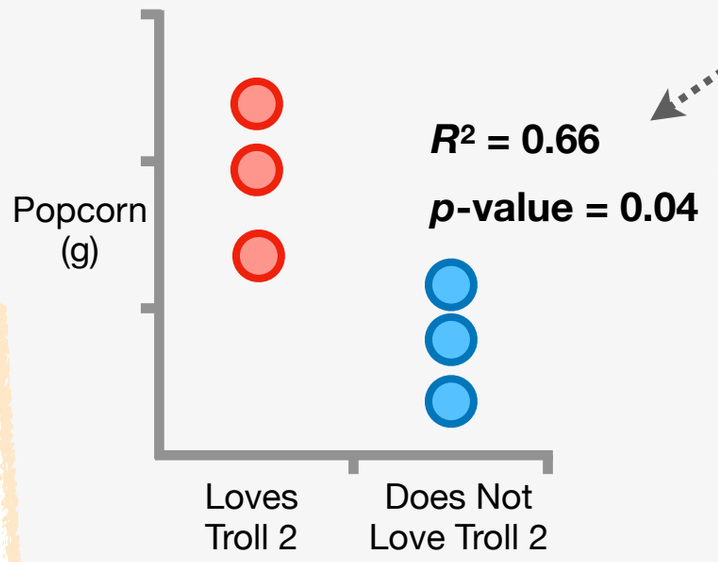
Beyond Linear Regression

1 As mentioned at the start of this chapter, **Linear Regression** is the gateway to something called **Linear Models**, which are incredibly flexible and powerful.

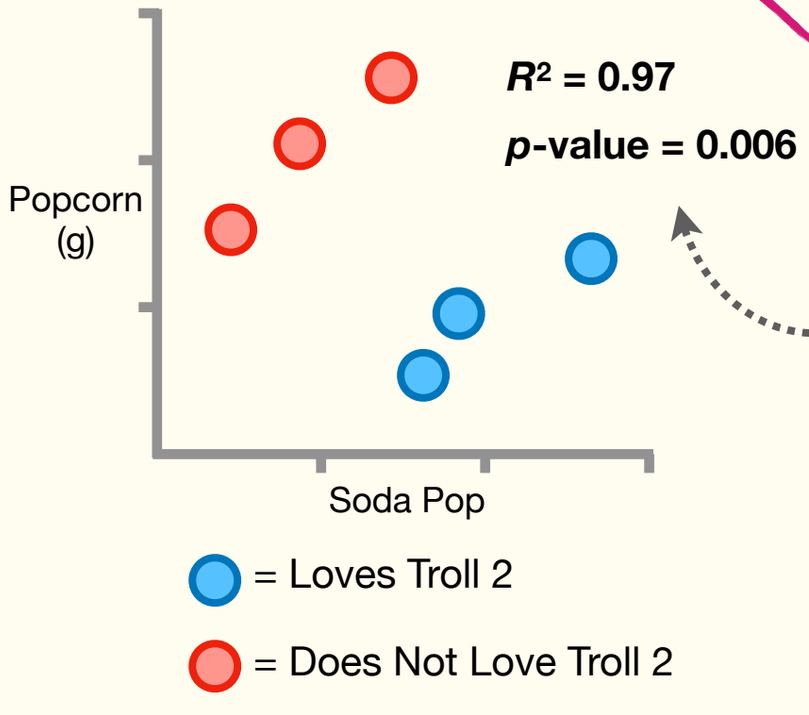
2 **Linear Models** allow us to use ***discrete*** data, like whether or not someone loves the movie Troll 2, to predict something ***continuous***, like how many grams of Popcorn they eat each day.



3 Just like when we used Weight to predict Height, **Linear Models** will give us an R^2 for this prediction, which gives us a sense of how accurate the predictions will be, and a ***p*-value** that lets us know how much confidence we should have in the predictions.

In this case, the ***p*-value** of **0.04** is relatively *small*, which suggests that it would be unlikely for random data to give us the same result or something more extreme. In other words, we can be confident that knowing whether or not someone loves Troll 2 will improve our prediction of how much Popcorn they will eat.

4 **Linear Models** also easily combine ***discrete*** data, like whether or not someone loves Troll 2**,** with ***continuous*** data, like how much Soda Pop they drink, to predict something ***continuous***, like how much Popcorn they will eat.

