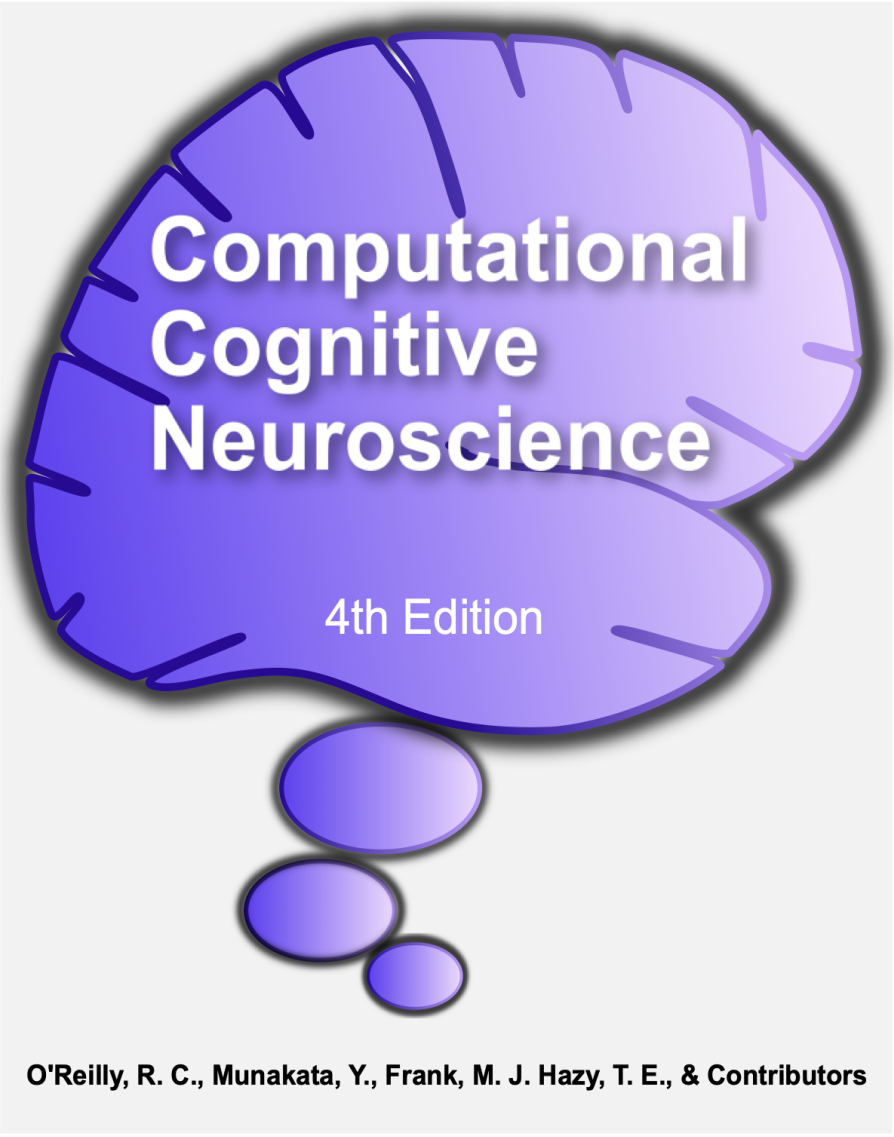
Computational Cognitive Neuroscience

Randall C. O'Reilly Yuko Munakata Michael J. Frank Thomas E. Hazy
Contributors



O'Reilly, R. C., Munakata, Y., Frank, M. J. Hazy, T. E., & Contributors

Contents

To our families.

Preface

This is the fourth edition of this book. The source of this book is hosted on:

https://github.com/CompCogNeuro/ed4

The hands-on simulations to explore the models described in this book are available at:

https://github.com/CompCogNeuro/sims

Earlier editions were hosted via a MediaWiki platform at:

https://grey.colorado.edu/CompCogNeuro/index.php/CCNBook/Main

Chapter 1: Introduction

You are about to embark on one of the most fascinating scientific journeys possible: inside your own brain! We start this journey by understanding what individual **neurons** (Chapter 2) in your neocortex do with the roughly 10,000 synaptic input signals that they receive from other neurons. The **neocortex** is the most evolutionarily recent part of the brain, which is also most enlarged in humans, and is where most of your thinking takes place. The numbers of neurons and synapses between neurons in the neocortex are astounding: roughly 20 billion neurons, each of which is interconnected with roughly 10,000 others. That is several times more neurons than people on earth. And each neuron is far more social than we are as people – estimates of the size of stable human social networks are only around 150-200 people, compared to the 10,000 for neurons.

We've got a lot going on under the hood. At these scales, the influence of any one neuron on any other is relatively small. We'll see that these small influences can be shaped in powerful ways through *learning mechanisms* (Chapter 4), to achieve complex and powerful forms of information processing. And this information processing prowess does not require much complexity from the individual neurons themselves – fairly simple forms of information integration both accurately describe the response properties of actual neocortical neurons, and enable sophisticated information processing at the level of aggregate neural *networks* (Chapter 3).

After developing an understanding of these basic neural information processing mechanisms in Part I of this book, we continue our journey in Part II by exploring many different aspects of human thought (cognition), including perception and attention (Chapter 6) motor control and reinforcement learning (Chapter 7), learning and memory (Chapter 8), language (Chapter 9), and executive function (Chapter 10). Amazingly, all these seemingly different cognitive functions can be understood using the small set of common neural mechanisms developed in Part I. In effect, our neocortex is a fantastic form of silly putty, which can be molded by the learning process to take on many different cognitive tasks. For example, we will find striking similarities across different brain areas and cognitive functions – the development of primary visual cortex turns out to tell us a lot about the development of rich semantic knowledge of word meanings!

Some Phenomena We'll Explore

Here is a list of some of the cognitive neuroscience phenomena we'll explore in Part II of the book:

- **Vision:** We can effortlessly recognize countless people, places, and things. Why is this so hard for robots? We will explore this issue in a network that views natural scenes (mountains, trees, etc.), and develops brain-like ways of encoding them using principles of learning.
- **Attention:** Where's Waldo? We'll see in a model how two visual processing pathways work together to help focus our attention in different locations in space (whether we are searching for something or just taking things in), and why damage to one of these pathways leads people to ignore half of space.
- **Dopamine and Reward:** Why do we get bored with things so quickly? Because our dopamine system is constantly adapting to everything we know, and only gives us rewards when something new or different occurs. We'll see how this all happens through interacting brain systems that drive phasic dopamine release.
- **Episodic memory:** How can damage to a small part of our brain cause amnesia? We'll see how in a model that replicates the structure of the hippocampus. This model provides insight into why the rest of the brain isn't well-suited to take on the job of forming new episodic memories.
- **Reading:** What causes dyslexia, and why do people who have it vary so much in their struggles with reading? We'll explore these issues in a network that learns to read and pronounce nearly 3,000 English words, and generalizes to novel nonwords (e.g., "mave" or "nust") just like people do. We'll see why damaging the network in different ways simulates various forms of dyslexia.
- **Meaning:** "A rose is a rose is a rose." But how do we know what a rose is in the first place? We'll explore this through a network that "reads" every paragraph in a textbook, and acquires a surprisingly good semantic understanding by noting which words tend to be used together or in similar contexts.
- **Task directed behavior:** How do we stay focused on tasks that we need to get done or things that we need to pay attention to, in the face of an ever-growing number of distractions (like email, text

messages, and tweets)? We'll explore this issue through a network that simulates the "executive" part of the brain, the prefrontal cortex. We will see how this area is uniquely-suited to protect us from distraction, and how this can change with age.

The Computational Approach

An important feature of our journey through the brain is that we use the vehicle of *computer models* to understand cognitive neuroscience (i.e., *Computational Cognitive Neuroscience*). These computer models enrich the learning experience in important ways – we routinely hear from our students that they didn't really understand anything until they pulled up the computer model and played around with it for a few hours. Being able to manipulate and visualize the brain using a powerful 3D graphical interface brings abstract concepts to life, and enables many experiments to be conducted easily, cleanly, and safely in the comfort of your own laptop. This stuff is fun, like a video game – think "sim brain", as in the popular "sim city" game from a few years ago.

At a more serious level, the use of computer models to understand how the brain works has been a critical contributor to scientific progress in this area over the past few decades. A key advantage of computer modeling is its ability to wrestle with complexity that often proves daunting to otherwise unaided human understanding. How could we possibly hope to understand how billions of neurons interacting with 10's of thousands of other neurons produce complex human cognition, just by talking in vague verbal terms, or simple paper diagrams? Certainly, nobody questions the need to use computer models in climate modeling, to make accurate predictions and understand how the many complex factors interact with each other. The situation is only more dire in cognitive neuroscience.

Nevertheless, in all fields where computer models are used, there is a fundamental distrust of the models. They are themselves complex, created by people, and have no necessary relationship to the real system in question. How do we know these models aren't just completely made-up fantasies? The answer seems simple: the models must be constrained by data at as many levels as possible, and they must generate predictions that can then be tested empirically. In what follows, we discuss different approaches that people might take to this challenge – this is intended to give a sense of the scientific approach behind the work described in this book – as a student this is perhaps not so relevant, but it might help give some perspective on how science really works.

In an ideal world, one might imagine that the neurons in the neural model would be mirror images of those in the actual brain, replicating as much detail as is possible given the technical limitations for obtaining the necessary details. They would be connected exactly as they are in the real brain. And they would produce detailed behaviors that replicate exactly how the organism in question behaves across a wide range of different situations. Then you would feel confident that your model is sufficiently "real" to trust some of its predictions.

But even if this were technically feasible, you might wonder whether the resulting system would be any more comprehensible than the brain itself! In other words, we would only have succeeded in transporting the fundamental mysteries from the brain into our model, without developing any actual understanding about how the thing really works. From this perspective, the most important thing is to develop *the simplest possible model that captures the most possible data* – this is basically the principle of *Ockham's razor*, which is widely regarded as a central principle for all scientific theorizing.

In some cases, it is easy to apply this razor to cut away unnecessary detail. Certainly many biological properties of neurons are irrelevant for their core information processing function (e.g., cellular processes that are common to all biological cells, not just neurons). But often it comes down to a judgment call about what phenomena you regard as being important, which will vary depending on the scientific questions being addressed with the model.

The approach taken for the models in this book is to find some kind of happy (or unhappy) middle ground between biological detail and cognitive functionality. This middle ground is unhappy to the extent that researchers concerned with either end of this continuum are dissatisfied with the level of the models. Biologists will worry that our neurons and networks are overly simplified. Cognitive psychologists will be concerned that our models are too biologically detailed, and they can make much simpler models that capture the same cognitive phenomena. We who relish this "golden middle" ground are happy when we've achieved

important simplifications on the neural side, while still capturing important cognitive phenomena. This level of modeling explores how consideration of neural mechanisms inform the workings of the mind, and reciprocally, how cognitive and computational constraints afford a richer understanding of the problems these mechanisms evolved to solve. It can thus make predictions for how a cognitive phenomenon (e.g., memory interference) is affected by changes at the neural level (due to disease, pharmacology, genetics, or similarly due to changes in the cognitive task parameters). The model can then be tested, falsified and refined. In this sense, a model of cognitive neuroscience is just like any other 'theory', except that it is explicitly specified and formalized, forcing the modeler to be accountable for their theory if/when the data don't match up. Conversely, models can sometimes show that when an existing theory is faced with challenging data, the theory may hold up after all due to a particular dynamic that may not be considered from verbal theorizing.

Ultimately, it comes down to aesthetic or personality-driven factors, which cause different people to prefer different overall strategies to computer modeling. Each of these different approaches has value, and science would not progress without them, so it is fortunate that people vary in their personalities so different people end up doing different things. Some people value simplicity, elegance, and cleanliness most highly – these people will tend to favor abstract mathematical (e.g., Bayesian) cognitive models. Other people value biological detail above all else, and don't feel very comfortable straying beyond the most firmly established facts – they will prefer to make highly elaborated individual neuron models incorporating everything that is known. To live in the middle, you need to be willing to take some risks, and value most highly the process of *emergence*, where complex phenomena can be shown to emerge from simpler underlying mechanisms.

The criteria for success here are a bit murkier and subjective – basically it boils down to whether the model is sufficiently simple to be comprehensible, but not so simple as to make its behavior trivial or otherwise so fully transparent that it doesn't seem to be doing you any good in the first place. One last note on this issue is that the different levels of models are not mutually exclusive. Each of the low level biophysical and high level cognitive models have made enormous contributions to understanding and analysis in their respective domains (much of which is a basis for further simplification or elaboration in the book). In fact, much ground can be (and to some extent already has been) gained by attempts to understand one level of modeling in terms of the other. At the end of the day, linking from molecule to mind spans multiple levels of analysis, and like studying the laws of particle physics to planetary motion, require multiple formal tools.

Emergent Phenomena

What makes something a satisfying scientific explanation? A satisfying answer is that you can explain a seemingly complex phenomenon in terms of simpler underlying mechanisms, that interact in specific ways. The classic scientific process of *reductionism* plays a critical role here, where the complex system is reduced to simpler parts. However, one also needs to go in the opposite, oft-neglected direction, *reconstructionism*, where the complex system is actually reconstructed from these simpler parts. Often the only way to practically achieve this reconstruction is through computational modeling. The result is an attempt to capture the essence of emergence.

Emergence can be illustrated in a very simple physical system, two interacting gears, as shown in Figure 1.1. It is not mysterious or magical. On the other hand, it really is. You can make the gears out of any kind of sufficiently hard material, and they will still work. There might be subtle factors like friction and durability that vary. But over a wide range, it doesn't matter what the gears are made from. Thus, there is a level of *transcendence* that occurs with emergence, where the behavior of the more complex interacting system does not depend on many of the detailed properties of the lower level parts. In effect, the interaction itself is what matters, and the parts are mere place holders. Of course, they have to be there, and meet some basic criteria, but they are nevertheless replaceable.

Taking this example into the domain of interest here, does this mean that we can switch out our biological neurons for artificial ones, and everything should still function the same, *as long as we capture the essential interactions in the right way?* Some of us believe this to be the case, and that when we finally manage to put enough neurons in the right configuration into a big computer simulation, the resulting brain will support consciousness and everything else, just like the ones in our own heads. One interesting further question arises: how important are all the interactions between our physical bodies and the physical environment? There is good reason to believe that this is critical. Thus, we'll have to put this brain in a robot. Or perhaps

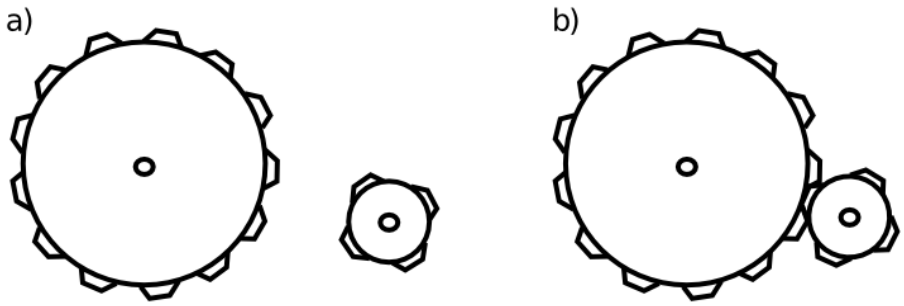


Figure 1.1: The principle of *emergence*, simply illustrated. The gears on the left do not interact, and nothing interesting happens. However, on the right, the interaction between the gears produces interesting, useful phenomena that *cannot* be reduced to the individual gears *separately*. For example, the little gear will spin faster, but the larger one will have higher torque at its axel – these properties would be entirely different if either gear interacted with a different sized gear. Furthermore, the material that the gear is made from really doesn't matter very much – the same basic behavior would be produced by plastic, metal, wood, etc. Thus, even in this simple case, there is something just slightly magical and irreducible going on – when two gears get together, something emerges that is more than the sum of the parts, and exists in a way independent of the parts, even while being entirely dependent on actually *having* those parts to make it happen. This is a good analogy for the relationship between the mind and the brain, and computer models can capture many complex interactions between neurons in the brain, and reveal nonobvious kinds of emergence.

more challengingly, in a virtual environment in a virtual reality, still stuck inside the computer. It will be fascinating to ponder this question on your journey through the simulated brain...

Why Should We Care about the Brain?

One of the things you'll discover on this journey is that *Computational Cognitive Neuroscience is hard*. There is a lot of material at multiple levels to master. We get into details of ion channels in neurons, names of pathways in different parts of the brain, effects of lesions to different brain areas, and patterns of neural activity, on top of all the details about behavioral paradigms and reaction time patterns. Wouldn't it just be a lot simpler if we could ignore all these brain details, and just focus on what we *really* care about – how does cognition itself work? By way of analogy, we don't need to know much of anything about how computer hardware works to program in Visual Basic or Python, for example. Vastly different kinds of hardware can all run the same programming languages and software. Can't we just focus on the *software* of the mind and ignore the *hardware*?

Exactly this argument has been promulgated in many different forms over the years, and indeed has a bit of a resurgence recently in the form of abstract Bayesian models of cognition. David Marr was perhaps the most influential in arguing that one can somewhat independently examine cognition at three different levels (Marr 1977):

- **Computational** – what computations are being performed? What information is being processed?
- **Algorithmic** – how are these computations being performed, in terms of a sequence of information processing steps?
- **Implementational** – how does the hardware actually implement these algorithms?

This way of dividing up the problem has been used to argue that one can safely ignore the implementation (i.e., the brain), and focus on the computational and algorithmic levels, because, like in a computer, the hardware really doesn't matter so much.

However, the key oversight of this approach is that the reason hardware doesn't matter in standard computers is that *they are all specifically designed to be functionally equivalent in the first place!* Sure, there are lots of different details, but they are all implementing a basic serial Von Neumann architecture. What if the brain has a vastly different architecture, which makes some algorithms and computations work extremely efficiently, while it cannot even support others? Then the implementational level would matter a great deal.

There is every reason to believe that this is the case. The brain is *not* at all like a general purpose computational device. Instead, it is really a custom piece of hardware that implements a very specific set of computations in massive parallelism across its 20 billion neurons. In this respect, it is much more like the specialized graphics processing units (GPUs) in modern computers, which are custom designed to efficiently carry out in massive parallelism the specific computations necessary to render complex 3D graphics. More generally, the field of computer science is discovering that parallel computation is exceptionally difficult to program, and one has to completely rethink the algorithms and computations to obtain efficient parallel computation. Thus, the hardware of the brain matters a huge amount, and provides many important clues as to what kind of algorithms and computations are being performed.

Historically, the "ignore the brain" approaches have taken an interesting trajectory. In the 1960's through the early 1990's, the dominant approach was to assume that the brain actually operates much like a standard computer, and researchers tended to use concepts like logic and symbolic propositions in their cognitive models. Since then, a more statistical metaphor has become popular, with the Bayesian probabilistic framework being widely used in particular. This is an advance in many respects, as it emphasizes the graded nature of information processing in the brain (e.g., integrating various graded probabilities to arrive at an overall estimate of the likelihood of some event), as contrasted with hard symbols and logic, which didn't seem to be a particularly good fit with the way that many (though not all!) aspects of cognition actually operate.

However, the actual mathematics of Bayesian probability computations are not a particularly good fit to how the brain operates at the neural level, and much of this research operates without much consideration for how the brain actually functions. Instead, a version of Marr's computational level has been adopted, by assuming that whatever the brain is doing, it must be at least close to optimal, and Bayesian models can often tell us how to optimally combine uncertain pieces of information. Regardless of the validity of this optimality assumption, it is definitely useful to know what the optimal computations are for given problems, so this approach certainly has a lot of value in general. However, optimality is typically conditional on a number of assumptions, and it is often difficult to decide among these different assumptions.

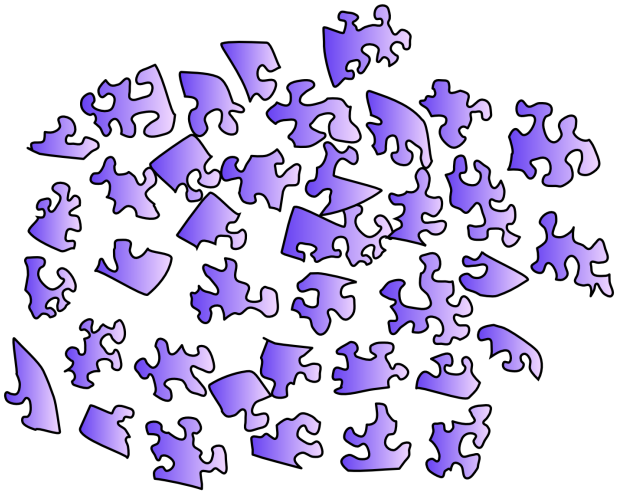


Figure 1.2: Models that are relatively unconstrained, e.g., by not addressing biological constraints, or detailed behavioral data, are like jigsaw puzzles of a featureless blue sky – very hard to solve – you just don't have enough clues to how everything fits together.

If you really want to know for sure how the brain is actually producing cognition, clearly you need to know how the brain actually functions. Yes, this is hard. But it is not impossible, and the state of neuroscience these days is such that there is a wealth of useful information to inform all manner of insights into how the brain actually works. It is like working on a jigsaw puzzle – the easiest puzzles are full of distinctive textures and junk everywhere, so you can really see when the pieces fit together (Figure 1.3). The rich tableau of neuroscience data provides all this distinctive junk to constrain the process of puzzling together cognition. In contrast, abstract, purely cognitive models are like a jigsaw puzzle with only a big

featureless blue sky (Figure 1.2). You only have the logical constraints of the piece shapes, which are all highly similar and difficult to discriminate. It takes forever.

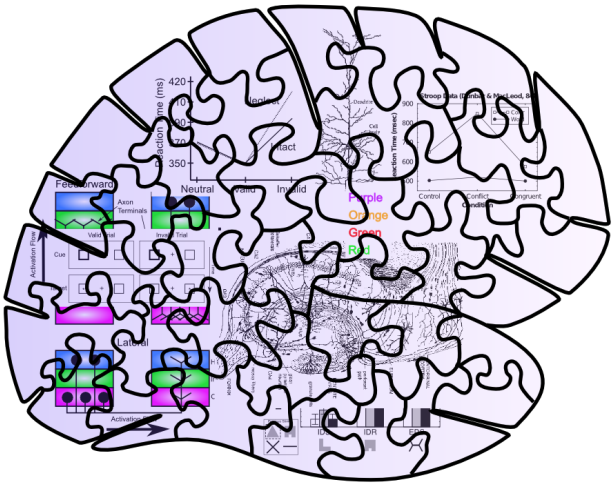


Figure 1.3: Adding constraints from biology and detailed consideration of behavior provide a rich set of clues for figuring out how to solve the puzzle of the brain!

A couple of the most satisfying instances of all the pieces coming together to complete a puzzle include:

- The detailed biology of the hippocampus, including high levels of inhibition and broad diffuse connectivity, fit together with its unique role in rapidly learning new episodic information, and the remarkable data from patient HM who had his hippocampus resected to prevent intractable epilepsy. Through computational models in Chapter 8 (Memory), we can see that these biological details produce high levels of *pattern separation* which keep memories highly distinct, and thus enable rapid learning without creating catastrophic levels of interference.
- The detailed biology of the connections between dopamine, basal ganglia, and prefrontal cortex fit together with the computational requirements for making decisions based on prior reward history, and learning what information is important to hold on to, versus what can be ignored. Computational models in Chapter 10 (Executive Function) show that the dopamine system can exhibit a kind of time travel needed to translate later utility into an earlier decision of what information to maintain, and those in Chapter 7 (Motor) show that the effects of dopamine on the basal ganglia circuitry are just right to facilitate decision making based on both positive and negative outcomes. And the interaction between the basal ganglia and the prefrontal cortex enables basal ganglia decisions to influence what is maintained and acted upon in the prefrontal cortex. There are a lot of pieces here, but the fact that they all fit together so well into a functional model – and that many aspects of them have withstood the test of direct experimentation – makes it that much more likely that this is really what is going on.

AI, ML, and Neuroscience

The core material in this textbook has been around for 20 years now, since 2000 (O'Reilly and Munakata 2000), and the overall popularity of neural network models has fluctuated considerably during that time. Currently, as of the 4th edition in 2020, there has been a major resurgence of interest in using "deep" neural networks for artificial intelligence (AI) and machine learning (ML) (LeCun, Bengio, and Hinton 2015; Schmidhuber 2015). These models are now powering practical, though still very narrow, applications on modern smart phones, outperforming other techniques in a wide range of ML competitions (Krizhevsky, Sutskever, and Hinton 2012), and beating humans at their most cherished games (Silver et al. 2017). Interestingly, the core principles powering these new AI models are consistent with many of those based on how the brain functions, including the error backpropagation learning algorithm which we discuss in detail in the *Learning* Chapter.

However, these new AI models also include many mechanisms that are *not* consistent with the underlying biology, and the performance-based goals for these models are often at odds with the more purely *scientific*

goals of understanding how the brain works. Thus, the material in this text is complementary to AI / ML approaches. Furthermore, it is still the case that the human brain is the undisputed champion at *general intelligence:* the ability to do many different tasks reasonably well, and learn with relatively minimal levels of explicit instruction to perform new tasks.

Interestingly, one core feature of the human brain, which is present in the models developed in this book, but largely absent in current AI models, is the pervasive *bidirectional connectivity* among neurons: neurons in the brain are *mutually* interacting, highly "social" little cells. Furthermore, this bidirectional connectivity or *recurrence* has been tied to *consciousness* (Lamme 2006), thus suggesting the interesting possibility that the human capacity for consciousness, based on this bidirectional connectivity, may be a key element of our general intelligence capacity. In particular, consciousness may enable flexible access to knowledge by many different routes, as knowledge can "float" across domains in the "plenary" *global workspace* of consciousness (Baars 1988), and we can directly, *consciousiously* manipulate and control our knowledge to direct it toward desired cognitive goals.

The models in this textbook provide a foundation for understanding this kind of flexible cognitive function, and current research is directly focused on exploring these ideas further, so that ultimately we may yet understand more of the deep secrets of the human mind!

How to Read this Book

This book is intended to accommodate many different levels of background and interests. The main chapters are relatively short, and provide a high-level introduction to the major themes. There will be an increasing number of detailed subsections added over time, to support more advanced treatment of specific issues. The ability to support these multiple levels of readers is a major advantage of the wiki format. We also encourage usage of this material as an adjunct for other courses on related topics. The simulation models can be used by themselves in many different courses.

Due to the complexity and interconnected nature of the material (mirroring the brain itself), it may be useful to revisit earlier chapters after having read later chapters. Also, we strongly recommend reading Chapter 5 (Brain Areas) chapter *now*, and then re-reading it in its regular sequence after having made it all the way through Part I. It provides a nice high-level summary of functional brain organization, that bridges the two parts of the book, and gives an overall roadmap of the content we'll be covering. Some of it won't make as much sense until after you've read Part I, but doing a quick first read now will provide a lot of useful perspective.

External Resources

- Gary Cottrell's solicited compilation of important computational modeling papers

Part I: Chapter 2: The Neuron

One major reason the brain can be so plastic and learn to do so many different things, is that it is made up of a highly-sculptable form of *silly putty*: billions of individual neurons that are densely interconnected with each other, and capable of shaping what they do by changing these patterns of interconnections. The brain is like a massive LEGO set, where each of the individual pieces is quite simple (like a single LEGO piece), and all the power comes from the nearly infinite ways that these simple pieces can be recombined to do different things.

So the good news for you the student is, the neuron is fundamentally *simple*. Lots of people will try to tell you otherwise, but as you'll see as you go through this book, *simple neurons can account for much of what we know about how the brain functions.* So, even though they have a lot of moving parts and you can spend an entire career learning about even just one tiny part of a neuron, we strongly believe that all this complexity is in the service of a very simple overall function.

What is that function? Fundamentally, it is about **detection**. Neurons receive thousands of different input signals from other neurons, looking for specific patterns that are "meaningful" to them. A very simple analogy is with a smoke detector, which samples the air and looks for telltale traces of smoke. When these exceed a specified threshold limit, the alarm goes off. Similarly, the neuron has a **threshold** and only sends an "alarm" signal to other neurons when it detects something significant enough to cross this threshold. The alarm is called an **action potential** or **spike** and it is the fundamental unit of communication between neurons.

Our goal in this chapter is to understand how the neuron receives input signals from other neurons, integrates them into an overall signal strength that is compared against the threshold, and communicates the result to other neurons. We will see how these processes can be characterized mathematically in computer simulations (summarized in Figure 2.1). In the rest of the book, we will see how this simple overall function of the neuron ultimately enables us to perceive the world, to think, to communicate, and to remember.

Math warning: This chapter and the Learning Chapter (Chapter 4) are the only two in the entire book with significant amounts of math, because these two chapters develop the core equations that power our neural simulations. We have separated the conceptual from the mathematical content, and those with an aversion to math can get by without understanding all the details. So, don't be put off or overwhelmed by the math here – just focus on the core conceptual ideas and get what you can out of the math (even if it is not much, you'll be OK)!

Basic Biology of a Neuron as Detector

Figure 2.2 shows the correspondence between neural biology and the detection functions they serve. **Synapses** are the connection points between **sending neurons** (the ones firing an alarm and sending a signal) and **receiving neurons** (the ones receiving that signal). Most synapses are on **dendrites,** which are the large branching trees (the word "dendrite" is derived from the Greek "dendros," meaning tree), which is where the neuron integrates all the input signals. Like tributaries flowing into a major river, all these signals flow into the main dendritic trunk and into the **cell body**, where the final integration of the signal takes place. The thresholding takes place at the very start of the output-end of the neuron, called the **axon** (this starting place is called the **axon hillock** – apparently it looks like a little hill or something). The axon also branches widely and is what forms the other side of the synapses onto other neuron's dendrites, completing the next chain of communication. And onward it goes.

Everything you need to know about the neuron biology to understand the basic detector functionality is that simple: It just receives inputs, integrates them, and decides whether the integrated input is sufficiently strong to trigger an output signal. However, there are some additional biological properties regarding the nature of these input signals, which we'll see have implications for neural function, including making the integration process better able to deal with large changes in overall input signal strength.

There are at least three major sources of input signals to the neuron:

- **Excitatory inputs** – these are the "normal", most prevalent type of input from other neurons (roughly 85% of all inputs), which have the effect of exciting the receiving neuron (making it more likely to get over threshold and fire an "alarm"). They are conveyed via a synaptic channel called **AMPA**, which is

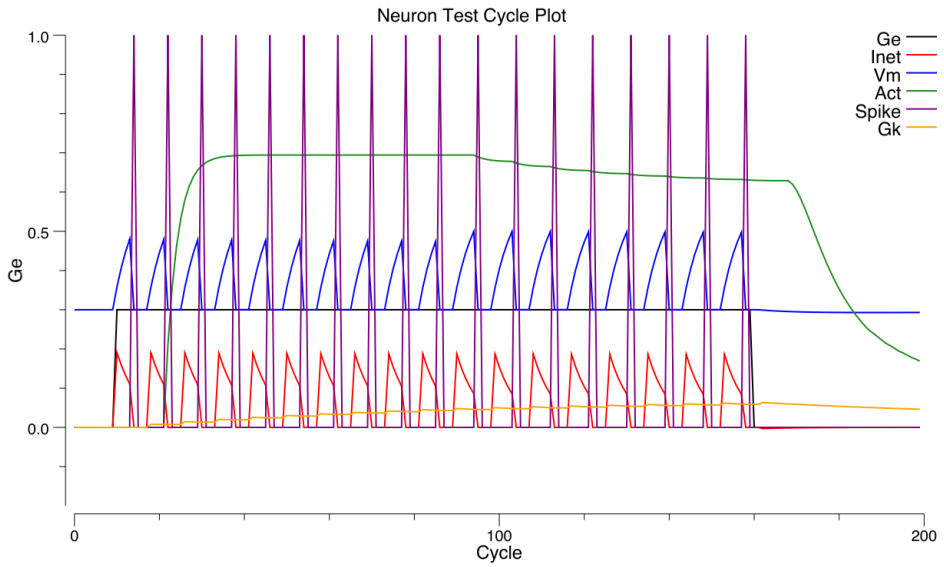


Figure 2.1: Trace of a simulated neuron spiking action potentials in response to an excitatory input – the blue **Vm** membrane potential (voltage of the neuron) increases (driven by the excitatory net input, **Ge**) until it reaches threshold (around .5), at which point a purple **Spike** (action potential) is triggered, which then resets the membrane potential back to its starting value (.3) and the process continues. The spike is communicated other neurons, and the overall rate of spiking (tracked by the green **Act** value) is proportional to the level of excitatory net input (relative to other opposing factors such as inhibition – the balance of all these factors is reflected in the net current **Inet**, in red). You can produce this graph and manipulate all the relevant parameters in the `neuron` exploration for this chapter.

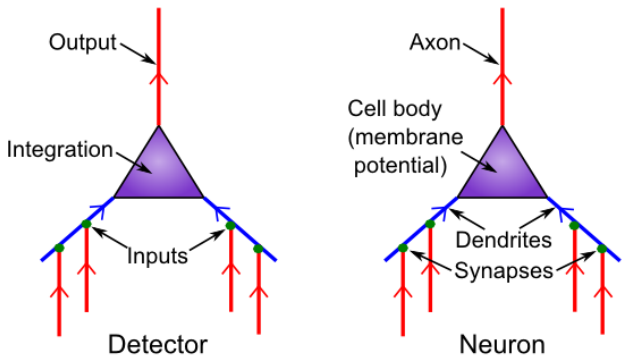


Figure 2.2: Neuron as a detector, with corresponding biological components.

opened by the neurotransmitter **glutamate**.

- **Inhibitory inputs** – these are the other 15% of inputs, which have the opposite effect to the excitatory inputs – they cause the neuron to be *less* likely to fire, and serve to make the integration process much more robust by keeping the excitation in check. There are specialized neurons in the brain called **inhibitory interneurons** that generate this inhibitory input (we'll learn a lot more about these in the *Networks* Chapter (3). This input comes in via **GABA** synaptic channels, driven by the neurotransmitter GABA.
- **Leak inputs** – these aren't technically inputs, as they are always present and active, but they serve a similar function to the inhibitory inputs, by counteracting the excitation and keeping the neuron in balance overall. Biologically, leak channels are **potassium channels (K)**.

The inhibitory and excitatory inputs come from *different* neurons in the cortex: a given neuron can only send either excitatory or inhibitory outputs to other neurons, not both (although neurons in other brain areas do violate this constraint, neocortical pyramidal neurons appear to obey it). We will see the multiple implications of this constraint throughout the text.

Finally, we introduce the notion of the **net synaptic efficacy** or **weight**, which represents the total impact that a sending neuron activity signal can have on the receiving neuron, via its synaptic connection. *The synaptic weight is one of the most important concepts in the entire field of computational cognitive neuroscience!* We will be exploring it in many different ways as we go along. Biologically, it represents the net ability of the sending neuron's action potential to release **neurotransmitter**, and the ability of that neurotransmitter to open synaptic channels on the postsynaptic side (including the total number of such channels that are available to be opened). For the excitatory inputs, the weight depends on the amount of glutamate released by the sending neuron into the synapse, and the number and efficacy of AMPA channels on the receiving neuron's side of the synapse (Figure 2.5).

Computationally, the weights determine what a neuron is detecting. A strong weight value indicates that the neuron is very sensitive to that particular input neuron, while a low weight means that that input is relatively unimportant. The entire process of learning (Chapter 4) amounts to changing these synaptic weights as a function of neural activity patterns in the sending and receiving neurons. In short, *everything you know, every cherished memory in your brain, is encoded as a pattern of synaptic weights!* In Chapter 3, we'll see that this detection process supports *categorization* and also, in linear algebra terms, the *projection* of high-dimensional input spaces along specific *dimensions* or *bases* or *axes* (all synonyms for our purposes). Thus, this detection process is truly the fundamental engine of neural computation, and can be described in many different ways that all amount to the same essential process we deconstruct here.

Biology Details

Figure 2.3 shows a tracing of a typical excitatory neuron in the cortex called a **pyramidal neuron**, which is the primary type that we simulate in our models. The major elements of dendrites, cell body, and axon as discussed in the main chapter are shown. Note that the dendrites have small **spines** on them – these are where the axons from sending neurons synapse, forming connections between neurons.

Figure 2.4 shows a high-resolution image of a synapse, while Figure 2.5 shows a schematic with all of the major elements of the synaptic signaling cascade represented. The primary behavior of a synapse is for an action potential to trigger release of neurotransmitter (NT) from the **presynaptic terminal button**, and this NT then binds to postsynaptic receptors that open to allow ions to flow, and thus communicating a signal to the postsynaptic neuron. In the predominant case of **excitatory AMPA-receptor** activation by the NT **glutamate**, the AMPA channels open to allow Sodium (Na+) ions to enter the postsynaptic neuron, which then have the effect of increasing the membrane potential and thus exciting the neuron. This excitatory input is called an **excitatory postsynaptic potential or EPSP**.

The other major types of postsynaptic receptors are: * **NMDA**, which is involved in learning and allows Calcium ions to flow (Ca++) – we will discuss these receptors in more detail in the [[CCN-Book/Learning|Learning chapter]]. * **mGluR**, which is also involved in learning and also possibly active maintenance of information in working memory – these receptors do not pass ions, and instead affect complex chemical processes in the postsynaptic cell.

Inhibitory synapses arising from inhibitory interneurons release GABA NT, and the corresponding GABA

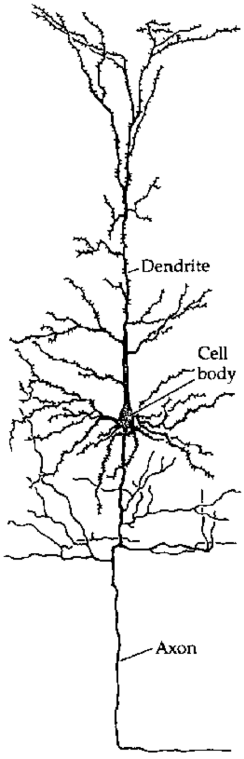


Figure 2.3: Tracing of a cortical pyramidal neuron.

Figure 2.4: Electron microscope image of a synapse. The arrows indicate synaptic release sites, where neurotransmitter is released to the receiving neuron. The small circles are synaptic vesicles, which contain the neurotransmitter.

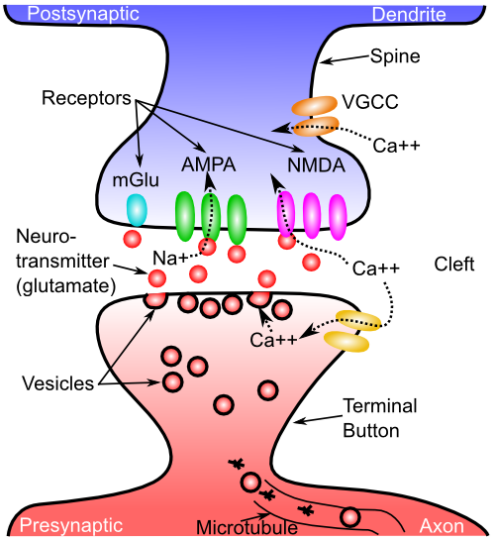


Figure 2.5: Schematic of a synapse, showing presynaptic terminal button which releases neurotransmitter (NT) into the synaptic cleft. The NT binds to postsynaptic receptors, causing ion channels to open (e.g., Sodium or Na+ ions for excitatory AMPA channels), and thus activating the receiving neuron. Metabotropic receptors such as mGluR do no open to allow ions to flow (as ionotropic ones do), and instead they trigger second-messenger cascades of reactions that can affect learning and other processes in the postsynaptic cell.

receptors on the receiving neurons open to allow Chloride (CL-) ions to flow, producing a net negative or inhibitory effect on the postsynaptic cell (called an **inhibitory postsynaptic potential or IPSP**).

Importantly, the biology shows that synapses in the cortex can either be excitatory or inhibitory, but not both. This has implications for our computational models as we explore in the *Networks* Chapter.

Dynamics of Integration: Excitation vs. Inhibition and Leak

The process of integrating the three different types of input signals (excitation, inhibition, leak) lies at the heart of neural computation. This section provides a conceptual, intuitive understanding of this process, and how it relates to the underlying electrical properties of neurons. Later, we'll see how to translate this process into mathematical equations that can actually be simulated on the computer.

The integration process can be understood in terms of a **tug-of-war** (Figure 2.6). This tug-of-war takes place in the space of **electrical potentials** that exist in the neuron relative to the surrounding extracellular medium in which neurons live (interestingly, this medium, and the insides of neurons and other cells as well, is basically salt water with sodium (Na+), chloride (Cl-) and other ions floating around – we carry our remote evolutionary environment around within us at all times). The core function of a neuron can be understood entirely in electrical terms: voltages (electrical potentials) and currents (flow of electrically charged ions in and out of the neuron through tiny pores called **ion channels**).

To see how this works, let's just consider excitation versus inhibition (inhibition and leak are effectively the same for our purposes at this time). The key point is that **the integration process reflects the relative strength of excitation versus inhibition** – if excitation is stronger than inhibition, then the neuron's electrical potential (voltage) increases, perhaps to the point of getting over threshold and firing an output action potential. If inhibition is stronger, then the neuron's electrical potential decreases, and thus moves further away from getting over the threshold for firing.

Before we consider specific cases, let's introduce some obscure terminology that neuroscientists use to label the various actors in our tug-of-war drama (going from left to right in the Figure):

- g_i – the **inhibitory conductance** (g is the symbol for a conductance, and i indicates inhibition) – this is the total strength of the inhibitory input (i.e., how strong the inhibitory guy is tugging), and plays a major role in determining how strong of an inhibitory current there is. This corresponds biologically to

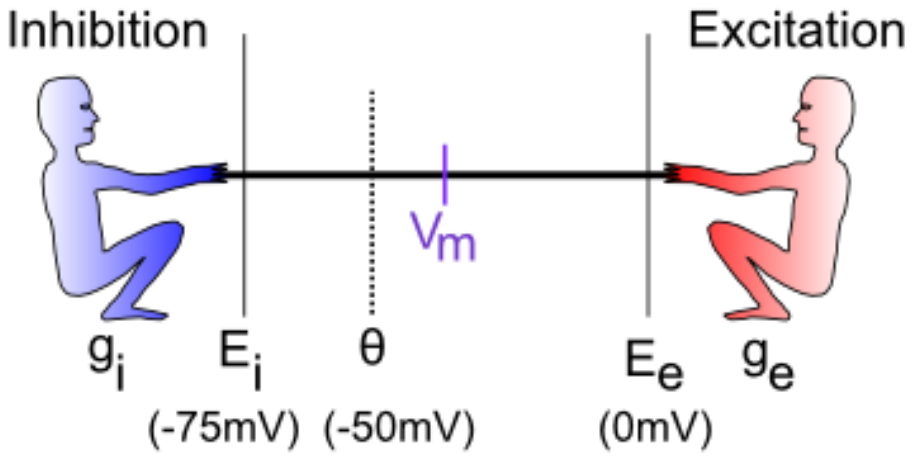


Figure 2.6: The neuron is a tug-of-war battleground between inhibition and excitation – the relative strength of each is what determines the membrane potential, Vm, which is what must get over threshold to fire an action potential output from the neuron.

the proportion of inhibitory ion channels that are currently open and allowing inhibitory ions to flow (these are **chloride** or **Cl-** ions in the case of GABA **inhibition**, and **potassium** or **K+** ions in the case of **leak** currents). For electricity buffs, the conductance is the inverse of resistance – most people find conductance more intuitive than resistance, so we'll stick with it.

- E_i – the **inhibitory driving potential** – in the tug-of-war metaphor, this just amounts to where the inhibitory guy happens to be standing relative to the electrical potential scale that operates within the neuron. Typically, this value is around -75mV where **mV** stands for **millivolts** – one thousandth (1/1,000) of a volt. These are very small electrical potentials for very small neurons.
- Θ – the **action potential threshold** – this is the electrical potential at which the neuron will fire an action potential output to signal other neurons. This is typically around -50mV. This is also called the **firing threshold** or the **spiking threshold**, because neurons are described as "firing a spike" when they get over this threshold.
- V_m – the **membrane potential** of the neuron (V = voltage or electrical potential, and m = membrane). This is the current electrical potential of the neuron relative to the extracellular space outside the neuron. It is called the membrane potential because it is the cell membrane (thin layer of fat basically) that separates the inside and outside of the neuron, and that is where the electrical potential really happens. An electrical potential or voltage is a relative comparison between the amount of electric charge in one location versus another. It is called a "potential" because when there is a difference, there is the potential to make stuff happen. For example, when there is a big potential difference between the charge in a cloud and that on the ground, it creates the potential for lightning. Just like water, differences in charge always flow "downhill" to try to balance things out. So if you have a lot of charge (water) in one location, it will flow until everything is all level. The cell membrane is effectively a dam against this flow, enabling the charge inside the cell to be different from that outside the cell. The ion channels in this context are like little tunnels in the dam wall that allow things to flow in a controlled manner. And when things flow, the membrane potential changes! In the tug-of-war metaphor, think of the membrane potential as the flag attached to the rope that marks where the balance of tugging is at the current moment.
- E_e – the **excitatory driving potential** – this is where the excitatory guy is standing in the electrical potential space (typically around 0 mV).
- g_e – the **excitatory conductance** – this is the total strength of the excitatory input, reflecting the proportion of excitatory ion channels that are open (these channels pass **sodium** or **Na+** ions – our deepest thoughts are all just salt water moving around).

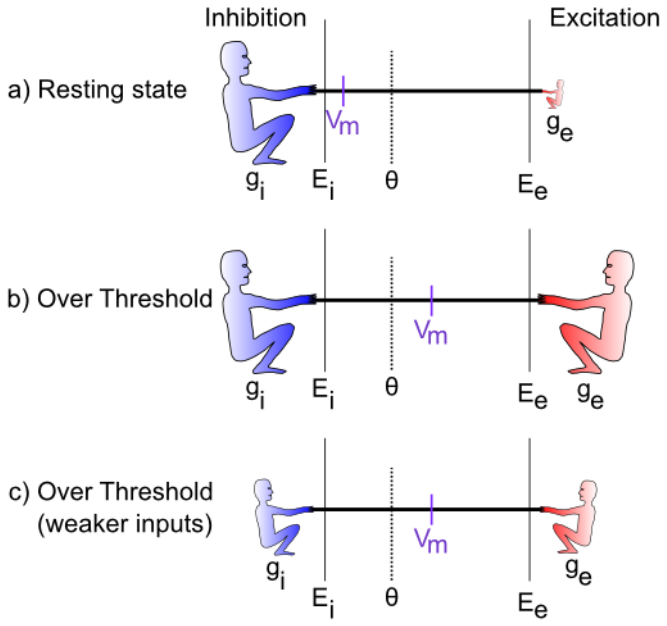


Figure 2.7: Specific cases in the tug-of-war scenario.

Figure 2.7 shows specific cases in the tug-of-war scenario. In the first case, the excitatory conductance g_e is very low (indicated by the small size of the excitatory guy), which represents a neuron at rest, not receiving many excitatory input signals from other neurons. In this case, the inhibition/leak pulls much more strongly, and keeps the membrane potential (Vm) down near the -70mV territory, which is also called the **resting potential** of the neuron. As such, it is below the action potential threshold Θ, and so the neuron does not output any signals itself. Everyone is just chillin'.

In the next case (b), the excitation is as strong as the inhibition, and this means that it can pull the membrane potential up to about the middle of the range. Because the firing threshold is toward the lower-end of the range, this is enough to get over threshold and fire a spike! The neuron will now communicate its signal to other neurons, and contribute to the overall flow of information in the brain's network.

The last case (c) is particularly interesting, because it illustrates that the integration process is fundamentally **relative** – what matters is how strong excitation is *relative* to the inhibition. If both are overall weaker, then neurons can still get over firing threshold. Can you think of any real-world example where this might be important? Consider the neurons in your visual system, which can experience huge variation in the overall amount of light coming into them depending on what you're looking at (e.g., compare snowboarding on a bright sunny day versus walking through thick woods after sunset). It turns out that the total amount of light coming into the visual system drives both a "background" level of inhibition, in addition to the amount of excitation that visual neurons experience. Thus, when it's bright, neurons get greater amounts of both excitation and inhibition compared to when it is dark. *This enables the neurons to remain in their sensitive range for detecting things* despite large differences in overall input levels.

Computing Activation Output

The membrane potential Vm is not communicated directly to other neurons – instead it is subjected to a **threshold** and only the strongest levels of excitation are then communicated, resulting in a much more efficient and compact encoding of information in the brain. In human terms, neurons are sensitive to "TMI" (too much information) constraints, also known as "Gricean Maxims" wikipedia link – e.g., only communicate relevant, important information.

Actual neurons in the Neocortex compute discrete **spikes** or **action potentials**, which are very brief (< 1 ms) and trigger the release of neurotransmitter that then drives the excitation or inhibition of the

neurons they are sending to. After the spike, the membrane potential Vm is reset back to a low value (at or even below the resting potential), and it must then climb back up again to the level of the threshold before another spike can occur. This process results in different *rates of spiking* associated with different levels of excitation – it is clear from eletrophysiological recordings of neurons all over the neocortex that this **spike rate** information is highly informative about behaviorally and cognitively relevant information. There remains considerable debate about the degree to which more precise differences in spike timing contain additional useful information.

In our computer models, we can simulate discrete spiking behavior directly in a very straightforward way (see below for details). However, we often use a **rate code** approximation instead, where the activation output of the neuron is a *real valued number* between 0-1 that corresponds to the overall rate of neural spiking. We typically think of this rate code as reflecting the net output of a small population of roughly 100 neurons that all respond to similar information – the neocortex is organized anatomically with **microcolumns** of roughly this number of neurons, where all of the neurons do indeed code for similar information. Use of this rate code activation enables smaller-scale models that converge on a stable interpretation of the input patterns rapidly, with an overall savings in computational time and model complexity. Nevertheless, there are tradeoffs in using these approximations, which we will discuss more in the Networks and other chapters. Getting the rate code to produce a good approximation to discrete spiking behavior has been somewhat challenging in the Leabra framework, and only recently has a truly satisfactory solution been developed, which is now the standard in the emergent software.

Mathematical Formulations

Now you've got an intuitive understanding of how the neuron integrates excitation and inhibition. We can capture this dynamic in a set of mathematical equations that can be used to simulate neurons on the computer. The first set of equations focuses on the effects of inputs to a neuron. The second set focuses on generating outputs from the neuron. We will cover a fair amount of mathematical ground here. Don't worry if you don't follow all of the details. As long as you follow conceptually what the equations are doing, you should be able to build on this understanding when you get your hands on the actual equations themselves and explore how they behave with different inputs and parameters. You will see that despite all the math, the neuron's behavior is indeed simple: the amount of excitatory input determines how excited it gets, in balance with the amount of inhibition and leak. And the resulting output signals behave pretty much as you would expect.

Computing Inputs

We begin by formalizing the "strength" by which each side of the tug-of-war pulls, and then show how that causes the Vm "flag" to move as a result. This provides explicit equations for the tug-of-war dynamic integration process. Then, we show how to actually compute the conductance factors in this tug-of-war equation as a function of the inputs coming into the neuron, and the synaptic weights (focusing on the excitatory inputs for now). Finally, we provide a summary equation for the tug-of-war which can tell you where the flag will end up in the end, to complement the dynamical equations which show you how it moves over time.

Neural Integration

The key idea behind these equations is that each guy in the tug-of-war pulls with a strength that is proportional to both its overall strength (conductance), and how far the "flag" (Vm) is away from its position (indicated by the driving potential E). Imagine that the tuggers are planted in their position, and their arms are fully contracted when the Vm flag gets to their position (E), and they can't re-grip the rope, such that they can't pull any more at this point. To put this idea into an equation, we can write the "force" or **current** that the excitatory guy exerts as:

$$I_e = g_e \left(E_e - V_m\right)$$

The excitatory current is I_e (I is the traditional term for an electrical current, and e again for excitation), and it is the product of the conductance g_e times *how far the membrane potential is away from the excitatory*

driving potential. If $V_m = E_e$ then the excitatory guy has "won" the tug of war, and it no longer pulls anymore, and the current goes to zero (regardless of how big the conductance might be – anything times 0 is 0). Interestingly, this also means that the excitatory guy pulls the strongest when the Vm "flag" is furthest away from it – i.e., when the neuron is at its resting potential. Thus, it is easiest to excite a neuron when it's well rested.

The same basic equation can be written for the inhibition guy, and also separately for the leak guy (which we can now reintroduce as a basic clone of the inhibition term):

$$I_i = g_i \left(E_i - V_m\right)$$

leak current:

$$I_l = g_l \left(E_l - V_m\right)$$

(only the subscripts are different).

Next, we can add together these three different currents to get the **net current**, which represents the net flow of charged ions across the neuron's membrane (through the ion channels):

$$\begin{aligned} I_{net} =& I_e + I_i + I_l = \quad g_e \left(E_e - V_m\right) + g_i \left(E_i - V_m\right) \\ &+ g_l \left(E_l - V_m\right) \end{aligned}$$

So what good is a net current? Recall that electricity is like water, and it flows to even itself out. When water flows from a place where there is a lot of water to a place where there is less, the result is that there is less water in the first place and more in the second. The same thing happens with our currents: the flow of current changes the membrane potential (height of the water) inside the neuron:

$$V_m\left(t\right) = V_m\left(t-1\right) + dt_{vm} I_{net}$$

$V_m(t)$ is the current value of Vm, which is updated from value on the previous time step $V_m(t-1)$, and the dt_{vm} is a **rate constant** that determines how fast the membrane potential changes – it mainly reflects the capacitance of the neuron's membrane).

The above two equations are the essence of what we need to be able to simulate a neuron on a computer! It tells us how the membrane potential changes as a function of the inhibitory, leak and excitatory inputs – given specific numbers for these input conductances, and a starting Vm value, we can then **iteratively** compute the new Vm value according to the above equations, and this will accurately reflect how a real neuron would respond to similar such inputs!

To summarize, here's a single version of the above equations that does everything:

$$\begin{aligned} V_m(t) =& V_m(t-1) + dt_{vm} \\ & \left[g_e(E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m)\right] \end{aligned}$$

For those of you who noticed the issue with the minus sign above, or are curious how all of this relates to **Ohm's law** and the process of diffusion, please see the Chapter Appendix section *Electrophysiology of the Neuron*. If you're happy enough with where we've come, feel free to move along to finding out how we compute these input conductances, and what we then do with the Vm value to drive the output signal of the neuron.

Computing Input Conductances

The excitatory and inhibitory input conductances represent the total number of ion channels of each type that are currently open and thus allowing ions to flow. In real neurons, these conductances are typically measured in nanosiemens (nS), which is 10^{-9} siemens (a very small number – neurons are very tiny). Typically, neuroscientists divide these conductances into two components:

- $\bar{g}$ ("g-bar") – a constant value that determines the **maximum conductance** that would occur if every ion channel were to be open, and:

- $g\,(t)$ – a dynamically changing variable that indicates at the present moment, what fraction of the total number of ion channels are currently open (goes between 0 and 1).

Thus, the total conductances of interest are written as: **excitatory conductance:**

$$\bar{g}_e g_e(t)$$

and the **inhibitory conductance:**

$$\bar{g}_i g_i(t)$$

and the **leak conductance:**

$$\bar{g}_l$$

(note that because leak is a constant, it does not have a dynamically changing value, only the constant g-bar value).

This separation of terms makes it easier to compute the conductance, because all we need to focus on is computing the proportion or fraction of open ion channels of each type. This can be done by computing the average number of ion channels open at each synaptic input to the neuron:

$$g_e(t) = \frac{1}{n} \sum_i x_i w_i$$

where x_i is the **activity** of a particular sending neuron indexed by the subscript i, w_i is the **synaptic weight strength** that connects sending neuron i to the receiving neuron, and n is the total number of channels of that type (in this case, excitatory) across all synaptic inputs to the cell. As noted above, the synaptic weight determines what patterns the receiving neuron is sensitive to, and is what adapts with learning – this equation shows how it enters mathematically into computing the total amount of excitatory conductance.

The above equation suggests that the neuron performs a very simple function to determine how much input it is getting: it just adds it all up from all of its different sources (and takes the average to compute a proportion instead of a sum – so that this proportion is then multiplied by $\bar{g}_e$ to get an actual conductance value). Each input source contributes in proportion to how active the sender is, multiplied by how much the receiving neuron cares about that information – the synaptic weight value. We also refer to this average total input as the **net input**.

The same equation holds for inhibitory input conductances, which are computed in terms of the activations of inhibitory sending neurons, times the inhibitory weight values.

There are some further complexities about how we integrate inputs from different categories of input sources (i.e., projections from different source brain areas into a given receiving neuron), which we deal with in the Chapter Appendix subsection *Net Input Detail*. But overall, this aspect of the computation is relatively simple and we can now move on to the next step, of comparing the membrane potential to the threshold and generating some output.

Equilibrium Membrane Potential

Before finishing up the final step in the detection process (generating an output), we will need to use the concept of the **equilibrium membrane potential**, which is the value of Vm that the neuron will settle into and stay at, *given a fixed set of excitatory and inhibitory input conductances* (if these aren't steady, then the the Vm will likely be constantly changing as they change). This equilibrium value is interesting because it tells us more clearly how the tug-of-war process inside the neuron actually balances out in the end. Also, we will see in the next section that it is useful mathematically.

To compute the equilibrium membrane potential (V_m^{eq}), we can use an important mathematical technique: set the change in membrane potential (according to the iterative Vm updating equation from above) to 0, and then solve the equation for the value of Vm under this condition. In other words, if we want to find out what the equilibrium state is, we simply compute what the numbers need to be such that Vm is no longer changing (i.e., its rate of change is 0). Here are the mathematical steps that do this:

$$\begin{aligned} V_m(t) =& V_m(t-1) + dt_{vm} \\ & [g_e(E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m)] \end{aligned}$$

(the **iterative Vm update equation:**)

This is the part that is driving the changes (time constant omitted as we are looking for equilibrium):

$$\Delta V_m = g_e (E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m)$$

which we set to zero to find when it stops changing:

$$0 = g_e (E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m)$$

and then do some algebra to solve for Vm:

$$V_m = \frac{g_e}{g_e + g_i + g_l} E_e + \frac{g_i}{g_e + g_i + g_l} E_i + \frac{g_l}{g_e + g_i + g_l} E_l$$

The detailed math is shown in the Chapter Appendix section *Math Derivations*.

In words, this says that the excitatory drive E_e contributes to the overall Vm as a function of the proportion of the excitatory conductance g_e relative to the sum of all the conductances ($g_e + g_i + g_l$). And the same for each of the others (inhibition, leak). This is just what we expect from the tug-of-war picture: if we ignore g_l, then the Vm "flag" is positioned as a function of the relative balance between g_e and g_i – if they are equal, then $g_e/(g_e + g_i)$ is .5 (e.g., just put a "1" in for each of the g's – 1/2 = .5), which means that the Vm flag is half-way between E_i and E_e. So, all this math just to rediscover what we knew already intuitively! (Actually, that is the best way to do math – if you draw the right picture, it should tell you the answers before you do all the algebra). But we'll see that this math will come in handy next.

Here is a version with the conductance terms explicitly broken out into the "g-bar" constants and the time-varying "g(t)" parts:

$$V_m = \frac{\bar{g}_e g_e(t)}{\bar{g}_e g_e(t) + \bar{g}_i g_i(t) + \bar{g}_l} E_e + \frac{\bar{g}_i g_i(t)}{\bar{g}_e g_e(t) + \bar{g}_i g_i(t) + \bar{g}_l} E_i + \frac{\bar{g}_l}{\bar{g}_e g_e(t) + \bar{g}_i g_i(t) + \bar{g}_l} E_l$$

For those who really like math, the equilibrium membrane potential equation is shown to be a Bayesian Optimal Detector in the Appendix.

Generating Outputs

The output of the neuron can be simulated at two different levels: discrete spiking (which is how neurons actually behave biologically), or using a rate code approximation. We cover each in turn, and show how the rate code must be derived to match the behavior of the discrete spiking neuron, when averaged over time (it is important that our approximations are valid in the sense that they match the more detailed biological behavior where possible, even as they provide some simplification).

Discrete Spiking

To compute discrete action potential spiking behavior from the neural equations we have so far, we need to determine when the membrane potential gets above the firing threshold, and then emit a spike, and subsequently reset the membrane potential back down to a value, from which it can then climb back up and trigger another spike again, etc. This is actually best expressed as a kind of simple computer program:

```
if (Vm > Theta) then: y = 1; Vm = Vm_r; else y = 0
```

where y is the activation output value of the neuron, and Vm_r is the *reset potential* that the membrane potential is reset to after a spike is triggered. Biologically, there are special potassium (K+) channels that bring the membrane potential back down after a spike.

This simplest of spiking models is not *quite* sufficient to account for the detailed spiking behavior of actual cortical neurons. However, a slightly more complex model can account for actual spiking data with great accuracy (as shown by Gerstner and colleagues (Brette and Gerstner 2005), and winning several international competitions even!). This model is known as the *Adaptive Exponential* or AdEx model (Scholarpedia Article on AdEx. We typically use this AdEx model when simulating discrete spiking, although the simpler model described above is also still an option. The critical feature of the AdEx model is that the effective firing threshold adapts over time, as a function of the excitation coming into the cell, and its recent firing history. The net result is a phenomenon called **spike rate adaptation**, where the rate of spiking tends to decrease over time for otherwise static input levels. Otherwise, however, the AdEx model is identical to the one described above.

Rate Code Approximation to Spiking

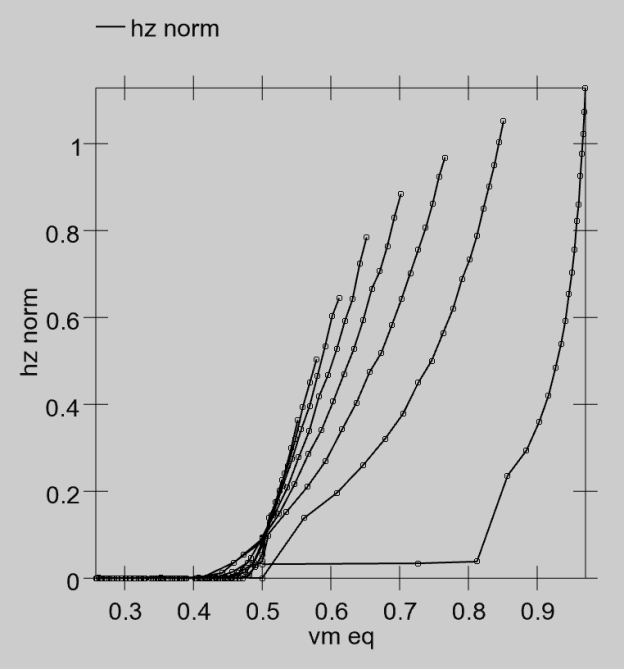


Figure 2.8: Normalized actual firing rate from the spiking model (**hz norm**, where hz stands for Hertz which are the units of firing rate) as a function of the equilibrium membrane potential (**vm_eq**) for a range of different excitatory and inhibitory input conductances. For every value of vm_eq, there are multiple different hz_norm values, indicating that you cannot base a rate code approximation on the Vm value directly.

Even though actual neurons communicate via discrete spiking (action potential) events, it is often useful in our computational models to adopt a somewhat more abstract **rate code approximation**, where the neuron continuously outputs a single graded value (typically normalized between 0-1) that reflects the overall rate of spiking that the neuron should be exhibiting given the level of inputs it is receiving. In other words, we could count up the number of discrete spikes the neuron fires, and divide that by the amount of time we did the counting over, and this would give us an average spiking rate. Instead of having the neuron communicate this rate of spiking distributed in discrete spikes over that period of time, we can have it communicate that rate value instantly, as a graded number. Computationally, this should be more efficient, because it is compressing the amount of time required to communicate a particular spiking rate, and it also tends to reduce the overall level of noise in the network, because instead of switching between spiking and not-spiking, the neuron can continuously output a more steady rate code value.

As noted earlier, the rate code value can be thought of in biological terms as the output of a small population (e.g., 100) of neurons that are generally receiving the same inputs, and giving similar output responses – averaging the number of spikes at any given point in time over this population of neurons is

roughly equivalent to averaging over time from a single spiking neuron. As such, we can consider our simulated rate code computational neurons to correspond to a small population of actual discrete spiking neurons.

To actually compute the rate code output, we need an equation that provides a real-valued number that matches the number of spikes emitted by a spiking neuron with the same level of inputs. Interestingly, you cannot use the membrane potential Vm as the input to this equation – it does *not* have a one-to-one relationship with spiking rate! That is, when we run our spiking model and measure the actual rate of spiking for different combinations of excitatory and inhibitory input, and then plot that against the equilibrium Vm value that those input values produce (without any spiking taking place), there are multiple spiking rate values for each Vm value – you cannot predict the correct firing rate value knowing only the Vm (Figure 2.8).

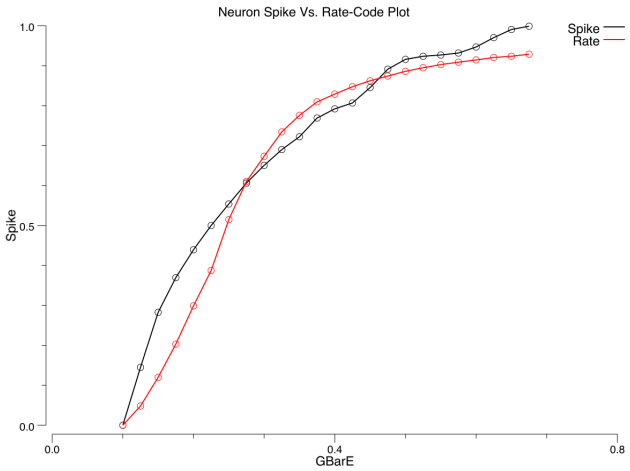


Figure 2.9: Quality of the rate code approximation (rate line) to actual spiking rate (**Spike** line), over a range of excitatory input levels (**GBarE**). The rate code approximation is based on the "gelin" (linear in Ge) model comparing Ge to g_e^Θ, using the Noisy XX1 sigmoidal function, and also including spike rate adaptation as included in the AdEx model.

Instead, it turns out that the excitatory net input g_e enables a good prediction of actual spiking rate, when it is compared to an appropriate threshold value (Figure 2.9). For the membrane potential, we know that Vm is compared to the threshold Θ to determine when output occurs. What is the appropriate threshold to use for the excitatory net input? We need to somehow convert Θ into a g_e^Θ value – a threshold in excitatory input terms. Here, we can leverage the equilibrium membrane potential equation, derived above. We can use this equation to solve for the level of excitatory input conductance that would put the equilibrium membrane potential right at the firing threshold Θ:

$$\Theta = \frac{g_e^\Theta E_e + g_i E_i + g_l E_l}{g_e^\Theta + g_i + g_l}$$

solved for g_e^Θ:

$$g_e^\Theta = \frac{g_i(E_i - \Theta) + g_l(E_l - \Theta)}{\Theta - E_e}$$

(see the Chapter Appendix on *Math Derivations* for the algebra to derive this solution).

Now, we can say that our rate coded output activation value will be some function of the difference between the excitatory net input g_e and this threshold value:

$$y = f(g_e - g_e^\Theta)$$

and all we need to do is figure out what this function f() should look like.

There are three important properties that this function should have:

- **threshold** – it should be 0 (or close to it) when g_e is less than its threshold value (neurons should not respond when below threshold).
- **saturation** – when g_e gets very strong relative to the threshold, the neuron cannot actually keep firing at increasingly high rates – there is an upper limit to how fast it can spike (typically around 100-200 Hz or spikes per second). Thus, our rate code function also needs to exhibit this leveling-off or saturation at the high end.
- **smoothness** – there shouldn't be any abrupt transitions (sharp edges) to the function, so that the neuron's behavior is smooth and continuous.

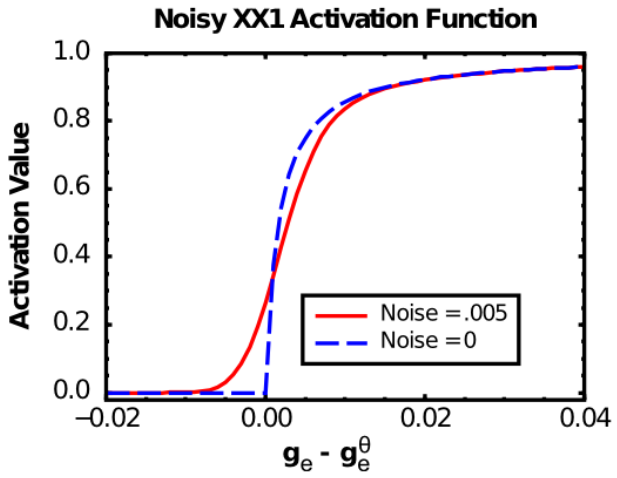


Figure 2.10: The X-over-X-plus 1 (XX1) function (Noise = 0) and the Noisy-XX1 (NXX1) function (Noise = .005).

The **X-over-X-plus-1 (XX1)** function (Figure 2.10, Noise=0 case), also known as the Michaelis-Mentin kinetics function – wikipedia link) exhibits the first two of these properties:

$$f_{xx1}(x) = \frac{x}{x+1}$$

where x is the *positive* portion of $g_e - g_e^{\Theta}$, with an extra **gain** factor γ, which just multiplies everything:

$$x = \gamma[g_e - g_e^{\Theta}]_+$$

So the full equation is:

$$y = \frac{\gamma[g_e - g_e^{\Theta}]_+}{\gamma[g_e - g_e^{\Theta}]_+ + 1}$$

Which can also be written as:

$$y = \frac{1}{\left(1 + \frac{1}{\gamma[g_e - g_e^{\Theta}]_+}\right)}$$

As you can see in Figure 2.10 (Noise=0), the basic XX1 function is not smooth at the point of the threshold. To remedy this problem, we **convolve** the XX1 function with normally-distributed (gaussian) noise, which smooths it out as shown in the Noise=0.005 case in Figure 2.10. Convolving amounts to adding to each point in the function some contribution from its nearby neighbors, weighted by the gaussian (bell-shaped) curve. It is what photo editing programs do when they do "smoothing" or "blurring" on an image. In the software, we use piecewise approximation function that is very quick to compute and closely approximates the noise-convolved version.

Biologically, this convolution process reflects the fact that neurons experience a large amount of noise (random fluctuations in the inputs and membrane potential), so that even if they are slightly below the firing threshold, a random fluctuation can sometimes push it over threshold and generate a spike. Thus, the spiking rate around the threshold is smooth, not sharp as in the plain XX1 function.

For completeness sake, and strictly for the mathematically inclined, here is the equation for the convolution operation:

$$y^*(x) = \int_{z=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-z^2/(2\sigma^2)} y(x-z) dz$$

where $y(x-z)$ is the XX1 function applied to the $x-z$ input instead of just x. In practice, a finite kernel of width 3σ on either side of x is used in the numerical convolution.

After convolution, the XX1 function (Figure 2.10) approximates the average firing rate of many neuronal models with discrete spiking, including AdEx (Figure 2.9). A mathematical explanation is in the Chapter Appendix section *Frequency-Current Curve*.

Restoring Iterative Dynamics in the Activation

There is just one last problem with the equations as written above. They don't evolve over time in a graded fashion. In contrast, the Vm value does evolve in a graded fashion by virtue of being iteratively computed, where it incrementally approaches the equilibrium value over a number of time steps of updating. Instead the activation produced by the above equations goes directly to its equilibrium value very quickly, because it is calculated based on excitatory conductance and does not take into account the sluggishness with which changes in conductance lead to changes in membrane potentials (due to capacitance). As discussed in the *Introduction* Chapter, graded processing is very important, and we can see this very directly in this case, because the above equations do not work very well in many cases because they lack this gradual evolution over time.

