Introduction to Neural Networks

Chapter 1 *of* DEEP LEARNING, Part I

by

Rita Singh *and* Bhiksha Raj

DEEP LEARNING

FIRST EDITION

PUBLISHED INDEPENDENTLY BY BHIKSHA RAJ AND RITA SINGH

Some base artwork licensed from shutterstock.com

First printing, September 2020 ISBN: 978-1-7344465-0-0

Preface

This is Chapter 1 (of Part I) of our book titled DEEP LEARNING.

This book contains nine parts with multiple chapters under each, numbered consecutively throughout the book. The nine parts of DEEP LEARNING are listed below.

Part I	:	Introductory Block
Part II	:	Training
Part III	:	Convolutional Neural Networks
Part IV	:	Recurrent Networks
Part V	:	What a Network Learns
Part VI	:	Networks as Generators
Part VII	:	Hopfield Networks and Boltzmann Machines
Part VIII	:	Reinforcement Learning
Part IX	:	Applications in AI

Deep Learning is a rapidly evolving field, and it is our intention to update the parts and chapters of this book synchronously with the developments in the field. As such, the content of this chapter may be periodically updated, and (rarely) new parts or chapters may be added to this book's milieu.This book will also be paired with a supplementary volume called DEEP LEARNING: EXTENSIONS AND NEW PARADIGMS, which will contain code, exercises and new ideas (tested or proposed) for projects and research.

The style of this book is highly pictorial, and is designed so to make it accessible to readers from a wide range of backgrounds. If you, as a reader, have comments, spot errors or find any part of it lacking in some aspect, please feel free to email us at bhiksha@gmail.com or rita.singh@gmail.com.

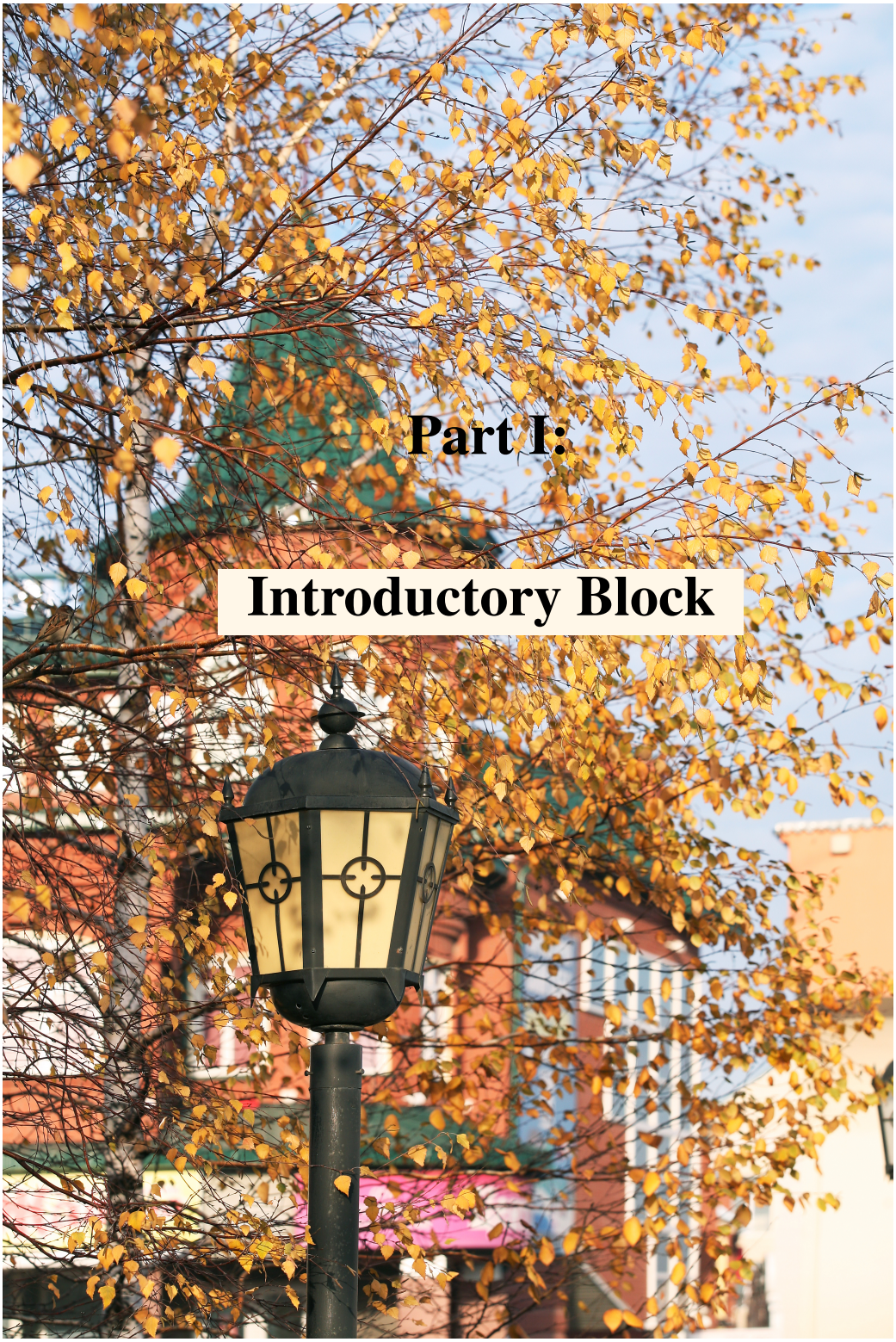
We hope you enjoy this book, and are able to learn easily from it.

Pittsburgh, Pennsylvania.

Bhiksha Raj and Rita Singh
July 2022

Contents

List of Figures



Part I:

Introductory Block

Chapter 1

Introduction to neural networks

The use of neural networks in artificial intelligence tasks has become so ubiquitous that the terms "AI" and "neural networks" have almost become synonymous. They are used in multiple fields of science, within which they have been applied with high degree of success to various pattern recognition, prediction and analysis problems. In many of these, they have established the state of the art, often beating benchmarks set using earlier machine learning techniques by large margins. In some cases they have even managed to solve problems that were difficult or impossible to solve using earlier methods.

1.1 Neural network applications: a glimpse

Chronologically, the greatest success stories of neural networks are associated with fields that deal with difficult pattern recognition tasks – often so difficult that even humans have to spend significant effort in their lifetimes to get them right.

One example is that of **speech recognition**. This is the task of understanding the content of spoken language – a difficult one even for humans. Accurate understanding of the spoken content of speech requires knowledge of the language spoken, clear audition (or correct inference of what was spoken in the case of poor audition), and good generalization skills to parse content in the face of tremendous variability in speakers, speaking styles and the effect of myriad phys-

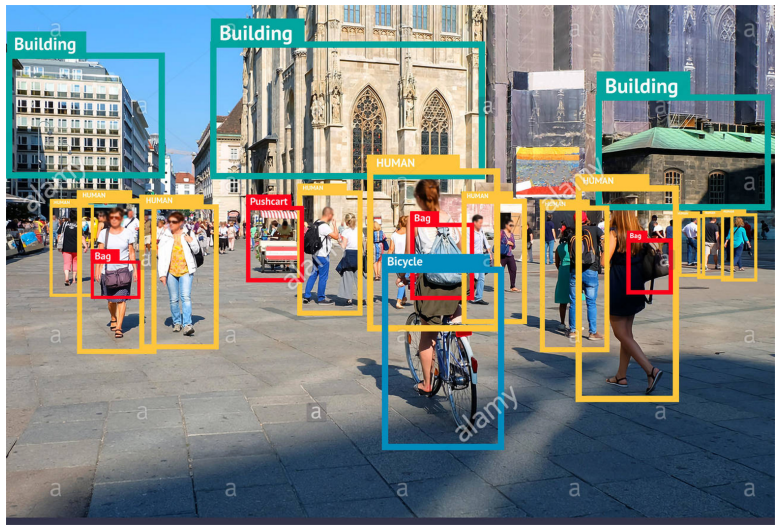
ical, physiological, emotional and other influences on the human voice [1]. It is no surprise that automatic speech recognition (ASR) – the computational equivalent of human speech recognition – was a tremendously challenging task that took decades of concentrated work to create working strategies. The latter were based on concepts from signal processing, statistical machine learning, pattern recognition, psychoacoustics and many other disciplines.

Work on ASR systems began in the early 1970's. After crossing milestone after milestone for nearly 45 years, by 2015, ASR systems still left a lot to be desired. ASR performance had plateaued at levels where a machine could do a good job of speech recognition within low-noise low-reverberation moderately-spontaneous conditions, but a typical large vocabulary speech recognition system made just too many mistakes to be really useful in any but the most constrained tasks, such as careful dictation in a quiet environment.

The use of neural networks for ASR broke this impasse in 2016 [2]. For the first time, automatic speech recognition systems outperformed humans on the classic Switchboard task [3], which comprised recordings of people calling up each other on the phone and talking freely about various topics. Since then the technology has progressed in leaps and bounds, and today neural network based ASR systems generally outperform humans in speech recognition on most ASR benchmarks. Speech (recognition) has currently become a staple user interface.

A second example is that of **automatic language translation**, often loosely called *machine translation* or just "MT". After almost 40 years of research, in 2015 MT systems still performed very poorly. It was standard practice to demonstrate this by translating a sentence in English to some other major language (such as Spanish) using the best machine translation systems of the time (e.g. Google Translate), and then translating it back to English – a practice that usually resulted in an output that read and meant nothing like the original text. Novemeber 2016 saw a remarkable turn of events, when Google Translate's latest version became *significantly* better overnight, resulting from Google's transition to using neural networks for translation. The technology has since continued to improve, and at the time of writing this book, the best machine translation systems are bet-

ter than all but the most expert human translators for most languages. All current machine translation systems use neural networks.



The result of a typical neural network based scene analysis on an image. The girl on the bicycle, the bicycle, and even the bag she is carrying are all correctly detected and segmented.

Figure 1.1: Image segmentation by neural networks

Figure 1.1 shows another impressive example of the effectiveness of neural networks – from the field of **computer vision**. The figure shows an example of image analysis of a rather complicated image. All objects in the image have been successfully recognized, isolated/segmented and tagged. Neural networks have made this level of performance routinely possible in today's world. Similarly, impressive examples of video analysis abound in the world today – scene analyses in videos using neural networks are similarly detailed and impressive. For example, the continual assessment of their visual environment by self-driving cars is currently based on neural networks.

Yet another example is drawn from a task that for centuries was thought to epitomize human intelligence – **playing games** like Chess! For the longest time, it was assumed that the ultimate challenge in AI was to achieve human-like intelligence, and this goal could be realized if machines could play complex thought-based games like *Chess* and *Go*. Until the 1990s, the best computer-played Chess games were little better than human novices. The first Chess game to beat a

grandmaster, called *Deep Thought*, was built at Carnegie Mellon University in 1988. By 1995, computers were able to beat reigning world champions at Chess. However, the game of Go still proved to be a task that machines could not beat a human at. This was because Go is vastly more complex than Chess. While a Chess game has about 10^{120} possible states, Go has 10^{180}. So it was assumed that a computer would never be able to beat a human at Go.

This changed in 2016, when the computer-played version of Go – AlphaGo – beat the human world champion at Go. It was based on neural networks (see Fig. 1.2). Today, the technology has progressed to the point where we have programs that don't even need to know the rules, and can begin learning such games from scratch on their own. In just a few hours of self-learning, a computer algorithm based on neural networks can learn the game of Go well enough to beat human champions!

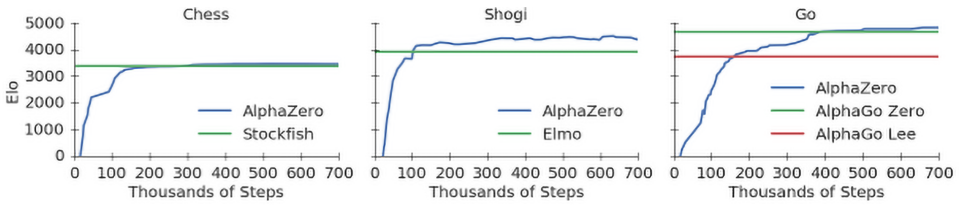


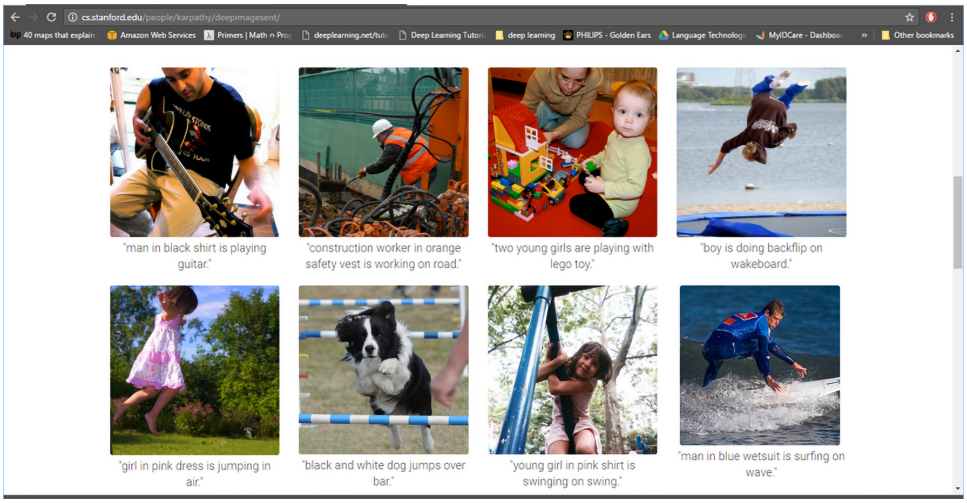
Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (*29*).

The performance of Alphazero (a system that can learn to play games) as a function of the number of "update" steps – effectively the time the game has "practiced," for Chess, Shogi and Go. The vertical axis shows standard ratings for the game. The green lines show the performance of previously world-beating systems on these games. On the plot for Go, the red line shows the performance of the first automatic game that beat the human world champion. Alphazero begins each game with no knowledge of the game, and in just a few hundreds of thousands of steps it learns to beat the best current system, which in all cases is much better than the human world champion.

