{dt^n}\exp t\right|_{t=0} = 1 \tag{12}$$

for all $n$ so

$$f(t) = \sum_{n=0}^{\infty} \frac{1}{n!} t^n \tag{13}$$

Next let's consider

$$f(t) = \sin t \tag{14}$$

Now

$$\frac{d}{dt}f(t) = \cos t \tag{15}$$

and

$$\frac{d^2}{dt^2}f(t) = -\sin t \tag{16}$$

and so on. Putting $t = 0$ and using $\sin 0 = 0$ and $\cos 0 = 1$ we get

$$\sin t = \sum_{n\,\text{odd}} \frac{(-1)^{(n-1)/2}t^n}{n!} \tag{17}$$

Finally, we have been expanding around $t = 0$, but you can expand around any point, here we expand around $t = t_0$

$$f(t) = \sum_{n=0}^{\infty} \frac{1}{n!} \left.\frac{d^n f}{dt^n}\right|_{t=t_0} (t - t_0)^n \tag{18}$$

or, writing $\epsilon = t - t_0$

$$f(t_0 + \epsilon) = \sum_{n=0}^{\infty} \frac{1}{n!} \left.\frac{d^n f}{dt^n}\right|_{t=t_0} \epsilon^n \tag{19}$$

## Numerical solutions of differential equations

Consider the differential equation

$$\frac{df}{dt} = G(t, f) \tag{20}$$

This class of differential equations would include the equation we looked at before:

$$\frac{df}{dt} = -\frac{1}{\tau}f \tag{21}$$

with $G(f) = -f/\tau$ or

$$\frac{df}{dt} = \frac{1}{\tau}[g(t) - f] \tag{22}$$

with $G(t, f) = (g - f)/\tau$. In fact, for most of this discussion we will restrict ourselves to the case where $G$ doesn't depend on $t$ except through $f$, this case is a small bit simpler.

Now imagine we want to find numerical values for $f(t)$ where we know $f(0) = f_0$, some value, and

$$\frac{df}{dt} = G(f) \tag{23}$$

Imagine further that we can't solve the equation analytically, so we resort to solving it approximately on the computer. This might be necessary because the equation is too hard to solve, or, in the case of the equation

$$\tau\frac{df}{dt} = g(t) - f(t) \tag{24}$$

it might be that we don't know $g(t)$ analytically.

The normal approach would be to discretize time and to work out the solution approximately for each time step in turn, depending on the previous time step. In other words, say we choose $\delta t$, a small value, as the time step then we would work out $f(\delta t)$, then use that to work out $f(2\delta t)$ and so on. Now, if $\delta t$ is small, then $\delta t^2$ will be smaller and $\delta t^3$ smaller still; the idea behind numerical solutions to differential equations is that we drop higher powers of $\delta t$.

Let us use the notation $f_n = f(n\delta t)$ and consider how we might get a computer to work out $f_{n+1}$ approximately if $f_n$ is already known. Now, by the Taylor expansion

$$f(n\delta t + \delta t) = f(n\delta t) + \left.\frac{df}{dt}\right|_{t=n\delta t}\delta t + \frac{1}{2}\left.\frac{d^2f}{dt^2}\right|_{t=n\delta t}(\delta t)^2 + \ldots \tag{25}$$

so one simple approach is to ignore the $(\delta t)^2$ and smaller terms, since $df/dt = G(f)$ this gives

$$f_{n+1} = f_n + G(f_n)\delta t \tag{26}$$

This approximation is known as the *Euler method*. A simple example is plotted in Fig. 1.

Since the Euler method takes into account the $\delta t$ part of the Taylor approximation but not the $\delta t^2$ term, we say it is accurate up to $O(\delta t^2)$, in other words

$$f(t + \delta t) = (\text{Euler approximation}) + O(\delta t^2) \tag{27}$$

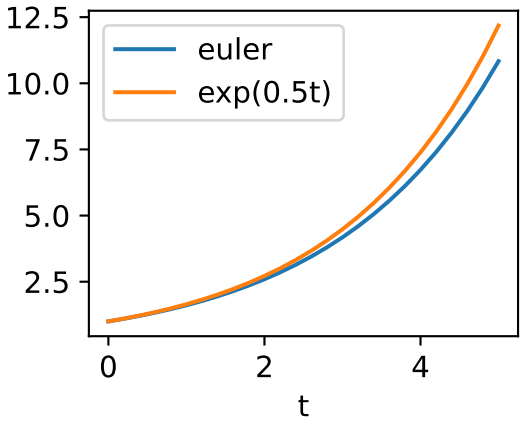



Figure 1: A comparison of the Euler method with the true solution for the equation $df/dt = rf$ with $r = 0.5$. This equation in actually one where the Euler method works quite well for modest values of $t$, though the error always has the same sign and starts to accumulate, here a large time step of $\delta t = 0.2$ is used to emphasise the error, the actual solution is plotted for comparison.

where, roughly, the $O(\delta t^2)$ stands for stuff that behaves like $\delta t^2$ as $\delta t$ gets small.

## The Runge-Kutta method

The Runge-Kutta method uses the Taylor expansion in a clever way to find a better approximation than the Euler method. It is a bit convoluted, so there is a lot of notation, but it does give a very useful numerical algorithm. We are not going to explicitely derive the Runge-Kutta method here; you can see a longer discussion of where it comes from in `runge_kutta.pdf` in the same notes. The key idea behind Runge-Kutta, as with Euler, is to find a way to approximate $f(t+\delta t)$, the Runge-Kutta approximation, however, accounts for more terms of the Taylor expansion. In fact, the fourth order Runge-Kutta we will describe here gets everything up to the fourth order, the errors are like $\delta t^5$:

$$f(t+\delta t) = (\text{Runge-Kutta approximation}) + O(\delta t^5) \tag{28}$$

So in the fourth order Runge Kutta approximation we have, as before

$$\frac{df}{dt} = G(f) \tag{29}$$

and we calculate

$$\begin{aligned} k_1 &= G(f_n) \\ k_2 &= G\left(f_n + \frac{1}{2}\delta t k_1\right) \\ k_3 &= G\left(f_n + \frac{1}{2}\delta t k_2\right) \\ k_4 &= G\left(f_n + \delta t k_3\right) \end{aligned} \tag{30}$$

then the approximation is

$$f_{n+1} = f_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\delta t \tag{31}$$

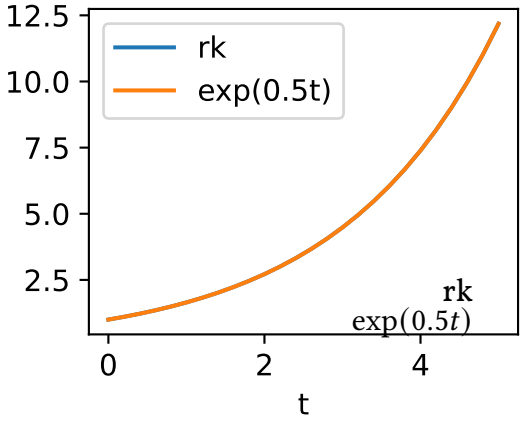



Figure 2: A comparison of the fourth order Runge Kutta method with the true solution for the growth equation. This has the same values of $r$ and $\delta t$ as in Fig. 1, but the fourth order Runge-Kutta method is used instead of the Euler method. As you can see, the approximation and true solution are indistinguishable.

# McCulloch–Pitts neurons

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]. 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:

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

$\phi$ is an activation function, $\theta$ 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 $\phi$:

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

Thus, the neuron has two states, it is in the on state, $x_i = 1$ if the weighted input exceeds a threshold $\theta_i$ 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

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

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.

## 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.

## Hopfield network

A **Hopfield network** is a recurrently connected network; it is intended to perform pattern completion and was proposed by John Hopfield in 1982 [1], 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

$$\hat{x}_i = \phi\left(\sum_j w_{ij} x_j\right) \tag{1}$$

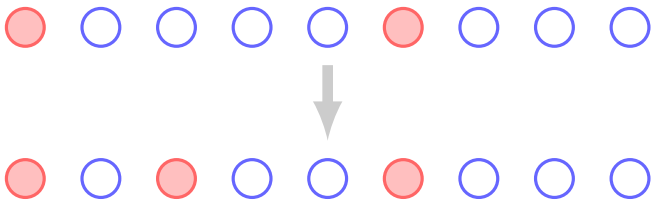
and then $x_i \longrightarrow \hat{x}$; that is all the nodes update using the old values. In the most common version of a Hopfield network, the $w_{ij}$ are symmetric, that is

$$w_{ij} = w_{ji} \tag{2}$$

The threshold values $\theta_i$ 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; for simplicity we stick to synchronous updates here.

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:

$$E = -\frac{1}{2}\sum_{ij} w_{ij}x_i x_j \tag{3}$$

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. Thus, updating will tend to make terms like $w_{ij}x_i x_j$ into a positive number if $w_{ij}$ is large and so 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, if the dynamics reduces the energy, $E$, the goal is to pick the connection strenghts so that the minima of $E$ correspond to the patterns that the network is charged with storing. Now the question is how to create the correct local minima? Here is a rules to achieve this:

$$w_{ij} = \frac{1}{N}\sum_{a} x_i^a x_j^a \tag{4}$$

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 this rule, a sort of 'top down' rule is inspired by Hebbian plasticity. There is an online rule that is even closer to Hebbian plasticity where $w_{ij}$ is changed for each presentation:

$$\delta w_{ij} = \frac{\eta}{4}(x_i^a + 1 - 2\alpha)(x_j^a + 1 - 2\alpha) \tag{5}$$

Following this online rule will bring you to the values in the formula for creating the correct minima: Eqn. 4. The $\alpha$'s are an ofset value which allow the network to reach a stable equilibrium since $\delta w_{ij}$ should stop changing (on average) if the average fraction of iterations where $x_i$ or $x_j$ is plus one is equal to $\alpha$.

These changes in the $w_{ij}$ mimick a simple version of **Hebbian plasticity**, a putative description of how the synapses in the brain change in response to

neuronal activity. 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.

The closest thing to an overall rule was formulated by Hebb in 1949 when he said [2]:

> 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

$$\delta w_{ij} = \eta x_i x_j \tag{6}$$

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 $\eta$ is a learning rate. Another version is

$$\delta w_{ij} = \eta(x_i - \alpha)(x_j - \alpha) \tag{7}$$

where having $\alpha$ allows for different cut-off points between the behaviour that causes potentiation or depression.

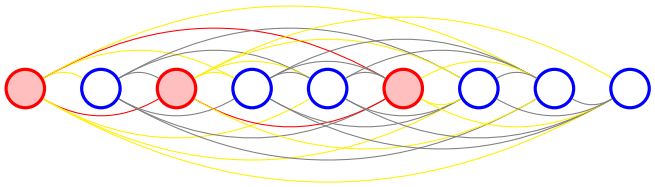


Figure 1: Learning in the associate network. The pattern has been imposed and connection strengths are changed. The red links increase by $\eta(1-\alpha)^2$ and the gray by $\eta(-\alpha)^2$, the yellow links decrease by $\eta\alpha(1-\alpha)$.

This is clearly, up to a change in variables for the $\alpha$ and rescaling of $\eta$, this is the same as the rule

$$\delta w_{ij} = \frac{\eta}{4}(x_i^a + 1 - 2\alpha)(x_j^a + 1 - 2\alpha) \tag{8}$$

mentioned for Hopfield networks; in fact, in the Hopfield network $\alpha$ 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. In the brain, the idea is that outside activity, signals from outside the auto-associative network, will cause some of the neurons, the pattern to be learned, to be active while others remain dormant. During this time plasticity occurs. Later, during recall, the outside activity causes a fraction of the pattern to become activity. The internal dynamics of the network, the McCulloch-Pitts updates, cause other neurons to also become active, allowing the pattern to repeat.

Typically $\alpha$ 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. 1.

As discussed, 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

$$r_i = \sum w_{ij} x_j \tag{9}$$

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. 2. 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 $N^2/N = N$ [3].

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 $\alpha$ 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 of the order of $1/\alpha$ patterns where a connection is weakened to wipe out the strengthening that results if the connection is part of a pattern. In fact, it is estimated that the capacity of a network is

$$P = \frac{k}{\alpha} N \tag{10}$$

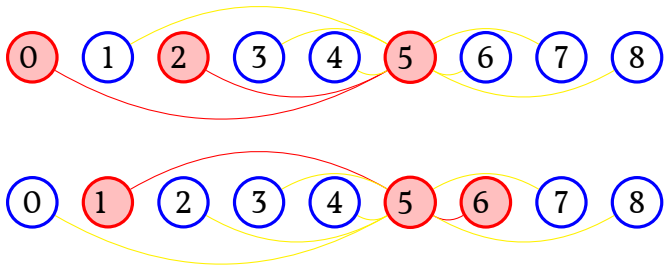



Figure 2: Interference in an auto-associative network. Neuron 5 is involved in two patterns and, as a consequence, some of its connections are strengthen for one pattern and weakened for the other, if these strengthening and weakening effects are similar in size it makes it unlikely that either pattern will be accurately retrieved.

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

$$P = c\frac{k}{\alpha}N \tag{11}$$

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] 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.

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

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

## The Hippocampus

These notes are about the hippocampus; for convenience some of the figures here repeat the figures already used in the notes on the McCulloch-Pitts neurons.

### Anatomy of the hippocampus

The **hippocampus** is situated at the medial ("toward the middle") edge of the temporal lobe, along the floor of the temporal horn of the lateral ventricle. It is divided into two main areas. The name *hippocampus* means seahorse in Latin, and comes from the appearance of the hippocampal formation and it's output tract. The hippocampus complex, and contains sub-regions also named for their shape:

- **Cornu Ammonis (CA)** - meaning the *horn of Ammon*, an Egypian god of ferility with curved horns. The CA is usually divided into four regions, labelled CA1 through to CA4.
- **Dentate Gyrus (DG)** - gyrus is the name given to the ridges in the cortex, dentate means *with teeth*. The dentate gyrus is one of the few areas of the adult brain that exhibits neurogenesis.

In addition, the main input to the hippocampus comes from the

- **Entorhinal Cortex (EC)** - entorhinal means *near the smell processing area*.

and in this discussion this will be treated along with the hippocampus since it participates in hippocampal processing.

### The role of the hippocampus

The role of any brain region is complex and the hippocampus is no exception. It may play a role in olfaction for example, however, it is widely believed that its principle role is in memory and in constructing spatial maps. Studying patients with hippocampal damage shows that it is involved in declarative memory, that is the sort of memory that can be described in words. It does not play a role in procedural memory, the memory process which allows us to learn new motor skills. It appears, again from patients with hippocampal damage, that some long term memories are stored outside the hippocampus.

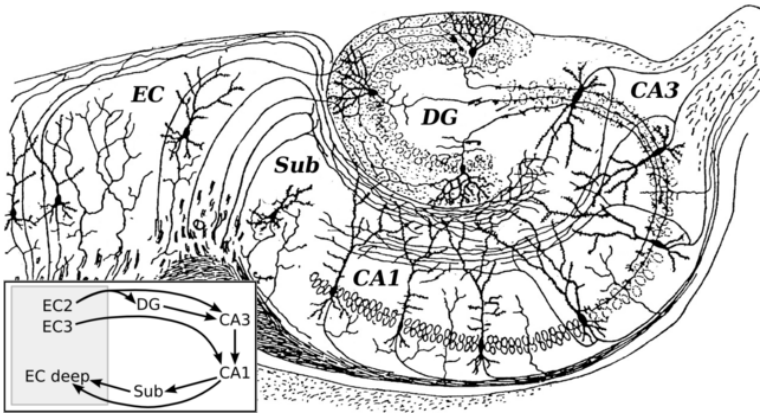



Figure 1: The hippocampus. This is a modified image originally due to Cajal, probably from his 1911 book, the inset shows the approximate connectivity. [From `http://en.wikipedia.org/wiki/File:CajalHippocampus_(modified).png`]

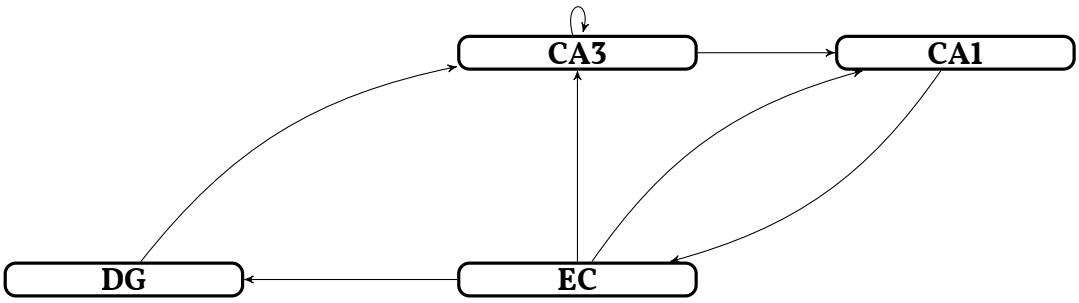



Figure 2: Connectivity of the hippocampus. A rough diagram showing the major connections between the areas of the connectivity. The set of axons running from EC to DG, CA3 and CA1 is called the perforant pathway, the mossy fibres run from DG to CA3 and the Schaffer collateral fibers go from CA3 to CA1. The loop on CA3 is supposed to represent the high level of recurrent connections in that regio.

