

Before Machine Learning

Volume 2

CALCULUS

Jorge Brasil

Copyright © 2023 Jorge Brasil all rights are reserved.

This is the point where you are made aware that, even the mere thought of trying to duplicate this content could lead to the risk of a prolonged jail sentence. Ah! Just joking. If you wish to reproduce bits of the book, don't hesitate to contact me. I hope you enjoy the reading and that by the end of it, you have become a Calculus mean machine. If you wish to complement your reading, you'll find a link below to download a document filled with practical and coding exercises. The solutions for the practical exercises and the corresponding Python code will also be in that same location.

Link for exercises!

This book is the second of a three-volume series called Before Machine Learning. The first volume was dedicated to linear algebra and is available on Amazon or my website. The remaining book will be on Probability and Statistics.

www.mldepot.co.uk

Subscribe for updates!

Subscribe here!

For a better reading experience on your MAC, please use the Kindle Classic app.

Contents

Contents	3
1 Why Calculus?	8
2 Pointing Fingers and Crossing Lines: The last breath of just linearity.	10
2.1 Swipe Right for co-domains: Functions	18
3 Changing Times and Tangent Lines: The Derivative. 26	
3.1 The Tale of the Eight with Wobbly Knees: Infinity	44
3.1.1 Forever Close, Never Met: The Untouchables Asymptotes	52
3.1.2 Better Sharpen Our Pencils: Continuity , The No-Lift Zone!	58
3.1.2.1 Boomerang Bonanza: The Inverse Function 's Return Trip.	58
3.1.2.2 The Mood Swings of the Graph World: Piecewise Functions	61
3.2 Claustrophobia in the Calculus Corridor: The Limit	63
3.3 Deciphering the Undefined - L'Hopital's Rule	77
3.3.1 After You, Please: The Cordial Conduct of Function Composition	86
3.3.1.1 The Key to the Calculus Handcuffs: Chain Rule	86
3.4 Get Me Higher and I'll Give You Better Curves: Taylor Series	94
3.5 Valleys and Hills: A Critical Situation for Points and Their Roots	105
3.5.1 Zeroing in: Newton-Raphson Method . . .	114

3.5.2	The Original GPS: Gradient Descent	134
3.5.3	Dare You Cross Me, Line? Convexity	139
3.5.4	Really?! Dishing Out More Dough? The Cost Function	143
4	Cleaning Up The Derivatives Debris: The Integral.	157
4.1	That Riemann Guy Is a Bit Square, Isn't He?	162
4.2	This seems important: The Fundamental Theorem of Calculus	167
4.3	Hold On, Logs Aren't Just for Fireplaces? Scaling Data	194
5	A Free Upgrade: More Dimensions.	200
5.1	Numerous Entries But Single Exit: Scalar Functions .201	
5.1.1	Grab Your 3D Glasses for these Multidimensional Integrals	202
5.1.2	I Am Quite Partial to These Derivatives.	205
5.1.3	Just Follow My Lead: The Gradient	208
5.1.4	GPS 2.0: Multidimensional Gradient Descent	211
5.1.5	Bending Realities, Unveiling Curvatures: The Hessian	212
5.1.6	I Thought We Were Past Lines: Linear Regression	219
5.1.7	Newton, Raphson, and Hessian Walk Into a Bar: No Joke, They Found a Root!	226
5.1.8	A Mathematical Marriage: Scalar Functions and Neural Networks	232
5.1.8.1	Wake Up Call for Neurons: Curve Alert with Activation Functions	243
5.1.8.2	Wait, Which Way? The Min-Max Scaling	249
5.1.8.3	The Price of Disorder: Cross Entropy as a Cost Function	257
5.1.8.4	A Detour Through Randomness: Stochastic Gradient Descent	260
5.1.8.5	Upgrading Our Locksmith Skills: The Multivariate Chain Rule	262

5.1.8.6	One Step Back to Give Two Steps Forward: Backpropagation	269
5.1.8.7	Customized Routes to Minima: RM-SProp	278
5.2	A Crowd Comes In, A Parade Goes Out: Vector Functions	286
5.2.1	A Clandestine Mathematical Liaison: Vector Functions and Neural Networks	294
5.2.1.1	Gathering Steam: The Momentum Algorithm	303
5.2.2	Smooth Sailing with a Customized Map: Embarking on the Adaptive Moment Estimation	310

Forward

I've known Jorge, aka Arguaramy since we were kids, but we grew closer in the early 2000s. Our group of friends always knew that his future would be brilliant regardless of what, where, or how.

The first time he mentioned the words "writing a book," I was there. "I was there, Gandalf! I was there 3000 years ago!" It was one of those nights in Oporto at a club called Blhá Blhá.

The establishment in question possessed a pair of funny, notable characteristics. Firstly, a line on the dancefloor arbitrarily divided the territories of smoking and non-smoking. A mere ten centimeters became the boundary between a cloud of smoke and clear air. Secondly, and of greater significance in this narrative, was the bathroom door that lacked a functional lock. It was Jorge who fell victim to this oversight. A sudden and urgent call of nature found him ensconced within for the duration of not one, but three anthems, 'Stairway to Heaven' among them, suffering for having to hold the door so that no one got in.

After that while ordering a drink at the bar, he said: "Damn. I spent so much time there that I could nearly have written a book!". He did not do it then, but more than 15 years later, here we are, and we are killing it! In this book, Jorge weaves together bees and mathematics, transforming what I anticipated to be as daunting as a 17-hour chess match against Kasparov into a leisurely walk in the park. The average person, including myself, can only understand these topics through an approach similar to Jorge's, as demonstrated in his first book and now again in his second.

For this remarkable achievement, extra kudos. I hope you like

Contents

this book as much as I did. To many more publications! And, of course, I want to thank my "filhinho" for letting me be a part of this book and his life.

The Dream is alive! Keep living it!

Your friend in time,

Fernando "bébé" Vieira

Chapter 1

Why Calculus?

Linear algebra has graced us with the tools to navigate through grids and planes, unveiling the secrets lying within data sets and multidimensional spaces. It allows for a clean, systematic exploration of structures, where every vector has its place, and every matrix transformation follows a clear and predictable path. In this orderly realm, everything adheres to the principles of linearity, a domain ruled by straight lines.

It may seem like we set ourselves to completely dismiss Linear Algebra, but that is not the case. Calculus doesn't overthrow the principles established in linear algebra. Still, it enhances and elevates them, building upon the firm foundations to explore new, dynamic terrains. It's an invitation to view the world through a different lens, one with curves that define the natural phenomena around us.

A curvilinear world is only possible with a particular fellow, that guy we call infinity, a fascinating notion, standing at the crossroads where the finite meets the boundless. Infinity invites us to stretch our understanding beyond the conventional to grapple with infinitely small or vastly expansive ideas. It challenges us to visualize a world where lines can extend eternally, where areas under curves can be dissected into infinitesimally small portions yet sum up to finite quantities.

Just as linear algebra serves as the underlying framework for understanding complex data structures, calculus is the driving force

behind the optimization and functionality of machine learning algorithms, taking the role of an assistant that aids in fine-tuning processes, enabling machines to learn and adapt, and allowing us to better represent real-world phenomena. Speaking of the world, our journey through the curves will be accompanied by a fascinating black and yellow creature that will not sting you but rather provide assistance and help, which is the aim of this book. So, let's get this party started with the simplest element of mathematics.



Chapter 2

Pointing Fingers and Crossing Lines: The last breath of just linearity.

Let's start simple, and I do mean simple. We are kicking off things with the plainest mathematical element, at least to my recollection. I am talking about the point. In simple terms, a point is just a little dot. However, to this seemingly simple shape, we can assign coordinates, with no limit to their number. Therefore, a point gives us an exact position in an n -dimensional space. For example, if $n = 2$:

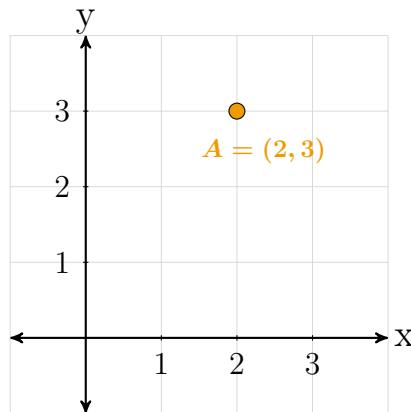


Figure 2.1: There is nothing exciting here. It is just a point.

$$A = (x, y) = (2, 3)$$

What can we do with one point? With just one, not much, but

Chapter 2. Pointing Fingers and Crossing Lines: The last breath of just linearity.

it happens that with two points, we can define a line. Let's explore this concept by introducing a new point, one that we will name B , and B will have $(-1, -3)$ for coordinates:

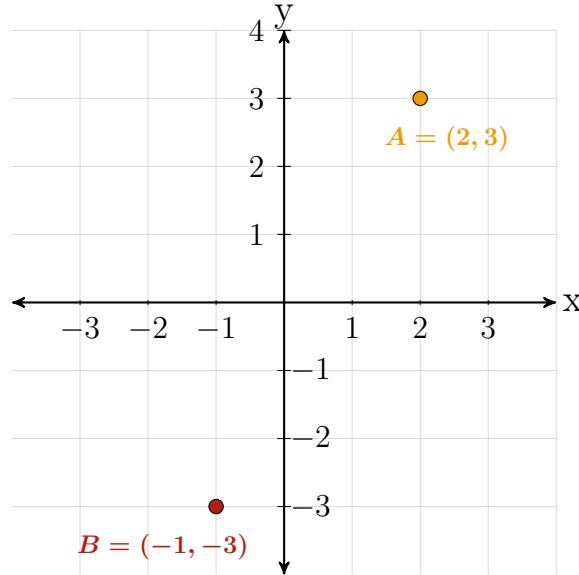


Figure 2.2: What is better than one point? Perhaps two?

One day, out of the blue, I received a call from a childhood friend inviting me to join him to watch a Metallica concert, the heavy rock band. At the time, residing in London, the prospect of traveling to Lisbon for the show posed quite the trek. Despite my reservations, I agreed to go and booked my flights.

As it turned out, my friend had also invited someone else along, a guy who I didn't recognise. It was strange, considering we had all grown up in the same small place and were roughly the same age. But I soon discovered that this guy was an interesting character. He was a police officer and a beekeeper. Bees had never sparked that much interest in me, but my curiosity increased as the night wore on and the whisky flowed to inundate my brain. I found myself continually asking the beekeeping cop about these animals. Astonishment took hold upon learning about the intricate and fascinating behaviors of these creatures. Looking back, I was so drunk that he could have said anything, like "Bees do photosynthesis..." and I probably would have believed that.

Imagine that the points A and B , which we defined earlier, repre-

sent two of many waypoints a bee will pass through on a given route. Bees have remarkable capabilities that make them more impressive than a fighter jet. They use the sun as a compass and calculate angles between the hive and it. It's also believed that they can detect Earth's magnetic field, which may assist their navigation, much like a built-in magnet. So you'll know what is going on if you ever see a bee flying with a tiny needle attached to its body!

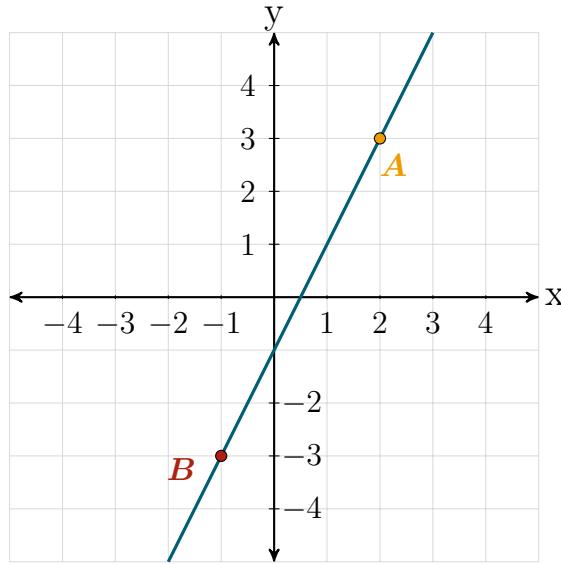


Figure 2.3: A line, but one that is safe for consumption.

Currently, we only have access to two positions in space. How can we have access to the bee's locations at all times? At any given x -coordinate, what would the y -coordinate be? We are only concerned with its safety; by no means do we want to exert control over this creature or invade its privacy. If you're wondering whether they can fly in perfectly straight lines, the answer is that it depends on factors like the wind. For now, let's assume that there is no wind, and a straight line represents the bee's route. Perhaps we could create an equation where we could plug in the x -coordinate and get the y or vice versa. One of the parameters that can aid us in defining that line is the angle between it and the x -axis. Changing this angle will mean that we are creating different lines:

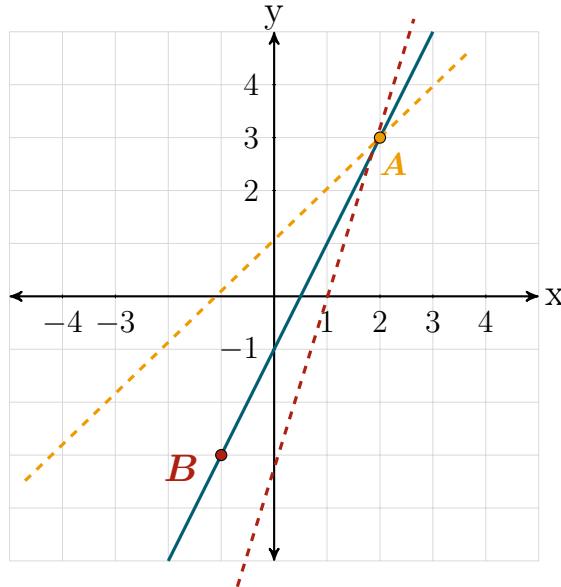


Figure 2.4: The effect of rotating a line around a point.

Figure 2.4 displays two dashed lines, each resulting from a change in the angle between the blue line and the x -axis. If changing an angle creates a new line, this suggests that, to define the equation of such a mathematical concept, we must include a parameter that carries information about this tilt. One way to do this is to create a square triangle and leverage all the mathematics that Mister P for Pythagoras deduced:

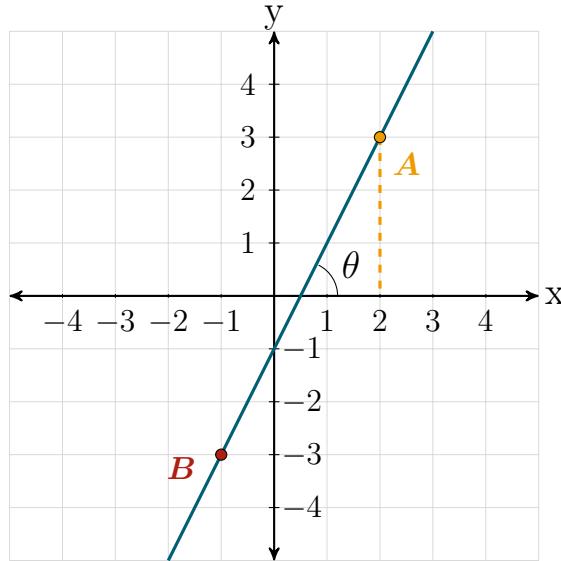


Figure 2.5: Mister P would be proud of our first square triangle.

We have a square triangle formed by the blue line, the x -axis,

and the dashed yellow line. Now is the time to revert to high school and start smoking cigarettes behind the school's gym... excuse me, I meant it is time to bring back the Pythagoras theorem:

$$\tan(\theta) = \frac{\text{length of the opposite side}}{\text{length of the adjacent side}}$$

The division of the length of the opposite side by the length of the adjacent side defines the tangent of the angle θ . Well, this is the same as saying that for us to get such a measure of θ , we can divide y by x , so:

$$\tan(\theta) = \frac{y}{x}$$

However, we don't have x and y ; our only available information is the two points A and B . Thus, we need to define the angle θ in a way that allows us to calculate it using the coordinates of these points. For that, let's go back to the drawing board and create a different triangle:

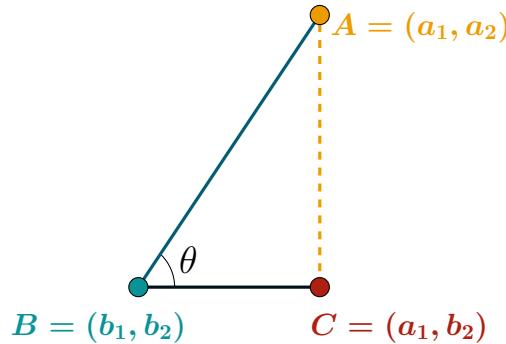


Figure 2.6: A big square triangle to help with the slope.

Our goal is to define an equation for the tangent of the angle between the blue line and the x -axis so that our formula for it depends on the coordinates of both A and B . I am a bit tired of the sentence "the angle between blah blah blah," so let's call that magnitude the "slope". Previously, we defined the slope using Pythagoras theorem, and it worked out to be a division of two distances: the length of the opposite side divided by the length of the adjacent side.

In the new triangle shown in figure 2.6, we have only changed the lengths of the sides. All the angles remained the same, so if

Chapter 2. Pointing Fingers and Crossing Lines: The last breath of just linearity.

we define a point C , whose coordinates can be written by what we know from A and B , we can have a new equation for the slope that will come just with coordinates of those two points.

Point C is horizontally aligned with B , which means the y -coordinate for C should be the same as that for B , that is why we have the b_2 for this value. Because we have a vertical alignment with A , the x -coordinate of the fellow point C must be the same as A . This defines C as (a_1, b_2) , so this gringo slope can be defined as:

$$m = \frac{(b_2 - a_2)}{(b_1 - a_1)} \quad \text{with } A = (a_1, a_2) \text{ and } B = (b_1, b_2) \quad (2.1)$$

We have assigned the letter m to the concept of slope, and there is a new equation for it, the one defined by 2.1. It is crucial to mention that you can also see the slope as being defined by the division of the “rise” and the “run”. These are two new words for concepts that refer to change: the rise is the vertical change between points, whereas the run depicts horizontal change.

I know, I know; where are all the curves? So far, everything is pretty straight and linear, but fear nothing, as they will be coming soon. We have merely stumbled across a fundamental concept that will be present throughout this book; changes in y are proportional to modifications on x ; this is what the equation 2.1 above tells us. So we can define an equation for that line as:

$$y = mx$$

The question now becomes whether this is enough or not. Well, it never is. For example, check these two lines with the same slope:

- $y = 2x + 1$
- $y = 2x - 1$

Let's plot these sisters:

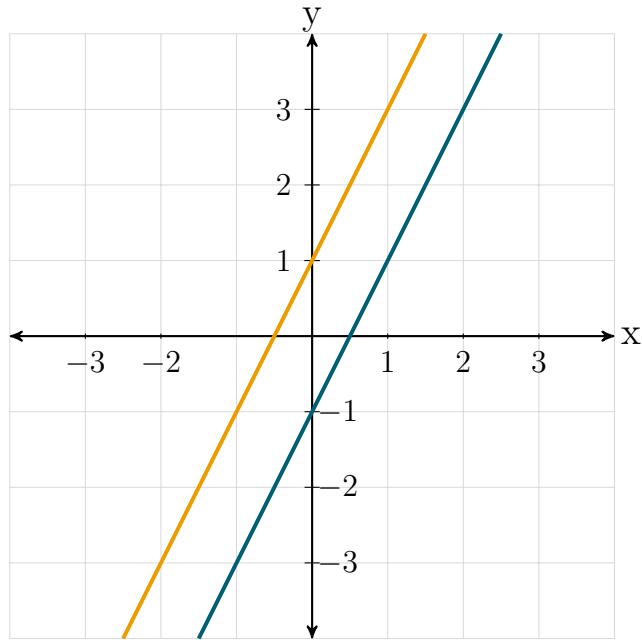


Figure 2.7: One more line?!? We want CURVES!

Although both lines share the same slope, they are visually distinct. Therefore just mx won't be enough to define these individuals. Right, allow me to introduce one more line, $\mathbf{y = 2x}$:

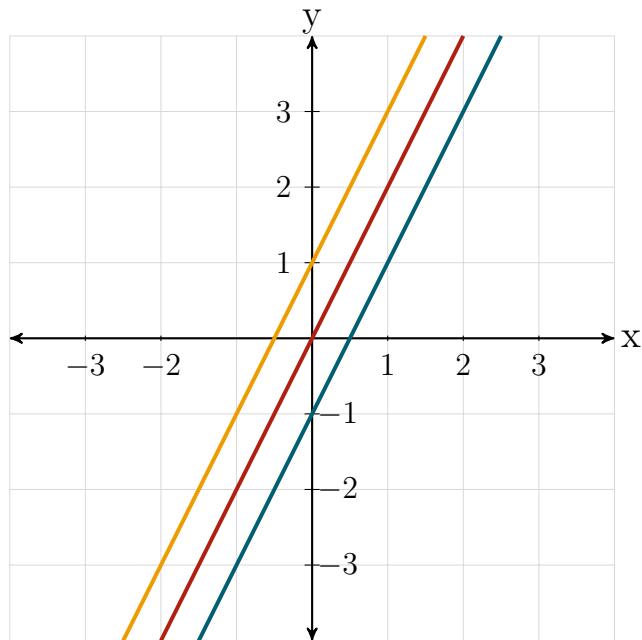


Figure 2.8: One more and now b is nailed!

Chapter 2. Pointing Fingers and Crossing Lines: The last breath of just linearity.

The graph 2.8 implies that the values of the blue line lie entirely to the right of the red line. Conversely, the ones on the yellow line are to the left of the red line. What we need to understand is by how much, as this piece of information will probably be just what we are missing to define a line. For that, let's check the values of each line when they cross the y -axis, which is the same as calculating the values of y when $x = 0$:

- $y = 2x + 1 \rightsquigarrow y = 1$
- $y = 2x - 1 \rightsquigarrow y = -1$
- $y = 2x \rightsquigarrow y = 0$

It's not coincidental that the values for $x = 0$ correspond to the term we add or subtract from mx in all the above equations. The point where each line intersects the y -axis, combined with the term mx , sufficiently defines a line. Let's denote the intercept point where $x = 0$ as b . This gives us the equation for a fully defined line:

$$y = mx + b \quad (2.2)$$

Let's dress that equation up 2.2, just as if it was prom night:

$$f(x) = 2x - 1 \quad (2.3)$$

Before we proceed, we need to address a crucial piece of missing information: how does $f(x)$ behave when b changes sign? While it's clear that this change results in the graph crossing the y -axis at a different point, the situation becomes more complex with a negative slope. Let's consider a function $g(x)$ to explore this scenario:

$$g(x) = -2x - 1$$

If we plot both $f(x)$ and $g(x)$ this is what we will see:

2.1. Swipe Right for co-domains: **Functions**.

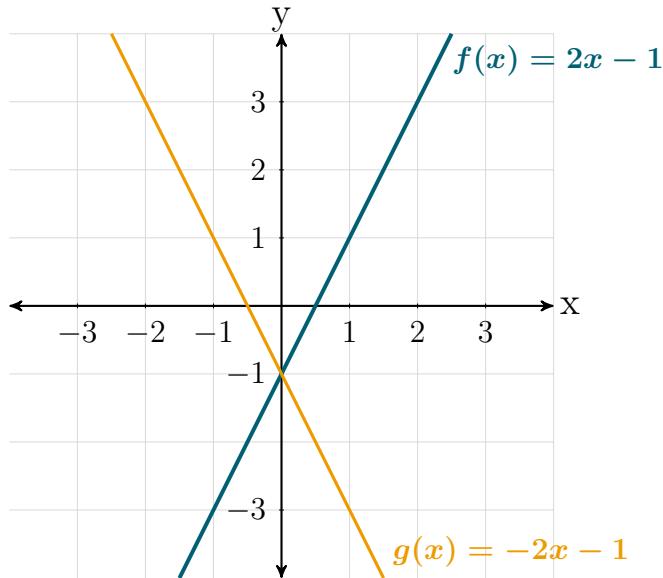


Figure 2.9: The impact of the m 's on the line.

The slope signal will determine the line's orientation: positive values of m will mean that instances of $f(x)$ increase from left to right. On the other hand, negative slopes indicate decreasing values of $g(x)$ if considering the same orientation.

We have created a mathematical framework that gives us a relationship between an input and an output. If these frameworks follow some conditions that we will explore, they are called a function.

2.1 Swipe Right for co-domains: Functions.

Allow me to introduce you to a function. The word on the street is that a function can be a formula, table, or graph, each representing a relationship between inputs and outputs. Now, as always, we have names for these inputs and outputs; they are called Donald and Duck. I sure wish they were, but, in reality, they go by domain and codomain. The domain contains all possible inputs, whereas the codomain includes all the possible outputs.

It is time for an example, so let's explore polyethism. Polye-

thism is the doctrine that encourages sexual relationships inside bushes ...no, no, and no, it is not that. It is a term used for a non-reproductive specialization within a colony of social organisms. For example, bees create divisions of labor with age polyethism, meaning that workers perform tasks depending on their age. Young bees don't leave the nest. Instead, they clean and attend to the queen.

As they age, they transition to tasks beyond the nest center, like comb building and food processing. Finally, when they are older and wiser, they become foragers, venturing to collect nectar, pollen, water, and propolis, a catchy name for a shot of rum. Actually, propolis is tree resin used for sealing and protecting the hive, almost like our cement.

It's fascinating how they prioritize wisdom over youth, unlike us humans. But let's shift our focus back to the foraging process. These black and yellow creatures must fly out and return food to the colony. Otherwise, they get beheaded with a tiny wax-made guillotine. Actually, they are social and cooperative beings, so if one is struggling, the other will help. To collect nectar, they need to go into a flower, so let's try to model the distance covered over time when a bee comes down to the flower and exits it.

To model the path of a bee's flight, we initially considered a straight line for simplicity. However, capturing the nuances of a bee's movement over time, a flight that isn't uniform in speed or direction will demand a more complex approach. We'll continue to let y represent the distance, while the x -axis will denote time. Our focus will be on the domain where time is positive, as negative time values, which would suggest a look into the past, are not of interest to us.

The task at hand is to express a distance that can only be positive or zero and varies non-linearly with time. Moreover, we need to illustrate the bee's approach to a flower, where the distance decreases until it reaches the flower, and then, post-pollination, as the bee departs, the distance increases again. A \cup -shaped curve suits

2.1. Swipe Right for co-domains: **Functions**.

our purpose here, setting aside the V shape for now, with plans to explore it later.

A promising candidate for our model is the function $f(x) = x^2$. This function inherently generates positive y values due to the squaring of x , ensuring that the distance cannot be negative. Furthermore, the relationship between y and x isn't proportional; the squaring of x introduces a non-linear element. To confirm that it's the \cup shape we're looking for, consider the points: $f(-2) = 4$, $f(0) = 0$, and $f(2) = 4$. Although we're not accounting for negative time values in our model, these points confirm the curve's shape. Now, let's proceed to plot this function and examine its suitability for our bee's flight path:

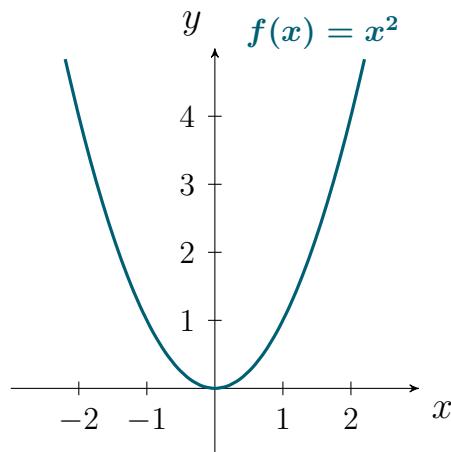


Figure 2.10: The first attempt of a model for a non-linear bee flight.

Where are my manners? Allow me to apologize. Welcome to the world of non-linearity! Here, we stop exclusively using straight lines and introduce some curves.

Looking at the graph 2.10, we can see we are close but not there yet. If x represents time, unless we have some time-traveling vehicle such as a flying DeLorean here. (If you have never watched Back to the Future, do yourself a big favor and do so. Not right now, after the calculus...! Oh dear, I feel that I've lost a few readers right there, although the movies are great!). Anyhow, we can't have a negative time period in this scenario. To overcome this, we will do two things, define a new domain, as previously mentioned and also translate the

function to the right. Within the mathematical world, “translation” is a word we use to represent movement. In this case, we will do a horizontal translation to the right. The function will travel right on the x -axis, whilst keeping the same shape. It is equally possible to do this on the y -axis. For that, we would move the function up or down.

We can think of functions as shapes that allow us to model real-life dynamics or phenomena. They are tools that will aid us in our machine-learning journeys, and the more we know about them, the better equipped we will be. Think of it this way, if a gardener has to dig a hole, she won’t use a lawnmower. She will go for a shovel. Here the phenomenon is the hole that has to be dug, and the function is the shovel. Now, there is an entire selection of shovels. Some are longer or wider than others. The gardener’s knowledge will determine the optimal tool for the task ahead. We also have a selection of functions within a class of functions. For example, with x^2 , we can either add a constant to $f(x) \rightarrow f(x) + c$ or alter the quadratic term, so x^2 becomes $(x + c)^2$, where $c \in \mathbb{R}$. Welcome back \mathbb{R} . I am unsure if we missed you, but you are always useful, so stick around. Throughout the book, we will study different calculus topics by introducing new function shapes and studying how they behave.

To fully understand the impact of the constant c , let’s consider two examples: $c = 3$ and $c = -3$. By having one value for c and its symmetrical, we can better understand how the original function $f(x)$ gets altered. OK, let’s set this up:

$$f(x) = (x + c)^2$$

Let’s create a box with the three of the possible iterations we are speaking of:

- $f(x) = x^2$
- $c = -3 \rightarrow g(x) = (x - 3)^2$
- $c = 3 \rightarrow h(x) = (x + 3)^2$

2.1. Swipe Right for co-domains: **Functions**.

It would be beneficial to view all those functions on the same plot. After all, we are looking for a way to understand what happens when we alter a quadratic term of a function, so let's plot all of them:

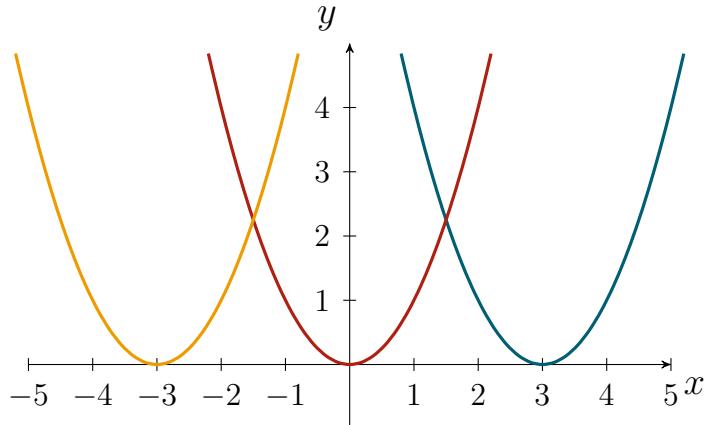


Figure 2.11: Playing with the Argument: x^2 's dance moves.

Negative values of c mean that we will subtract c from the quadratic term of $f(x)$, so we will translate the function c units to the right. Conversely, altering the quadratic element from x^2 to $(x + 3)^2$ will have the opposite effect with the translation being to the left this time. Now in the cases where we add c to the function $f(x)$ the translation will occur along the y -axis:

$$f(x) = x^2 + c$$

We will make use of another box for the new iterations:

- $f(x) = x^2$
- $c = -3 \rightarrow g(x) = x^2 - 3$
- $c = 3 \rightarrow h(x) = x^2 + 3$

In this case, when we go from x^2 to $x^2 + c$, the function $f(x)$ will shift up c units for positive values of this guy and will be moved down the same amount when this magnitude has a negative sign. We can analyse this behaviour in the plot below:

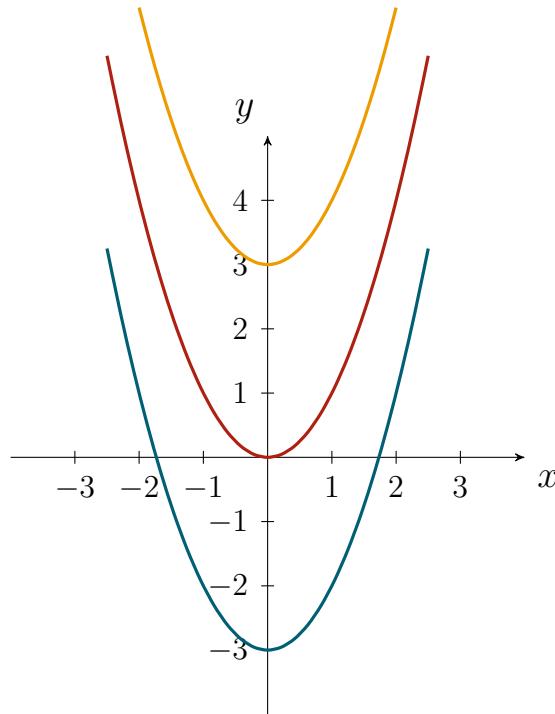


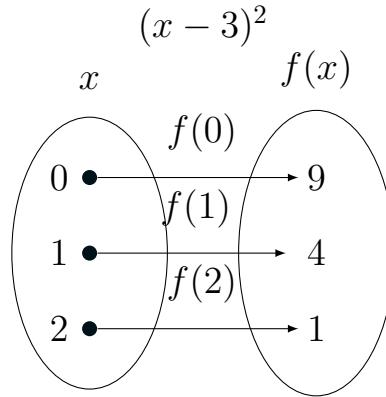
Figure 2.12: Jumping up and down: The c 's dance.

That translation to the right will help us improve our model. For example, if we pick $(x - 3)^2$, the blue function on graph 2.11 for our function $f(x)$, we have eliminated the negative time period. In this case, the bee lands in the flower, whose location is $(0,3)$ at time three instead of 0. To fully assure ourselves that $f(x)$ represents what we need, specifically by excluding any negative time period, we can tailor both the domain and the codomain accordingly. Let's start with the domain, all values must be positive, or 0. The symbol 0 is an old fellow; it was introduced into arithmetic operations in the 7th century by a mathematician called Brahmagupta. There was a need to represent nothing, imagine having the necessity of representing the void. What an excellent problem to have! Back to the bee, it can't fly forever; the flight duration is just 6 minutes. Thus, our domain is designated by $[0, 6]$. As for the codomain, the yellow and black mean machine departs from the beehive that is at 9 meters distance from the flower to later return to it. Because the lowest value for length is 0 when the bee is in the flower and distances can't be negative, the codomain is $[0, 9]$:

2.1. Swipe Right for co-domains: **Functions**.

$$\begin{aligned} f(x) : [0, 6] &\rightarrow [0, 9] \\ x \rightarrow (x - 3)^2 \end{aligned} \tag{2.4}$$

There is more than one way to represent a function and 2.4 is one of them. Another example, is to have the same mathematical concept displayed using the figure below:



In the representation above, we have a few domain elements and the codomain's corresponding values, also known as images. It is time for us to see the face of our new function $f(x)$:

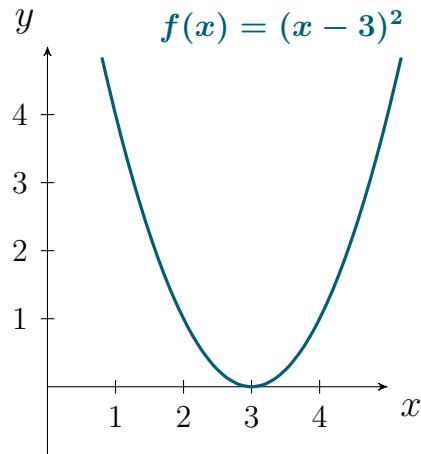


Figure 2.13: Finally, a model for the bee's flight.

The graph 2.13 represents our final function. On the x -axis, we have the time in minutes that the bee's flight takes. The flower is located at the point $(3, 0)$. After pollinating it, the bee leaves it and flies away to the beehive that is at 9 meters away.

Foraging bees gather nectar from flowers and store it in their honey stomachs before returning to the hive. Once there, they transfer the nectar to other bees by regurgitating it into their mouths. This nectar is then passed among the bees multiple times, mixing with enzymes from their salivary glands. This mixing breaks down the complex sugars in the nectar into simpler ones, making them more digestible. Finally, the nectar is stored in honeycomb cells. Here, bees fan it with their wings to evaporate excess water, transforming the nectar into honey.

Just as we analyze the intricate process of honey production, we can apply similar analytical methods to understand other aspects of a bee's behavior. Therefore, our function $f(x) = (x - 3)^2$ represents the distance travelled per unit of time. One of the things we can extract from this graph is the velocity at which the bees fly. Velocity is defined by the rate at which an object covers distances. It is a ratio between the distance and the time. In fact, understanding the velocity at which bees fly can help estimate their pollination efficiency, which then can help in optimising crop yields.

Chapter 3

Changing Times and Tangent Lines: The Derivative.

The derivative measures how much a function's output changes in response to a small change in its input. While there's more nuance to this definition, let's unravel it progressively as we introduce more concepts. Given that a derivative signifies the rate of change in output relative to a change in input, it aligns well with the concept of velocity, which is defined as the rate at which an object covers distance over a given time. Therefore, the derivative is apt for representing the velocity of the bee's flight. To illustrate, consider two points: $(1, 4)$ and $(2, 1)$. These coordinates give us two distinct positions and times, enabling the calculation of the bee's velocity between these points. This can be done by calculating the displacement over the time interval:

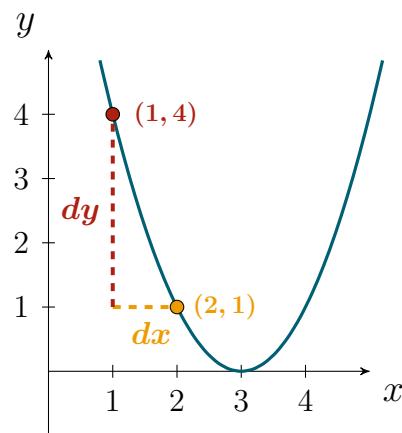


Figure 3.1: Two tiny changes: One on the x -axis and another on the y -axis.

There is some new notation on the plot, we have some d 's, but they are still not for Donald Duck, DAMN IT! In mathematics, the d 's refer to difference or change, so dx is the change in x , whereas dy is the change in y . We can then define the rate of change as:

$$\frac{dy}{dx} \tag{3.1}$$

We can also represent the change in mathematics by the Greek letter Δ , so that expression 3.1 can also be written as:

$$\frac{\Delta y}{\Delta x}$$

When we defined the equation for the straight line, we mentioned "change" several times when calculating the slope. Let's explore this idea again with $(2, 1)$ and $(1, 4)$. It turns out that:

$$\frac{dy}{dx} = \frac{4 - 1}{1 - 2} = -3 \tag{3.2}$$

We can calculate the distance travelled, or dy , by subtracting the origin's y coordinate from the destination's y coordinate. Similarly, the time taken to cover this distance, or dx , is the final time minus the initial time. When we divide these two magnitudes, we have our velocity or the rate of change. We have these calculations in equation 3.2, and it comes out as a velocity of -3. Velocity is a vector, defined by both direction and magnitude. Therefore, a velocity of -3 indicates that the bee is flying to the left or downwards. For example, if we were to pick two points on the other side of the plot, say $(4, 1)$ and $(5, 4)$, the velocity would come to:

$$\frac{dy}{dx} = \frac{4 - 1}{5 - 4} = 3$$

A positive value means that the bee leaves the flower, making its flight to the right or upwards. More generically, if we consider two points A and B such that: $A = (a, f(a))$ and $B = (b, f(b))$ we can write equation 3.2 as:

$$\frac{dy}{dx} = \frac{f(a) - f(b)}{a - b} \tag{3.3}$$

The equation 3.3 is the same as the one we got to when we defined the slope; the only difference is in the notation. Before we defined $A = (a_1, a_2)$ and $B = (b_1, b_2)$ and now $A = (a, f(a))$ and $B = (b, f(b))$ but the formula is the same. The fact of the matter is that, we don't necessarily need two points to find the equation to calculate a derivative; if we are after a measurement of change, all we need is a starting point and a quantity to gauge how much both x and y change. This same quantity can be defined as h , so our setup becomes:

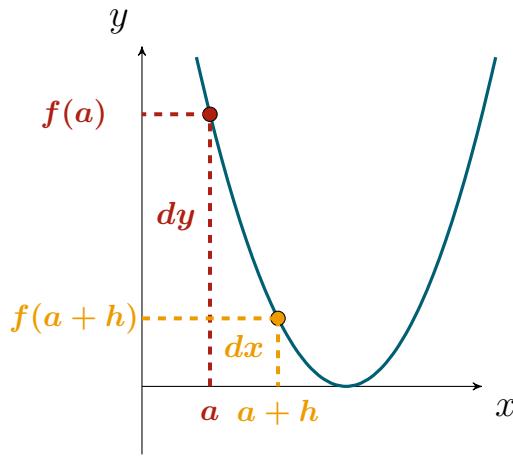


Figure 3.2: Where we represent the tiny changes by h .

The quantity h denotes the change in x , which is often referred to as the independent variable. On the other hand, $f(a + h)$ represents a change in y . Since y is determined by the values of x , it's termed the dependent variable (and yes, you can think of it as still living with its "parents", x). With these clarifications, we can express equation 3.3 as:

$$\frac{f(a + h) - f(a)}{a + h - a}$$

Which comes out as:

$$\frac{f(a + h) - f(a)}{h}$$

Thus for a generic point x we can derive the expression:

$$\frac{f(x + h) - f(x)}{h} \tag{3.4}$$

At first glance, it seems that we haven't got anything new, as equation 3.4 still represents the slope of a line that crosses the points $(x, f(x))$ and $(x+h, f(x+h))$, but now we have an h whose variation is yet to be understood. We can experiment with this new term and see what emerges; for example, we can reduce it and bring the two points closer together. As we stand, we can leave the a 's behind, and work with a generic point x :

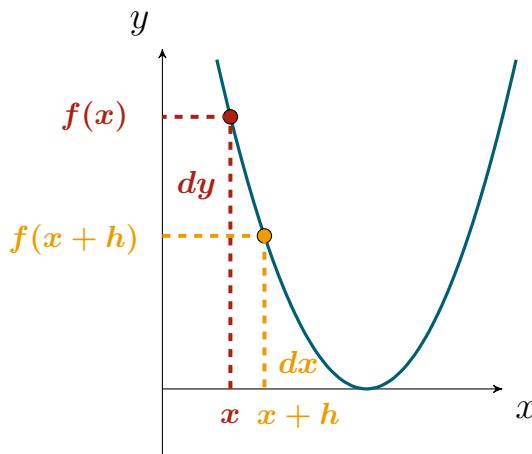


Figure 3.3: When we reduce the size of h .

Since h represents a measurement of change, we can keep making it smaller and smaller, turning the distance between x and $x+h$ a number very close to 0. Initially, we were computing the rate of change between two points, but now, because h can be such a tiny number, we can probably calculate the rate of change of a single point. Excuse me? Well, let me introduce you to the complete definition of a derivative. Previously, we stated that the derivative measures how much a function changes as its input varies, wording that defined the concept of the rate of change. Although the definition provided is accurate, it's important to note that when h is a number close to 0, we have the ability to measure an instantaneous rate of change. This is essentially the same as computing the derivative at a single point.

This is a bit of a brain-twister, I know. The rate of change in a single point? When we said we needed two points to calculate it? But wouldn't it be useful? It sure would! By defining it like this, we can calculate derivatives at a given point, and with this, we can have

functions that represent rates of change. Oh boy, this can open up a new realm of opportunities, so let's introduce all this information into the formula we were constructing, equation 3.4.

We want to understand what happens to the rate of change when h is in the proximity of 0, meaning that h is a finitely small number that we will let tend to 0. This can be defined by:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h} \quad (3.5)$$

There are two new players in town: $f'(x)$ represents the derivative function, whereas the "lim" denotes a limit. This new concept of limit captures the essence of proximity. What happens to that expression when we get closer and closer to 0? This same question is what we represent by the limit in mathematics. If this is still foreign to you, don't worry, it was to me at one point also, and I trust it won't be for long. We will explore both of these concepts, starting with the limit, as we now have a definition of a derivative that is dependent on it. Since we are in exploration mode, we can see what comes out of 3.5 if we plug our function $f(x)$ into that formula:

$$f(x + h) = (x + h - 3)^2$$

$$f(x) = (x - 3)^2$$

Replacing this in the equation 3.5, we get:

$$f'(x) = \lim_{h \rightarrow 0} \frac{(x + h - 3)^2 - (x - 3)^2}{h} \quad (3.6)$$

There's quite a bit we can do to make 3.5 more friendly; for instance, the expansions of those two terms $(x + h - 3)^2$ and $(x - 3)^2$ might get us somewhere. OK, in case anyone has forgotten how to do this, I will do it for one of them, and yes, I will do it for the one with the h ; by now, we shouldn't fear a letter though; they are just there to represent numbers. More work, less talking, I can appreciate that:

$$(x + h - 3)^2 = (x + h - 3)(x + h - 3)$$

Applying distributivity, we can expand that expression:

$$(x + h - 3)(x + h - 3) = (x + h)(x + h) - 3(x + h) - 3(x + h) + 3^2$$

Well, why not do the same, but now for $(x + h - 3)(x + h - 3)$?

$$(x + h - 3)(x + h - 3) = (x^2 + 2hx + h^2) + (-3x - 3h) + (-3x - 3h) + 9$$

And finally:

$$(x + h - 3)(x + h - 3) = x^2 + 2hx + h^2 - 6x - 6h + 9$$

The calculations will be omitted for the other term, the $(x - 3)^2$, as they are very similar to what we just did. So, equation 3.6 takes the following form:

$$f'(x) = \lim_{h \rightarrow 0} \frac{(x^2 + 2hx + h^2 - 6x - 6h + 9) - (x^2 - 6x + 9)}{h}$$

Then in turn:

$$f'(x) = \lim_{h \rightarrow 0} \frac{(x^2 - 6x + 9 + 2h(x - 3) + h^2) - x^2 + 6x - 9}{h}$$

After some work:

$$f'(x) = \lim_{h \rightarrow 0} \frac{2h(x - 3) + h^2}{h}$$

Finally:

$$f'(x) = \lim_{h \rightarrow 0} 2(x - 3) + h$$

We need to understand what happens to h when its values are in the neighbourhood of 0. We can replace this with 0 and end up with the following:

$$f'(x) = 2x - 6$$

Substitutions are one of the techniques that we use to solve a limit. As the name implies, we replace the variable on the limit with the value it is tending to. Unfortunately, this is not always possible, but for now, let's enjoy that it worked for our function $f(x)$. However, be sure that sooner rather than later, we will have a bastard limit for which that substitution won't work.

Well, but for now, after the substitution, we ended up with a function that is familiar to us, the equation of a line:

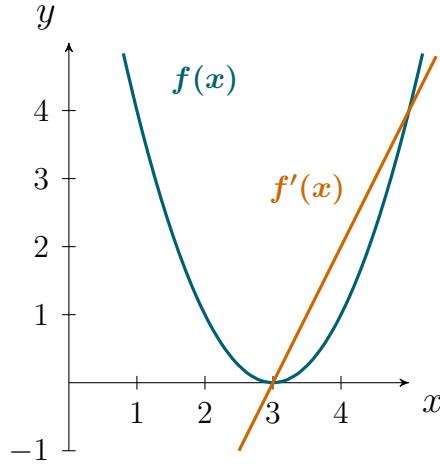


Figure 3.4: A family photo: A function and its derivative.

On the above graph 3.4, we have our function $f(x)$ represented by the blue curve while its derivative, $f'(x)$, is represented by the orange line. There are some interesting dynamics between the sign of the derivative $f'(x)$ and the function $f(x)$. When $f'(x)$ is negative, evidenced by the orange line lying below the x -axis the function $f(x)$ exhibits a decrease in its values. In contrast, the opposite behaviour is true when $f'(x)$ takes positive values, and the function $f(x)$ increases. This makes sense; if the flower location is at $(0,3)$, the bee needs to decrease velocity to land and then increase it to leave.

Derivatives carry a lot of helpful information about a function; in fact, they will be the central theme of this book as they are super important in machine learning. I would like us to keep exploring them and present to you some techniques that will help us in computing such concepts. My goal is not for us to be super calculators for derivatives but to have this concept well solidified. For this purpose, why not combine the need to learn more about these instantaneous rates of change with introducing a new type of function? Just like machine learning algorithms, bees also get more efficient with more attempts. For example, on flower recognition, bees learn to recognise flowers that provide rich nectar sources by associating color, shapes, and patterns with rewards. They don't like to mess around and are

quick learners, so the time required to perform a task decreases with the number of attempts on that same endeavour.

Let's consider y to represent the time taken to perform a task, such as recognising rewards based on flower patterns, and we use x to reflect the number of attempts. If we wish to create a simple model that demonstrates this dynamic, y and x must have opposite behaviours. As x grows, y decreases, because the bee becomes more proficient at the given task. So $f(x) = x$ or $f(x) = x^2$ won't work because they don't reflect this behaviour. We need to get creative here, so let's start by defining our domain. For the values of x , we want to study the number of attempts between 1 and 5 and given our conjecture, what this means is that if $g(x)$ represents our function, then, $g(1) > g(5)$, and $g(1) > g(2)$. Such behaviour is inversely observed in $f(x) = x$. Sorry, that is a fancy sentence to say that in $f(x) = x$, the images grow as x increases. I just felt fancy. Saying "inversely" is just suggesting that we could try the following function:

$$g : [1, 5] \rightarrow \left[\frac{1}{5}, 1 \right] \quad (3.7)$$

$$g(x) = \frac{1}{x}$$

Let's see if that works by plotting it:

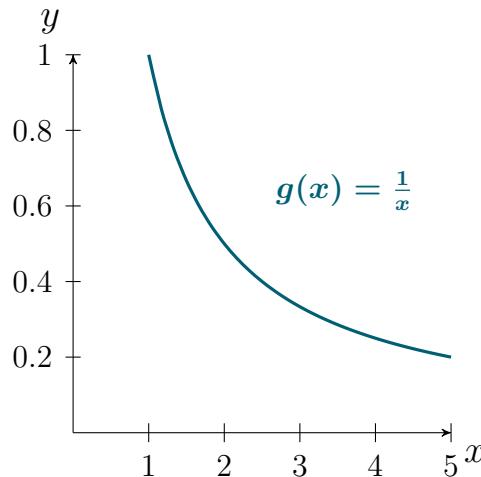


Figure 3.5: The number of attempts vs. time to perform a task.

The behavior we intended to model is well captured by the func-

tion shown in equation 3.7, as illustrated in graph 3.5. From all the knowledge we have gathered, we can understand another cool thing: the velocity at which a bee learns a skill. The process will be the same as the one we did before for the function $f(x)$, so it follows that:

$$g'(x) = \lim_{h \rightarrow 0} \frac{\frac{1}{x+h} - \frac{1}{h}}{h}$$

To get out of that mess, one strategy would be to find a common denominator on the terms that are in the top of that fraction:

$$g'(x) = \lim_{h \rightarrow 0} \frac{\frac{x-(x+h)}{x(x+h)}}{h}$$

Which simplifies to:

$$g'(x) = \lim_{h \rightarrow 0} \frac{-h}{h(x+h)x}$$

Finally:

$$g'(x) = \lim_{h \rightarrow 0} -\frac{1}{(x+h)x}$$

Again, if h is tending to 0; the function $g'(x)$ has the following expression:

$$g'(x) = -\frac{1}{x^2}$$

If you are wondering if we need to apply the definition of a derivative every time we want to compute this instant rate of change, don't worry. I bring good news. There is a set of rules that we can leverage to make our lives easier. I LOVE SOME RULES! (I wish you could sense the irony in the air).

1. You shall not drive with an uncased bear.

By this rule, the derivative of a generic function $f(x)$ is ... well, this is a joke, at the same time, it is actual law in Missouri. So be careful if you are going for a drive with your bear and happen to be there of all places. Now for the authentic list of rules.

- **Constant rule:** The derivative of a constant c is 0. Well, the proof is in the name, constant means something that does not vary, and if the derivative represents the instant rate of change, this guy has to be zero. Otherwise, mathematics explodes.

$$\frac{d}{dx}c = 0$$

- **The product rule:** Say that we have a function $z(x) = f(x) \cdot g(x)$ then its derivative can be expressed as:

$$\frac{dz}{dx} = f(x) \frac{dg}{dx} + g(x) \frac{df}{dx} \quad (3.8)$$

Why? It was on the Internet, so it must be true. Let's try and calculate the derivative of $z(x)$ via the definition:

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{z(x+h) - z(x)}{h}$$

If we replace $z(x)$ by $f(x) \cdot g(x)$ we get :

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h)g(x+h) - f(x)g(x)}{h} \quad (3.9)$$

So far, we just replaced the function $z(x)$ with $f(x) \cdot g(x)$; now we must find a way to make equation 3.9 a bit easier to digest. Taking a closer look, we have both $f(x+h)$ and $f(x)$, which are exactly the terms from the equation of the derivative definition. The same can be said for $g(x+h)$ and $g(x)$. Ideally, we would have something of the form $f(x+h) - f(x)$ and $g(x+h) - g(x)$ so we could simplify it with the definition of a derivative. Since we have the f 's being multiplied by the g 's, it will not be straightforward to get to those subtractions. But one way to do this would be, for example, if we had something of this form $f(x+h)g(x+h) - f(x+h)g(x)$. This way we could then have $f(x+h)(g(x+h) - g(x))$. Let's throw this term into the mix:

$$\frac{f(x+h)g(x+h) - f(x+h)g(x)}{h}$$

That term is zero, so it won't change anything on equation 3.9, but it will help us:

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h)g(x+h) - f(x)g(x) + f(x+h)g(x) - f(x+h)g(x)}{h}$$

OK, let's move some terms around.

$$= \lim_{h \rightarrow 0} \frac{(f(x+h)g(x+h) - f(x+h)g(x)) + (f(x+h)g(x) - f(x)g(x))}{h}$$

I feel we might get somewhere!

$$= \lim_{h \rightarrow 0} \frac{f(x+h)(g(x+h) - g(x)) + g(x)(f(x+h) - f(x))}{h}$$

We are definitely getting somewhere!

$$= \lim_{h \rightarrow 0} \frac{f(x+h)(g(x+h) - g(x))}{h} + \lim_{h \rightarrow 0} \frac{g(x)(f(x+h) - f(x))}{h}$$

Take a good look at that; if we now break those two limits, we can make the expression above look even better:

$$\begin{aligned} &= \lim_{h \rightarrow 0} f(x+h) \lim_{h \rightarrow 0} \frac{(g(x+h) - g(x))}{h} \\ &\quad + \lim_{h \rightarrow 0} g(x) \lim_{h \rightarrow 0} \frac{(f(x+h) - f(x))}{h} \end{aligned}$$

We can work with this! When brother h tends to 0 by substitution, the limit of $f(x+h)$ is $f(x)$. The other one is even easier. When h tends to 0, $g(x)$ will tend to ... yes, that's right, $g(x)$. The remaining terms are the definitions of a derivative for $f(x)$ and $g(x)$, so:

$$\frac{dz}{dx} = f(x) \frac{dg}{dx} + g(x) \frac{df}{dx}$$

Which is exactly equation 3.8.

- **Quotient rule:** If we have $z(x) = \frac{f(x)}{g(x)}$ then:

$$\frac{dz}{dx} = \frac{\frac{df}{dx}g(x) - \frac{dg}{dx}f(x)}{g(x)^2} \quad (3.10)$$

Once again, we will use the definition, just as we did for the product rule. So the equation becomes :

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{\frac{f(x+h)}{g(x+h)} - \frac{f(x)}{g(x)}}{h}$$

A first good step will be to find a common dominator (This was a typo. The word is denominator. I just thought it would like to be referred to as dominator at least once in its life, so I left it):

$$= \lim_{h \rightarrow 0} \frac{\frac{f(x+h)}{g(x+h)} - \frac{f(x)}{g(x)}}{h}$$

Which we can turn into:

$$= \lim_{h \rightarrow 0} \frac{f(x+h)g(x) - f(x)g(x+h)}{h \cdot g(x)g(x+h)}$$

We can create two limits out of that one:

$$= \lim_{h \rightarrow 0} \frac{f(x+h)g(x) - f(x)g(x+h)}{h} \cdot \lim_{h \rightarrow 0} \frac{1}{g(x)g(x+h)}$$

Let's take care of the guy on the right. When h goes to zero, that denominator is $g(x)^2$. So, one can do a similar maneuver to the one we did with the proof of multiplication to help us simplify things. This time, we will be adding the term $\frac{f(x)g(x) - f(x)g(x)}{h}$.

$$= \left(\lim_{h \rightarrow 0} \frac{f(x+h)g(x) - f(x)g(x)}{h} - \lim_{h \rightarrow 0} \frac{g(x+h)f(x) - f(x)g(x)}{h} \right) \cdot \frac{1}{g(x)^2}$$

We also split the fraction on the left into two limits, and with one more step, we can get to 3.10:

$$= \left(\lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} g(x) - \lim_{h \rightarrow 0} \frac{g(x+h) - g(x)}{h} f(x) \right) \cdot \frac{1}{g(x)^2}$$

The limits of $f(x)$ and $g(x)$ when h tends to 0, are $f(x)$ and $g(x)$ respectively. Regarding the remaining fractions, they are precisely the definition of the derivative for $f(x)$ and $g(x)$, therefore:

$$= \frac{\frac{df}{dx}g(x) - \frac{dg}{dx}f(x)}{g(x)^2}$$

Let's go! This is just what we wanted. We have three more to go, the sum rule, the subtraction rule and the power rule. Moving into the sum:

- **The sum rule:** If $z(x) = f(x) + g(x)$, the derivative can be computed with:

$$\frac{dz}{dx} = \frac{df}{dx} + \frac{dg}{dx} \quad (3.11)$$

Happily this proof is not only simple, but we can also use it to verify the subtraction rule, so:

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) + g(x+h) - (f(x) + g(x))}{h}$$

We can split that fraction and create two limits:

$$= \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} + \lim_{h \rightarrow 0} \frac{g(x+h) - g(x)}{h}$$

And there we go:

$$\frac{dz}{dx} = \frac{df}{dx} + \frac{dg}{dx}$$

- **The subtraction rule** Given that $z(x) = f(x) - g(x)$ the derivative is:

$$\frac{dz}{dx} = \frac{df}{dx} - \frac{dg}{dx} \quad (3.12)$$

One more and we are done with the rules:

- **Power rule:** When $z(x) = x^n$, we can compute the instantaneous rate of change by:

$$\frac{dz}{dx} = nx^{n-1} \quad (3.13)$$

For this particular proof, we will be using something called induction. The method of mathematical induction begins by confirming the truth of a statement for an initial value, usually $n = 1$. Next, an inductive hypothesis is formulated by assuming that the statement holds true for a general case, represented by $n = k$, where k is a positive integer.

The crux of this method lies in proving that, if the statement is true for $n = k$, it logically extends to be true for $n = k + 1$. This iterative proof allows us to conclude that the statement is valid for all subsequent positive integers n , demonstrating

the universal truth of the statement through induction. The first step is the base step, whereas the second is the inductive step. OK, so if $n = 1$ we have:

$$z(x) = x$$

We need to use the definition to calculate this derivative:

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{z(x+h) - z(x)}{h}$$

Which comes to:

$$\frac{dz}{dx} = \lim_{h \rightarrow 0} \frac{x+h-x}{h} = 1$$

Right, now let's try the induction method:

$$\frac{dz}{dx} = 1 \cdot x^{1-1} = 1$$

A good start, wouldn't you agree? The base case can be verified. Now, for the inductive step, we must assume that the following statement is true:

$$z(x) = x^k \quad \text{then} \quad \frac{dz}{dx} = kx^{k-1}$$

That statement needs to hold for $k + 1$. Otherwise, we are in big, big trouble:

$$z(x) = x^{k+1} \quad \text{then} \quad \frac{dz}{dx} = (k+1)x^k \quad (3.14)$$

Let's try and be a smart-ass, I am usually opposed to this type of behaviour, but I do have to admit that sometimes it pays off. What if we write x^{k+1} as xx^k and use the derivative multiplication rule? After all the work we did on that demonstration, we might as well use some of it, so:

$$\frac{d}{dx}(xx^k) = \left(\frac{d}{dx}x\right)x^k + \left(\frac{d}{dx}x^k\right)x$$

Those derivatives are both the base case and the inductive step, so:

$$\frac{d}{dx}xx^k = x^k + xkx^{k-1} = x^k + kx^k = (k+1)x^k \quad (3.15)$$

That formula is just what we were looking for. It perfectly matches equation 3.14. The caveat is that this proof only works for values of $n \geq 0$. In reality, the power rule works for any value of n . This means that, for negative exponents, it can also be verified. The proof is not here because it requires combinatory, a concept that we will only learn in the next volume. However, it so happens that we have the perfect candidate to try out this rule!

$$g(x) = \frac{1}{x} = x^{-1}$$

We can also have $g(x)$ defined with a negative exponent and if we apply the power rule, the derivative will have the following shape:

$$g'(x) = (-1)x^{-1-1} = -x^{-2} = -\frac{1}{x^2}$$

Now that we have the methodology to make our lives easier, let's analyse both the function $g(x)$ and its derivative $g'(x)$:

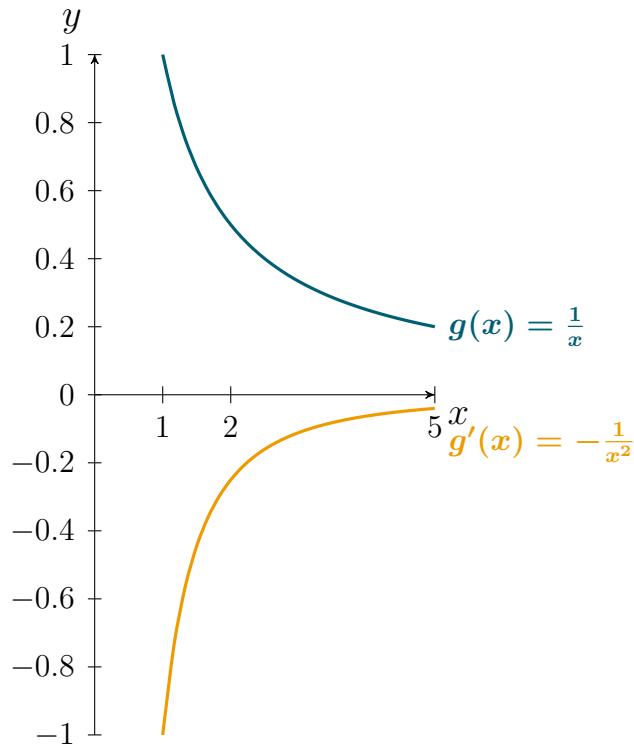


Figure 3.6: Our second family photo: Another function and its derivative.

We now have derivative $g'(x)$ that is always negative and a function $g(x)$ whose values decrease in the domain we have considered.

Negative derivative values imply that each additional attempt requires less time, indicating an increasing speed of learning. With $g'(x)$, we can indeed calculate this exciting concept of an instantaneous rate of change! No? It did not excite you that much? Oh well, at least the calculations are straightforward; for example, if $x = 2$:

$$g'(2) = -\frac{1}{4}$$

When we defined the equation of a line, we stated that the slope was a rate of change. This same concept came along again when we were defining the derivative. We followed the same strategy of establishing two points to calculate from, just as we did with the slope. Then we went a bit further and introduced a quantity h that would reflect the distance between two points, and not only that, we said that guy could get very close to zero. With this, we defined an instantaneous rate of change, the rate of change of a function at a given point, just like the one we calculated for $g'(2)$. We used slopes to define lines, but they were also the starting point for understanding a derivative, a concept that represents instantaneous rate of change. The curious part now is to understand whether this instantaneous rate of change will somehow be part of the definition of any particular line. Let's experiment; for that we will plot $g(x)$ and the point $(2, \frac{1}{2})$:

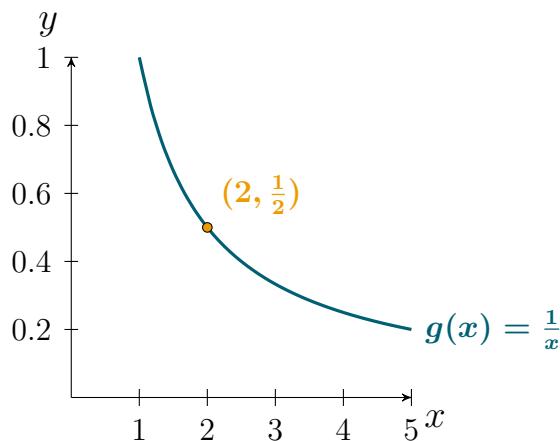


Figure 3.7: The start of something new, but so far just a point on the plot.

What's new Jorge? A curve and a point? Scotland's national animal is a unicorn! Sorry, I panicked and just threw in a random fact. The reality is that we are going to get somewhere with the experiment. I wonder what will happen if we zoom in on the point:

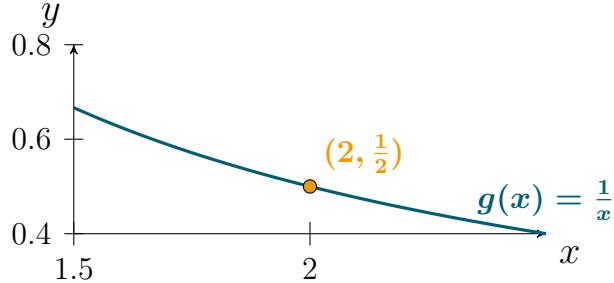


Figure 3.8: Perhaps if we zoom in, we will find something.

All of a sudden, that curve got a bit less curvy; it is now starting to resemble a line; let's zoom in again:

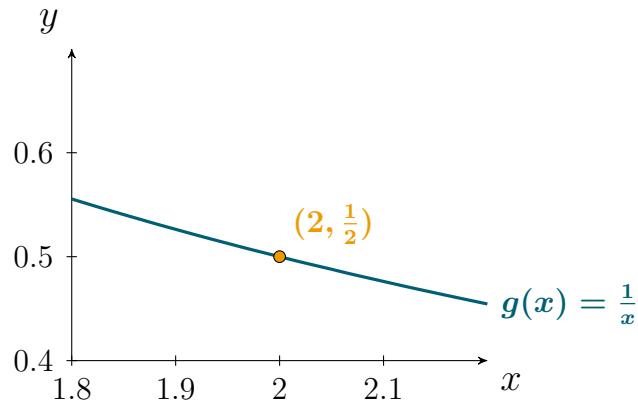


Figure 3.9: One more zoom? Sure, why not?

