Chapter 1: Reverse engineering the brain

If the brain is a computer, what is it computing? This chapter argues that the brain computes solutions to statistical decision problems. The fundamental problem facing the brain is choosing utility-maximizing actions under uncertainty about the state of the world. This runs into tractability issues, necessitating approximate solutions, which can be realized using an elementary set of neural building blocks.

How does the brain work? You've probably heard this question many times, but what is it really asking? What kind of answer are we looking for?

Asking how something works is fundamentally a question about how it serves a function. The heart pumps, the stomach digests, the brain thinks. Thus, the question is really asking: "How does the brain produce thought?" To answer this question, we need to define thought in a precise way. A widely accepted modern definition is that *thought is computation*—the manipulation of representations for some purpose.

Philosophers have debated how exactly to define computation, and how to determine whether the brain (or any other object) can be characterized as a computer. The definition here captures a fairly standard (but not universal) view. See Maley (2022) for further discussion.

Let's unpack this definition while trying not to get too bogged down in philosophical issues. Roughly speaking, representations are *about* other things: the word "apple" is a representation of the concept APPLE or some particular apple in the sense that when I say "apple" a listener knows that I am referring to one of those things. Similarly, a collection of neurons represents an apple in the sense that a downstream neuron can interpret their activity pattern in terms of information about the apple. The downstream neuron can then do something with this information by participating in a computation (e.g., planning a reaching movement, comparing the apple to other apples in memory, deciding whether to eat it, etc.). Planning a movement might involve predicting what happens if we grasp the apple in a particular way, but critically we don't have to grasp the actual apple to make this prediction—we might mentally simulate grasping a representation of the apple. This illustrates how mental computations are manipulations of representations. Finally, it's obvious that we don't do these manipulations willy-nilly; they are purposeful. Another way of saying this is that the manipulations are part of algorithms that solve particular problems.

For a more nuanced treatment of representation in neuroscience, see Baker et al. (2022).

Drawing upon these ideas, the computational neuroscientist David Marr articulated an influential framework for studying computational systems at multiple levels of analysis (Marr, 1982):

1. **Computational level**: What is the problem being solved by the

system?

2. **Representational/algorithmic level**: How is the problem solved algorithmically?

3. **Implementation level**: How is the algorithm realized physically?

In motivating his levels of analysis, Marr famously declared, "Trying to understand perception by studying only neurons is like trying to understand bird flight by studying only feathers: It just cannot be done." This declaration has conceptual and methodological interpretations. The conceptual interpretation is that there is no such thing as perception defined only in terms of neurons—it's a category error, because there is something about perception that is irreducibly mental. Note that this is not the same as endorsing a Cartesian dualism in which mind and brain are separate entities; rather, the point is that cognitive processes like perception are *defined* in terms of mental representations of objects, shapes, and so on. Neuroscience can study how neurons implement these processes, but it cannot simply replace them with neural descriptions.

A more recent rendition of Marr's argument can be found in Krakauer et al. (2017).

If one looks really closely at the atomic structure of a transistor, one won't recognize a computing unit, but of course digital computers are made out of transistors. Likewise, there is nothing recognizable as "cognition" if one only looks at the details of neuronal biochemistry. Marr's levels allow us to adopt a level of abstraction at which cognition comes into view.

"The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise." (Dijkstra, 1972)

The methodological interpretation is that in practice it's virtually impossible to make progress in cognitive neuroscience by pursuing a purely bottom-up approach of "just looking" at neurons. This point is vividly illustrated by the exercise of applying analogues of modern neuroscience techniques to understanding a microprocessor (Jonas and Kording, 2017), a case where we know how it works because we designed it, and yet these techniques are largely useless in the absence of structured hypotheses about the microprocessor's functions and design principles. For example, Jonas and Kording identified which "behaviors" (video games running on the microprocessor) are disrupted by lesioning individual transistors (Figure 1), a common approach to establishing causality in neuroscience. They write:

See Gershman (2021) for a discussion of the "innocent eye" fallacy in neuroscience.