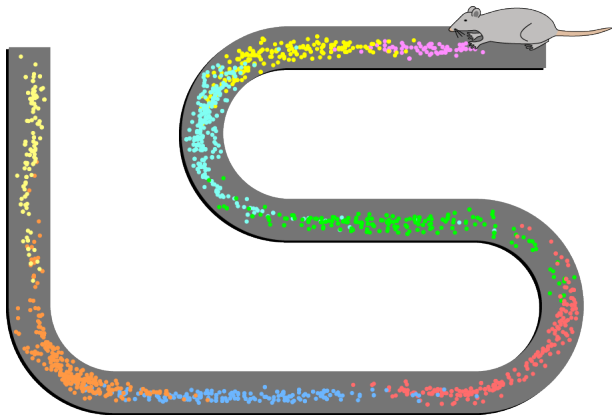


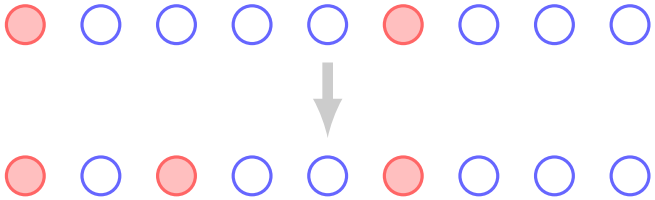
Figure 3: Place cells. This shows the firing activity of eight different cells in CA1 of a rat moving along a path, the dots correspond to spikes. [From `http://en.wikipedia.org/wiki/Place_cell`]

The best studied function of the hippocampus is spatial memory. There are cells in rat CA1 known as place cells which fire in response to specific location, an example is shown in Fig. 3. These were first discovered in 1971 [1] and show both that the hippocampus has a role in spatial memory and that the memory encoding is very sparse. Another, striking, example was discovered in 2005 in humans [2]. Recordings were taken from electrodes implanted for medical reasons in patients with focal epilepsy, these patients are required as part of an investigation into their epilepsy to spend time with the electrodes implanted; during this time many generously agree to take part in scientific investigations. They were shown many images while the activity of individual cells were monitored. It was found that some cells respond to very specific stimuli and that this stimulus can be quite abstract. In the most quoted example a cell in hippocampus in one patient was found which responded to pictures of Jennifer Aniston. It did this irrespective of how she appeared, but did not respond to other famous and non-famous faces, or curiously to pictures of Jennifer Aniston with Brad Pitt, her spouse at that time. Again, this demonstrates a role in memory, but one that involves very sparse responses.

### Auto-associative memory

The standard paradigm for memory in the hippocampus is *auto-associative* memory. As we discussed in the last chapter auto-associative memories are patterns representing memories along with some dynamics that complete partial patters. Imagine a sequence of on-off 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 idea is that the hippocampus implements a network that performs auto-associative memory. This view is in some ways associated with David Marr, although his work preceded our modern knowledge of hippocampal anatomy. The most plausible site for this auto-associative network is CA3: this is the only area with recurrent connections between the principal cells. Often when discussing brain regions, the connectivity is classified as feed-forward or recurrent according to the connectivity of the principal cells (the most common cell type whose axons extend to other brain regions). There are almost always recurrent connections with other cells, such as inhibitory interneurons. This is true in the CA3 and CA2: In both cases the principal cells are pyramidal cells whose axons extend to other brain regions. There are also inhibitory neurons called basket cells, which are connected to the pyramidal cells. However, only in CA3 are the pyramidal cells typically connected to other pyramidal cells.

### A model of CA3

Here a highly simplified model of CA3 is presented [4]. In this model CA3 is all-to-all connected and made up of McCulloch-Pitts neurons. As before let $N$ be the number of neurons, $x_i$ the activity of neuron $i$ and $w_{ij}$ the strength of the connection for $i$ to $j$. The sparseness, the average proportion of neurons

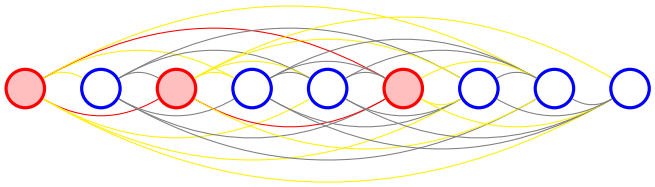


Figure 4: Learning in the associate network. The pattern has been imposed and connection strengths are changed. The red links increase by $\eta(1-a)^2$ and the gray by $\eta(-a)^2$, the yellow links decrease by $\eta a(1-a)$.

active at any one time is $\alpha$, this is believed to be very small in actual neurons. For this simple model $w_{ij} = w_{ji}$.

During learning the patterns are activated and plastic changes are made to the synapse strength according to a simple correlation based Hebbian plasticity rule.

$$\Delta w_{ij} = \frac{\eta}{4}(x_i + 1 - 2\alpha)(x_j + 1 - 2\alpha) \tag{1}$$

where $\eta/4$ is the learning rate, often a small number, the four is just for notational convenience, but, in hippocampus were memories need to be learned quickly, possibly during a single presentation, $\eta$ is large. Since $\alpha$ is very small too for real networks there will be a large increase, $\tilde{\eta}$ for the connection between two neurons that are active at the same time, a tiny increase $\tilde{\eta}\alpha^2$ for pairs neurons that are inactive at the same time and a medium size decrease $-\eta\alpha$ for pairs of neurons where one is active and one inactive. See Fig. 4.

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

$$h_i = \sum w_{ij} x_j \tag{2}$$

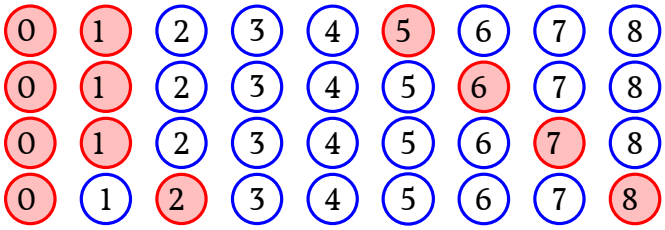
and is set in the active state if $h_i > \theta$ where $\theta$ is a threshold which is set to different values for different networks. The idea is that after learning the pattern $\{0, 2, 5\}$

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

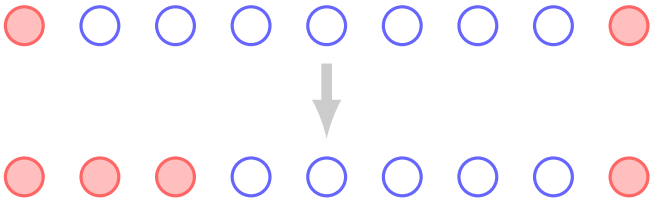

the value $h_2 = w_{12} + w_{52}$ will be larger than threshold and the subsequent dynamics will switch neuron 2 on. However, in this network, if a different initial set of neurons are activated, the activity will die away because the $h_i$ will all be sub-threshold.

### Correlated patterns

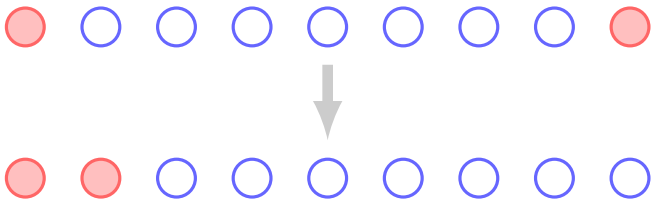
The estimates of capacity assume that the patterns are all independent. If they aren't the capacity is reduced. If patterns share some fragments or subpatterns then the connections in these subpatterns become very strong, perhaps dominating other elements in the patterns, the elements that make them different. Consider the four patterns




The connection between neurons 0 and 1 will become very strong because this connection is present in three patterns out of four. It is likely that this will result in this erroneous completion

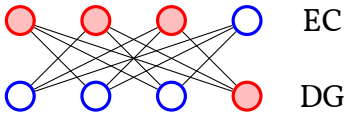


or even

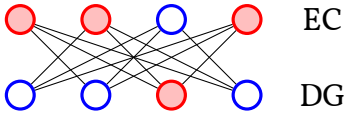


This means that auto-associative networks are not able to effectively store anything except random patterns! This is why they have never proved useful for machine learning; in cortical memory, since there are multiple presentations supported by a hippocampal representation, the memory system has the opportunity to learn different, similar, memories. The goal here, however, is to learn the memory quickly after a small number of presentations.

In the case of hippocampus it has been proposed [6] that this problem is solved through the EC-DG-CA3 pathway, and that the role of the dentate gyrus is to randomise the connectivity between EC and CA3. In short, during learning neurons in EC and CA3 are matched via DG and that the connections from EC to DG and from DG to EC are essentially random. This reduces overlap through a $k$-winner takes all mechanism. Roughly, $k$-winner-take-all assumes that local inhibition ensures that the $k$ most active neurons in the DG layer "win" and suppress the activity of the other neurons [6]. The way this might reduce overlap is shown in this cartoon where $k = 1$ and two similar patters result in a different neuron being active:



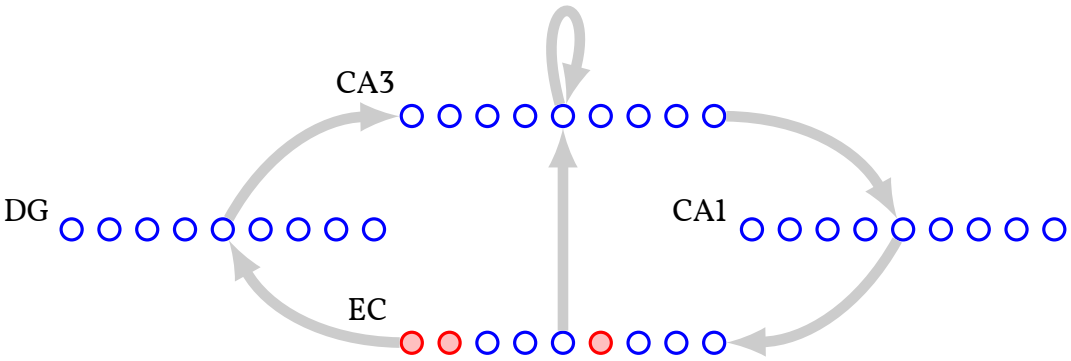

versus



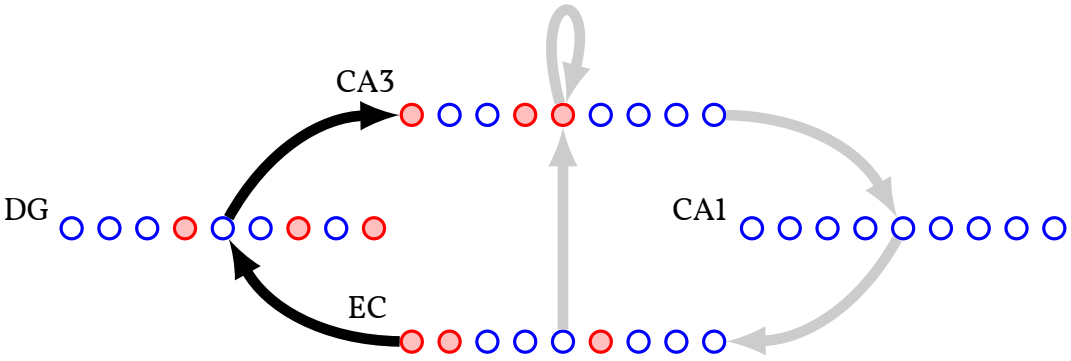

with the idea being that this randomization might be repeated in the subsequent connection between DG and CA3. This mechanism may explain why neurons in DG are being born all the time, perhaps their role is to create these random connections.

### Models of the whole hippocampus

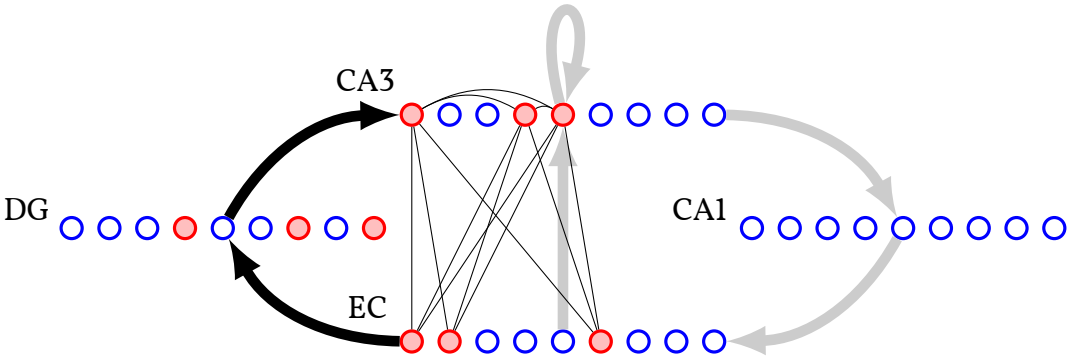
We are now almost in a position to consider models of the hippocampus as a whole; so far CA1 hasn't been mentioned. It has been suggested that the role of CA1 is to relay patterns back to EC. In learning, according to the standard model, a pattern is presented by EC:



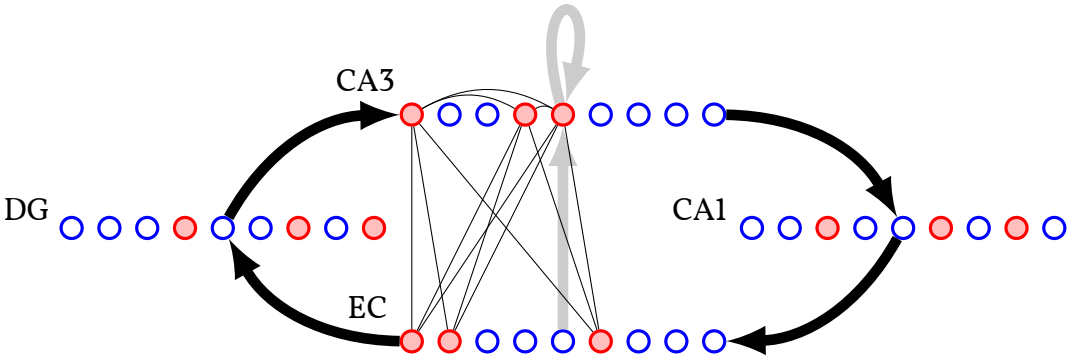

Connections, possibily random, between EC and DG and between DG and CA3, along with the '$k$-winner takes all' mechanism, causes activity in CA3.



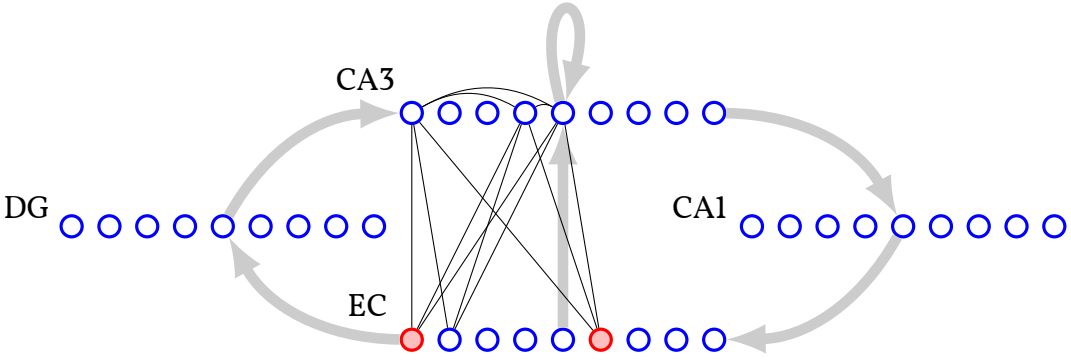

This representation is then learned by Hebbian plasticity between EC and CA3.




Hebbian learning also stregthens links to map the pattern to CA1 and link that to EC.

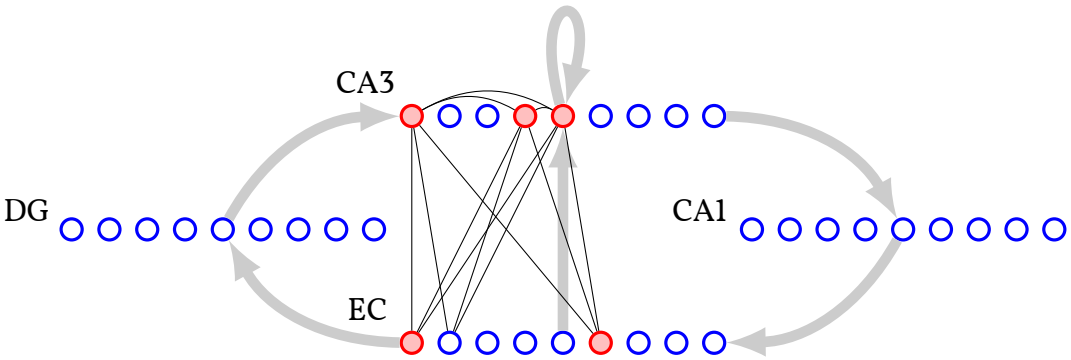



Now, during retrieval, part of the pattern is presented to EC.

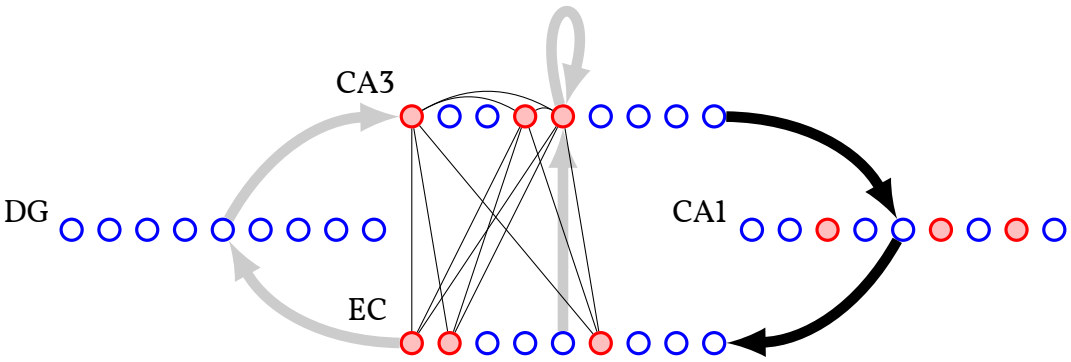



Because of the connections from EC to CA3 and the recurrent connections

in CA3, this excites the pattern in CA3:




This memory is sent back to EC via CA1, and recall has occurred!




This doesn't explain how the hippocampus switches between the learning and retrieval phase. One suggestion that the level of stimulus is different—that larger activity during learning excites the pathway that goes via DG [6], another is that the neuromodulator acetylcholine [7] or dopamine [8] is important.

## References

[1] O'Keefe J, Dostrovsky J. (1971) The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. Brain Research 34: 171–5.

[2] Quian Quiroga R, Reddy L, Kreiman G, Koch C, Fried I. (2005) Invariant visual representation by single neurons in the human brain. Nature 435: 1102–7.

[3] McCulloch W, Pitts W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 7: 115–33.

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

[5] O'Reilly RC, Munakata Y (2000) Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT press, Cambridge MA.

[6] O'Reilly RC, McClelland JL (1994) Hippocampal conjunctive encoding, storage, and recall: Avoiding a tradeoff. Hippocampus 4: 661–82.