Now, we will pick two points and calculate a slope. Say $A = (1.9, \frac{1}{1.9})$ and $B = (2.1, \frac{1}{2.1})$. We can calculate the slope of the line that passes through those two points:

$$m = \frac{\left(\frac{1}{2.1} - \frac{1}{1.9}\right)}{2.1 - 1.9} \approx -0.268$$

What about if we zoom in once more and calculate the slope of the line again?

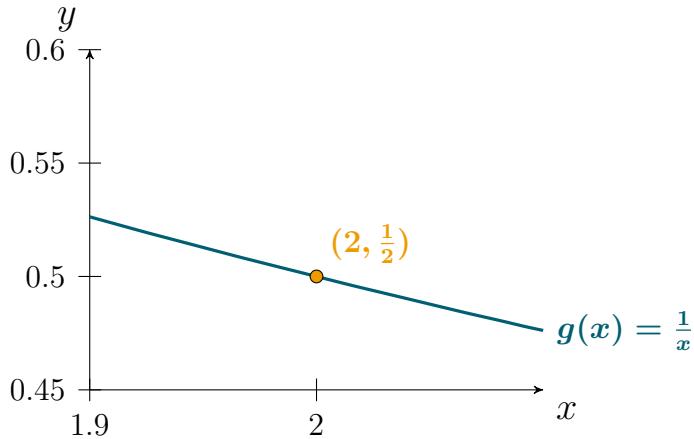


Figure 3.10: I feel one more zoom will get us somewhere.

This time we will consider the points, $C = (1.95, \frac{1}{1.95})$ and $D = (2.05, \frac{1}{2.05})$. With this the slope value is:

$$m = \frac{\left(\frac{1}{2.05} - \frac{1}{1.95}\right)}{2.05 - 1.95} \approx -0.250$$

I can spot a trend! Recall that when we calculated the instant rate of change of the point $(2, \frac{1}{2})$, we found it to be $-\frac{1}{4}$, or -0.25 . It is no coincidence that when we zoom in on the plot on that specific point, not only does the curve seem to be getting straighter, but also, if we keep decreasing the distance between the point from which we calculate the slope, the value of this will get closer and closer to -0.25 . The reason is that the derivative at a point gives us the slope of the line tangent to the plot on that exact location:

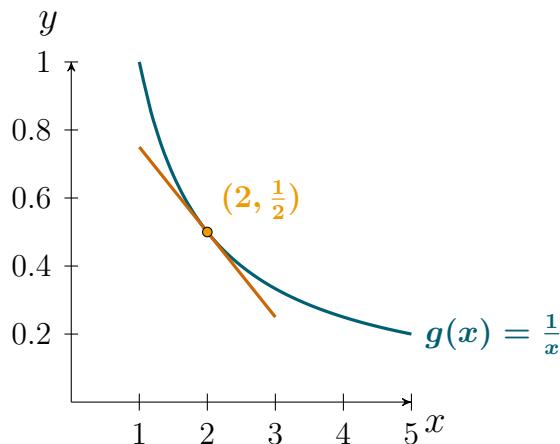


Figure 3.11: Is that a tangerine? Ah, a tangent line!

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

A tangent line is a geometric form that touches the function only at that specific point. If we keep doing this exercise of zooming and calculating a slope until... I am ready to drop it, 3,2,1... INFINITY, we will have a tangent line.

3.1 The Tale of the Eight with Wobbly Knees: **Infinity**.

I will take the liberty of doing something new. I will start this section with a little poem.

*Infinity, you son of a bitch
Your lack of meaning made my skin itch
An endless road, a boundless sea
Infinity, you frightened me*

*Small becomes all, curves become blocks
Just the thought of you, made me shit my socks
But then, in every direction you brought light
So now we are friends and no longer fight*

My hope is to get away with "not bad for a mathematician". Now let's get back to business. Infinity is indeed an abstract concept, a notion without end but at the same time a vital component in calculus. The idea of infinity allowed us to create local linearity by zooming in an infinite amount of times onto a curve, to the point where it eventually became straight; that created the possibility to define a tangent line to a function. Infinity also plays a role in determining the area of specific shapes.

For example, bees are around 10 to 20 millimeters long, and, instead of a skeleton, they have something called an exoskeleton; a term I find infinitely cooler and, frankly, I'm a very envious. An exoskeleton is a hard outer covering that provides protection. Sup-

pose we wish to calculate the volume of a bee. To do so, let's select a geometrical object resembling a bee shape, such as a cylinder:

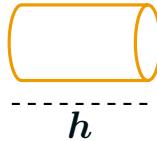


Figure 3.12: A bee, well, nearly a bee.

With a measuring tape in hand, we could measure the length of the cylinder and the perimeter of its base. However, to compute the volume, we require an additional crucial detail: the area of the circular base of the cylinder. This necessitates knowing the radius of the circle that forms the base, denoted by r :

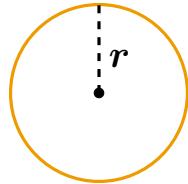


Figure 3.13: It is what is says on the tin: A circle.

Let's denote the perimeter as P . This nomenclature may lack originality, but, much like my name, Jorge, and I managed to survive, so P will do. Sticking to a traditional view of the circle leaves us at an impasse when calculating its area (assuming we don't know the formula for the area of a circle), but a shift in perspective could prove fruitful. For example, imagine dividing the circle into four equal parts:

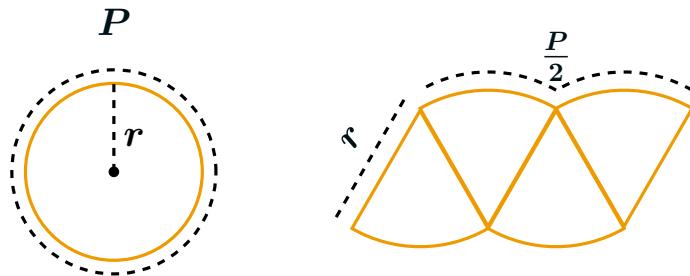


Figure 3.14: A circle with a perimeter, and a sliced circle.

Because we defined the perimeter of our circle to be P , the length of the top side of our new representation of this same shape is $\frac{P}{2}$;

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

this is because we have two pieces of the circle facing up and two facing down. What stops us from representing it with eight pieces instead of four? Let's check:

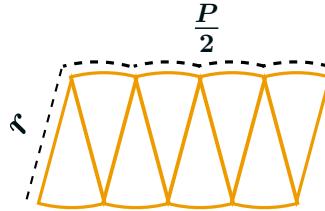


Figure 3.15: Same circle, but more slices.

Those dashed lines on the top of the figure 3.15 got a bit straighter when we increased the number of slices. You probably know where this is going, but let us proceed by dividing into 24 segments to see if this trend continues:

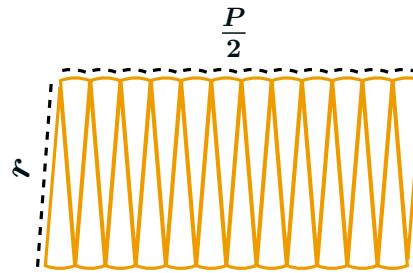


Figure 3.16: We have lost our minds, slice it more!

As expected, the dashed lines are even straighter, and, if we carry on slicing the circle until infinity, we will end up with a rectangle:

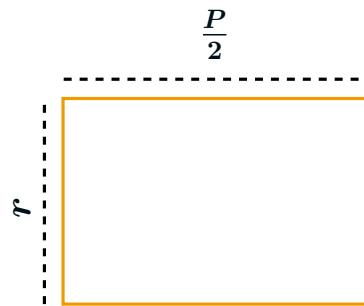


Figure 3.17: Maybe, we might have sliced it too thin.

And that area, we know how to calculate:

$$A = r \frac{P}{2}$$

A is the area that we were looking for. So, the volume of the cylinder that is the approximate volume of a bee comes to:

$$V = r \frac{P}{2} h = \pi r^2 h$$

Indeed, a circle's area is πr^2 , and its perimeter equals $2r\pi$. By substituting P with $2r\pi$, we arrive at the familiar volume equation of a cylinder. But where did that guy come from? A T with an extra leg...? Well, it turns out that infinity is again here to assist us! To understand how this fellow can help us again, let's create an hexagon.

Honeycombs are composed of hexagons. Somehow bees worked out that this particular shape allowed them to maximize space while using the least amount of material. The hexagons offer stability and strength to the honeycomb while providing symmetry and uniformity, making this shape ideal for storing honey or pollen and developing larvae.

So let's consider one hexagon shape, and as we are trying to find a relationship between π and a circle, we will encircle this guy:

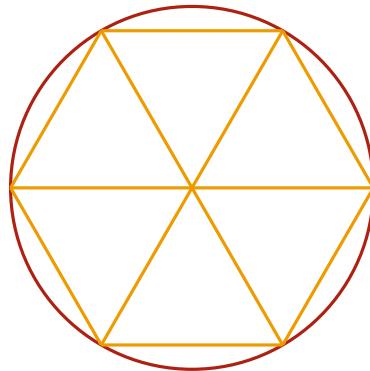


Figure 3.18: What could be a cult symbol is just a hexagon inside a circle.

In a regular hexagon inscribed in a circle, its perimeter is always less than the circle's circumference. We can divide the hexagon into six equilateral triangles, with the side length of each triangle equal to the radius of the circle, denoted by r . Therefore, the hexagon's perimeter, being the total of its sides, is exactly $6r$, so:

$$2\pi r > 6r$$

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

Which in means that:

$$\pi > 3$$

Alright, we have a lower bound for π ; this is now calling for an upper limit. This way, we can keep π nice and cozy, not doing much thinking, entertained with TV shows, hypnotized with dopamine releases of sugar and colorful screens, outraged with subjects that it does not understand, somehow part of a cult that makes it blind to seeing things objectively.

Oops, sorry, got sidetracked there... something else must have crossed my mind... anyway, upper bound! Well, we can draw another hexagon but this time, outside the circle:

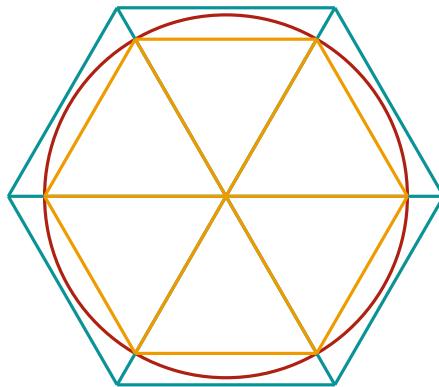


Figure 3.19: Entrapment: the same circle in the middle of two hexagons.

On this occasion, say that the sides of the blue hexagons are of size $1.1r$; this way, the perimeter of this guy is equal to $6.6r$, meaning that, for the upper boundary, we have the following:

$$6.6r > 2\pi r$$

So:

$$3.3 > \pi$$

Cool, so we have our first interval for π :

$$3.3 > \pi > 3$$

Good! We know that $\pi \approx 3.14$, so that interval works quite well. We can increase the size of the polygon an infinite number of

times while getting the outside shape closer and closer to the circle. This way, we will get a better approximation for the value of π . For example, if we consider a polygon with 24 sides:

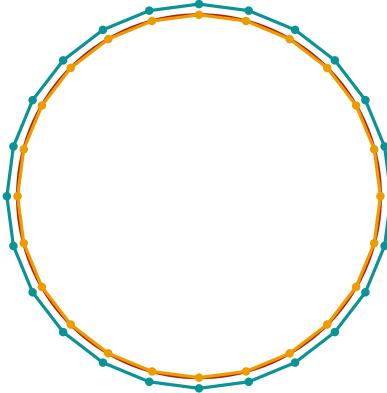


Figure 3.20: See you later hexagons and welcome circles.

The yellow polygon closely resembles a circle. No need for extensive arithmetic here; Archimedes already did the heavy lifting with a 96-sided polygon:

$$3.141 < \pi < 3.143$$

Pretty good! So far, we've encountered two scenarios where infinity is crucial: achieving local linearity and determining areas and volumes. The concept of local linearity emerges when we zoom into a curve until it seems straight. With this came the phenomenon that every curve can be approximated as a straight line when we zoom in sufficiently. This revelation is pivotal, as it leads us to see how to use the slope of a tangent line to a function at a specific point. When we first delved into the slope, it became apparent how much we could infer from it. A negative slope suggests a decreasing behavior in a function, while a positive one indicates growth. The tangent line, which often closely traces the function, provides this slope.

Thus, by understanding the sign of the slope of a tangent, which is the same as calculating the instantaneous rate of change at a point, we can quickly discern the behavior of the function around that exact location:

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

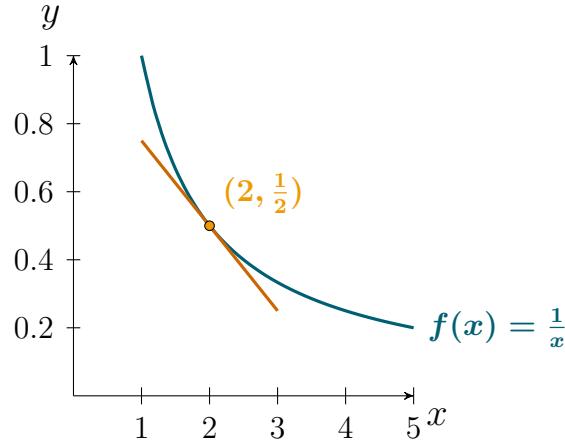


Figure 3.21: Oh, the tangerine is back... hang on, no, it is the tangent!

The slope of the red line, calculated by the derivative at $x = 2$, resulted in -0.25 , indicating a negative value. It is possible to see that the function around that point is decreasing. Identifying these behaviors is a paramount task in machine learning, as it will allow us to identify points where functions are at their lowest values. For such points to exist, the function must transition from decreasing to increasing, a change that the derivative's sign can help us detect. With irregular shapes (shapes with sides of different lengths and angles), infinity, allowed us to break down these shapes into infinitesimally small pieces, letting us compute the area or the volumes of these shapes.

I have good news and bad news. The good news is that infinity has more applications, and we will explore a few more of those. Regarding the bad news, infinity has more applications, and we will explore a few more of those, ha! Infinity can be directional, taking on $+\infty$ or $-\infty$ to represent vast positive or negative quantities. Let's revisit our earlier function, equation 3.7, but now extend its domain to $\mathbb{R} \setminus 0$:

$$\begin{aligned} g(x) : \mathbb{R} \setminus \{0\} &\rightarrow \mathbb{R} \\ x &\rightarrow \frac{1}{x} \end{aligned} \tag{3.16}$$

As usual, we will be better off with a visualization:

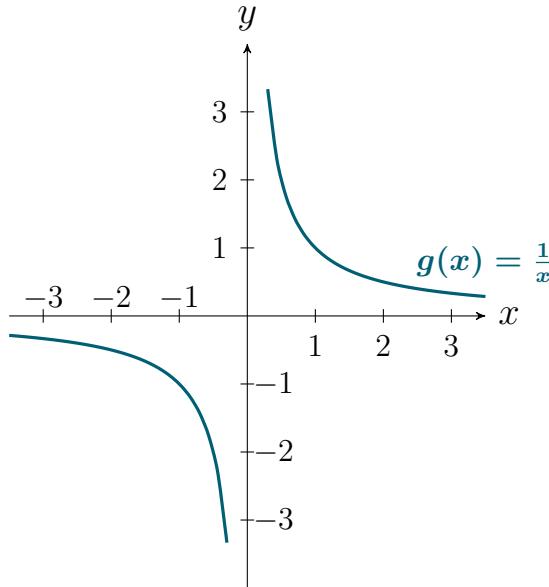


Figure 3.22: The full picture of $\frac{1}{x}$

When observing our function as it nears the x and y -axes, some intriguing behavior emerges. To dive deeper into this, let's rekindle our relationship with the idea of a limit. While we still need to formally define this 'proximity' concept, let's navigate it with what we already understand. It's similar to dating someone without having introduced them to our parents: things are getting serious, but there's still that final step awaiting.

Rest assured, though, we're not breaking up with the limits any-time soon. Instead, we'll become even more familiar with it. Stay with me, and together we'll clarify this idea.

Speaking of proximity, let us consider the behavior of our function as x increases without bound, detailed in 3.16. As x approaches positive infinity, $g(x)$ tends towards 0 from the positive side, yet crucially, it never intersects the y -axis. In a similar vein, as x approaches negative infinity, $g(x)$ approaches 0 from the negative side, again without ever touching the axis. Furthermore, as x approaches 0 from the right, $g(x)$ ascends towards positive infinity, and conversely, as x approaches 0 from the left, $g(x)$ descends to negative infinity, in both cases without making contact with the x -axis. Such behaviors aren't just random; they are recognized phenomena in

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

mathematics and are termed as 'asymptotes' for both the x and y -axes.

3.1.1 Forever Close, Never Met: The Untouchables Asymptotes.

An asymptote is a line or a curve that a function approaches but never intersects. It describes the behaviour of such a concept at a certain point or at the extremes of its domain, which is where we use infinity:

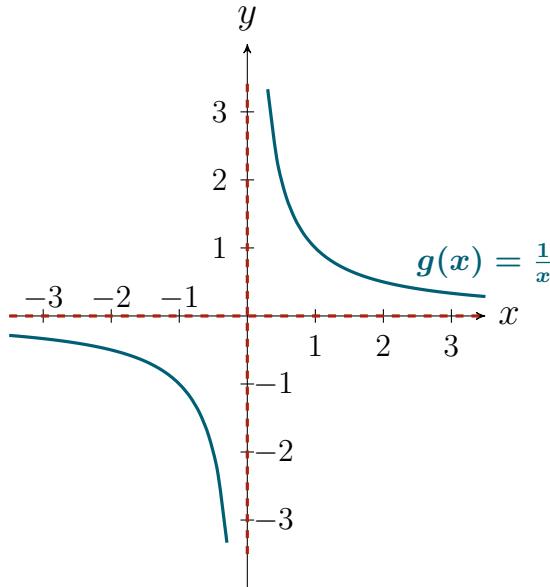


Figure 3.23: The infamous dashed *untouchables* lines.

There are three types of asymptotes:

- **Horizontal:** This is one of the ones we just found; when we made the function tend to values of either $-\infty$ or $+\infty$, we get images very close to 0, so we can represent this asymptote by $y = 0$.
- **Vertical:** Visually, it seems like we also have one of these; when we get closer to the y -axis from the left or the right, we see that our function gets closer and closer to the y -axis, but it

will never intercept it. Therefore, the vertical asymptote can be defined by $x = 0$. The mathematical way to represent this for the left of zero is 0^- , and the right of zero is 0^+ .

- **Oblique:** The term "oblique" in this context refers to a line that forms an angle other than 90 or 180 degrees with the x -axis. These typically appear in rational functions when the degree of the numerator polynomial is exactly one greater than that of the denominator polynomial.
- **Curved:** Curved asymptotes are less common and occur when the function approaches a curved line, rather than a straight line, as it tends towards infinity. These are often found in more complex functions where the behavior of the function is not linear as it moves towards infinity.

To study the case of an oblique asymptote, let's consider a case concerning bee productivity. Bees don't have social media mentors that drink black coffee inside a cold running shower while brushing their teeth so they save five minutes to make yet another video on how productive they are. They have to do stuff. Otherwise, there are serious consequences, one of them being the defense of the hive. If the guards at the hive's entrance (bees armed with machine guns and helmets) spot an intruder, such as a wasp, they will try to stop it. If this task proves unsuccessful and the wasp gets in, a group of bees will form a ball around the intruder. They can disconnect the muscles from their wings, allowing them to move these muscles without actually moving their wings. When they do this rapidly, it creates vibration and generates heat. The wasp can't stand the temperature and the rise in carbon dioxide level and drops dead. Bye bye bitch.

Let's pick a function that represents the impact on productivity given a deviation from the average task duration. If bees take more time than expected on a given task, it will cause delays in the other chores that must be completed, causing problems in the hive. So x represents the deviation from the average task time, meaning that for $x > 0$, the bee takes longer than average to complete the task.

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

Conversely, for negative values of x , the bees work faster than the average. One way to represent these dynamics is by $y = x$. For positive values of x , y is positive, meaning that more significant delays in the tasks will have a greater impact on the overall hive productivity. The opposite scenario also makes sense, if jobs are completed in a faster fashion, the impact on productivity will be positive. So, negative values of y mean less delays and, therefore, more productivity.

Now, two problems: First, $y = x$, is boring and linear. Second, where is the oblique asymptote? We mentioned before that bees help each other, so in a collection of bees, one may finish its task faster than others and help the ones in trouble. We could have a term representing this help, which would slow down the impact. Why not $\frac{1}{x}$? when x is large, many bees are not completing their tasks in time; therefore, less will be available to aid the others.

This dynamic is well reflected by $\frac{1}{x}$; high values of x make the result of that fraction smaller. We now need to verify what happens for the negative part of x . Here the term $\frac{1}{x}$ will represent the penalty of having too many fast bees, which might mean poor task distribution in the bee hive. Cool, we will have a good model if we add $\frac{1}{x}$ to x .

Look, I know you're thinking, "the hell with the bees and give me the function already." There is enormous value in thinking about the problem and what to use before just copying whatever is on the most recent blog posts. Bees are just incredible animals, so we can have some fun, ease the mathematics and learn something new. Anyhow:

$$\begin{aligned} f(x) : \mathbb{R} \setminus \{0\} &\rightarrow \mathbb{R} \\ x &\rightarrow x + \frac{1}{x} \end{aligned} \tag{3.17}$$

Now let's create a graph:

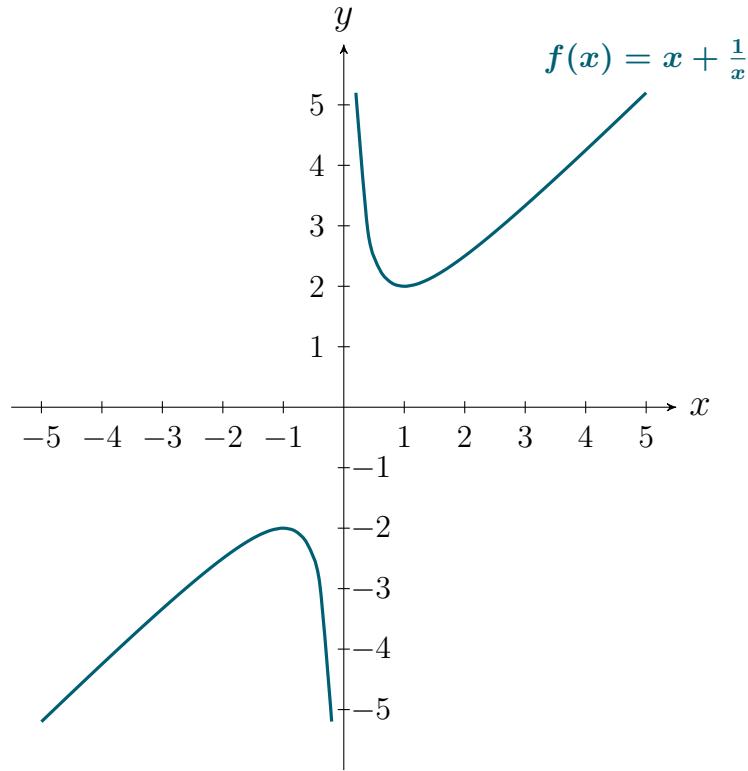


Figure 3.24: Why are we titling the function?

Finally, for the asymptotes. We have a vertical one, which is when $x = 0$. The function is not defined at that point, as $\frac{1}{0}$ is not specified, but it will tend towards positive or negative infinity as x tends to 0 from both positive (0^+) and negative (0^-) directions. Looking at the graph above, we can draw a line that goes through the middle without touching it. Let's analyse the function 3.17 for that. We have two terms in that equation, and we want to understand what happens when the function goes to the realms of infinity, both positive and negative:

- x : As we make it tend to the ends of the world where our good old friend infinity lives, this term will take on larger and larger positive or negative values.
- $\frac{1}{x}$: This term tends to 0 as we make x tend to infinity or negative infinity. We have just studied this in the section on productivity in bees.

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

So, if $\frac{1}{x}$ goes to 0, then the dominant behavior of the function is determined by x . Therefore, the asymptote will be $y = x$:

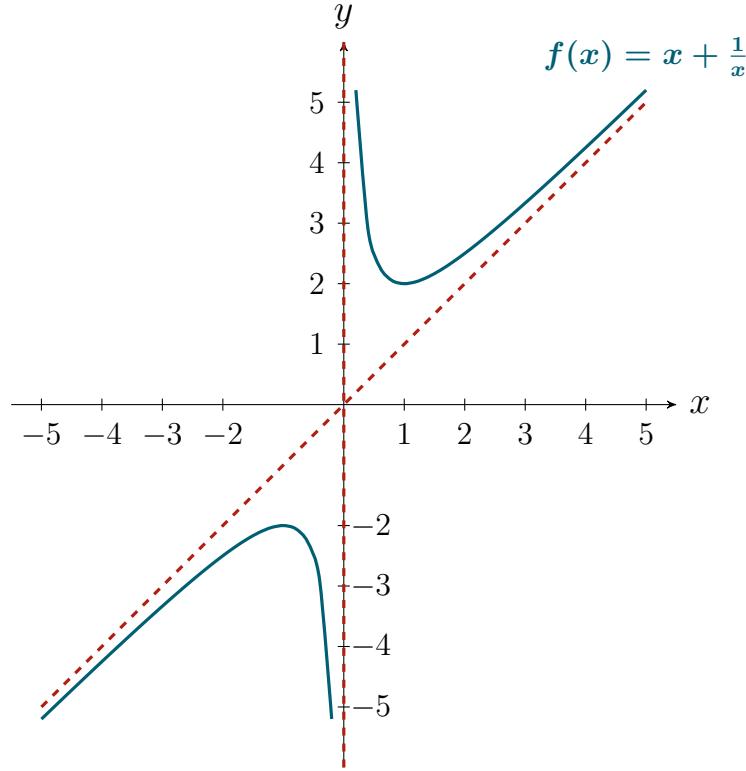


Figure 3.25: Oh! The oblique asymptote, yes!

The concept of an oblique asymptote, as seen in the example where $y = x$, illustrates a balance or proportionality between time spent on a task and task efficiency, especially as the time becomes very large. This implies that even if bees spend a substantial amount of time on a task, their efficiency asymptotically approaches but does not exceed a certain limit.

Asymptotes symbolize lines or curves that a function approaches but never actually intercepts. In the context of machine learning, they can represent plateaus or limits in performance. For example, when training an algorithm, we aim to adjust the model's parameters to better learn the patterns in data. To do so, we will use a technique called optimization. This method can often be an iterative process, where we compute the difference between our predictions and the actual outcome present in the data. How good our model is can be reflected by a function, often called cost or loss function. Generally,

the lower cost, the better, as it usually indicates good performance of the given model. At some point, improving the cost function and, consequently, the algorithm, will not be possible. This will mean that we have reached the minimum cost possible; we have plateaued. Well, the idea of an asymptote helps us to understand this:

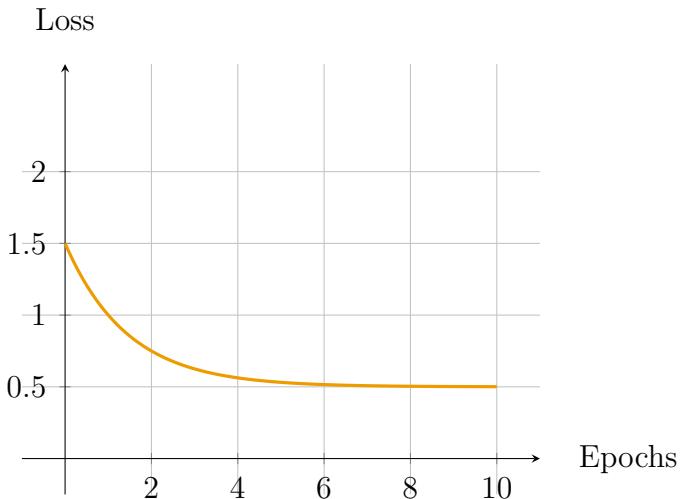


Figure 3.26: An example of an asymptote in Machine Learning.

Let's create a hypothetical scenario with a dataset of 1,000 images of bees and flowers. Each image will have a label describing what is in the picture, whether it is a bee or a flower. Our goal is to create a model that distinguishes between bees and flowers. We begin by training the model with 1,000 labeled images. Each training iteration, called an epoch, involves feeding these images to the model, assessing the cost (a measure of prediction accuracy), and adjusting the model parameters. Despite repeating this process multiple times, there comes a point when further improvement in the cost function ceases. This plateau can be represented by an asymptote, as illustrated in the plot above

So, good old infinity gave us three opportunities: local linearity, areas with no regular shapes, and asymptotes. There is one more concept that I would like for us to go over before leaving infinity. I never thought I would say or write those words, “leaving infinity”, unless we were talking about a bar called Infinity. Anyway, what I am talking about is the notion of continuity.

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

3.1.2 Better Sharpen Our Pencils: Continuity, The No-Lift Zone!

The concept of continuity is crucial in mathematics as it describes the smoothness and absence of abrupt changes or breaks within the behavior of a function. Think of it as if the function plot was drawn without lifting the pencil. Let's start by doing some visual analysis, and then we can formalise the conditions.

Consider the queen bee's task of regulating the hive's population. She adjusts her egg-laying rate according to the hive's occupancy. As the hive reaches its capacity, the rate of new bees hatching decreases. Say that we wish to represent this phenomena; we could do so by modeling the relationship between the area of the hive occupied and the number of bees inside it. As the volume of occupancy increases, the rate of newborn bees will slow down, meaning that a polynomial function won't be adequate for this situation; we need something that, at some point, will start growing at a slower rate. What about a square root? Let's see:

$$f(x) = \sqrt{x} \quad (3.18)$$

Freeze! You have the right to remain silent, everything... Sorry, that might be old trauma popping up. The reality is that before moving into studying the behavior of square root functions we have to address something, so we do have to pause a moment and refer back to a quadratic function such as:

$$y = x^2 \quad (3.19)$$

3.1.2.1 Boomerang Bonanza: The Inverse Function's Return Trip.

What if we decided to isolate x on 3.19 instead of y ? Well let's give it a go:

$$x = \sqrt{y} \quad (3.20)$$

That guy represented by 3.20 is very similar to equation 3.18 and there is a reason for this, they are inverse functions. Essentially, if we have a function $f(x)$ that takes an input x and maps it to an output $f(x)$, its inverse function, denoted as $f^{-1}(x)$, performs the reverse operations: it takes an input $f(x)$ and brings it back to the original x . So if we have that:

$$f(x) = \sqrt{x}$$

Then:

$$f^{-1}(x) = x^2$$

Analytically, this is how you find a function's inverse by isolating x instead of y . Graphically, the inverse of $f(x)$ is mirrored along the line $y = x$. This reflective property means that a point (a, b) on the graph of $f(x)$ will find its counterpart at point (b, a) on the graph of $f^{-1}(x)$:

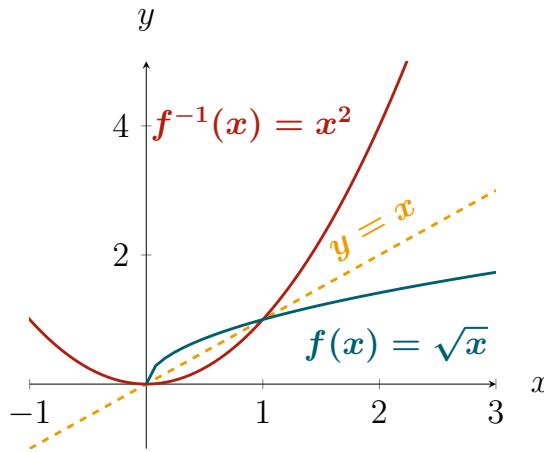


Figure 3.27: An oblique reflection of the inverses.

An important caveat to remember is that not all functions have inverses. A function will have an inverse only if it is one-to-one, meaning it has a unique output for every unique input. If a function is not one-to-one, we might restrict its domain to allow for an inverse to exist. The inverse of a function plays a crucial role in the field of machine learning. Often we use techniques to transform data, either by scaling it, projecting it onto a new space using linear algebra (I have a book on the subject!). All these techniques can be considered functions; the way to bring the data back to our original space or

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

scale is by doing the inverse operation, hence the need for these bad boys!

OK, a slight detour to the inverse of a function but it was just there for grabs with the x^2 and the \sqrt{x} so I thought we could cover it. In essence, it is the same concept as of the inverse of a matrix. So, let's start studying this concept with our function $f(x) = \sqrt{x}$; spoiler alert it will be the same as the blue line that is on 3.27:

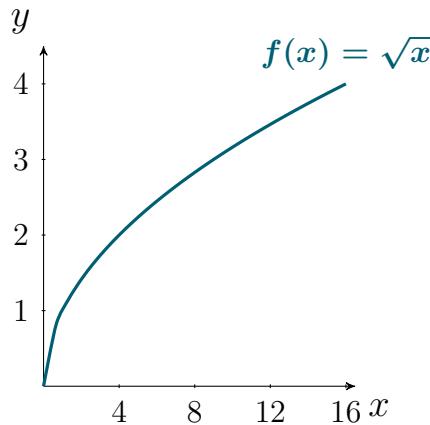


Figure 3.28: Roots exposed: When functions show their other half.

The x -axis represents a unit of time, while the y -axis depicts the total volume of occupancy in m^3 . The shape of the function in graph 3.28 captures the behaviour we are looking for. The growth slows down with time, meaning the rate of newborn bees follows that same trend. In terms of what brought us here, continuity, we have no sudden jumps or breaks in the function. In the case of the square root model, as the time x increases, the volume of occupancy $f(x)$ also increases. However, there are no sudden jumps or abrupt changes in the function.

An excellent way to visually solidify the concept of continuity is to provide an example of a non-continuous function. In some cases, if the hive becomes too crowded or the queen's pheromone production declines, the worker bees may decide to replace the current queen with a new one. This process is called supersEDURE. Worker bees will rear a new queen from selected larvae, and, once mature, she will either take over the colony or lead a swarm to establish a new hive.

Revolution, like the good old days! That means that the population of bees will, at a certain point, decline and then stabilize:

$$f(x) = \begin{cases} \sqrt{x} & \text{if } x \leq 12 \\ 2.5 & \text{if } x > 12 \end{cases} \quad (3.21)$$

3.1.2.2 The Mood Swings of the Graph World: Piecewise Functions.

The function for an equation such as 3.21 is called a piecewise function. These new fellows are functions defined by multiple sub-functions for different parts of their domain. In our specific case, we have two sub-functions because at a volume value of $f(12) = 3.46$, the workers rear a new queen. Subsequently, some bees, along with this new queen, depart to establish a new colony, causing the population of the original beehive to sharply decrease, and then remain stable for a period. This behavior is what the function 3.21 reflects. Visually we have the following:

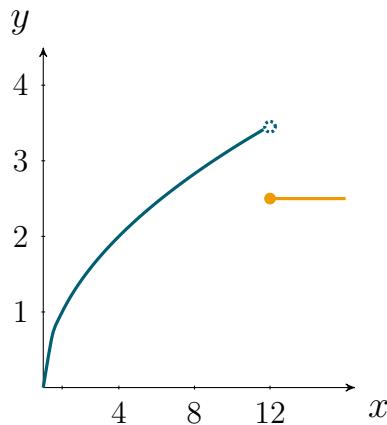


Figure 3.29: Damn, did it brake a leg?

In this particular situation, the function is not continuous; we have a point of discontinuity at $x = 12$, where we see a jump. There is one piecewise function that is famous in the field of machine learning; we even have a name for it; it is called ReLU, which stands for rectified linear unit, and it has this shape:

3.1. The Tale of the Eight with Wobbly Knees: **Infinity**.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.22)$$

The ReLU function belongs to a crucial category known as activation functions, which play a pivotal role in neural networks. Neural networks are, in essence, a party of dot products, which means that all the operations are linear. The world is often not linear, so one way to introduce nonlinearity to all the linear algebra under the hood of a neural network is via an activation function. This is a function that receives the results of some dot products (check before machine learning volume 1 for more linear algebra information) and introduces nonlinearity. Later in this book, we will dive in-depth into these topics; for now, I am just showcasing an application of these functions in the field.

We now need the aid of notation to construct a formal definition of this concept. For that, let's return a generic function $f(x)$ and some point of its domain denominated by c . For $f(x)$ to be continuous at some point c , these three conditions have to be satisfied:

- The function $f(x)$ must be defined at the point c , meaning that $f(c) \in \mathbb{R}$.
- The limit of $f(x)$ as x approaches c must exist.
- The limit of $f(x)$ as x approaches a given point c must be equal to $f(c)$.

If these conditions are satisfied at every point in the function's domain, then we have continuity in the entirety of this same set. Some functions can be continuous locally, meaning that they can have chunks of continuity followed by discontinuity. So for a function f to be locally continuous at a point x_0 it must exist a $\delta > 0$ such that f is continuous at $]x_0 - \delta, x_0 + \delta[$.. ermeeeeeee [means that the number is included and $]$ represents the opposite. Just in case...

In most machine learning algorithms training phases, we will deal with optimization tasks based on derivation. For these optimization techniques to work smoothly, the functions involved, such as the cost function, should ideally have well-defined derivatives. Discontinuities can pose challenges, as when a model encounters a data point near a discontinuity, it may produce erratic or unpredictable results, leading to a lack of robustness. We have re-encountered the word limit, so it is time to define this concept formally.

3.2 Claustrophobia in the Calculus Corridor: The Limit.

In mathematics, the definition of a limit refers to reaching a certain point where you throw your notebook to the wall while screaming, "Enough of this Greek letter nonsense!" It can certainly feel like that. The good news is that if you ever reach this state, you are not alone; I was there a lot, and since good news never comes without bad ones, allow me to tell you that I still often go back to that place.

That could indeed be called a limit, but I am talking about something else. A limit is a notion related to proximity, which is reflected in a generic function $f(x)$ when x gets closer and closer to a specific value. To formally define this guy, let's dive into the world of honey production again. Bees need to visit two million flowers to produce one pound of honey. It takes 12 bees their entire lifetime to produce one teaspoon of honey. Imagine that! When I learned that, eating honey made me feel like I was exploiting somebody, as if I was buying two cellphones a year. OK, the lifetime honey production of a bee would be hard to track, but we can assess a 24-hour period and see how the honey production varies within that time frame. Consider the following:

$$\begin{aligned} f(x) : [0, 23] &\rightarrow \mathbb{R}^+ \cup \{0\} \\ x &\rightarrow -2x^3 + 45x^2 \end{aligned} \tag{3.23}$$

Bees are diurnal creatures, so honey production is typically high-

3.2. Claustrophobia in the Calculus Corridor: The Limit.

est during mid-morning to early afternoon. They prefer the warm temperature, not because of pina coladas and white sandy beaches, but because nectar and pollen sources are readily available. Even though they favor warmer temperatures to produce honey, when these get too high, the yellow and black machines can cool themselves down by fanning their wings as air circulates over their bodies. I wish I had this ability; I would have been spared from the embarrassment of being caught using my flamenco-style fan.

We have a new player in town in equation 3.23, x^3 , but fear not as we know what the deal is with x^2 , so this fellow should not be too much a headache. Let's plot it:

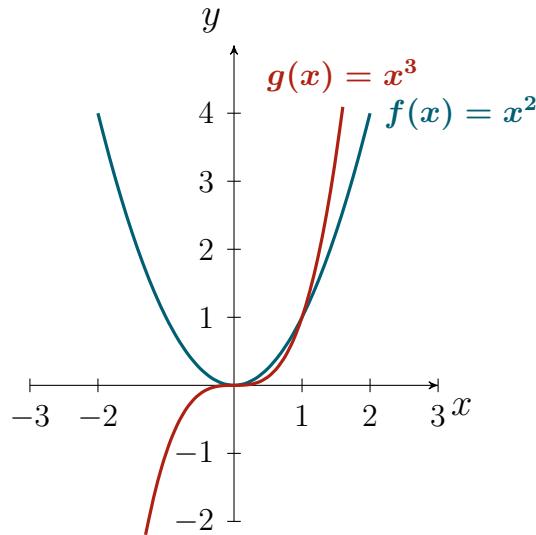


Figure 3.30: The start of the function mushing behaviour study.

For x^2 , all values in the codomain are positive, irrespective of whether x is positive or negative. However, this is not the case for x^3 . In a cubic function, the sign of x influences the outputs sign, meaning negative values of x will yield negative outputs. The minus signs mean that the graph's left hand side will be different for x^2 and x^3 as one will point upwards and the other downwards, as shown in figure 3.30. Also, the absolute values of x^3 increase more rapidly compared to those of x^2 , especially as x moves away from zero.

Nice to meet you x^3 . You seem like a pleasant fellow, not much fun but also not that boring, just like almond milk. We understand these functions in isolation, but we have more happening in equation

3.23. Not only are we multiplying constants to those terms with exponents, but also we are adding them. To comprehend what is happening, let's think about the images of x^3 when we multiply them by -2:

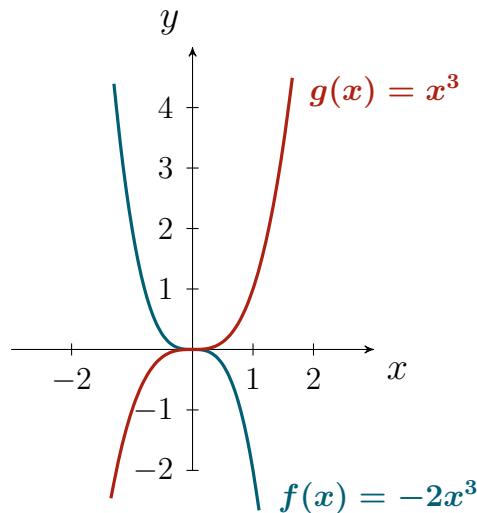


Figure 3.31: Another mush mash scenario.

The minus sign causes the function to "flip" as if the x -axis were a mirror. We have a name for this; reflection. The scalar, in this particular case, the number 2, makes the function grow faster in absolute value. Now for the other part of equation 3.23, the term $z(x) = 45x^2$, we have a quadratic function, and its plot is a similar shape to the one in 2.10. The coefficient behind the quadratic term in our particular case, 45, will widen the curve:

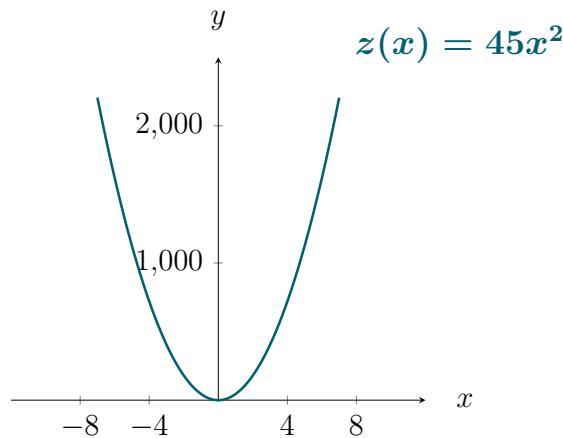


Figure 3.32: To ease the mood, this plot is just $z(x) = 45x^2$.

3.2. Claustrophobia in the Calculus Corridor: **The Limit.**

But what will happen when we have add quadratic term to a cubic one? Nothing better than a plot to understand what is going on!

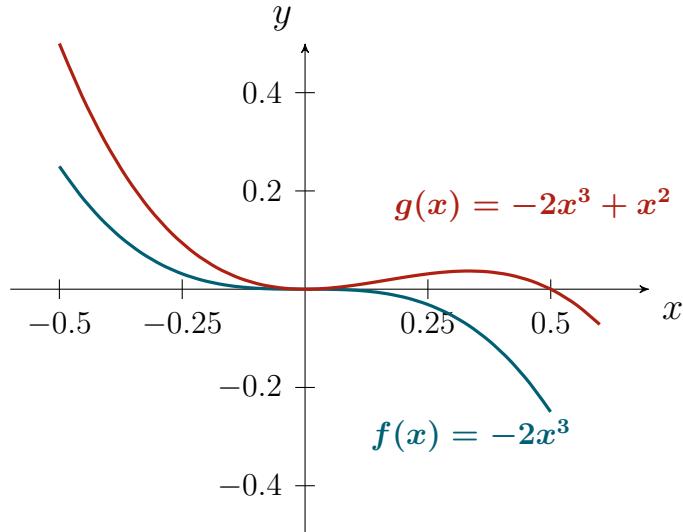


Figure 3.33: The last of our miscellaneous functions.

Let's call $l(x)$ the combination of $f(x)$ and $g(x)$, meaning that $l(x) = f(x) + g(x) = ax^3 + bx^2$. The quadratic term has a smoothing effect on the curve around the origin. It can also be called the smooth operator, and there is a song about this by Sade; (please don't believe this, it was a joke to ease all the mathematical vernacular). What is important to understand is that this action of smoothing happens because the quadratic term grows at a slower pace than the cubic one and, therefore, provides a counteracting effect on the growth speed of the fellow with the 3 for exponent.

Consider the terms a and b ; these sons of guns will also play a role in the curves of the function $l(x)$. If they are far apart in value, the term with the larger magnitude will dominate, especially when x increases. If they are not that distant, we observe the most effects near the origin, because the quadratic function has slower growth, making the function smoother around that area.

Why don't we describe the function $f(x)$ with the knowledge we have so far? The more often we do this type of exercise, the better we will be at combining different functions. Remember that

these guys are our tools for translating the world into letters, so by understanding what happens when we add and subtract terms, we are increasing our chances of building robust models.

We will start near the origin when $x = 0$. The term $45x^2$ grows at a slower rate than $-2x^3$, so this will create a smoother curve around this area. As x increases, the cubic term will prevail since this individual function will have negative values. But because of the quadratic term, the function will reach the negative magnitudes slower when compared to $-2x^3$. The curve will be broader around the origin. The missing bit is that, when x decreases; this will be very similar to what happens for $x > 0$, but the function will reach positive values instead of negative ones at a similarly slow rate. Again, this is due to the quadratic term. Slow-ass quadratic term!

Before we move on to the final graph of the combination of the functions whose behaviors we have been describing, there is an important concept we can't leave untouched. We paid a lot of attention to the cubic and the quadratic terms of the polynomial function $-2x^3 + 45x^2$, but those sneaky -2 and 45 have a significant role. They are called degrees of freedom. This refers to the number of independent parameters in a function's equation. They impact the shapes of our plots, making them good candidates to manipulate to obtain the desired results when trying to fit a model into data. I will provide one more example for clarity.

$$f(x) = ax^3 + bx^2 + cx + d$$

This particular function has four degrees of freedom, a, b, c, d and can therefore accommodate more complex shapes than for example:

$$g(x) = ax^3 + bx^2$$

This has only two degrees of freedom, and what also stands out is the polynomial's degree. Yes, that's right; it's another name to add to the list of terms that an equation representing a function has:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0 \quad \text{where } n > 0, a_n \neq 0$$

The order of the polynomial is equal to the highest exponent of the equation that represents it. In this particular case, the order

3.2. Claustrophobia in the Calculus Corridor: **The Limit**.

of $p(x)$ is n . Understanding the intertwined concepts of degrees of freedom and order in modeling is fundamental for practical machine learning applications. The order of a model delineates its inherent complexity, offering the requisite curvatures and behaviors that allow the model to capture underlying patterns in the data. This is especially true in polynomial regression and similar models, where the order defines the degree of the polynomial and, consequently, the potential shapes the model can assume, groooovyy! On the other hand, the degrees of freedom, represented by the model's adjustable parameters, are the essence of the learning process. We refine our model to best fit observed patterns by optimizing these degrees of freedom, as they are driven by data.

Well, well, well, time for the truth. This is in fact a drawing course and not a calculus book; apologies, I thought I could get more sales like that! Anyhow, back to our function $f(x)$ and our task of representing the honey production in a 24-hour period:

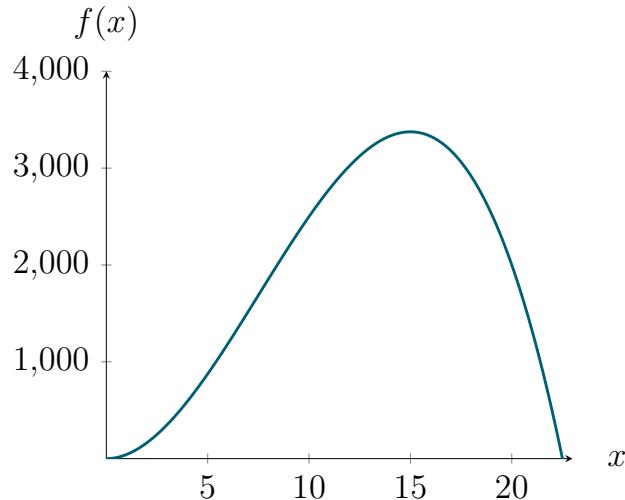


Figure 3.34: The plot of $f(x) = 45x^2 - 2x^3$ to represent the honey production.

Graph 3.34 comes from the function 3.23. The production of honey ramps up from the early morning, at its peak at around 3 p.m., and then decreases with the arrival of the darkness and the appearance of the stars; this is when bees go outside to smoke reefer. Oh, I meant they get some rest and start over again the next day.

Say that we wish to understand what occurs to honey production at around for example, 10 in the morning; what will happen to the

quantities of honey when we get closer and closer to the limit of 10 a.m.? It has been a while since we introduced new notation, and this is not how we do things around here; but let's get a nice one for our pal the limit:

$$\lim_{x \rightarrow a} f(x) = L \quad (3.24)$$

Equation 3.24 reads as the limit of $f(x)$ when x is in the proximity of a is equal to L . So when the functions receive input values that are getting closer and closer to a , their value is L . Well, proximity is a broad term; how close are we speaking? Is it a finger pad? A British red double-decker? A light year? Think of it this way, for a given range of values around a , another interval of $f(x)$'s values must exist. We spent so much time defining the function 3.23 that it would almost be a crime not to use it. If we are after the limit of honey production when we get closer to ten in the morning, we can describe it as:

$$\lim_{x \rightarrow 10} 45x^2 - 3x^3 \quad (3.25)$$

We have the mathematical formulation; Now, so we can have a better understanding of the concepts of ranges and proximity, let us zoom in around the point $x = 10$ and create some ranges around bot $x = 10$ and $f(10)$:

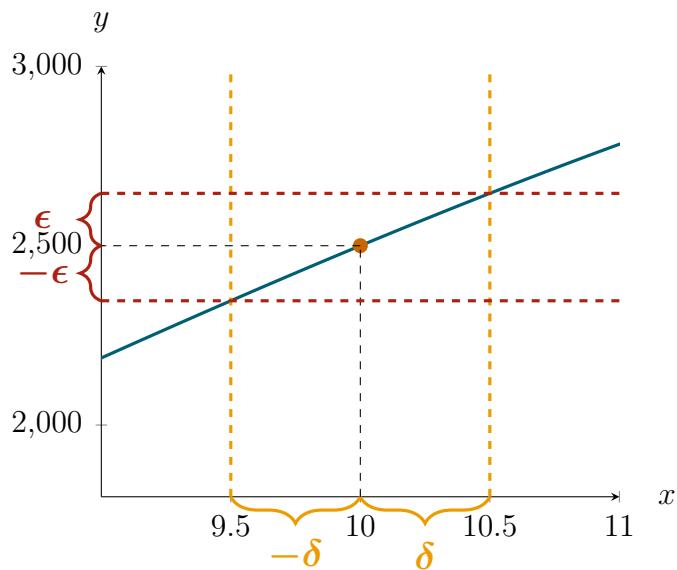


Figure 3.35: Oh (10, 2500) is entrapped by δ 's and ϵ 's.

In graph 3.35, the yellow lines represent a range around the point $x = 10$ for which we want to compute the limit, defined by the letter

3.2. Claustrophobia in the Calculus Corridor: **The Limit.**

δ . The red lines correspond to the values of $f(x)$ for the specific x values within this range. The Greek letter ϵ is used to describe the range of output values.

There is a relationship between the interval defined by δ and the one described by ϵ . In this particular case, if we keep decreasing the size of ϵ , the interval defined by δ will also get shorter no matter how small we choose ϵ to be, which means that the area of the rectangle formed by the intersection of the red and yellow dashed lines will also be getting smaller and smaller. This relationship between δ and ϵ is crucial for understanding how the function behaves as x approaches a specific value, such as 10 in our case:

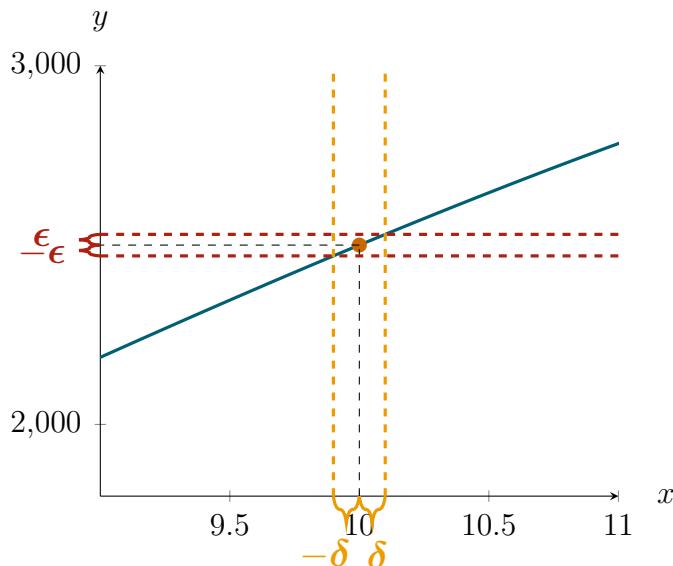


Figure 3.36: The walls are getting tighter!

If we keep up with this shenanigans, we will eventually end up with a rectangle so tiny that the value of y will be 2500, which is the same as $f(10)$.

Now, let's establish notation for the intervals involving ϵ and δ . It's essential to emphasize that ϵ must always be greater than 0. As we examine the plots above, we observe that when approaching $f(x)$ from the right side, it remains below $L + \epsilon$, signifying that all function values must be less than the limit plus half the interval's size. Similarly, when approaching from the left, the inequality $f(x) > L - \epsilon$ must be satisfied. Come on, even if we don't know what

it is, undoubtedly, there is some notation we can use to make those look better. Inefficiency is not a current theme in mathematics, and we do have something called the module function that we can use:

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases} \quad (3.26)$$

In equation 3.26, we have represented the module or absolute value function. Graphically, this is as if we pulled the left side of the function $g(x) = x$ up:

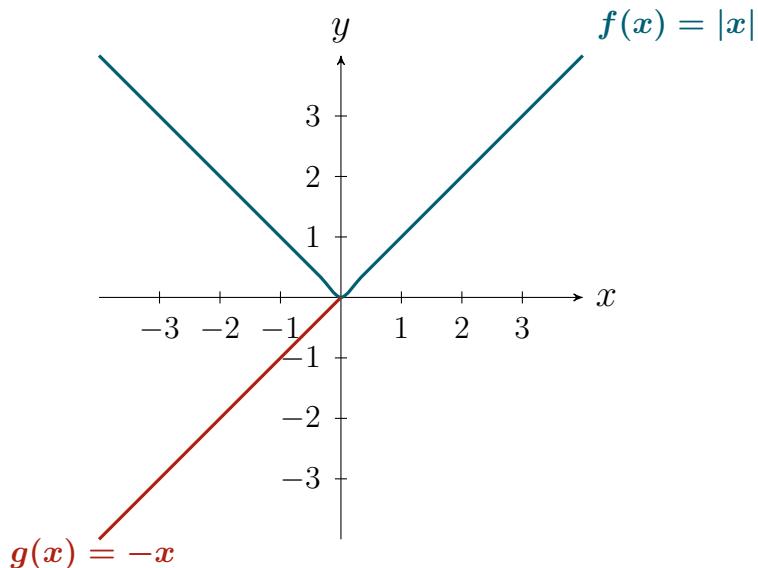


Figure 3.37: A module function with an extra leg.

In the graph 3.37 we have the blue lines representing the function $|x|$. The red piece is to showcase the part of $g(x) = x$ that got changed as the module only returns positive values. Because of this feature, this function is a good fit for representing distances. For example the distance of both $x = 3$ and $x = -3$ to the origin 0 is 3 which is the same as $|3|$ or $|-3|$. Suppose we want to describe all numbers within 2 units of the point $x = 3$. This means we're looking at the interval of numbers that lie between 1 and 5. Using the modulus, we can represent this as:

$$|x - 3| < 2 \quad (3.27)$$

3.2. Claustrophobia in the Calculus Corridor: **The Limit.**

If x is 2 units greater than 3, it means that $x - 3 = 2$. For cases where x is 2 units less than 3 we have that $x - 3 = -2$, meaning that any number x within 2 units of 3 will satisfy:

$$-2 < x - 3 < 2 \quad (3.28)$$

Equations 3.27 and 3.28 are equivalent and in fact we can use 3.28 to express the notation with the ϵ 's and the L 's we are trying to improve, so:

$$-\epsilon < f(x) - L < \epsilon \quad (3.29)$$

So, by the equivalence of 3.27 and 3.28 we can write 3.29 as:

$$|f(x) - L| < \epsilon$$

We have a new piece to the puzzle of defining a limit. We can represent the intervals defined by the ϵ 's by $|f(x) - L| < \epsilon$, so:

$$\forall \epsilon > 0, \exists \delta > 0 \text{ such that } |f(x) - L| < \epsilon$$

The only thing missing is the consideration of distances dependent on δ . No matter how small δ is, we must ensure that $|x - a| < \delta$. However, if we leave it at that, we're essentially saying that x can be equal to a , implying that we're not merely approaching a , but we can actually reach it. Considering what led us here, the definition of a derivative allowing x to be equal to a seems like a bad idea, and we aim to avoid such setbacks. To do that, let's shift our attention to the formula for the instantaneous rate of change at a point a and explore what happens when $x = a$:

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a + h) - f(a)}{h}$$

Letting $x = a$ will be the equivalent of allowing $h = 0$. If we let this happen, that quotient becomes indeterminate, and we would lose the instantaneous meaning as we wouldn't be looking at a change anymore – we'd be fixed at a single point. The derivative's purpose is to study how the function changes near that point, not how the function behaves at that exact point. So we must say that:

$$0 < |x - a| < \delta$$

This not only sidesteps potential mathematical pitfalls (like division by zero) but also retains the core intent of these concepts: studying behaviors and changes as we approach, but don't quite reach, a particular point. So our last bit of the definition has to be $0 < |x - a| < \delta$. With this we arrived at the definition of a limit using ϵ and δ :

$$\forall \epsilon > 0, \exists \delta > 0 \text{ such that } |f(x) - L| < \epsilon \text{ whenever } 0 < |x - a| < \delta \quad (3.30)$$

What that formula 3.30 says is that, for a limit to exist, the function $f(x)$ has to have the same value L whether if we are approaching a from the left or the right. OK so with this we can calculate our limit:

$$\lim_{x \rightarrow 10} 45x^2 - 3x^3 = 1500$$

The limit will be $f(10)$, which is 1500 because this is the value that the function tends to come to from both the left and the right. Now imagine the following scenario; we know that several factors can impact honey production, for example, weather conditions, pesticides, and human interference. Say that at 10 am, when we were trying to understand the limit of honey production, it rained like crazy. Excessive rain can limit the availability of nectar and pollen, which can decrease honey production. Because of this event, we have no data collection for the point $x = 10$, meaning it is not defined:

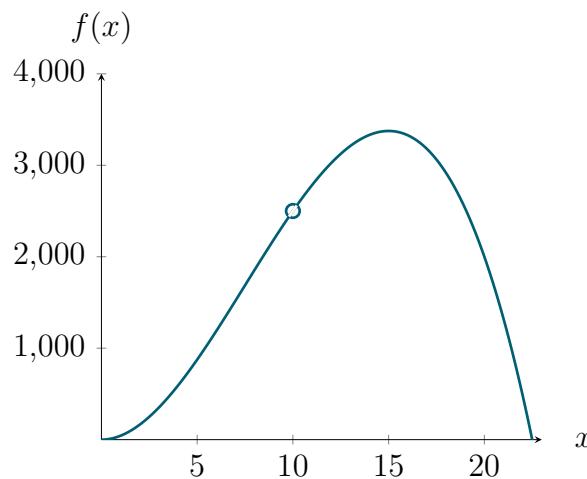


Figure 3.38: Ups, the bees missed work at 10 am.

The circle on Figure 3.38 represents the value for which we don't

3.2. Claustrophobia in the Calculus Corridor: **The Limit.**

have any data. Even though we don't have the value of the function at $x = 10$, we can still calculate the limit when it tends to 10. Remember, if we are coming from both sides, the value of $f(x)$ tends to be the same number, and this means we have a limit. On top of that, because $f(x)$ is continuous, we can evaluate it at $x = 10$, so:

$$\lim_{x \rightarrow 10} 45x^2 - 3x^3 = 1500$$

We can still have an idea of what the honey production is on a day when it does not rain at 10 a.m.

Just as with derivatives our proximity pals also come equipped with some properties that will help us with not only with calculations but also proofs:

1. Constant Rule:

$$\lim_{x \rightarrow a} c = c$$

This rule says that the limit of a constant is simply the constant itself.

2. Identity Rule:

$$\lim_{x \rightarrow a} x = a$$

This means the limit of x as x approaches a value a is just a .

3. Sum/Difference Rule:

$$\lim_{x \rightarrow a} [f(x) \pm g(x)] = \lim_{x \rightarrow a} f(x) \pm \lim_{x \rightarrow a} g(x)$$

When finding the limit of a sum or difference, you can find the limits of the individual functions and then add or subtract them.

4. Product Rule:

$$\lim_{x \rightarrow a} [f(x) \cdot g(x)] = \lim_{x \rightarrow a} f(x) \cdot \lim_{x \rightarrow a} g(x)$$

For the product of two functions, the limit is the product of their individual limits.

5. Quotient Rule:

$$\lim_{x \rightarrow a} \frac{f(x)}{g(x)} = \frac{\lim_{x \rightarrow a} f(x)}{\lim_{x \rightarrow a} g(x)}$$

For a quotient, you can take the limits of the numerator and denominator separately, but make sure the limit of the denominator isn't zero.

6. Power Rule:

$$\lim_{x \rightarrow a} [f(x)]^n = [\lim_{x \rightarrow a} f(x)]^n$$

For a function raised to a power, the limit is the limit of the function itself raised to that power.

We have now covered derivatives, infinity, continuity, and limits. Quite a bit of material, but they all have implications for each other. Some of the synergy comes from the conditions for a function $f(x)$, to be differentiable:

1. The function must be defined at the point c :

$$f(c) \in \mathbb{R}$$

2. The limit of the difference quotient as x approaches c must exist:

$$\lim_{x \rightarrow c} \frac{f(x) - f(c)}{x - c} \text{ exists.}$$

3. The limit of the difference quotient as x approaches c must be finite:

$$\lim_{x \rightarrow c} \frac{f(x) - f(c)}{x - c} \text{ is a finite number.}$$

It's essential to recognize the intricate relationship between continuity and differentiability when examining the concept of the instantaneous rate of change. Continuity ensures that a function doesn't exhibit abrupt jumps or disruptions, providing a seamless transition between neighboring points on its graph. An illustrative example

3.2. Claustrophobia in the Calculus Corridor: **The Limit.**

is the function $f(x) = |x|$, which is continuous everywhere but is not differentiable at $x = 0$ due to a sharp corner. While a function must be continuous at a specific point to be potentially differentiable there, continuity doesn't automatically guarantee differentiability. Differentiability requires that a unique tangent line exists at that point, capturing the function's instantaneous rate of change. Hence, to deeply understand and apply the notion of an instantaneous rate of change, one must first appreciate the foundational roles of continuity and differentiability.

The conditions for differentiability presented above can be summarized as follows: First, the function must be defined at the point of interest, meaning it has a well-defined value. Second, the limit of the difference quotient as the input values approach the point must exist. This limit measures the average rate of change between two points, and its existence indicates the presence of a well-defined instantaneous rate of change. Lastly, the limit of the difference quotient must be finite, ensuring that the instantaneous rate of change is well-defined and not infinite. When combined with continuity, these conditions guarantee that a function exhibits smooth and well-behaved behavior, allowing for a precise analysis of its rates of change and behavior at specific points.

The third item above speaks about the limit being a finite number; this suggests that in some given situations, a limit does not exist. One specific case is when the limit is not determined, for example, $\frac{0}{0}$. At first glance, one could conclude that this limit does not exist. It's crucial to note that if a function's value can't be directly determined when approaching a number from both the left and right, we shouldn't immediately conclude that the limit doesn't exist. Sometimes, further analysis or different methods are needed to ascertain the existence of a limit.

3.3 Deciphering the Undefined - L'Hopital's Rule.

Foraging bees have the task of going outside and locating food. When they find flowers, their food sources, a return trip is made to the hive to recruit other bees to aid in collecting resources such as nectar and pollen. These foragers release a trail of pheromones to help them find food. Intense scents on this trail mean proximity to the sources, whereas weak smells represent the opposite. Right, so we need a function. Surprised? Should I write another poem? no?! I understand, and am not offended. We will start by setting an anchor at the point $x = 0$, the location of the beehive. The negative values of x represent distances to the left of their home, whereas positive values represent distances to the right.

The y -axis represents the level of pheromones and consequently the attractiveness or desirability of a location, with high values indicating the presence of food sources or alluring sites and low values indicating non-attractive or less desirable areas:

$$f(x) = \frac{\sin(\pi x)}{x^2 - 1} \quad (3.31)$$

This will be our function, the division of a trigonometric fellow by a polynomial and it looks like this:

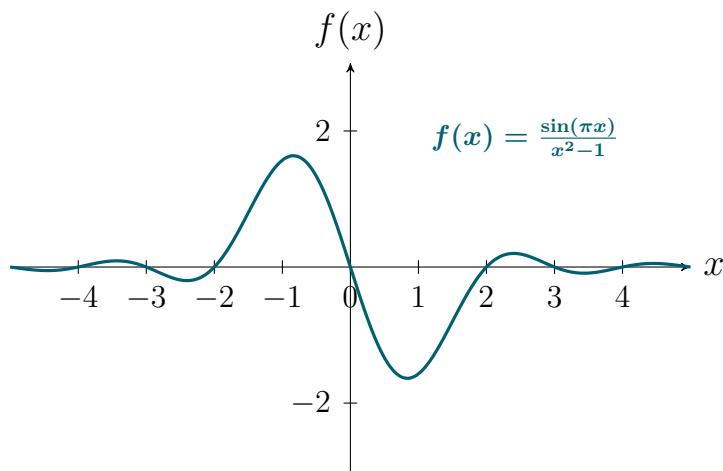


Figure 3.39: But where are those flowers?