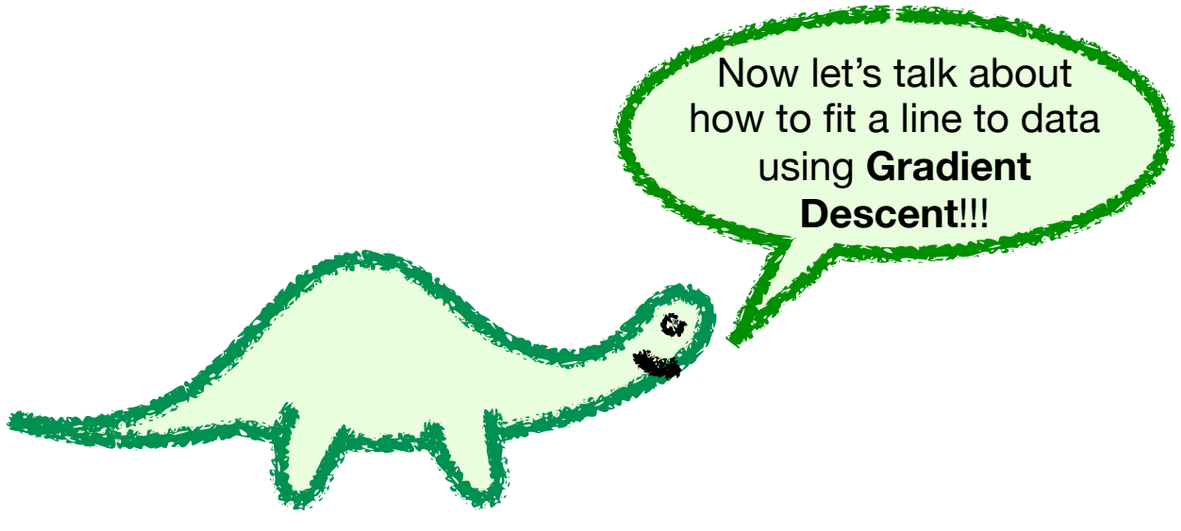


In this case, adding how much Soda Pop someone drinks to the model dramatically increased the R^2 value, which means the predictions will be more accurate, and reduced the ***p*-value**, suggesting we can have more confidence in the predictions.

DOUBLE BAM!!!

5 If you'd like to learn more about **Linear Models**, scan, click, or tap this QR code to check out the **'Quests** on YouTube!!!

bam.


Now let's talk about how to fit a line to data using **Gradient Descent**!!!

Chapter 05

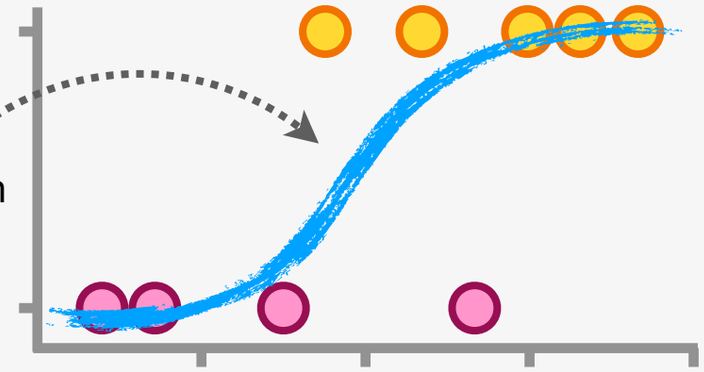
Gradient Descent!!!

Gradient Descent: Main Ideas

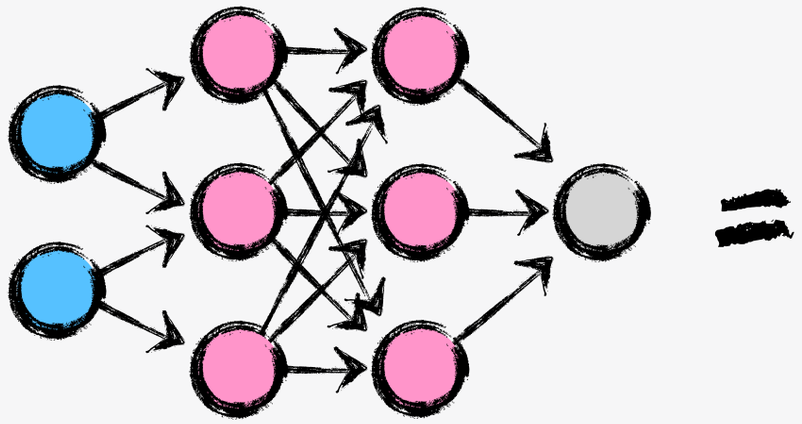
1

The Problem: A major part of machine learning is optimizing a model's fit to the data. Sometimes this can be done with an analytical solution, but it's not always possible.

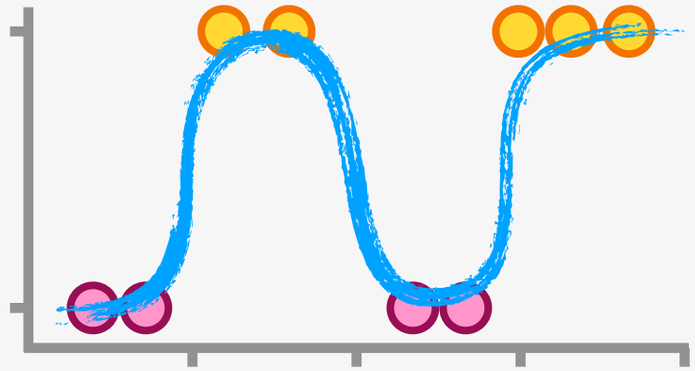
For example, there is no analytical solution for **Logistic Regression** (**Chapter 6**), which fits an **s-shaped squiggle** to data.



Likewise, there is no analytical solution for **Neural Networks** (**Chapter 12**), which fit **fancy squiggles** to data.



=



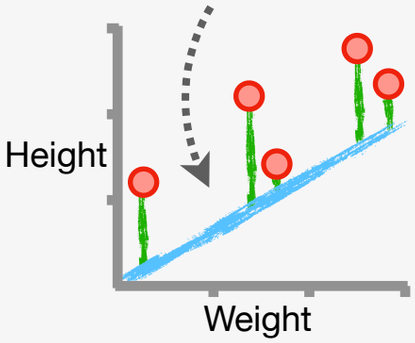
2

A Solution: When there's no analytical solution, **Gradient Descent** can save the day!

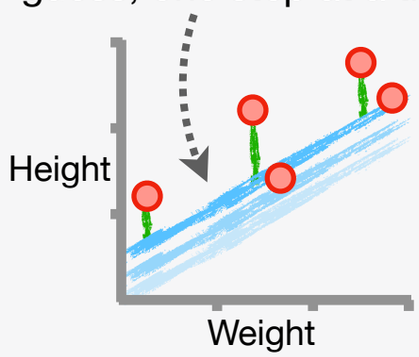
Gradient Descent is an ***iterative solution*** that incrementally steps toward an optimal solution and is used in a very wide variety of situations.