To introduce graded iterative dynamics into the activation function, we just use the activation value ($y^*(x)$) from the above equation as a *driving force* to an iterative temporally-extended update equation:

$$y(t) = y(t-1) + dt_{vm} \left(y^*(x) - y(t-1)\right)$$

This causes the actual final rate code activation output at the current time t, y(t) to iteratively approach the driving value given by $y^*(x)$, with the same time constant dt_{vm} that is used in updating the membrane potential. In practice this works extremely well, better than any prior activation function used with Leabra.

Summary of Neuron Equations and Normalized Parameters

Table 2.1: The parameters used in our simulations are normalized using the above conversion factors so that the typical values that arise in a simulation fall within the 0..1 normalized range. For example, the membrane potential is represented in the range between 0 and 2 where 0 corresponds to -100mV and 2 corresponds to +100mV and 1 is thus 0mV (and most membrane potential values stay within 0-1 in this scale). The biological values given are the default values for the AdEx model.

Parameter	Bio Val	Norm Val	Parameter	Bio Val	Norm Val
Time	0.001 sec	1 ms	Voltage	0.1 V or 100mV	0..2
Current	$1x10^{-8}$ A	10 nA	Conductance	$1x10^{-9}$ S	1 nS
Capacitance	$1x10^{-12}$ F	1 pF	C (memb cap)	281 pF	Dt = .355
GbarL (leak)	10 nS	0.1	GBarI (inhib)	100 nS	1
GbarE (excite)	100 nS	1	ErevL (leak)	-70mV	0.3
ErevI (inhib)	-75mV	0.25	ErevE (excite)	0mV	1
θ (Thr)	-50mV	0.5	SpikeThr	20mV	1.2

Table 2.1 shows the normalized values of the parameters used in our simulations. We use these normalized values instead of the normal biological parameters so that everything fits naturally within a 0..1 range, thereby simplifying many practical aspects of working with the simulations.

The final equations used to update the neuron, in computational order, are shown here, with all variables that change over time indicated as a function of (t):

1. Compute the excitatory input conductance (inhibition would be similar, but we'll discuss this more in the next chapter, so we'll omit it here):

$$g_e(t) = \frac{1}{n}\sum_i x_i(t) w_i$$

2. Update the membrane potential one time step at a time, as a function of input conductances (separating conductances into dynamic and constant "g-bar" parts):

$$\begin{aligned} V_m(t) =& V_m(t-1) + dt_{vm} \\ & [\bar{g}_e g_e(t)(E_e - V_m) + \bar{g}_i g_i(t)(E_i - V_m) + g_l(E_l - V_m)] \end{aligned}$$

3a. For discrete spiking, compare membrane potential to threshold and trigger a spike and reset Vm if above threshold:

```
if (Vm(t) > Theta) then: y(t) = 1; Vm(t) = Vm_r; else y(t) = 0`
```

3b. For rate code approximation, compute output activation as NXX1 function of g_e and Vm:

$$y^*(x) = f_{nxx1}(g_e^*(t)) \approx \frac{1}{\left(1 + \frac{1}{\gamma[g_e - g_e^\Theta]_+}\right)}$$

(convolution with noise not shown)

$$y(t) = y(t-1) + dt_{vm}\left(y^*(x) - y(t-1)\right)$$

(restoring iterative dynamics based on time constant of membrane potential changes)

Exploration of the Individual Neuron

To get your hands dirty, run the `neuron` simulation, available in the CCN Sims.

Back to the Detector

Now that you can see how the individual neuron integrates a given excitatory signal relative to leak/inhibition, it is important to put this into the larger perspective of the detection process. In this simulation, you'll see how a neuron can pick out a particular input pattern from a set of inputs, and also how it can have different patterns of responding depending on its parameters ("loose" or "strict").

You can run this simulation in the `detect` model in CCN Sims.

Appendix

There are a number of optional in-depth topics here in this Chapter Appendix.

- **Neuron Electrophysiology:** more detailed description of the electrophysiology of the neuron, and how the underlying concentration gradients of ions give rise to the electrical integration properties of the neuron.
- **Net Input Detail:** details on how net inputs are computed across multiple different input projections.
- **Math Derivations:** shows how to derive the equilibrium Vm value based on the update equation, and g_e^Θ.
- **Frequency-Current Curve:** Derives an explanation for why the XX1 function approximates discrete spiking (courtesy of Sergio Verduzco-Flores).

- **Sodium-Gated Potassium Channels for Adaptation (kNa Adapt):** describes how sodium that comes in during spiking
- **Bayesian Optimal Detector:** how the equilibrium membrane potential represents a Bayesian optimal way of integrating the different inputs to the neuron.

Neuron Electrophysiology

This optional section provides a full treatment of the electrophysiology of the neuron – how differential concentrations of individual ions lead to the electrical dynamics of the neuron.

First, some basic facts of electricity. Electrons and protons, which together make up atoms (along with neutrons), have electrical charge (the electron is negative, and the proton is positive). An **ion** is an atom where these positive and negative charges are out of balance, so that it carries a **net charge**. Because the brain carries its own salt-water ocean around with it, the primary ions of interest are:

- **sodium (Na+)** which has a net positive charge.
- **chloride (Cl-)** which has a net negative charge.
- **potassium (K+)** which has a net positive charge.
- **calcium (Ca++)** which has ''two" net positive charges.

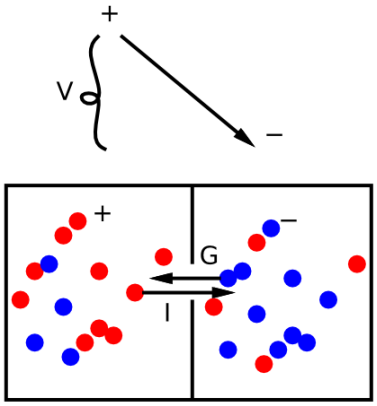


Figure 2.11: Basic principles of electricity: when there is an imbalance of positive and negative charged ions, these ions will flow so as to cancel out this imbalance. The flow of ions is called a current **I**, driven by the potential (level of imbalance) **V** with the conductance **G** (e.g., size of the opening between the two chambers) determining how quickly the ions can flow.

As we noted in the main chapter, these ions tend to flow under the influence of an electrical potential (voltage), driven by the basic principle that **opposite charges attract and like charges repel**. If there is an area with more positive charge than negative charge (i.e., and **electrical potential**), then any negative charges nearby will be drawn into this area (creating an electrical **current**), thus nullifying that imbalance, and restoring everything to a neutral potential. Figure 2.11 shows a simple diagram of this dynamic. The **conductance** is effectively how wide the opening or path is between the imbalanced charges, which determines how quickly the current can flow.

Ohm's law formalizes the situation mathematically:

$$I = GV$$

(i.e., current = conductance times potential).

The other major force at work in the neuron is **diffusion**, which causes individual ions to move around until they are uniformly distributed across space (Figure 2.12). Interestingly, the diffusion force originates from random movements of the ions driven by heat – ions are constantly bumping around through space, with a mean velocity proportional to the temperature of the environment they're in. This constant motion creates the diffusion force as a result of the inevitable increase in **entropy** of a system – the maximum entropy state

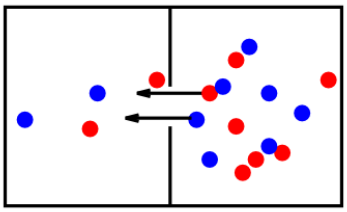


Figure 2.12: Diffusion is the other major force at work in neurons – it causes each ion individually to balance out its concentration uniformly across space (i.e., on both sides of the chamber). Concentration imbalances can then cause ions to flow, creating a current, just like electrical potential forces.

is where each ion is uniformly distributed, and this is in effect what the diffusion force represents. The key difference between the diffusion and electrical force is:

- Diffusion operates individually on each ion, regardless of its charge compared to other ions etc – each ion is driven by the diffusion force to spread itself uniformly around. In contrast, electrical forces ignore the identity of the ion, and only care about the net electrical charge. From electricity's perspective, Na+ and K+ are effectively equivalent.

It is this critical difference between diffusion and electrical forces that causes different ions to have different driving potentials, and thus exert different influences over the neuron.

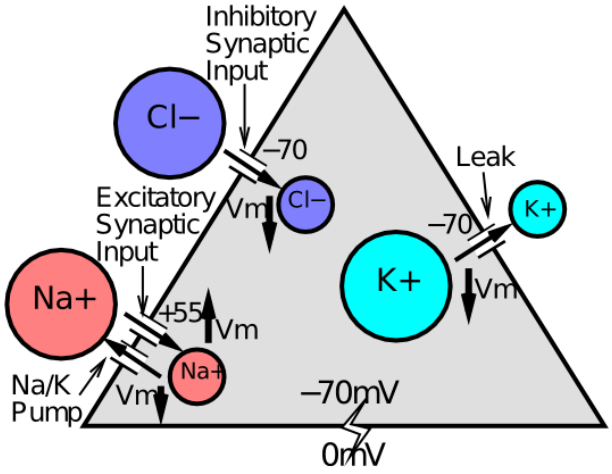


Figure 2.13: Major ions and their relative concentrations inside and outside the neuron (indicated by the size of the circles). These relative concentration differences give rise to the different driving potentials for different ions, and thus determine their net effect on the neuron (whether they pull it "up" for excitation or "down" for inhibition).

Figure 2.13 shows the situation inside and outside the neuron for the major ion types. The concentration imbalances all stem from a steady **sodium pump** that pumps Na+ ions out of the cell. This creates an imbalance in electrical charge, such that the inside of the neuron is more negative (missing all those Na+ ions) and the outside is more positive (has an excess of these Na+ ions). This negative net charge (i.e., **negative resting potential**) of about -70mV pushes the negative Cl- ions outside the cell as well (equivalently, they are drawn to the positive charge outside the cell), creating a concentration imbalance in chloride as well. Similarly, the K+ ions are drawn ''into" the cell by the extra negative charge within, creating an opposite concentration imbalance for the potassium ions.

All of these concentration imbalances create a strong diffusion force, where these ions are trying to distribute themselves more uniformly. But this diffusion force is counteracted by the electrical force, and when the neuron is at rest, it achieves an **equilibrium** state where the electrical and diffusion forces exactly balance and cancel each other out. Another name for the diving potential for an ion (i.e., which direction it

pulls the cell's membrane potential) is the **equilibrium potential** – the electrical potential at which the diffusion and electrical forces exactly balance.

As shown in Figure 2.13, the Cl- and K+ ions have driving potentials that are essentially equivalent to the resting potential, -70mV. This means that when the cell's membrane potential is at this -70mV, there is no net current across the membrane for these ions – everything will basically stay put.

Mathematically, we can capture this phenomenon using the same equation we derived from the tug-of-war analogy:

$$I = G(E - V)$$

Notice that this is just a simple modification of Ohm's law – the E value (the driving potential) "corrects" Ohm's law to take into account any concentration imbalances and the diffusion forces that they engender. If there are no concentration imbalances, then E = 0, and you get Ohm's law (modulo a minus sign that we'll deal with later).

If we plug an E value of -70mV into this equation, then we see that the current is 0 when V = -70mV. This is the definition of an equilibrium state. No net current.

Now consider the Na+ ion. Both the negative potential inside the neuron, and the concentration imbalance, drive this ion to want to move into the cell. Thus, at the resting potential of -70mV, the current for this ion will be quite high if it is allowed to flow into the cell. Indeed, it will not stop coming into the cell until the membrane potential gets all the way up to +55mV or so. This equilibrium or driving potential for Na+ is positive, because it would take a significant positive potential to force the Na+ ions back out against their concentration difference.

The bottom line of all this is that synaptic channels that allow Na+ ions to flow will cause Na+ to flow *into* the neuron, and thereby excite the receiving neuron. In effect, the sodium pump "winds up" the neuron by creating these concentration imbalances, and thus the potential for excitation to come into the cell against a default background of the negative resting potential.

Finally, when excitatory inputs do cause the membrane potential to increase, this has the effect of drawing more Cl- ions back into the cell, creating an inhibitory pull back to the -70mV resting value, and similarly it pushes K+ ions out of the cell, which also makes the inside of the cell more negative, and has a net inhibitory effect. The Cl- ions only flow when inhibitory GABA channels are open, and the K+ ions flow all the time through the always-open leak channels.

Net Input Detail

Here we describe in full detail how the excitatory conductance *Ge* or *net input* to the neuron is computed, taking into account differences across different sources of input to a given neuron. In the main chapter, the core computation is summarized, as an average of the weights times sending activations:

$$g_e(t) = \frac{1}{n} \sum_i x_i w_i$$

where n is the total number of channels, and x_i is the **activity** of a particular sending neuron indexed by the subscript i, and w_i is the **synaptic weight strength** that connects sending neuron i to the receiving neuron.

The overall goal of the more elaborate net input calculation described here, which is what is actually used in the *emergent* software, is to ensure that inputs from different layers having different overall levels of activity have a similar impact on the receiving neuron in terms of overall magnitude of net input, while also allowing for the strength of these different inputs to be manipulated in ways that continue to work functionally. For example, a "localist" input layer may have only one unit active out of 100 (1%) whereas a hidden layer may have 25% activity (e.g., 25 out of 100) – this vast difference in overall activity level would make these layers have very disparate influence on a receiving layer if not otherwise compensated for. Terminologically, we refer to the set of connections from a given sending layer as a **projection**.

The full equation for the net input is as follows, which contains a double sum, first over the different projections, indexed by the letter k, and then within that by the receiving connections for each projection,

indexed by the letter i (where these are understood to vary according to the outer projection loop):

$$g_e(t) = \sum_k \left[s_k \left(\frac{r_k}{\sum_p r_p} \right) \frac{1}{\alpha_k} \frac{1}{n_k} \sum_i (x_i w_i) \right]$$

The factors in this equation are:

- s_k = absolute scaling parameter (set at the user's discretion) for the projection, represented by `WtScale.Abs` parameter in the LeabraConSpec in *emergent*.
- r_k = relative scaling parameter for the projection, which is always normalized by the sum of the relative parameters for all the other projections, which is what makes it relative – the total is constant and one can only alter the relative contributions – represented by `WtScale.Rel` in *emergent*.
- α_k = effective expected activity level for the sending layer, computed as described below, which serves to equalize projections regardless of differences in expected activity levels.
- n_k = number of connections within this projection

The equations for computing the effective expected activity level α_k are based on the integer counts of numbers of expected active inputs on a given projection – this takes into account both the sending layer expected activation, and the number of connections being received. For example, consider a projection from a layer having 1% activity (1 out of 100 units active), with only a single incoming connection from that layer. Even though the odds of this single incoming connection having an active sending unit are 1% on average, *some* receiving unit in the layer is highly likely to be getting that 1 sending unit active. Thus, we use the "highest expected activity level" on the layer, which is 1, rather than the average expected sending probability, which is 1%.

Specifically, the equations, using pseudo-programming variables with longer names instead of obscure mathematical symbols, are:

- `alpha_k = MIN(%_activity * n_recv_cons + sem_extra, r_max_act_n)`
  - `%_activity` = % expected activity on sending layer
  - `n_recv_cons` = number of receiving connections in projection
  - `sem_extra` = standard error of the mean (SEM) extra buffer, set to 2 by default – this makes it the highest expected activity level by including effectively 4 SEM's above the mean, where the real SEM depends on `%_activity` and is a maximum of .5 when `%_activity` = .5.
  - `r_max_act_n = MIN(n_recv_cons, %_activity * n_units_in_layer)` = hard upper limit maximum on number of active inputs – can't be any more than either the number of connections we receive, or the total number of active units in the layer

See the emer/leabra README docs for more detailed information about parameters, monitoring the actual relative net input contributions from different projections, etc.

Math Derivation

This shows all the algebra to derive the equilibrium membrane potential from the update equation – it will only be viewable on the PDF version.

iterative Vm update equation:

$$V_m(t) = V_m(t-1) + dt_{vm} \left[g_e(E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m) \right]$$

just the change part:

$$\Delta Vm = g_e \left(E_e - V_m \right) + g_i(E_i - V_m) + g_l(E_l - V_m)$$

set it to zero:

$$0 = g_e \left(E_e - V_m \right) + g_i(E_i - V_m) + g_l(E_l - V_m)$$

solve for Vm: (multiply all the g's through)

$$0 = g_e E_e - g_e V_m + g_i E_i - g_i V_m + g_l E_l - g_l V_m$$

solve for Vm: (gather Vm terms on other side)

$$g_e V_m + g_i V_m + g_l V_m = g_e E_e + g_i E_i + g_l E_l$$

solve for Vm: (get only one Vm guy, then divide each side by g's to get..)

$$V_m(g_e + g_i + g_l) = g_e E_e + g_i E_i + g_l E_l$$

solution!

$$V_m = \frac{g_e E_e + g_i E_i + g_l E_l}{g_e + g_i + g_l}$$

Another way of writing this solution, which makes its meaning a bit clearer, is:

$$V_m = \frac{g_e}{g_e + g_i + g_l} E_e + \frac{g_i}{g_e + g_i + g_l} E_i + \frac{g_l}{g_e + g_i + g_l} E_l$$

In the Adaptive Exponential function, there is an adaptive factor ω (greek omega) that enters into the membrane update equation, which we can include in our equilibrium calculation:

iterative Vm update equation:

$$V_m(t) = V_m(t-1) + dt_{vm}\left[g_e(E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m) - \omega\right]$$

just the change part:

$$\Delta Vm = g_e\left(E_e - V_m\right) + g_i(E_i - V_m) + g_l(E_l - V_m) - \omega$$

set it to zero:

$$0 = g_e\left(E_e - V_m\right) + g_i(E_i - V_m) + g_l(E_l - V_m) - \omega$$

solve for Vm: (multiply all the g's through)

$$0 = g_e E_e - g_e V_m + g_i E_i - g_i V_m + g_l E_l - g_l V_m - \omega$$

solve for Vm: (gather Vm terms on other side)

$$g_e V_m + g_i V_m + g_l V_m = g_e E_e + g_i E_i + g_l E_l - \omega$$

solve for Vm: (get only one Vm guy, then divide each side by g's to get..)

$$V_m(g_e + g_i + g_l) = g_e E_e + g_i E_i + g_l E_l - \omega$$

solution:

$$V_m = \frac{g_e E_e + g_i E_i + g_l E_l - \omega}{g_e + g_i + g_l}$$

And here is the derivation of the equation for g_e^{Θ}:

equilibrium Vm at threshold:

$$\Theta = \frac{g_e^{\Theta} E_e + g_i E_i + g_l E_l}{g_e^{\Theta} + g_i + g_l}$$

solve for g__e:

$$\Theta(g_e^{\Theta} + g_i + g_l) = g_e^{\Theta} E_e + g_i E_i + g_l E_l$$

(multiply both sides by g's), then solve for g_e:

$$\Theta g_e^{\Theta} = g_e^{\Theta} E_e + g_i E_i + g_l E_l - \Theta g_i - \Theta g_l$$

(bring non-g_e's back to other side), then solve for g_e:

$$g_e^{\Theta} \Theta - g_e^{\Theta} E_e = g_i(E_i - \Theta) + g_l(E_l - \Theta)$$

(bring other g_e over and consolidate other side, then divide both sides to isolate g_e to get), solution:

$$g_e^{\Theta} = \frac{g_i(E_i - \Theta) + g_l(E_l - \Theta)}{\Theta - E_e}$$

In the AdEx function, there is an adaptive factor ω (greek omega) that enters into the membrane update equation, which we can include in our equilibrium calculation as above:

equilibrium Vm at threshold:

$$\Theta = \frac{g_e^{\Theta} E_e + g_i E_i + g_l E_l - \omega}{g_e^{\Theta} + g_i + g_l}$$

solve for g_e:

$$\Theta(g_e^{\Theta} + g_i + g_l) = g_e^{\Theta} E_e + g_i E_i + g_l E_l - \omega$$

(multiply both sides by g's), then solve for g_e:

$$\Theta g_e^{\Theta} = g_e^{\Theta} E_e + g_i E_i + g_l E_l - \Theta g_i - \Theta g_l - \omega$$

(bring non-g_e's back to other side), then solve for g_e:

$$g_e^{\Theta} \Theta - g_e^{\Theta} E_e = g_i(E_i - \Theta) + g_l(E_l - \Theta) - \omega$$

(bring other g_e over and consolidate other side, then divide both sides to isolate g_e to get), solution:

$$g_e^{\Theta} = \frac{g_i(E_i - \Theta) + g_l(E_l - \Theta) - \omega}{\Theta - E_e}$$

Frequency Current Curve

This is written by Sergio Verduzco-Flores:

The smoothed version of the XX1 function has a sigmoidal shape which represents the average firing-rate of a population of discrete spiking neurons given one constant excitatory stimulus and some noise. To a certain extent, it can also represent the average firing-rate of a single spiking neuron receiving a constant input. For simplicity, we will only discuss why the smoothed XX1 function can approximate the firing-rate of a single neuron.

The firing-rate resulting from a constant input is something that can be obtained for any spiking neuronal model using numerical simulations, and in a few cases using math (as shown below). This frequency response can also be measured in real neurons, and is helpful to characterize their behavior. Physiologists can use very sharp glass pipettes to inject electrical current into neurons and measure the resulting frequency of spiking. The function which maps each constant current input to its corresponding steady-state frequency is called the frequency-current curve (or **f-I curve**).

One property of the f-I curve for a large class of neuronal models (and real neurons) is that they have a shape resembling that of the smoothed XX1 function. This implies that the smoothed XX1 function can be used to approximate the firing-rate of neurons in this class. The discussion below is meant to show where the shape of the f-I curve comes from. Since the inputs to the XX1 model are conductances, we will actually analyze the shape of the function $f(g_e)$, which maps the excitatory conductance g_e into a response frequency f. The case for the current input is qualitatively similar.

Consider the simplified spiking model discussed before, which emits a spike whenever the the membrane voltage reaches a threshold value. This is known as an "integrate-and-fire" model. In our case the membrane potential obeys this differential equation:

$$C_m \frac{dV_m}{dt} = g_e(E_e - V_m) + g_i(E_i - V_m) + g_l(E_l - V_m)$$

where C_m is the membrane capacitance.

Excitatory inputs to the neuron will increase the conductance g_e, leading to a current $I = g_e(E_e - Vm)$ that increases the membrane potential, and leads to a spike when $V_m > \Theta$. In the case of a constant input such that $g_e > g_e^{\Theta}$, the neuron will fire periodically with a frequency f. We will now obtain $f(g_e)$.

Rewrite the Vm equation:

$$C_m \frac{dV_m}{dt} = -(g_e + g_i + g_l)V_m + (g_e E_e + g_i E_i + g_l E_l).$$

$$\frac{dV_m}{dt} = -AV_m + B$$

where

$$A = (g_e + g_i + g_l)/C_m, B = (g_e E_e + g_i E_i + g_l E_l)/C_m.$$

This is a first-order linear differential equation, which can be solved using various methods. We will assume that all conductances are constant and solve the differential equation using separation of variables. An additional assumption is that at $V_m(0) = V_r$ with V_r being the reset potential.

Solve the differential equation:

$$\frac{dV_m}{B - AV_m} = dt,$$

$$\int_0^t \frac{dV_m}{B - AV_m} = \int_0^t dt.$$

The left-hand integral can be solved using the substitution $Y = AV_m + B$:

$$\int_0^t \frac{dV_m}{B - AV_m} = -\frac{1}{A} \int_{B - AV_r}^{B - AV_m(t)} \frac{dY}{Y}$$

$$= ln\left(\frac{B - AV_m(t)}{B - AV_r}\right) = t.$$

Solving the last equation with respect to $V_m(t)$ we obtain:

$$V_m(t) = \left(V_r - \frac{B}{A}\right) e^{-At} + \frac{B}{A}.$$

Notice that as $t \to \infty$, $V_m(t) \to B/A$. Thus, the condition for spiking is $B/A > \Theta$. This is exactly equivalent to $g_e > g_e^{\Theta}$, and will be assumed in the next step.

Find the time to reach the threshold. Let T be the time required to go from the reset voltage to the spike threshold. From the solution to the differential equation we have:

$$\Theta = \left(V_r - \frac{B}{A}\right) e^{-AT} + \frac{B}{A}.$$

Solving for T we have:

$$T = (1/A) ln\left(\frac{Vr - (B/A)}{\Theta - (B/A)}\right).$$

$$f = \frac{1}{T} = \frac{A}{ln\left(\frac{Vr-(B/A)}{\Theta-(B/A)}\right)}.$$

It is simple to see that the function f just obtained is asymptotically linear. Notice that the term:

$$B/A = \frac{(g_e E_e + g_i E_i + g_l E_l)}{(g_e + g_i + g_l)}$$

tends towards the constant E_e as g_e grows. Therefore, the whole denominator in the f function becomes roughly a constant as g_e grows, causing f to be a straight line.

The firing-rate of the integrate-and-fire neuron becomes linear for larger values of the excitatory conductance. This is unlike any real neuron, because for larger values of g_e our model neuron will always respond with larger firing rates, without any limit to how fast it can be. A real neuron, on the other hand, is limited to respond with frequencies always smaller than a maximum frequency. There are three main factors which cause this saturation in the firing rate:

- The neuron's refractory period, which is a brief lapse right after spiking during which the neuron becomes unresponsive.
- The time required for the neuron to produce a spike and return to its reset value after it reaches the threshold voltage.
- The balance of electrical ions inside the cell.

We can easily add a refractory period to our integrate-and-fire model, by requiring that the neuron remains at the reset voltage V_r for T_r milliseconds after spiking. The period T^* of this new model will simply be the period of the old integrate-and-fire model plus the T_r term:

$$T^* = T + T_r,$$

$$f^* = \frac{1}{T + T_r}.$$

This now bends the straight line response in the firing rate to become flattened by the introduction of the refractory period. Clearly, no frequency can be larger than $f^*_{max} = 1/T_r$. The qualitative effect of introducing a time to spike and return to the reset value is similar.

The AdEx model does not explicitly introduce a refractory period, but it has two features which flatten its f-I curve and avoid arbitrarily high firing rates. The first feature is a term in the voltage equation which generates the upswing of its action potentials, so that spikes don't happen instantaneously. The second feature is a term which introduces spike frequency adaptation. When a real neuron is stimulated with a constant current injection it often starts by firing quickly, and then gradually reduces its firing rate. This phenomenon is known as **frequency adaptation**.

In the case of the AdEx model it is not possible to obtain $f(g_e)$ analytically as above, but it is possible to use simulations to numerically approximate it. These simulations show that for a certain parameter regime the frequency response is similar to that which can be approximated by the smooth XX1 function (see Figure 2.9).

Sodium-Gated Potassium Channels for Adaptation (kNa Adapt)

The longer-term adaptation (accomodation / fatigue) dynamics of neural firing in our models are based on sodium (Na) gated potassium (K) currents. As neurons spike, driving an influx of Na, this activates the K channels, which, like leak channels, pull the membrane potential back down toward rest (or even below). Multiple different time constants have been identified and this implementation supports 3: M-type (fast), Slick (medium), and Slack (slow) (Kaczmarek 2013; Kohn 2007; Sanchez-Vives, Nowak, and McCormick 2000; Benda, Maler, and Longtin 2010).

The logic is simplest for the spiking case, and can be expressed in conditional program code:

```
if spike {
    gKNa += Rise * (Max - gKNa)
```

```
} else {
    gKNa -= 1/Tau * gKNa
}
```

The KNa conductance (g_{kna} in mathematical terminology, `gKNa` in the program) rises to a `Max` value with a `Rise` rate constant, when the neuron spikes, and otherwise it decays back down to zero with another time constant `Tau`.

The equivalent rate-code equation just substitutes the rate-coded activation variable in as a multiplier on the rise term:

```
gKNa += act * Rise * (Max - gKNa) - (1/Tau * gKNa)
```

The default parameters, which were fit to various empirical firing patterns and also have proven useful in simulations, are:

Channel Type	Tau (ms)	Rise	Max
Fast (M-type)	50	0.05	0.1
Medium (Slick)	200	0.02	0.1
Slow (Slack)	1000	0.001	1.0

Bayesian Optimal Detector

This optional section shows how the equilibrium membrane potential equation, derived based on the biology of the neuron, can also be understood in terms of Bayesian hypothesis testing (Hinton and Sejnowski 1983; McClelland 1998). In this framework, one is comparing different hypotheses in the face of relevant data, which is analogous to how the detector is testing whether the signal it is looking for is present (h), or not ($\bar{h}$). The probability of h given the current input data d (which is written as $P(h|d)$) is a simple ratio function of two other functions of the relationship between the hypotheses and the data (written here as $f(h,d)$ and $f(\bar{h},d)$):

$$P(h|d) = \frac{f(h,d)}{f(h,d) + f(\bar{h},d)}$$

Thus, the resulting probability is just a function of how strong the support for the detection hypothesis h is over the support for the *null hypothesis* $\bar{h}$. This ratio function may be familiar to some psychologists as the **Luce choice ratio** used in mathematical psychology models for a number of years.

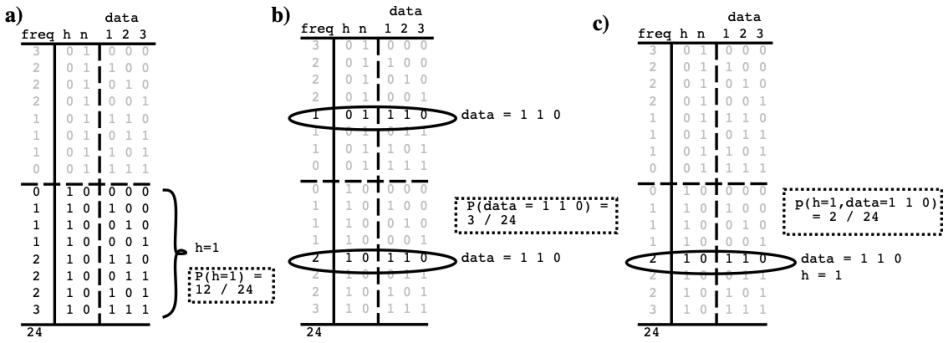


Figure 2.14: Simple example data to compute probabilities from, for line detecto – just add up number of cases where a given condition is true, and divide by the total number of cases (24): **a** $P(h = 1) = 12/24 = .5$. **b** $P(d = 110) = 3/24 = .125$. $$c** $P(h = 1, d = 110) = 2/24 = .0833$.

To have a concrete example to work with, consider a detector that receives inputs from three sources, such that when a vertical line is present (which is what it is trying to detect), all three sources are likely to be activated (Figure 2.14). The hypothesis h is thus that a vertical line is actually present in the world, and $\bar{h}$ is that it is not. h and $\bar{h}$ are *mutually exclusive* alternatives: their summed probability is always 1. There

are three basic probabilities that we are interested in that can be computed directly from the world state table – you just add up the number of cases where a given situation is true, and divide by the total number of cases (with explicit and complete data, probability computations are just accounting):

- The probability that the hypothesis h is true, or $P(h = 1)$ or just $P(h)$ for short = 12/24 or .5.
- The probability of the current input data, e.g., $d = 110$, which is $P(d = 110)$ or $P(d)$ for short = 3/24 (.125) because it occurs 1 time when the hypothesis is false, and 2 times when it is true.
- The *intersection* of the first two, known as the *joint probability* of the hypothesis *and* the data, written $P(h = 1, d = 110)$ or $P(h, d)$, which is 2/24 (.083).

The joint probability tells us how often two different states co-occur compared to all other possible states, but we really just want to know how often the hypothesis is true *when we receive the particular input data* we just got. This is the **conditional probability** of the hypothesis given the data, which is written as $P(h|d)$, and is defined as follows:

$$P(h|d) = \frac{P(h, d)}{P(d)}$$

So, in our example where we got $d = 110$, we want to know:

$$P(h = 1|d = 110) = \frac{P(h = 1, d = 110)}{P(d = 110)}$$

which is (2/24) / (3/24), or .67 according to our table. Thus, matching our intuitions, this tells us that having 2 out of 3 inputs active indicates that it is more likely than not that the hypothesis of a vertical line being present is true. The basic information about how well correlated this input data and the hypothesis are comes from the joint probability in the numerator, but the denominator is critical for *scoping* this information to the appropriate context (cases where the particular input data actually occurred).

The above equation is what we want the detector to solve, and if we had a table like the one in Figure 2.14, then we have just seen that this equation is easy to solve. However, having such a table is nearly impossible in the real world, and that is the problem that Bayesian math helps to solve, by flipping around the conditional probability the other way, using what is called the **likelihood**:

$$P(d|h) = \frac{P(h, d)}{P(h)}$$

It is a little bit strange to think about computing the probability of the *data*, which is, after all, just what was given to you by your inputs (or your experiment), based on your hypothesis, which is the thing you aren't so sure about! However, think of it instead as how likely you would have *predicted* the data based on the assumptions of your hypothesis. In other words, the likelihood computes how well the data fit with the hypothesis.

Mathematically, the likelihood depends on the same joint probability of the hypothesis and the data, we used before, but it is *scoped* in a different way. This time, we scope by all the cases where the hypothesis was true, and determine what fraction of this total had the particular input data state:

$$P(d = 110|h = 1) = \frac{P(h = 1, d = 110)}{P(h = 1)}$$

which is (2/24) / (12/24) or .167. Thus, one would expect to receive this data .167 of the time when the hypothesis is true, which tells you how likely it is you would predict getting this data knowing only that the hypothesis is true.

The main advantage of a likelihood function is that we can often compute it directly as a function of the way our hypothesis is specified, without requiring that we actually know the joint probability $P(h, d)$ (i.e., without requiring a table of all possible events and their frequencies). Assuming that we have a likelihood function that can be computed directly, **Bayes formula** is just a simple bit of algebra that eliminates the need for the joint probability:

$$P(h, d) = P(d|h)P(h)$$

such that:

$$P(h|d) = \frac{P(d|h)P(h)}{P(d)}$$

It allows you to write $P(h|d)$, which is called the *posterior* in Bayesian terminology, in terms of the likelihood times the *prior*, which is what $P(h)$ is called. The prior indicates how likely the hypothesis is to be true without having seen any data at all — some hypotheses are just more plausible (true more often) than others, and this can be reflected in this term. Priors are often used to favor *simpler* hypotheses as more likely, but this is not necessary. In our application here, the prior terms will end up being constants, which can actually be measured (at least approximately) from the underlying biology.

The last barrier to actually using Bayes formula is the denominator $P(d)$, which requires somehow knowing how likely this data is compared to any other. Conveniently, we can replace $P(d)$ with an expression involving only likelihood and prior terms if we make use of the null hypothesis $\bar{h}$. Because the hypothesis and null hypothesis are mutually exclusive and sum to 1, we can write the probability of the data in terms of the part of it that overlaps with the hypothesis plus the part that overlaps with the null hypothesis:

$$P(d) = P(h, d) + P(\bar{h}, d)$$

In Figure 2.14, this amounts to computing $P(d)$ in the top and bottom halves separately, and then adding these results to get the overall result:

$$P(d) = P(d|h)P(h) + P(d|\bar{h})P(\bar{h})$$

which can then be substituted into Bayes formula, resulting in:

$$P(h|d) = \frac{P(d|h)P(h)}{P(d|h)P(h) + P(d|\bar{h})P(\bar{h})}$$

This is now an expression that is strictly in terms of just the likelihoods and priors for the two hypotheses! Furthermore, it is this is the same equation that we showed at the outset, with $f(h, d) = P(d|h)P(h)$ and $f(\bar{h}, d) = P(d|\bar{h})P(\bar{h})$. It has a very simple $\frac{h}{h+\bar{h}}$ form, which reflects a *balancing* of the likelihood in favor of the hypothesis with that against it. It is this form that the biological properties of the neuron implement. You can use the table in Figure 2.14 to verify that this equation gives the same results (.67) as we got using the joint probability directly.

The reason we cannot use something like the table in Figure 2.14 in the real world is that it quickly becomes intractably large due to the huge number of different unique combinations of input states. For example, if the inputs are binary (which is not actually true for neurons, so it's even worse), the table requires 2^{n+1} entries for n inputs, with the extra factor of two (accounting for the +1 in the exponent) reflecting the fact that all possibilities must be considered twice, once under each hypothesis. This is roughly $1.1x10^{301}$ for just 1,000 inputs (and our calculator gives Inf as a result if we plug in a conservative guess of 5,000 inputs for a cortical neuron).

In lieu of the real data, we have to fall back on coming up with plausible ways of directly computing the likelihood terms. One plausible assumption for a detector is that the likelihood is directly (linearly) proportional to the number of inputs that match what the detector is trying to detect, with a linear factor to specify to what extent each input source is representative of the hypothesis. These parameters are just our standard weight parameters w. Together with the linear proportionality assumption, this gives a likelihood function that is a normalized linear function of the weighted inputs:

$$P(d|h) = \frac{1}{z} \sum_i d_i w_i$$

where d_i is the value of one input source i (e.g., $d_i = 1$ if that source detected something, and 0 otherwise), and the normalizing term $\frac{1}{z}$ ensures that the result is a valid probability between 0 and 1.

The fact that we are *defining* probabilities, not *measuring* them, makes these probabilities *subjective*, as compared to frequencies of objectively measurable events in the world. Nevertheless, the Bayesian math ensures that you're integrating the relevant information in the mathematically correct way, at least.

To proceed, one could define the following likelihood function:

$$P(d|h) = \frac{1}{12}\sum_i x_i w_i$$

and similarly for the null hypothesis, which is effectively the negation:

$$P(d|\bar{h}) = \frac{1}{12}\sum_i (1 - x_i) w_i$$

If you plug these into the Bayesian equation, together with the simple assumption that the prior probabilities are equal, $P(h) = P(\bar{h}) = .5$, you get the same results we got from the table.

Finally, we compare the equilibrium membrane potential equation:

$$V_m = \frac{g_e \bar{g}_e E_e + g_i \bar{g}_i E_i + \bar{g}_l E_l}{g_e \bar{g}_e + g_i \bar{g}_i + \bar{g}_l}$$

ot the Bayesian formula, where the excitatory input plays the role of the likelihood or support for the hypothesis, and the inhibitory input and leak current both play the role of support for null hypotheses. Because we have considered only one null hypothesis in the preceding analysis (though it is easy to extend it to two), we will just ignore the leak current for the time being, so that the inhibitory input will play the role of the null hypothesis.

Interestingly, the reversal potentials have to be 0's and 1's to fit the numerical values of probabilities, such that excitatory input drives the potential toward 1 (i.e., $E_e = 1$), and that the inhibitory (and leak) currents drive the potential toward 0 (i.e., $E_i = E_l = 0$).

$$V_m \approx P(h|d)\frac{g_e \bar{g}_e}{g_e \bar{g}_e + g_i \bar{g}_i}$$

$$V_m \approx \frac{P(d|h)P(h)}{P(d|h)P(h) + P(d|\bar{h})P(\bar{h})}$$

The full equation for V_m with the leak current can be interpreted as reflecting the case where there are two different (and independent) null hypotheses, represented by inhibition and leak. As we will see in more detail in the *Network* Chapter, inhibition dynamically changes as a function of the activation of other units in the network, whereas leak is a constant that sets a basic minimum standard against which the detection hypothesis is compared. Thus, each of these can be seen as supporting a different kind of null hypothesis.

Taken together, this analysis provides a satisfying computational-level interpretation of the biological activation mechanism, and assures us that the neuron is integrating its information in a way that makes good statistical sense.

Chapter 3: Networks

In this chapter, we build upon the previous *Neuron* Chapter to understand how networks of detectors can produce emergent behavior that is more than the sum of their simple neural constituents. We focus on the networks of the **neocortex** ("new cortex", often just referred to as "cortex"), which is the evolutionarily most recent, outer portion of the brain where most of advanced cognitive functions take place. There are three major categories of emergent network phenomena:

- **Categorization** of diverse patterns of activity into relevant groups: For example, faces can look very different from one another in terms of their raw "pixel" inputs, but we can categorize these diverse inputs in many different ways, to treat some patterns as more similar than others: male vs. female, young vs. old, happy vs. sad, "my mother" vs. "someone other", etc. Forming these categories is essential for enabling us to make the appropriate behavioral and cognitive responses (approach vs. avoid, borrow money from, etc.). Imagine trying to relate all the raw inputs of a visual image of a face to appropriate behavioral responses, without the benefit of such categories. The relationship ("mapping") between pixels and responses is just too complex. These intermediate, abstract categories organize and simplify cognition, just like file folders organize and simplify documents on your computer. One can argue that much of intelligence amounts to developing and using these abstract categories in the right ways. Biologically, we'll see how successive layers of neural detectors, organized into a **hierarchy**, enable this kind of increasingly abstract categorization of the world. We will also see that many individual neural detectors at each stage of processing can work together to capture the subtlety and complexity necessary to encode complex conceptual categories, in the form of a **distributed representation**. These distributed representations are also critical for enabling multiple different ways of categorizing an input to be active at the same time – e.g., a given face can be simultaneously recognized as female, old, and happy. A great deal of the emergent intelligence of the human brain arises from multiple successive levels of cascading distributed representations, constituting the collective actions of billions of **excitatory pyramidal** neurons working together in the cortex.
- **Bidirectional excitatory dynamics** are produced by the pervasive bidirectional (e.g., bottom-up and top-down or **feedforward** and **feedback**) connectivity in the neocortex. The ability of information to flow in all directions throughout the brain is critical for understanding phenomena like our ability to focus on the task at hand and not get distracted by irrelevant incoming stimuli (did my email inbox just beep??), and our ability to resolve ambiguity in inputs by bringing higher-level knowledge to bear on lower-level processing stages. For example, if you are trying to search for a companion in a big crowd of people (e.g., at a sporting event or shopping mall), you can maintain an image of what you are looking for (e.g., a red jacket), which helps to boost the relevant processing in lower-level stages. The overall effects of bidirectional connectivity can be summarized in terms of an **attractor dynamic** or **multiple constraint satisfaction**, where the network can start off in a variety of different states of activity, and end up getting "sucked into" a common attractor state, representing a cleaned-up, stable interpretation of a noisy or ambiguous input pattern. Probably the best subjective experience of this attractor dynamic is when viewing an Autostereogram (wikipedia link) – you just stare at this random-looking pattern with your eyes crossed, until slowly your brain starts to fall into the 3D attractor, and the image slowly emerges. The underlying image contains many individual matches of the random patterns between the two eyes at different lateral offsets – these are the constraints in the multiple constraint satisfaction problem that eventually work together to cause the 3D image to appear – this 3D image is the one that best satisfies all those constraints.
- **Inhibitory competition**, mediated by specialized **inhibitory interneurons** is important for providing dynamic regulation of overall network activity, which is especially important when there are positive feedback loops between neurons as in the case of bidirectional connectivity. The existence of epilepsy in the human neocortex indicates that achieving the right balance between inhibition and excitation is difficult – the brain obtains so many benefits from this bidirectional excitation that it apparently lives right on the edge of controlling it with inhibition. Inhibition gives rise to **sparse distributed representations** (having a relatively small percentage of neurons active at a time, e.g., 15% or so), which have numerous advantages over distributed representations that have many neurons active at a time. In addition, we'll see in the Learning Chapter that inhibition plays a key role in the learning

process, analogous to the Darwinian "survival of the fittest" dynamic, as a result of the competitive dynamic produced by inhibition.

We begin with a brief overview of the biology of neural networks in the neocortex.

Biology of the Neocortex

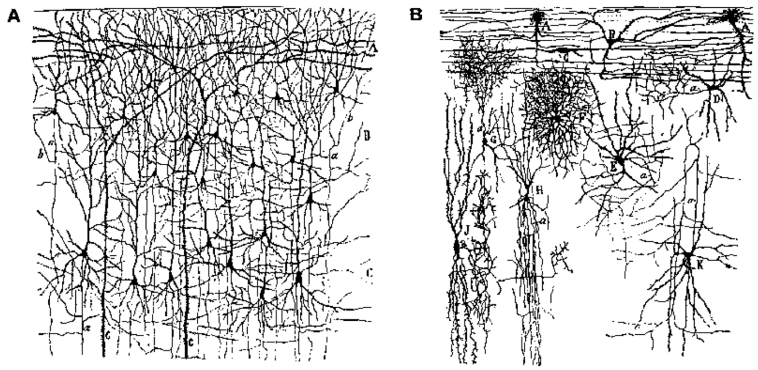


Figure 3.1: Neural constituents of the neocortex. (A) shows excitatory pyramidal neurons, which constitute roughly 85% of neurons, and convey the bulk of the information content via longer-range axonal projections (some of which can go all the way across the brain). (B) shows inhibitory interneurons, which have much more local patterns of connectivity, and represent the remaining 15% of neurons. Reproduced from Crick & Asanuma (1986).

The **cerebral cortex** or **neocortex** is composed of roughly 85% **excitatory** neurons (mainly **pyramidal** neurons, but also **stellate** cells in layer 4), and 15% **inhibitory interneurons** (Figure 3.1). We focus primarily on the excitatory pyramidal neurons, which perform the bulk of the information processing in the cortex. Unlike the local inhibitory interneurons, they engage in long-range connections between different cortical areas, and it is clear that learning takes place in the synapses between these excitatory neurons (evidence is more mixed for the inhibitory neurons). The inhibitory neurons can be understood as "cooling off" the excitatory heat generated by the pyramidal neurons, much like the cooling system (radiator and coolant) in a car engine. Without these inhibitory interneurons, the system would overheat with excitation and lock up in epileptic seizures (this is easily seen by blocking inhibitory GABA channels, for example). There are, however, areas outside of the cortex (e.g., the basal ganglia and cerebellum) where important information processing does take place via inhibitory neurons, and certainly some researchers will object to this stark division of labor even within cortex, but it is nevertheless a very useful simplification.

Layered Structure

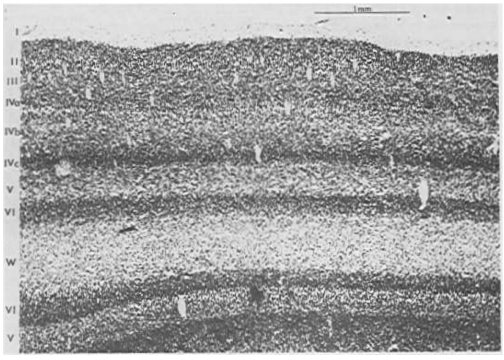


Figure 3.2: A slice of the visual cortex of a cat, showing the six major cortical layers (I - VI), with sublayers of layer IV that are only present in visual cortex. The first layer (I) is primarily axons ("white matter"). Reproduced from Sejnowski and Churchland (1989).

The neocortex has a characteristic 6-layer structure (Figure 3.2), which is present throughout all areas of cortex (Figure 3.3). However, the different cortical areas, which have different functions, have different thicknesses of each of the 6 layers, which provides an important clue to the function of these layers, as summarized in (Figure 3.4). The anatomical patterns of connectivity in the cortex are also an important source of information giving rise to the following functional picture:

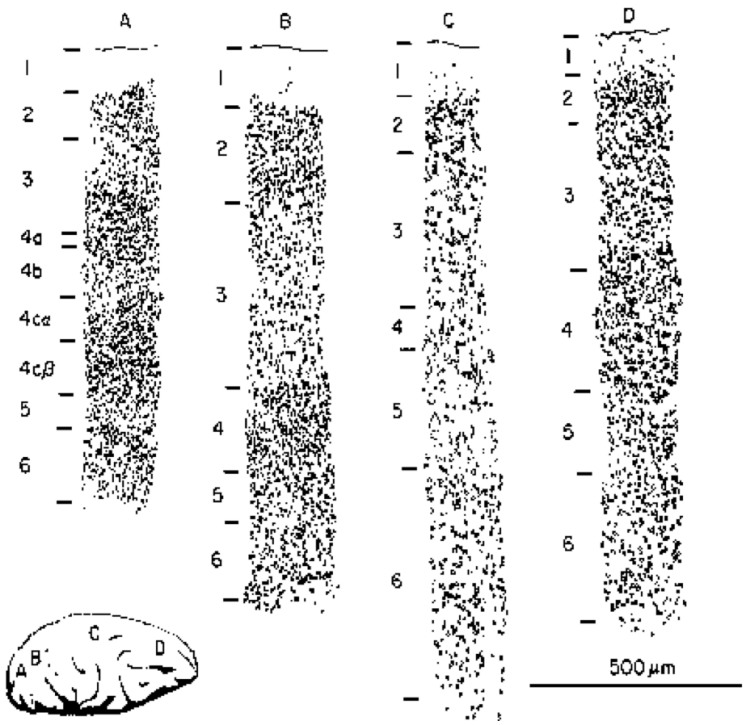


Figure 3.3: The thickness of the different cortical layers varies depending on the location in cortex – this is an important clue to the function of these layers (and the cortical areas). A) shows primary visual cortex (same as Figure 3.2) which emphasizes input layer 4. B) shows extrastriate cortex which processes visual information, and emphasizes superficial layers 2/3. C) shows primary motor cortex, which emphasizes deep layers 5/6. D) shows prefrontal cortex ("executive function") which has an even blend of all layers. Reproduced from Shepherd (1990).

- **Input** areas of the cortex (e.g., primary visual cortex) receive sensory input (typically via the thalamus), and these areas have a greatly enlarged **layer 4**, which is where the axons from the thalamus primarily terminate. The input layer contains a specialized type of excitatory neuron called the **stellate** cell, which has a dense bushy dendrite that is relatively localized, and seems particularly good at collecting the local axonal input to this layer.
- **Hidden** areas of the cortex are so-called because they don't directly receive sensory input, nor do they directly drive motor output – they are "hidden" somewhere in between. The bulk of the cortex is "hidden" by this definition, and this makes sense if we think of these areas as creating increasingly sophisticated and abstract categories from the sensory inputs, and helping to select appropriate behavioral responses based on these high-level categories. This is what most of the cortex does, in one way or another. These areas have thicker **superficial layers 2/3**, which contain many pyramidal neurons that are well positioned for performing this critical categorization function.
- **Output** areas of cortex have neurons that synapse directly onto muscle control areas ("motor outputs"), and are capable of causing physical movement when directly stimulated electrically. These areas have much thicker **deep layers 5/6**, which send axonal projections back down into many different subcortical areas.

In summary, the layer-wise (*laminar*) structure of the cortex and the area-wise function of different cortical areas converge to paint a clear picture about what the cortex does: it takes in sensory inputs,

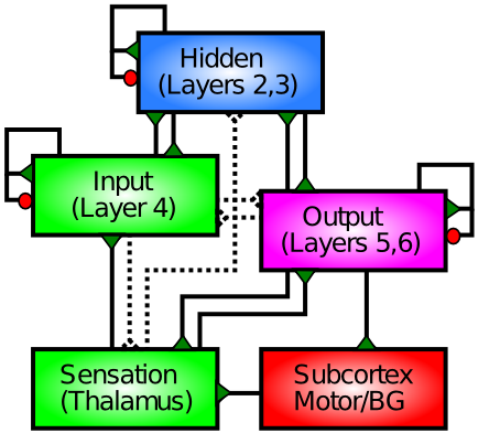


Figure 3.4: Function of the cortical layers: layer 4 processes input information (e.g., from sensory inputs) and drives superficial layers 2/3, which provide a "hidden" internal re-processing of the inputs (extracting behaviorally-relevant categories), which then drive deep layers 5/6 to output a motor response. Green triangles indicate excitation, and red circles indicate inhibition via inhibitory interneurons. BG = basal ganglia which is important for driving motor outputs, and Subcortex includes a large number of other subcortical areas.

processes them in many different important ways to extract behaviorally relevant categories, which can then drive appropriate motor responses. We will adopt this same basic structure for most of the models we explore.

Patterns of Connectivity

The dominant patterns of longer-range connectivity between cortical areas, and lateral connections within cortical areas are shown in Figure 3.5. Consistent with the Input-Hidden-Output laminar structure described above, the **feedforward** flow of information "up" the cortical hierarchy of areas (i.e., moving further away from sensory inputs) goes from Input to Hidden in one area, and then to Input to Hidden in the next area, and so on. This flow of information from sensory inputs deeper into the higher levels of the brain is what supports the formation of increasingly abstract hierarchies of categories that we discuss in greater detail in the next section.

Information flowing in the reverse direction (**feedback**) goes from Hidden & Output in one area to Hidden & Output in the previous area, and so on. We will see later in this chapter how this backward flow of information can support top-down cognitive control over behavior, direct attention, and help resolve ambiguities in the sensory inputs (which are ubiquitous). One might have expected this pattern to go Hidden to Output in one area, to Hidden to Output in the previous area, but this pattern is only part of the story. In addition, the Hidden layers can communicate directly to each other across areas. Furthermore, Output areas can also directly communicate with each other. We can simplify this pattern by assuming that the Output layers in many cortical areas serve more as extra copies of the Hidden layer patterns, which help make additional connections (especially to subcortical areas – all cortical areas project to multiple subcortical areas). Thus, the essential computational functions are taking place directly in the Hidden to Hidden connections between areas (mediated by intervening Input layers for the feedforward direction), and Output layers provide an "external interface" to communicate these Hidden representations more broadly. The exception to this general idea would be in the motor output areas of cortex, where the Output layers may be doing something more independent (they are at least considerably larger in these areas).

Each cortical area also has extensive **lateral** connectivity among neurons within the same area, and this follows the same general pattern as the feedback projection, except that it also terminates in layer 4. These lateral connections serve a very similar functional role as the feedback projections as well – essentially they represent "self feedback".

The other significant aspect of cortical connectivity that will become quite important for our models, is that the connectivity is largely **bidirectional** . Thus, an area that sends a feedforward projection to

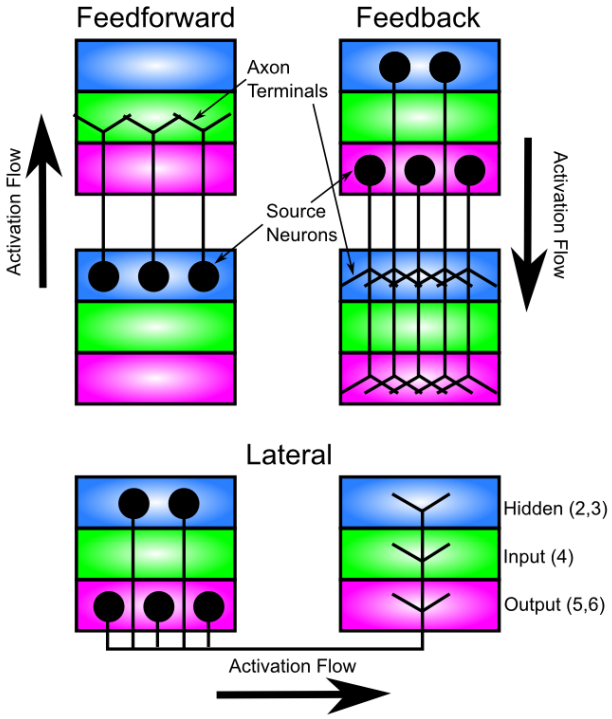


Figure 3.5: Typical patterns of connectivity between cortical areas (Feedforward and Feedback) and within cortical areas (Lateral). Information flows from the Hidden layers "up" in a feedforward direction into the Input layers of "higher" areas (from which it flows into the Hidden layer of that area), and flows back down in a feedback direction from Hidden and Output (output typically stronger as indicated) back to Hidden and Output layers in "lower" areas. All areas have lateral connections that similarly originate in the Hidden and Output layers, and connect to all three layers within another part of the same cortical area. Based on Figure 3 from Felleman and Van Essen (1991).

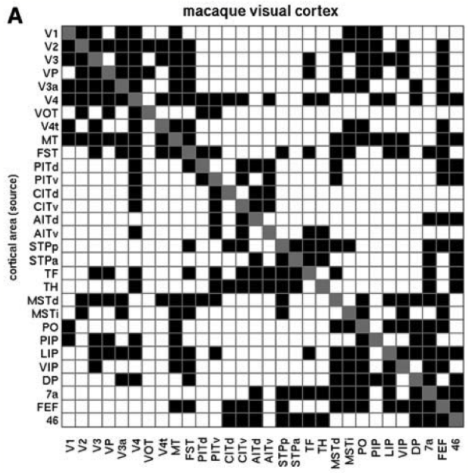


Figure 3.6: Connectivity matrix between cortical areas, showing that when a given area sends a feedforward projection to another area, it typically also receives a feedback projection from that same area. Thus, cortical connectivity is predominantly **bidirectional**. Reproduced from Sporns & Zwi (2004).

another area also typically receives a reciprocal feedback projection from that same area. This bidirectional connectivity is important for enabling the network to converge into a coherent overall state of activity across layers, and is also important for driving error-driven learning as we'll see in the Learning Chapter.

Next, let's see how feedforward excitatory connections among areas can support intelligent behavior by developing categorical representations of inputs.

Categorization and Distributed Representations

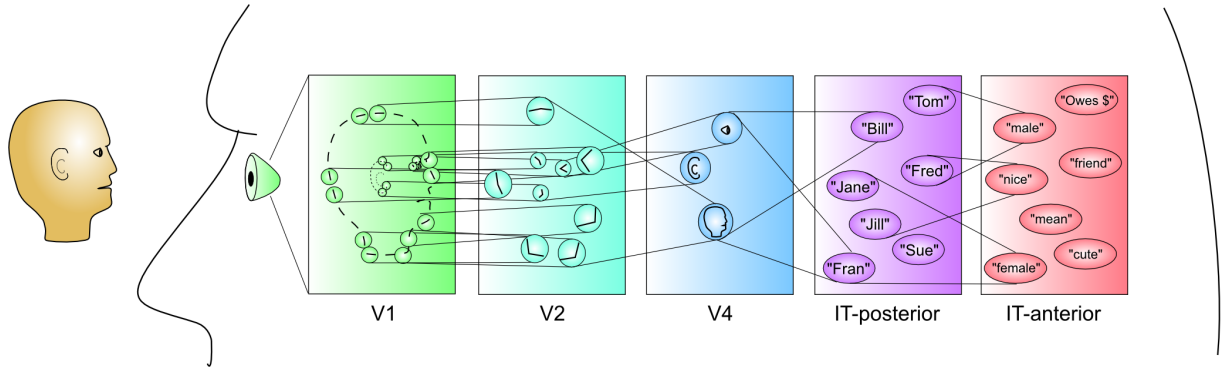


Figure 3.7: Schematic of a hierarchical sequence of categorical representations processing a face input stimulus. Representations are distributed at each level (multiple neural detectors active). At the lowest level, there are elementary feature detectors (oriented edges). Next, these are combined into junctions of lines, followed by more complex visual features. Individual faces are recognized at the next level (even here multiple face units are active in graded proportion to how similar people look). Finally, at the highest level are important functional "semantic" categories that serve as a good basis for actions that one might take – being able to develop such high level categories is critical for intelligent behavior.

As explained in the introduction to this chapter, the process of forming **categorical representations** of inputs coming into a network enables the system to behave in a much more powerful and "intelligent" fashion (Figure 3.7). Philosophically, it is an interesting question as to where our mental categories come from – is there something objectively real underlying our mental categories, or are they merely illusions we impose upon reality? Does the notion of a "chair" really exist *in the real world*, or is it just something that our brains construct for us to enable us to get by (and rest our weary legs)? This issue has been contemplated since the dawn of philosophy, e.g., by Plato with his notion that we live in a cave perceiving only shadows on the wall of the true reality beyond the cave. It seems plausible that there is *something* "objective" about chairs that enables us to categorize them as such (i.e., they are not purely a collective hallucination), but providing a rigorous, exact definition thereof seems to be a remarkably challenging endeavor (try it! don't forget the cardboard box, or the lump of snow, or the miniature chair in a dollhouse, or the one in the museum that nobody ever sat on..). It doesn't seem like most of our concepts are likely to be true "natural kinds" that have a very precise basis in nature. Things like Newton's laws of physics, which would seem to have a strong objective basis, are probably dwarfed by everyday things like chairs that are not nearly so well defined (and "naive" understanding of physics is often not actually correct in many cases either).

The messy ontological status of conceptual categories doesn't bother us very much. As we saw in the previous chapter, Neurons are very capable detectors that can integrate many thousands of different input signals, and can thereby deal with complex and amorphous categories. Furthermore, we will see that learning can shape these category representations to pick up on things that are behaviorally relevant, without requiring any formality or rigor in defining what these things might be. In short, our mental categories develop because they are useful to us in some way or another, and the outside world produces enough reliable signals for our detectors to pick up on these things. Importantly, a major driver for learning these categories is social and linguistic interaction, which enables very complex and obscure things to be learned and shared – the strangest things can be learned through social interactions (e.g., you now know that the considerable extra space in a

bag of chips is called the "snackmosphere", courtesy of Rich Hall). Thus, our cultural milieu plays a critical role in shaping our mental representations, and is clearly a major force in what enables us to be as intelligent as we are (we do occasionally pick up some useful ideas along with things like "snackmosphere"). If you want to dive deeper into the philosophical issues of truth and relativism that arise from this lax perspective on mental categories, see the Chapter Appendix *Philosophy of Categories*.

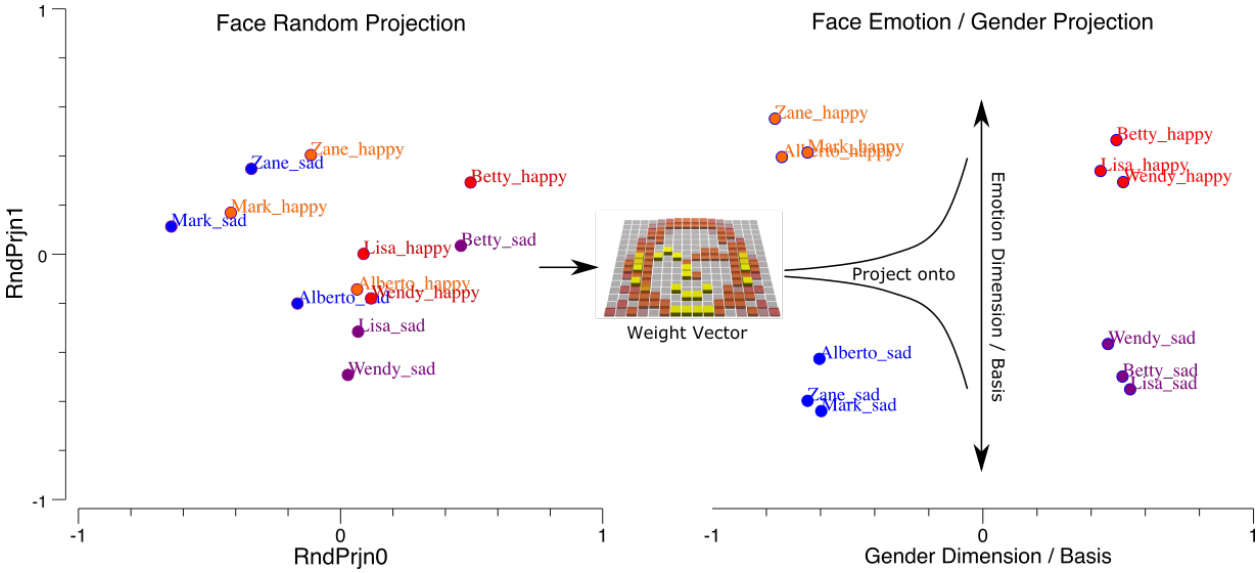


Figure 3.8: How synaptic weights act to *project* input patterns along specific *dimensions* or bases, in this case projecting the inputs along the dimensions of Emotion and Gender. In the left panel, the very high-dimensional face inputs (256 dimensions for a 16x16 image) are projected along two random weight vectors, allowing us to visualize this high-dimensional input space in a 2D plot. In the right panel, the specific synaptic weights trained for discriminating along the emotion vs. gender dimensions have *transformed* or *rotated* the input space into a much more systematic and well-organized, low-dimensional space. This is fundamentally what neurons do: organize and transform input patterns along relevant dimensions, and that is another way of stating that neurons detect stimuli along these dimensions.

Figure 3.8 provides a complementary view of the neuron and its weights, as *projecting input patterns along a specific* ***dimension*** *in a high-dimensional space*. Mathematically, the synaptic weights are a *vector* that multiplies the high-dimensional *input vector* of neural activity signals using a *dot product*, which is just multiplying weights times activations and adding up the total – that is also known as the **projection** of the input space onto the weight vector dimension. This projection operation organizes and systematizes the inputs along dimensions of behavioral importance (e.g., emotion and gender in the case shown in the figure, which is used in the exploration below).

In linear algebra terms, the neural weights *rotate* the input space along a new *basis set*, where a *basis set* is a collection of different *axes* (like the X and Y axes) or dimensions that provides a different way of *encoding* the inputs. Furthermore, in these terms, learning is the process of finding a good such basis set for encoding the inputs, and current deep neural networks used in AI are primarily doing exactly that, over many successive layers that each applies a different such "rotation", such that at the "top" of such a network, a few very informative such dimensions have been extracted (e.g., the object category extracted from a set of input images).

The detector way of looking at the neuron is useful for understanding the roles of inhibition and the neural firing threshold as we saw in the previous chapter – it specifically differentiates between *active* firing for detected items, vs. not firing for everything else, and provides a more "discrete" view of what the neuron is doing. By contrast, the dimension projection framework provides a more continuous, mathematical view. Both are useful ways of understanding what is going on in the brain.

One intuitive way of understanding the importance of having the right categories (and choosing them appropriately for the given situation) comes from **insight problems**. These problems are often designed so that our normal default way of categorizing the situation leads us in the wrong direction, and it is necessary

to **re-represent** the problem in a new way ("thinking outside the box"), to solve it. For example, consider this "conundrum" problem: "two men are dead in a cabin in the woods. what happened?" – you then proceed to ask a bunch of true/false questions and eventually realize that you need to select a different way of categorizing the word "cabin" in order to solve the puzzle. Here is a list of some of these kinds of conundrums (external link).

For computer programmers, one of the most important lessons one learns is that choosing the correct representation is the most important step in solving a given problem. As a simple example, using the notion of a "heap" enables a particularly elegant solution to the sorting problem. Binary trees are also a widely used form of representation that often greatly reduce the computational time of various problems. In general, you simply want to find a representation that makes it easy to do the things you need to do. This is exactly what the brain does.

One prevalent example of the brain's propensity to develop categorical encodings of things are **stereotypes.** A stereotype is really just a mental category applied to a group of people. The fact that everyone seems to have them is strong evidence that this is fundamentally how the brain works. We cannot help but think in terms of abstract categories like this, and as we've argued above, categories in general are essential for allowing us to deal with the world in an intelligent manner. But the obvious problems with stereotypical thinking also indicate that these categories can also be problematic (for stereotypes specifically and categorical thinking more generally), and limit our ability to accurately represent the details of any given individual or situation. As we discuss next, having many different categorical representations active at the same time can potentially help mitigate these problems. The ability to entertain multiple such potential categories at the same time may be an individual difference variable associated with things like political and religious beliefs (todo: find citations). This stuff can get interesting!

Distributed Representations

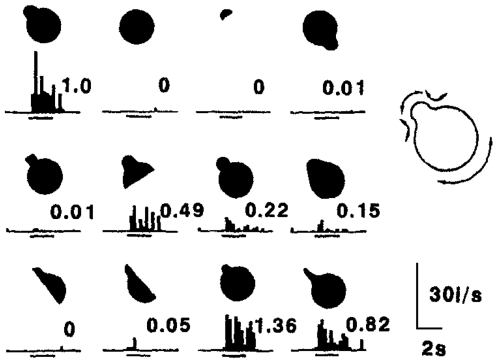


Figure 3.9: Graded response as a function of similarity. This is one aspect of distributed representations, shown here in a neuron in the visual cortex of a monkey – this neuron responds in a graded fashion to different input stimuli, in proportion to how similar they are to the thing that it responds most actively to (as far as is known from presenting a wide sample of different input images). With such graded responses ubiquitous in cortex, it follows that any given input will activate many different neuron detectors. Reproduced from Tanaka (1996).

In addition to our mental categories being somewhat **amorphous**, they are also highly **polymorphous**: any given input can be categorized in many different ways at the same time – there is no such thing as *the* appropriate level of categorization for any given thing. A chair can also be *furniture*, *art*, *trash*, *firewood*, *doorstopper*, *plastic* and any number of other such things. Both the amorphous and polymorphous nature of categories are nicely accommodated by the notion of a **distributed representation**. Distributed representations are made up of many individual neurons-as-detectors, each of which is detecting something different. The aggregate pattern of output activity ("detection alarms") across this population of detectors can capture the amorphousness of a mental category, because it isn't just one single discrete factor that goes into it. There are many factors, each of which plays a role. Chairs have seating surfaces, and sometimes have a backrest, and typically have a chair-like shape, but their shapes can also be highly variable and strange.

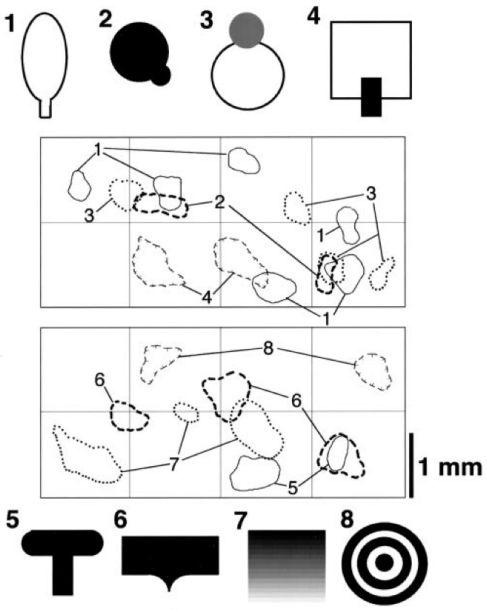


Figure 3.10: Distributed representations of different shapes mapped across regions of inferotemporal (IT) cortex in the monkey. Each shape activates a large number of different neurons distributed across the IT cortex, and these neurons overlap partially in some places. Reproduced from Tanaka (2003).

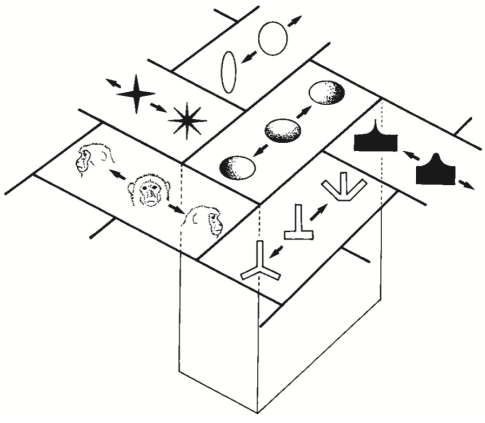


Figure 3.11: Schematic diagram of topographically organized shape representations in monkey IT cortex, from Tanaka (2003) – each small area of IT responds optimally to a different stimulus shape, and neighboring areas tend to have similar but not identical representations.

They are often made of wood or plastic or metal, but can also be made of cardboard or even glass. All of these different factors can be captured by the whole population of neurons firing away to encode these and many other features (e.g., including surrounding context, history of actions and activities involving the object in question).

The same goes for the polymorphous nature of categories. One set of neurons may be detecting chair-like aspects of a chair, while others are activating based on all the different things that it might represent (material, broader categories, appearance, style etc). All of these different possible meanings of the chair input can be active **simultaneously**, which is well captured by a distributed representation with neurons detecting all these different categories *at the same time*.