3.3. Deciphering the Undefined - L'Hopital's Rule.

We can see in figure that 3.39 the pheromones' intensity seems to peak at a value close to 2 at one unit of distance to the left. At a value of x around -3.5, it appears that there is something of interest and on the right side, we have one little peak at about 2.5. It is a rudimentary model, but it will be enough to learn a few new concepts. The question we now have to answer is; what is the level of interest when the food sources tend to be at 1 unit of distance from the beehive? Mathematically we can write it like:

$$\lim_{x \rightarrow 1} \frac{\sin(\pi x)}{x^2 - 1} \quad (3.32)$$

The first thing to do when presented with a limit to calculate is the most obvious one, substitute the value of x in the function and check what comes out of it:

$$f(1) = \frac{\sin(\pi \cdot 1)}{1^2 - 1} = \frac{0}{0}$$

Isn't that something? Considering that we are after methodology to tackle this kind of situation, this was to be expected. Before jumping into new concepts, let's remind ourselves of the behavior of that guy on the numerator, $\sin(\pi x)$ but, first $\sin(x)$:

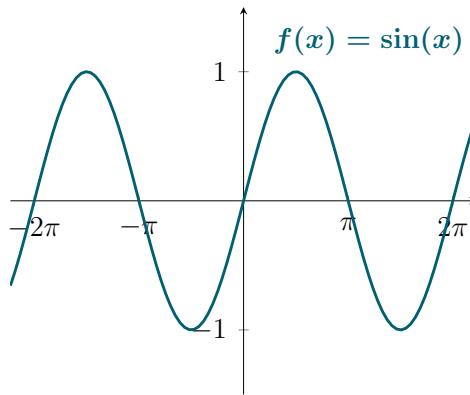


Figure 3.40: Some cool curves. The behaviour of $\sin(x)$.

One of the cool things we can leverage with these types of functions, the trigonometric ones, is their periodicity. This is a concept that implies repetition; for every given length of interval on the x -axis, the function will have the same behaviour. For example, in this case of $\sin(x)$, the period is 2π , so the function will repeat its

behavior after every interval of 2π . If you are working with data that contains repetition, like music or electrical circuits, a function with periodicity will probably be suitable. Having established the basic behavior of $\sin(x)$, let's now examine how scaling the argument by π alters this function, which is crucial for our bee foraging model. We started by checking $\sin(x)$, but our numerator is $\sin(\pi x)$ instead. For π or any scalar really, when we multiply them by the argument of $\sin(x)$, the behaviour of the function will change. To fully grasp the extent of this impact on the function's graph what we need is ... yes, a graph:

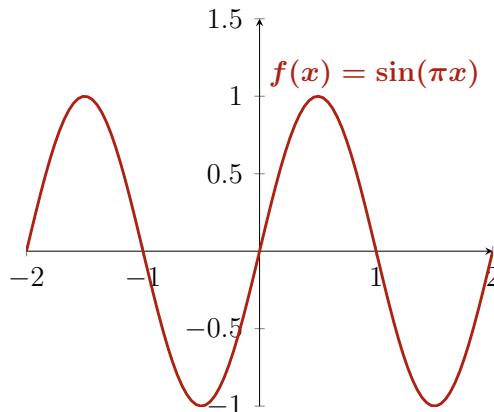


Figure 3.41: Some more curves, but from a new fellow $\sin(\pi x)$.

They are indeed very similar. A closer look at the x -axis indicates that something has changed, the period! Now we have a period of 2 instead of 2π , so the function repeats itself in a smaller interval. Generically, if we have a function like:

$$f(x) = \sin(bx)$$

We can calculate the periodicity by using the knowledge that $\sin(x)$ repeats itself every 2π so:

$$bT = 2\pi$$

So:

$$T = \frac{2\pi}{|b|}$$

3.3. Deciphering the Undefined - L'Hopital's Rule.

The letter T represents the period. The module of b is there because we need a positive magnitude as the period is a measure of length. While we are in the discovery phase, we can make things more generic and cover everything we need to understand about this trigonometric function:

$$f(x) = a \sin(b(x - c)) + d$$

Each one of those letters will impact the function differently:

- **a** : Represents the amplitude. It dictates the vertical stretch or compression of the function, with the range of the function oscillating between a and $-a$. A negative value of a results in a reflection across the x -axis.
- **b** : Known as the frequency, it controls the number of oscillations within a 2π interval, effectively altering the horizontal scale of the function. The period of the function is $\frac{2\pi}{|b|}$, so a greater b decreases the period, increasing the frequency of the cycles.
- **c** : Manages the phase shift, moving the graph horizontally. A positive c shifts the graph to the right, while a negative c shifts it to the left, adjusting the starting point of the sine wave.
- **d** : Accounts for the vertical shift, moving the graph up or down. Positive values of d elevate the function, and negative values lower it, shifting the midpoint of the oscillation accordingly.

$$g(x) = 2 \sin(\pi(x - 1)) - 1 \quad (3.33)$$

If we plot both equation 3.33 and $\sin(x)$ we can better understand the impact of a, b, c, d for both:

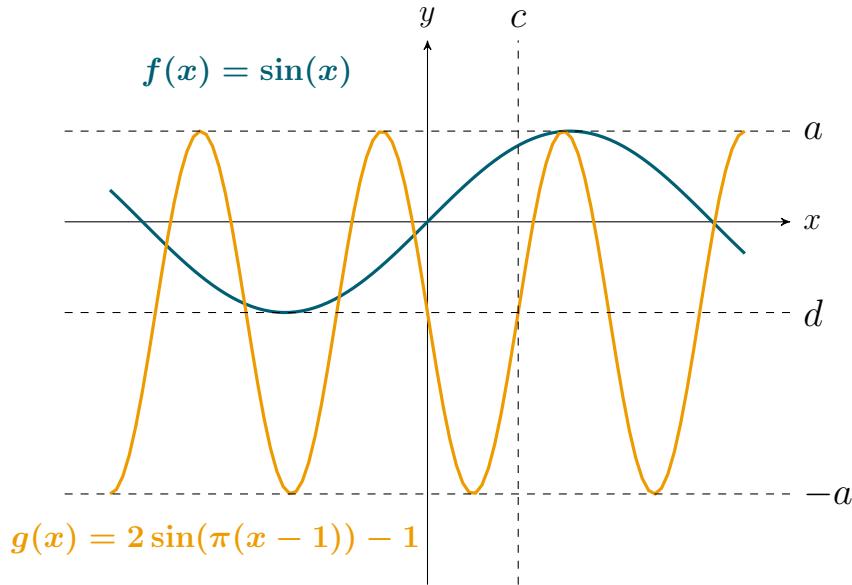


Figure 3.42: The impact of the parameters on trigonometry.

In equation 3.33, we have a value of $a = 2$, so the function should range from -2 to 2 , but as we have a value of $d = -1$, this means that we will shift it by a value of -1 on the y -axis. In regards to the period, it is 2 because $T = \frac{2\pi}{b}$ and it has a frequency of π , meaning that a full oscillation is completed in the interval $[-1, 1]$. The value of $c = 1$ means that from $\sin(x)$, we will observe a translation of 1 unit to the right.

While the transformations provided by those parameters have the same effects, the fundamental shapes and properties of the other trigonometric functions, namely, $\cos(x)$ and $\tan(x)$, are different. For example, the sine and cosine functions have values ranging between -1 and 1 for all real numbers. However, the tangent function has vertical asymptotes (values of x where the function approaches infinity) and can take on any real value. We will do more exploration of these trigonometric functions as they will keep showing up, and their role in machine learning is more critical than just modelling events with seasonality. For example, their curved and smooth shapes allow us to turn linearity into non-linearity, a process used in neural networks via activation functions. Don't worry. We will go over this later.

3.3. Deciphering the Undefined - L'Hopital's Rule.

The denominator of equation 3.31 is a polynomial, a type of function that is already familiar to us as we looked at $x^2 - 3$. This time we have $x^2 - 1$, a quadratic function translated one unit down on the y -axis.

Now that we understand the behavior of both the numerator and the denominator, it is time to compute this damn limit. If we refer back to the definition of a limit that we just went over, that one with the ϵ 's and the δ 's, we can at least suspect that the limit exists. Visually we can see that whichever side the function we use to approach 1 on the x -axis, we will probably end up with the same value for y :

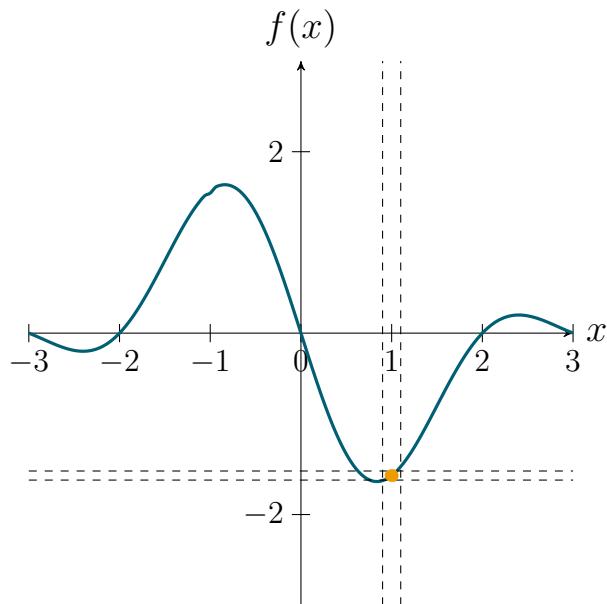


Figure 3.43: A dashed box is formed by lines for the limit representations.

If we keep decreasing the space between those dashed lines, the function will tend to one value only. Because calculating the limit with the ratio is not working, let's take care of each of those functions individually:

$$\lim_{x \rightarrow 1} \frac{t(x)}{g(x)} \quad (3.34)$$

Where:

$$t(x) = \sin(\pi x) \quad \text{and} \quad g(x) = x^2 - 1$$

Alright, how does this look ?

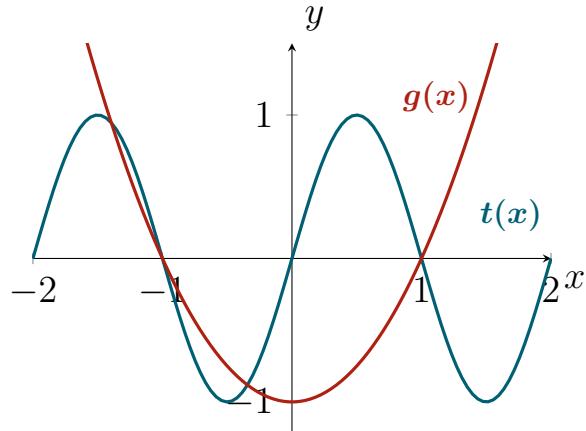


Figure 3.44: The red lead and blue bead made a deal, uniting at the same place with a playful zeal.

The functions intercept each other at two distinct points, with 0 for the y -coordinate and 1 and -1 for x , which is fancy wording for $(-1,0)$ and $(1,0)$. Because we got a $\frac{0}{0}$ when we tried to compute the limit with a substitution, using the point $(1,0)$ this crossing was to be expected. The $(-1,0)$ I had saved as a bonus, a way of thanking you for having bought this book. What do we do when a point is causing us trouble? We zoom in!

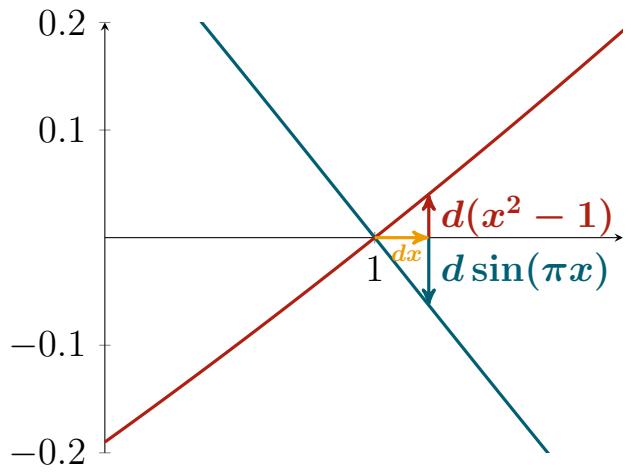


Figure 3.45: The changes around the point where we need to get really, really close.

3.3. Deciphering the Undefined - L'Hopital's Rule.

While it's possible to estimate limits by choosing values close to the point of interest, like 1.001 for instance, this raises the question: how close is close enough? Plus, this is not like us; we only work with bulletproof stuff around here, not weak statements; leave those for the people in Parliament. Even though this approximation suggestion sucks, it at least hints that we could solve this with a slight change in x . Oh, if only we had a way to... hold on... the derivative! Wow, it's almost destiny. The limit for defining a derivative and the derivative to helps us calculate a limit. Romance!

So, rather than relying on approximations, let's turn to derivatives, a more precise tool in our mathematical arsenal, to tackle this limit calculation.

The value, 1.001, represents a small change in x , which we'll denote as dx . To narrow our focus, let's concentrate on a specific point, a , located in the vicinity of 1. Think of it as the neighborhood of 1 where these minute changes are happening. In our analysis, we're examining the ratio described in equation 3.34 and how it evolves under these small alterations. This naturally raises the question: by how much does it change? To gain insight, we can evaluate the values of both $t(x)$ and $g(x)$ at this nearby point, a . At this point, the derivative of $t(x)$ encapsulates the rate at which $t(x)$ changes. When x undergoes a small change, denoted as dx , the derivative at point a becomes particularly significant. This derivative represents the instantaneous rate of change of $t(x)$ at a . So, when x changes by a tiny amount dx , this rate effectively dictates how much $t(x)$ will adjust in response. To quantify this change, we multiply the derivative of $t(x)$ at a by dx . This multiplication yields an estimated change in $t(x)$, reflecting how $t(x)$ adjusts in response to the small shift in x . This concept is not just confined to $t(x)$; it applies equally to $g(x)$. In the case of $g(x)$, its derivative at a would provide the rate at which $g(x)$ changes, and multiplying this derivative by dx gives an approximation of the change in $g(x)$ due to the small variation in x :

$$\lim_{x \rightarrow a} \frac{\frac{dt}{dx}(a)dx}{\frac{dg}{dx}(a)dx} \quad (3.35)$$

Urggggh let's simplify that equation 3.35:

$$\lim_{x \rightarrow a} \frac{\frac{dt}{dx}(a)}{\frac{dg}{dx}(a)}$$

One more time:

$$\lim_{x \rightarrow a} \frac{t'(a)}{g'(a)} \quad (3.36)$$

And that equation, 3.36 is the formula for the L'Hopital's rule. This means that we can calculate our limit by computing the derivatives of the numerator and the denominator and for our specific case check their values when $x = 1$:

$$\lim_{x \rightarrow 1} \frac{\sin(\pi x)'}{(x^2 - 1)'} \quad (3.36)$$

Let's start with the denominator, it is a polynomial and they are always friendly:

$$d(x^2 - 1) = d(x^2) - d(1)$$

By the rule of subtraction, we can write the derivative $d(x^2 - 1)$ as a subtraction of two derivatives. Well, now we need to apply the power rule and the constant rule, and we are sorted:

$$d(x^2) = 2x \quad \text{and} \quad d(1) = 0$$

So:

$$d(x^2 - 1) = 2x$$

I have another confession: I not only shot the sheriff but also the deputy. Of course, that is not the real confession, which is I left one rule of derivation out when we studied them before – but there is a reason. It is a rule widely used in machine learning, called the chain rule. Now, let's address that missing piece, focusing on the numerator, which is the function $\sin(\pi x)$. This function isn't just $\sin(x)$ or πx ; it's a combination of both. We have a specific way to name and represent such functions. Consider two functions, $f(x)$ and $g(x)$, as defined below:

$$f(x) = \sin(x) \quad \text{and} \quad g(x) = \pi x$$

3.3. Deciphering the Undefined - L'Hopital's Rule.

Combining those two will give us the function $\sin(\pi x)$, so if we apply $\sin(x)$ to the images of $g(x)$, we will get exactly what we need. One representation of this phenomenon is:

$$f(g(x))$$

3.3.1 After You, Please: The Cordial Conduct of Function Composition.

That notation translated to English means f after g , and no, we can't assume that g is not a polite individual. So if $f(x) = \sin(x)$ and $g(x) = \pi x$, then:

$$f(g(x)) = \sin(\pi x)$$

The composition of functions can also be represented by:

$$f \quad o \quad g$$

This concept of one function after another is everywhere in machine learning; it allows us to build sophisticated models by combining simpler operations. The composition of functions will come back when we introduce practical applications in machine learning, for example, neural networks. But for now, let's get the missing derivative rule:

3.3.1.1 The Key to the Calculus Handcuffs: Chain Rule.

Consider two differentiable functions $f(x)$ and $g(x)$. If we have a composition defined by $z(x) = f(g(x))$, $z(x)$ is differentiable at x with:

$$\frac{d}{dx}(f(g(x))) = f'(g(x))g'(x) \quad (3.37)$$

OK, so now we need to prove the results and for that, yes, you have guessed it correctly, we need the definition of a derivative:

$$\lim_{h \rightarrow 0} \frac{z(x + h) - z(x)}{h}$$

Now we can replace, $z(x)$ with $f(g(x))$:

$$\lim_{h \rightarrow 0} \frac{f(g(x+h)) - f(g(x))}{h} \quad (3.38)$$

Looking at equation 3.37, we can see that we have $g'(x)$. One possible strategy for us to get on with this proof would be to try to make that guy show up at this party by manipulating 3.38. For that, we can multiply 3.38 by:

$$\frac{g(x+h) - g(x)}{g(x+h) - g(x)} \quad (3.39)$$

The result of that division 3.39 is 1, so if we multiply it by 3.38, we can be confident that we are not altering anything:

$$\lim_{h \rightarrow 0} \frac{f(g(x+h)) - f(g(x))}{h} \cdot \frac{g(x+h) - g(x)}{g(x+h) - g(x)}$$

Because we are multiplying two fractions, we can swap the denominators:

$$\lim_{h \rightarrow 0} \frac{f(g(x+h)) - f(g(x))}{g(x+h) - g(x)} \cdot \lim_{h \rightarrow 0} \frac{g(x+h) - g(x)}{h} \quad (3.40)$$

We also used the rules of limits and created a multiplication of two limits represented in the equation 3.40. If we pay close attention to the limit on the right side of the multiplication, we can see that is $g(x)'$. This is a great, as it completes half of the expressions we are looking for:

$$\lim_{h \rightarrow 0} \frac{f(g(x+h)) - f(g(x))}{g(x+h) - g(x)} \cdot g'(x)$$

Let's define $g(x)$ as u and $g(x+h)$ as t to make it easier on the eye:

$$\lim_{h \rightarrow 0} \frac{f(t) - f(u)}{t - u} \cdot u'$$

Now the only thing left is to work on that fraction, and we are golden. Say that $g(x) = u$ and $g(x+h) = t$. OK, so when h tends to 0:

$$g(x) = t \rightarrow t = u$$

3.3. Deciphering the Undefined - L'Hopital's Rule.

So we can have the following expression:

$$\lim_{t \rightarrow u} \frac{f(t) - f(u)}{t - u} \cdot g'(x)$$

Let's work more on the right side of the equation. Say that we have the definition of a derivative at a generic point a , but now, instead of h , let's have $h = t - a$. So if h goes to 0, $t = a$. Initially, we had:

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a + h) - f(a)}{h}$$

If we implemented the changes described above, the result is:

$$f'(a) = \lim_{t \rightarrow a} \frac{f(t) - f(a)}{t - a}$$

This means that:

$$\lim_{t \rightarrow u} \frac{f(t) - f(u)}{t - u} = f'(u)$$

And because we defined $u = g(x)$ it comes that:

$$\lim_{t \rightarrow u} \frac{f(t) - f(u)}{t - u} = f'(g(x))$$

So:

$$f'(g(x))g'(x)$$

Just like with equation 3.37. Right, we have the rule to calculate the derivative of $\sin(\pi x)$ where $f(x) = \sin(x)$ and $g(x) = \pi x$. We will start with $g'(x)$ and it comes that:

$$g'(x) = \pi$$

Now the only thing missing is $f'(g(x))$, and for that, we need the derivative of $\sin(x)$. Well, it is $\cos(x)$, and the definition can prove it. I lost count of the times we used that definition already. It may be time for something different, for us to break the monotony. If the derivative of $\sin(x)$ is the $\cos(x)$, it implies that the slope of the tangent line at a given point a on the plot of $\sin(x)$ must be equal to $\cos(a)$. Let's check a plot:

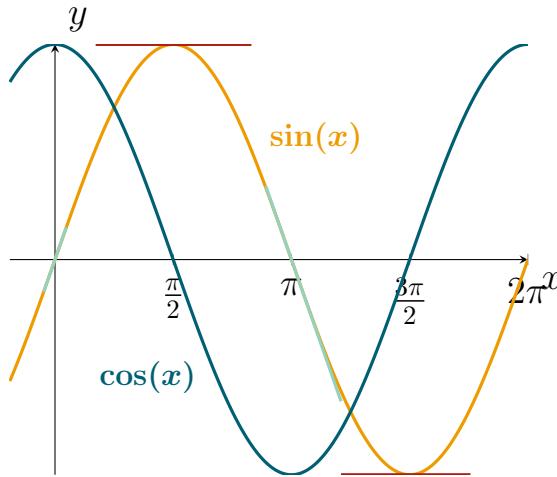


Figure 3.46: A roller-coaster project? The curves of $\sin(x)$ and $\cos(x)$.

We have two horizontal lines that are dressed in red and two oblique lines represented by the ugly blue. We can eyeball the plot to calculate the slope value for the horizontal lines. It is 0.

In our graph, we are focusing on two specific points along the x -axis: $\frac{\pi}{2}$ and π (since $\frac{2\pi}{2}$ simplifies to π). At these x -coordinates, the value of the $\cos(x)$ function is zero. This is an intrinsic property of the cosine function, which intersects the x -axis at these points due to its periodic nature. Hence, in our plot, we expect the blue line, representing the $\cos(x)$ function, to cross the x -axis at both $\frac{\pi}{2}$ and π and it does!

The oblique lines' slope can't be calculated by just looking at the plot. However, we can determine the derivative's sign by checking the corresponding tangent's orientation. So for $x = 0$, we have a line going from the left to the right, meaning that values of y increase when x increases, so the slope is positive, and so must $\cos(x)$ be.

Well, $\cos(0) = 1$, so we can verify this. On the other hand, the tangent to the point π has a different orientation; the values of y grow inversely to x , and the slope of the tangent on this point is negative. If we calculate the $\cos(x)$ value at π , it comes out as -1 . This is negative as expected.

3.3. Deciphering the Undefined - L'Hopital's Rule.

So the expression $f'(g(x))$ has the following value:

$$f'(g(x)) = \cos(\pi x)$$

Meaning that:

$$\frac{d}{dx} \sin(\pi x) = \cos(\pi x)\pi$$

So finally, we are ready to compute this limit:

$$\lim_{x \rightarrow 1} \frac{\sin(\pi x)}{x^2 - 1} = \lim_{x \rightarrow 1} \frac{\cos(\pi x)\pi}{2x} = -\frac{\pi}{2} \quad (3.41)$$

L'Hôpital's Rule works on cases in which we have undefined limits such as $\frac{0}{0}$ or $\frac{\infty}{\infty}$. We will explore one more example of a function that has an undefined limit before moving on to some more topics on calculus; for that let's consider this:

$$f(x) = \frac{\sin^2(x)}{x^2} \quad (3.42)$$

Let's check how the function 3.42 behaves:

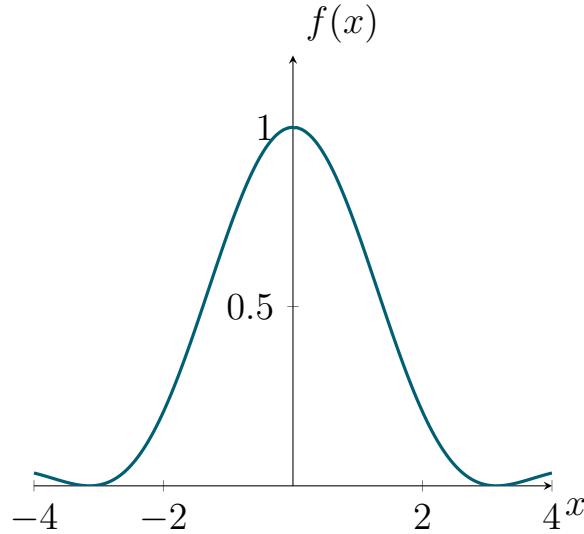


Figure 3.47: This seems to be a friendly curve!

The factor that will dominate the behaviour of $f(x)$ is the term $\sin^2(x)$. As x approaches 0, the sine function oscillates between -1 and 1, but $\sin^2(x)$ will always be positive and will tend towards

0. The denominator x^2 behaves similarly. As x approaches 0 from either direction, x^2 also tends towards 0. Because the function behaves this way around $x = 0$, $f(x)$ is undefined at that particular location. So, let's calculate the limit as x approaches 0:

$$\lim_{x \rightarrow 0} \frac{\sin^2(x)}{x^2}$$

OK, if we work with substitution, we get $\frac{0}{0}$, which shouldn't be a problem as we have L'Hopital's rule. For that we need the derivatives of $\sin^2(x)$ as well as x^2 :

$$\frac{d}{dx} (\sin^2(x)) = 2 \sin(x) \cos(x)$$

If we wish to apply the chain rule, we consider the derivative of $\sin(x)$ which is $\cos(x)$, and then apply it twice due to the power of 2. For the denominator, the derivative is straightforward:

$$\frac{d}{dx} x^2 = 2x$$

All right, we can calculate the limit with the derivatives:

$$\lim_{x \rightarrow 0} \frac{2 \sin(x) \cos(x)}{2x}$$

And, we get $\frac{0}{0}$ – phenomenal... I can only see one solution to this. Throw the book in the bin! No, I am just joking. Please don't do that. We have a rule for calculating undetermined limits, but the only thing we did was to calculate derivatives, so what about if we calculate a derivative of the derivative? The rate of change of a rate of change; now we are talking. So, for the numerator:

$$\frac{d}{dx} (2 \sin(x) \cos(x)) = 2[\cos^2(x) - \sin^2(x)]$$

For the denominator:

$$\frac{d}{dx} 2x = 2$$

This means that the limit can be computed by the following:

$$\lim_{x \rightarrow 0} \frac{2[\cos^2(x) - \sin^2(x)]}{2} = 2$$

3.3. Deciphering the Undefined - L'Hopital's Rule.

OK, there are quite a few things we need to talk about here. First and foremost, we are not limited to deriving a function once. A function can have as many derivatives as its nature allows. Each of these derivatives gives us insights into the function's characteristics at various levels. These successive derivatives are categorized in order: first derivative, second derivative, and so on.

The intuition about deriving the function again when we get an undefined limit after the first derivative is the same as original intuition, but now the function happens to be the first derivative. What we are doing by calculating the second derivative is to go to look for further information.

When we speak of the first derivative of a function, we examine its rate of change. For a function representing distance over time, this derivative equates to velocity. But when we go a level deeper and look at the derivative of that velocity, we encounter the concept of acceleration, a measure of how quickly the velocity changes. Let's bring back the function $g(x) = (x - 3)^2$ and calculate both the first and second derivatives:

$$g(x) = (x - 3)^2, \quad g'(x) = 2(x - 3), \quad g''(x) = 2$$

For the graph this is what we end up with:

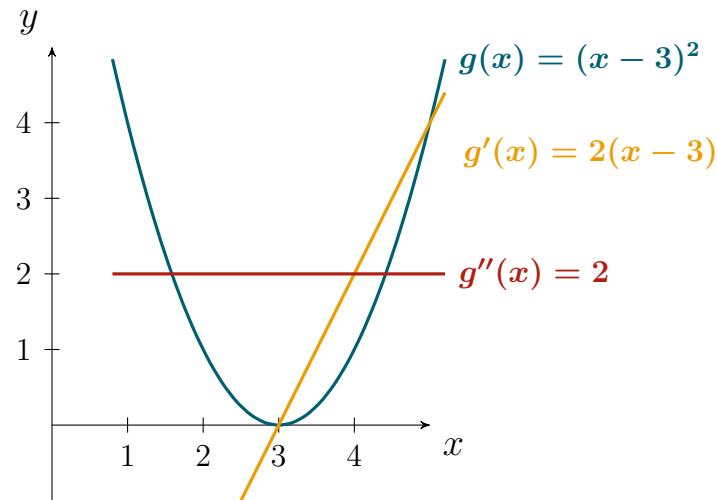


Figure 3.48: A family picture: The function alongside it's first and second derivatives.

In the graph above, we have all the players represented; the blue line is our function $g(x)$, the yellow line is the first derivative, and

lastly, the orange line is the second derivative. We know that the first derivative gives us the slope of the tangent line at any given point on the function. However, when we consider the second derivative, it's about understanding the variation of this slope along the function's path. To think of it visually, imagine drawing tangent lines at various points along a curve. The slope of these lines could change: they might get steeper, remain constant, or become flatter. The second derivative is a tool that quantifies this change. If it's positive, the slopes of these tangent lines are getting steeper, indicating that our function is bending or curving upward. On the contrary, if it's negative, the slopes are becoming less steep or more negative, which signifies that our function is curving downward. We can use this notation for it:

$$\frac{d^2g}{dx^2}$$

When we were deriving L'Hôpital's rule, we leveraged the idea that it might simplify our calculations if we could approximate a function's behavior at a specific point using its derivative. This is based on the fact that a function's derivative at a point captures its rate of change there, allowing us to infer the function's behavior in that proximity by considering tiny changes in the input variable (denoted as dx). In the early days, our world revolved around first derivatives; however, now we have the luxury of exploring as many derivatives as we wish. As we delve deeper, each derivative provides us with additional insights about the function. So, if one derivative gives us the rate of change, and the second provides curvature, it's reasonable to expect that continuing this process will yield even more information, leading us to perhaps having the ability to create approximations. These are strong words, and indeed, we can approximate a function around a point, applying the same mathematical framework but expressing it in a more polynomial-style fashion rather than, for instance, in the case of a non-polynomial function like $\sin(x)$. The reality is that polynomials are far easier to work with, so if there's a way to approximate a non-polynomial function as a polynomial, it's a valuable tool and we should explore it.

3.4 Get Me Higher and I'll Give You Better Curves: Taylor Series.

An infinite series allows us to represent or approximate functions using polynomials. Among these, the Taylor series stands out. It enables us to create precise polynomial approximations of functions around a specific point. I won't lie: trying to wrap my head around these mathematical constructs resulted in many sleepless nights. Initially, I couldn't understand this to save my life. Maybe I'm being dramatic, but man, this was a headache. So, let's start with the basics of infinite series and then delve into what Mr. Taylor cooked up for us. At its core, these series are straightforward; each term is added to the sum of the preceding terms. Mathematically we have:

$$S = \sum_{i=1}^{\infty} a_i \quad (3.43)$$

Which can be expanded like:

$$S = a_1 + a_2 + a_3 + a_4 + \dots$$

The sum represented by equation 3.43 is indeed one of those infinity bastards. So to help you not feel as hopeless I did about this mathematical concept, let's introduce some practical examples.

Pollen serves as a crucial source of nutrition for bees. It is a nutrient-dense food they use for their growth and development. It can also be used as a signal; bees exchange information about the availability and quality of pollen sources through odor cues on their bodies. They can detect and identify different types of pollen, helping them communicate to other foragers where to find suitable food sources.

Consider a scenario where we have one flower and one bee. Every time the bee goes to the flower, it returns half of the quantity of pollen from the previous trip. They can carry around 20 milligrams of flower dust in their pollen baskets, which are located on their hind legs. OK, what we want to understand is how much pollen a bee

extracts from a flower, and for that, we will use a series:

$$S_N = 20 + \frac{20}{2} + \frac{20}{4} + \frac{20}{8} + \dots \quad (3.44)$$

On the first trip, the yellow and black drone brings back 20 milligrams of pollen; on the second, it will bring back 10, then 5, and so forth, until the flower has no more pollen. We don't know when this will be, so let's assume it happens at infinity, our good old pal. So, equation 3.44 can be written as:

$$S_N = \sum_{n=0}^N \frac{20}{2^n} \quad (3.45)$$

Because we need the total amount of pollen available on the flower, we are required to add up all its elements. That quantity, we will represent by S_N . The assumption was that the bee would extract every bit of pollen the flower has to offer, meaning that as long as some is available, it will bring it back. One way to calculate S_N is via a limit, and because we don't know the exact trip on which the pollen will run out, to be safe, we will let S_N go to infinity:

$$\begin{aligned} & \lim_{N \rightarrow \infty} S_N \\ &= \lim_{N \rightarrow \infty} \left(20 + \frac{20}{2} + \frac{20}{4} + \frac{20}{8} + \dots + \frac{20}{2^N} \right) \end{aligned} \quad (3.46)$$

We currently face a limitation with this equation 3.46 – calculating that limit seems challenging. However, we're not giving up just yet. What we're in search of is a method to transform it into a more approachable form. Ideally, we'd like an expression where we can input values and readily obtain the total sum of the series. To craft such an equation, we must maximize our understanding of this series. For instance, we can explore how each value in the series changes concerning the previous one, such as the shift from a_0 to a_1 :

$$a_0 = 20, \quad a_1 = 10, \quad \frac{a_1}{a_0} = \frac{1}{2}$$

We can do this for the following two terms also:

$$a_1 = 10, \quad a_2 = 5, \quad \frac{a_2}{a_1} = \frac{1}{2}$$

3.4. Get Me Higher and I'll Give You Better Curves: Taylor Series.

What if we have any two consecutive terms of the series? Let's check what happens for the terms with indexes k and $k + 1$:

$$a_k = \frac{20}{2^k}, \quad a_{k+1} = \frac{20}{2^{k+1}}, \quad \frac{a_{k+1}}{a_k} = \frac{\frac{20}{2^{k+1}}}{\frac{20}{2^k}} = \frac{1}{2}$$

We'll call the ratio we just finished deducing r ; in this particular situation, $r = \frac{1}{2}$. Cool, Equation 3.45 can be written as:

$$S_N = a + ar + ar^2 + ar^3 + \dots + ar^N \quad (3.47)$$

Where $a = 20$ and $r = \frac{1}{2}$. We are still stuck with infinite terms, but at least now, everything is defined as a function of r and a . Right, we clearly need to do something to this equation, but what? Our problem is that 3.47 carries on until the end of the world and beyond, so the only way I can see us having some chance is by creating another series with the same length where we can either cancel out terms or find a relationship between them. We have a 's and r 's; we had better stay within these two, as creating new variables would only increase complexity. Now I can add, subtract, divide, or multiply our original series 3.47 by a , r , or both, but because r is the element with exponents, we will multiply S_n by r :

$$rS_N = ar + ar^2 + ar^3 + \dots + ar^{N+1} \quad (3.48)$$

By subtracting equation 3.48 from equation 3.47, we arrive at:

$$S_N - rS_N = a - ar^{N+1} \quad (3.49)$$

This simplifies to:

$$(1 - r)S_N = a(1 - r^{N+1}) \quad (3.50)$$

From equation 3.50, we can deduce the sum of the series up to the N^{th} term:

$$S_N = \frac{a(1 - r^{N+1})}{1 - r} \quad (3.51)$$

For our infinite geometric series, since $|r| < 1$, r^{N+1} approaches 0 as N tends toward infinity. Hence, the sum of the infinite series, denoted as S_∞ , is:

$$S_\infty = \frac{a}{1 - r} \quad (3.52)$$

Substituting $a = 20$ and $r = \frac{1}{2}$ into our formula gives:

$$S_{\infty} = \frac{20}{1 - \frac{1}{2}} = 40$$

So we can expect 40 milligrams of pollen from that given flower. When the limit given by 3.46 is defined, we say that the series converges. This idea of convergence isn't just a theatricality's pivotal. Why? The concept is crucial because convergence assures us that our infinite series has a definitive sum. In simpler terms, even if we're summing up an endless number of elements, the total doesn't just shoot off to infinity. It settles at a particular value. Without convergence, we'd be uncertain if our calculations and approximations hold real-world relevance or are just Greek letters on a piece of paper that can only be used to impress people who don't care about us.

Previously, we touched on how problematic non-polynomials can be. In doing so, we stumbled not only upon the realization of unlimited derivatives but also the fact that we can leverage the information they carry about a function to create polynomial approximations around a point. Given their beneficial properties, especially in differentiation and integration, polynomials will be our focus in the upcoming discussion. Here, convergence becomes paramount, we want our polynomial approximation to align seamlessly with the original function at the given point.

Now, our journey takes us to Mr. Taylor, as we put derivatives, series, convergence, and polynomials to work in synergy, just like the bees. To start deriving this method, we will revert to the non-polynomial function:

$$f(x) = \frac{\sin(\pi x)}{x^2 - 1} \quad (3.53)$$

The derivative of that expression 3.53 is done using the chain rule. I will omit the calculations and provide the final expression:

$$f'(x) = \frac{-2x \sin(\pi x) + \pi(x^2 - 1) \cos(\pi x)}{(x^2 - 1)^2}$$

3.4. Get Me Higher and I'll Give You Better Curves: Taylor Series.

We are just missing one little thing to get this show on the road, a point, say $x = 2.5$. Alright, let's plot the function $f(x)$ around this point:

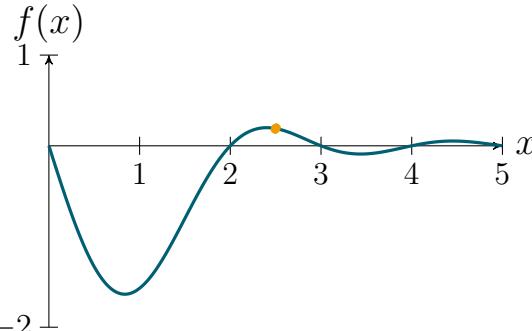


Figure 3.49: The yellow point that we will do some work on.

Eye on the prize; the aim is to approximate a function via a polynomial. We want to go from that nasty division that contains a $\sin(\pi x)$ to something more like this:

$$P(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n \quad (3.54)$$

Equation 3.54 represents a power series, and we can also write it like this:

$$P(x) = \sum_{n=0}^{\infty} c_n x^n \quad (3.55)$$

That is nearly the format we want; the only thing is that equation 3.55 is centered around 0; we want it to be centered around any given point a or, in this particular case, $x = 2.5$:

$$P(x) = \sum_{n=0}^{\infty} c_n (x - 2.5)^n \quad (3.56)$$

The c 's are constants, and 2.5 is the point we want the approximation of the function to be around. The n represents the range of the summation, and it is up to us to select a value for it. From 3.56, we have nearly everything we need to define the polynomial from which we will approximate a function; the only thing missing is the coefficients c 's. We need to find a way to calculate them; for that, we will revert to the simplest form of approximation when $n = 0$:

$$f(x) = c_0$$

OK, we are after an approximation of $f(x)$ around the point $x = 2.5$ so naturally the first thing to include must be $f(2.5)$, meaning that when $n = 0$ our approximation is:

$$f(x) = f(2.5) = 0.19$$

All right, a graph will help us in understanding how good this approximation is:

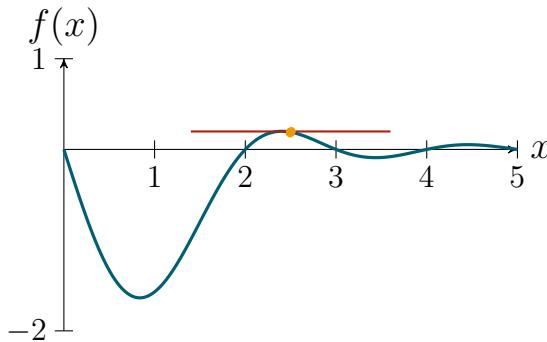


Figure 3.50: Our first approximation! Quality? Below average.

Yeah, we can't say that is the pinnacle of all our work so far, but it is a starting point. And if it sucks? The only way is up. We're currently at a horizontal baseline. To refine our approximation, let's incorporate the some function's curvature. Fortunately, we've got a toolkit filled with information about rates of change and curvatures: the derivatives. And we've got as many as the function permits! Let's kick off with the first one, $f'(x)$. We'll harness this to better understand when the function rises and falls, transforming our basic horizontal approximation into something more insightful. But keep in mind, our endgame is to pin down expressions for the constant c . And given our focus on derivatives, it only makes sense to kick things off by calculating one, doesn't it?

$$f(x) = c_0 + c_1(x - 2.5) + c_2(x - 2.5)^2 + c_3(x - 2.5)^3 + c_4(x - 2.5)^4 + \dots \quad (3.57)$$

Equation 3.57 represents the polynomial approximation of $f(x)$ by a power series. We used 3.54 to expand $f(x)$ around $x = 2.5$. Now we will compute $f'(x)$:

$$f'(x) = c_1 + 2c_2(x - 2.5) \cdot 1 + 3c_3(x - 2.5)^2 \cdot 1 + 4c_4(x - 2.5)^3 \cdot 1 + \dots \quad (3.58)$$

3.4. Get Me Higher and I'll Give You Better Curves: Taylor Series.

Cool, we used two derivation rules, the sum and chain rules. Those multiplications by 1 reflect the chain rule as they are the derivative of $(x - 2.5)$. If we are after what happens near to $x = 2.5$, a piece of good information would be the rate of change near that exact point. We can get this by replacing x by 2.5 in 3.58.

$$f'(2.5) = c_1 + 2c_2(2.5 - 2.5) \cdot 1 + 3c_3(2.5 - 2.5)^2 \cdot 1 + \dots$$

That results in:

$$f'(2.5) = c_1$$

Oh! Double whammy, we have c_1 and information from the first derivative. Let's check our new approximation! This is the one where we expand up until $n = 1$:

$$f(x) = c_0 + c_1(x - 2.5)$$

We now know the value of those two bad boys c_0 and c_1 :

$$f(x) = f(2.5) + f'(2.5)(x - 2.5)$$

To establish the value of $f'(2.5)$, we just have to input $x = 2.5$ into the equation for $f'(x)$. I feel some improvements have been made. Graph, please:

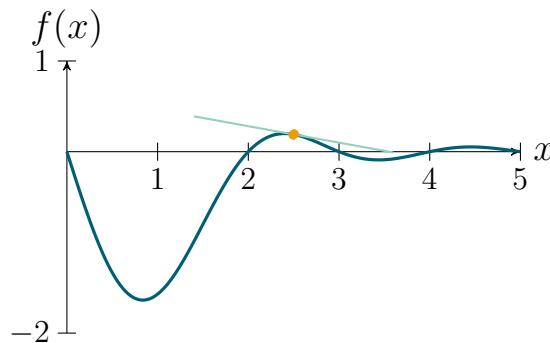


Figure 3.51: The second attempt and we have improvements.

The light blue line represents the new approximation of $f(x)$ around $x = 2.5$. It is still a linear approximation as we have no terms of order higher than 1, but it's already showing some promise compared to the horizontal line we previously had. The refinements came from the information we introduced by including the

first derivative, so in theory, if we get more information on the behavior of $f(x)$ around that point, we will get a better approximation, for example, something around curvature. For that we can use the second derivative:

$$f'(x) = c_1 + 2c_2(x - 2.5) \cdot 1 + 3c_3(x - 2.5)^2 \cdot 1 + 4c_4(x - 2.5)^3 \cdot 1 + \dots$$

If we compute the derivative of the function represented above, we get:

$$f''(x) = 2c_2 + 6c_3(x - 2.5) + 12c_4(x - 2.5)^2 + \dots \quad (3.59)$$

Replacing $x = 2.5$ in 3.59:

$$f''(2.5) = 2c_2 \Rightarrow c_2 = \frac{f''(2.5)}{2}$$

Making our new approximation:

$$f(x) = f(2.5) + f'(2.5)(x - 2.5) + \frac{f''(2.5)}{2}(x - 2.5)^2$$

Will one more term produce a better approximation? Let's use a visual for verification:

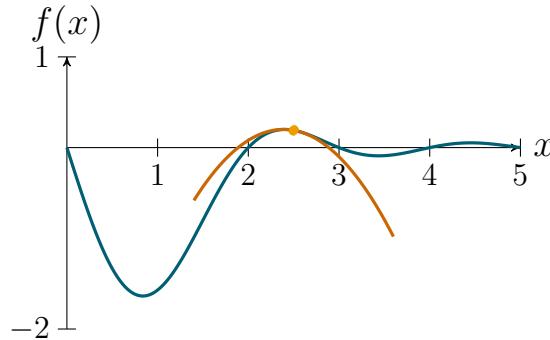


Figure 3.52: We have some curves! No wonder the second derivative has come to play a part in the approximations.

That new approximation, represented by the orange line, is indeed better. Let's try one more term. The procedure will be the same since it seems to be working, so let's derive $f''(x)$:

$$f''(x) = 2c_2 + 6c_3(x - 2.5) + 12c_4(x - 2.5)^2 + \dots$$

3.4. Get Me Higher and I'll Give You Better Curves: Taylor Series.

This means that $f'''(x)$ is equal to:

$$f'''(x) = 6c_3 + 24c_4(x - 2.5) + \dots \quad (3.60)$$

Replacing x by 2.5 in 3.60 we have:

$$f'''(2.5) = 6c_3$$

So:

$$c_3 = \frac{f'''(2.5)}{6}$$

Making our new approximation:

$$f(x) = f(2.5) + f'(2.5)(x - 2.5) + \frac{f''(2.5)}{2}(x - 2.5)^2 + \frac{f'''(2.5)}{6}(x - 2.5)^3$$

Visually:

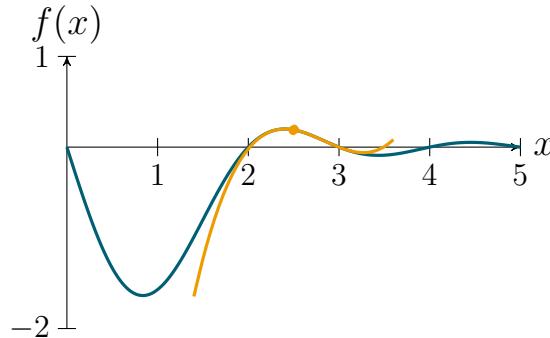


Figure 3.53: One more derivative and better curves, let's go!

The continuous improvement in our approximations, suggests that we are on the right track. Also, this shows that derivatives are potent tools that capture a lot of the function's behavior. Our next step is to formulate a general expression for the Taylor series approximation. We are nearly there, but those denominators 2 and 6 must be examined. For that, let's calculate c_4 . I will omit the calculations as it is more of the same:

$$c_4 = \frac{f'''(2.5)}{24}$$

So:

$$f(x) = f(2.5) + f'(2.5)(x - 2.5) + \frac{f''(2.5)}{2}(x - 2.5)^2 + \frac{f'''(2.5)}{6}(x - 2.5)^3 + \frac{f''''(2.5)}{24}(x - 2.5)^4$$

There seems to be a relationship between the exponents and the denominators:

$$\begin{aligned} 2 &\rightarrow 2 \rightarrow 2! \\ 3 &\rightarrow 6 \rightarrow 3! \\ 4 &\rightarrow 24 \rightarrow 4! \end{aligned}$$

In this representation, the left column lists the exponents. The middle column shows the denominators, which are determined by taking the factorial of the corresponding exponents. The right column explicitly indicates these factorials. Understanding this relationship can assist us in constructing a series expansion of a function around a specific point. With this knowledge, we can generalize a method to approximate functions around $x = 2.5$:

$$f(x) = \sum_{n=0}^{\infty} c_n (x - 2.5)^n \quad \text{Where} \quad c_n = \frac{f^{(n)}(2.5)}{n!} \quad (3.61)$$

With 3.61, we are seeing a representation of a Taylor series, which helps us to approximate functions as a sum of power series where c_n represents the n^{th} coefficient calculated as the n^{th} derivative of the function at $x = 2.5$, divided by $n!$. More generically, we base 3.61 around a given point a :

$$f(x) = \sum_{n=0}^{\infty} c_n (x - a)^n \quad \text{Where} \quad c_n = \frac{f^{(n)}(a)}{n!} \quad (3.62)$$

In this equation 3.62, the series is described around a more general point a , allowing us to tailor (the pun was needed) our approximation to different central points. This flexibility is one of the great strengths of the Taylor series, as it enables more accurate approximations of complex functions in the vicinity of other points. As seen in the previous plot illustrations, the approximation quality decays

3.4. Get Me Higher and I'll Give You Better Curves: Taylor Series.

as we move further from the target approximation point. This creates the need to understand how far we can go, and be sure that we won't be engaging with really poor approximations. We thus need a "safe zone" or a radius where our series approximation remains valid and accurate:

$$R = \frac{1}{\limsup_{n \rightarrow \infty} \sqrt[n]{|c_n|}}$$

Let's shift our focus to the radius of convergence, R . It essentially delineates the interval around the central point (here $x = 2.5$ or a) where our Taylor series will converge. We call that weird limit the "supremum"; it refers to the least upper bound of a set. It is the smallest number that is greater than or equal to every number in a particular set, offering a kind of "ceiling value". In the context of the radius of convergence, it aids us in determining the boundary within which the Taylor series accurately approximates the function, thus establishing a region in which the approximation holds. For the specific case of the Taylor series, we have:

$$R = \frac{1}{\limsup_{n \rightarrow \infty} \sqrt[n]{\left| \frac{f^{(n)}(c)}{n!} \right|}} \quad (3.63)$$

In this formula 3.63, the supremum comes into play, helping us identify the sequence's upper boundary formed by the series' coefficients, c_n . It gauges the "speed" of the growth of c_n and, consequently, delineates the boundary where the series approximation remains reliable, forming a crucial part of defining the radius of convergence.

"Series" can be a bit of an abstract concept, but in reality, that is just a big sum of a sequence of numbers that we define. The Taylor series is more than just an intimidating equation in machine learning. When building predictive models, we sometimes run into complex, non-linear functions that are computationally intense. Well, brother Taylor can help us with this by letting us swap out these intricate functions with simpler polynomials, streamlining calculations without skimping on performance.

When deriving this method, we improved the approximations by adding information about the function provided by the derivatives. Well, that can't be left untouched, and here is the reason why. We've briefly touched upon the "cost function," which helps gauge how well our model performs. We're hunting for a set of parameters that minimize this cost to get that guy at its peak performance. Essentially, we will be looking for the point with the lowest value on the cost function. It turns out that these points happen to be located at the bottom of the valley when functions change behavior, and this is where the derivatives come into play. They will guide us and tell us where to move to so we can find such points.

3.5 Valleys and Hills: A Critical Situation for Points and Their Roots.

Critical points are essentially the points on a given function that are in a critical state and need the defibrillator to zap them before it is too late. On a more serious note, this is where the fun starts. When introducing this topic, we discussed minimum points; these are not the only subject deemed critical; we also have maximum points, inflection points, and roots. The nomenclature of maximum and minimum suggests some "optimality". This is particularly relevant in machine learning, where we rigorously fine-tune algorithms. Such fine-tuning often involves minimizing a cost function, which essentially measures an algorithm's performance. However, the quest for optimization doesn't halt at minimizing errors; it also encompasses maximizing desired outcomes, like increasing a beehive's productivity within given constraints. Moving beyond maxima and minima, we venture into the territory of inflection points, where the function undergoes a subtle yet pivotal shift in curvature. These are the nuanced locations where a function subtly shifts its curvature, marking a change in the narrative of our mathematical plot. Meanwhile, roots, in this specific case, are not vegetables. They are the points where some mapping crosses the x -axis. I invite you to join

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

me and let the derivatives lead the way.

Flying bees are positively charged; in contrast, as flowers are grounded, their charge is negative. The crazy thing is that bees transfer this charge when they visit a flower to pollinate it. When doing so, they will create a temporary positive electrical field around the flower. This electrical phenome is an indicator for other bees to understand that a specific flower was recently visited; therefore, the pollen levels are low, so they are better off going elsewhere. By getting these signals, the bees allow the flowers to replenish their nectar, making pollination more efficient. To model this intriguing bee-pollination phenomenon, let's construct a function that encapsulates this dynamic:

$$\begin{aligned} f(x) : [-1.55, 1.7] &\rightarrow \mathbb{R} \\ x \rightarrow x^5 - 3x^3 + 0.5x^2 + x \end{aligned} \tag{3.64}$$

We can interpret the behavior of $f(x)$ in terms of a bee's nectar gathering. Consider the point $x = 0$ as the central position. Negative values of x correspond to positions to the left of this point, while positive values indicate positions to the right. The output of the function, given by the y -axis, illustrates the electric field's charge. When the function outputs a positive value, it signals that the corresponding flower has been recently visited, suggesting a lower availability of nectar. Conversely, negative outputs suggest the presence of abundant nectar, indicating that it's less likely another bee has visited that flower in the recent past. The plot of the equation 3.64 visually portrays this behavior:

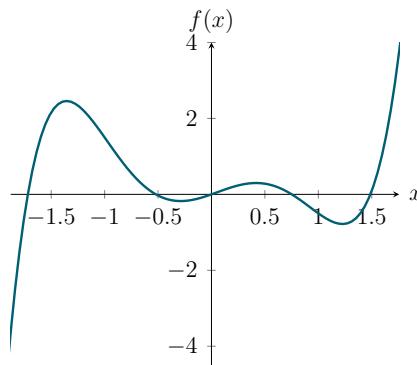


Figure 3.54: Somewhere around those electrical levels, there are flowers.

The fifth-degree monomial, x^5 , is somewhat of a wild one. As it is of an odd degree, it means that both ends of the function head in opposite directions, either upwards or downwards, as we move away from the origin. However, its higher degree compared to x^3 means it changes values more dramatically, creating steeper regions on our graph. This steepness is what leads to the most pronounced peaks and valleys. These prominent ups and downs correspond to local maximum and minimum points, where the function reaches temporary highs and lows before changing direction. The $-3x^3$ factor, a third-degree term, introduces additional curvature and influences the overall shape of the curve. Now, for the second-degree monomial that we have, the one represented by $0.5x^2$; this will create the parabolic shape that, in this particular case, is upwards as the coefficient is positive. Lastly, x contributes to the linear component, influencing the slope and direction of the function.

Our goal is to pinpoint the locations on 3.64 where the function crosses the x -axis, indicating zero energy levels, i.e., where $y = 0$. These are, in essence, the roots; because we set the coordinates images to be zero, these points will occur when the blue line crosses the x -axis:

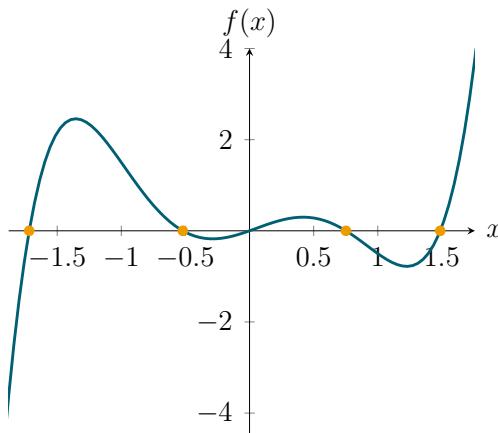


Figure 3.55: Yellow's zest in sun's embrace, energy zero finds its space.

The yellow dots on image 3.55 represent the points we are after, so we must get their coordinates. We do have some formulae that can aid us on the given task, but they work for polynomials whose degree is lower or equal to 4. I am not fond of conditioning, and my love for formulae is at an all-time low; I would much prefer us to have a

method whose application accommodates all cases; considering what we know from the information derivatives provide us with, surely we can come up with something. Perhaps we should zoom in on one of those roots to start a brainstorming session? What about that one to the right, the point between $x = 1.45$ and $x = 1.6$:

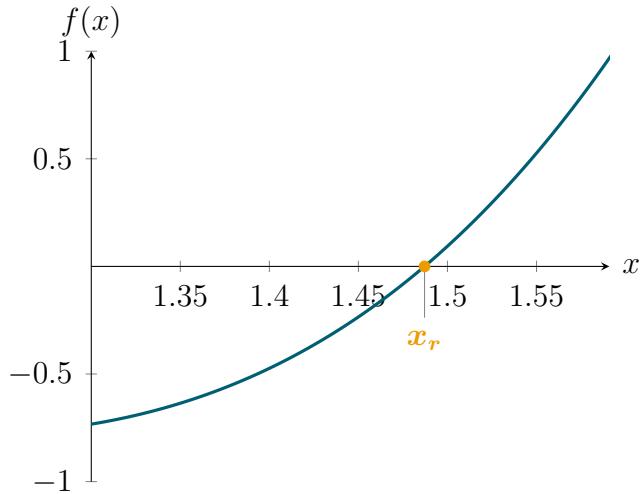


Figure 3.56: Let's check what is happening around one of the yellow guys.

Like most things in life, there is also a trade-off here; we rejected a formula and chose a path of applicability and some simplicity. With this comes the loss of exactitude and the need for a different solution that will revolve around a method of approximation that will provide us, not with the exact root but with a point close to it. But fear nothing, as similar approximation methods have worked for us in the past, for example, when we used the Taylor series; let's not lose confidence now. If we can't be precise on where this damn root is, we can only guess... good luck to us... actually, let's make this more interactive.

Think of a number, and if I can guess it, you subscribe to the mailing list. How about $x_0 = 1.6$? Yes! I'm good at this game. Please subscribe.

Subscribe here!

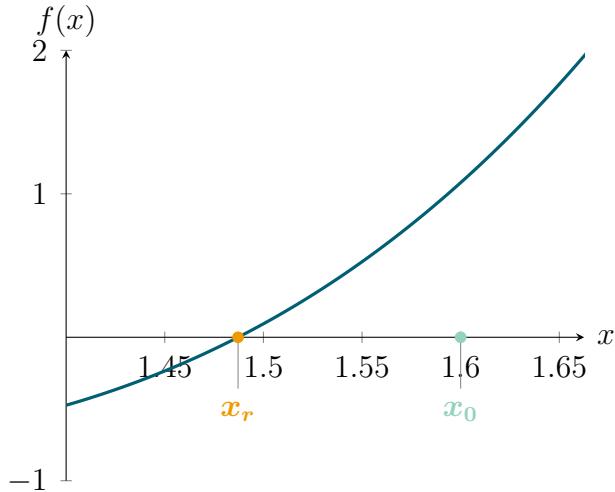


Figure 3.57: Who's that blue fellow? Will it help us? Let's give him a chance.

The light blue dot, denoted as x_0 , marks our initial guess. Our ultimate goal is to accurately determine the position of x_r , the root we are seeking. No, I am not suggesting that we simply guess until we get it right, mainly because we don't yet have a method to accurately determine the coordinates of x_r . What else can we get from what we have so far? We can compute the image of x_0 . It is $f(x_0)$:

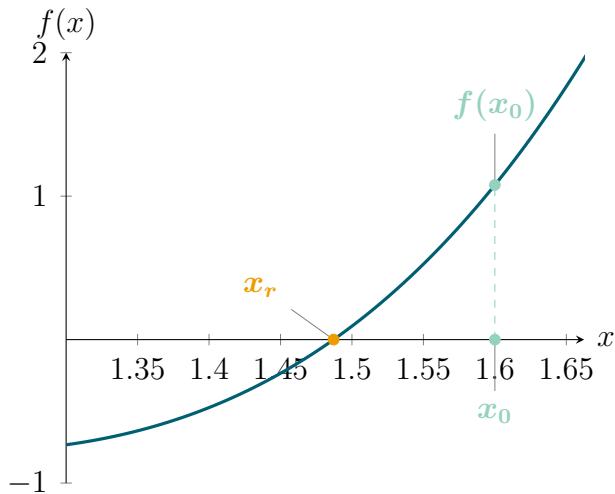


Figure 3.58: Some more information around that new fellow x_0 , its image.

Because $f(x)$ is a polynomial whose degree is too high, we couldn't analytically find the point x_r , and, perhaps somewhat arrogantly, we decided not to use formulae. Well, I did that, but I thought we were a team. So our strategy was to start with a guess that we named x_0 . As the plot above shows, we are still a fair way from the root

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

x_r , represented by the yellow dot. One way to improve our guess would be to have an approximation of $f(x)$ around the point x_0 and then calculate the point at which this exact approximation crosses the x -axis.

Well, well, well, if you were still not convinced about the utility of the Taylor series; there you have it! For this particular case, let's go with a linear approximation, meaning that the approximation of $f(x)$ around x_0 is:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) \quad (3.65)$$

Equation 3.65 is a line; it might not seem like the best approximation but before you jump to conclusions, let's visualize this narrative we are constructing:

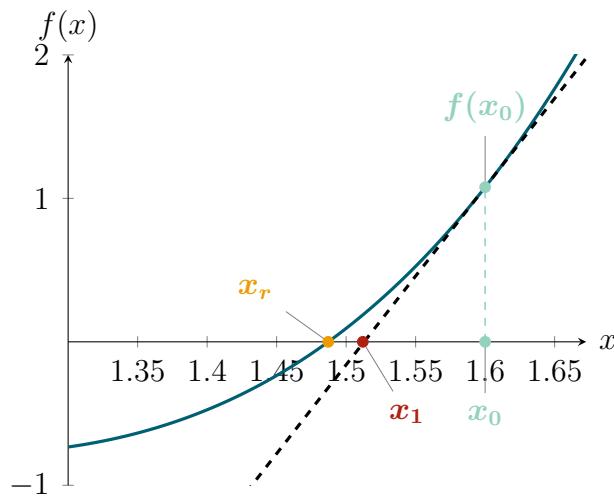


Figure 3.59: We are moving closer to the original yellow fellow.

We have made improvements, as x_1 is closer to x_r than x_0 was. Well, it was unlikely that a random guess would be better than a linear approximation. After all, that dashed line represents a linear approximation of the function $f(x)$, which is the tangent of $f(x)$ on x_0 . Everything seems to be in order, but we still need an expression to compute the value of x_1 . For that, we can work on 3.65 a bit:

$$f(x_0) + f'(x_0)(x - x_0) = 0 \quad (3.66)$$

We are computing the value of the function at the point $(x_1, 0)$ so:

$$f(x_0) + f'(x_0)(x_1 - x_0) = 0$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

We need x_0 , $f(x_0)$ and $f'(x_0)$ to compute x_1 and from those as we stand we just have x_0 , so let's get to work:

$$f'(x) = 5x^4 - 9x^2 + x + 1 \quad (3.67)$$

Easy, some friendly polynomials and a call back to derivation rules, the recipe for temporary mathematical happiness.

Now, for the slope of the tangent line at the point x_0 , fancy wording for the value of that exact point calculated with the derivatives function:

$$f'(x_0) = 5(1.6)^4 - 9(1.6)^2 + 1.6 + 1 = 12.328$$

Now, the only guy missing is $f(x_0)$:

$$f(x_0) = (0.6)^5 - 3(0.6)^3 + 0.5(0.6)^2 + 0.6 = 1.077$$

So x_1 is equal to:

$$x_1 = 1.6 - \frac{1.077}{12.328}$$

That results in:

$$x_1 = 1.512$$

I consider this a significant success for two reasons: First, our new estimate x_1 is closer to the actual root x_r . Second, if we now create a linear approximation of $f(x)$ starting from x_1 , we're likely to get even closer to x_r :

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

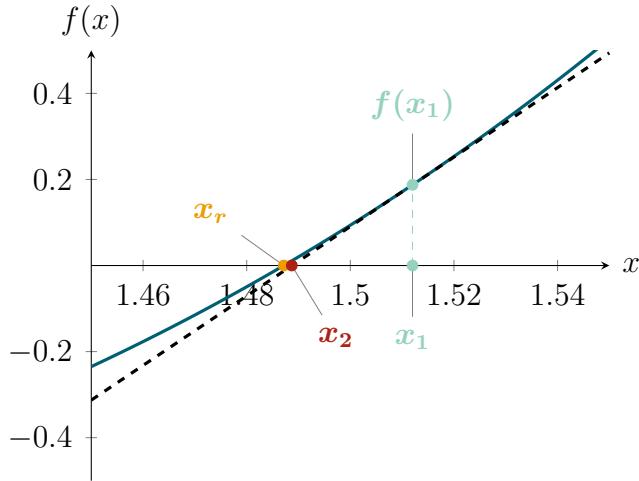


Figure 3.60: One more dashed approximation that moved us even closer to the gold pot.

Good! The solution has improved visually, but let's do the analytical calculations to confirm this. The logic remains the same; we aim to find a local approximation of $f(x)$ around x_1 to then calculate its root. So x_2 can be represented by:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

For $f(x_1)$ and $f'(x_1)$ we just have to input x_1 in equations 3.64 and 3.67 respectively:

$$f(x_1) = 0.187 \quad f'(x_1) = 8.068$$

So:

$$x_2 = 1.512 - \frac{0.187}{8.068}$$

Resulting in:

$$x_2 = 1.488$$

Is this enough? Do we carry on? When do we stop? These are all fair questions, but please allow me to answer them before more show up. We have an iterative method; the iterations refer to each of the steps we took to get a new root, and in this particular case, we have done two, x_1 and x_2 . As with most iterative methods, we need to set some criteria that will allow us to evaluate the new solution in comparison to the old one and decide whether we should do more

calculations or stop. For iterative methods, knowing when to halt the iterations is crucial. Not having stopping criteria can provide us with a bad solution, wasted computational time, or, in some cases, both. We need a way to ensure accuracy and efficiency, so here are some possibilities:

1. $|f(x)| < \epsilon$: Where $f(x)$ represents a generic function and ϵ is the tolerance. We are looking for a root, a value of x where $f(x) = 0$, meaning it makes sense to stop when $f(x)$ is close to zero. This tolerance is usually a small number close to zero, for example, 0.001. But you can choose any ϵ that satisfies your needs. But no, it can't be an oatmilk, ginger, turmeric, kale, chia seeds, and cinnamon organic latte.
2. $\frac{|x_n - x_{n-1}|}{|x_n|} < \epsilon$: This ratio represents a relative change in our approximation of the root, meaning that if, between iterations, the root value does not change more than the tolerance we can stop.

Let's apply point 1 and point 2 to our more recently calculated solution, x_2 . Oh! I nearly forgot, sorry, mister ϵ , you are equal to 0.02. By point 1 we must have $f(x_2) < \epsilon$ so if:

$$f(x_2) = 1.488^5 - 3(1.488)^3 + 0.5(1.488)^2 + 1.488 \approx 0.005$$

This means that:

$$|f(x_2)| < 0.02 \quad f(x_2) \approx 0.005 \Rightarrow f(x_2) < \epsilon$$

Point 1 is verified. Now for point 2:

$$\frac{|x_2 - x_1|}{|x_2|} = \frac{|1.488 - 1.512|}{|1.488|} \approx 0.01 < 0.02$$

Point 2 is also verified, meaning our method has converged. The theme of convergence came up when we introduced series, and within this domain this means we reach a finite number. The meaning is slightly different for iterative methods such as the one we are

currently discussing. Imagine you are tuning one of the strings of a guitar. You can only do a couple of things to that string: increase or release the tension. So, you will pluck the string after each iteration of adding or releasing tension until you think that the sound has finally converged to the desired note. At each attempt, the amount of change in tension that you apply to it should get smaller. Here we managed to iterate until we reached the point $x_2 = 1.488$, which approximates the desired root. There are three more roots, as we can see on the plot 3.55, and to calculate them, we would do this three more times. As the methodology to do so is the same, we will skip that part and instead focus on a general equation for the method.