[7] Hasselmo ME. (2006) The role of acetylcholine in learning and memory. Current Opinion in Neurobiology 16: 710–71

[8] Lisman JE, Grace AA. (2005) The hippocampal-VTA loop: controlling the entry of information into long-term memory. Neuron, 46: 703–713.

## 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 [1] 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 forebearer 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.

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

$$y = \phi(r) \tag{1}$$

where

$$r = \sum_j w_j x_j - \theta = \mathbf{w}^\top \mathbf{x} - \theta \tag{2}$$

and $\phi$ here is the Heaviside-like threshold activation function described before. 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:

$$\delta w_j = \eta(d - y)x_j \tag{3}$$

and

$$\delta\theta = \eta(d - y) \tag{4}$$

where $\eta$ is some small learning rate and

$$\begin{aligned} w_j &\rightarrow w_j + \delta w_j \\ \theta &\rightarrow \theta + \delta\theta \end{aligned} \tag{5}$$

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.

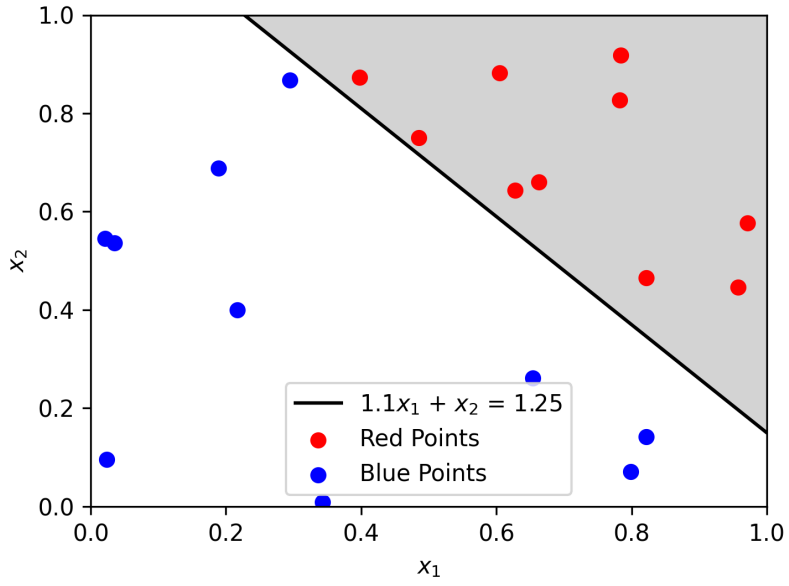



Figure 1: Here the shaded region corresponds to $1.1x_1+x_2 > 0.25$, so if $w_1 = 1.1$ and $w_2 = 1$ with $\theta = 0.25$ then the corresponding perceptron neuron will be one for all the circle points and -1 for all the square points.

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 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 min-

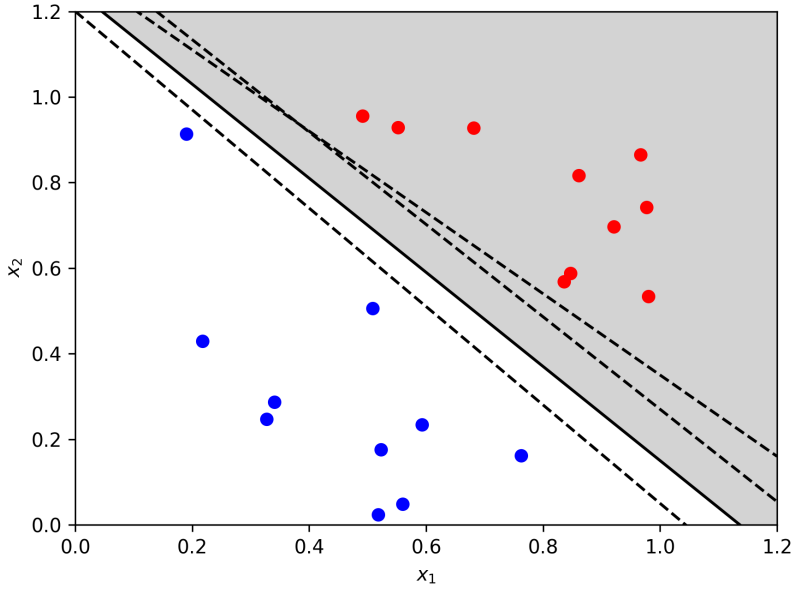



Figure 2: All of these lines separate the points into two classes

imization. Consider an error function

$$E = \tfrac{1}{2}\langle (y_a^* - \hat{y}_a)^2 \rangle_a \tag{6}$$

In Equation (6) above,

- The subscript $a$ indexes individual input–output training pairs $(\mathbf{x}_a, y_a^*)$,
- $y_a^*$ is a **desired** or **target** activation,
- $\hat{y}_a$ is the **predicted** output from the perceptron computed via Equations (1) and (2) for input vector $\mathbf{x}^a$,
- The angle brackets $\langle \cdot \rangle_a$ denote an average over training pairs.

Now consider the gradient of $E$ with $w_i$:

$$\frac{\partial E}{\partial w_i} = -\left\langle \frac{\partial \hat{y}_a}{\partial w_i} \cdot (y_a^* - \hat{y}_a) \right\rangle_a \tag{7}$$

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

$$\hat{y} = \phi(r) = \phi\left(\mathbf{w}^\top \mathbf{x} - \theta\right) \tag{8}$$

we have

$$\frac{\partial \hat{y}_a}{\partial w_i} = \frac{\partial \phi}{\partial r} \cdot \frac{\partial r}{\partial w_i} \tag{9}$$

so using

$$\frac{\partial r}{\partial w_i} = x_i \tag{10}$$

we get

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= -\left\langle \frac{\partial \phi}{\partial r_a} \cdot \frac{\partial r_a}{\partial w_i} \cdot (y_a^* - \hat{y}_a) \right\rangle_a \\ &= -\left\langle \frac{\partial \phi}{\partial r_a} \cdot x_{a;i} \cdot (y_a^* - \hat{y}_a) \right\rangle_a \end{aligned} \tag{11}$$

Of course, in the McCulloch-Pitts case $\frac{\partial \phi}{\partial r_a}$ is either zero or undefined, but we can pretend that $\frac{\partial \phi}{\partial r_a} = 1$ (a form of **surrogate gradient**). 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 $\theta$.

To side-step this messy business of non-differentiable activation function, most modern perceptron neural networks use smooth (or mostly smooth) activation functions. However, you will still see various surrogate-gradient approaches applied to incorporate steps or spikes in model neural networks for biological plausibility.

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


Figure 1: A drawing by Santiago Ramón y Cajal of a Purkinje cell. [Picture taken from `http://en.wikipedia.org/wiki/Golgi's_method`]

## 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.

Purkinje cells have a distinctive structure with a huge, highly branched, but flat dendritic arbour, see Fig. 1; 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 arbours 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 arbours, and a strong input from a climbing fibre, a single axon which winds around the Purkinje cell and makes multiple con-

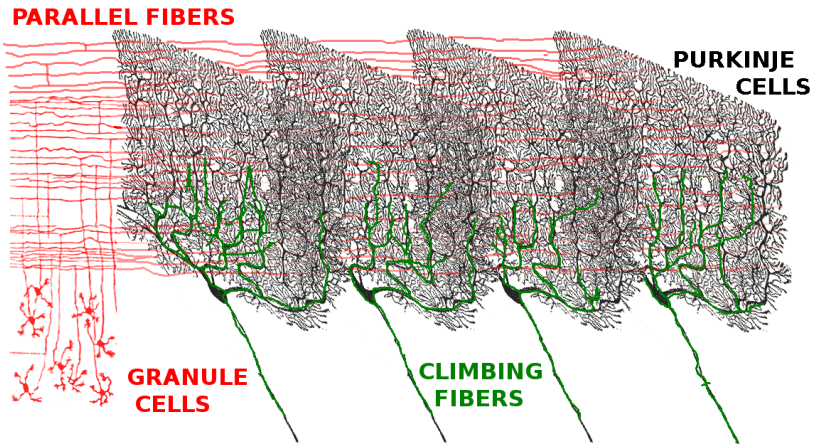



Figure 2: A cartoon of the cerebellar circuitry. A vertical axon rises from each granule cells, splits once and then extends horizontally in two directions making connections with multiple Purkinje cells. Each Purkinje cell has its own climbing fiber which winds up around it.

tacts with it, see Fig. 2.

Another peculiarity is that the Purkinje cell emits dramatically different types of spikes to different inputs. In response to multiple weak inputs from the parallel fibres it fires a normal spike, called in this context a **simple spike**; In response to single spike from the climbing fibre is fires a special spike, called a **complex spike**, with a leading spike, a number of small 'spikelets' and a sustained after-period of depolarisation; this is illustrated in Fig. 3.

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, 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 [1], or for predicting the sensory or proprioceptive consequences of motor actions [2]; 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.

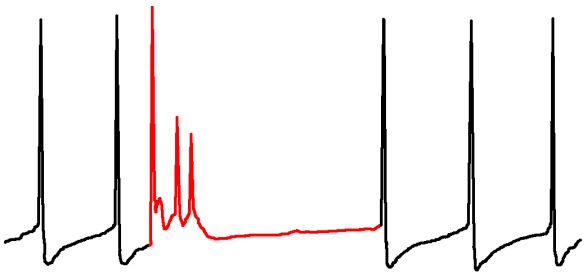


Figure 3: A complex spike. This drawing shows simple spikes in black and a complex spike in red. The complex spike is followed by a long refractory period during which spiking is not possible. This is a sketch, not an actual recording, but a typical time scale would have this refractory period 50 ms long.

Whatever exactly it does, it is widely believed, in accordance with the Marr-Albus model [3, 1], 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

$$y = -\sum w_i x_i \tag{1}$$

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

## References

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

[2] 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.

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

# Rate neurons

In this short note we introduce the idea of **rate-based neurons**. In the previous section we considered McCulloch-Pitts neurons. These have an on-off behaviour, we know that neurons actually communicate using spikes, also called action potentials, which are discrete pulses of voltage. Of course, this is not necessarily inconsistent; it is possible, but unlikely, that neurons have a low and high firing rate and that we can effectively analyse that part of the behaviour of the brain that is relevant to computation by considering neurons only as on / off thresholding units. However, it is much more likely that this would only give a partial account of neural computation, but that there are nonetheless useful insights to be gained in considering what algorithms could be supported by on-off neurons.

The next step in our consideration of neuronal models is to imagine that spikes are irrelevant but spike firing rates are important and that we can understand neural computation by describing computation based on neural firing rates. It is known that this picture does not apply to some specialised circuits such at the one used by owls to locate their prey. In the owl auditory system the precise temporal difference between sounds arriving at each of the owls ears is calculated by finding the point individual spikes from each ear arrive at the same point. However, it is not uncommon for people to believe that, in general, neuronal computation is based on the communication and transformation of firing rates; in fact, one common point-of-view is that computation is rate based but that plasticity, that is learning through changes in synapse strengths, depends on spike times.

The idea of rate coding dates back to the earliest detailed measurements of the electrical activity of nervous system. Edgar Adrian was one of the great pioneers, he was a neurophysiologist, but, as is often the case, his success relied on innovation in equipment, coupled, of course, to careful experimentation. He pioneered the use of vacuum tubes to amplify electrical signals allowing him and his co-workers to record the activity in individual nerve fibres. He recorded from the sensory nerves carrying signals from muscles in frog, by attaching a silk thread to the muscle, passing it over a pulley and attaching a weight he was able to observe a linear relationship between the frequency of spikes along the nerve and the size of the weight. He also noted that the frequency decreased with time, see Fig. 1.

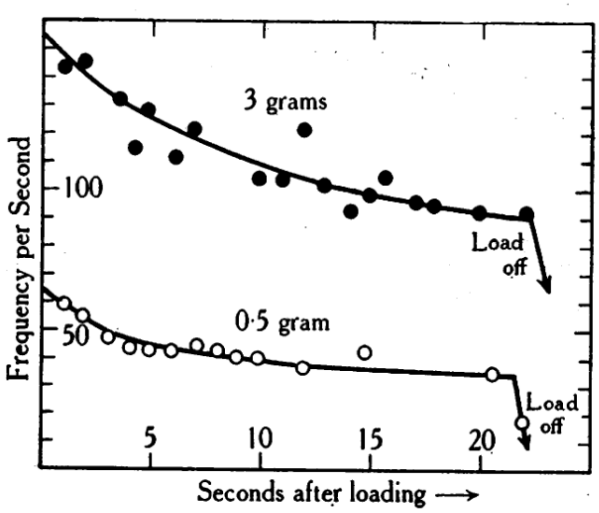



Figure 1: A figure from [1] showing the response to two different weights.

# References

[1] Adrian, ED. and Zotterman Y (1926) The impulses produced by sensory nerve-endings: Part II. The response of a Single End-Organ. The Journal of Physiology, 61.2:151

# Vision

## Introduction

This lecture is about vision, it will discuss how **simple cells** in **V1** are modelled and how their behaviour may be explained by sparseness.

## The visual pathway

The visual system starts at the eye, where photons are detected and some denoising occurs; the optical nerve then carries the information to the thalamus, in the very centre of the brain, there it is further processed, compressed, and denoised before being relayed on to the visual cortex, at the very back of the cortex. It is processed in stages in the cortex with the information being passed forward, as objects are recognised the information fans out and is integrated with other signals, from memory, from other sensory modalities and other aspects of our cognition. The basic pathway is shown in a very old drawing in Fig. 1 and is summarised in Fig. 2. One notable aspect is that different sides of the brain deal with different sides of the visual field, so signals from the left sides of the retina of both eyes go to the right side of the brain and signals from the right sides so to the left side.

Light is detected at the retina. The retina is a surprising organ in that it is backwards compared to how you'd expect it to be organised; the layer with light detectors is at the back instead of the front. Leaving that aside though, light is detected in specialised cells called **photoreceptors**. These don't spike, but they do convert light into electrical activity. There are two types of photoreceptors: rods, which are important for vision in low light, and the cones, which are responsible for colour vision and important for vision in normal lighting conditions. The electrical activity of the photoreceptors is passed forward through **bipolar cells** to **ganglion cells**.

Ganglion cells aggregate activity from a number of photoreceptors, along with activity from some inhibitory cells in the intermediate layer and their axons form the optic nerve, carrying information to the thalamus. A sketch of the retina is given in Fig. 3 and the uneven distribution of cones and rods across the retina is illustrated in Fig. 4.

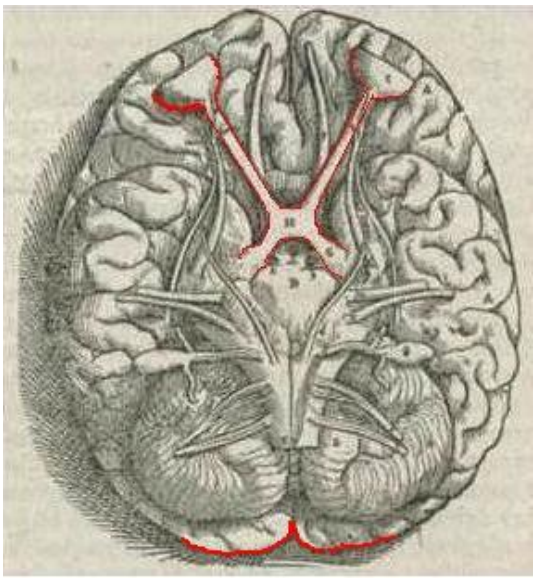


Figure 1: The visual pathway. This is an old drawing due to the sixteenth century Belgian anatomist Andreas Vesalius taken from his influential 1543 textbook *De Humani Corporis Fabrica*. In red are marked the retina, the optic nerves, the thalamus where they cross and the primary visual cortex (V1). [Image from Wikipedia].

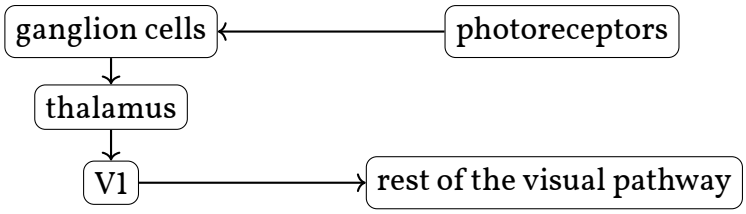



Figure 2: The visual pathway. This is a very rough diagram showing the visual pathway; V1 is the first visual area in the cortex.