Some real-world data on distributed representations is shown in Figures 3.8 and 3.9. These show that individual neurons respond in a **graded** fashion as a function of **similarity** to inputs relative to the optimal thing that activates them (we saw this same property in the detector exploration from the Neuron Chapter, when we lowered the leak level so that it would respond to multiple inputs). Figure 3.11 shows an overall summary map of the topology of shape representations in monkey inferotemporal (IT) cortex, where each area has a given optimal stimulus that activates it, while neighboring areas have similar but distinct such optimal stimuli. Thus, any given shape input will be encoded as a distributed pattern across all of these areas to the extent that it has features that are sufficiently similar to activate the different detectors.

Another demonstration of distributed representations comes from a landmark study by (Haxby et al. 2001), using functional magnetic resonance imaging (fMRI) of the human brain, while viewing different visual stimuli (Figure 3.12). They showed that contrary to prior claims that the visual system was organized in a strictly modular fashion, with completely distinct areas for faces vs. other visual categories, for example, there is in fact a high level of overlap in activation over a wide region of the visual system for these different visual inputs. They showed that you can distinguish which object is being viewed by the person in the fMRI machine based on these distributed activity patterns, at a high level of accuracy. Critically, this accuracy level does not go down appreciably when you exclude the area that exhibits the maximal response for that object. Prior "modularist" studies had only reported the existence of these maximally responding areas. But as we know from the monkey data, neurons will respond in a graded way even if the stimulus is not a perfect fit to their maximally activating input, and Haxby et al. showed that these graded responses convey a lot of information about the nature of the input stimulus.

Coarse Coding

Figure 3.13 illustrates an important specific case of a distributed representation known as **coarse coding**. This is not actually different from what we've described above, but the particular example of how the eye uses only 3 photoreceptors to capture the entire visible spectrum of light is a particularly good example of the power of distributed representations. Each individual frequency of light is uniquely encoded in terms of the *relative balance* of graded activity across the different detectors. For example, a color between red and green (e.g., a particular shade of yellow) is encoded as partial activity of the red and green units, with the relative strength of red vs. green determining how much it looks more orange vs. chartreuse. In summary, coarse coding is very important for efficiently encoding information using relatively few neurons.

Localist Representations

The opposite of a distributed representation is a **localist** representation, where a single neuron is active to encode a given category of information. Although we do not think that localist representations are characteristic of the actual brain, they are nevertheless quite convenient to use for computational models, especially for input and output patterns to present to a network. It is often quite difficult to construct a suitable distributed pattern of activity to realistically capture the similarities between different inputs, so we often resort to a localist input pattern with a single input neuron active for each different type of input, and just let the network develop its own distributed representations from there.

Figure 3.14 shows the famous case of a "Halle Berry" neuron, recorded from a person with epilepsy who had electrodes implanted in their brain (Quiroga et al. 2005). This would appear to be evidence for an extreme form of localist representation, known as a **grandmother cell** (a term apparently coined by Jerry Lettvin in 1969), denoting a neuron so specific yet abstract that it only responds to one's grandmother, based

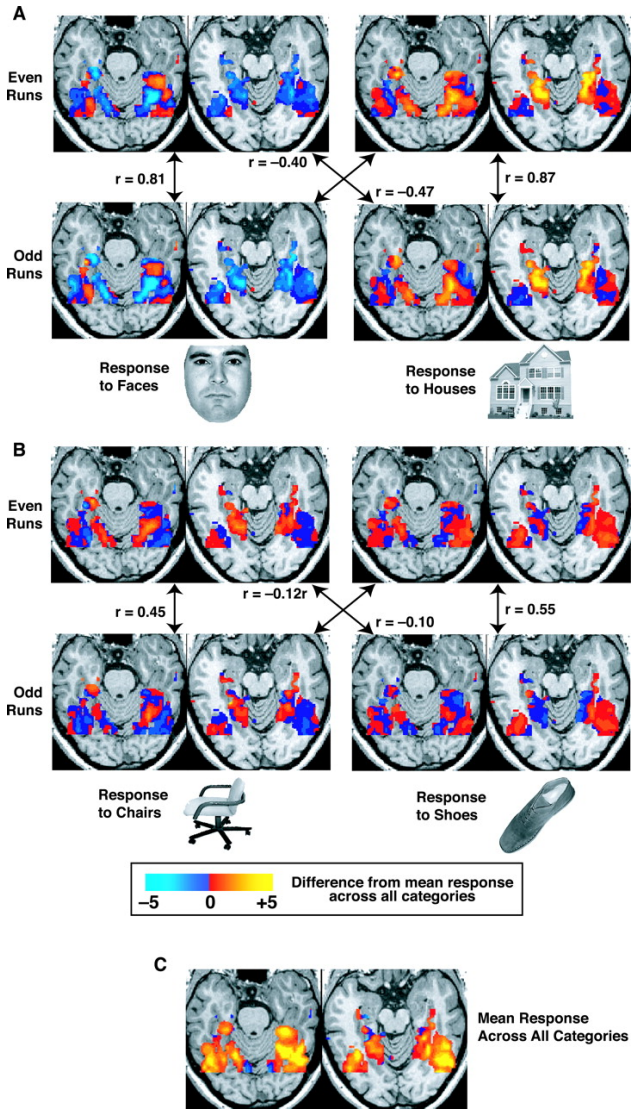


Figure 3.12: Maps of neural activity in the human brain in response to different visual input stimuli (as shown – faces, houses, chairs, shoes), recorded using functional magnetic resonance imaging (fMRI). There is a high level of overlap in neural activity across these different stimuli, in addition to some level of specialization. This is the hallmark of a distributed representation. Reproduced from Haxby et al. (2001).

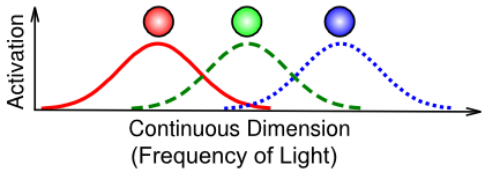


Figure 3.13: Coarse coding, which is an instance of a distributed representation with neurons that respond in a graded fashion. This example is based on the coding of color in the eye, which uses only 3 different photoreceptors tuned to different frequencies of light (red, green blue) to cover the entire visible spectrum. This is a very efficient representation compared to having many more receptors tuned more narrowly and discretely to different frequencies along the spectrum.

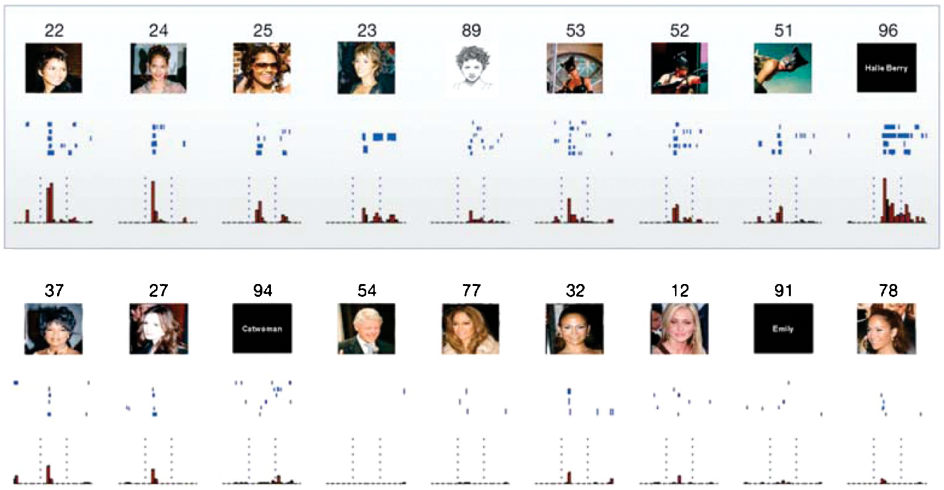


Figure 3.14: The famous case of a Halle Berry neuron recorded from a person with epilepsy who had electrodes implanted in their brain. The neuron appears sensitive to many different presentations of Halle Berry (including just seeing her name in text), but not to otherwise potentially similar people. Although this would seem to suggest the presence of localist "grandmother cells", in fact there are many other distributed neurons activated by any given input such as this within the same area, and even this neuron does exhibit some level of firing to similar distractor cases. Reproduced from Quiroga et al. (2005).

on any kind of input, but not to any other people or things. People had long scoffed at the notion of such grandmother cells. Even though the evidence for them is fascinating (including also other neurons for Bill Clinton and Jennifer Aniston), it does little to change our basic understanding of how the vast majority of neurons in the cortex respond. Clearly, when an image of Halle Berry is viewed, a huge number of neurons at all levels of the cortex will respond, so the overall representation is still highly distributed. But it does appear that, amongst all the different ways of categorizing such inputs, there are a few highly selective "grandmother" neurons! One other outstanding question is the extent to which these neurons actually do show graded responses to other inputs – there is some indication of this in the figure, and more data would be required to really test this more extensively.

Explorations

See the `face_categ` simulation in CCN Sims (Part I only) for an exploration of how face images can be categorized in different ways (emotion, gender, identity), each of which emphasizes some aspect of the input stimuli and collapses across others.

Bidirectional Excitatory Dynamics and Attractors

The feedforward flow of excitation through multiple layers of the neocortex can make us intelligent, but the **feedback** flow of excitation in the opposite direction is what makes us **robust**, **flexible**, and **adaptive**. Without this feedback pathway, the system can only respond on the basis of whatever happens to drive the system most strongly in the feedforward, bottom-up flow of information. But often our first impression is wrong, or at least incomplete. In the "searching for a friend" example from the introduction, we might not get sufficiently detailed information from scanning the crowd to drive the appropriate representation of the person. Top-down activation flow can help focus us on relevant perceptual information that we can spot (like the red coat). As this information interacts with the bottom-up information coming in as we scan the crowd, our brains suddenly converge on the right answer: There's my friend, in the red coat!

The overall process of converging on a good internal representation given a noisy, weak or otherwise ambiguous input can be summarized in terms of **attractor dynamics** (Figure 3.15). An attractor is a concept from *dynamical systems* theory, representing a stable configuration that a dynamical system will tend to gravitate toward. A familiar example of attractor dynamics is the coin gravity well, often found in

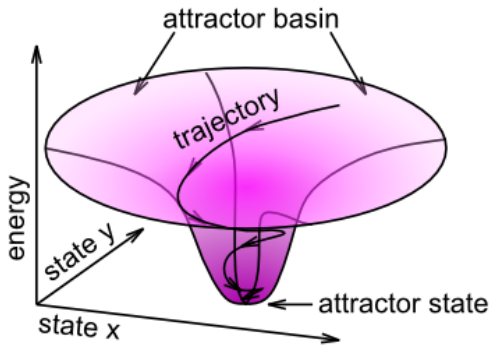


Figure 3.15: Illustration of attractor dynamics, in terms of a "gravity well". In the familiar gravity wells that suck in coins at science museums, the attractor state is the bottom hole in the well, where the coin inevitably ends up. This same dynamic can operate in more abstract cases inside bidirectionally connected networks. For example, the x and y axes in this diagram could represent the activities of two different neurons, and the attractor state indicates that the network connectivity prefers to have neuron x highly active, while neuron y is weakly active. The attractor basin indicates that regardless of what configuration of activations these two neurons start in, they'll end up in this same overall attractor state.

science museums. You roll your coin down a slot at the top of the device, and it rolls out around the rim of an upside-down bell-shaped "gravity well". It keeps orbiting around the central hole of this well, but every revolution brings it closer to the "attractor" state in the middle. No matter where you start your coin, it will always get sucked into the same final state. This is the key idea behind an attractor: many different inputs all get sucked into the same final state. If the attractor dynamic is successful, then this final state should be the correct categorization of the input pattern.

Figure 3.16: A well-known example of an image that is highly ambiguous, but we can figure out what is going on if an appropriate high-level cue is provided, e.g., "Dalmatian". This process of top-down knowledge helping resolve bottom-up ambiguity is a great example of bidirectional processing.

There are many different instances where bidirectional excitatory dynamics are evident:

- **Top-down imagery** – I can ask you to imagine what a purple hippopotamus looks like, and you can probably do it pretty well, even if you've never seen one before. Via top-down excitatory connections, high-level verbal inputs can drive corresponding visual representations. For example, imagining the locations of different things in your home or apartment produces reaction times that mirror the actual spatial distances between those objects – we seem to be using a real spatial/visual representation in our

imagery.

- **Top-down ambiguity resolution** – Many stimuli are ambiguous without further top-down constraints. For example, if you've never seen Figure 3.16 before, you probably won't be able to find the Dalmatian dog in it. But now that you've read that clue, your top-down semantic knowledge about what a dalmatian looks like can help your attractor dynamics converge on a coherent view of the scene.
- **Pattern completion** – If I ask you "what did you have for dinner last night", this partial input cue can partially excite the appropriate memory representation in your brain (likely in the hippocampus), but you need a bidirectional excitatory dynamic to enable this partial excitation to reverberate through the memory circuits and fill in the missing parts of the full memory trace. This reverberatory process is just like the coin orbiting around the gravity well – different neurons get activated and inhibited as the system "orbits" around the correct memory trace, eventually converging on the full correct memory trace (or not!). Sometimes, in so-called **tip of the tongue** states, the memory you're trying to retrieve is *just* beyond grasp, and the system cannot quite converge into its attractor state. Man, that can be frustrating! Usually you try everything to get into that final attractor. We don't like to be in an unresolved state for very long.

Energy and Harmony

There is a mathematical way to capture something like the vertical axis in the attractor (Figure 3.15), which in the physical terms of a gravity well is potential energy. Perhaps not surprisingly, this measure is called **energy** and it was developed by a physicist named John Hopfield. He showed that local updating of unit activation states ends up reducing a global energy measure, much in the same way that local motion of the coin in the gravity well reduces its overall potential energy (Hopfield 1982, 1984). Another physicist, Paul Smolensky, developed an alternative framework with the sign reversed, where local updating of unit activation states *increases global Harmony* (Smolensky 1986). That sounds nice, doesn't it? To see the mathematical details, see Chapter Appendix on *Energy and Harmony*. We don't actually need these equations to run our models, and the basic intuition for what they tell us is captured by the notion of an attractor, so we won't spend any more time on this idea in this main chapter.

Explorations

See `face_categ` in CCN Sims (Part II) for an exploration of how top-down and bottom-up processing interact to produce imagery and help resolve ambiguous inputs (partially occluded faces). These additional simulations provide further elaboration of bidirectional computation:

- `cats-and-dogs` – fun example of attractor dynamics in a simple semantic network.
- `necker-cube`– another fun example of attractor dynamics, showing also the important role of noise, and neural fatigue.

Inhibitory Competition and Activity Regulation

Inhibitory competition plays a critical role in enabling us to focus on a few things at a time, which we can then process effectively without getting overloaded. Inhibition also ensures that those detectors that do get activated are the ones that are the most excited by a given input – in Darwinian evolutionary terms, these are the *fittest* detectors.

Without inhibition, the bidirectional excitatory connectivity in the cortex would quickly cause every neuron to become highly excited, because there would be nothing to check the spread of activation. There are so many excitatory connections among neurons that it doesn't take long for every neuron to become activated. A good analogy is placing a microphone near a speaker that is playing the sound from that microphone – this is a bidirectional excitatory system, and it quickly leads to that familiar, very loud "feedback" squeal. If one's audio system had the equivalent of the inhibitory system in the cortex, it would actually be able to prevent this feedback by dynamically turning down the input gain on the microphone, and/or the output volume of the speaker.

Another helpful analogy is to an air conditioner (AC), which has a thermostat control that determines when it kicks in (and potentially how strong it is). This kind of **feedback control** system allows the room

to warm up to a given **set point** (e.g., 75 degrees F) before it starts to counter the heat. Similarly, inhibition in the cortex is proportional to the amount of excitation, and it produces a similar set point behavior, where activity is prevented from getting too high: typically no more than roughly 15-25% of neurons in any given area are active at a time.

The importance of inhibition goes well beyond this basic regulatory function, however. Inhibition gives rise to **competition** – only the most strongly excited neurons are capable of overcoming the inhibitory feedback signal to get activated and send action potentials to other neurons. This competitive dynamic has numerous benefits in processing and learning. For example, **selective attention** depends critically on inhibitory competition. In the visual domain, selective attention is evident when searching for a stimulus in a crowded scene (e.g., searching for a friend in a crowd as described in the introduction). You cannot process all of the people in the crowd at once, so only a relatively few capture your attention, while the rest are ignored. In neural terms, we say that the detectors for the attended few were sufficiently excited to out-compete all the others, which remain below the firing threshold due to the high levels of inhibition. Both bottom-up and top-down factors can contribute to which neural detectors get over threshold or not, but without inhibition, there wouldn't be any ability to select only a few to focus on in the first place. Interestingly, people with Balint's syndrome, who have bilateral damage to the parietal cortex (which plays a critical role in spatial attention of this sort), show reduced attentional effects and also are typically unable to process anything if a visual display contains more than one item (i.e., "simultanagnosia" – the inability to recognize objects when there are multiple simultaneously present in a scene). We will explore these phenomena in the Perception Chapter.

We will see in the Learning Chapter that inhibitory competition facilitates learning by providing this *selection pressure,* whereby only the most excited detectors get activated, which then gets reinforced through the learning process to make the most active detectors even better tuned for the current inputs, and thus more likely to respond to them again in the future. This kind of positive feedback loop over episodes of learning leads to the development of very good detectors for the kinds of things that tend to arise in the environment. Without the inhibitory competition, a large percentage of neurons would get trained up for each input, and there would be no **specialization** of detectors for specific categories in the environment. Every neuron would end up weakly detecting everything, and thus accomplish nothing. Thus, again we see that competition and limitations can actually be extremely beneficial.

A summary term for the kinds of neural patterns of activity that develop in the presence of inhibitory competition is **sparse distributed representations**. These have relatively few (15-25%) neurons active at a time, and thus these neurons are more highly tuned for the current inputs than they would otherwise be in a fully distributed representation with much higher levels of overall activity. Thus, although technically inhibition does not contribute directly to the basic information processing functions like categorization, because inhibitory connectivity is strictly local within a given cortical area, inhibition does play a critical *indirect* role in shaping neural activity patterns at each level.

Feedforward and Feedback Inhibition

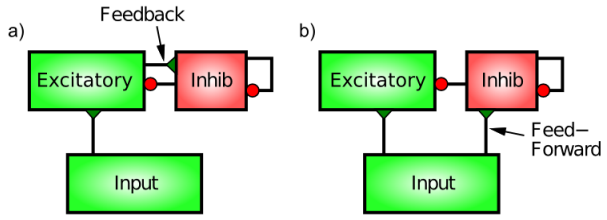


Figure 3.17: Feedforward and Feedback Inhibition. Feedback inhibition reacts to the actual level of activity in the excitatory neurons, by directly responding to this activity (much like an air conditioner reacts to excess heat). Feedforward inhibition anticipates the level of excitation of the excitatory neurons by measuring the level of excitatory input they are getting from the Input area. A balance of both types works best.

There are two distinct patterns of neural connectivity that drive inhibitory interneurons in the cortex, **feedforward** and **feedback** (Figure 3.17). Just to keep things interesting, these are not the same as the

connections among excitatory neurons. Functionally, feedforward inhibition can *anticipate* how excited the excitatory neurons will become, whereas feedback accurately reflects the actual level of activation they achieve.

Feedback inhibition is the most intuitive, so we'll start with it. Here, the inhibitory interneurons are driven by the same excitatory neurons that they then project back to and inhibit. This is the classical "feedback" circuit from the AC example. When a set of excitatory neurons starts to get active, they then communicate this activation to the inhibitory interneurons (via *excitatory glutamatergic* synapses onto inhibitory interneurons – inhibitory neurons have to get excited just like everyone else). This excitation of the inhibitory neurons then causes them to fire action potentials that come right back to the excitatory neurons, opening up their inhibitory ion channels via GABA release. The influx of Cl- (chloride) ions from the inhibitory input channels on these excitatory neurons acts to drive them back down in the direction of the inhibitory driving potential (in the tug-of-war analogy, the inhibitory guy gets bigger and pulls harder). Thus, excitation begets inhibition which counteracts the excitation and keeps everything under control, just like a blast of cold air from the AC unit.

Feedforward inhibition is perhaps a bit more subtle. It operates when the excitatory synaptic inputs to excitatory neurons in a given area also drive the inhibitory interneurons in that area, causing the interneurons to inhibit the excitatory neurons *in proportion to the amount of excitatory input they are currently receiving.* This would be like a thermostat reacting to the anticipated amount of heat, for example, by turning on the AC based on the outside temperature. Thus, the key difference between feedforward and feedback inhibition is that **feedforward reflects the net excitatory input**, whereas **feedback reflects the actual activation output** of a given set of excitatory neurons.

As we will see in the exploration, the anticipatory function of feedforward inhibition is crucial for limiting the kinds of dramatic feedback oscillations that can develop in a purely feedback-driven system. However, too much feedforward inhibition makes the system very slow to respond, so there is an optimal balance of the two types that results in a very robust inhibitory dynamic. Furthermore, the way in which inhibition and excitation interact through the tug-of-war dynamic as we saw in the previous chapter is *essential* for enabling these inhibitory dynamics to be as robust as they are. For example, the shunting nature of inhibition, which only starts to resist once the membrane potential starts to rise, enables the neurons to get some level of activity and then get pulled back down – an alternative form of inhibition (e.g., simply subtracting away from excitation) would either prevent activation entirely or not generate enough inhibition to control the excitation.

Exploration of Inhibitory Interneuron Dynamics

- See the `inhib` simulation in CCN Sims – this simulation shows how feedforward and feedback inhibitory dynamics lead to the robust control of excitatory pyramidal neurons, even in the presence of bidirectional excitation.

FFFB Inhibition Function

We can efficiently implement the feedforward (FF) and feedback (FB) form of inhibition without actually requiring the inhibitory interneurons, by using the average net input and activity levels in a given layer, in a simple equation shown below. This works surprisingly well, without requiring subsequent parameter adaptation during learning, and this **FFFB** form of inhibition is now the default, replacing the *k-Winners-Take-All* (kWTA) form of inhibition used in the 1st Edition of the textbook (O'Reilly and Munakata 2000).

The average excitatory net input to a layer (or group of units within a layer, if inhibition is operating at that level) is just the average of the net input (η_i) of each unit in the layer / group:

$$\eta = \sum_n \frac{1}{n} \eta_i$$

Similarly, the average activation is just the average of the activation values (y_i):

$$\langle y \rangle = \sum_n \frac{1}{n} y_i$$

We compute the overall inhibitory conductance applied uniformly to all the units in the layer / group with just a few key parameters applied to each of these two averages. Because the feedback component tends to drive oscillations (alternately over and under reacting to the average activation), we apply a simple time integration dynamic on that term. The feedforward does not require this time integration, but it does require an offset term, which was determined by fitting the actual inhibition generated by our earlier kWTA equations. Thus, the overall inhibitory conductance is just the sum of the two terms (ff and fb), with an overall inhibitory gain factor **gi**:

$$g_i(t) = \text{gi}\,[\text{ff}(t) + \text{fb}(t)]$$

This gi factor is typically the only parameter manipulated to determine how active overall a layer is. Typically a value of 1.5 is as low as is used, to give a more widely distributed activation pattern, with values around 2.0 (often 2.1 or 2.2 works best) being very typical. For very sparse layers (e.g., a single output unit active), values up to around 3.5 or so can be used.

The feedforward (ff) term is:

$$\text{ff}(t) = \text{ff}\,[\langle\eta\rangle - \text{ff0}]_+$$

where **ff** is the overall gain factor for the feedforward component (set to 1.0 by default), and **ff0** is an offset (set to 0.1 by default) that is subtracted from the average netinput value $\langle\eta\rangle$.

The feedback (fb) term is:

$$\text{fb}(t) = \text{fb}(t-1) + dt\,[\text{fb}\langle y\rangle - \text{fb}(t-1)]$$

where **fb** is the overall gain factor for the feedback component (0.5 default), **dt** is the time constant for integrating the feedback inhibition (0.7 default), and the t-1 indicates the previous value of the feedback inhibition – this equation specifies a graded folding-in of the new inhibition factor on top of what was there before, and the relatively fast dt value of 0.7 makes it track the new value fairly quickly – there is just enough lag to iron out the oscillations.

Overall, it should be clear that this FFFB inhibition is extremely simple to compute (much simpler than the previous kWTA computation), and it behaves in a much more *proportional* manner relative to the excitatory drive on the units – if there is higher overall excitatory input, then the average activation overall in the layer will be higher, and vice-versa. The previous kWTA-based computation tended to be more rigid and imposed a stronger set-point like behavior. The FFFB dynamics, being much more closely tied to the way inhibitory interneurons actually function, should provide a more biologically accurate simulation.

Exploration of FFFB Inhibition

To see FFFB inhibition in action, you can follow the instructions at the last part of the `inhib` simulation at CCN Sims.

Appendix

The following optional additional topics are covered here:

- **Philosophy of Categories:** philosophical issues about the truth value of mental categories.
- **Energy and Harmony:** mathematics of attractor dynamics in terms of Hopfield energy or Smolensky's Harmony.

Philosophy of Categories

This section delves a bit more into the philosophical issues associated with mental categories, and their apparent lack of obvious "truth" value, and what the implications of this might be.

IMPORTANT DISCLAIMER: this will probably be an interesting topic for various folks and certainly a lot has been written on this topic in the philosophical literature. At this point, however, the views represented here are those of the first author and perhaps a few other co-authors.

- As we noted in the main chapter, it seems that mental categories are shaped by learning, and social interaction via language etc, and that there is likely some kind of underlying regularity that drives our ability to form stable internal category representations, but really, they are not "grounded" in any solid kind of "reality".
- This accords with many facts about human cognition: it is highly fallible, people believe all manner of completely wrong things all the time (and often hold these beliefs extremely dearly..), etc.
- But it is somewhat unsettling to embrace this view, as it seems to put one square in the full "cultural relativism" camp, with no hope of ever having any sense "universal truth". This makes objectivists puke, and is generally not great for scientists, who seek to discover the "true nature of the world".
- However, there is a very good solution to this problem, even though it is in no way "absolute" and certainly takes a lot of time and patience (and cooperation among individuals). It also happens to be the bedrock of science. This solution is to develop an ever-broader **self-consistent** set of mental categories, based on **replicable** experiences that can be **shared across individuals**. In short, any given mental category you might happen to develop has a good chance of being wrong, but if you and a group of other people can all agree on a very reliable set of basic experiences and ways of categorizing those that is self-consistent over time and across the whole set of such categories, then it seems quite likely that these are "true".
- In scientific terms, the "experiences" are **experiments** that can be replicated across different labs. And the mental categories are **scientific theories** which have to be consistent not only with a given set of experiments, but also with each other, and all the other experiments that support other such theories.
- At this point in time, there is a collective understanding in science that encompasses a great deal of phenomena in the natural world, e.g., the "standard model" in physics, and all of chemistry, biology, molecular genetics, etc. Higher-level more complex phenomena such as human cognition and neuroscience have a lot more unresolved issues, but progress is being made and people would probably be surprised about how many important things for which there really is a strong overall consensus. But of course, no one individual knows all this stuff. But anyway, it is there for the knowing, and seems to constitute the closest approximation to the truth that we're going to get.
- Short answer: if you want to find the truth, become a scientist! If not, be content to just make stuff up. The brain is very good at it, and it might serve you just fine.. If you don't want to go all the way to being a scientist, you can also try to just think about your different mental categories (beliefs) and see which ones seem consistent with each other and which ones don't. Then, try to resolve the inconsistencies, in a way that best matches your actual physical experiences in the real world. In so doing, you will likely improve the quality of your mental categories, making them closer approximations to some kind of underlying truth!

Energy and Harmony

This section describes the (Hopfield 1982, 1984) energy and Smolensky Harmony equations, and how they help us understand more formally what our networks are doing as they settle into an attractor state.

The Hopfield energy equation is:

$$E = -\frac{1}{2}\sum_j \sum_i x_i w_{ij} y_j$$

where x and y represent the sending and receiving unit activations (indexed by i and j), respectively, and w is the weight between them.

Harmony is literally the same thing without the minus sign:

$$H = \frac{1}{2}\sum_j \sum_i x_i w_{ij} y_j$$

You can see that Harmony is maximized to the extent that, for each pair of sending and receiving units, the activations of these units x_i and y_j are consistent with the weight between these two units. If the weight

is large and positive, the network is configured such that it is harmonious if these two units are both active together. If the weight is negative (a simple version of inhibitory projections), then those units contribute to greater harmony only if they have opposite sign (one is active and the other not active).

A key feature of these equations is that *local updates drive reliable global effects* on energy or Harmony (decreasing the energy or increasing Harmony). To see this, we can use the mathematics of calculus to take the derivative of the global equation with respect to changes in the receiving unit's activation:

$$\frac{\partial H}{\partial y_j} = \sum_i x_i w_{ij}$$

Taking the derivative allows us to find the maximum of a function, which occurs when the derivative is zero. So, this gives us a prescriptive formula for deciding how y_j should be changed (updated) as a function of inputs and weights so as to maximize Harmony. You might recognize this equation as essentially the net excitatory conductance or *net input* to a neuron, from the *Neuron* Chapter. This means that updating units with a *linear* activation function (where activation y = net input directly) would serve to maximize Harmony or minimize energy. To accommodate a non-linear activation function (e.g., a "sigmoidal" function of the same general shape as the XX1 function), one needs to introduce and additional "penalty" term (called *entropy* in the Hopfield framework, and *stress* in the Smolensky one), that essentially drives the saturation of the neural activation function for high or low values of net input.

Chapter 4: Learning

How do we learn to read, do math, and play sports? Learning in a neural network amounts to the **modification of synaptic weights**, in response to the local activity patterns of the sending and receiving neurons. As emphasized in previous chapters, these synaptic weights are what determine what an individual neuron detects, and thus are *the* critical parameters for determining neuron and network behavior.

In other words, *everything you know is encoded in the patterns of your synaptic weights*, and these have been shaped by every experience you've had (as long as those experiences got your neurons sufficiently active). Many of those experiences don't leave a very strong mark, and in much of the brain, traces of individual experiences are all blended together, so it is difficult to remember them distinctly (we'll see in the Memory Chapter that this blending can be quite beneficial for overall intelligence, actually). But each experience nevertheless drives some level of learning, and our big challenge in this chapter is to figure out how the mere influences of patterns of activity among individual neurons can add up to enable us to learn big things.

Biologically, **synaptic plasticity** (the modification of synaptic weights through learning) has been extensively studied, and we now know a tremendous amount about the detailed chemical processes that take place as a result of neural activity. We'll provide multiple levels of detail here (including a discussion of **spike timing dependent plasticity (STDP)**, which has captured the imaginations of many researchers in this area), but the high-level story is fairly straightforward: the overall level of neural activity on both ends of the synapse (sending and receiving neural firing) drives the influx of calcium ions (Ca^{++}) via NMDA channels, and synaptic weight changes are driven by the **level of postsynaptic** Ca^{++} in the *dendritic spine* associated with a given synapse. Low levels of Ca^{++} cause synapses to get weaker, and higher levels cause them to get stronger.

Computationally, many different sets of equations have been developed that can drive synaptic weight changes to accomplish many different computational goals. Which of these correspond to what the biology is actually doing? That is the big question. While a definitive answer remains elusive, we nevertheless have a reasonable candidate that aligns well with the biological data, and also performs computationally very useful forms of learning, which can solve the most challenging of cognitive tasks (e.g., learning to read or recognize objects).

There are two primary types of learning:

- **Self-organizing** learning, which extracts longer time-scale statistics about the environment, and can thus be useful for developing an effective **internal model** of the outside world (i.e., what kinds of things tend to reliably happen in the world – we call these **statistical regularities**).
- **Error-driven** learning, which uses more **rapid contrasts between expectations and outcomes** to correct these expectations, and thus form more detailed, specific knowledge about contingencies in the world. For example, young children seem endlessly fascinated learning about what happens when they push stuff off their high chair trays: will it still fall to the ground and make a huge mess *this* time? Once they develop a sufficiently accurate expectation about exactly what will happen, it starts to get a bit less interesting, and other more unpredictable things start to capture their interest. As we can see in this example, error-driven learning is likely intimately tied up with curiosity, surprise, and other such motivational factors. For this reason, we hypothesize that neuromodulators such as **dopamine**, **norepinephrine** and **acetylcholine** likely play an important role in modulating this form of learning, as they have been implicated in different versions of surprise, that is, when there is a discrepancy between expectations and outcomes.

Interestingly, the main computational difference between these two forms of learning has to do with the time scale over which one of the critical variables is updated – self-organizing learning involves averaging over a long time scale, whereas error-driven learning is much quicker. This difference is emphasized in the above descriptions as well, and provides an important source of intuition about the differences between these types of learning. Self-organizing learning is what happens when you blur your eyes and just take stuff in over a period of time, whereas error-driven learning requires much more alert and rapid forms of neural activity. In the framework that we will use in the rest of the book, we combine these types of learning into a single set of learning equations, to explore how we come to perceive, remember, read, and plan.

Biology of Synaptic Plasticity

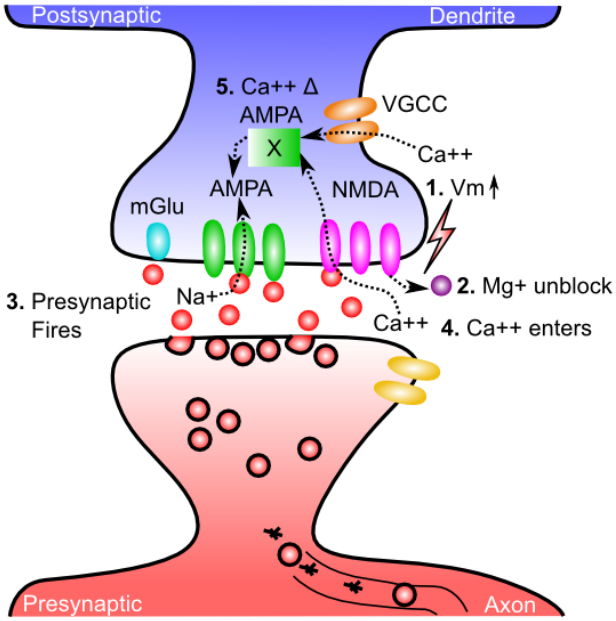


Figure 4.1: Critical steps in allowing calcium ions (Ca^{++}) to enter postsynaptic cell via NMDA channels, inducing synaptic plasticity. 1. The postsynaptic membrane potential (Vm) must be elevated (from collective excitatory synaptic inputs to existing AMPA receptors, and backpropagating action potential that comes back down the dendrite when the postsynaptic neuron fires). 2. Elevated Vm causes magnesium (Mg+) ions to be expelled from NMDA channel openings, thus unblocking them. 3. Presynaptic neuron fires an action potential, releasing glutamate. 4. Glutamate binds to NMDA receptors, causing them to open, allowing Ca^{++} to enter (only when also unblocked, per step 2). 5. The concentration of Ca^{++} in the postsynaptic spine drives second messenger systems (indicated by the X) that result in change in AMPA receptor efficacy, thereby changing the synaptic weight. Ca^{++} can also enter from voltage-gated calcium channels (VGCC's), which depend only on postsynaptic Vm levels, and not sending activity – these are weaker contributors to Ca^{++} levels.

Learning amounts to changing the overall synaptic efficacy of the synapse connecting two neurons. The synapse has a lot of moving parts (see the *Neuron* Chapter), any one of which could potentially be the critical factor in causing its overall efficacy to change. How many can you think of? The search for the critical factor(s) dominated the early phase of research on synaptic plasticity, and evidence for the involvement of a range of different factors has been found over the years, from the amount of presynaptic neurotransmitter released, to number and efficacy of postsynaptic AMPA receptors, and even more subtle things such as the alignment of pre and postsynaptic components, and more dramatic changes such as the cloning of multiple synapses. However, the dominant factor for long-lasting learning changes appears to be the number and efficacy of postsynaptic AMPA receptors.

Figure 4.1 shows the five critical steps in the cascade of events that drives change in AMPA receptor efficacy. The **NMDA** receptors and the calcium ion (Ca^{++}) play a central role – NMDA channels allow Ca^{++} to enter the postsynaptic spine. Across all cells in the body, Ca^{++} typically plays an important role in regulating cellular function, and in the neuron, it is capable of setting off a series of chemical reactions that ends up controlling how many AMPA receptors are functional in the synapse. For details on these reactions, see Chapter Appendix on *Detailed Biology of Learning*. Here's what it takes for the Ca^{++} to get into the postsynaptic cell:

1. The postsynaptic membrane potential (Vm) must be elevated, as a result of all the excitatory synaptic inputs coming into the cell. The most important contributor to this Vm level is actually the **back-propagating action potential** – when a neuron fires an action potential, it not only goes forward out the axon, but also backward down the dendrites (via active voltage-sensitive Na+ channels along the dendrites). Thus, the entire neuron gets to know when it fires – we'll see that this is incredibly

useful computationally.

2. The elevated Vm causes magnesium ions (Mg+) to be repelled (positive charges repel each other) out of the openings of NMDA channels, unblocking them.
3. The presynaptic neuron fires an action potential, releasing glutamate neurotransmitter into the synaptic cleft.
4. Glutamate binds to the NMDA receptor, opening it to allow Ca^{++} ions to flow into the postsynaptic cell. This only occurs if the NMDA is also unblocked. This dependence of NMDA on both pre and postsynaptic activity was one of the early important clues to the nature of learning, as we see later.
5. The concentration of Ca^{++} in the postsynaptic spine drives those complex chemical reactions that end up changing the number and efficacy of AMPA receptors. Because these AMPA receptors provide the primary excitatory input drive on the neuron, changing them changes the net excitatory effect of a presynaptic action potential on the postsynaptic neuron. This is what is meant by changing the synaptic efficacy, or *weight*.

Ca^{++} can also enter the postsynaptic cell via **voltage gated calcium channels (VGCC)'s** which are calcium channels that only open when the membrane potential is elevated. Unlike NMDA, however, they are *not* sensitive to presynaptic neural activity – they only depend on postsynaptic activity. This has important computational implications, as we discuss later. VGCCs contribute less to Ca^{++} levels than NMDA, so NMDA is still the dominant player.

Metabotropic glutamate receptors (mGlu) also play an important role in synaptic plasticity. These receptors do not allow ions to flow across the membrane (i.e., they are not *ionotropic*), and instead they directly trigger chemical reactions when neurotransmitter binds to them. These chemical reactions can then modulate the changes in AMPA receptors triggered by Ca^{++}.

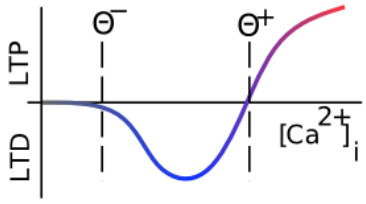


Figure 4.2: Direction of synaptic plasticity (LTP = increase, LTD = decrease) as a function of Ca^{++} concentration in the postsynaptic spine (accumulated over several 100 milliseconds). Low levels of Ca^{++} cause LTD, while higher levels drive LTP. Threshold levels indicated by theta values represent levels where the function changes sign.

We have been talking about changes in AMPA receptor efficacy without specifying which direction they change. **Long Term Potentiation (LTP)** is the biological term for long-lasting *increases* in AMPA efficacy, and **Long Term Depression (LTD)** means long-lasting *decreases* in AMPA efficacy. For a long time, researchers focused mainly on LTP (which is generally easier to induce), but eventually they realized that both directions of synaptic plasticity are equally important for learning. Figure 4.2 shows how this direction of change depends on the overall level of Ca^{++} in the postsynaptic spine (accumulated over a few 100's of milliseconds at least – the relevant time constants for effects of Ca^{++} on synaptic plasticity are fairly slow) – low levels drive LTD, while high levels produce LTP. This property will be critical for our computational model. Note that the delay in synaptic plasticity effects based on Ca^{++} levels means that the synapse doesn't always have to do LTD on its way up to LTP – there is time for the Ca^{++} to reach a high level to drive LTP before the weights start to change.

Hebbian Learning and NMDA Channels

The famous Canadian psychologist Donald O. Hebb predicted the nature of the NMDA channel many years in advance of its discovery, just by thinking about how learning should work at a functional level. Here is a key quote:

> *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.

This can be more concisely summarized as *cells that fire together, wire together*. The NMDA channel is essential for this process, because it requires both pre and postsynaptic activity to allow Ca^{++} to enter and drive learning. It can detect the *coincidence* of neural firing. Interestingly, Hebb is reputed to have said something to the effect of "big deal, I knew it had to be that way already" when someone told him that his learning principle had been discovered in the form of the NMDA receptor.

Mathematically, we can summarize Hebbian learning as:

$$\Delta w = xy$$

where Δw is the change in synaptic weight w, as a function of sending activity x and receiving activity y.

Anytime you see this kind of pre-post product in a learning rule, it tends to be described as a form of Hebbian learning. For a more detailed treatment of Hebbian learning and various popular variants of it, see the *Hebbian Learning* Appendix.

As we'll elaborate below, this most basic form of Hebbian learning is very limited, because weights will only go up (given that neural activities are rates of spiking and thus only positive quantities), and will do so without bound. Interestingly, Hebb himself only seemed to have contemplated LTP, not LTD, so perhaps this is fitting. But it won't do anything useful in a computational model. Before we get to the computational side of things, we cover one more important result in the biology.

Spike Timing Dependent Plasticity

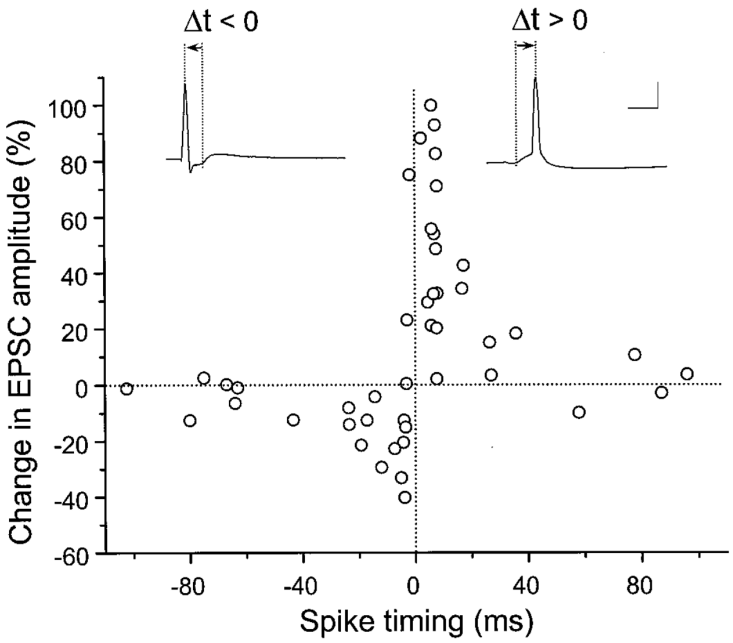


Figure 4.3: Spike timing dependent plasticity demonstrated in terms of temporal offset of firing of pre and postsynaptic neurons. If post fires after pre ($\Delta t > 0$, right side), then the weights go up, and otherwise ($\Delta t < 0$, left side) they go down. This fits with a causal flow of information from pre to post. However, more complex sequences of spikes wash out such precise timing and result in more generic forms of Hebbian-like learning. Reproduced from Bi and Poo (1998).

Figure 4.3 shows the results from an experiment (Bi and Poo 1998) that captured the imagination of many a scientist, and has resulted in extensive computational modeling work. This experiment showed that the precise order of firing between a pre and postsynaptic neuron determined the sign of synaptic plasticity,

with LTP resulting when the presynaptic neuron fired before the postsynaptic one, while LTD resulted otherwise. This **spike timing dependent plasticity (STDP)** was so exciting because it fits with the *causal* role of the presynaptic neuron in driving the postsynaptic one. If a given pre neuron actually played a role in driving the post neuron to fire, then it will necessarily have to have fired in advance of it, and according to the STDP results, its weights will increase in strength. Meanwhile, pre neurons that have no causal role in firing the postsynaptic cell will have their weights decreased. However, this STDP pattern does not generalize well to realistic spike trains, where neurons are constantly firing and interacting with each other over 100's of milliseconds (Shouval, Wang, and Wittenberg 2010). Nevertheless, the STDP data does provide a useful stringent test for computational models of synaptic plasticity. We base our learning equations on a detailed model using more basic, biologically-grounded synaptic plasticity mechanisms that does capture these STDP findings (Urakubo et al. 2008), but which nevertheless result in quite simple learning equations when considered at the level of firing rate.

The eXtended Contrastive Attractor Learning (XCAL) Model

The learning function we adopt for the models in the rest of this text is called the **eXtended Contrastive Attractor Learning (XCAL)** rule. (The basis for this naming will become clear later). This learning function was derived through a convergence of bottom-up (motivated by detailed biological considerations) and top-down (motivated by computational desiderata) approaches. In the bottom-up derivation, we extracted an empirical learning function (called the **XCAL dWt function**) from a highly biologically detailed computational model of the known synaptic plasticity mechanisms, by (Urakubo et al. 2008) (see Chapter Appendix on *Detailed Biology of Learning* for more details). Their model builds in detailed chemical rate parameters and diffusion constants, etc, based on empirical measurements, for all of the major biological processes involved in synaptic plasticity. We capture much of the incredible complexity of the model (and by extension, hopefully, the complexity of the actual synaptic plasticity mechanisms in the brain) using a simple piecewise-linear function, shown below, that emerges from it. This XCAL dWt function closely resembles the function shown in Figure 4.2, plotting the dependence of synaptic plasticity on Ca^{++} levels. It also closely resembles the (**BCM**) learning function.

The top-down approach leverages the key idea behind the BCM learning function, which is the use of a **floating threshold** for determining the amount of activity needed to elicit LTP vs LTD (see). Specifically, the threshold is not fixed at a particular value, but instead adjusts as a function of average activity levels of the postsynaptic neuron in question over a long time frame, resulting in a **homeostatic** dynamic. Neurons that have been relatively inactive can more easily increase their synaptic weights at lower activity levels, and can thus "get back in the game". Conversely, neurons that have been relatively overactive are more likely to decrease their synaptic weights, and "stop hogging everything".

As we'll see below, this function contributes to useful **self-organizing** learning, where different neurons come to extract distinct aspects of statistical structure in a given environment. But purely self-organizing mechanisms are strongly limited in what they can learn – they are driven by statistical generalities (e.g., animals tend to have four legs), and are incapable of adapting more pragmatically to the functional demands that the organism faces. For example, some objects are more important to recognize than others (e.g., friends and foes are important, random plants or pieces of trash or debris, not so much).

To achieve these more pragmatic goals, we need **error-driven** learning, where learning is focused specifically on correcting errors, not just categorizing statistical patterns. Fortunately, we can use the same floating threshold mechanism to achieve error-driven learning within the same overall mathematical framework, by adapting the threshold on a faster time scale. In this case, weights are increased if activity states are greater than their very recent levels, and conversely, weights decrease if the activity levels go down relative to prior states. Thus, we can think of the recent activity levels (the threshold) as reflecting **expectations** which are subsequently compared to actual **outcomes**, with the difference (or "error") driving learning. Because both forms of learning (self-organizing and error-driven) are quite useful, and use the exact same mathematical framework, we integrate them both into a single set of equations with two thresholds reflecting integrated activity levels across different time scales (recent and long-term average).

Next, we describe the XCAL dWt function (dWt = change in weight), before describing how it captures both forms of learning, followed by their integration into a single unified framework (including the promised

explanation for its name!).

The XCAL dWt Function

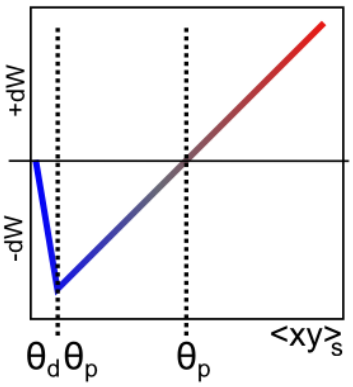


Figure 4.4: The XCAL dWt function, showing direction and magnitude of synaptic weight changes (dWt) as a function of the short-term average activity of the sending neuron (x) times the receiving neuron (y). This quantity is a simple mathematical approximation to the level of postsynaptic Ca^{++}, reflecting the dependence of the NMDA channel on both sending and receiving neural activity. This function was extracted directly from the detailed biophysical Urakubo, Honda, Froemke, & Kuroda (2008) model, by fitting a piecewise linear function to the synaptic weight change behavior that emerges from it as a function of a wide range of sending and receiving spiking patterns.

The XCAL dWt function extracted from the (Urakubo et al. 2008) model is shown in Figure 4.4. First, the main input into this function is the **total synaptic activity** reflecting the firing rate and duration of activity of the sending and receiving neurons. In mathematical terms for a rate-code model with sending activity rate x and receiving activity rate y, this would just be the "Hebbian" product we described above:

$$\Delta w = f_{xcal}\left(xy, \theta_p\right)$$

where f_{xcal} is the piecewise linear function shown in Figure 4.4. The weight change also depends on an additional dynamic threshold parameter θ_p, which determines the point at which it crosses over from negative to positive weight changes – i.e., the point at which weight changes reverse sign. For completeness, here is the mathematical expression of this function, but you only need to understand its shape as shown in the figure:

$$f_{xcal}(xy, \theta_p) = \begin{array}{l} (xy \ - \ \theta_p) \ \text{ if } \ xy \ > \ \theta_p \ \theta_d \\ -xy \ (1 \ - \ \theta_d) \ / \ \theta_d \ \ \text{otherwise} \end{array}$$

where $\theta_d = .1$ is a constant that determines the point where the function reverses direction (i.e., back toward zero within the weight decrease regime) – this reversal point occurs at $\theta_p \theta_d$, so that it adapts according to the dynamic θ_p value.

As noted in the previous section, the dependence of the NMDA channel on activity of both sending and receiving neurons can be summarized with this simple Hebbian product, and the level of intracellular Ca^{++} is likely to reflect this value. Thus, the XCAL dWt function makes very good sense in these terms: it reflects the qualitative nature of weight changes as a function of Ca^{++} that has been established from empirical studies and postulated by other theoretical models for a long time. The Urakubo model simulates detailed effects of pre/postsynaptic spike timing on Ca^{++} levels and associated LTP/LTD, but what emerges from these effects at the level of firing rates is this much simpler fundamental function.

As a learning function, this basic XCAL dWt function has some advantages over a plain Hebbian function, while sharing its basic nature due to the "pre * post" term at its core. For example, because of the shape of the dWt function, weights will go down as well as up, whereas the Hebbian function only causes weights to increase. But it still has the problem that weights will increase without bound (as long as activity levels are often greater than the threshold). We'll see in the next section that some other top-down

computationally-motivated modifications can result in a more powerful form of learning while maintaining this basic form.

Self-Organizing Learning: Long Time Scales and the BCM Model

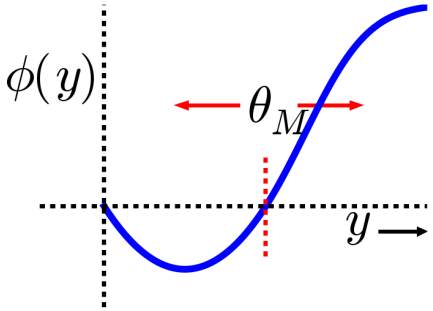


Figure 4.5: The shape of the BCM learning function. Note the similarity in qualitative shape to both the XCAL dWt function (Figure 4.4) and synaptic plasticity as function of Ca^{++} (Figure 4.2).

The major computational motivation comes from a line of learning functions that began with (Bienenstock, Cooper, and Munro 1982), with these initials giving rise to the name of the function: **BCM**. (Interestingly Leon Cooper, a Nobel Laurate in Physics, was also "central" in the BCS theory of superconductivity). The BCM function is a modified form of Hebbian learning, which includes an interesting **homeostatic** mechanism that keeps individual neurons from firing too much or too little over time:

$$\Delta w = xy\,(y - \theta)$$

where again x = sending activity, y = receiving activity, and θ is a **floating threshold** reflecting a **long time average** of the receiving neuron's activity:

$$\theta = \langle y^2 \rangle$$

where $\langle\rangle$ indicates the expected value or average, in this case of the square of the receiving neuron's activation.

Figure 4.5 shows what this function looks like – a shape that should be becoming rather familiar. Indeed, the fact that the BCM learning function anticipated the qualitative nature of synaptic plasticity as a function of Ca^{++} (Figure 4.2) is an amazing instance of theoretical prescience. Furthermore, BCM researchers have shown that it does a good job of accounting for various behavioral learning phenomena, providing a better fit than a comparable Hebbian learning mechanism (Cooper et al. 2004; Kirkwood, Rioult, and Bear 1996) (Figure 4.6).

BCM has typically been applied in simple feedforward networks in which, given an input pattern, there is only one activation value for each neuron. But how should weights be updated in a more realistic bidirectionally connected system with attractor dynamics in which activity states continuously evolve through time? We confront this issue in the XCAL version of the BCM equations:

$$\Delta w = f_{xcal}(xy, \langle y \rangle_l) = f_{xcal}(xy, y_l)$$

where xy is understood to be the **short-term average synaptic activity** (on a time scale of a few hundred milliseconds – the time scale of Ca^{++} accumulation that drives synaptic plasticity), which could be more formally expressed as: $\langle xy \rangle_s$, and $y_l = \langle y \rangle_l$ is the **long-term average activity of the postsynaptic neuron** (i.e., essentially the same as in BCM, but without the squaring), which plays the role of the θ_p floating threshold value in the XCAL function.

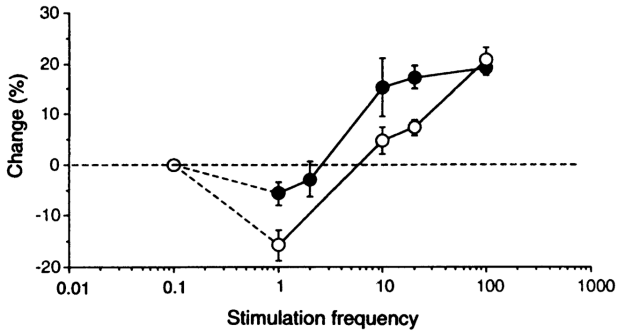


Figure 4.6: Synaptic plasticity data from dark reared (filled circles) and normally-reared (open circles) rats, showing that dark reared rats appear to have a lower threshold for LTP, consistent with the BCM floating threshold. Neurons in these animals are presumably much less active overall, and thus their threshold moves down, making them more likely to exhibit LTP relative to LTD. Reproduced from Kirkwood, Rioult, & Bear (1996).

After considerable experimentation, we have found the following way of computing the y_l floating threshold to provide the best ability to control the threshold and achieve the best overall learning dynamics:

$$\text{if } y > .2 \text{ then } y_l = y_l + \frac{1}{\tau_l}(\max - y_l)$$
$$\text{else } y_l = y_l + \frac{1}{\tau_l}(\min - y_l)$$

This produces a well-controlled exponential approach to either the *max* or *min* extremes depending on whether the receiving unit activity exceeds the basic activity threshold of .2. The time constant for integration τ_l is 10 by default – integrating over around 10 trials. See Chapter Appendix *Leabra Details* for more discussion.

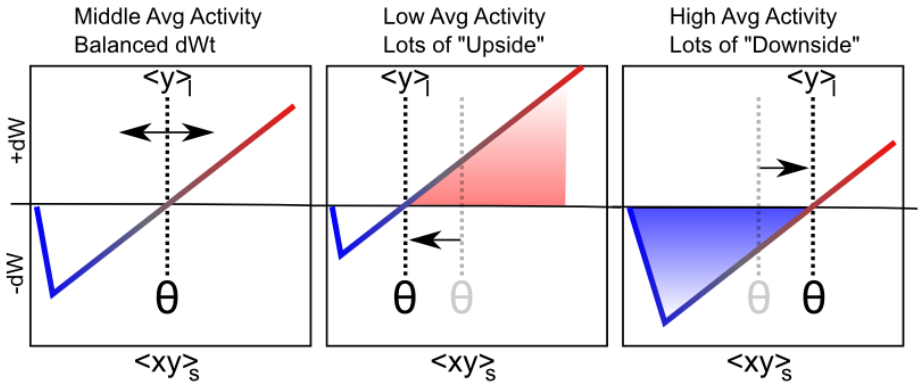


Figure 4.7: How the floating threshold as a function of long-term average receiver neural activity $\langle y \rangle_l$ drives homeostatic behavior. Neurons that have low average activity are much more likely to increase their weights because the threshold is low, while those that have high average activity are much more likely to decrease their weights because the threshold is high.

Figure 4.7 shows the main qualitative behavior of this learning mechanism: when the long term average activity of the receiver is low, the threshold moves down, and thus it is more likely that the short term synaptic activity value will fall into the positive weight change territory. This will tend to increase synaptic weights overall, and thus make the neuron more likely to get active in the future, achieving the homeostatic objective. Conversely, when the long term average activity of the receiver is high, the threshold is also high, and thus the short term synaptic activity is more likely to drive weight decreases than increases. This will

take these over-active neurons down a notch or two, so they don't end up dominating the activity of the network.

Self-organizing Learning Dynamics

This ability to spread the neural activity around in a more equitable fashion turns out to be critical for self-organizing learning, because it enables neurons to more efficiently and effectively cover the space of things to represent. To see why, here are the critical elements of the self-organizing learning dynamic (see subsequent simulation exploration to really get a feel for how this all works in practice):

- **Inhibitory competition** – only the most strongly driven neurons get over the inhibitory threshold, and can get active. These are the ones whose current synaptic weights best fit ("detect") the current input pattern.
- **Rich get richer** positive feedback loop – due to the nature of the learning function, only those neurons that actually get active are capable of learning (when receiver activity y = 0, then xy = 0 too, and the XCAL dWt function is 0 at 0). Thus, the neurons that already detect the current input the best are the ones that get to further strengthen their ability to detect these inputs. This is the essential insight that Hebb had with why the Hebbian learning function should strengthen an "engram".
- **Homeostasis** to balance the positive feedback loop – if left unchecked, the rich-get-richer dynamic ends up with a few units dominating everything, and as a result, all the inputs get categorized into one useless, overly-broad category ("everything"). The homeostatic mechanism in BCM helps fight against this by raising the floating threshold for highly active neurons, causing their weights to decrease for all but their most preferred input patterns, and thus restoring a balance. Similarly, under-active neurons experience net weight increases that get them participating and competing more effectively, and hence they come to represent distinct features.

The net result is the development of a set of neural detectors that relatively evenly cover the space of different inputs patterns, with systematic categories that encompass the statistical regularities. For example, cats like milk, and dogs like bones, and we can learn this just by observing the reliable co-occurrence of cats with milk and dogs with bones. This kind of reliable co-occurrence is what we mean by "statistical regularity". See Chapter Appendix on *Hebbian Learning* for a very simple illustration of why Hebbian-style learning mechanisms capture patterns of co-occurrence. It is really just a variant on the basic maxim that "things that fire together, wire together".

The Learning Rate

There is an important factor missing from the above equations, which is the **learning rate** – we typically use the greek epsilon ϵ to represent this parameter, which simply multiplies the rate with which the weights change:

$$\Delta w = \epsilon f_{xcal}(xy, y_l)$$

Thus, a bigger epsilon means larger weight changes, and thus quicker learning, and vice-versa for a smaller value. A typical starting value for the learning rate is .04, and we often have it decrease over time (which is true of the brain as well – younger brains are much more plastic than older ones) – this typically results in the fastest overall learning and best final performance.

Many researchers (and drug companies) have the potentially dangerous belief that a faster learning rate is better, and various drugs have been developed that effectively increase the learning rate, causing rats to learn some kind of standard task faster than normal, for example. However, we will see in the Memory Chapter that actually a slow learning rate has some very important advantages. Specifically, a slower learning rate enables the system to incorporate *more statistics* into learning – the learning rate determines the effective time window over which experiences are averaged together, and a slower learning rate gives a longer time window, which enables more information to be integrated. Thus, learning can be much smarter with a slower learning rate. But the tradeoff of course is that the results of this smarter learning take that much longer to impact actual behavior. Many have argued that humans are distinctive in our extremely protracted period of developmental learning, so we can learn a lot before we need to start earning a paycheck. This allows us to have a pretty slow learning rate, without too many negative consequences.

Exploration of Self-Organizing Learning

The best way to see this dynamic is via the computational exploration. Open the `self_org` simulation from CCN Sims and follow the directions from there.

Error-Driven Learning

Although self-organizing learning is very useful, we'll see that it is significantly limited in the kinds of things that it can learn. It is great for extracting generalities, but not so great when it comes to learning specific, complicated patterns. To learn these more challenging types of problems, we need error-driven learning. Intuitively, error-driven learning is much more powerful because it drives learning based on *differences*, not *raw signals*. Differences (errors) tell you much more precisely what you need to do to fix a problem. Raw signals (overall patterns of neural activity) are not nearly as informative – it is easy to become overwhelmed by the forest and lose sight of the trees. We'll see more specific examples later, after first figuring out how we can get error-driven learning to work in the first place.

We begin with the simplest form of error-driven learning, the **delta rule**, which was developed by (Widrow and Hoff 1960):

$$\Delta w = x(t - y)$$

which can be directly compared to the simple Hebbian learning rule showed earlier:

$$\Delta w = xy$$

This shows how we're just replacing the y receiver activation with a *difference* (delta) or **error** term $(t - y)$, where t is the *target* activation that *should have been produced* and y is the activation that was *actually* produced. This error value $(t - y)$ is zero if the actual activation equals the target, and furthermore tells you exactly how to adjust the synaptic weights to *fix the error* when it is not equal to the target. If the actual activation y was too low relative to t, then the weights will be increased, and vice-versa if it was too high.

By contrast, the Hebbian learning rule just increases the weights regardless of any particular target state – it simply does not take into account any kind of target, desired behavior (activation). You could replace the y value in the Hebbian rule with the target value t, and that *might* work, but we'll see in a bit that it actually does not in general – you really need to learn as a function of errors, not raw activations.

Interestingly, the delta rule does include the sending unit activation x, much like the Hebbian equation, and this turns out to be essential for allocating the **credit** (or **blame**) for the error on those sending units that are actually active – those are the ones for which a change in synaptic weights will actually result in a different outcome next time, and adjusting weights in proportion to the activity level provides a graded form of this logic (the most active get the most credit or blame). Another more "Hebbian" way of thinking about this is that you're *associating* the sending activation with the error, instead of associating it with the raw output activation. The technical term for this property is **credit assignment** and it is an important feature of learning algorithms more generally.

A Biological Basis for Error-driven Learning: Faster Time-Scale Floating Threshold

Although a seemingly simple step from the Hebbian learning rule, error-driven learning introduces a number of important challenges from a biological perspective, such as: where does the target value come from in the first place, and how can biological synapses compute something like the delta rule? We'll go into greater depth on the nature of the target signal later, but for now, we can think in terms of the neural activation y representing something like a **prediction** or **expectation**, and the target being a actual **outcome** – what actually happens.

In this case, there is a natural time-flow from expectation to outcome, which helps us think more specifically about the second question, by taking advantage of the floating threshold property of BCM-like self-organizing aspect of XCAL learning (Figure 4.8). Specifically, we speed up the time scale for computing the floating threshold (and also have it reflect synaptic activity, not just receiver activity):

$$\Theta_p = \langle xy \rangle_m$$

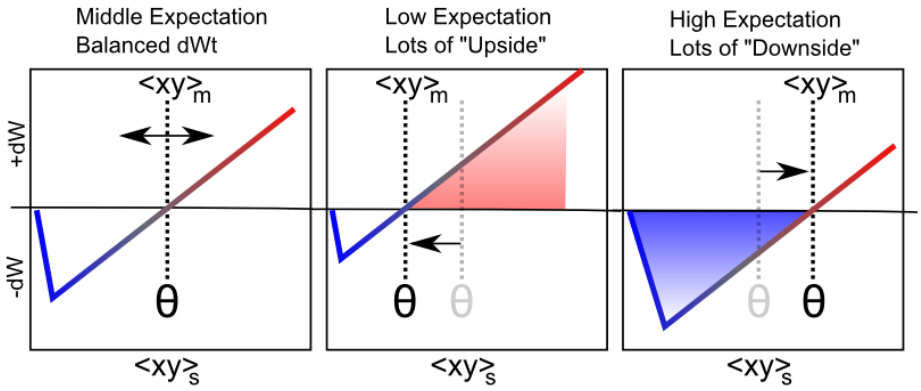


Figure 4.8: How the floating threshold as a function of medium-term average synaptic activity $\langle xy \rangle_m$ can produce error-driven learning. This medium time frame reflects the development of a pattern of neural activity that encodes an expectation about what will happen next. The most recent short term synaptic activity (which drives learning) represents the actual outcome of what did happen next. Because of the (nearly) linear nature of the dWt function, it effectively computes the difference between outcome and expectation. Qualitatively, if the outcome produces greater activation of a population of neurons than did expectation, corresponding weights go up, while neurons that decreased their activity states as a result of the outcome will have their weights go down. This is illustrated above in the case of low vs. high expectations.

$$\begin{aligned} \Delta w &= f_{xcal}(\langle xy \rangle_s, \langle xy \rangle_m) \\ &= f_{xcal}(x_s y_s, x_m y_m) \end{aligned}$$

where $\langle xy \rangle_m$ is the **medium-time scale average synaptic activity**, which we think of as reflecting an emerging expectation about the current situation, which develops over approximately 75 msec of neural activity. The most recent, short-term (last 25 msec) neural activity ($\langle xy \rangle_s$) reflects the actual outcome, and it is the same calcium-based signal that drives learning in the Hebbian case.

In the simulator, the period of time during which this expectation is represented by the network, before it gets to see the outcome, is referred to as the **minus phase**, based on the *Boltzmann machine* terminology (Ackley, Hinton, and Sejnowski 1985). The subsequent period in which the outcome is observed (and the activations evolve to reflect the influence of that outcome) is referred to as the **plus phase**. It is the difference between this expectation and outcome that represents the error signal in error-driven learning (hence the terms minus and plus – the minus phase activations are subtracted from those in the plus phase to drive weight changes).

Although this expectation-outcome comparison is the fundamental requirement for error-driven learning, a weight change based on this difference by itself begs the question of how the neurons would ever 'know' which phase they are in. We have explored many possible answers to this question, and the most recent involves an internally-generated alpha-frequency (10 Hz, 100 msec periods) cycle of expectation followed by outcome, supported by neocortical circuitry in the deep layers and the thalamus (O'Reilly, Wyatte, and Rohrlich 2017; Kachergis et al. 2014). A later revision of this textbook will describe this in more detail. For now, the main implications of this framework are to organize the timing of processing and learning as follows:

- A **Trial** lasts 100 msec (10 Hz, alpha frequency), and comprises one sequence of expectation – outcome learning, organized into 4 quarters.
  - Biologically, the deep neocortical layers (layers 5, 6) and the thalamus have a natural oscillatory rhythm at the alpha frequency (Buffalo et al. 2011; Lorincz et al. 2009; Franceschetti et al. 1995; Luczak, Bartho, and Harris 2013). Specific dynamics in these layers organize the cycle of expectation vs. outcome within the alpha cycle.
- A **Quarter** lasts 25 msec (40 Hz, gamma frequency) – the first 3 quarters (75 msec) form the expectation / minus phase, and the final quarter are the outcome / plus phase.
  - Biologically, the superficial neocortical layers (layers 2, 3) have a gamma frequency oscillation (Buffalo et al. 2011), supporting the quarter-level organization.

- A **Cycle** represents 1 msec of processing, where each neuron updates its membrane potential according to the equations covered in the Neuron Chapter.

The XCAL learning mechanism coordinates with this timing by comparing the most recent synaptic activity (predominantly driven by plus phase / outcome states) to that integrated over the medium-time scale, which effectively includes both minus and plus phases. Because the XCAL learning function is (mostly) linear, the association of the floating threshold with this synaptic activity over the medium time frame (including expectation states), to which the short-term outcome is compared, directly computes their difference:

$$\Delta w \approx x_s y_s - x_m y_m$$

which is very similar to the delta rule shown above, when the sending activation has been multiplied through (and separated into the two different time averages), and y_s represents the outcome target signal t, and y_m represents the actual activation y:

$$\Delta w \approx x(y_s - y_m)$$

Intuitively, this rule behaves like the delta rule, and we can elaborate the error-driven dynamic by considering three different cases. The easiest case is when the expectation is equivalent to the outcome (i.e., a correct expectation) – the two terms above will be the same, and thus their subtraction is zero, and the weights remain the same. So once you obtain perfection, you stop learning. What if your expectation was higher than your outcome? The difference will be a negative number, and the weights will thus decrease, so that you will lower your expectations next time around. Intuitively, this makes perfect sense – if you have an expectation that all movies by M. Night Shyamalan are going to be as cool as *The Sixth Sense*, you might end up having to reduce your weights to better align with actual outcomes. Conversely, if the expectation is lower than the outcome, the weight change will be positive, and thus increase the expectation.

You might have thought this class was going to be deadly boring, but maybe you were amused by the above mention of M. Night Shyamalan, and now you'll have to increase your weights just a bit. It should hopefully be intuitively clear that this form of learning will work to minimize the differences between expectations and outcomes over time. Note that while the example given here was cast in terms of deviations from expectations having value (i.e., things turned out better or worse than expected, as we cover in more detail in the Motor control and Reinforcement Learning Chapter), the same principle applies when outcomes deviate from other sorts of expectations.

Because of its explicitly temporal nature, there are a few other interesting ways of thinking about what this learning rule does, in addition to the explicit timing defined above. To reiterate, the rule says that the outcome comes *immediately after* a preceding expectation – this is a direct consequence of making it learn toward the short-term (most immediate) average synaptic activity, compared to a slightly longer medium-term average that includes the time just before the immediate present.

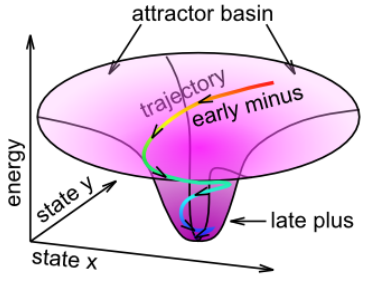


Figure 4.9: Illustration of Contrastive Attractor Learning (CAL) principle, which is core idea behind XCAL error-driven learning mechanism. The network learns on the contrast between the early phase of settling (the minus phase, or medium time frame activation average $\langle xy \rangle_m$) versus the late phase of settling (the plus phase or short time frame activation average $\langle xy \rangle_s$). The late phase has integrated more of the overall constraints in the network and thus represents a "better" overall interpretation or representation of the current situation than the early phase, so it makes sense for the late phase to serve as the "training signal" relative to the earlier phase.

We can think of this learning in terms of the attractor dynamics discussed in the Networks Chapter. Specifically, the name **Contrastive Attractor Learning (CAL)** reflects the idea that the network is settling into an attractor state, and it is the contrast between the final attractor state that the network settles into (i.e., the "outcome" in this case), versus the network's activation trajectory as it approaches the attractor, that drives learning (Figure 4.9). The short-time scale average reflects the final attractor state (the 'target'), and the medium time-scale average reflects the entire trajectory during settling. When the pattern of activity associated with the expectation is far from the actual outcome, the difference between these two attractor states will be large, and learning will drive weight changes so that in future encounters, the expectation will more closely reflect the outcome (assuming the environment is reliable). The **X** part of XCAL simply reflects the fact that the same objective is achieved without having to explicitly compare two attractors at discrete points in time, but instead by using a time-averaged activity state eXtended across the entire settling trajectory as the baseline comparison, which is more biologically realistic because such variables are readily accessible by local neuronal activity.

Mathematically, this CAL learning rule represents a simpler version of the oscillating learning function developed by Norman and colleagues (Norman et al. 2006; Ritvo, Turk-Browne, and Norman 2019).