3.5.1 Zeroing in: Newton-Raphson Method.

I have some sad news, I did not come up with the elegant method used above to calculate roots. Newton originally did, and then Raphson refined it a bit more. I am sorry if I caused any disappointment. Now, the Newton-Raphson method has the following formula:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})} \quad (3.68)$$

To put that equation 3.68 in English, the next iteration is the result of the subtraction of the previous point and the ratio between the values of both the function and the derivative at the same location. Now for a more intuitive perspective, it is interesting to look at that ratio; The term $f(x_{n-1})$ gives us the value of $f(x)$ at that point, and if we seek the root, the function value at this point indicates our distance from the root. The derivative at this point, representing the slope of the tangent to the function at x_{n-1} , acts like a compass; it tells us where we need to move to. So, with this we're essentially “normalizing our error” (how far off our guess is) by using the “rate of change” (how fast the function is changing). This ratio determines how much we should adjust our guess to get closer to the root.

To successfully find a root with this method, we must confirm the following scenarios: The function must be differentiable; I guess this one is obvious but, at the same time, critical. Next, the derivative $f'(x)$ can't be zero at the approximation. Otherwise, we have an undefined division. Also, not everything works with this method, as we also have potential problems. The first one is the initial guess. If it happens to be very far from the root, the method can converge to the wrong one:

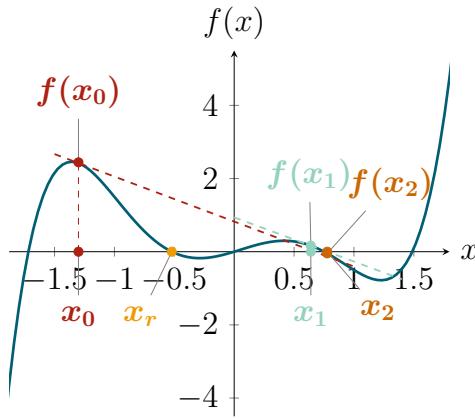


Figure 3.61: Steven, we got the wrong point!

The situation is the same, and we seek the root x_r represented on the plot 3.61. Our initial guess, the red dot x_0 , led us to a different root than the one we were initially targeting after a few iterations. Another big issue that can arise when the initial guess is too far away. This is that the method can converge slowly or not converge at all. For example, if we pick for an initial guess the point $x_0 = -0.287$:

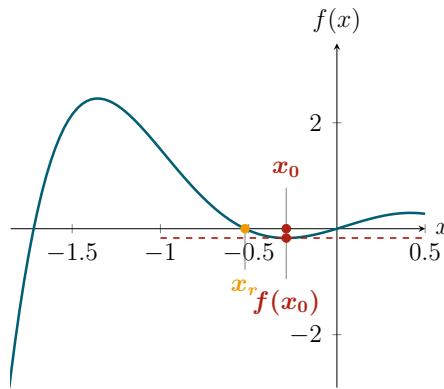


Figure 3.62: This thing is so flat that we can't move anywhere from here.

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

Here, the nearly horizontal tangent line at x_0 results in minimal progress towards the root, indicating that the method may not converge. The initial guess can cause us some problems, that is evident, but we can deal with them in a few ways:

- **A good initial guess** : I know; thank you for stating the obvious, Jorge! Let me explain. If you can visualize the function you are dealing with or have good knowledge of the domain, there is the possibility of nailing an excellent initial guess for x_0 .
- **Multiple initial guesses** : Gladly, we do have computers to experiment with multiple initial guesses. This is a brute force method, but could be a viable solution.
- **Interval analysis** : If we can isolate an interval on the x -axis where a function $f(x)$ goes from negative to positive or vice versa, that will imply that it crosses the x -axis on that interval. The only caveat is that $f(x)$ must be differentiable throughout that interval.

Having traversed the intricacies of Newton's method, we've come to appreciate the importance of derivatives in locating a function's root. We've seen how the derivative, representing the rate of change, acts as a guiding compass that directs our path toward the root. When describing this process, we also showcased some problems that can arise with, one of them being when the derivative at a given point is zero, which would create an undefined division on our formula. Taking a step away from Newton's method, this tells us that the tangent line's slope at that exact point is 0, and we have a horizontal line similar to the one on the graph 3.62. A zero slope also means an instantaneous rate of change of the same value, which implies that, if a function represents the derivative, it will also have one or more roots. If, to this fact, we add that the derivative comes with a plenitude of information, this raises suspicion about whether those points are of value to us. Let's explore this:

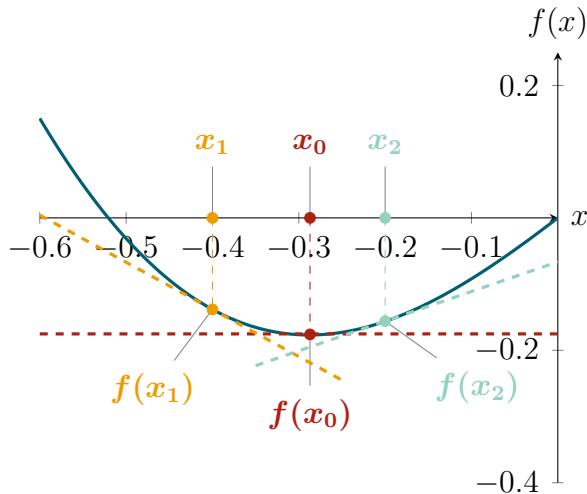


Figure 3.63: What really happens at the peak ?

In graph 3.63, we showcased three points and their respective tangent lines for $f(x)$. What jumps out is that those three tangent lines have different slope values; the yellow line's slope transitions from positive to negative, indicating a negative slope. By contrast, we have the blue line coming from negative to positive values, resulting in a slope with a positive value. Before jumping into any specific case, first, let's consider what it means for a derivative to be negative. A negative rate of change, tells us that as x changes, the function decreases. For positive values of derivatives, changes in the input lead to the values of the function increasing. So, if we have an interval where the slope of the tangent lines goes from negative to positive, as we saw with $[x_1, x_2]$, it also means that $f(x)$ has switched from decreasing to increasing; this happened at the point where $f'(x) = 0$ which represents the horizontal tangent. These guys are called critical points. These points are of great importance as they represent peaks in functions, a point where the function behavior changes. Either they go from increasing to decreasing or vice versa. These particular locations provide a great deal of information.

For example, say that our task is to find the location x where a flower is likely to be, and in addition to that, we also need the measured value of the electric field. Because bees alter this magnitude, the most probable location for the flowers will be where we observe a "peak." When these have positive values, a bee has been

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

there recently. If the value of the electric field dips, it is probably a flower with higher levels of nectar concentration, therefore, more attractive to the bee. In essence, our task is again to find the roots. However, now, instead of doing it to the original function $f(x)$, we will do so to its derivative $f'(x)$. Let me bring that equation 3.67 back, just as a reminder:

$$5x^4 - 9x^2 + x + 1 \quad (3.69)$$

Visually, we should observe the derivative function $f'(x)$ crossing the x -axis just where $f(x)$ has a peak; let's verify this:

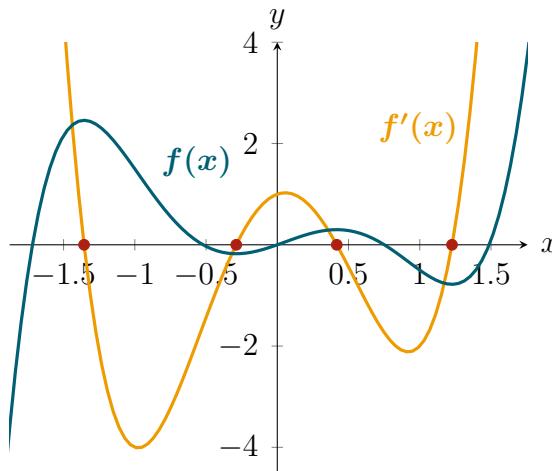


Figure 3.64: The synchrony between the function and its derivative.

Above, in figure 3.64 we have the representation of two functions; the blue line represents our original function $f(x)$, while the mellow yellow fellow is its derivative, the function whose equation is defined by formula 3.69.

Those red dots represent points where the derivative $f'(x)$ is equal to 0. The graph confirms we are on the right track, as the roots of $f'(x)$ align with the peaks and valleys of $f(x)$.

We should probably use the proper terminology instead of just calling them 'peaks'. The points that represent a period of positive growth followed by a period of decreasing values are called maxima (or singularly, a maximum). They can be visualized as being at the top of a mountain. The points at the bottom of the valleys, rep-

resenting a period of decreasing values followed by positive growth, are called minima (or singularly, a minimum):

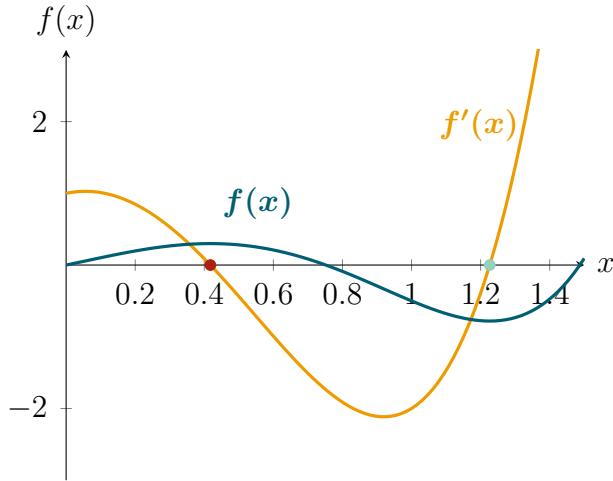


Figure 3.65: A valley and a peak.

Graph 3.65 zooms in on a section of 3.64; it showcases one example of a maximum and minimum. The root of the derivative, represented by the red dot is a maximum whereas the one that is a light blue color is a minimum. These type of points play a crucial role in the field of machine learning. As stated before, machine learning is about optimising tasks; to do so, we will be on the lookout for maxima or minima, depending on the problem in hand. If we are training a model to predict an event, we will want to minimize cost. Also, sometimes we might want to maximize the amount of clicks that an ad will get.

If we are still looking for roots, the Newton-Raphson method will also apply. After all, the derivative in this case is still a function, which is what the technique requires, alongside the conditions we listed before. Most mathematical methods are blind to the input, meaning a function for this iterative technique is simple a function. It does not care if it comes from derivation or not. This is a current theme in mathematics, and a situation that allows us to get creative, meaning we can use the same methods for different purposes.

So, if the Newton-Raphson method receives a function and an initial guess to gives us a root, we can adapt its formula when we are working with the derivative of $f(x)$ instead of $f(x)$ itself:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

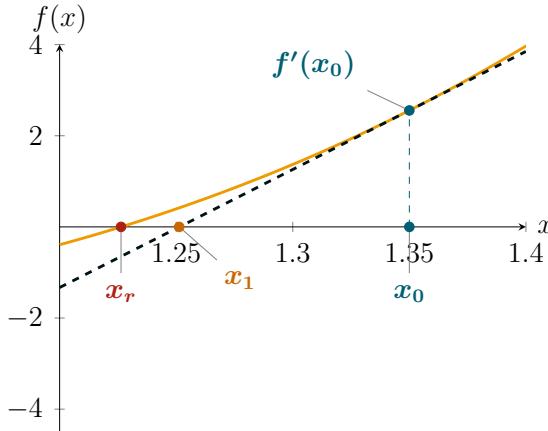


Figure 3.66: Do we really need this whole thing again? I mean another dashed line...

Same method, same procedure. What this means is that we will select one of the roots, an initial guess and create a linear approximation of the function in question around that given point. Then the location where this newly created line crosses the x -axis is the new estimation for the root. Cool! For the equation of the linear approximation, well that is nearly the same but, instead of $f(x)$ we have $f'(x)$:

$$f'(x) = f'(x_0) + f''(x_0)(x - x_0) \quad (3.70)$$

We just replaced $f(x)$ with $f'(x)$ and because we need the derivative of $f'(x)$ we are presented with $f''(x)$. Now if we wish to have a way to compute x_1 we manipulate equation 3.70 just as we did before:

$$f'(x_0) + f''(x_0)(x_1 - x_0) = 0$$

Similarly we have that:

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

That generalises as:

$$x_n = x_{n-1} - \frac{f'(x_{n-1})}{f''(x_{n-1})} \quad (3.71)$$

And because the roots of the derivatives are the critical points that can include maxima and minima, I present you with our first optimization method. The excitement in the air is contagious, as we just opened the gates for the world of optimization, but let's not

forget that we still must try equation 3.71, at least once. We need an initial guess, say $x_0 = 1.35$, and the second derivative of $f(x)$:

$$f''(x) = 20x^3 - 18x + 1 \quad (3.72)$$

The next step will be to calculate both $f'(x_0)$ and $f''(x_0)$:

$$f'(1.35) = 2.55 \quad f''(1.35) = 25.91$$

We are now ready to compute x_1 :

$$x_1 = 1.35 - \frac{2.55}{25.91} = 1.25$$

Suppose we follow the same methodology, meaning defining a stopping criteria and keeping iterating until the method converges. In that case, we will have a good estimation for one of the roots of $f'(x)$ and, consequently, the location of a critical point. Contrary to the exercise done before on finding the root of $f(x)$, with this case, we still need to include precisely what type of critical point this is. Is it a maximum? Or is it a minimum? These are important questions that require definite answers because if we need a minimum and come up with a maximum... oh boy, that would be a painful day. Imagine you're tasked with minimizing marketing spend on a campaign. Identifying a maximum instead of a minimum could lead to an undesirable outcome, highlighting the importance of accurate critical point identification. Well, this raises the opportunity for further knowledge acquisition, and what a privilege this is! Alongside this methodology, we need another piece of information that will aid in identifying the root we find.

Gladly a rollercoaster engineer, just showed up, someone who knows how vital curves are. I am talking about the second derivative, a concept we know is linked to curvature. Because of that, it can provide us with the extra information we need to distinguish between a maximum and a minimum. How so? Let's start with a visual interpretation of it:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

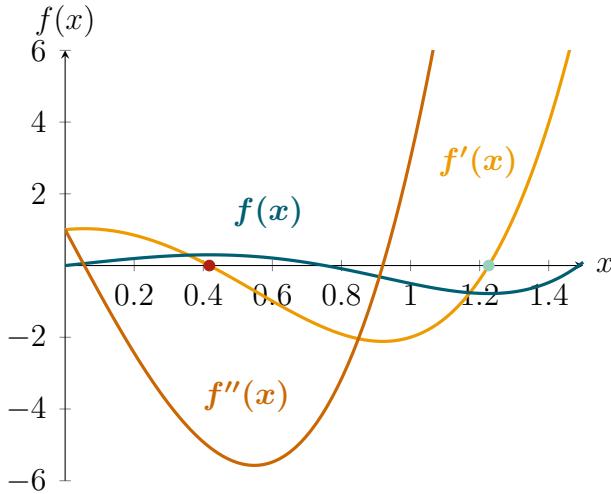


Figure 3.67: Yes, get in there! The curvature information that the second derivative provides.

In graph 3.67, we have zoomed in on a section of $f(x)$, represented by the blue line. Two critical points are highlighted in red and light blue. The derivative, $f'(x)$, represented by the yellow line, intersects the x -axis exactly at these critical points. Observing the behavior of $f(x)$ around the red dot, which resembles the peak of a mountain, we can infer that this point is a local maximum. Conversely, the light blue point sits within a valley, indicating a local minimum. This observation is corroborated by examining the behavior of $f'(x)$. For instance, at the red dot, the derivative is positive to the left, signifying an increasing $f(x)$. However, to the right of this point, the derivative is negative, implying a decreasing trend in $f(x)$.

Around the other root, we observe a different behavior. The yellow derivative function goes from negative to positive, meaning that the function $f(x)$ goes from a period where the y-value declines into a period of growth. This behavior changes after the root, creating a \cup shape as opposed to the \cap observed on the other point we studied before.

Visually we can distinguish between maxima and minima. But this is because we have a tiny example with one variable. You will most likely be working with feature sets with multiple dimensions, so applying this visual "technique" won't be possible if we have to dis-

cern between a maximum and a minimum. With the first derivative, we know where the critical points are, so we need a way to describe the behavior of this function around those points. By knowing the values to the right and left of the roots, we can identify which shape we are dealing with. Well, if we used $f'(x)$ to describe the behaviour of $f(x)$, we can extend this concept to $f'(x)$ by studying its behavior, making use of $f''(x)$, the second derivative.

We will be examining how the rate of change itself is changing. If the concept of growth of a rate of change is still abstract, let's provide some context. The x -axis acts like the "boundary" between positive and negative values for y , and our goal is to study what happens around the roots of the derivative, meaning that this derivative function will cross the x -axis at some point. Traversing the horizontal line from below or above implies that the first derivative will either increase or decrease. By looking into the transition in the signal of $f'(x)$, we can infer the behavior $f(x)$ is displaying around the root; positive to negative corresponds to a \cap shape and negative to positive corresponds to a \cup shape. If we do the same analysis, but now wish to understand what happens to $f'(x)$, we could do so by using $f''(x)$:

- If $f'(x)$ is decreasing $\Rightarrow f''(x) < 0$ then $f'(a) = 0$ is a maximum point at a and $f(x)$ has the shape of a \cap , where a belongs to the domain of the function.
- If $f'(x)$ is increasing $\Rightarrow f''(x) > 0$ then $f'(a) = 0$ is a minimum point at a and $f(x)$ has the shape of a \cup , where a belongs to the domain of the function.

Well, now we understand the value of studying the signal of the second derivative $f''(x)$ but should we have any interest in the roots of that curvilinear specialist? Let's plot it alongside $f(x)$:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

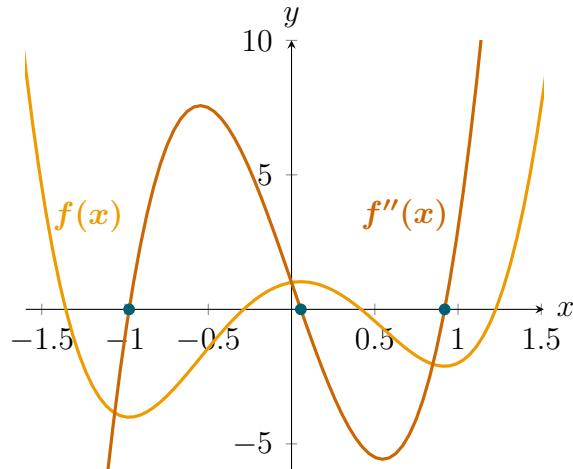


Figure 3.68: Do you think $f'(x)$ is jealous? I mean, here we have $f(x)$ and $f''(x)$ hanging out by themselves.

Question number 12 for 50,000 euros; What is the name of the blue dots representing the location of where $f''(x)$ crosses the x -axis in graph 3.68 ?

- (A) A horse with no name (B) Bananas
- (C) Montgomery Burns (D) Roots

The correct answer is D! Now, to understand what these roots of the second derivative signify, we can refer to the knowledge we just gathered from the analogous points in the first derivative. These points correlate to locations where the tangent line to the function lies horizontally, indicating a halt in the function's upward or downward trajectory, potentially earmarking local maxima or minima, locations where $f(x)$ will change its behavior.

If we think about the first and second derivatives the same way, these specific coordinates mark points where $f'(x)$ transitions between periods of increasing and decreasing values. So, when $f''(x) = 0$, we find critical points for $f'(x)$, which are potential locations where the rate of change of $f'(x)$, and therefore the curvature of $f(x)$ is changing. These points are candidate inflection points, where the function $f(x)$ could be switching from a bowl shape to a dome shape or vice versa, and there is thus an inflection!

The critical thing is that if $f''(x) = 0$ creates a candidate for an inflection, then we choose via a democratic system to elect or neglect the candidate. Here, instead of using ballots, we check the sign of $f''(x) = 0$ around the point where it equals zero :

- **Confirmed Inflection Point:** If $f''(a) = 0$ and $f''(x)$ changes sign around $x = a$, then $x = a$ is an inflection point, indicating a change in curvature of $f(x)$.
- **Not an Inflection Point:** If $f''(a) = 0$ but $f''(x)$ maintains its sign around $x = a$, then $x = a$ is not an inflection point, meaning the curvature of $f(x)$ remains unchanged. Basically, when $f''(x)$ does not cross the x -axis, e.g: $f''(x) = x^2$.

I remember being confused about these inflection points almost as much as when I tried to understand why my mother stated that it was hot 50 times on the same summer day. I really thought that she could make the air cooler with her words, but I was wrong.

You might wonder, aren't inflection points the same as maximum and minimum points? Well, an inflection point occurs where the function changes its curvature, meaning that they may well be maximal or minimal but not always. Check this out:

$$f(x) = x^3$$

Its first and second derivatives are given by:

$$f'(x) = 3x^2, \text{ and } f''(x) = 6x$$

To find potential inflection points, we set the second derivative to zero:

$$f''(x) = 0 \Rightarrow x = 0$$

At $x = 0$, $f'(x) = 0$, indicating a possible minimum or maximum. However, if we analyze the sign of $f'(x)$ around 0 (e.g., at $x = -0.1$ and $x = 0.1$), we see that the sign does not change, hence $x = 0$ is not a local minimum or maximum. But since $f''(x)$ does change its

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

sign at $x = 0$, we know that $x = 0$ is an inflection point, showcasing a change in the curvature of $f(x)$ but not a local extremum.

These points can be as dangerous as tequila shots. While drinking them, life's a flower, but just for a short period until you black out and wake up with a headache and plenty of regrets. Because inflection points represent changes in curvature, they can deceive optimization algorithms into thinking they have reached an optimal point, which is a big problem. Much like the deceptive nature of tequila shots, inflection points can mislead optimization algorithms. Recognizing their existence is crucial in careful optimization.

As it turns out, the derivatives are a great tool for discovering not only critical points, i.e., maxima and minima, but also for understanding the shape of the function, in the calculation of roots and lastly the inflection points. We still need to know where those flowers are! Remember? That was our task. OK, the electric field represented by $f(x)$ will probably peak either where the pollen was taken, or where there is a flower full of nectar. These are the critical points, and we can get their location by examining the roots of the first derivative, $f'(x)$. This will only provide us with the x -coordinate, from which we can calculate the y -coordinate.

However, we won't know if it is a maximum or minimum. For that, we can analyze the second derivative, as we know that the signal of $f''(x)$, when calculated at the root of the first derivative, will give us the remaining puzzle piece. If it is negative, it means the function has a \cap shape; therefore, we are in the presence of a maximum. For positive values, we will have a \cup and hence a minimum.

Right, we need the roots of $f'(x)$. We have $x_4 = 1.221$. We got this by using the Newton-Raphson method. I will provide the other three: $x_3 = 0.4176$, $x_2 = -0.288$, and $x_1 = -1.356$. We need the sign of the second derivative on those specific points. For that we will individually input x_1 , x_2 , x_3 , and x_4 into equation 3.72. Let me remind us of that equation, as it is a charming one:

$$f''(x) = 20x^3 - 18x + 1$$

What about a little table with the three function $f(x)$, $f'(x)$, and $f''(x)$ and their behaviour at the points ?

	x_1	x_2	x_3	x_4
	-1.356	-0.288	0.4176	1.226
$f(x)$	2.458	-0.176	0.299	-0.780
$f'(x)$	0	0	0	0
$f''(x)$	-24.458	5.706	-5.060	15.8125
Point	maximum	minimum	maximum	minimum

Table 3.1: A cool table with the critical points details.

Great, so the two maximum points represent flowers recently visited by bees: $x_1 = (-1.356, 2.458)$ and $x_3 = (0.417, 0.299)$. For some juicy flowers full of nectar, we have the locations of the minima: $x_2 = (-0.288, -0.176)$ and $x_4 = (1.226, -0.780)$. Just remember the shape of the plot at that point, the \cup for positive values and the \cap for negative ones. We have to introduce a mathematical definition for those two points. Let's select one maximum and one minimum from the four we identified in the table above:

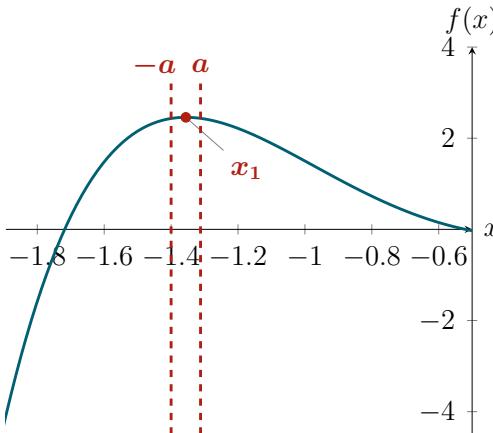


Figure 3.69: The peak and the maximum point dressed in a red jacket.

In graph 3.69, we have the same $f(x)$ function, but it is a zoomed-in version around the point x_1 . The red dashed lines represent an interval of size $2a$, and we say that x_1 is a maximum if:

$$f(x_1) \geq f(x) \quad \forall x \in [x_1 - a, x_1 + a]$$

The image of the x_1 has to be greater than any point around it, which is what has to happen for us to have the \cap shape. If we do something similar for the minimum x_4 :

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

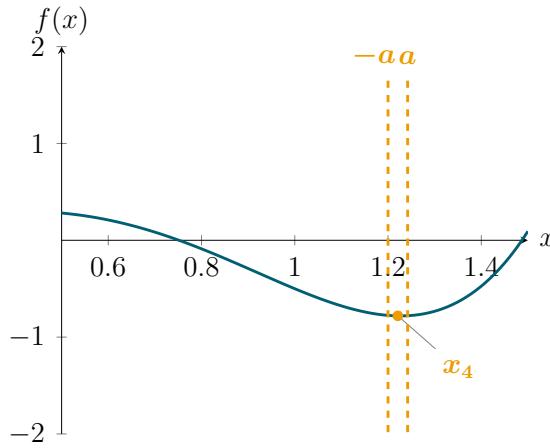


Figure 3.70: The valley and the minimum point dressed in a yellow jacket.

On the representation 3.70, we are looking for a point whose image is lower than all the other points in the neighbourhoods of x_4 , so we can define it like this:

$$f(x_4) \leq f(x) \quad \forall x \in [x_4 - a, x_4 + a]$$

A closer look at table 3.1 shows that one of the maxima has a higher value than the other. Also, one minimum has a lower value than its pal. We call the maximum point with the highest value the global maximum, and you can guess that the global minimum is the minimum point with the lowest value for y .

We began by devising a method to determine the roots of a function. Through this exploration, we understood that the same technique could be applied to the first derivative, given that a derivative can be a function in its own right. With the realization that the roots of $f'(x)$ represent the critical points of $f(x)$, it became clear we had a dual-purpose method that can be used both for finding roots and for optimization. As we progressed, we applied the Newton-Raphson method to the first derivative, leading us to the second derivative, $f''(x)$. This second derivative was crucial in serving as a test with which to ascertain the nature of the critical points we identified earlier. Were they maxima or minima? However, a challenge becomes evident. The use of the Newton-Raphson method for optimization introduced a second-degree process due to the need to compute the second derivative. When working with larger datasets, such computational demands become a concern. This raises the question:

”Can’t we harness all the essential information from just the first derivative?”

To frame this in a more intuitive manner, imagine I’m a bee, whose one task is to find the richest source of nectar. Essentially, I need to locate the global minimum. If I were to randomly select a point on a plot and determine its derivative, this value would reveal the rate at which the function changes at that specific location. This rate of change is visually represented by the slope of the tangent line drawn at that point.

When I compute the derivative at any given point, it hints at the direction of the minimum. A positive derivative indicates growth in the function; so, if my quest is to find a minimum, I should navigate in the direction in which the function decreases. Conversely, a negative derivative suggests that I should proceed in the other direction, which will lead me towards decreasing function values.

The magnitude of our steps can be dictated by the value of the derivative. We must be cognizant of the fact that, at a critical point, the rate of change is zero. Therefore, points in close proximity to this critical juncture have tangents with milder slopes compared to points situated further away. In essence, if I’m using the slope as a travel guide, larger steps need to be taken further away from the minimum, while smaller, more cautious steps are taken as we approach the critical point, given the variance in slope magnitudes. However, it’s important to note that this rule presumes the derivative is defined; in instances of discontinuity, cusps, or vertical tangents where the derivative may not exist, alternative strategies must be employed. Despite these exceptions, this methodology appears to be a promising approach:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

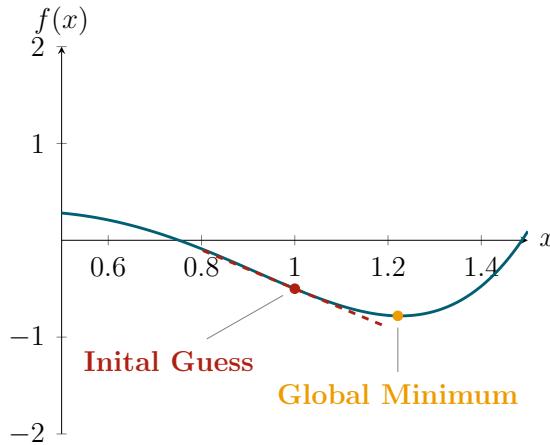


Figure 3.71: Is this a new way to get to a minimum ?

Graph 3.71 is the same as 3.70, the one we use to define the global minimum, which is again represented by the yellow dot. The red dot is our initial guess, as we are using the same technique we used to kick-start the other iterative method we just went over. Spoiler alert, this is another method with iteration involved. We have $x_0 = (1, -0.5)$ for the coordinates of our red guess.

The **r(ed d) ash(ed)** line represents the tangent line to $f(x)$ at the point x_0 . Because the slope of the tangent line at x_0 has a negative value, we must follow the function $f(x)$ from the left to the right, and the step size will depend on the slope's value. OK, so let's compute the derivative at that point and start with the calculations:

$$f'(x) = 5x^4 - 9x^2 + x + 1$$

So $f'(x_0)$ comes as:

$$f'(x_0) = 5(1)^4 - 9(1)^2 + 1 + 1 = -2$$

Cool, so by subtracting the slope from x_0 , we will move towards our goal. Let's call this new point x_1 :

$$x_1 = x_0 - f'(x_0) = 1 + 2 = 3$$

Our minimum lies to the right, hence the need for subtraction. With the new point as $x_1 = 3$, we seem to have overshot the yellow dot. This oversight can be attributed to the pronounced slope value at x_0 . One way to mitigate this would be to introduce a learning rate, something that we can use to control the speed at which we move along, the equivalent of saying, "Easy tiger" but in mathematical terms. In this specific case, we want to "slow down" the algorithm, but equally, we can make it move faster if need be. Say that instead of 1, which is, in essence, what we had on the previous calculations, we pick 0.05:

$$x_1 = x_0 - 0.05 \cdot f'(x_0) = 1 - 0.05 \cdot (-2) = 1.1$$

Much better!

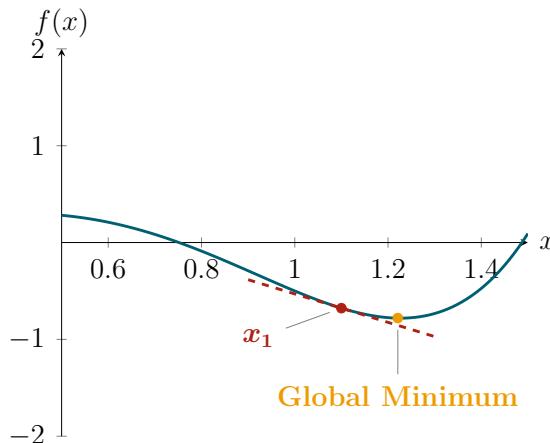


Figure 3.72: It indeed seems as we are getting closer.

We are getting closer; if we keep iterating, I suspect we will soon either be at our destination or at least very nearby, so let's proceed. For x_2 :

$$f'(x_1) = -1.469$$

So:

$$x_2 = x_1 - 0.05 \cdot f'(x_1) = 1.1 + 0.07 = 1.17$$

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

If we plot this new approximation:

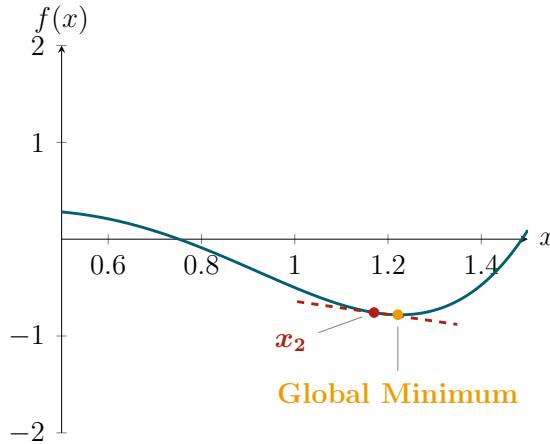


Figure 3.73: Oh wow, we are nearly there. It is working.

Before doing any more calculations, I propose comparing the slopes of the three tangent lines we calculated so far. We have x_0 , x_1 and x_2 so:

$$f'(x_0) = -2$$

For x_1 :

$$f'(x_1) = -1.469$$

Finally x_2 :

$$f'(x_2) = -0.706$$

The slope, which is our chosen measure of inclination, decreases in value as the number of iterations increases.

Consequently, we will take more significant steps when we are further away from the minimum, whereas, as we get closer to the intended point, the step size will get smaller and smaller due to the decreasing slope value. That will save us time, which is always good. Now you tell me if this isn't an elegant solution! Because half the job of computing x_3 is already done, why not calculate it?

$$x_3 = x_2 - 0.05f'(x_2) = 1.17 + 0.03 = 1.205$$

Let's visualise how close we are to the yellow dot:

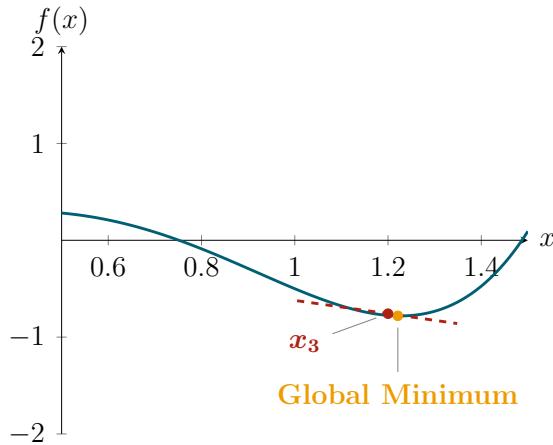


Figure 3.74: Are they kissing ?!?

I don't know about you, but I am curious about x_4 , so allow me to calculate it. You can sit back and relax:

$$f'(x_3) = -0.392$$

Come on x_4 !

$$x_4 = x_3 - 0.05f'(x_3) = 1.2 + 0.02 = 1.22$$

Our prior knowledge tells us that the value for the yellow dot is 1.226, and with x_4 , we have ended up at 1.22. These two are very close to each other! Indeed, the essence of iterative methods like the Newton-Raphson is that we do not need to know the exact location of the minimum point in advance. If we already had that information, there would be no need for an approximation process. Furthermore, this algorithm is often applied to complex functions of higher dimensionality, which are beyond our ability to visualize. This complexity necessitates well-defined stopping criteria, similar to those used in the Newton-Raphson method. We are also in the process of creating a general formula, so let's define this thing rigorously.

3.5.2 The Original GPS: Gradient Descent.

That is right; I was introducing the gradient descent, a fundamental optimization algorithm in machine learning due to its efficiency and simplicity, so let's ensure we cover it well. We say "optimization" alongside "descent" because this guy will be trying to find a minimum. Equally, we can have a gradient ascent to find a maximum the reason that the most popular of these semi-identical twins is the gradient descent is its application in minimizing cost functions. The reality is that the main advantage of applying this methodology to a function with only one variable is for us to have a good foundation, as later, when we delve into higher dimensional functions, we will revisit this topic, and this knowledge will be of value.

So, what are the equation, the stopping criteria, and the problems with this? Yeah, nothing is perfect; we have to see the good things about these heuristics and work with them while tackling the issues that might arise. Right, equation time:

$$x_n = x_{n-1} - \eta f'(x_{n-1}) \quad (3.73)$$

Here n defines the number of iterations and η the learning rate. The following approximation of a critical point equals the previous one minus a learning rate times the slope of the tangent line at the last point. These are the steps:

1. **Select an initial point:** in many cases, this is done randomly as this helps explore different regions of the space. If you have knowledge in the domain, this guess can be made in a more educated fashion. In cases of functions with both local and global minima, this initial guess can make the algorithm converge to a local minimum; for example, say that instead of $x_0 = 1$, we define $x_0 = -0.5$:

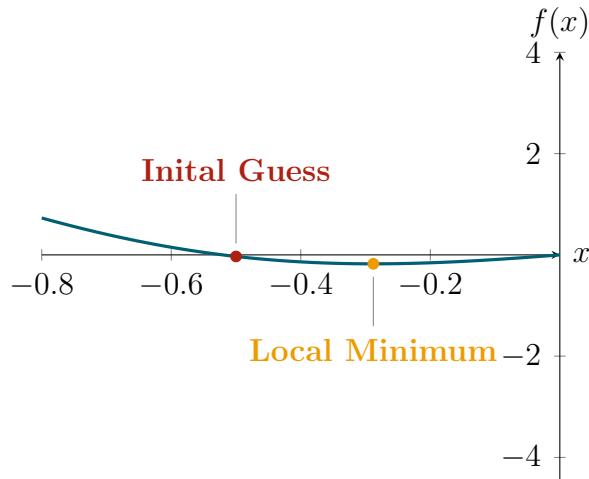


Figure 3.75: An initial guess that would lead us to the wrong yellow fellow.

Figure 3.75 represents a close-up version of the section around the other minimum of $f(x)$, because we took a random guess closer to that point, which happens to be a local minimum, if we use the gradient descent, we risk it converging to this point instead of the to the global one.

To overcome this type of situation, where we can't rely on domain knowledge, running the algorithm several times with different starting points is a viable strategy.

2. **Define a learning rate:** for larger values, we will have faster convergence. Still, we risk overshooting, just like when we had this rate set for one, the case where we simply subtracted the slope of the tangent. This can cause the behavior of the algorithm to be more chaotic as it can bounce back and forward, causing problems with convergence. On the other hand, a lower learning rate will cause slower convergence. Nonetheless, it will reduce the risk of overshooting and lead to fewer fluctuations. Some typical values for these metrics are 0.01, 0.001, and 0.0001.

The learning rate could mitigate the initial guess problems, i.e., if it is large enough, we could jump over the local minima and hopefully converge to the global minima. Although this does not happen often, it is worth a try. Now let's set $\eta = 0.5$ and $x_0 = -0.5$. After 100 iterations I did with a computer, we find that $x_{100} = 0.247$, meaning we are clear! We jumped over

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

the local minima at $x = -0.288$, maybe 100 more iterations, and we will get closer to the global minimum $x = 1.226$. After 200 iterations we get $x_{200} = -0.173$, this is no bueno. We just came a long way in the other direction; maybe with 300, we will get back on track, $x_{300} = -0.244$. This could be better, maybe with x_{600} ? Damn it! $x_{600} = 0.09$. Let's plot all this madness:

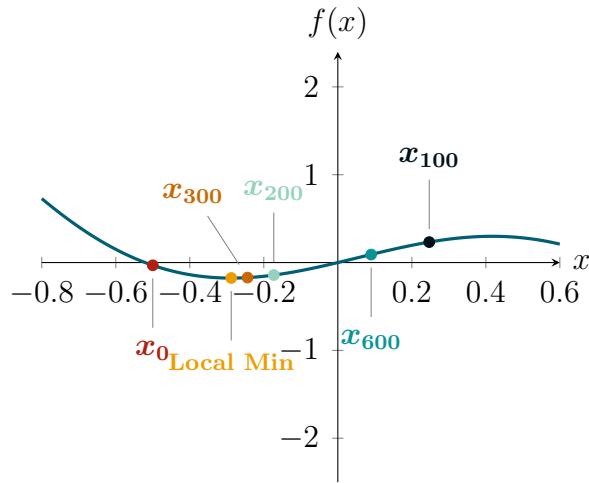


Figure 3.76: The dance off the iterations.

An algorithm that oscillates back and forth will probably not converge to the global minimum. A learning rate that is too large resulted in overshooting across the valley, moving from the starting point x_0 directly to x_{100} . However, the jump wasn't sufficient to bypass the local minimum.

When we evaluated the rate of change at x_{100} , represented by $f'(x_{100})$, we found it to be 0.716. Since this is positive, it indicates that the function is increasing around that point. Following the gradient descent's instructions, we moved to the left, aiming to find a minimum. At the next evaluation point, x_{200} , the rate of change was $f'(x_{200}) = 0.237$, signalling a shift to the left.

This shift brought us closer to the local minima again, which raises concerns about the algorithm's trajectory towards the global minimum.

To address this oscillation issue, we increased our iterations up to x_{600} , but still ended up overshooting. The lesson here? The journey involves a lot of experimentation to pinpoint an optimal learning rate, ensuring that the algorithm converges smoothly.

3. **Update the value of the estimate x_n :** here we calculate the derivative at x_{n-1} and get the value for our following estimation by using equation 3.73
4. **Check the stopping conditions:** as with the Netwon-Rasphon method, we need some criteria to stop iterating. For the gradient descent, we have two possible ways:
 - The goal is for the algorithm to converge, so if the new solution is similar to the old one, it suggests that we will not get a significant improvement in the estimates going forward. A strategy to refine this procedure and avoid extensive iterations is to establish a threshold, denoted as ϵ . The iterations are terminated once the absolute difference between successive value; $|x_n - x_{n-1}|$, falls below this threshold, indicating that the difference between consecutive solutions is sufficiently small.
 - Another way to stop the show is to define a number of iterations. Setting a fixed number of iterations can serve as a safeguard against endless processing, especially in cases where the algorithm fails to converge rapidly. If your gradient descent takes a while to converge, it might indicate that something is not working right; for example, consider the following plot:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

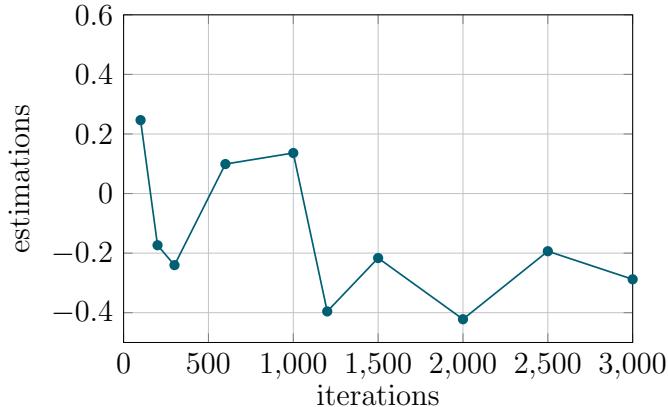


Figure 3.77: A non-converging gradient descent run.

Graph 3.77 represents the same case we explored when we described the challenges with the learning rate, the one we iterated up to x_{600} . As we can see, the solutions are bouncing up and down, suggesting that the algorithm is struggling to converge. This is because we choose a learning rate that is too large for this task; for example, if instead of $\eta = 0.5$, we go for $\eta = 0.05$:

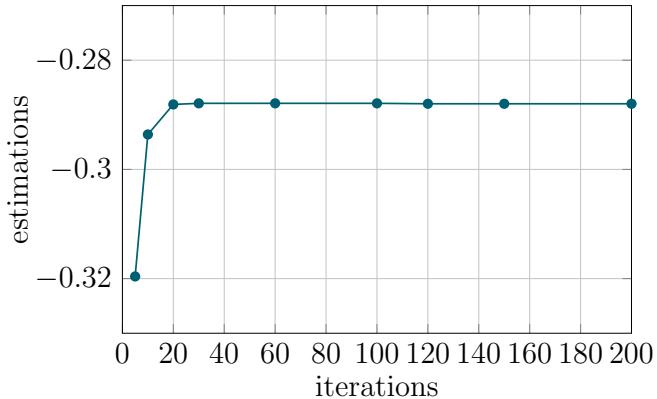


Figure 3.78: A converging gradient descent run.

In this instance, the algorithm converges and does it fast; we would only require around 30 iterations. The problem is that we got stuck in a local minimum; we could, for example, keep the learning rate of $\eta = 0.05$ and experiment with different starting points.

5. If the stop conditions are not met, repeat steps 3 and 4.

We started our journey through optimization techniques with two methods with unique merits: the Newton-Raphson method and the gradient descent. However, it's crucial to understand that neither approach is a guaranteed path to the global minimum. Their performance can be significantly influenced by factors like the initial guesses and the nature of the function they're navigating. Sometimes, they may get trapped in local minima or may not converge quickly due to an unfortunate starting point. So, local and global critical points can be a source of some headaches. However, it is not all bad news because we have a trick! There is a type of function with some properties worth exploring. I am talking about convex functions.

3.5.3 Dare You Cross Me, Line? Convexity.

The notion of convexity can be explained by a visual definition. If we draw a segment line (a part of a line) between any two distinct points on the graph it will implicate that all values of the same function lie below that segment line:

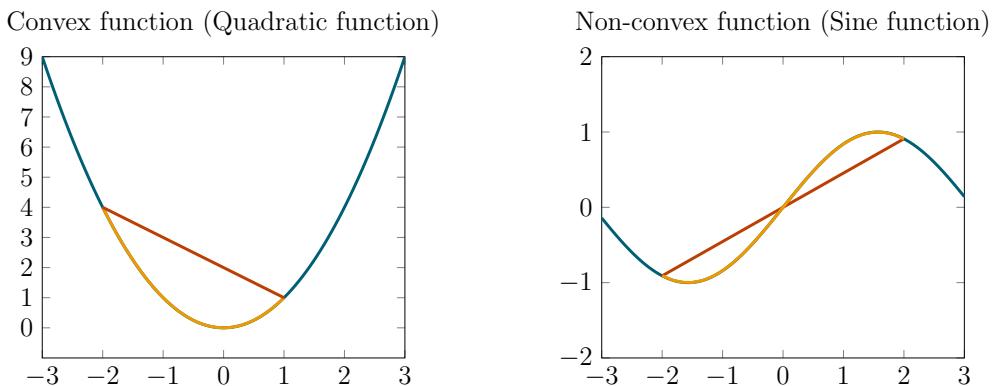


Figure 3.79: Left panel: A convex functions vs. right panel: A non-convex example.

The left panel on plot 3.79 represents a convex function. Any line segment we draw between two points will have all the function values underneath it. On the right panel, we can't state this. For example, if we consider the segment between $x = 0$ and $x = 2$, some $f(x)$ values are above the segment, which means that convexity is

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

not verified. What follows should not come as a surprise. We need Greek letters and equations, so let's start by defining the segment.

We need a strategy, and it happens that I have one in mind. Today I woke up inspired. We will start by defining a segment similar to the one in the left panel. A segment can be determined by two connected points, say x_0 and x_1 , and their corresponding images, $f(x_0)$ and $f(x_1)$. Now we need an expression dependent on x_0 and x_1 to retrieve all the points that belong to the orange line. Hmm so far we don't know how to get all of them, but the middle point we can get:

$$f(x_{middle}) = \frac{f(x_0) + f(x_1)}{2}$$

That is the same as having this equation:

$$f(x_{middle}) = \frac{1}{2}f(x_0) + \frac{1}{2}f(x_1)$$

Then x_1 can be defined as:

$$f(x_1) = 0f(x_0) + 1f(x_1)$$

Similarly, if we wish to represent x_0 :

$$f(x_0) = 1f(x_0) + 0f(x_1)$$

All of those equations are representations of a point via a weighted sum, so if we wish to go closer to x_1 , we increase the factor that multiplies that same value while decreasing the magnitude of the scalar we have just behind x_0 . To move closer to x_0 , we can balance the equation towards it, increasing the weight of x_0 while decreasing the value of x_1 . One way to represent this is with an equation:

$$f(x_{\text{new}}) = \lambda f(x_0) + (1 - \lambda) f(x_1) \quad \text{with } \lambda \in [0, 1] \quad (3.74)$$

If we want to get the edges of the segment, we could do so by having a combination of values for weights of 0 and 1, defining our boundaries. The middle point was defined by a set of values for the same parameters of 0.5 and 0.5. So if we wish to move within the

segment and get new points that belong to the orange line, we must keep the sum of both weights to 1; this way we can guarantee that we will never pass the boundaries defined by $f(x_1)$ and $f(x_0)$.

Equation 3.74 defines a segment between two points that belong to a function; as for the initial definition of convexity that is half of what we need, we now have to guarantee that the values of the function are all below that line. Well, if our function is $f(x)$ and we have a new point given by a combination of x_0 and x_1 , we can represent its image as:

$$f(\lambda x_0 + (1 - \lambda)x_1) \quad \text{with } \lambda \in [0, 1] \quad (3.75)$$

With equation 3.75, we can extract the value of a position on the x -axis of a given point of the segment. We are nearly there; we can find the value of point between x_1 and x_0 represented by equation 3.75. We also have a formula for the point in the segment; this is equation 3.74. If the definition of a convex function is one where all the values that belong to a specific segment are below this same line, we can represent this by:

$$f(\lambda x_0 + (1 - \lambda)x_1) \leq \lambda f(x_0) + (1 - \lambda)f(x_1) \quad \text{with } \lambda \in [0, 1] \quad (3.76)$$

Why do we care about all of this? Let me try and make a good case for convex functions. If we have a convex function, their minima are all global. It's a pretty powerful one, considering the issues with both the optimization methods we just covered, meaning that if they converge, we are guaranteed to find the best solutions.

That definition of equation 3.76 implies that once the function starts growing, it can no longer come down. Otherwise, the concept of convexity will not hold; the right panel of plot 3.79 is an example of that, so not only will we have a minimum, but this minimum will also be global. Now for the analytical proof—you thought I had forgotten that?

Consider a point x^* that's defined as a local minimum on an interval $[x^* - \delta, x^* + \delta]$ for some $\delta > 0$. Within this interval, it holds that $f(x^*) \leq f(x)$. Our primary goal is to show that this relationship $f(x^*) \leq f(x)$, holds true for all x in the domain of f .

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

When we identify a local minimum, it signifies a transition in the function's slope. In the context of a convex function, once its slope becomes positive (indicating growth), it continues in that manner. If this wasn't the scenario, it would lead to behavior akin to the function depicted in the left panel of 3.79. Owing to the convex nature of $f(x)$, we can form a segment connecting $f(x)$ and $f(x^*)$. Utilizing equation 3.76, we can delve deeper into this relationship. To push our analysis further, let's introduce a new data point, y , defined as:

$$y = ax + bx^* \quad \text{where} \quad a + b = 1 \quad (3.77)$$

On 3.77, we have a weighted sum where the values for a and b can be adjusted so that y lies within a δ -neighborhood of x^* , i.e., y is in the interval $[x^* - \delta, x^* + \delta]$. Applying the definition of convexity, we obtain:

$$f(ax + bx^*) \leq af(x) + bf(x^*) \quad (3.78)$$

We can use 3.77 to rewrite 3.78 as:

$$f(y) \leq af(x) + bf(x^*) \quad (3.79)$$

Since y is a convex combination of x and x^* , and due to our choice of a and b , y lies in a region close to x^* where $f(x^*) \leq f(y)$ is a valid assumption. Therefore, we have:

$$f(x^*) \leq f(y) \quad (3.80)$$

Combining 3.79 and 3.80, we derive:

$$f(x^*) \leq f(y) \leq af(x) + bf(x^*) \Rightarrow f(x^*) \leq af(x) + bf(x^*) \quad (3.81)$$

Solving the inequalities for $f(x)$, we deduce:

$$\begin{aligned} f(x^*) &\leq af(x) + bf(x^*) \\ f(x^*) - bf(x^*) &\leq af(x) \\ (1 - b)f(x^*) &\leq af(x) \\ \frac{1 - b}{a}f(x^*) &\leq f(x) \end{aligned} \quad (3.82)$$

Because $a + b = 1$ we have that $b = 1 - a$, so:

$$\begin{aligned} \frac{1-b}{a}f(x^*) &\leq f(x) \\ \frac{1-(1-a)}{a}f(x^*) &\leq f(x) \\ f(x^*) &\leq f(x) \end{aligned}$$

From this derivation, we see that the function value at x^* , which we have identified as a local minimum, is less than or equal to the function value at any point x within the domain of $f(x)$, validating that x^* serves as a global minimum for $f(x)$.

In addition to this method, remember our good old friend, the second derivative, which can provide invaluable insights into a function's curvature? Positive values of the second derivative hint at a \cup shape, implying the function increases as we move along the x -axis. Thus, if our curve is bending upwards, it is convex, allowing us to use the second derivative as a quicker method to confirm convexity.

3.5.4 Really?! Dishing Out More Dough? The Cost Function.

We began our journey with two optimization methods known for their rapid convergence and relative simplicity. However, these methods come with their own set of challenges. In our quest for solutions, we discovered a particular category of functions, known as convex functions, which exhibit a unique and valuable property: any local minimum is also a global minimum. This discovery holds significant importance in machine learning, particularly in the realm of cost functions. These functions are a cornerstone of many algorithms, and if we can design them to be convex within our specific context, we can capitalize on the simple and efficient optimization methods we discussed. More importantly, due to the nature of convex functions, we are assured that once these methods converge, the result is a globally optimal solution.

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

This principle of finding the best solution effortlessly aligns with natural phenomena as well. Let's consider an example from the world of bees to explore this concept further. We humans have taste buds. Our friends the bees have taste receptors located in their antennas and feet. When I learned about this, two scenarios popped into my head. The equivalent of antennas for us would perhaps be our ears? Well, imagine if you could taste with your ears; how wonderful it would be to swim in the ocean? Salty! The other one is around tasting with the feet and imagining the unfortunate scenario where your neighbor did not clean up after its dog did what it had to do, and you happened to step on it.

Back to what is essential, because bees can taste with their feet, when they land on a flower, just by walking on top of it, they can understand whether it has nectar or not. Imagine that we could count the number of steps a bee took and then measure the nectar collected:

Steps Taken x	Nectar Collected (mg) y
1	2
2	4
3	6
4	8

Table 3.2: The nectar collected based on steps taken by bees.

Table 3.2 represents our data collection, it is not a big sample, but it will be enough to understand some cool concepts. Our task is to predict the amount of collected nectar given the number of steps. For that, we will create a function! Because we know quite a few shapes of these mathematical elements, let's plot the data on the table 3.2 and see which type of function is most relevant:

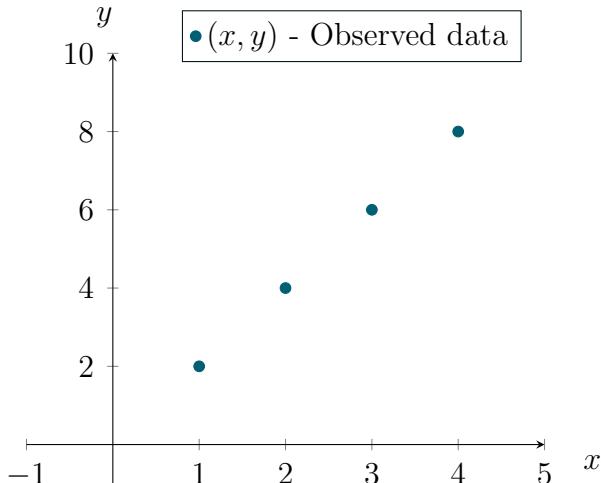


Figure 3.80: The dots representing the steps vs. the nectar collected.

Right, it seems that we can get away with a linear model, so let's try:

$$y = mx \quad (3.83)$$

OK, time to stop and be real with each other. The values of y in table 3.2 are double those of x , so $y = 2x$ is the perfect fit for that data. It is an elementary example and a slight modification of a machine learning algorithm called linear regression. With linear regression, you will be using the equation $y = mx + b$ as a model hypothesis, whereas ours is just $y = mx$. We're excluding the intercept term b to simplify our task to optimizing just m . Had we included b , making our equation $y = mx + b$, we would need to optimize two variables, m and b . This is certainly possible, but it requires partial derivatives. We will study this fundamental concept later. But, with this example, we can achieve several things: set up a cost function that is convex, optimize m with the gradient descent, and set up some groundwork, so later, when we get into higher dimensionality concepts, we can complete this example and finally, fully cover linear regression. One way to start this party is to randomly assign a value to m and check what comes from it. For example if $m = 0.5$ our model becomes:

$$y = 0.5x \quad (3.84)$$

The logical next step will be to verify how well our equation suits

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

the task. For that, we will input all the values of x in equation 3.84 and check what y value will result from it:

Steps Taken x	Nectar Collected (mg) y	\hat{y}
1	2	0.5
2	4	1
3	6	1.5
4	8	2

Table 3.3: The same table as 3.2 but we have some guesses.

We have the first set of predictions represented by \hat{y} . I think it is time to celebrate, and what better way than with a graph?

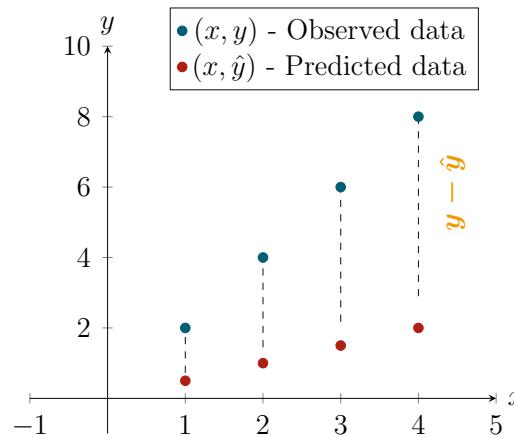


Figure 3.81: How far were we from the observed nectar values with our first guesses?

Consider the red dots on the graph 3.81, which denote the predicted values from our model $y = 0.5x$. The dashed lines depict the differences between the observed data points and these predicted values. The quality of our model depends on these distances. Specifically, shorter distances between predicted and observed data points signify a better fit of our model to the data. Thus, these discrepancies can be used as a metric to gauge the accuracy of our chosen slope m and, by extension, the overall fit of our equation:

x	y	\hat{y}	$y - \hat{y}$
1	2	0.5	1.5
2	4	1	3
3	6	1.5	4.5
4	8	2	6

Table 3.4: We now include the length of those dashed lines with $y - \hat{y}$.

In our updated table, x represents the number of steps, and y the collected nectar. We use the notation \hat{y} for our predictions. The term $y - \hat{y}$ indicates the distance between our prediction and the actual observed value of y . This distance corresponds to the length of each dashed line in our graph. We have a local understanding of how well our model is performing and how far each prediction is from the real value; this is good, but we would do better with an overall metric. We can add all those errors up, the metric that defines how far away y is from \hat{y} . Let's call this new metric "cost"; it's about time we had something palpable, so:

$$c = 1.5 + 3 + 4.5 + 6 = 15$$

If we consider c our total cost and calculate it by simply adding the entries of $y - \hat{y}$, we will soon encounter a problem. What if the red dot is above the blue one? In that specific case, because $y - \hat{y}$ will turn out negative, our total cost will be reduced, with a bad prediction. In fact, the worse the prediction, the more significant the cost reduction, which will constitute a huge disaster.

One solution for this could be to use the module of $y - \hat{y}$, like $|y - \hat{y}|$. However, this approach has its shortcomings. The absolute value function isn't differentiable everywhere in its domain, which complicates matters if we're discussing gradient-based optimization methods. Therefore, the most suitable alternative is to square the difference, yielding $(y - \hat{y})^2$. This method ensures that all positive or negative differences contribute positively and are amenable to differentiation, making it compatible with techniques like gradient descent:

x	y	\hat{y}	$(y - \hat{y})^2$
1	2	0.5	2.25
2	4	1	9
3	6	1.5	20.25
4	8	2	36

Table 3.5: We might need to square the $y - \hat{y}$, we can't deal with negative distances.

If we compute the sum of $(y - \hat{y})^2$ we have:

$$c = 2.25 + 9 + 20.25 + 36 = 67.5$$

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

This calculation gives us the total cost. Let's create an equation for the metrics that we are currently exploring:

$$c = \sum_{i=1}^4 (y_i - \hat{y}_i)^2 \quad (3.85)$$

Equation 3.85 represents the total cost for this specific problem. It is the summation of $(y - \hat{y})^2$ for the four entries we have in table 3.5. Totals can be of great help, but in this case, they won't be the best measurement of cost because we are lacking an understanding of how well we are doing individually. Also, the comparison with other costs will be challenging. To overcome this, we could divide the total result by 4, which is the number of observations, so:

$$c = \frac{1}{4} \sum_{i=1}^4 (y_i - \hat{y}_i)^2 = 16.875 \quad (3.86)$$

And if we wish to generalize to N observations:

$$c = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (3.87)$$

Cool, we have a function! So we need to minimize the cost to have the best possible model that accommodates our data meaning that we must have a way to select a value of m such that the cost is minimum. The problem is that equation 3.87 has no m at all. Why are you worried? There is always a way. The \hat{y} values are the result of inputting x to equation 3.84 so we say that:

$$\hat{y} = 0.5x$$

With this we can turn equation 3.87 into:

$$\text{MSE} = \frac{1}{4} \sum_{i=1}^4 (y_i - (0.5x_i))^2 \quad (3.88)$$

And for the general case :

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 \quad (3.89)$$

Where $f(x)$ represents the selected function for a given task, there was a sneaky change in the name of the cost function from c to MSE. That is the name given to this particular cost function; it stands for Maybe Suspensory Extol, it doesn't, it doesn't... Mean Squared Error is the name, and it is essential to note that this is not the only option for a cost function; we will explore some more through this book.

Now it comes the moment of truth. Is that thing convex? Well, let's calculate the second derivative and check what's up. To get to the second, we must start with the first one; and what needs to be clarified is the variable we will be respecting. Since we are trying to understand how the changes in m affect the cost function, it would make sense to derive the cost function with respect to m :

$$\frac{d}{dm} \frac{1}{4} \sum_{i=1}^4 (y_i - mx_i)^2 \quad (3.90)$$

To compute the derivative in 3.90 we will consider x and y as constants and use the chain rule as follows:

$$f(u) = u^2 \quad g(m) = y_i - mx_i$$

We need the derivative of $f(g(m))$; that is equal to $f'(g(m)).g'(m)$ so:

$$f'(g(m)).g'(m) = 2(y_i - mx_i)(-x_i)$$

It turns out that:

$$\frac{dMSE}{dm} = \frac{1}{4} \sum_{i=1}^4 (-2x_i)(y_i - mx_i)$$

That in turn is equal to:

$$\frac{dMSE}{dm} = \frac{1}{4} \sum_{i=1}^4 -2x_i y_i + 2mx_i^2 \quad (3.91)$$

If that summation symbol in equation 3.90 is confusing about how to compute the derivative of that same expression, we can think of it as a polynomial; for example, if we expand it:

$$\frac{(y_1 - mx_1)^2}{4} + \frac{(y_2 - mx_2)^2}{4} + \frac{(y_3 - mx_3)^2}{4} + \frac{(y_4 - mx_4)^2}{4}$$

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

Because the y 's and the x 's are constants, we can apply both the derivatives' power and sum rule and end up with equation 3.91. Now if we compute the derivative of 3.91 we will be equipped with everything necessary to check if our cost function is convex:

$$\frac{d^2}{dm^2} \frac{1}{4} \sum_{i=1}^4 (y_i - mx_i)^2$$

If we derive equation 3.91 we get:

$$\frac{d^2}{dm^2} \text{MSE} = \frac{1}{4} \sum_{i=1}^4 2x_i^2 \quad (3.92)$$

Success, equation 3.92 is always positive and therefore convex! We have a convex cost function that is dependent on m . So, if we apply, for example, the gradient descent and the method converges, we are guaranteed that the method had found a global minimum; on top of this, we already have the equation for the first derivative, equation 3.91, so we are ready for it baby! Not so fast, I might have gotten overexcited; we do still need a learning rate also, so let's say $\eta = 0.01$. Now we are ready. So for the next m we have:

$$m_{n+1} = m_n - \eta \frac{d}{dm} \text{MSE} \quad (3.93)$$

Here n is the iteration number; so for the first iteration we have that:

$$m_1 = m_0 - \eta \frac{d}{dm} \text{MSE}$$

Now we can input m_0 with the first selection for m , 0.5, and η , with the learning rate 0.01:

$$m_1 = 0.5 - 0.01 \left(\frac{1}{4} \sum_{i=1}^4 -2x_i y_i + 2m_{old} x_i^2 \right)$$

We have that bigass term with a summation to the right of the equation, so let's take care of it first.

For that, we will input the values of x and y from table 3.5 into that expression with the summation and calculate what comes from that term:

$$\begin{aligned} \sum_{i=1}^4 -2x_i y_i + 2m_{old} x_i^2 &= -2x_1 y_1 + 2m_{old} x_1^2 - 2x_2 y_2 + 2m_{old} x_2^2 \\ &\quad - 2x_3 y_3 + 2m_{old} x_3^2 - 2x_4 y_4 + 2m_{old} x_4^2 \end{aligned}$$

Replacing the variables with numbers:

$$\begin{aligned} \sum_{i=1}^4 -2x_i y_i + 2m_{old} x_i^2 &= -2(1)2 + 2(0.5)(1)^2 - 2(2)4 + 2(0.5)2^2 \\ &\quad - 2(3)6 + 2(0.5)3^2 - 2(4)8 + 2(0.5)4^2 = -90 \end{aligned}$$

So, for m_1 we have that:

$$m_1 = 0.5 - 0.01 \left(\frac{-90}{4} \right) = 0.725$$

After one iteration, our model is the following:

$$y = 0.725x$$

Verification time!

x	y	\hat{y}
1	2	0.725
2	4	1.45
3	6	2.175
4	8	2.9

Table 3.6: A new set of \hat{y} , hopefully with some improvements.

A graph is always a good idea:

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

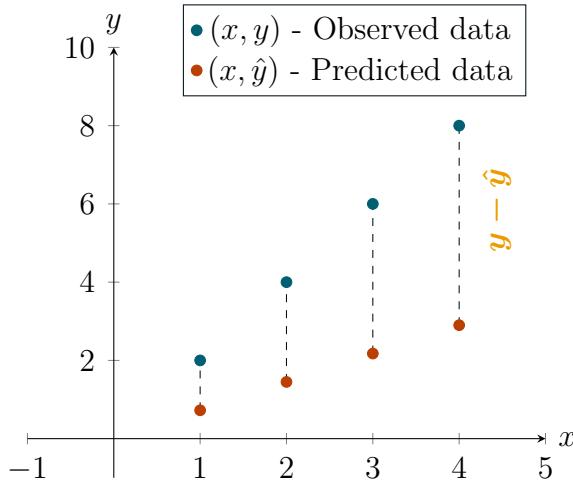


Figure 3.82: Another set of distances between points, but this time, they are shorter.