Figure 1.2: Learning in Alphazero

In addition to these, there came a spate of successes in the ultimate human-like task: that of **creativity**! As humans, we observe, assimilate, analyze and are then able to describe what we observe using language. This is often a one-to-

many mapping. There is no single correct way to achieve the goal of accurate description. Image captioning is a good example of a task that requires such creativity. We may look at a photograph of a scene and be able to instantly describe the scene in such a way that other humans understand what is in the scene, or get a "sense" of it. This too became possible, around 2015, through the use of neural networks. Fig. 1.3 shows an example of automatically generated image captions produced by a neural network, which could well be mistaken for human-produced descriptions.



The result of an image captioning task from 2015: captions such as "Man in black shirt is playing a guitar," "Construction worker in orange safety vest working on road," "Boy doing backflip on wakeboard," etc. are all very human-like in a creative sense.

Figure 1.3: Example of image captioning using neural networks

The use of neural networks in other creative areas has lately seen an explosion – neural nets can generate paintings in the style of the greatest of painters, create videos of people dancing or playing musical instruments in the style of the greatest dancers and instrumentalists (they cannot currently sing with as much authenticity, but that may not be far in the future). In fact neural networks can even hallucinate and dream up entirely new objects and scenarios, just like humans!

Fig. 1.4 shows photographs of people who don't exist, generated by a neural network [4]. The kind of creativity exhibited by a neural network in this case is

not possible with other types of machine learning models. Even humans can't be this good at hallucinating up pictures of people in such intricate detail as is exhibited in the example in Fig. 1.4. Neural nets can in fact paint imaginary faces, draw landscapes, generate music and write stories. All of this has been achieved in just the past four or five years preceding the time of writing this book.

Images of people created by a neural network. These people have never existed in the real world.

Figure 1.4: Images of fictitious people created by a neural network

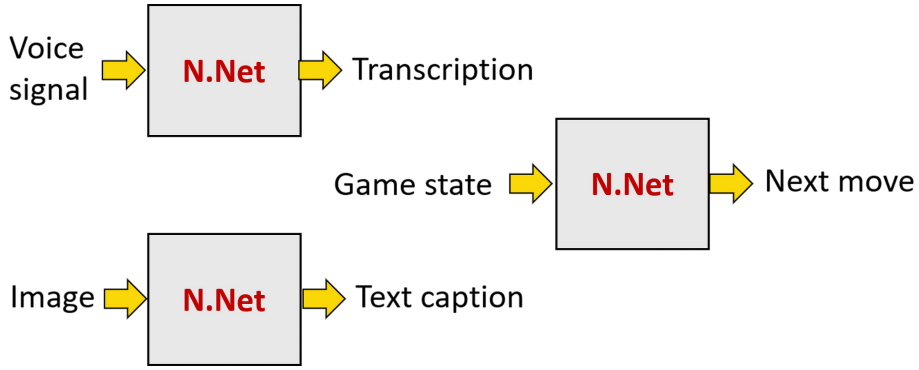
In these, and in many other such problems, ranging from astronomy and health care to even stock prediction, neural networks prevail today. Neural networks represent the best performing models used by investment firms these days. Basically, the belief today is that in due course, neural nets will be able to do any human-like task *better* than a human.

All of this bring us to the question: what *are* neural networks? To understand neural networks, we must understand the thoughts and principles on which they are based. This is best done by (briefly) going through the milestones in the process of discovery that scientists and philosophers went through for centuries,

pondering about human intelligence, and its relationship to the structure of the brain. We do this in the following section.

1.2 Theories that led to the formulation of neural networks

What are neural networks? From a purely operational perspective, they can be thought of as computational black boxes that take in some data as input, and map it to some output. In that respect, like all other machine learning algorithms, they are essentially mathematical functions. This extremely simplistic view of neural networks as black boxes is depicted schematically in Fig. 1.5.



In this example, a neural network is represented as a black box, which takes in data as input and transforms it to some output. When a voice recording is the input, a transcription of it may be the output. When an image is the input, its caption may be the output. A game state may be input, and a recommendation for the next move may be output.

Figure 1.5: Neural networks as functions in blackboxes

To understand the foundational principles of neural networks, let us reconsider the success stories that were outlined earlier in this chapter. Fundamentally, all of the tasks mentioned therein – playing a game, labeling an image, recognizing speech etc. – were human activities, enabled by human intelligence. The foundations of neural network were laid with the surmise that for machines to emulate human intelligence, they must perhaps emulate the biological core of all human intelligence – the human brain.

However, even before this came about, the general nature of human intelligence and (eventually) its connection to the brain had been the subject of deep thought and inquiry for centuries. That humans have a remarkable ability for cognition: to think, to analyze, and to create, defined what it was to be human in an existential sense. The mystery of human intelligence has remained a subject of philosophical thought, presumably from unrecorded times. What is it to be intelligent? What is it to be cognizant? What, indeed is cognition? In 1637 René Descartes summed up the flavor of the time on this subject in his famous statement "Cogito, ergo sum" (Latin; "I think, therefore I am"), although in a slightly different context.

Fun facts 2.1: *Cogito, ergo sum*

Cogito, ergo sum

(Latin; "I think, therefore I am") was a dictum coined by the French philosopher René Descartes in his *Discourse on Method* (1637) as a first step in demonstrating the attainability of certain knowledge. It is the only statement to survive the test of his methodic doubt. The statement is indubitable, as Descartes argued in the second of his six Meditations on First Philosophy (1641), because even if an all-powerful demon were to try to deceive him into thinking that he exists when he does not, he would have to exist in order for the demon to deceive him. Therefore, whenever he thinks, he exists. Furthermore, as he argued in his replies to critics in the second edition (1642) of the Meditations, the statement "I am" (sum) expresses an immediate intuition, not the conclusion of a piece of reasoning (regarding the steps of which he could be deceived), and is thus indubitable.

–Quoted from the **Encyclopedia Brittanica**

1.3 Cognition and models of cognition

Human cognition is a pervasive concept that defines every aspect of existance as an intelligent being. Our ability to learn, to solve complex problems, to recognize patterns, to create new things and new concepts, to think, to cogitate (think and reflect and imagine and dream without external stimulus) and thereby to do all of the above, are all facets of our cognitive abilities. These abilities make us who we are, their unique combination gives each of us our unique identity. As all of this was recognized through the centuries. It stood to reason that if we wanted to build a machine with human-like capabilities, then we must find ways to emulate our cognitive abilities.

Easier said than done, of course! How does the brain perform all of these feats of cognition? How, indeed do we *actually* produce intelligence? There was general consensus that this could be extremely difficult to understand because the problem is that the brain is an immensely complex organ. It has to be. As Marvin Minsky put it [5], if our brain was simple enough to be understood, we wouldn't have the intelligence to understand it!

Questions such as these led to different answers based on different lines of reasoning. All of these collectively fell into the category of *models of cognition*, which we discuss below.

1.3.1 Associationism

The earliest known model for human cognition dates back to at least 400 BC, to within the lifetimes of the Greek philosophers Plato and Aristotle.

This model, known as "Associationism," is ascribed to Plato, and is based on the hypothesis that the process of human cognition is fundamentally that of learning *associations*. Associations are literally what the word itself refers to – relationships – that are perceived between *percepts*. Percepts, in turn, are objects,

feelings and all other entities (real or imagined) that are perceived by a human.

In forming an association, we link, or associate, one percept with another. For example, in our experience we find that lightning is generally followed by thunder. So we make a causal and temporal association between these percepts (in this case, they are events). Thus we expect that lightning occurs when thunder does, and that when we see a bolt of lightning, we will hear the sound of thunder shortly thereafter. The theory of associationism stipulates that we use such associations to also make *inferences*, based on which we do other intelligent tasks, such as making decisions. Thus if we only hear thunder, we *infer* that lightning must have occured or struck somewhere nearby, out of our sight.

Fun facts 3.1: Aristotle on Associationism

In 360 BC, the Greek philosopher Aristotle (thought to be the first of the world's known Associationist thinkers) put forth in his *De Memoria et Reminiscentia*, the postulate that we memorize and rationalize through association: *"Hence, too, it is that we hunt through the mental train, excogitating from the present or some other, and from similar or contrary or coadjacent. Through this process reminiscence takes place. For the movements are, in these cases, sometimes at the same time, sometimes parts of the same whole, so that the subsequent movement is already more than half accomplished."*

[6]

The main ideas stating the basic philosophy of associationism were:

- Pairs of thoughts become associated based on the organism's past experience.
- Learning is a mental process that forms associations between temporally related phenomena.

Associationism was, in fact, the primary model for human cognition until the early 20^{th} century, and scientists such as David Hume and Ivan Pavlov developed theories around it that continue to influence multiple areas of behavioral psychology to this day.

Fun facts 3.2: Pavlovian conditioning

In behavioral psychology, the term Pavlovian conditioning is used to refer to a famous accidental discovery in the 1880s by the Russian physiologist, Ivan Pavlov. At the time Pavlov was investigating the causes for salivation in dogs. He expected that dogs would salivate when food was placed before them, and he measured their saliva levels to confirm this. However, to his surprise he discovered instead that the saliva levels increased even *before* food was placed before his dogs – in fact they began to salivate as soon as they heard the footsteps of his assistant who brought them food! Pavlov's reasoning for this was based on associationism: that dogs responded with increased salivation to any object or person they associated with food. Pavlov devoted the rest of his life to studying this new associationist model of learning.

This led to the notion of *Pavlovian Conditioning*, a theory he put forward in 1902 based on the conjecture that some things need not be explicitly learned in order to factor into an associationist model. For example, a dog doesn't explicitly learn to salivate when it sees food, or something associated with food, in fact the salivation is a reflex that is intrinsic to the dog, and the associations merely "condition" the reflex. This was also an important addition to the notion of "intelligence," conjecturing that not only are inferences made consciously through associationist learning, but also that subconscious reactions could be based on associations. The latter were called "conditioned responses" by him – and are now widely known as "Pavlovian responses"

Fun facts 3.2 continued.....

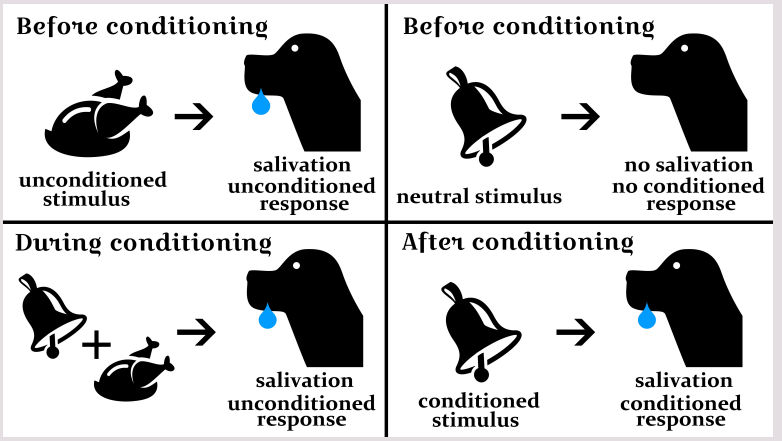


Figure 1.6: The mechanics of the Pavlov's conditioned response in psychology

With the widespread acceptance of the models of associationism as those that could explain human cognition, or cogitation, one question began to haunt the scientists and thinkers of the times. As it became an undisputed fact that the brain was the physical seat of cognition, they wondered: *how and where are these associations stored in the human brain?*

There were several, sometimes strange, theories on how the brain actually stored these associations. But the confusions within these began to clear up as more began to be understood about the structure of the brain itself. With significant advances in optics and improvements in microscopy starting from the sixteenth century, it had become known by the late 1800s that the brain was a large network of special kinds of cells, called *neurons*.

Fun facts 3.3: Cognition and the neuro-physiology of the brain

Figure 1.7: David Hartley: Engraving by William Blake, 1791.

In 1749, David Hartley (1705-57) wrote the treatise *Observations on Man, his Frame, his Duty, and his Expectations*. It contained his thoughts on the interrelations between neurology, moral psychology, and spirituality, a unique presentation that gathered much scientific and social attention in Europe and Britain. It ascribed consciousness and cognition to the neuro-physiology of the brain, a revolutionary proposition at the time. It dwelled on associationism specifically as a physiological process that generates "ideas", followed by the psychological processes by which associations between different percepts are either formed or erased. The work attempted (for the first time) to explain the concepts of learning and the performance of skilled actions such as speech and language, process of scientific inquiry etc. on this basis. Interestingly, the work was also transformative to the spiritual sectors of the society in those times. From a spiritual perspective, this same treatise advocated universal salvation, and argued that all humans would eventually be "partakers of the divine nature."

Fun facts 3.3 continued.....

This formed the basis for his new model for psychological growth that explained how the self both forms (sympathy) and transforms (theopathy), from loving oneself, others and eventually, God. The treatise was remarkable in that it maintained connections between all the different aspects of human existence that it touched upon.

Fun facts 3.4: The theory of Vibratiuncles

Also in David Hartley's treatise *Observations on Man, his Frame, his Duty, and his Expectations* appeared the "The theory of Vibratiuncles." It postulated that we receive input through vibrations, and those are transferred to the brain, that memories could also be small vibrations (called vibratiuncles) in the same regions and that our brain represents compound or connected ideas by connecting our memories with our current senses.

Fun facts 3.5: Naming the neuron

Prior to 1873, it was only known that the brain was a messy network of cellular projections, with no clear structure apparent [7]. It was against this backdrop that Bain came up with his astonishingly prescient models. It was only in 1873 that Camillo Golgi came up with his new staining techniques [8] that enabled people to visually isolate individual neurons and trace their connections. The neuron itself wasn't named "neuron" until 1891, and the axon wasn't named until 1896, although dendrites had already been named by 1889.

1.3.2 Connectionism

With Hartley's work, especially, the brain became the center of scientific attention for unraveling the mysteries of human cognition. The observation that neurons in the brain were all connected to one another and that the brain was essentailly a mass of interconnected neurons triggered great interest. What was so special about such a mass of neurons that it almost magically gave rise to human intelligence? The first person to postulate about the significance of such a conglomerate was Alexander Bain, a polymath of his times. In his book "Mind and Body" written in 1873, Bain put forward the hypothesis that the *connections* were key to the mystery of cognition. His postulate was that the brain stores all of its information, regarding associations between percepts, or more generally between stimulus and responses, *through the structure of the connections* between neurons. This was, in fact, the first *connectionist model*.