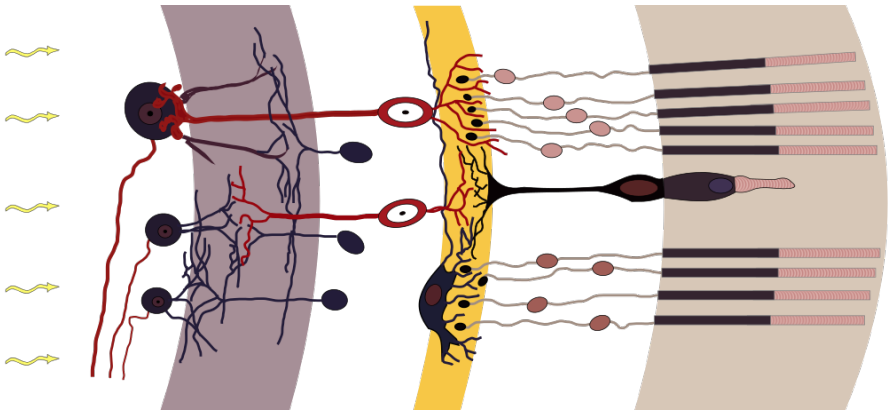


Figure 3: Rods, cones and nerve layers in the retina. The front of the eye is on the left. Light (from the left) passes through several transparent nerve layers to reach the rods and cones (far right). A chemical change in the rods and cones send a signal back to the nerves. The signal goes first to the bipolar and horizontal cells (middle yellow layer), then to the amacrine cells and ganglion cells (left-most purple layer), then to the optic nerve fibres. The signals are processed in these layers. First, the signals start as raw outputs of points in the rod and cone cells. Then the nerve layers identify simple shapes, such as bright points surrounded by dark points, edges, and movement. (Based on a drawing by Ramón y Cajal, 1911.) [Caption and drawing taken from Wikipedia: Cajal derivative work: Anka Friedrich via Wikimedia Commons]

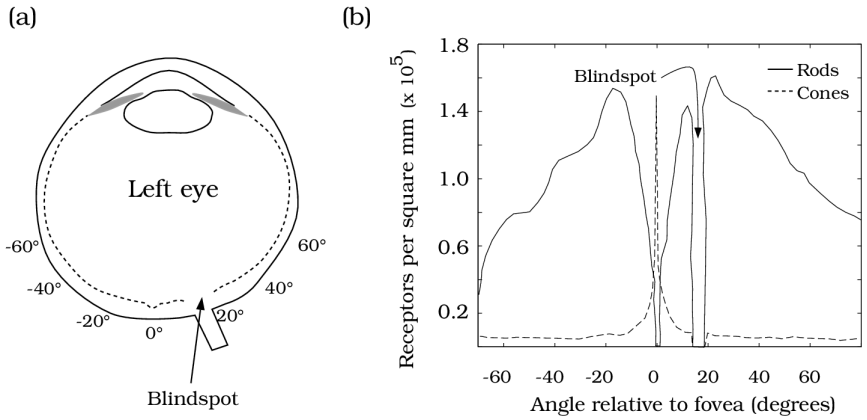



Figure 4: The distribution of rods and cones in the retina. [Image from [1]]

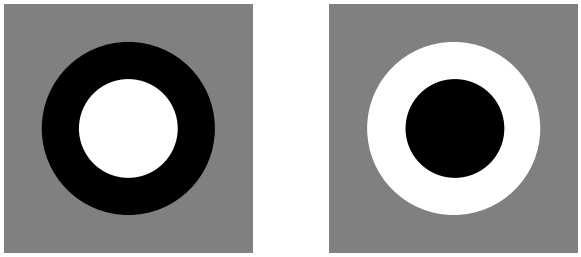
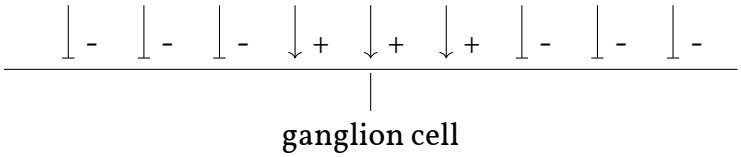


Figure 5: On and off cells respond to small contrast patches.

## Receptive fields

**Receptive fields** are often described as the stimuli giving the largest response from a neuron. For ganglion and thalamic cells these are contrast patches, see Fig. 5, in **on-cells** small patches of the visual field where an illuminated region surrounded by an unilluminated one causes firing, different cells will respond to different locations. The width of the receptive fields vary from the size of full stop at reading distance in the center, to the size of a page near the periphery. In **off-cells** the contrast is reverse, the cell responds to an unilluminated region surrounded by an illuminated region. In practice these patterns are the result of excitatory and inhibitory synapses relaying information from these regions of the visual field.




In V1 there are cells called **simple cells** and cells called **complex cells**; we will concentrate on the simple cells, these have edge-like receptive fields; different cells respond to particular orientations in particular locations in the visual field.

The edge-like receptive fields in V1 were first discovered by Hubel and Wiesel [2]. They used an electrode to record from V1 neurons in anaesthetised cat; they moved an edge-like stimulus around until they found the position that caused the highest firing rate, they observed that the firing rate depended on orientation as well as position, see Fig. 7 and Fig. 8.

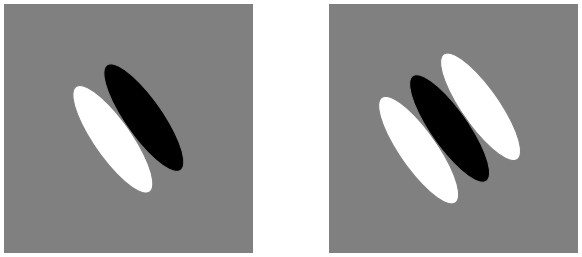


Figure 6: Simple cells in V1 respond to edges.

## Linear models

One way to think about it is to imagine the entries in the receptive field are synapse strengths for inputs from cells responding to illumination at points in the visual field. To formalize this consider linear models of the neuron's activity. Let $I_{ij}$ denote the illumination level at point $(i, j)$ in the visual field, $i$ and $j$ are discrete coordinates, for simplicity we will treat everything discretely. Now, imagine a linear model of the activity of the neuron, with the firing rate depending linearly on the illuminations; leaving out any messing with the firing rate having to be positive, this means

$$\tilde{r} = r_0 + \sum w_{ij} I_{ij} \tag{1}$$

where $r_0$ is the background firing rate and $w_{ij}$ give the receptive field. Of course the firing rate of a neuron doesn't satisfy a linear model but the idea is to choose the linear model which best approximates the neuron, that is, for example, to choose $w_{ij}$ to minimize the average square error

$$\text{average square error} = \langle (r - \tilde{r})^2 \rangle \tag{2}$$

between $r$, the observed firing rate and $\tilde{r}$ is the estimated firing rate from the linear model.

As an example consider

$$[w_{ij}] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1/8 & -1/8 & -1/8 & 0 \\ 0 & -1/8 & 1 & -1/8 & 0 \\ 0 & -1/8 & -1/8 & -1/8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

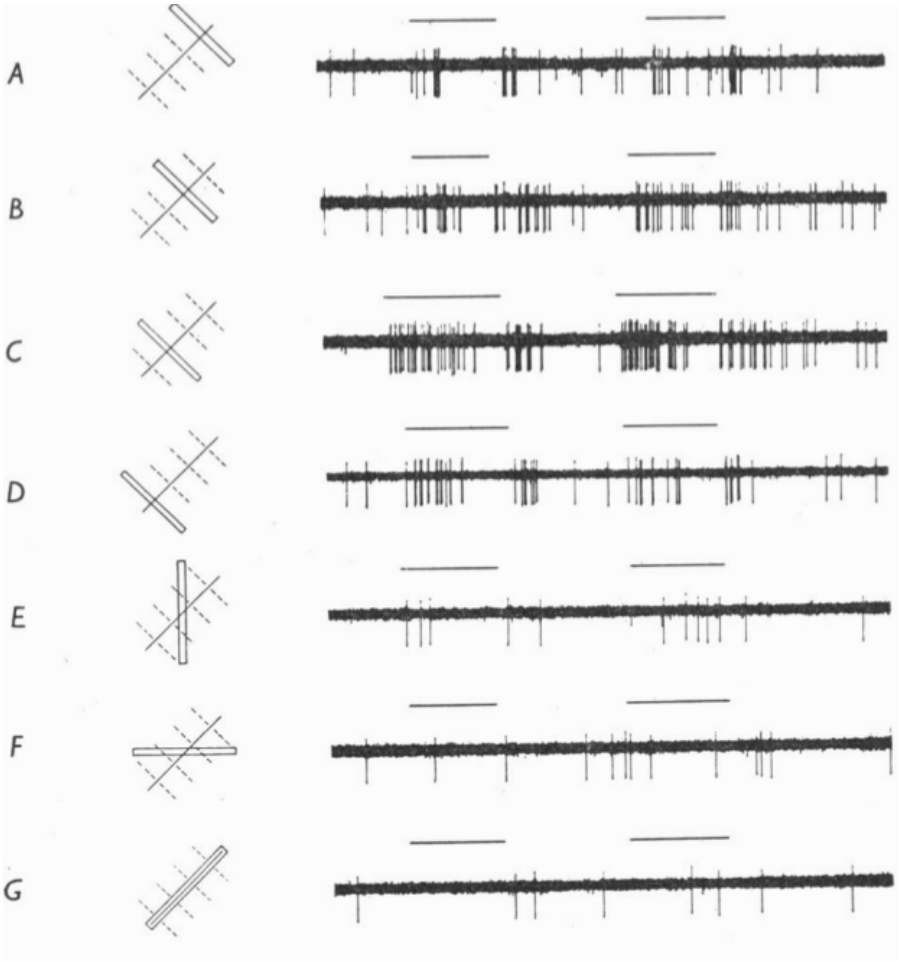
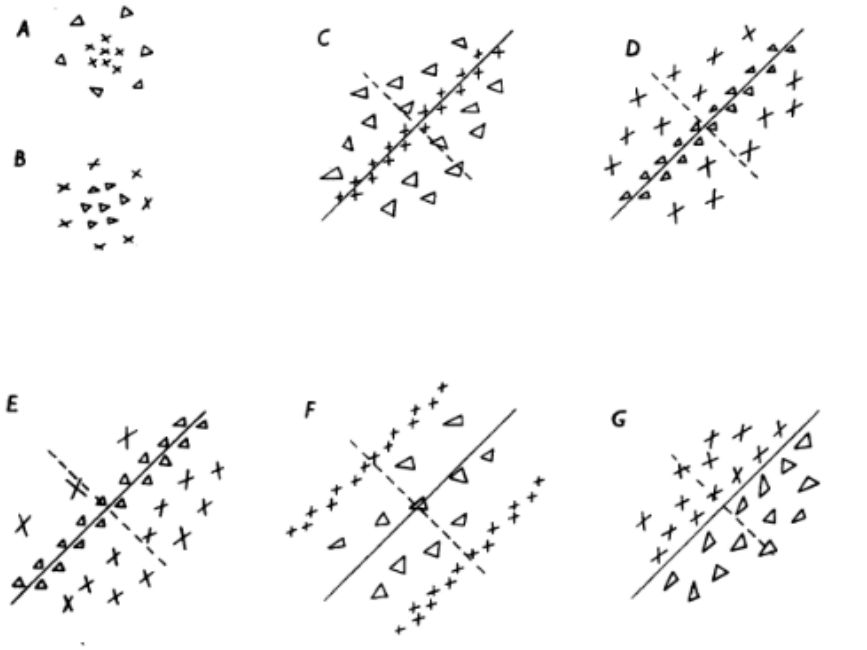



Figure 7: Experimental results from Hubel and Wiesel; the stimulus is a slit that allows light through from a souce, it is $0.125° \times 2.5°$ and is presented during the one-second period marked by the two bars over the plots. In the plots the vertical lines correspond to spikes. [Image from [2]].




Text-fig. 2. Common arrangements of lateral geniculate and cortical receptive fields. *A*. 'On'-centre geniculate receptive field. *B*. 'Off'-centre geniculate receptive field. *C–G*. Various arrangements of simple cortical receptive fields. ×, areas giving excitatory responses ('on' responses); △, areas giving inhibitory responses ('off' responses). Receptive-field axes are shown by continuous lines through field centres; in the figure these are all oblique, but each arrangement occurs in all orientations.

Figure 8: More experimental results from Hubel and Wiesel; here they have mapped out the excitatory (crosses) and inhibitory (triangles) areas for a number of neurons. [Image from [2]].

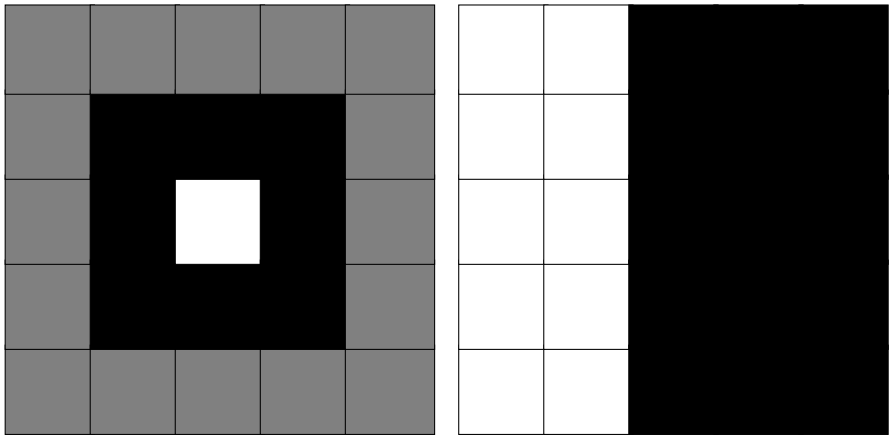


Figure 9: Receptive field and visual stimulus.

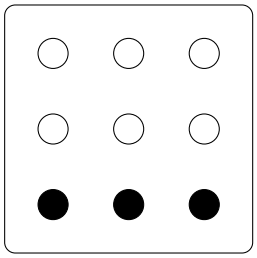
and

$$[I_{ij}] = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{4}$$

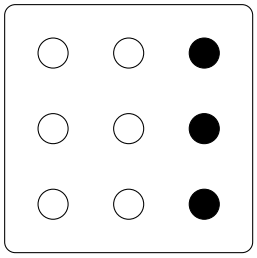
which is like a ganglion cell responding to an edge and is illustated in Fig. 9. If $r_0 = 2$ say then $\tilde{r} = 13/8$.

## Features

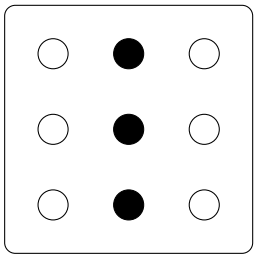
Knowing why V1 receptive fields have the particular stucture they do is likely to tell us something about what it is that the brain does to information in the sensory pathways. One idea is that it is related to feature extraction. To motivate this we will consider a fictitious world of simplified creatures; imagine we are one of these creatures and wish to decide how to react to other creatures we encounter. As in the real world, when we encounter a creature we need to decide between what are sometimes called the three Fs: fighting, fleeing and mating. Now imagine that the creatures all have a three by three pattern on their stomachs:



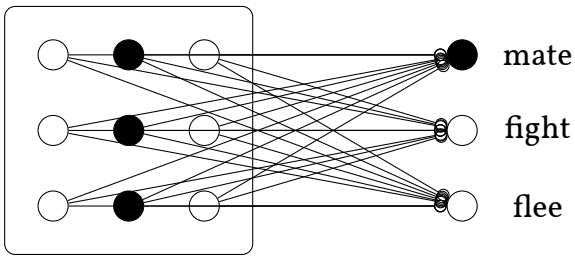
and that creatures to fight have a horizontal strip to the top or the bottom, as above, creatures to flee from, a vertical strip on the left or right, for example



and creatures to mate with, a central line, either horizontal or vertical like:



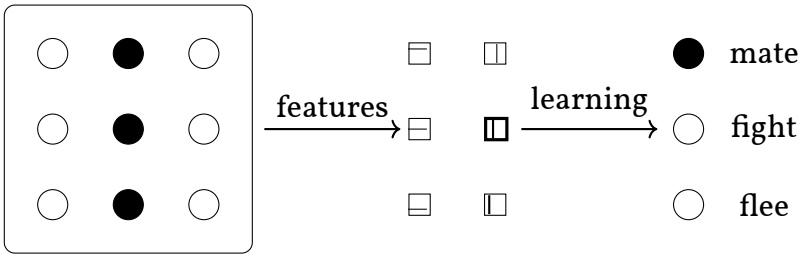
Now, imagine processing this information so as to rapidly decide what to do, the simplest neural network to process the patterns would look like




Clearly this would be very hard to learn; the connection from, say the bot-

tom middle node is on in two of the patterns above despite these patterns corresponding to different types of creature.

A far better strategy would be to first learn features and then learn the association between these features and the creature type. Here the features are clearly the horizontal and vertical bars




In this case the problem has been split in two; first the connections summarized as 'features' above are learned from the data, possibly using the sort of correllation structure learning provided for by STDP, the interpretation of these feature is then learned, this is clearly easier, the connections summarized as 'learning' have a far simpler task, which is good, since it is crucial to learn this sort of salient ecological information quickly.

## Feature selection

Here we will consider what properties we would expect features to have. To do this lets imagine that there are neurons that correspond to the features and their activity represents the image. Say the feature=code neurons each have a receptive field and respond linearly and for simplicity we will leave out the background firing rate: for the $s$th feature neuron

$$a_s = \sum_{ij} w_{ij}^s I_{ij} \tag{5}$$

and conversely, the output can be represented by

$$I_{ij} = \sum_{s} a^s W_{ij}^s \tag{6}$$

**Confusing aside - rate versus reconstruction**

The slightly confusing thing here is that we are moving between the linear model

$$a_s = \sum_{ij} w_{ij}^s I_{ij} \tag{7}$$

with $a_s$ the firing rate, we have changed to $a$s rather than $r$s to be consistient with the papers this is based on, and the reconstruction

$$I_{ij} = \sum_s a_s W_{ij}^s \tag{8}$$

which estimates the image using the firing rates $a_s$.

We do this all the time with vectors:

$$\mathbf{v} = v_1\mathbf{i} + v_2\mathbf{j} + v_3\mathbf{k} \tag{9}$$

is the reconstruction where the corresponding project, for example

$$v_1 - \mathbf{v} \cdot \mathbf{i} \tag{10}$$

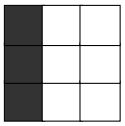
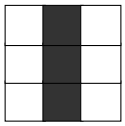
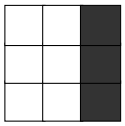
is like the linear model. However, the situation in this case is more straightforward, because the basis vectors are orthonormal

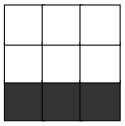
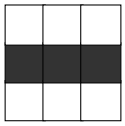
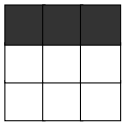
$$\mathbf{i} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{i} = 0 \tag{11}$$

and

$$\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1 \tag{12}$$

the same basis vector appears in the reconstruction and the projection: the coefficent $v_1$ of $\mathbf{i}$ in the reconstruction is the projection of $\mathbf{v}$ onto $\mathbf{i}$. However, in the case of vision the basis elements, the $W_{ij}^s$ and $w_{ij}^s$ are not orthonormal and therefore are not the same, working out the relationship between involves vectorizing the matrix indices $i$ and $j$, so we won't go into it here, morally one is the inverse transpose of the other. In fact, here we will consider an example where the dimensions are different, where the image patches are $3 \times 3$ but there are only six features,

$W^1 =$  $W^2 =$  $W^3 =$ 

$W^4 =$  $W^5 =$  $W^6 =$ 

where the almost-black corresponds to one and white to zero, so put another way

$$[W^1_{ij}] = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{13}$$

This means that the reconstructed image may not be equal to the original image and will just be an approximation to it.

### Sparseness

The question now is what principle to use to select these features. In considering the mythical creatures we decided that features should be horizontal and vertical lines since that's the pattern that the creatures have. The question is how to find these features in general.