Visually those dashed lines in figure 3.82 seem shorter. Therefore, our MSE got smaller, so we made some improvements after just one iteration of gradient descent. To be certain of this, let's analytically compute the MSE for the first iteration of the gradient descent. We could do so with the values from table 3.6:

$$\text{MSE}_1 = \frac{1}{4} \sum_{i=1}^4 (y_i - (0.725x_i))^2 = 12.19$$

We introduced some new notation with MSE_1 . This directed us to the average error resultant when we calculated the \hat{y} with the value of m that we got after one passage of the gradient descent, meaning that MSE_n refers to the cost of iteration n . Now we must compare MSE_1 with MSE_0 and because $\text{MSE}_1 = 12.19$ and $\text{MSE}_0 = 16.975$:

$$\text{MSE}_0 > \text{MSE}_1$$

That improvement is substantial enough to convince me that we should keep iterating on that specimen. I'll spare you the heavy notation and repetitive calculations since they're basically the same as what we did for m_1 .

Using the 3 steps outlined below, I carried out 10 more iterations:

1. Compute the new m_i with equation 3.93 where i is the iteration number.

2. Calculate the cost function. It can help understand if the algorithm is converging.
3. Check if the number of iterations is lower than 10; repeat 1 and 2 if yes. Otherwise, stop.

For this scenario, we chose the number of iterations as the stopping criteria; you can perform step 3 with any stopping condition you wish. Here is the promised table:

Iteration	m_i	Gradient	Cost
1	0.725	-22.5	12.192
2	0.916	-19.125	8.809
3	1.079	-16.256	6.364
4	1.217	-13.818	4.598
5	1.334	-11.745	3.322
6	1.434	-9.983	2.4
7	1.519	-8.486	1.734
8	1.591	-7.213	1.253
9	1.653	-6.131	0.905
10	1.705	-5.211	0.654

Table 3.7: Gradient Descent Iterations

The right-hand side column, which has the cost function values, gets smaller and smaller as we move on through iterations, a behavior that implies that the observed values of y and the new predictions are getting closer and closer.

You know what would be beneficial? To construct both a table and a graph. These should display the computed values of \hat{y} when applying the equation $y = 1.705x$. Specifically, we should iterate this equation 10 times:

x	y	\hat{y}
1	2	1.705
2	4	3.41
3	6	5.115
4	8	6.82

Table 3.8: Another little table, but at least we have better \hat{y} .

To have a clearer picture of the improvement, we included on the graph 3.83 the original points, represented by the blue color, the

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

first set of \hat{y} 's we got with $m_0 = 0.5$ are the red guys, and finally the yellow color is highlights the model values we get after 10 iterations, meaning that $m_{10} = 0.75$:

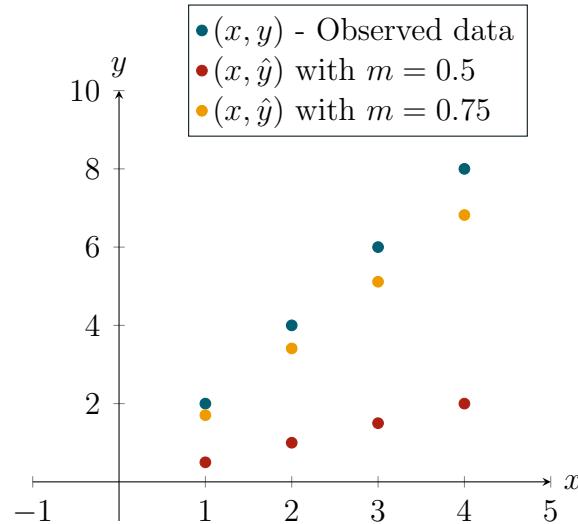


Figure 3.83: Yellow for the win!

The yellow dots are much closer to the blue ones than the red fellows. Because our cost is related to the proximity between the collection of the same color points and the blue ones, our new model is a much better fit for the data we were working with, and we did this by optimizing the cost function with the gradient descent. Finally, let's plot the cost over the values of m we get from each iteration and see what behavior we get:

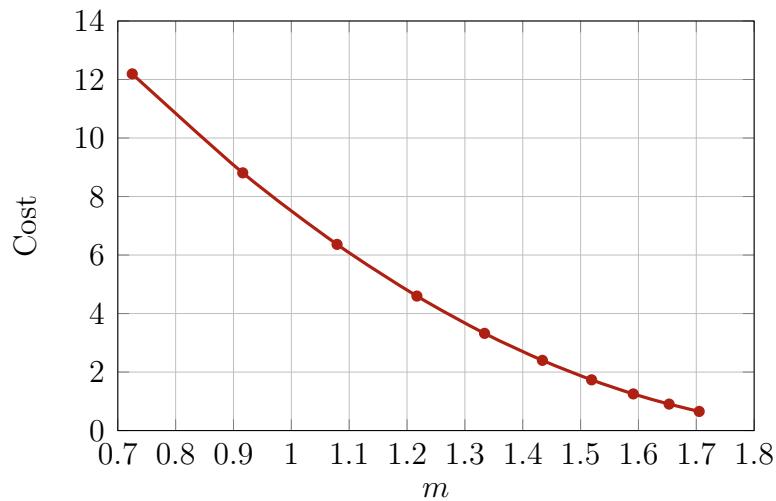


Figure 3.84: The cost behaviour by the value of m .

Look at that! On the y -axis of 3.84, we have the value of the MSE, whereas the x -axis represents the values of m_n where n is the iteration number. The behavior is what we expected or at least hoped for, bigger jumps at the first iterations followed by smaller and smaller steps as we reach values that are getting closer to the minimum. As we proved that the MSE is a convex function because the second derivative is always positive, that minimum is the global minimum.

Cool, we delved into applying gradient descent to optimize a cost function dependent on a single parameter to fit some data. But, in terms of a machine learning algorithms, we only did half the job, as the processes we discussed predominantly center on the training phase of a machine learning algorithm, in which calculus is prominent. It's important to note, however, that the journey of machine learning doesn't end with training. The true efficacy of our model is tested when it encounters new, unseen data. This step is crucial for assessing how well the model generalizes beyond the data it was trained on.

We must venture into the testing and performance metrics to fully understand how well our model is doing. Think of our model as a student who has been studying a subject. While they may ace the exercises and problems from their textbook (data they've "seen"), the real test lies in their performance on a brand new set of questions or an unfamiliar exam (data they haven't seen before). Machine learning involves testing our model on entirely new data, which is untouched during the training phase. This new data is essential because it evaluates how well our model generalizes its learning to predict information outside its training experience. We can only confidently claim the model's effectiveness or robustness with this step.

We typically split our initial dataset into a training set and a test set or gather new data to do this. Using this previously unseen data, we predict values and calculate the MSE for these new predictions, \hat{y} . It's worth noting that our emphasis on the MSE was mainly to illustrate the convergent behavior of gradient descent, the optimiza-

3.5. Valleys and Hills: A Critical Situation for Points and Their Roots.

tion technique we were evaluating, and not on fully evaluating the model.

While testing and performance metrics are critical in machine learning, this book focuses on calculus and its various applications. In machine learning, performance metrics often resemble simple values, like percentages or ratios, which summarize how well a model performs. But don't be deceived by their simplicity. While these metrics offer a snapshot, genuinely understanding what they signify requires a deep dive into the algorithm's mechanics. Mastering the foundational concepts of the training stage is crucial. After all, if we encounter unfavorable metrics, how can we pinpoint and rectify the underlying issues if we're unaware of what's happening under the hood of the algorithm? We started with a definition of the rate of change which led us to explore the abstract notion of an instantaneous rate of change, which indicates how much a function changes at a point! The definition of such a technique created the necessity to learn the concept of proximity, which we defined in terms of limits. Then it finally came to that lazy eight, that son of a bitch that never stands up, but he helped us to zoom in on a graph an infinite number of times and conclude that the instantaneous rate of change on a point gives us the slope of the tangent line on that same point. We also explored different types of functions, namely some trigonometric ones, which made us miss the good old polynomials where calculations are made simple for us. But as adversity breeds genius, we explored a way of local approximation by introducing series, culminating in the Taylor series. This specific series leverages derivatives to their best; as we keep increasing the value of n , we get extra information on the function via the features captured.

Derivatives break everything into tiny pieces and allow us to have a loads of information about a function, so much so that we can build optimization algorithms. But... if they excel at breaking things down, does this mean something else has to put them back together? Yes indeed. Please welcome the integrals!

Chapter 4

Cleaning Up The Derivatives Debris: The Integral.

We are getting deeper into our odyssey on the curves of calculus and are about to cross worlds, from the realm of derivatives, where everything got broken into tiny pieces, into the realm of the integrals. As the name suggests, integrals are about integration, a concept allowing a macroscopic perspective, putting together the microscopic wreckage that the derivatives left behind. Since we introduced the derivatives with bee's flight's speed, let's stick with the same concept for consistency. Bees possess remarkable navigation skills. They use a combination of visual cues, such as landmarks, the sun's position, and odor cues, to navigate from the hive to a specific flower. Plus, they recognize particular flower species and remember their characteristics, such as color, shape, and scent. Those factors contribute to locating and targeting the flowers that provide the most rewarding nectar and pollen resources:

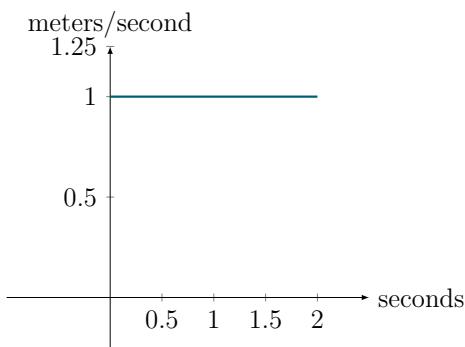


Figure 4.1: Look how fast that bee is flying.

Remember when I mentioned that training a model is half the job? That was not very nice of me so let me redeem myself with a little treat. The bees' flight velocity measurements have arrived, and they are represented by graph 4.1. So we can study them! Isn't that exciting? Tough crowd... Anyhow, we have them, they came in meters per second, and they happen to be constant.

Before, we explored tasks that involved calculating rates of changes, such as velocity; now, the paradigm has flipped on us; we will be after cumulative metrics such as the total distance covered by a bee in a given time interval. Alright, if I have a measurement of time and another of time per distance, one way to get distance is to multiply them:

$$\text{meters} = \text{seconds} \cdot \frac{\text{meters}}{\text{seconds}}$$

Yeah, I know. So much talk about mathematics to end up calculating the area of a rectangle. Stick with me, as we will get somewhere more interesting soon:

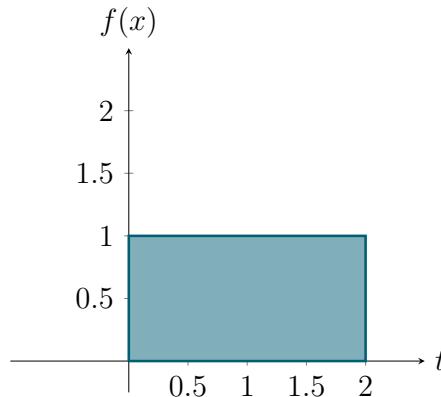


Figure 4.2: The cumulative distance covered by the bee.

So the total distance traveled by the bee as represented by the blue rectangle can be computed by:

$$\text{total distance} = 2 \text{seconds} \cdot 1 \frac{\text{meters}}{\text{seconds}} = 2 \text{meters}$$

That was an easy task to solve as we were not looking at curves, but this is about to change as a second measurement of the velocity of a bee's flight just came in, and this time, it is not constant. We have a velocity that varies over time:

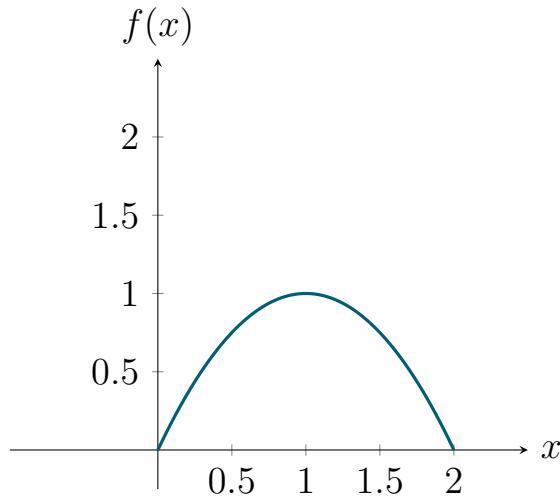


Figure 4.3: The variation of the bee's' velocity over time.

The function that represents the graph 4.3 is:

$$f(x) = x(2 - x) \quad (4.1)$$

A first glance of the plot 4.3 suggests that calculating that shape's area will be complicated, but we can always try to approximate it. Because rectangles are shapes for which we can quickly get a value for their area, one way to have an approximation of that area would be to fit a certain number of rectangles under that curve; let's try with 4:

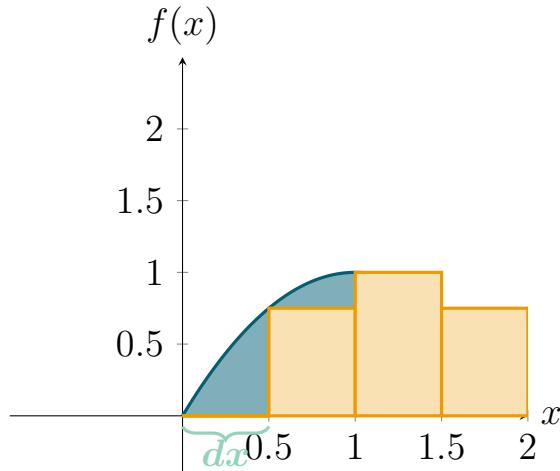


Figure 4.4: Could we get a good measurement by using rectangles?

The yellow rectangles on 4.4, are the ones we will use to estimate the area under the curve. They all have the same basis size, defined

by dx ; so all the rectangles have the same width, meaning that the following formula can be used to calculate dx :

$$dx = \frac{2 - 0}{4}$$

The 2 and the 0 show up because we are after the total distance covered by a bee between 0 and 2. The number 4 comes after the total number of rectangles we wish to introduce, which means that $dx = 0.5$. With the basis length calculated, we are halfway on the quest for our first area approximation. What is missing is the heights of those shapes, and we just have 3 different options for such magnitudes; The image of the first point that is part of the base, the image of the middle point, or, lastly, the image of the last point of the base. We will go with the first option, the image of the first point. Remember that when you choose one criteria, you must stick with it for all rectangles:

$$\text{Area}_1 = 0.5f(0) = 0$$

For the second one, we have:

$$\text{Area}_2 = 0.5f(0.5) = 0.375$$

The third one is a charm. At least, this is what I was always told:

$$\text{Area}_3 = 0.5f(1.0) = 0.5$$

Lastly:

$$\text{Area}_4 = 0.5f(1.5) = 0.375$$

So for the total area, we have:

$$\text{Area}_{\text{total}} = 0 + 0.375 + 0.5 + 0.375 = 1.25$$

Visually we can see that this value for the area of the curve could be better, as there is quite a bit of blue area that the rectangles can't fill and a part of them are over the curve. One thing we can do is to create more rectangles. 12? Why not?

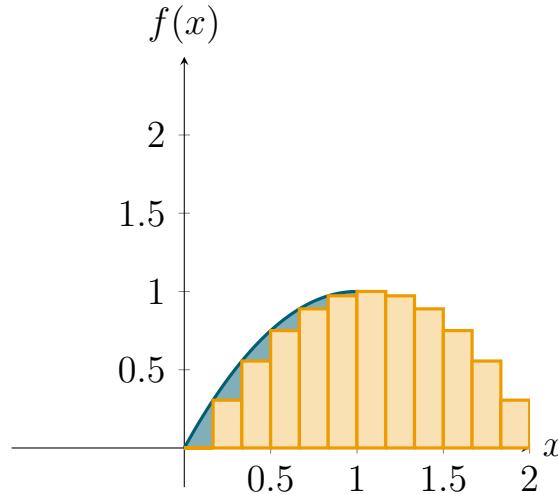


Figure 4.5: Dam, the more the merrier? It seems so, as the approximation is better.

By looking at figure 4.5, we all know where this is going. The route is to keep creating more and more smaller rectangles, so let's use this case of 12 rectangles to build a generic formula that represents the sum of their total areas. We need dx as it represents the width of those guys and in this specific case:

$$dx = \frac{2 - 0}{12}$$

OK, so we need a formula to represent a summation of the multiplication of an image of a selected point by dx or the base of the rectangles' areas:

$$R_{12} = \sum_{i=1}^{12} f(x_i)dx$$

Yes, we are ready for generalization:

$$R_n = \sum_{i=1}^n f(x_i)dx \tag{4.2}$$

Where:

$$dx = \frac{b - a}{n}$$

4.1 That Riemann Guy Is a Bit Square, Isn't He?

The values b and a represent the interval from which we wish to derive an approximation; in our example, we have $a = 0$ and $b = 2$. The R_n , which is named after Riemann, the mathematician who came up with it, defines the area we are after. Alright, so for 12 rectangles, the area is estimated as:

$$R_{12} = \sum_{i=1}^{12} f(x_i)dx = 1.32$$

We don't have a point of reference, i.e., the real value for the area, but visually we can see this is a better approximation; there is less blue uncovered on the left side and, where the rectangles are over the curve also, the right side, it seems to happen less than before where we had 4 rectangles. Logically 36 rectangles will create a better approximation:

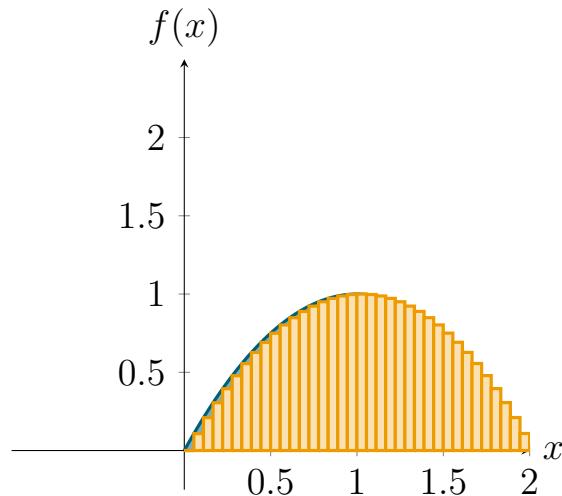


Figure 4.6: 36 hun? Ok, Ok, we are convinced that shorter rectangles provide better approximations.

Figure 4.6 looks like a better approximation, and we can find that $R_{36} = 1.33$ to confirm it. Because we keep getting better approximations of the areas as we keep creating more and thinner rectangles, the actual value of the area will be returned if we generate an infinite

number of them:

$$R = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(x_i) dx \quad (4.3)$$

Equation 4.3 can also have the following notation:

$$\int_a^b f(x) dx \quad (4.4)$$

Would you look at that? We have a new symbol, the \int , and it happens to represent the concept we are currently studying, the integrals. The integral symbol, \int , represents the continuous accumulation of quantities, much like summing an infinite number of infinitesimally small rectangles under a curve. It's a powerful tool that allows us to find the total area under a curve, even when the curve's shape is complex or changing. We can read equation 4.4 as the integral of $f(x)$ with respect to x between a and b , and the conditions for this to be calculable are: $a < b$ and $f(x)$ is continuous in the interval $[a, b]$. The reason behind replacing the \sum is that we are interested in understanding what happens to R_n when $dx \rightarrow 0$, which is the total area under the curve.

This transition from a finite number of rectangles to an infinite sum, represented by the integral, embodies one of the most profound ideas in calculus: capturing the whole by summing its infinitely small parts:

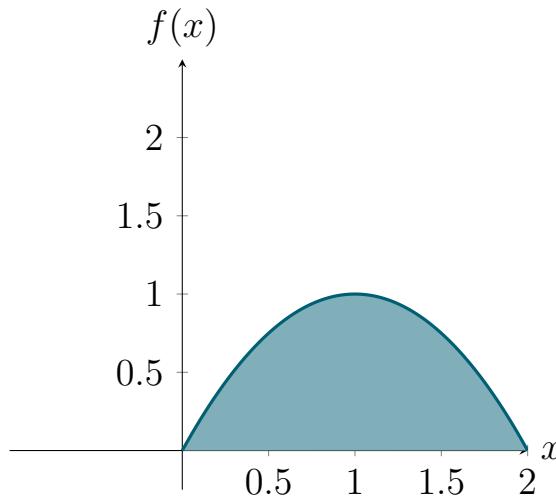


Figure 4.7: The area we are really after.

4.1. That **Riemann** Guy Is a Bit Square, Isn't He?

We began with a straightforward scenario: a constant velocity over a fixed time interval. This setup allowed us simply to calculate the total distance as the area of a rectangle. However, things got more interesting when we introduced a non-constant velocity measurement leading to a curved path. To tackle this, we turned to approximations using four rectangles, each of height $f(x)$ and width dx . By making dx infinitesimally small, we found that our approximation improved due to the increased number of rectangles. This behavior of getting better approximations by shrinking dx led us to a limit, illustrated in 4.3. We are seeking to capture the total area under the curve by summing areas of these numerous tiny rectangles, so we introduced a new notation encapsulated by equation 4.4. Essentially, as our approximation becomes near-perfect with dx tending to 0, the most accurate representation of the area under the curve, which in this case is the total distance, is provided by 4.4.

Now, let's adapt 4.4 to our specific context and learn how to compute it. Starting with the limits of the integral, a and b : the lower limit a is straightforward, set at 0. For b , considering our goal to compute the total distance covered between 0 and 2, we initially set $b = 2$. However, to generalize this for any interval within $]0, 2]$, we introduce a variable T in place of b , representing any number within this range. Consequently, we define a distance function:

$$s(T) = \int_0^T f(x) dx \quad (4.5)$$

When we first explored the concept of derivatives, we computed the rate of change in distance over time to find velocity. Now, given a velocity function $f(x)$ and our newly defined distance function $s(T)$, we see the integral symbol \int , signifying the accumulation of velocity over time. To elucidate the relationship between $f(x)$ and $s(T)$, we differentiate $s(T)$ with respect to T :

$$\frac{ds}{dT}(T) = f(T) \quad (4.6)$$

This equation 4.6 confirms that the rate of change of distance with respect to time is the velocity, which is represented by $f(x)$ evaluated at $x = T$. What this means is that to calculate expressions

like the 4.4, we have to think about which function has $f(x)$ for a derivative:

$$\frac{d}{dx}(?) = 2x - x^2 \quad (4.7)$$

Just like with derivatives, when performing integration, we apply the following rules:

1. $\int_a^b cf(x) dx = c \int_a^b f(x) dx.$
2. $\int_a^b (f(x) + g(x)) dx = \int_a^b f(x) dx + \int_a^b g(x) dx.$
3. $\int_a^b (f(x) - g(x)) dx = \int_a^b f(x) dx - \int_a^b g(x) dx.$

These rules help us calculate the area under the curve of a function, which in the context of our discussion, allows us to find the total distance traveled as represented by the integral.

The definition of the integral involves dividing the interval $[a, b]$ into small sub-intervals, computing the sum of the areas of the rectangles defined by the function values over these sub-intervals, and then taking the limit as the width of these sub-intervals approaches zero. When you add or subtract the function values $f(x)$ and $g(x)$ over each sub-interval, it doesn't matter whether you add or subtract the function values first and then take the integral or take the integrals and add or subtract them. This is because addition is associative, meaning that how numbers are grouped doesn't change their sum or subtraction. If we multiply an integral by a constant, we can use a similar explanation to understand why rule 1 is valid. OK, so we will make use of those rules to compute:

$$\int_0^2 (2x - x^2) dx$$

If we apply point 3 we have:

$$\int_0^2 (2x - x^2) dx = \int_0^2 2x dx - \int_0^2 x^2 dx$$

4.1. That **Riemann** Guy Is a Bit Square, Isn't He?

Let's start with the guy on the right:

$$\int_0^2 x^2 dx$$

The question is, what function has x^2 for its derivative? Well, we know that the power rule states that:

$$x^n = nx^{n-1}$$

Before we use the expression on the left to come up with a solution, we need to consider what equation on the right gives us the left part. Now, Instead of going from x^n to nx^{n-1} , we must find which x^n created nx^{n-1} . As we have x^2 , then $n - 1 = 2$, so $n = 3$ therefore x^3 might be what we are after. Let's check:

$$(x^3)' = 3x^2$$

We are nearly there. The only problem we have is the 3 that is multiplying with x^2 ; we are on the hunt for just a x^2 . We can divide x^2 by 3 and get rid of that constant that will come after applying the power rule:

$$\left(\frac{x^3}{3}\right)' = x^2 \quad (4.8)$$

The integral of x^2 is represented as $\frac{x^3}{3}$. To confirm this, if we differentiate $\frac{x^3}{3}$ with respect to x (as shown in 4.8), the result is x^2 , our starting function. This highlights a significant property: $\frac{x^3}{3}$ is also termed an "anti-derivative" of x^2 . Why? Because differentiating it returns us to the initial function. Now, let's consider the upper and lower limits of the integral. When integrating a function over an interval, such as from 0 to 2, we're essentially calculating the cumulative area beneath the curve between these boundaries. To find this total area using our established anti-derivative, evaluate $\frac{x^3}{3}$ at the upper limit 2 and subtract its value when assessed at the lower limit 0.

$$\int x^2 dx = \frac{x^3}{3}$$

Therefore:

$$\int_0^2 x^2 dx = \frac{2^3}{3} - \frac{0^3}{3} = \frac{8}{3}$$

For the missing bit:

$$\int_0^2 2x dx$$

We have an antiderivative of:

$$x^2$$

So:

$$\int_0^2 2x dx = 2^2 - 0^2 = 4$$

Finally:

$$\int_0^2 (2x - x^2) dx = 4 - \frac{8}{3} = \frac{4}{3}$$

The bee covered around 1.33 meters in total! To sum up, when we wish to calculate an area under a curve, we can do so with an integral:

$$\int_a^b f(x) dx = F(b) - F(a) \quad (4.9)$$

Here, F is an anti-derivative with the following formula:

$$\frac{dF}{dx}(x) = f(x) \quad (4.10)$$

4.2 This seems important: The Fundamental Theorem of Calculus.

We are adding the areas of rectangles that are calculated by multiplying $f(x)$ by dx and then asking what happens to the sum of these areas when $dx \rightarrow 0$. A fundamental fact that we need to reference is that equations 4.9 and 4.10 form the **fundamental theorem of calculus**.

And suddenly, we got lucky again, as some new measurements came in, and now, instead of data for 2 seconds, we have data for 3. Let's see what the bee is doing then:

4.2. This seems important: **The Fundamental Theorem of Calculus.**

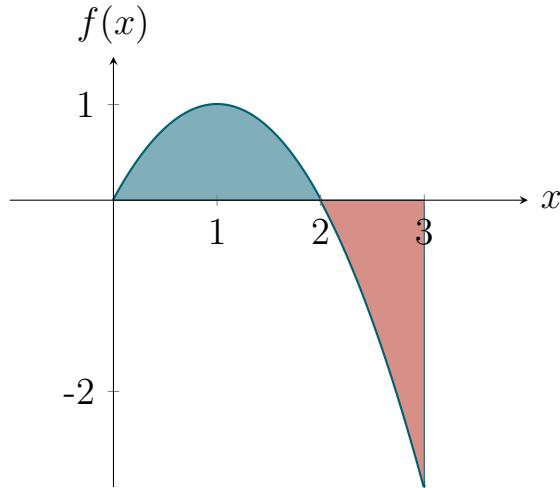


Figure 4.8: Wait a second. Negative values for velocity? Come on, dude.

The graph 4.8 represents the new measurement of the velocity function of the bee, and we see a new scenario that we haven't covered previously: negative values for velocity. This magnitude is a vector, and negative signs represent a change in direction, but how will this impact the integrals? Well, there is only one way to find out, and it involves some calculations:

$$\int_0^3 (2x - x^2) dx \quad (4.11)$$

We can apply the equation 4.9 to calculate the value of 4.11:

$$\int_0^3 (2x - x^2) dx = F(3) - F(0)$$

Where:

$$F(x) = x^2 - \frac{x^3}{3}$$

$$\int_0^3 (2x - x^2) dx = F(3) - F(0)$$

Substituting the values, we get:

$$F(3) = 3^2 - \frac{3^3}{3} = 9 - 9 = 0$$

$$F(0) = 0^2 - \frac{0^3}{3} = 0$$

Therefore:

$$\int_0^3 (2x - x^2) dx = 0 - 0 = 0$$

What does an integral of 0 mean? In this particular situation, the bee ended up in the exact same location after 3 seconds of flight. It traveled a distance from second 0 to 2, but then, it changed the direction and traveled the same distance back, ending up at the original location. The goal is not for us to be integral calculation machines but to understand the concept behind them. Before moving into the world of high dimensionality, I would like to present another use case for integrals.

In a beehive, the queen rules. She really is the boss. The queen bee releases pheromones that allow the other bees to know what's up; she is in charge, and her dominance is to be respected. Not only that, but these same pheromones also have an inhibitory effect on the development of new queens within the colony. It suppresses worker bees' ovaries, preventing them from becoming fertile and laying eggs. There's no messing around inside the hive. The plot below represent how rapidly the pheromone concentration dissipates. It is similar to the velocity we studied above, but on the y -axis, we have "ppt (part per trillion)". The x -axis still measures time, but now it is represented in hours:

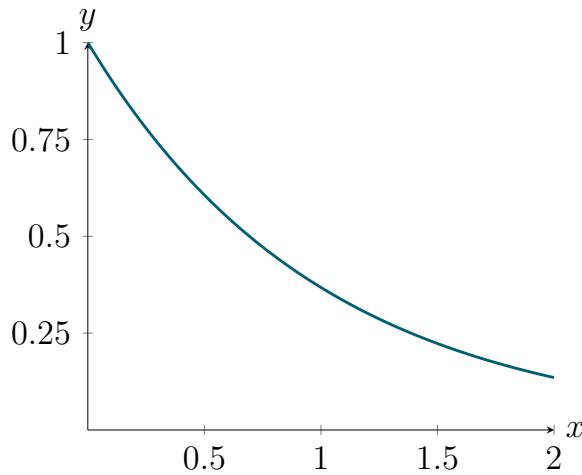


Figure 4.9: The representation of how fast the concentration of this pheromone dissipates.

The task is to calculate the cumulative concentration of pher-

4.2. This seems important: **The Fundamental Theorem of Calculus.**

omones in the first 2 seconds. For that, we first need to define a function that can fit the data, something with a shape similar to the curve in the plot 4.9. Looking back at what we have studied so far, we have seen some shapes like that curve when we looked at $\frac{1}{x}$. However, this curve appears to decrease more rapidly, prompting a visual comparison:

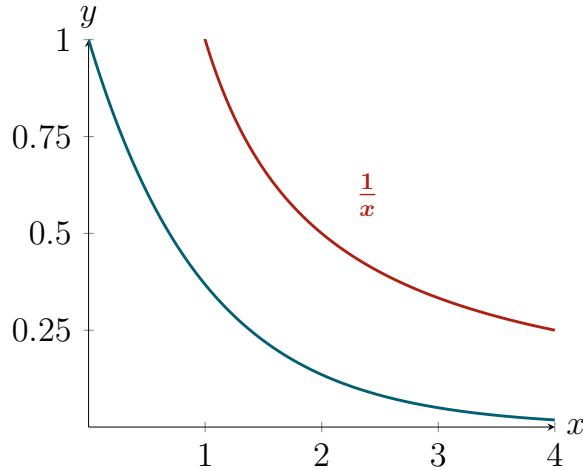


Figure 4.10: On the race to 0 the blue won wins!

In our previous explorations, we examined functions like x^{-n} for positive values of x , observing that increasing n enhances the function's decline. Though it decays, it is not steep enough for this particular case, because, fundamentally, it is a polynomial. So we need something that decreases faster, and honestly, I just see one option: what about a little switcheroo? We could put the variable as an exponent, something like:

$$f(x) = b^x \quad (4.12)$$

In essence, for every unit increase in x , the function's value gets the value of the function multiplied by b , and this constant multiplication leads to a much faster growth in value than we can get with any polynomials. These guys, the ones with the shape of equation 4.12, are functions called exponentials, and they are instrumental in modeling any event that shows rapid growth, such as population growth, the spread of viruses, and natural resource depletion. Cool, so let's dive a bit deeper into the behavior of these individuals as we

have two components to explore; we wish to understand what happens to exponential functions when we vary b and also how fiddling around with x will impact these functions:

1. $0 < b < 1$:

- **Positive Exponent $x > 0$:** As x increases, the function $f(x)$ decreases. The rate of decrease becomes slower as x increases, approaching zero but never reaching it.

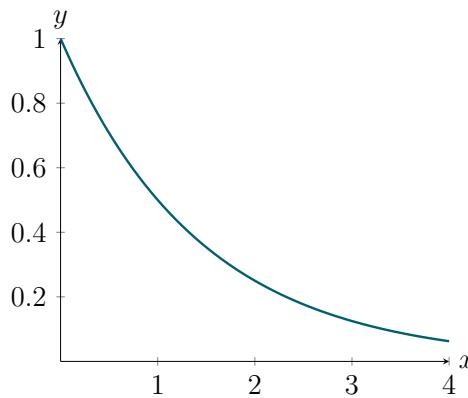


Figure 4.11: A positive exponent and a base $0 < b < 1$.

- **Negative Exponent $x < 0$:** As x becomes more negative, the function $f(x)$ increases. The rate of increase becomes slower as x approaches "negative infinity" but never reaches infinity:

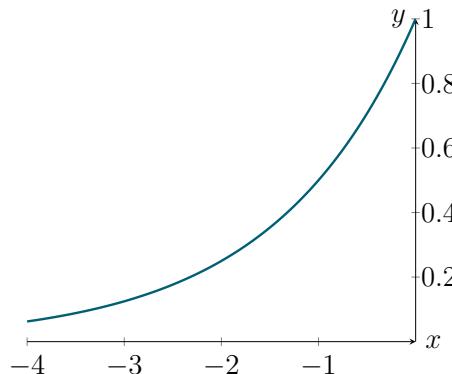


Figure 4.12: The same base $0 < b < 1$ but a negative exponent.

4.2. This seems important: **The Fundamental Theorem of Calculus.**

- **Fractional Positive Exponent $0 < x < 1$:** The function's behavior depends on the specific value of x . For example, if $x = \frac{1}{2}$, the function $f(x)$ represents the square root of b , and as x approaches zero, the function approaches 1:

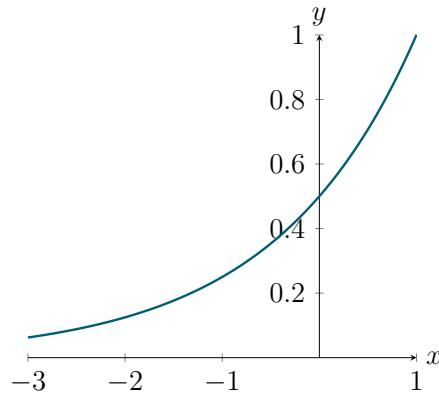


Figure 4.13: Well again the base $0 < b < 1$ but a fractional positive exponent.

- **Fractional Negative Exponent $-1 < x < 0$:** The behavior of the function also depends on the specific value of x . For example, if $x = -\frac{1}{2}$, the function $f(x)$ represents the reciprocal of the square root of b , and as x approaches zero, the function approaches 1:

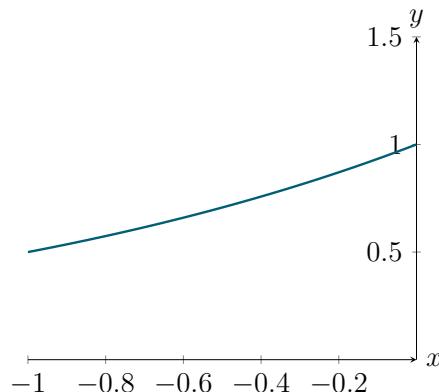


Figure 4.14: Guess what ? b is $0 < b < 1$ but the exponent is fractional and negative.

2. **$b = 1$** : In this case, the function $f(x) = a \cdot 1^{-x}$ simplifies to a constant value a for all values of x . The function remains constant, representing a horizontal line:

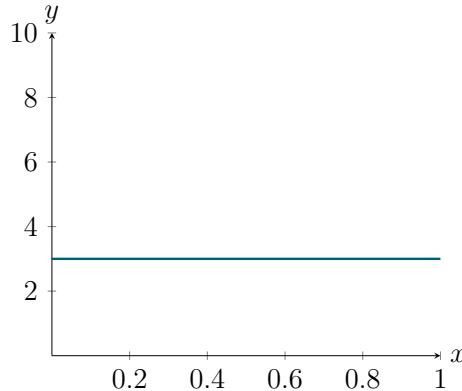


Figure 4.15: A boring line but still a case when $b = 1$.

3. **$b > 1$** :

- **Positive Exponent $x > 0$** : As x increases, the function $f(x)$ increases. The rate of increase becomes faster as x grows, approaching zero but never reaching it:

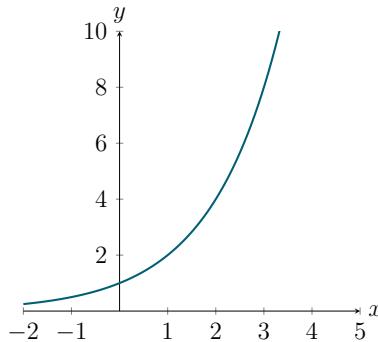


Figure 4.16: Now for the positive cases of b we start with a positive exponent $x > 0$.

- **Negative Exponent $x < 0$** : As x becomes more negative, the function $f(x)$ decreases. The rate of decrease becomes slower as x approaches negative 0, but never reaches it:

4.2. This seems important: **The Fundamental Theorem of Calculus.**

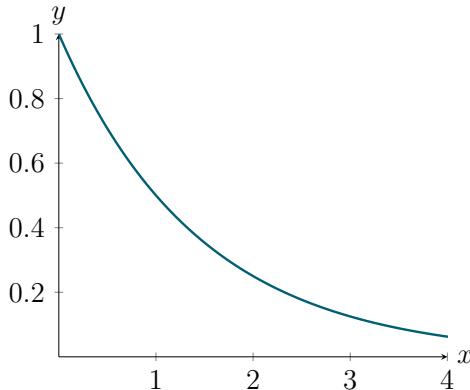


Figure 4.17: We follow with a negative exponent $x < 0$.

- **Fractional Positive Exponent $0 < x < 1$:** The behavior of the function depends on the specific value of x . For example, if $x = \frac{1}{2}$, the function $f(x)$ represents the square root of b , and as x approaches zero, the function approaches 1:

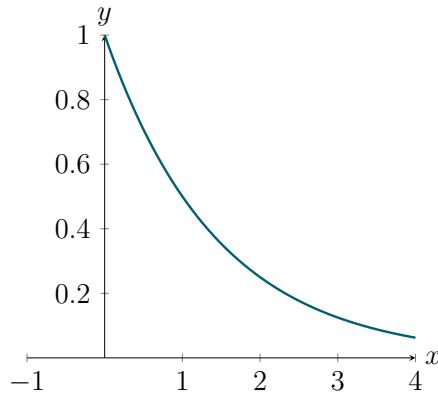


Figure 4.18: Still a positive base but now a fractional positive exponent.

- **Fractional Negative Exponent $-1 < x < 0$:** The behavior of the function also depends on the specific value of x . For example, if $x = -\frac{1}{2}$, the function $f(x)$ represents the reciprocal of the square root of b , and as x approaches zero, the function approaches 1:

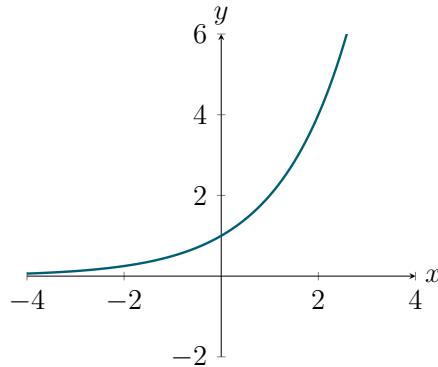


Figure 4.19: Lastly, a positive base but a negative fractional exponent.

In summary, for the exponential function $f(x) = b^x$ with positive values of b and varying x , the base b largely dictates the fundamental behavior, while the exponent x determines its direction and intensity. A value of b greater than 1 will lead to growth as x increases, but if x is negative, it translates to decay. Conversely, if b is between 0 and 1, it results in decay as x increases and growth when x is negative. For $b = 1$, the function remains constant regardless of x . Hence, the combined influence of b and x orchestrates the intricate dance of growth, decay, or stability in the exponential world.

Now, we are in a situation that allows us to pick a model to represent graph 4.9. From our analysis above, we need a positive base b and a negative exponent $-x$. But what base? The b can take any value, but these are the three most common:

- **Base 10:** This is a widely used base; here are a few examples: the decimal system, scientific notation, in which we use powers of 10, scales such as the Richter magnitude of an earthquake, and lastly, we have 10 fingers, which people believe led to the adoption of the decimal system.
- **Base 2:** While a base 10 seems more natural for us, base 2 is intuitive for computers as they operate on binary systems of 0 and 1's.
- **Base e :** This guy is called Euler's number. One way to deduce this value is to think about compound interest. Say that you

4.2. This seems important: **The Fundamental Theorem of Calculus.**

have an initial quantity of money P . When looking for a place to invest, we find a product that gives you 100% of interest a year. At the end of that period, we will have:

$$P(1 + 1) = 2P$$

That is good, but after researching, we find a bank that gives you 50% every 6 months, which is better, so:

$$P + \frac{1}{2}P + \frac{1}{2} \left(P + \frac{1}{2}P \right) = P \left(1 + \frac{1}{2} \right)^2 = 2.25P$$

This type of investment is more beneficial because it compounds, meaning that we will have more than with annual compounding. This scenario makes us wonder what would happen if we keep compounding, for example, every 4 months:

$$P \left(1 + \frac{1}{3} \right)^3 = 2.37P$$

What will happen if we compound n times?

$$P \left(1 + \frac{1}{n} \right)^n$$

Well, to maximize the amount of dough we could get, we need to make n tend to ∞ :

$$\lim_{n \rightarrow \infty} P \left(1 + \frac{1}{n} \right)^n$$

If we now focus on the limit:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n$$

The calculation of that limit is complex, and I will omit it:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n \approx 2.71828 \approx e$$

Before fully defining our function and moving into the calculation of derivatives and integrals (yeah, of course, we have to do that), there is something I have just spotted. When we introduced the notion of the inverse of a function, we said that we needed to swap the roles of the equation, meaning that we had to define x in the function of y . If we wish to translate this to English in the case of exponentials, it means: to what power must a given base be raised to obtain a certain number? We have no notation for this, but we will introduce some. Please meet the logarithm. This fellow is the inverse of the exponential function, and be sure that it is a super helpful function. We represent it like:

$$x = \log_b(y)$$

This might still seem abstract but think of it this way. Consider the following equality:

$$2^3 = 8 = 2 \cdot 2 \cdot 2 \quad (4.13)$$

Say that the base is a growth rate and the exponent is the number of times it will grow in the future. Equation 4.13 says that our quantity will double 3 times. So now, if we write 4.13 as a log:

$$\log_2 8 = 3 \quad (4.14)$$

What that tells us is: If our rate doubled and you ended up with 8, how long did we have to do it for? The answer is 3, the exponent of 4.13, which defines the unit of time. Now apply this concept to, for example, an exponent of 0.

$$2^0 = 1 \quad \log_2 1 = 0$$

If it grows by 0, then what? Nothing; it will remain the same. Similarly:

$$2^{-1} = \frac{1}{2} \quad \log_2 \frac{1}{2} = -1$$

This one is like going to the past; instead of going ahead and growing, we are shrinking! Because exponents and logarithms are a regular presence in machine learning, this is a less abstract way of looking into these logarithmic functions to grasp these concepts

4.2. This seems important: **The Fundamental Theorem of Calculus**.

better. With exponents and logarithms come some relationships that are worth exploring:

Exponential Laws	Logarithmic Laws
1. $a^m \cdot a^n = a^{m+n}$	1. $\log_b(mn) = \log_b(m) + \log_b(n)$
2. $\frac{a^m}{a^n} = a^{m-n}$	2. $\log_b\left(\frac{m}{n}\right) = \log_b(m) - \log_b(n)$
3. $(a^m)^n = a^{mn}$	3. $\log_b(m^n) = n \cdot \log_b(m)$
4. $a^{1/n} = \sqrt[n]{a}$	4. $\log_b \sqrt[n]{m} = \frac{1}{n} \cdot \log_b(m)$
5. $a^0 = 1$	5. $\log_b(b) = 1$

I think it will be an excellent exercise to go from the exponential law to the logarithmic one; we haven't dealt with these functions, so it will be a way for us to get a bit more familiarized with these concepts:

1. If we have $\log_b(m) = x$ then $m = b^x$; equally if $\log_b(n) = y$ then $n = b^y$. If we now multiply the quantities m and n we have that:

$$mn = b^x b^y = b^{x+y} \quad (4.15)$$

Now, we need to convert that exponential into a logarithm. For that let's do a switcheroo: $s = x + y$ and $p = mn$, so the exponential 4.15 can be written as:

$$p = b^s$$

If we invert it to have a logarithmic function, it is:

$$\log_b p = s$$

But $s = x + y$ and $p = mn$ so:

$$\log_b(mn) = x + y$$

Nearly there! When we started this proof we said that $\log_b(m) = x$ and $\log_b(n) = y$. Gotcha!

$$\log_b(mn) = \log_b(m) + \log_b(n)$$

Point 1 done and dusted.

2. This proof is very similar to the one done for the multiplication. We just have to change the multiplication to a division, and we will get that $\log_b\left(\frac{m}{n}\right) = \log_b(m) - \log_b(n)$.
3. The power rule... no, this is not politics... We are just dealing with math and logarithms. This proof is very similar to the one for point 1. If we consider that $m = b^x$ then $m^n = (b^x)^n$. By the rule of exponentials, we have:

$$m^n = (b^x)^n = b^{nx} \quad (4.16)$$

Again, what we need is some log's somewhere so let's do this: Take the logarithm base b of both sides of equation 4.16, we get:

$$\log_b(m^n) = \log_b(b^{nx})$$

By the property of logarithms, $\log_b(b^a) = a$, we simplify the right side to:

$$\log_b(m^n) = nx \quad (4.17)$$

We just had to make the correct substitution. Initially, we defined $m = b^x$, so $x = \log_b m$. We can replace this in equation 4.17:

$$\log_b(m^n) = n \log_b m$$

One more in the bag.

4. Points 3 and 4 also have similar proofs to the ones for 1 and 2.
5. Finally, we arrived at point 5, which is not bad, I promise. Consider the following equation:

$$b^1 = b$$

Let's apply the log rule on that thing:

$$\log_b(b) = 1$$

This represents the identity for logarithms.

4.2. This seems important: The Fundamental Theorem of Calculus.

What about the graph of this type of function, and what happens when we iterate over the possible values of x ? All good points, indeed, and something that we need to cover, so let's not try to avoid the inevitable and get on with it. For that consider a $f(x)$ such that:

$$f(x) = \log_a(x) \quad (4.18)$$

1. $x > 0$: The logarithmic graph increases towards negative infinity as x approaches 0 from the positive side. This indicates that the logarithm grows very large (in the negative direction). Conversely, as x increases from a small positive value, the logarithmic plot rises slowly and accelerates. The rate of increase becomes milder as x gets larger:

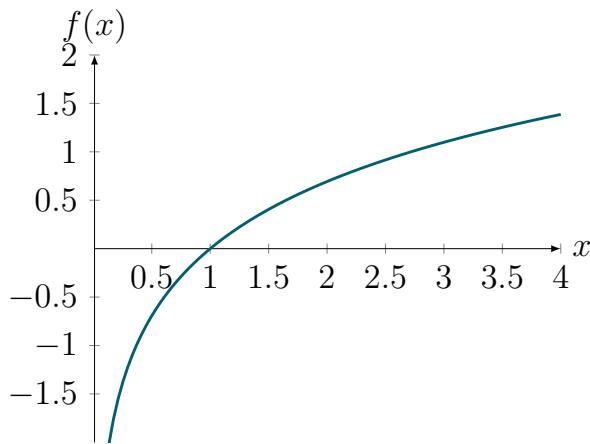


Figure 4.20: The first sighting of the logarithm.

2. $x = 0$: The logarithmic plot is undefined for $x = 0$, and we have a vertical asymptote. If we think about the question that the logarithm provides the answer to, what power must a given base be raised to obtain a certain number?

$$b^x = 0 \quad (4.19)$$

There is no value for x that satisfies equation 4.19.

3. $x < 0$: This is entering into the realm of complex analysis, something we won't cover in this book. So, in the world of real numbers, we can't define logarithms with negative values for arguments.

Regarding the impact of the logarithmic base selection, it works inversely to the exponentiation. This is to be expected as these two functions are inverse, so a logarithm with a base greater than 1 grows more slowly as the input increases, while a logarithm with a base between 0 and 1 decreases more rapidly. As the input values increase, the logarithm with a base greater than 1 increases at a decreasing rate, approaching but never reaching infinity. On the other hand, a logarithm with a base between 0 and 1 decreases in value more rapidly, approaching negative infinity but never reaching it.

So, finally, we are ready to pick the equation to reflect the phenomena we are trying to capture, the one we introduced before all that madness of the exponentials and logarithms. Because the methodology we will use to devise the total pheromone concentration is an integral, we need an equation that accommodates plot 4.9. I understand it was a while back that the plot was introduced, so here it is again as a reminder:

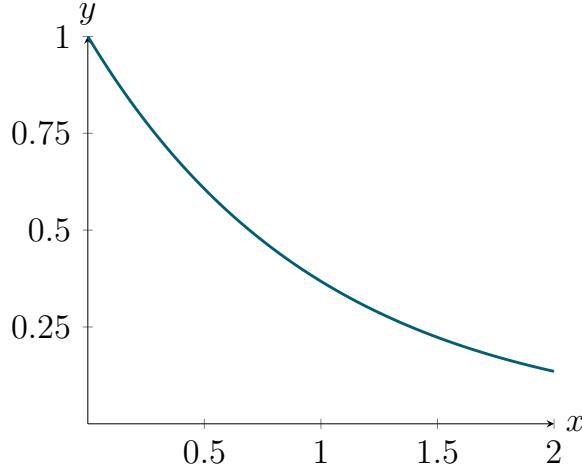


Figure 4.21: A reminder of how fast the concentration of this pheromone dissipates.

Well, from all of what we saw on the several iterations of the exponential terms, we need an exponential with a negative exponent, a decay. For the base, let's go with e . So our model is:

$$f(x) = e^{-x} \quad (4.20)$$

If we are after the total concentration of pheromones released

4.2. This seems important: **The Fundamental Theorem of Calculus**.

between 0 and 2, we need to integrate 4.20:

$$\int_0^2 e^{-x} \quad (4.21)$$

We need to find an anti-derivative. The question is: which function, when derived, gives us e^{-x} ? Let's take one step back in hopes of then taking several forward and deducing the derivative of e^x ; yeah, don't tell me you were not missing it already; that "lovely" definition of a derivative:

$$\frac{f(x)}{dx} = \frac{f(x+h) - f(x)}{h}$$

Replacing $f(x)$ with e^x :

$$\frac{d}{dx} e^x = \lim_{h \rightarrow 0} \frac{e^{x+h} - e^x}{h}$$

Now, we can use the power rule for the exponential and manipulate that equation into something like this:

$$\frac{d}{dx} e^x = \lim_{h \rightarrow 0} \frac{e^x (e^h - 1)}{h}$$

Since we calculate the limit as h tends to 0 we can "push" e^x outside:

$$\frac{d}{dx} e^x = e^x \lim_{h \rightarrow 0} \frac{(e^h - 1)}{h}$$

We have a $\frac{0}{0}$, and we can't use the L'Hopital's rule as we don't know the derivative of e^h . Throughout this book, we've frequently changed variables with great success. Let's apply this approach again:

$$e^h - 1 = m$$

Then:

$$e^h = m + 1$$

Applying the logarithmic properties:

$$\ln(m + 1) = h$$

When the logarithm base is e , we can use the notation \ln , which stands for natural logarithm. We should leverage all of these equations and substitute them in the limit formula:

$$= e^x \lim_{h \rightarrow 0} \frac{m}{\ln(m + 1)}$$

We have a limit with a h tending to 0, but no h 's in the expression. So what does it mean when $h \rightarrow 0$?

$$e^h - 1 = m \Rightarrow e^0 - 1 = m$$

Which implies that:

$$m = 0$$

When h approaches 0, m will also approach 0:

$$= e^x \lim_{m \rightarrow 0} \frac{m}{\ln(m + 1)} \tag{4.22}$$

Now, it is time to look back at everything we have learned so far and try to find something that will help us with that nasty expression. In equation 4.22, we have both exponentials and logarithms of the same base e , so let's gather intelligence around those assets. Sorry, I just wanted to sound like some CIA person when I wrote that last sentence. What we need is the definition of e :

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n \tag{4.23}$$

On the right side of equation 4.22, we have a limit that is not that different to 4.23, so perhaps we can work with this. One issue is that in 4.22 m tends to zero, but in 4.23 n is tending to ∞ . Okay, but if we defined a new variable m such that:

$$m = \frac{1}{n}$$

We have that when $n \rightarrow \infty$, then, $m \rightarrow 0$. If we do a variable substitution in 4.23, comes:

$$e = \lim_{m \rightarrow 0} (m + 1)^{\frac{1}{m}} \tag{4.24}$$

4.2. This seems important: **The Fundamental Theorem of Calculus**.

Equation 4.24 has more in common with equation 4.22, but we still need to do a bit more manipulation. I want to draw your attention to two things: the exponent of 4.24, which is equal to $\frac{1}{m}$ and also the third logarithmic rule:

$$\log_b(m^n) = n \cdot \log_b(m)$$

Indeed, by introducing a factor of $\frac{1}{m}$ to multiply the denominator in equation 4.22, we are likely to facilitate some meaningful manipulation with the limit depicted in equation 4.23:

$$e^x \lim_{m \rightarrow 0} \frac{\frac{1}{m}m}{\frac{1}{m} \ln(m + 1)}$$

The numerator simplifies to 1, whereas if we apply the logarithmic rule we just spoke about, the denominator takes the form of $\ln(1 + m)^{\frac{1}{m}}$ so:

$$e^x \lim_{m \rightarrow 0} \frac{1}{\ln((m + 1)^{\frac{1}{m}})} \quad (4.25)$$

The denominator looks a lot like the definition of e , which we have represented by 4.24. When we studied the limits, we covered one case that will allow us to finish this madness in glory:

$$\lim_{x \rightarrow c} f(g(x)) = f\left(\lim_{x \rightarrow c} g(x)\right)$$

So for the logarithm function:

$$\lim_{x \rightarrow c} \log(g(x)) = \log\left(\lim_{x \rightarrow c} g(x)\right)$$

Alright, this means that 4.25 can then be written as:

$$e^x \frac{1}{\ln\left(\lim_{m \rightarrow 0} (m + 1)^{\frac{1}{m}}\right)}$$

But that is the definition of e !

$$e^x \frac{1}{\ln(e)} = e^x$$

What this means is that:

$$\frac{d}{dx} e^x = e^x$$

Really? At any point on the curve, $y = e^x$, the slope of the tangent line at that point is equal to the image of that point? Indeed! Such characteristics are significant as they simplify many problems and equations. The question now is, what happens if, for a base, we have something other than e ? Consider the exponential defined as:

$$f(x) = a^x \quad (4.26)$$

We know the derivative of e^x , so, if we can find a way to involve that term in equation 4.26, it would be a good start. We can't turn x or y into e^x , so the only option is to manipulate a . Sorry a , but we will have to put you to work:

$$a = e^x \rightarrow x = \ln(a)$$

So:

$$a = e^{\ln(a)}$$

We just substituted x for $\ln(a)$ on the right side equation. Now, if we use these results and substitution in equation 4.26, we have that:

$$f(x) = e^{\ln(a^x)}$$

By applying law 3 for logarithms:

$$f(x) = e^{x \ln(a)}$$

Cool, so we are after the derivative of $f(x)$, and for that, we will use the chain rule:

$$f'(g(x)) = f'(g(x))g'(x)$$

For $f(x)$ and $g(x)$ we can have them as:

$$f(x) = e^x, \quad g(x) = x \ln(a)$$

Then:

$$f'(x) = e^x, \quad g'(x) = \ln(a)$$

4.2. This seems important: **The Fundamental Theorem of Calculus.**

So:

$$f'(g(x)) = e^{x \ln(a)} \ln(a)$$

But we know that:

$$e^{x \ln(a)} = a^x$$

Finally:

$$\frac{d}{dx} a^x = a^x \ln(a) \quad (4.27)$$

We are now ready to calculate the antiderivative of e^{-x} so we can finally get that total concentration of pheromones. So, what function can I derive to obtain e^{-x} ? $-e^{-x}$? Let's see!

$$f(x) = -e^{-x}$$

Chain rule here we go again:

$$f(x) = -e^{-x} \quad g(x) = -x$$

We have that:

$$f'(x) = e^{-x} \quad g'(x) = -1$$

Applying the rule:

$$f'(g(x)) = e^{-x} \cdot (-1) = -e^{-x}$$

If we define $F(x)$ as the anti-derivative, we can see that our integral can be calculated with the following expressions:

$$\int_0^2 e^{-x} = F(2) - F(0)$$

And in turn:

$$\int_0^2 e^{-x} = -e^{-2} - (-e^0) = 0.864$$

There: we have it! The total concentration in ppt of the pheromones between 0 and 2 in terms of time is 0.864. Integral calculus, mainly through its application in aggregating cumulative measures, is central to statistics, a fundamental element in machine learning. This significance becomes apparent in density functions, where integrals

are indispensable for calculating probabilities. Statistics and probabilities are the topics of volume 3 of this series, so we will cover them there. However, there is still some work to do with these exponentials.

So, we now know how to integrate exponential functions and consequently derive them. We are also capable of answering questions such as "At what time did we have a ppt per hour with a value of 0.15?" We don't? Well, check this out! The inverse of e^{-x} has the following expression:

$$f^{-1}(x) = -\ln(x) \quad (4.28)$$

If we want to know at what time we had a ppt/time of 0.15, we just have to input this number in equation 4.28:

$$f^{-1}(0.15) = -\ln(0.15) = 1.89$$

We still haven't gone over the derivatives of the logarithmic functions, though, something that can't be ignored. In the same way we did with the exponentials, let's start with $\ln(x)$, and the tactic will be to try to use the definition. Brace for impact, as this one is long:

$$\frac{d}{dx} \ln(x) = \lim_{h \rightarrow 0} \frac{\ln(x+h) - \ln(x)}{h}$$

There it is, that definition, as if we haven't seen it enough. GO AWAY! I think we need it once more, but, this time we can make our job a bit easier by leveraging this law: $\ln(a) - \ln(b) = \ln(\frac{a}{b})$:

$$= \lim_{h \rightarrow 0} \frac{\ln\left(\frac{x+h}{x}\right)}{h}$$

Which comes to:

$$\begin{aligned} &= \lim_{h \rightarrow 0} \frac{1}{h} \ln\left(1 + \frac{h}{x}\right) \\ &= \lim_{h \rightarrow 0} \ln\left[\left(1 + \frac{h}{x}\right)^{\frac{1}{h}}\right] \end{aligned}$$

Now we will have to do a change of variable:

$$m = \frac{h}{x} \Rightarrow h = mx$$

4.2. This seems important: **The Fundamental Theorem of Calculus**.

Which means that when $h \rightarrow 0$ then $m \rightarrow 0$, so:

$$= \lim_{h \rightarrow 0} \ln \left[(1 + m)^{\frac{1}{mx}} \right]$$

We can work with that:

$$= \lim_{h \rightarrow 0} \frac{1}{x} \ln \left[(1 + m)^{\frac{1}{m}} \right]$$

Pushing the fraction with the x outside:

$$= \frac{1}{x} \lim_{h \rightarrow 0} \ln \left[(1 + m)^{\frac{1}{m}} \right] \quad (4.29)$$

If we recall one of the definitions of e , that nasty one with the limit:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n$$

This is almost what we have on equation 4.29, if we now do another change of variable:

$$n = \frac{1}{m}$$

If $m \rightarrow 0$ that implies $n \rightarrow \infty$ and consequently:

$$= \frac{1}{x} \ln \left[\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n \right]$$

Well, that is very convenient as the right side represents the number e :

$$= \frac{1}{x} \ln(e) = \frac{1}{x}$$

So:

$$\frac{d}{dx} \ln(x) = \frac{1}{x}$$

That was long, but at least the result is a nice, neat little expression. Bad news: we are not done yet, as we also need a formula for all bases of logarithm and not just e . We have acquired quite a bit of knowledge around the base e , and it has been shown to have properties that allow the simplification of calculations, so, as we did when deducing the formula to calculate the derivatives of the exponentials we are gonna try to find some relationship between \log_a

and \ln . For that, we will use a technique called a change of basis. Alright, let's get this party started; we know that:

$$b^x = a \Rightarrow \log_b(a) = x$$

Let's work with the left side first. If we apply \log_c on both sides of the equation, where c is a different base to b :

$$\log_c(b^x) = \log_c(a)$$

We can apply law 3 of the logarithmic laws on the left side:

$$x \log_c(b) = \log_c(a)$$

Solving with respect to x :

$$x = \frac{\log_c(a)}{\log_c(b)}$$

This all started with $x = \log_b(a)$ so:

$$\log_b(a) = \frac{\log_c(a)}{\log_c(b)}$$

So, for any given x :

$$\log_b(x) = \frac{\log_c(x)}{\log_c(b)} \quad (4.30)$$

Equation 4.30 shows how we can go from base b to base c . Before moving into the derivative calculations, let's bring a bit of context to this concept of change of basis. Say we want to check at what time we have a ppt of 0.05:

$$-\ln(0.05) = 3$$

This equation states that it takes 3 units of time for the pheromone to decay to 0.05 ppt according to your model with a natural logarithm. Now, we could change the base from e to 10.

$$\log_{10}(0.05) = \frac{\ln(0.05)}{\ln(10)} = 1.302$$

4.2. This seems important: The Fundamental Theorem of Calculus.

OK, so it takes approximately 1.302 units of time for the gas to decay to 0.05 ppt according to the base 10 model. Changing the base essentially involves using a different "scale" or perspective to measure an exponential relationship. Logarithms in different bases are proportional to each other and represent the same underlying relationships, just expressed in different ways. It is a bit like the way we can measure time in seconds, minutes or hours.

Now, for the derivative calculation, let's consider a $\log_b(x)$ function of a given base b , and since e is a base that so far has helped us with the calculation, let's perform a change of basis, from b to e :

$$\log_b(x) = \frac{\ln(x)}{\ln(b)}$$

Now we have it easy, finally!

$$\frac{d}{dx} \log_b(x) = \frac{d}{dx} \frac{\ln(x)}{\ln(b)}$$

As $\ln(b)$ is a constant, we have that:

$$\frac{d}{dx} \log_b(x) = \frac{1}{x \ln(b)}$$

Before delving into the complexities of high dimensionality, it's essential to highlight another critical use case for logarithms. Beyond their ability to change the basis, logarithms possess a unique property: they can convert multiplicative changes into additive ones. This transformation is invaluable, especially when observing data that spans multiple orders of magnitude.

The additive perspective granted by logarithms offers a potent tool in data analysis and machine learning. Specifically, logarithms come to our rescue when datasets present wide ranges. By applying logarithmic scaling, we can simplify calculations, comparisons, and interpretations of widespread data, making it more tractable and understandable. Drawing upon our understanding of logarithms and their capability to simplify wide ranges, let's bring back our example where we fitted a line to the data, the one where we had the measurements of the number of steps taken by the bees, and the nectar collected:

Steps taken x	Nectar collected (mg) y
1	2
2	4
3	6
4	8

Table 4.1: The case of steps vs. nectar collected once more.

This is precisely the same table as before, and, in addition, we will also plot the data as another helpful reminder:

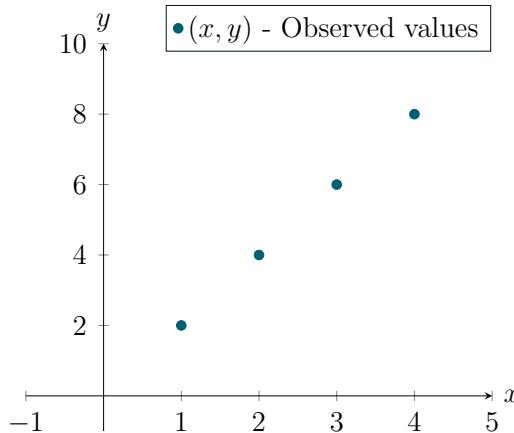


Figure 4.22: Again the steps and the nectar?

Previously, we selected a simple linear regression model for our data, given by the equation $y = mx$. This model aims to fit a straight line to the data, where y is our prediction and x is the input. To find the value of m that minimizes the difference between our predictions and the actual data points (measured by a convex cost function like the sum of squared errors), we used an optimization technique called gradient descent. By iteratively adjusting m , gradient descent finds the value that best fits our data. The resulting model, after optimization, was:

$$y = 1.705x$$

Well, we know that $y = 2x$ fits the data perfectly and that would have been achieved had we iterated more, so let's work with that model, meaning:

$$y = 2x \tag{4.31}$$

4.2. This seems important: **The Fundamental Theorem of Calculus**.

Excellent, if we plot the line represented by 4.31 alongside the dots, we can have an idea of how well it fits the data:

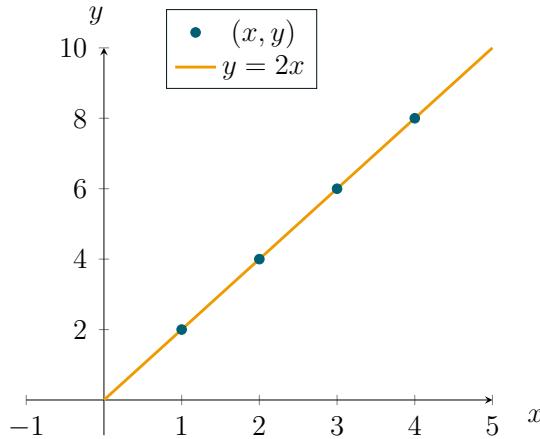


Figure 4.23: The yellow line that best fits our data, $y = 2x$.

The cost function we used was the MSE, that guy with the following equation:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2$$

That \hat{y}_i represents the i_{th} prediction given by our model, so to calculate the error, we must calculate what value of y (or the amount of honey produced) our model spits out when we input the number of steps taken, represented by x :

x	y	\hat{y}	$(y - \hat{y})^2$
1	2	2	0
2	4	4	0
3	6	6	0
4	8	8	0

Table 4.2: A wishful look at a error of 0.

So for MSE, we have a value of 0; every prediction matches the exact number of observed honey collected, meaning $y_i = \hat{y}_i$ for all the y 's. Now, if this happens when you're analyzing data, it most likely indicates poopoo.