> This finding of course is grossly misleading. The transistors are not specific to any one behavior or game but rather implement simple functions, like full adders. The finding that some of them are important while others are not for a given game is only indirectly indicative of the transistor's role and is unlikely to generalize to other games.

The moral of the story is that we can make swifter progress by (i) positing the computational problems that the brain is trying to solve,

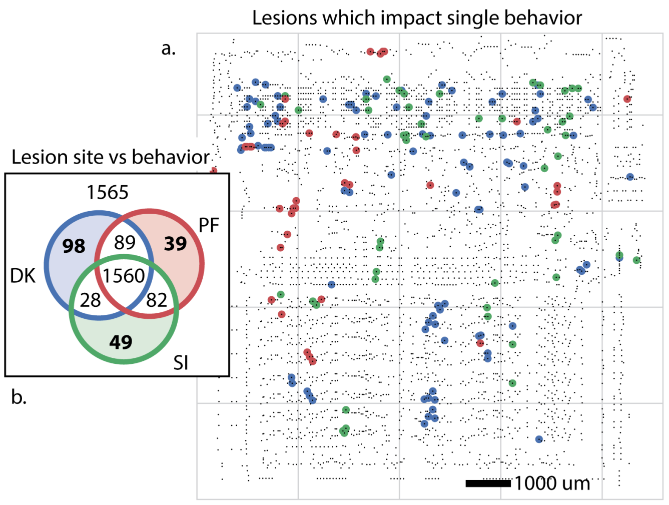

Figure 1: **Lesioning a microprocessor to identify the functions of individual transistors: an exercise in futility?** Colors indicate different "behaviors" (video games). Reproduced from Jonas and Kording (2017).

(ii) engineering algorithmic solutions to these problems, and (iii) modeling how the brain could implement the algorithmic solutions under biological constraints. This is the reverse engineering approach to understanding the brain, structured according to Marr's levels of analysis. Importantly, the levels mutually constrain one another. For example, we can eliminate certain algorithmic hypotheses that have no plausible biological implementation, and we can eliminate biological implementations of algorithms that lack psychological plausibility.

Reverse engineering comes with its own risks. What if we invent a "just-so" story about the brain, a wishful fantasy about its computational principles? My answer is that reverse engineering is subject to the same empirical discipline as any other theoretical approach; we have to always be asking, *what can it explain?* The theory is good if it can explain a large number of phenomena with a small number of principles. We'll know we're succeeding when the textbooks start to get shorter rather than longer (Figure 2).

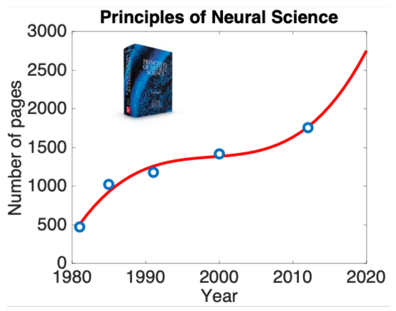

Figure 2: **A standard neuroscience textbook gets longer with each edition.** If theorists are doing their jobs right, this book should be getting shorter!

1 *The computational level*

At first glance, the brain seems to be solving many different kinds of problems: perception, decision making, motor control, learning, memory, language understanding, and so on. On further inspection, these all appear to be variations on one kind of problem, namely a statistical decision problem. This might sound like a radically narrow construal of brain function, but it isn't; the technical definition of a statistical decision problem is general enough to encompass many, if not all, specific functions carried out by the brain. Let's take a closer look at the technical definition.