One idea, due to [3, 4], is to use **sparseness**. Continuing with the example with creatures with patterned bellies, looking at flee-from animal, for example, three dots are black so among neurons that code for individual dots three would be active, but among neurons coding for vertical or horizontal lines, only one would be active.

Of course this is a very artificial made-up example, nonetheless it is thought that 'sparseness' is a good way to define features [3, 4]. Very roughly, the fewer neurons needed to reconstruct an image, the more of the image each neuron is coding for; for this to work without having a vast number of neurons covering every possible combination of pixels, the neurons must code for features, pieces of image that occur regularly. The assumption, in short, is that all the images are mostly made of the same few building blocks.

The idea is as follows, let $I$ be a image and $\tilde{I}$ an approximation to that image formed using features

$$\tilde{I}_{ij} = \sum_s a_s W^s_{ij} \tag{14}$$

Now $W^s$ could be under-complete, like above, or over-complete, as it may be in the visual system, or the dimensions could be chosen to match. Either

way, even if the basis is not under-complete, $\tilde{I}$ is an approximation because the $a_s$ are not just chosen to give an accurate reconstruction, but to do so in a sparse way; they are chosen to minimize

$$E = \sum_{ij}(I_{ij} - \tilde{I}_{ij})^2 + \beta \sum_s f(a_s) \tag{15}$$

This has two terms, the first measures the square error between $I$ and $\tilde{I}$, the second is intended as a measure of sparseness, there are different choices possible, one example would be

$$f(x) = \log_2(1 + x^2) \tag{16}$$

Now, just looking at two dimensions $(1, 0)$ gives

$$\sum_s f(a_s) = \log_2(2) + \log_2(1) = 1 \tag{17}$$

whereas the less sparse $(1/\sqrt{2}, 1/\sqrt{2})$, which as a vector is the same length, gives

$$\sum_s f(a_s) = 2\log_2(3/2) = 1.17 \tag{18}$$

which is larger. The $\beta$ here determines the trade-off between accuracy and sparseness, if $\beta$ is small the square error is more important, if $\beta$ is big the sparseness is.

The idea now is to take a corpus of image patches and find the best features, the ones that on average give the lowest values of $E$. The algorithm proceeds in two stages, for each image patch the $a_s$ are chosen to minimise $E$ basically by numerically solving the system of differential equations

$$\frac{\partial E}{\partial a_s} = 0 \tag{19}$$

Next, using the results of this calculation for all the images in the corpus, the features $W_{ij}^s$ are adjusted

$$W_{ij}^s \rightarrow W_{ij}^s - \eta \frac{\partial \langle E \rangle}{\partial W_{ij}^s} \tag{20}$$

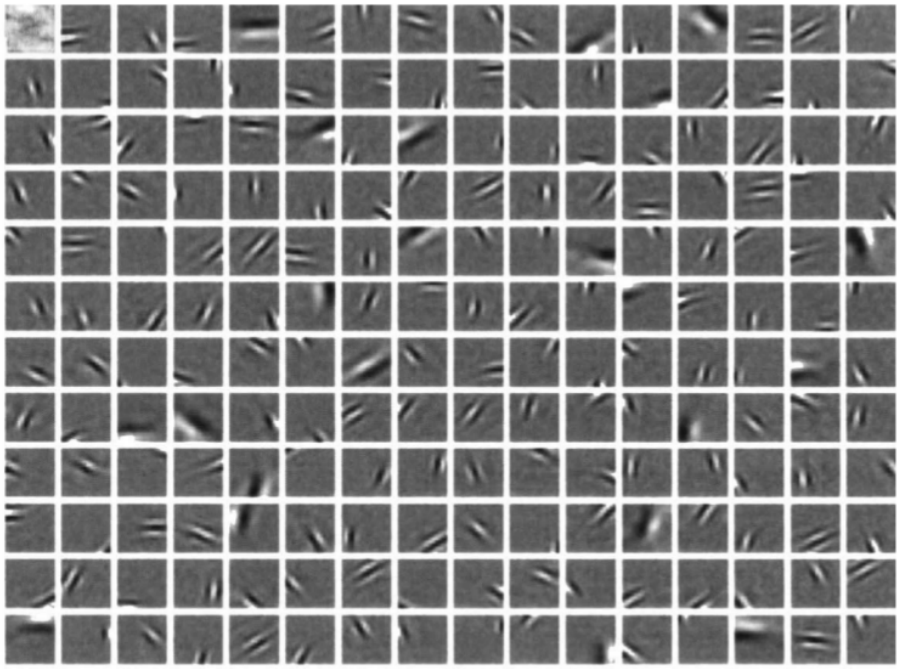


Figure 10: Sparse filters; the optimal features $W_{ij}^s$ using $16 \times 16$ image patches cut from a corpus of pictures of the American northwest. [From [3]].

where $\eta$ is a learning rate and $\langle E \rangle$ this average error. This should reduce the average error in the next run through the corpus. This is repeated until the best features are found. The details of how $E$ is minimized over possible choices of the $a_s$ and how to adjust the $W_{ij}^s$ can be found in [3, 4]. The results are shown in Fig. 10 and, ignoring the complication that these are not actually the receptive fields, they do clearly resemble the receptive fields measured from V1.

As described, none of this seems very biological, the sparse filters were discovered using numerical optimisation routines. However, there are biologically plausible implementations using **Hebbian learning**, see for example [5]. There are other approaches which parallel sparseness as a way of distinguishing features, for example, Infomax, which examines the informativeness of putative features [6].

# References

[1] Wandell, Brian A. Foundations of vision. (Sinauer Associates, 1995) foundationsofvision.stanford.edu/.

[2] Hubel DH, Wiesel TN. (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of Physiology 160: 106.

[3] Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381: 607–609.

[4] Olshausen BA, Field DJ (1997) Sparse coding with an overcomplete basis set: A strategy employed by V1?. Vision Research 37: 3311–3325.

[5] O'Reilly RC, Munakata Y (2000) Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT Press.

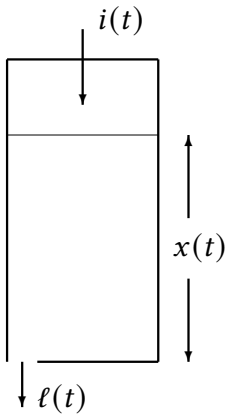
[6] Bell AJ, Sejnowski TJ (1995) An information-maximization approach to blind separation and blind deconvolution. Neural Computation 7: 1129–1159.

# Equation for leaky buckets

These notes are about a simple but very useful model of spiking neurons: the integrate and fire model. First, though, we consider the dynamics of a leaky bucket of water. These dynamics are very similar to those seen in the integrate and fire model.

## Buckets of water

In the simplest model of neurons their voltage dynamics is similar to the dynamics of a bucket with a leak and the class of equations that apply in this case will also be applied to synapses, for example.




Consider a bucket with straight sides which is filled to a height $x$ with water. Imagine the water leaks out of a hole in the bottom. The rate the water leaks out depends on $x$; the larger $x$ is the larger the pressure at the bottom is and hence the faster the water pours out. In other words

$$\ell(t) \propto x(t) \tag{1}$$

or

$$\ell(t) = Gx(t) \tag{2}$$

where $G$ is a constant which will depend on the size of the hole and complicated things like the viscosity of water. Of course, we are ignoring lots of complicated stuff, like turbulence and so forth, but since we are interested in the equation rather than the amount of water in a bucket, this is fine.

Imagine water also pours in the top at a rate $i(t)$. This means the total rate of change of the amount of water is $i - Gx$.

Now, $x$ is the height of the water not the volume: the volume is $Cx$ where $C$ is the cross-sectional area of the bucket. The rate of change of the volume is therefore

$$C\dot{x} = i - Gx \tag{3}$$

or

$$\dot{x} = \frac{1}{C}\left[i - Gx\right] \tag{4}$$

or perhaps most usefully

$$\tau\dot{x} = Ri - x \tag{5}$$

where $R = 1/G$ and, more interestingly, $\tau = C/G$.

The $\tau$ is called "tau" because it is a timescale. This can be checked in two ways, first, in this equation on the right hand side there is a $x$, a length, and $Ri$ that must also be a length. In fact, from $\ell = Gx$ we can see that $R$ converts flow rates like $\ell$ and $i$ to heights. This implies the left hand side must also be a height and $\tau$ is therefore a timescale to balance the 'per time' from the derivative. We can also check directly. From the equation defining $G$ we can see $[G] = L^2/T$ and from we know $[C] = L^2$ so $[\tau] = [C]/[G] = T$.

This is an equation we know how to solve; we saw it at the start of the course when learning about differential equations. This same equation will appear when we look at the dynamics of neuron and for similar reasons, something, water in the case of the bucket, charge in the case of a neuron, is being added to a leaky container.

## Constant input

This equation can be solved analytically if the current flowing in is constant, but we won't try that calculation here since it only works for this specific special case, normally we have to solve numerically, using a computer. The

solution is

$$x(t) = [x(0) - Ri]e^{-t/\tau} + Ri \tag{6}$$

This makes sense, when $x = Ri$ then the equation says $\dot{x} = 0$ so this is an equilibrium point, a value where everything stops changing. The dynamics describe the systems as decaying exponentially to the equilibrium.

As for actually finding the solution, that can be done a few different ways, I would usually solve it by ansatz, that is by guessing the solution and checking I am right, so I would substitute in

$$x(t) = A + Be^{rt} \tag{7}$$

with $A$, $B$ and $r$ constants to get

$$Br\tau e^{rt} = Ri - A - Be^{rt} \tag{8}$$

and the solution follows by matching up the terms to give $A$ and $r$ so, equating the coefficients of the exponential I get $r = -1/\tau$ and $A = Ri$. To find $B$ you just look at the initial condition, the value when $t = 0$. Another common approach to these equations is the integrating factor, this avoids knowing a good guess as the most of ansatz requires, here you rewrite the equation

$$\dot{x} + \frac{1}{\tau}h = \frac{R}{\tau}i \tag{9}$$

and notice that multiplying by $\exp(t/\tau)$ allows you to write the left hand side as a product:

$$\partial_t \left( he^{t/\tau} \right) = \tfrac{R}{\tau} ie^{t/\tau} \tag{10}$$

Next you integrate both sides. There are still other approachs, for example, using Laplace transforms.

These dynamics are intuitive: The more water there is in the bucket, the higher the pressure will be at the leak and the quicker the water will pour out. If there is just the right amount of water the rate the water pours out the leak will precisely match the rate it pours in, this is the equilibrium. If there is more water than required for equilibrium it will pour out faster than the flow coming in, if there is less, it will pour out slower. Either way, as time passes the height of the water will reach the equilibrium. The plot in Fig. 1 illustrates this.

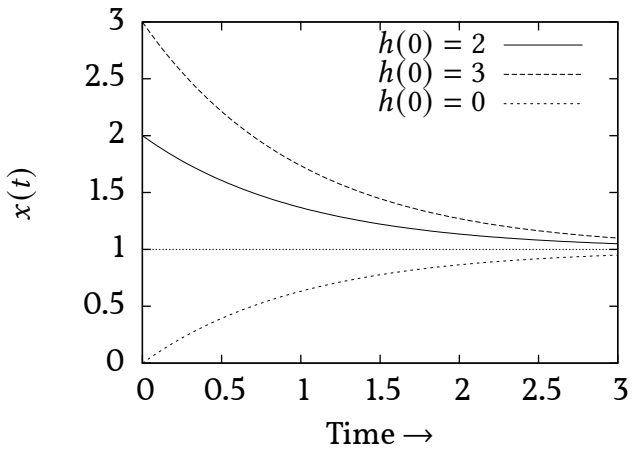



Figure 1: Exponential relaxation. The dynamics described by the "bucket equation" are very common. Here $x(t)$ is plotted with $i = G$, $\tau = 1$ and three different values of $x(0)$. $x(t)$ relaxes towards the equilibrium value, the closer it gets, the slower it approaches.

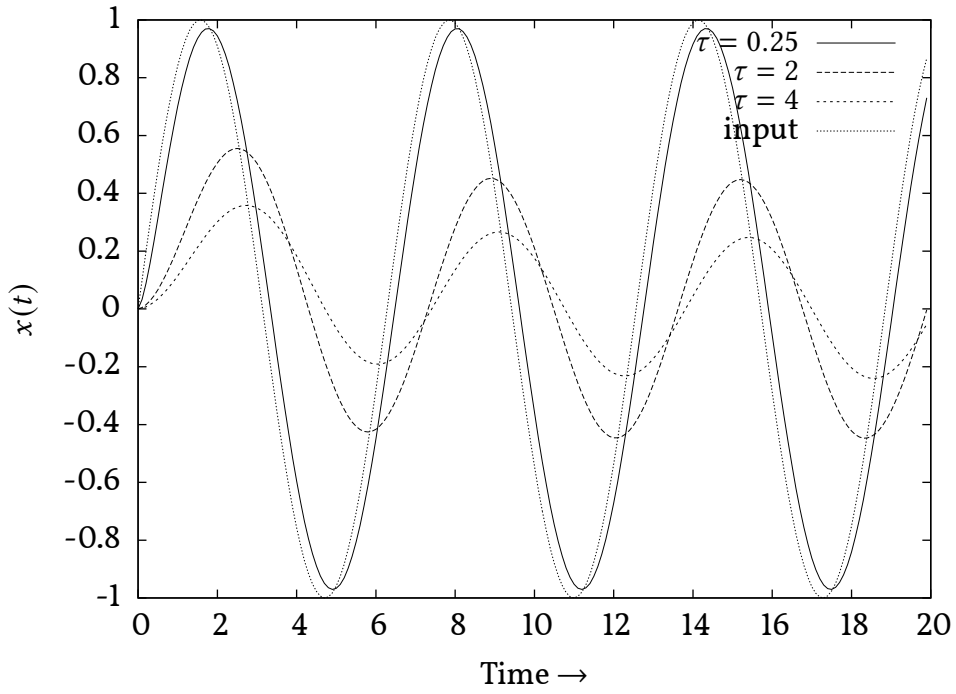



Figure 2: Variable input. Here the input is a sine wave $i(t) = \sin(t)$ and the equation is evolved with $x(0) = 0$ and three different $\tau$ values. For $\tau = 0.25$ we see $x(t)$ closely matches the input whereas for larger $\tau$ it is smoother and lags behind.

## Variable input

We have only discussed constant inputs; the variable input case where $i$ depends on time is harder and although it can sometimes be solved it is often easier just to compute it numerically. We will look briefly at how to do this, but first note that the effect of variable input is that the solution follows the input with a timescale set by $\tau$. For very small $\tau$ is chases it quickly, so it is close to the input, but for large $\tau$ it lags behind it and smooths it out. This is sometimes described by saying that it **filters** the input. There is an illustration in Fig. 2.

## Numerical solutions

Consider a situation where we want to solve

$$\dot{x} = \frac{1}{\tau}(Ri - x), \tag{11}$$

but can't do it analytically—as will be the case for many values of the time dependent input $i(t)$ or, indeed, if $i(t)$ itself is only known numerically.

### Forward Euler

The simplest numerical approach is the **forward Euler** method. Say you know the value of $x(t)$ and want to approximate $x(t + \Delta)$ where $\Delta$ is a small increment in time. By Taylor expanding $x(t)$ and truncating at first order, we know that

$$x(t + \Delta) = x(t) + \dot{x}(t)\,\Delta + O(\Delta^2). \tag{12}$$

The $O(\Delta^2)$ says there are $\Delta^2$ sized errors in this approximation. These arise because $\dot{x}(t)$ is itself changing but our approximation assumes it is fixed. If we substitute the expression $\dot{x}(t)$ given in right-hand side of Equation (11), we obtain

$$x(t + \Delta) = \frac{1}{\tau}[Ri(t) - x(t)]\Delta + O(\Delta^2). \tag{13}$$

A common implementation of forward Euler uses fixed steps in time, with time bins indexed by $n = 0, 1, 2, \ldots$. We can denote this fixed-stepping form of (13) as

$$x_{n+1} \leftarrow \frac{\Delta}{\tau}(Ri_n - x_n). \tag{14}$$

The "$\leftarrow$" here means "gets" or "assign to", and emphasises that at this point our mathematical equation also serves as pseudo-code for how one might implement fixed-timestep forward Euler in a computer program.

There are more sophisticated approaches like *Runge-Kutta* or *backwards Euler* which are more complex and computational costly for each step, but that are more accurate. In the next subsection, we detail common approach used in computational neuroscience that balances accuracy and simplicity.

### Exponential Euler

In the previous subsection, did it seem odd to you that we numerically approximated a solution to (11) using a first-order Taylor expansion, when we already had an *exact* analytic solution for constant input in Equation (6)?

Why not use this exact solution to solve forward $\Delta$ time-units into the future, and approximate $i(t)$ as constant over this step?

This approach is known as forward **Exponential Euler**. It leads to the fixed-timestep update

$$x_{n+1} \leftarrow [x_n - Ri_n]e^{-\Delta/\tau} + Ri_n. \tag{15}$$

We can also define $\alpha = e^{-\Delta/\tau}$ to write (15) as

$$x_{n+1} \leftarrow x_n\alpha + (1-\alpha)Ri_n. \tag{16}$$

**Forward exponential Euler** can be more stable than forward Euler: Since $\Delta/\tau$ is always positive, $\alpha = e^{-\Delta/\tau}$ is always between 0 and 1. Equation (16) therefore updates $x_{n+1}$ as a **weighted average** of the present $x_n$ and the driving input $Ri_n$. This avoids overshoot and instability. As an exercise, you may wish to simulate what happens if we choose $\Delta \geq 2\tau$ in forward Euler, and compare this to the behaviour of forward exponential Euler for the same step size. You may encounter code resembling Equation (16) in the source code for computational neuroscience papers in that simulate neuronal membrane voltage.