This is an excellent opportunity to review the methodology and the code, as some of these pitfalls might have happened:

1. **Overfitting:** Your model might be too detailed for the data. It's like using a map with every single rock drawn on it when all you need are the main roads.
2. **Data Leakage:** Imagine accidentally getting a peek at the answers before a test. That's what data leakage is like for models. If your model somehow saw the test data before being tested, it'll perform too well.
3. **Duplicates:** It's like studying the same page of a textbook over and over. If your data has too many repeats, your model might think it's learned much when it hasn't.
4. **Coding Mistakes:** Sometimes, a misplaced line of code can make it seem like the model's doing fantastically well when it is just echoing what it saw.
5. **Data that is too Simple:** If all the questions in a quiz are the same, everyone will score 100%. Even a basic model can look like a star if your data lacks variety.

As we are showcasing how to apply the techniques and concepts we learned, the dataset for the example is elementary, which is how we get an error of 0. Let's explore a new scenario, one where we get an additional point:

$$\text{Point}_5 = (5, 20)$$

Let's check how well our previous model can predict this new measure:

4.3. Hold On, Logs Aren't Just for Fireplaces? Scaling Data.

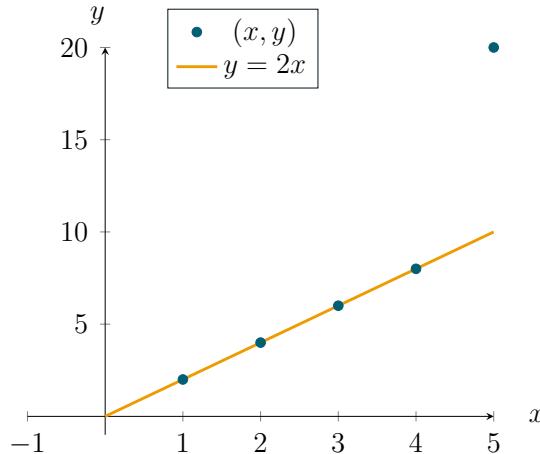


Figure 4.24: That point looks like trouble. All of a sudden, the yellow line doesn't look that good.

4.3 Hold On, Logs Aren't Just for Fireplaces? Scaling Data.

Oh yeah! We definitely have some work to do, as where the new point stays is a long way from the yellow line compared to the previous ones we used to fit the line to the data. What this suggests is that equation 4.31 is no longer the best fit for our data, but first let's see what happened to the MSE when we calculate the \hat{y} of $x = 5$:

x	y	\hat{y}	$(y - \hat{y})^2$
1	2	2	0
2	4	4	0
3	6	6	0
4	8	8	0
5	20	10	100

Table 4.3: There we go, the zero error is gone and now we have to deal with that big 100 there.

So for the MSE calculation we have:

$$\text{MSE} = \frac{1}{5} (0 + 0 + 0 + 0 + 100) = 20$$

It's disappointing, isn't it? We had a smooth linear relationship, and then that bastard new point came along and disrupted it. The audacity! But all hope isn't lost. Remember our friends the logarithms? They transform multiplicative relationships into additive ones. This property could be our way to reintroduce some linearity. We'll leverage the natural logarithm, denoted as \ln , and apply it to our nectar quantities. So, we're looking at computing $\ln(y)$:

x	y	$\ln(y)$
1	2	0.693
2	4	1.386
3	6	1.792
4	8	2.079
5	20	2.991

Table 4.4: A cry for help. Log please helps us with the scale of the data.

Huuuuuuuh, that appears to be far better! The $\ln(y)$ points seem pretty close to each other!

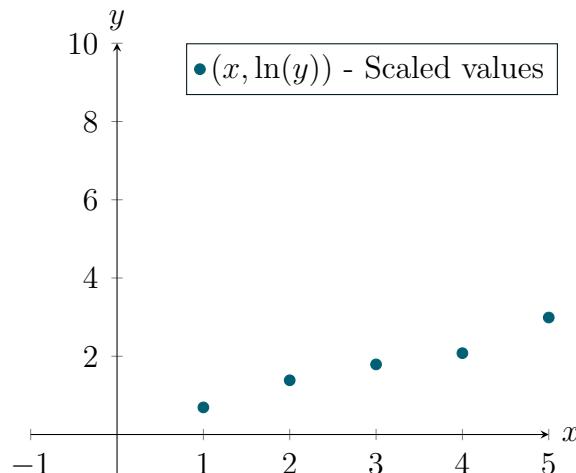


Figure 4.25: A way better representation of the same data, the scaled version.

Well, the \ln came to the rescue, and we can enjoy the simplicity of a linear world again:

4.3. Hold On, Logs Aren't Just for Fireplaces? Scaling Data.

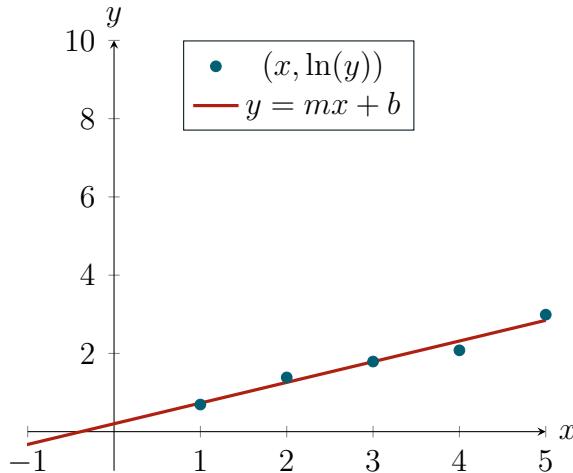


Figure 4.26: Another straight line that fits our scaled data like a glove.

On graph 4.26, the red line represents an ideal fit for our data. It's a visual representation of our goal. However, a simple model like $y = mx$ won't be sufficient this time. Unlike in previous scenarios, where the line was expected to pass through the origin, here it isn't guaranteed to cross the y -axis at the point $x = 0$. This indicates the need for a model with an added degree of freedom, we need a gangster like b , meaning $y = mx + b$. One possible equation is:

$$y = 0.5287x + 0.2019 \quad (4.32)$$

As always, m and b are random choices. Well, in all fairness, I cheated a bit as I picked values that made the line nearly fit to perfection, but don't worry, there is still room for improvement. If we now create a little table to introduce the calculation of the new MSE, this is what it would look like:

x	$\ln(y)$	\hat{y}
1	0.693	0.730
2	1.259	0.934
3	1.787	1.149
4	2.079	2.316
5	2.991	2.845

Table 4.5: The \hat{y} values given by the red line.

Now we must see if this new model, with an extra parameter, is producing better results than our original $y = 2x$, and what better method of comparison than measuring how bad each of them is?

We have a metric for that, the MSE. However, there's a caveat when comparing the MSE values generated by the model referenced in 4.32. The predicted values \hat{y} in table 4.6 were derived from scaled data. We must revert these values to their original scale to ensure a fair comparison, something we can do by applying the inverse of the logarithmic function, the exponential:

x	y	\hat{y}	$e^{\hat{y}}$	$(y - e^{\hat{y}})^2$
1	2	0.730	2.075	0.001
2	4	0.934	2.544	2.119
3	6	1.149	3.155	8.094
4	8	2.316	10.135	4.558
5	20	2.845	17.201	7.834

Table 4.6: A table to aid us with the MSE calculation for the new red line model.

Since we are utilizing the original scale for comparison purposes, we will calculate our error using the squared difference $(y - e^{\hat{y}})^2$, rather than the simple difference $y - \hat{y}$. Consequently, the following relationship holds:

$$\text{MSE} = \frac{1}{5} (0.001 + 2.119 + 8.094 + 4.558 + 7.834) = 4.537$$

Well, that's a significant leap forward. Our MSE dropped from 20 to just 4.437, underscoring the potential of logarithmic functions. In machine learning, scaling data is often of paramount importance. A large portion of machine learning tasks revolve around optimization. The objective is clear: find the best parameters for our equation to achieve the closest fit to our data. Gradient descent is one of the commonly used algorithms for this purpose. This optimization technique converges more swiftly when data features are on a similar scale. However, when scales differ significantly, the gradient updates for one part might eclipse those for another, potentially leading to slower convergence or even non-convergence. With this foundation, it's time to revisit the model that yielded the red line:

$$y = 0.5287x + 0.2019 \tag{4.33}$$

Equation 4.33 represents our initial guess; although it visually seemed like a good fit, we are still determining whether it is the best we can get. We must optimize our cost function according to

4.3. Hold On, Logs Aren't Just for Fireplaces? Scaling Data.

the model's parameters. With our new equation, we have a tiny difference that makes a huge impact, just like a small kidney stone. This is because, in the new model, we have an extra parameter. So we must revisit our cost function, as now we have two levers we can pull: m and b :

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$$

Because the MSE is convex, we decided to use the gradient descent by calculating the derivative of MSE with respect to m :

$$\frac{d}{dm} \text{MSE}$$

OK, so that means that the MSE is a function of only m , which we can define as:

$$J(m) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 \quad (4.34)$$

Where:

$$y = mx$$

But now we have:

$$y = mx + b$$

So, both m and b will impact the error of our model, the MSE. For example, if we change the b value of our model from 0.2019 to 1:

$$y = 0.5287x + 1$$

Let's check what happens to the predictions:

x	y	\hat{y}	$e^{\hat{y}}$	$(y - e^{\hat{y}})^2$
1	2	1.528	1.413	0.3445
2	4	2.057	7.822	14.607
3	6	2.586	6.667	0.444
4	8	3.114	22.511	210.569
5	20	3.644	39.017	361.646

Table 4.7: Another table with metrics to see if we improved the MSE.

Our MSE is:

$$\text{MSE} = \frac{1}{5} (0.3445 + 14.607 + 0.444 + 210.569 + 361.646) = 117.43$$

Oh wow! That looks like crap, meaning that the b parameter is important, so we must also optimize. In other words, it has to be part of the cost function, so equation 4.34. For that, we will start by changing the notation:

$$J(m, b) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 \quad (4.35)$$

We also have:

$$f(x) = mx + b$$

Well, well, well, now we have two variables to optimize ... don't worry; this is when things get interesting because we need a new concept: partial derivatives.

However, to understand such an idea, we are going to have to dive into higher dimensionality; finally, something more applicable to real-world phenomena!



Chapter 5

A Free Upgrade: More Dimensions.

We started with a single input to a single output function, something that is not that complex, but it allowed us to understand some crucial concepts. Now it is time to scale everything we learned up, by exploring two more types of functions: scalar functions, which are mappings of multiple inputs to a single output, and various inputs to multiple outputs, which are called vector functions. By doing this, we can contextualize all these techniques with machine learning and understand that, with derivatives and some linear algebra, there is a lot we can accomplish. Here is an overview:

1. Single output and single input functions:

$$y = f(x)$$

2. Single output and multiple input functions:

$$y = f(x_1, x_2, \dots, x_n)$$

3. Multiple output and multiple input functions:

$$(y_1, y_2, \dots, y_n) = f(x_1, x_2, \dots, x_m)$$

We are now ready to start dealing with scalar functions, which receive multiple inputs but produce one output. With these mappings will come some new notions, which should be exciting as we are here to learn. So let's introduce them with the following scenario.

5.1 Numerous Entries But Single Exit: Scalar Functions.

Bees have to build their houses. Unfortunately, we haven't got real estate agent bees or beehive construction firms. They construct their beehive through a collective process that involves several steps. First, worker bees produce beeswax by converting sugar-rich honey into wax flakes. They shape these flakes by chewing and manipulating them with their mandibles and legs. The pliable beeswax is then moulded into hexagonal cells, forming the honeycomb's foundation. The construction starts from the top of the hive and progresses downward.

During the construction process, bees also build the outside structure of the beehive. They attach the comb to the walls and frames of the hive using beeswax as a natural adhesive. This attachment process occurs throughout the construction of the honeycomb. As the bees build new cells and extend the comb, they secure it to the hive's structure. The bees create a cohesive and stable beehive by continuously adding new wax and connecting it to the existing comb. This collaborative effort, encompassing both the internal honeycomb construction and the attachment to the outside structure, creates a remarkable beehive that serves as the bees' home, food storage, and nursery. Let's say that the following function represents a similar shape to a beehive:

$$\begin{aligned} f : \mathbb{R}^2 &\rightarrow \mathbb{R} \\ f(x, y) &\rightarrow x^2 + y^2 \end{aligned} \tag{5.1}$$

The notation on equation 5.1 describes a scalar function, a type of function that takes inputs from a two-dimensional space (represented by \mathbb{R}^2) and maps them to a single-dimensional output (represented by $f : \mathbb{R}^2 \rightarrow \mathbb{R}$). In simpler terms, imagine taking two different factors, such as temperature and pressure, and using them to calculate a single outcome like density. This is what a scalar function does; it synthesizes multiple inputs into one conclusive output.

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Now for the beehive shape, the best way to see if this resembles it will be to plot it:

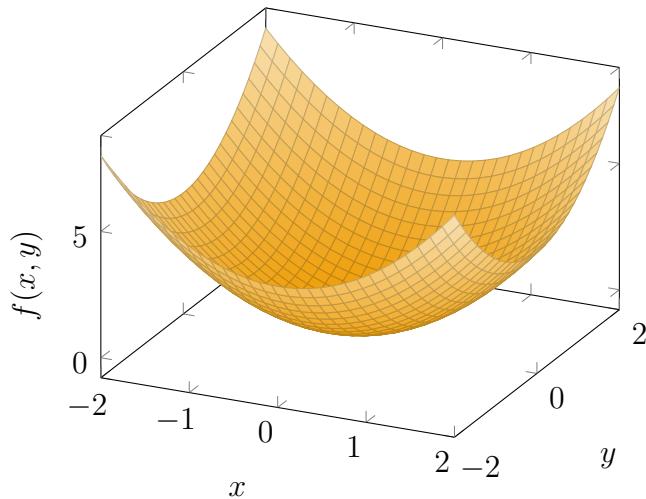


Figure 5.1: The beehive of a colony of bees, well, half of it.

As you can see in figure 5.1, what we're dealing with isn't quite the perfect beehive. The function 5.1 that generates it will never close at the top. As x and y stretch towards infinity, so does $f(x, y)$. But that doesn't mean we can't learn something from that attempted representation of half a beehive.

5.1.1 Grab Your 3D Glasses for these Multidimensional Integrals.

For example, we could ask how many bees we can possibly fit in half of it. Before you even think about crafting a colorful nickname for me, let's see what we already have that might help us answer this question.

Using the concept of infinity, we created an approximation for the volume of a bee. That volume is a good starting point for the task at hand, as we need to understand the volume of occupancy of a single bee. We also require the size of half a beehive, and here is where we can take a moment to appreciate the wonders of integrals. In the simplistic world of one-dimensional functions, integrals grace

us with a representation of area. But now we are dealing with a 3-D scenario. Does this mean we will get a volume instead of an area? Oh yes, baby, it does!

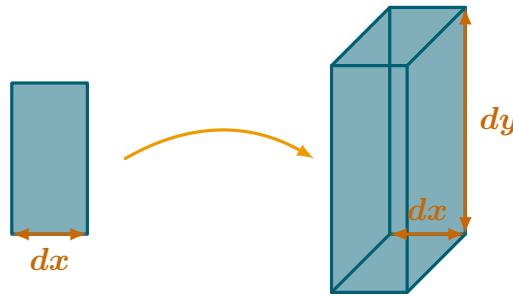


Figure 5.2: From 2d to 3d approximations.

Previously, we started by approximating the area under a curve using rectangles, gradually making these rectangles infinitesimally small to arrive at the definition of an integral. Similarly, for a 3-D surface, we'll use parallelepipeds in place of rectangles. However, our goal now shifts from calculating an area to determining a volume:

$$\text{Vol}_{\text{par}} = f(x, y)dx dy \quad (5.2)$$

Equation 5.2 represents the formula to calculate the volume of figure 5.2. OK, so we know you to compute the volume of one of the parallelepiped, but we need the sum of several of them so we can have an approximated value for the volume:

$$R = \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j)dx dy \quad (5.3)$$

As we stand now, equation 5.3 is kind of cute. It will approximate the volume if we consider N parallelepipeds, but that is for kids. We are interested in understanding what happens where the world ends: yeah, at infinity:

$$\lim_{N \rightarrow \infty} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j)dx dy \quad (5.4)$$

The limit represented by 5.4 is exactly the double integral of $f(x, y)$ over the region R :

$$\int \int_R f(x, y)dx dy \quad (5.5)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

We can compute this double integral to find the exact volume under the surface over the region R , which, in our terms, means half of a beehive. We need to consider what limits form half of this beehive. A random kind of criteria like this is just an example, so let's consider a "roof" defined at $0 \leq x \leq 3$ and $0 \leq y \leq 3$ so that:

$$\int_0^3 \int_0^3 (x^2 + y^2) dy dx \quad (5.6)$$

To compute the integral represented by 5.6 we will work inside out, just as we would at a fancy dinner, using multiple pieces of cutlery, from the inside out! As we have dy first, we will get the antiderivative of $x^2 + y^2$ by considering x to be a constant:

$$\int_0^3 (x^2 + y^2) dy = x^2 y + \frac{1}{3} y^3 \Big|_0^3$$

We introduced some new notation, with that bar. It just means we will be inputting the limits of integration and, in this case, the limits of y :

$$3x^2 + \frac{1}{3}(27) - (0x^2 + 0)$$

Simplify this to get:

$$3x^2 + 9 \quad (5.7)$$

Now integrate 5.7 with respect to x :

$$\int_0^3 (3x^2 + 9) dx = x^3 + 9x \Big|_0^3$$

Evaluate this at $x = 3$ and $x = 0$ to find the final volume:

$$27 + 27 - (0 + 0) = 54 \text{ cm}^3$$

So, the volume under the surface $f(x, y) = x^2 + y^2$ and above the region in the xy -plane is 54 cm^3 . The only thing missing is an approximation of a bee's volume; for that, we need to estimate the base of the cylinder, the shape we decided to approximate the bee with.

A bee's head houses a pair of large, compound eyes covering most of the head's surface, allowing it to have a nearly 360-degree field of

vision. These eyes, placed on a head about 2 to 3 mm wide, comprise thousands of tiny lenses that help the bee perceive its surroundings in great detail.

We will use this information about the head to estimate the radius and perimeter of the circle that forms the cylinder's base. Given a width of about 2.5 mm (midway between 2 and 3 mm), the radius would be about 1.25 mm. Because the perimeter is given by $2\pi r$, it comes to 7.85 mm. So the volume of the bee is:

$$V_{bee} = 1.25 \cdot \frac{7.85}{2} \cdot 20 = 98.125 \text{ mm}^3 \quad (5.8)$$

That formula 5.8 gives volume of a cylinder; the 20 represents the length of a bee. Now we need to convert the volume to cm^3 and divide it by the volume of half of the beehive:

$$\text{Capacity} = \frac{54}{0.0981} \approx 550$$

So we can fit 550 bees; what a bad beehive! This happened because of the limits of integration, but at least we were learning to operate with multi-dimensional integrals.

On the topic of bees, usually, the entrance for the beehive is at the bottom of it; in this case, the bees create it at the very bottom. In mathematical terms, the fact that the way into the hive is at the very bottom means that we are on the quest for a global minimum. For that, we need the gradient descent or the Newton method, which implies derivatives.

5.1.2 I Am Quite Partial to These Derivatives.

Firstly, we need to introduce a new concept, partial derivatives. Because we now have multiple inputs, we will check how the rate of change of this function behaves, one variable at a time. So, we will analyze these rates in the direction of one variable while making the others constant. The notation for this new concept is:

$$\frac{\partial f}{\partial x} \quad (5.9)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

The expression 5.9 represents the partial derivative with respect to x ; the definition of it is:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x + h, y) - f(x, y)}{h} \quad (5.10)$$

Cool, let's do it step by step. Firstly, we will replace $f(x, y)$ with $x^2 + y^2$:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{(x + h)^2 + y^2 - (x^2 + y^2)}{h}$$

We can expand the term $(x + h)^2$ and hope there is some simplification we can do:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 + y^2 - (x^2 + y^2)}{h}$$

Would you look at that? Amazingly, that turns into:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{2xh + h^2}{h}$$

If we factor h out:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{h(2x + h)}{h}$$

And that simplifies to:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} 2x + h$$

Finally:

$$\frac{\partial f}{\partial x} = 2x$$

When dealing with functions of two variables, such as $f(x, y)$, we employ the concept of partial derivatives. This involves differentiating with respect to one variable while holding the other constant. For instance, when computing the partial derivative with respect to x , we treat y as if it's a constant. This partial derivative, evaluated at a point (x, y) , gives the slope of the tangent line to the curve obtained by fixing y and varying x in the 3D space spanned by x, y , and $f(x, y)$:

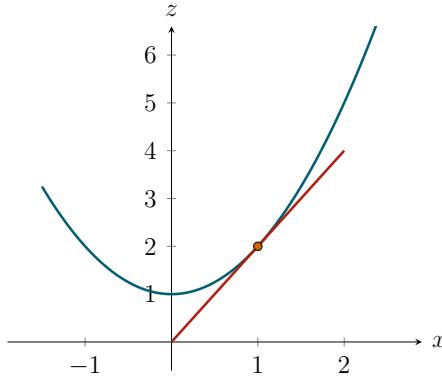


Figure 5.3: A slice of the beehive, with a tangent line at a given point.

Defining y as a constant is analogous to slicing the curve with a plane. For instance, consider the plane corresponding to $y = 1$ depicted in graph 5.3. With this slice, the equation for $f(x, y)$ simplifies to:

$$f(x, y) = x^2 + 1 \quad (5.11)$$

In the plot 5.3, the blue line represents a parabola, whereas the red color line is tangent to that curve at the orange dot, a point with coordinates $x = 1$ and $z = 2$. To derive the equation of that line, we can start with the slope, which will be the value of the partial derivative of x at $x = 1$ so we that:

$$m = 2(1) = 2$$

To calculate the value b , we have the point $(1, 2)$, so we can input it to equation 5.11 and compute that fellow:

$$z = 2x + b$$

This means that the equation for the yellow line is:

$$z = 2x$$

Cool, so the only thing missing is to fix x and compute the partial derivatives, but this time with respect to y :

$$\frac{\partial f}{\partial y} \quad (5.12)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

This time, we will leave that definition aside and use the derivation rules. So, if x is a constant, its derivative is 0. Now, for the term y^2 , we can use the power rule:

$$\frac{\partial f}{\partial y} = 2y$$

5.1.3 Just Follow My Lead: The Gradient.

Alright, now that we've determined both partial derivatives, it's worth noting that they retain the same properties as the standard derivatives we've previously studied. Therefore, they will guide us right to the entrance of that beehive, as we know its location coincides with a minimum point. Now, we have a tiny difference in our concept of derivatives. Before, we had an equation to represent this concept, but now, as we have more than one dimension, we have something called a gradient vector - Nice to meet you!

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \quad (5.13)$$

Moving forward, we'll rock linear algebra and calculus together. If you need a linear algebra refresher, refer to this series' first volume. Vectors carry both a magnitude and a direction, so if we define the gradient as such, it will point in a specific direction. To understand what lies at the end of this oriented arrow, we must consider what this little bastard is constituted of. Well, it is no surprise that the inputs to this vector are the partial derivatives of a given function, meaning that it will point in the direction of more significant change. So, if we want to comprehend how to locate critical points, we must nail our understanding of this alignment. As it happens, linear algebra gives us a cool concept for this endeavor: the dot product! A linear algebra tool that returns a scalar showing how much one vector aligns with another's direction.

Alright, so if we are going to explore the gradient's dot product with another vector, we need another vector, for example, $u =$

(u_1, u_2) . The result of this operation between these two vectors will measure how much the gradient aligns with the direction of u :

$$u \cdot v = \|u\| \|v\| \cos(\theta) \quad (5.14)$$

On 5.14 we have the formula for the dot product, where u and v are vectors and θ is the angle formed between them. In today's world, we cannot often define something in a way that favors us, but now this is about to change, at least for a few seconds. We want to express u so it has a norm equal to 1. It seems that it worked; they must be distracted, so let's move on:

$$\|u\| = 1 \quad (5.15)$$

The dot product between u and $\nabla f(x_0, y_0)$, where (x_0, y_0) is a generic point, is:

$$\nabla f(x_0, y_0) \cdot u = \|\nabla f(x_0, y_0)\| \|u\| \cos(\theta) \quad (5.16)$$

The term 5.16 is known as the *directional derivative*. It describes the rate at which a function changes at a specific point in a specified direction. Specifically, given a unit vector u and an arbitrary point (x_0, y_0) , the directional derivative tells us how fast the function $f(x, y)$ is changing at (x_0, y_0) in the direction of u . We use the notation $D_u f(x_0, y_0)$ for this, so 5.16 can be written as:

$$D_u f(x_0, y_0) = \|\nabla f(x_0, y_0)\| \|u\| \cos(\theta)$$

Now the question is, what values of directional derivative would be of interest to us? I would say both maximum and minimum values.

As the gradient represents the most pronounced rates of change in our function, if we can pinpoint the direction of u that renders these extreme values, it will point right where we need to go, the steep descent or ascent, where the critical points will be. Since the direction of u hinges on $\cos(\theta)$, our task will begin by understanding where this trigonometric guy has a maximum and minimum value:

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

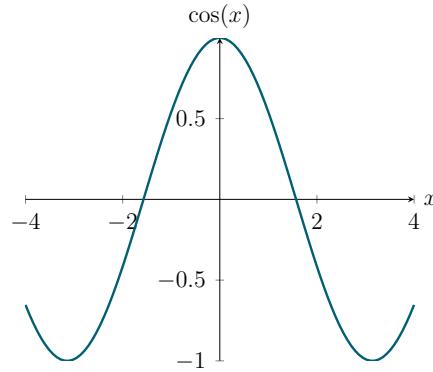


Figure 5.4: Some more curves, but this time, we have our friend $\cos(x)$.

We can see on graph 5.4 that $\cos(\theta)$ has a maximum when $\theta = 0$ and a minimum for values of $\theta = \pi$. Well, we can't just glance at something and move on. We must also verify it analytically:

$$\frac{d}{d\theta} \cos(\theta) = -\sin(\theta)$$

So, we have that $-\sin(\theta) = 0$ when:

$$\theta = 0, \pm\pi, \pm 2\pi, \pm 3\pi\dots$$

The reason for all those zeros comes from the phenomenon of the period that we covered before. The values of $\theta = 0$ and $\theta = \pi$ represent a maximum and a minimum respectively. However, let's verify this rather than taking my word for it. For that, we need to analyze the signal of the second derivative.

$$\frac{d^2}{d\theta^2} \cos(\theta) = -\cos(\theta)$$

Replacing those values of θ in the second derivative:

$$-\cos(0) = -1 \quad -\cos(\pi) = 1$$

So, the value of the second derivative at $\theta = 0$ is -1, meaning it is negative. That represents a \cap shape in the $\cos(\theta)$ function, so this guy is a maximum. For the other value of $\theta = -1$, we find 1. This is positive, which indicates a \cup , shape, so a minimum. Let's input this to 5.16, starting with $\theta = 0$:

$$D_u f(x_0, y_0) = \|\nabla f(x_0, y_0)\| \cdot 1 \cdot 1$$

When the angle θ between two vectors is 0, implying they point in the same direction, the function $\cos(\theta)$ reaches its maximum value of 1. Taking into account the fact that the dot product formula involves norms (which are always positive), the dot product is also maximized when $\cos(\theta)$ is at its peak. This ensures that $D_u f(x_0, y_0)$ achieves its highest value. Consequently, the gradient points in the direction of the steepest ascent, symbolizing the maximum positive rate of change:

$$D_u f(x_0, y_0) = \|\nabla f(x_0, y_0)\|$$

The opposite will happen when we select $\theta = \pi$, which is the same as saying that the vectors point in opposite directions. The π measures radians and the conversion to degrees is 180:

$$D_u f(x_0, y_0) = \|\nabla f(x_0, y_0)\| \cdot 1 \cdot (-1)$$

Simplifying:

$$D_u f(x_0, y_0) = -\|\nabla f(x_0, y_0)\|$$

Therefore, if $\cos(\theta) = -1$, the unit vector u points in the direction of steepest descent or maximum negative rate of change. That's because moving in the direction of u would decrease the value of the function as quickly as possible. This is not the direction of the hidden city of Atlantis, but it is still essential to us; it is where the minima are, the points we set ourselves to discover when applying algorithms such as the gradient descent algorithm.

5.1.4 GPS 2.0: Multidimensional Gradient Descent.

This may come as no surprise. Well, the name of the algorithm is gradient descent. Cool, so we have our GPS; not only that, but we can also generalize the gradient descent to n dimensions:

$$X_{n+1} = X_n - \eta \nabla f(X_n) \quad (5.17)$$

Where:

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

- $X = (x_1, x_2, \dots, x_n)$ represents a vector with n dimensions.
- η is the learning rate.
- $\nabla f(X_n)$ is the gradient of the function f with respect to the parameters X_n , our GPS that indicates the direction of steepest descent.

“What about an example, Mister Big Shot?” I agree. Let’s minimize the function $x^2 + y^2$ with this shiny new version of the gradient descent algorithm, but first, is that even convex? Well, I bring good news; if you recall, when we introduced the notion of convexity after the earlier tricky demonstration, we inferred that, if the second derivative is positive, we are golden. We have that \cup shape! So, we need to calculate:

$$\frac{\partial^2 f}{\partial x^2} \quad \text{and} \quad \frac{\partial^2 f}{\partial y^2}$$

For that, we derive the first-order partial derivatives with respect to the given variables:

$$\frac{\partial^2 f}{\partial x^2} = 2 \quad \text{and} \quad \frac{\partial^2 f}{\partial y^2} = 2$$

Contrary to the case of one dimension, things here are more complex than checking whether the second derivative is positive; by doing that, we will only consider what happens to the function $f(x, y)$ when we change one variable at a time and not when we do it simultaneously. The following partial derivatives can capture these phenomena:

$$\frac{\partial^2 f}{\partial x \partial y} \tag{5.18}$$

5.1.5 Bending Realities, Unveiling Curvatures: The Hessian.

This new piece of information brings a new player into town, a matrix called the Hessian. It is a square matrix of the second-

order partial derivatives of a scalar function. The Hessian has the following structure:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (5.19)$$

Where f is a function with n entries, $f(x_1, x_2, \dots, x_n)$. OK, but what do we need to do with that matrix to conclude whether a function is convex or not? Remember, this property ensures global critical points, which is extremely important for us. That means that we need to find a relationship between the Hessian and the definition of convexity to be able to prove that a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex for any two points x and y . There is nothing like starting with the definition of a concept when we are trying to study it, so a function is convex when:

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$$

Now, how can we get the Hessian involved? Let's look closer at the left hand side of the inequality, representing f at the point $(tx + (1 - t)y)$. We need to relate that to derivatives somehow, hopefully, the second one. I am not one to say "I told you so", but I did tell you that ugly equation was going to be helpful. I am talking about Taylor boy. So, we need to scale Taylor's expansion for multivariate functions. Let's remind ourselves of the equation for one dimension:

$$f(x) \approx f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \dots + \frac{f^{(n)}(a)}{n!}(x - a)^n$$

For the expansion of a multivariate function, rather than a point, we have a vector, and instead of the derivatives at those points, we will have the dot products between tensors and vectors. A tensor is a mathematical object that generalizes several algebraic concepts, such as scalars, vectors, and matrices, to higher dimensions. The Hessian is a second-order tensor in this particular situation, so with

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

all of this, we can derive the Taylor series expansion for more than one dimension:

$$\begin{aligned} f(x) \approx & f(a) + \nabla f(a)^T(x - a) \\ & + \frac{1}{2}(x - a)^T H(a)(x - a) + \dots \\ & + \frac{1}{n!}(x - a)^T T(a)(x - a)^n \end{aligned} \quad (5.20)$$

The formula 5.20 outlines the Taylor series expansion of the function f around point a , incorporating gradients and higher-order derivatives. It is essential to ensure that the dimensions align correctly when dealing with dot products between vectors and matrices; this is why the transposition of vectors, denoted by the superscript T , is used to facilitate valid dot product operations. To delve deeper, let's consider the second-order Taylor expansion of f around the point x , directed towards $(y - x)$:

$$f(x + t(y - x)) = f(x) + t(y - x)^T \nabla f(x) + 0.5t^2(y - x)^T H(x)(y - x)$$

These two expressions convey the same mathematical concept, but are framed differently. The second expression is a more detailed case of the first one, where $a = x$ and the direction of expansion is $(y - x)$, not $(x - a)$. Moreover, it introduces a parameter t to indicate the scale of the step towards y . This parameter is integrated into the vectors in the first expression, allowing it to describe a more general case. When $t = 1$, the second expression directly aligns with the terms in the first expression, thus demonstrating their equivalence and showcasing the adaptability and consistency of the Taylor series in approximating functions from various angles and positions.

Now we also know that, If f is convex, then by the definition of convexity, we have:

$$f(x) + t(y - x)^T \nabla f(x) + 0.5t^2(y - x)^T H(x)(y - x) \leq t f(y) + (1 - t) f(x)$$

Rearranging terms and dividing by t yields:

$$(y - x)^T \nabla f(x) + 0.5t(y - x)^T H(x)(y - x) \leq f(y) - f(x) \quad (5.21)$$

In the inequality 5.21 we have a linear term:

$$(y - x)^T \nabla f(x)$$

Let's focus on the quadratic component of the Taylor expansion given by:

$$0.5t(y - x)^T H(x)(y - x)$$

To ensure the convexity of the function f , we need to scrutinize the properties encapsulated in this term. Equation 5.21 stipulates criteria for verifying convexity. A crucial aspect of this verification process is analyzing the quadratic term associated with the Hessian matrix, denoted by H .

The Hessian matrix is central to our discussion as it contains second partial derivatives that convey vital information about the curvature of f . To establish convexity, we look for a \cap or \cup shaped curvature in the function graph. This necessitates that for all values of x and y , the second-order term in the Taylor series satisfies the inequality:

$$(y - x)^T H(x)(y - x) \geq 0 \quad (5.22)$$

Meeting this inequality condition would signify that f exhibits convex characteristics, essentially implying that the graph of f remains below the secant line connecting any two points x and y . Hence, affirming this inequality is a robust testament to the convexity of f , edging us closer to demonstrating condition 5.21.

To corroborate the non-negative curvature between points x and y , we recognize that the $0.5t$ term merely scales the quadratic form without altering its essential properties. We now resort to a foundational principle in linear algebra, which is that a symmetric matrix A is positive semi-definite if, and only if, the quadratic form $v^T A v$ is non-negative for all non-zero vectors v . From this, we deduce that:

$$v^T A v \geq 0 \quad \forall v \neq 0 \Rightarrow A \text{ is a positive semi definite matrix}$$

So, if the Hessian is a positive semi-definite matrix, the inequality 5.22 is verified, and therefore f is convex. Now, how the hell do we know if a matrix is of that type? The requirements never stop, I know, but the more you know, the better you will be at this! There are several methods to verify if a symmetric matrix is positive semi-definite. We will explore one of them:

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

- **All eigenvalues are non-negative:** If the eigenvalues of H are all non-negative, then H is positive semi-definite. This is because a matrix's eigenvalues represent the matrix's scaling factors, and a positive semi-definite matrix shouldn't "reverse" the direction of any vector (which would correspond to a negative eigenvalue).

It has been a while since we did some calculations, so I suggest we understand if the function $f(x, y) = x^2 + y^2$ is convex. First, the Hessian! It happens that we have most of the work done for constructing this matrix:

$$\frac{\partial^2 f}{\partial x^2} = 2 \quad \frac{\partial^2 f}{\partial y^2} = 2$$

And:

$$\frac{\partial^2 f}{\partial x \partial y} = 0 \quad \frac{\partial^2 f}{\partial y \partial x} = 0$$

We have two variables x and y so our Hessian will be a 2×2 matrix with the following shape:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

Cool, so that becomes :

$$H = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Next, we need the eigenvectors, which can be calculated using the expression:

$$\det(H - \lambda I) = 0$$

In this particular case, we have it easy:

$$\begin{vmatrix} 2 - \lambda & 0 \\ 0 & 2 - \lambda \end{vmatrix} = 0$$

So:

$$(2 - \lambda)(2 - \lambda) = 0 \Rightarrow \lambda = 2$$

Let's go; we have a non-negative eigenvalue. Therefore, H is symmetric positive semi-definite; consequently, our function is convex. This is crucial because it ensures that, if we identify a minimum on $f(x, y)$, it will indeed be the global minimum – the entrance point to the beehive that we're after. Now, there's an important concept I should underscore before we start iterating with the gradient descent.

Recall the fundamental definition of an eigenvector: it's a vector that, upon being transformed, lands either on itself or some linear combination thereof. By applying this to the Hessian matrix, we determine directions (given by the eigenvectors) where the function's second-order behavior is characterized by simple scaling, with the eigenvalues quantifying the magnitude of this scaling. So, the eigenvectors and eigenvalues give you a coordinate system aligned with the function's curvature.

With a learning rate and an initial guess (x_0, y_0) , we have everything we need to start with the gradient descent:

$$\eta = 0.05 \quad \text{and} \quad (x_0, y_0) = (0.5, 0.6)$$

As a reminder our gradient is:

$$\nabla f(x, y) = (2x, 2y)$$

We are ready!

• **Iteration 1:**

$$(x_1, y_1) = (x_0, y_0) - \eta \nabla f(x_0, y_0)$$

$$(x_1, y_1) = (0.5, 0.6) - 0.05(1.0, 1.2)$$

$$(x_1, y_1) = (0.45, 0.54)$$

• **Iteration 2:**

$$(x_2, y_2) = (x_1, y_1) - \eta \nabla f(x_1, y_1)$$

$$(x_2, y_2) = (0.45, 0.54) - 0.05(0.9, 1.08)$$

$$(x_2, y_2) = (0.405, 0.486)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

We need stopping criteria for this algorithm, and they are very similar to the ones we studied before:

- **The number of iterations:** This is the most common stopping criterion; we have to set a number of iterations for the algorithm to perform.
- **Convergence threshold:** This is where we check if the difference between the values of the images of the function over consecutive iterations k and $k+1$ is less than a defined threshold ϵ :

$$|f(X_k) - f(X_{k-1})| < \epsilon$$

- **Zero gradient:** If the value of the gradient at the current point on iteration k is very close to zero, it indicates that we are in the realm of our intended point.

These two iterations demonstrate the gradient descent process in action. However, more iterations will be performed to further refine our current estimate:

k	X_k	$\nabla f(X_k)$	$f(X_k)$	$f(X_k) - f(X_{k-1})$
3	(0.365, 0.437)	(0.810, 0.972)	0.324	0.076
4	(0.328, 0.394)	(0.729, 0.875)	0.263	0.062
5	(0.295, 0.354)	(0.656, 0.787)	0.213	0.050
6	(0.266, 0.319)	(0.591, 0.709)	0.172	0.040
7	(0.239, 0.287)	(0.531, 0.638)	0.140	0.033
8	(0.215, 0.258)	(0.478, 0.574)	0.113	0.027
9	(0.194, 0.233)	(0.431, 0.517)	0.092	0.022
10	(0.174, 0.209)	(0.387, 0.465)	0.074	0.017
11	(0.157, 0.188)	(0.349, 0.418)	0.060	0.014
12	(0.141, 0.170)	(0.314, 0.377)	0.049	0.011
13	(0.127, 0.153)	(0.282, 0.339)	0.039	0.009

Table 5.1: Iterations, Gradients values and a measure of conversion.

Table 5.1 presents 13 iterations of the gradient descent. In this table, the letter k in the column headings represents the iteration number, while X denotes a point, in this particular case, $X = (x, y)$.

For $\nabla f(X_k)$ we have the value of the gradient at the point (x_k, y_k) ; the last column represents the difference between consecutive estimates. This column will serve as the basis for our stopping criterion. If we consider a value of $\epsilon = 0.01$ we can see that on iteration 13, we have got that level of convergence as:

$$f(X_{13}) - f(X_{12}) < 0.01$$

Given our estimation for the entrance at the point $(0.127, 0.153)$, we are reasonably close to the global minimum of $x^2 + y^2$ located at $(0,0)$. Selecting a smaller value for ϵ would bring our estimate even closer to the global minimum.

Reflecting on our journey so far, we've extended several core concepts to handle more complex scenarios: Partial derivatives have been expanded into gradients, guiding our path for optimization. By building upon the Taylor series to cater to multi-variable contexts, we were able to use the Hessian matrix, a structure with information on the curvature of the function, as its elements are the second derivatives, to help us ascertain convexity and verify the nature of the critical points we encountered. Finally, we scaled the gradient descent algorithm and can now apply it to any dimension.

5.1.6 I Thought We Were Past Lines: Linear Regression.

Recall that our dive into these mathematical concepts was driven by the need to optimize two parameters for a better fit of a line to data- As with the previous case, we got stuck with a cost function dependent on m and b instead of just m . Let's remind ourselves of what equation it was:

$$J(m, b) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 \quad (5.23)$$

Where:

$$y = mx + b \quad (5.24)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

The function J represents a convex cost function, and we wish to minimize it to have the best fit possible for the linear relationship represented by 5.24. For that, we will use the gradient descent to obtain the best estimates for m and b , a task that requires some partial derivatives:

$$\frac{\partial J}{\partial m} \quad \text{and} \quad \frac{\partial J}{\partial b}$$

Alright, let's proceed with some calculations:

$$\frac{\partial J}{\partial m} = \frac{-2}{N} \sum_{i=1}^N (y_i - (mx_i + b))x_i \quad (5.25)$$

For the b parameter, we have:

$$\frac{\partial J}{\partial b} = \frac{-2}{N} \sum_{i=1}^N (y_i - (mx_i + b)) \quad (5.26)$$

We can then define the gradient as :

$$\nabla J(m, b) = \frac{1}{N} \left(\sum_{i=1}^N -2x_i(y_i - (mx_i + b)), \sum_{i=1}^N -2(y_i - (mx_i + b)) \right) \quad (5.27)$$

For the update rule, we have the following:

$$(m_{n+1}, b_{n+1}) = (m_n, b_n) - \eta \nabla J(m_n, b_n) \quad (5.28)$$

It is also common and possible to update one value at a time; after all we are dealing with partial derivatives, so we are considering the individual rates of change:

$$m_{n+1} = m_n - \eta \frac{\partial J}{\partial m}$$

$$b_{n+1} = b_n - \eta \frac{\partial J}{\partial b}$$

I will go through the calculation for one iteration and then present a table showing the rest. We need to define a learning rate and an initial point, oh... and the data would indeed be helpful:

x	$\ln(y)$
1	0.693
2	1.386
3	1.792
4	2.079
5	2.991

Table 5.2: Just the x 's and the scaled y 's.

For initial guess and learning rate, let's choose:

$$\eta = 0.01, \quad m = 0.6, \quad b = 0.3$$

Before starting, let's calculate the MSE for our initial guess:

$$\begin{aligned} J(m, b) &= \frac{1}{5} \left[(0.693 - (0.6 \cdot 1 + 0.3))^2 + (1.386 - (0.6 \cdot 2 + 0.3))^2 \right. \\ &\quad + (1.792 - (0.6 \cdot 3 + 0.3))^2 + (2.079 - (0.6 \cdot 4 + 0.3))^2 \\ &\quad \left. + (2.991 - (0.6 \cdot 5 + 0.3))^2 \right] \\ J(m, b) &= 0.126 \end{aligned}$$

Next, by expanding the formulas 5.25 and 5.26 we will do some updates, first for m and then b :

$$\begin{aligned} \frac{\partial J}{\partial m} &= -\frac{2}{5} [(1 \cdot (0.693 - (0.6 \cdot 1 + 0.3))) \\ &\quad + (2 \cdot (1.386 - (0.6 \cdot 2 + 0.3))) \\ &\quad + (3 \cdot (1.792 - (0.6 \cdot 3 + 0.3))) \\ &\quad + (4 \cdot (2.079 - (0.6 \cdot 4 + 0.3))) \\ &\quad + (5 \cdot (2.991 - (0.6 \cdot 5 + 0.3)))] \\ \frac{\partial J}{\partial m} &= 0.578 \end{aligned}$$

So for the first iteration of m which we call of m_1 we have:

$$m_1 = 0.6 - 0.01 \cdot 0.578 = 0.594$$

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

Now for b_1 , starting with the partial derivative:

$$\begin{aligned}\frac{\partial J}{\partial b} = -\frac{2}{5} & [(0.693 - (0.6 \cdot 1 + 0.3)) + (1.386 - (0.6 \cdot 2 + 0.3)) \\ & + (1.792 - (0.6 \cdot 3 + 0.3)) + (2.079 - (0.6 \cdot 4 + 0.3)) \\ & + (2.991 - (0.6 \cdot 5 + 0.3))]\end{aligned}$$

$$\frac{\partial J}{\partial b} = 0.293$$

$$b_1 = 0.3 - 0.01 \cdot 0.293 = 0.297$$

After one iteration, this is what our model looks like:

$$y = 0.594x + 0.297$$

If we were to calculate the cost after one iteration, this is the result:

$$J(m, b) = 0.082$$

Nice! There are some improvements in the cost of our predictions; so we are moving in the right direction! Let's use a stopping criterion of the number of iterations, say 10:

n	m	b	Cost
2	0.562	0.289	0.056
3	0.549	0.285	0.041
4	0.540	0.282	0.032
5	0.533	0.280	0.027
6	0.527	0.278	0.024
7	0.523	0.277	0.022
8	0.520	0.276	0.021
9	0.517	0.275	0.021
10	0.515	0.274	0.020

Table 5.3: The iterations, the change in the line parameters, and the cost values.

Great, so now to assess if we made improvements on our model or not, let's create a table with some metrics to help us compute the cost of the model whose parameters are the ones in iteration 10:

$$y = 0.515x + 0.274$$

Now for the table to help us with the cost function J :

x	y	\hat{y}	$e^{\hat{y}}$	$(y - e^{\hat{y}})^2$
1	2	0.789	2.201	0.040
2	4	1.304	3.684	0.099
3	6	1.819	6.165	0.027
4	8	2.334	10.319	5.378
5	20	2.849	17.270	7.450

Table 5.4: It seems we did make improvements in the cost. Let's check with some further calculations.

To maintain consistency, as with previous errors, we'll calculate the MSE on the standard scale:

$$\text{MSE} = \frac{1}{5} (0.040 + 0.099 + 0.027 + 5.378 + 7.450) = 2.559$$

After 10 iterations, our model yielded a cost of 2.559. This represents a substantial improvement from our initial error measure of 20 (using the model $y = 2x$). It also outperforms the starting model of the form $y = mx + b$. Remember? The guy that gave us a cost of 4.537:

$$y = 0.5287x + 0.2019$$

This refinement was mainly due to our strategy adjustment to accommodate an outlier in the data. By employing a logarithmic transformation on the y -values, we could tailor a linear regression model to our dataset. In contrast to the book's first model, where we solely optimized for m , we utilized multi-variate gradient descent in our most recent approach.

This technique necessitated simultaneous optimization of two parameters, m and b , thus improving our model's alignment with the data and diminishing the cost J . The outcome is:

$$y = 0.515x + 0.274$$

Should we plot it?

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

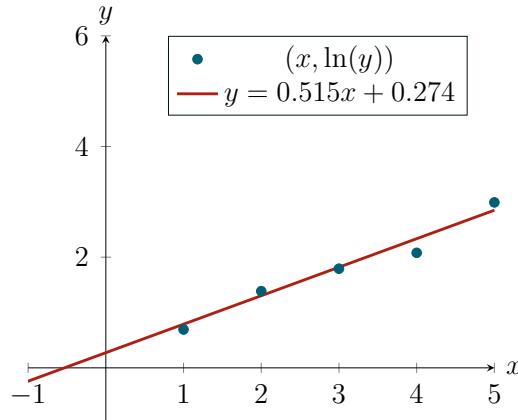


Figure 5.5: A better red fellow that fits the data smoothly.

Pretty good! If you wish to forecast with this model, we could do so by inputting the value of x we want. Let's experiment; we can estimate what nectar value would be collected with 6 steps, meaning that $x = 6$ and this is an observation we don't have:

$$\text{Point}_6 = (6, ?)$$

Let's input $x = 6$ in our newly trained model:

$$y = 0.515 \cdot 6 + 0.274$$

That Results in:

$$y = 3.364$$

Remember we are in the \ln scale, so to bring it back to the original measurement, we have to use e^x , the inverse of $\ln(x)$:

$$y_{\text{original}} = e^{3.364}$$

I am nervous...

$$y_{\text{original}} = 29.9$$

It makes sense, doesn't it? Yes, we are using a linear model so the estimated value of $x = 6$ must be higher than $x = 5$.

We started with a model of $y = 2x$ which revealed problems with the new observation $(5, 20)$. To overcome this, we not only created a new model with one more parameter, $y = mx + b$, but also scaled the y values with the \ln . Consequently, the cost that we got with the

original model of 20 came down to 2.559 which is a big improvement. But to truly showcase the power of scaling data with \ln we should fit another $y = mx + b$ to the data on the original scale and check the cost for it; let's go.

We have to perform another optimization task, why not use a different method? We initially introduced the Newton-Raphson method to approximate roots, but then we learned that the roots of the derivatives are the critical points of the function. Therefore, we can use this same technique to optimize functions. As a reminder, the update step for this algorithm is given by:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} \quad (5.29)$$

Right, that bad boy 5.29 is good when dealing with a function of one input and one output. However, that type of stuff belongs to the past, and now we are dealing with serious stuff. So, we need to adapt that equation 5.29 to a multivariate case. There is a division between the first and the second derivative, which, in the multivariate case, means that we need to divide a vector by a matrix. This operation does not exist, so here is where the book ends. Thank you very much for your support, and I wish you all the best.

I am joking, well, maybe I am half joking. In fact, there is no division among matrices, but we can use their inverse to get a similar concept. To understand why it would make sense to use such a technique, let's start by considering the world of scalars:

$$ax = b \quad (5.30)$$

In equation 5.30, both a and b are constants, so one way to solve it with respect to x is to divide both sides by a :

$$\frac{ax}{a} = \frac{b}{a}$$

Which is the same as:

$$1 \cdot x = \frac{b}{a} \quad (5.31)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

We now have an equation similar to 5.30 but with matrices instead of scalars:

$$Ax = B \quad (5.32)$$

Equation 5.32 is very similar to 5.30, but instead of scalars, A and B represent matrices. The strategy to isolate x was to have a 1 multiplied by the left side of the equation, as represented in 5.31. To get to that stage, we divided both sides by a , but things have changed as we are dealing with matrices, and matrix multiplication is not commutative, meaning $AB \neq BA$. Also, the "1" in linear algebra has a different representation. We call it the identity matrix and represent it by I ; one way to make it come to the party is by multiplying a matrix by its inverse:

$$A^{-1} \cdot Ax = A^{-1} \cdot B$$

When we multiply both sides by the inverse of A :

$$x = A^{-1}B$$

So the closest we have to a division between matrices is to multiply by the inverse; this means we need to perform this operation with the Hessian and the gradient. The first derivative of a function f at a point x is a vector that gives the direction in which the function increases the most. The second derivative, represented by the Hessian, provides information about how the first derivative changes as we move in space. It tells us about the curvature of the function in each direction.

5.1.7 Newton, Raphson, and Hessian Walk Into a Bar: No Joke, They Found a Root!

We want to make a step in the direction opposite to the gradient (since we want to decrease the function), and we want this step to be smaller if the function is highly curved in that direction. Well this can be achieved if we multiply the gradient by the inverse of the Hessian. If a particular eigenvalue of the Hessian is large, meaning the function is highly curved in the corresponding eigendirection,

the step we make in that direction will be smaller. Therefore, if we want to adapt equation 5.29 for a multi-dimensional case we have:

$$X_{n+1} = X_n - H^{-1} \cdot \nabla(f(X_n)) \quad (5.33)$$

As with the gradient descent case, we need the equations to update m and b . Looking to the terms in equation 5.33, we are just missing the Hessian, as we have the gradient ∇f . On we go with the calculations of some second derivatives:

$$\frac{\partial^2 J}{\partial m^2} = \frac{1}{N} \sum_{i=1}^N 2x_i^2 \quad (5.34)$$

$$\frac{\partial^2 J}{\partial m \partial j} = \frac{\partial^2 J}{\partial j \partial m} = \frac{1}{N} \sum_{i=1}^N 2 \cdot x_i \quad (5.35)$$

$$\frac{\partial^2 J}{\partial b^2} = \frac{1}{N} \sum_{i=1}^N 2 \quad (5.36)$$

So, for the Hessian matrix, we have the following:

$$H = \frac{1}{N} \cdot \begin{pmatrix} \sum_{i=1}^N 2x_i^2 & \sum_{i=1}^N 2x_i \\ \sum_{i=1}^N 2x_i & \sum_{i=1}^N 2 \end{pmatrix} \quad (5.37)$$

We are now ready for the revolution... one day, it will happen; one must be optimistic. But I mean, we are ready for the update equations for both m and b by the Newton-Raphson's method:

$$(m_{n+1}, b_{n+1}) = (m_n, b_n) - H^{-1}(m_n, b_n) \cdot \nabla J(m, b)(m_n, b_n) \quad (5.38)$$

As for the gradient descent case, the calculations for the first iteration will serve as a demonstration, but for the remaining ones, a computer will do the work, so, deep breath, let's go! For the initial guesses and the stopping criteria:

$$m_0 = 2,$$

$$b_0 = 2,$$

Stopping criteria = 10 iterations.

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

The Hessian matrix will be the same at every iteration of the algorithm, as it does not depend on the value of m or b :

$$H = \begin{pmatrix} 22 & 6 \\ 6 & 2 \end{pmatrix} \quad (5.39)$$

With this method, we can't do one parameter at a time, because we have to perform matrix-to-vector multiplications. Previously, we said that when we had the MSE for a cost function, we would be dealing with a convex shape; we can verify this by calculating the eigenvalues of the Hessian. If they are all non-negative, we can be confident that the minimum we find is global:

$$\lambda_1 = 23.662, \quad \lambda_2 = 0.338$$

There we have it; the eigenvalues are non-negative. Therefore, the Hessian is semi-definite and positive, which means we have a convex function. Cool, so the next step is to invert matrix 5.39:

$$H^{-1} = \begin{pmatrix} 0.25 & -0.75 \\ -0.75 & 2.75 \end{pmatrix} \quad (5.40)$$

To calculate the point (m_1, b_1) , we just need the gradient at (m_0, b_0) . Plugging in $(2, 2)$ into our gradient function 5.27:

$$\nabla J(2, 2) = \begin{pmatrix} -8 \\ 0 \end{pmatrix}$$

So for iteration 1 of the Newton-Raphson method, we have:

$$\begin{aligned} (m_1, b_1) &= (m_0, b_0) - H^{-1} \cdot \nabla J(2, 2) \\ \begin{pmatrix} m_1 \\ b_1 \end{pmatrix} &= \begin{pmatrix} 2 \\ 2 \end{pmatrix} - \begin{pmatrix} 0.25 & -0.75 \\ -0.75 & 2.75 \end{pmatrix} \cdot \begin{pmatrix} -8 \\ 0 \end{pmatrix} = \begin{pmatrix} 4 \\ -4 \end{pmatrix} \end{aligned}$$

Since the computer is about to do some work, why don't we optimize m and b with both methods instead of just one? This way, we can compare the gradient descent and the Newton-Raphson technique:

Iteration	m_{Newton}	b_{Newton}	J by Newton	m_{GD}	b_{GD}	
0	2.000	2.000	16.000	2.000	2.000	16.000
1	4.000	-4.000	8.000	2.080	2.000	15.430
2	4.000	-4.000	8.000	2.142	1.995	15.080
3	4.000	-4.000	8.000	2.191	1.987	14.857
4	4.000	-4.000	8.000	2.230	1.976	14.709
5	4.000	-4.000	8.000	2.261	1.962	14.604
6	4.000	-4.000	8.000	2.286	1.947	14.525
7	4.000	-4.000	8.000	2.306	1.931	14.460
8	4.000	-4.000	8.000	2.323	1.914	14.405
9	4.000	-4.000	8.000	2.337	1.897	14.355
10	4.000	-4.000	8.000	2.349	1.878	14.308

Table 5.5: Comparison of the Newton-Raphson method and gradient descent after convergence of GD

By iteration 1, Newton's method has converged; the cost does not change significantly after that. On the other hand, the gradient descent still shows signs that more iterations would make the technique improve on the solutions; if we look at the cost values, they are still decreasing. For now, let's see how well these two regression lines fit the data:

$$y_{GD} = 2.349x + 1.878 \quad (5.41)$$

$$y_{Newton} = 4x - 4 \quad (5.42)$$

Equation 5.42 represents the line defined by the parameters optimized by the Newton-Raphson method, whereas 5.41 depicts a line but optimized by the gradient descent:

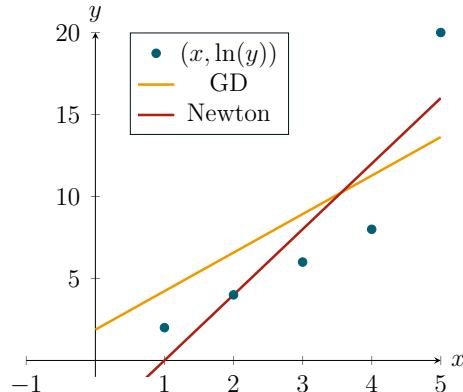


Figure 5.6: Newton's with the red trunks vs. Gradients descent wearing the yellow shorts.

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

As expected, and I say, this is due to the difference in the cost of both methods. We have a cost of 8 for the Newton-Raphson method versus one of 14.308 for gradient descent after 10 iterations; the yellow line shows that the former method does a better job at fitting all the points.

Continuing with gradient descent iterations, by iteration 1585, we observe the following values:

$$\text{Cost} = 8.01, \quad m = 3.992, \quad b = -3.971$$

Which equates to:

$$y = 3.992x - 3.971$$

Plotting it:

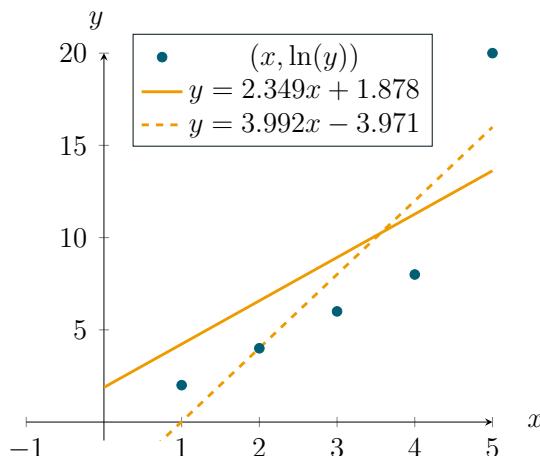


Figure 5.7: It will get there! We just need to give Newton a bit more time!

The dashed line depicts the optimized linear model configuration obtained using gradient descent. Interestingly, with enough iterations, gradient descent would converge to nearly the same parameters as those secured by Newton's method; all it requires is a bit of patience. When we initially compared these methods, Newton's approach was characterized by the encapsulation of more information, especially with the curvature insights offered by the Hessian. Consequently, it typically required fewer iterations to converge.

However, fewer iterations don't necessarily translate to faster convergence. The computational complexity increases when calcu-

lating the Hessian and its inverse. In the context of our small example, Newton's method converged quickly, but it's essential not to equate fewer iterations with the speed of execution. It's possible for one iteration of a particular method to be five times more time-consuming than its counterpart. If we have a massive Hessian, the matrix inversion step becomes notably resource-intensive, making each iteration of the method potentially lengthy.

Sure, but we did all this to compare a linear model fitted with a y value scaled by the \ln function, originally with one model but subsequently with two methods, trained on the original dataset, one on which we did nothing to the y . But for consistency, let's compare the MSE obtained by the two methods:

Case	J
y scaled by \ln	2.559
y not scale by \ln	8.000

Table 5.6: Comparison of the costs between scaled and unscaled data.

Yeah, baby, a 2.559 for cost with the \ln scale versus an 8.000 cost obtained by fitting a model to the variables without scaling. This showcases a practical use case for our friends, the logarithms, beyond just making pretty graphs.

Upon scaling our y value with the natural logarithm, our results notably improved. This led us to venture into the fascinating realm of multi-dimensional spaces. We established a compelling argument for the superiority of the model $y = mx + b$ over the simpler $y = mx$ when fitting our data. This journey deepened our understanding of functions that accept multiple inputs but yield a single output.

With determination, we grappled with sophisticated topics: from the nuances of first and second-order partial derivatives to the complexities of the Taylor series. We unraveled the intricacies of gradients, Hessians, convexity, gradient descent, and Newton's method. In addition, we learned to fit a line to our data using linear regression and experimented with two distinct optimization methods.

Building on this foundation, we are naturally advancing to the

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

next level of complexity in our mathematical exploration. Remember those functions we introduced at the beginning of this chapter, the ones with multiple inputs leading to multiple outputs? Well, what better method to delve into these with than neural networks? Yeah, baby... surprise! Although, now that I think about it, I might have given away that surprise by mentioning neural networks in both the table of contents and the book description. But hey, the excitement is still genuine. We're finally there!

5.1.8 A Mathematical Marriage: **Scalar Functions and Neural Networks.**

Flowers have developed various strategies to attract bees for pollination. They use bright colors, distinct patterns, and enticing fragrances to lure bees and ensure the transfer of pollen from one flower to another. I tried this strategy, but unfortunately, I just attracted boring conversations, and, in most cases, I went home alone. Sometimes, even worse, I didn't even get a fake phone number and the false hope of a date.

Anyhow, the honey's flavor can depend on the types of flowers the bees feed on, and a bee colony can produce 45 kg of honey a year. Say that we had access to the following table:

Bees	Flowers	Honey	Honey Class
500	1500	30	0
550	1600	35	0
700	1700	40	0
750	1800	50	1
800	2000	60	1
850	2200	70	1
950	2500	80	1

Table 5.7: Bees, Flowers and Honey measures.

We have 4 measurements on table 5.7:

1. **Bees** : This column indicates the number of bees in a specific location or bee colony.
2. **Flowers** : This is the count of flowers available in the area surrounding the bee colony's location. It is a major factor affecting honey production, as flowers are the source of nectar.
3. **Honey** : This is the quantity of Honey produced by the bee colony. It forms the target variable for a possible regression task. But we will use it to create a label.
4. **Honey Class** : This column has a binary representation of the volumes of honey produced in the colony. A value of 0 indicates that the production level was below 45 kg, and we represent this by 0. Conversely, we label a quantity of honey production greater than or equal to 45 kg as 1.

Our goal is to devise a mathematical framework that predicts the classification of honey production based on the number of bees and flowers. Specifically, we need equations, or a system of them, to take inputs (bees, flowers) and produce an output: 1 or 0, signifying high and low honey production respectively.

However, understanding this relationship is intricate. Hence, as good mathematicians, we can read more about the dynamics of the system for which we are trying to create models. It turns out that bees love dessert; Research indicates that bees are drawn to areas where the nectar has a higher sugar concentration; the sweeter the nectar, the more honey they tend to produce. Additionally, over-crowding can harm honey production.

A higher bee population might lead to competition for the same flowers, potentially decreasing honey yields. This information is valuable, and we shouldn't ignore it, but at the same time, we don't have values for these two metrics on our table. Perhaps we could infer them with a model that uses the following representation:

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

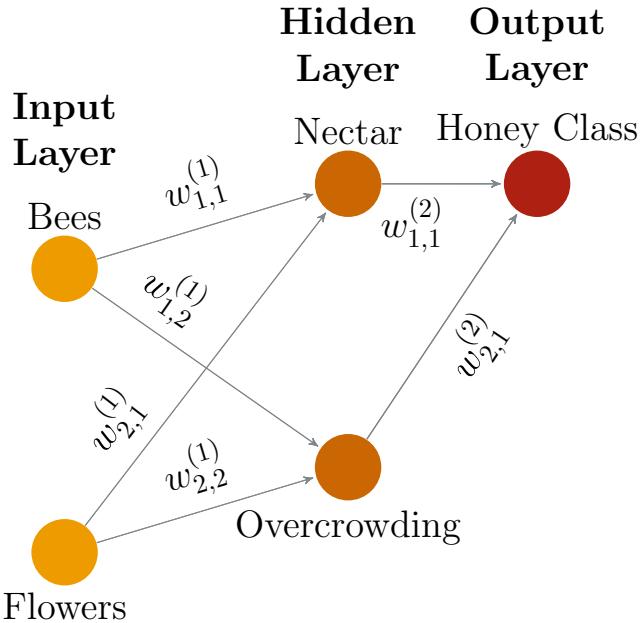


Figure 5.8: A possible neural network model.

We have a diagram! Now, there is some stuff we need to talk about before moving on. I do have to confess that I like to eat bananas with tuna and avocado. Oh, wrong forum... here we need to speak about that diagram, so let's do it.

Firstly, the color schemes: each color represents a different layer on the network. The yellow section represents the input layer, where we input the values of our observations. For the orange fellows, we have the factors we discovered when we went to read more about the problem at hand: the sugar in the nectar and the overcrowding factors. These are in a part of the model called the hidden layer; they represent latent variables that are hidden relationships in the data.

The output layer, represented by our red node, is where the model produces its final results, predicting either a 0 or a 1. Naturally, a question surfaces: why use only one node for two potential outcomes? The reason behind this is simplicity and efficiency. Although we could allocate two neurons to each label on the output layer, our current structure, with its table of data and weights, relies on scalar values, making it inherently linear. Incorporating curves or complex boundaries isn't feasible with such a linear setup. We'll

infuse nonlinearity after each layer to add complexity by steering the values through nonlinear functions. This approach amplifies our model’s capabilities beyond being a mere (albeit big) regression model. Particularly after the hidden layer, one of these functions will transform the final value from the network into a probability range between 0 and 1. With this probability for label 1 in hand, deducing the probability for label 0 becomes trivial. It is only necessary to subtract the former from one. Thus, having two output neurons becomes redundant, and a single neuron suffices to capture the model’s predictions.

Zooming out, we find ourselves exploring a specific niche within machine learning: ‘supervised learning’. We must grasp its broader landscape to become fluent in this domain’s lexicon. We also have to familiarize ourselves with its algorithmic suite. Machine learning is a trifecta: supervised, unsupervised, and reinforcement learning. What distinguishes these is the dataset we are working with. For example, supervised learning, the type we are dealing with in this neural network model, is where models predict labels (classification) or real-valued outputs (regression). It mandates datasets with target variables, be it honey quantity (regression) or honey classification (classification). Such tasks can have single or multiple targets.

In contrast, unsupervised learning operates on datasets without explicit labels. It’s the framework behind clustering models and recommendation systems, where we can make sense of data without pre-defined categories.