There are also more general reasons for **later information** (short time scale average) to train **earlier information** (medium time scale average). Typically, the longer one waits, the better quality the information is – at the start of a sentence, you might have some idea about what is coming next, but as it unfolds, the meaning becomes clearer and clearer. This later information can serve to train up the earlier expectations, so that you can more efficiently understand things next time around. Overall, these alternative ways of thinking about XCAL learning represent more self-organizing forms of learning without requiring an explicit outcome training signal, while using the more rapid contrast (short vs. medium time) for the error-driven learning mechanism.

Before continuing, you might be wondering about the biological basis of this error-driven form of the floating threshold. Unlike the BCM-style floating threshold, which has solid empirical data consistent with it, the idea that the threshold changes on this quicker time scale to reflect the medium time-scale average synaptic activity has not yet been tested empirically. Thus, it stands as an important prediction of this computational model. Because it is so easily computed, and results in such a powerful form of learning, it seems plausible that the brain would take advantage of just such a mechanism, but we'll have to see how it stands up to empirical testing. One initial suggestion of such a dynamic comes from this paper: (Lim et al. 2015), which showed a BCM-like learning dynamic with rapid changes in the threshold depending on recent activity. Also, there *is* substantial evidence that transient changes in neuromodulation that occur during salient, unexpected events, are important for modifying synaptic plasticity – and may functionally contribute to this type of error-driven learning mechanism. Also, we discuss a little bit later another larger concern about the nature and origin of the expectation vs. outcome distinction, which is central to this form of error-driven learning.

Advantages of Error-Driven Learning

As noted above, error-driven learning is much more computationally powerful than self-organizing learning. For example, all computational models that perform well at the difficult challenge of learning to recognize objects based on their visual appearance (see the Perception Chapter) utilize a form of error-driven learning. Some also use various forms of self-organizing learning, but this tends to play more of a supporting role, whereas the models would be entirely non-functional without error-driven learning. Error-driven learning ensures that the model makes the kinds of categorical discriminations that are relevant, while avoiding those that are irrelevant. For example, whether a side view of a car is facing left or right is not relevant for determining that this is a car. But the presence of wheels is very important for discriminating a car from a fish. A purely self-organizing model has no way of knowing that these differences, which may be quite statistically reliable and strong signals in the input, differ in their utility for the categories that people care about.

Mathematically, the history of error-driven learning functions provides a fascinating window into the sociology of science, and how seemingly simple ideas can take a while to develop. In the Chapter Appendix on *Backpropagation*, we trace this history through the derivation of error-driven learning rules, from the

delta rule (Widrow and Hoff 1960), to the very widely used **backpropagation** learning rule (Rumelhart, Hinton, and Williams 1986). At the start of that subsection, we show how the XCAL form of error-driven learning (specifically the CAL version of it) can be derived directly from backpropagation, thus providing a mathematically satisfying account as to why it is capable of solving so many difficult problems.

The key idea behind the backpropagation learning function is that error signals arising in an output layer can *propagate backward* down to earlier hidden layers to drive learning in these earlier layers so that it will solve the overall problem facing the network (i.e., it will ensure that the network can produce the correct expectations or answers on the output layer). This is essential for enabling the system as a whole to solve difficult problems – as we discussed in the Networks Chapter, a lot of intelligence arises from multiple layers of cascading steps of categorization – to get all of these intervening steps to focus on the relevant categories, error signals need to propagate across these layers and shape learning in all of them.

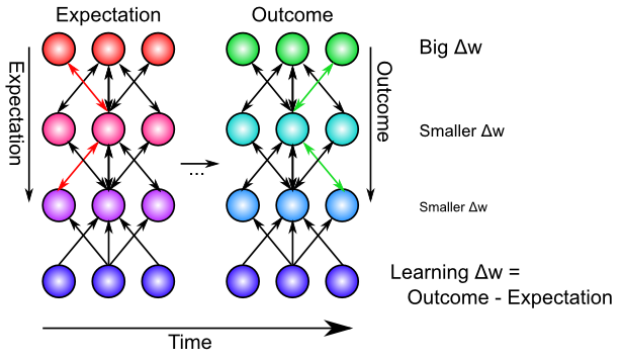


Figure 4.10: Intuition for how bidirectional connections enable the backpropagation of learning signals from other parts of the network – when there is a difference between an expectation and an outcome in any part of the network, neurons in other parts of the network "feel" that difference via the bidirectional connection. All neurons experience an impact on their own activation of both the expectation and the outcome, and thus when they learn on the difference between these two points in time (later training earlier), they are learning about their own impact on the outcome - expectation error, and weight changes based on this difference will end up minimizing the overall error in the network as a whole. Neurons closer to the source of the error learn the most, with error decreasing with distance from this source.

Biologically, the bidirectional connectivity in our models enables these error signals to propagate in this manner (Figure 4.10). Thus, changes in any given location in the network radiate backward (and every which way the connections go) to affect activation states in all other layers, via bidirectional connectivity, and this then influences the learning in these other layers. In other words, XCAL uses bidirectional activation dynamics to communicate error signals throughout the network, whereas backpropagation uses a biologically implausible procedure that propagates error signals backward across synaptic connections, in the opposite direction of the way that activation typically flows. Furthermore, the XCAL network experiences a sequence of activation states, going from an expectation to experiencing a subsequent outcome, and learns on the difference between these two states. In contrast, backpropagation computes a single error *delta* value that is effectively the difference between the outcome and the expectation, and then sends this single value backwards across the connections. See the *Backpropagation* Appendix for how these two different things can be mathematically equivalent.

Exploration of Error-Driven Learning

The `pat_assoc` simulation from CCN Sims provides a nice demonstration of the limitations of self-organizing Hebbian-style learning, and how error-driven learning overcomes these limitations, in the context of a simple two-layer pattern associator that learns basic input/output mappings. Follow the directions in that simulation link to run the exploration.

You should have seen that one of the input/output mapping tasks was *impossible* for even error-driven learning to solve, in the two-layer network. The next exploration, `err_driven_hidden` shows that the addition of a hidden layer, combined with the powerful error-driven learning mechanism, enables even this

"impossible" problem to now be solved. This demonstrates the computational power of the Backpropagation algorithm.

Combined Self-Organizing and Error-Driven Learning

Although scientists have a tendency to want to choose sides strongly and declare that either self-organizing learning or error-driven learning is the best way to go, there are actually many advantages to combining both forms of learning together. Each form of learning has complementary strengths and weaknesses:

- Self-organizing is more *robust*, because it only depends on local statistics of firing, whereas error-driven learning implicitly depends on error signals coming from potentially distant areas. Self-organizing can achieve something useful even when the error signals are remote or not yet very coherent.
- But self-organizing learning is also very *myopic* – it does not coordinate with learning in other layers, and thus tends to be "greedy". In contrast, error-driven learning achieves this coordination, and can learn to solve problems that require collective action of multiple units across multiple layers.

One analogy that may prove useful is that error-driven learning is like left-wing politics – it requires all the different layers and units to be working together to achieve common goals, whereas self-organizing learning is like right-wing politics, emphasizing local, greedy actions that somehow also benefit society as a whole, without explicitly coordinating with others. The tradeoffs of these political approaches are similar to those of the respective forms of learning. Socialist approaches can leave individual people feeling not very motivated, as they are just a little cog in a huge faceless machine. Similarly, neurons that depend strictly on error-driven learning can end up not learning very much, as they only need to make a very small and somewhat "anonymous" contribution to solving the overall problem. Once the error signals have been eliminated (i.e., expectations match outcomes), learning stops. We will see that networks that rely on pure error-driven learning often have very random-looking weights, reflecting this minimum of effort expended toward solving the overall problem. On the other side, more strongly right-wing capitalist approaches can end up with excessive positive feedback loops (rich get ever richer), and are typically not good at dealing with longer-term, larger-scale problems that require coordination and planning. Similarly, purely self-organizing models tend to end up with more uneven distributions of "representational wealth" and almost never end up solving challenging problems, preferring instead to just greedily encode whatever interesting statistics come their way. Interestingly, our models suggest that a balance of both approaches – a *centrist* approach – seems to work best! Perhaps this lesson can be generalized back to the political arena.

Colorful analogies aside, the actual mechanics of combining both forms of learning within the XCAL framework amounts to merging the two different definitions of the floating threshold value. Biologically, we think that there is a combined weighted average of the two thresholds, using a "lambda" parameter λ to weight the long-term receiver average (self-organizing) relative to the medium-term synaptic co-product:

$$\theta_p = \lambda y_l + (1 - \lambda) x_m y_m$$

However, computationally, it is clearer and simpler to just combine separate XCAL functions, each with their own weighting function – due to the linearity of the function, this is mathematically equivalent:

$$\Delta w = \lambda_l f_{xcal}(x_s y_s, y_l) + \lambda_m f_{xcal}(x_s y_s, x_m y_m)$$

It is reasonable that these lambda parameters may differ according to brain area (i.e., some brain systems learn more about statistical regularities, whereas others are more focused on minimizing error), and even that it may be dynamically regulated (i.e. transient changes in neuromodulators like dopamine and acetylcholine can influence the degree to which error signals are emphasized). To see a concrete example of how Hebbian learning in early perceptual areas can complement error-driven learning to allow networks to generalize to novel situations, see the `hebberr_combo` simulation in CCN Sims.

There are small but reliable computational advantages to automating this balancing of self-organizing vs. error-driven learning (i.e., a dynamically-computed λ_l value, while keeping $\lambda_m = 1$), based on two factors: the magnitude of the y_l receiving-unit running average activation, and the average magnitude of the error signals present in a layer (see *Leabra Details* Chapter Appendix).

Weight Bounding and Contrast Enhancement

The one last issue we need to address computationally is the problem of synaptic weights growing without bound. In LTP experiments, it is clear that there is a maximum synaptic weight value – you cannot continue to get LTP on the same synapse by driving it again and again. The weight value **saturates**. There is a natural bound on the lower end, for LTD, of zero. Mathematically, the simplest way to achieve this kind of weight bounding is through an **exponential approach** function, where weight changes become exponentially smaller as the bounds are approached. This function is most directly expressed in a programming language format, as it involves a conditional:

```
if   dWt > 0 then Wt = Wt + (1 - Wt) * dWt
else              Wt = Wt + Wt * dWt
```

In words: if weights are supposed to increase (dWt is positive), then multiply the rate of increase by 1-Wt, where 1 is the upper bound, and otherwise, multiply by the weight value itself. As the weight approaches 1, the weight increases get smaller and smaller, and similarly as the weight value approaches 0.

The exponential approach function works well at keeping weights bounded in a graded way (much better than simply clipping weight values at the bounds, which loses all the signal for saturated weights), but it also creates a strong tendency for weights to hang out in the middle of the range, around .5. This creates problems because then neurons don't have sufficiently distinct responses to different input patterns, and then the inhibitory competition breaks down (many neurons become weakly activated), which then interferes with the positive feedback loop that is essential for learning, etc. To counteract these problems, while maintaining the exponential bounding, we introduce a **contrast enhancement** function on the weights:

$$\hat{w} = \frac{1}{1 + \left(\frac{w}{\theta(1-w)}\right)^{-\gamma}}$$

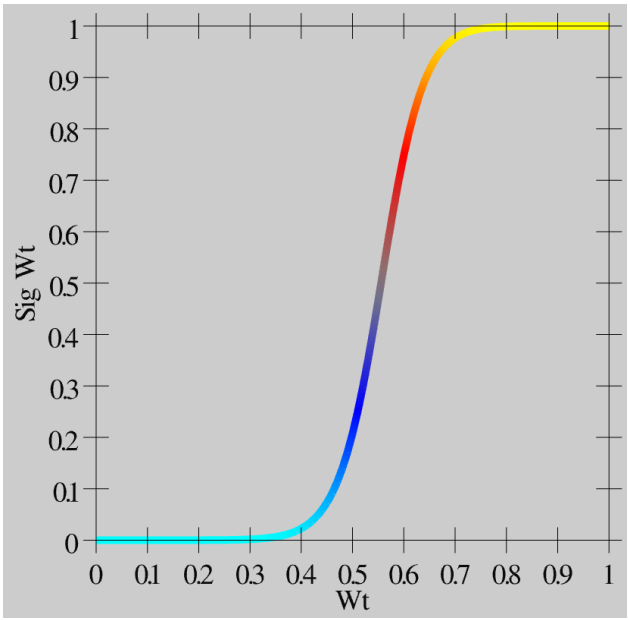


Figure 4.11: Weight contrast enhancement function, gain (gamma) = 6, offset (theta) = 1.25.

As you can see in Figure 4.11, this function creates greater contrast for weight values around this .5 central value – they get pushed up or down to the extremes. This contrast-enhanced weight value is then used for communication among the neurons, and is what shows up as the wt value in the simulator.

Biologically, we think of the plain weight value w, which is involved in the learning functions, as an *internal* variable that accurately tracks the statistics of the learning functions, while the contrast-enhanced weight value is the actual synaptic efficacy value that you measure and observe as the strength of interaction

among neurons. Thus, the plain w value may correspond to the phosphorylation state of CAMKII or some other appropriate internal value that mediates synaptic plasticity.

Finally, see the *Leabra Details* Appendix for a few implementational details about the way that the time averages are computed, which don't affect anything conceptually, but if you really want to know *exactly* what is going on..

When, Exactly, is there an Outcome that should Drive Learning?

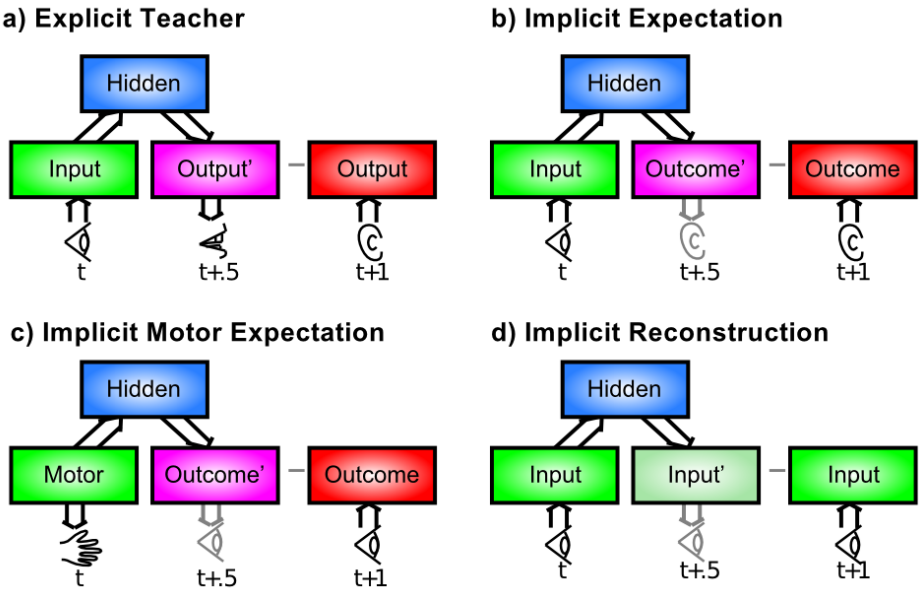


Figure 4.12: Different situations that give rise to a contrast between expectations and outcomes. a) The simplest case of explicit teacher / parent input – a visual input (e.g., an object) at time t drives a verbal output (e.g., the name of the object), and the teacher then corrects (or confirms) the output. b) The same scenario can go through without actually producing a verbal output – instead just an expectation of what someone else might say, and this can be compared with what is actually said to derive useful error signals. c) Is a specific instance of when many expectations are likely to be generated, when a motor action (e.g., pushing food off of a high chair) drives an expectation about the visual outcomes associated with the action, which then occur (to the seemingly endless delight of the mischievous infant). d) Involves making an "expectation" about what you actually just saw – reconstructing or generating the input (otherwise known as generative model or an auto-encoder) – the input itself serves as its own training signal in this case.

This is the biggest remaining question for error-driven learning. You may not have even noticed this issue, but once you start to think about implementing the XCAL equations on a computer, it quickly becomes a major problem. We have talked about how the error-driven learning reflects the difference between an outcome and an expectation, but it really matters that the short-term average activation representing the outcome state reflects some kind of actual outcome that is worth learning about. Figure 4.12 illustrates four primary categories of situations in which an outcome state can arise, which can play out in myriad ways in different real-world situations.

In our most recent framework described briefly above (O'Reilly, Wyatte, and Rohrlich 2017), the expectation-outcome timing is specified in terms of the 100 msec alpha trial. And within this trial, the combined circuitry between the deep neocortical layers and the thalamus end up producing an outcome state that drives *predictive auto-encoder* learning, which is basically the last case (d) in Figure 4.12, with an extra twist that during every 100 msec alpha trial, the network attempts to predict what will happen in the next 100 msec – the predictive aspect of the auto-encoder idea. Specifically, the deep layers attempt to predict what the bottom-up driven activity pattern over the thalamus will look like in the final plus-phase quarter of the alpha trial, based on activations present during the prior alpha trial. Because of the extensive

bidirectional connectivity between brain areas, the cross-modal expectation / output sequence shown in panel (b) of is also supported by this mechanism. A later revision of this text will cover these ideas in more detail. Preliminary versions are available: (O'Reilly, Wyatte, and Rohrlich 2017; Kachergis et al. 2014).

Another hypothesis for something that "marks" the presence of an important outcome is a phasic burst of a neuromodulator like **dopamine**. It is well established that dopamine bursts occur when an unexpected outcome arises, at least in the context of expectations of reward or punishment (we'll discuss this in detail in the Motor Control and Reinforcement Learning Chapter. Furthermore, we know from a number of studies that dopamine plays a strong role in modulating synaptic plasticity. Under this hypothesis, the cortical network is always humming along doing standard BCM-like self-organizing learning at a relatively low learning rate (due to a small lambda parameter in the combined XCAL equation, which presumably corresponds to the rate of synaptic plasticity associated with the baseline tonic levels of dopamine), and then, when something unexpected occurs, a dopamine burst drives stronger error-driven learning, with the immediate short-term average "marked" by the dopamine burst as being associated with this important (salient) outcome. The XCAL learning will automatically contrast this immediate short-term average with the immediately available medium-term average, which presumably reflects an important contribution from the prior expectation state that was just violated by the outcome.

There are many other possible ideas for how the time for error-driven learning is marked, some of which involve local emergent dynamics in the network itself, and others that involve other neuromodulators, or networks with broad connectivity to broadcast an appropriate "learn now" signal. From everything we know about the brain, there are likely several such learning signals, each of which being useful in some particular subset of situations. This is an active area of ongoing research.

The Leabra Framework

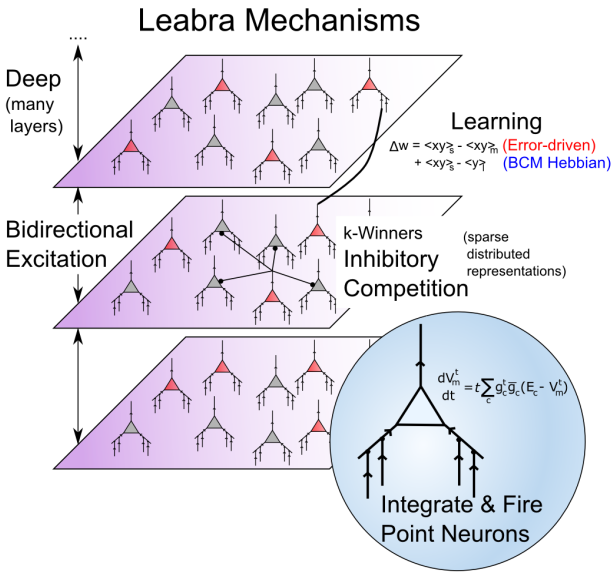


Figure 4.13: Summary of all the mechanisms in the Leabra framework used in this text, providing a summary of the last three chapters.

Figure 4.13 provides a summary of the **Leabra** framework, which is the name given to the combination of all the neural mechanisms that have been developed to this point in the text. Leabra stands for *Learning in an Error-driven and Associative, Biologically Realistic Algorithm* – the name is intended to evoke the "Libra" balance scale, where in this case the balance is reflected in the combination of error-driven and self-organizing learning ("associative" is another name for Hebbian learning). It also represents a balance between low-level, biologically-detailed models, and more abstract computationally-motivated models. The biologically-based way of doing error-driven learning requires bidirectional connectivity, and the Leabra framework is relatively

unique in its ability to learn complex computational tasks in the context of this pervasive bidirectional connectivity. Also, the FFFB inhibitory function producing k-Winners-Take-All dynamics is unique to the Leabra framework, and is also very important for its overall behavior, especially in managing the dynamics that arise with the bidirectional connectivity.

The different elements of the Leabra framework are therefore synergistic with each other, and as we have discussed, highly compatible with the known biological features of the neocortex. Thus, the Leabra framework provides a solid foundation for the cognitive neuroscience models that we explore next in the second part of the text.

Exploration of Leabra

Open the `family_trees` simulation in CCN Sims to explore Leabra learning in a deep multi-layered network running a more complex task with some real-world relevance. This simulation is very interesting for showing how networks can create their own similarity structure based on functional relationships, refuting the common misconception that networks are driven purely by input similarity structure.

Appendix

Here are all the sub-topics within the Learning chapter, collected in one place for easy browsing. These may or may not be optional for a given course, depending on the instructor's specifications of what to read:

- **Detailed Biology of Learning:** more in-depth treatment of postsynaptic signaling cascades that mediate LTP and LTD, described in context of the (Urakubo et al. 2008) model of synaptic plasticity.
- **Hebbian Learning:** extensive treatment of computational properties of Hebbian learning – starts with a simple manual simulation of Hebbian learning showing exactly how and why it captures patterns of co-occurrence.
- **Backpropagation:** history and mathematical derivation of error-driven learning functions – strongly recommended to obtain greater insight into the computational nature of error-driven learning (starts with some important conceptual points before getting into the math).
- **Leabra Details:** contains misc implementational details about the learning mechanisms, including how time averaged activations are computed.
- Full set of Leabra equations on *emergent* leabra site.

Detailed Biology of Learning

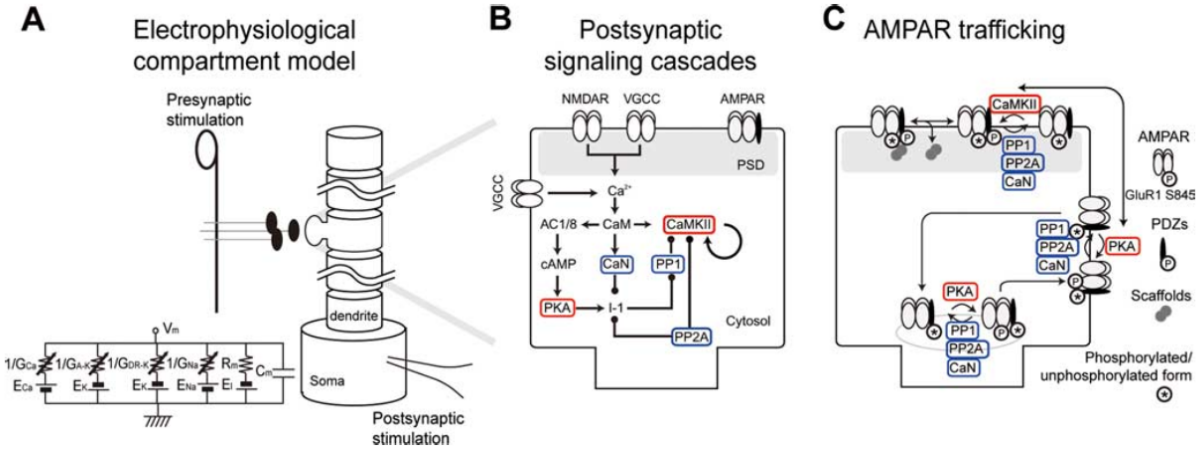


Figure 4.14: Diagram of full set of steps involved in synaptic plasticity, including kinases driven by Ca++ binding, and the effect they have on AMPA receptor expression in the synapse. Reproduced from Urakubo et al, 2008

Figure 4.14 shows a full set of chemical processes that are triggered by Ca++ influx, and result in

changes in AMPA receptor expression in the synapse. This figure is from the very detailed computational model by (Urakubo et al. 2008), which is highly recommended reading for those interested in the time course and dynamics of these chemical processes.

The Urakubo et al. (2008) model was constructed in a very "bottom up" fashion, by building in detailed chemical rate parameters and diffusion constants, etc, based on empirical measurements, for all of the major biological processes involved in synaptic plasticity. Having built this model, they found that it did *not* capture the classic spike timing dependent plasticity (STDP) curve, when driven by the exact STDP pairwise induction protocol (see figure of this in the main chapter text). However, by changing one aspect of the way the NMDA receptors function (adding what is known as an *allosteric* mechanism, where the NMDA receptor functions differently depending on binding by a substance called calmodulin), they were able to capture not only pairwise STDP, but also the weight changes that result from more complex patterns of spiking, in triplet and quadruplet experiments. Furthermore, they accurately capture the effects of changing the timing parameters on pairwise STDP experiments (e.g., interval between pairwise spikes, and number of repetitions thereof).

Thus, this model represents a remarkable bridge between detailed biological mechanisms, and the overall synaptic plasticity that results in actual experiments. Either this is a fantastic coincidence, or this model has managed to capture a reasonable chunk of the critical mechanisms of synaptic plasticity. We adopt the later view, and therefore leverage this model as a basis for our computational models described in the main chapter.

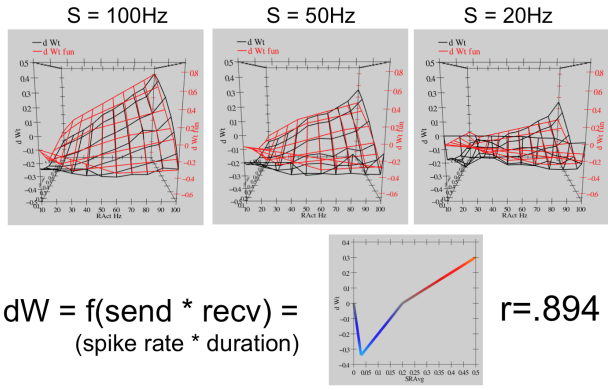


Figure 4.15: Fit of the Urakubo et al. (2008) model with a simple learning function driven by the product of the total sending and receiving neural activity (frequency of firing in Hertz (Hz) times duration of firing in milliseconds). This simple linear function (called the ''XCAL dWt function'') can capture a considerable amount of the regularity present in the behavior of the Urakubo et al. (2008) model, with a correlation value of"'r=0.894". The top portion of the figure shows graphs of three different sending Hz, with the X (horizontal) axis being the receiving unit Hz, Z (depth) is the duration of activity in fractions of a second (.1 to 1), and Y (vertical) is net change in synaptic weight. The black lines are the measured results from Urakubo et al. (2008), and the red are the values computed from the simple piecewise-linear function shown at the bottom of the figure.

For the bottom-up derivation of XCAL, we systematically subjected the biologically detailed Urakubo et al. (2008) model to a range of different *pre* and *post* spike trains, with durations from 100 msec to a second, and spike rates from 10 to 100 Hz (Hertz or spikes per second). We then tried to fit the pattern of weight changes that resulted using a piecewise linear function of some form. Figure 4.15 shows the results. The resulting function is shown at the bottom of the figure – if you compare with Figure 4.4, you should see that this is essentially the qualitative shape of the function relating weight change to level of Ca++. The top part of the figure is probably too complex to parse very well, but you should get the general impression that the red lines (generated by the piecewise linear function) fit the black lines (data from the Urakubo et al. (2008) model) pretty well. The correlation value of .894 represents a very good fit of the function to the data.

Thus, we are able to capture much of the incredible complexity of the Urakubo et al. (2008) model (and by extension, hopefully, the complexity of the actual synaptic plasticity mechanisms in the brain) using an extremely simple function. This is a very powerful simplification. But what does it mean?

First, the main input into this function is the *total synaptic activity* reflecting the firing rate and duration of activity of the sending and receiving neurons. In mathematical terms for a rate-code model with sending activity rate x and receiving activity rate y, this would just be the "Hebbian" product we described above:

$$\Delta w = f_{xcal}\left(xy, \theta_p\right)$$

Where f_{xcal} is the piecewise linear function shown in Figure 4.15 or 4.4, which we can call the *XCAL dWt function*. It also takes an additional dynamic parameter θ_p, which determines the point at which it crosses over from negative to positive weight changes – we'll discuss this at length in a moment. Just for kicks, here is the mathematical expression of this function:

$$f_{xcal}(xy, \theta_p) = \begin{cases} (xy - \theta_p) & \text{if } xy > \theta_p\theta_d \\ -xy(1 - \theta_d)/\theta_d & \text{otherwise} \end{cases}$$

where $\theta_d = .1$ is a constant that determines the point where the function reverses back toward zero within the weight decrease regime – this reversal point occurs at $\theta_p\theta_d$, so that it adapts according to the dynamic θ_p value.

As noted in the main chapter, the dependence of the NMDA channel on activity of both sending and receiving neurons can be summarized with this simple Hebbian product, and the level of Ca++ is likely to reflect this value. Thus, the XCAL dWt function makes very good sense in these terms: it reflects the qualitative nature of weight changes as a function of Ca++ that has been established from empirical studies and postulated by other theoretical models for a long time. When realistic spike trains with many spikes drive the complex synaptic plasticity mechanisms, this fundamental function emerges.

As a learning function, this basic XCAL dWt function has some advantages over a plain Hebbian function, while sharing its basic nature due to the "pre * post" term at its core. For example, because of the shape of the dWt function, weights will go down as well as up, whereas the Hebbian function only causes weights to increase. But it still has the problem that weights will increase without bound, and we'll see in the next section that some other top-down computationally-motivated modifications can result in a much more powerful form of learning.

Hebbian Learning

This subsection provides a detailed treatment of Hebbian learning and popular variants thereof.

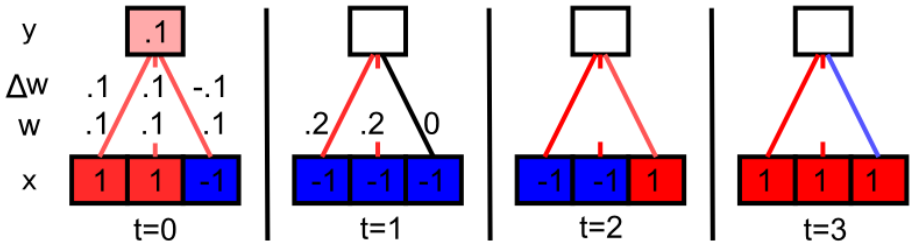


Figure 4.16: Simple Hebbian learning demonstration across 4 time steps (t=0 thru 3). Bottom row of network has 3 input units, the last of which fires in an ''uncorrelated" fashion with the other two. They all start out with weights w = .1. Receiving activity is just a linear sum of the sending activations times weights: $y = \sum xw = .1$ for the first time step. Learning is simple Hebbian: $\Delta w = xy$. As you fill in the remainder of the activations, weights, and weight changes, you will find that the two correlated input units dominate the receiving unit activation, and thus they end up being correlated in their activity, causing their weights to always increase. The third unit sometimes goes up and sometimes down, with no net increase over time. Thus, Hebbian learning discovers correlations in the inputs.

Figure 4.16 shows a simple demonstration of how Hebbian learning causes the receiving network to discover correlations in the patterns of input unit activation. The input units that are correlated end up dominating the receiving unit activity, and thus the receiving unit ends up being correlated with this subset

of correlated inputs, and their weights always increase under the Hebbian learning function. Uncorrelated inputs bounce around without a systematic trend. If you keep going, you'll see that the weights grow quickly without bound, so this is not a practical learning function, but it illustrates the essence of Hebbian learning.

Next, we do some math to show that the simplest version of Hebbian correlational learning, in the case of a single linear receiving unit that receives input from a set of input units, result in the unit extracting the first **principle component** of correlation in the patterns of activity over the input units.

Because it is linear, the receiving unit's activation function is just the weighted sum of its inputs

$$y_j = \sum_k x_k w_{kj}$$

where k (rather than the usual i) indexes over input units, for reasons that will become clear (and all of the variables are a function of the current time step t reflecting different inputs). The weight change is:

$$\Delta_t w_{ij} = \epsilon x_i y_j$$

where ϵ is the *learning rate* and i is the index of a particular input unit, and weights just increment these changes over time:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta_t w_{ij}$$

To understand the aggregate effects of learning over many patterns, we can just sum the changes over time:

$$\Delta w_{ij} = \epsilon \sum_t x_i y_j$$

and we assume that $\epsilon = 1/N$, where N is the total number of patterns in the input. This turns the sum into an *average*:

$$\Delta w_{ij} = \langle x_i y_j \rangle_t$$

Next, substitute into this equation the formula for y_j, showing that the weight changes are a function of the *correlations* between the input units:

$$\Delta w_{ij} = \langle x_i \sum_k x_k w_{kj} \rangle_t$$

$$= \sum_k \langle x_i x_k \rangle_t \langle w_{kj} \rangle_t$$

$$= \sum_k \mathbf{C}_{ik} \langle w_{kj} \rangle_t$$

This new variable $\mathbf{C}_{ik}$ is an element of the *correlation matrix* between the two input units i and k, where correlation is defined here as the expected value (average) of the product of their activity values over time ($\mathbf{C}_{ik} = \langle x_i x_k \rangle_t$). You might be familiar with the more standard correlation measure:

$$\mathbf{C}_{ik} = \frac{\langle (x_i - \mu_i)(x_k - \mu_k) \rangle_t}{\sqrt{\sigma_i^2 \sigma_k^2}}$$

which subtracts away the mean values (μ) of the variables before taking their product, and normalizes the result by their variances (σ^2). Thus, an important simplification in this form of Hebbian correlational learning is that it assumes that the activation variables have zero mean and unit variance.

The implication of all this is that where strong correlations exist across input units, the weights for those units will increase because this average correlation value will be relatively large. Interestingly, if we run this learning rule long enough, the weights will become dominated by the strongest set of correlations present in

the input, with the gap between the strongest set and the next strongest becoming increasingly large. Thus, this simple Hebbian rule learns the *first* (strongest) principal component of the input data.

One problem with the simple Hebbian learning rule is that the weights become infinitely large as learning continues. One solution to this problem was proposed by (Oja 1982), known as *subtractive normalization*:

$$\Delta w_{ij} = \epsilon(x_i y_j - y_j^2 w_{ij})$$

As we did in Chapter 2, you just set the equation equal to zero and solve for the *equilibrium* or *asymptotic* weight values:

$$0 = \epsilon(x_i y_j - y_j^2 w_{ij})$$

$$w_{ij} = \frac{x_i}{y_j}$$

$$w_{ij} = \frac{x_i}{\sum_k x_k w_{kj}}$$

Thus, the weight from a given input unit will end up representing the proportion of that input's activation relative to the total weighted activation over all the other inputs. This will keep the weights from growing without bound. Finally, because it is primarily based on the same correlation terms $\mathbf{C}_{ik}$ as the previous simple Hebbian learning rule, this Oja rule still computes the first principal component of the input data (though the proof of this is somewhat more involved, see (Hertz, Krogh, and Palmer 1991) for a nice treatment).

Moving beyond a single hidden unit, there are ways of configuring inhibition so that the units end up learning the sequence of PCA values of the correlation matrix in eigenvalue order (Sanger 1989; Oja 1989). In (O'Reilly and Munakata 2000), we developed a different alternative known as **conditional principal components analysis** or CPCA, which assumes that we want the weights for a given input unit to represent the conditional probability that the input unit (x_i) was active given that the receiving unit (y_j) was also active:

$$w_{ij} = P(x_i = 1 | y_j = 1)$$

$$w_{ij} = P(x_i | y_j)$$

where the second form uses simplified notation that will continue to be used below.

The important characteristic of CPCA is that the weights will reflect the extent to which a given input unit is active across the subset of input patterns represented by the receiving unit (i.e., conditioned on this receiving unit). If an input pattern is a very typical aspect of such inputs, then the weights from it will be large (near 1), and if it is not so typical, they will be small (near 0).

Following the analysis of (Rumelhart and Zipser 1985), the CPCA learning rule can be derived as:

$$\Delta w_{ij} = \epsilon[y_j x_i - y_j w_{ij}]$$

$$= \epsilon y_j (x_i - w_{ij})$$

The two equivalent forms of this equation are shown to emphasize the similarity of this learning rule to Oja's normalized PCA learning rule, while also showing its simpler form, which emphasizes that the weights are adjusted to match the value of the sending unit activation x_i (i.e., minimizing the difference between x_i and w_{ij}), weighted in proportion to the activation of the receiving unit (y_j).

We use the expression $P(y_j|t)$ to represent the probability that the receiving unit y_j is active given that some particular input pattern t was presented. $P(x_i|t)$ represents the corresponding thing for the sending unit x_i. Substituting these into the learning rule, the total weight update computed over all the possible patterns t (and multiplying by the probability that each pattern occurs, $P(t)$) is:

$$\Delta w_{ij} = \epsilon \sum_t [P(y_j|t)P(x_i|t) - P(y_j|t)w_{ij}]P(t)$$

$$= \epsilon \left(\sum_t P(y_j|t)P(x_i|t)P(t) - \sum_t P(y_j|t)P(t)w_{ij} \right)$$

As usual, we set Δw_{ij} to zero and solve:

$$w_{ij} = \frac{\sum_t P(y_j|t)P(x_i|t)P(t)}{\sum_t P(y_j|t)P(t)}$$

Interestingly, the numerator is the definition of the joint probability of the sending and receiving units both being active together across all the patterns t, which is just $P(y_j, x_i)$. Similarly, the denominator gives the probability of the receiving unit being active over all the patterns, or $P(y_j)$. Thus, we can rewrite the preceding equation as:

$$w_{ij} = \frac{P(y_j, x_i)}{P(y_j)}$$

$$w_{ij} = P(x_i|y_j)$$

at which point it becomes clear that this fraction of the joint probability over the probability of the receiver is just the definition of the conditional probability of the sender given the receiver.

Although CPCA is effective and well-defined mathematically, it suffers one major problem relative to the BCM formulation that we now use: it drives significant LTD (weight decrease) when a sending neuron is *not* active, and the receiving unit is active. This results in a significant amount of interference of learning across time. By contrast, the XCAL dWt function specifically returns to zero when either sending or receiving neuron has zero activity, and that significantly reduces interference, preserving existing weight values for inactive neurons.

Backpropagation

In this subtopic, we trace the mathematical progression of error-driven learning from a simple two-layer network, which was developed in 1960, to a network with three or more layers, which took 26 years to be invented for the last time (several others invented it earlier, but it didn't really catch on). In the process, we develop a much more rigorous understanding of what error-driven learning is, which can also be applied directly to understanding what the XCAL learning function in its error-driven mode is doing. We start off with a high-level conceptual summary, working backward from XCAL, that should be accessible to those with a basic mathematical background (requiring only basic algebra), and then get progressively more into the math, where we take advantage of concepts from calculus (namely, the notion of a partial derivative).

The highest-level summary is that XCAL provides a very good approximation to an optimal form of error-driven learning, called **error backpropagation**, which works by directly minimizing a computed error statistic through steepest gradient descent. In other words, backpropagation is mathematically designed to learn whatever you throw at it in the most direct way possible, and XCAL basically does the same thing. If you want to first understand the principled math behind backpropagation, skip down to read the *Gradient Descent on Error and the Delta Rule* section below, and then return here to see how XCAL approximates this function.

The critical difference is that XCAL uses bidirectional activation dynamics to communicate error signals throughout the network, whereas backpropagation uses a biologically implausible procedure that propagates error signals backward across weight values, in the opposite direction of the way that activation typically flows (hence the name). As discussed in the main chapter, the XCAL network experiences a sequence of activation states, going from an expectation to experiencing a subsequent outcome, and learns on the difference between these two states. In contrast, backpropagation computes a single error *delta* value that is effectively the difference between the outcome and the expectation, and then sends this single value backwards across the weights. In the following math, we show how these two ways of doing error-driven learning are approximately equivalent.

The primary value of this exercise is to first establish that XCAL can perform a powerful, effective form of error-driven learning, and also to obtain further insights into the essential character of this error-driven

learning by understanding how it is derived from first principles. One of the most important intuitive ideas that emerges from this analysis is the notion of **credit assignment** – you are encouraged to read up through that section (or just skip ahead to it if you really can't stand the following math, which again only requires basic algebra).

To begin, the error-driven aspect of XCAL is effectively:

$$\Delta w \approx \langle xy \rangle_s - \langle xy \rangle_m$$

which reflects a contrast between the average firing rate during the outcome, represented by the first term, and that over the expectation, represented by the second term.

We will see how this XCAL rule is related to the backpropagation error-minimizing rule, but achieves this function in a more biologically constrained way. This was the same goal of previous attempts including the *GeneRec (generalized recirculation)* algorithm (O'Reilly 1996), which is equivalent to the *Contrastive Hebbian Learning (CHL)* equation (Movellan and McClelland 1993):

$$\Delta w = \left(x^+ y^+\right) - \left(x^- y^-\right)$$

Here, the first term is the activity of the sending and receiving units during the outcome (in the *plus phase*), while the second term is the activity during the expectation (in the *minus phase*). CHL is so-named because it involves the contrast or difference between two Hebbian-like terms. As you can see, XCAL is essentially equivalent to CHL, despite a few differences:

- XCAL actually uses the XCAL dWt function instead of a direct subtraction, which causes weight changes to go to 0 at when short term activity is 0 (as dictated by the biology).
- XCAL is based on average activations across the entire evolution of attractors (reflected by accumulated Ca++ levels), instead of based on single points of activation (i.e., the final attractor state in each of two phases, as used somewhat unrealistically in CHL – how would the plasticity rules 'know' exactly what counts as the final state of each phase?).

For the present purposes, we can safely ignore these factors, which allows us to leverage all of the analysis that went into understanding GeneRec – itself a large step towards biological plausibility relative to backpropagation.

The core of this analysis revolves around the following simpler version of the GeneRec equation, which we call the *GeneRec delta equation*:

$$\Delta w = x^- \left(y^+ - y^-\right)$$

where the weight change is driven only by the *delta* in activity on the receiving unit y between the plus (outcome) and minus (expectation) phases, multiplied by the sending unit activation x. One can derive the full CHL equation from this simpler GeneRec delta equation by adding a constraint that the weight changes computed by the sending unit to the receiving unit be the same as those of the receiving unit to the sending unit (i.e., a *symmetry constraint* based on bidirectional connectivity), and by replacing the minus phase activation for the sending unit with the average of the minus and plus phase activations (which ends up being equivalent to the *midpoint method* for integrating a differential equation). You can find the actual mathematics of this derivation later in this section, but you can take our word for it for the time being.

Interestingly, the GeneRec delta equation is equivalent in form to the *delta rule*, which we derive below as the optimal way to reduce error in a two layer network (input units sending to output units, with no hidden units in between). The delta rule was originally derived by (Widrow and Hoff 1960), and it is also basically equivalent to a gradient descent solution to linear regression. This is very basic old-school math.

But two-layer networks are very limited in what they can compute. As we discussed in the *Networks* Chapter, you really need those hidden layers to form higher-level ways of re-categorizing the input, to solve challenging problems (you will also see this directly in the simulation explorations in this chapter). As we discuss more below, the limitations of the delta rule and two-layer networks were highlighted in a very critical paper by (Minsky and Papert 1969), which brought research in the field of neural network models nearly to a standstill for nearly 20 years.

In 1986, David Rumelhart and colleagues (Rumelhart, Hinton, and Williams 1986) published a landmark paper on the *backpropagation* learning algorithm, which essentially extended the delta rule to networks with

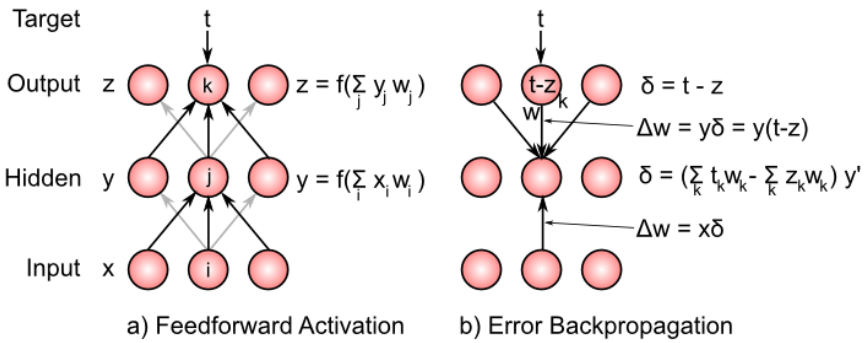


Figure 4.17: Illustration of backpropgation computation in three-layer network. First, the feedforward activation pass generates a pattern of activations across the units in the network, cascading from input, to hidden to output. Then, "delta" values are propagated *backward* in the reverse direction across the same weights. The delta sum is broken out in the hidden layer to facilitate comparison with the GeneRec algorithm as shown in the next figure.

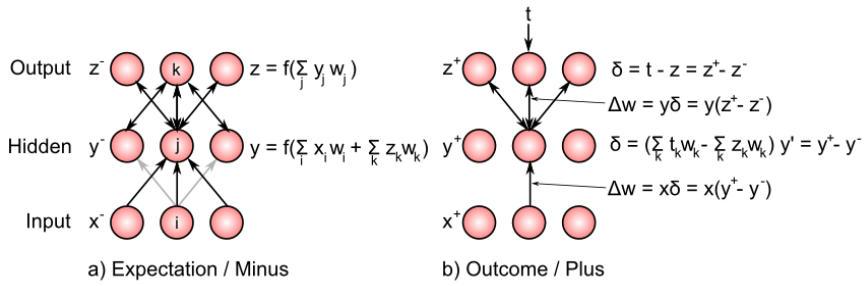


Figure 4.18: Illustration of GeneRec/XCAL computation in three-layer network, for comparison with previous figure showing backpropagation. Activations settle in the expectation/minus phase, in response to input activations presented to the input layer. Activation flows bidirectionally, so that the hidden units are driven both by inputs and activations that arise on the output units. In the outcome/plus phase, "target" values drive the output unit activations, and due to the bidirectional connectivity, these also influence the hidden units in the plus phase. Mathematically, changing the weights based on the difference in hidden layer activation states between the plus and minus phases results in a close approximation to the delta value computed by backpropagation. This same rule is then used to change the weights into the hidden units from the input units (delta times sending activation), which is the same form used in backpropagation, and identical in form to the delta rule.

three or more layers (Figure 4.18). These models have no limitations on what they can learn, and they opened up a huge revival in neural network research, with backpropagation neural networks providing practical and theoretically interesting solutions to a very wide range of problems.

The essence of the backpropagation (also called "backprop") algorithm is captured in this *delta backpropagation equation*:

$$\Delta w = x \left(\sum_k \delta_k w_k \right) y'$$

where x is again the sending activity value, δ is the error derivative for the units in the next layer *above* the layer containing the current receiving unit y (with each such unit indexed by the subscript k), and w_k is the weight *from* the receiving unit y to the k'th such unit in the next layer above (see Figure 4.18). Ignore the z' term for the time being – it is the derivative of the receiving units activation function, and it will come in handy in a bit.

So we're propagating this "delta" (error) value *backward* across the weights, in the opposite direction that the activation typically flows in the "feedforward" direction, which is from the input to the hidden to the output (backprop networks are typically feedforward, though bidirectional versions have been developed as discussed below). This is the origin of the "backpropagation" name.

Before we unpack this equation a bit more, let's consider what happens at the *output* layer in a standard three-layer backprop network like that pictured in the Figure. In these networks, there is no outcome/plus phase, but instead we just compare the output activity of units in the output layer (effectively the expectation) and compute externally the difference between these activities and the *target* activity values t. The difference is the *delta* value:

$$\delta = t - z$$

and is used to drive learning by changing the weight from sending unit y in the hidden layer to a given output unit z is:

$$\Delta w = y\delta = y(t - z)$$

You should recognize that this is exactly the *delta rule* as described above (where we keep in mind that y is now a sending activation to the output units). The delta rule is really the essence of all error-driven learning methods.

Now let's get back to the delta backpropagation equation, and see how we can get from it to GeneRec (and thus to XCAL). We just need to replace the δ_k term with the value for the output units, and then do some basic rearranging of terms, and we get very close to the GeneRec delta equation:

$$\Delta w = x \left(\sum_k (t_k - z_k) w_k \right) y'$$

$$\Delta w = x \left(\sum_k t_k w_k - \sum_k z_k w_k \right) y'$$

If you compare this last equation with the GeneRec delta equation, they would be equivalent (except for the y' term that we're still ignoring) if we made the following definitions:

$$y^+ = \sum_k t_k w_k$$

$$y^- = \sum_k z_k w_k$$

$$x^- = x$$

Interestingly, these sum terms are identical to the *net input* that unit y would receive from unit z if the weight went the other way, or, critically, if y also received a *symmetric, bidirectional connection* from z, in addition to sending activity to z. Thus, we arrive at the critical insight behind the GeneRec algorithm relative to the backpropagation algorithm:

Symmetric bidirectional connectivity can convey error signals as the difference between two activity states (plus/outcome vs. minus/expectation), instead of sending a single "delta" error value backward down a single weight in the opposite (backpropagation) direction.

The only wrinkle in this argument at this point is that we had to assign the activation states of the receiving unit to be equal to those net-input like terms (even though we use non-linear thresholded activation functions), and also those net input terms ignore the other inputs that the receiving unit should also receive from the sending units in the input layer. The second problem is easily dispensed with, because those inputs from the input layer would be common to both "phases" of activation, and thus they cancel out when we subtract $y^+ - y^-$. The first problem can be solved by finally no longer ignoring the *y'* term – it turns out that the difference between a function evaluated at two different points can be approximated as the difference between the two points, times the derivative of the function:

$$f(a) - f(b) \approx f'(a)(a - b)$$

So we can now say that the activations states of y are a function of these net input terms:

$$y^+ = f\left(\sum_k t_k w_k\right)$$

$$y^- = f\left(\sum_k z_k w_k\right)$$

and thus their difference can be approximated by the difference in net inputs times the activation function derivative:

$$y^+ - y^- \approx y'\left(\sum_k t_k w_k - \sum_k z_k w_k\right)$$

Which gets us right back to the GeneRec delta equation as being a good approximation to the delta backpropagation equation:

$$\Delta w = x^-\left(y^+ - y^-\right) \approx x\left(\sum_k \delta_k w_k\right) y'$$

So if you've followed along to this point, you can now rest easy by knowing that the GeneRec (and thus XCAL) learning functions are actually very good approximations to error backpropagation. As we noted at the outset, XCAL uses bidirectional activation dynamics to communicate error signals throughout the network, in terms of averaged activity over two distinct states of activation (expectation followed by outcome), whereas backpropagation uses a biologically implausible procedure that propagates a single error value (outcome - expectation) backward across weight values, in the opposite direction of the way that activation typically flows.

Gradient Descent on Error and the Delta Rule

Now, we'll back up a bit and trace more of a historical trajectory through error-driven learning, starting by deriving the delta rule through the principle of *steepest gradient descent* on an error function. To really understand the mathematics here, you'll need to understand calculus and the notion of a *derivative*. Interestingly, we only need the most basic forms of derivatives to do this math – it really isn't very fancy. The basic strategy is to define an *error function* which tells you how poorly your network is doing at a task, and then take the negative of the derivative of this error function relative to the synaptic weights in the network, which then tells you how to *adjust the synaptic weights so as to minimize error*. This is what error-driven learning does, and mathematically, we take the simplest, most direct approach.

A very standard error function, commonly used in statistics, is the *sum squared error (SSE)*:

$$SSE = \sum_k \left(t_k - z_k\right)^2$$

which is the sum over output units (indexed by k) of the *target* activation t minus the actual output activation that the network produced (z), squared. There is typically an extra sum here too, over all the different input/output patterns that the network is being trained on, but it cancels out for all of the following math, so we can safely ignore it.

In the context of the expectation and outcome framework of the main chapter, the outcomes are the targets, and the expectations are the output activity of the network.

For the time being, we assume a linear activation function of activations from sending units y, and that we just have a simple two-layer network with these sending units projecting directly to the output units:

$$z_k = \sum_j y_j w_{jk}$$

Taking the negative of the derivative of SSE with respect to the weight w, which is easier computed by breaking it down into two parts using the *chain rule* to first get the derivative of SSE with respect to the output activation z, and multiply that by the derivative of z with respect to the weight:

$$\Delta w_{jk} = -\frac{\partial SSE}{\partial w_{jk}} = -\frac{\partial SSE}{\partial z_k}$$

$$= \frac{\partial z_k}{\partial w_{jk}} = (t_k - z_k) y_j$$

When you break down each step separately, it is all very straightforward:

$$\frac{\partial SSE}{\partial z_k} = -2(t_k - z_k)$$

$$\frac{\partial z_k}{\partial w_{jk}} = y_j$$

(the other elements of the sums drop out because the first partial derivative is with respect to z_k so derivative for all other z's is zero, and similarly the second partial derivative is with respect to y_j so the derivative for the other y's is zero.)

Thus, the negative of $\partial SSE / \partial w_{jk}$ is $2(t_k - z_k)$ and since 2 is a constant, we can just absorb it into the learning rate parameter.

Breaking down the error-minimization in this way, it becomes apparent that the weight change should be adjusted in proportion to both the error (difference between the target and the output) *and* the extent to which the sending unit y was active. This modulation of weight change by activity of the sending unit achieves a critical **credit assignment** function (or rather blame assignment in this case), so that when an error is made at the output, weights should only change for the sending units that contributed to that error. Sending units that were not active did not cause the error, and their weights are not adjusted.

As noted above, the original delta rule was published by (Widrow and Hoff 1960), followed in 1969 by the critique by (Minsky and Papert 1969), showing that such models could not learn a large class of basic but nonlinear logical functions, for example the XOR function. XOR states that the output should be true (active) if *either* one of two inputs are true, *but not both*. This requires a strong form of nonlinearity that simply could not be represented by such models. In retrospect, it should have been obvious that the problem was the use of a two-layer network, but as often happens, this critique left a bad "odor" over the field, and people simply pursued other approaches (mainly symbolic AI, which Minsky was an advocate for).

Then, roughly 26 years later, David Rumelhart and colleagues published a paper on the backpropagation learning algorithm, which extended the delta-rule style error-driven learning to networks with three or more layers. The addition of the extra layer(s) now allows such networks to solve XOR and any other kind of problem (there are proofs about the universality of the learning procedure). The problem is that above, we only considered how to change weights from a sending unit y to an output unit z, based on the error between the target t and actual output activity. But for multiple stages of hidden layers, how do we adjust the weights from the inputs to the hidden units? Interestingly, the mathematics of this involves simply adding a few more steps to the chain rule.

The overall derivation is as follows. The goal is to again minimize the error (SSE) as a function of the weights,

$$\Delta w_{ij} = -\frac{\partial SSE}{\partial w_{ij}}$$

$$= -\frac{\partial SSE}{\partial z_k}\frac{\partial z_k}{\partial \eta_k}\frac{\partial \eta_k}{\partial y_j}\frac{\partial y_j}{\partial \eta_j}\frac{\partial \eta_j}{\partial w_{ij}}$$

Although this looks like a lot, it is really just applying the same chain rule as above repeatedly. To know how to change the weights from input unit x_i to hidden unit y_j, we have to know how changes in this weight w_{ij} are related to changes in the SSE. This involves computing how the SSE changes with output activity, how output activity changes with its net input, how this net input changes with hidden unit activity y_j, how in turn this activity changes with its net input η_j, and finally, how this net input changes with the weights from sending unit x_i to hidden unit y_j. Once all of these factors are computed, they can be multiplied together to determine how the weight w_{ij} should be adjusted to minimize error, and this can be done for all sending units to all hidden units (and also as derived earlier, for all hidden units to all output units).

We again assume a linear activation function at the output for simplicity, so that $\partial z_k/\partial \eta_k = 1$. We allow for non-linear activation functions in the hidden units y, and simply refer to the derivative of this activation function as y' (which for the common sigmoidal activation functions turns out to be $y(1-y)$ but we leave it in generic form here so that it can be applied to any differentiable activation function. The solution to the above equation is then, applying each step in order,

$$-\frac{\partial SSE}{\partial w_{ij}} = \sum_k (t_k - z_k) * 1 * w_{jk} * y' * x_i$$

$$= x\left(\sum_k \delta_k w_{jk}\right) y'$$

as specified earlier.

You can see that this weight change occurs not only in proportion to the error at the output, and the 'backpropagated' error at the hidden unit activity y, but also to the activity of the sending unit x_i. So, once again, the learning rule assigns credit/blame to change weights based on active input units that contributed to the error, weighted by the degree to the error at the level of hidden unit activities contribute to errors at the output (which is the weight w_{jk}). At all steps along the process, the appropriate units and weights are factored in to minimize errors. This procedure can be repeated for any arbitrary number of layers, with repeated application of the chain rule.

GeneRec and Activation Differences

Finally, we can derive GeneRec in full (see (O'Reilly 1996) for more details). First reconsider the equation for the δ_j variable on the hidden unit in backpropagation:

$$\delta_j = -\sum_k (t_k - o_k) w_{jk} h_j (1 - h_j)$$

The main biological implausibility in this equation comes from the passing of the error information on the outputs backward, and the multiplying of this information by the feedforward weights and by the derivative of the activation function. There is simply no know biological way for such a signal to be transported backward like this (Crick 1989).

We avoid the implausible error propagation procedure by converting the computation of error information multiplied by the weights into a computation of the net input to the hidden units. For mathematical purposes, we assume for the moment that our bidirectional connections are symmetric, or $w_{jk} = w_{kj}$. We will see later that the argument below holds even if we do not assume exact symmetry. With $w_{jk} = w_{kj}$:

$$\delta_j = -\sum_k (t_k - o_k) w_{jk} h_j (1 - h_j)$$

$$= -\sum_k (t_k - o_k) w_{kj} h_j (1 - h_j)$$

$$= -\left(\sum_k (t_k w_{kj}) - \sum_k (o_k w_{kj}) \right) h_j (1 - h_j)$$

$$= -(\eta_j^+ - \eta_j^-) h_j (1 - h_j)$$

That is, w_{kj} can be substituted for w_{jk} and then this term can be multiplied through $(t_k - o_k)$ to get the difference between the net inputs to the hidden units (from the output units) in the two phases. Bidirectional connectivity thus allows error information to be communicated in terms of net input to the hidden units, rather than in terms of δs propagated backward and multiplied by the strength of the feedforward synapse.

Next, we can deal with the remaining $h_j(1 - h_j)$ by applying the difference-of-activation-states approximation. $h_j(1 - h_j)$ can be expressed as $\sigma'(\eta_j)$, the derivative of the activation function, so:

$$\delta_j = -(\eta_j^+ - \eta_j^-) \sigma'(\eta_j)$$

This product can be approximated by just the difference of the two sigmoidal activation values computed on these net inputs:

$$\delta_j \approx -(h_j^+ - h_j^-)$$

That is, the difference in a hidden unit's activation values is approximately equivalent to the difference in net inputs times the slope of the activation function. This also has the benefit of *implicitly* computing the derivative of the activation function, so we can use more complex biologically-based ones without worrying about needing to compute their derivatives.

Once we have the δ terms for the hidden units computed as the difference in activation across the two phases, we end up with:

$$\Delta w_{ij} = -\epsilon \delta_j s_i = \epsilon (h_j^+ - h_j^-) s_i^-$$

Thus, through bidirectional connectivity and the approximation of the product of net input differences and the derivative of the activation function, hidden units implicitly compute the information needed to minimize error as in backpropagation, but using only locally available activity signals.

To get to CHL, we can improve upon in two small but significant ways. First, there is a more sophisticated way of updating weights, known as the *midpoint method*, that uses the average of both the minus and plus phase activation of the sending unit x_i, instead of just the minus phase alone:

$$\Delta w_{ij} = \epsilon (y_j^+ - y_j^-) \frac{x_i^- + x_i^+}{2}$$

Second, the mathematical derivation of the learning rule depends on the weights being symmetric, and yet the basic GeneRec equation is not symmetric (i.e., the weight changes computed by unit j from unit i are not the same as those computed by unit i from unit j). So, even if the weights started out symmetric, they would not likely remain that way under the basic GeneRec equation. Making the weight changes symmetric (the same in both directions) both preserves any existing weight symmetry, and, when combined with a small amount of weight decay (Hinton 1989) and/or soft weight bounding, actually works to symmetrize initially asymmetric weights. A simple way of preserving symmetry is to take the average of the weight updates for the different weight directions:

$$\Delta w_{ij} = \epsilon \frac{1}{2} \left[(y_j^+ - y_j^-) \frac{(x_i^+ + x_i^-)}{2} + (x_i^+ - x_i^-) \frac{(y_j^+ + y_j^-)}{2} \right]$$

$$= \epsilon \left[x_i^+ y_j^+ - x_i^- y_j^- \right]$$

(where the 1/2 for averaging the weight updates in the two different directions gets folded into the arbitrary learning rate constant ϵ). Because many terms end up canceling, the weight change rule that results is just the CHL equation as shown above.

Leabra Details

The full set of Leabra equations on *emergent* leabra site.

First, it turns out in practice that computing the average of the sending and receiving activations over the short time period:

$$\langle xy \rangle_s$$

can be approximated by the computationally less expensive product of the averages:

$$\langle xy \rangle_s \approx \langle x \rangle_s \langle y \rangle_s = x_s y_s$$

(the last expression is a convenient short-hand for expressing the short-term average of the relevant variables).

Thus, we separately compute these averages on each neuron, and then multiply the averages when it is time to compute synaptic weight changes – in contrast, computing the average of the product requires integrating the average at every synapse – there are typically many many more synapses than neurons, so this represents a significant computational savings. Furthermore, the learning results (i.e., the ability of networks to learn efficiently) are typically somewhat better computing these averages separately, for reasons that are not entirely clear at this point.

The default time constant for the long-term running-average activation value used as a floating threshold in the BCM learning rule is tau = 10, which means that it integrates over roughly 10 trials – this is fairly fast relative to some expectations of "long term", but that does tend to work better computationally when combined with error-driven learning, to keep the floating threshold more fluid and responsive to the current state of the neuron. Better biological data is needed to pin down the appropriate constraints on this parameter, to see if this is biologically plausible or not.

Part II: Chapter 5: Brain Areas

In Part I of this book, we have developed a toolkit of basic neural mechanisms, going from the activation dynamics of individual neurons, to networks of neurons, and the learning mechanisms that configure them in both self-organizing and error-driven ways. At this start of Part II, we begin the transition to exploring a wide range of cognitive phenomena. As an important foundational step along this path, this chapter attempts to provide a big picture view of the overall functional organization of the brain, in a relatively non-controversial way that is roughly meant to correspond to what is generally agreed upon in the literature. This should help you understand at a broad level how different brain areas work together to perform different cognitive functions, and situate the more specific models in the subsequent chapters into a larger overall framework.

We proceed in the same sequence as the subsequent chapters, which roughly follows the evolutionary trajectory of the brain itself, starting with basic perceptual and motor systems, and then proceeding to explore different forms of learning and memory (including the role of the hippocampus in episodic memory). Building upon these core capacities, we then examine language and executive function, which build upon and extend the functionality of these basic cognitive systems.

As usual, we begin with a basic foundation in biology: the gross anatomy of the brain.

Navigating the Functional Anatomy of the Brain

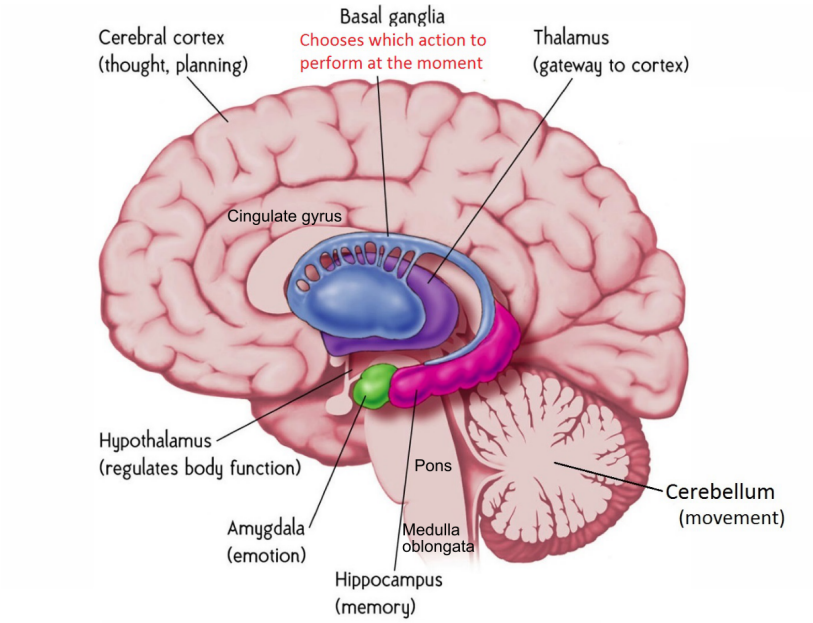


Figure 5.1: Gross anatomy of the brain. Left panel shows the major lobes of the outer neocortex layer of the brain, and right panel shows some of the major brain areas internal to the neocortex.

Figure 5.1 shows the "gross" (actually quite beautiful and amazing!) anatomy of the brain. The outer portion is the "wrinkled sheet" (upon which our thoughts rest) of the **neocortex**, showing all of the major lobes. This is where most of our complex cognitive function occurs, and what we have been focusing on to this point in the text. The rest of the brain lives inside the neocortex, with some important areas shown in the figure. These are generally referred to as **subcortical** brain areas, and we include some of them in our computational models, including:

- **Hippocampus** – this brain area is actually an "ancient" form of cortex called "archicortex", and we'll see in the *Memory* Chapter how it plays a critical role in learning new "everyday" memories about events and facts (called *episodic* memories).
- **Amygdala** – this brain area is important for recognizing emotionally salient stimuli, and alerting the rest of the brain about them. We'll explore it in the *Motor Control and Reinforcement Learning*

Chapter, where it plays an important role in reinforcing motor (and cognitive) actions based on reward (and punishment).

- **Cerebellum** – this massive brain structure contains 1/2 of the neurons in the brain, and plays an important role in motor coordination. It is also active in most cognitive tasks, but understanding exactly what its functional role is in cognition remains somewhat elusive. We'll explore it in the *Motor Control and Reinforcement Learning* Chapter.
- **Thalamus** – provides the primary pathway for sensory information on its way to the neocortex, and is also likely important for attention, arousal, and other modulatory functions. We'll explore the role of visual thalamus in the *Perception and Attention* Chapter and of motor thalamus in the *Motor Control and Reinforcement Learning* Chapter.
- **Basal Ganglia** – this is a collection of subcortical areas that plays a critical role in the *Motor Control and Reinforcement Learning* Chapter, and also in *Executive Function* Chapter. It helps to make the final "Go" call on whether (or not) to execute particular actions that the cortex 'proposes', and whether or not to update cognitive plans in the prefrontal cortex. Its policy for making these choices is learned based on their prior history of reinforcement/punishment.

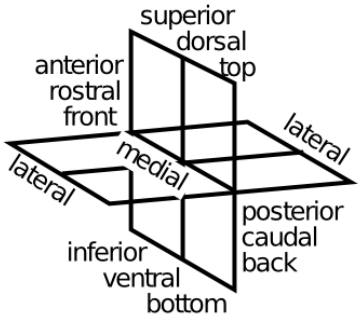


Figure 5.2: Terminology for referring to different parts of the brain – for everything except lateral and medial, three different terms for the same thing are given.

Figure 5.2 shows the terminology that anatomist's use to talk about different parts of the brain – it is a good idea to get familiar with these terms – we'll put them to good use right now.

Figures 5.3 and 5.4 show more detail on the structure of the neocortex, in terms of **Brodmann areas** – these areas were identified by Korbinian Brodmann on the basis of anatomical differences (principally the differences in thickness of different cortical layers, which we covered in the *Networks* Chapter. We won't refer too much to things at this level of detail, but learning some of these numbers is a good idea for being able to read the primary literature in cognitive neuroscience. Here is a quick overview of the functions of the cortical lobes (Figure 5.5):

- **Occipital lobe** – this contains **primary visual cortex (V1)** (Brodmann's area 17 or BA17), located at the very back tip of the neocortex, and higher-level visual areas that radiate out (forward) from it. Clearly, its main function is in visual processing.
- **Temporal lobe** – departing from the occipital lobe, the **what** pathway of visual processing dives down into **inferotemporal cortex (IT)**, where visual objects are recognized. Meanwhile, superior temporal cortex contains **primary auditory cortex (A1)**, and associated higher-level auditory and **language-processing** areas. Thus, the temporal lobes (one on each side) are where the visual appearance of objects gets translated into verbal labels (and vice-versa), and also where we learn to read. The most anterior region of the temporal lobes appears to be important for **semantic knowledge** – where all your high-level understanding of things like lawyers and government and all that good stuff you learn in school. The **medial temporal lobe** (MTL) area transitions into the hippocampus, and areas here play an increasingly important role in storing and retrieving memories of life events (**episodic memory**). When you are doing rote memorization without deeper semantic learning, the MTL and hippocampus are hard at work. Eventually, as you learn things more deeply and systematically, they get encoded in the anterior temporal cortex (and other brain areas too). In summary, the temporal lobes contain a

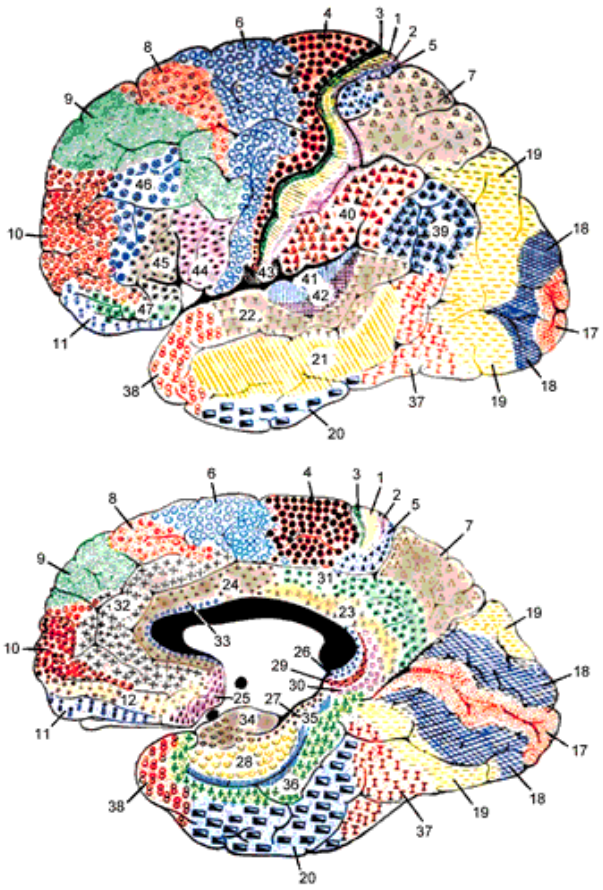


Figure 5.3: Brodmann's numbering system for the different areas of the neocortex, based on anatomical distinctions such as the thickness of different cortical layers, as we discussed in the Networks Chapter. These anatomical distinctions are remarkably well correlated with the functional differences in what different brain areas do.