# 1 Summary

Water filling a leaky bucket gives a useful example of the dynamics that will describe a neuron. The idea is that the leak is proportional to the height of the water, so we write:

$$\ell(t) = Gx(t) \tag{17}$$

The volume of water is also proportional to the height, so we write:

$$\text{volume} = Cx(t) \tag{18}$$

where $G$ and $C$ are both constants of proportionality, the $G$ will depend in a complicated way on the viscosity of water, the size of the hole, the $C$ is

more straight-forwardly the cross-sectional area of the bucket. Given that the rate of change of volume is given by water coming in minus water going out, an writing $i(t)$ for the inflow:

$$C\dot{x}(t) = i(t) - Gx(t) \tag{19}$$

This can also be written

$$\tau\dot{x} = R \cdot i(t) - x \tag{20}$$

where $R = 1/G$ and $\tau = C/G$ is a timescale. By solving for constant input and discussing the dynamics we see that the solution to this equation 'chases' the input $Ri$ over a timescale corresponding to $\tau$. Finally we look at numerical approaches: for the Euler approximation, the approximate solution at a time $t_n = n\Delta$ is

$$x_n \leftarrow x_{n-1} + \Delta\ \dot{x}|_{t_n} \tag{21}$$

We also introduced forward exponential Euler method for linear first-order systems that track (filter) an input:

$$x_n \leftarrow \alpha x_{n-1} + (1 - \alpha) \cdot \text{input}, \tag{22}$$

where $\alpha = e^{-\Delta/\tau}$.

## Electrical properties of a neuron

The potential inside a neuron is lower than the potential on the outside; this difference is created by **ion pumps**, small molecular machines that use energy to pump ions across the membrane separating the inside and outside of the cell.

One typical ion pump is Na+/K+-ATPase (Sodium-potassium adenosine triphosphatase); this uses energy in the form of ATP, the energy carrying molecule in the body, and through each cycle it moves three sodium ions out of the cell and two potassium ions into the cell. Since both sodium and potassium ions have a charge of plus one, this leads to a net loss of one atomic charge to the inside of the cell lowering its potential. It also creates an excess of sodium outside the cell and an excess of potassium inside it. We will return to these chemical imbalances later.

The potential difference across the membrane is called the **membrane potential**. At rest a typical value of the membrane potential is $E_L = -70$ mV. It is useful to remember that the excessive sodium is outside the cell and potassium inside, like islands which are surrounded by salty water, as in Fig. 1.

## Spikes

The summary version of what happens in neurons is that **synapses** cause a small increase or decrease in the voltage; **excitatory synapses** cause an increase, **inhibitory synapses** a decrease. This drives the internal voltage dynamics of the cell, these dynamics are what we will learn about here. If the voltage exceeds a threshold, say $v_\theta = -55$ mV there is a nonlinear cascade that produces a **spike** or **action potential**, a spike in voltage 1–2 ms wide which rises above 0 mV before, in the usual description, falling to a reset value of $V_R = -65$ mV, the cell then remains unable to produce another spike for a **refractory period** which may last about 5 ms. We will examine how spikes are formed later, this involves the nonlinear dynamics of ion channels in the membrane; first though we will consider the integrate and fire model which ignores the details of how spikes are produced and simplifies the voltage dynamics.

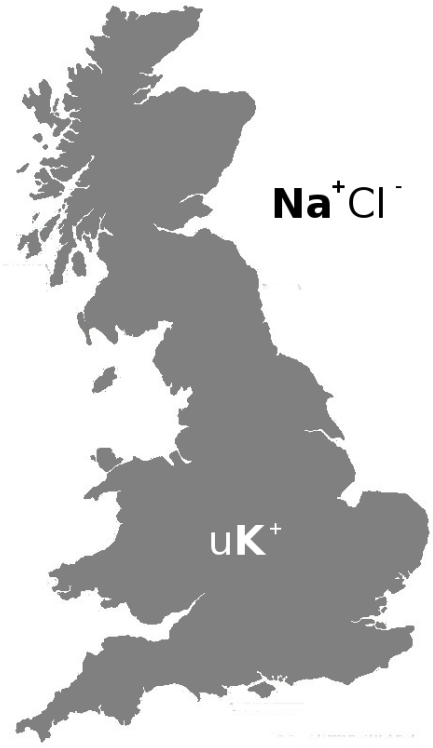



Figure 1: A cartoon to help you remember that there is more sodium outside the cell and more potassium inside.

## The bucket-like equation for neurons

We will now try to extend the bucket-like equation we looked at before so that it applies to neurons. First off we replace $x(t)$, the height of the water, by $v(t)$ the voltage in the cell and $C$ will be replaced by $C_m$, the capacitance of the membrane, the amount of electrical charge that can be stored at the membrane is $C_m v$. The amount of electrical charge is the analogue of the volume of water. Thus, voltage is like height, charge is like the amount of water.

The leak is a bit more complicated, because of the chemical gradients, that is the effects of the differing levels of ions inside and outside the cell along and their propensity to diffuse, the voltage at which there is no leaking of charge is not zero, it is $E_L = -70$ mV, roughly. This is an important aspect of how neurons behave, and one we will encounter again looking at the Hodgkin-Huxley equation.

You might at first expect that if the voltage inside the cell was, say, -60 mV then even if there was a high conductivity for potassium at the membrane, the potassium ions would stay in the cell: they are positive ions after all and so a negative voltage means the electrical force is attracting them to the inside of the cell. However, this isn't quite what happens. There is a high concentration of potassium inside the cell.

Because of the random motion of particles associated with temperature, these have a tendency to diffuse, that is to increase the entropy of the situation by spreading out. It takes a force to counteract this. This is the reversal potential, $E_L$, the voltage required for zero current even if there is some conductivity. It turns out that the normal Ohm's law applies around the reversal potential so that the current out of the cell is proportional to $v - E_L$.

$G$ is now $G_m$, a conductance, measuring the porousness of the membrane to the flow of ions. It gives the constant of proportionality for the leak current: the leak current out of the cell is $G_m(v - E_L)$. We actually divide across by the conductance, and write $R_m = 1/G_m$, the resistance. Finally, we write $\tau_m = C_m/G_m$ to get

$$\tau_m \dot{v} = E_L - v + R_m i(t) \tag{1}$$

$i(t)$ might end up being synaptic input, but traditionally we write the equation to match the **in vivo** experiment where $t$ is an injected current from an electrode, so we write $i_e$, "e" for electrode. $\tau_m$ is a time constant, using the

notation of dimensional analysis we have $[\tau_m] = T$. To check this note that the units of capacitance are charge per voltage: $[C_m] = QV^{-1}$, the units of resistance is voltage per current $[R_m] = VI^{-1}$ and current is charge per time, $[I] = QT^{-1}$ so $[C_m R_m] = T$, time.

The equation above leaves out the possibility that there are other non-linear changes in the currents through the membrane as $v(t)$ changes. This is a problem since, in general, the conductance (inverse: resistance) of voltage-gated ion channels depend on $v(t)$, introducing nonlinear dependence. In fact, the nonlinear effects are strongest for values of $v$ near where a spike is produced.

The **leaky-integrate-and-fire** model handles this spiking nonlinearity in a particularly simple way: It uses the linear equation unless $v$ exceeds a threshold value $v_\theta$. Above threshold, a spike is added 'by hand', and the membrane voltage is changes to a **reset** value $v_r$ to mimic the neuron returning to a hyperpolarized voltage after the spike is complete. To summarise:

- Below the threshold voltage $v_\theta$, the membrane voltage $v(t)$ satisfies $\tau_m \dot{v} = E_L - v + R_m i_e(t)$
- Above threshold ($v \geq v_\theta$), a spike is recorded and the voltage is set to a reset value $v(t) \leftarrow v_r$.

A common choice for $v_r$ is the leak potential. The **leaky integrate and fire model** is surprisingly old, and was first introduced in [1]. It lacks important details in the dynamics of neurons, but is useful and often used for modelling the behaviour of large neuronal networks or for exploring ideas about neuronal computation in a relatively straight-forward setting.

This model is easy to solve. If $i_e$ is constant we have already solved it above, up to messing around with constants:

$$v(t) = (E_L + R_m i_e) + [v(0) - (E_L + R_m i_e)]e^{-t/\tau_m} \tag{2}$$

We can write Equation (2) in terms of a steady-state voltage $\bar{v} = E_L + R_m i_e$ as

$$v(t) = \bar{v} + [v(0) - \bar{v}]e^{-t/\tau_m}. \tag{3}$$

If $i_e$ is not constant it may still be possible to solve the equation, but in any case the equation can be solved numerically on a computer, for example us-

ing a forward exponential Euler update

$$\begin{aligned} v_{t+\Delta} &\leftarrow \alpha v_\theta + (1-\alpha) v_{\text{ss}}(t) \\ \alpha &= e^{-\Delta/\tau_m} \\ v_{\text{ss}}(t) &= E_L + R_m i_e(t). \end{aligned} \tag{4}$$

An example in given in Fig. 2.

One thing to notice is that there are no spikes for low values of the current. Looking at the equation

$$\tau_m \dot{v} = E_L - v + R_m i_e \tag{5}$$

so the equilibrium value for constant $i_e$, the value where $v$ stops changing, is

$$\bar{v} = E_L + R_m i_e \tag{6}$$

Now if this value $\bar{v} > v_\theta$ then as the neuron voltage increased towards its equilibrium value, $\bar{v}$, it would reach the threshold, $v_\theta$, and spike. Hence, if $\bar{v} > v_\theta$ the neuron will spike repeatedly. However if $\bar{v} < v_\theta$ then the neuron will not spike for that input because it will never reach threshold.

In fact, we can work out the curve that represents the firing rate as a function of the current. This is the called the f-I curve and is shown in Fig. 3. In the model

$$\tau_m \dot{v} = E_L + R_m i_e - v = \bar{v} - v \tag{7}$$

which we can solve from our study of ODEs, it gives

$$v(t) = \bar{v} + [v(0) - \bar{v}] e^{-t/\tau_m} \tag{8}$$

so if the neuron has spiked and is reset at time $t = 0$ and reaches threshold at time $t = T$, assume $v_r = E_L$ we have

$$\begin{aligned} v(t) = v_\theta &= \bar{v} + [v_r - \bar{v}] e^{-T/\tau_m} && \text{using assumption } v(0) = v_r \\ &= \bar{v} + [E_L - \bar{v}] e^{-T/\tau_m} && \text{using definition } v_r = E_l \\ &= \bar{v} + [E_L - (E_L + R_m i_e)] e^{-T/\tau_m} && \text{using definition of } \bar{v} \\ &= \bar{v} - R_m i_e \, e^{-T/\tau_m} \end{aligned} \tag{9}$$

so

$$e^{-T/\tau_m} = \frac{\bar{v} - v_\theta}{R_m i_e} \tag{10}$$

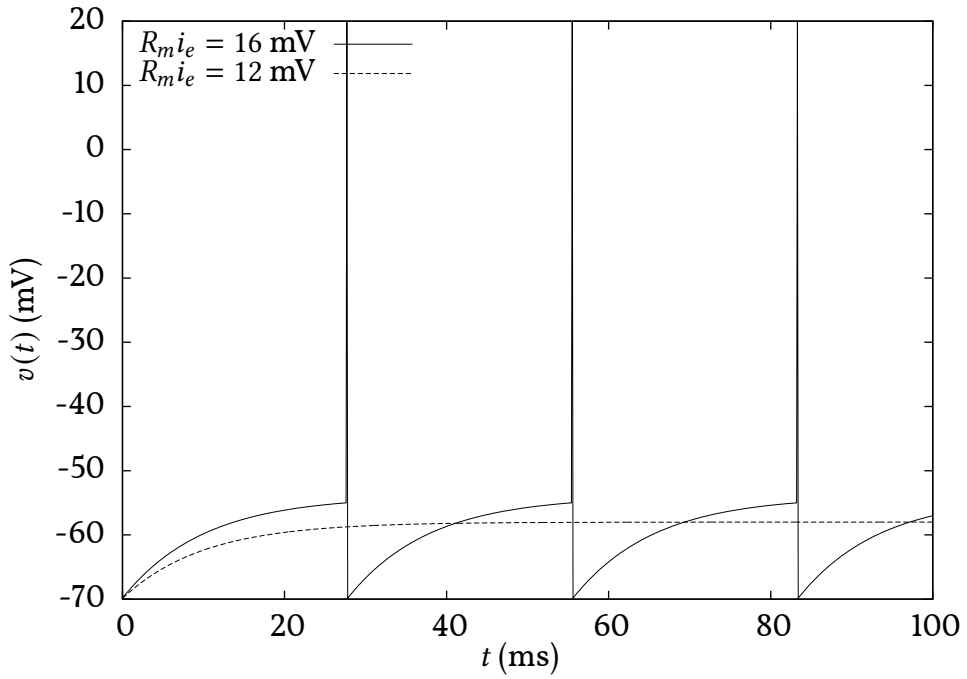



Figure 2: An integrate and fire neuron with different inputs. For $R_m i_e = 12$ mV the voltage relaxes towards the equilibrium value $v = E_L + R_m i_e = -58$ mV. It never reaches the threshold value of $v_\theta = -55$ mV. For $R_m i_e = 16$ mV the voltage reaches threshold and so there is a spike; the spike is added by hand, in this case by setting $V$ to 20 mV for one time step. The voltage is then reset. Here $\tau_m = 10$ ms.

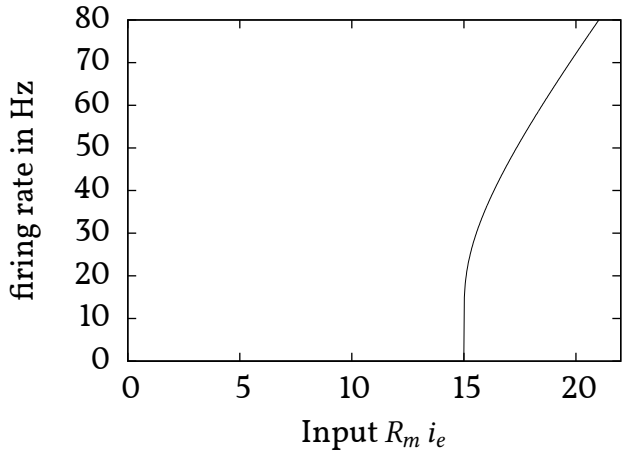



Figure 3: The firing rate, that is spikes per second, for the integrate and fire neuron with different constant inputs with $\tau_m = 10$ ms, $v_\theta = -55$ mV and both the leak and reset given by $-70$ mV. Notice how there is no firing until a threshold is reached and after that the firing increases very quickly.

Taking the log of both sides we get

$$T = \tau_m \ln \left[ \frac{R_m i_e}{\bar{v} - v_\theta} \right] \tag{11}$$

Finally, this is the time between spikes, so the frequency is one over this. It is only defined for $\bar{v} > v_\theta$, below that there is no spiking and the frequency is zero. The actual gnuplot command used to make the figure was

```
plot [0:22] x>15 ? 1/(.01*log(x/(x-15))) : 0
```

We have been ignoring the refractory period, this can easily be added to the model. The easiest approach is just to adjust the rule for spiking by specifying the voltage remains fixed at $v_r$ for some time, $T_r$, corresponding to the refractory period. In fact, the refractory period tends to be divided in the **absolute refractory period** when the neuron can't spike at all and a **relative refractory period** when spiking is just harder. Holding $v$ at $v_r$ models an absolute refractory period, to model a relative refractory period a negative current can be added to the equation.

In fact, neurons tend not to have a regular firing rate; they often, for example, show spike rate adaptation. This means that the firing rate decreases even if the incoming current remains constant. As we have seen this isn't modelling in the integrate and fire model, in the integrate and fire model the response to a constant current is a constant firing rate. The model can be extended to include it. For example a slow potassium current could be added, a slow potassium current is a current that would increase every time there is a spike and decrease between spikes. Since a potassium current reduces the voltage inside the cell, it decreases the firing rate; thus, if the cell fires quickly the potassium current increases, decreasing the firing rate until the increase in the potassium when there is a spike balances the decrease between spikes. These slow currents are common, they may help protect the cell from 'exhaustion' by reducing its spike rate for ongoing stimuli and they may play a computation role, helping the brain infer the nature of stimuli across timescales. The 'face of Jesus' illusion demonstrates the existence of these currents.

A simple approach, which would approximate adding a potassium would be to just add a second input $u$

$$\tau \dot{v} = R_m i_e - v + u \tag{12}$$

where

$$\tau_u \dot{u} = -u \tag{13}$$

where there is an additional rule that $u \leftarrow u + \delta u$ whenever there is a spike. Thus if $u$ is negative it will reduce the equilibrium value of $v$, reducing the spike rate if the cell is spiking. $u$ will decay towards zero with a timescale $\tau_u$ but spiking will change it by $\delta u$. In the case we have discussed here, where we want to model a reduction in spiking $\delta u$ would be negative and $\tau_u$ would have a moderate timescale, perhaps 200 ms.

## Summary

Chemical gradients mean that current flows across the membranes of the cell are not zero for zero potential There is a reversal potential for each type of ion, this is the value of the potential for which no current flows. For our purposes here we consider the leak potential; at lower voltages most of the channels that allow ions to pass through the membrane are closed, this means

the conductance is near zero. The are some open channels, mostly for potassium and the corresponding reversal potential is called the leak potential $E_l$ while the conductance is $g_l$. $E_l$ is often take to be somewhere around -80 mV. Using Ohms law the equation for the voltage in a cell is then

$$C_m \dot{v} = g_l(E_l - v) + i_e(t) \tag{14}$$

where $C_m$ is the capacitance of the membrane and $i_e(t)$ is the input current. This is also written

$$\tau_m \dot{v} = E_l - v + R_m i_e(t) \tag{15}$$

where $\tau_m$ is a timescale, often around 12–20 ms and $R_m = 1/g_l$. However, this linear equation does not include the nonlinear dynamics that supports spiking, we add spiking 'by hand' with a rule that says if $v > v_\theta$, a threshold value, usually about -55 mV, then there is a spike and the voltage is reset to a reset value $v_r$ often set equal to, or a little greater than, $E_l$. This is the leaky integrate and fire equation. We know something of what the solutions look like from the leaky bucket equation considered before, but now there are spikes. We can see that if the equilibrium value $\bar{v} = E_l + R_m i_e$ is greater than $v_\theta$ the neuron will spike regularly, if it is less, it won't spike at all. The equation leaves out the actual spiking dynamics, it also ignores the actual spatial organization of the neuron, treating it as a point. It does not include some other aspects of neuronal spiking like the refractory period and slow currents, though these can be added, at least approximately.

# References

[1] Lapicque, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. J. Physiol. Pathol. Gen, 9:620–635.