Finally comes reinforcement learning, the name for a learning paradigm through experimentation. Think of it as a system’s quest to perfect decision-making. Exploring different actions and observing the results is really just a fancy way of saying we’re learning by trial and error. It’s like testing the waters in various situations to see what works and what doesn’t.

Having established this foundation, let’s now tackle the supervised classification task of honey production using a neural network. Speaking of networks, let’s start working on ours. Starting at the

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

network's entrance, our input layer consists of two nodes. These nodes represent our primary variables: the number of bees and flowers. The essence of a neural network is to manipulate these inputs in a way that will eventually lead us to our desired output, in this case, the classification of honey. Such manipulations are done via the network weights, which reflect each input value's contribution to the desired outcome. So, the trick now is to find a mathematical concept that can reflect the relationship between inputs, weights, and outputs. Fortunately, linear algebra provides us with the dot product:

$$v \cdot w = v_1 \cdot w_1 + v_2 \cdot w_2 + \dots + v_n \cdot w_n \quad (5.43)$$

Where v is a vector with some input values, and w is another vector with weights, equation 5.43 represents a linear combination that can accurately reflect the contribution of each entry of v . Cool, so this seems like a good idea, as it captures what we are trying to represent by the network. Before jumping into the calculation, let's go over some notation, and this is about to get heavy:

$$w_{1,1}^{(1)}$$

The indices of the weight, in this particular case 1,1, represent what neuron connections this weight refers to. This guy, $w_{1,1}^{(1)}$ connects the first neuron from the input layer to the first neuron of the hidden layer:

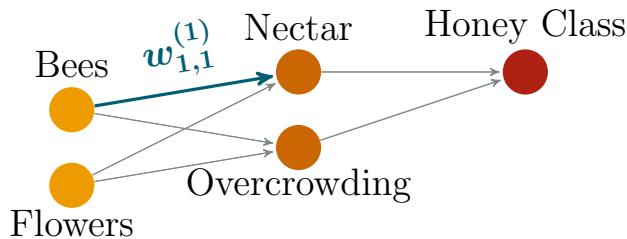


Figure 5.9: The weight of the connections of our network model.

Superscript 1 allows us to identify what layer connections are represented by a matrix or a vector. In this example, we will have two possible values for the superscript: the value of 1 when we wish to refer to the connections between the input layer and the hidden layer and the number 2 when we desire to get the relationships between the hidden layer and the output layers:

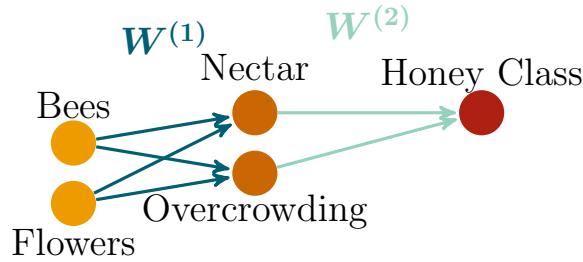


Figure 5.10: All of the weights can be stored as matrices.

In essence, the weights are numerical values that determine the strength and type (positive or negative) of the relationship between nodes across consecutive layers. In our particular case we can define them as $W^{(1)}$ and $W^{(2)}$:

$$W^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} \end{pmatrix}, \quad W^{(2)} = \begin{pmatrix} w_{1,1}^{(2)} \\ w_{2,1}^{(2)} \end{pmatrix}$$

- $W^{(1)}$: This set of weights, highlighted in dark blue in diagram 5.10, captures the relationships between the input layer (Bees and Flowers) and the hidden layer (Nectar and Overcrowding). Each connection between an input node and a hidden node has a specific weight that signifies how much the corresponding input influences that particular hidden node's value.
- $W^{(2)}$: This set of weights, highlighted in light blue, connects the hidden layer to the output layer. These weights dictate how the values in the hidden layer combine to produce the final output of "Honey Class".

Finally, we have to define the vectors for both the input layer and the hidden layer:

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad H^{(1)} = \begin{pmatrix} h_1^{(1)} \\ h_2^{(1)} \end{pmatrix}$$

Visually, we have:

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

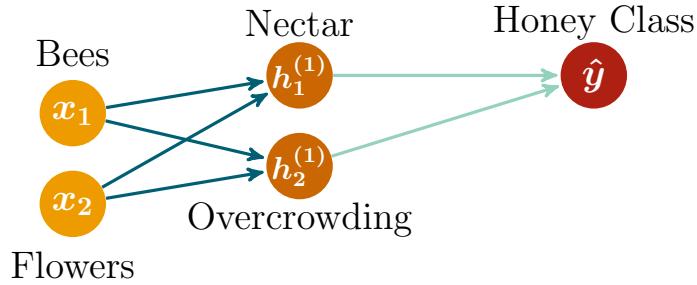


Figure 5.11: The hidden layers can't hide forever.

We also described the output layers as \hat{y} ; this variable will be the network's output. With this, we have defined our neural network, and, if we put all the notation together on the same diagram, this is what it looks like:

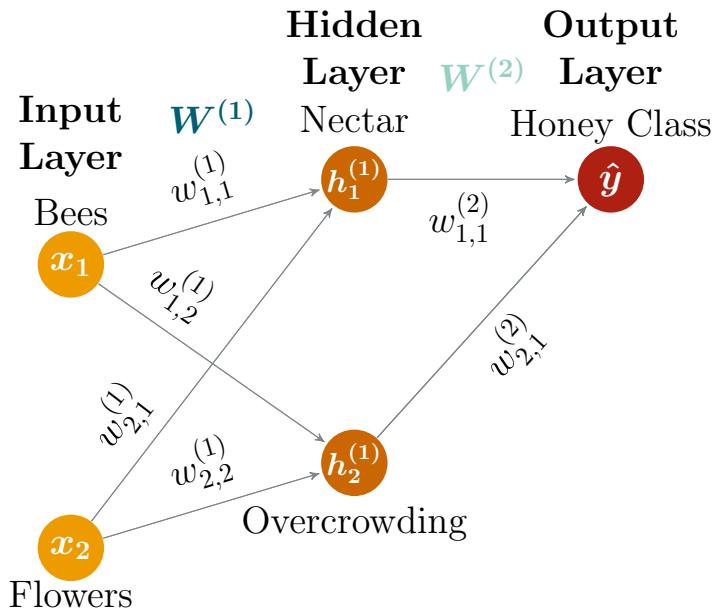


Figure 5.12: Everything on the same diagram.

Our neural network diagram is complete! Now it is time to stop with the drawings and go back to mathematics. We will move from the left to the right in a process called feed-forward, so let's start with the calculations of the value for the neurons in the hidden layer defined as Nectar and Overcrowding:

$$\text{Nectar} = w_{1,1}^{(1)} \cdot \text{Number of bees} + w_{2,1}^{(1)} \cdot \text{Number of flowers}$$

For the overcrowding, we have:

$$\text{Overcrowding} = w_{1,2}^{(1)} \cdot \text{Number of bees} + w_{2,2}^{(1)} \cdot \text{Number of flowers}$$

We can now replace the text with the notation previously defined:

$$h_1^{(1)} = w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 \quad (5.44)$$

$$h_2^{(1)} = w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2 \quad (5.45)$$

Equations 5.44 and 5.45 are how we calculate the entries that will form the hidden layer $H^{(1)}$. Because they represents dot products, we can also use matrix vector notation:

$$H^{(1)} = W^{(1)} \cdot X \quad (5.46)$$

Where:

$$H^{(1)} = \begin{bmatrix} h_1^{(1)} \\ h_2^{(1)} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \text{and} \quad W^{(1)} = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} \end{bmatrix}$$

The matrix $W^{(1)}$ represents the first set of weights as we defined it before. That player X is a vector with the entries that make our input layer, so in this particular case, $x_1 = \text{Number of bees}$ and $x_2 = \text{Number of flowers}$. There is another way to represent equations 5.44 and 5.45 in a singular equation, one that will be more relevant to us as, at some points, we will certainly be computing derivatives:

$$h_i^{(1)} = \sum_{j=1}^2 w_{j,i}^{(1)} \cdot x_j \quad (5.47)$$

Time to make a point of the situation; we defined two vectors X and $H^{(1)}$. They have magnitudes that respectively characterize both the input and the hidden layers. We also described a weight matrix $W^{(1)}$ that allows us to calculate the values of $h_1^{(1)}$ and $h_2^{(1)}$ given the vector X . Cool, so if we go from the hidden layer to the output layer, we have a completed set of equations for the whole network:

$$\text{Honey Class} = w_{1,1}^{(2)} \cdot \text{Nectar} + w_{2,1}^{(2)} \cdot \text{Overcrowding}$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

This is equivalent to:

$$\hat{y} = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)}$$

If we want to have an algebraic representation:

$$\hat{y} = (H^{(1)})^T \cdot W^{(2)}$$

Finally:

$$\hat{y} = \sum_{i=1}^2 w_{i,1}^{(2)} \cdot h_i^{(1)} \quad (5.48)$$

The next step is to do some calculations, but, before hopping into that, I want to point out that we can represent this whole neural network with a composition of functions. To do so, we can start by considering the output value \hat{y} that is calculated by equation 5.48. However, this equation depends on $h_i^{(1)}$ which is given by equation 5.47. Thus, we can represent the whole network thus:

$$\hat{y}(x) = \sum_{i=1}^M w_{i,1}^{(2)} \cdot \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j \right) \quad (5.49)$$

So, if we define a function $z_i(x)$ for each hidden neuron as:

$$z_i(x) = \sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j \quad \text{for } i = 1, 2, \dots, \text{number of hidden neurons} \quad (5.50)$$

We can then represent the entire set of outputs of the hidden layer as a vector $z(x)$. Furthermore, we can define a function $g(h^{(1)})$ as:

$$g(h^{(1)}) = \sum_{i=1}^M w_{i,1}^{(2)} \cdot h_i^{(1)} \quad (5.51)$$

Where M is the number of output neurons. Now equation 5.49 can be described as a composition of functions by:

$$\hat{y}(x) = g(z(x)) \quad (5.52)$$

Substituting in the full form:

$$\hat{y}(x) = \sum_{i=1}^M w_{i,1}^{(2)} \cdot z_i(x) = \sum_{i=1}^M w_{i,1}^{(2)} \cdot \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j \right) \quad (5.53)$$

That is indeed neat. I know; enough is enough, the calculations, right? We do have possible values for the entry vectors; what we don't have are the weights, but just as we did with all the methods we studied so far, we can assign them randomly, for example:

$w_{1,1}$	$w_{1,2}$	$w_{2,1}$	$w_{2,2}$
1.5	2.3	0.7	-1.2

Table 5.8: Our initial configurations for the weights that connect the input with the hidden layer.

So our matrix $M^{(1)}$ is:

$$M^{(1)} = \begin{pmatrix} 1.5 & 2.3 \\ 0.7 & -1.2 \end{pmatrix}$$

Alright, so we have our first set of weights $M^{(1)}$ that represents the impact of the number of bees and flowers on the collection of nectar and the overcrowding factor. Now for $M^{(2)}$, which represents the effect of the sugar in the nectar and the effect the overcrowding will have on the final nectar production, we have:

$w_{1,1}$	$w_{2,1}$
0.9	-0.5

Table 5.9: The second set of weights, the ones connecting the hidden and output layers.

Meaning that $M^{(2)}$:

$$M^{(2)} = \begin{pmatrix} 0.9 \\ -0.5 \end{pmatrix}$$

Initializing the weights randomly is just one of several ways to have a set of them to kick-start the training process of a neural network. We won't dive into the other methods as they rely heavily on probability distribution, a subject that will be extensively studied in volume 3 of this series.

Finally, some calculations are about to come! We have a table with data, weights, and a neural and network diagram, which means

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

we are ready to start experimenting with the data provided in table 5.7 and see what we get for the values of $h_1^{(1)}$, $h_2^{(1)}$ and \hat{y} . For that, let's start with the first line of our data:

$$X = \begin{pmatrix} 500 \\ 1500 \end{pmatrix}, \quad y_{\text{obs}=0}$$

Cool, so we have X as our entry vector, and the variable y_{obs} represents the observed class, which in this case is zero. Right so for $h_1^{(1)}$ or the sugar in the nectar value we have:

$$h_1^{(1)} = w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2$$

We have numbers for all those variables:

$$h_1^{(1)} = 1.5 \cdot 500 + 0.7 \cdot 1500$$

$$h_1^{(1)} = 1800$$

For $h_2^{(1)}$:

$$h_2^{(1)} = w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2$$

That comes to:

$$h_2^{(1)} = 2.3 \cdot 500 - 1.2 \cdot 1500$$

$$h_2^{(1)} = -650$$

Excellent, we have the hidden layer covered. Now we can move on to our final output value \hat{y} :

$$\hat{y} = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)}$$

Plugging in the values for the weights and the hidden layer inputs that we just calculated:

$$\hat{y} = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)}$$

$$\hat{y} = 0.9 \cdot 1800 - 0.5 \cdot (-650)$$

$$\hat{y} = 1945$$

Yes, we have got our first value for \hat{y} , so now we need to see how well this set of weights performed. Because our final goal is to have

a neural network from which, given an input vector, we can get a class of honey production, either a 0 or a 1. We can compare the output value \hat{y} , with the y_{obs} :

$$\hat{y} = 1945, \quad y_{\text{obs}} = 0$$

Yeah... but what can we do with this? The numbers are far apart, but this should not surprise us, as we haven't set any restrictions on the output \hat{y} . It can take any real value. However, we do need to address this issue, because at some point, as you might have suspected, we are going to try to optimize those weights we randomly selected, and that will imply having a cost function that will establish a relationship between \hat{y} and y_{obs} . If we leave the neural network to work with such a disparity between the predictions and observations, this shit will converge maybe... well, never.

Don't worry. As the people who developed these algorithms realized, we can solve this issue with an activation function and simultaneously introduce non-linearity to the previous layers. If we check all the equations presented with this model, we are still in a world of scalars represented by weights multiplied by vectors, so there are no curves, meaning linearity all the way. Well, the real world is often not linear, and by now, we have enough knowledge to deal with curves, so let's explore these activation functions:

5.1.8.1 Wake Up Call for Neurons: Curve Alert with Activation Functions.

Activation functions are a pivotal component of neural networks. They introduce nonlinearity by applying specific transformations to each neuron's weighted inputs. Without them, a neural network would merely be a big-ass linear regression model. It's essential to understand that real-world datasets often exhibit nonlinear relationships, hence the need for such transformations in our models. Furthermore, these activation functions are differentiable, which is crucial for optimization algorithms like the gradient descent one. As you may have anticipated, optimizing the weights is a central aspect

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

of training a neural network. Right, let's go over the most common activation functions:

- **Sigmoid:** The sigmoid function transforms its input into a value ranging between 0 and 1, ideal for output neurons in binary classification tasks like ours, where outputs represent probabilities:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This is how it looks:

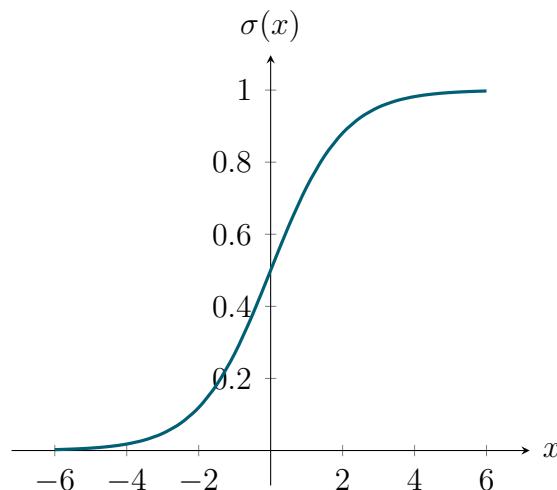


Figure 5.13: Sigmoid: The Latin word for an exponential that has taken a steroid (just a joke).

- **Hyperbolic Tangent (tanh):** This function maps its input to a value between -1 and 1 making it particularly suitable for processes and datasets where the data is zero-centered. Its mathematical representation is similar to the sigmoid, but it can model data that is centered around 0. For the equation, we have:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The graph:

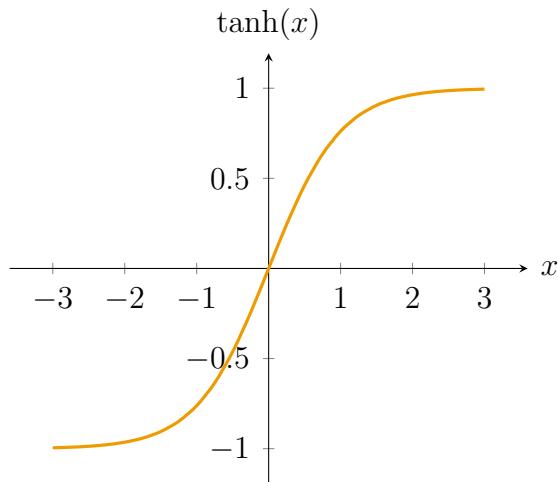


Figure 5.14: Hyperbolically speaking this is a tangent.

Unlike the sigmoid function, which has outputs in the range $[0, 1]$, \tanh outputs are in the range $[-1, 1]$. This means that the outputs are zero-centered, which can make the optimization process more manageable and the convergence faster.

- **Rectified Linear Unit (ReLU)** : This function outputs the input directly if it is positive; otherwise, it outputs 0. It's the most widely used activation function in convolutional neural networks (CNNs) and deep learning. This is because it trains faster. However, it can cause "dead neurons" that only output 0. This is a piece-wise function. See, I said that it would be handy to know about these functions earlier:

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Graph!

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

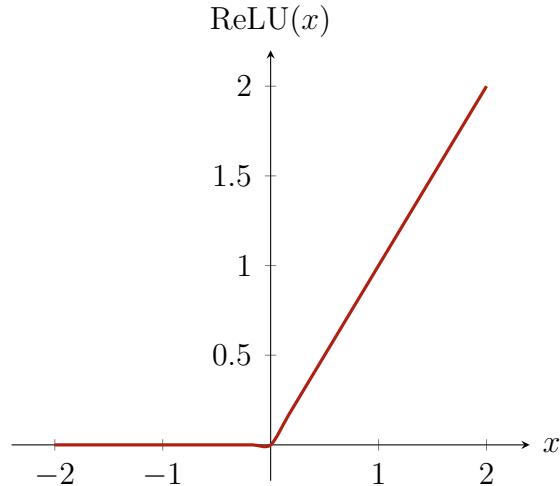


Figure 5.15: A strange case of nonlinearity - the ReLU.

- **Softmax:** This function is often used in the output layer of a network for multi-class classification problems. It maps each input to a probability between 0 and 1, and the outputs sum to 1. Because it is a function that operates over a vector and returns a vector, it's challenging to visualize it using 2-D plots like I have done for the other activation functions.

$$S_i(x) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

There you go, now we at least have some curves to make things more interesting. If ReLU is confusing you because of the straight line, don't feel that bad, as it initially confused me too. The nonlinearity of this fellow comes from $\max(0, x)$, which is an equivalent expression to the ReLU equation. This clipping of the negative values brings the nonlinear factor; we can see a bend around 0 if we take a closer look.

Now, when to use what? The descriptions of each of the functions give us some hints, but they are just a starting point; the best way would be to do some empirical testing. This means trying different ones and check what is happening to the model.

Okay, so we should apply this valuable knowledge to our model. Let's define $\phi_i(x)$ as a generic activation function. Because we have

two functions in which calculations are done, we need two activation buddies, one after the dot products between matrix $W^{(1)}$ and the input vector X and another one just after multiplying $H^{(1)}$ by the vector $W^{(2)}$:

$$\phi_1(x) = \tanh(x)$$

And because our output is a 0 or a 1, for $\phi_2(x)$, we can use:

$$\phi_2(x) = \sigma(x)$$

The sigmoid function provides us with a number between 0 and 1. When we have a value like this as the neural network's output, we can then define a threshold and consider that the network output is either a 0 or a 1. For example, if $\hat{y} \geq 0.5$, we have a 1; otherwise, it is 0. Okay, so let's complete our equations by adding the activation functions:

$$h_i^{(1)} = \phi_1 \left(\sum_{j=1}^2 w_{j,i}^{(1)} \cdot x_j \right) \quad (5.54)$$

Equation (5.54) gives us the values for the hidden layer function with an activation function:

$$H^{(1)} = \phi_1 \left(W^{(1)} \cdot X \right) \quad (5.55)$$

If we want to use matrix to vector notation, we could do so with equation (5.55). Now for the missing layer, the output one, we have the following equations:

$$\hat{y} = \phi_2 \left(\sum_{i=1}^2 w_{i,1}^{(2)} \cdot h_i^{(1)} \right) \quad (5.56)$$

For matrix vector notation, we have:

$$\hat{y} = \phi_2 \left((H^{(1)})^T \cdot W^{(2)} \right) \quad (5.57)$$

Finally, the full network comes as:

$$\hat{y} = \phi_2 \left(\sum_{i=1}^M w_{i,1}^{(2)} \cdot \phi_1 \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j \right) \right) \quad (5.58)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Okay, so let's adjust our calculations, starting with the h 's:

$$h_1^{(1)} = w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2$$

So now, instead of :

$$h_1^{(1)} = 1800$$

We have:

$$h_1^{(1)} = \phi_1(1800)$$

We need to input the value of 1800 in the $\tanh(x)$ equation:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Then:

$$\tanh(1800) = \frac{e^{1800} - e^{-1800}}{e^{1800} + e^{-1800}} = 1$$

For $h_2^{(1)}$:

$$h_2^{(1)} = \phi_1(-650) = -1$$

Proceeding to y :

$$\hat{y} = \phi_2 \left(w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} \right)$$

We can now plug in our data:

$$\hat{y} = \phi_2 (0.9 \cdot 1 - 0.5 \cdot (-1))$$

Time to use the sigmoid:

$$\hat{y} = \sigma(1.4)$$

And here we have it:

$$\hat{y} = \frac{1}{1 + e^{-1.4}} = 0.802$$

Much better, so if we define a threshold of 0.5, the value of y we obtained gives us a classification of 1, but our $y_{obs} = 0$. This is not ideal, it would be better if both values matched, but at least we have an output value \hat{y} between 0 and 1, so we are moving in the

right direction. The weights were selected randomly, so a miss classification like the one presented above is to be expected. But before moving on to the optimization, where we'll fine-tune our weights to better reflect the patterns in our data, we can still do at least one more thing. If we revert to when we calculated the neuron values of the hidden layer $H^{(1)}$, the quantities $h_1^{(1)}$ and $h_2^{(1)}$ we input an 1800 and a -650 into the activation function. The activation function we chose was the $\tanh(x)$ one. Well, this guy has two asymptotes: one at -1 and one at 1, and they are reached with some ease, meaning that x does not have to be very large for it to spit out 1 or -1. What can come from this rapid reach behavior is something called saturation of the activation function, which happens when all the inputs are rather large.

In our dataset, we find substantial numbers both for flowers and bees, which can lead to oversaturation of the activation function on the hidden layer. Before, we observed some similar behavior where one nasty number, an outlier observation, threw all our balance out and caused issues with our linear regression model. While our current situation doesn't present an outlier per se, our approach to addressing it is similar: we can use data scaling.

By scaling, we adjust the numerical range of our dataset while preserving the inherent relationships between data points. We previously utilized the natural logarithm (\ln) for this purpose. Though that was effective, I suggest we venture into new territory this time. There's a popular technique that leverages maximum and minimum values for scaling; it seems like a worthy prospect, so let's explore it.

5.1.8.2 Wait, Which Way? The Min-Max Scaling.

Given an input feature (column in your data), min-max scaling first identifies that feature's minimum and maximum values. Then, it subtracts the minimum value from each value in the components (thus shifting the whole distribution so that the minimum value is 0) and divides it by the range (min-max) of original values. This

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

results in a distribution where all the values are in the interval $[0, 1]$. I know what you're thinking. We need an equation:

$$x_i^* = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (5.59)$$

Where x_i represents the point of the feature set we are scaling, x_i^* is the scaled version of this exact point, and $\min(x)$, $\max(x)$ are the minimum and the maximum of the feature we are working with. So, we have two input features, the number of bees and the number of flowers; let's start by retrieving both the min and the max for these two:

$$\min_{\text{bees}} = 500, \quad \max_{\text{bees}} = 950$$

And:

$$\min_{\text{flowers}} = 1500, \quad \max_{\text{flowers}} = 2500$$

To demonstrate the calculations, I will manually do them for one input vector, the one with 500 bees and 1500 flowers. The remaining ones will also be scaled via the same technique but using a machine:

$$x_{1_{\text{bees}}}^* = \frac{500 - 500}{950 - 500} = 0$$

$$x_{1_{\text{flowers}}}^* = \frac{1500 - 1500}{2500 - 1500} = 0$$

The first scaled vector is $(0, 0)$, and the table below contains all the vectors scaled:

Bees	Scaled Bees	Flowers	Scaled Flowers	Honey Class
500	0.000	1500	0.000	0
550	0.111	1600	0.100	0
700	0.444	1700	0.200	0
750	0.556	1800	0.300	1
800	0.667	2000	0.500	1
850	0.778	2200	0.700	1
950	1.000	2500	1.000	1

Table 5.10: The scaled values of our dataset.

Alright, those scaled numbers seem a bit more friendly. We are now ready to see if the scaling technique improves the results we

were getting for the honey class. We needed a 0 but were getting a 1. So, instead of the entry vector (500,1500) we do the calculations for (0,0), the scaled version of it:

$$h_1^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 \right)$$

We can now enter the values for x_1 and x_2 :

$$h_1^{(1)} = \phi_1 (1.5 \cdot 0 + 0.7 \cdot 0)$$

$$h_1^{(1)} = \phi_1 (0) = 0$$

For $h_2^{(1)}$:

$$h_2^{(1)} = \phi_1 \left(w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2 \right)$$

That comes to:

$$h_2^{(1)} = \phi_1 (2.3 \cdot 0 - 1.2 \cdot 0)$$

$$h_2^{(1)} = \phi_1 (0) = 0$$

OK, this shows that both $h_1^{(1)}$ and $h_2^{(1)}$ are equal to 0, so let's proceed to calculate \hat{y} :

$$\hat{y} = \phi_2 \left(w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} \right)$$

We can now plug in our data:

$$\hat{y} = \phi_2 (0.9 \cdot 0 - 0.5 \cdot 0)$$

Time for the sigmoid:

$$\hat{y} = \sigma(0) = 0.5 = 1$$

Indeed, our model produced an output of 1. This is due to the nature of the sigmoid function, which returns 0.5 when its input is zero. However, an underlying issue remains: none of the neurons were activated, which means our model's output is solely reliant on the sigmoid function's behavior at zero, rather than any meaningful computation of the input. This highlights a limitation in our model. If the observed value is 1, or if we're tackling a regression

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

problem aiming to predict a continuous value, relying on the sigmoid function's default behavior becomes problematic. To overcome this issue, meet the bias!

Bias serves as an adjustable parameter that fine-tunes the output from the weighted sums. We enhance the model's flexibility by introducing a scalar to these sums before the activation functions process them. Doing so allows us to dodge scenarios like the one we discussed earlier, where the network is fed with 0's. Moreover, the bias ensures that the neurons can still be activated in situations where the weights are zero. To illustrate, when considering, $h_1^{(1)}$ the bias can be applied as follows:

$$h_1^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 + b_1^{(1)} \right)$$

If we have $h_2^{(1)}$:

$$h_2^{(1)} = \phi_1 \left(w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2 + b_2^{(1)} \right)$$

Here, $b_1^{(1)}$ and $b_2^{(1)}$ are real numbers. Their role can be likened to the b in linear equations such as $y = mx + b$. Just as b allows us to adjust the vertical positioning of a straight line, these bias terms enable us to shift the activation function either to the left or right. This added flexibility enhances the neuron's capacity to map inputs to outputs. With that understanding in place, let's turn our attention to the equation for the output layer:

$$\hat{y} = \phi_2 \left(w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b^{(2)} \right)$$

With this, we have fully defined our neural network feed forward pass, so let's recap. For the hidden layer equations we have:

$$h_1^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 + b_1^{(1)} \right)$$

$$h_2^{(1)} = \phi_1 \left(w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2 + b_2^{(1)} \right)$$

This is equivalent to:

$$h_i^{(1)} = \phi_1 \left(\sum_{j=1}^2 w_{j,i}^{(1)} \cdot x_j + b_i^{(1)} \right) \quad (5.60)$$

We can use matrix/vector notation, bringing back $W^{(1)}$:

$$W^{(1)} = \begin{pmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} \end{pmatrix}$$

Also, we need the input vector X , welcome back!

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Now we need a matrix to represent the bias:

$$B^{(1)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \end{pmatrix}$$

So, for $H^{(1)}$ we have:

$$H^{(1)} = \phi_1 \left(W^{(1)} \cdot X + B^{(1)} \right) \quad (5.61)$$

For the remaining layer, the output layer, we will have the following equations:

$$\hat{y} = \phi_2 \left(\sum_{i=1}^2 w_{i,1}^{(2)} \cdot h_i^{(1)} + b^{(2)} \right) \quad (5.62)$$

For matrix-vector notation, we have to do the same maneuver:

$$W^{(2)} = \begin{pmatrix} w_{1,1}^{(2)} \\ w_{2,1}^{(2)} \end{pmatrix}, \quad \text{and} \quad H^{(1)} = \begin{pmatrix} h_1^{(1)} \\ h_2^{(1)} \end{pmatrix}$$

Because we just have one output neuron, the bias is a scalar:

$$b^{(2)}$$

Finally the equation for \hat{y} :

$$\hat{y} = \phi_2 \left((H^{(1)})^T \cdot W^{(2)} + b^{(2)} \right) \quad (5.63)$$

And our composition of functions that will give us the full network:

$$\hat{y}(x) = \phi_2 \left(\sum_{i=1}^M w_{i,1}^{(2)} \cdot \phi_1 \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j + b_i^{(1)} \right) + b^{(2)} \right) \quad (5.64)$$

Where:

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

- M is the number of neurons in the hidden layer.
- N is the number of input features.
- $w_{j,i}^{(1)}$ are the weights from the input features to the i -th neuron in the hidden layer.
- $b_i^{(1)}$ is the bias for the i^{th} neuron in the hidden layer.
- $w_{i,1}^{(2)}$ are the weights from the i^{th} neuron in the hidden layer to the single output neuron.
- $b^{(2)}$ is the single bias term for the output neuron (no index needed since there is only one output neuron).
- ϕ_1 is the activation function for the hidden layer.
- ϕ_2 is the activation function for the output layer.

The diagram? Good point!

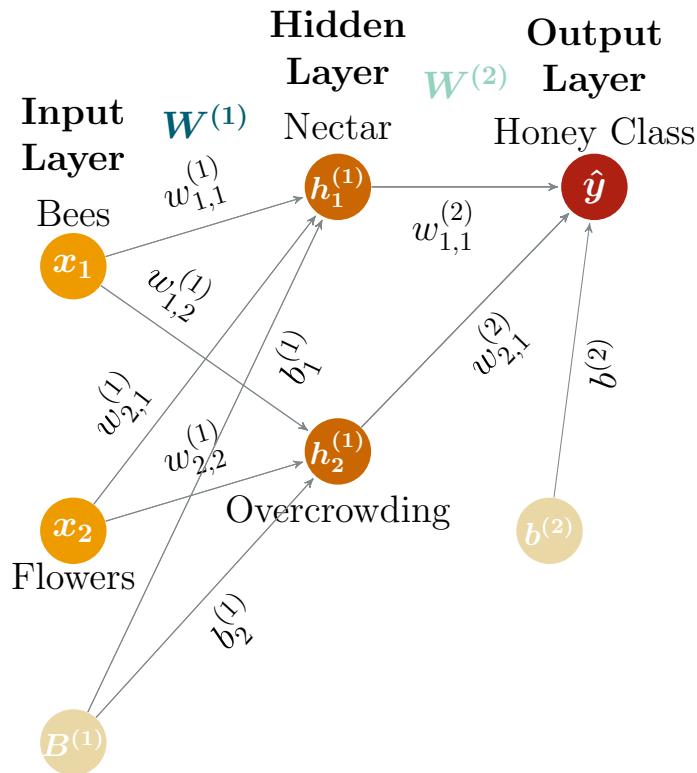


Figure 5.16: Everything on the same diagram.

Our neural network is fully defined; the only thing missing is selecting values for our bias. It will not come as a surprise that we randomly assign a value to these bad boys just like ... well... always, really. This random stuff seems to be a theme. Anyhow, here it goes:

$$b_1^{(1)} = 1, \quad b_2^{(1)} = 1, \quad b^{(2)} = 1$$

They are all one! Cool, now let's go back to the calculations and do a feed-forward pass for the same scaled entry vector $(0, 0)$:

$$h_1^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 + b_1^{(1)} \right)$$

If we replace the variables with numbers:

$$h_1^{(1)} = \phi_1 (1.5 \cdot 0 + 0.7 \cdot 0 + 1)$$

$$h_1^{(1)} = \phi_1 (1) = 0.761$$

For $h_2^{(1)}$:

$$h_2^{(1)} = \phi_1 \left(w_{1,2}^{(1)} \cdot x_1 + w_{2,2}^{(1)} \cdot x_2 + b_2^{(1)} \right)$$

It comes to:

$$h_2^{(1)} = \phi_1 (2.3 \cdot 0 - 1.2 \cdot 0 + 1)$$

$$h_2^{(1)} = \phi_1 (1) = 0.761$$

Now for the output layer:

$$\hat{y} = \phi_2 \left(w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b^{(2)} \right)$$

With numbers:

$$\hat{y} = \phi_2 (0.9 \cdot 0.761 - 0.5 \cdot 0.761 + 1)$$

$$\hat{y} = \phi_2 (1.304) = 0.787$$

Oh no, another $\hat{y} = 1$ when we have an $y_{obs} = 0$. Even after all those tricks? Weights, bias, and activation functions?

Well, it would have been too good to randomly select all of those parameters and nail something amazing at the first go; Yeah that

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

is right, we need to optimize those weights and biases. The truth is that all of the knowledge we acquired so far is enough to apply to these functions with multiple inputs and one output. Okay, so if we are talking about optimizing parameters, we need a cost function, and please give us a convex one! We used the Mean Squared Error (MSE) to quantify prediction errors in our previous tasks. But for our current classification challenge, the MSE doesn't fit the bill. The reason is that we are dealing with a classification and not a regression task. The softmax activation function we're employing gives outputs restricted between 0 and 1.

Given that our labels are binary (0 or 1), using the MSE can usher in misleadingly minor error values, even for big misclassifications. For instance, if our model predicts 0.04 when the actual label is 1, the cost would indicate a small error of 0.921. But in reality, that's a significant misclassification. Well, if the MSE is no bueno, we need something else, a concept called "cross-entropy"; this guy has the following function:

$$J = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (5.65)$$

Firstly let's unpack equation 5.65:

- n is the number of training examples.
- y_i is the actual output.
- \hat{y}_i is the predicted output.

The function we're considering, as shown in equation 5.65, is grounded in the concept of entropy, a fundamental notion in information theory. Let me make the case for why it is a better choice than the MSE.

5.1.8.3 The Price of Disorder: Cross Entropy as a Cost Function.

Entropy measures a dataset's uncertainty or randomness. A higher degree of randomness or unpredictability results in greater entropy values. In simpler terms, if it's challenging to make accurate predictions based on a dataset, that dataset probably exhibits high entropy. This idea, rooted in information theory, can be captured by the following equation:

$$H(p) = - \sum (p(x) \log(p(x))) \quad (5.66)$$

Here, $p(x)$ is the probability of event x occurring; for example, if we have a fair coin, the likelihood of heads is 0.5. The interesting part comes with the multiplication by $\log(p(x))$. The logarithm is a monotonically increasing function defined for positive values; it transforms multiplicative operations into additive ones. In the context of entropy, it correctly penalizes rare events with a high informational value.

When an event x is likely, its probability $p(x)$ is close to 1, and $\log(p(x))$ is close to 0. Conversely, when an event is unlikely, $p(x)$ is small, close to 0, and $\log(p(x))$ is a significant negative number. Multiplying $p(x)$ and $\log(p(x))$ together gives us a measure of the "surprise" or "information" of event x . Events that are very probable don't contribute much to the entropy because they aren't surprising (meaning they have low information content), while rare instances contribute a great deal to the entropy because they are surprising (meaning they have high information content).

Cool, so we now understand entropy, but our function definition calls for cross-entropy; this is when we combine two different likelihoods for the same outcome, which is the case with our neural network.

$$H(p, q) = - \sum (p(x) \log(q(x))) \quad (5.67)$$

We are dealing with a binary classification. Either we have a 0 or 1 as our neural network outcome; therefore, we can adapt the

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

cross-entropy function in equation 5.67 to tailor our needs better:

$$H(p, q) = - (p(0) \log(q(0)) + p(1) \log(q(1))) \quad (5.68)$$

Here, $p(x)$ represents the distribution of our observed labels, the y 's on table . Because $p(x)$ has those characteristics, $p(1)$ is the probability that an instance belongs to class 1, whereas $p(0)$ represents the likelihood of the input vector belonging to class 0. If $y = 1$, the instance is genuinely from class 1, and we can say that $p(1) = y$. Consequently, if $y = 0$, it means that the input vector is part of class 0, and, if $p(1) = y$, we can say $p(0) = 1 - y$ because we either have a 0 or a 1.

Now, for $q(x)$, we have represented the probability that our neural network outputs the value of the output layer after we pass it through the activation function. We defined this as \hat{y} , and we can do the same exercise as we did with y and $p(x)$. This means that, if $q(1) = \hat{y}$ then $q(0) = 1 - \hat{y}$. Let's plug all this new information into our equation 5.68:

$$J(y, \hat{y}) = - ((1 - y) \log(1 - \hat{y}) + y \log(\hat{y})) \quad (5.69)$$

Equation 5.69 is the representation for a single input vector, but we can not only extrapolate this for all of them, but also average the error out, as we previously did with the MSE:

$$J(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n ((1 - y_i) \log(1 - \hat{y}_i) + y_i \log(\hat{y}_i)) \quad (5.70)$$

For consistency of notation we will call the cost function J . Alright, let's recap what we have so far. Our neural network's cost function is showcased in equation 5.70. While the cross-entropy function is inherently convex, when applied within the framework of our neural network, the prediction \hat{y} is influenced by non-linear activation functions and various weights. Because of the non-linearity introduced by the activation fellow, this culminates in a non-convex optimization challenge. Don't worry; this is not all bad news as we get to learn some new algorithms.

We initiated our journey with a drawing which looked like it came from a coloring-by-numbers book. However, we soon realized that it was a cool neural network model whose aim was predicting honey production, using the input vectors of bee and flower counts as our guideposts. Subsequently, we introduced activation functions, which play a dual role: they infuse non-linearity into our computations and ensure our output comes out as a number between 0 and 1. Following that, we came across some lazy neurons. The sons of guns were dormant, so we called in our chief of staff, the bias: this is a scalar that put them back to work. These numbers serve as a nifty adjustment to our dot product calculations, refining them before they undergo processing via the activation functions.

Recalling our work with linear regression, we leaned on a cost function for guidance, specifically embracing cross-entropy as depicted in equation 5.70. The primary objective of this function is evaluation: it provides a metric with which to gauge the effectiveness of our optimization, something we needed because we randomly selected the first sets of weights and biases.

Cool, so now we need to understand how to simultaneously adjust all the weights and the biases and, at the same time, navigate a non-convex cost function. For that, we will start by calculating the cost for our network with our randomly generated set of weights and biases. We have the outcome for the input vector $(0, 0)$. Let's get the cost for this guy:

$$\hat{y} = 0.787 \quad y = 0$$

Therefore the cost for this instance is:

$$J(y, \hat{y}) = -((1 - 0) \cdot \log(1 - 0.787) + 0 \cdot \log(0.787))$$

$$J(y, \hat{y}) = 1.545$$

The calculations for all the other entries can be done the same way, so I will provide them in the table below:

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Bees	Flowers	\hat{y}	y	Cost
0.000	0.000	0.787	0	1.545
0.110	0.100	0.806	0	1.640
0.440	0.200	0.815	0	1.685
0.560	0.300	0.818	1	0.201
0.670	0.500	0.824	1	0.193
0.780	0.700	0.831	1	0.185
1.000	1.000	0.841	1	0.173

Table 5.11: The values of cost for the calculations done with the scaled data.

So the average total cost for our whole feed forward passage is:

$$J(y, \hat{y}) = 0.803$$

Previously, when dealing with linear regression, we optimized our cost function using gradient descent. This endeavor involved iteratively adjusting the parameters b and m based on the Mean Squared Error (MSE) gradient. Then, we computed the average error across all data points for each iteration, and updated our parameters accordingly. This approach was streamlined and effective because we were working with two parameters and a convex cost function.

However, those days are over, and now we are involved in a messier situation. We're navigating a more intricate model with an increased parameter set encapsulating multiple weights and biases. Furthermore, the cost function's landscape is no longer purely convex, a complexity introduced by the non-linear nature of our chosen activation functions. It is time to introduce something new: an individual with a name that makes it sound far more complicated than it actually is. I am talking about the “stochastic gradient descent”.

5.1.8.4 A Detour Through Randomness: **Stochastic Gradient Descent.**

Don't let the word “stochastic” throw you off, as it did for me. It's just a more nuanced version of the original gradient descent. We'll make updates individually or in smaller chunks instead of iterating over all the parameters after calculating the cost for all the

predictions. In this context, rather than performing a complete feed-forward pass, computing the average cross-entropy cost, and then determining the partial derivatives to update both the weights and the bias for the entire set of parameters, we'll make these updates after each individual or batched pass. A "batch" refers to small chunks of data on which we perform, in this case, optimizations; for example, if we have 1 million data entries, we can update the weights and bias every 1,000 entries. Here, with a batch size of 1,000, we compute the average cross-entropy error for each batch, followed by a single iteration of gradient descent.

The stochastic version of the gradient descent has a couple of tricks up its sleeve. For starters, it helps us to bypass local minima, the mathematician's nightmare. By randomly selecting individual data points or batches, we introduce randomness into the system. This variability can assist the algorithm in navigating beyond these troublesome local points. Additionally, there's the bonus of enhanced memory efficiency. Since our calculations are focused on gradients from either a single entry vector or a batch, the computational cost is lower, which leads to faster convergence.

Cool, so for this process, we start with a feed-forward pass to gauge the error. Following this, we must decipher how tweaking the parameters influences the cost function, just as we did with linear regression:

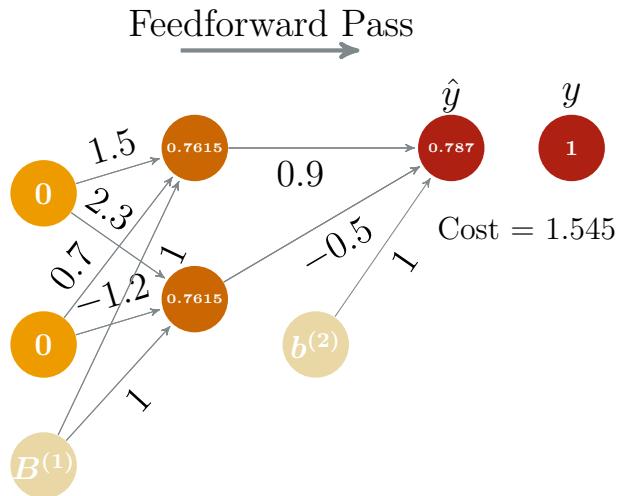


Figure 5.17: A single instance feed-forward pass.

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Right now, as we have the error quantified, we can walk backward and adjust the weights and the bias to reduce the cost. Because we have a cost function, we can stop with the randomness madness and use our new friend, the SGD (stochastic gradient descent), to minimize it. However, to effectively leverage SGD, it's essential to grasp how weight and bias changes affect the cost. This understanding will help us formulate a precise update rule.

Since we have many weights and biases, we will start with a single instance of those groups of numbers. Oh, we have a volunteer; thank you $w_{1,1}^{(2)}$:

$$\frac{\partial J}{\partial w_{1,1}^{(2)}} \quad (5.71)$$

In equation 5.71, we are trying to comprehend the effect on the cost when altering the weight $w_{1,1}^{(2)}$. It's crucial to understand that this weight, like others, does not directly influence the cost. Instead, its role is pivotal in determining the value of \hat{y} value, which subsequently impacts the cost function. Further complicating matters is the fact we threw a sigmoid activation function in there. Given this intricate composition of functions and the consequent cascade of events, we'll turn to a high-dimensional chain rule to escape that mess.

5.1.8.5 Upgrading Our Locksmith Skills: The Multivariate Chain Rule.

Alright, I will propose a plan of attack. First, we will use a generic function to see how we can "expand" our previously-learned formula for the chain rule for a single input to single output function, to functions of multiple inputs to a single output. Then, once this concept is solidified, we will proceed with a simple numerical example. To finish in glory we will see how this can be applied to what we are studying; a neural network, cool right? I assumed so, but moving on! Recall that for two differentiable functions $f(x)$ and $g(x)$, their composition is represented as:

$$f(g(x))$$

After studying the derivatives, we got to the result that the derivative of that composition is equal to the derivative of f analyzed at $g(x)$ multiplied by the derivative of $g(x)$, which, in mathematical notation is:

$$f'(g(x)) = f'(g(x))g'(x)$$

This can also have the following equation:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} \quad (5.72)$$

Where $y = f(u)$ and $u = g(x)$. But now instead of $f(x)$ and $g(x)$ we have a function $w(x, y)$ where both x and y depend on another variable z , so:

$$w = (x(z), y(z))$$

We can represent the correspondent derivatives thus:

$$\frac{dx}{dz} \quad \text{and} \quad \frac{dy}{dz}$$

So there is a dependency of w on x , while y that depends on z . This dynamic is captured in the following diagram:

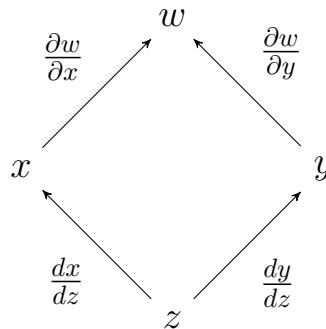


Figure 5.18: The diamond of knowledge! The chain rule dependencies.

Because they have one variable, we use d to represent the total derivative, not ∂ , which represents a partial derivative. Now, w is a different specimen, as it depends on x and y , meaning do we need some partial derivatives:

$$\frac{\partial w}{\partial x} \quad \text{and} \quad \frac{\partial w}{\partial y}$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Then, if we wish to calculate $\frac{dw}{dz}$ we have that:

$$\frac{dw}{dz} = \frac{dx}{dz} \cdot \frac{\partial w}{\partial x} + \frac{dy}{dz} \cdot \frac{\partial w}{\partial y} \quad (5.73)$$

Well, well, well, a new formula; we should explain what it means and how those terms that constitute the summation appeared. Our primary goal? To understand how w changes in relation to z . The variable z is the input for two functions: x and y , both of which provide information for our function w . We've illustrated these relationships in graph 5.18 to clarify this. The left hand side of this plot mirrors the left-hand component of the summation in equation 5.73. Essentially, this is the chain rule from equation 5.72, but applied to $w = f(x, y)$ where $x = g(z)$. Using similar logic for the right side of our equation and considering $w = f(x, y)$ where $y = h(z)$, applying equation 5.72 gets us the right component. Since both x and y contribute to our understanding of w , we sum these components, leading us to 5.73.

Are you feeling overwhelmed? Let's lighten things up with a numerical example:

$$w = f(x, y) = 3x^2 + 2y$$

Where:

$$x = g(z) = z^2, \quad \text{and} \quad y = h(z) = 2z$$

If we wish to calculate the derivative of w with respect to z , we can do so by using equation 5.73. Let's start with the partial derivatives:

$$\frac{\partial w}{\partial x} = 6x$$

$$\frac{\partial w}{\partial y} = 2$$

Now we need the derivatives of x and y with respect to z :

$$\frac{dx}{dz} = 2z$$

$$\frac{dy}{dz} = 2$$

Cool, so if we replace these values that we calculated in our formula for the chain rule it means that:

$$\frac{dw}{dz} = 6x \cdot 2z + 2 \cdot 2$$

$$\frac{dw}{dz} = 12xz + 4$$

We know that $x = z^2$ so:

$$\frac{dw}{dz} = 12z^3 + 4$$

And there we have it! This is a representation of how w changes with respect to z , taking into account the influence of x and y , which are functions of z . That completed the process of going through the chain rule definition with a generic function, and we followed it with a numerical example. Returning to our primary focus, the cost function $J(y, \hat{y})$ is defined as:

$$J(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n ((1 - y_i) \cdot \log(1 - \hat{y}_i) + y_i \cdot \log(\hat{y}_i))$$

With:

$$\hat{y} = \phi_2 \left(w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b^{(2)} \right) \quad (5.74)$$

So now, if we represent the cost function as $J(y, u)$ where $u = \hat{y}$, we can apply the chain rule to this fellow and calculate the partial derivative of the cost with respect to a weight:

$$\frac{\partial J}{\partial w_{1,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{1,1}^{(2)}} \quad (5.75)$$

Easy, right? Well, what about that ϕ_2 ? That is also a function, so we don't have $f(g(x))$. It is more like $f(g(h(x)))$, but don't worry, we can just apply the chain rule iteratively:

$$f'(g(h(x))) = f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x) \quad (5.76)$$

Which we can also write as:

$$\frac{df}{dh} = \frac{df}{dg} \cdot \frac{dg}{dx} \cdot \frac{dx}{dh}$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

Let's clarify the last part. The function, denoted as f , represents our cost function J . So, f' refers to the derivative with respect to its input. The term g isn't about being 'gangsta,' of course; it actually stands for the activation function ϕ , which is applied to the output layer. Finally, h represents the linear function, generating the scalar that then passes through the activation function:

$$z = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b^{(2)}$$

OK, so the first term of 5.76 corresponds to the derivative of the cost with respect to the output because $g(h(x))$ represents the weighted sum passed by the activation function, so:

$$f'(g(h(x))) \rightarrow \frac{\partial J}{\partial \hat{y}}$$

As $g(x)$ is the activation function, $h(x)$ is the weighted sum of the elements of the hidden layer with the weights, and $g(h(x))$ is the same as 5.74 therefore we can use the \hat{y} in our formula.

For the second term, $g'(h(x))$, we have the derivative of the activation function with respect to the weighted sum, but as the result of the activation function is the output, \hat{y} we have that:

$$g'(h(x)) \rightarrow \frac{\partial \hat{y}}{\partial z}$$

Finally, for the last term, $h'(x)$ is the derivative of the weighted sum with respect to the weight so:

$$h'(x) \rightarrow \frac{\partial z}{\partial w_{1,1}^{(2)}}$$

So we can write the change of cost with respect to the weight $w_{1,1}^{(2)}$ as:

$$\frac{\partial J}{\partial w_{1,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{1,1}^{(2)}} \quad (5.77)$$

If we calculate those three partial derivatives, we will be more than halfway done with what we need to do to get an equation to

update that weight with the gradient descent:

$$\frac{\partial J}{\partial \hat{y}} = - \left(\frac{y}{\hat{y}} - \frac{(1-y)}{1-\hat{y}} \right)$$

Because we are dealing with a single instance of the data, we are deducing how to update a weight after doing a forward pass on a single input vector (0,0). Our cost function does not have the summation; it is, in fact, represented by 5.69. I know it was way back up in the book, so I will put it here again as a reminder:

$$J(y, \hat{y}) = -((1-y) \cdot \log(1-\hat{y}) + y \cdot \log(\hat{y}))$$

Let's work it out from right to left; because y is a constant, the derivative of $y \cdot \log(\hat{y})$ is $\frac{y}{\hat{y}}$. This comes from the derivative of $\log(x)$ being $\frac{1}{x}$.

The same argument can be made for the other term. Now, for the derivative of the activation function with respect to the weighted sum, we have:

$$\frac{\partial \hat{y}}{\partial z} = \phi'_2(z)$$

Now we need the derivative of the sigmoid function:

$$\phi'_2(z) = -\frac{1}{(1+e^{-z})^2} \cdot (-e^{-z})$$

This expression simplifies to:

$$\phi'_2(z) = \frac{e^{-z}}{(1+e^{-z})^2}$$

Which, upon further simplification using the original sigmoid function $\phi_2(z)$, gives us the derivative:

$$\phi'_2(z) = \phi_2(z) \cdot (1 - \phi_2(z))$$

But we know that $\phi_2(z) = \hat{y}$:

$$\phi'_2(z) = \hat{y} \cdot (1 - \hat{y})$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

The last guy is easy. I think it is well deserved after all of the notation and work done so far:

$$\frac{\partial z}{\partial w_{1,1}^{(2)}} = h_1^{(1)}$$

In equation 5.74, we observe a nice weighted sum. When deriving this equation in terms of z with respect to the specific weight under consideration, the derivative is simply the element being multiplied by that weight.

With this understanding in hand, we can now construct our update equation for $w_{1,1}^{(2)}$:

$$w_{1,1new}^{(2)} = w_{1,1old}^{(2)} - \eta \cdot \frac{\partial J}{\partial w_{1,1}^{(2)}} \quad (5.78)$$

The value η is again our learning rate. Say that we have a value of 0.01. Right, there's not much else left to do other than computing that fellow:

$$w_{1,1old}^{(2)} = 0.9$$

For the partial derivative, we have:

$$\frac{\partial J}{\partial w_{1,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{1,1}^{(2)}}$$

Well, we know that:

$$\frac{\partial J}{\partial \hat{y}} = - \left(\frac{y}{\hat{y}} - \frac{(1-y)}{1-\hat{y}} \right)$$

Because $y = 0$ and $\hat{y} = 0.787$:

$$\frac{\partial J}{\partial \hat{y}} = \frac{1}{1-0.787} = 4.694$$

For the second guy in that nasty derivation:

$$\frac{\partial \hat{y}}{\partial z} = \hat{y} \cdot (1 - \hat{y}) = 0.787 \cdot (1 - 0.787) = 0.167$$

Finally:

$$\frac{\partial z}{\partial w_{1,1}^{(2)}} = h_1^{(1)} = 0.761$$

So for our new value of the weight $w_{1,1}^{(2)}$:

$$w_{1,1new}^{(2)} = 0.9 - (0.01 \cdot 4.694 \cdot 0.167 \cdot 0.761)$$

$$w_{1,1new}^{(2)} = 0.894$$

We grabbed the input vector (0,0) and passed it through the neural network. Then, we checked the error and used the stochastic gradient descent to come back and adjust one of the weights to improve our cost:

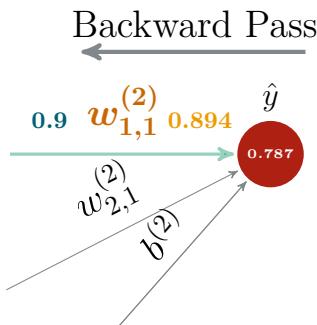


Figure 5.19: This is where we hit reverse on the neural network.

5.1.8.6 One Step Back to Give Two Steps Forward: Backpropagation.

Figure 5.19 depicts the process of the backward pass. For each input vector, we first evaluate the error. Based on this error, we propagate adjustments backward through the network, tuning the weights and biases to reduce the discrepancy in the network output, denoted as \hat{y} . This recursive parameter-tuning process is commonly known as backpropagation. You might have come across this term before – in the realm of neural networks, it's almost as famous as Elvis Presley is in the real world! While 'backpropagation' refers to the technique, the implementation method is through an optimization algorithm. In this context, we're employing stochastic gradient

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

descent, but there are many other algorithms out there, some of which we'll delve into later in this book. Our primary goal is to update every network parameter for each input vector, which calls for adjusting every weight and bias across the network.

Alright, a logical next step is to proceed layer by layer. Given that we already have the equation for the update of $w_{1,1}^{(2)}$, as given by equation 5.78, we should continue to focus on the output layer. For $w_{2,1}^{(2)}$, the formulation will be similar, as it's within the same layer. The crux of these equations lies in understanding the influence on J when we adjust a specific parameter. Let's delve into the specifics for $w_{2,1}^{(2)}$:

$$\frac{\partial J}{\partial w_{2,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{2,1}^{(2)}}$$

The chain rule is applied the same way as for $w_{1,1}^{(2)}$; this is because both these weights are on the same layer, so:

$$w_{2,1new}^{(2)} = w_{2,1old}^{(2)} - \eta \cdot \frac{\partial J}{\partial w_{2,1}^{(2)}} \quad (5.79)$$

The calculation is very similar to what we did with $w_{1,1}^{(2)}$, as the elements of the partial derivative of J with respect to $w_{2,1}^{(2)}$ are the same, therefore:

$$\begin{aligned} w_{2,1new}^{(2)} &= -0.5 - (0.01 \cdot 4.950 \cdot 0.161 \cdot 0.761) \\ w_{2,1new}^{(2)} &= -0.506 \end{aligned}$$

We are only missing the b girl; it grew up wanting to be as cool as a natural break-dancer, but sorry, that is not happening; it still requires calculations:

$$\frac{\partial J}{\partial b^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b^{(2)}} \quad (5.80)$$

The good news is that we are only missing the calculation of the change in equation z with respect to $b^{(2)}$:

$$z = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b^{(2)}$$

So:

$$\frac{\partial z}{\partial b^{(2)}} = 1$$

Now for the update rule:

$$b_{new}^{(2)} = b_{old}^{(2)} - \eta \cdot \frac{\partial J}{\partial b^{(2)}} \quad (5.81)$$

$$\begin{aligned} b_{new}^{(2)} &= 1 - (0.01 \cdot 4.694 \cdot 0.167 \cdot 1) \\ b_{new}^{(2)} &= 0.992 \end{aligned}$$

That completes the updates for the output layer:

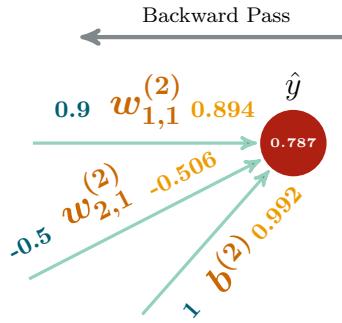


Figure 5.20: The backpropagation step of the layer connecting the hidden to the output neurons.

The value to the right (yellow) of the parameters is the new updated magnitude, whereas the one to the left (blue) is the original one. We can now create some more generic equations for an update of the output layer's weights and bias:

$$w_{i,1new}^{(2)} = w_{i,1old}^{(2)} - \eta \cdot \frac{\partial J}{\partial w_{i,1}^{(2)}} \quad (5.82)$$

Where $i = 1, 2$ and:

$$\frac{\partial J}{\partial w_{i,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{i,1}^{(2)}}$$

For the parameter b , equation 5.81 is enough, as we only have one set of biases.

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

So now we have to do the same procedure for the input layer, and that means we are required to calculate all the partial derivatives of the weights and parameters with respect to the cost J . The thing is, they are "further away" on the neural network:

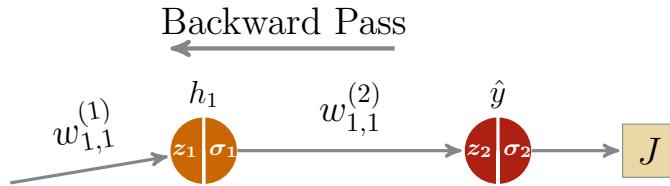


Figure 5.21: From the cost to the entry neuron. A birds-eye view of the backpropagation.

Figure 5.21 visualizes a segment of our neural network. To understand the update process for the weight $w_{1,1}^{(1)}$, we need to determine how much J (our cost function) changes when this specific weight is adjusted. With this comes the necessity to compute a partial derivative and apply the chain rule. Thus, we get:

$$\frac{\partial J}{\partial w_{1,1}^{(1)}} = \frac{\partial J}{\partial \phi_2} \cdot \frac{\partial \phi_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial \phi_1} \cdot \frac{\partial \phi_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{1,1}^{(1)}} \quad (5.83)$$

Is it a caterpillar?! Ah no, it is just an equation. On equation 5.83, the underlying concept mirrors our previous discussions for the output layer. Starting from the right-most term, we examine how the cost function J is affected by changes in the activation function ϕ_2 . Next, since ϕ_2 is a function of the weighted sum z_2 , we incorporate its derivative with respect to z_2 .

Moving further back, we note that z_2 is influenced by the preceding activation output ϕ_1 . Hence, we take the derivative of z_2 with respect to ϕ_1 . Following this, we consider how ϕ_1 is influenced by z_1 , and incorporate the derivative of ϕ_1 with respect to z_1 .

Finally, $w_{1,1}^{(1)}$ directly affects z_1 . Therefore, to complete the chain rule expression, we consider the derivative of z_1 with respect to $w_{1,1}^{(1)}$ to complete the chain rule expression.

Phew... I could use a glass of whis... I mean, water after all that! But here's the silver lining: Once we've wrapped your head around the backpropagation process for a simple case, we've essentially cracked the code for adjusting weights and biases in any neural network architecture, regardless of its complexity. Now, if we peek at equation 5.83, we can spot a way to make it look friendlier.

We're already aware that the output from ϕ_2 is \hat{y} and the output from ϕ_1 is $h_1^{(1)}$. So, let's rewrite it:

$$\frac{\partial J}{\partial w_{1,1}^{(1)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial h_1^{(1)}} \cdot \frac{\partial h_1^{(1)}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{1,1}^{(1)}}$$

We already know what some of those guys are, namely:

$$\begin{aligned}\frac{\partial J}{\partial \hat{y}} &= - \left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right) \\ \frac{\partial \hat{y}}{\partial z_2} &= \hat{y} \cdot (1 - \hat{y}) \\ \frac{\partial z_2}{\partial h_1^{(1)}} &= w_{1,1}^{(2)}\end{aligned}$$

Cool, so to proceed, we need z_1 and ϕ_1 . I will put them here as a reminder:

$$z_1 = w_{1,1}^{(1)} \cdot x_1 + w_{2,1}^{(1)} \cdot x_2 + b_1^{(1)}$$

And:

$$\phi_1 = \tanh(z_1) = \frac{e^{z_1} - e^{(-z_1)}}{e^{z_1} + e^{(-z_1)}}$$

Now, we are in good shape to calculate the remaining derivatives:

$$\begin{aligned}\frac{\partial h_1^{(1)}}{\partial z_1} &= 1 - \tanh^2(z_1) \\ \frac{\partial z_1}{\partial w_{1,1}^{(1)}} &= x_1\end{aligned}$$

The update rule is very similar to the one for the other weights and biases:

$$w_{1,1}^{(1)_{new}} = w_{1,1}^{(1)_{old}} - \eta \cdot \frac{\partial J}{\partial w_{1,1}^{(1)}}$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

I'll walk you through the calculations for this specific example. After that, we'll derive the update rule for the bias of this layer. Once we have that sorted out, we'll let the computer handle the remaining calculations and present the results here. So, let's dive in:

$$w_{1,1new}^{(1)} = 1.5 - (0.01 \cdot 4.694 \cdot 0.167 \cdot 0.9 \cdot 1 \cdot 0)$$

$$w_{1,1new}^{(1)} = 1.5$$

When the input is 0, this particular weight remains unchanged during this specific gradient descent iteration. This observation can be generalized; when we formulate the update rule for weights on the input layer, it takes the following form:

$$w_{i,jnew}^{(1)} = w_{i,jold}^{(1)} - \eta \cdot \frac{\partial J}{\partial w_{i,j}^{(1)}} \quad (5.84)$$

Where $i = 1, 2$, $j = 1, 2$, and:

$$\frac{\partial J}{\partial w_{i,j}^{(1)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial h_i^{(1)}} \cdot \frac{\partial h_i^{(1)}}{\partial z_{1,i}} \cdot \frac{\partial z_{1,i}}{\partial w_{i,j}^{(1)}}$$

We are just missing a way to understand how we can update the bias:

$$\frac{\partial J}{\partial b_i^{(1)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial h_i^{(1)}} \cdot \frac{\partial h_i^{(1)}}{\partial z_{1,i}} \cdot \frac{\partial z_{1,i}}{\partial b_i}$$

That bias derivative guy is easy to calculate:

$$\frac{\partial z_{1,i}}{\partial b_i} = 1$$

Finally! We have everything we need to train our neural network, so let's pass on all the entry vectors, and at each of these stages, we will update all the weights and biases. The results of this exercise are recorded in the table below:

x_1	x_2	$w_{1,1}^{(1)}$	$w_{1,2}^{(1)}$	$w_{2,1}^{(1)}$	$w_{2,2}^{(1)}$	$b_1^{(1)}$	$b_2^{(1)}$	$w_{1,1}^{(2)}$	$w_{2,1}^{(2)}$	$b^{(2)}$
0.000	0.000	1.500	2.300	0.700	-1.200	0.996	1.002	0.894	-0.506	0.992
0.110	0.100	1.500	2.300	0.700	-1.200	0.992	1.005	0.887	-0.512	0.984
0.440	0.200	1.498	2.299	0.701	-1.199	0.989	1.007	0.879	-0.518	0.976
0.560	0.300	1.499	2.299	0.701	-1.199	0.990	1.007	0.881	-0.517	0.978
0.670	0.500	1.499	2.300	0.701	-1.200	0.991	1.006	0.883	-0.516	0.980
0.780	0.700	1.500	2.300	0.700	-1.200	0.991	1.005	0.885	-0.514	0.982
1.000	1.000	1.500	2.301	0.699	-1.201	0.992	1.005	0.886	-0.514	0.983

Table 5.12: The last workout routine of the SGD ($\eta = 0.01$).

Table 5.12 showcases the variations in weights and bias after a forward pass for each entry vector. When we process the entire dataset through the network for backpropagation, this complete cycle is called one “epoch”. It’s worth noting that, for any given entry vector at index t , we utilize the weights and biases that have been updated based on the entry vector at index $t - 1$. For instance, during a feed-forward pass of the entry vector $(0.440, 0.200)$ — the third in our sequence — the weights and biases derived from the stochastic gradient descent process applied to the second entry vector will be used. The final set of parameters from the epoch are the ones with the light blue background in table 5.12, and they will be the ones we retain for subsequent operations. Hence:

$$M_1^{(1)} = \begin{pmatrix} 1.500 & 2.301 \\ 0.699 & -1.201 \end{pmatrix}$$

The matrix $M_1^{(1)}$ represents the set of weights for the input layer, and the subscript 1 refers to the first epoch. Similarly, the bias vector can be represented as:

$$B_1^{(1)} = \begin{pmatrix} 0.992 \\ 1.005 \end{pmatrix}$$

For the output layer set of weights, we have:

$$M_1^{(2)} = \begin{pmatrix} 0.886 \\ -0.514 \end{pmatrix}$$

Finally the bias is:

$$b^{(2)} = (0.983)$$

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

Let's see if these new sets of weights have improved the average error of our network. Previously, we got a value of 0.803 by making a prediction with the weights and biases we randomly selected:

x_1	x_2	\hat{y}	y	Cost
0.000	0.00	0.779	0	1.511
0.110	0.100	0.799	0	1.606
0.440	0.200	0.808	0	1.650
0.560	0.300	0.811	1	0.209
0.670	0.500	0.818	1	0.201
0.780	0.700	0.826	1	0.192
1.000	1.000	0.863	1	0.179

Table 5.13: Predictions and losses for each input sample.