1.1 *Decision theory*

The basic setup of decision theory consists of two objects: an agent and the world (Figure 3). The agent has a perceptual apparatus to measure signals and rewards from the world, and a motor apparatus to produce actions operating on the world. In between these two apparatuses is the agent's brain, which computes probabilistic beliefs about the state of the world based on the sensory signals that it receives. These beliefs are used to calculate the expected reward for each possible action. Finally, the agent chooses an action, which induces reward delivery and changes in the state of the world.

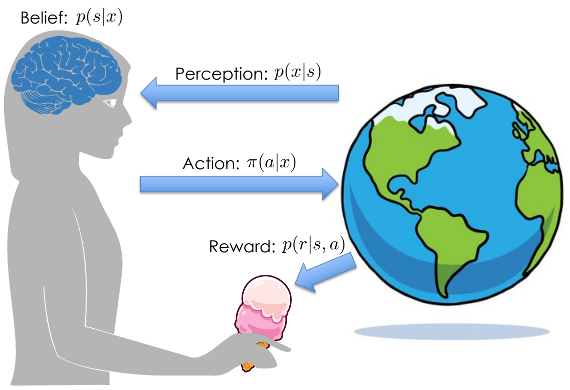

Figure 3: **Elements of the decision problem.**

More formally, the true state of the world is denoted by s, drawn from $p(s)$. For example (Figure 4), s might be the color of an apple, drawn from the distribution of apple colors, $p(s)$. Conditional on the state, a signal x is generated from an observation distribution $p(x|s)$. In general, the state may not be fully observable (x can include states, actions, and rewards whenever these are observed). In our example, x corresponds to the activation of photoreceptors in the retina, and $p(x|s)$ corresponds to the process by which photons of a particular wavelength travel from the apple's surface to the photoreceptors, where they interact with light-sensitive proteins (opsins). This process is probabilistic due to several sources of randomness (e.g., the distribution of opsins across photoreceptors, the stochastic absorption of photons by opsins).

All the variables (state, signal, action, reward) can be multidimensional (e.g., indexed by space or time), discrete or continuous, stochastic or deterministic. Later we will explore more specific models that make different assumptions about the structure of these variables. We will assume discrete variables in what follows.

The agent samples an action a from its (possibly stochastic) *policy* $\pi(a|x)$, and then collects a reward r from a reward distribution $p(r|s,a)$. For example, I might ask you to report the color of the apple using a color wheel and reward you based on your accuracy. The action generation process in this case is probabilistic due to motor

In Chapter 11, we'll look at cases where random action selection arises from particular optimization problems.

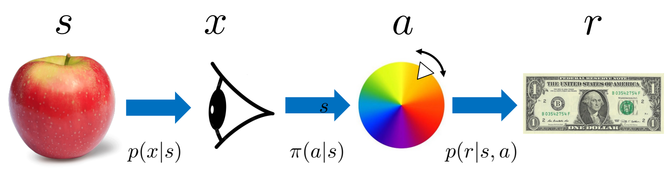

Figure 4: **Example decision problem (color perception).**

noise.

The agent derives utility $u(r)$ from reward r. For example, r might be money you earn from the task I give you, and $u(r)$ is how much you value the money, which depends on factors like your current wealth level and the purchasing power of the money. This emphasizes the fact that utility is distinct from nominal quantities like dollars, number of calories, etc. Generally speaking, utility is internally generated. Where this comes from is a deep question. Singh et al. (2010) take an evolutionary approach to the origin of utility. The basic idea is that agents need to maximize fitness for survival and reproduction, but this is not easily measurable. Therefore, evolution has equipped us with utility functions that serve as imperfect proxies for fitness.

At first glance, the setup of decision theory seems too austere to encompass everything that the brain does. What about language, mathematics, logic, analogy? All of these are (at least in principle) reducible to decision theory, provided we consider a rich enough space of states, actions, and signals. For example, we can think of language production as a decision problem where the state space consists of referents (i.e., what the speaker is talking about), the action space consists of utterances, and the signal space consists of the speaker's sensory inputs (potentially including the utterances of other speakers). The listener needs to infer the referent of the speaker based on the signals they observe, while the speaker needs to choose utterances that convey the referent efficiently to the listener (Gibson et al., 2019). This illustrates how we can conceptualize even high-level cognition in terms of decision theory.