Fun facts 3.6

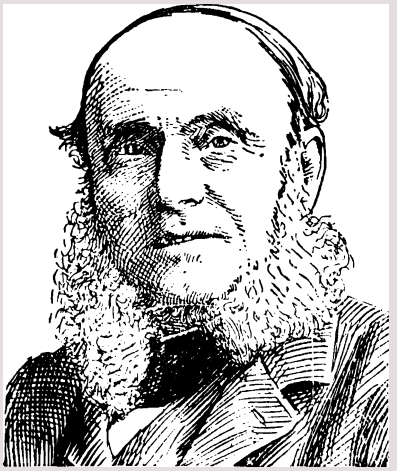


Figure 1.8: Alexander Bain (1818-1903), vintage line drawing.

Alexander Bain was a Scottish philosopher, he was the leading figure in establishing and applying the scientific method to psychology. Like all the scientists of the day back then, Bain was a bit of everything. He was a philosopher, a psychologist, a mathematician, a logician, a linguist. He was also a university professor.

Before we discuss Bain's ideas about the significance of neuronal groupings further, let us briefly recall what is known today about neurons, and their role in the human brain.

1.3.2.1 The human brain as a conglomerate of interconnected neurons

The human brain can be viewed as a large network of interconnected neurons. Each neuron has many neurons connecting *into* it, and each in turn connects out to many neurons. The structure of each neuron is such that impulses (in the form of electrical potentials) can be received by a neuron through incoming connections to its main cell body, and the impulses generated by it can be propagated

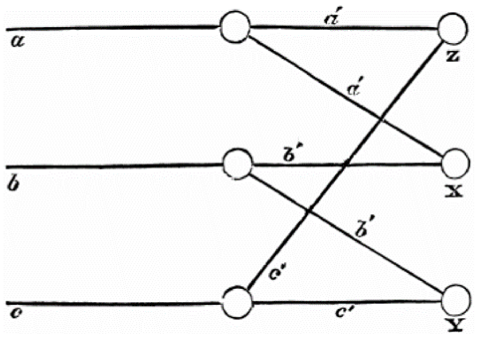
to other neurons through outgoing connections. We will revisit the structure of individual neurons shortly.

1.3.2.2 Bain's first idea: Neural groupings

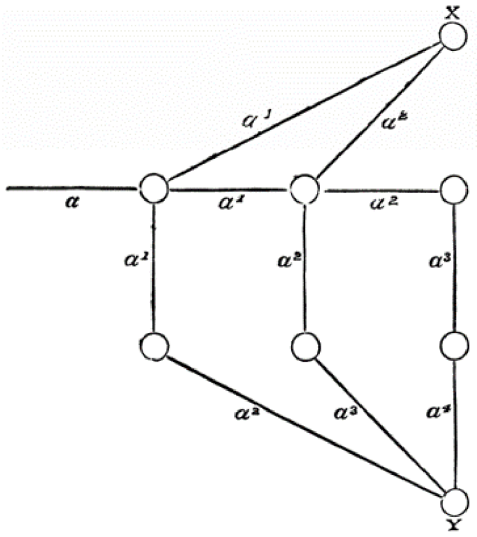
Bain was the first person to translate his connectionist theory – that cognition-relevant information was stored within neuronal connections – into computational models. The key idea that was carried over from his theory of connectionism into his computational models was that within groups of neurons, *how* they were connected could make a difference as to *what* their net output was. In other words, the same connections, if appropriately arranged, could result in different outputs for different inputs.

This may seem obvious to us today, but in Bain's time this was revolutionary. The idea is best understood with the help of Fig. 1.9 [9]. The figure shows a computational circuit that encapsulates his idea: in this figure, the circles depict neurons, and the lines depict the connections between them. Neurons excite and stimulate each other. Depending on which input neurons fire, the output neuron that fires is different. Thus although there is a single circuit in the figure, for different input firing patterns, we obtain different firing patterns for the output.

Bain proposed several such models that demonstrated his ideas about the functional aspects of neuronal groupings. The set included models where the output of a neuronal circuit could depend on just the *intensity* of the input. For instance, in Fig. 1.9, if the input is of low intensity, only y could be set to fire, but if it is of high intensity, both x and y could be set to fire. These represent modern neural network models, except for the fact that Bain proposed them almost 150 years ago! Neural networks have thus been in existence for almost 150 years at the time of writing this book. Bain not only showed the circuit-equivalent structures for neuronal groupings, but also proposed a *learning mechanism*, whereby the such groupings (including the brain) – or any computational model of them – could *learn* these connections.



(a) In this computational model, a,b, and c are the inputs, and x, y, z are the outputs. All inputs and outputs are assumed to be binary. It takes the combined output of two neurons in the input, for an output neuron to fire. Due to the nature of the connections shown, different combinations of inputs can result in different outputs. It is easy to see that if a and b fire, x fires; if a and c fire, z fires; and If b and c fire, y fires.



(b) In this computational model, Different intensities of activation of a lead to the differences in when x and y are activated. Low intensities of a only excite y while higher intensities excite both x and y.

Figure 1.9: Examples of Bain's connectionist models for neurons

1.3.2.3 Bain's second idea: Creating memories

Bain's proposed mechanism for how neuronal connections could be learned, in his own words, was this:

"When two impressions concur, or closely succeed one another, the nerve-currents find some bridge or place of continuity, better or worse, according to the abundance of nerve-matter available for the transition."

According to this, the finding/formation of a connection was postulated to happen when there was an occurrence of concurrent impulses or stimuli. This postulate effectively predicted *Hebbian learning* (explained later in this chapter) three quarters of a century before Donald Hebb!

Fun facts 3.7: Bain's doubts

Sadly, Bain's peers in the 19^{th} century thought he was speaking nonsense, and eventually even he began doubting himself. To quote Bertrand Russell: *"The fundamental cause of the trouble is that in the modern world the stupid are cocksure while the intelligent are full of doubt."*

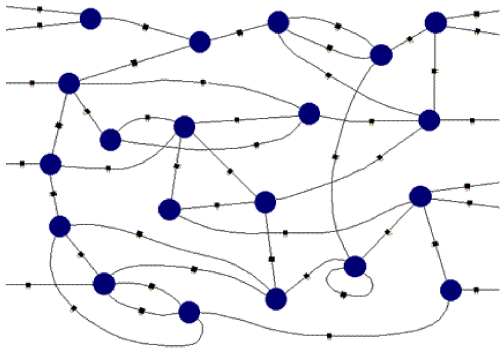
.

In 1873 Bain postulated that there must be one million neurons and five billion connections relating to 200,000 "acquisitions" that capture 200,000 percepts. These accounted for memory, recall and learning. This of course was a very large number when he boldly suggested it, but then by 1883 he realized that it had to be larger because he had not taken into account the number of "partially-formed associations" between neurons, to account for which he would have to add more neurons and connections to his postulate. His hesitation to extend his numbers stemmed from the skepticism of his peers at the time. By 1903, toward the end of his life, he decided that the human brain was simply not large enough to hold the vast number of neurons and connections needed for his model to work, and therefore his postulates about neurons and their significance in the brain was wrong, and recanted all his ideas. In doing so, of course we know today that he was wrong. All of his original postulates had been correct – he was largely correct in 1873 and until the time he recanted his postulates. The human brain, as is known for a fact today, has about 80 billion neurons and about one hundred trillion connections – *more* than the largest numbers that Bain guessed was required for his model to work! Bain's model is in fact more or less how we think our brain works today.

1.3.2.4 Modern connectionist machines

As a model for cognition, connectionism survives to this day. It is in fact a widely accepted fact that a brain is a connectionist machine, as predicted by Bain in 1873 and Ferrier in 1876 [11]. In it, neurons connect to other neurons, and the processing and information capacity of the brain is a function of these connections. Modern artificial neural networks emulate this architecture, and are in fact connectionist machines.

A early model of a connectionist machine, proposed by Alan Turing, is shown in Fig. 1.10. It comprises a network of processing elements. All world knowledge about the task(s) that the machine performs is stored in the connections between the elements.



All world knowledge about the task(s) that this machine performs is stored in the connections between the elements.

Figure 1.10: Turing's connectionist model

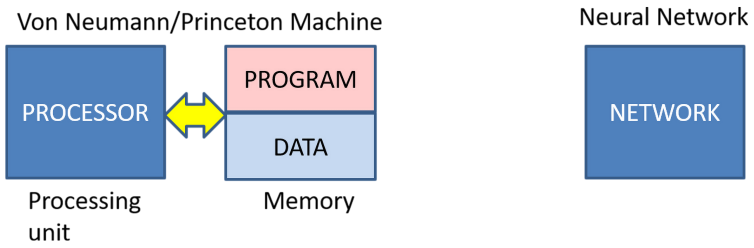
1.3.2.5 The difference between a connectionist machine and a modern computer

In this context, it is important to note the difference between a modern computer, and a connectionist machine: A standard modern computer's architecture is a variant of what is known as the *Von Neumann architecture* or the *Princeton architecture*. In such an architecture, there is a processing unit, and a separate

memory where programs and data are stored. A further variant of this is the *Harvard architecture* which comprises separate memories for data and programs. In most cases, modern computing architectures use a hybrid approach: the processor core uses a Harvard architecture, while the main memory has a Princeton architecture.

In the context of this chapter though, these differences are less important to us than the *form* of the architecture itself. In a modern computer, the processor is separated from the memory which holds the programs and data. Thus, to execute a new program, all that needs to be done is to upload the new program into the memory, and the processor can run it. This makes modern computers versatile, in that a typical computer, regardless of its size, is usually able to do so many different things. To make it perform a new task, we simply load the corresponding program into the memory and the computer is able to run it.

That is not the case for connectionist machines. For connectionist machines, the computer *is the program*, because the program and its memory are *all* encoded in the connections. To change the program, one must change the machine itself. For this reason, we usually emulate the connectionist machine on regular Harvard and Princeton architecture machines. The difference between modern computing architectures and a connectionist machine is highlighted in Fig. 1.11



The connectionist machine has many non-linear processing units The program is the connections between these units. Connections may also define memory.

Figure 1.11: The difference between a modern computer's architecture and a connectionist machine

Recap 3.1

1. In current times, neural network based AI has taken over most AI tasks
2. Neural networks originally began as computational models of the brain, or more generally, models of cognition
3. The earliest model of cognition was Associationism
4. Associationism was followed by Connectionism
5. The more recent model of the brain is connectionist: Neurons connect to neurons and the workings of the brain are encoded in these connections
6. Current neural network models are connectionist machines

Fun facts 3.8: Turing's connectionist paradigms

The view that a connectionist machine comprises a network of processing elements and all world-knowledge is assumed to be stored in the connections between the elements, formed the basis of multiple connectionist paradigms. One of the earliest of these architectures was proposed by Alan Turing [10]. Turing's "unorganized" machines consisted of randomly connected networks of NAND gates. **A-type** machines were those with straightforward connections. The operation of such networks was random. **B-type** machines (shown in Fig. 1.10) were modified A-type machines where the connection between two units had a "modifier" as shown in the figure below, which gates the connections. In the example shown, if the green line is on, the signal sails through; if the red is on, the output is fixed to 1. The modifiers are themselves composed of A-type units. "Learning" comprises determining how to manipulate the colored wires to elicit a desired response from the network.

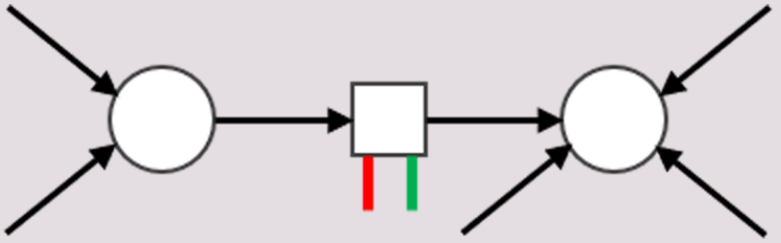


Figure 1.12: Example of a B-type machine

Fun facts 3.8 continued.....

A second connectionist paradigm is the *Parallel Distributed Processing*, or the PDP paradigm. This was put forward in 1986 by Rumelhart, Hinton, McClelland and Smolensky [15, 12, 13, 14] and fulfills the following requirements [17, 16]:

- A set of processing units
- A state of activation for each unit
- An output function for each unit
- A pattern of connectivity among units
- A propagation rule for propagating patterns of activities through the network of connectivities
- An activation rule for combining the inputs impinging on a unit with the current state of that unit to produce a new level of activation for the unit
- A learning rule whereby patterns of connectivity are modified by experience
- An environment within which the system must operate
- A mechanism whereby the network is interpreted semantically

1.3.3 Modern connectionism

Modern connectionist models began with a close emulation of what became known about the biological workings of the neurons and the brain as a whole. Foremost in this was the structure of a single neuron: the individual processing element of the brain.

Fun facts 3.9

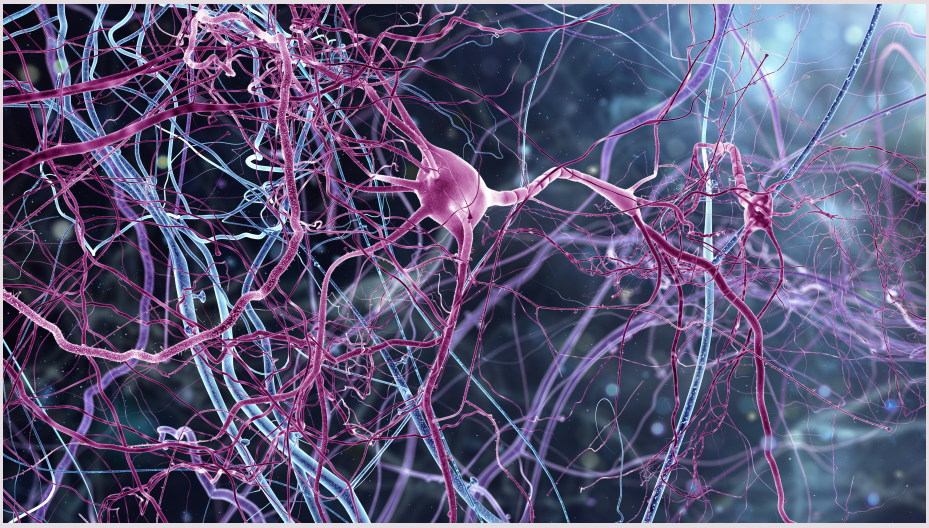


Figure 1.13: Closeup of neurons in the human brain.