The average cost is then:

$$J(y, \hat{y}) = 0.793$$

Well, it's not a vast improvement, but it does seem we are going in the right direction. To ensure that the algorithm is converging, we can perform 100 epochs and, after each, compute the average error and check its behavior. If it behaves poorly, we will lock it inside its room, and tell it that it is grounded!

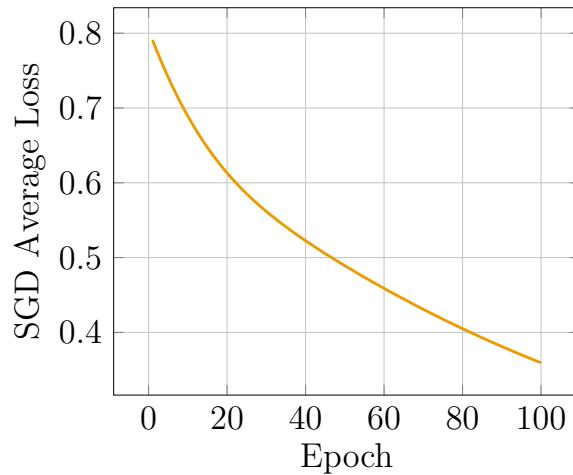


Figure 5.22: My sports betting account balance over time... It could be, but this one is for the average loss with the SGD.

Fortunately, the stochastic gradient descent appears to be converging, as indicated by the continual reduction in the error value with each iteration. We can't quantify how good this conversion

is, because we don't have a reference for comparison. Still, the downward trend in the error is a good sign as it suggests that the parameters are being optimized appropriately. Let's examine the outcomes when we reach epoch 100:

$$M_1^{(1)} = \begin{pmatrix} 1.500 & 2.301 \\ 0.699 & -1.201 \end{pmatrix} \quad M_{100}^{(1)} = \begin{pmatrix} 1.645 & 2.443 \\ 0.222 & -1.646 \end{pmatrix}$$

For the first set of biases:

$$B_{1_1}^{(1)} = \begin{pmatrix} 0.992 \\ 1.005 \end{pmatrix} \quad B_{1_{100}}^{(1)} = \begin{pmatrix} 0.683 \\ 1.054 \end{pmatrix}$$

For the second layer:

$$M_1^{(2)} = \begin{pmatrix} 0.886 \\ -0.514 \end{pmatrix} \quad M_{100}^{(2)} = \begin{pmatrix} 0.638 \\ -1.241 \end{pmatrix}$$

Finally, the bias is:

$$b_1^{(2)} = (0.983) \quad b_{100}^{(2)} = (0.432)$$

The subscripts 1 and 100 correspond to the epoch number we got the above parameters. It seems that, there's no cause for concern: the weights and biases did change, and the error is converging, but there is room for improvement and exploring other options. In fact, the selection of the optimization algorithm often depends more on the specific problem at hand than on a universal ranking of which method is better than another.

If we examine $M_1^{(1)}$ and $M_{100}^{(1)}$, we can see that some parameters have changed more than others. For instance, the weight $w_{1,2}^{(1)}$ was 2.301 and ended at 2.443 after 100 iterations, a 5% change. Conversely, the weight $w_{2,1}^{(1)}$ had 0.699 for value but ended at 0.222 after 100 iterations, a substantial 71% change. We can find more examples of such differences when comparing the parameters from iteration 1 and iteration 100.

These differences don't imply that one rate of change is inherently better than another. Instead, they suggest that different parameters might require different rates of change. As these rates are

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

primarily determined by the learning rate and the value of the partial derivative, using a fixed learning rate in SGD may not be optimal.

This brings us to the idea of personalized learning rates. If we can find a way to adjust the learning rate for each weight individually, we can make the parameters that require more significant adjustments change faster. In contrast, those that need smaller adjustments change more slowly.

We set up a scenario where the algorithm called RMSProp comes into play. RMSProp individually adapts the learning rate for each weight by calculating a moving average of the squared gradients, which is then used to adjust the learning rate for each parameter. That all sounds very fancy and complicated, but in reality, this is just some gradient descent with a little modification again.

5.1.8.7 Customized Routes to Minima: **RMSProp.**

RMSProp, an acronym for 'Root Mean Square Propagation', is an optimization algorithm that attributes a personalized learning rate to each parameter, instead of using a fixed learning rate for all. Different parameters may need to be updated at different rates in the context of neural networks. For instance, some parameters might be associated with rarely activated neurons and need a more significant learning rate. On the other hand, parameters related to frequently activated neurons might need a lower learning rate to avoid overshooting the optimal solution.

Since we are living through a period in which everything is about ourselves, perhaps in terms of temporal context, this one fits. So, let's translate this concept of individualization into mathematical terms. Let's embark on this expedition, building from the foundational equations of gradient descent:

$$x_{n+1} = x_n - \eta \nabla f(x_n) \quad (5.85)$$

From that equation, the only term we can change is the multiplication of the learning rate by the gradient of the function f at

the point x_n . OK, but now, how should we do this personalization?

We need to think about what might cause an overshoot; for example, if the gradient of parameters is frequently oversized, if we make it even more significant by increasing the learning rate, this will be no bueno. So, we need to decrease the learning rate for parameters for which the gradients are sequentially large. Conversely, if the gradients are often small, we need to crank them up to speed things up, meaning we can take some more significant steps, as the risk of going over the minimum is smaller.

Excellent, so we need to formulate a mechanism that encapsulates the historical gradient behavior and the current gradient magnitude for individual parameters. One effective approach to this is through a weighted sum:

$$v(n) = \beta v(n-1) + (1 - \beta) (\nabla f(x_n))^2 \quad (5.86)$$

In equation 5.86, two terms deserve our special attention: the parameters' gradient at the current iteration, and the same function v but analyzed in the previous iteration. This set-up ensures we're not just dwelling on the past, but actively responding to what's happening right now. Now, we do have to understand why we square the gradient on that same equation. This comes because our function v doesn't really care about which way the gradient is pointing; it's more concerned about how "strong" or "big" the gradient is. So, we square it to focus solely on the gradient's magnitude, allowing us to adjust the weights just right.

The term β is our balancing act between the past and present. With values between 0 and 1, β allows us to choose our priority: Do we give more relevance to what happened before, or do we prioritize what's happening right now?

If the value of $v(n)$ is large, we've consistently encountered high gradient values in the present and past iterations. In such cases, we should reduce the learning rate to avoid overshooting the optimal solution. Conversely, a small $v(n)$ suggests that our gradients have been relatively modest. We might benefit from increasing the learn-

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

ing rate to speed up convergence in these situations. Based on this reasoning, we can formulate the following equation:

$$x_{n+1} = x_n - \eta \frac{\nabla f(x_n)}{v(n)} \quad (5.87)$$

Keeping the original learning rate of η allows us to ensure that the learning rates for each parameter remain within a reasonable range. This helps avoid excessively large or small updates that could hinder the convergence process.

However, to ensure stability in our approach, we must consider a modification to equation 5.87. Because the learning rates grow proportionately with the squared gradients, this can lead to an extreme where learning rates are substantial, creating one of the same problems we are trying to solve: overshooting and, therefore, issues with conversions. We can deal with this by scaling it down with a square root, so:

$$x_{n+1} = x_n - \eta \frac{\nabla f(x_n)}{\sqrt{v(n)}} \quad (5.88)$$

Well, but what happens if $v(n)$ is tiny? Yes, that will be a problem also as, if $v(n)$ is zero or very close to it, we will get a division by zero. To overcome this we can defined a small fixed value ϵ that we will add to $\sqrt{v(n)}$:

$$x_{n+1} = x_n - \eta \frac{\nabla f(x_n)}{\sqrt{v(n)} + \epsilon} \quad (5.89)$$

There we have it, the root mean square propagation equation, the bad boy represented by equation 5.89. Well, now we need to see this thing working, so here is my proposal. We'll walk through the steps of using RMSProp to update a weight. As with our exploration of stochastic gradient descent, I'll run the network for 25 iterations on a computer to capture the nuances in parameter adjustments and errors. This approach sets the stage for directly comparing the two optimization techniques.

As before, the updates will be done stochastically, meaning we will update all the weights and biases for each entry vector. Let's get

this party started with the weight $w_{2,1}^{(2)}$ and the entry vector (0.00, 0.00). We need to calculate a few things:

$$\frac{\partial J}{\partial w_{2,1}^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_{2,1}^{(2)}}$$

Let's go from the left to right:

$$\frac{\partial J}{\partial \hat{y}} = - \left(\frac{y}{\hat{y}} - \frac{(1-y)}{1-\hat{y}} \right) = - \left(\frac{0}{0.787} - \frac{(1-0)}{1-0.787} \right) = 4.694$$

For the derivative of \hat{y} with respect to the linear combination z_2 we have:

$$\frac{\partial \hat{y}}{\partial z_2} = \hat{y} \cdot (1 - \hat{y}) = 0.787 \cdot (1 - 0.787) = 0.167$$

Lastly, we need the changes in z_2 when we change the weight $w_{2,1}^{(2)}$:

$$\frac{\partial z_2}{\partial w_{2,1}^{(2)}} = h_2^{(1)} = 0.761$$

Nice one, so:

$$\frac{\partial J}{\partial w_{2,1}^{(2)}} = 4.694 \cdot 0.167 \cdot 0.761 = 0.630$$

We can now calculate the value of $v(n)$ for iteration 1, more concretely $v(1)$. For that, we need to adapt our equations of the RMS prop, and, to define β . Hmmmmmmmm 0.9 (it is actually a very common choice)?!

$$v(n) = \beta \cdot v(n-1) + (1 - \beta) \cdot \left(\nabla J(w_{2,1}^{(2),n}) \right)^2$$

The function $v(n)$ is the weighted sum of the gradients so $v(0) = 0$:

$$v(1) = 0.9 \cdot 0 + 0.1 \cdot (0.630)^2 = 0.039$$

So the new iteration of $w_{21}^{(2)}$ comes with the following formula:

$$w_{2,1}^{(2)}_{new} = w_{2,1}^{(2)}_{old} - \eta \frac{\nabla J(w_{2,1}^{(2)}_{old})}{\sqrt{v(1)} + \epsilon}$$

5.1. Numerous Entries But Single Exit: **Scalar Functions.**

For η we will have a value of 0.01 and $\epsilon = 1 \times 10^{-8}$, so:

$$w_{2,1new}^{(2)} = -0.5 - 0.01 \cdot \frac{0.630}{\sqrt{0.039} + 1 \times 10^{-8}}$$

$$w_{2,1new}^{(2)} = -0.532$$

There we have it: simple stuff but of value. I would encourage you to see mathematics as being less complicated than we think it is, because this is often true; it just requires us to sit down and make a little effort. The idea that just only gifted people can understand it is a load of crap. With that in mind and our goal of fine-tuning parameters, let's proceed with 100 epochs of backpropagation, optimizing with RMSProp:

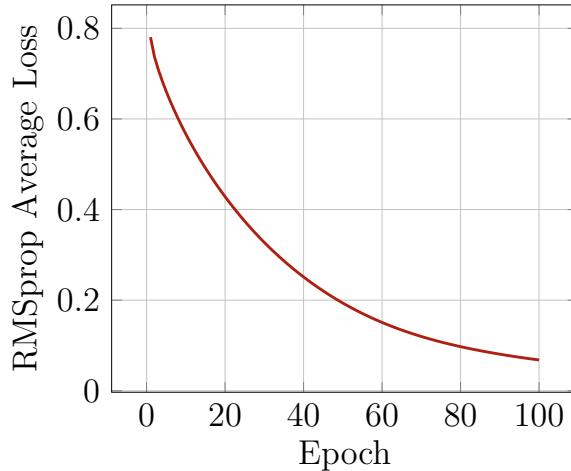


Figure 5.23: The average loss by epoch when we use the RMSprop.

Oh, la la! Check that conversion! Alright, we seem to be optimizing our weights and bias, so let's observe what we end up with after 100 epochs:

$$M_1^{(1)} = \begin{pmatrix} 1.485 & 2.305 \\ 0.705 & -1.213 \end{pmatrix} \quad M_{100}^{(1)} = \begin{pmatrix} 2.417 & 3.896 \\ -0.884 & -3.395 \end{pmatrix}$$

The bias sisters:

$$B_{1_1}^{(1)} = \begin{pmatrix} 0.948 \\ 1.049 \end{pmatrix} \quad B_{1_{100}}^{(1)} = \begin{pmatrix} -0.827 \\ 1.433 \end{pmatrix}$$

The last layer:

$$M_1^{(2)} = \begin{pmatrix} 0.847 \\ -0.555 \end{pmatrix} \quad M_{100}^{(2)} = \begin{pmatrix} 0.856 \\ -4.396 \end{pmatrix}$$

Lastly:

$$b_1^{(2)} = (0.947) \quad b_{100}^{(2)} = (0.008)$$

Notably, there's an increased variation in the parameter values. To further explore this notion of fluctuating learning rates, let's visualize the parameter values of two of them across 100 epochs. We'll compare optimizations done using stochastic gradient descent and RMSProp:

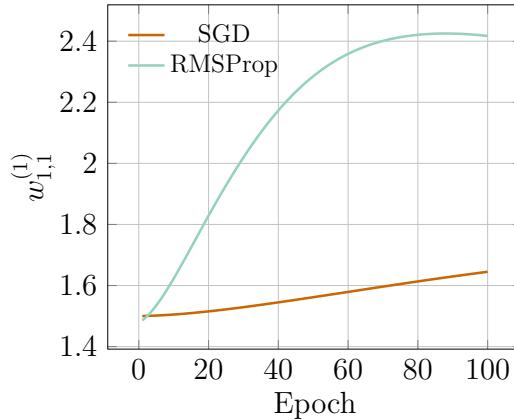


Figure 5.24: A comparison of the weight changes by epoch: RMSProp vs SGD

Graph 5.24 represents the change on the parameter $w_{1,1}^{(1)}$ over 100 epochs. The ugly blue line reflects the behavior of that individual when optimized with the RMSProp. In contrast, the orange line describes what is happening to the same parameters, but this time, when optimized by the stochastic gradient descent:

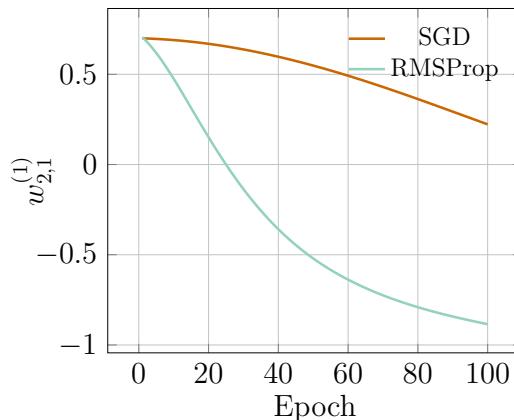


Figure 5.25: Another comparison of weight changes by epoch: RMSProp vs SGD

It's the same story in 5.25 as in 5.24 except for the parameter $w_{2,1}^{(1)}$. We can see that RMSProp can make faster and bigger changes

5.1. Numerous Entries But Single Exit: **Scalar Functions**.

in the neural network parameters while performing the backpropagation tasks; the question now is whether this is better. Well, we can answer that question in two ways: but only one is within this book's remit; we will compare the speed of conversion of the errors of these two optimization methods.

The second would be to measure the performance of the algorithm. Model performance is an essential and extensive topic that is outside of the content of the "before machine learning series"; we will focus predominantly on the mathematics that is necessary to understand these algorithms and provide full applicability for them.

Now, let's visualize both methodologies' cost evolution across epochs to gain further insights:

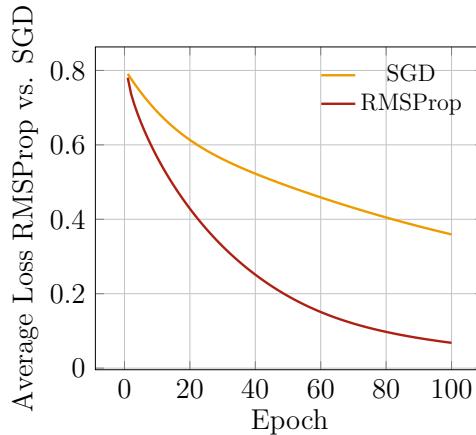


Figure 5.26: A comparison on the losses: SGD vs. RMSProp

In figure 5.26, the RMSProp is converging faster than the stochastic gradient descent, so in this particular situation, it seems like a better option with which to train our network. The reality is that each has its unique strengths and weaknesses.

The stochastic gradient descent is a straightforward and efficient optimization algorithm that approximates the actual gradient of the loss function by considering one training sample at a time. The main advantages of SGD are its simplicity, efficiency, and ease of implementation. It's also known to handle large-scale datasets well, since it only requires one batch at a time. However, SGD's primary disadvantage is that it tends to have issues converging, par-

ticularly in settings where the data or the objective function is not well-conditioned. The learning rate in SGD is the same for all parameters, which can result in sub-optimal convergence speed. Also, choosing an appropriate learning rate for SGD can be challenging.

On the other hand, RMSProp is an adaptive learning rate method that tries to resolve some of SGD's shortcomings. RMSProp modifies the learning rate for each weight individually, using the moving average of the squared gradients. This feature makes it more robust for the initialization of the parameters and the scale of the problem. Therefore, it often converges faster in practice and is less sensitive to the initial learning rate. However, RMSProp introduces additional complexity and hyper-parameters (like decay factor) that need to be tuned, potentially increasing the time it takes to optimize the model. Another disadvantage is that RMSProp might not be as effective as SGD in scenarios where the data is extensive and can't all fit into the memory at once because RMSProp needs to store gradient information for each parameter.

In conclusion, SGD offers simplicity and scalability, while RMSProp provides the advantage of adaptive learning rates, often leading to faster convergence. The choice between the two often depends on the specific application and its requirements.

Having explored these optimization algorithms, let's turn our attention to the versatility of neural networks in building diverse models. For example, we can have two outcomes instead of one, a scenario worth exploring, as we will require vector functions. This is a new concept that has high applicability in the field of machine learning, so let's go!

5.2 A Crowd Comes In, A Parade Goes Out: Vector Functions.

Previously, our neural network model focused on mapping multiple inputs, such as weights and biases, to a single output like a label. We're now transitioning to a more complex scenario where the network is configured to predict two distinct outcomes.

Sometimes, a single output isn't sufficient to capture the nuances of a scenario. Consider a situation where, instead of just distinguishing between high and low honey production, we also wanted to identify medium production. In this extended scenario, we'd label them as 0 for low production, 1 for medium production, and 2 for high production. Furthermore, we could have a case where the output is a vector whose entries are real numbers, meaning that we have a regression neural network with multiple outputs, a double whammy! We can study a neural network regression case and learn about vector functions.

I wanted a theme for this book, something that would make the heaviness of equations lighter; this was why I thought of bees. Honestly, I knew very little about them, just that they can fly, and some random facts a cop buddy of mine dropped on me. Flying means covering distance over time, meaning I could cover at least derivatives. That got me diving into bee biology, and let me tell you, those little guys are wild. Like, they dance to chat with each other! It's not quite the Macarena, but it's close. They're telling their buddies where the good stuff is by the angle and duration of their groove.

This dance takes the form of an infinity pattern (here it is again!). The central component of this pattern is the “waggle run” or “waggle phase” – a straight line in which the bee vigorously shakes, or ‘waggles’ its body. Following the waggle phase, the bee circles back to the starting point of the waggle run, tracing a path that forms half of the infinity symbol. The bee then performs another waggle run, followed by a return loop in the opposite direction, completing

the symbol. The waggle phase is performed at an angle relative to the vertical direction of the honeycomb, which encodes the direction of the food source relative to the position of the sun, while the duration of the waggle phase encodes the distance to the food source: the longer the waggle phase, the further away the food source is.

Okay, so let's create a function f to decipher this location of the food based on the information we get from the waggle dance. We stated that the intensity of the dance was linked to the amount of rum consumed by the bee; oh, sorry, that might apply to me and not the bee ...anyway, we know that the length and the intensity of the dance dictate the distance to the food source. If we define the distance by d , for length pick x and represent the intensity by y , we can have the following function for distance:

$$d(x, y) = x^4 + y^4 - 2x^2 - 3y^2 \quad (5.90)$$

The x^4 and y^4 terms represent the primary contribution to the distance. This part of the function is there because a more intense and prolonged waggle dance indicates a more distant food source. The term $-2x^2 - 3y^2$ represents the fact that there is a certain optimal intensity and duration that results in the minimum distance to the food source.

Now, using this bee dance as a springboard, consider other mathematical models like linear regression. Nothing stops us from evolving a linear equation, e.g., $y = mx + b$, to a non-linear model $y = mx^2 + b$. At its core, it's the same deal: we have two parameters, m and b , which we need to optimize.

The other component of the dance is the angle relative to the honeycomb's vertical direction, which will indicate the direction. We can capture this with:

$$\theta(x, y) = \arctan\left(\frac{x}{y}\right) \quad (5.91)$$

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

A highly intense waggle could indicate a food source in one direction, while a less intense waggle may signal a food source in another direction. However, the time elapsed since sunrise is critical because the sun's position changes throughout the day. The first phenomenon is captured by x , whereas y is a good metric for the second. Therefore, we can define θ with equation 5.91. So, the location of the food source is represented by:

$$f(d(x, y), \theta(x, y)) = [d(x, y), \theta(x, y)] \quad (5.92)$$

Or:

$$f(d(x, y), \theta(x, y)) = \left[x^4 + y^4 - 2x^2 - 3y^2, \arctan\left(\frac{x}{y}\right) \right]$$

We can produce some visuals but would have to do it individually for each function as we have yet to visualize 4 dimensions. For the functions that represent the distance to the food source, $d(x, y)$, we get:

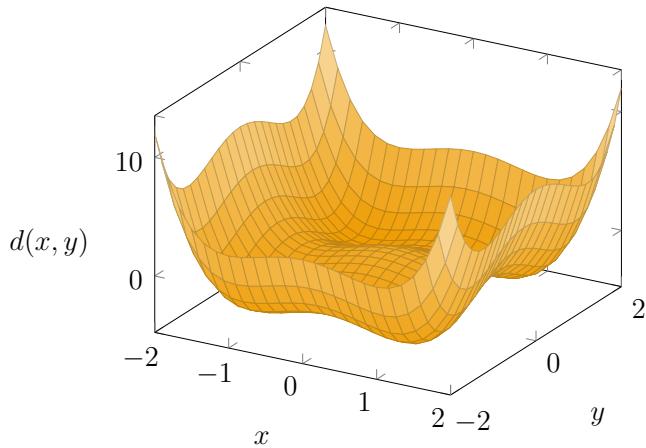


Figure 5.27: A curvy representation of the distance function $d(x, y)$.

Would you look at that shape? It's definitely not convex! Now, for $\theta(x, y)$ we have:

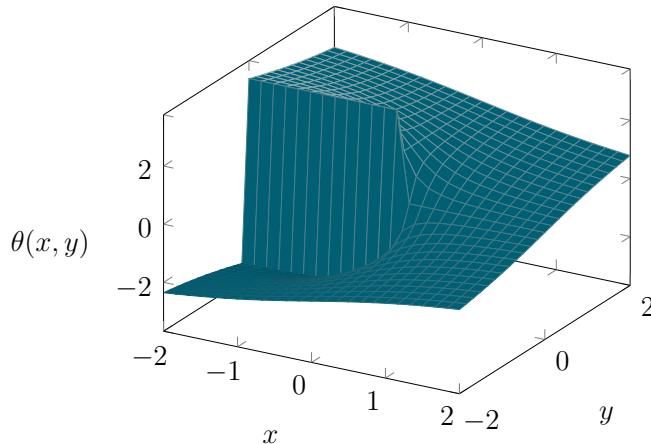


Figure 5.28: The angle to the honeycomb, $\theta(x, y)$.

Cool, so we have a function that returns two points, which is why these are also called vector functions. What comes next is more of the same. The points of interest to us are the critical ones, which means we need to find the points at which the gradient is 0. Our next step involves calculating the gradients for both functions, d and θ :

$$\nabla(f(d(x, y), \theta(x, y)) = (\nabla d(x, y), \nabla \theta(x, y)) \quad (5.93)$$

So, the gradient vector of the function f is a vector of gradients, but surely we can compute the entries of that vector. We just need a lot of partial derivatives! Well, not that many, actually, just four of them, starting with $d(x, y)$:

$$\begin{aligned} \frac{\partial d}{\partial x} &= 4x^3 - 4x \\ \frac{\partial d}{\partial y} &= 4y^3 - 6y \end{aligned}$$

And the gradient come as:

$$\nabla d(x, y) = (4x^3 - 4x, 4y^3 - 6y) \quad (5.94)$$

Now we need to do the same but for the function θ :

$$\frac{\partial \theta}{\partial x} \quad \text{and} \quad \frac{\partial \theta}{\partial y}$$

A bit of maneuvering with this one is necessary, but nothing that we haven't seen before:

$$\frac{\partial \theta}{\partial x} = \frac{\partial}{\partial x} \left[\arctan \left(\frac{x}{y} \right) \right]$$

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

We need our good buddy, the chain rule. As the outer function is $\arctan(u)$, its derivative is :

$$\frac{1}{1+u^2}$$

But $u = \frac{x}{y}$:

$$\begin{aligned} &= \frac{1}{1 + \left(\frac{x}{y}\right)^2} \cdot \frac{\partial}{\partial x} \left[\frac{x}{y} \right] \\ &= \frac{1}{1 + \left(\frac{x}{y}\right)^2} \cdot \frac{1}{y} \\ &= \frac{y}{x^2 + y^2} \end{aligned}$$

The logic behind the other partial derivative is the same, so the gradient is:

$$\nabla \theta(x, y) = \left(\frac{y}{x^2 + y^2}, -\frac{x}{x^2 + y^2} \right) \quad (5.95)$$

Finally the gradient of f can be displayed as:

$$\nabla f(d(x, y), \theta(x, y)) = \left((4x^3 - 4x, 4y^3 - 6y), \left(\frac{y}{x^2 + y^2}, -\frac{x}{x^2 + y^2} \right) \right) \quad (5.96)$$

Now, for the critical points, we need to understand where the gradient is equal to zero:

$$\nabla f((d(x, y), \theta(x, y))) = 0$$

Yeah, you got it right. We need to calculate the roots of all those equations, and we better start now! For $d(x, y)$ we have the following system:

$$\begin{cases} 4x^3 - 4x = 0 \\ 4y^3 - 6y = 0 \end{cases}$$

We can work those guys a bit:

$$\begin{cases} 4x(x^2 - 1) = 0 \\ 2y(2y^2 - 3) = 0 \end{cases} \quad (5.97)$$

For the first equation on the system 5.97, we have that either $x = 0$ or $x^2 = 1 \Rightarrow x = \pm 1$. Now, for equation numero dos of the same system, we have $y = 0$ or $y = \pm\sqrt{1.5}$.

So that means we have 4 critical points: $(1, \sqrt{1.5})$, $(1, -\sqrt{1.5})$, $(-1, \sqrt{1.5})$ and $(-1, -\sqrt{1.5})$. Because $(0,0)$ is not defined for the $\arctan(\frac{x}{y})$, it is not a critical point of f . Now we must do the same exercise for the $\nabla\theta(x, y)$:

$$\begin{aligned}\frac{\partial\theta}{\partial x} &= \frac{y}{x^2 + y^2} \\ \frac{\partial\theta}{\partial y} &= -\frac{x}{x^2 + y^2}\end{aligned}$$

We must check when:

$$\begin{cases} \frac{y}{x^2 + y^2} = 0 \\ -\frac{x}{x^2 + y^2} = 0 \end{cases} \quad (5.98)$$

Referring back to Equation 5.98, it turns out there is no solution, which implies that all the critical points of f arise from the function h . Our next step? Determine the nature of these critical points: Are they maxima, minima, or saddle points? While we don't have a specific goal, as we did before, it's crucial to understand the characteristics of these points.

Recalling our previous exploration of single-variable functions, we know that the second derivative provides crucial insights into a function's curvature. A positive second derivative, characterized by a \cup shape, indicates a minimum. Conversely, a negative value, signified by a \cap shape, suggests a maximum. Applying this knowledge will help in understanding our multi-variable function's critical points and, for that, we will once again resort to the Hessian matrix:

$$\frac{\partial^2 d}{\partial x^2} = 12x^2 - 4$$

$$\frac{\partial^2 d}{\partial y^2} = 12y^2 - 6$$

5.2. A Crowd Comes In, A Parade Goes Out: **Vector Functions**.

We are just missing the derivatives with respect to xy and yx :

$$\frac{\partial^2 d}{\partial xy} = 0$$

$$\frac{\partial^2 d}{\partial yx} = 0$$

In this particular case, they are the same. Right, so we have everything we need to form the Hessian matrix:

$$H_d(x, y) = \begin{pmatrix} 12x^2 - 4 & 0 \\ 0 & 12y^2 - 6 \end{pmatrix} \quad (5.99)$$

When we studied the convexity of a multiply entry to a scalar function, we established that, if the Hessian was a positive semi-definite matrix, then the function to which the Hessian represents the curvature was convex. Furthermore, we said there was a relationship between the eigenvalues and a positive semi-definite matrix; these guys had to be non-negative for this to be verified; so let's move on with the calculations:

- If we have the point $(1, \sqrt{1.5})$:

$$H(1, \sqrt{1.5}) = \begin{pmatrix} 8 & 0 \\ 0 & 12 \end{pmatrix}$$

Now, if we calculate the eigenvalues and the eigenvectors, we have the following:

$$\lambda_1 = 8, \quad \lambda_2 = 12, \quad v_1 = (1, 0), \quad v_2 = (0, 1)$$

Both eigenvalues of the Hessian matrix are positive, indicating the presence of a local minimum at the critical points under consideration. Given the structure of the Hessian matrix, specifically, the presence of squared terms and the symmetrical nature of the remaining critical points (which have the same absolute values for their coordinates), we can deduce that the matrices evaluated at

these points will be identical. Consequently, without further computations, we can assert that these critical points also represent local minima in the function's landscape. But we must also understand how to classify critical point other than the minima. So, let's get all the conditions listed:

- **Positive Eigenvalues (Local Minimum):**

- If all the eigenvalues of the Hessian matrix at a critical point are positive, then the function has a local minimum at that point. This is because the positive eigenvalues indicate that the function is concave up along all directions in the vicinity of that point, forming a bowl shape or an upward-facing paraboloid.

- **Negative Eigenvalues (Local Maximum):**

- If all the eigenvalues of the Hessian matrix at a critical point are negative, then the function has a local maximum at that point. This represents a situation where the function is concave down in all directions at that point, resembling an inverted bowl or a downward-facing paraboloid.

- **Mixed Sign Eigenvalues (Saddle Point):**

- If the Hessian matrix at a critical point has eigenvalues with mixed signs (some positive and some negative), then the function has a saddle point at that location. A saddle point represents a situation where the function is concave up in some directions and concave down in others, resembling a saddle shape on the graph.

The global or local minima concept here is slightly different than with a scalar function. Two functions dictate the output of f : The function d represents the distance to the food source whereas the dance angle is depicted by that θ guy. Therefore, selecting one or more of these critical points depends on the context of the problem we are dealing with.

5.2. A Crowd Comes In, A Parade Goes Out: **Vector Functions**.

Understanding this complex interplay between the distance and dance angle in the waggle dance, and the context-specific selection of critical points, leads us to consider another vital aspect of bee behavior: energy efficiency.

It requires energy for bees to fly. A closer nectar source means bees can conserve energy and make more trips in a shorter time. The closer a food source is, the less energy a bee has to expend to get there. Over time, especially if nectar flow is good, the bee can accumulate more nectar for the hive with less energy expenditure, making close sources highly attractive. The time of day can significantly influence the behavior and priorities of bees, which can, in turn, affect their choice of critical points in the waggle dance. Here's a possible scenario:

5.2.1 A Clandestine Mathematical Liaison: Vector Functions and Neural Networks.

At midday, as the sun is at its zenith, flowers in specific patches, like those in open fields (to the north-east), get more sunlight and produce more nectar; in this situation, the point $(1, \sqrt{1.5})$ might be the best option.

Because not all bees can go to the same patch due to competition, the other point that indicates a different direction, $(-1, \sqrt{1.5})$ can also be a great option. By late afternoon, the sun has moved, and certain areas, like a grove or valley in the south-east, offer shade and colder spots, making it more comfortable for bees to collect nectar without overheating. At this time of the day, both points $(-1, -\sqrt{1.5})$ and $(1, -\sqrt{1.5})$ can be better locations as bees will prioritize these less hot areas in the late afternoon to avoid the intense heat.

These are just hypotheticals, and the reality is that this behavior, the waggle dance, is a bit more complex than what is reflected by the function f . Given the importance of location for energy conserva-

tion and nectar accumulation in bees, understanding their foraging patterns becomes crucial. Say we were tasked with the challenge of forecasting the geographic position of nectar sources based on measurements from the bees' waggle dance.

Specifically, we aim to use the duration and angle of the dance to predict the latitude and longitude of these locations. The following table has been provided to us for this purpose:

Duration of Dance	Angle of Dance	Latitude	Longitude
15	30	42.1	58.7
8	145	41.9	59.1
20	270	42.4	58.5
5	90	42.0	59.2
12	210	41.8	58.8
18	330	42.2	59.0

Table 5.14: Let's get another neural network going.

The entries in the table contain the following information:

- **Duration of Dance:** The total time in seconds that the waggle phase of the dance lasted, which is directly correlated with the distance to the food source.
- **Angle of Dance:** The angle in degrees relative to the sun that indicates the direction of the food source from the hive.
- **Latitude:** The geographical latitude of the foraging location that the dance is indicating.
- **Longitude:** The geographical longitude of the foraging location that the dance is indicating.

Given this new information, it's time to adjust our model and neural network architecture to accommodate these changes:

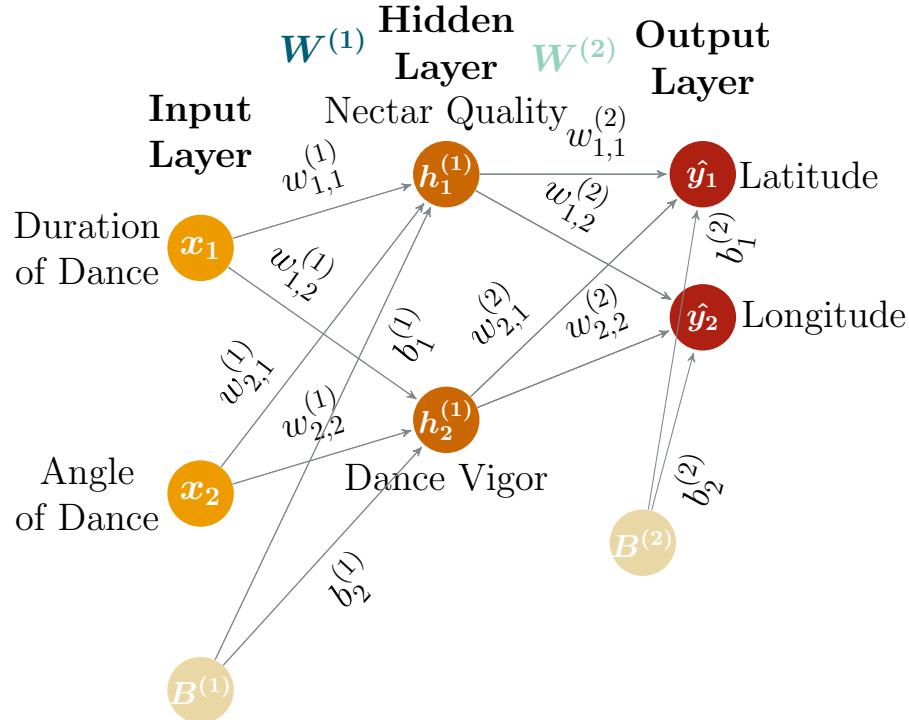


Figure 5.29: Everything on the same diagram with two output nodes.

It is similar to our original neural network. We just added one more neuron and a few more connections. The core ideas behind the model are the same: We will have a feed-forward pass based on dot products that get passed by activation functions to introduce some non-linearity in the system. The result of this pass will differ, as now we will have a vector instead of a scalar; our problem is a regression task rather than a classification task. We then need to define a cost function to create the update rules that our optimization algorithm will follow, making this the backpropagation task that will allow us to get optimized values for the weights and biases that we will randomly define. It is important to note that the configuration of the hidden neurons is but one of many possibilities, meaning that they are up to interpretation. This is possible because of the simplicity of the example and when neural networks get really deep this interpretation becomes extremely hard. Cool, so let's start defining this guy. The first part of the network is the same, so we have it that:

$$h_i^{(1)} = \phi_1 \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j + b_i^{(1)} \right) \quad (5.100)$$

The w 's are the weights that connect the input layer to the hidden layer, the b 's are the bias for the same connections, and ϕ_1 is the activation function. Let's pick a different one for this situation, the ReLU:

$$\phi_1(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

We will thus have a different equation for the other part of the network that will produce our outcomes. Before, we had:

$$\hat{y} = \phi_2 \left(\sum_{i=1}^M w_{i,1}^{(2)} \cdot h_i^{(1)} + b^{(2)} \right) \quad (5.101)$$

We now have to alter this guy as we have two outputs, one for latitude and another one for longitude, so:

$$\hat{y}_1(x) = \phi_2 \left(\sum_{i=1}^M w_{i,1}^{(2)} \cdot h_i^{(1)} + b_1^{(2)} \right) \quad (5.102)$$

And:

$$\hat{y}_2(x) = \phi_2 \left(\sum_{i=1}^M w_{i,2}^{(2)} \cdot h_i^{(1)} + b_2^{(2)} \right) \quad (5.103)$$

We can combine 5.102 and 5.103 into one by introducing an index k :

$$\hat{y}_k(x) = \phi_2 \left(\sum_{i=1}^M w_{i,k}^{(2)} \cdot h_i^{(1)} + b_k^{(2)} \right) \quad \text{for } k = 1, 2 \quad (5.104)$$

Finally, for our representation of a composition of functions:

$$\hat{y}_k(x) = \phi_2 \left(\sum_{i=1}^M w_{i,k}^{(2)} \cdot \phi_1 \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j + b_i^{(1)} \right) + b_k^{(2)} \right) \quad (5.105)$$

Where:

- \mathbf{N} : Number of neurons in the input layer.
- \mathbf{M} : Number of neurons in the hidden layer.
- $\mathbf{k} = 1, 2$

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

Since we have two distinct output equations for the \hat{y} values, we must accordingly adapt our linear combination equations, previously defined as z :

$$z_1 = \sum_{i=1}^M w_{i,1}^{(2)} \cdot h_i^{(1)} + b_1^{(2)}$$

And:

$$z_2 = \sum_{i=1}^M w_{i,2}^{(2)} \cdot h_i^{(1)} + b_2^{(2)}$$

We still have to define ϕ_2 , but first, let's see if we really need an activation function here. We are trying to predict values for both latitude and longitude, so it is not necessary to have these within a range like the previous neural network, where the output was either zero or one. Here, we are looking for two real numbers. So, we can leave that as a linear part of the network:

$$\hat{y}_k(x) = \sum_{i=1}^M w_{i,k}^{(2)} \cdot \phi_1 \left(\sum_{j=1}^N w_{j,i}^{(1)} \cdot x_j + b_i^{(1)} \right) + b_k^{(2)} \quad (5.106)$$

Right, if our output isn't strictly confined to a binary 0 or 1, we'll need to rethink our cost function. While binary cross-entropy served us well for those specific cases, it's time we revisited our reliable companion: the Mean Squared Error (MSE). This function was our go-to when plotting lines to predict real numbers. Now, the situation is somewhat similar, but with a twist: instead of a single \hat{y} , we have a pair to consider. Ensuring we account for errors in both is crucial. A straightforward way to do this? Sum up the errors, so, something along the lines of:

$$J_1(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(\hat{y}_1^{(i)} - y_1^{(i)} \right)^2 \quad (5.107)$$

Equation 5.107 represents the error we get for the output for the latitude. Next, we do the same for the longitude:

$$J_2(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(\hat{y}_2^{(i)} - y_2^{(i)} \right)^2 \quad (5.108)$$

We can combine 5.107 and 5.108:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left[\left(\hat{y}_1^{(i)} - y_1^{(i)} \right)^2 + \left(\hat{y}_2^{(i)} - y_2^{(i)} \right)^2 \right] \quad (5.109)$$

Where θ represents the set of parameters of the network, which includes all the weights and biases. We'll need to compute the partial derivatives for these components to set the optimization process in motion. The bright side is that the derivatives are determined the same way we did with the other neural network, and on top of that, I have some great news to share; we also have less to do concerning partial derivatives, as we have one less activation function than before. Our output layer doesn't have one:

$$\frac{\partial J_1}{\partial w_{1,1}^{(2)}} = \frac{\partial J_1}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{1,1}^{(2)}} \quad (5.110)$$

Because we don't have any activation function:

$$\frac{\partial \hat{y}_1}{\partial z_1} = 1$$

The function simplifies to 5.110:

$$\frac{\partial J_1}{\partial w_{1,1}^{(2)}} = \frac{\partial J_1}{\partial \hat{y}_1} \cdot \frac{\partial z_1}{\partial w_{1,1}^{(2)}} \quad (5.111)$$

Let's start with the first partial derivative:

$$\frac{\partial J_1}{\partial \hat{y}_1} = \frac{1}{m} (\hat{y}_1 - y_1)$$

Now for the other one:

$$\frac{\partial z_1}{\partial w_{1,1}^{(2)}} = h_1^{(1)}$$

Because z_1 is also defined as previously:

$$z_1 = w_{1,1}^{(2)} \cdot h_1^{(1)} + w_{2,1}^{(2)} \cdot h_2^{(1)} + b_1^{(2)}$$

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

We can generalize the equations to update the weights on this specific layer as:

$$\frac{\partial J_k}{\partial w_{i,k}^{(2)}} = \frac{\partial J_k}{\partial \hat{y}_k} \cdot \frac{\partial z_k}{\partial w_{i,k}^{(2)}} \quad (5.112)$$

Where i represents the neuron and $k = 1, 2$ is the latitude and the longitude, respectively. With this, the equation for the bias updates comes to:

$$\frac{\partial J_k}{\partial b_k^{(2)}} = \frac{\partial J_k}{\partial \hat{y}_k} \cdot \frac{\partial z_k}{\partial b_k^{(2)}}$$

But as:

$$\frac{\partial z_k}{\partial b_k^{(2)}} = 1$$

We have that:

$$\frac{\partial J_k}{\partial b_k^{(2)}} = \frac{\partial J_k}{\partial \hat{y}_k} \quad (5.113)$$

That is done! For the other parameters, the equations are very similar to the ones in the previous network, but, as we have two weighted sums z , they are:

$$\frac{\partial J_k}{\partial w_{i,j}^{(1)}} = \frac{\partial J_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial h_i} \cdot \frac{\partial h_i}{\partial z_{1,i}} \cdot \frac{\partial z_{1,i}}{\partial w_{i,j}^{(1)}} \quad (5.114)$$

For the bias:

$$\frac{\partial J_k}{\partial b_i^{(1)}} = \frac{\partial J_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial h_i} \cdot \frac{\partial h_i}{\partial z_{1,i}} \cdot \frac{\partial z_{1,i}}{\partial b_i^{(1)}} \quad (5.115)$$

We are missing two fundamental things we need to do some calculations: the weights and the bias. Well, we can use a technique that is prevalent these days. Say you are trying to defend a complex subject about which you only watched a 30-second clip or read a headline; when you are in need of an argument, make up some shit, and the problem is solved. So, by translating this technique to our world of mathematics, where logic prevails, we can randomly attribute values for the weights and bias... oh no... this is the same method we used before. Did we take this from them? Or did they

take them from us? Still, we will own our random guesses so we can optimize the hell out of them... moving on!

$$W^{(1)} = \begin{pmatrix} 0.34 & -0.56 \\ -0.21 & 0.78 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} 0.45 \\ -0.67 \end{pmatrix}$$

For the other layers:

$$W^{(2)} = \begin{pmatrix} -0.78 & 0.56 \\ 0.65 & -0.23 \end{pmatrix} \quad b^{(2)} = \begin{pmatrix} 0.34 \\ -0.56 \end{pmatrix}$$

OK, we have everything we need in order to proceed with a feed-forward pass. Well we nearly have everything, the data... is not yet scaled:

x_1	x_1^*	x_2	x_2^*	y_1	y_1^*	y_2	y_2^*
15	0.667	30	0.000	42.1	0.500	58.7	0.286
8	0.200	145	0.383	41.9	0.167	59.1	0.857
20	1.000	270	0.800	42.4	1.000	58.5	0.000
5	0.000	90	0.200	42.0	0.333	59.2	1.000
12	0.467	210	0.600	41.8	0.000	58.8	0.429
18	0.867	330	1.000	42.2	0.667	59.0	0.714

Table 5.15: Feed-forward predictions and observed values for each data point.

In Table 5.15, we've relabeled "Duration of Dance" and "Angle of Dance" as x_1 and x_2 for simplicity. These inputs have been subjected to Min-Max scaling, denoted as x_1^* and x_2^* , to ensure they contribute equally during the neural network's training.

We have also scaled the target variables latitude y_1 and longitude y_2 , respectively y_1^* and y_2^* . This was done to streamline gradient computation within the neural network. This ensures gradients are calculated on a uniform scale, aligning the treatment of inputs and outputs. Without scaling, we would require a transformation of the error terms each time we compare predictions to actual values, complicating the optimization process due to varying scales. Post-training, we will perform an inverse transformation to interpret the outputs in their original geographical context.

*In a realm of nodes and edges, I embark on a quest,
 From neuron to neuron, I travel without rest.
 Through layers of logic, my path is cast,
 Gathering knowledge as I go fast.*

What am I?

A FEED FORWARD PASS!!! Well, let's get one of those done:

x_1^*	x_2^*	\hat{y}_1	\hat{y}_2	y_1^*	y_2^*
0.666	0.000	-0.188	-0.181	0.500	0.286
0.200	0.383	-0.001	-0.315	0.167	0.857
1.000	0.800	-0.145	-0.212	1.000	0.000
0.000	0.200	0.022	-0.332	0.333	1.000
0.466	0.600	-0.036	-0.290	0.000	0.429
0.866	1.000	-0.077	-0.261	0.667	0.714

Table 5.16: Comparison of predicted and actual outputs for scaled input vectors.

Table 5.16 showcases the results of one epoch, calculated using a computer on which all the data points were processed through the network, and boy oh boy are those y 's with a hat far away from the observed y 's. We have some work to do here. Up to this point, we've delved into various optimization methods reliant on gradients and derivatives, with RMSProp being the most recent addition. RMSProp distinguishes itself from SGD by dynamically adjusting the learning rate for each parameter. This adaptability ensures an efficient traversal of the optimization landscape, particularly mitigating oscillations and slow convergence in either too steep or too flat regions. It's essential to note that these regions can exhibit vast differences in gradient values across various parameters.

However, RMSProp doesn't adequately address issues like path dependency and oscillations in the updates of parameters. The intricate landscapes of loss functions, such as valleys, ravines, or plateaued areas might also present challenges. In these zones, our optimization steps might show oscillatory or inconsistent behavior. This inconsistency could prevent us from reaching the minima effectively, or at least make our path to the minima overly zigzagged.

5.2.1.1 Gathering Steam: The Momentum Algorithm.

As we know by now, an optimization algorithm is required to train neural networks, and those with many layers and parameters, like in deep learning methods, challenges of navigating valleys often arise. Why not consider an algorithm specifically designed to address this? Moreover, as previously mentioned, it's not mandatory to perform an iteration after processing each entry vector in every epoch. There are alternative approaches we can consider. We can complete this task in batch mode after processing a set number of data points. Before diving deeper into batching, let's introduce an optimization technique named Momentum.

The concept of Momentum was inspired by physics. Picture a ball rolling downhill: as it descends, it gains speed (or Momentum), ensuring a swift and consistent movement, irrespective of changes in the hill's gradient. We have to track past and current gradients to implement this in our algorithm. We can use a similar approach to the one we used in the RMSProp, a weighted sum:

$$v_n = \beta v_{n-1} - \eta \nabla J(\theta) \quad (5.116)$$

To break this equation down, let's traverse it from left to right. First up, we have the parameter β , which essentially weighs the importance we assign to the preceding momentum, denoted by v_{n-1} . Here, v_{n-1} aggregates the insights from previous gradients, which will help the algorithm update the parameters in a consistent direction, smoothing out the oscillations, as the gradient represents the slope of the loss function with respect to the inputs.

The term $\nabla J(\theta)$ denotes the gradient, indicating the rate of change of our loss function with regard to the parameters. It provides a sense of how much the parameters need to be adjusted and which direction to reduce the loss.

Then there's η , our learning rate. By incorporating the term $\eta \nabla J(\theta)$, we can ensure that the updates are oriented toward decreasing loss. All of those factors combined essentially encapsulates

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

the essence of our update rule:

$$x_{n+1} = x_n + v_n \quad (5.117)$$

The term v_n represents the direction of our movement; hence, we update it based on the previous iteration's value. While RMSProp tailors the learning rate based on recent iterations, Momentum provides a consistent direction to our updates.

So we now have: a network architecture, some initial sets and weights, formulae to calculate partial derivatives of the cost function with respect to all weights and biases, plus a brand new optimization algorithm. This is everything we need to start backpropagating, but this time we will do in batches of two entries. Just so we have an example of this technique. Let's start with a parameter that belongs to the set of weights that connects the hidden layer to the output layer, say:

$$\frac{\partial J_1}{\partial w_{1,1}^{(2)}} = \frac{\partial J_1}{\partial \hat{y}_1} \cdot \frac{\partial z_1}{\partial w_{1,1}^{(2)}}$$

That formula would work if we wished to perform the update on a single entry. For a batch, as we will be dealing with multiple entry vectors that imply numerous errors, we must consider all of them. A way to do so is to average them out:

$$\frac{\partial J_1}{\partial w_{1,1}^{(2)}} = \frac{1}{B} \sum_{i=1}^B \frac{\partial J_{1,i}}{\partial \hat{y}_{1,i}} \cdot \frac{\partial z_{1,i}}{\partial w_{1,1}^{(2)}} \quad (5.118)$$

In equation 5.118 i represents the entry point concerning the mini-batch. It counts the entries within the mini-batch and not on the whole dataset. We are averaging the changes in the cost function with respect to a parameter. We can also generalize 5.118:

$$\frac{\partial J_{k,\text{mini-batch}}}{\partial w_{i,j}^{(l)}} = \frac{1}{B} \sum_{p=1}^B \frac{\partial J_{k,p}}{\partial \hat{y}_{k,p}} \cdot \frac{\partial z_{k,p}}{\partial w_{i,j}^{(l)}} \quad (5.119)$$

Where:

- $J_{k,p}$ is the cost for the p^{th} example with respect to the k^{th} output type (like latitude or longitude).

- $w_{i,j}^{(l)}$ is the weight connecting the i^{th} neuron in layer $l - 1$ to the j^{th} neuron in layer l .
- $\hat{y}_{k,p}$ is the predicted value for the k^{th} output type of the p^{th} example.
- $z_{k,p}$ is the pre-activation value for the k^{th} output type of the p^{th} example.
- B is the size of the mini-batch.

The bias equations are pretty much the same; the only difference is that we need to calculate the partial derivative of $z_{z,p}$ with respect to the bias term instead of the weight. Cool, so let's update that $w_{1,1}^{(2)}$ then:

- **Data Point 1 - $(x_{1,1}, x_{1,2}) = (0.666, 0.000)$:**

We now turn our attention to Data Point 1, represented by the coordinates $(x_{1,1}, x_{1,2})$ that stand for the first and second coordinates of data point 1. We'll start by examining the influence of this data point on the weight adjustment in the context of mini-batch learning. Specifically, we're interested in the gradient of the cost function $J_{1,1}$ with respect to the weight $w_{1,1}^{(2)}$, and this is represented by:

$$\frac{\partial J_{1,1}}{\partial w_{1,1}^{(2)}} = (\hat{y}_{1,1} - y_{1,1}) \cdot h_{1,1}^{(1)} \quad (5.120)$$

All of the sudden the notation got a bit heavier so let's make a point of the situation. Equation 5.120 captures the sensitivity of the cost function for the first output neuron, the latitude one, to changes in $w_{1,1}^{(2)}$. The term $\hat{y}_{1,1}$ is the predicted latitude for this data point, while $y_{1,1}$ is the actual observed latitude. The value $h_{1,1}^{(1)}$ represents the activation of the first hidden neuron after the data point has been fed through the neural network. Let's proceed with the calculation of the h fellow:

$$h_{1,1}^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_{1,1} + w_{2,1}^{(1)} \cdot x_{1,2} + b_1^{(1)} \right)$$

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

We have everything we need to get a value for $h_{1,1}^{(1)}$:

$$h_{1,1}^{(1)} = \phi_1 (0.34 \cdot 0.666 + (-0.21) \cdot 0.000 + 0.45)$$

$$h_{1,1}^{(1)} = \phi_1 (0.676)$$

Because we selected the ReLU for our activation function ϕ_1 , we have:

$$h_{1,1}^{(1)} = \max(0, 0.676) = 0.676$$

This value, in conjunction with our y 's fellows ($\hat{y}_{1,1}$ and $y_{1,1}$) that are represented in table 5.16, leads us to the gradient change for J_1 :

$$\frac{\partial J_{1,1}}{\partial w_{1,1}^{(2)}} = (-0.188 - 0.500) \cdot 0.676 = -0.465$$

- **Data Point 2 - $(x_{1,2}, x_{2,2}) = (0.200, 0.383)$:**

Now let's dive into Data Point 2 and just like before, we're on the lookout for how this data point affects our weight adjustments:

$$\frac{\partial J_{1,2}}{\partial w_{1,1}^{(2)}} = (\hat{y}_{1,2} - y_{1,2}) \cdot h_{1,2}^{(1)}$$

If to the formula of $h_{1,1}^{(1)}$ we input the values $x_{1,2}$ and $x_{2,2}$ we will get $h_{1,2}^{(1)}$:

$$h_{1,2}^{(1)} = \phi_1 \left(w_{1,1}^{(1)} \cdot x_{1,2} + w_{2,1}^{(1)} \cdot x_{2,2} + b_1^{(1)} \right)$$

$$h_{1,2}^{(1)} = \phi_1 (0.34 \cdot 0.200 + (-0.21) \cdot 0.383 + 0.45) = \phi_1(0.437)$$

When we zap it with our ReLU activation function, we get:

$$h_{1,2}^{(1)} = \max(0, 0.437)$$

With all of this we have that:

$$\frac{\partial J_{1,2}}{\partial w_{1,1}^{(2)}} = (-0.001 - 0.167) \cdot 0.437 = -0.073$$

With our gradient in hand, we're ready to figure out the average change in the cost function:

$$\frac{\partial J_1}{\partial w_{1,1}^{(2)}} = \frac{1}{2} (-0.465 - 0.073) = -0.269$$

Alright it is time for what we came here for, the Momentum algorithm iteration. First we need to set the dials for our parameters β and η :

$$\beta = 0.9, \quad \eta = 0.01$$

And let's roll out the new velocity:

$$v_1 = \beta \cdot v_0 - \eta \cdot \frac{\partial J_1}{\partial w_{1,1}^{(2)}}$$

Since we're starting from a standstill with $v_0 = 0$:

$$v_1 = -0.01 \cdot (-0.269)$$

$$v_1 = 0.002$$

So, here's the update formula for $w_{1,1}^{(2)}$:

$$w_{1,1,\text{new}}^{(2)} = w_{1,1,\text{old}}^{(2)} + v_1 \quad (5.121)$$

Let's plug the numbers into equation 5.121 and see what we get:

$$w_{1,1,\text{new}}^{(2)} = -0.78 + 0.002$$

$$w_{1,1,\text{new}}^{(2)} = -0.778$$

The process is the same on the next batch, points 3 and 4, with the velocity equation this time having a memory value $v_1 = 0.002$. The process to update all the other weights and biases on the sets of weights and biases that connect the input layer to the hidden layer is similar; we need to calculate the partial derivatives of the cost function with respect to each of the weights with equations 5.114 and 5.115.

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

We used a mini-batch technique to calculate them for two points at a time and then calculated their partial derivatives and took the average. After this process, we need to ensure convergence of this optimization guy. So, let's put it to the test over 100 epochs and observe the error trends:

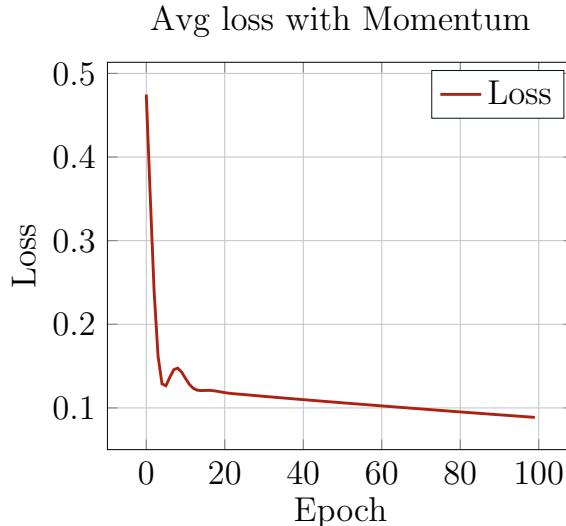


Figure 5.30: Plot of loss values over epochs.

There is a little bump on the graph 5.30, that camel back shape between iteration 5 and 10. Well this can happen because of a couple of factors:

- **Momentum Overshoot:** The algorithm's momentum term can cause it to overshoot the minimum of the loss function. This is because momentum accumulates gradients from previous steps, which can cause the updates to go past the minimum when the accumulated velocity is high.
- **Learning Rate Sensitivity:** If the learning rate is too large, it can exacerbate the effects of momentum, leading to more pronounced overshooting.

Despite the bump, the loss stabilizes shortly afterwards, indicating that the optimizer is effectively navigating the parameter space towards convergence.

This stabilization is a sign that the algorithm is compensating for the overshoot by adjusting the direction of the updates accordingly, just like we wanted to!

Momentum effectively accelerates the gradient vectors toward optimal directions, as seen in plot 5.30, a behavior that promotes faster convergence. It achieves this by accumulating a velocity for each parameter, a mix of the present gradient and the preceding velocity. Meanwhile, RMSProp refines the learning process by dynamically adjusting the learning rate for each parameter. It uses a moving average of the squared gradient, providing individualized learning rates, which is particularly beneficial when specific parameters demand more nuanced adjustments than their counterparts. Both methods seem strong but unfortunately, sometimes that bump that we observe in plot 5.30 does not stabilize, meaning that we have to find some way to introduce a stop to the somewhat chaotic behavior of the optimizer:

1. **Adjust the Learning Rate:** A smaller learning rate can reduce the magnitude of each update, thereby minimizing the risk of overshooting and helping the loss to stabilize.
2. **Fine-Tune Momentum Hyperparameters:** Adjusting the hyperparameter can help manage the velocity of the updates, potentially leading to a smoother descent.
3. **Use Adaptive Optimizers:** Given the strengths of both methods, it's only logical to contemplate whether we could merge their capabilities for an even superior algorithm. We not only can, but it has already been done, and it has the name of ADAM.

5.2.2 Smooth Sailing with a Customized Map: Embarking on the Adaptive Moment Estimation.

This algorithm combines the two algorithms we just studied: RMSProp and Momentum. This method calculates an exponential moving average of the gradients, using them to update the weights in a way that mirrors the velocity update in the Momentum method. Simultaneously, it maintains an exponential moving average of the squared gradients, characteristic of the RMSProp algorithm, which it uses to adjust the learning rates adaptively. By doing so, we can harness both Momentum's ability to navigate the optimization landscape with a velocity-driven approach and RMSProp's capacity to adapt learning rates for each parameter. The resultant algorithm thus offers faster convergence and improved performance across many scenarios, solidifying its status as one of the most recommended optimizers for neural networks, especially the ones with a ton of neurons, you know those trendy things that are used in deep learning. Now we need equations!

1. $v_t = \beta_1 \cdot v_{t-1} + (1 - \beta_1) \nabla J(\theta)$ - The velocity or momentum.
2. $s_t = \beta_2 \cdot s_{t-1} + (1 - \beta_2) \nabla (J(\theta))^2$ - The correction function for the learning rate from RMSProp.

So, the update rule that comes when we combine these two equations is:

$$x_{n+1} = x_n - \eta \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon} \quad (5.122)$$

Both \hat{v}_t and \hat{s}_t arise from a necessary correction in our algorithm. Recall that at the start, both v_t and s_t are initialized to zero. This is due to the need for prior information in the initial iteration. Such a setup, however, introduces an inherent bias towards zero, especially pronounced when the decay rates, β_1 and β_2 , are near 1. As the algorithm begins its early iterations, both v (from the momentum component) and s (from the RMSProp feature) can show a strong

tendency towards zero. To rectify this skewed behavior, it's imperative to apply a correction, introducing a counteracting bias, just as we did with the neural networks, oh bias bias. I better stop there, before another poem comes out:

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t}$$

$$\hat{s}_t = \frac{s_t}{1 - \beta_2^t}$$

In the initial iterations, when t is small, both β_1^t and β_2^t remain close to 1. This results in a small denominator, which amplifies the values of \hat{v}_t and \hat{s}_t . However, as t grows, the correction factor tends towards 1, making the bias correction increasingly insignificant. Without this adjustment, the algorithm would face difficulties during its early stages.

I lived in London for roughly six years, and amongst many things, one that stood out for me was that most of the standard measure of success is uniquely linked to material wealth and financial assets. To a certain extent, I understand it, but it also raises the question of whether there are better use cases for personalization of success. Perhaps we should create our own parameters needed for success, say, Effort, Perseverance and Learning, and strive to optimize it. This would be an approach similar to the one of exploring with RMSProp (remember that guy?) It introduced us to employing a distinct learning rate for each parameter, thus fine-tuning the optimization process to the task at hand.

To get to that particular algorithm, we had to start with the fundamental concepts of derivatives, limits, critical points, and integrals within the comfortable boundaries of one-dimensional space. As we progressed, we ventured into higher-dimensional functions, exploring the intricacies of scalar and vector functions and grappling with complex concepts such as gradients and Hessians. This voyage of discovery culminated in the creation of neural networks, bringing together nearly all the theories and techniques we had covered previously.

5.2. A Crowd Comes In, A Parade Goes Out: Vector Functions.

Indeed, mastering the mathematical concepts presented in this book necessitated a concerted effort and sustained perseverance, promoting our learning step by step. As we approach the end of our journey, we need a final stretch: an application for the ADAM algorithm. Well, that creates the need for a function, so why not use success as the dependent variable?

$$\begin{aligned} \text{Success} = & - (a(\text{Effort} - b)^2 \\ & + c(\text{Perseverance} - d)^2 \\ & + e(\text{Learning} - f)^2) \end{aligned}$$

If we consider Effort to be x_1 , Perseverance x_2 and Learning x_3 :

$$F(x_1, x_2, x_3) = - (a(x_1 - b)^2 + c(x_2 - d)^2 + e(x_3 - f)^2) \quad (5.123)$$

The function 5.123 has 6 parameters, so we can optimize a, b, c, d, e and f . The negative sign is introduced to convert the minima of the quadratic functions to maxima, facilitating the search for maximum points during the optimization process. Now, we will select one of those parameters, let's choose a , and apply one iteration of the ADAM optimization algorithm. For the initial iteration, the values are provided as follows:

$$\begin{aligned} \eta &= 0.001, \quad \beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \epsilon = 1e^{-8} \\ v_0 &= 0, \quad s_0 = 0 \end{aligned}$$

The parameters for equation 5.123 are initialized as:

$$a_0 = 4, \quad b = 2, \quad c = 3, \quad d = 3, \quad e = 4, \quad f = 5$$

Considering 1 unit of hard work, 2 units of perseverance, and 3 units of learning, we have:

$$x_1 = 1, \quad x_2 = 2, \quad x_3 = 3$$

After one iteration, we assume the following parameter values (to be validated with calculations):

$$a_1 = 3.999, \quad v_1 = 1.999, \quad s_1 = 0.400, \quad \hat{v}_1 = 20.0, \quad \hat{s}_1 = 400.0$$

Moving to iteration 2, we proceed as follows:

1. Compute Gradient at iteration 2:

To find the gradient of function F with respect to parameter a at iteration 2, we calculate:

$$\nabla F = \frac{\partial F}{\partial a} \bigg|_{a=a_1} = -2(x_1 - b)^2 = -2$$

2. Update Moment Estimates:

The estimates for the velocity (v_t) and the scaling (s_t) components are updated using the gradients calculated in the step above:

$$\begin{aligned} v_2 &= \beta_1 v_1 + (1 - \beta_1) \nabla F = 3.799 \\ s_2 &= \beta_2 s_1 + (1 - \beta_2) (\nabla F)^2 = 0.799 \end{aligned}$$

3. Correct Bias in Moment Estimates:

The zero-initialization bias in the moment estimates is corrected to ensure more accurate updates, especially in the early iterations:

$$\begin{aligned} \hat{v}_2 &= \frac{v_2}{1 - \beta_1^2} = 20.0 \\ \hat{s}_2 &= \frac{s_2}{1 - \beta_2^2} = 400.0 \end{aligned}$$

4. Update Parameter:

The parameter updates are applied, combining the corrected estimates of the velocity and scaling components to move towards the point of optimization:

$$a_2 = a_1 - \eta \frac{\hat{v}_2}{\sqrt{\hat{s}_2} + \epsilon} = 3.996$$

So, we have applied our final iterative method to a modest equation representing an individual's perception of success and that is it for calculus.

I had a cool journey writing this book, and I hope the document has been of valuable to you.

Throughout our discussions on machine learning, the concept of 'labels' emerged as a focal point to make machines understand how

5.2. A Crowd Comes In, A Parade Goes Out: **Vector Functions**.

to separate data. It's fascinating to observe the paradox when we all agree that an increase in labels escalates the complexity and challenges for a model to excel. However, when this notion transposes to human behavior, we show an unyielding belief in our categorizations.

This predisposition towards “labeling” often leads to tribalism, impeding our capacity to cultivate a broader perspective. Such a process inherently fosters a hierarchical structure, wherein a dominant classification presupposes a series of underlying attributes, sidelining the ability to understand further nuances.

Perhaps in future, we should leave the labels for the machines, let them do the separation, and then leverage this knowledge to gain perspective, so we can iterate and optimize our perceptions.

Peace

Jorge

Don't forget that there are complementary practical and coding exercises with solutions! Also, please consider leaving the book a review.

[Link for exercises!](#)