3

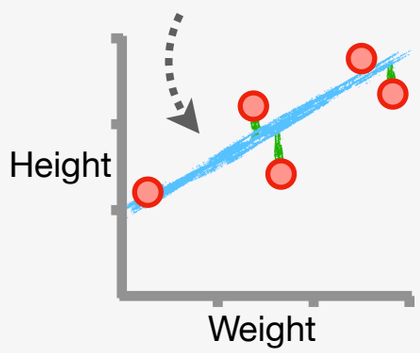
Gradient Descent starts with an initial guess…



…and then improves the guess, one step at a time…



…until it finds an optimal solution or reaches a maximum number of steps.

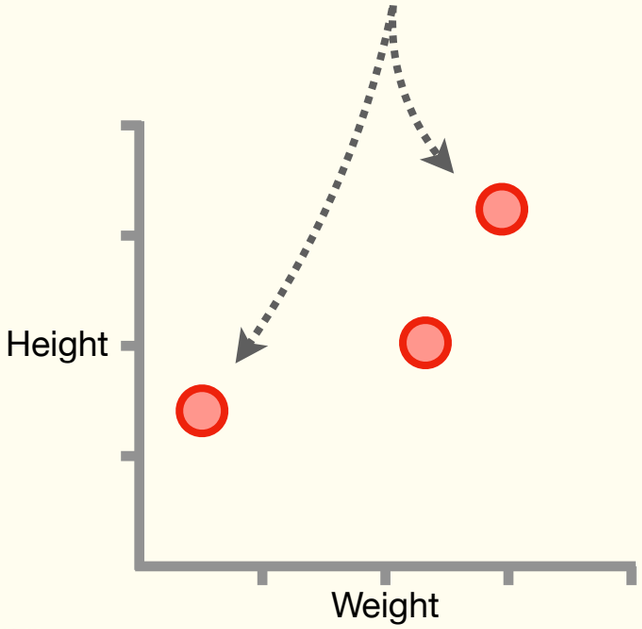


BAM!!!

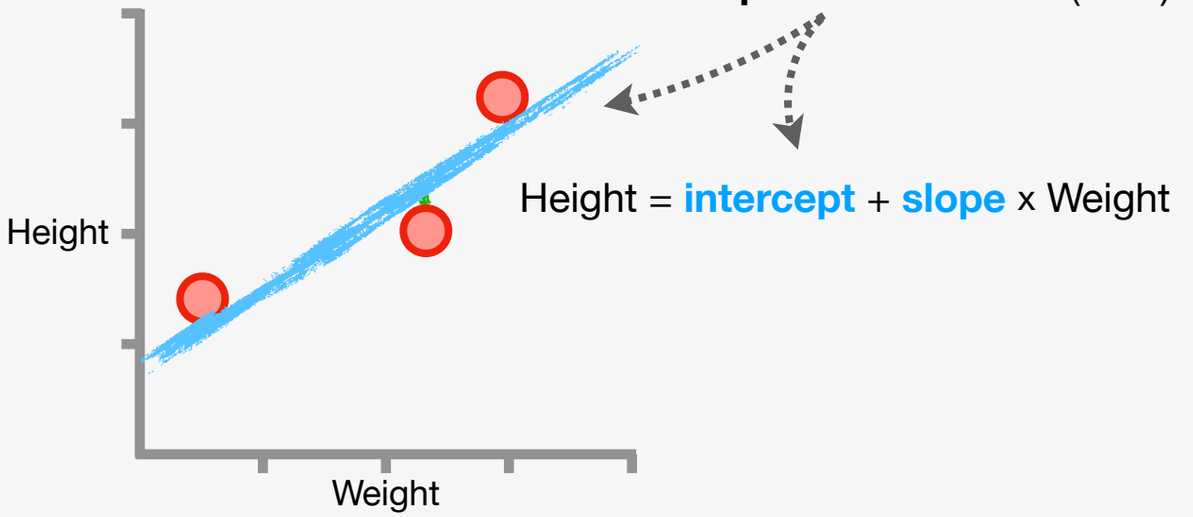
Gradient Descent: Details Part 1

NOTE: Even though there's an analytical solution for **Linear Regression**, we're using it to demonstrate how **Gradient Descent** works because we can compare the output from **Gradient Descent** to the known optimal values.

1 Let's show how **Gradient Descent** fits a line to these Height and Weight measurements.



2 Specifically, we'll show how **Gradient Descent** estimates the **intercept** and the **slope** of this line so that we minimize the **Sum of the Squared Residuals** (**SSR**).