## Chemical synapses

Spikes—the voltage pulses that carry signals from neuron to neuron—are notably stereotypical; there aren't big spikes and small spikes, to a good approximation, there are just spikes. However, the effect one neuron has on the other can vary considerably, not just from neuron to neuron, but from time to time. This variability can occur because of chemical synapses, the complicated biochemical machinery responsible for connect the axon of one neuron to the dendrite of another.

Chemical synapses are not the only synapses, there are also **gap junctions**. If an axon is connected to a dendrite by a gap junction there is a small hole directly connecting the inside of one neuron through to the inside of the other, usually this means that the axon of one neuron is connected to the dendrite of the other, though axon to axon gap junctions are also found. For an axon to dendrite gap junction this means that when a spike travelling along the axon reaches the gap junction some of the charged ions diffuse through the gap changing the charge in the dendrite. In some simple animals like jelly fish most or all of the synapses are gap junctions. There are gap junctions in the mammalian brain, for example gap junctions are thought to be responsible for the dynamics which supports very rapid oscillations in the hippocampus, however, most of the synapses in the mammalian brain are chemical synapses. We will see that this allows a more variable effect of a pre-synaptic spike on the voltage of the post-synaptic dendrite.

In a chemical synapse the pre-synaptic spike does not affect the post-synaptic voltage directly, instead it causes a cascade of bio-electrodynamics events which ultimately causes a transient change in conductance of the post-synaptic membrane.

Roughly, the synapse consists of a protuberance in the axon called the **terminal bouton**, the terminal bouton is held by astrocytes, supporting non-neuronal brain cells, so that it is separated by a tiny gap, called the **synaptic cleft** from a protuberance in the dendrite called the **dendritic spine**; depending on the neurons involved this protuberance might be a small bump, or a substantial spine. Figure 1 indicates the range of spine shapes. The shape of the spine is thought to be important in the modulation of synaptic signalling; this isn't an aspect of synapses we will consider here.

The terminal bouton is filled with tiny bags or bubbles called **vesicles**, these

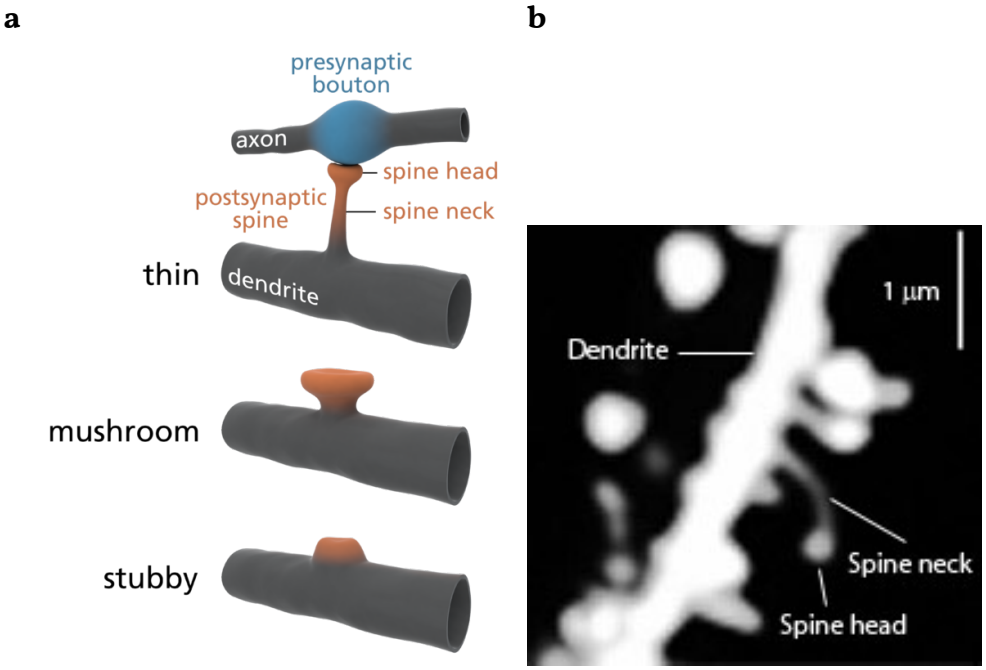



Figure 1: Types of dendritic spines: **a** shows a set of different spine types, **b** shows a photograph of a spiny dendrite of a striatal medium spiny neuron. [Both pictures from `https://en.wikipedia.org/wiki/Dendritic_spine`]

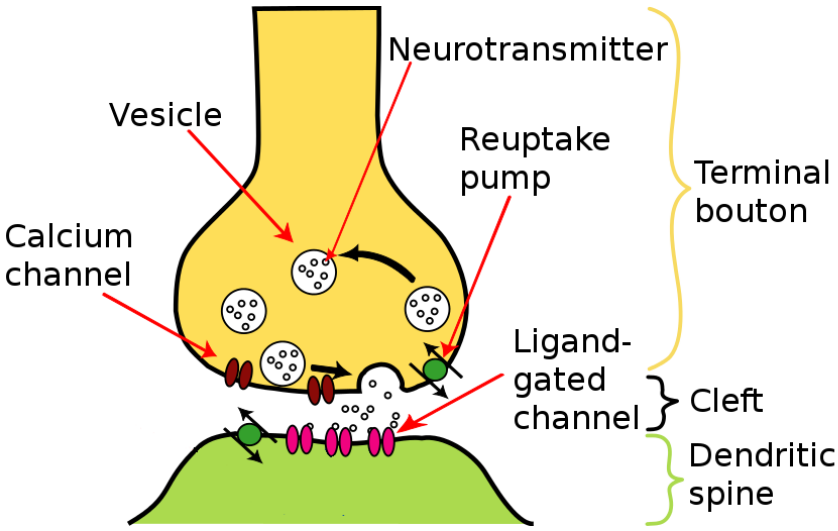



Figure 2: The major parts of the synapse; this shows a vesicle bursting, releasing neurotransmitter into the cleft, this will bind with the ligand-gated channels to allow a current across the membrane of the dendrite. Reuptake pumps are shown in the bouton and the spine, there are also pumps in the astrocyte that surrounds the cleft but isn't shown here. Some is also lost to diffusion. [Diagram modified from one in wikipedia.]

contain special molecules called **neurotransmitters**. When a spike arrives at the terminal bouton it causes calcium gates to open in the cellular membrane, the resulting influx of calcium ions causes some of the vesicles to migrate to the membrane separating the bouton from the synaptic cleft, they burst releasing neurotransmitter into the cleft.

The membrane of the dendritic is pieced by gated ion channels; these are **ligand gated** channels. This means that they contain a receptor site which binds with a particular type of molecule, like a key designed for the receptor site's lock. When the receptor has a molecule bound to it, the gate is open and so ions can pass through the channel, like the other channels we have seen the channel is ion specific, so only one type of ion can pass through it. In the case of the ligand-gated channels in the dendritic spine, the neurotransmitter binds with the receptor, opening the gate. Hence, after a spike arrives at the synapse the cleft is filled with neurotransmitter and some of that neurotransmitter binds to the gated channels, causing them to open. This in turn

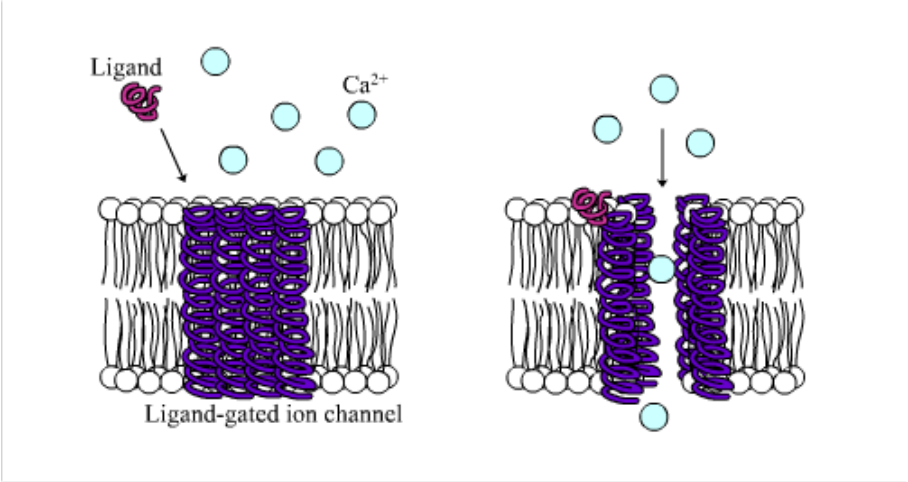



Figure 3: Sketch of a ligand-gated channel; when the neurotransmitter, the ligand, binds to the gate it opens allowing the ions to pass through. This is a Calcium channel, calcium, like sodium, is found in higher concentrations outside the cell so the chemical gradient, as well as the voltage gradient, means these ions flow into the cell from outside, increasing its potential. This is therefore part of an excitatory synapse. Inhibitory synapses have ligand-gated potassium and chlorine channels, potassium, as we have discussed, is at a higher concentration inside the cell and so will flow out, depending on the voltage gradient, lowering the potential inside the cell. Chlorine is at a higher concentration outside the cell but is a negative ion, so when it flows in it also lowers the potential. [Diagram modified from wikipedia.]

allows a flow of ions in or out of the dendrite, changing the voltage there. A cartoon of a ligand-gated channel is given in Fig. 3.

Which ion and which direction, depends on the synapses, we will return to that. For now, though, let us continue describing what happen; after the neurotransmitter floods the cleft it is quickly reabsorbed through neurotransmitter reuptake pumps. Some of the neurotransmitter is absorbed into the bouton, some into the spine and some is absorbed by the astrocyte, the important thing is that the concentration of neurotransmitter in the cleft falls rapidly. Now, the fluid of the cleft has little neurotransmitter, but there is still neurotransmitter bound to the receptors of the ligand gated channels.

This gradually unbinds, this is usually imagined to be a random process, because of the Brownian motion of molecules in the fluid of the cleft and the thermal vibration of the receptor itself, the neurotransmitters unbind as the result of random collisions and thermal variations. As they do so, the channels close again and the conductivity of the dendritic spine's membrane falls back towards zero.

## Post-synaptic potential

If we put aside messy nuances such as neuromodulation and co-release of multiple neurotransmitters, the vast majority of neurons the adult brain can be roughly classed as excitatory or inhibitory. Excitatory neurons release neurotransmitters that open sodium or calcium channels in their targets—allowing positive ions to enter the dendrite and increasing membrane voltage. Inhibitory neurons release neurotransmitters that open potassium or chloride channels, which to pull the post-synaptic cell's membrane voltage toward a value below threshold. All of this is a simplification, and many exceptions exist, but it remains a useful simplification.

The post-synaptic change in potential that results from a pre-synaptic spike is called a **post-synaptic potential**; if the synapse is excitatory this is called an **excitatory post-synaptic potential** or EPSP, if it is inhibitory it is called an **inhibitory post-synaptic potential** or IPSP. The profile of PSPs reflects the neurotransmitter dynamics, it rises fast as the neurotransmitter floods the cleft and the ion-channels open, it then decays back to zero following an exponential decay, reflecting the constant rate unbinding process: since any bound molecule has a constant probability of shaking free the number of unbinding events depends on the number of bound molecules, giving an exponential decay.

Often the post-synaptic conductivity is taken to be a what is called an **alpha** function:

$$i_s(t) = g_s s(t)(E_s - v) \tag{1}$$

where $i_s(t)$ is the synpatic current, $E_s$ is the reversal potential of the synapse and $g_s s(t)$ is the conductance, $g_s$ is a constant describing the strength of the synapse and $s(t)$ is

$$s(t) = te^{-t/\tau_s} \tag{2}$$

where $\tau_s$ is a time scale, see Fig. 4.

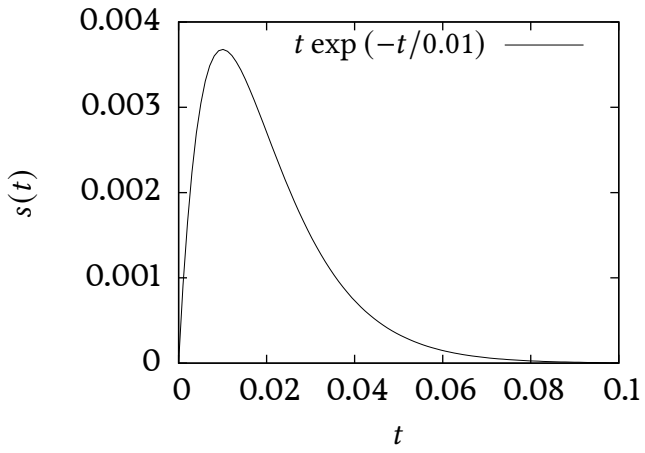



Figure 4: The $\alpha$-function profile often used to model synaptic conductances, shown here with $\tau_m = 10$ ms.

The rising part of the $\alpha$ function models the period when there is neurotransmitter in the cleft, this is binding to the channels increasing the conductance; the falling part represents the period where the unbound neurotransmitter has been cleared from the cleft and the bound neurotransmitter is unbinding randomly due to the thermal motion of molecules. It is possible to understand these dynamics in terms of the bucket-like equations we have examined before, but this won't be done here. It is also common to leave out the rising part and just model the conductance as

$$\tau_s \dot{s} = -s \tag{3}$$

with

$$s(t) \rightarrow s(t) + 1 \tag{4}$$

whenever there is a spike. This is what is done in the coursework, for example.

# Synaptic plasticity

Synaptic plasticity usually refers to the long-term changes in synapse strength, an 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 behaviour 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.

The closest thing to an overall rule was formulated by Hebb in 1949 when he said [2]:

> 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 neural networks the nodes, modelling neurons, often lack spiking dynamics and so 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 favour of

> Neurons that fire together wire together.

This leads to a plasticity rule

$$\delta w_{ij} = \eta x_i x_j \tag{1}$$

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 $\eta$ is a learning rate. Another version is

$$\delta w_{ij} = \eta (x_i - \theta)(x_j - \theta) \tag{2}$$

where $\theta$ is a threshold, this allows negative changes, when $x_i > \theta$ and $x_j < \theta$, or visa versa.

## Spike-timing dependent plasticity

The late nineties saw a revival of interest in causal, spike-timing dependent plasticity (STDP). A series of papers pointed to experimental evidence for timing effects in plasticity [3, 4, 5, 6, 7] including a definitive demonstration of a STDP changes **in vitro** in [4], the observation of asymmetric STDP **in vivo** in developing Xenopus in [8] and a clear graph of the time dependence of plastic changes **in vitro** in [9].

The famous graph of STDP is shown in Fig. 1. This shows measurements of plastic changes made in an **in vitro** preparation. Electrodes are inserted into two synaptically connected cells and currents are used to cause both to spike periodically with a gap of $\delta t$ between the pre- and post-synaptic spikes. This causes the synaptic strength to change, if the pre-synaptic spike precedes the post-synaptic spike the synapse gets stronger with the degree of strengthening depending on the size of $|\delta t|$, the bigger the gap the smaller the effect with a roughly exponential profile. The opposite is observed if the pre-synaptic spike arrives after the post-synaptic spike has left, in this case the synapse gets weaker, again the size of the effect falls like an exponential as $|\delta t|$ gets bigger.

There are lots of caveats to be added to this, it is quite an artificial situation, in vitro with periodic spiking; since the changes are only tracked over a show period it isn't clear whether the changes are additive

$$w \leftarrow w + \delta w \tag{3}$$

or multiplicative

$$w \leftarrow \lambda_w w \tag{4}$$

However, it does give a striking picture of how STDP might work.

In fact, an example of how STDP might support unsupervised learning was given in [10, 11]. A toy model is introduced with multiple neuron inputs feeding forward to a single integrate-and-fire neuron. The inputs are divided into two groups and given a correlation structure so that inputs in the same group are more likely to spike at roughly the same time. This is sketched in Fig. 2. The synapses are then adjusted according to a simple STDP rule. It is seen that one of the two groups 'wins out', its synapses get stronger while the synapses corresponding to the other group gets weaker. Basically if one group, by chance, is slightly more likely to cause the post-synaptic neuron to

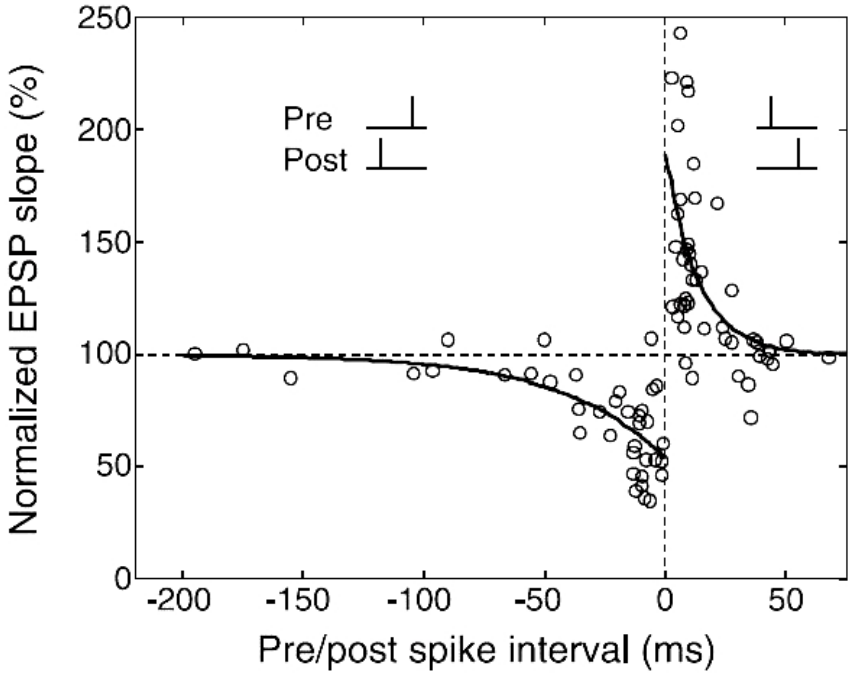



Figure 1: Spike timing dependent plasticity. This shows the change in synapse strength as a function of the timing gap between the pre- and post-synaptic spike.