In biology, a neuron is a cell that is typically found in the brain of creatures that have a brain – namely animals – or members of the taxonomic kingdom Animalia. Bacteria, viruses, plants etc. are of course not animals and do not have brains.

A neuron has a specific structure. Each neuron has a main cell body, called the *soma*, within which the nucleus resides. The soma has several smaller projections called *dendrites*, through which it receives signals from other neurons. The neuron also has a long, stem-like projection from the soma called an *axon*, which terminates in a set of branches called *telodendria*. The telodendria of one neuron connect to the dendrites of another by physical proximity achieved through a *synapse*, which is the junction between the two that has specific chemical properties that allows a chemical connection to be formed between them.

Fun facts 3.9 continued.....

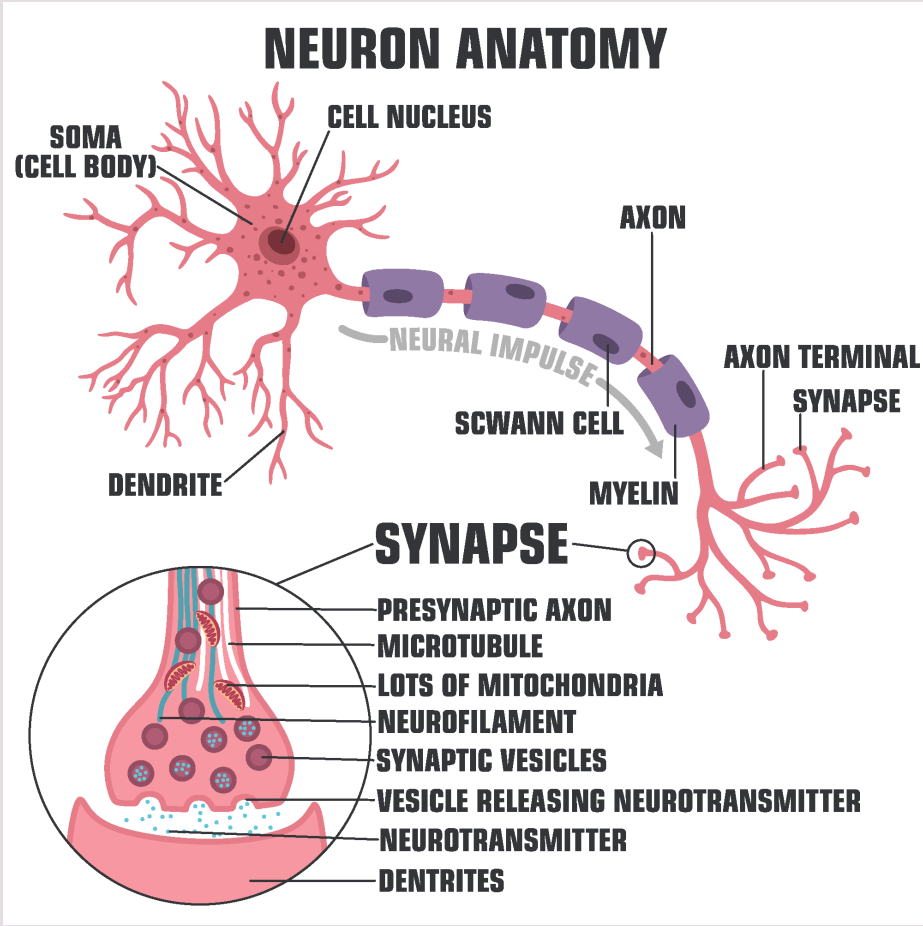


Figure 1.14: The anatomy of a neuron.

The soma receives signals from other neurons through its dendrites. The neuron generates signals that are spike trains. The rate of which these spike trains are generated depends on the total strength of the input signals received by the neuron through its dendrites (from other neurons). The signal then travels down the axon to other neurons. At a finer level, this activity may be modeled through a Boolean analog: if the total strength of incoming signals to the neuron exceeds a threshold, the neuron *fires*, or releases a signal of its own that is communicated to other neurons it is connected to.

Fun facts 3.10: Neural factoids

Since the axon is a relatively long structure, it is protected by a sheath of fat, called the myelin sheath. This fat is crucial for the functioning of the brain. In fact scientists have found that the key distinction between Einstein's brain and those of people of "normal" intelligence was that he had more *glial cells* – the cells that produce myelin – than normal people.

Neurons do not undergo cell division. They are created from neural stem cells, and neurogenesis reduces to almost zero shortly after birth. In other words, we are born with almost all the brain we will ever have.

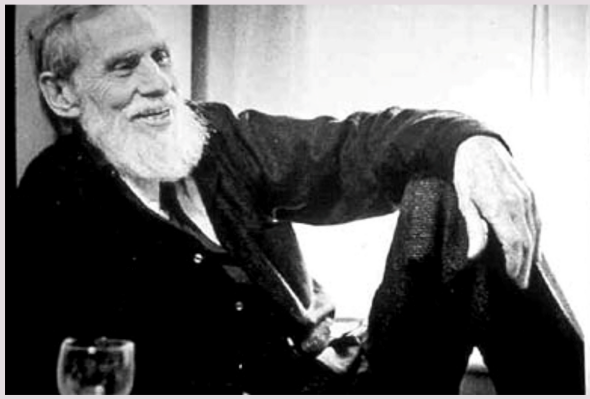
1.4 Computational models of the brain

Computational models of the brain (and neorons within it) use computing units that emulate the operations of biological neurons.

1.4.1 The McCulloch-Pitts model

The first real computational model for the neuron was proposed by Warren McCulloch and Walter Pitts. Fig. 1.16 is a representation of their model.

Fun facts 4.1



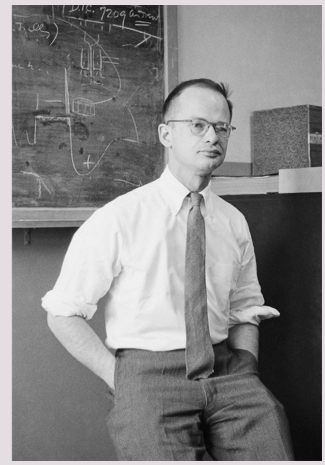
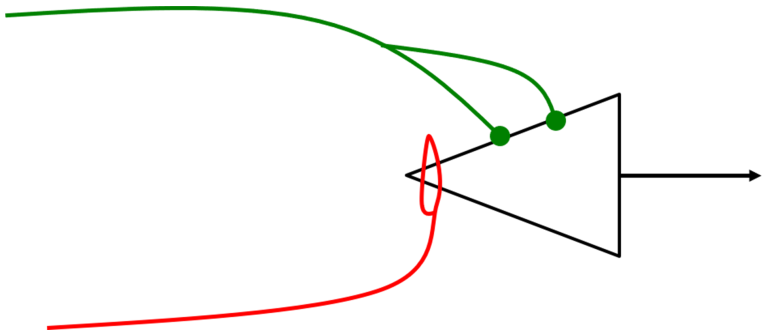


Figure 1.15: Warren McCulloch (left) and Walter Pitts (right).

Warren Sturgis McCulloch (November 16, 1898 – September 24, 1969) was an American neurophysiologist at the University of Chicago. Walter Harry Pitts, Jr. (23 April 1923 – 14 May 1969) was a logician who worked in the field of computational neuroscience. At the time he met McCulloch, he was a homeless teen-aged autodidact who corresponded with various luminaries such as Bertrand Russel on topics related to mathematical logic. He took up residence with McCulloch, at the latter's invitation, and together, they created computational models of the brain that used threshold logic. The first artificial neuron as a Threshold Logic Unit (TLU), or Linear Threshold Unit, was proposed by them in 1943 [18]. Pitts was only 20 at the time they proposed this model.

In Fig. 1.16, neurons connect to each other through synaptic connections. There are two types of synapses: *Excitatory synapses* and *Inhibitory synapses*. The excitatory synapse transmits weighted inputs into the neuron. The number of connections represents the weight. Any signal on an inhibitory synapse absolutely prevents the neuron from firing, regardless of what else is input to it. If the inhibitory input is turned on, the output of the neuron will always be turned off.



This figure shows the working of a single neuron as per the McCulloch-Pitts model. The actual neuron is shown by the triangle. The line going out represents the outgoing signal. The colored lines represent incoming signals from other neurons, which connect to this neuron through synaptic connections. The green lines connect through an *Excitatory synapse*, and transmit a weighted input to the neuron. The red line denotes an *Inhibitory synapse*. Any signal from an inhibitory synapse prevents the neuron from firing, regardless of other inputs.

Figure 1.16: The McCulloch and Pitts mathematical model of a neuron

An important point to note is that by appropriate choice of excitatory and inhibitory synapses, the McCulloch and Pitts neuron can be made to emulate *any Boolean gate*! Examples are shown in Fig. 1.17. In each instance shown in this figure, one or two "input" neurons connect to an "output" neuron through either one or two connections. In each case, the output neuron fires if the total input across all of its incoming connections is at least 2. Through different arrangements of the connections, as shown in the figure, different Boolean gates can be realized.

With these basic gates, it becomes possible to construct *any* Boolean circuit using this model. The McCulloch-Pitts model can in fact compute arbitrary Boolean propositions. Since any Boolean function can be emulated by it, any Boolean function can be composed.

With their model, McCullough and Pitts also showed that it was possible to produce fairly complex responses. An example is shown by the circuit in Fig. 1.18.

The circuit shown in Fig. 1.18 models temperature response. The input consists

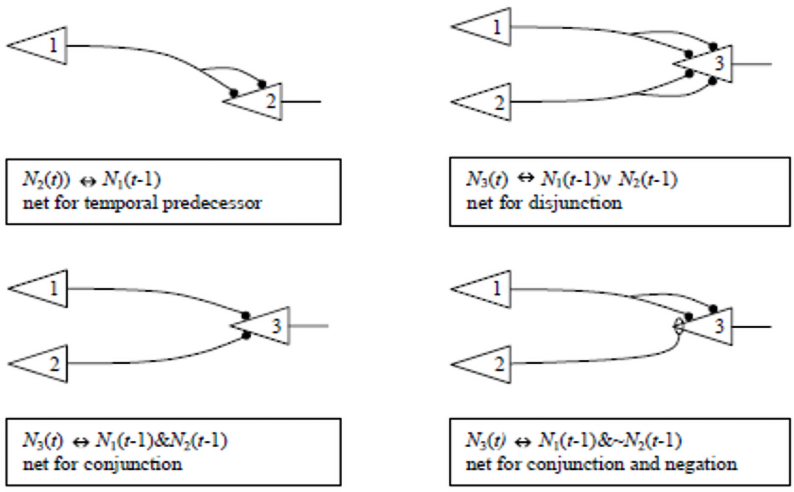


Figure 1. Diagrams of McCulloch and Pitts nets. In order to send an output pulse, each neuron must receive two excitory inputs and no inhibitory inputs. Lines ending in a dot represent excitatory connections; lines ending in a hoop represent inhibitory connections.

This figure shows how simple networks of neurons can perform Boolean operations. In each case the neuron fires if the total input is at least 2. **Top left:** If 1 fires, 2 subsequently receives two inputs through the two synapses and fires. This is a delay. **Top right:** if either 1 or 2 fire, 3 receives two inputs and fires. The output is 1 OR 2. **Bottom left:** If and only if 1 and 2 both fire does 3 receive two inputs and fire. The output is thus 1 AND 2. **Bottom right:** If 1 fires, 3 receives 2 inputs, but 2 is connected to 3 through an inhibitory synapse, so if 2 fires, 3 won't fire regardless of the status of 1. This output is 1 AND NOT 2.

Figure 1.17: Boolean operations using the McCulloch-Pitts model

of two neurons, of which the first (1) detects a heat signal and the second (2) detects cold. The two output neurons represent *sensations* of heat (3) and cold (4) respectively. There is also a chain of two intermediate neurons, the first of which directly receives input from the cold signal detector, and the second which receives excitatory inputs from the first intermediate sensor and an inhibitory input directly from the cold signal detector. Each connecting line between neurons corresponds to a delay of one unit of time.

Given cold and hot signals as input, if the cold signal persists for two time units, at t=1 the first intermediate neuron fires in response to the cold signal at t=0, and at t=2, the responses from both the first intermediate neuron and the cold signal

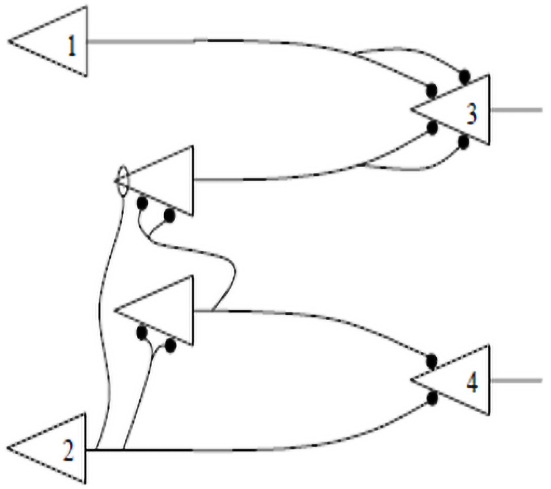


Figure 2. Net explaining the heat illusion. Neuron 3 (heat sensation) fires if and only if it receives two inputs, represented by the lines terminating on its body. This happens when either neuron 1 (heat reception) fires or neuron 2 (cold reception) fires once and then immediately stops firing. When neuron 2 fires twice in a row, the intermediate (unnumbered) neurons excite neuron 4 rather than neuron 3, generating a sensation of cold.

This figure shows how the McCollough-Pitts model can induce a complex response: the illusion of "perception."

Figure 1.18: An example of a complex response using the McCollough-Pitts model: percepts and inhibition in action.

detector, both arrive at neuron 4 (the cold sensor), which fires, giving a sensation of cold. Thus 4 (the cold sensor) only senses cold if the cold signal persists for at least 2 time units.