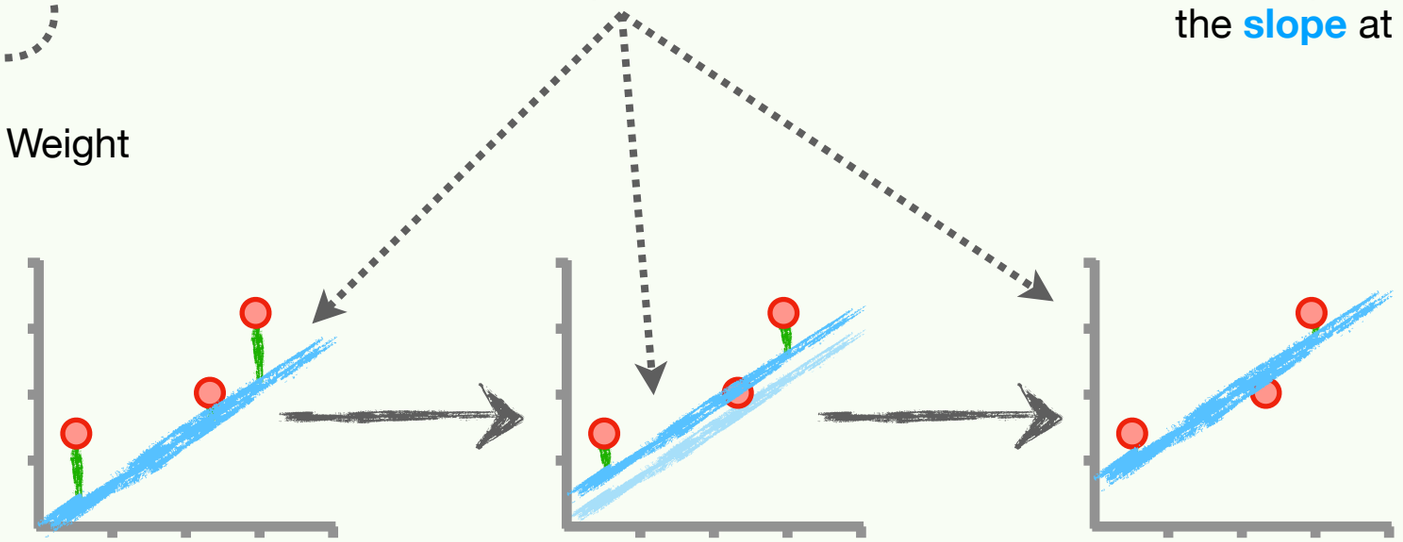


3 To keep things simple at the start, let's plug in the analytical solution for the **slope**, **0.64**…

Height = **intercept** + **0.64** x Weight

…and show how **Gradient Descent** optimizes the **intercept** one step at a time.

Once we understand how **Gradient Descent** optimizes the **intercept**, we'll show how it optimizes the **intercept** *and* the **slope** at the same time.

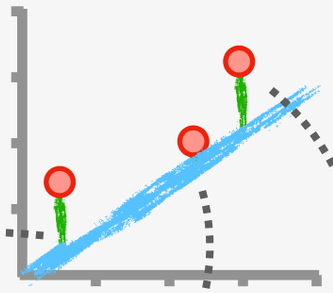


Gradient Descent: Details Part 2

4 In this example, we're fitting a line to data, and we can evaluate how well that line fits with the **Sum of the Squared Residuals** (**SSR**).

Remember, **Residuals** are the difference between the Observed and Predicted values…

5 Now, because we have **3** data points, and thus, **3 Residuals**, the **SSR** has **3** terms.



$$SSR = (\text{Observed Height}_1 - (\textbf{intercept} + 0.64 \times \text{Weight}_1))^2 + (\text{Observed Height}_2 - (\textbf{intercept} + 0.64 \times \text{Weight}_2))^2 + (\text{Observed Height}_3 - (\textbf{intercept} + 0.64 \times \text{Weight}_3))^2$$

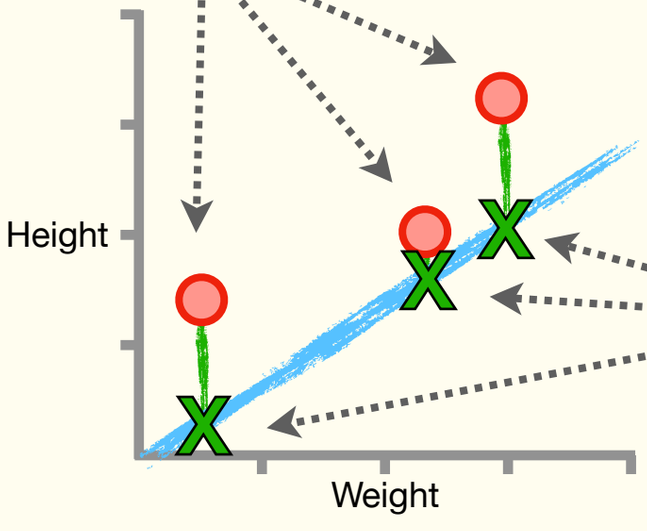
$$\text{Residual} = (\text{Observed Height} - \text{Predicted Height}) = (\text{Height} - (\textbf{intercept} + 0.64 \times \text{Weight}))$$

…and the Observed Heights are the values we originally measured…

…and the Predicted Heights come from the equation for the line…

…so we can plug the equation for the line in for the Predicted value.

Psst! Don't forget to read **Step 5**!!!

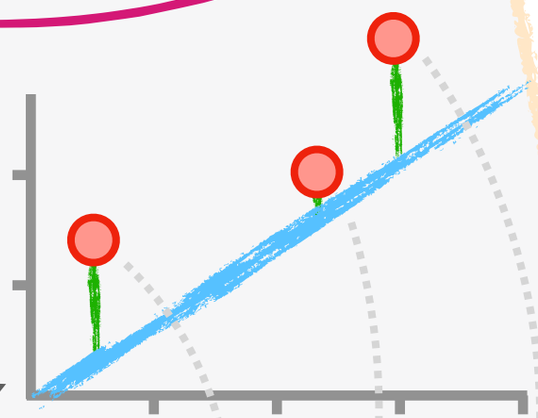


$$\text{Predicted Height} = \textbf{intercept} + 0.64 \times \text{Weight}$$

Gradient Descent: Details Part 3

6 In this first example, since we're only optimizing the y-axis **intercept**, we'll start by assigning it a random value. In this case, we'll initialize the **intercept** by setting it to **0**.