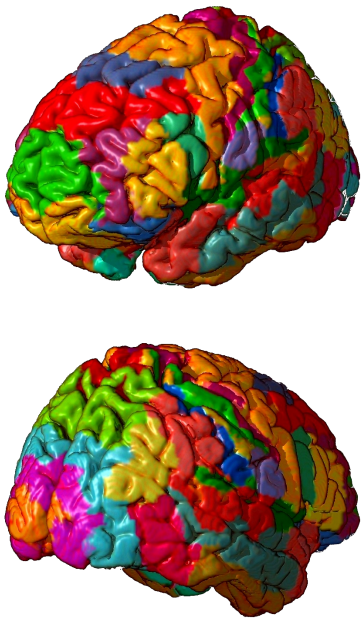


Figure 5.4: Color delineated 3D map of Brodmann areas on the external cortical surface. *Top:* anterior view. *Bottom:* posterior view.

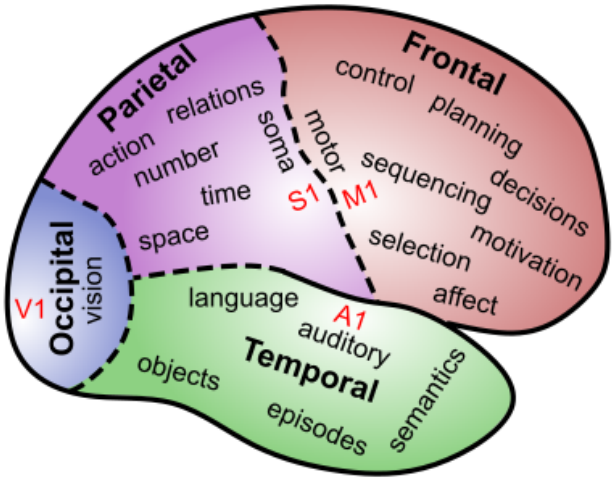


Figure 5.5: Summary of functions of cortical lobes – see text for details.

huge amount of the stuff that we are consciously aware of – facts, events, names, faces, objects, words, etc. One broad characterization is that temporal cortex is good at **categorizing** the world in myriad ways.

- **Parietal lobe** – in contrast to temporal lobe, the parietal lobe is much murkier and subconscious. It is important for encoding spatial locations (i.e., the **where** pathway, in complement to the IT what pathway), and damage to certain parts of parietal gives rise to the phenomenon of hemispatial neglect – people just forget about an entire half of space! But its functionality goes well beyond mere spatial locations. It is important for encoding **number, mathematics, abstract relationships**, and many other "smart" things. At a more down-to-earth level, parietal cortex provides the major pathway where visual information can **guide motor actions**, leading it to be characterized as the **how** pathway. It also contains the **primary somatosensory cortex (S1)**, which is important for guiding and informing motor actions as well. In some parts of parietal cortex, neurons serve to translate between different **frames of reference,** for example converting spatial locations on the body (from somatosensation) to visual coordinates. And visual information can be encoded in terms of the patterns of activity on the retinal (retinotopic coordinates), or head, body, or environment-based reference frames. One broad characterization of parietal cortex is that it is specialized for processing **metrical** information – things that vary along a continuum, in direct contrast with the discrete, categorical nature of temporal lobe. A similar distinction is popularly discussed in terms of left vs. right sides of the brain, but the evidence for this in terms of temporal vs. parietal is stronger overall.
- **Frontal lobe** – this starts at the posterior end with **primary motor cortex (M1)**, and moving forward, there is a hierarchy of higher-levels of motor control, from low level motor control in M1 and supplementary motor areas (SMA), up to higher-level action sequences and contingent behavior encoded in premotor areas (higher motor areas). Beyond this is the **prefrontal cortex (PFC)**, known as the brain's **executive** – this is where all the high-level shots are called – where your big plans are sorted out and influenced by basic motivations and emotions, to determine what you really should do next. The PFC also has a posterior-anterior organization, with more anterior areas encoding higher-level, longer-term plans and goals. The most anterior area of PFC (the **frontal pole**) seems to be particularly important for the most abstract, challenging forms of cognitive reasoning – when you're really trying hard to figure out a puzzle, or sort through those tricky questions on the GRE or an IQ test. The medial and ventral regions of the frontal cortex are particularly important for **emotion and motivation** – for example the **orbital frontal cortex (OFC)** seems to be important for maintaining and manipulating information about how rewarding a given stimulus or possible outcome might be (it receives a strong input from the Amygdala to help it learn and represent this information). The **anterior cingulate cortex (ACC)** is important for encoding the consequences of your actions, including the difficulty, uncertainty, or likelihood of failure associated with prospective actions in the current state (it lights up when you look down that double-black diamond run at the ski area!). Both the OFC and the ACC can influence choices via interactions with other frontal motor plan areas, and also via interactions with the basal ganglia. The **ventromedial PFC (VMPFC)** interacts with a lot of subcortical areas, to control basic bodily functions like heart rate, breathing, and neuromodulatory areas that then influence the brain more broadly (e.g., the ventral tegmental area (VTA) and locus coeruleus (LC), which release **dopamine** and **norepinephrine**, both of which have broad effects all over the cortex, but especially back in frontal cortex). The biggest mystery about the frontal lobe is how to understand how it does all of these amazing things, without using terms like "executive", because we're pretty sure you don't have a little guy in a pinstripe suit sitting in there. It is all just neurons!

Comparing and Contrasting Major Brain Areas

Table 5.1 Comparison of learning mechanisms and activity/representational dynamics across four primary areas of the brain. +++ means that the area definitely has given property, with fewer +'s indicating less confidence in and/or importance of this feature. — means that the area definitely does not have the given property, again with fewer -'s indicating lower confidence or importance.

	Learning Signal			*Dynamics*		
Area	Reward	Error	Self Org	Separator	Integrator	Attractor
Basal Ganglia	+++	—	—	++	-	—
Cerebellum	—	+++	—	+++	—	—
Hippocampus	+	+	+++	+++	—	+++
Neocortex	++	+++	++	—	+++	+++

Table 5.1 shows a comparison of four major brain areas according to the learning rules and activation dynamics that they employ. The evolutionarily older areas of the basal ganglia, cerebellum, and hippocampus employ a separating form of activation dynamics, meaning that they tend to make even somewhat similar inputs map onto more separated patterns of neural activity within the structure. This is a very conservative, robust strategy akin to "memorizing" specific answers to specific inputs – it is likely to work OK, even though it is not very efficient, and does not generalize to new situations very well. Each of these structures can be seen as optimizing a different form of learning within this overall separating dynamic. The basal ganglia are specialized for learning on the basis of reward expectations and outcomes. The cerebellum uses a simple yet effective form of error-driven learning (basically the delta rule as discussed in the *Learning* Chapter. And the hippocampus relies more on hebbian-style self-organizing learning. Thus, the hippocampus is constantly encoding new episodic memories regardless of error or reward (though these can certainly modulate the rate of learning, as indicated by the weaker + signs in the table), while the basal ganglia is learning to select motor actions on the basis of potential reward or lack thereof (and is also a control system for regulating the timing of action selection), while the cerebellum is learning to swiftly perform those motor actions by using error signals generated from differences in the sensory feedback relative to the motor plan. Taken together, these three systems are sufficient to cover the basic needs of an organism to survive and adapt to the environment, at least to some degree.

The hippocampus does introduce one critical innovation beyond what is present in the basal ganglia and cerebellum: it has attractor dynamics. Specifically the recurrent connections between CA3 neurons are important for retrieving previously-encoded memories, via pattern completion as we explored in the *Networks* Chapter. The price for this innovation is that the balance between excitation and inhibition must be precisely maintained, to prevent epileptic activity dynamics. Indeed, the hippocampus is the single most prevalent source of epileptic activity, in people at least.

Against this backdrop of evolutionarily older systems, the neocortex represents a few important innovations. In terms of activation dynamics, it builds upon the attractor dynamic innovation from the hippocampus (appropriately so, given that hippocampus represents an ancient "proto" cortex), and adds to this a strong ability to develop representations that integrate across experiences to extract generalities, instead of always keeping everything separate all the time. The cost for this integration ability is that the system can now form the wrong kinds of generalizations, which might lead to bad overall behavior. But the advantages apparently outweigh the risks, by giving the system a strong ability to apply previous learning to novel situations. In terms of learning mechanisms, the neocortex employs a solid blend of all three major forms of learning, integrating the best of all the available learning signals into one system.

Perception and Attention: What vs. Where

The perceptual system provides an excellent example of the power of hierarchically organized layers of neural detectors, as we discussed in the *Networks* Chapter. Figure 5.6 summarizes this process, with associated cortical areas noted below each stage of processing. Figure 5.7 shows the current best estimate of the actual anatomical connectivity patterns of all of the major visual areas (Markov et al. 2014; Felleman and Van Essen 1991), showing that information really is processed in a hierarchical fashion in the brain (although

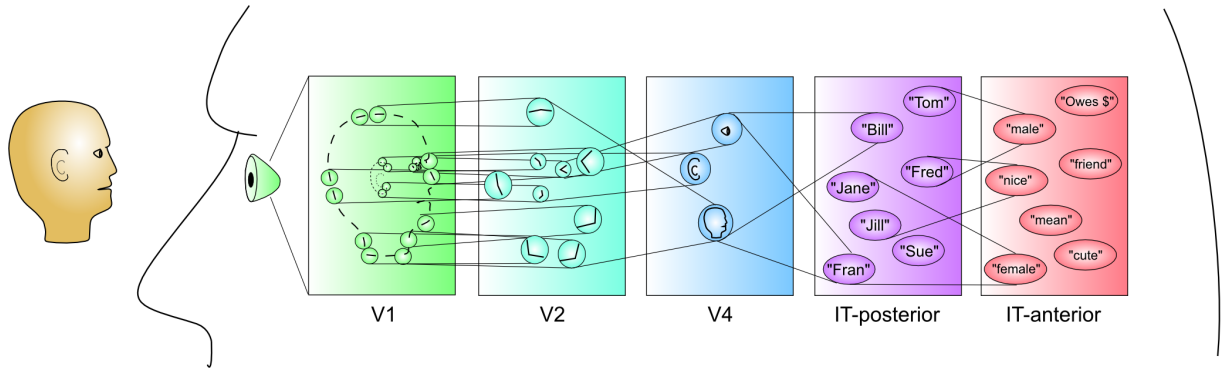


Figure 5.6: Hierarchy of visual detectors of increasing complexity achieves sophisticated perceptual categorization, with the higher levels being able to recognize 1000's of different objects, people, etc.

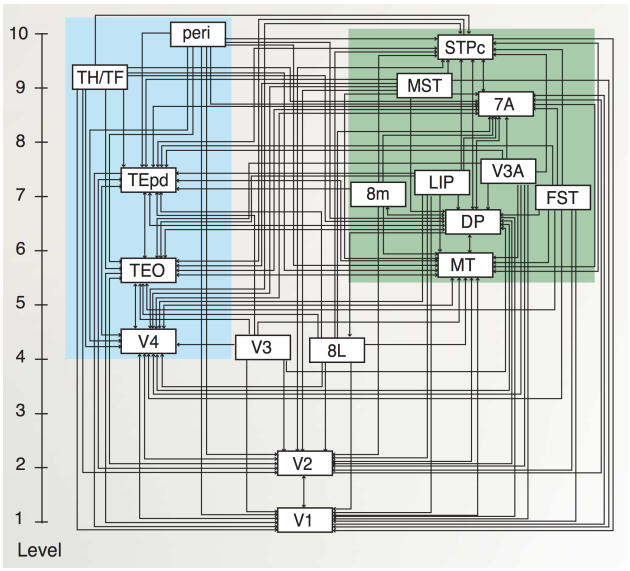


Figure 5.7: Updated and simplified version of Felleman & Van Essen's (1991) diagram of the anatomical connectivity of visual processing pathways, starting with primary visual cortex (V1) and on up. The blue-shaded areas comprise the ventral *What* pathway, and green-shaded are the dorsal *Where*. Reproduced from Markov et al., (2014).

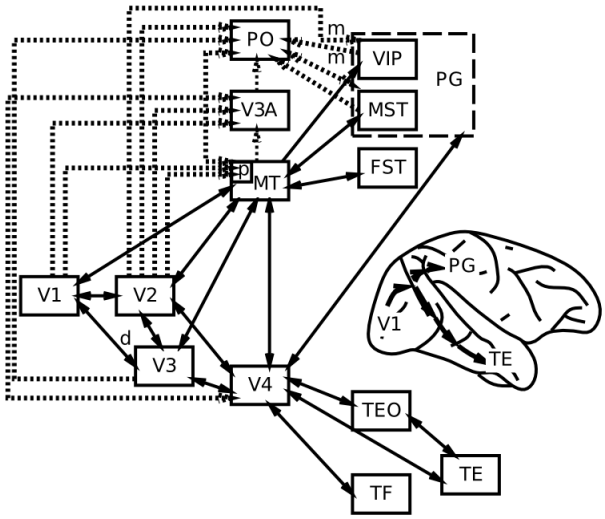


Figure 5.8: Division of *What* vs *Where* (ventral vs. dorsal) pathways in visual processing, based on the classic Ungerlieder and Mishkin (1982) framework.

there are many interconnections outside of a strict hierarchy as well). Figure 5.8 puts these areas into their anatomical locations, showing more clearly the **what vs where** (**ventral vs dorsal**) split in visual processing (Ungerleider and Mishkin 1982). Here is a quick summary of the flow of information up the **what** side of the visual pathway (pictured on the left side of Figure 5.7):

- **V1** – primary visual cortex, which encodes the image in terms of oriented edge detectors that respond to edges (transitions in illumination) along different angles of orientation. We will see in the *Perception and Attention* Chapter how these edge detectors develop through self-organizing learning, driven by the reliable statistics of natural images.
- **V2** – secondary visual cortex, which encodes combinations of edge detectors to develop a vocabulary of intersections and junctions, along with many other basic visual features (e.g., 3D depth selectivity, basic textures, etc), that provide the foundation for detecting more complex shapes. These V2 neurons also encode these features in a broader range of locations, starting a process that ends up with IT neurons being able to recognize an object regardless of where it appears in the visual field (i.e., *invariant* object recognition).
- **V4** – detects more complex shape features, over an even larger range of locations (and sizes, angles, etc).
- **IT-posterior** (PIT) – detects entire object shapes, over a wide range of locations, sizes, and angles. For example, there is an area near the fusiform gyrus on the bottom surface of the temporal lobe, called the **fusiform face area (FFA)**, that appears especially responsive to faces. As we saw in the *Networks* Chapter, however, objects are encoded in distributed representations over a broad range of areas in IT.
- **IT-anterior** (AIT) – this is where visual information becomes extremely abstract and **semantic** in nature – as shown in the Figure, it can encode all manner of important information about different people, places and things.

We'll explore a model of invariant object recognition in the *Perception and Attention* Chapter that shows how this deep hierarchy of detectors can develop through learning. The *Language* Chapter builds upon this object recognition process to understand how words are recognized and translated into associated verbal motor outputs during reading, and associated with semantic knowledge as well.

The **where** aspect of visual processing going up in a dorsal directly through the parietal cortex (areas MT, VIP, LIP, MST) contains areas that are important for processing motion, depth, and other spatial features. As noted above, these areas are also critical for translating visual input into appropriate motor output, leading Goodale and Milner to characterize this as the **how** pathway (Goodale and Milner 1992).

In the *Perception and Attention* Chapter, we'll see how this dorsal pathway can interact with the ventral what pathway in the context of visual attention, producing the characteristic effects of parietal damage in hemispatial neglect, for example. There is also increasing evidence for three distinct parietal-lobe pathways, corresponding to *looking* (LIP / FEF), *reaching* (VIP / SMA), and *navigating* (medial parietal networks, including *posterior cingulate cortex (PCC)* and *retrosplenial cortex (RSC)*) (Kravitz et al. 2011; Ranganath and Ritchey 2012).

Motor Control: Parietal and Motor Cortex Interacting with Basal Ganglia and Cerebellum

Carrying the parietal **how** pathway forward, visual information going along the dorsal pathway through the parietal cortex heads directly into the frontal cortex, where it can drive motor neurons in primary motor cortex, which can directly drive the muscles to produce overt motor actions. This completes the critical sensory-motor loop that lies at the core of all behavior. Motor control also critically involves many subcortical brain areas, including the basal ganglia and cerebellum. The rough division of labor between these areas is:

- **Neocortex (parietal to frontal)** – does high-level metrical processing of sensory information, integrating multiple modalities and translating between different reference frames as necessary, to arrive at a **range of possible responses to the current sensory environment.**
- **Basal Ganglia** – receives both sensory inputs and the potential responses being "considered" in frontal cortex, and can then trigger a **disinhibitory Go** signal that enables the *best* of the possible actions to get over threshold and actually drive behavior (Mink 1996; Frank 2005). This process of **action selection** is shaped by **reinforcement learning** – the basal ganglia are bathed in **dopamine**, which drives learning in response to rewards and punishments, and also influences the speed of the selection process itself (Sutton and Barto 1998; Montague, Dayan, and Sejnowski 1996). Thus, the basal ganglia selects the action that is most likely to result in reward, and least likely to result in punishment. The **amygdala** plays a key role in driving these dopamine signals in response to sensory cues associated with reward and punishment.
- **Cerebellum** – is richly interconnected with the parietal and motor cortex, and it is capable of using a simple yet powerful form of error-driven learning to acquire high-resolution metrical maps between sensory inputs and motor outputs. Thus, it is critical for generating smooth, coordinated motor movements that properly integrate sensory and motor feedback information to move in an efficient and controlled manner. It also likely serves to teach the parietal and motor cortex what it has learned.

In the *Motor Control and Reinforcement Learning* Chapter, we will see how dopamine signals shape basal ganglia learning and performance in a basic action selection task. Then, we'll explore a fascinating model of cerebellar motor learning in a virtual robot that performs coordinated eye and head movements to fixate objects – this model shows how the error signals needed for cerebellar learning can arise naturally.

Interestingly, all of these "low level" motor control systems end up being co-opted by "higher level" executive function systems (e.g., the prefrontal cortex), so although some don't think of motor control as a particularly cognitive domain, it actually provides a solid foundation for understanding some of the highest levels of cognitive function!

Memory: Temporal Cortex and the Hippocampus

When you think of memory, probably things like "what did I have for dinner last night?" and "how can I remember people's names better?" tend to come to mind. These represent just one category of memory, however. Indeed, memory is ubiquitous in neural networks – every synapse has the capacity for storing memory, and any given "memory" requires the coordinated actions of millions of such synapses to encode and retrieve. There are many taxonomies of memory, but really the only one you need to know is identical to the functional organization of the brain being provided here. Memory is embedded in every brain area, and the nature of that memory is intimately tied up with what that area does. Motor cortex learns motor memories. Parietal cortex learns things like motor skills – how to hit a baseball (hint: keep your eye on the ball – parietal cortex needs visual input!).

There is one brain area, however, that looms so large in the domain of memory, that we'll spend a

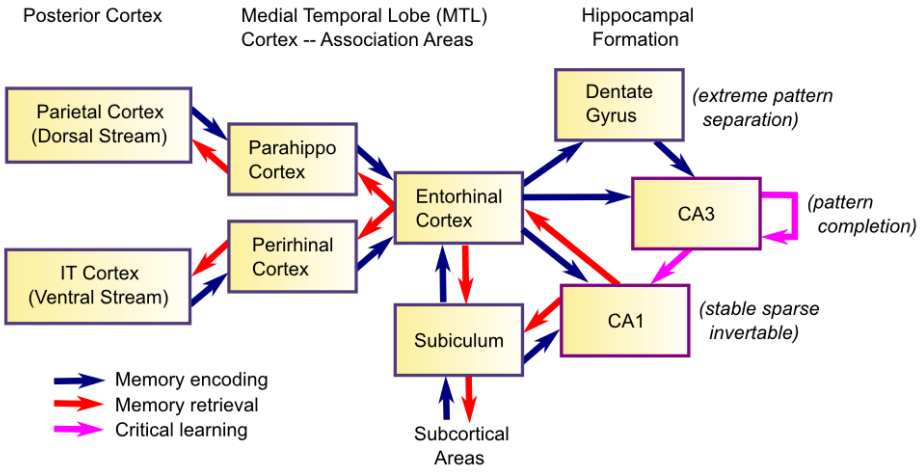


Figure 5.9: The hippocampus sits on "top" of the cortical hierarchy and can encode information from all over the brain, binding it together into an episodic memory.

while focusing on it. This is the **hippocampus**, which seems to be particularly good at rapidly learning new information, in a way that doesn't interfere too much with previously learned information (Figure 5.9). When you need to remember the name associated with a person you recently met, you're relying on this rapid learning ability of the hippocampus. We'll see that the neural properties of the hippocampal system are ideally suited to producing this rapid learning ability. One key neural property is the use of **extremely sparse representations**, which produce a phenomenon called **pattern separation**, where the neural activity pattern associated with one memory is highly distinct from that associated with other similar memories (Marr 1971; McClelland, McNaughton, and O'Reilly 1995). This is what minimizes interference with prior learning – interference arises as a function of overlap. We'll see how this pattern separation process is complemented by a pattern completion process for recovering memories during retrieval from partial information (O'Reilly and McClelland 1994).

We'll also see how the **learning rate** plays a crucial role in learning. Obviously, to learn rapidly, you need a fast learning rate. But what happens with a slow learning rate? Turns out this enables you to integrate across many different experiences, to produce *wisdom* and *semantic knowledge*. This slower learning rate is characteristic of most of the neocortex (it also enables the basal ganglia to learn probabilities of positive and negative outcomes for each action across a range of experience, rather than just their most recent outcomes) (McClelland, McNaughton, and O'Reilly 1995). Interestingly, even with a slow learning rate, neocortex can exhibit measurable effects of a single trial of learning, in the form of **priming effects** and **familiarity signals** that can drive recognition memory (i.e., your ability to recognize something as familiar, without any more explicit episodic memory). This form of recognition memory seems to depend on **medial temporal lobe (MTL)** areas including **perirhinal cortex**. Another form of one trial behavioral learning involves mechanisms that support active maintenance of memories in an attractor state (working memory in the prefrontal cortex). This form of memory does not require a weight change at all, but can nevertheless rapidly influence behavioral performance from one instance to the next.

Language: All Together Now

Language taps the coordinated function of many of the brain areas discussed above. Language requires highly sophisticated perceptual abilities, to be able to discriminate different speech sounds and different letters and combinations thereof (just listen and look at an unfamiliar foreign language to experience how amazing your own native language perceptual abilities are, which you take for granted). Likewise, sophisticated motor output abilities are required to produce the sounds and write the letters and words of language. In between, language requires some of the most demanding forms of cognitive processing, to keep track of the grammatical and semantic information streaming past you, often at high speed. This requires sophisticated executive function

and working memory abilities, in addition to powerful distributed posterior-cortical semantic representations to integrate all the semantic information.

Our exploration in the *Language* Chapter starts with a small-scale model of reading, that interconnects **orthographic** (writing), **phonological** (speech), and **semantic** representations of individual words, to form a **distributed lexicon** – there isn't one place where all word information is stored – instead it is distributed across brain areas that are specialized for processing the relevant perceptual, motor, and semantic information. Interestingly, we can simulate various forms of **acquired dyslexia** (e.g., from stroke or other forms of brain injury) by damaging specific pathways in this model, providing an important way of establishing neural correlates of language function in humans, where invasive experiments are not possible (Plaut and Shallice 1993).

We then zoom in on the orthography to phonology pathway to explore issues with **regularities and exceptions** in this spelling-to-sound mapping, which has been the topic of considerable debate (Seidenberg and McClelland 1989; Pinker and Prince 1988; Plaut et al. 1996). We show that the object recognition model from the perception chapter has an important blend of features that support both regular and exception mappings, and this model pronounces nonword probe inputs much like people do, demonstrating that it has extracted similar underlying knowledge about the English mapping structure.

Next, we zoom in on the semantics pathway, exploring how a self-organizing network can learn to encode statistical regularities in **word co-occurance**, that give rise to semantic representations that are remarkably effective in capturing the similarity structure of words (Landauer and Dumais 1997). We train this network on an early draft of the first edition of this text, so you should be familiar with the relevant semantics!

Finally, we tackle the interactions between syntax and semantics in the context of processing the meaning of sentences, using the notion of a **sentence gestalt** representation, that uses coarse-coded distributed representations to encode the overall meaning of the sentence, integrating both syntactic and semantic cues (St John and McClelland 1990). This is a distinctly neural approach to syntax, as contrasted with the symbolic, highly structured approaches often employed in linguistic theories.

Executive Function: Prefrontal Cortex and Basal Ganglia

Finally, we build upon the motor control functions of frontal cortex and basal ganglia to understand how these two areas interact to support high-level executive function.

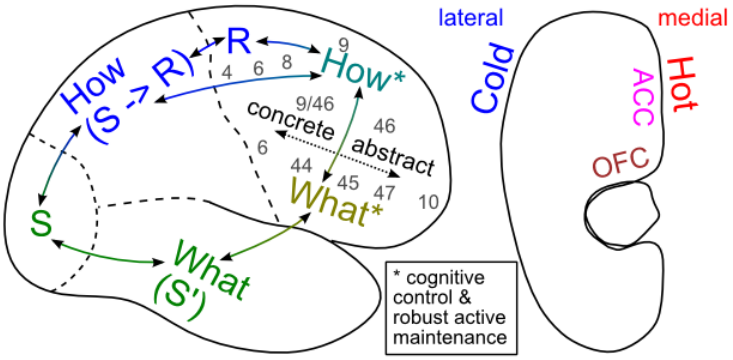


Figure 5.10: The *What* vs. *How* distinction for posterior cortex can be carried forward into prefrontal cortex, to understand the distinctive roles of the ventral and dorsal areas of PFC.

We also build upon the functional divisions of the posterior cortex to understand how the ventral vs. dorsal areas of prefrontal cortex are functionally organized. Figure 5.10 shows an overall schematic for how this occurs. It also illustrates how the lateral surface is more associated with "cold" cognitive function, while the medial surface is more involved in "hot" emotional and motivational processing.

We'll see how the PFC can provide **top-down cognitive control** over processing in the posterior cortex, with the classic example being the Stroop task.

Then we'll explore how PFC and BG can interact to produce a dynamically gated working memory

system that allows the system to hold multiple pieces of information 'in mind', and to separately update some pieces of information while continuing to maintain some existing information. The role of the BG in this system builds on the more established role of the BG in motor control, by interacting in very similar circuits with PFC instead of motor cortex. In both cases, the BG provide a gating signal for determining whether or not a given frontal cortical 'action' should be executed or not. It's just that PFC actions are more cognitive than motor cortex, and include things like updating of working memory states, or of goals, plans, etc. Once updated, these PFC representations can then provide that top-down cognitive control mentioned above, and hence can shape action selection in BG-motor circuits, but also influence attention to task-relevant features in sensory cortex. Interestingly, the mechanisms for reinforcing which cognitive actions to execute (including whether or not to update working memory, or to attend to particular features, or to initiate a high level plan) seem to depend on very similar dopaminergic reinforcement learning mechanisms that are so central to motor control. This framework also provides a link between motivation and cognition which is very similar to the well established link between motivation and action.

Chapter 6: Perception and Attention

Perception is at once obvious and mysterious. It is so effortless to us that we have little appreciation for all the amazing computation that goes on under the hood. And yet we often use terms like "vision" as a metaphor for higher-level concepts (does the President have a vision or not?) – perhaps this actually reflects a deep truth: that much of our higher-level cognitive abilities depend upon our perceptual processing systems for doing a lot of the hard work. Perception is not the mere act of seeing, but is leveraged whenever we imagine new ideas, solutions to hard problems, etc. Many of our most innovative scientists (e.g., Einstein, Richard Feynman) used visual reasoning processes to come up with their greatest insights. Einstein tried to visualize catching up to a speeding ray of light (in addition to trains stretching and contracting in interesting ways), and one of Feynman's major contributions was a means of visually diagramming complex mathematical operations in quantum physics.

Pedagogically, perception serves as the foundation for our entry into cognitive phenomena. It is the most well-studied and biologically grounded of the cognitive domains. As a result, we will cover only a small fraction of the many fascinating phenomena of perception, focusing mostly on vision. But we do focus on a core set of issues that capture many of the general principles behind other perceptual phenomena.

We begin with a computational model of **primary visual cortex (V1)**, which shows how self-organizing learning principles can explain the origin of oriented edge detectors, which capture the dominant statistical regularities present in natural images. This model also shows how excitatory lateral connections can result in the development of **topography** in V1 – neighboring neurons tend to encode similar features, because they have a tendency to activate each other, and learning is determined by activity.

Building on the features learned in V1, we explore how higher levels of the **ventral what** pathway can learn to **recognize objects** regardless of considerable variability in the superficial appearance of these objects as they project onto the retina. Object recognition is the paradigmatic example of how a hierarchically-organized sequence of feature category detectors can incrementally solve a very difficult overall problem. Computational models based on this principle can exhibit high levels of object recognition performance on realistic visual images, and thus provide a compelling suggestion that this is likely how the brain solves this problem as well.

Next, we consider the role of the **dorsal where** (or how) pathway in **spatial attention**. Spatial attention is important for many things, including object recognition when there are multiple objects in view – it helps focus processing on one of the objects, while degrading the activity of features associated with the other objects, reducing potential confusion. Our computational model of this interaction between what and where processing streams can account for the effects of brain damage to the *where* pathway, giving rise to hemispatial neglect for damage to only one side of the brain, and a phenomenon called Balint's syndrome with bilateral damage. This ability to account for both neurologically intact and brain damaged behavior is a powerful advantage of using neurally-based models.

As usual, we begin with a review of the biological systems involved in perception.

Biology of Perception

Our goal in this section is to understand just enough about the biology to get an overall sense of how information flows through the visual system, and the basic facts about how different parts of the system operate. This will serve to situate the models that come later, which provide a much more complete picture of each step of information processing.

Figure 6.1 shows the basic optics and transmission pathways of visual signals, which come in through the retina, and progress to the lateral geniculate nucleus of the thalamus (LGN), and then to primary visual cortex (V1). The primary organizing principles at work here, and in other perceptual modalities and perceptual areas more generally, are:

- **Transduction of different information** – in the retina, photoreceptors are sensitive to different **wavelengths of light** (red = long wavelengths, green = medium wavelengths, and blue = short wavelengths), giving us color vision, but the retinal signals also differ in their **spatial frequency** (how coarse or fine of a feature they detect – photoreceptors in the central *fovea* region can have high spatial frequency = fine resolution, while those in the periphery are lower resolution), and in their **temporal**

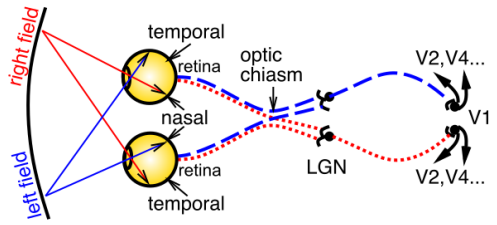


Figure 6.1: The pathway of early visual processing from the retina through lateral geniculate nucleus of the thalamus (LGN) to primary visual cortex (V1), showing how information from the different visual fields (left vs. right) are routed to the opposite hemisphere.

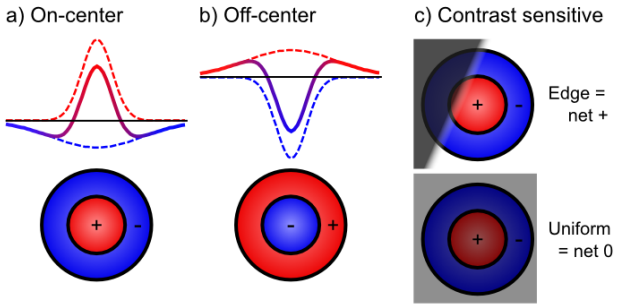


Figure 6.2: How the retina compresses information by only responding to areas of contrasting illumination, not solid uniform illumination. The response properties of retinal cells can be summarized by these Difference-of-Gaussian (DoG) filters, with a narrow central region and a wider surround (also called center-surround receptive fields). The excitatory and inhibitory components exactly cancel when both are uniformly illuminated, but when light falls more on the center vs. the surround (or vice-versa), they respond, as illustrated with an edge where illumination transitions between darker and lighter.

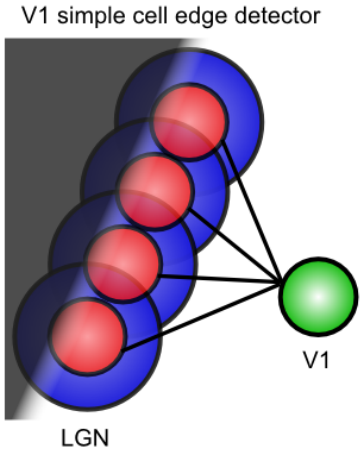


Figure 6.3: A V1 simple cell that detects an oriented edge of contrast in the image, by receiving from a line of LGN on-center cells aligned along the edge. The LGN cells will fire due to the differential excitation vs. inhibition they receive (see previous figure), and then will activate the V1 neuron that receives from them.

response (fast vs. slow responding, including differential sensitivity to motion).

- **Organization of information in a topographic fashion** – for example, the left vs. right visual fields are organized into the contralateral hemispheres of cortex – as the figure shows, signals from the left part of visual space are routed to the right hemisphere, and vice-versa. Information within LGN and V1 is also organized topographically in various ways. This organization generally allows similar information to be contrasted, producing an enhanced signal, and also grouped together to simplify processing at higher levels.
- **Extracting relevant signals, while filtering irrelevant ones** – Figure 6.2 shows how retinal cells respond only to **contrast**, not uniform illumination, by using **center-surround** receptive fields (e.g., on-center, off-surround, or vice-versa). Only when one part of this receptive field gets different amounts of light compared to the others do these neurons respond. Typically this arises with edges of contrast, where illumination transitions between light and dark, as shown in the figure – these transitions are the most informative aspects of an image, while regions of constant illumination can be safely ignored. Figure 6.3 shows how these center-surround signals (which are present in the LGN as well) can be integrated together in V1 simple cells to detect the orientation of these edges – these edge detectors form the basic vocabulary for describing images in V1. It should be easy to see how more complex shapes can then be constructed from these basic line/edge elements. V1 also contains complex cells that build upon the simple cell responses (Figure 6.4), providing a somewhat richer basic vocabulary.

The following videos show how we know what these receptive fields look like:

- Classic Hubel & Wiesel V1 receptive field mapping using old school projector stimuli: YouTube Video
- Newer reverse correlation V1 receptive field mapping: YouTube Video

In the auditory pathway, the cochlear membrane plays an analogous role to the retina, and it also has a topographic organization according to the frequency of sounds, producing the rough equivalent of a fourier transformation of sound into a spectrogram. This basic sound signal is then processed in auditory pathways to extract relevant patterns of sound over time, in much the same way as occurs in vision.

Moving up beyond the primary visual cortex, the perceptual system provides an excellent example of the power of hierarchically organized layers of neural detectors, as we discussed in the *Networks* Chapter. Figure 6.5 shows the anatomical connectivity patterns of all of the major visual areas, from V1 and on up (Markov et al. 2014; Felleman and Van Essen 1991). The specific patterns of connectivity allow a hierarchical structure to be extracted, as shown, even though there are many interconnections outside of a strict hierarchy as well.

Figure 6.6 puts these areas into their anatomical locations, showing more clearly a **what vs where** (**ventral vs dorsal**) split in visual processing (Ungerleider and Mishkin 1982). The projections going in a ventral direction from V1 to V4 to areas of **inferotemporal cortex (IT)** (TE, TEO, labeled as PIT for posterior IT in the previous figure) are important for recognizing the identity ("what") of objects in the visual input, while those going up through parietal cortex extract spatial ("where") information, including motion signals in area MT and MST. We will see later in this chapter how each of these visual streams of processing can function independently, and also interact together to solve important computational problems in perception.

Here is a quick summary of the flow of information up the **what** side of the visual pathway (pictured on the left side of Figure 6.5):

- **V1** – primary visual cortex, which encodes the image in terms of oriented edge detectors that respond to edges (transitions in illumination) along different angles of orientation. We will see in the first simulation in this chapter how these edge detectors develop through self-organizing learning, driven by the reliable statistics of natural images.
- **V2** – secondary visual cortex, which encodes combinations of edge detectors to develop a vocabulary of intersections and junctions, along with many other basic visual features (e.g., 3D depth selectivity, basic textures, etc), that provide the foundation for detecting more complex shapes. These V2 neurons also encode these features in a broader range of locations, starting a process that ends up with IT neurons being able to recognize an object regardless of where it appears in the visual field (i.e., *invariant* object recognition).
- **V4** – detects more complex shape features, over an even larger range of locations (and sizes, angles,

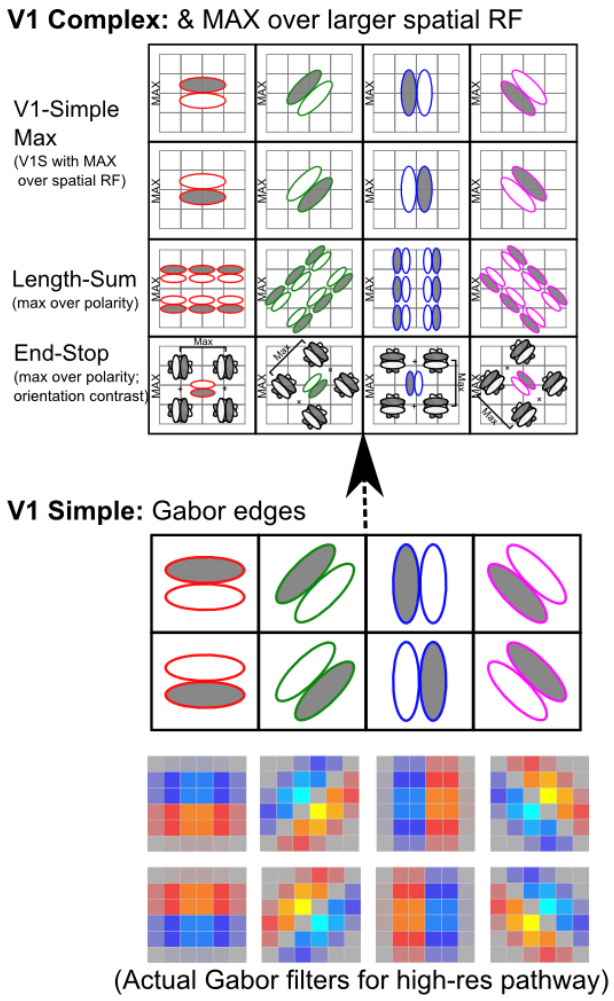


Figure 6.4: Simple and complex cell types within V1 – the complex cells integrate over the simple cell properties, including abstracting across the polarity (positions of the on vs. off coding regions), and creating larger receptive fields by integrating over multiple locations as well (the V1-Simple-Max cells are only doing this spatial integration). The end stop cells are the most complex, detecting any form of contrasting orientation adjacent to a given simple cell. In the simulator, the V1 simple cells are encoded more directly using gabor filters, which mathematically describe their oriented edge sensitivity.

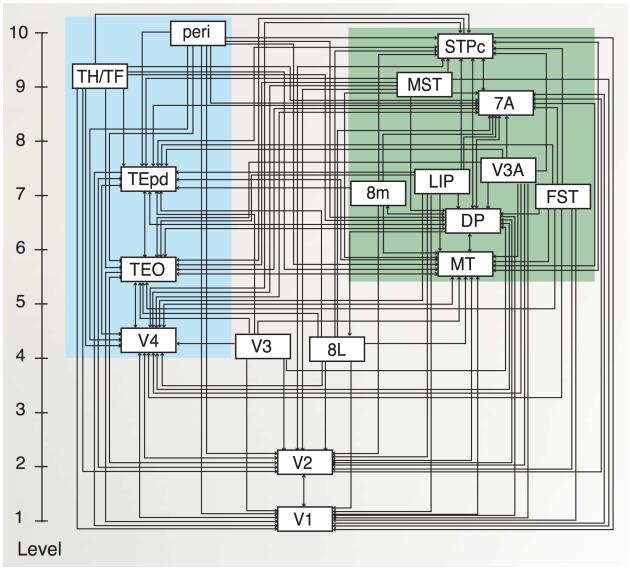


Figure 6.5: Updated and simplified version of Felleman & Van Essen's (1991) diagram of the anatomical connectivity of visual processing pathways, starting with primary visual cortex (V1) and on up. The blue-shaded areas comprise the ventral *What* pathway, and green-shaded are the dorsal *Where*. Reproduced from Markov et al., (2014).

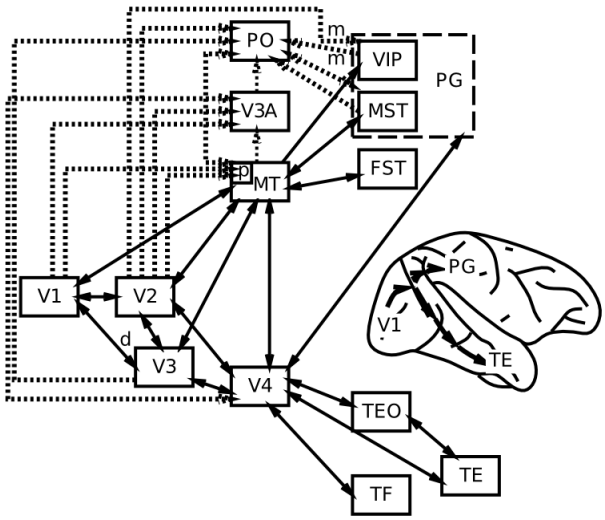


Figure 6.6: Division of *What* vs *Where* (ventral vs. dorsal) pathways in visual processing, based on the classic Ungerlieder and Mishkin (1982) framework.

etc).

- **IT-posterior** (PIT) – detects entire object shapes, over a wide range of locations, sizes, and angles. For example, there is an area near the fusiform gyrus on the bottom surface of the temporal lobe, called the **fusiform face area (FFA)**, that appears especially responsive to faces. As we saw in the *Networks* Chapter, however, objects are encoded in distributed representations over a broad range of areas in IT.
- **IT-anterior** (AIT) – this is where visual information becomes extremely abstract and **semantic** in nature – it can encode all manner of important information about different people, places and things.

In contrast, the **where** aspect of visual processing going up in a dorsal directly through the parietal cortex (areas MT, VIP, LIP, MST) contains areas that are important for processing motion, depth, and other spatial features.

Oriented Edge Detectors in Primary Visual Cortex

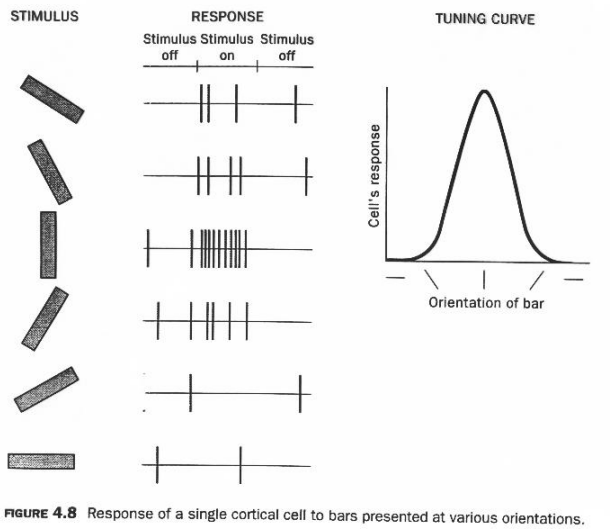


Figure 6.7: Orientation tuning of an individual V1 neuron in response to bar stimuli at different orientations – this neuron shows a preference for vertically oriented stimuli.

Neurons in primary visual cortex (V1) detect the orientation of edges or bars of light within their receptive field (RF – the region of the visual field that a given neuron receives input from). Figure 6.7 shows characteristic data from electrophysiological recordings of an individual V1 neuron in response to oriented bars. This neuron responds maximally to the vertical orientation, with a graded fall off on either side of that. This is a very typical form of **tuning curve**. Figure 6.8 shows that these orientation tuned neurons are organized **topographically**, such that neighbors tend to encode similar orientations, and the orientation tuning varies fairly continuously over the surface of the cortex.

The question we attempt to address in this section is why such a topographical organization of oriented edge detectors would exist in primary visual cortex? There are multiple levels of answer to this question. At the most abstract level, these oriented edges are the basic constituents of the kinds of images that typically fall on our retinas. These are the most obvious **statistical regularities** of natural images (Olshausen and Field 1996). If this is indeed the case, then we would expect that the self-organizing aspect of the XCAL learning algorithm used in our models (as discussed in the *Learning* Chapter) would naturally extract these statistical regularities, providing another level of explanation: V1 represents oriented edge detectors because this is what the learning mechanisms will naturally develop.

The situation here is essentially equivalent to the self organizing learning model explored in the *Learning Chapter, which was exposed to horizontal and vertical lines, and learned to represent these strong statistical regularities in the environment.

However, that earlier simulation did nothing to address the topography of the V1 neurons – why do

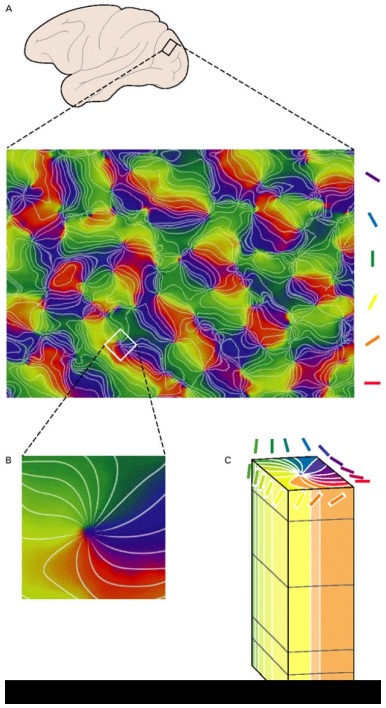


Figure 6.8: Topographic organization of oriented edge detectors in V1 – neighboring regions of neurons have similar orientation tuning, as shown in this colorized map where different colors indicate orientation preference as shown in panel C. Panel B shows how a full 360 degree loop of orientations nucleate around a central point – these are known as pinwheel structures.

neighbors tend to encode similar information? The answer we explore in the following simulation is that neighborhood-level connectivity can cause nearby neurons to tend to activate together, and because activity drives learning, this then causes them to tend to learn similar things.

Simulation Exploration

Open `v1rf` in CCN Sims to explore the development of oriented edge detectors in V1. This model gets exposed to a set of natural images, and learns to encode oriented edges, because they are the statistical regularity present in these images. Figure 6.9 shows the resulting map of orientations that develops.

Invariant Object Recognition in the *What* Pathway

Object recognition is the defining function of the ventral "what" pathway of visual processing: identifying what you are looking at. Neurons in the inferotemporal (IT) cortex can detect whole objects, such as faces, cars, etc, over a large region of visual space. This **spatial invariance** (where the neural response remains the same or invariant over spatial locations) is critical for effective behavior in the world – objects can show up in all different locations, and we need to recognize them regardless of where they appear. Achieving this outcome is a very challenging process, one which has stumped artificial intelligence (AI) researchers for a long time – in the early days of AI, the 1960's, it was optimistically thought that object recognition could be solved as a summer research project, and 50 years later we are making a lot of progress, but it remains unsolved in the sense that people are still much better than our models. Because our brains do object recognition effortlessly all the time, we do not really appreciate how hard of a problem it is.

The reason object recognition is so hard is that there can often be no overlap at all among visual inputs of the same object in different locations (sizes, rotations, colors, etc), while there can be high levels of overlap among different objects in the same location (Figure 6.10). Therefore, you cannot rely on the bottom-up visual similarity structure – instead it often works directly against the desired output categorization of these

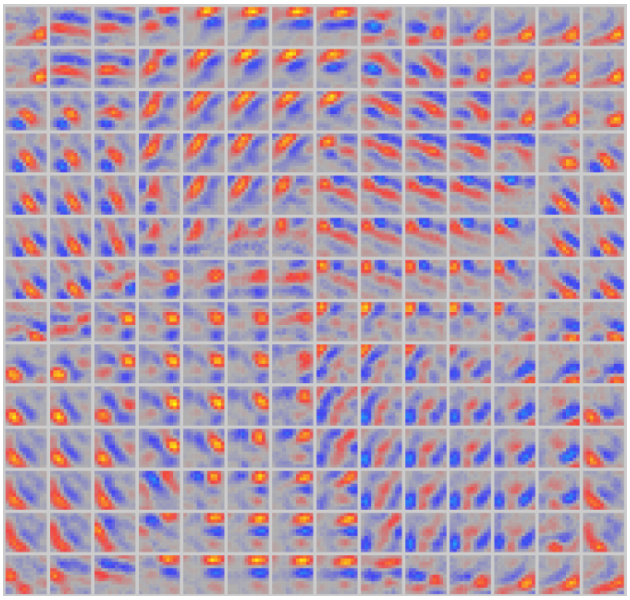


Figure 6.9: Topographic organization of oriented edge detectors in simulation of V1 neurons exposed to small windows of natural images (mountains, trees, etc). The neighborhood connectivity of neurons causes a topographic organization to develop.

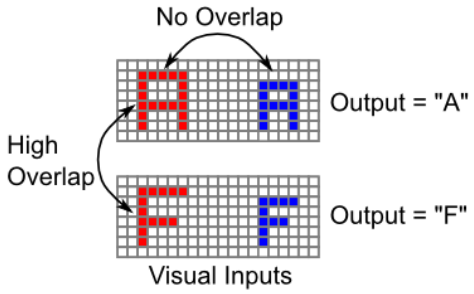


Figure 6.10: Why object recognition is hard: things that should be categorized as the same (i.e., have the same output label) often have no overlap in their retinal input features when they show up in different locations, sizes, etc, but things that should be categorized as different often have high levels of overlap when they show up in the same location. Thus, the bottom-up similarity structure is directly opposed to the desired output similarity structure, making the problem very difficult.

stimuli. As we saw in the *Learning* Chapter, successful learning in this situation requires error-driven learning, because self-organizing learning tends to be strongly driven by the input similarity structure.

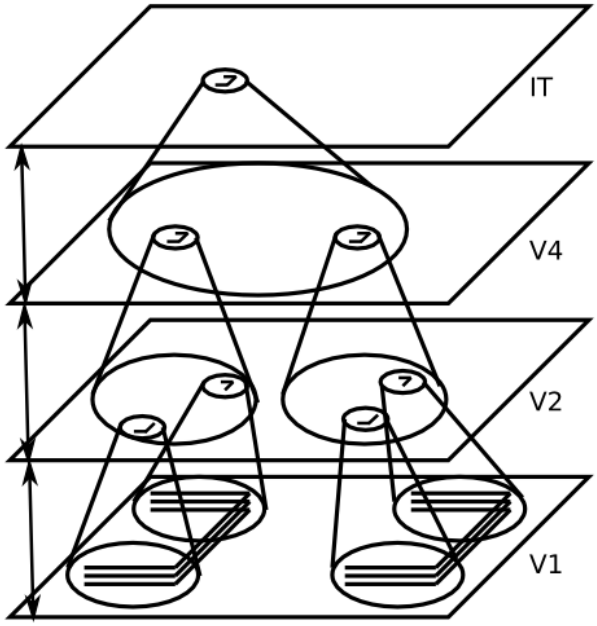


Figure 6.11: Schematic for how multiple levels of processing can result in invariant object recognition, where an object can be recognized at any location across the input. Each level of processing incrementally increases the featural complexity and spatial invariance of what it detects. Doing this incrementally allows the system to appropriately bind together features and their relationships, while also gradually building up overall spatial invariance.

The most successful approach to the object recognition problem, which was advocated initially in a model by (Fukushima 1980), is to incrementally solve two problems over a hierarchically organized sequence of layers (Figures 6.11, 6.12):

- The invariance problem, by having each layer *integrate over a range of locations* (and sizes, rotations, etc) for the features in the previous layer, such that neurons become increasingly invariant as one moves up the hierarchy.
- The pattern discrimination problem (distinguishing an A from an F, for example), by having each layer build up more complex combinations of feature detectors, as a result of detecting *combinations of the features* present in the previous layer, such that neurons are better able to discriminate even similar input patterns as one moves up the hierarchy.

The critical insight from these models is that breaking these two problems down into incremental, hierarchical steps enables the system to solve both problems without one causing trouble for the other. For example, if you had a simple fully invariant vertical line detector that responded to a vertical line in any location, it would be impossible to know what spatial relationship this line has with other input features, and this relationship information is critical for distinguishing different objects (e.g., a T and L differ only in the relationship of the two line elements). So you cannot solve the invariance problem in one initial pass, and then try to solve the pattern discrimination problem on top of that. They must be interleaved, in an incremental fashion. Similarly, it would be completely impractical to attempt to recognize highly complex object patterns at each possible location in the visual input, and then just do spatial invariance integration over locations after that. There are way too many different objects to discriminate, and you'd have to learn about them anew in each different visual location. It is much more practical to incrementally build up a "part library" of visual features that are increasingly invariant, so that you can learn about complex objects only toward the top of the hierarchy, in a way that is already spatially invariant and thus only needs to be learned once.

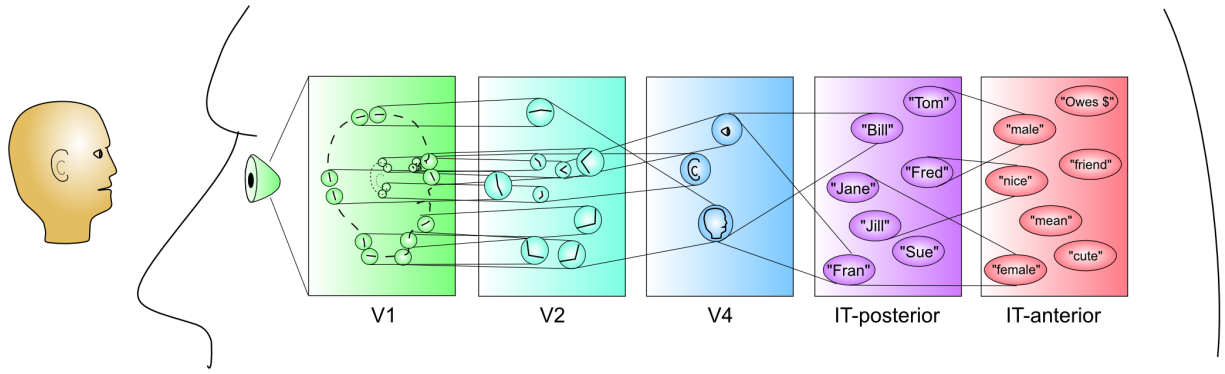


Figure 6.12: Another way of representing the hierarchy of increasing featural complexity that arises over the areas of the ventral visual pathways. V1 has elementary feature detectors (oriented edges). Next, these are combined into junctions of lines in V2, followed by more complex visual features in V4. Individual faces are recognized at the next level in IT (even here multiple face units are active in graded proportion to how similar people look). Finally, at the highest level are important functional "semantic" categories that serve as a good basis for actions that one might take – being able to develop such high level categories is critical for intelligent behavior – this level corresponds to more anterior areas of IT.

In a satisfying convergence of top-down computational motivation and bottom-up neuroscience data, this incremental, hierarchical solution provides a nice fit to the known properties of the visual areas along the ventral what pathway (V1, V2, V4, IT). Figure 6.13 summarizes neural recordings from these areas in the macaque monkey (Kobatake and Tanaka 1994), and shows that neurons increase in the complexity of the stimuli that drive their responding, and the size of the receptive field over which they exhibit an invariant response to these stimuli, as one proceeds up the hierarchy of areas. Figure 6.14 shows example complex stimuli that evoked maximal responding in each of these areas, to give a sense of what kind of complex feature conjunctions these neurons can detect.

Exploration of Object Recognition

Open up the `objrec` simulation in CCN Sims for the computational model of object recognition, which demonstrates the incremental hierarchical solution to the object recognition problem. We use a simplified set of "objects" (Figure 6.15) composed from vertical and horizontal line elements. This simplified set of visual features allows us to better understand how the model works, and also enables testing generalization to novel objects composed from these same sets of features. You will see that the model learns simpler combinations of line elements in area V4, and more complex combinations of features in IT, which are also invariant over the full receptive field. These IT representations are not identical to entire objects – instead they represent an invariant distributed code for objects in terms of their constituent features. The generalization test shows how this distributed code can support rapid learning of new objects, as long as they share this set of features. Although they are likely much more complex and less well defined, it seems that a similar such vocabulary of visual shape features are learned in primate IT representations.

Spatial Attention and Neglect in the *Where/How* Pathway

The dorsal visual pathway that goes into the parietal cortex is more heterogeneous in its functionality relative to the object recognition processing taking place in the ventral *what* pathway, which appears to be the primary function of that pathway. Originally, the dorsal pathway was described as a *where* pathway, in contrast to the ventral *what* pathway (Ungerleider and Mishkin 1982). However, (Goodale and Milner 1992) provide a compelling broader interpretation of this pathway as performing a *how* function – mapping from perception to action. One aspect of this *how* functionality involves spatial location information, in that this information is highly relevant for controlling motor actions in 3D space, but spatial information is too narrow of a definition for the wide range of functions supported by the parietal lobe. Parietal areas are important for

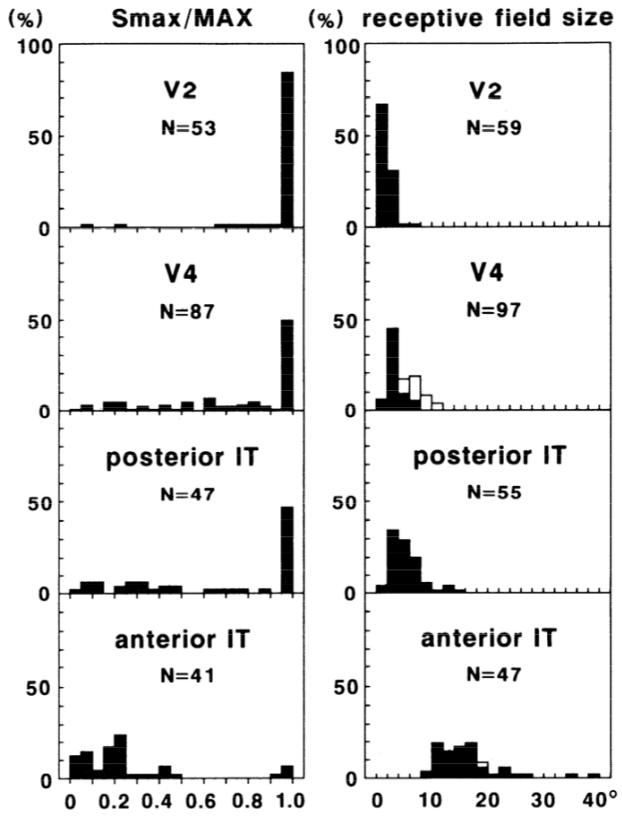


Figure 6.13: Summary of neural response properties in V2, V4, and IT for the macaque monkey, according to both the extent to which the areas respond to complex vs. simple visual features (Smax / MAX column, showing how the response to simple visual inputs (Smax) compares to the maximum response to any visual input image tested (MAX), and the overall size of the visual receptive field, over which the neurons exhibit relatively invariant responding to visual features. For V2, nearly all neurons responded maximally to simple stimuli, and the receptive field sizes were the smallest. For V4, only 50% of neurons had simple responses as their maximal response, and the receptive field sizes increase over V2. Posterior IT increases (slightly) on both dimensions, while anterior IT exhibits almost entirely complex featural responding and significantly larger receptive fields. These incremental increases in complexity and invariance (receptive field size) are exactly as predicted by the incremental computational solution to invariant object recognition as shown in the previous figure. Reproduced from Kobatake & Tanaka (1994).

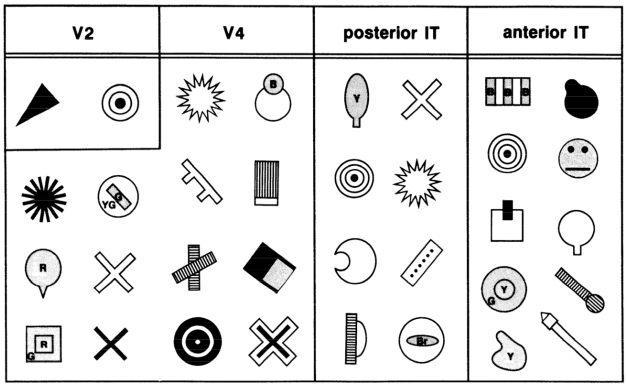


Figure 6.14: Complex stimuli that evoked a maximal response from neurons in V2, V4, and IT, providing some suggestion for what kinds of complex features these neurons can detect. Most V2 neurons responded maximally to simple stimuli (oriented edges, not shown). Reproduced from Kobatake & Tanaka (1994).

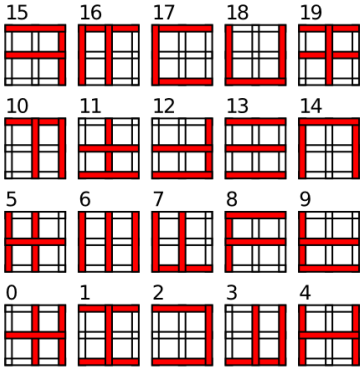


Figure 6.15: Set of 20 objects composed from horizontal and vertical line elements used for the object recognition simulation. By using a restricted set of visual feature elements, we can more easily understand how the model works, and also test for generalization to novel objects (object 18 and 19 are not trained initially, and then subsequently trained only in a relatively few locations – learning there generalizes well to other locations).

numerical and mathematical processing, and representation of abstract relationship information, for example. Areas of parietal cortex also appear to be important for modulating episodic memory function in the medial temporal lobe, and various other functions. This later example may represent a broader set of functions associated with prefrontal cortical cognitive control – areas of parietal cortex are almost always active in combination with prefrontal cortex in demanding cognitive control tasks, although there is typically little understanding of what precise role they might be playing.

In this chapter, we focus on the established *where* aspect of parietal function, and we'll take up some of the *how* functionality in the next chapter on Motor Control. Even within the domain of spatial processing, there are many cognitive functions that can be performed using parietal spatial representations, but we focus here on their role in focusing attention to spatial locations. In relation to the previous section, one crucial function of spatial attention is to enable object recognition to function in visual scenes that have multiple different objects present. For example, consider one of those "where's Waldo" puzzles () that is packed with rich visual detail. Is it possible to perceive such a scene all in one take? No. You have to scan over the image using a "spotlight" of visual attention to focus on small areas of the image, which can then be processed effectively by the object recognition pathway. The ability to direct this spotlight of attention depends on spatial representations in the dorsal pathway, which then interact with lower levels of the object recognition pathway (V1, V2, V4) to constrain the inputs to reflect only those visual features that come from within this spotlight of attention.

Hemispatial Neglect

Some of the most striking evidence that the parietal cortex is important for spatial attention comes from patients with hemispatial neglect, who tend to ignore or neglect one side of space (Figures 6.17, 6.18, 6.19). This condition typically arises from a stroke or other form of brain injury affecting the right parietal cortex, which then gives rise to a neglect of the left half of space (due to the crossing over of visual information shown in the biology section). Interestingly, the neglect applies to multiple different spatial reference frames, as shown in , where lines on the left side of the image tend to be neglected, and also each individual line is bisected more toward the right, indicating a neglect of the left portion of each line.

The Posner Spatial Cueing Task

One of the most widely used tasks to study the spotlight of spatial attention is the Posner spatial cueing task, developed by Michael Posner (Posner 1980) (Figure 6.20). One side of visual space is cued, and the effects of this cue on subsequent target detection are measured. If the cue and target show up in the same side of space (*valid* cue condition), then reaction times are faster compared to when they show up on different sides of space (*invalid* cue condition) (). This difference in reaction time (RT) suggests that spatial attention is

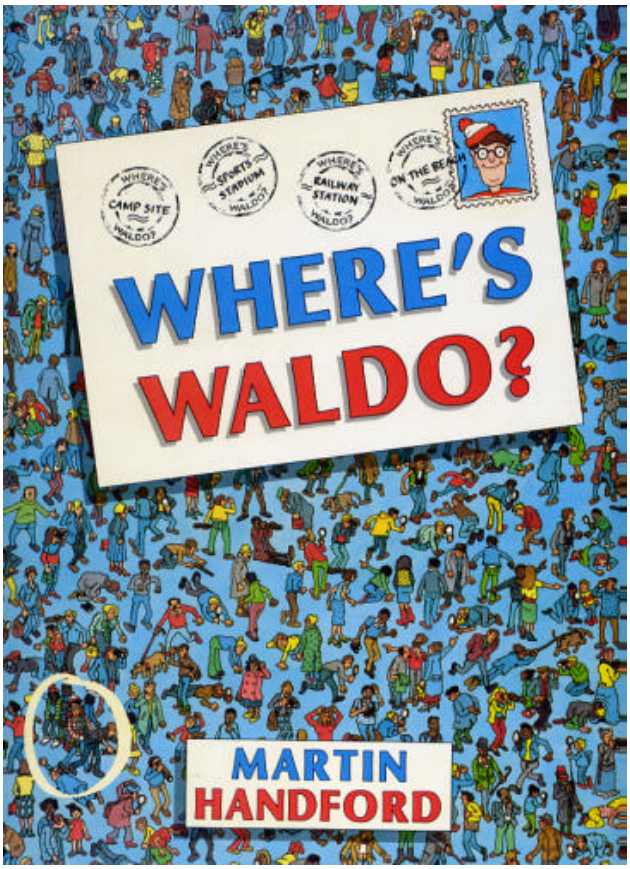


Figure 6.16: Where's Waldo visual search example.

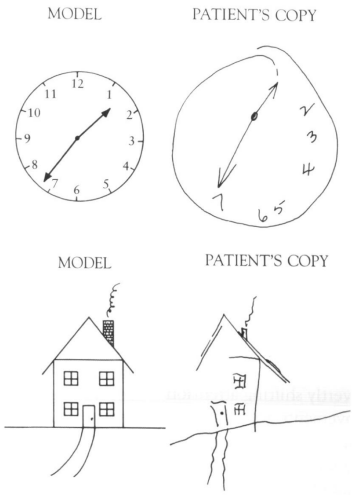


Figure 6.17: Drawings of given target objects by patients with hemispatial neglect, showing profound neglect of the left side of the drawings.

Figure 6.18: Progression of self portraits by an artist with hemispatial neglect, showing gradual remediation of the neglect over time.

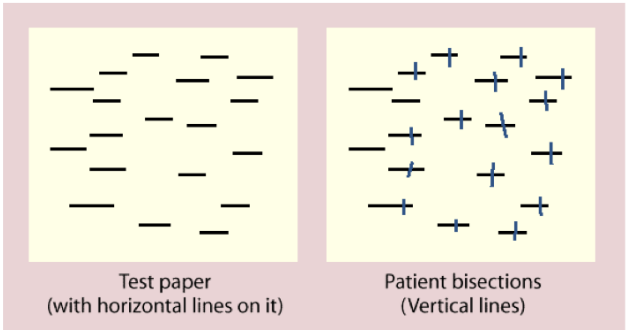


Figure 6.19: Results of a line bisection task for a person with hemispatial neglect. Notice that neglect appears to operate at two different spatial scales here: for the entire set of lines, and within each individual line.

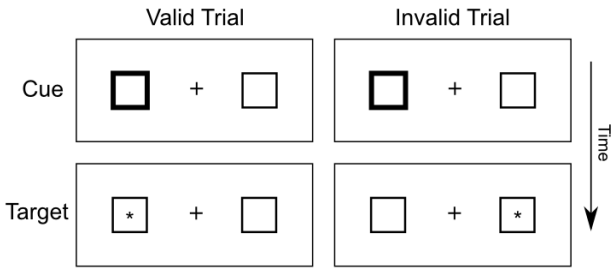


Figure 6.20: The Posner spatial cueing task, widely used to explore spatial attention effects. The participant is shown a display with two boxes and a central fixation cross – on some trials, one of the boxes is cued (e.g., the lines get transiently thicker), and then a target appears in one of the boxes (or not at all on catch trials). The participant just presses a key when they first detect the target. Reaction time is quicker for valid cues vs. invalid ones, suggesting that spatial attention was drawn to that side of space. Patients with hemispatial neglect exhibit slowing for targets that appear in the neglected side of space, particularly when invalidly cued.

drawn to the cued side of space, and thus facilitates target detection. The invalid case is actually worse than a neutral condition with no cue at all, indicating that the process of reallocating spatial attention to the correct side of space takes some amount of time. Interestingly, this task is typically run with the time interval between cue and target sufficiently brief as to prevent eye movements to the cue – thus, these attentional effects are described as *covert attention*.

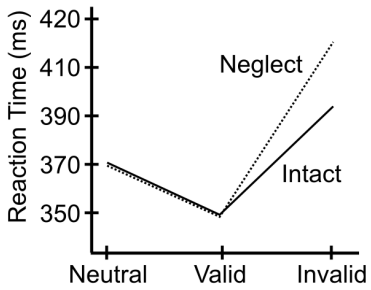


Figure 6.21: Typical data from the Posner spatial cueing task, showing a speedup for valid trials, and a slowdown for invalid trials, compared to a neutral trial with no cueing. The data for patients with hemispatial neglect is also shown, with their overall slower reaction times normalized to that of the intact case.

As shown in Figure 6.21, patients with hemispatial neglect show a disproportionate increase in reaction times for the invalid cue case, specifically when the cue is presented to the good visual field (typically the right), while the target appears in the left. Posner took this data to suggest that these patients have difficulty disengaging attention, according to his box-and-arrow model of the spatial cueing task (Figure 6.22).

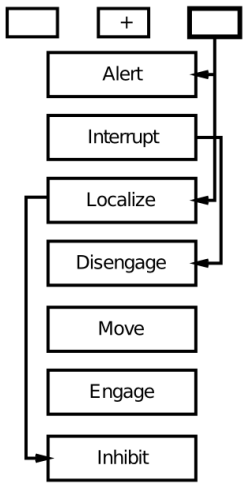


Figure 6.22: Posner's box-and-arrow model of the spatial cueing task. He suggests that spatial neglect deficits are attributable to damage to the disengage mechanism. However, this fails to account for the effects of bilateral parietal damage, for example.

We explore an alternative account here, based on bidirectional interactions between spatial and object processing pathways (Figure 6.23). In this account, damage to one half of the spatial processing pathway leads to an inability of that side to compete against the intact side of the network. Thus, when there is something to compete against (e.g., the cue in the cueing paradigm), the effects of the damage are most pronounced.

Importantly, these models make distinct predictions regarding the effects of *bilateral* parietal damage. Patients with this condition are known to suffer from Balint's syndrome, which is characterized by a profound inability to recognize objects when more than one is present in the visual field (Coslett and Saffran 1991). This is suggestive of the important role that spatial attention plays in facilitating object recognition in

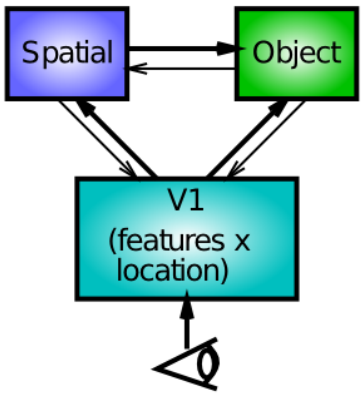


Figure 6.23: Interacting spatial and object recognition pathways can explain Posner spatial attention effects in terms of spatial influences on early object recognition processing, in addition to top-down influence on V1 representations.

crowded visual scenes. According to Posner's disengage model, bilateral damage should result in difficulty disengaging from *both* sides of space, producing slowing in invalid trials for both sides of space. In contrast, the competition-based model makes the opposite prediction: the lesions serve to reduce competition on *both* sides of space, such that there should be reduced attentional effects on both sides. That is, the effect of the invalid cue actually decreases in magnitude. The data is consistent with the competition model, and not Posner's model.