Now consider the upper intermediate neuron. As long as there is a cold signal at the cold receptor, the upper intermediate neuron is inhibited and does not fire. Thus, in the presence of extended code signals it does not influence heat sensation. However, if there were only a brief cold impulse at t=0, 4 would not fire. Instead, in this case, at t=1 the first intermediate neuron fires, and at t=2 the upper intermediate neuron is not inhibited (since it receives no inhibitory input from the cold signal), and does fire in response to the inputs from the lower intermediate

neuron. This in turn causes the heat sensor to fire, triggering a sensation of heat. Of course, whenever the heat receptor fires, 3 does fire, giving a sensation of heat as well. Thus, Fig. 1.18 represents a model where a brief cold signal triggers a sensation of heat, but an extended cold signal triggers a sensation of cold.

This gives a synthetic illusion of known characteristics of temperature "perception" on the part of the model.

1.4.1.1 Criticisms

Although the McCollough-Pitts model was quite powerful, it faced a lot of criticism. McCullough and Pitts claimed that their networks could compute a small class of functions, which was true. They also claimed that if we provided the networks with tapes, they would become Turing machines, and would be able to solve any problem that a Turing machine could solve. This is of course easily seen to be false (these are finite state machines and cannot be Turing-complete). Also they did not actually prove any results themselves; these were just claims.

In addition to this, they did not provide any learning mechanism whereby the network could learn its connections – intensities of weights thereof – from experience, i.e., from data. This was rectified by Donald Hebb, with the introduction of *Hebbian learning*, which we discuss next.

1.4.2 Hebbian Learning

In 1949, Donald Hebb pubished a book called "Organization of Behavior," wherein he outlined a mechanism for neuronal cells in the brain to learn the weights of their connections from experience, i.e., from data. He described the mechanism as follows (quote): *"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."*

To summarize, if neuron A is connected to neuron B, each time that a firing of A is immediately followed by a firing of B, their connection gets stronger. This is often stated as: *Neurons that fire together wire together*.

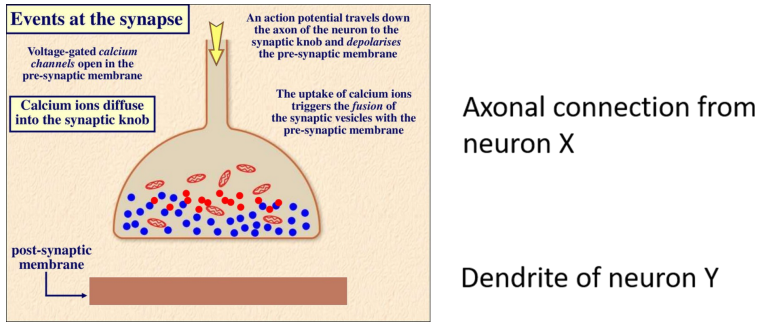
Fun facts 4.2

Figure 1.19: Donald Hebb.

Donald Olding Hebb (July 22, 1904 - August 20, 1985) was an interesting character. He was a Canadian. He initially trained to be a novelist, then wanted to be a farmer, then literally a hobo, subsequently figured he wanted to be a school teacher, and eventually decided that he would become a psychologist. To the latter end he obtained a doctorate degree from Harvard, and then went on to work professionally in a variety of places. He introduced his theory of Hebbian learning in his 1949 publication *The Organization of Behavior* [19]. He is often described as the father of neuro-psychology and neural networks.

Figure 1.20 shows the essential elements of Hebbian learning. In this figure, we focus on the connections between two neurons X and Y.

Neuron X connects to neuron Y at a synapse. At the synapse, the axonal connec-



Hebb's model for how neuronal learning happens at neuronal connections.

Figure 1.20: Hebbian learning

tion from X has a knob, called the synaptic knob, through which it connects to the dendrite of neuron Y. This is not actually a physical connection, but rather a chemically mediated one: the actual signal transfer happens through chemicals called neuro-transmitters that flow from the synaptic knob of X to the dendrite of Y. Each time a signal from X successfully triggers a response from Y, this synaptic knob becomes slightly larger, making it easier for X to trigger Y the next time, effectively increasing the strength of the connection. This can be translated into mathematical form as the *Hebbian learning rule* as follows:

$$w_{xy} = w_{xy} + \eta xy \tag{1.1}$$

where w_{xy} is the weight of the connection between neuron x and neuron y. η is a positive number. In this model, the outputs and inputs to neurons are binary and can take the value 1 or 0, representing whether a neuron is firing or not, respectively. According to this rule, whenever both x and y take the value 1, w_{xy} increases. Note that the term ηxy in this equation is 1 only if both x and y are 1, so w_{xy} increases by η each time both x and y fire together. The Hebbian learning rule of Eq. 1.1 is in fact the basis for many machine learning algorithms today.

However, the Hebbian learning rule as Donald Hebb defined it above had problems: it is fundamentally unstable. The main issues were:

1. The learning is unbounded.

2. Stronger connections will enforce themselves.

3. There is no notion of "competition."

4. There is no reduction in weights. The weights can only increase, and never decrease. Over time, eventually *all* weights can become large, and in that case the entire model would become useless.

Various proposals were made for how to modify it, allowing for weights to be normalized, forgotten, etc. For instance in *Generalized Hebbian learning*, also known as *Sanger's rule*,

$$w_{ij} = w_{ij} + \eta y_i \left(x_i - \sum_{k=1}^{j} w_{ik} y_k \right) \tag{1.2}$$

an input x_i is allowed to contribute to many outputs – basically saying that a neuron can connect out to multiple other neurons, but in order to determine exactly by *how much* the weight between an input and a particular output must be updated (using the Hebbian rule) – we must subtract out the contributions of the input to other outputs. In effect, the contribution of an input is incrementally distributed over multiple outputs.

However, the best and the most lasting update to the Hebbian rule came with the introduction of a new model: the *Perceptron*.

1.4.3 The perceptron

In 1958, a better model, called the perceptron, was proposed by Frank Rosenblatt. At the time he was sure that his model would solve all AI problems.

Fun facts 4.3

Figure 1.21: Frank Rosenblatt, the inventor of the Perceptron.

Frank Rosenblatt (July 11, 1928 – July 11, 1971) was a psychologist and logician. He obtained his Ph.D. in 1956 from Cornell University, and then worked at the Cornell Aeronautical Laboratory in Buffalo, New York, where he conducted his early work on perceptrons. His work resulted in the conceptual development and hardware construction of the Mark I Perceptron in 1960 – the first computer that could learn new skills by trial and error. He died on his 43^{rd} birthday in a boating accident in Chesapeake Bay.

Fun facts 4.3 continued.....

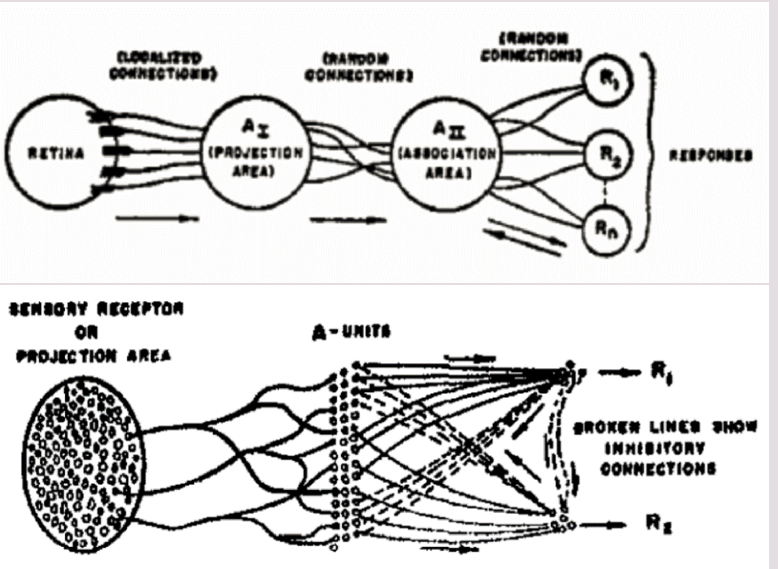


Figure 1.22: Rosenblatt's original model of the perceptron.

Rosenblatt's original model of the perceptron related to the eye. In this model (on the left in the figure above), groups of sensors (S) on the retina combine into the cells in an "association" area A1. Groups of A1 cells combine into cells in an Association area A2. Signals from A2 cells combine into response cells R. All connections may be excitatory or inhibitory. The model even included feedback between A and R cells (right panel in the figure above), which ensures mutually exclusive outputs.

1.4.4 A simplified model of the perceptron

Rosenblatt's original proposal related to perception in the eye, and was a little complicated. Mathematically though, the model simplifies to the one shown in Fig. 1.23. This was called the *Perceptron*.

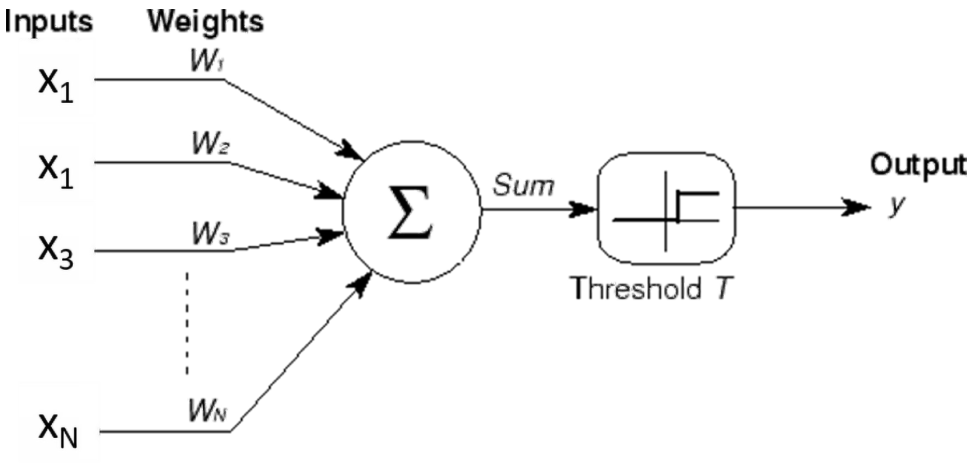


Figure 1.23: The perceptron

The perceptron is shown in Fig. 1.23. It incorporated the following "Threshold logic":

$$Y = \begin{cases} 1 & if \\ \sum_i w_i x_i - T \geq 0 & \\ 0 & else \end{cases} \tag{1.3}$$

In this simple model, each neuron receives a number of inputs $x_1, x_2, \ldots, x_N$. It computes a weighted combination of these inputs, where the weights could be positive or negative. If the weighted combination of inputs equals or exceeds some threshold T, the neuron fires, i.e. its output becomes 1. Otherwise it does not fire, and the output is 0. This is often referred to as *Threshold logic* in computing literature.

Fun facts 4.4: Rosenblatt's belief and media response

Rosenblatt originally assumed that with the appropriate choice of weights, the perceptron model could represent any Boolean function and perform any logic. Roseblatt's assumption was not right, but the media of the time believed it to be true.

"...the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence" – New York Times, July 8, 1958

"Frankenstein Monster Designed by Navy That Thinks" – Tulsa, Oklahoma Times, 1958

Rosenblatt originally assumed that with the appropriate choice of weights, the perceptron model could represent any Boolean function and perform any logic – an assumption that was not true. He also provided a learning algorithm which could be used to learn the weights of the connections to perform any Boolean task:

$$w = w + \eta(d(x) - y(x)) \tag{1.4}$$

In Eq. 1.4 which represents the proposed (sequential) learning algorithm, $d(x)$ is the desired output in response to input x, and $y(x)$ is the actual output of the perceptron. Let w be the weight with which an input x is connected to the perceptron. According to Eq. 1.4, after each new input, the weight is adjusted to account for the difference between the actual output of the perceptron and the desired output of the perceptron.

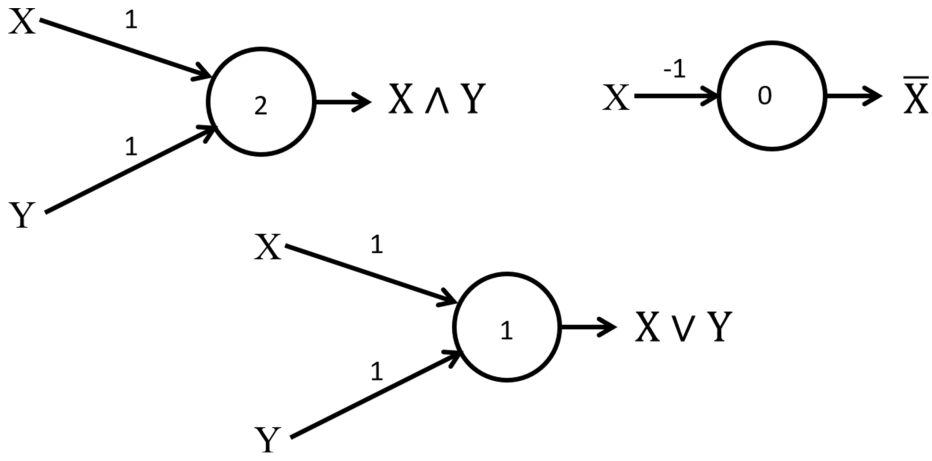
Observe that this is very similar to the Hebbian learning rule, but there is a small but extremely significant difference: while in Hebb's rule the weights are simply updated by the product of the output and the input, here we update them by the product of the input and the *error* between the actual and desired outpus. If the perceptron produces the right output, the weights are not adjusted. They are adjusted only when the output is *not* the desired output, and the adjustment is

proportional to the difference between the two.

In addition to proposing this rule, Rosenblatt also proved its convergence for linearly separable classes. In other words, he showed that if we have linearly separable data, i.e. data such that the instances for which y is desired to be 1 can be separated by a hyperplane from the instances for which y is desired to be 0, then the perceptron learning rule represented by Eq. 1.4 will converge with even a limited amount of training data to learn from.

1.4.5 The Perceptron and Boolean operations

It is relatively easy to see how the perceptron can mimic any Boolean function. Consider the examples shown in Fig. 1.24. In the figures the number shown above each edge is its weight, and the number within the circle is the threshold that must be matched or exceeded for the neuron to fire.



The values shown on the edges are the weights associated with the connection, and the numbers in the circles represent thresholds.