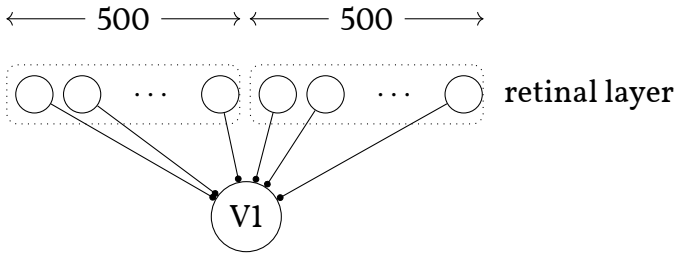



Figure 2: The STDP of Song and Abbott. 1000 input neurons, refered to as retinal neurons, feed forward to a single V1 neuron. The retinal neurons are divided into two groups: the first 500 and the second 500, the two groups provide noisy output, these give the input to the V1 neuron. The inputs from neurons in the same group are correllated, meaning they are more likely to be similar to each other in their activity than neurons from different groups.

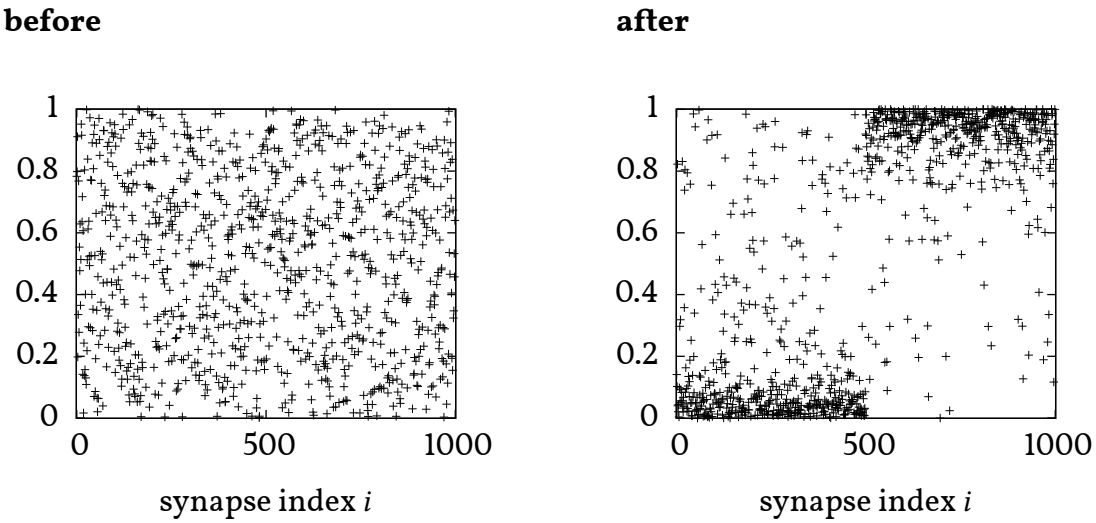



Figure 3: Synapse strengths before and after in Song and Abbotts simple model. At first they are all random, after the STDP has had an effect the synapses from one of the two groups have approached their maximum value, the others are near zero.

spike than the other, then the post-synaptic spikes are more likely to occur after the presynaptic spikes for that group, so the synapses will get stronger, increasing the effect.

# References

[1] Abbott LF, Varela JA, Sen K and Nelson SB. (1997) Synaptic depression and cortical gain control. Science, 275: 221–224.

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

[3] Markram H, Sakmann B (1995) Action potentials propogating back into dendrites triggers changes in efficacy of single-axon synapses between layer V pyramidal cells. Society of Neuroscience Abstract 21.

[4] Markram H, Lübke J, Frotscher M, Sakmann B (1997) Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. Science 275: 213-215.

[5] Bell CC, Han VZ, Sugawara Y, Grant K (1997) Synaptic plasticity in a cerebellum-like structure depends on temporal order. Nature 387: 278–281.

[6] Magee JC, Johnston D (1997) A synaptically controlled, associative signal for Hebbian plasticity in hippocampal neurons. Science 275: 209–213.

[7] Debanne D, Gähwiler BH, Thompson SM (1998) Long-term synaptic plasticity between pairs of individual ca3 pyramidal cells in rat hippocampal slice cultures. Journal of Physiology 507: 237–247.

[8] Zhang LI, Tao HW, Holt CE, Harris WA, Poo MM (1998). A critical window for cooperation and competition among developing retinotectal synapses. Nature 395: 37–44.

[9] Bi G, Poo M (1998) Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience 18: 10464–10472.

[10] Song S, Miller K, Abbott L (2000) Competitive hebbian learning through spike-timing-dependent synaptic plasticity. Nature Neuroscience 3: 919–926.

[11] Song S, Abbott L (2001) Cortical development and remapping through spike timing-dependent plasticity. Neuron 32: 339-350.

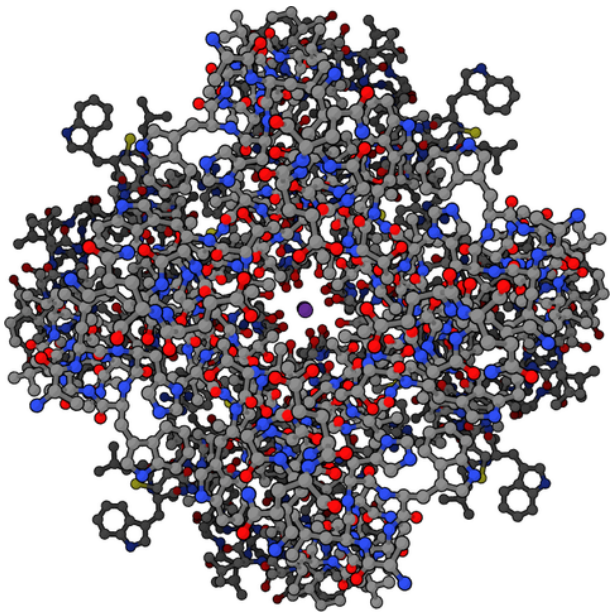


Figure 1: An open potassium channel, picture from wikipedia, which in turn took it from the Protein Data Bank. `http://en.wikipedia.org/wiki/Potassium_channel`

## Gated channels

The nonlinear dynamics that neurons rely on to form spikes arise from the **voltage-gated channels**; these are ion channels whose conductance varies as the voltage varies. They are tiny molecular machines which, crucially, are ion selective: only sodium ions can pass through a sodium gate, only potassium ions through a potassium gate. Each individual gate has a number of different gating states. We will briefly examine this, but ultimately each one is either open or closed, the overall smooth, though rapid, variation in these conductances comes from average a large number of individual discrete step-like changes as the individual gates open and close.

The potassium channel is a **persistent** gate; this is actually a little complicated, but roughly speaking it has one type of closed state and one type of open state; the sodium channel, which we will look at after the potassium channel also has one open state, but it has two types of closed states.

The potassium gate is actually composed of four independent subgates, all

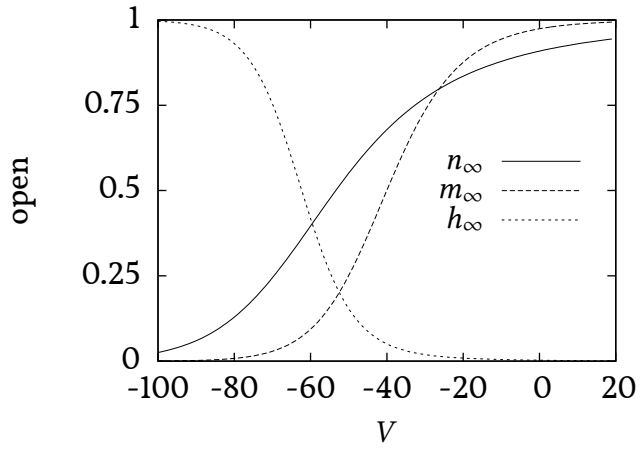



Figure 2: The asymptotic values of the gating probabilities.

these gates must be open to allow potassium ions through, but each has its own independent dynamics. The membrane is usually modelled as having overall potassium conductance

$$g_K = \bar{g}_K n^4 \tag{1}$$

where $\bar{g}_K$ would be conductance if all the channels were open and $n$ is the probability an individual subgate is open so $n^4$ is the probability an individual gated channel is open. The dynamical equation for $n$ is quite complicated, it is of the standard form

$$\tau_n(v)\dot{n} = n_\infty(v) - n(t) \tag{2}$$

If $\tau_n(v)$ and $n_\infty(v)$ where constant this would be simple, $n(t)$ would decay to $n_\infty$ with a timescale of $\tau_n$, however they aren't constants, they are functions of the membrane potential.

A graph of $n_\infty(v)$ is shown in Fig. 2. We can see that $n$ is small when the voltage is near the resting value but climbs towards one as $v$ increases. Now, $n$ isn't equal to $n_\infty$, rather it decays towards it with a time constant given

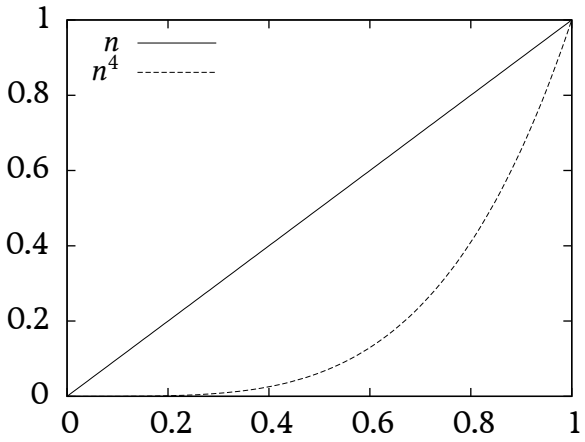



Figure 3: The fourth power gives crisper behavior than $n$ itself would.

by $\tau_n(v)$, but we can see that the potassium channels open as the voltage increases. Before looking at how these play a role in spiking we will look at the sodium gates. It is worth noting though the way having four independent gates makes the dynamics crisper: if $n$ is near zero, $n^4$ is very small indeed. This is illustrated in Fig. 3.

We also need to discuss reversal potentials. You would expect the flow of potassium to be determined by $g_K v$, but it isn't. Because there are more potassium ions inside the cell than outside they would flow out even if $v = 0$. In fact, as we discussed when looking at the integrate and flow model, we assume this doesn't change Ohm's law, the relationship between potential difference and current, rather, it just changes the zero point:

$$i_K = g_K(E_K - v) \tag{3}$$

where $E_K = -70$ mV, approximately, is called the reversal potential and can be calculated using an equation called the Nernst equation.

The sodium channel is called a transient channel because it has two closed states and one open one; generally its dynamics during the spike is

$$\text{closed I} \rightarrow \text{open} \rightarrow \text{closed II} \tag{4}$$

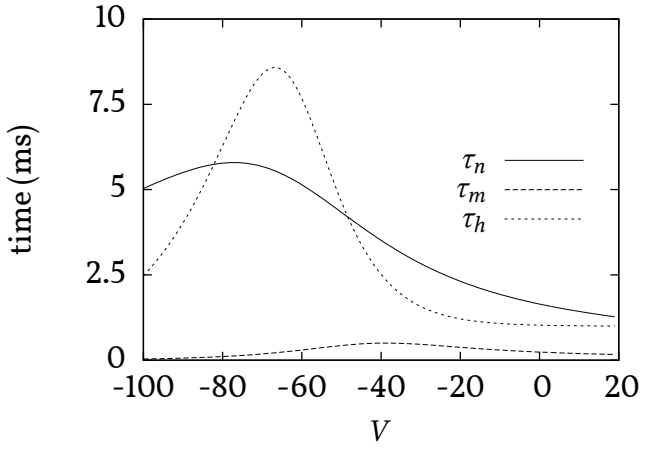



Figure 4: The time constants for the gating probabilities.

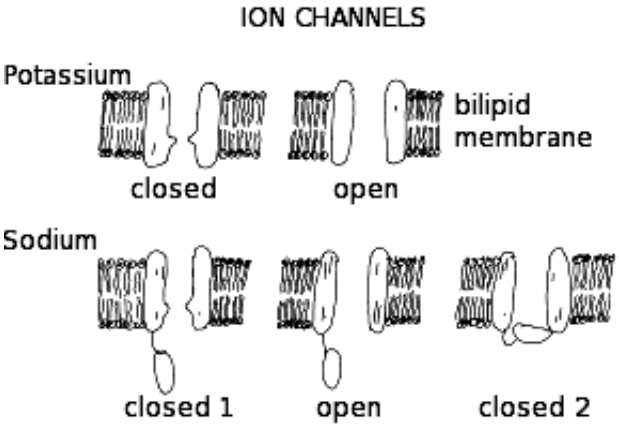



Figure 5: A cartoon of the two types of voltage gated channel we are discussing; the persistent potassium channels open as the voltage rise, the transient sodium channel opens and then closes again.

After that there is a slower process of resetting. The part of the gate that is closed to give the initial closed state is very like the potassium gate, but with three sub-gates; the probability of these sub-gates being open is usually called $m$; the other part, the gate that closes to give the second closed state is different in that it is not made of sub-gates, its probability of being open is usually called $h$ and its asymptotic value, $h_\infty$, is near one for lower $V$ and near zero for larger. A cartoon of all this is given in Fig. 5. Fig. 2 includes graphs of $m_\infty(V)$ and $h_\infty(V)$. Finally, the reversal potential for sodium is $E_{\text{Na}} = 50$ mV. The sodium current is therefore

$$i_{\text{Na}} = g_{\text{Na}}(E_{\text{Na}} - v) \tag{5}$$

with

$$g_{\text{Na}} = \bar{g}_{\text{Na}} m^3 h \tag{6}$$

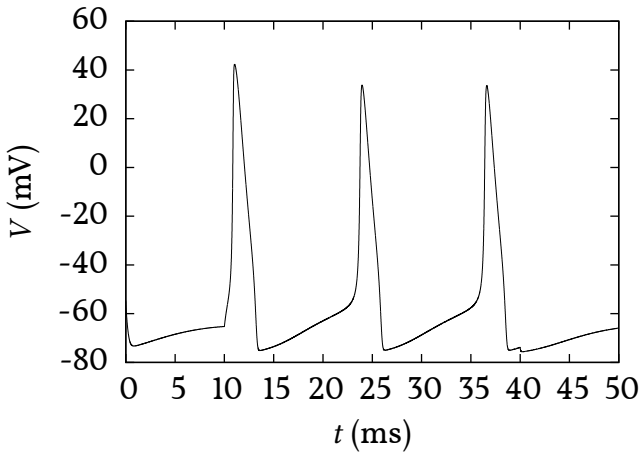
All of this together gives the Hodgkin-Huxley equation, which equates the rate of change of $v$ to a set of currents, the leak current giving the roughly linear behaviour below threshold we saw in the integrate and fire model and the gated channels forming the spike.

$$C_m \dot{v} = \text{currents} \tag{7}$$

We can now give a rough description of how spikes are formed. The time constants $\tau$ for the three gating probabilities are given in Fig. 4, these are quite complicated, but the key thing is that $\tau_m$ is very small, no matter what the value of $V$ is. This means that $m$ stays very close to its asymptotic value $m_\infty$. As $V$ approaches the threshold of about $-55$ mV, $m$ increases towards one, with $m^3$ increasing even more dramatically. Opening the sodium gates allows sodium to flood the cell, increasing the $V$ further and further opening the gates. This gives the rapid upswing in voltage, the rising part of the spike. The other two gating probabilities have slower dynamics and it takes $n$ and $h$ a while to catch up with $n_\infty$ and $h_\infty$. However, as $h$ decreases, it closes the sodium gates again, preventing more sodium getting in to the cell; $n$ increases opens the potassium gates, potassium flows out reducing the $V$ again, back towards -70 mV. This gives the downswing of the spike. Afterwards everything resets. An example spike is shown in Fig. 6.

For a more accurate model further channels, and therefore further currents, can be added, other sodium and potassium channels with different dynamics, or a calcium channel. It is also common to investigate models 'between'

**A**




**B**

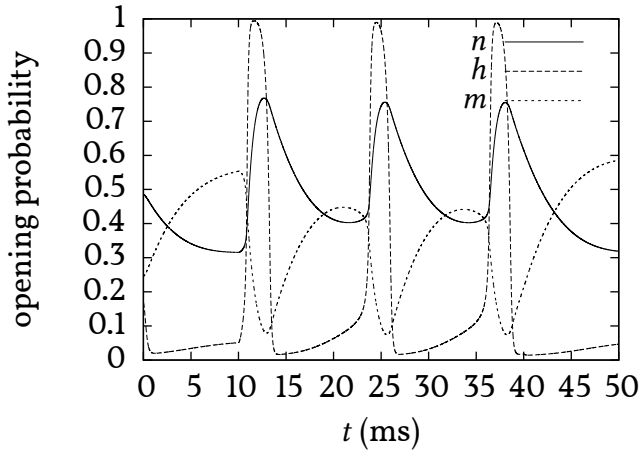



Figure 6: Some spikes in the HH model. **A** shows the spikes produced by a standard HH model in response to a current input. **B** shows how the gating probabilities vary during the spike, in this graph $h$ and $m$ are labelled the wrong way around.

the integrate and fire model and the Hodgkin-Huxley equation which add some of the nonlinearity to the integrate and fire dynamics.

## Summary

There are voltage-dependant channels in the cell membrane. These are modelled with the equation

$$\tau_\ell(v)\dot{\ell} = \ell_\infty(v) - \ell \tag{8}$$

where $\ell$ is standing in for one of the so called gating probabilities. Here we consider the channels found in the giant axon of the squid. This has a potassium channel with conductance:

$$g_{\mathrm{K}} = \bar{g}_{\mathrm{K}} n^4 \tag{9}$$

with gating probability $n$ and a sodium channel with conductance

$$g_{\mathrm{Na}} = \bar{g}_{\mathrm{Na}} m^3 h \tag{10}$$

with gating probabilities $m$ and $h$. In the case of the the sodium channel the timescale for $m$ is small so $m$ closely tracks $m_\infty$; this rises towards one as voltage increase beyond the threshold, so, provided $h$ is not close to zero, the sodium channel opens as the voltage crosses zero, since the reversal potential for sodium is high, this means sodium rushes in, causing $m$ to increase still further. However, $h$ does not remain above zero, as the votage increases $h_\infty$ falls to zero and $\tau_\infty$ decreases, causing the sodium conductance to fall. At the same time, the potassium conductance increases, since the reversal potential for potassium is low this causes a current out of the cell, pushing the voltage back down. Thus a spike is formed.

## References

[1] Hodgkin, AL and Huxley HF (1952) "Propagation of electrical signals along giant nerve fibres." Proceedings of the Royal Society of London. Series B, Biological Sciences 140: 177-183.