Exploration of Spatial Attention

Open `attn_simpl` in CCN Sims to explore a model with spatial and object pathways interacting in the context of multiple spatial attention tasks, including perceiving multiple objects, and the Posner spatial cueing task. It reproduces the behavioral data shown above, and correctly demonstrates the observed pattern of reduced attentional effects for Balint's patients.

Chapter 7: Motor Control and Reinforcement Learning

The foundations of cognition are built upon the sensory-motor loop – processing sensory inputs to determine which motor action to perform next. This is the most basic function of any nervous system. The human brain has a huge number of such loops, spanning the evolutionary timescale from the most primitive reflexes in the peripheral nervous system, up to the most abstract and inscrutable plans, such as the decision to apply to, and attend, graduate school, which probably involves the highest levels of processing in the prefrontal cortex (PFC) (or perhaps some basic level of insanity... who knows ;).

Table 7.1 Comparison of learning mechanisms and activity/representational dynamics across four primary areas of the brain. +++ means that the area definitely has given property, with fewer +'s indicating less confidence in and/or importance of this feature. — means that the area definitely does not have the given property, again with fewer -'s indicating lower confidence or importance.

	Learning Signal			*Dynamics*		
Area	Reward	Error	Self Org	Separator	Integrator	Attractor
Basal Ganglia	+++	—	—	++	-	—
Cerebellum	—	+++	—	+++	—	—
Hippocampus	+	+	+++	+++	—	+++
Neocortex	++	+++	++	—	+++	+++

In this chapter, we complete the loop that started in the previous chapter on *Perception and Attention*, by covering a few of the most important motor output and control systems, and the learning mechanisms that govern their behavior. At the subcortical level, the **cerebellum** and **basal ganglia** are the two major motor control areas, each of which has specially adapted learning mechanisms that differ from the general-purpose cortical learning mechanisms described in the *Learning* Chapter (Table 7.1). The basal ganglia are specialized for learning from **reward/punishment** signals, in comparison to expectations for reward/punishment, and this learning then shapes the **action selection** that the organism will make under different circumstances (selecting the most rewarding actions and avoiding punishing ones; Figure 7.1). This form of learning is called **reinforcement learning**. The cerebellum is specialized for learning from **error**, specifically errors between the sensory outcomes associated with motor actions, relative to expectations for these sensory outcomes associated with those motor actions. Thus, the cerebellum can refine the implementation of a given motor plan, to make it more accurate, efficient, and well-coordinated.

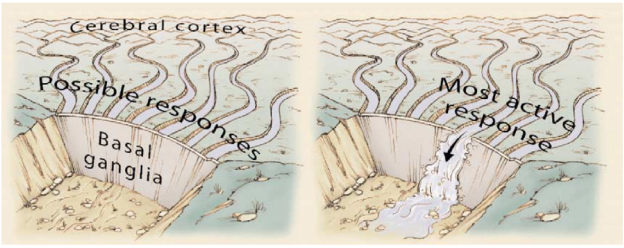


Figure 7.1: Illustration of the role of the basal ganglia in action selection – multiple possible actions are considered in the cortex, and the basal ganglia selects the best (most rewarding) one to actually execute. Reproduced from Gazzaniga et al (2002).

There is a nice division of labor here, where the basal ganglia help to select one out of many possible actions to perform, and the cerebellum then makes sure that the selected action is performed well. Consistent with this rather clean division of labor, there are no direct connections between the basal ganglia and cerebellum – instead, each operates in interaction with various areas in the cortex, where the action plans are formulated and coordinated. Both basal ganglia and cerebellum are densely interconnected with the **frontal cortex**, including motor control areas in posterior frontal cortex, and the **prefrontal cortex** anterior

to those. Also, as discussed in the prior chapter, the **parietal cortex** is important for mapping sensory information to motor outputs (i.e., the "how" pathway), by way of computing things like spatial maps, and relative spatial relationships among objects in the environment. Thus, parietal representations drive motor action execution as coordinated by the cerebellum, and cerebellum is also densely interconnected with parietal cortex. In contrast, the basal ganglia are driven to a much greater extent by the ventral pathway "what" information, which indicates the kinds of rewarding objects that might be present in the environment (e.g., a particular type of food). They do also receive some input from parietal, but just not to the great extent that the cerebellum does.

Both the cerebellum and basal ganglia have a complex **disinhibitory** output dynamic, which produces a gating-like effect on the brain areas they control. For example, the basal ganglia can disinhibit neurons in specific nuclei of the thalamus, which have bidirectional excitatory circuits through frontal and prefrontal cortical areas. The net effect of this disinhibition is to *enable* an action to proceed, without needing to specify any of the details for how to perform that action. This is what is meant by a *gate* – something that broadly modulates the flow of other forms of activation. The cerebellum similarly disinhibits parietal and frontal neurons to effect its form of precise control over the shape of motor actions. It also projects directly to motor outputs in the brain stem, something that is not true of most basal ganglia areas.

We begin the chapter with the basal ganglia system, including the reinforcement learning mechanisms (which involve other brain areas as well). Then we introduce the cerebellar system, and its unique form of error-driven learning. Each section starts with a review of the relevant neurobiology of each system.

Basal Ganglia, Action Selection and Reinforcement Learning

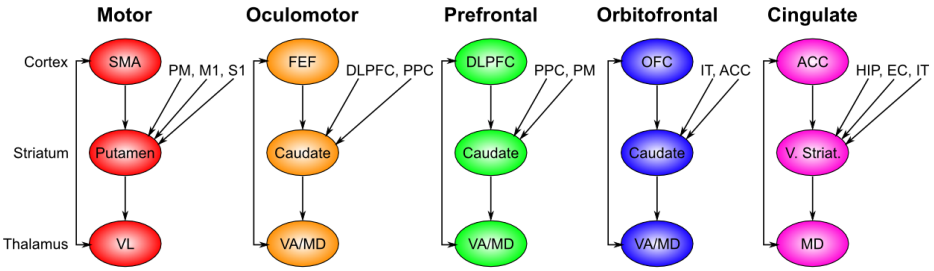


Figure 7.2: Parallel circuits through the basal ganglia for different regions of the frontal cortex – each region of frontal cortex has a corresponding basal ganglia circuit, for controlling action selection/initiation in that frontal area. Motor loop: SMA = supplementary motor area – the associated striatum (putamen) also receives from premotor cortex (PM), and primary motor (M1) and somatosensory (S1) areas – everything needed to properly contextualize motor actions. Oculomotor loop: FEF = frontal eye fields, also receives from dorsolateral PFC (DLPFC), and posterior parietal cortex (PPC) – appropriate context for programming eye movements. Prefrontal loop: DLPFC also controlled by posterior parietal cortex, and premotor cortex. Orbitofrontal loop: OFC = orbitofrontal cortex, also receives from inferotemporal cortex (IT), and anterior cingulate cortex (ACC). Cingulate loop: ACC also modulated by hippocampus (HIP), entorhinal cortex (EC), and IT.

The basal ganglia performs its action selection function over a wide range of frontal cortical areas, by virtue of a sequence of parallel loops of connectivity (Figure 7.2). These areas include motor (skeletal muscle control) and oculomotor (eye movement control), but also prefrontal cortex, orbitofrontal cortex, and anterior cingulate cortex, which are not directly motor control areas. Thus, we need to generalize our notion of action selection to include **cognitive action selection** – more abstract forms of selection that operate in higher-level cognitive areas of prefrontal cortex. For example, the basal ganglia can control the selection of large-scale action plans and strategies in its connections to the prefrontal cortex. The orbitofrontal cortex is important for encoding the reward value associated with different possible stimulus outcomes, so the basal ganglia connection here is important for driving the updating of these representations as a function of contingencies in the environment. The anterior cingulate cortex is important for encoding the costs of motor actions (time, effort, uncertainty), and basal ganglia similarly can help control updating of these costs as different actions are considered. We can summarize the role of basal ganglia in these more abstract frontal

areas as controlling **working memory updating**, as is discussed further in the *Executive Function* Chapter.

Interestingly, the additional inputs that converge into the basal ganglia for a given area all make good sense. Motor control needs to know about the current somatosensory state, as well as inputs from the slightly higher-level motor control area known as premotor cortex. Orbitofrontal cortex is all about encoding the reward value of stimuli, and thus needs to get input from IT cortex, which provides the identity of relevant objects in the environment.

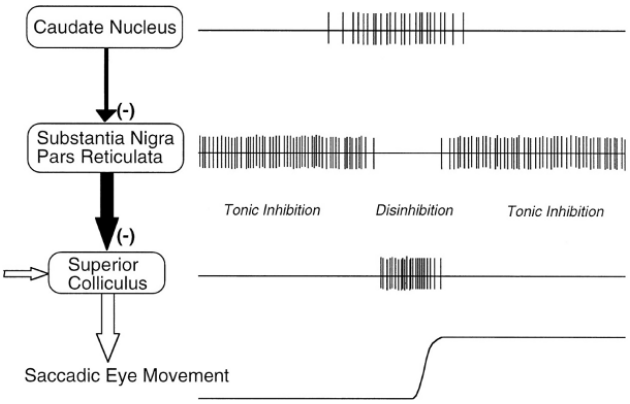


Figure 7.3: Gating mechanisms of the direct pathway in the oculomotor circuit. An eye saccade movement is made when the superior colliculus (SC) neurons coding for the saccade direction exhibit burst firing. The SC receives excitatory input from elsewhere (e.g., frontal cortex) indicating planned eye movements. However, the SC is under tonic inhibitory regulation from the output of the basal ganglia (in this circuit, it is the substantia nigra pars reticulata (SNr), equivalent to the GPi for other movements). SNr neurons fire at high tonic rates in the absence of input, and prevent the SC from initiating a burst. Neurons in the caudate nucleus (part of the striatum), upstream of the SNr, are normally silent but fire when detecting the appropriate conditions under which to initiate the eye movement (e.g., when it is predictive of reward). Caudate neurons inhibit the SNr, causing a pause in tonic firing, and disinhibit the SC. This disinhibition acts as a gating mechanism because the Caudate does not directly elicit SC firing but instead allows SC to burst fire in particular SC neurons that also receive excitatory input about the planned movement. Not shown here are indirect pathway Caudate neurons which would have the opposite effect, increasing SNr activity and preventing gating of particular movements. From Hikosaka et al, 2000.

Zooming in on any one of these loops, the critical elements of the basal ganglia system are diagrammed in Figure 7.4, with two important activation patterns shown. First, the basal ganglia system involves the following subregions:

- The **striatum**, which is the major input region, consisting of the **caudate** and **putamen** subdivisions (as shown in 7.2). The striatum is anatomically subdivided into many small clusters of neurons, with two major types of clusters: *patch/striosomes* and *matrix/matrisomes*. The matrix clusters contain **direct (Go)** and **indirect (NoGo)** pathway *medium spiny* neurons, which together make up 95% of striatal cells, both of which receive excitatory inputs from all over the cortex but are inhibitory on their downstream targets in the globus pallidus as described next. The patch cells project to the dopaminergic system, and thus appear to play a more indirect role in modulating learning signals. There are also a relatively few widely spaced tonically active neurons (TANs), which release acetylcholine as a neurotransmitter and appear to play a modulatory role, and inhibitory interneurons, which likely perform the same kind of dynamic gain control that they play in the cortex.
- The **globus pallidus, internal segment (GPi)**, which is a much smaller structure than the striatum, and contains neurons that are constantly (tonically) active even with no additional input. These neurons send inhibition to specific nuclei in the thalamus. When the direct/Go pathway striatum neurons fire, they inhibit these GPi neurons, and thus *disinhibit* the thalamus, resulting ultimately in the initiation of a specific motor or cognitive action (depending on which circuit is involved). Note that in other fronto-basal ganglia circuits, the role of the GPi is taken up by the substantia nigra pars reticulata (SNr), which is situated identically to the GPi anatomically, but receives from other areas of striatum and projects to outputs regulating other actions (e.g., eye movements in the superior colliculus).

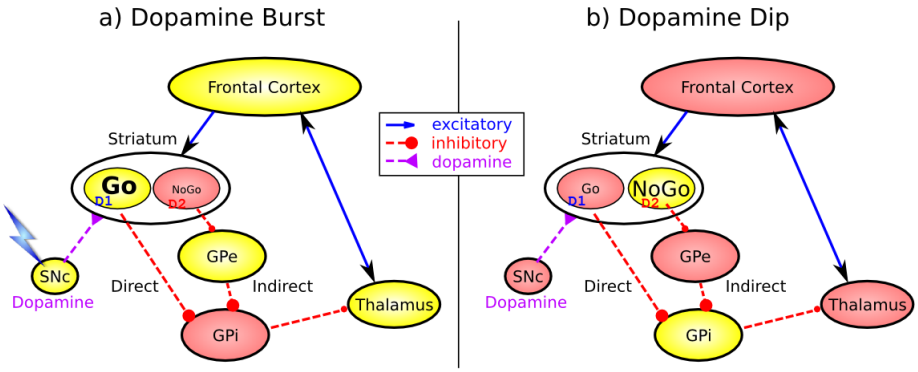


Figure 7.4: Biology of the basal ganglia system, with two cases shown: a) Dopamine burst activity that drives the direct "Go" pathway neurons in the striatum, which then inhibit the tonic activation in the globus pallidus internal segment (GPi), which releases specific nuclei in the thalamus from this inhibition, allowing them to complete a bidirectional excitatory circuit with the frontal cortex, resulting in the initiation of a motor action. The increased Go activity during dopamine bursts results in potentiation of corticostriatal synapses, and hence learning to select actions that tend to result in positive outcomes. b) Dopamine dip (pause in tonic dopamine neuron firing), leading to preferential activity of indirect "NoGo" pathway neurons in the striatum, which inhibit the external segment globus pallidus neurons (GPe), which are otherwise tonically active, and inhibiting the GPi. Increased NoGo activity thus results in disinhibition of GPi, making it more active and thus inhibiting the thalamus, preventing initiation of the corresponding motor action. The dopamine dip results in potentiation of corticostriatal NoGo synapses, and hence learning to avoid selection actions that tend to result in negative outcomes. From Frank, 2005

- The **globus pallidus, external segment (GPe)**, which is also small, and contains tonically active neurons that send focused inhibitory projections to corresponding GPi neurons. When the indirect/NoGo pathway neurons in the striatum fire, they inhibit the GPe neurons, and thus disinhibit the GPi neurons, causing them to provide even greater inhibition onto the thalamus. This blocks the initiation of specific actions coded by the population of active NoGo neurons.
- The **thalamus**, specifically the medial dorsal (MD), ventral anterior (VA), and ventrolateral (VL) nuclei (as shown in Figure 7.3 from (Hikosaka, Takikawa, and Kawagoe 2000)). When the thalamic neurons get disinhibited by Go pathway firing, they can fire, but only when driven by top-down excitatory input from the frontal cortex. In this way, the basal ganglia serve as a **gate** on the thalamocortical circuit – Go firing opens the gate, while NoGo firing closes it, but the contents of the information that go through the gate (e.g., the specifics of the motor action plan) depend on the thalamocortical system. In the oculomotor circuit (as shown in , the role of the thalamus is taken up by the superior colliculus, the burst firing of which initiates eye saccades).
- The **substantia nigra pars compacta (SNc)** has neurons that release the neuromodulator dopamine, and specifically innervate the striatum. Interestingly, there are two different kinds of dopamine receptors in the striatum. D1 receptors are prevalent in Go pathway neurons, and dopamine has an excitatory effect on neurons with D1 receptors (particularly those neurons that are receiving convergent glutamatergic excitatory input from cortex). In contrast, D2 receptors are prevalent in NoGo pathway neurons, and dopamine has an inhibitory effect via the D2 receptors. Thus, when a burst of dopamine hits the striatum, it further excites active Go units and inhibits NoGo units. This change in activity results in activity-dependent plasticity, and thus leads to an increased propensity for initiating motor and cognitive actions. In contrast, when a dip in dopamine firing occurs, Go neurons are less excited, while NoGo neurons are disinhibited, and thus those NoGo neurons receiving excitatory input from cortex (representing the current state and action) will become more excited due to the dopamine dip. Again, this change in activity results in potentiation of synapses, such that this specific population of NoGo neurons will be more likely to become active in future encounters of this sensory state and candidate motor action. Both of these effects of dopamine bursts and dips make perfect sense: dopamine bursts are associated with positive reward prediction errors (when rewards are better than expected), and thus reinforce selection of actions that lead to good results. Conversely, dopamine dips are associated

with negative reward prediction errors (worse than expected) and thus lead to avoidance (NoGo) of those actions that tend to result in these bad results. Also, tonic levels of dopamine can influence the relative balance of activity of these pathways, so that even if learning has already occurred, changes in dopamine can affect whether action selection is influenced primarily by learned Go vs learned NoGo values – roughly speaking, the higher the dopamine, the more risky the choices (insensitivity to negative outcomes).

- The **subthalamic nucleus** is also a major component of the basal ganglia (not pictured in the figure), which acts as the third *hyperdirect pathway*, so named because it receives input directly from frontal cortex and sends excitatory projections directly to BG output (GPi), bypassing the striatum altogether. These STN-GPi projections are diffuse, meaning that a single STN neuron projects broadly to many GPi neurons, and as such the STN is thought to provide a *global NoGo* function that prevents gating of any motor or cognitive action (technically, it raises the threshold for gating). This area has been shown in models and empirical data to become more active with increasing demands for response inhibition or when there is conflict between alternative cortical action plans, so that the STN buys more time for striatal gating to settle on the best action (Michael J. Frank 2006).

This is a fairly complex circuit, and it probably takes a few iterations through it to really understand how all the parts fit together. The bottom line should nevertheless be easier to understand: the basal ganglia learn to select rewarding actions (including more abstract cognitive actions), via a disinhibitory gating relationship with different areas of frontal cortex. Moreover, the general depiction above, motivated by computational considerations and a lot of detailed anatomical, physiological, and pharmacological data, has been overwhelmingly been supported by empirical data across species. For example, in mice, selective stimulation of D1 striatal neurons resulted in inhibition of BG output nuclei and disinhibition of motor actions, whereas selective stimulation of D2 striatal neurons resulted in excitation of output nuclei and suppression of motor actions (Kravitz et al. 2010). A follow up paper in 2012 showed that transient stimulations of these pathways after movements causes the mouse to be more likely (go unit stimulation) or less likely (nogo unit stimulation) to repeat that same movement in the future, consistent with a learning effect (Kravitz, Tye, and Kreitzer 2012).

Other research showed that when a mouse experiences a negative reward prediction error (i.e. they expect a reward but don't get one), the D2 neurons respond by increasing their activity levels, and the extent of this is related to their subsequent avoidance of the action in favor of a safer option leading to certain reward (Zalocusky et al. 2016). There is also evidence for the model prediction that D1 and D2 receptors oppositely modulate synaptic plasticity in the two pathways (Shen et al. 2008). Furthermore, selective blockade of neurotransmission along the Go pathway resulted in impairments in learning to select rewarding actions but no deficits in avoiding punishing actions, and exactly the opposite pattern of impairments was observed after blockade of the NoGo pathway (Hikida et al. 2010). In humans, striatal dopamine depletions associated with Parkinson's disease result in impaired "Go learning" in probabilistic reinforcement learning tasks, but enhanced "NoGo learning", with the opposite pattern of findings elicited by medications that increase striatal dopamine (Frank, Seeberger, and O'Reilly 2004). Even individual differences in young healthy human performance in Go vs NoGo learning are associated with genetic variants that affect striatal D1 and D2 receptor function and D1 vs D2 receptor expression in PET studies (Cox et al. 2015; Frank and Fossella 2011).

The division of labor between frontal cortex and basal ganglia is such that the frontal cortex entertains many different possible actions, by virtue of rich patterns of connectivity from other cortical areas providing high-level summaries of the current environment, which then activate a range of different possible actions, and the basal ganglia then selects the best (most likely to be rewarding) of these actions to actually execute. In more anthropomorphic terms, the frontal cortex is the fuzzy creative type, with a million ideas, but no ability to focus on the real world, and it has a hard time narrowing things down to the point of actually doing anything: kind of a dreamer. Meanwhile, the basal ganglia is a real take-charge type who always has the bottom line in mind, and can make the tough decisions and get things done. We need both of these personalities in our heads (although people clearly differ in how much of each they have), and the neural systems that support these different modes of behavior are clearly different. This is presumably why there are two separable systems (frontal cortex and basal ganglia) that nevertheless work very closely together to solve the overall action selection problem.

Exploration of the Basal Ganglia

Open the `bg` simulation from CCN Sims for an exploration of a basic model of go vs. nogo action selection and learning dynamics in the basal ganglia. This model also allows you to investigate the effects of Parkinson's disease and dopaminergic medications.

Dopamine and Temporal Difference Reinforcement Learning

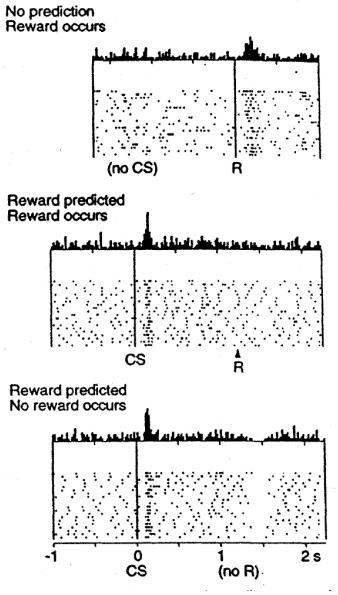


Figure 7.5: Characteristic patterns of neural firing of the dopaminergic neurons in the ventral tegmental area (VTA) and substantia nigra pars compacta (SNc), in a simple conditioning task (Schultz et al, 1997). Prior to conditioning, when a reward is delivered, the dopamine neurons fire a burst of activity (top panel – histogram on top shows sum of neural spikes across the repeated recording traces shown below, with each row being a different recording trial). After the animal has learned to associate a conditioned stimulus (CS) (e.g., a tone) with the reward, the dopamine neurons now fire to the onset of the CS, and not to the reward itself. If a reward is withheld after the CS, there is a dip or pause in dopamine firing, indicating that there was some kind of prediction of the reward, and when it failed to arrive, there was a negative prediction error. This overall pattern of firing across conditions is highly consistent with reinforcement learning models based on reward prediction error. Reproduced from Schultz et al, 1997

Although we considered above how phasic changes in dopamine can drive Go and NoGo learning to select the most rewarding actions and to avoid less rewarding ones, we have not addressed above how dopamine neurons come to represent these phasic signals for driving learning. One of the most exciting discoveries in recent years was the finding that dopamine neurons in the ventral tegmental area (VTA) and substantia nigra pars compacta (SNc) behave in accord with reinforcement learning models based on reward prediction error. Unlike some popular misconceptions, these dopamine neurons do not encode raw reward value directly. Instead, they encode the *difference* between reward received versus an expectation of reward. This is shown in Figure 7.5 (Schultz, Dayan, and Montague 1997): if there is no expectation of reward, then dopamine neurons fire to the reward, reflecting a positive reward prediction error (zero expectation, positive reward). If a conditioned stimulus (CS, e.g., a tone or light) reliably predicts a subsequent reward, then the neurons no longer fire to the reward itself, reflecting the lack of reward prediction error (expectation = reward). Instead, the dopamine neurons fire to the onset of the CS. If the reward is omitted following the CS, then the dopamine neurons actually go the other way (a "dip" or "pause" in the otherwise low tonic level of dopamine neuron firing), reflecting a negative reward prediction error (positive reward prediction, zero reward).

Computationally, the simplest model of reward prediction error is the **Rescorla-Wagner** conditioning model (Rescorla and Wagner 1972), which is mathematically identical to the **delta rule** as discussed in the

Learning Chapter, and is simply the difference between the actual reward and the expected reward:

$$\delta = r - \hat{r}$$

$$\delta = r - \sum xw$$

where δ ("delta") is the reward prediction error, r is the amount of reward actually received, and $\hat{r} = \sum xw$ is the amount of reward *expected*, which is computed as a weighted sum over input stimuli x with weights w. The weights adapt to try to accurately predict the actual reward values, and in fact this delta value specifies the direction in which the weights should change:

$$\Delta w = \delta x$$

This is identical to the delta learning rule, including the important dependence on the stimulus activity x – you only want to change the weights for stimuli that are actually present (i.e., non-zero x's).

When the reward prediction is correct, then the actual reward value is *canceled out* by the prediction, as shown in the second panel in . This rule also accurately predicts the other cases shown in the figure too (positive and negative reward prediction errors).

What the Rescorla-Wagner model fails to capture is the firing of dopamine to the onset of the CS in the second panel in Figure 7.5. However, a slightly more complex model known as the **temporal differences (TD)** learning rule does capture this CS-onset firing, by introducing time into the equation (as the name suggests) (Sutton and Barto 1981, 1998). Relative to Rescorla-Wagner, TD just adds one additional term to the delta equation, representing the *future* reward values that might come later in time:

$$\delta = (r + f) - \hat{r}$$

where f represents the future rewards, and now the reward expectation $\hat{r} = \sum xw$ has to try to anticipate both the current reward r and this future reward f. In a simple conditioning task, where the CS reliably predicts a subsequent reward, the onset of the CS results in an increase in this f value, because once the CS arrives, there is a high probability of reward in the near future. Furthermore, this f itself is not predictable, because the onset of the CS is not predicted by any earlier cue (and if it was, then that earlier cue would be the real CS, and drive the dopamine burst). Therefore, the r-hat expectation cannot cancel out the f value, and a dopamine burst ensues.

Although this f value explains CS-onset dopamine firing, it raises the question of how can the system know what kind of rewards are coming in the future? Like anything having to do with the future, you fundamentally just have to guess, using the past as your guide as best as possible. TD does this by trying to *enforce consistency in reward estimates over time*. In effect, the estimate at time t is used to train the estimate at time *t-1*, and so on, to keep everything as consistent as possible across time, and consistent with the actual rewards that are received over time.

This can all be derived in a very satisfying way by specifying something known as a **value function, V(t)** that is a sum of all present and future rewards, with the future rewards **discounted** by a "gamma" factor, which captures the intuitive notion that rewards further in the future are worth less than those that will occur sooner. As the Wimpy character says in Popeye, "I'll gladly pay you Tuesday for a hamburger today." Here is that value function, which is an infinite sum going into the future:

$$V(t) = r(t) + \gamma^1 r(t+1) + \gamma^2 r(t+2)...$$

We can get rid of the infinity by writing this equation *recursively*:

$$V(t) = r(t) + \gamma V(t+1)$$

And because we don't know anything for certain, all of these value terms are really estimates, denoted by the little "hats" above them:

$$\hat{V}(t) = r(t) + \gamma \hat{V}(t+1)$$

So this equation tells us what our estimate at the current time t should be, in terms of the future estimate at time t+1. Next, we subtract V-hat from both sides, which gives us an expression that is another way of expressing the above equality – that the difference between these terms should be equal to zero:

$$0 = \Big(r(t) + \hat{V}(t+1)\Big) - \hat{V}(t)$$

This is mathematically stating the point that TD tries to keep the estimates consistent over time – their difference should be zero. But as we are learning our V-hat estimates, this difference will *not* be zero, and in fact, the extent to which it is not zero is the extent to which there is a reward prediction error:

$$\delta = \Big(r(t) + \hat{V}(t+1)\Big) - \hat{V}(t)$$

If you compare this to the equation with f in it above, you can see that:

$$f = \gamma\hat{V}(t+1)$$

and otherwise everything else is the same, except we've clarified the time dependence of all the variables, and our reward expectation is now a "value expectation" instead (replacing the r-hat with a V-hat). Also, as with Rescorla-Wagner, the delta value here drives learning of the value expectations.

The TD learning rule can be used to explain a large number of different conditioning phenomena, and its fit with the firing of dopamine neurons in the brain has led to a large amount of research progress. It represents a real triumph of the computational modeling approach for understanding (and predicting) brain function.

Exploration of TD Learning

Open the `rl` simulation from CCN Sims for an exploration of TD-based reinforcement learning in simple conditioning paradigms. This exploration should help solidify your understanding of reinforcement learning, reward prediction error, and simple classical conditioning.

The Actor-Critic Architecture for Motor Learning

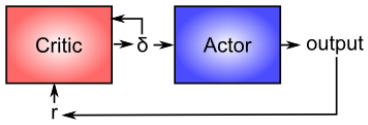


Figure 7.6: Basic structure of the actor critic architecture for motor control. The critic is responsible for processing reward inputs (r), turning them into reward prediction errors (delta), which are suitable for driving learning in both the critic and the actor. The actor is responsible for producing motor output given relevant sensory input, and doesn't process reward or reward expectations directly. This is an efficient division of labor, and it is essential for learning to transform rewards into reward prediction errors, otherwise the system would overlearn on simple tasks that it mastered long ago.

Now that you have a better idea about how dopamine works, we can revisit its role in modulating learning in the basal ganglia (as shown in Figure 7.4). From a computational perspective, the key idea is the distinction between an **actor** and a **critic** (), where it is assumed that rewards result at least in part from correct performance by the actor. The basal ganglia is the actor in this case, and the dopamine signal is the output of the critic, which then serves as a training signal for the actor (and the critic too as we saw earlier). The reward prediction error signal produced by the dopamine system is a good training signal because it drives stronger learning early in a skill acquisition process, when rewards are more unpredictable, and reduces learning as the skill is perfected, and rewards are thus more predictable. If the system instead learned directly on the basis of external rewards, it would continue to learn about skills that have long been mastered, and this would likely lead to a number of bad consequences (synaptic weights growing ever stronger, interference with other newer learning, etc).

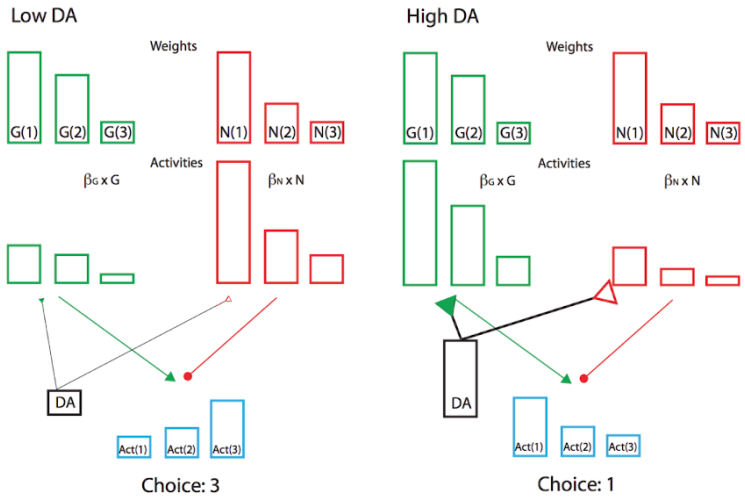


Figure 7.7: The Opponent Actor Learning (OpAL) scheme. This is a modified actor critic whereby the actor contains separate G and N weights representing the Go and NoGo pathways. The activities of the pathways are scaled by dopamine levels during choice, and the relative activation differences for each action are compared to make a choice. The figure depicts selection among three actions that have different learned costs and benefits (think coffee, tea, water: clearly coffee has a better benefit than tea, but it also has higher costs (jitters etc)). When dopamine levels are low (left), the costs are amplified, and the benefits diminished, and the system chooses to avoid the highest cost and selects action 3 (water). When dopamine levels are high, the benefits are amplified and the costs diminished, and it chooses action 1 (coffee). Moderate dopamine levels are associated with action 2 (tea; not shown). This accounts for differential effects of dopamine on learning and choice among actions with different costs and benefits. From Collins & Frank, 2014.

Furthermore, the sign of the reward prediction error is appropriate for the effects of dopamine on the Go and NoGo pathways in the striatum, as we saw in the BG model project above. Positive reward prediction errors, when unexpected rewards are received, indicate that the selected action was better than expected, and thus Go firing for that action should be increased in the future. The increased activation produced by dopamine on these Go neurons will have this effect, assuming learning is driven by these activation levels. Conversely, negative reward prediction errors will facilitate NoGo firing, causing the system to avoid that action in the future. Indeed, the complex neural network model of BG Go/NoGo circuitry can be simplified with more formal analysis in a modified actor-critic architecture called Opponent Actor Learning (OpAL; Figure 7.7), where the actor is divided into independent G and N opponent weights, and where their relative contribution is itself affected by dopamine levels during both learning and choice (Collins and Frank 2014).

Finally, the ability of the dopamine signal to propagate backward in time is critical for spanning the inevitable delays between motor actions and subsequent rewards. Specifically, the dopamine response should move from the time of the reward to the time of the action that reliably predicts reward, in the same way that it moves in time to the onset of the CS in a classical conditioning paradigm.

The PVLV Model of DA Biology

You might have noticed that we haven't yet explained at a biological level how the dopamine neurons in the VTA and SNc actually come to exhibit their reward prediction error firing. There is a growing body of data supporting the involvement of the brain areas shown in Figure 7.8:

- **Lateral hypothalamus (LHA)** provides a primary reward signal for basic rewards like food, water etc.
- **Patch-like neurons in the ventral striatum (VS-patch)** have direct inhibitory connections onto the dopamine neurons in the VTA and SNc, and likely play the role of canceling out the influence of primary reward signals when these rewards have successfully been predicted.
- **Central nucleus of the amygdala (CNA)** is important for driving dopamine firing to the onset of

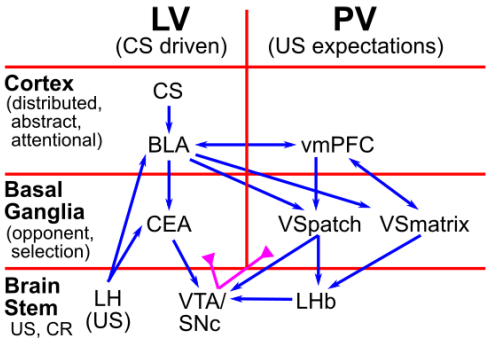


Figure 7.8: Biological mapping of the PVLV algorithm, which has two separate subsystems: Primary Value (PV) and Learned Value (LV). Each subsystem has excitatory and inhibitory subcomponents, so named for their effect on dopamine firing. PVe = primary rewards that excite dopamine, associated with lateral hypothalamic nucleus (LHA). PVi = inhibitory canceling of dopamine firing to rewards, driven by patch-like neurons in the ventral striatum (VS_Patch). LVe = excitatory drive from the central nucleus of the amygdala (CNA), which represents CS's. LVi = inhibitory canceling of LVe excitation, also via patch-like neurons in the ventral striatum. The pedunculopontine tegmental nucleus (PPTN) may transform sustained inputs into phasic dopamine responses via a simple temporal delta operation. From Mollick et al, (in press).

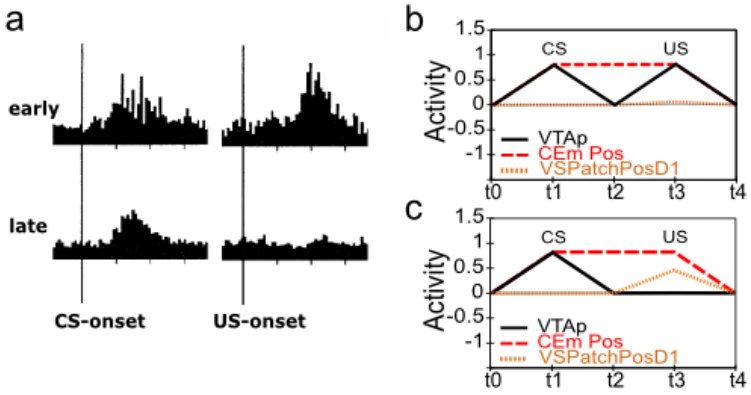


Figure 7.9:

conditioned stimuli. It receives broadly from the cortex, and projects directly and indirectly to the VTA and SNc. Neurons in the CNA exhibit CS-related firing.

Given that there are distinct brain areas involved in these different aspects of the dopamine firing, it raises the question as to how the seemingly unified TD learning algorithm could be implemented across such different brain areas? In response to this basic question, the PVLV model of dopamine firing was developed. PVLV stands for Primary Value, Learned Value, and the key idea is that different brain structures are involved at the time when primary values are being experienced, versus when conditioned stimuli (learned values) are being experienced. This then requires a different mathematical formulation, as compared to TD.

The dopamine signal in PVLV for primary values (PV), which is in effect at the time external rewards are delivered or expected, is identical to Rescorla-Wagner, just using different labels for the variables:

$$\delta_{pv} = r - \hat{r}$$

$$\delta_{pv} = PV_e - PV_i$$

Where excitatory (e) and inhibitory (i) subscripts denote the two components of the primary value system, and the sign of their influence on dopamine firing.

The dopamine signal for learned values (LV) applies whenever PV does not (i.e., when external rewards are *not* present or expected), and it has a similar form:

$$\delta_{lv} = LV_e - LV_i$$

Where LVe is the excitatory drive on dopamine from the CNA, which learns to respond to CSs. LVi is a counteracting inhibitory drive, again thought to be associated with the patch-like neurons of the ventral striatum. It learns much more slowly than the LVe system, and will eventually learn to cancel out CS-associated dopamine responses, once these CS's become highly highly familiar (beyond the short timescale of most experiments).

The PVi values are learned in the same way as in the delta rule or Rescorla-Wagner, and the LVe and LVi values are learned in a similar fashion as well, except that their training signal is driven directly from the PVe reward values, and *only occurs when external rewards are present or expected.* This is critical for allowing LVe for example to get activated at the time of CS onset, when there isn't any actual reward value present. If LVe was always learning to match the current value of PVe, then this absence of PVe value at CS onset would quickly eliminate the LVe response then. See the *PVLV Learning* Appendix for the full set of equations governing the learning of the LV and PV components.

There are a number of interesting properties of the learning constraints in the PVLV system. First, the CS must still be active at the time of the external reward in order for the LV_e system to learn about it, since LV only learns at the time of external reward. If the CS itself goes off, then some memory of it must be sustained. This fits well with known constraints on CS learning in conditioning paradigms. Second, the dopamine burst at the time of CS onset cannot influence learning in the LV system itself – otherwise there would be an unchecked positive feedback loop. One implication of this is that the LV system cannot support second-order conditioning, where a first CS predicts a second CS which then predicts reward. Consistent with this constraint, the CNA (i.e., LV_e) appears to only be involved in first order conditioning, while the basolateral nucleus of the amygdala (BLA) is necessary for second-order conditioning. Furthermore, there does not appear to be much of any evidence for third or higher orders of conditioning. Finally, there is a wealth of specific data on differences in CS vs. US associated learning that are consistent with the PVLV framework (Mollick et al., n.d.; Hazy, Frank, and O'Reilly 2010; O'Reilly et al. 2007)

In short, the PVLV system can explain how the different biological systems are involved in generating phasic dopamine responses as a function of reward associations, in a way that seems to fit with otherwise somewhat peculiar constraints on the system. Also, we will see in the *Executive Function* Chapter that PVLV provides a cleaner learning signal for controlling the basal ganglia's role in the prefrontal cortex working memory system.

Exploration of PVLV

Open the `pvlv` simulation in CCN Sims, which runs the same conditioning paradigms as explored in the TD model.

Cerebellum and Error-Driven Learning

Figure 7.10: Areas of the cortex that project to the cerebellum – unlike the basal ganglia, the cerebellum receives exclusively from motor-related areas, including the parietal cortex (which includes primary somatosensory cortex), and motor areas of frontal cortex. Notably, it does not receive from prefrontal cortex or temporal cortex.

Now that we understand how the basal ganglia can select an action to perform based on reinforcement learning, we turn to the cerebellum, which takes over once the action has been initiated, and uses error-driven learning to shape the performance of the action so that it is accurate and well-coordinated. As shown in Figure 7.10, the cerebellum only receives from cortical areas directly involved in motor production, including the parietal cortex and the motor areas of frontal cortex. Unlike the basal ganglia, it does not receive from prefrontal cortex or temporal cortex, which makes sense according to their respective functions. Prefrontal cortex and temporal cortex are really important for high-level planning and action selection, but not for action execution. However, we do know from neuroimaging experiments that the cerebellum is engaged in many cognitive tasks – this must reflect its extensive connectivity with the parietal cortex, which is also activated in many cognitive tasks. One idea is that the cerebellum can help shape learning and processing in parietal cortex by virtue of its powerful error-driven learning mechanisms – this may help to explain how the parietal cortex can learn to do all the complex things it does. However, at this point both the parietal cortex and cerebellum are much better understood from a motor standpoint than a cognitive one.

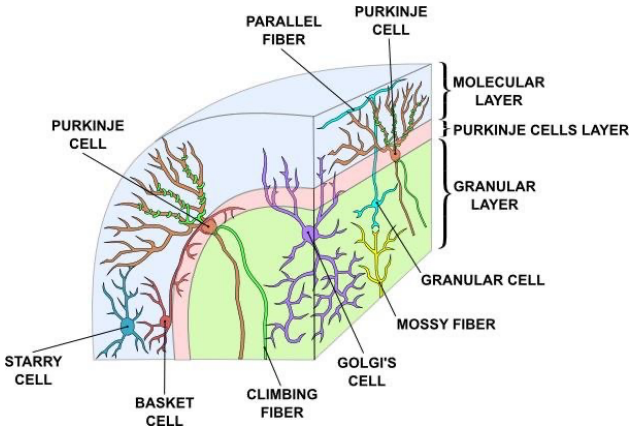


Figure 7.11: Circuitry and structure of the cerebellum – see text for details.

The cerebellum has a very well-defined anatomy (Figure 7.11), with the same basic circuit replicated throughout. Thus, like the basal ganglia, it seems to be performing the same basic function replicated over a wide range of different content domains (e.g., for different motor effectors, and for different areas of parietal and frontal cortex). The basic circuit involves input signals coming from various sources, which are conveyed into the cerebellum via **mossy fiber** axons. These terminate onto **granule cells**, of which there are roughly 40 billion in the human brain! Each granule cell receives only 4-5 mossy fiber inputs, and there are roughly 200 million mossy fiber inputs, with each mossy fiber synapsing on roughly 500 granule cells. Thus, there is a great expansion of information coding in the granule cells relative to the input – we'll revisit this important fact in a moment. To complete the circuit, the granule cells send out **parallel fiber** axons, that synapse into the very dense dendritic trees of **Purkinje cells**, which can receive as many as 200,000 inputs from granule cells. There are roughly 15 million Purkinje cells in the human brain, and these cells produce the output signal from the cerebellum. Thus, there is a massive convergence operation from the granule cells onto the Purkinje cells. The Purkinje cells are tonically active, and the granule cells are excitatory onto them, making

it a bit puzzling to figure out how the granule cells convey a useful signal to the Purkinje cells. The other cell types in the cerebellum (stellate, basket, and golgi) are inhibitory interneurons that provide inhibitory control over both granule cell and Purkinje cell firing. It is possible that granule cells work in concert with these inhibitory cells to alter the balance of excitation and inhibition in the Purkinje cells, but this remains somewhat unclear.

The final piece of the cerebellar puzzle is the **climbing fiber** input from the inferior olivary nucleus – there is exactly one such climbing fiber per Purkinje cell, and it has a very powerful effect on the neuron, producing a series of *complex spikes*. It is thought that climbing fiber inputs convey a training or error signal to the Purkinje's, which then drives synaptic plasticity in its associated granule cell inputs. One prominent idea is that this synaptic plasticity tends to produce LTD (weight decrease) for synaptic inputs where the granule cells are active, which then makes these neurons less likely to fire the Purkinje cell in the future. This would make sense given that the Purkinje cells are inhibitory on the **deep cerebellar nuclei** neurons, so to produce an output from them, the Purkinje cell needs to be turned off.

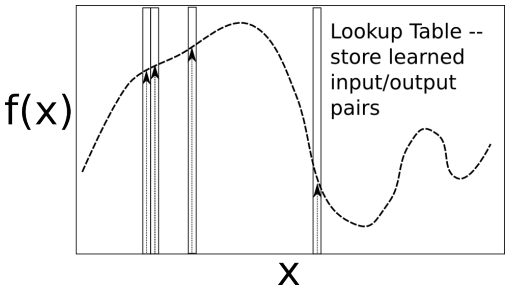


Figure 7.12: Lookup table solution to function learning – the appropriate value for the function can be memorized for each input value X, with perhaps just a little bit of interpolation around the X values. For the cerebellum, X are the inputs (sensory signals, etc), and f(X) is the motor output commands.

Putting all these pieces together, David Marr (Marr 1969) and James Albus (Albus 1971) argued that the cerebellum is a system for error-driven learning, with the error signal coming from the climbing fibers. It is clear that it has the machinery to associate stimulus inputs with motor output commands, under the command of the climbing fiber inputs. One important principle of cerebellar function is the projection of inputs into a very high-dimensional space over the granule cells – computationally this achieves the *separation* form of learning, where each combination of inputs activates a unique pattern of granule cell neurons. This unique pattern can then be associated with a different output signal from the cerebellum, producing something approximating a *lookup table* of input/output values (Figure 7.12). A lookup table provides a robust solution to learning even very complex, arbitrary functions – it will always be able to encode any kind of function. The drawback is that it does not generalize to novel input patterns very well. However, it may be better overall in motor control to avoid improper generalization, rather than eek out a bit more efficiency from some form of generalization. This high-dimensional expansion is also used successfully by the support vector machine (SVM), one of the most successful machine learning algorithms.

Exploration of Cerebellum

Open the `cereb` model in CCN Sims of motor learning.

Appendix

- *PVLV Learning:* – full set of equations governing the learning in the PVLV model of phasic dopamine.

PVLV Learning

Chapter 8: Memory

When you think of memory, you probably think of **episodic memory** – memory for specific episodes or events. Maybe you can remember some special times from childhood (birthdays, family trips, etc), or some traumatic times (ever get lost in a supermarket, or get left behind on a hike or other family outing?). Probably you can remember what you had for dinner last night, and who you ate with? Although this aspect of memory is the most salient for us, it is just one of many different types of memory.

One broad division in memory in mechanistic, computational terms is between **weight-based** and **activation-based** forms of memory. Weight based memory is a result of synaptic plasticity, and is generally relatively long lasting (at least several 10's of minutes, and sometimes several decades, up to a whole lifetime). Activation-based memory is supported by ongoing neural activity, and is thus much more transient and fleeting, but also more flexible. Because weight-based memory exists at every modifiable synapse in the brain, it can manifest in innumerable ways. In this chapter, we focus on some of the most prominent types of memory studied by psychologists, starting with episodic memory, then looking at familiarity-based recognition memory, followed by weight-based priming, and activation-based priming. We'll look at more robust forms of activation-based memory, including working memory, in the *Executive Function* Chapter.

Probably most people have heard of the **hippocampus** and its critical role in episodic memory – the movie *Memento* for example does a great job of portraying what it is like to not have a functional hippocampus. We'll find out through our computational models *why* the hippocampus is so good at episodic memory – it has highly *sparse* patterns of neural activity (relatively few neurons active at a time), which allows even relatively similar memories to have very different, non-overlapping neural representations. These distinct neural patterns dramatically reduce *interference*, which is the primary nemesis of memory. Indeed, the highly distributed, overlapping representations in the neocortex – while useful for reasons outlined in the first half of this book – by themselves produce **catastrophic interference** when they are driven to learn too rapidly. But it is this rapid *one-shot* learning that is required for episodic memory! Instead, it seems that the brain leverages two specialized, **complementary learning systems** – the hippocampus for rapid encoding of new episodic memories, and the neocortex for slow acquisition of rich webs of semantic knowledge, which benefit considerably from the overlapping distributed learning and slower learning rates, as we'll see.

Countering the seemingly ever-present urge to oversimplify and modularize the brain, it is critical to appreciate that memory is a highly distributed phenomena, with billions of synapses throughout the brain being tweaked by any given experience. Several studies have shown preserved learning of new memories of relatively specific information in people with significant hippocampal damage – but it is critical to consider how these memories are cued. This is an essential aspect to remember about memory in general: whether a given memory can actually be retrieved depends critically on how the system is probed. We've probably all had the experience of a flood of memories coming back as a result of visiting an old haunt – the myriad of cues available enable (seemingly spontaneous) recall of memories that otherwise are not quite strong enough to rise to the surface. The memories encoded without the benefit of the hippocampus are weaker and more vague, but they do exist.

In addition to being highly distributed, memory in the brain is also highly interactive. Information that is initially encoded in one part of the brain can appear to "spread" to other parts of the brain, if those memories are reactivated and these other brain areas get further opportunities to learn them. A classic example is that episodic memories initially encoded in the hippocampus can be strengthened in the surrounding neocortical areas through repeated retrieval of those memories. This can even happen while we are sleeping, when patterns of memories experienced during the day have shown to be re-activated! Furthermore, influences of the prefrontal cortex system, and affective states, can significantly influence the encoding and retrieval of memory. Thus, far from the static "hard drive" metaphor from computers, memory in the brain is a highly dynamic, constantly evolving process that reflects the complexity and interactions present across all the areas of the brain.

Episodic Memory

We begin with episodic memory, because it is such a major part of our conscious lives, and really of our identities. For example, the movie *Total Recall,* loosely based on the Philip K. Dick novel We Can Remember

it for You Wholesale (wikipedia link), explores this connection between episodic memories and our sense of self. All people with a functioning hippocampus have this remarkable "tape recorder" constantly encoding everything that happens during our waking lives – we don't have to exert particular effort to recall what happened 20 minutes or a few hours ago – it is just automatically there. Most people end up forgetting the vast majority of the daily flux of our lives, retaining only the particularly salient or meaningful events.

However, a tiny percentage of otherwise seemingly "normal" people have Exceptional memory (wikipedia link), or *hyperthymesia*. Interestingly, it is not the hippocampus itself that differentiates these people from you and me – instead they are characterized by the obsessive rehearsal and retrieval of episodic memories, with areas of the basal ganglia apparently enlarged (which is associated with obsessive compulsive disorder (OCD)). As we'll see in the *Executive Function* Chapter, the basal ganglia participate not only in motor control and reinforcement learning, but also the reinforcement of updating and maintenance of active memory. This suggests that in normal human brains, the hippocampus has the raw ability to encode and remember every day of our lives, but most people just don't bother to rehearse these memories to the point where they can all be reliably retrieved. Indeed, a major complaint that these people with exceptional memory have is that they are unable to forget all the unpleasant stuff in life that most people just let go.

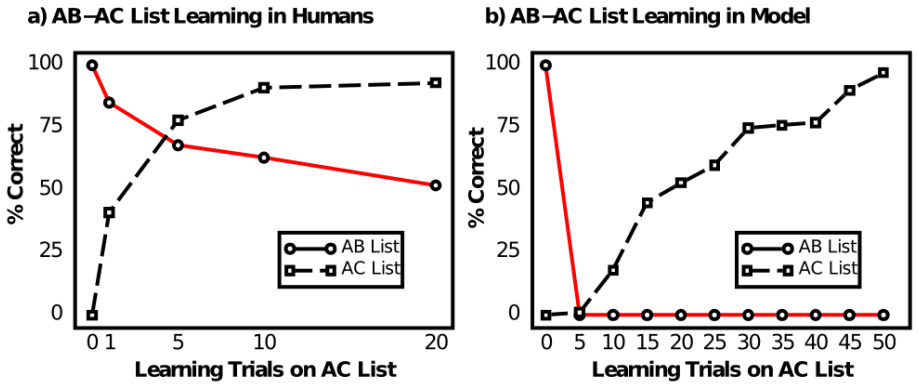


Figure 8.1: Data from humans (a) and a generic (cortical) neural network model (b) on the classic AB-AC list learning task, which generates considerable interference by re-pairing the A list items with new associates in the AC list after having first learned the AB list. People's performance on the AB items after learning the AC list definitely degrades (red line), but nowhere near as catastrophically as in the neural network model. Data reproduced from McCloskey and Cohen (1989).

So what exactly makes the hippocampus such an exceptionally good episodic memory system? Our investigation begins with failure. Specifically, the failure of a "generic" cortical neural network model of the sort we've been exploring in this textbook to exhibit any kind of useful episodic memory ability. This failure was first documented by (McCloskey and Cohen 1989), using a generic backpropagation network trained on the AB-AC paired associate list learning task (Barnes and Underwood 1960) (Figure 8.1). This task involves learning an initial list of arbitrary word pairs, called the AB list – for example:

- locomotive - dishtowel
- window - reason
- bicycle - tree
- ...

People are tested on their ability to recall the B associate for each A item, and training on the AB list ends when they achieve perfect recall. Then, they start learning the AC list, which involves new associates for the previous A items:

- locomotive - cloud
- window - book
- bicycle - couch
- ...

After 1, 5, 10, and 20 iterations of learning this AC list, people are tested on their ability to recall the

original *AB* items, without any additional training on those items. shows that there is a significant amount of interference on the AB list as a result of learning the AC items, due to the considerable overlap between the two lists, but even after 20 iterations through the AC items, people can still recall about 50% of the AB list. In contrast, (McCloskey and Cohen 1989) showed that the network model exhibited **catastrophic interference** – performance on the AB list went to 0% immediately. They concluded that this invalidated all neural network models of human cognition, because obviously people have much better episodic memory abilities.

But we'll see that this kind of whole-sale abandonment of neural networks is unjustified (indeed, the brain is a massive neural network, so there must be some neural network description of any phenomenon, and we take these kind of challenges as informative opportunities to identify the relevant mechanisms). Indeed, in the following exploration we will see that there are certain network parameters that reduce the levels of interference. The most important manipulation required is to increase the level of inhibition so that fewer neurons are active, which reduces the overlap between the internal representation of the AB and AC list items, thus allowing the system to learn AC without overwriting the prior AB memories. We'll then see that the hippocampal system exploits this trick to an extreme degree (along with a few others), making it an exceptionally good episodic memory system.

Exploration of Catastrophic Interference

Run the `abac` simulation from CCN Sims.

The Hippocampus and Pattern Separation / Pattern Completion

The hippocampus is specifically optimized to rapidly record episodic memories using highly **sparse** representations (i.e., having relatively few neurons active) that minimize overlap (through **pattern separation**) and thus interference. This idea is consistent with such a large quantity of data, that it is close to established fact (a rarity in cognitive neuroscience). This data includes the basic finding of episodic memory impairments (and particularly in pattern separation) that result from selective hippocampal lesions, the unique features of the hippocampal anatomy, which are so distinctive relative to other brain areas that they cry out for an explanation, and the vast repertoire of neural recording data from different hippocampal areas. We start with an overview of hippocampal anatomy, followed by the neural recording data and an understanding of how relatively sparse neural activity levels also results in pattern separation, which minimizes interference.

Hippocampal Anatomy

The anatomy of the hippocampus proper and the areas that feed into it is shown in Figure 8.2. The hippocampus represents one of two "summits" on top of the hierarchy of interconnected cortical areas (where the bottom are sensory input areas, e.g., primary visual cortex) – the other such summit is the prefrontal cortex explored in the *Executive Function* Chapter. Thus, it possesses a critical feature for an episodic memory system: access to a very high-level summary of everything of note going on in your brain at the moment. This information, organized along the dual-pathway dorsal vs. ventral pathways explored in the *Perception and Attention* Chapter, converges on the **parahippocampal (PHC)** (dorsal) and **perirhinal (PRC)** (ventral) areas, which then feed into the **entorhinal cortex (EC)**, and then into the hippocampus proper. The major hippocampal areas include the **dentate gyrus (DG)** and the areas of "ammon's horn" (cornu ammonis (CA) in latin), **CA3** and **CA1** (what happened to CA2? turns out it is basically the same as CA3 so we just use that label). All of these strange names have to do with the shapes of these areas, including the term "hippocampus" itself, which refers to the seahorse shape it has in the human brain (hippocampus is greek for seahorse)).

The basic episodic memory encoding story in terms of this anatomy goes like this. The high-level summary of everything in the brain is activated in EC, which then drives the DG and CA3 areas via the **perforant pathway** – the end result of this is a highly sparse, distinct pattern of neural firing in CA3, which represents the main "engram" of the hippocampus. The EC also drives activity in CA1, which has the critical feature of being able to then re-activate this same EC pattern all by itself (i.e., an *invertible mapping* or *auto-encoder* relationship between CA1 and EC). These patterns of activity then drive synaptic plasticity (learning) in all

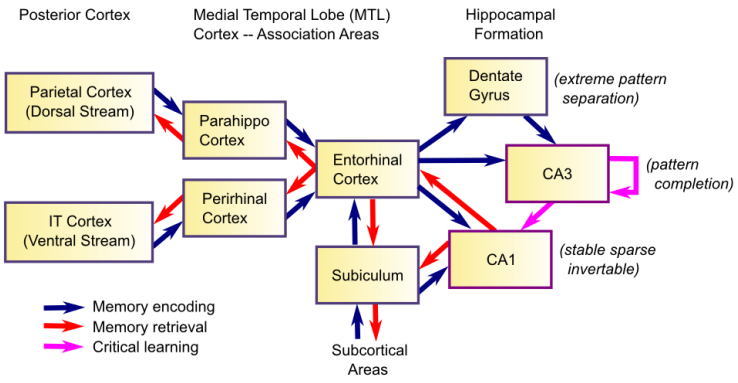


Figure 8.2: The hippocampus sits on "top" of the cortical hierarchy and can encode information from all over the brain, binding it together into an episodic memory. Dorsal (parahippocampal) and Ventral (perirhinal) pathways from posterior cortex converge into the entorhinal cortex, which is then the input and output pathway of the hippocampus proper, consisting of the dentate gyrus (DG) and areas of "ammon's horn" (cornu ammonis, CA) – CA3 and CA1. CA3 represents the primary "engram" for the episodic memory, while CA1 is an invertible encoding of EC, such that subsequent recall of the CA3 engram can activate CA1 and then EC, to reactivate the full episodic memory out into the cortex.

the interconnected synapses, with the most important being the synaptic connections among CA3 neurons (in the CA3 recurrent pathway), and the connections between CA3 and CA1 (the **Schaffer collateral** pathway). These plastic changes effectively "glue together" the different neurons in the CA3 engram, and associate them with the CA1 invertible pattern, so that subsequent retrieval of the CA3 engram can then activate the CA1, then EC, and back out to the cortex. Thus, the primary function of the hippocampus is to bind together all the disparate elements of an episode, and then be able to retrieve this *conjunctive memory* and reinstate it out into the cortex during recall. This is how a memory can come "flooding back" – it floods back from CA3 to CA1 to EC to cortex, reactivating something approximating the original brain pattern at the time the memory was encoded.

As noted in the introduction, every attempt to simplify and modularize memory in this fashion is inaccurate, and in fact memory encoding is distributed among all the neurons that are active at the time of the episode. For example, learning in the perforant pathway is important for reactivating the CA3 engram from the EC inputs (especially when they represent only a partial memory retrieval cue). In addition, learning all the way through the cortical pathways into and out of the hippocampus "greases" the retrieval process. Indeed, if a memory pattern is reactivated frequently, then these cortical connections can be strong enough to drive reactivation of the full memory, without the benefit of the hippocampus at all. We discuss this *consolidation* process in detail later. Finally, the retrieval process can be enhanced by controlled retrieval of memory using top-down strategies using the prefrontal cortex. We don't consider this aspect of controlled retrieval here, but it depends on a combination of activation and weight based memory analogous to some features we will explore in *Executive Function* Chapter.

Properties of Hippocampal Neurons: Sparseness, Pattern Separation

A representative picture of a critical difference between the hippocampus (CA3, CA1) and cortex is shown in Figure 8.3, where it is clear that CA3 and CA1 neurons fire much less often than those in the cortex (entorhinal cortex and subiculum). This is what we mean by *sparseness* in the hippocampal representation – for any given episode, only relatively few neurons are firing, and conversely, each neuron only fires under a very specific circumstance. In rats, these circumstances tend to be identifiable as spatial locations, i.e., *place cells*, but this is not generally true of primate hippocampus. This sparseness is thought to result from high levels of GABA inhibition in these areas, keeping many neurons below threshold, and requiring active neurons to receive a relatively high level of excitatory input to overcome this inhibition. The direct benefit of this sparseness is that the engrams for different episodes will overlap less, just from basic probabilities (Figure 8.4). For example, if the probability of a neuron being active for a given episode is 1% (typical of the

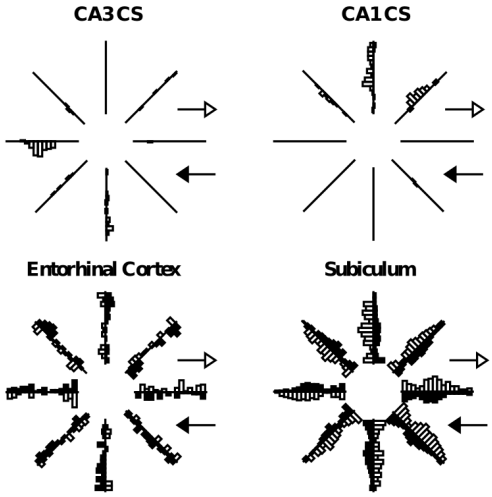


Figure 8.3: Comparison of activity patterns across different areas of the hippocampus, showing that CA fields (CA3, CA1) are much more sparse and selective than the cortical input areas (Entorhinal cortex (EC) and subiculum). This sparse, pattern separated encoding within the hippocampus enables it to rapidly learn new episodes while suffering minimal interference. Activation of sample neurons within each area are shown for a rat running on an 8 arm radial maze, with the bars along each arm indicating how much the neuron fired for each direction of motion along the arm. The CA3 neuron fires only for one direction in one arm, while EC has activity in all arms (i.e., a much more overlapping, distributed representation).

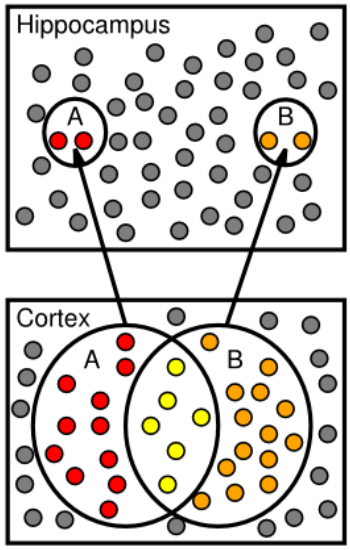


Figure 8.4: Schematic illustration of how more sparse activity levels can produce pattern separation, just because the odds of overlapping are that much lower. Graphically, this is evident in that the smaller circles (sparser activation) in the hippocampus are less likely to overlap than the larger ones in the cortex.

DG), then the probability for any two random episodes is that value squared, which is .01% (a very small number). In comparison, if the probability is higher, e.g., 25% (typical of cortex), then there is a 6.25% chance of overlap for two episodes. David Marr appears to have been the first one to point out this *pattern separation* property of sparse representations, in an influential paper (Marr 1971).

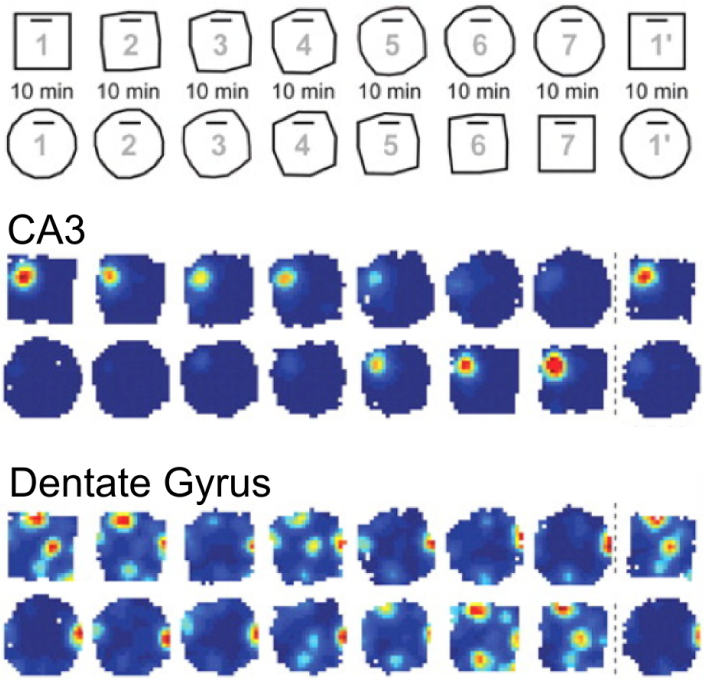


Figure 8.5: Pattern separation in the CA3 and Dentate Gyrus of the hippocampus, as a function of a rat's location in an environment that morphs gradually from a square into a circle, and vice-versa (indicated in top panel). The CA3 neuron shown here has two distinct "place cell" firing patterns, one for the square and one for the circle. In contrast, The DG neuron exhibits somewhat greater pattern separation by responding differentially in the middle of the morph sequence. Data from Leutgeb et al. (2007).

The connection between activity levels and pattern separation can also be observed within the hippocampus itself, by comparing the firing properties of DG vs. CA3 neurons, where DG neurons have the sparsest activity levels, even compared to the somewhat less sparse CA3 (roughly 2-5% activity level). Figure 8.5 from a study by (Leutgeb et al. 2007) shows that the DG exhibits more pattern separation than the CA3, as a function of systematic morphing of an environment from a square to a circle and back again. The DG neurons exhibit a greater variety of neural firing as a function of this environmental change, suggesting that they separate these different environments to a greater extent than the CA3. There are many other compelling demonstrations of pattern separation in various hippocampal areas relative to cortex, and in particular in DG relative to other areas (see e.g., the extensive work of Kesner on this topic).

Another factor that contributes to effective pattern separation is the broad and diffuse connectivity from EC to DG and CA3, via the perforant pathway. This allows many different features in EC to be randomly combined in DG and CA3, enabling them to be sensitive to combinations or *conjunctions* of inputs. Because of the high inhibitory threshold associated with sparse activations, this means a given neuron in these areas must receive significant excitation from multiple of these diffuse input sources. In other words, these neurons have **conjunctive representations**.

Pattern separation is important for enabling the hippocampus to rapidly encode novel episodes with a minimum of interference on prior learning, because the patterns of neurons involved overlap relatively little.

Pattern Completion: Cued Recall

While pattern separation is important for encoding new memories, this encoding would be useless unless these memories can be subsequently recalled. This recall process is also known as **pattern completion**, where a partial retrieval cue triggers the completion of the full original pattern associated with the memory. For example, if I cue you with the question: "did you go to summer camp as a kid?" you can pattern complete from this to memories of summer camp, or not, as the case may be. The amazing thing about human memory is that it is **content addressable memory** – any sufficiently specific subset of information can serve as a retrieval cue, enabling recovery of previously-encoded episodic memories. In contrast, memory in a computer is accessed by a memory address or a variable pointer, which has no relationship to the actual content stored in that memory. The modern web search engines like Google demonstrate the importance of content addressability, and function much like the human memory system, taking search terms as retrieval cues to find relevant "memories" (web pages) with related information. As you probably know from searching the web, the more specific you can make your query, the more likely you will retrieve relevant information – the same principle applies to human memory as well.

In the hippocampus, pattern completion is facilitated by the recurrent connections among CA3 neurons, which glues them together during encoding, such that a subset of CA3 neurons can trigger recall of the remainder. In addition, the synaptic changes during encoding in the perforant pathway make it more likely that the original DG and CA3 neurons will become reactivated by a partial retrieval cue.

Interestingly, there is a direct tension or tradeoff between pattern separation and pattern completion, and the detailed parameters of the hippocampal anatomy can be seen as optimizing this tradeoff (O'Reilly and McClelland 1994). Pattern separation makes it more likely that the system will treat the retrieval cue like a novel stimulus, and thus encode a new distinct engram pattern in CA3, instead of completing to the old one. Likewise, if the system is too good at pattern completion, it will reactivate old memories instead of encoding new pattern separated ones, for truly novel episodes. Although the anatomical parameters in our model do help to find a good balance between these different forces of completion and separation, it is also likely that the hippocampus benefits from strategic influences from other brain areas, e.g., prefrontal cortex executive control areas, to emphasize either completion or separation depending on whether the current demands require recall or encoding, respectively. We will explore this issue further in the *Executive Function* Chapter.

Exploration

To explore how the hippocampus encodes and recalls memories, using the AB-AC task, run the `hip` simulation in CCN Sims.

Complementary Learning Systems

As noted earlier, when McCloskey & Cohen first discovered the phenomenon of catastrophic interference, they concluded that neural networks are fatally flawed and should not be considered viable models of human cognition. This is the same thing that happened with (Minsky and Papert 1969), in the context of networks that lack a hidden layer and thus cannot learn more difficult mappings such as XOR (see the *Learning* Chapter for more details). In both cases, there are ready solutions to these problems, but people seem all too willing to seize upon an excuse to discount the neural network model of the mind. Perhaps it is just too reductionistic or otherwise scary to think that everything that goes on in your brain could really boil down to mere neurons... However, this problem may not be unique to neural networks – researchers often discount various theories of the mind, including Bayesian models for example, when they don't accord with some pattern of data. The trick is to identify when any given theory is fundamentally flawed given challenging data; the devil is in the details, and oftentimes there are ways to reconcile or refine an existing theory without "throwing out the baby with the bathwater".

Such musings aside, there are (at least) two possible solutions to the catastrophic interference problem. One would be to somehow improve the performance of a generic neural network model in episodic memory tasks, inevitably by reducing overlap in one way or another among the representations that form. The other would be to introduce a specialized episodic memory system, i.e., the hippocampus, which has

Complementary Learning Systems

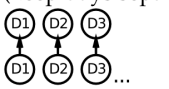
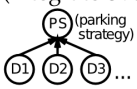
Goals:	Remember Specifics	Extract Generalities
Example:	Where is car parked?	Best parking strategy?
Need to:	Avoid interference	Accumulate experience
Solution:		
1.	Separate reps (keep days separate) 	Overlapping reps (integrate over days) 
2.	Fast learning (encode immediately)	Slow learning (integrate over days)
3.	Learn automatically (encode everything)	Task-driven learning (extract relevant stuff)
These are incompatible, need two different systems:		
System:	Hippocampus	Neocortex

Figure 8.6: Summary of the Complementary Learning Systems perspective on functional roles of Hippocampus vs. Neocortex, in context of memory about parking spaces. The hippocampus can rapidly encode in a relatively interference-free way where you parked your car today, as distinct from previous days. The neocortex in contrast can integrate across many different experiences (using a slow learning rate) to extract an overall parking strategy that reflects effects of many different factors on likelihood of finding parking in a given lot. The functional demands for these two different kinds of learning are in direct conflict, so the best overall functionality can be achieved by having two complementary learning systems, each separately optimized for these different functions.

parameters that are specifically optimized for low-interference rapid learning through pattern separation, while retaining the generic neural network functionality as a model of neocortical learning. The advantage of this latter perspective, known as the **complementary learning systems (CLS)** framework ((McClelland, McNaughton, and O'Reilly 1995; Norman and O'Reilly 2003)), is that the things you do to make the generic neural model better at episodic memory actually interfere with its ability to be a good model of neocortex. Specifically, neocortical learning for things like object recognition (as we saw in the *Perception* Chapter), and semantic inference (as we'll see in the *Language* Chapter) really benefit from highly overlapping distributed representations, and slow interleaved learning. These overlapping distributed representations enable patterns of neural activity to encode complex, high-dimensional similarity structures among items (objects, words, etc), which is critical for obtaining a "common sense" understanding of the world. Figure 8.6 summarizes this fundamental tradeoff between statistical or semantic learning (associated with the neocortex) and episodic memory (associated with the hippocampus).

Consistent with this basic tradeoff, people with exceptional episodic memory abilities (as discussed earlier) often suffer from a commensurate difficulty with generalizing knowledge across episodes. Even more extreme, autistic memory savants, who can memorize all manner of detailed information on various topics, generally show an even more profound lack of common sense reasoning and general ability to get by in the real world. In these cases, it was speculated that the neocortex also functions much more like a hippocampus, with sparser activity patterns, resulting in overall greater capacity for memorizing specifics, but correspondingly poor abilities to generalize across experiences to produce common sense reasoning (McClelland 2000).