1.2 *The optimal policy*

There are different ways to think about what it means for a policy to be optimal. An agent could choose a policy that maximizes the worst possible utility (*minimax* optimality) or it could choose a policy that is not dominated by any other policy (i.e., no other policy yields as good or better utility across all possible states, a property known as *admissibility*). We will focus on *Bayesian decision theory*, which turns out to be equivalent to these other notions of optimality under certain conditions (see next section). The key idea in Bayesian

A good general introduction to Bayesian decision theory is Berger (1985).

decision theory is that the agent is trying to maximize its *expected utility* $\bar{u}(\pi) = \mathbb{E}[u(r)|\pi]$ given its beliefs about the hidden state of the world:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \bar{u}(\pi). \tag{1}$$

There are several sources of randomness going into the expectation. First, the policy may be stochastic. Second, the agent doesn't have access to the hidden state. Third, the reward may be stochastic. The agent's expected utility is how much utility it believes it will gain under some policy, averaging over these sources of randomness:

$$\bar{u}(\pi) = \sum_x p(x) \sum_a \pi(a|x) \sum_s p(s|x) \sum_r p(r|s,a)u(r). \tag{2}$$

The term $p(s|x)$ is the agent's probabilistic belief about the hidden state s given the signal x. It is commonly referred to as the *posterior distribution*, to contrast it with the agent's *prior distribution* $p(s)$. We can obtain the posterior from the prior via Bayes' rule:

Although here the prior is taken to be the ground truth state distribution, in reality we rarely know this ground truth. Our priors are typically *subjective*. This causes discomfort for some people.

$$p(s|x) = \frac{p(x|s)p(s)}{p(x)}. \tag{3}$$

The first term in the numerator, $p(x|s)$, is the *likelihood* of state s. Intuitively, this term expresses how well a hypothetical state "fits" the data. The denominator is the *marginal likelihood* $p(x) = \sum_s p(x|s)p(s)$, a normalizing constant which ensures that the posterior probabilities sum to 1. The marginal likelihood is actually the thorniest part of Bayesian computation, because it is generally intractable when the state space is large. This leads to approximation techniques that we will cover later.

Marginalization refers to summing over one variable to get the marginal distribution of another variable.

1.3 *Why be Bayesian?*

In the previous section, we took Bayesian decision theory as our standard for optimality. Why? This section briefly summarizes 3 arguments.

The logician's argument. In Boolean logic, the truth value of a proposition is represented by 0 (false) or 1 (true), and the truth value of complex propositions can be calculated by combining elementary propositions using algebraic operations (the Boolean algebra). For example, if A and B represent the truth values of two propositions, then AB gives the truth value of their conjunction (A and B are both true) and $A + B - AB$ gives the truth value of their disjunction (A or B is true). We can also negate a proposition using $1 - A$ (A is not true). The operations of Boolean algebra always yield values of 0 or 1; truth values are known with complete certainty. What if you're unsure about the truth value of elementary propositions like A and B? Is there a "soft" version of Boolean logic that correctly represents and propagates some measure of "plausibility"? We'd like this measure to be a real number (so that it can encode continuous degrees of plausibility) and internally consistent (logically equivalent propositions should have the same plausibility). We'd also like it to recover Boolean logic as a special case when plausibility is maximal or minimal (corresponding to complete certainty). It turns out that only probabilities updated according to Bayes' rule satisfy these requirements. Thus, Bayesian probability theory can be viewed as a natural extension of Boolean logic.

This result is known as *Cox's Theorem* (Cox, 1946). See Jaynes (2003) for an accessible introduction.