Figure 1.24: Boolean operations using the perceptron

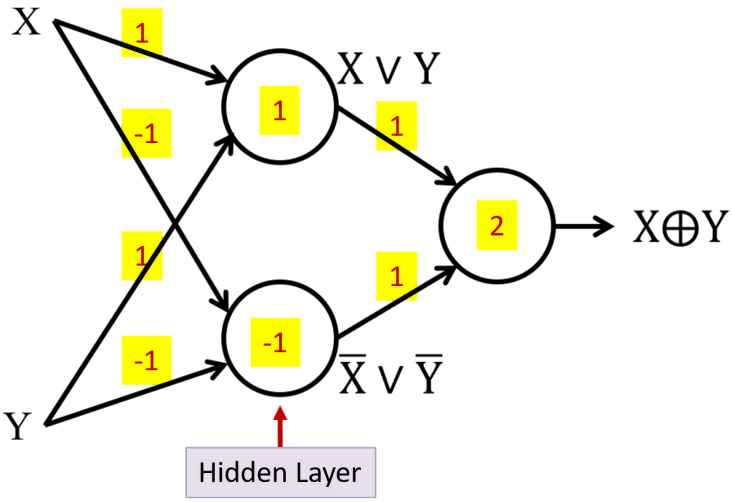
In the panel on the top left, if both x and y are 1, then the total weighted sum of inputs matches the threshold and the perceptron fires. If either of them is 0, the

total input is less than 2 and the unit will not fire. This represents the Boolean AND operation: the output is x AND y. In the panel on the top right, whenever the input x is 0, the total input is 0. This matches the threshold, so the unit will fire. On the other hand if the input x is 1, the total weighted input is -1, which is less than the threshold and the unit will not fire. This perceptron represents a Boolean NOT opearation: y = NOT x. In the bottom panel, the weighted sum of inputs becomes 1 and matches the threshold if either x or y are 1. This models the Boolean operation x OR y.

This example clearly shows that the perceptron can model Boolen functions. It was Rosenblatt's – and general – belief that perceptrons can indeed model all Boolean functions. However, this belief was shattered when Minsky and Papert proved that there is no setting of the weights of a perceptron for which it could emulate the Boolean XOR operation, showing in fact that the perceptron is *not* a universal Boolean machine [20]. This discovery caused so much dismay that research funding into perceptrons and neural networks dried up for many years.

How could this impasse be overcome? For this, scientists had to reconsider the brain. The problem really lay in the fact that a perceptron focused on single neurons at a time. A brain is not just one single neuron – it is in fact a network of neurons. It was realized that for better computational models, although individual perceptrons are rather weak computational elements, one could greatly extend the scope of what could be done with them by networking many of them (also originally suggested by Minsky and Papert in 1969).

In fact, with a network of just three perceptrons, it becomes possible to do an XOR operation, as shown in Fig. 1.25. In this figure, three perceptrons are connected in a layered manner. There are two perceptrons in the first layer. These directly operate on the inputs. The third perceptron in the network operates on the outputs of the first two perceptrons. The first perceptron computes X OR Y. The second perceptron computes (NOT X) OR (NOT Y). The final perceptron computes the AND of the output of the first two perceptrons, to result in X XOR Y.



The first layer has two perceptrons and is a "hidden" layer.

Figure 1.25: The Boolean XOR operation with three perceptrons

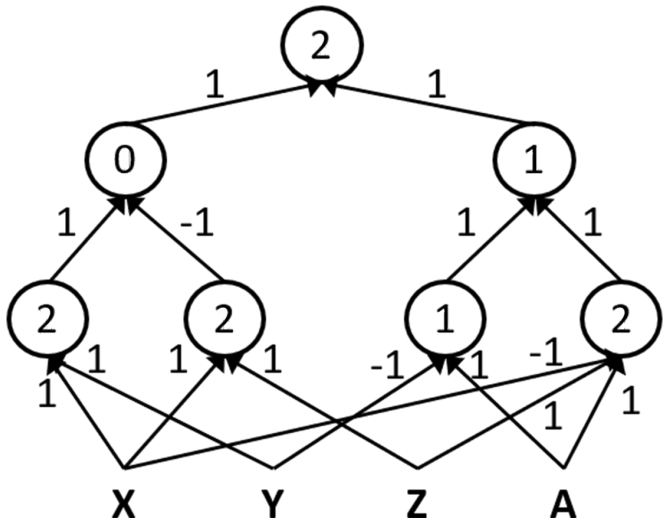
1.4.6 The multi-layer perceptron

Fig. 1.25 in fact represents a perceptron with multiple layers. The first layer has two perceptrons in this example, and is called a *hidden layer*, since their output is not directly observed, but is fed into the third neuron. The third neuron is called the output neuron. The observed output of the network is that of this third neuron. This entire layered network of perceptrons is an example of a *multi-layer perceptron*.

The multi-layer perceptron is in fact a generic model that refers to any layered network of neurons. Networking of neurons into multi-layer perceptrons allowed the computation of complex Boolean functions. In fact, arbitrarily complex Boolean functions can be computed by simply connecting multiple perceptrons in the right way. In cognitive terms, we can thus compute arbitrary Boolean functions of sensory input! An example is shown in Fig. 1.26.

A network of perceptrons can indeed be a universal model for Boolean functions.

$$((A\&\bar{X}\&Z)|(A\&\bar{Y}))\&((X\ \&\ Y)|\overline{(X\&Z)})$$



This network operates on four inputs: X, Y, Z and A, and computes the function depicted in the figure.

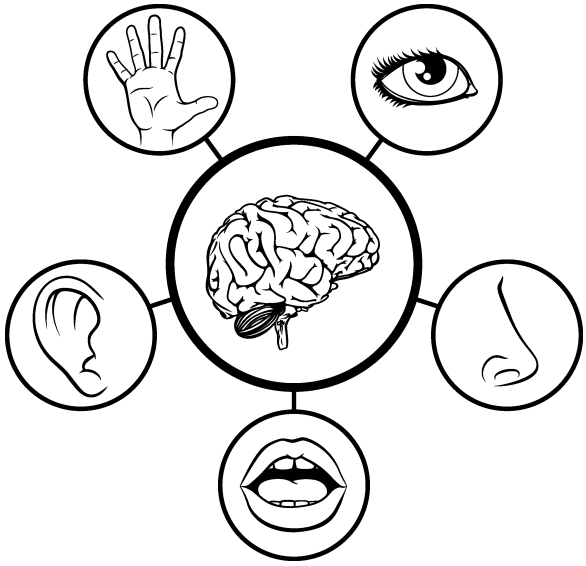
Figure 1.26: A multi-layer perceptron can compute complex Boolean functions

Recap 4.1

1. Neural networks began as computational models of the brain.
2. Modern neural network models are connectionist machines.
3. They comprise networks of neural units.
4. **McCullough and Pitt model:** Uses neurons as units that can perform Boolean operations. It models the brain as performing propositional logic, or theshold logic, but no learning rule.
5. **Hebb's learning rule:** Neurons that fire together wire together. Model is Unstable.
6. **Rosenblatt's perceptron:** A variant of the McCulloch and Pitt neuron with a provably convergent learning rule, but individual perceptrons are limited in their capacity to do all Boolean operation (Minsky and Papert).
7. Multi-layer perceptrons can model arbitrarily complex Boolean functions.

1.4.7 The perceptron with real inputs

So far, we have seen that in the development of computational neural units, much attention was focused on the need for performing Boolean operations, and all of our examples used Boolean inputs and outputs. However, our brain is *not* a Boolean machine. We obtain *real-valued inputs* and make non-Boolean inferences, as Fig. 1.27 reminds us.

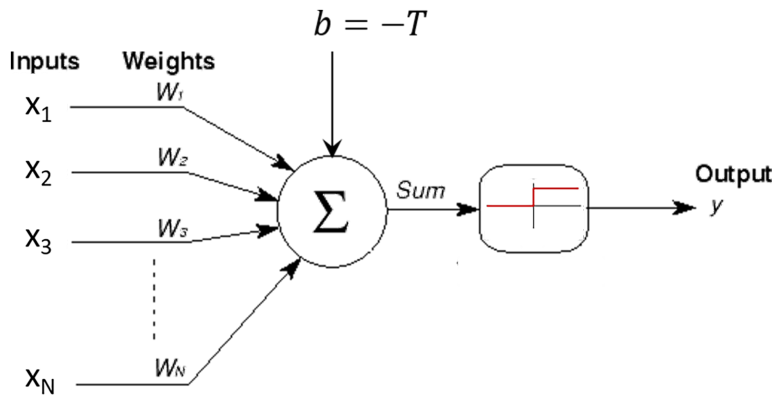


Real inputs to the brain: sight (light), smell (chemical), hearing (acoustic), touch (haptic) and taste (chemical).

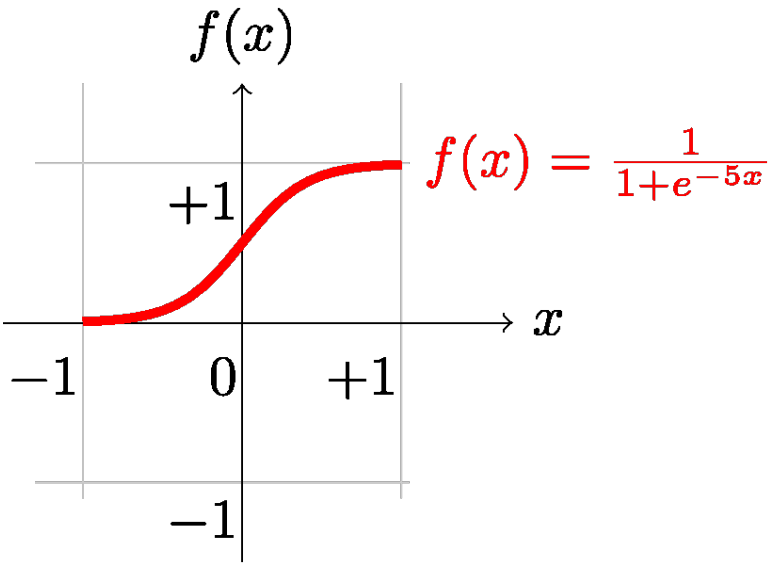
Figure 1.27: Real inputs to the brain

Let us understand the implications of this a little further before we discuss how perceptrons can handle real (for our purposes, real-valued) inputs. Fig. 1.28(a) shows a perceptron. While there is no change in its structure from that shown in Fig. 1.23, the arrangement of the computation is slightly modified.

The inputs $x_1, x_2, \ldots, x_N$ to the perceptron are now real numbers. The weights too are real numbers. The perceptron first computes an affine combination of the inputs: $z = \sum_i w_i x_i + b$, and subsequently applies a threshold "activation" $\theta(z)$ to the affine value z, which outputs a 1 if this is non-negative, and 0 otherwise.



(a) The perceptron is implemented by computing an *affine* combination of the inputs, followed by an activation. An affine combination of inputs is the weighted sum of the inputs, plus a bias. Here the bias term is set to -T. The activation function is set to a *threshold* activation which outputs a 1 when the input is greater than or equal to 0, and 0 otherwise. In effect, the perceptron fires if the weighted sum of inputs exceeds the threshold T.



(b) A sigmoid activation: this is a smoother version of the threshold activation that goes continuous from 0, when the input is very negative, to 1, when it is highly positive.

Figure 1.28: Real inputs to a perceptron

If the bias $b = -T$, then the perceptron is identical to the one in Fig. 1.23, which fires if the weighted sum of inputs exceeds a threshold z.

$$z = \sum_i w_i x_i + b \tag{1.5}$$

$$y = \theta(z) \tag{1.6}$$

$$\theta(z) = \begin{cases} 1 & if \\ z \geq 0 & \\ 0 & else \end{cases} \tag{1.7}$$

Note that with $b = -T$, this is simply a restatement of the threshold logic presented earlier in the case of Resenblatt's perceptron.

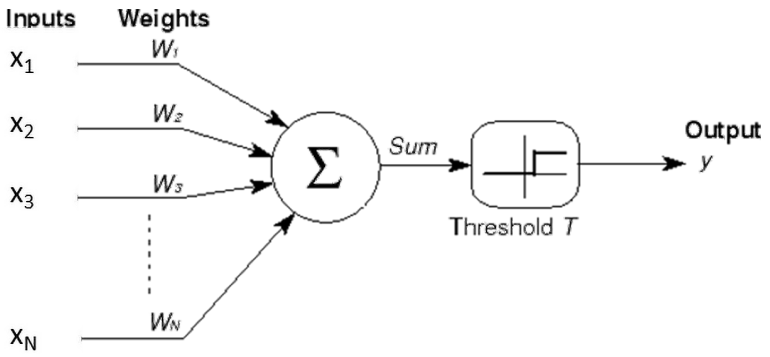
We can now easily extend the perceptron model to have real-valued outputs too. Instead of a threshold activation, we can replace it with a sigmoid (shown in Fig. 1.28(b)), which is a smooth version of a threshold. Now the perceptron produces real valued outputs between 0 and 1, which can be viewed as a *probability of firing*.

In the example shown in Fig. 1.28, the threshold activation function can be replaced with *any* real-valued activation function that operates on the affine combination of inputs. We will discuss many other activation functions later in this book.

The net result of this change is that now the perceptron maps real-valued inputs to real-valued outputs. However, to understand a perceptron's workings intuitively and for easy interpretation, let us for now continue to assume real-valued inputs, but Boolean outputs.

1.4.8 Boolean functions with a real-valued perceptron

Fig. 1.29(a) shows a perceptron with real inputs again. It operates within a real-valued space of inputs. If the weighted combination of inputs exceeds a threshold, it outputs a 1. If not, it outputs a 0. The boundary between the region of the input space where it takes the value 0, and the region where it takes the



$$y = \begin{cases} 1 \; if \; \sum_i w_i \mathrm{x}_i \geq T \\ 0 \; else \end{cases}$$

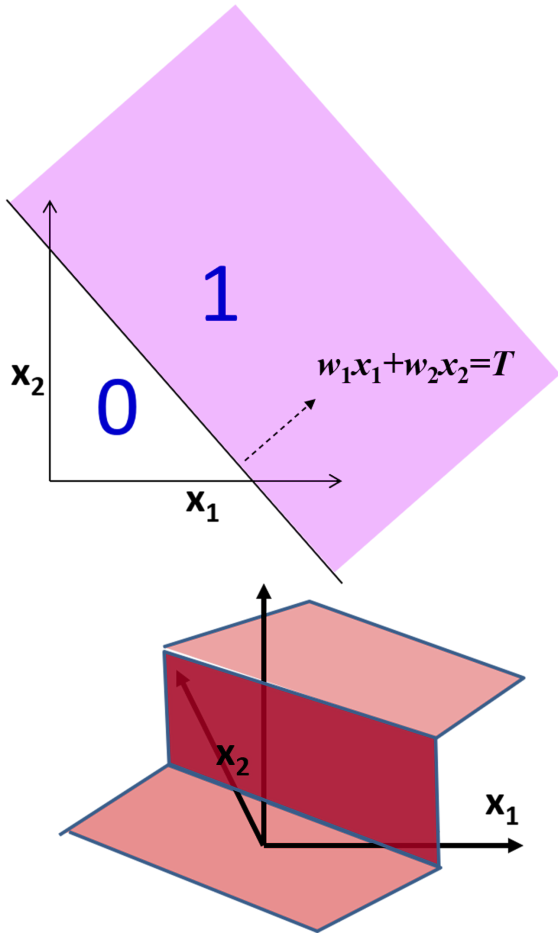
(a) Perceptron with real inputs. Here the bias term is set to -T. The perceptron fires if the weighted sum of inputs exceeds the threshold T.