Height = **0** + 0.64 x Weight



7 Now, to calculate the **SSR**, we first plug the value for the y-axis **intercept**, **0**, into the equation we derived in **Steps 4** and **5**…

$$SSR = (Observed\ Height_1 - (\mathbf{intercept} + 0.64 \times Weight_1))^2$$
$$+ (Observed\ Height_2 - (\mathbf{intercept} + 0.64 \times Weight_2))^2$$
$$+ (Observed\ Height_3 - (\mathbf{intercept} + 0.64 \times Weight_3))^2$$

$$SSR = (Observed\ Height_1 - (\mathbf{0} + 0.64 \times Weight_1))^2$$
$$+ (Observed\ Height_2 - (\mathbf{0} + 0.64 \times Weight_2))^2$$
$$+ (Observed\ Height_3 - (\mathbf{0} + 0.64 \times Weight_3))^2$$

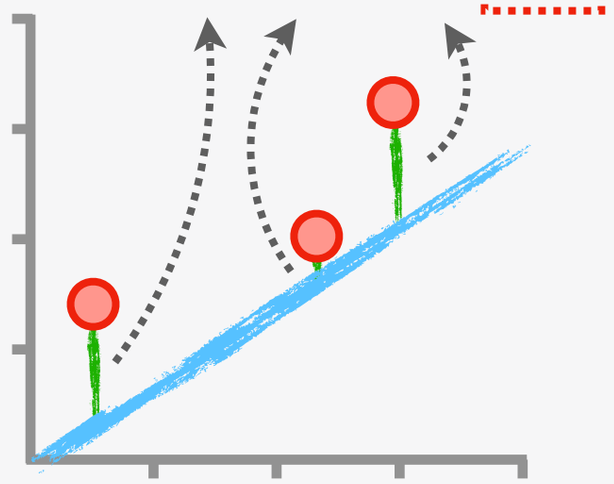
8 …then we plug in the **Observed** values for Height and Weight for each data point.

$$SSR = (Observed\ Height_1 - (\mathbf{0} + 0.64 \times Weight_1))^2$$
$$+ (Observed\ Height_2 - (\mathbf{0} + 0.64 \times Weight_2))^2$$
$$+ (Observed\ Height_3 - (\mathbf{0} + 0.64 \times Weight_3))^2$$

$$SSR = (\mathbf{1.4} - (\mathbf{0} + 0.64 \times \mathbf{0.5}))^2$$
$$+ (\mathbf{1.9} - (\mathbf{0} + 0.64 \times \mathbf{2.3}))^2$$
$$+ (\mathbf{3.2} - (\mathbf{0} + 0.64 \times \mathbf{2.9}))^2$$

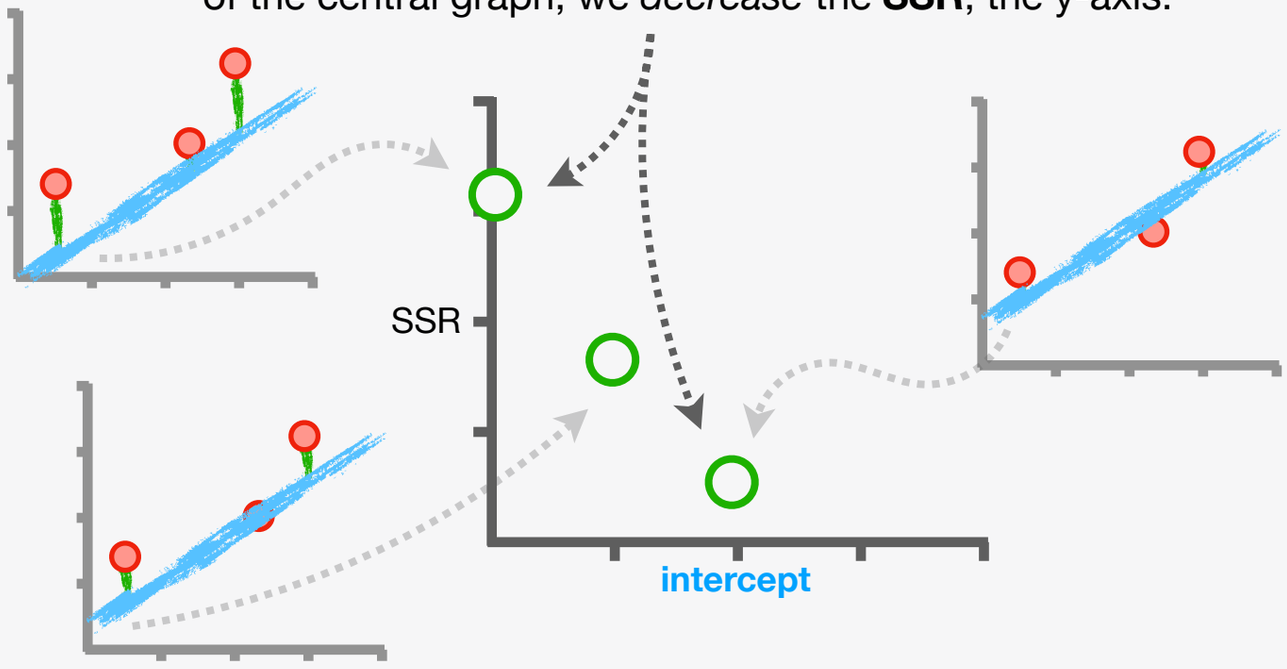
9 Lastly, we just do the math. The **SSR** for when the y-axis **intercept** is set to **0** is **3.1**. Bam!

$$1.1^2 + 0.4^2 + 1.3^2 = 3.1$$



Gradient Descent: Details Part 4

10 Now, because the goal is to minimize the **SSR**, it's a type of **Loss** or **Cost Function** (see **Terminology Alert** on the right). In **Gradient Descent**, we minimize the **Loss** or **Cost Function** by taking steps away from the initial guess toward the optimal value. In this case, we see that as we *increase* the **intercept**, the x-axis of the central graph, we *decrease* the **SSR**, the y-axis.