Amnesia: Anterograde vs. Retrograde

Having seen how the intact hippocampus functions, you may be wondering what goes wrong to produce **amnesia.** The hollywood version of amnesia involves getting hit on the head, followed by a complete forgetting of everything you know (e.g., your spouse becomes a stranger). Then of course another good whack restores those memories, but not before many zany hijinks have ensued. In reality, there are many different sources of amnesia, and memory researchers typically focus on the kind that is caused by direct damage to the hippocampus and related structures, known as *hippocampal amnesia.* The most celebrated case of this is a person known to science as H.M. (Henry Molaison), who had his hippocampus removed to prevent otherwise intractable epilepsy, in 1957. He then developed the inability to learn new episodic information

(**anterograde amnesia**), as well as some degree of forgetting of previously learned knowledge (**retrograde amnesia**). But he remembered how to talk, the meanings of different words and objects, how to ride a bike, and could learn all manner of new motor skills. This was a clear indication that the hippocampus is critical for learning only some kinds of new knowledge.

More careful studies with HM showed that he could also learn new semantic information, but that this occurred relatively slowly, and the learned knowledge was more brittle in the way it could be accessed, compared to neurologically intact people. This further clarifies that the hippocampus is critical for episodic, but not semantic learning. However, for most people semantic information can be learned initially via the hippocampus, and then more slowly acquired by the neocortex over time. One indication that this process occurs is that HM lost his most recent memories prior to the surgery, more than older memories (i.e., a temporally-graded retrograde gradient, also know as a Ribot gradient). Thus, the older memories had somehow become *consolidated* outside of the hippocampus, suggesting that this gradual process of the neocortex learning information that is initially encoded in the hippocampus, is actually taking place. We discuss this process in the next section.

Certain drugs can cause a selective case of anterograde amnesia. For example, the benzodiazepines (including the widely-studied drug *midazolam*) activate GABA inhibitory neurons throughout the brain, but benzodiazepene (GABA-A) receptors are densely expressed in the hippocampus, and because of the high levels of inhibition, it is very sensitive to this. At the right dosage, this inhibition is sufficient to prevent synaptic plasticity from occurring within the hippocampus, to form new memories, but previously-learned memories can still be reactivated. This then gives rise to a more pure case of anterograde, without retrograde, amnesia. Experimentally, midazolam impairs hippocampal-dependent rapid memory encoding but spares other forms of integrative learning such as reinforcement learning (Hirshman, Passannante, and Arndt 2001; Michael J. Frank, O'Reilly, and Curran 2006).

Another source of amnesia comes from Korsakoff's syndrome, typically resulting from lack of vitamin B1 due to long-term alcoholism. This apparently affects parts of the thalamus and the mammillary bodies, which in turn influence the hippocampus via various neuromodulatory pathways, including GABA innervation from the medial septum, which can then influence learning and recall dynamics in the hippocampus.

Memory Consolidation from Hippocampus to Neocortex

Why do we dream? Is there something useful happening in our brains while we sleep, or is it just random noise and jumbled nonsensical associations? Can you actually learn a foreign language while sleeping? Our enduring fascination with the mysteries of sleep and dreaming may explain the excitement surrounding the idea that memories can somehow migrate from the hippocampus to the neocortex while we sleep. This process, known as **memory consolidation**, was initially motivated by the observation that more recent memories were more likely to be lost when people suffer from acquired amnesia, as in the case of H.M. discussed above. More recently, neural recordings in the hippocampus during wakefulness and sleep have revealed that patterns of activity that occur while a rat is running a maze seem to also be reactivated when the animal is then asleep. However, the measured levels of reactivation are relatively weak compared to the patterns that were active during the actual behavior, so it is not clear how strong of a learning signal could be generated from this. Furthermore, there is considerable controversy over the presence of the temporally-graded retrograde gradients in well-controlled animal studies, raising some doubts about the existence of the consolidation phenomenon in the first place. Nevertheless, on balance it seems safe to conclude that this process does occur at least to some extent, in at least some situations, even if not fully ubiquitous. In humans, slow wave oscillations during non-REM sleep are thought to be associated with memory consolidation. Indeed, one recent study showed that external induction of slow wave oscillations during sleep actually resulted in enhanced subsequent hippocampal-dependent memories for items encoded just prior to sleep (Marshall et al. 2006).

One prediction from the complementary learning systems perspective regarding this consolidation process is that the information encoded in the neocortex will be of a different character to that initially encoded by the hippocampus, due to the very different nature of the learning and representations in these two systems. Thus, to the extent that episodic memories can be encoded in the neocortex, they will become more "semanticized" and generalized, integrating with other existing memories, as compared to the more distinct and crisp pattern

separated representations originally encoded in the hippocampus. Available evidence appears to support this idea, for example by comparing the nature of the intact memories from hippocampal amnesics to neurologically intact controls.

Role of Space in the Hippocampus

A large amount of research on the hippocampus takes place in the rat, and spatial navigation is one of the most important behavioral functions for a rat. Thus, it is perhaps not too surprising that the rat hippocampus exhibits robust **place cell** firing (as shown in the figures above), where individual DG, CA3 and CA1 neurons respond to a particular location in space. A given neuron will have a different place cell location in different environments, and there does not appear to be any kind of topography or other systematic organization to these place cells. This is consistent with the random, diffuse nature of the perforant pathway projections into these areas, and the effects of pattern separation.

More recently, spatial coding in the entorhinal cortex has been discovered, in the form of **grid cells**. These grid cells form a regular hexagonal lattice or grid over space, and appear to depend on various forms of oscillations. These grid cells may then provide the raw spatial information that gets integrated into the place cells within the hippocampus proper. In addition, **head direction cells** have been found in a number of different areas that project into the hippocampus, and these cells provide a nice *dead reckoning* signal about where the rat is facing based on the accumulation of recent movements.

The combination of all these cell types provides a solid basis for spatial navigation in the rat, and various computational models have been developed that show how these different signals can work together to support navigation behavior. An exploration model of this domain will be available in a future edition.

Theta Waves

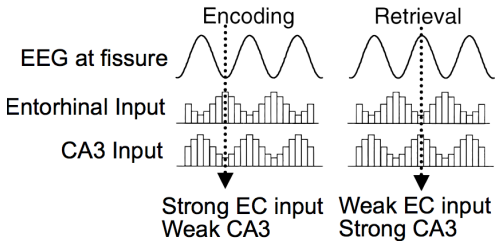


Figure 8.7: Different areas of the hippocampal system fire out of phase with respect to the overall theta rhythm, producing dynamics that optimize encoding vs. retrieval. We consider the strength of the EC and CA3 inputs to CA1. When the EC input is strong and CA3 is weak, CA1 can learn to encode the EC inputs. This serves as a plus phase for an error-driven learning dynamic in the Leabra framework. When CA3 is strong and EC is weak, the system recalls information driven by prior CA3 -> CA1 learning. This serves as a minus phase for Leabra error-driven learning, relative to the plus phase encoding state. (adapted from Hasselmo et al, 2002)

An important property of the hippocampus is an overall oscillation in the rate of neural firing, in the so-called *theta* frequency band in rats, which ranges from about 8-12 times per second. As shown in Figure 8.7, different areas of the hippocampus are out of phase with each other with respect to this theta oscillation, and this raises the possibility that these phase differences may enable the hippocampus to learn more effectively. Hasselmo and colleagues argued that this theta phase relationship enables the system to alternate between encoding of new information vs. recall of existing information (Hasselmo, Bodelon, and Wyble 2002). This is an appealing idea, because as we discussed earlier, there can be a benefit by altering the hippocampal parameters to optimize encoding or retrieval based on various other kinds of demands.

The emergent software now supports an extension to this basic theta encoding vs. retrieval idea that enables Leabra error-driven learning to shape two different pathways of learning in the hippocampus, all within one standard trial of processing (Ketz, Morkonda, and O'Reilly 2013). Each pathway has an effective minus and plus phase activation state (although in fact they share the same plus phase). The main pathway, trained on the standard minus to plus phase difference, involves CA3-driven recall of the corresponding CA1 activity pattern, which can then reactivate EC and so on out to cortex. The second pathway, trained using a

special initial phase of settling within the minus phase, is the CA1 <-> EC invertible auto-encoder, which ensures that CA1 can actually reactivate the EC if it is correctly recalled. In our standard hippocampal model explored previously, this auto-encoder pathway is trained in advance on all possible sub-patterns within a single subgroup of EC and CA1 units (which we call a "slot"). This new model suggests how this auto-encoder can instead be learned via the theta phase cycle. See the Appendix *Hippocampus Theta Phase* for details on this *theta phase* version of the hippocampus.

Theta oscillations are also thought to play a critical role in the grid cell activations in the EC layers, and perhaps may also serve to encode temporal sequence information, because place field activity firing shows a *theta phase procession*, with different place fields firing at different points within the unfolding theta wave. We will cover these topics in greater detail in a subsequent revision.

The Function of the Subiculum

The subiculum is often neglected in theories of hippocampal function, and yet it likely plays various important roles. Anatomically, it is situated in a similar location as the entorhinal cortex (EC) relative to the other hippocampal areas, but instead of being interconnected with neocortical areas, it is interconnected more directly with subcortical areas (Figure 8.2). Thus, by analogy to the EC, we can think of it as the input/output pathway for subcortical information to/from the hippocampus. One very important function that the subiculum may perform is computing the relative novelty of a given situation, and communicating this to the midbrain dopamine systems and thence to basal ganglia, to modulate behavior appropriately (Lisman and Grace 2005). Novelty can have complex affective consequences, being both anxiogenic (anxiety producing) and motivational for driving further exploration, and generally increases overall arousal levels. The hippocampus is uniquely capable of determining how novel a situation is, taking into account the full *conjunction* of the relevant spatial and other contextual information. The subiculum could potentially compute novelty by comparing CA1 and EC states during the recall phase of the theta oscillation, for example, but this is purely conjecture at this point. Incorporating this novelty signal is an important goal for future computational models.

Familiarity and Recognition Memory

Stepping back now from the specific memory contributions of the hippocampus, we consider a broader perspective of how the hippocampal system fits into the larger space of human memory capacities. One of the most important questions that researchers have focused on here is whether the neocortex can contribute anything at all to single trial episodic memory. Does a single exposure to a given stimulus leave a big enough trace anywhere in the cortex so as to influence overt behavior? As noted previously, we feel confident that synapses throughout the brain are likely to be affected by every learning experience, but is neocortical learning simply too slow, and the representations too overlapping, to produce a behaviorally significant change from a single experience?

A large body of data suggests that indeed the neocortex can support episodic memory traces, but that they have very different properties compared to those supported by the hippocampus. Specifically, it seems that the perirhinal cortex can produce a useful *familiarity* signal, that indicates in a very coarse manner whether a given stimulus was experienced recently or not. This familiarity signal can be contrasted with the *recollective* memory signal provided by the hippocampus: a full explicit recall of the details of the previous episode when the item was last experienced. The familiarity signal is instead more like a single graded value that varies in intensity depending on how strongly familiar the item is. One hypothesis about the neural basis for this signal is the *sharpness* of the representations in perirhinal cortex – single trials of learning in a generic cortical model leave measurable traces on the overall pattern of neural activity, such that the contrast between strongly active and more weakly active neurons is enhanced (Norman & O'Reilly, 2003). This results from the basic self-organizing learning dynamic we observed in the *Learning* Chapter, where the most strongly activated neurons strengthen their synaptic connections, and thus are better able to out-compete other neurons.

Interestingly, people can obtain subjective conscious access to this familiarity signal, and use it to make overt, conscious evaluations of how familiar they think an item is. The neural mechanism for this explicit readout of a sharpness signal has not been identified. This main challenge here is identifying why signals

in perirhinal cortex are consciously accessible, while similar such signals in other neocortical areas do not appear to be accessible to consciousness (as we discuss in the next section).

This combination of hippocampal recall and perirhinal familiarity memory systems is called a *dual process* model of recognition memory, and after many years of controversy, it is now widely accepted in the field. Some of the data consistent with this dual process model include preserved familiarity signals in people with substantial hippocampal lesions, and a variety of neuroimaging and behavioral studies that have been able to distinguish between these two memory signals in various ways.

Priming: Weight and Activation-Based

Moving further afield from the hippocampus and surrounding cortical areas (e.g., the familiarity signal in the perirhinal cortex), can perceptual and other association cortex areas make useful memory contributions based on single or small numbers of exposures? The answer here is also in the affirmative, but unlike the familiarity signal, these memory traces remain almost entirely below the radar of conscious awareness – scientists can measure memory effects in terms of various behavioral measures, but we are not subjectively aware of having these memories. The general term for this form of memory is **priming,** because the main behavioral manifestation is a speedup in reaction time, or an increased probability of making a particular behavioral response – as if the "pump is being primed" by these memory traces. Indeed, we think of the slow incremental neocortical learning effects as doing exactly this pump priming level of tweaking to the underlying neural representations. Only sustained changes over many experiences can truly reshape these more stable neural representations in more dramatic ways. And as we get older, it seems that perhaps the learning rate gets slower, making it even more difficult to fundamentally reshape the most basic neocortical representations.

In addition to the subtle effects of slow learning changes, priming can also result from residual activation – neural firing that persists from previously processed information. Thus, we can distinguish between weight-based priming and activation-based priming. As might be expected, activation-based priming is very short-lived, disappearing as soon as the neural firing dissipates. By contrast, weight-based priming can be remarkably persistent, with some cases of priming lasting a year or more, from a single exposure! This kind of behavioral result puts strong constraints on the stability of synaptic plasticity – various computational models introduce forms of synaptic weight decay, but this seems inconsistent with the extreme durability of priming, and of our long-term memories more generally.

One behavioral paradigm used to reveal priming effects is called *stem completion*. Here, the first letters of a word are presented, and the participant is asked to complete the stem with the first word that comes to mind. For example, you might see stems like this:

- win____
- let____

and respond with words like "window" or "winter", "letter" or "lettuce". The priming effect is revealed by first exposing people to one of the possible words for these stems, often in a fairly disguised, incidental manner, and then comparing how much this influences the subsequent likelihood of completing the stem with it. By randomizing which of the different words people are exposed to, you can isolate the effects of prior exposure relative to whatever baseline preferences people might otherwise have. We know that those priming effects are not due to learning in the hippocampus, because they remain intact in people with hippocampal lesions.

Exploration

You can explore both weight-based and activation-based priming on a simple stem-completion like task, using a very generic cortical learning model, in the `priming` simulation in CCN Sims.

Appendix

- *Hippocampus Theta Phase:* theta phase learning version of the hippocampus.

Hippocampus Theta Phase

This appendix provides more information about the theta phase hippocampus implementation (Ketz, Morkonda, and O'Reilly 2013), which is used in the `hip` exploration. See leabra/hip on github for the source code and more details about the current implementation.

Here are the three phases of activation dynamics in the network:

- First half of minus phase: the EC input layer drives CA1, which then drives the EC output layer, and CA1 is not influenced by CA3 (in the theta cycle, CA3 is inhibited, but we actually just set its effective weight scale for influencing CA1 to 0). This is a minus phase for training the EC <-> CA1 auto encoder pathway.
- Second half of the minus phase: CA3 now influences CA1 to drive recall, while EC input does not drive CA1. This state at the end of settling is the regular minus phase, which drives learning relative to the plus phase, to train the CA3 -> CA1 recall pathway.
- Plus phase: the EC output layer units are activated directly by the EC input layer activities, such that EC output learns to reproduce the EC input pattern, and CA1 is also in the same state as the first half of the minus phase, where it is being driven by EC input but not CA3. Thus, the target CA1 activity pattern for the CA3 -> CA1 recall connections is the state that properly recalls EC input on the EC output layer, and similarly this same pattern is the target for the EC <-> CA1 auto encoder.

Chapter 9: Language

Language involves almost every part of the brain, as covered in other chapters in the text:

- *Perception and attention:* language requires the perception of words from auditory sound waves, and written text. Attention is critical for pulling out individual words on the page, and individual speakers in a crowded room. In this chapter, we see how a version of the object recognition model from the perception chapter can perform written word recognition, in a way that specifically leverages the spatial invariance property of this model.
- *Motor control*: Language production obviously requires motor output in the form of speech, writing, etc. Fluent speech depends on an intact cerebellum, and the basal ganglia have been implicated in a number of linguistic phenomena.
- *Learning and memory:* early word learning likely depends on episodic memory in the hippocampus, while longer-term memory for word meaning depends on slow integrated learning in the cortex. Memory for recent topics of discourse and reading (which can span months in the case of reading a novel) likely involves the hippocampus and sophisticated semantic representations in temporal cortex.
- *Executive Function:* language is a complex mental facility that depends critically on coordination and working memory from the prefrontal cortex (PFC) and basal ganglia – for example encoding syntactic structures over time, pronoun binding, and other more transient forms of memory.

One could conclude from this that language is not particularly special, and instead represents a natural specialization of *domain general* cognitive mechanisms. Of course, people have specialized articulatory apparatus for producing speech sounds, which are not shared by other primate species, but one could argue that everything on top of this is just language infecting pre-existing cognitive brain structures. Certainly reading and writing is too recent to have any evolutionary adaptations to support it (but it is also the least "natural" aspect of language, requiring explicit schooling, compared to the essentially automatic manner in which people absorb spoken language).

But language is fundamentally different from any other cognitive activity in a number of important ways:

- **Symbols** – language requires thought to be reduced to a sequence of symbols, transported across space and time, to be reconstructed in the receiver's brain.
- **Syntax** – language obeys complex abstract regularities in the ordering of words and letters/phonemes.
- **Temporal extent and complexity** – language can unfold over a very long time frame (e.g., Tolstoy's *War and Peace*), with a level of complexity and richness conveyed that far exceeds any naturally occurring experiences that might arise outside of the linguistic environment. If you ever find yourself watching a movie on an airplane without the sound, you'll appreciate that visual imagery represents the lesser half of most movie's content (the interesting ones anyway).
- **Generativity** – language is "infinite" in the sense that the number of different possible sentences that could be constructed is so large as to be effectively infinite. Language is routinely used to express new ideas. You may find some of those here.
- **Culture** – much of our intelligence is imparted through cultural transmission, conveyed through language. Thus, language shapes cognition in the brain in profound ways.

The "special" nature of language, and its dependence on domain-general mechanisms, represent two poles in the continuum of approaches taken by different researchers. Within this broad span, there is plenty of room for controversy and contradictory opinions. Noam Chomsky famously and influentially theorized that we are all born with an innate universal grammar, with language learning amounting to discovering the specific parameters of that language instance. On the other extreme, connectionist language modelers such as Jay McClelland argue that completely unstructured, generic neural mechanisms (e.g., backpropagation networks) are sufficient for explaining (at least some of) the special things about language.

Our overall approach is clearly based in the domain-general approach, given that the same general-purpose neural mechanisms used to explore a wide range of other cognitive phenomena are brought to bear on language here. However, we also think that certain features of the PFC / basal ganglia system play a special role in symbolic, syntactic processing. At present, these special contributions are only briefly touched upon here, and elaborated just a bit more in the executive function chapter, but future plans call for further elaboration. One

hint at these special contributions comes from *mirror neurons* discovered in the frontal cortex of monkeys, in an area thought to be analogous to Broca's area in humans – these neurons appear to encode the intentions of actions performed by other people (or monkeys), and thus may constitute a critical capacity to understand what other people are trying to communicate.

We start as usual with a biological grounding to language, in terms of particularly important brain areas and the biology of speech. Then we look at the basic perceptual and motor pathways in the context of reading, including an account of how damage to different areas can give rise to distinctive patterns of acquired dyslexia. We explore a large-scale reading model, based on our object recognition model from the perception chapter, that is capable of pronouncing the roughly 3,000 monosyllabic words in English, and generalizing this knowledge to nonwords. Next, we consider the nature of semantic knowledge, and see how a self-organizing model can encode word meaning in terms of the statistics of word co-occurrence, as developed in the Latent Semantic Analysis (LSA) model. Finally, we explore syntax at the level of sentences in the Sentence Gestalt model, where syntactic and semantic information are integrated over time to form a "gestalt" like understanding of sentence meaning.

Biology of Language

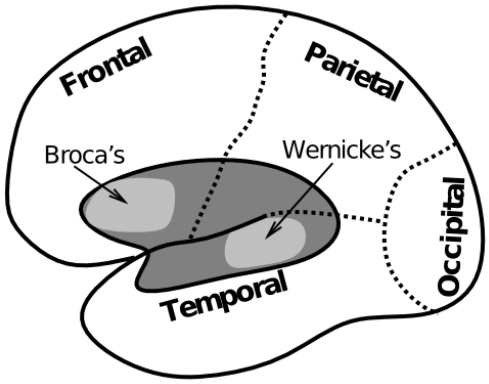


Figure 9.1: Brain areas associated with two of the most well-known forms of aphasia, or deficit in speech produced by damage to these areas. Broca's aphasia is associated with impaired syntax but intact semantics, while Wernicke's is the opposite. This makes sense given their respective locations in brain: temporal cortex for semantics, and frontal cortex for syntax.

The classic "textbook" brain areas for language are **Broca's** and **Wernicke's** areas (Figure 9.1), which have been associated with syntax and semantics, respectively. For example, a person who suffers a stroke or other form of damage to Wernicke's area can produce fluent, syntactically-correct speech, which is essentially devoid of meaning. Here is one example:

> *"You know that smoodle pinkered and that I want to get him round and take care of him like you want before"*

which apparently was intended to mean: "The dog needs to go out so I will take him for a walk."

In contrast, a person with damage to Broca's area has difficulty producing syntactically correct speech output, typically producing single content words with some effort, e.g., "dog. . . .walk".

The more modern term for Broca's aphasia is *expressive aphasia*, indicating a primary deficit in expressing speech. Comprehension is typically intact, although interestingly there can be deficits in understanding more syntactically complex sentences. Wernicke's aphasia is known as *receptive aphasia*, indicating a deficit in comprehension, but also expression of meaning.

Biologically, the locations of the damage associated with these aphasias are consistent with what we know about these areas more generally. The ventral posterior area of frontal cortex known as Broca's area (corresponding to Brodmann's areas 44 and 45) is adjacent to the primary motor area associated with control over the mouth, and thus it represents supplementary motor cortex for vocal output. Even though

Broca's patient's can physically move their mouths and other articulatory systems, they cannot perform the complex sequencing of these motor commands that is necessary to produce fluid speech. Interestingly, these higher-order motor control areas also seem to be important for syntactic processing, even for comprehension. This is consistent with the idea that frontal cortex is important for temporally-extended patterning of behavior according to increasingly complex plans as one moves more anterior in frontal cortex.

The location of Wernicke's area in temporal cortex is sensible, given that we know that the temporal lobe represents the semantic meanings of objects and other things.

There are still some controversies about the exact nature of damage required to produce each of these aphasias (and likely a large amount of individual variability across people as well), but the basic distinction between these broad areas remains quite valid.

The Articulatory Apparatus and Phonology

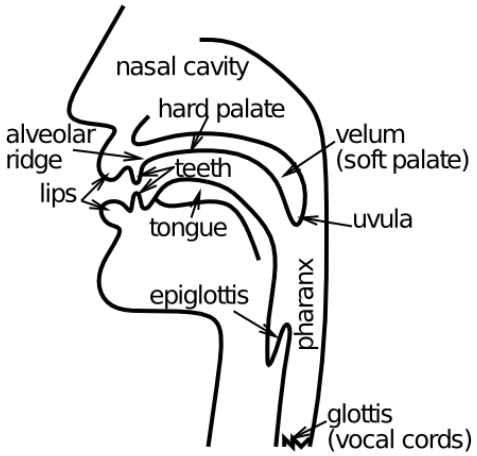


Figure 9.2: The different components of the vocal tract, which are important for producing the range of speech sounds that people can produce.

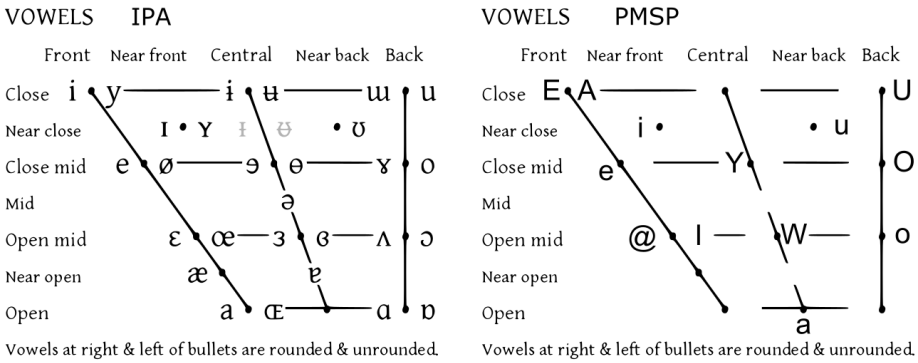


Figure 9.3: Left panel: International Phonological Alphabet (IPA) for vowels, as a function of where the tongue is positioned (front vs. back, organized horizontally in figure), and the shape of the lips (vertical axis in figure) – these two dimensions define a space of vowel sounds. Right panel: Version of IPA vowel space with vowel labels used by PMSP and in our simulations – these are all standard roman letters and thus easier to manipulate in computer programs. Only the subset present in English is used.

The vocal tract in people (Figure 9.2) is capable of producing a wide range of different speech sounds, by controlling the location and manner in which sound waves are blocked or allowed to pass. There are two basic categories of speech sounds: vowels and consonants. Vowels occur with unobstructed airflow (you can sing a

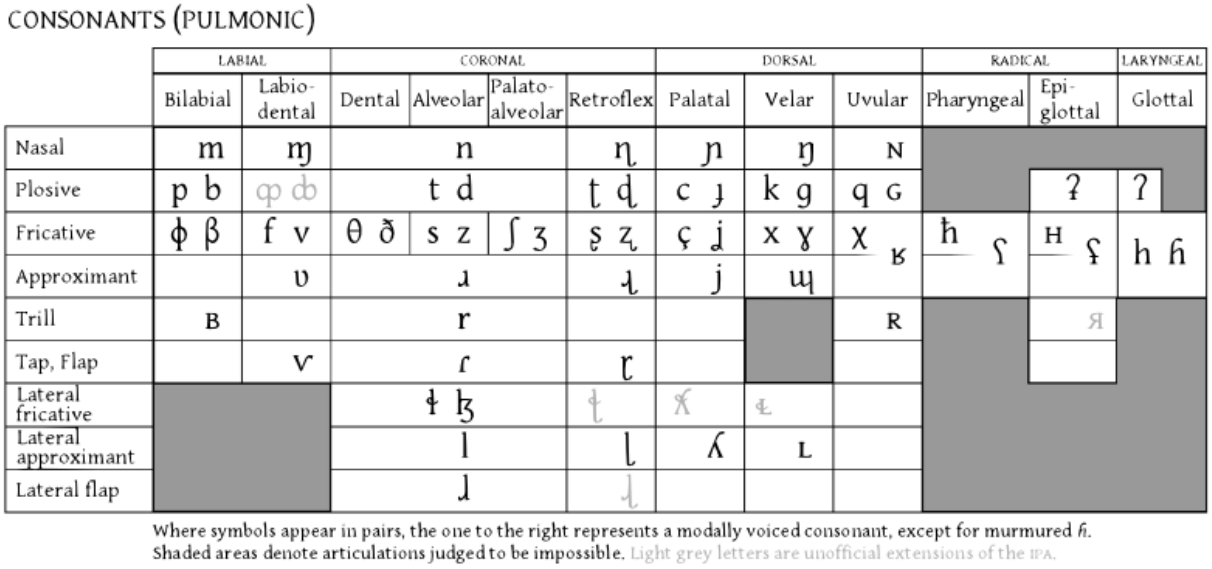


Figure 9.4: International Phonological Alphabet (IPA) for consonants, which are defined in terms of the location where the flow of air is restricted (place, organized horizontally in the table) and the manner in which it is restricted (plosive, fricative, etc, organized vertically).

vowel sound over an extended period), and differ in the location of the tongue and lips (Figure 9.3). For example, the long "E" vowel sound as in "seen" is produced with the tongue forward and the lips relatively closed. Consonants involve the blockage of airflow, in a variety of locations, and with a variety of different manners (Figure 9.4). The "s" consonant is a "fricative" (friction-like obstruction of the sound) with the tongue placed at the aveloar ridge. It is also unvoiced, which means that the vocal chords are not vibrating for it – the "z" sound is just like an "s" except it is voiced.

To see a video of the movements of the tongue in vocal output, see this YouTube link.

We'll take advantage of these phonological features in the output of our detailed reading model – using these features ensures that the spelling-to-sound correspondences actually capture the real phonological structure of the English language (at least at a fairly abstract level). A more detailed motor model of speech output developed by Frank Guenther, which we hope to include in our models at some point, can be found here.

Reading and Dyslexia in the Triangle Model

The first language model we explore simulates the major pathways involved in reading, according to the so-called *triangle model* (9.5) (Plaut and Shallice 1993). This model provides a basic understanding of the functional roles of visual perception of written words (*orthography*), spoken motor output of word *phonology*, and *semantic* representations of word meaning in between. This set of language pathways is sufficient to simulate the processes involved in reading words aloud, and damage to these pathways can simulate the critical features of different types of acquired dyslexia. Acquired dyslexia, which results from strokes or other brain damage, is distinct from *developmental* dyslexia, which is the more common form that many people associate with the term *dyslexia* (which generically refers to any form of reading impairment).

There are three major forms of acquired dyslexia that can be simulated with the model:

- **Phonological** – characterized by difficulty reading nonwords (e.g., "nust" or "mave"). This can be produced by damage to the direct pathway between orthography and phonology (there shouldn't be any activation in semantics for nonwords), such that people have difficulty mapping spelling to sound according to learned regularities that can be applied to nonwords. We'll explore this phenomenon in greater detail in the next simulation.
- **Deep** – is a more severe form of phonological dyslexia, with the striking feature that people sometimes

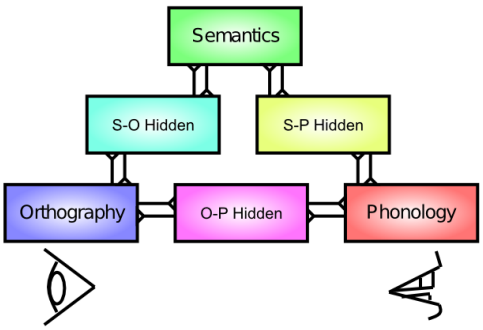


Figure 9.5: Triangle model of reading pathways: Visual word input (orthography) can produce speech output of the word either directly via projections to phonology (direct path), or indirectly via projections to semantics that encode the meanings of words. There is no single "lexicon" of words in this model – word representations are instead distributed across these different pathways. Damage to different pathways can account for properties of acquired dyslexia.

make semantic substitutions for words, pronouncing the word "orchestra" as "symphony" for example. There are also *visual* errors, so-named because they seem to reflect a misperception of the word inputs (e.g, reading the word "dog" as "dot"). Interestingly, we'll see how more significant damage to the direct pathway can give rise to this profile – the semantic errors occur due to everything going through the semantic layer, such that related semantic representations can be activated. In the normal intact brain, the direct pathway provides the relevant constraints to produce the actual written word, but absent this constraint, an entirely different but semantically related word can be output.

- **Surface** – here nonword reading is intact, but access to semantics is impaired (as in Wernicke's aphasia), strongly implicating a lesion in the semantics pathway. Interestingly, pronunciation of exception words (e.g., "yacht") is impaired. This suggests that people typically rely on the semantic pathway to "memorize" how to pronounce odd words like yacht, and the direct pathway is used more for pronouncing regular words.

That these different forms of dyslexia can be reliably observed in different patients, and fit so well with expected patterns of reading deficits according to the triangle model, provides a strong source of support for the validity of the model. It would be even more compelling if the specific foci of damage associated with these different forms of dyslexia make sense anatomically according to the mapping of the triangle model onto brain areas.

Exploration

Run `dyslexia` from CCN Sims for the simulation of the triangle model and associated forms of dyslexia. This model allows you to simulate the different forms of acquired dyslexia, in addition to normal reading, using the small corpus of words as shown in Figure 9.6. In the next section, we expand upon the direct pathway and examine nonword reading, which requires a much larger corpus of words to acquire the relevant statistical regularities that support generalization.

Spelling to Sound Mappings in Word Reading

We now zoom in on the direct pathway between visual word inputs (orthography) and verbal speech output (phonology), using a much larger set of words comprising most of the monosyllabic words in English (nearly 3,000 words). By learning on such a large selection of words, sampled according to their frequency of occurrence in English, the network has a chance to extract the "rules" that govern the mapping between spelling and sound in English (such as they are), and thus be able to successfully pronounce *nonwords*.

English is a particularly difficult language from a pronunciation perspective, as anyone knows who has tried to acquire it as a second language. There are very few (if any) absolute rules. Everything is more of a *partial, context-dependent regularity*, which is also called a *subregularity*. For example, compare the

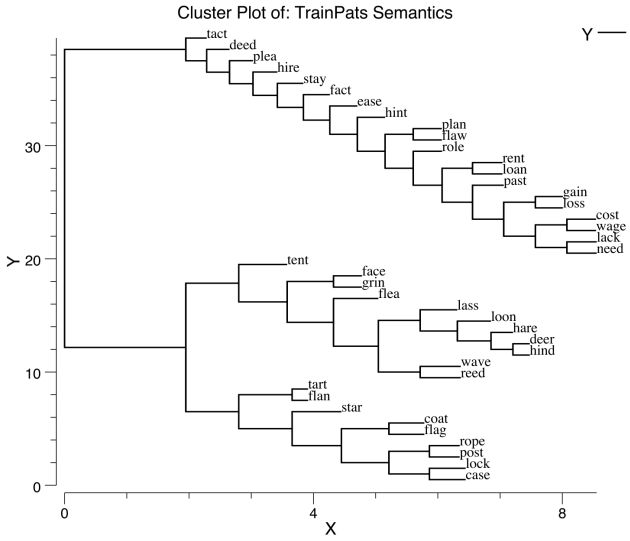


Figure 9.6: Cluster plot of semantic similarity for words in the simple triangle model of reading and dyslexia. Words that are semantically close (e.g., within the same terminal cluster) are sometimes confused for each other in simulated deep dyslexia.

pronunciation of the letter *i* in *mint* and *hint* (short *i* sound) to that in *mind* and *find* (long *I* sound). The final consonant (*t* vs. *d*) determines the pronunciation, and of course there are always exceptions such as *pint* (long *I* sound).

One way to classify how strong a regularity is, is to count how many other letters the pronunciation depends upon. A complete exception like *pint* or *yacht* depends on *all* the letters in the word, while *mint* vs. *mind* depends on one other letter in the word (the final *t* or *d*). There are many silent letter examples, such as the final *e* in many words. A nice subregularity is the letter *m*, which depends on whether there is an *n* next to it, in which case it goes silent, as in *damn*, *column*, or *mnemonic*. Many other consonants can be silent with varying degrees of subregularity, including *b* (*debt*), *d* (*handsome*), *h* (*honest*), *l* (*halve*), *p* (*coup*), *r* (*iron*), *s* (*aisle*), *t* (*castle*), *w* (*sword*), and *z* (*rendezvous*).

Another factor that determines how much context is required to pronounce a given letter is the preponderance of multi-letter groups like *th* (*think*), which have a particular regular pronunciation that differs from the individual letters separately. Other examples of these include: *sch* (*school*), *tch* (*batch*), *gh* (*ghost*), *ght* (*right*), *kn* (*knock*), *ph* (*photo*), *wh* (*what*). One of the most context sensitive set of letters is the *ough* group, as in *though, tough, cough, plough, through, nought*, where the pronunciation varies widely.

So English is a mess. The constructed word *ghoti* is a famous example of how crazy it can get. It is pronounced "fish", where the *gh* is an *f* sound as in *tough*, *o* is an *i* sound as in *women*, and *ti* is a *sh* sound as in *nation*.

For any system to be able to have any chance of producing correct pronunciation of English, it must be capable of taking into account a range of context around a given letter in a word, all the way up to the entire word itself. An influential early approach to simulating spelling to sound in a neural network (Seidenberg and McClelland 1989) used a so-called *Wickelfeature* representation (named after Wayne Wickelgren), where the written letters were encoded in pairs of three. For example, the word "think" would be encoded as *thi*, *hin*, and *ink*. This is good for capturing context, but it is a bit rigid, and doesn't allow for the considerable amount of regularity in individual letters themselves (most of the time, an *m* is just an *m*). As a result, this model did not generalize very well to nonwords, where letters showed up in different company than in the real words used in training. A subsequent model by (Plaut et al. 1996) (hereafter *PMSP*) achieved good nonword generalization by representing input words through a hand-coded combination of individual letter units and useful multi-letter contexts (e.g., a *th* unit).

We take a different approach in our spelling-to-sound model (Figure 9.7), leveraging ideas from the object recognition model that was explored in the *Perception* Chapter. Specifically, we saw that the object

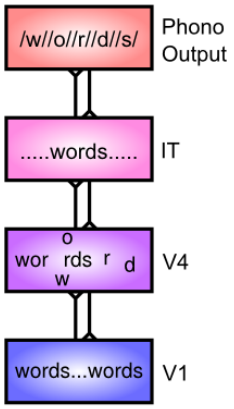


Figure 9.7: Word reading as a process of spatially invariant object recognition. Words show up in different locations in the input, and the next level up, equivalent to the V4 level in the object recognition model, extracts more complex combinations of letters, while also developing more invariant representations that integrate individual letters or multi-letter features over multiple different locations. The IT level representation then has a fully spatially invariant representation of the word (as a distributed representation integrating over individual letters and letter groups), which then provides a nice mapping to the phonological output.

recognition model could learn to build up increasingly complex combinations of features, while also developing spatial invariance, over multiple levels of processing in the hierarchy from V1 through IT. In the context of word recognition, these complex features could include combinations of letters, while spatial invariance allows the system to recognize that an *m* in any location is the same as any other *m* (most of the time).

One compelling demonstration of the importance of spatial invariance in reading comes from this example, which made the rounds in email a few years ago:

> I cnduo't bvleiee taht I culod aulaclty uesdtannrd waht I was rdnaieg. Unisg the icndeblire pweor of the hmuan mnid, aocdcrnig to rseecrah at Cmabrigde Uinervtisy, it dseno't mttaer in waht oderr the lterets in a wrod are, the olny irpoamtnt tihng is taht the frsit and lsat ltteer be in the rhgit pclae. The rset can be a taotl mses and you can sitll raed it whoutit a pboerlm. Tihs is bucseae the huamn mnid deos not raed ervey ltteer by istlef, but the wrod as a wlohe. Aaznmig, huh? Yaeh and I awlyas tghhuot slelinpg was ipmorantt! See if yuor fdreins can raed tihs too.

Clearly this is more effortful than properly spelled text, but the ability to read it at all indicates that just extracting individual letters in an invariant manner goes a long way.

To test the performance of this object-recognition based approach, we ran it through a set of different standard sets of nonwords, several of which were also used to test the PMSP model. The results are shown in Table 9.1.

- **Glushko regulars** – nonwords constructed to match strong regularities, for example *nust*, which is completely regular (e.g., *must*, *bust*, *trust*, etc).
- **Glushko exceptions** – nonwords that have similar English exceptions and conflicting regularities, such as *bint* (could be like *mint*, but also could be like *pint*). We score these items either according to the predominant regularity, or also including close exceptional cases (alt OK in the table).
- **McCann & Besner ctrls** – these are pseudo-homophones and matched controls, that sound like actual words, but are spelled in a novel way, for example *choyce* (pronounced like *choice*), and the matched control is *phoyce*.
- **Taraban & McClelland** – has frequency matched regular and exception nonwords, for example *poes* (like high frequency words *goes* or *does*), and *mose*, like lower frequency *pose* or *lose*.

The results indicate that the model does a remarkably good job of capturing the performance of people's performance on these nonword reading sets. This suggests that the model is capable of learning the appropriate regularities and subregularities that are present in the statistics of English pronunciation.

Table 9.1 Comparison of nonword reading performance for our spelling-to-sound model (ss Model), the PMSP model, and data from people, across a range of different nonword datasets as described in the text. Our model performs comparably to people, after learning on nearly 3,000 English monosyllabic words.

Nonword Set	ss Model	PMSP	People
Glushko regulars	95.3	97.7	93.8
Glushko exceptions raw	79.0	72.1	78.3
Glushko exceptions alt OK	97.6	100.0	95.9
McCann & Besner ctrls	85.9	85.0	88.6
McCann & Besner homoph	92.3	n/a	94.3
Taraban & McClelland	97.9	n/a	100.0

Exploration

Run `ss` (spelling to sound) in CCN Sims to explore the spelling-to-sound model, and test its performance on both word and nonword stimuli.

Latent Semantics in Word Co-Occurrence

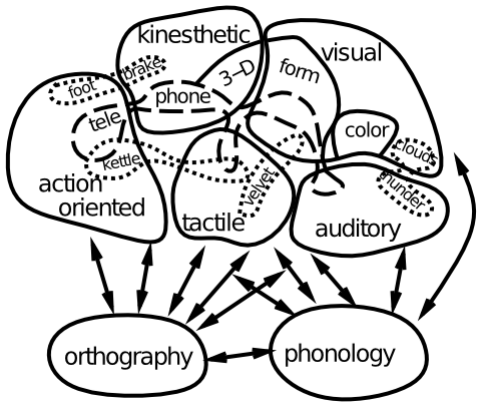


Figure 9.8: Distributed semantics for concrete words, where different aspects of a word's meaning are encoded in domain-specific brain areas (e.g., the sound of thunder in auditory areas, and the feel of velvet in somatosensory areas). Figure adapted from Allport, 1985.

Completing our more in-depth tour of the major pathways in the triangle model of reading, we now turn to the issue of semantics. What is the nature of the semantic representations shown at the top of Figure 9.5? An increasing body of data supports the idea shown in Figure 9.8, where the meaning of concrete words is encoded by patterns of activity within domain-specific brain areas that process sensory and motor information (Allport 1985). Thus, semantics is distributed throughout a wide swath of the brain, and it is fundamentally *embodied* and *grounded* in the sensory-motor primitives that we first acquire in life. Thus, the single "semantics" area shown in the triangle model is a major simplification relative to the actual widely distributed nature of semantic meaning in the brain.

However, there is also increasing evidence that the anterior tip or "pole" of the temporal lobe plays a particularly important role in representing semantic information, perhaps most importantly for more abstract words that lack a strong sensory or motor correlate. One theory is that this area acts as a central "hub" for coordinating the otherwise distributed semantic information (Patterson, Nestor, and Rogers 2007).

How do we learn the meanings of these more abstract words in the first place? Unlike the more concrete words shown in Figure 9.8, the meanings of more abstract words cannot be so easily pushed off to sensory and motor areas. One compelling idea here is that words obtain their meaning in part from the company they keep – the statistics of word co-occurrence across the large volume of verbal input that we are exposed

to can actually provide clues as to what different words mean. One successful approach to capturing this idea in a functioning model is called *Latent Semantic Analysis* (LSA) (Landauer and Dumais 1997) – see LSA Website for full details and access to this model.

LSA works by recording the statistics of how often words co-occur with each other within semantically-relevant chunks of text, typically paragraphs. However, these surface statistics themselves are not sufficient, because for example synonyms of words occur together relatively rarely, compared to how closely related they should be. And in general, there is a lot of variability in word choice and idiosyncrasies of word choices that are reflected in these detailed statistics. The key step that LSA takes in dealing with this problem is to apply a *dimensionality reduction* technique called *Singular Value Decomposition* (SVD), which is closely related to *Principal Components Analysis* (PCA), which in turn is closely related to the Hebbian self-organizing learning that our neural network models perform.

The key result of this SVD/PCA/Hebbian process is to extract the *strongest groupings* or clusters of words that co-occur together, in a way that integrates over many partially-overlapping subsets of word groups. Thus, even though synonyms do not tend to occur with each other, they do co-occur with many of the same other sets of words, and this whole group of words represents a strong statistical grouping that will be pulled out by the dimensionality reduction / Hebbian self-organizing learning process.

This process is exactly the same as what we saw with the V1 receptive field model in the *Perception* Chapter. In that model, Hebbian learning extracted the statistical regularity of oriented edges from a set of natural images. Any given image typically contains a noisy, partial version of an oriented edge, with perhaps several pixels occluded or blurry or otherwise distorted. However, as the self-organizing learning process integrates over many such inputs, these idiosyncrasies wash away, and the strongest statistical groupings of features emerge as oriented edges.

Unlike the V1 model, however, the individual statistical clusters that emerge from the LSA model (including our Hebbian version of it) do not have any clear interpretation equivalent to "oriented edges". As you'll see in the exploration, you can typically make some sense of small subsets of the words, but no obvious overall meaning elements are apparent. But this is not a problem – what really matters is that the overall distributed pattern of activity across the semantic layer appropriately captures the meanings of words. And indeed this turns out to be the case.

Exploration

Run the `sem` model from CCN Sims for the exploration of semantic learning of word co-occurrences. The model here was trained on an early draft of the first edition of this textbook, and thus has relatively specialized knowledge, hopefully much of which is now shared by you the reader.

Syntax and Semantics in a Sentence Gestalt

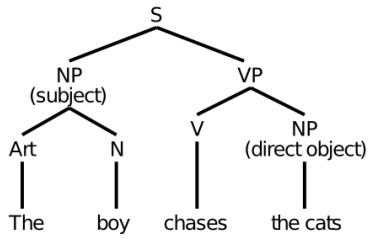


Figure 9.9: Syntactic diagram of a basic sentence. S = sentence; NP = noun phrase; Art = article; N = noun; VP = verb phrase; V = verb.

Having covered some of the interesting properties of language at the level of individual words, we now take one step higher, to the level of sentences. This step brings us face-to-face with the thorny issue of *syntax*. The traditional approach to syntax assumes that people assemble something akin to those tree-like syntactic structures you learned (or maybe not) in school (Figure 9.9). But given that these things need to be explicitly

taught, and don't seem to be the most natural way of thinking for many people, it seems perhaps unlikely that this is how our brains actually process language.

These syntactic structures also assume a capacity for role-filler binding that is actually rather challenging to achieve in neural networks. For example, the assumption is that you somehow "bind" the noun *boy* into a variable slot that is designated to contain the subject of the sentence. And once you move on to the next sentence, this binding is replaced with the next one. This constant binding and unbinding is rather like the rotation of a wheel on a car – it tends to rip apart anything that might otherwise try to attach to the wheel. One important reason people have legs instead of wheels is that we need to provide those legs with a blood supply, nerves, etc, all of which could not survive the rotation of a wheel. Similarly, our neurons thrive on developing longer-term stable connections via physical synapses, and are not good at this rapid binding and unbinding process. We focus on these issues in greater depth in the *Executive Function* Chapter.

An alternative way of thinking about sentence processing that is based more directly on neural network principles is captured in the **Sentence Gestalt** model of (St John and McClelland 1990). The key idea is that both syntax and semantics merge into an evolving distributed representation that captures the overall gestalt meaning of a sentence, without requiring all the precise syntactic bindings assumed in the traditional approach. We don't explicitly bind *boy* to subject, but rather encode the larger meaning of the overall sentence, which implies that the boy is the subject (or more precisely, the *agent*), because he is doing the chasing.

One advantage of this way of thinking is that it more naturally deals with all the ambiguity surrounding the process of parsing syntax, where the specific semantics of the collection of words can dramatically alter the syntactic interpretation. A classic demonstration of this ambiguity is the sentence:

Time flies like an arrow.

which may not seem very ambiguous, until you consider alternatives, such as:

Fruit flies like a banana.

The word *flies* can be either a verb or noun depending on the semantic context. Further reflection reveals several more ambiguous interpretations of the first sentence, which are fun to have take hold over your brain as you re-read the sentence. Another example from (Rohde 2002) is:

The slippers were found by the nosy dog.

The slippers were found by the sleeping dog.

just a single subtle word change recasts the entire meaning of the sentence, from one where the dog is the agent to one where it plays a more peripheral role.

If you don't bother with the syntactic parse in the first place, and just try to capture the meaning of the sentence, then none of this ambiguity really matters. The meaning of a sentence is generally much less ambiguous than the syntactic parse – getting the syntax exactly right requires making a lot of fine-grained distinctions that people may not actually bother with. But the meaning does depend on the exact combination of words, so there is a lot of emergent meaning in a sentence – here's another example from (Rohde 2002) where the two sentences are syntactically identical but have very different meaning:

We finally put the baby to sleep.

We finally put the dog to sleep.

The notion of a semantically-oriented gestalt representation of a sentence seems appealing, but until an implemented model actually shows that such a thing actually works, it is all just a nice story. The St. John & McClelland (1990) model does demonstrate that a distributed representation formed incrementally as words are processed in a sentence can then be used to answer various comprehension questions about that sentence. However, it does so using a very small space of language, and it is not clear how well it generalizes to new words, or scales to a more realistically complex language. A more sophisticated model by (Rohde 2002) that adopts a similar overall strategy does provide some promise for positive answers to these challenges. The training of the the Rohde model uses structured semantic representations in the form of slot-filler propositions about the thematic roles of various elements of the sentence. These include the roles: agent, experiencer, goal, instrument, patient, source, theme, beneficiary, companion, location, author, possession, subtype, property, if, because, while, and although. This thematic role binding approach is widely used in the natural language processing field for encoding semantics, but it moves away from the notion of an

unstructured gestalt representation of semantic meaning. The sentence gestalt model uses a much simpler form of this thematic role training, which seems less controversial in this respect.

The Sentence Gestalt Model

The sentence gestalt (SG) model is trained on a very small toy world, consisting of the following elements:

- People: *busdriver* (adult male), *teacher*, (adult female), *schoolgirl*, *pitcher* (boy). *adult, child, someone* also used.
- Actions: *eat, drink, stir, spread, kiss, give, hit, throw, drive, rise.*
- Objects: *spot* (the dog), *steak, soup, ice cream, crackers, jelly, iced tea, kool aid, spoon, knife, finger, rose, bat* (animal), *bat* (baseball), *ball* (sphere), *ball* (party), *bus, pitcher, fur.*
- Locations: *kitchen, living room, shed, park.*

The semantic roles used to probe the network during training are: *agent, action, patient, instrument, co-agent, co-patient, location, adverb, recipient.*

The main syntactic variable is the presence of active vs. passive construction, and clauses that further specify events. Also, as you can see, several of the words are ambiguous so that context must be used to disambiguate.

The model is trained on randomly-generated sentences according to a semantic and syntactic grammar that specifies which words tend to co-occur etc. It is then tested on a set of key test sentences to probe its behavior in various ways:

- Active semantic: *The schoolgirl stirred the kool-aid with a spoon.* (kool-aid can only be the patient, not the agent of this sentence)
- Active syntactic: *The busdriver gave the rose to the teacher.* (teacher could be either patient or agent – word order syntax determines it).
- Passive semantic: *The jelly was spread by the busdriver with the knife.* (jelly can't be agent, so must be patient)
- Passive syntactic: *The teacher was kissed by the busdriver.* vs. *The busdriver kissed the teacher.* (either teacher or busdriver could be agent, syntax alone determines which it is).
- Word ambiguity: *The busdriver threw the ball in the park.*, *The teacher threw the ball in the living room.* (ball is ambiguous, but semantically, busdriver throws balls in park, while teacher throws balls in living room)
- Concept instantiation: *The teacher kissed someone.* (male). (teacher always kisses a male – has model picked up on this?)
- Role elaboration: *The schoolgirl ate crackers.* (with finger); *The schoolgirl ate.* (soup) (these are predominant cases)
- Online update: *The child ate soup with daintiness.* vs. *The pitcher ate soup with daintiness.* (schoolgirl usually eats soup, so ambiguous *child* is resolved as schoolgirl in first case after seeing soup, but specific input of *pitcher* in second case prevents this updating).
- Conflict: *The adult drank iced-tea in the kitchen.* (living-room) (iced-tea is always had in the living room).

The model structure (Figure 9.10) has single word inputs (using localist single-unit representations of words) projecting up through an encoding hidden layer to the gestalt layer, which is where the distributed representation of sentence meaning develops. The memory for prior words and meaning interpretations of the sentence is encoded via a context layer, which effectively retains a copy of the gestalt layer activation state from the previous word input. Historically, this context layer is known as a **simple recurrent network (SRN)**, and it is widely used in neural network models of temporally extended tasks (Elman 1990; Jordan 1989; Cleeremans and McClelland 1991). In this model, we are using a biologically-based version of the SRN, based on the thalamocortical connections between the Pulvinar nucleus of the thalamus and the deep layers of the cortex (O'Reilly, Wyatte, and Rohrlich 2017), which is implemented in the *DeepLeabra* version of Leabra. The network training comes from repeated probing of the network for the various semantic roles enumerated above (e.g., *agent* vs. *patient*). A role input unit is activated, and then the network is trained to activate the

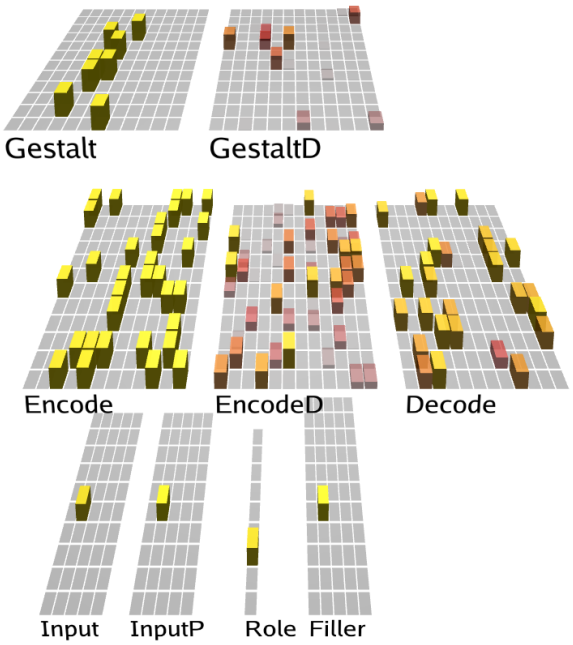


Figure 9.10: The sentence gestalt model, implemented in the DeepLeabra framework that does predictive learning with deep layer (D suffix) temporal context representations, that function like the context layer in a standard SRN (simple recurrent network). A single word at a time is presented in the input, which is encoded into the gestalt layer. The gestalt deep context layer (`GestaltD`) effectively maintains a copy of the gestalt from the previous time step, enabling integration of information across words in the sentence. This gestalt is probed by querying the semantic role, with training based on the ability to produce the correct filler output.

appropriate response in the filler output layer. In addition, as in other DeepLeabra models (and other SRN models), the encoder layer attempts to predict the next input, and learns from errors in these predictions.

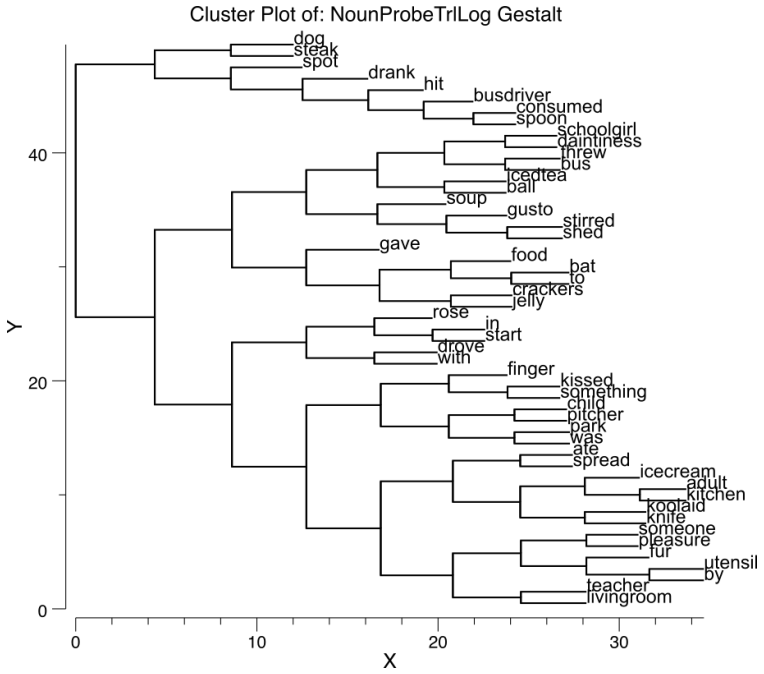


Figure 9.11: Cluster plot over the gestalt layer of patterns associated with the different nouns, showing that these distributed representations capture the semantic similarities of the words (much as in the LSA-like semantics model explored in the previous section).

Figure 9.11 shows a cluster plot of the gestalt layer representations of the different nouns, indicating that the network does develop sensible semantic similarity structure for these words. Probing further, Figure 9.12 shows the cluster plot for a range of related sentences, indicating a sensible verb-centric semantic organization – sentences sharing the same verb are all clustered together, and then agents within that form a second level of organization.

Exploration

Run the `sg` model from CCN Sims to explore the sentence gestalt model.

Next Steps in Language Modeling of Sentences and Beyond

The primary function of language is to communicate. It is fundamentally about semantics. And semantics represents a major barrier to further progress in language modeling. The sentence gestalt model has very simplistic semantics, and the more advanced version of it developed by (Rohde 2002) introduces more complex semantics, at the cost of injecting externally more of what the model should be developing on its own. Thus, the fundamental challenge for models of sentence-level or higher-level language is to develop a more naturalistic way of training the corresponding semantics. In an ideal case, a virtual humanoid robot would be wandering around a rich simulated naturalistic environment, and receiving and producing language in order to understand and survive in this environment. This would mimic the way in which people acquire and use language, and would undoubtedly provide considerable insight into the nature of language acquisition and higher-level semantic representations. But clearly this will require a lot of work.

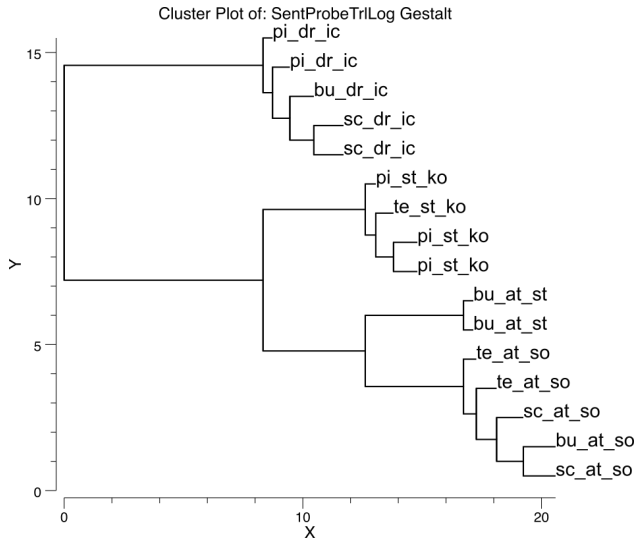


Figure 9.12: Cluster plot over the gestalt layer of patterns associated with a set of test sentences designed to test for appropriate similarity relationships. sc = schoolgirl; st = stirred; ko = kool-aid; te = teacher; bu= busddriver; pi = pitcher; dr = drank; ic = iced-tea; at = ate; so = soup; st = steak.

Chapter 10: Executive Function

We have now reached the top of the cognitive neuroscience hierarchy: the "executive" level. In a business, an executive makes important decisions and plans, based on high-level information coming in from all the different divisions of the company, and with a strong consideration of "the bottom line." In a person, the executive level of processing, thought to occur primarily within the **prefrontal cortex (PFC)**, similarly receives high-level information from posterior cortical association areas, and is also directly interconnected with motivational and emotional areas that convey "the bottom line" forces that ultimately guide behavior. Although many of us walk around with the impression (delusion?) that our actions are based on rational thought and planning, instead it is highly likely that basic biological motivations and affective signals play a critical role in shaping what we do. At least, this is what the underlying biology of the PFC and associated brain areas suggests. And yet, it is also clear that the PFC is critical for supporting more abstract reasoning and planning abilities, including the ability to ignore distraction and other influences in the pursuit of a given goal. We will try to unravel the mystery of this seemingly contradictory coexistence of abilities in the PFC in this chapter.

Evidence for the importance of the PFC in higher-level cognitive control comes from the *environmental dependency syndrome* associated with damage to PFC. In one classic example, a patient with PFC damage visited a researcher's home and, upon seeing the bed, proceeded to get undressed (including removal of his toupee!), got into bed, and prepared to sleep. The environmental cues overwhelmed any prior context about what one should do in the home of someone you don't know very well. In other words, without the PFC, behavior is much more reflexive and unthinking, driven by the affordances of the immediate sensory environment, instead of by some more abstract and considered plan or goals. You don't need actual PFC damage to experience this syndrome – certainly you have experienced yourself absent-mindedly doing something cued by the immediate sensory environment that you hadn't otherwise planned to do (e.g., brushing your teeth a second time before going to bed because you happened to see the toothbrush). We all experience lapses in attention – the classic stereotype of an absent-minded professor is not explained by lack of PFC in professors, but rather that the PFC is apparently working on something else and thus leaves the rest of the brain to fend for itself in an environmentally-dependent manner.

Another great source of insight into the cognitive contributions of the PFC is available to each of us every night, in the form of our dreams. It turns out that the PFC is one of the brain areas most inactivated during dreaming phases of sleep. As a result, our dreams often lack continuity, and seem to jump from one disconnected scene to another, with only the most tangential thread connecting them. For example, one

moment you might be reliving a tense social situation from high school, and the next you're trying to find out when the airplane is supposed to leave, with a feeling of general dread that you're hopelessly late for it.

So what makes the PFC uniquely capable of serving as the brain's executive? Part of the answer is its connectivity, as alluded to above – it sits on top of the overall information processing hierarchy of the brain, and thus receives highly-processed "status reports" about everything important going on in your brain. In this sense it is similar to the hippocampus as we saw in the *Memory* Chapter, and indeed these areas appear to work together. However, the PFC is also especially well placed to exert control over our actions – the PFC is just in front of the frontal motor areas (see the *Motor* Chapter), and has extensive connectivity to drive overt (and covert) motor behavior. Furthermore, the medial and ventral areas of PFC are directly interconnected with affective processing areas in subcortical regions such as the amygdala, thus enabling it to be driven by, and reciprocally, to amplify or override, motivational and affective signals.

In addition to being in the right place, the PFC also has some special biological properties that enable it to hold onto information in the face of distraction, e.g., from incoming sensory signals. Thus, with an intact PFC, you can resist the idea of laying down in someone else's bed, and remain focused on the purpose of your visit. We refer to this ability as **robust active maintenance** because it depends on the ability to keep a population of neurons actively firing over the duration needed to maintain a goal or other relevant pieces of information. This ability is also referred to as **working memory**, but this latter term has been used in many different ways in the literature, so we are careful to define it as synonymous with robust active maintenance of information in the PFC, in this context. We will see later how active maintenance works together with a gating system that allows us to hold in mind more than one item at a time, to selectively update and manipulate some information while continuing to maintain others, in a way that makes the integrated system support more sophisticated forms of working memory.

Recordings of neurons in the PFC of monkeys in the 1970's showed that they exhibit this robust active firing over delays (aka *delay period activity*). One of the most widely-used tasks is the oculomotor delayed response task, where a stimulus is flashed in a particular location of a video display, but the monkey is trained to maintain its eyes focused on a central fixation cross until that cross goes off, at which point it must then move its eyes to the previously flashed location in order to receive a juice reward. Neurons in the frontal eye fields (an area of PFC) show robust delay-period firing that is tuned to the location of the stimulus, and this activity terminates just after the monkey correctly moves its eyes after the delay. There are many other demonstrations of this robust active maintenance in the PFC of humans as well.

The computational models we explore in this chapter show how these two factors of connectivity and robust active maintenance can combine to support a wide range of executive function abilities that have been attributed to the PFC. The goal is to provide a unifying model of executive function, as compared to a laundry list of cognitive abilities that it is thought to support.

One of the most important executive function abilities is the ability to *rapidly shift* behavior or thought in a strategic manner (often referred to as *cognitive flexibility*). For example, when attempting to solve a puzzle or other challenging problem, you often need to try out many different ideas before discovering a good solution. Without the PFC, behavior is repetitive and stereotypical (banging your head against the wall again and again), lacking this hallmark flexibility. The ability to **rapidly update what is being actively maintained in the PFC** is what enables the PFC system to rapidly shift behavior or thought – instead of requiring relatively slow synaptic weight modification to change how the system behaves, updating the pattern of active neural firing in PFC can change behavior immediately. In short, the PFC system contributes to behavioral adaptation by dynamically updating activation states, which then shape posterior cortical representations or motor actions via *top-down* biasing of the associated patterns of activity. In contrast, behavioral adaption in the posterior cortex or basal ganglia relies much more on slowly adapting weight changes. Evidence for this difference comes from task switching paradigms, including the widely-studied *Wisconsin card sorting task (WCST)* in adults, and the *dimensional change card sorting task (DCCS)* in children.

The computational models in this chapter show how the **basal ganglia (BG)** and **midbrain dopamine areas** (specifically the **ventral tegmental area, VTA**) play a critical role in the rapid, strategic updating of PFC activity states. Specifically, we'll see that robust active maintenance requires an additional control signal to switch between maintaining existing information vs. updating to encode new information. The BG, likely in conjunction with dopaminergic signals from the VTA, play this role of **dynamic gating** of the

maintenance of information in PFC. This dynamic gating function is identical to the role the BG plays in gating motor actions, as we saw in the *Motor* Chapter. Furthermore, the BG learning process is also identical to that in the Motor chapter based on reinforcement learning principles. Specifically, dopamine (from the SNc, which is next door to the VTA) shapes BG learning and thereby enables the gating mechanism to deal with the challenging problem of deciding what is important to maintain (and as such is task-relevant and therefore predictive of intrinsic reward), vs. what can be ignored (because it is not predictive of good task performance). These mechanisms embody the general notion that the PFC-BG cognitive system evolved by leveraging existing powerful mechanisms for gating motor behavior and learning. From this perspective, cognition cannot be divorced from motivation, as dopaminergic learning signals play a central and intimate role in the basic machinery of PFC/BG function. The analogous functions of BG and dopamine in cognitive and motor action selection and learning have been strongly supported by various data over the last 10 or 20 years, including evidence from monkey studies, and in humans, effects of disease impacting BG and/or dopamine, pharmacological manipulations, functional imaging, and genetics.

Biology of PFC/BG and Dopamine Supporting Robust Active Maintenance

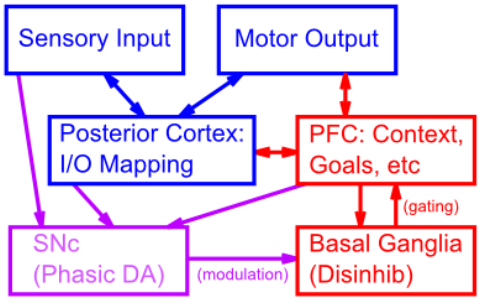


Figure 10.1: Schematic of functional relationships and connectivity of PFC, BG, and SNc phasic dopamine signals in relationship to basic loop between sensory input and motor output. The PFC provides top-down context and control over posterior cortical processing pathways to ensure that behavior is task and context appropriate. The BG exerts a disinhibitory gating influence over PFC, switching between robust maintenance and rapid updating. The SNc (substantia nigra pars compacta) exhibits phasic dopamine (DA) firing to drive learning and modulation of BG circuits, thereby training the BG gating signals in response to task demands (external reward signals).

The overall connectivity of the areas that are particularly important for executive function are shown in Figure 10.1, in relation to the sensory and motor processing associated with posterior cortex (temporal, parietal and occipital lobes) and motor frontal cortex (i.e., frontal cortex posterior to the prefrontal cortex). The PFC is interconnected with higher-level association cortical areas in posterior cortex where highly processed and abstracted information about the sensory world is encoded. It also interconnects with higher-level motor control areas (premotor cortex, supplementary motor areas), which coordinate lower-level motor control signals to execute sequences of coordinated motor outputs. With this pattern of connectivity, PFC is in a position to both receive from, and exert influence over, the processing going on in posterior and motor cortex.

The Basal Ganglia (BG), which consists principally of the striatum (caudate, putamen, nucleus accumbens), globus pallidus, and subthalamic nucleus, is densely interconnected with the PFC by way of specific nuclei of the thalamus. As described in detail in the *Motor Control and Reinforcement Learning* Chapter, the BG provides a dynamic, adaptive gating influence on the frontal cortex, by disinhibiting the excitatory loop between PFC and the thalamus. In the context of the PFC, this gating influence controls the updating of information that is actively maintained in the PFC, using the same mechanisms that control the initiation of motor actions in the context of motor control. Also, top-down projections from PFC to the subthalamic nucleus support a type of inhibitory control over behavior by detecting conditions under which ongoing action selection should be halted or switched and preventing the rest of the BG circuitry from gating the planned motor action.

The final major component of the executive control system consists of the substantia nigra pars compacta (SNc) and several other associated brain areas that together drive phasic dopamine neuromodulation of the

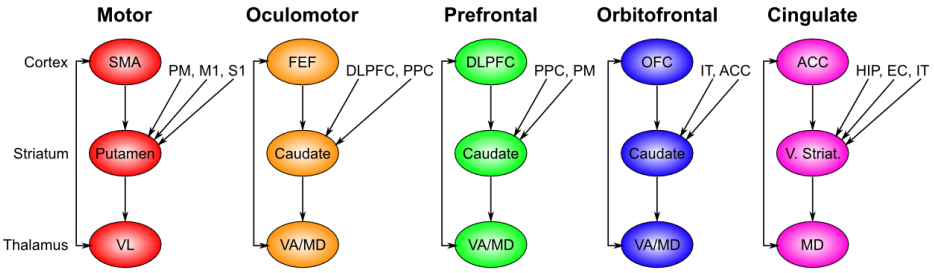


Figure 10.2: Parallel circuits through the basal ganglia for different regions of the frontal cortex – each region of frontal cortex has a corresponding basal ganglia circuit, for controlling action selection/initiation in that frontal area. Motor loop: SMA = supplementary motor area – the associated striatum (putamen) also receives from premotor cortex (PM), and primary motor (M1) and somatosensory (S1) areas – everything needed to properly contextualize motor actions. Oculomotor loop: FEF = frontal eye fields, also receives from dorsolateral PFC (DLPFC), and posterior parietal cortex (PPC) – appropriate context for programming eye movements. Prefrontal loop: DLPFC also controlled by posterior parietal cortex, and premotor cortex. Orbitofrontal loop: OFC = orbitofrontal cortex, also receives from inferotemporal cortex (IT), and anterior cingulate cortex (ACC). Cingulate loop: ACC also modulated by hippocampus (HIP), entorhinal cortex (EC), and IT.

BG, resulting in reinforcement learning of its gating actions. This system, summarized computationally using the PVLV model as described in the *Motor Control and Reinforcement Learning* Chapter, interacts with the active maintenance of information in PFC to be able to reinforce a gating signal in the BG that leads to subsequent good performance and reward later in time. This time-travel property of the phasic DA reinforcement learning is essential for training a system that maintains information over time.

In the following subsections, we summarize the biological properties of each of these systems and their relevance to executive function. The prefrontal cortex basal ganglia working memory model (PBWM) then integrates all of these elements into a functioning computational model that can perform complex executive function tasks, as we explore in the remainder of the chapter.