The decision theorist's argument. You might be uncomfortable with the idea that your beliefs, and hence your decisions, depend on a subjective prior distribution. Wouldn't it be better if we could devise a policy which was "objective" in some sense? One approach is to identify policies which are never worse than any other policy (this is the property of admissibility introduced above). It turns out that every admissible policy corresponds to a Bayesian policy (i.e., a policy that maximizes expected utility under the posterior) for *some* prior (not necessarily your subjective prior). Every minimax policy is also equivalent to a Bayesian policy under some prior. This means that Bayesian decision theory is in a sense inevitable for a decision maker who wants to avoid being dominated or avoid the worst possible outcome.

This result is known as the *Complete Class Theorem* (Wald, 1947). Note that technically the corresponding prior may be "improper" (doesn't sum to 1).

The gambler's argument. Suppose I offer you the following bet: if it rains tomorrow, I'll give you \$1, otherwise you get nothing. To purchase this bet, you need to pay me $\$q$. A Bayesian decision maker will purchase the bet if $q < b$, where b is the posterior probability that it will rain tomorrow. In this case, I can't make any money off of you in expectation. Now suppose that instead of posterior probability you assign a possibly non-Bayesian plausibility to each event outcome. If these plausibilities violate the axioms of probability (and hence are non-Bayesian), then it is possible to construct a bet that you will accept and yet you will be guaranteed to lose money. Thus, there is a financial incentive to be Bayesian.

This result is known as the *Dutch Book Theorem* (De Finetti, 1931).

2 *The algorithmic level*

Real agents have constraints on computation, memory, and data. These constraints delimit what kinds of algorithms are realizable. To reverse engineer the brain's algorithms, we need to understand both the requirements of different algorithms and the brain's physical constraints. The algorithmic level of analysis focuses on the former; we will come to the brain's physical constraints when we discuss the implementational level.

2.1 *Complexity, efficiency, tractability*

We can characterize the requirements of an algorithm along several dimensions:

1. **Time complexity**: how much computation is required?
2. **Space complexity**: how much memory is required?
3. **Sample complexity**: how much data are required?

Computer scientists use these dimensions to determine the efficiency of algorithms: if any of the complexity measures cannot be expressed as a polynomial function of the input size N (e.g., they scale exponentially with N), an algorithm is considered inefficient (Arora and Barak, 2009). A problem for which no efficient algorithm exists is considered intractable.

A polynomial function of order N is defined as $f(x) = \sum_{n=0}^{N} \omega_n x^n$, with coefficients $(\omega_1, \ldots, \omega_N)$. The Cobham-Edmonds thesis states that a problem is tractable only if there exists an efficient (polynomial-time) algorithm that can compute a solution.

To illustrate, suppose the state space consists of N variables, $s = (s_1, \ldots, s_N)$, where each variable can take one of K discrete values. Computing the normalizing constant for Bayes' rule then requires summing over K^N possible configurations. This might occur, for example, in the setting where x corresponds to images and s corresponds to the set of N objects in a scene, each of which could belong to K possible categories. Naively trying to enumerate all possible states is inefficient because N appears in the exponent, and thus the time and space complexity of exhaustive enumeration is exponential in the input size (N in this case).

Combinatorial problems of this kind are everywhere. We will encounter them again in the context of learning and decision making. They cannot be efficiently solved by algorithms that rely on exhaustive enumeration. More generally, exponential complexity frequently arises in high-dimensional problems where some computation requires exhaustive coverage of the space—the *curse of dimensionality* (Bellman, 1957).

If a problem is intractable, it is unlikely that our brains evolved to solve it. This suggests that we should only try to reverse engineer the brain's efficient solutions to tractable problems. To this end, we will focus on algorithms with polynomial complexity that have been shown to work in practice. Our strategy will be to identify behavioral and neural signatures of these algorithms, and to investigate how they could be implemented with neural machinery.