TERMINOLOGY ALERT!!!

The terms **Loss Function** and **Cost Function** refer to anything we want to optimize when we fit a model to data. For example, we want to optimize the **SSR** or the **Mean Squared Error** (**MSE**) when we fit a straight line with **Regression** or a squiggly line (in **Neural Networks**). That said, some people use the term **Loss Function** to specifically refer to a function (like the **SSR**) applied to *only one data point*, and use the term **Cost Function** to specifically refer to a function (like the **SSR**) applied to *all* of the data. Unfortunately, these specific meanings are not universal, so be aware of the context and be prepared to be flexible. In this book, we'll use them together and interchangeably, as in "The **Loss** or **Cost Function** is the **SSR**."

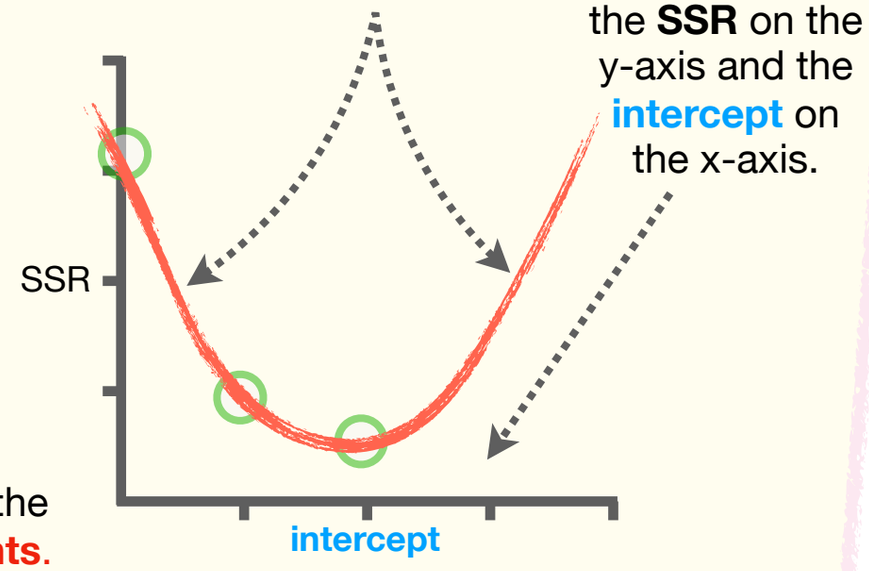
11 However, rather than just randomly trying a bunch of values for the y-axis **intercept** and plotting the resulting **SSR** on a graph, we can plot the **SSR** as a function of the y-axis **intercept**. In other words, this equation for the **SSR**…

$$SSR = (\mathbf{1.4} - (\mathbf{intercept} + 0.64 \times \mathbf{0.5}))^2 + (\mathbf{1.9} - (\mathbf{intercept} + 0.64 \times \mathbf{2.3}))^2 + (\mathbf{3.2} - (\mathbf{intercept} + 0.64 \times \mathbf{2.9}))^2$$

Psst! Remember: these are the **Observed Heights**…

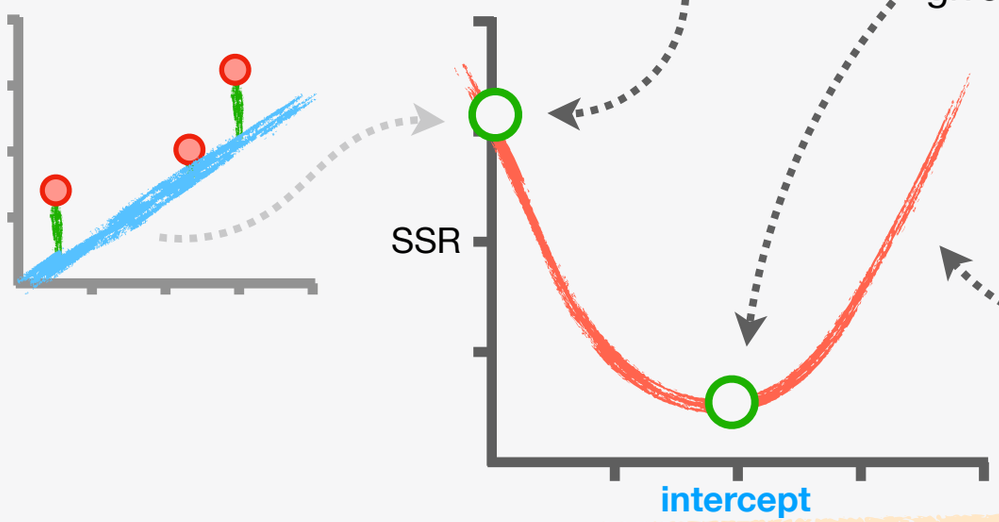
…and these are the **Observed Weights**.

…corresponds to this **curve** on a graph that has the **SSR** on the y-axis and the **intercept** on the x-axis.