Figure 1.29: The perceptron as a linear classifier

value 1, is the set of inputs where the (weighted) linear combination of inputs exactly equals the threshold, i.e $\sum w_i x_i = T$. This is in fact the equation of a hyperplane in the input space, shown in Fig. 1.30(). For inputs on one side of the plane, shown by the color pink in the upper panel, the output is 1, on the other side (white region), the output is 0. The lower panel of Fig. 1.30() schematically represents an equivalent 3-dimensional plot for 2-dimensional inputs, where two of the axes represent the inputs, and the third vertical axis represents the output. As shown in this panel, the input-output relation for the perceptron appears as a step (or "heavyside" function), where the output is 0 on one side of the edge of the boundary plane, at which point the output becomes exactly 1.

Since the boundary itself is a hyperplane – i.e., it is linear – this perceptron can operate as a linear classifier, predicting 0 on one side of the linear boundary, and 1 on the other.

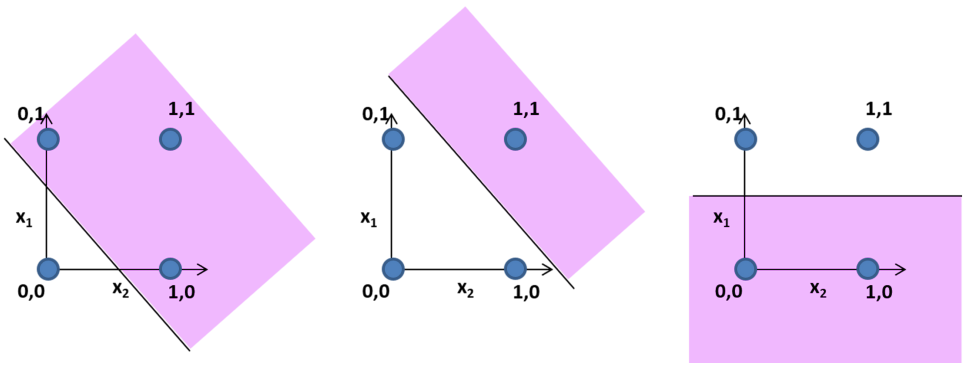
It is now possible to see how the perceptron of Fig. 1.29(a) can model different Boolean functions. For a Boolean function, all inputs are binary – either 1 or



This perceptron can operate as a linear classifier, predicting 0 on one side of the linear boundary, and 1 on the other.

Figure 1.30: The perceptron as a linear classifier

0. Thus, for a two-input perceptron, the inputs can take one of only four values (0,0), (0,1), (1,0) and (1,1), represented by the four balls shown in each figure. If we construct our perceptron to have the linear boundary shown in the leftmost panel of this figure, it will output 1 for the three combinations of inputs that are shown in the pink region on this panel. Clearly, this perceptron models the Boolean OR function. If, on the other hand, we construct the perceptron to have the boundary shown in the second (middle) panel, it will output 1 only when both inputs are 1. Thus this perceptron would model the Boolean AND function. Similarly, the perceptron the third panel outputs a 0 when x_1 is 0, and a 0 when it



Depending on how the boundaries are constructed, the perceptron shown can model different Boolean functions (from left to right: OR, AND, ...).

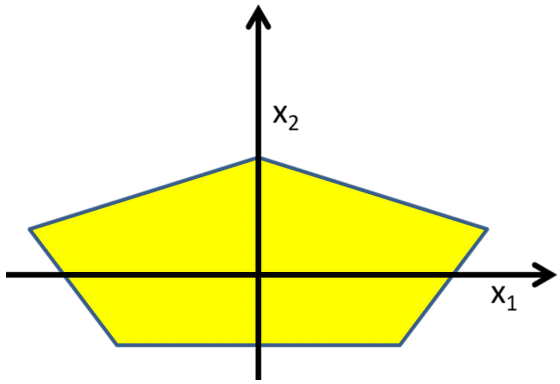
Figure 1.31: The perceptron as a linear classifier

is 1, so this perceptron computes NOT x_1. It must be apparent from these figures why the perceptron of Fig. 1.29 cannot model the Boolean XOR function. We leave the reader to reason why for now. We will explain this in Chapter 2.

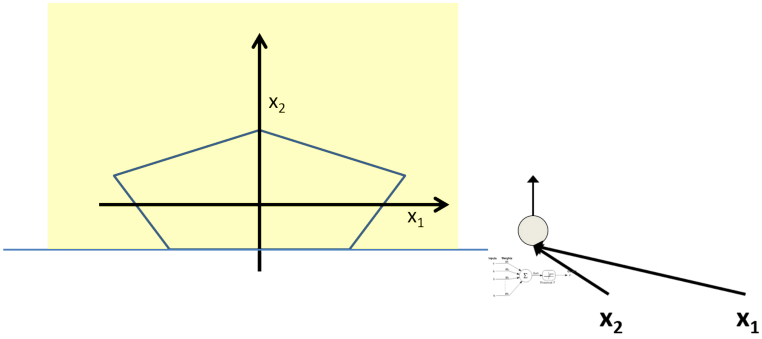
1.4.9 Constructing complex decision boundaries with a perceptron

With this understanding and view of the single perceptron as a linear classifier, it is easy to see how a *network* of perceptrons can be used to construct arbitrarily complex decision boundaries. Consider, for example, the pentagonal decision boundary shown in Fig. 1.32.

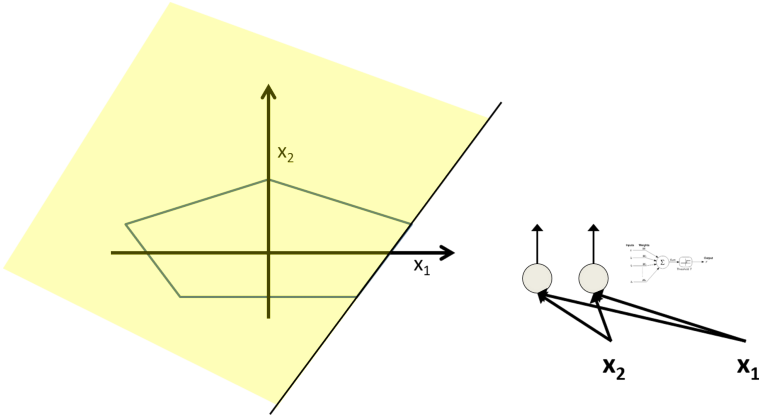
For a perceptron classifier to have such a decision boundary, it must be modeled such that it outputs a 1 for inputs that lie within the pentagonal region shaded yellow, and 0 outside. In order to achieve this, we construct one perceptron to capture each of the sides of the pentagon as a linear boundary. The perceptron modeling any boundary must output a 0 on the side *away* from the pentagon, and 1 on the side that the pentagon lies on. Figs. 1.32(b),(c), 1.33(a)-(c) and 1.34(c) show this for each of the boundaries. Thus for the five boundaries we require five first-layer perceptrons.



(a) A pentagon shaped decision boundary.



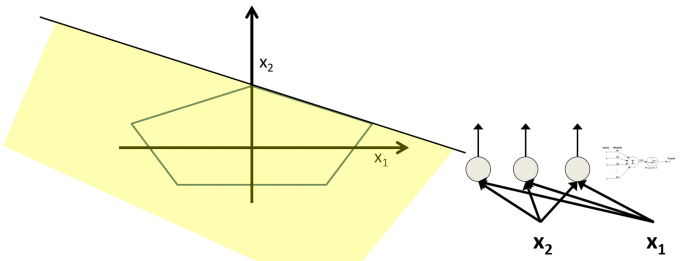
(b) The network must fire if the input is in the yellow area.



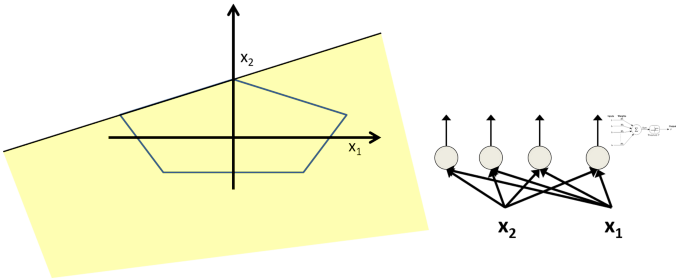
(c) The network must fire if the input is in the yellow area.

Figure 1.32: Construction of a complex decision boundary using a perceptron

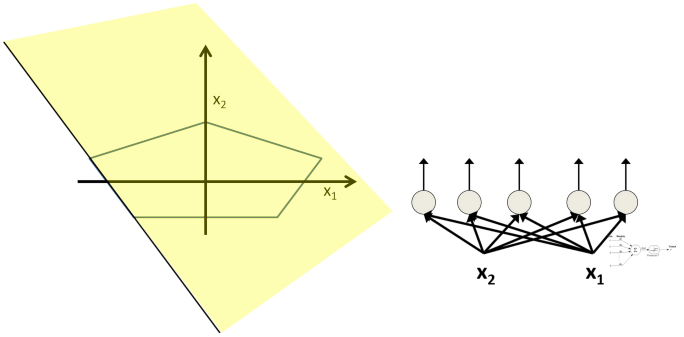
We see from this example, that the *only* region in which the output of all five perceptrons is 1 is within the pentagon. In all other areas, at least one perceptron does not fire. To complete this network of perceptrons, we must add a final out-



(a) The network must fire if the input is in the yellow area.



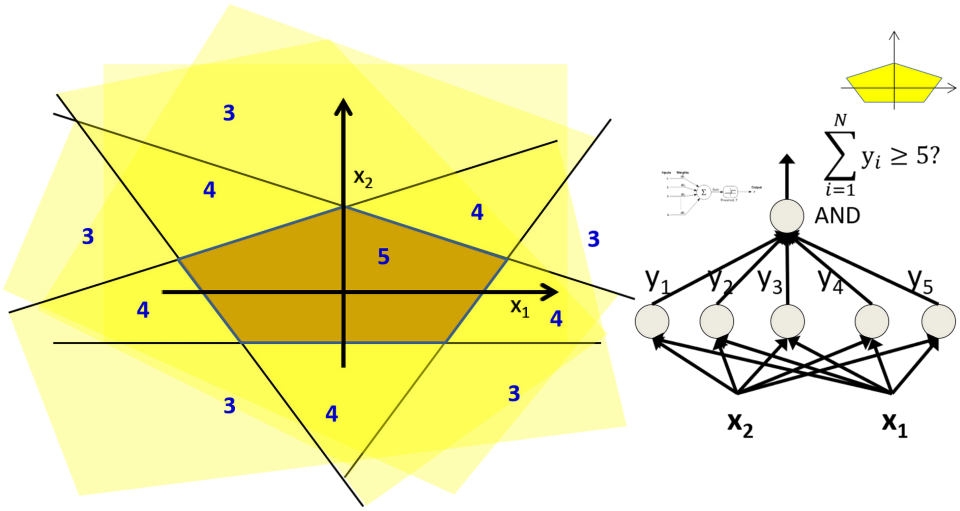
(b) The network must fire if the input is in the yellow area.



(c) The network must fire if the input is in the yellow area.

Figure 1.33: Construction of a complex decision boundary using a perceptron

put perceptron that identifies the region where all five first-layer perceptrons fire. This perceptron effectively simply ANDs the outputs of the five first-layer perceptrons. This can be done by comparing the sum of their outputs to a threshold value of 5, resulting in a network that outputs 1 exactly when the input is within the pentagon, and does not fire otherwise. Fig. 1.34(a) illustrates this logic and also shows the final network of perceptrons that has the desired decision boundary.



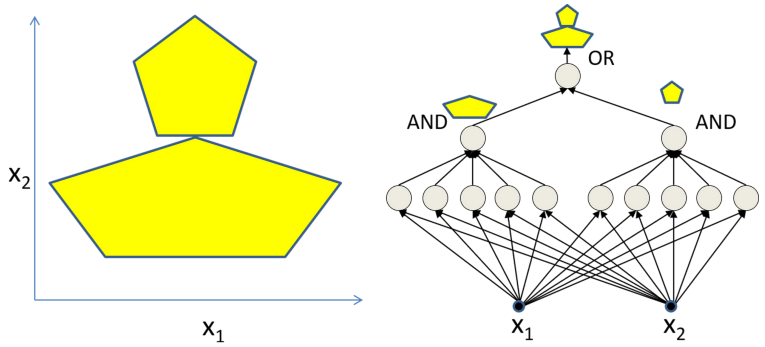
(a) The network must fire if the input is in the yellow area.

Figure 1.34: Construction of a complex decision boundary using a perceptron

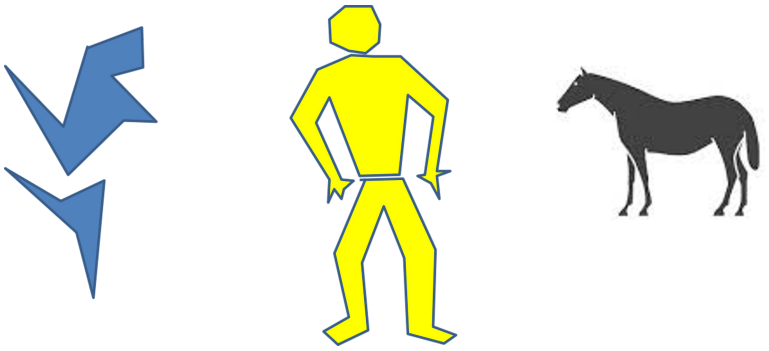
From this example, it is clear that it should be possible to similarly construct networks for even more complex decision regions, such as the ones shown in Fig. 1.35. To construct the double pentagon shown in Fig. 1.35(a), for example, we can use two subnets: one outputs a 1 only if the input lies within the first pentagon, the other does the same for the second pentagon. A network that has the union of the two decision boundaries must include a final perceptron to OR the outputs of the first two subnets.