2.2 Resource rationality

Efficiency is a rather weak constraint on the space of algorithms, since often many different algorithms enjoy polynomial complexity for a given problem. In some cases, we can make stronger inferences by explicitly specifying the resource constraints as part of the objective function and then asking what algorithms optimize this objective function. This is the essential idea of *resource rationality* (Lieder and Griffiths, 2020).

To formalize this idea, consider an agent with capacity $\mathcal{C}$, measured in resource units (e.g., time, memory, computation, information), which it can't exceed. The amount of resources consumed by implementing policy π is given by $c(\pi)$. The resource-rational policy optimizes expected utility subject to the resource capacity limit:

$c(\pi)$ includes not only the resource cost of executing the policy, but also all the antecedent computation (sensing, belief computation, etc.) needed to compute the policy.

$$\pi^* = \underset{\pi: c(\pi) \leq \mathcal{C}}{\operatorname{argmax}} \ \bar{u}(\pi). \tag{4}$$

We ultimately aim to ground the resource constraints in terms of biology (e.g., the brain's energy budget). In some cases, we lack sufficiently detailed knowledge about biological resource constraints, so we fall back on more abstract and heuristic constraints as placeholders.

One subtlety of resource rationality is that identifying the optimal resource-rational agent is itself computationally demanding. If we incorporate this cost into our analysis, then we would have a new optimization problem which itself requires optimization. This threatens an infinite regress that defeats the point of resource rationality. One way to get around the infinite regress problem is to assume that some of this optimization is happening on an evolutionary or developmental time scale, rather than in real time during task performance. For example, we may have some heuristics that help us efficiently find solutions to certain decision problems; these heuristics were not necessarily discovered through one agent's trial and error, but rather through the collective experience of multiple agents. Resource rationality isn't defined as an optimization procedure happening in one agent's head (although that might happen in some cases)—it is the outcome of an optimization procedure that is the product of multiple pathways (including biological evolution, cultural transmission, and cognitive development).

Pushing the optimization problem beyond a single agent does not completely solve the issue, because this optimization problem is itself intractable (Rich et al., 2020). Thus, it is unlikely that the brain has evolved to be fully resource-rational. In applying resource rationality, the goal is not to claim that the brain has been fully optimized in this way, but rather to show that the optimal solutions are able to explain some of the things that the brain actually does.

2.3 *Algorithmic design principles*

We haven't discussed any particular algorithms yet, but as a preview here are a few design principles which we will encounter throughout the book:

- **Divide and conquer**. Break complex problems down into smaller ones that are easier to solve.
- **Use randomness**. Rather than trying to exhaustively enumerate the elements of some space, sample the elements from a probability distribution.
- **Follow gradients**. Rather than searching blindly in some space, follow the direction of steepest ascent on the function you want to maximize (the objective function).

Bayesian inference can be approximated efficiently using a combination of sampling and gradients, as we will explore later.

- **Regularize**. The natural objective function for Bayesian decision theory is the expected utility. But directly optimizing this function might lead to getting stuck in local optima (suboptimal solutions). Moreover, we often don't have direct access to expected utility, but only an approximation of it, in which case the objective function is a noisy version of the function we actually care about. We can "smooth" the optimization landscape by adding terms of the objective function which make it better-behaved and more robust to noise.
- **Reduce redundancy**. Compress signals so that they consume fewer resources.
- **Reuse computations**. Rather than computing a new solution to each problem, store solutions to past problems so that they can be reused. This exploits patterns of similarity in the space of solutions: if many problems can be solved the same way, just learn one solution.

Much of the book will be devoted to explaining the logic of these design principles and how the brain implements them.

3 *The implementation level*

There are many physical implementations of a given algorithm. You can build a universal computer out of transistors, cells, ant colonies, pasta, and so on. It is commonly believed that the brain's elementary computing units are neurons. Each neuron implements a relatively simple computation; wiring up many neurons together makes complex computation possible. The same idea is the foundation of modern artificial neural networks. This section introduces the basic primitives that we will work with in subsequent chapters.