Gradient Descent: Details Part 5

12 Now, when we started with the y-axis **intercept = 0**, we got this **SSR**…

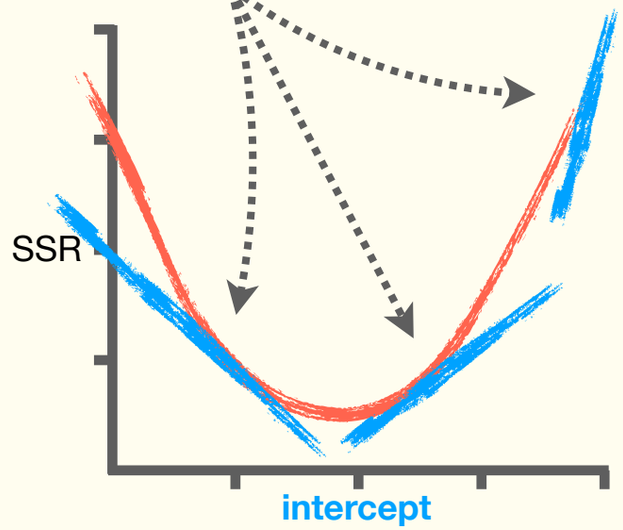


…so how do we take steps toward this y-axis intercept that gives us the lowest **SSR**…

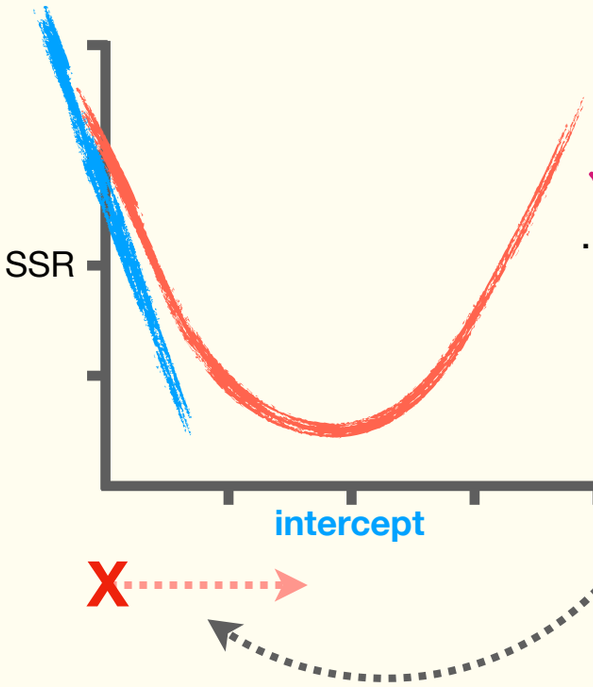
…and how do we know when to stop or if we've gone too far?

13 The answers to those questions come from the derivative of the curve, which tells us the slope of any **tangent line** that touches it.

NOTE: See **Appendix D** to learn more about derivatives.

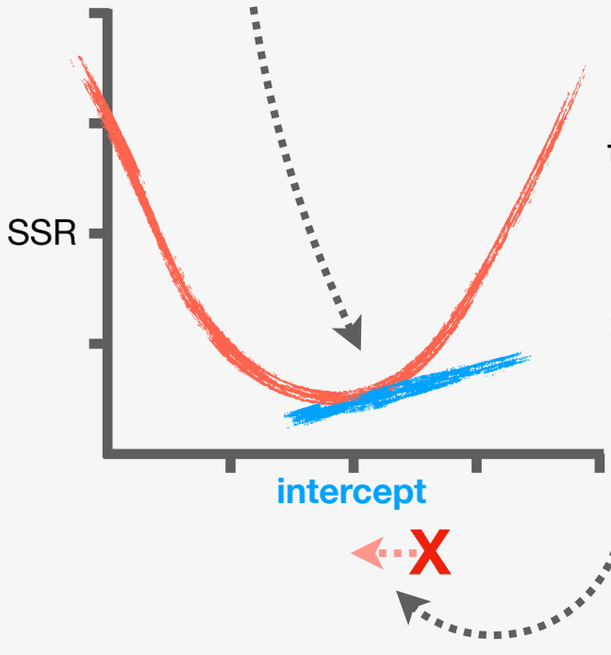


14 A relatively large value for the derivative, which corresponds to a relatively steep slope for the **tangent line**, suggests we're relatively far from the bottom of the curve, so we should take a relatively large step…



…and a *negative* derivative, or slope, tells us that we need to take a step to the right to get closer to the lowest **SSR**.

15 A relatively small value for the derivative suggests we're relatively close to the bottom of the curve, so we should take a relatively small step…



…and a *positive* derivative tells us that we need to take a step to the left to get closer to the lowest **SSR**.

In summary, the derivative tells us in which direction to take a step and how large that step should be, so let's learn how to take the derivative of the **SSR**!!!

Gradient Descent: Details Part 6

16 Because a single term of the **SSR** consists of a **Residual**… …wrapped in parentheses and squared… …one way to take the derivative of the **SSR** is to use **The Chain Rule** (see **Appendix F** if you need to refresh your memory about how **The Chain Rule** works).

$$SSR = (\text{Height} - (\text{intercept} + 0.64 \times \text{Weight}))^2$$

Step 1: Create a *link* between the **intercept** and the **SSR** by rewriting the **SSR** as the function of the **Residual**.

$$SSR = (\text{Residual})^2 \qquad \text{Residual} = \text{Height} - (\text{intercept} + 0.64 \times \text{Weight})$$

Step 2: Because the **Residual** links the **intercept** to the **SSR**, **The Chain Rule** tells us that the derivative of the **SSR** with respect to the **intercept** is…

$$\frac{d\,SSR}{d\,intercept} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,intercept}$$

Because of the subtraction, we can remove the parentheses by multiplying everything inside by **-1**.