In a similar manner, it is possible to construct arbitrarily complex decision boundaries, such as the ones shown in Figs. 1.35(b) and (c).

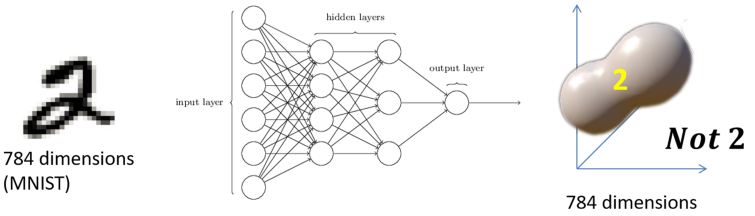
Most real-life tasks work on much higher-dimensional inputs, but the same principle applies. An MLP that performs classification on high-dimensional inputs models high-dimensional classification boundaries. For example, to build a network that will recognize (or fire for) the digit "two", in a digits classification task, where the image of each digit has 784 pixel values, the input must be 784 dimensional. As in Fig. 1.35(c), the region of the input space that corresponds the digit 2 will comprise some region of the space (illustrated roughly by the peanut shaped figure in this panel). We must construct a multi-layer perceptron (or MLP) that captures this boundary and outputs a 1 for inputs that fall within



(a) A double pentagon shaped decision bounday and the corresponding network structure that achieves it. To "OR" the two polygons a third layer is required.



(b) More complex decision boundaries. Networks can be constructed to achieve these.



(c) Finding decision boundaries in high-dimensional space can be performed by a multi-layer perceptron (MLP). MLPs can classify real-valued inputs. In this case the input is the image of a handwritten digit.

Figure 1.35: Examples of more complex decision boundaries that can be constructed using a network of perceptrons.

the peanut shaped region.

Recap 4.2

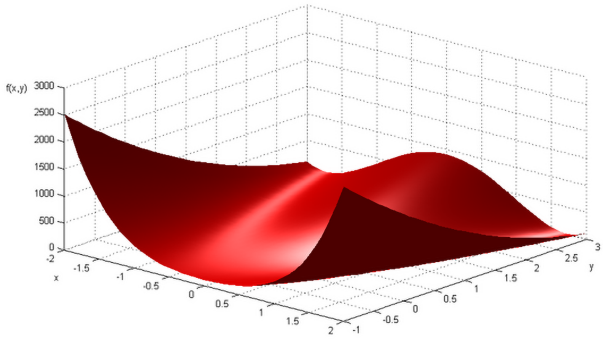
1. MLPs are connectionist computational models
   - Individual perceptrons are computational equivalent of neurons
   - The MLP is a layered composition of many perceptrons
2. MLPs can model Boolean functions
   - Individual perceptrons can act as Boolean gates
   - Networks of perceptrons are Boolean functions
3. MLPs are Boolean machines • They represent Boolean functions over linear boundaries • They can represent arbitrary decision boundaries • They can be used to classify data

1.4.10 When the outputs are real valued

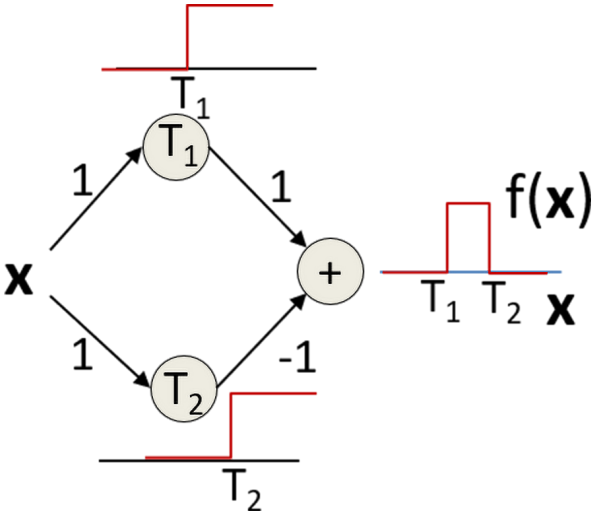
We have so far considered the case where the inputs are real valued, while the outputs are Boolean variables. How does a perceptron, or a network of perceptrons work for cases where even the outputs must be real valued? As our final topic for this chapter, we will consider this case.

Fig. 1.36 shows an example of a continuous-valued set of outputs. How would we construct a network to model such real-valued functions?

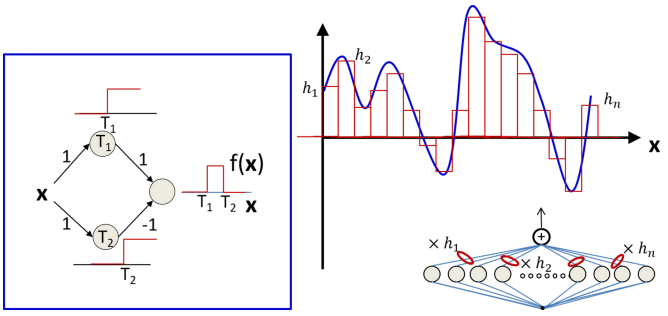
To understand how this can be done, consider a function of a scalar input as shown in Fig. 1.36(b). In this figure, we have two perceptrons, one of which outputs a 1 when the input exceeds a threshold T_1, and a second which outputs a 1 when the input exceeds threshold T_2. The outputs of the two perceptrons are summed with weights of 1 and -1 respectively. Let us now consider the overall response of this network. Assume without loss of generality that $T_1 < T_2$. When the input X is less than T_1 both perceptrons output 0, and the overall output of the net is 0. When $T_1 \leq X < T_2$, then the first perceptron outputs a 1, while the



(a) An example of a continuous-valued output.



(b) .



(c) .

Figure 1.36: MLPs as a continuous valued regression

second outputs a 0, and thus the overall output of the net is 1. When $X \geq T_2$ both perceptrons output 1 and cancel each other out in the weighted sum, thus the overall output of the net is 0. Thus the network shown in Fig. 1.36(b) produces an output of 1 when the input is between T_1 and T_2, and produces 0 otherwise.

Fig. 1.36(c) explains the case of MLPs as continuous-valued regression further. If we want to construct a network that approximately gives us the function represented by the blue curve on the upper right in Fig. 1.36(c), we partition the x axis into ranges, and use one subnet of 2 neurons to produce a square pulse of height 1 in each range. We then also scale the outputs of the individual subnets appropriately by weighting their outputs correctly. Adding up all of the subnets (and outputs) then results in the curve shown in the figure. The approximation to the curve can be made arbitrarily precise by making the partitions on the x axis (or the ranges) narrower. This construction can be extended to any scalar function of a scalar input. In other words, an MLP can approximate *any* scalar function of scalar inputs to arbitrary precision. It also generalizes to functions of multi-variate or vector inputs. We will discuss this in later chapters of this book.

Recap 4.3

1. MLPs are connectionist computational models
2. MLPs classification engines
   - They can identify classes in the data
   - Individual perceptrons are feature detectors
   - The network will fire if the combination of the detected basic features matches an "acceptable" pattern for a desired class of signal
3. MLPs can also model continuous valued functions

In later chapters of this book, we will see that there are many other things that MLPs can do, such as modeling memory, representing probability distributions over real, complex and integer domains, modeling both the *a posteriori* and *a*

priori probability distributions of data, generating data from complicated and potentially unknown distributions etc.

Fun facts 4.5: Models for memory

Lawrence Kubie in 1930 hypothesized that closed loops in the central nervous system explain memory. The perception of memory or "remembering" can indeed be created by neural networks with loops.

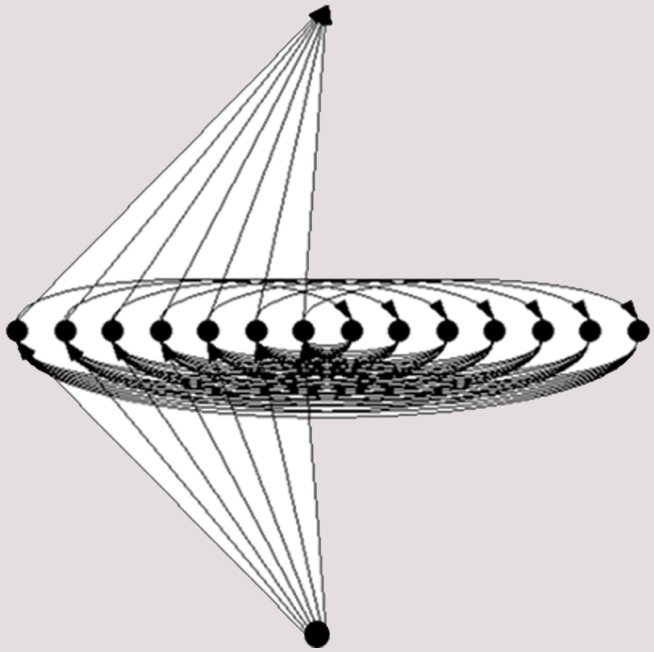
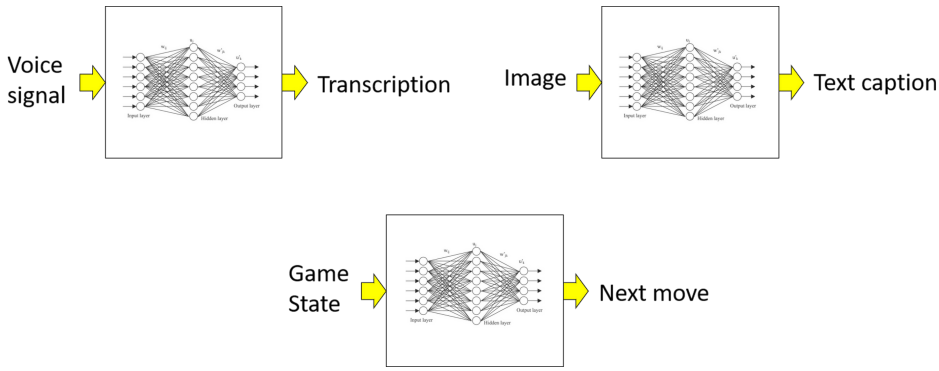


Figure 1.37: Networks with loops can create the perception of memory.

Returning to the question we began with, from an AI perspective, a neural network is just a function. Given one or more inputs, it computes the function to predict one or more outputs. Thus the black boxes shown in Fig. 1.5 are just functions: the voice recognition function takes in voice and produces a transcription. The image captioning function takes in an image and generates a caption. The game playing function takes in a game state and outputs the best next move. When we perform these tasks using neural nets, all of these functions are modeled or approximated by neural networks.



A glimpse into the blackboxes shown in Fig. 1.5 earlier in this chapter.

Figure 1.38: Neural networks are functions that perform AI tasks

Recap 4.4

1. Multi-layer perceptrons are connectionist computational models
2. MLPs are classification engines
3. MLPs can also model continuous valued functions
4. AI tasks can be viewed as functions that can be modeled by networks of perceptrons

We close this chapter with the observation that MLPs can basically model any kind of computational task – they are *universal approximators*. We will discuss this aspect of MLPs in the next chapter.

Bibliography

[1] Singh, R., 2019. Profiling humans from their voice. Springer.

[2] Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X. and Stolcke, A., 2018, April. The Microsoft 2017 conversational speech recognition system. In 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 5934-5938). IEEE.

[3] Godfrey, J.J., Holliman, E.C. and McDaniel, J., 1992, March. SWITCHBOARD: Telephone speech corpus for research and development. In Acoustics, Speech, and Signal Processing, IEEE International Conference on (Vol. 1, pp. 517-520). IEEE Computer Society.

[4] Karras, T., Laine, S. and Aila, T., 2019. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4401-4410).

[5] Minsky, M., 1988. Society of mind. Simon and Schuster.

[6] Gregoric, P., 2007. Aristotle on the common sense. Oxford University Press.

[7] Von Gerlach, J., 1858. Mikroskopische Studien aus dem Gebiete der menschlichen Morphologie. Enke.

[8] Bentivoglio, M., 1998. Life and discoveries of Camillo Golgi. Nobel e-Museum http://www.nobel.se/medicine/articles/golgi/index.html First published April, 20.

[9] Bain, A., 1874. Mind and body.

[10] Copeland, B.J. and Proudfoot, D., 1999. Alan Turing's forgotten ideas in computer science. Scientific American, 280(4), pp.98-103.

[11] Ferrier, D. (1876). The functions of the brain. G P Putnam's Sons. https://doi.org/10.1037/12860-000

[12] Rumelhart, D.E., Hinton, G.E. and McClelland, J.L., 1986. A general framework for parallel distributed processing. Parallel distributed processing: Explorations in the microstructure of cognition, 1(45-76), p.26.

[13] Rumelhart, D.E., Smolensky, P., McClelland, J.L. and Hinton, G., 1986. Sequential thought processes in PDP models. Parallel distributed processing: explorations in the microstructures of cognition, 2, pp.3-57.

[14] McClelland, J.L., Rumelhart, D.E. and Hinton, G.E., 1986. The appeal of parallel distributed processing. MIT Press, Cambridge MA, pp.3-44.

[15] McClelland, J.L., Rumelhart, D.E. and PDP Research Group, 1986. Parallel distributed processing. Explorations in the Microstructure of Cognition, 2, pp.216-271.

[16] Medler, D.A., 1998. A brief history of connectionism. Neural Computing Surveys, 1, pp.18-72.

[17] Bechtel, W. and Abrahamsen, A., 1991. Connectionism and the mind: An introduction to parallel processing in networks. Basil Blackwell.

[18] McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), pp.115-133.

[19] Hebb, D.O., 1949. The organization of behavior: a neuropsychological theory. J. Wiley; Chapman Hall. Republished: Hebb, D.O., 2005. The organization of behavior: A neuropsychological theory. Psychology Press.

[20] Minsky, M., 1969. Papert. S., Perceptrons., MIT Press, Cambridge, MA.