In reality, each neuron is extremely complex. We'll later consider possible computational roles for some of this complexity.

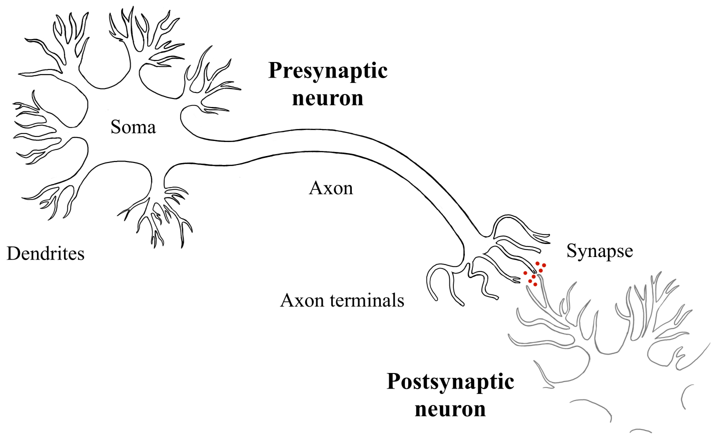

Figure 5: **Schematic of a neuron and synapse.** Red dots show neurotransmitters. Credit: Cindy Luo.

The most important elements of a neuron are shown in Figure 5. A neuron receives inputs from other cells along its dendrites. These inputs come in the form of neurotransmitters released by the axon terminals of other neurons, which then diffuse across the synapse and bind to receptors on dendrites. Neurotransmitters are released (probabilistically) when a neuron is strongly depolarized (an action potential, also known as a spike), producing an electrical signal that is propagated along the axon. Depolarization arises from integration of electrical potentials at dendrites induced by opening of ion channels, which are controlled by the binding state of neurotransmitter receptors. Action potentials are generated when the postsynaptic potential exceeds a threshold.

When discussing transmission of signals across synapses, we will use "postsynaptic" to refer to the receiving neuron and "presynaptic" to refer to the sending neuron.

The brain primarily uses two neurotransmitters: glutamate, which causes excitatory potentials (by opening channels for positively charged ions, typically sodium and calcium), and GABA, which causes inhibitory potentials (by opening channels for negatively charged ions, typically chloride). We will also encounter another class of chemicals released from axon terminals, neuromodulators such as dopamine, serotonin, acetylcholine, and norepinephrine (also known as noradrenaline). As their name suggests, neuromodulators typically modulate the response of neurons to neurotransmitters. The diversity of modulatory effects will be discussed in later chapters.

The number of receptors on the postsynaptic membrane determines its sensitivity to presynaptic signals. Roughly speaking, we can think of the total number of glutamate receptors as a summary of synaptic strength (or "weight"). A postsynaptic neuron with greater synaptic strength will be more likely to produce an action potential given the same amount of presynaptic neurotransmitter release.

Synaptic strengths are modifiable, a process known as *synaptic plasticity*. These modifications typically depend on a combination of presynaptic and postsynaptic activity. The classic example of this (*Hebbian plasticity*) is the strengthening of a synapse after the coincident firing of presynaptic and postsynaptic neurons. As we will see, biologically realistic plasticity rules are more complex than this simple formulation, but nonetheless it captures a key feature of learning in the brain. In addition to synaptic strength, the overall sensitivity of the neuron to inputs is modifiable, a process known as *intrinsic plasticity*. For example, brief stimulation of a neuron can enhance its propensity to generate spikes, whereas longer stimulation reduces this propensity.

"Neurons that fire together, wire together." Although this phrase is often attributed to Donald Hebb, it was actually coined by Shatz (1992).