Robust Active Maintenance in the PFC

The ability of PFC neurons to exhibit sustained active firing over delays, as initially discovered by (Fuster and Alexander 1971; Kubota and Niki 1971), is shown in Figure 10.3, panel B ("Neuron with delay signal"), in the context of the delayed saccading task described in the introduction. Other subsets of PFC neurons also exhibit other firing patterns, such as responding transiently to visual inputs (Panel C) and initiating movements (Panel D). This differentiation of neural response patterns in PFC has important functional implications that we capture in the PBWM model described later.

There are two primary biological mechanisms that enable PFC neurons to exhibit sustained active firing over time:

- **Recurrent excitatory connectivity:** Populations of PFC neurons have strong excitatory interconnections (Figure 10.4), such that neural firing reverberates back-and-forth among these interconnected neurons, resulting in sustained active firing. There are two types of such connections: 1) a *corticocortical loop* among pyramidal cells in the same PFC stripe, and; 2) a *corticothalamocortical loop* between lamina VI pyramidal cells in PFC and the thalamic relay cells that project to that particular group of cells.
- **Intrinsic excitatory maintenance currents:** At the synapses formed by both of the recurrent excitatory loops there are NMDA and metabotropic glutamate (mGluR) receptors that, once opened by high frequency activity, provide a longer time window of increased excitability so as to keep reverberant activity going. Recall from the *Learning* Chapter that the NMDA channel requires the neuron to be sufficiently depolarized to remove the Mg+ (magnesium) ions that otherwise block the channel. This activity-dependent nature of the NMDA channel makes it ideally suited to providing a "switched" or dynamically gated form of active maintenance – only those neurons that have already been sufficiently activated will benefit from the increased excitation provided by these NMDA channels. This provides a "hook" for the basal ganglia system to control active maintenance: when the thalamic neurons are

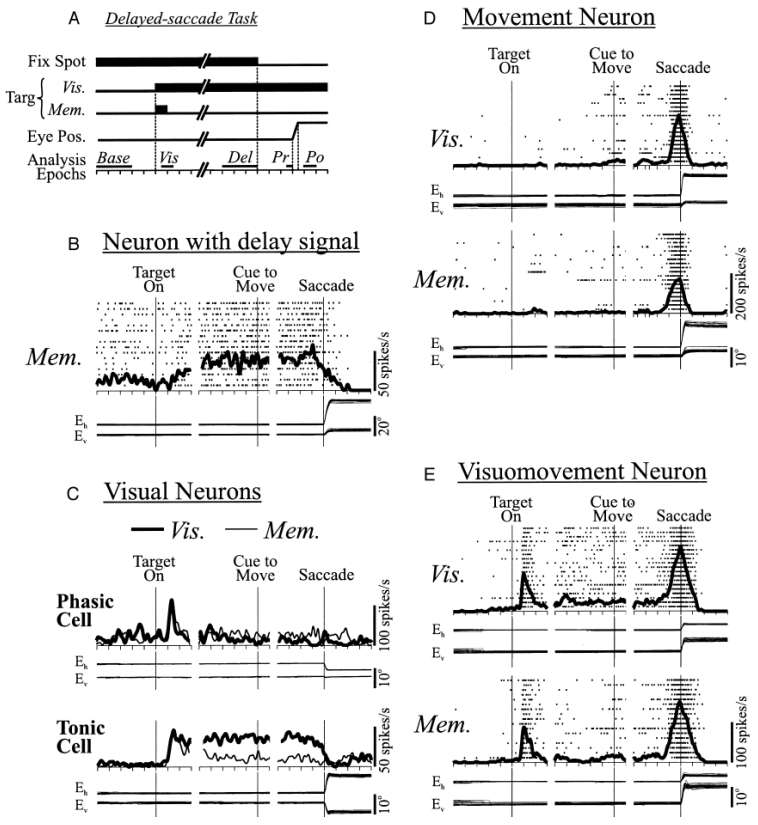


Figure 10.3: An example of sustained delay period activity (Panel B). Histogram (background dots) and curve of activity rate for an individual cell recorded in the frontal eye fields (FEF) during a delayed saccade task. The target stimulus is only on briefly at the beginning of the trial (Panel A, Targ, Mem.). This cell maintained its activity during the delay so as to enable other cells to generate a correct saccade at the end of the trial (Panels D,E). Adapted from Sommer & Wurtz, 2000, Figure 2.

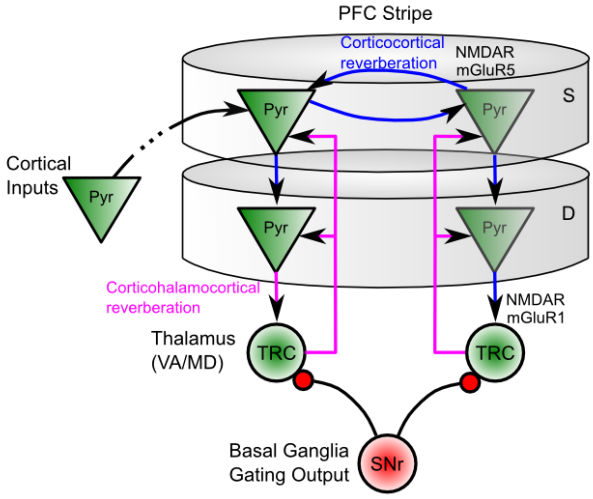


Figure 10.4: Two types of reverberant loops can support actively maintained representations in cortical tissue: 1) Corticocortical interconnections among pyramidal neurons within the same PFC stripe (horizontal blue arrows); 2) Thalamocortical connections between PFC and thalamic relay cells (TRC's) (vertical blue arrows). Both use mutually supportive recurrent excitation plus intrinsic maintenance currents (NMDARs; mGluRs).

disinhibited via a BG gating action, the ensuing burst of activity enables a subset of PFC neurons to get over their NMDA Mg+ block thresholds, and thereby continue to fire robustly over time.

Functional Specialization Across PFC Areas

The mechanisms for robust active maintenance exist across the PFC, but different PFC areas have been associated with different contributions to overall executive function. We will explore the idea later in this chapter that these different functional contributions can be explained in terms of differences in connectivity of these PFC areas with other parts of the brain, within the context of the unifying framework that all PFC areas share robust active maintenance as a critical feature.

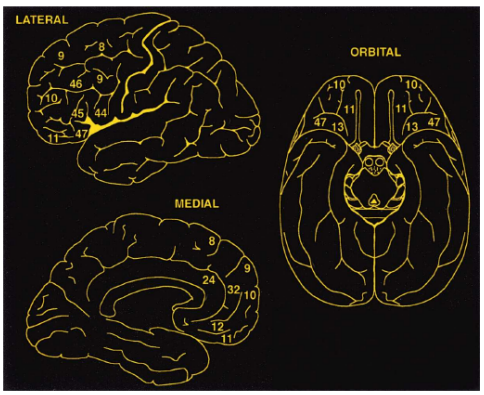


Figure 10.5: Brodmann numbers for areas of the prefrontal cortex, each of which has been associated with a different mixture of executive functions. Reproduced from Fuster, 2001.

Anatomically, the frontal lobes constitute those cortical areas anterior to the central sulcus. Immediately anterior to the central sulcus, and thus most posteriorly in frontal cortex, is the primary motor cortex (M1), which is most prominently seen on the lateral surface but extends all the way over the dorsal surface and onto the medial side. Contiguous tissue roughly anterior to M1 makes up planning motor areas, the premotor (PM) cortex (laterally) and supplementary motor areas (SMA, pre-SMA; medially). Then, anterior to that are the PFC areas, labeled with their Brodmann numbers in Figure 10.5.

At the broadest level, the PFC areas can be divided along the major axes of medial vs. lateral and dorsal vs. ventral. Generally speaking, the lateral PFC areas are interconnected with sensory and motor areas in more posterior cortex, and are thought to play a role in controlling the processing in these areas. In contrast, the medial PFC areas are more strongly interconnected with subcortical brain areas associated with affective and motivational functions. Functionally we can characterize the lateral areas as being important for "cold" cognitive control, while the medial areas are important for "hot" emotional and motivational processing (Figure 10.6). However, this distinction is not as clear cut as it sounds, as even the lateral areas are subject to modulation by motivational variables and BG/dopamine gating signals based on the extent to which maintained cognitive information is predictive of task success (a form of reward).

The functional significance of the dorsal vs. ventral distinction has been considerably more controversial in the literature, but anatomically it is clear that dorsal PFC areas interconnect more with the dorsal pathway in the posterior cortex, while ventral PFC interconnects with the ventral posterior cortex pathway. As we saw in the *Perception* Chapter, the dorsal pathway in posterior cortex is specialized for perception-for-action (*How* processing): extracting perceptual signals to drive motor control, while the ventral pathway is specialized for perception-for-identification (*What* processing). This functional specialization in posterior cortex can be carried forward to the associated dorsal and ventral areas of PFC (Figure 10.6), such that **dorsal lateral PFC (DLPFC)** areas are particularly important for executive control over motor planning and the parietal cortex pathways that drive motor control, while **ventral lateral PFC (VLPFC)** areas are particularly important for control over the temporal lobe pathways that identify entities in the world, and also form rich semantic associations about these entities.

On the medial side, the dorsal medial PFC is also known as the anterior cingulate cortex (ACC), which

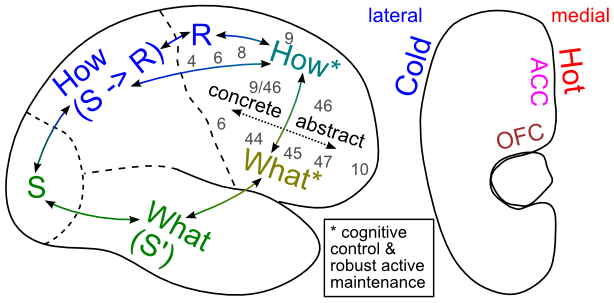


Figure 10.6: The *What* vs. *How* distinction for posterior cortex can be carried forward into prefrontal cortex, to understand the distinctive roles of the ventral and dorsal areas of PFC. Reproduced from O'Reilly (2010).

has been shown to encode the affective aspects of motor control variables (e.g., how much effort will an action take, what is its probability of success, how much conflict and uncertainty is there in selecting a response), which is consistent with a "hot how" functional specialization. Dorsomedial PFC areas also project to the subthalamic nucleus within the BG, and serve to delay motor responding to prevent impulsive choice under difficult response selection demands (Michael J. Frank 2006; Aron et al. 2007; Cavanagh et al. 2011). The ventromedial areas of PFC (VMPFC) including the orbital frontal cortex (OFC) have been shown to encode the affective value of different sensory stimuli, consistent with the idea that they are the "hot what" areas. See (O'Reilly 2010) for more discussion of the What/How functional specialization idea.

Substructure within PFC Areas: Stripes

Within each functional PFC area, there is some interesting topographic organization of neurons into *hypercolumns, macrocolumns* or **stripes** (each of these terms is typically associated with a similar type of neural organization, but in different parts of the cortex, with stripes being specific to the PFC; (Levitt et al. 1993)). In all areas of cortex, one can identify the smallest level of neural topological organization as a *cortical column* or *microcolumn* (to more clearly distinguish it from the larger *macrocolumn*), which contains roughly 20 pyramidal neurons in a region that is roughly 50 microns across. A stripe contains roughly 100 of these microcolumns, generally organized in an elongated shape that is roughly 5 microcolumns wide (250 microns) by 20 microcolumns long (1000 microns or 1 millimeter). Each such stripe is interconnected with a set of roughly 10 or more other stripes, which we can denote as a *stripe cluster*. Given the size of the human frontal cortex, there may be as many as 20,000 stripes within all of frontal cortex (including motor areas).

In PFC and other areas, neurons within a microcolumn tend to encode very similar information, and may be considered equivalent to a single rate-coded neuron of the sort that we typically use in our models. We can then consider an individual stripe as containing roughly 100 such rate-coded neuron-equivalents, which provides sufficient room to encode a reasonably large number of different things using sparse distributed representations across microcolumns.

Functionally, we hypothesize in the PBWM model that each stripe can be independently updated by a corresponding stripe-wise loop of connectivity with an associated stripe of neurons through the BG system. This allows for very fine-grained control by the BG over the updating and maintenance of information in PFC, as we describe next.

Basal Ganglia and Dynamic Gating

As we discussed in the *Motor and Reinforcement Learning* Chapter, the Basal Ganglia (BG) is in a position to modulate the activity of the PFC, by virtue of its control over the inhibition of the thalamic neurons that are bidirectionally connected with the PFC (Figure 10.7). In the default state of no striatal activity, or firing of indirect (NoGo) pathway neurons, the SNr (substantia nigra pars reticulata) or GPi (globus pallidus internal segment) neurons tonically inhibit the thalamus. This prevents the thalamocortical loop from being activated, and it is activation of this loop that is thought to be critical for initiating motor actions or updating PFC active memory representations. When the striatal Go (direct) pathway neurons fire, they

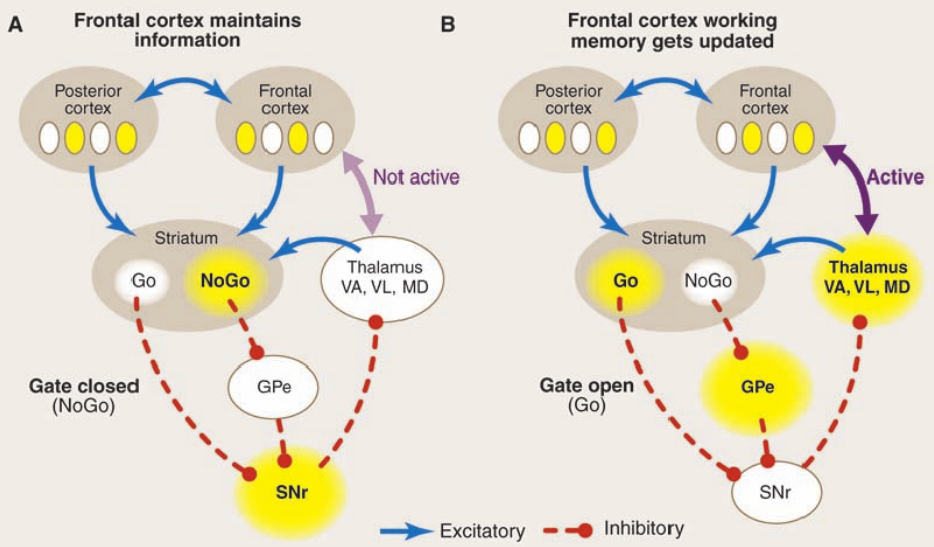


Figure 10.7: How the basal ganglia (BG) can modulate active maintenance in the PFC. (A) In the default state of no BG activity, or NoGo (indirect) pathway firing in the striatum, the PFC continues to maintain information in an active state. The SNr (substantia nigra pars reticulata) or GPi (globus pallidus internal segment) exhibits tonic (sustained) activity, that inhibits neurons in the thalamus, thereby shutting down the thalamocortical loop. (B) Go (direct) pathway firing triggers updating of a PFC stripe to encode new information (e.g., new stimulus inputs that are behaviorally relevant). This occurs by inhibiting the SNr/GPi neurons, thereby opening up the thalamocortical loop, resulting in a burst of activity in the PFC that drives updating to a new pattern of neural firing, including new intrinsic maintenance currents that will continue to sustain this new pattern going forward. This mechanism is identical to that for gating motor actions via BG interactions with pre/motor areas in parallel circuits (see Motor Chapter). Reproduced from O'Reilly, 2006, Science.

inhibit the tonic SNr/GPi inhibition, thereby allowing the excitatory thalamocortical loop to be activated. This wave of excitatory activation can activate a new population of PFC neurons, which are then actively maintained until a new Go signal is fired.

Phasic DA and Temporal Credit Assignment

Another critical biological mechanism for executive function, which we also discussed in the *Motor and Reinforcement Learning* Chapter, is the firing of phasic dopamine neurons in the midbrain (**ventral tegmental area (VTA)** and **substantia nigra pars compacta (SNc)**). These neurons initially respond to primary rewards (e.g., apple juice), but then learn to fire at the onset of conditioned stimuli (CS's) that reliably predict these primary rewards. This amounts to a form of *time travel* that solves a critical problem for the PFC active maintenance system: how does the system learn what to maintain, given that the decision for what to maintain typically occurs well in advance of the subsequent value of having maintained something useful. If you think of the maintenance of useful information in the PFC as a kind of CS (because they should reliably be associated with positive outcomes), then the dopamine neurons will learn to fire at the onset of such a CS. Having this phasic DA signal at CS onset can then reinforce the decision to maintain this information in the first place, thus solving the time travel problem.

The computational model described next incorporates this key idea, by having the phasic DA signal at CS onset drive learning of the BG Go neurons that update new information into PFC active maintenance. The model shows that this core idea is sufficient to support the learning of complex executive function tasks.

The PBWM Computational Model

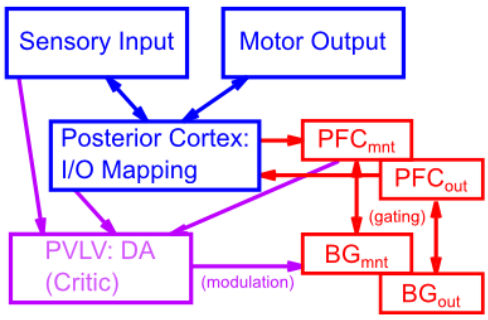


Figure 10.8: Components of a PBWM model, based on biological connections and functions of the PFC (robust active maintenance of task-relevant information), Basal Ganglia (BG, dynamic gating of PFC active maintenance), and PVLV (phasic dopamine signals for training the BG gating. Each specialized job, in interaction, produces a capable overall executive function system, after sufficient learning experience.

The biological properties of the PFC/BG system that we reviewed above are captured in a computational model called PBWM (prefrontal cortex basal ganglia working memory) (O'Reilly and Frank 2006; Hazy, Frank, and O'Reilly 2006, 2007) (Figure 10.8). The PFC neurons in this model are organized into separately-updatable stripes, and also into separate functional groups of maintenance and output gating (described more below). Furthermore, each PFC stripe is represented in terms of superficial layers (2,3) and deep layers (5,6) – the deep layer neurons specifically have the ability to sustain firing over time through a variety of mechanisms, representing the effects of NMDA and mGluR channels and excitatory loops through the thalamus. The flow of activation from the superficial to deep layers of a given PFC stripe is dependent on BG gating signals, with the BG layers also organized into corresponding maintenance and output gating stripes. The **Matrix** layer of the BG (representing the matrisomes of the striatum) has separate Go and NoGo neurons that project to a combined GPi and thalamus (**GPiThal**) layer with a single neuron per stripe that fires if the Go pathway is sufficiently stronger than the NoGo (this mechanism abstracts away from the detailed BG gating circuitry involving the GPe, GPi/SNr, STN and thalamus, as simulated in the motor chapter, and simply summarizes functionality in a single GPiThal layer). A GPiThal Go signal will update the PFC deep layer activations to

reflect the current superficial layer activations, while a NoGo leaves the PFC alone to continue to maintain prior information (or nothing at all).

The PVLV phasic dopamine system drives learning of the BG Go and NoGo neurons, with positive DA bursts leading to facilitation of Go and depression of NoGo weights, and vice-versa for DA dips – using the same reinforcement learning mechanisms described in the Motor chapter.

Perhaps the single most important key for understanding how the PBWM system works is that it uses *trial and error* exploration of different gating strategies in the BG, with DA reinforcing those strategies that are associated with positive reward, and punishing those that are not. In the current version of the model, Matrix learning is driven exclusively by dopamine firing at the time of rewards, and it uses a synaptic-tag-based trace mechanism to reinforce/punish all prior gating actions that led up to this dopaminergic outcome. Specifically, when a given Matrix unit fires for a gated action, synapses with active input establish a *synaptic tag,* which persists until a subsequent phasic dopaminergic outcome signal. Extensive research has shown that these synaptic tags, based on actin fiber networks in the synapse, can persist for up to 90 minutes, and when a subsequent strong learning event occurs, the tagged synapses are also strongly potentiated (Redondo and Morris 2011; Rudy 2015; Bosch and Hayashi 2012). This form of trace-based learning is very effective computationally, because it does not require any other mechanisms to enable learning about the reward implications of earlier gating events. In earlier versions of the PBWM model, we relied on CS (conditioned stimulus) based phasic dopamine to reinforce gating, but this scheme requires that the PFC maintained activations function as a kind of internal CS signal, and that the amygdala learn to decode these PFC activation states to determine if a useful item had been gated into memory. Compared to the trace-based mechanism, this CS-dopamine approach is much more complex and error-prone. Instead, in general, we assume that the CS's that drive Matrix learning are more of the standard external type, which signal progress toward a desired outcome, and thus reinforce actions that led up to that intermediate state (i.e., the CS represents the achievement of a *subgoal*).

The presence of multiple stripes is typically important for the PBWM model to learn rapidly, because it allows different gating strategies to be explored in *parallel*, instead of having a single stripe sequentially explore all the different such strategies. As long as one stripe can hit upon a useful gating strategy, the system can succeed, and it quickly learns to focus on that useful stripe while ignoring the others. Multiple stripes are also critical when more than one piece of information has to be maintained and updated in the course of a task – indeed, it is this demand that motivated the development of the original PBWM model to supersede earlier gating models, which used phasic dopamine signals to directly gate PFC representations but did not support multiple gating and hence was limited to a capacity of a single item. One interesting consequence of having these multiple stripes is that "superstitious" gating can occur in other stripes – if that gating happens to reliably enough coincide with the gating signals that are actually useful, it too will get reinforced. Perhaps this may shed light on our proclivity for being superstitious?

Output Gating

As we saw in Figure 10.3, some PFC neurons exhibit delay-period (active maintenance) firing, while others exhibit output response firing. These populations do not appear to mix: a given neuron does not typically exhibit a combination of both types of firing. This is captured in the PBWM framework by having a separate set of PFC stripes that are *output gated* instead of *maintenance gated*, which means that maintained information can be subject to further gating to determine whether or not it should influence downstream processing (e.g., attention or motor response selection). We typically use a simple pairing of maintenance and output gating stripes, with direct one-to-one projections from maintenance to output PFC units, but there can be any form of relationship between these stripes. The output PFC units are only activated, however, when their corresponding stripe-level BG/GPiThal Go pathway fires. Thus, information can be maintained in an active but somewhat "offline" form, before being actively output to drive behavior. Figure 10.9 illustrates this division of labor between the maintenance side and the output side for gating and how a "handoff" can occur.

For more PBWM details, including further considerations for output gating, how maintained information is cleared when no longer needed (after output gating), and gating biases that can help improve learning, see the *PBWM Details* Appendix, which also includes relevant equations and default parameters.

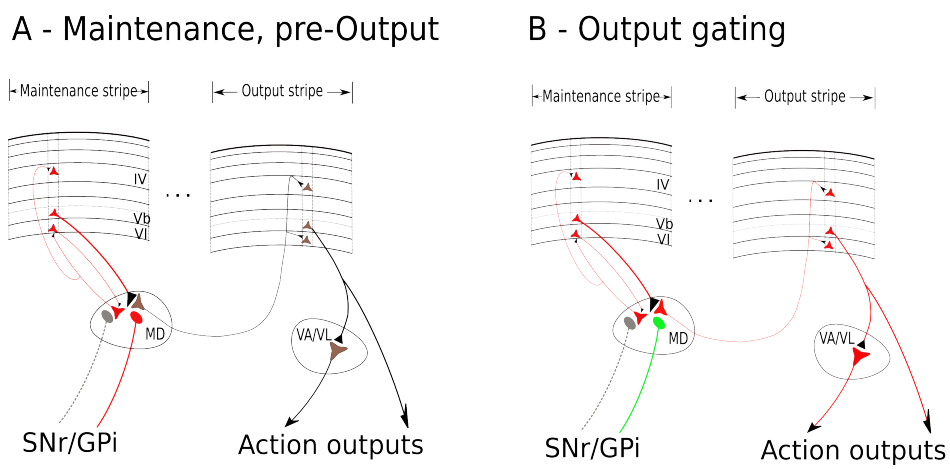


Figure 10.9: Schematic to illustrate the division of labor between maintenance-specialized stripes and corresponding output-specialized stripes. A - Maintenance stripe (left) in maintenance mode, with corticothalamocortical reverberant activity shown (red). Information from that stripe projects via layer Vb pyramidals to a thalamic relay cell for the corresponding output stripe, but the BG gate is closed from tonic SNr/GPi inhibition so nothing happens (gray). B - Output gate opens due to 'Go'-signal generated disinhibition of SNr/GPi output (green), triggering burst firing in the thalamic relay cell, which in turn activates the corresponding cortical stripe representation for the appropriate output. Projection from output stripe's layer Vb pyramidal cells then activates cortical and subcortical action/output areas, completing a handoff from maintenance to output. MD = mediodorsal nucleus of the thalamus; VP/VL = ventroposterior or ventrolateral (motor) thalamic nuclei.

Top-down Cognitive Control from Sustained PFC Firing: The Stroop Model

We now turn to a series of computer simulations to explore various facets of executive function. We begin with perhaps the single most studied task used to test for executive function, the *Stroop task*, named after John Ridley Stroop, who first described the basic phenomenon (Stroop 1935). The computational model of this task, developed initially by (Cohen, Dunbar, and McClelland 1990), has been applied (with appropriate change of labels) to a remarkably wide range of different phenomena. Thus, this deceptively simple task and model capture the most critical features of executive function.

Figure 10.10: The Stroop task requires either reading the word or naming the ink color of stimuli such as these. When there is a conflict between the word and the ink color, the word wins because reading is much more well-practiced. Top-down biasing (executive control) is required to overcome the dominance of word reading, by providing excitatory support for the weaker ink color naming pathway.

In the Stroop paradigm (Figure 10.10) subjects are presented with color words (e.g., "red", "green") one at a time and are required to either read the word (e.g., "red"), or name the color of the ink that the word is written in. Sometimes the word "red" appears in *green* ink, which represents the *incongruent* or *conflict* condition. The "Stroop effect" is that error rates and response times are larger for this incongruent condition, especially in the case of color naming (Figure 10.11). That color naming is particularly difficult in the incongruent condition has been attributed to the relatively "automatic", well-practiced nature of reading words, so that the natural tendency to read the word interferes with attending to, and naming, the color of the ink.

The Cohen et al. (1990) Stroop model showed how a maintained PFC representation can provide a

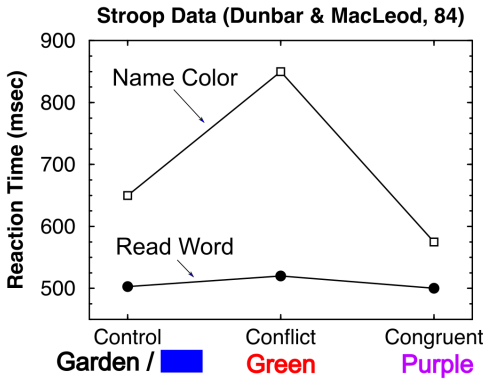


Figure 10.11: Typical data from neurologically intact participants on the Stroop task, showing differentially slowed performance on the conflict (incongruent) color naming condition. Damage to the PFC produces a differential impairment in this condition relative to the others, indicating that PFC is providing top-down excitatory biasing to support color naming.

strong top-down bias to support the weaker color processing channel in the face of the stronger word-reading pathway. They were able to establish the difference between word reading and color naming simply as a function of the amount of training provided on each of these tasks. Our simulation reproduces these same core features.

The Stroop model helps clarify the role of **inhibition** in executive function. Many people describe the Stroop task as requiring people to inhibit the prepotent word reading pathway, in order to focus on the ink color, and the model also does involve inhibitory dynamics. However, the PFC in the model does *not* provide a directed form of inhibition to the word reading pathway specifically. Instead, it provides *excitatory* top-down support to the weaker pathway (color naming), which then enables this pathway to better compete (via lateral inhibitory interactions) with the more dominant word reading pathway. Thus, inhibition is seen as a more collateral, automatic process operating throughout the cortex, and top-down biasing is involved in exciting relevant information, rather than inhibiting irrelevant information.

Exploration

Open the `stroop` model in CCN Sims.

Development of PFC Active Memory Strength and the A-not-B Task

The developmental process can provide important insights into various cognitive phenomena, often by making cognitive failures particularly stark. A great example of this is the A-not-B task developed by pioneering developmental researcher Jean Piaget (Piaget 1954). An infant is repeated shown a toy hidden in one location (labeled A), and when the toy is then hidden in a different location (B), they continue to reach back to A. The behavior is striking – the infant just saw the toy being hidden, tracking the experimenter's movements with great attention (typically novel, interesting toys are used). And yet they appear to forget all about this in a flash, reverting back to the previously established "habitual" behavior.

The computational model we explore here (Munakata 1998) shows how a range of behavioral phenomena, some of it quite subtle and complex, can be captured with a relatively simple model that shares much in common with the Stroop model explored above. Development in this model is operationalized simply as the strength of the reverberant excitatory connections among PFC neurons, which are the only mechanism for active maintenance in this simplified model. The "older" networks can hold onto information for a longer period of time due to their stronger recurrent connections, while information is much more fleeting in the "younger" ones with weaker recurrent connections.

Exploration

To see how this all plays out, open the `a-not-b` model in CCN Sims.

Dynamic Updating of PFC Active Memory: The SIR Model

Having seen in the Stroop and A-not-B models how sustained PFC activity can influence behavior through top-down biasing, we now turn to the more complex aspects of PFC function, involving the dynamic gating of PFC representations by the basal ganglia, and the ability to rapidly update and robustly maintain information. As a first introduction to this functionality, captured by the PBWM model, we use the simple **SIR (Store, Ignore, Recall)** task. Here is a sample sequence of trials in this task:

- **S - A** – this means that the network should store the *A* stimulus for later recall – network responds *A*.
- **I - C** – ignore the *C* stimulus, but you still have to respond to it – network responds *C*.
- **I - B** – ignore the *B* stimulus – network responds *B*.
- **R** – recall the most recently stored stimulus – network responds *A*.

The BG maintenance gating system has to learn to fire Go to drive updating of PFC on the Store trials to encode the associated stimulus for later recall. It also must learn to fire NoGo to the ignore stimuli, so they don't overwrite previously stored information. Finally, on recall trials, the output BG gating mechanism should drive output of the stored information from PFC. It is critical to appreciate that the network starts out knowing nothing about the semantics of these various inputs, and has to learn entirely through trial-and-error what to do with the different inputs.

Exploration

To see this learning unfold, open the `sir` model in CCN Sims.

While we don't consider it here for simplicity, the same PBWM model, when augmented to have multiple parallel stripes, can learn to separately update and maintain multiple pieces of information in working memory and to retrieve the correct information when needed. A good example of this demand is summarized by the SIR-2 task, where instead of involving a single store and recall task control signal, there are two such signals (i.e. S1 and S2 and R1 and R2). Thus, the network has to learn to separately store two stimuli, update them into separate buffers, and appropriately respond based on the maintained information in the correct buffer when cue to recall R1 vs. R2.

More Complex Dynamic Updating of PFC Active Memory: The N-Back Task

The N-back task has become one of the most widely used measures of complex working memory function in the PFC, in part because it so reliably drives the activation of the PFC in functional MRI (fMRI) experiments. Chatham et al (in press) developed a PBWM-based model of this task, which shows how a more complex cognitive task can be learned by PBWM. This model makes contact with a range of important findings in the cognitive neuroscience literature as well.

Hierarchical Organization of PFC: Subtasks, Goals, Cognitive Sequencing

For related models simulating hierarchical control over action across multiple PFC-BG circuits, see (Reynolds and O'Reilly 2009; Frank and Badre 2012; Collins and Frank 2013). The latter model considers situations in which there are multiple potential rule sets signifying which actions to select in particular sensory states, and where the appropriate rule set might depend on a higher level context. (For example, your tendency to greet someone with a hug, kiss, handshake, or wave might depend on the situation: your relationship to the person, whether you are in the street or at work, etc. And when you go to a new country (or city), the rule set to apply may be the same as that you've applied in other countries, or it might require creating a new rule set). More generally, we refer to the higher level rule as a "task-set" which contextualizes how to act in response to many different stimuli. Hierarchical PFC-BG networks can learn to create these PFC task-sets, and simultaneously, which actions to select in each task-set (Figure 10.12).

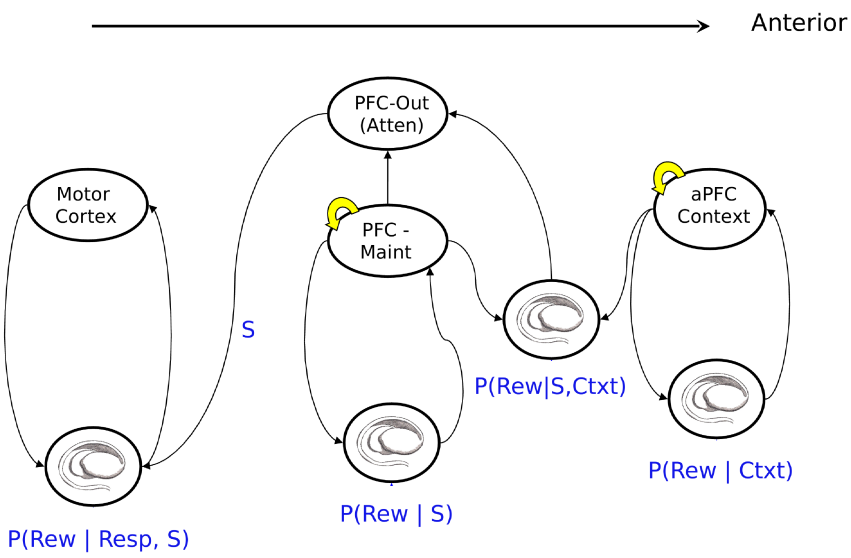


Figure 10.12: Hierarchical action selection across multiple prefrontal basal ganglia loops. On the far right, at the most anterior level, the PFC represents contextual information that is gated by its corresponding BG loop based on the probability that maintaining this context for guiding lower level actions is predictive of reward. The middle loop involves both input and output gating. The input gating mechanism allows stimulus representations S to update a PFC_maint layer, while the output gating mechanism gates out a subset of maintained information conditional on the context in anterior PFC. Its associated BG layer learns the reward probability of output gating given the maintained stimulus S and the context. Finally, the left-most motor loop learns to gate simple motor responses based on their reward probabilities conditional on the stimulus, as in the single loop BG model described in the Motor chapter, but where here relevant stimulus features are selected by the more anterior loops. Reproduced from Frank & Badre (2012).

Critically, with this hierarchical representation, the learned PFC representations are abstract and independent of the contexts that cue them, facilitating generalization and transfer to other contexts, while also identifying when new task-sets need to created. They also allow for new knowledge to be appended to existing abstract task structures, which then can be immediately transferred to other contexts that cue them (much like learning a new word in a language: you can immediately then re-use that word in other contexts and with other people). To see this network in action, including demonstrations of generalization and transfer, see the Collins & Frank network linked here. Various empirical data testing this model have shown that indeed humans (including babies!) represent such task-sets in a hierarchical manner (even when not cued to do so, and even when it is not beneficial for learning) in such a way that facilitates generalization and transfer; and that the extent of this hierarchical structure is related to neural signatures in PFC and BG (Badre and Frank 2012; Collins and Frank 2016).

To put many of the elements explored above to their most important use, we explore how the coordinated interactions of various regions of the PFC (including the affective areas explored previously), together with BG gating, enable the system to behave in a coherent, task-driven manner over multiple sequential steps of cognitive processing. This is really the hallmark of human intelligence: we can solve complex problems by performing a sequence of simpler cognitive steps, in a flexible, adaptive manner. More abstract cognitive models such as ACT-R provide a nice characterization of the functional properties of this level of cognition. The goal with the model we explore here is to understand how more detailed neural mechanisms can work together to produce this functionality.

- Higher (more anterior) levels of PFC encode context/goals/plans to organize sequence of cognitive actions, which are driven by more lower, more posterior PFC areas. Critically, these higher areas do not specify rigid sequences of actions, but rather encode the desired outcome states of the sequence of actions, and provide appropriate context so that appropriate lower-level steps will be selected.
- Each step in a sequence of actions involves a consideration of the reward outcomes and effort costs of the action relative to other possible options.

Affective Influences over Executive Function: Roles of the OFC and ACC

One of the most important features of the PFC and executive function is that it integrates emotional and motivational influences together with high-level cognitive control and planning. The medial and ventral regions of the PFC are particularly important for processing emotional and motivational factors, with considerable data converging on the idea that the ventral medial areas including the orbital prefrontal cortex (OFC) are important for encoding the affective value of stimuli, while the dorsal medial areas (principally the anterior cingulate cortex (ACC)) is important for encoding the affective value of motor actions and plans ().

Here, we explore a model of these areas in the context of relatively simple conditioning tasks in animals.

Other Executive Functions

The main models explored above are intended to cover some of the most central and important aspects of executive function, but this is a very large space and there are many important phenomena that we unfortunately cannot cover (though we plan to expand the scope of what is covered in future revisions, with optional models covering various of the following topics).

For many people, particularly in an academic setting, the first things that may come to mind if asked to name some higher-level cognitive functions might be things like: learning and/or using formal mathematics (like calculus or statistics); or, perhaps, the use of careful logical reasoning to make a major decision. But, in addition to these highly formalized domains, there are many other day-to-day, but none the less important, mental activities that also involve a highly sophisticated level of processing, activities like: planning one's day or a work project, or; resisting the temptation to have dessert when you are trying to lose ten pounds before bathing suit season, or; counting cards in working memory while playing blackjack. All these kinds of mental activities are now known to rely upon the frontal cortex and related structures for their optimal expression. Here is a list of some major categories of distinctive executive functions:

- *Highly structured cognitive activities, often involving formal symbol systems* – Mental activities like learning and/or using mathematics, formal logic, computer programming, creative and/or non-fiction writing, and structured, rational decision-making. All of these require temporally-extended maintenance of task-relevant information, especially of a highly abstract, symbolic nature. The role of language in these and many other executive functions is a very important aspect – language provides a highly flexible mental currency for active maintenance and control over behavior – by remembering specific words or phrases, we can remind ourselves of what we want to achieve, or what we have derived in an initial processing step, etc.
- *Control over encoding and retrieval of episodic information in the hippocampus* – it is highly likely that the hippocampus and PFC/BG systems interact significantly in many forms of executive function, with the rapid learning abilities of the hippocampus complementing the transient, flexible active maintenance properties of the PFC. If the PFC gets distracted, the information is typically gone forever, but the hippocampus can encode and retrieve information in terms of long-lasting synaptic changes. Often, it may be more efficient to use this hippocampal encoding and retrieval instead of persistent active maintenance of information in PFC.

Alternative frameworks and modeling approaches

In this chapter we have focused on one particular theoretical framework, but there have been many other approaches described over the years. Probably the most influential model came from (Baddeley 1986), who especially focused on working memory, but also argued for a "central executive." In particular, he postulated two specific forms of working memory: 1) a *phonological loop* for maintaining verbal information and; 2) a *visuospatial scratchpad* for spatial information. Another highly influential theoretical approach came from (Shallice 1988) who described a supervisory attentional system (SAS) framework. Finally, there is also the very influential traditional AI approach, which we will discuss briefly below.

Motivated largely by the kinds of cognitive functions listed above, traditional AI has largely focused on a design-oriented approach using symbols that has focused on trying to figure out what it would take to solve a particular kind of problem, and then designing a model that does things that way. There is an irony in this approach in that researchers taking this approach are using the very higher-level cognitive functionality they

are trying to explain in order to design a system that will reproduce it. A fundamental problem with this kind of approach is that it basically designs in the very functionality it aims to explain. This is not to say that these kinds of approaches are wholly without merit, only that they are fundamentally limited in what they can ultimately explain. Perhaps for obvious reasons, it has turned out that these kinds of models of cognitive function have been most successful in dealing with the kinds of cognitive function that we listed as being at the highest level - that is, in modeling systems able to do formal mathematics and logic. What they have done less well in has been in accounting for many of the kinds of things that might be considered less high-level, or even lower-level, things which we often take to be automatic. It is for these latter areas, that the biologically informed neural network approach has been most helpful. Thus, these two approaches can be nicely complementary and hybrid approaches are being pursued. For example, the Leabra approach is being hybridized with the ACT-R approach in an architecture called SAL.

All of these approaches are not mutually exclusive, but instead share many common ideas and can be complementary in many ways. In particular, the traditional AI approach, by going straight to solving a high level problem e.g., arithmetic. On the other hand, the goal of the neural network approach we advocate is to provide a more bottom-up model that tries to provide a reductionist account for the emergence of control-like processing based on underlying automatic mechanisms. This is the approach we take with the PBWM framework.

Summary of Key Points

- The prefrontal cortex (PFC) encodes information in an active state through sustained neural firing, which is more flexible and rapidly updatable than using synaptic weight changes.
- The basal ganglia (BG) drives updating (dynamic gating) of PFC active memory states, enhancing flexibility.
- Phasic dopamine signals from midbrain nuclei have the right properties for training BG gating, by transferring reward associations earlier in time to the onset of stimuli that predict subsequent rewards.
- The PFC influences cognitive processing elsewhere in the brain via top-down excitatory biasing, as demonstrated in the Stroop model.
- Developmental changes in active memory can be explained in terms of stronger PFC active maintenance abilities, as demonstrated in the A-not-B model.
- BG dynamic gating can support flexible cognitive function by dynamically encoding some information while ignore other irrelevant information, and updating the contents of active memory. The SIR and n-back models demonstrate these abilities.
- Medial and ventral areas of PFC (orbital prefrontal cortex (OFC) and anterior cingulate cortex (ACC)) convey affective information about stimuli and actions, respectively, and are important for properly evaluating potential actions to be taken (decision making, problem solving, etc).

Acknowledgments

We thank the large number of students who have helped improve this textbook over the years, in addition to the support and understanding of our families.

About the Authors

Randall C. O'Reilly and Yuko Munakata are Professors of Psychology and Neuroscience at the University of California Davis, and this text was written while at the University of Colorado Boulder.

Michael J. Frank is Professor at Brown University.

Thomas E. Hazy is a Senior Research Associate at the University of Colorado Boulder.

Many others have contributed to this text.

```
if this {
    print
}
else {
}
then
// adkf
```

References

Ackley, David H., G. E. Hinton, and T. J. Sejnowski. 1985. "A Learning Algorithm for Boltzmann Machines." *Cognitive Science* 9 (1): 147–69.

Albus, James S. 1971. "A Theory of Cerebellar Function." *Mathematical Biosciences* 10 (1-2): 25–61. http://www.sciencedirect.com/science/article/B6VHX-45F52M2-J8/2/bba55f65c1bf9b826444584ec64ee6c3.

Allport, D. A. 1985. "Distributed Memory, Modular Systems and Dysphasia." In *Current Perspectives in Dysphasia*, edited by S. K. Newman and R. Epstein. Edinburgh: Churchill Livingstone.

Aron, Adam R., Tim E. Behrens, Steve Smith, Michael J. Frank, and Russell A. Poldrack. 2007. "Triangulating a Cognitive Control Network Using Diffusion-Weighted Magnetic Resonance Imaging (MRI) and Functional MRI." *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience* 27 (14): 3743–52. http://www.ncbi.nlm.nih.gov/pubmed/17409238.

Baars, B. J. 1988. *A Cognitive Theory of Consciousness*. New York: Cambridge University Press.

Baddeley, A. D. 1986. *Working Memory*. Oxford University Press.

Badre, David, and Michael J Frank. 2012. "Mechanisms of Hierarchical Reinforcement Learning in Cortico-Striatal Circuits 2: Evidence from FMRI." *Cerebral Cortex* 22 (3). http://www.ncbi.nlm.nih.gov/pubmed/21693491.

Barnes, J. M., and B. J. Underwood. 1960. "Fate of First-List Associations in Transfer Theory." *Journal of Experimental Psychology* 58 (December): 97–105. http://www.ncbi.nlm.nih.gov/pubmed/13796886.

Benda, Jan, Leonard Maler, and André Longtin. 2010. "Linear Versus Nonlinear Signal Transmission in Neuron Models with Adaptation Currents or Dynamic Thresholds." *Journal of Neurophysiology* 104 (5): 2806–20. https://doi.org/10.1152/jn.00240.2010.

Bi, Guo-qiang, and Mu-ming Poo. 1998. "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type." *The Journal of Neuroscience* 18 (24): 10464–72. http://www.jneurosci.org/content/18/24/10464.

Bienenstock, Elie L., Leon N. Cooper, and Paul W. Munro. 1982. "Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex." *The Journal of Neuroscience* 2 (2): 32–48. http://www.ncbi.nlm.nih.gov/pubmed/7054394.

Bosch, Miquel, and Yasunori Hayashi. 2012. "Structural Plasticity of Dendritic Spines." *Current Opinion in Neurobiology*, Synaptic structure and function, 22 (3): 383–88. https://doi.org/10.1016/j.conb.2011.09.002.

Brette, Romain, and Wulfram Gerstner. 2005. "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity." *Journal of Neurophysiology* 94 (5): 3637–42. https://doi.org/10.1152/jn.00686.2005.

Buffalo, Elizabeth A, Pascal Fries, Rogier Landman, Timothy J Buschman, and Robert Desimone. 2011. "Laminar Differences in Gamma and Alpha Coherence in the Ventral Stream." *Proceedings of the National Academy of Sciences of the United States of America* 108 (27): 11262–7. http://www.ncbi.nlm.nih.gov/pubmed/21690410.

Cavanagh, James F, Thomas V Wiecki, Michael X Cohen, Christina M Figueroa, Johan Samanta, Scott J Sherman, and Michael J Frank. 2011. "Subthalamic Nucleus Stimulation Reverses Mediofrontal Influence over Decision Threshold." *Nature Neuroscience* 14 (September): 1462–7. http://www.ncbi.nlm.nih.gov/pubmed/21946325.

Cleeremans, A., and J. L. McClelland. 1991. "Learning the Structure of Event Sequences." *Journal of Experimental Psychology: General* 120 (January): 235–53.

Cohen, J. D., K. Dunbar, and J. L. McClelland. 1990. "On the Control of Automatic Processes: A Parallel Distributed Processing Model of the Stroop Effect." *Psychological Review* 97 (3): 332–61.

Collins, Anne Gabrielle Eva, and Michael Joshua Frank. 2016. "Neural Signature of Hierarchically Structured Expectations Predicts Clustering and Transfer of Rule Sets in Reinforcement Learning." *Cognition* 152 (July): 160–69. https://doi.org/10.1016/j.cognition.2016.04.002.

Collins, Anne G. E., and Michael J. Frank. 2013. "Cognitive Control over Learning: Creating, Clustering, and Generalizing Task-Set Structure." *Psychological Review* 120 (1): 190–229. http://www.ncbi.nlm.nih.

gov/pubmed/23356780.

———. 2014. "Opponent Actor Learning (OpAL): Modeling Interactive Effects of Striatal Dopamine on Reinforcement Learning and Choice Incentive." *Psychological Review* 121 (3): 337–66. http://www.ncbi.nlm.nih.gov/pubmed/25090423.

Cooper, L. N., N. Intrator, B. S. Blais, and H. Shouval. 2004. *Theory of Cortical Plasticity.* New Jersey: World Scientific.

Coslett, H. B., and E. Saffran. 1991. "Simultanagnosia. To See but Not Two See." *Brain* 114 (January): 1523–45.

Cox, Sylvia M. L., Michael J. Frank, Kevin Larcher, Lesley K. Fellows, Crystal A. Clark, Marco Leyton, and Alain Dagher. 2015. "Striatal D1 and D2 Signaling Differentially Predict Learning from Positive and Negative Outcomes." *NeuroImage* 109 (April): 95–101. https://doi.org/10.1016/j.neuroimage.2014.12.070.

Crick, F. 1989. "The Recent Excitement About Neural Networks." *Nature* 337 (February): 129–32. http://www.ncbi.nlm.nih.gov/pubmed/2911347.

Elman, J. L. 1990. "Finding Structure in Time." *Cognitive Science* 14 (2): 179–211.

Felleman, D. J., and D. C. Van Essen. 1991. "Distributed Hierarchical Processing in the Primate Cerebral Cortex." *Cerebral Cortex* 1 (1): 1–47. http://www.ncbi.nlm.nih.gov/pubmed/1822724.

Franceschetti, S, E Guatteo, F Panzica, G Sancini, E Wanke, and G Avanzini. 1995. "Ionic Mechanisms Underlying Burst Firing in Pyramidal Neurons: Intracellular Study in Rat Sensorimotor Cortex." *Brain Research* 696 (12): 127–39. http://www.ncbi.nlm.nih.gov/pubmed/8574660.

Frank, Michael J. 2006. "Hold Your Horses: A Dynamic Computational Role for the Subthalamic Nucleus in Decision Making." *Neural Networks* 19 (8): 1120–36. http://www.ncbi.nlm.nih.gov/pubmed/16945502.

Frank, Michael J., and David Badre. 2012. "Mechanisms of Hierarchical Reinforcement Learning in Corticostriatal Circuits 1: Computational Analysis." *Cerebral Cortex* 22 (3): 509–26. http://www.ncbi.nlm.nih.gov/pubmed/21693490.

Frank, Michael J, and John A Fossella. 2011. "Neurogenetics and Pharmacology of Learning, Motivation, and Cognition." *Neuropsychopharmacology* 36 (January). http://www.ncbi.nlm.nih.gov/pubmed/20631684.

Frank, Michael J., R. C. O'Reilly, and Tim Curran. 2006. "When Memory Fails, Intuition Reigns: Midazolam Enhances Implicit Inference in Humans." *Psychological Science : A Journal of the American Psychological Society / APS* 17 (August): 700–707. http://www.ncbi.nlm.nih.gov/pubmed/16913953.

Frank, M. J. 2005. "When and When Not to Use Your Subthalamic Nucleus: Lessons from a Computational Model of the Basal Ganglia." *Modelling Natural Action Selection: Proceedings of an International Workshop*, January, 53–60.

Frank, M. J., L. C. Seeberger, and R. C. O'Reilly. 2004. "By Carrot or by Stick: Cognitive Reinforcement Learning in Parkinsonism." *Science* 306 (5703): 1940–3. https://www.ncbi.nlm.nih.gov/pubmed/15528409.

Fukushima, Kunihiko. 1980. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position." *Biological Cybernetics* 36 (4): 193–202. http://dx.doi.org/10.1007/BF00344251.

Fuster, J. M., and G. E. Alexander. 1971. "Neuron Activity Related to Short-Term Memory." *Science* 173 (January): 652–54.

Goodale, Melvyn A., and A. David. Milner. 1992. "Separate Visual Pathways for Perception and Action." *Trends in Neurosciences* 15 (1): 20–25.

Hasselmo, Michael E., Clara Bodelon, and Bradley P. Wyble. 2002. "A Proposed Function for Hippocampal Theta Rhythm: Separate Phases of Encoding and Retrieval Enhance Reversal of Prior Learning." *Neural Computation* 14 (4): 793–818. http://www.ncbi.nlm.nih.gov/pubmed/11936962.

Haxby, J. V., M. I. Gobbini, M. L. Furey, A. Ishai, J. L. Schouten, and P. Pietrini. 2001. "Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex." *Science (New York, N.Y.)* 293 (September): 2425–9. http://www.ncbi.nlm.nih.gov/pubmed/11577229.

Hazy, T. E., M. J. Frank, and R. C. O'Reilly. 2006. "Banishing the Homunculus: Making Working Memory Work." *Neuroscience* 139 (April): 105–18. http://www.ncbi.nlm.nih.gov/pubmed/16343792.

Hazy, Thomas E., Michael J. Frank, and R. C. O'Reilly. 2007. "Towards an Executive Without a Homunculus: Computational Models of the Prefrontal Cortex/Basal Ganglia System." *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 362 (1485): 1601–13. http://www.ncbi.nlm.nih.gov/pubmed/17428778.

———. 2010. "Neural Mechanisms of Acquired Phasic Dopamine Responses in Learning." *Neuroscience and Biobehavioral Reviews* 34 (5): 701–20. http://www.ncbi.nlm.nih.gov/pubmed/19944716.

Hertz, John, Anders Krogh, and Richard G. Palmer. 1991. *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley.

Hikida, Takatoshi, Kensuke Kimura, Norio Wada, Kazuo Funabiki, and Shigetada Nakanishi. 2010. "Distinct Roles of Synaptic Transmission in Direct and Indirect Striatal Pathways to Reward and Aversive Behavior." *Neuron* 66: 896–907.

Hikosaka, O., Y. Takikawa, and R. Kawagoe. 2000. "Role of the Basal Ganglia in the Control of Purposive Saccadic Eye Movements." *Physiological Reviews* 80 (3): 953–78. http://www.ncbi.nlm.nih.gov/pubmed/10893428.

Hinton, G. E. 1989. "Learning Distributed Representations of Concepts." In *Parallel Distributed Processing: Implications for Psychology and Neurobiology*, edited by R. G. M. Morris, 46–61. Oxford, England: Clarendon Press.

Hinton, G. E., and T. J. Sejnowski. 1983. "Optimal Perceptual Inference." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC.

Hirshman, E., A. Passannante, and J. Arndt. 2001. "Midazolam Amnesia and Conceptual Processing in Implicit Memory." *Journal of Experimental Psychology. General* 130 (September): 453–65. http://www.ncbi.nlm.nih.gov/pubmed/11561920.

Hopfield, J. J. 1982. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceedings of the National Academy of Sciences of the United States of America* 79 (8): 2554–8. http://www.ncbi.nlm.nih.gov/pubmed/6953413.

———. 1984. "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons." *Proceedings of the National Academy of Sciences of the United States of America* 81 (July): 3088–92. http://www.ncbi.nlm.nih.gov/pubmed/6587342.

Jordan, M. I. 1989. "Serial Order: A Parallel, Distributed Processing Approach." In *Advances in Connectionist Theory: Speech*, edited by J. L. Elman and D. E. Rumelhart. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kachergis, George, Dean Wyatte, R. C. O'Reilly, Roy de Kleijn, and Bernhard Hommel. 2014. "A Continuous-Time Neural Model for Sequential Action." *Philosophical Transactions of the Royal Society B: Biological Sciences* 369 (1655): 20130623. https://doi.org/10.1098/rstb.2013.0623.

Kaczmarek, Leonard K. 2013. "Slack, Slick, and Sodium-Activated Potassium Channels." *ISRN Neuroscience* 2013 (May). https://doi.org/10.1155/2013/354262.

Ketz, Nicholas, Srinimisha G Morkonda, and R. C O'Reilly. 2013. "Theta Coordinated Error-Driven Learning in the Hippocampus." *PLoS Computational Biology* 9 (June): e1003067. http://www.ncbi.nlm.nih.gov/pubmed/23762019.

Kirkwood, A., M. G. Rioult, and M. F. Bear. 1996. "Experience-Dependent Modification of Synaptic Plasticity in Visual Cortex." *Nature* 381 (6582): 526–28. http://www.ncbi.nlm.nih.gov/pubmed/8632826.

Kobatake, E., and K. Tanaka. 1994. "Neuronal Selectivities to Complex Object Features in the Ventral Visual Pathway." *Journal of Neurophysiology* 71 (3): 856–67.

Kohn, Adam. 2007. "Visual Adaptation: Physiology, Mechanisms, and Functional Benefits." *Journal of Neurophysiology* 97 (5): 3155–64. https://doi.org/10.1152/jn.00086.2007.

Kravitz, Alexxai, Benjamin Freeze, Philip Parker, Kenneth Kay, Myo Thwin, Karl Deisseroth, and Anatol Kreitzer. 2010. "Regulation of {}P{}arkinsonian Motor Behaviours by Optogenetic Control of Basal Ganglia Circuitry." *Nature* 466 (7306): 622–26. http://www.nature.com/nature/journal/vaop/ncurrent/full/nature09159.html.

Kravitz, Alexxai V., Lynne D. Tye, and Anatol C. Kreitzer. 2012. "Distinct Roles for Direct and Indirect Pathway Striatal Neurons in Reinforcement." *Nature Neuroscience* 15 (6): 816–18. http://www.ncbi.nlm.

nih.gov/pubmed/22544310.

Kravitz, Dwight J., Kadharbatcha S. Saleem, Chris I. Baker, and Mortimer Mishkin. 2011. "A New Neural Framework for Visuospatial Processing." *Nature Reviews Neuroscience* 12 (4): 217–30. https://doi.org/10.1038/nrn3008.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, 1097–1105. Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kubota, K., and H. Niki. 1971. "Prefrontal Cortical Unit Activity and Delayed Alternation Performance in Monkeys." *Journal of Neurophysiology* 34 (3): 337–47. http://www.ncbi.nlm.nih.gov/pubmed/4997822.

Lamme, Victor A. F. 2006. "Towards a True Neural Stance on Consciousness." *Trends in Cognitive Sciences* 10 (11): 494–501. https://doi.org/10.1016/j.tics.2006.09.001.

Landauer, Thomas K., and Susan T. Dumais. 1997. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge." *Psychological Review* 104 (January): 211–40.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539.

Leutgeb, J. K., S. Leutgeb, M. Moser, and E. Moser. 2007. "Pattern Separation in the Dentate Gyrus and CA3 of the Hippocampus." *Science* 315 (5814): 961–66. http://www.sciencemag.org/cgi/content/abstract/315/5814/961.

Levitt, J. B., D. A. Lewis, T. Yoshioka, and J. S. Lund. 1993. "Topography of Pyramidal Neuron Intrinsic Connections in Macaque Monkey Prefrontal Cortex (Areas 9 & 46)." *Journal of Comparative Neurology* 338: 360–76.

Lim, Sukbin, Jillian L. McKee, Luke Woloszyn, Yali Amit, David J. Freedman, David L. Sheinberg, and Nicolas Brunel. 2015. "Inferring Learning Rules from Distributions of Firing Rates in Cortical Neurons." *Nature Neuroscience* 18 (12): 1804–10. https://doi.org/10.1038/nn.4158.

Lisman, John E., and Anthony A. Grace. 2005. "The Hippocampal-VTA Loop: Controlling the Entry of Information into Long-Term Memory." *Neuron* 46 (5): 703–13. http://www.ncbi.nlm.nih.gov/pubmed/15924857.

Lorincz, Magor L, Katalin A Kekesi, Gabor Juhasz, Vincenzo Crunelli, and Stuart W Hughes. 2009. "Temporal Framing of Thalamic Relay-Mode Firing by Phasic Inhibition During the Alpha Rhythm." *Neuron* 63 (5): 683–96. http://www.ncbi.nlm.nih.gov/pubmed/19755110.

Luczak, Artur, Peter Bartho, and Kenneth D. Harris. 2013. "Gating of Sensory Input by Spontaneous Cortical Activity." *The Journal of Neuroscience* 33 (4): 1684–95. http://www.ncbi.nlm.nih.gov/pubmed/23345241.

Markov, Nikola T., Julien Vezoli, Pascal Chameau, Arnaud Falchier, René Quilodran, Cyril Huissoud, Camille Lamy, et al. 2014. "Anatomy of Hierarchy: Feedforward and Feedback Pathways in Macaque Visual Cortex: Cortical Counterstreams." *Journal of Comparative Neurology* 522 (1): 225–59. https://doi.org/10.1002/cne.23458.

Marr, D. 1969. "A Theory of Cerebellar Cortex." *Journal of Physiology (London)* 202 (January): 437–70.

———. 1971. "Simple Memory: A Theory for Archicortex." *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 262 (841): 23–81. https://doi.org/10.1098/rstb.1971.0078.

Marr, David. 1977. "Artificial IntelligenceA Personal View." *Artificial Intelligence*, January, 37–48.

Marshall, Lisa, Halla Helgadóttir, Matthias Mölle, and Jan Born. 2006. "Boosting Slow Oscillations During Sleep Potentiates Memory." *Nature* 444 (7119). http://www.ncbi.nlm.nih.gov/pubmed/17086200.

McClelland, J. L. 1998. "Connectionist Models and Bayesian Inference." In *Rational Models of Cognitive Processes.*, edited by N. Chater and M. Oaksford. Oxford: Oxford University Press.

McClelland, J L. 2000. "The Basis of Hyperspecificity in Autism: A Preliminary Suggestion Based on Properties of Neural Nets." *Journal of Autism and Developmental Disorders* 30 (October): 497–502. http://www.ncbi.nlm.nih.gov/pubmed/11098891.

McClelland, J. L., B. L. McNaughton, and R. C. O'Reilly. 1995. "Why There Are Complementary Learning

Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory." *Psychological Review* 102 (3): 419–57. http://www.ncbi.nlm.nih.gov/pubmed/7624455.

McCloskey, M., and N. J. Cohen. 1989. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem." In *The Psychology of Learning and Motivation, Vol. 24*, edited by G. H. Bower, 109–64. San Diego, CA: Academic Press.

Mink, J. W. 1996. "The Basal Ganglia: Focused Selection and Inhibition of Competing Motor Programs." *Progress in Neurobiology* 50 (4): 381–425. http://www.ncbi.nlm.nih.gov/pubmed/9004351.

Minsky, M., and S. A. Papert. 1969. *Perceptrons*. Cambridge, MA: MIT Press.

Mollick, Jessica A., Thomas E. Hazy, Kai A. Krueger, Ananta Nair, Prescott Mackie, Seth A. Herd, and R. C. O'Reilly. n.d. "A Systems-Neuroscience Model of Phasic Dopamine."

Montague, P. Read, Peter Dayan, and Terrence J. Sejnowski. 1996. "A Framework for Mesencephalic Dopamine Systems Based on Predictive Hebbian Learning." *Journal of Neuroscience* 16 (5): 1936–47. http://www.ncbi.nlm.nih.gov/pubmed/8774460.

Movellan, J. R., and J. L. McClelland. 1993. "Learning Continuous Probability Distributions with Symmetric Diffusion Networks." *Cognitive Science* 17 (January): 463–96.

Munakata, Y. 1998. "Infant Perseveration: Rethinking Data, Theory, and the Role of Modelling." *Developmental Science* 1 (January): 205–12.

Norman, Kenneth A., Ehren Newman, Greg Detre, and Sean Polyn. 2006. "How Inhibitory Oscillations Can Train Neural Networks and Punish Competitors." *Neural Computation* 18 (7): 1577–1610. http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1577.

Norman, Kenneth A., and R. C. O'Reilly. 2003. "Modeling Hippocampal and Neocortical Contributions to Recognition Memory: A Complementary-Learning-Systems Approach." *Psychological Review* 110 (4): 611–46. http://www.ncbi.nlm.nih.gov/pubmed/14599236.

Oja, E. 1982. "A Simplified Neuron Model as a Principal Component Analyzer." *Journal of Mathematical Biology* 15: 267–73. http://www.ncbi.nlm.nih.gov/pubmed/7153672.

———. 1989. "Neural Networks, Principal Components, and Subspaces." *International Journal of Neural Systems* 1 (1): 61–68. https://www.worldscientific.com/doi/10.1142/S0129065789000475.

Olshausen, B. A., and D. J. Field. 1996. "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images." *Nature* 381 (July): 607. http://www.ncbi.nlm.nih.gov/pubmed/8637596.

O'Reilly, R. C. 1996. "Biologically Plausible Error-Driven Learning Using Local Activation Differences: The Generalized Recirculation Algorithm." *Neural Computation* 8 (5): 895–938. https://doi.org/https://doi.org/10.1162/neco.1996.8.5.895.

———. 2010. "The What and How of Prefrontal Cortical Organization." *Trends in Neurosciences* 33 (8): 355–61. https://doi.org/10.1016/j.tins.2010.05.002.

O'Reilly, R. C., and Michael J. Frank. 2006. "Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia." *Neural Computation* 18 (2): 283–328. http://www.ncbi.nlm.nih.gov/pubmed/16378516.

O'Reilly, R. C., Michael J. Frank, Thomas E. Hazy, and Brandon Watz. 2007. "PVLV: The Primary Value and Learned Value Pavlovian Learning Algorithm." *Behavioral Neuroscience* 121 (1): 31–49. http://www.ncbi.nlm.nih.gov/pubmed/17324049.

O'Reilly, R. C., and J. L. McClelland. 1994. "Hippocampal Conjunctive Encoding, Storage, and Recall: Avoiding a Tradeoff." *Hippocampus* 4 (6): 661–82.

O'Reilly, R. C., and Y. Munakata. 2000. *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Cambridge, MA: MIT Press.

O'Reilly, R. C., Dean R. Wyatte, and John Rohrlich. 2017. "Deep Predictive Learning: A Comprehensive Model of Three Visual Streams." *arXiv:1709.04654 [Q-Bio]*, September. http://arxiv.org/abs/1709.04654.

Patterson, K., P. J. Nestor, and T. T. Rogers. 2007. "Where Do You Know What You Know?: The Representation of Semantic Knowledge in the Human Brain." *Nature Reviews* 8 (12): 976–87. http://www.ncbi.nlm.nih.gov/pubmed/18026167.

Piaget, J. 1954. *The Construction of Reality in the Child*. New York: Basic Books.

Pinker, S., and A. Prince. 1988. "On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition." *Cognition* 28 (April): 73–193. http://www.ncbi.nlm.nih.gov/pubmed/2450717.

Plaut, David C., and Tim Shallice. 1993. "Deep Dyslexia: A Case Study of Connectionist Neuropsychology." *Cognitive Neuropsychology* 10 (5): 377–500.

Plaut, D. C., J. L. McClelland, M. S. Seidenberg, and K. Patterson. 1996. "Understanding Normal and Impaired Word Reading: Computational Principles in Quasi-Regular Domains." *Psychological Review* 103 (July): 56–115. http://www.ncbi.nlm.nih.gov/pubmed/8650300.

Posner, M. I. 1980. "Orienting of Attention." *Quarterly Journal of Experimental Psychology* 32 (1): 3–25.

Quiroga, R. Quian, L. Reddy, G. Kreiman, C. Koch, and I. Fried. 2005. "Invariant Visual Representation by Single Neurons in the Human Brain." *Nature* 435 (7045): 1102–7. https://doi.org/10.1038/nature03687.

Ranganath, Charan, and Maureen Ritchey. 2012. "Two Cortical Systems for Memory-Guided Behaviour." *Nature Reviews Neuroscience* 13 (10): 713–26. http://www.ncbi.nlm.nih.gov/pubmed/22992647.

Redondo, Roger L., and Richard G. M. Morris. 2011. "Making Memories Last: The Synaptic Tagging and Capture Hypothesis." *Nature Reviews Neuroscience* 12 (1): 17–30. https://doi.org/10.1038/nrn2963.

Rescorla, R. A., and A. R. Wagner. 1972. "A Theory of Pavlovian Conditioning: Variation in the Effectiveness of Reinforcement and Non-Reinforcement." In *Classical Conditioning II: Theory and Research*, edited by A. H. Black and W. F. Prokasy, 64–99. New York: Appleton-Century-Crofts.

Reynolds, Jeremy R., and R. C. O'Reilly. 2009. "Developing PFC Representations Using Reinforcement Learning." *Cognition* 113 (December): 281–92. http://www.ncbi.nlm.nih.gov/pubmed/19591977.

Ritvo, Victoria J. H., Nicholas B. Turk-Browne, and Kenneth A. Norman. 2019. "Nonmonotonic Plasticity: How Memory Retrieval Drives Learning." *Trends in Cognitive Sciences* 23 (9): 726–42. https://doi.org/10.1016/j.tics.2019.06.007.

Rohde, Douglas L. T. 2002. "A Connectionist Model of Sentence Comprehension and Production." PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Rudy, Jerry W. 2015. "Variation in the Persistence of Memory: An Interplay Between Actin Dynamics and AMPA Receptors." *Brain Research*, Brain and Memory: Old Arguments and New Perspectives, 1621 (September): 29–37. https://doi.org/10.1016/j.brainres.2014.12.009.

Rumelhart, David E., and David Zipser. 1985. "Feature Discovery by Competitive Learning*." *Cognitive Science* 9 (1): 75–112. https://doi.org/10.1207/s15516709cog0901_5.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323 (9): 533–36.

Sanchez-Vives, Maria V., Lionel G. Nowak, and David A. McCormick. 2000. "Cellular Mechanisms of Long-Lasting Adaptation in Visual Cortical Neurons in Vitro." *Journal of Neuroscience* 20 (11): 4286–99. http://www.jneurosci.org/content/20/11/4286.

Sanger, T. D. 1989. "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network." *Neural Networks* 2 (January): 459–73.

Schmidhuber, Jürgen. 2015. "Deep Learning in Neural Networks: An Overview." *Neural Networks* 61 (January): 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

Schultz, W., P. Dayan, and P. R. Montague. 1997. "A Neural Substrate of Prediction and Reward." *Science* 275 (5306): 1593–9. http://www.ncbi.nlm.nih.gov/pubmed/9054347.

Seidenberg, M. S., and J. L. McClelland. 1989. "A Distributed, Developmental Model of Word Recognition and Naming." *Psychological Review* 96 (November): 523–68. http://www.ncbi.nlm.nih.gov/pubmed/2798649.

Shallice, T. 1988. *From Neuropsychology to Mental Structure*. New York: Cambridge University Press.

Shen, Weixing, Marc Flajolet, Paul Greengard, and D. James Surmeier. 2008. "Dichotomous Dopaminergic Control of Striatal Synaptic Plasticity." *Science* 321 (5890): 848–51. http://www.ncbi.nlm.nih.gov/pubmed/18687967.

Shouval, Harel Z., Samuel S.-H. Wang, and Gayle M. Wittenberg. 2010. "Spike Timing Dependent Plasticity:

A Consequence of More Fundamental Learning Rules." *Frontiers in Computational Neuroscience* 4 (19). http://www.ncbi.nlm.nih.gov/pubmed/20725599.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, et al. 2017. "Mastering the Game of Go Without Human Knowledge." *Nature* 550 (7676): 354–59. https://doi.org/10.1038/nature24270.

Smolensky, P. 1986. "Information Processing in Dynamical Systems: Foundations of Harmony Theory." In *Parallel Distributed Processing. Volume 1: Foundations*, edited by D. E. Rumelhart, J. L. McClelland, and PDP Research Group, 282–317. Cambridge, MA: MIT Press.

St John, M. F., and J. L. McClelland. 1990. "Learning and Applying Contextual Constraints in Sentence Comprehension." *Artificial Intelligence* 46 (January): 217–57.

Stroop, J. R. 1935. "Studies of Interference in Serial Verbal Reactions." *Journal of Experimental Psychology* 18 (January): 643–62.

Sutton, R. S., and A. G. Barto. 1981. "Toward a Modern Theory of Adaptive Networks: Expectation and Prediction." *Psychological Review* 88 (2): 135–70. http://www.ncbi.nlm.nih.gov/pubmed/7291377.

———. 1998. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press. http://www.cs.ualberta.ca/%20sutton/book/ebook/the-book.html.

Ungerleider, L. G., and M. Mishkin. 1982. "Two Cortical Visual Systems." In *The Analysis of Visual Behavior*, edited by D. J. Ingle, M. A. Goodale, and R. J. W. Mansfield, 549–86. Cambridge, MA: MIT Press.

Urakubo, Hidetoshi, Minoru Honda, Robert C. Froemke, and Shinya Kuroda. 2008. "Requirement of an Allosteric Kinetics of NMDA Receptors for Spike Timing-Dependent Plasticity." *The Journal of Neuroscience* 28 (13): 3310–23. http://www.ncbi.nlm.nih.gov/pubmed/18367598.

Widrow, B., and M. E. Hoff. 1960. "Adaptive Switching Circuits." In *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, 96–104.

Zalocusky, Kelly A., Charu Ramakrishnan, Talia N. Lerner, Thomas J. Davidson, Brian Knutson, and Karl Deisseroth. 2016. "Nucleus Accumbens D2R Cells Signal Prior Outcomes and Control Risky Decision-Making." *Nature* 531 (7596): 642–46. https://doi.org/10.1038/nature17400.