Despite their simplicity, these neural primitives can be used to implement many of the algorithmic design principles described earlier. We can divide and conquer by wiring up neurons into modules that each solve a different part of a complex problem. The mapping from state s to spikes is probabilistic (as discussed in the next chapter); this can be used for sampling (e.g., for approximating Bayesian inference). The synaptic weights can be adjusted through learning (synaptic plasticity), and the plasticity rules can be constructed to approximately follow the gradient of some objective function (possibly with the help of non-local information provided by neuromodulators). The objective function can incorporate biases for simplicity (e.g., the weights should be close to 0) and compression (e.g., responses of different neurons to the same inputs should be non-redundant). The gain and threshold parameters can also be adjusted (a form of intrinsic plasticity) to optimize simplicity and compression. Finally, the learned weights can potentially be reused to solve multiple problems. These ideas will be elaborated later in the book.

The neural primitives also let us think more concretely about what the brain's limited resources are at the level of cell biology. Spikes are metabolically expensive, so cells cannot spike with arbitrarily high rates for arbitrarily long periods of time. Maintaining a reliable response to inputs (i.e., a high signal-to-noise ratio) is also metabolically expensive. Finally, maintenance of synaptic weights is metabolically expensive, so there may be evolutionary pressure for the weights to be close to 0 (another impetus for simplicity). These metabolic costs imply that the brain should economize on the number of neurons, their average firing rate, the reliability of firing, and the number of connections between neurons.

See Niven (2016) for a review of the brain's energy budget.

4 Conclusion

We started with the question, "How does the brain produce thought?" Now we have the broad strokes of an answer. The brain is designed by evolution to compute (approximately) resource-rational solutions to statistical decision problems. It achieves this by using a number of algorithmic design principles that achieve good performance with low resource requirements. These principles can be implemented with networks of simple neurons connected by plastic synapses.

Study questions

1. What are the advantages and disadvantages of the reverse engineering approach?
2. If priors are subjective, are Bayesian theories unfalsifiable?
3. How is it possible to find optimal resource-constrained policies when the policy search is itself resource-constrained? And doesn't this threaten an infinite regress, where each optimization is nested within an even more difficult optimization problem?

References

Arora, S. and Barak, B. (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press.

Baker, B., Lansdell, B., and Kording, K. P. (2022). Three aspects of representation in neuroscience. *Trends in Cognitive Sciences*, 26:942–958.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American Journal of Physics*, 14:1–13.

De Finetti, B. (1931). Sul significato soggettivo della probabilittextà. *Fundamenta Mathematicae*, 17:298–329.

Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, 15:859–866.

Gershman, S. J. (2021). Just looking: The innocent eye in neuroscience. *Neuron*, 109:2220–2223.

Gibson, E., Futrell, R., Piantadosi, S. P., Dautriche, I., Mahowald, K., Bergen, L., and Levy, R. (2019). How efficiency shapes human language. *Trends in Cognitive Sciences*, 23:389–407.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

Jonas, E. and Kording, K. P. (2017). Could a neuroscientist understand a microprocessor? *PLoS Computational Biology*, 13:e1005268.

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., and Poeppel, D. (2017). Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, 93:480–490.

Lieder, F. and Griffiths, T. L. (2020). Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1.

Maley, C. J. (2022). How (and why) to think that the brain is literally a computer. *Frontiers in Computer Science*, 4:970396.

Marr, D. (1982). *Vision: A Computational Approach*. Freeman.

Niven, J. E. (2016). Neuronal energy consumption: biophysics, efficiency and evolution. *Current Opinion in Neurobiology*, 41:129–135.

Rich, P., Blokpoel, M., de Haan, R., and van Rooij, I. (2020). How intractability spans the cognitive and evolutionary levels of explanation. *Topics in Cognitive Science*, 12:1382–1402.

Shatz, C. J. (1992). The developing brain. *Scientific American*, 267:60–67.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2:70–82.

Wald, A. (1947). An essentially complete class of admissible decision functions. *The Annals of Mathematical Statistics*, pages 549–555.