Step 3: Use **The Power Rule** (**Appendix E**) to solve for the two derivatives.

$$\frac{d\,SSR}{d\,Residual} = \frac{d}{d\,Residual}(\text{Residual})^2 = 2 \times \text{Residual}$$

$$\frac{d\,Residual}{d\,intercept} = \frac{d}{d\,intercept}(\text{Height} - (\text{intercept} + 0.64 \times \text{Weight}))$$

$$= \frac{d}{d\,intercept}(\text{Height} - \text{intercept} - 0.64 \times \text{Weight})$$

$$= 0 - 1 - 0 = -1$$

Because the first and last terms do not include the **intercept**, their derivatives, with respect to the **intercept**, are both **0**. However, the second term is the negative **intercept**, so its derivative is **-1**.

Step 4: Plug the derivatives into **The Chain Rule** to get the final derivative of the **SSR** with respect to the **intercept**.

$$\frac{d\,SSR}{d\,intercept} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,intercept} = 2 \times \text{Residual} \times -1$$

$$= 2 \times (\text{Height} - (\text{intercept} + 0.64 \times \text{Weight})) \times -1$$

$$= -2 \times (\text{Height} - (\text{intercept} + 0.64 \times \text{Weight}))$$

Multiply this **-1** on the right by the **2** on the left to get **-2**.

BAM!!!

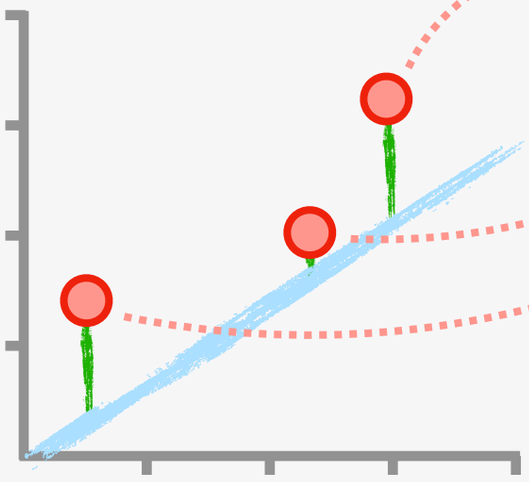
Gradient Descent: Details Part 7

17 So far, we've calculated the derivative of the **SSR** for a single observation.

However, we have three observations in the dataset, so the **SSR** and its derivative both have three terms.

$$SSR = (Height - (intercept + 0.64 \times Weight))^2$$

$$\frac{d\,SSR}{d\,intercept} = -2 \times \mathbf{(}\ Height - (intercept + 0.64 \times Weight)\ \mathbf{)}$$

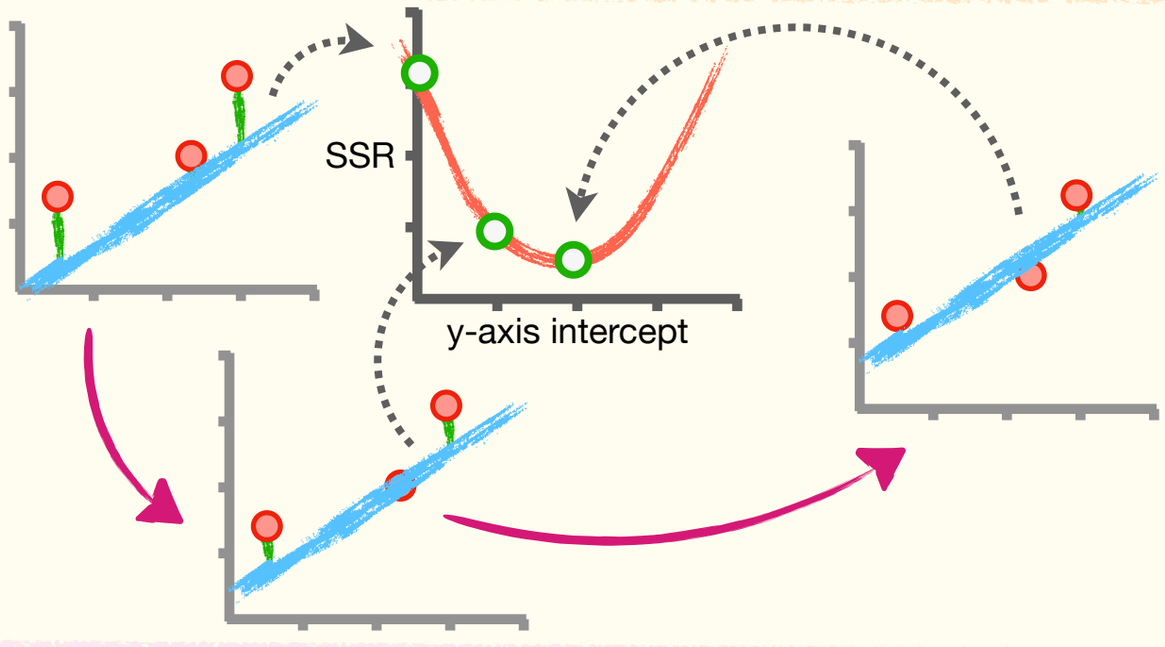


$$SSR = (Height - (intercept + 0.64 \times Weight))^2 + (Height - (intercept + 0.64 \times Weight))^2 + (Height - (intercept + 0.64 \times Weight))^2$$

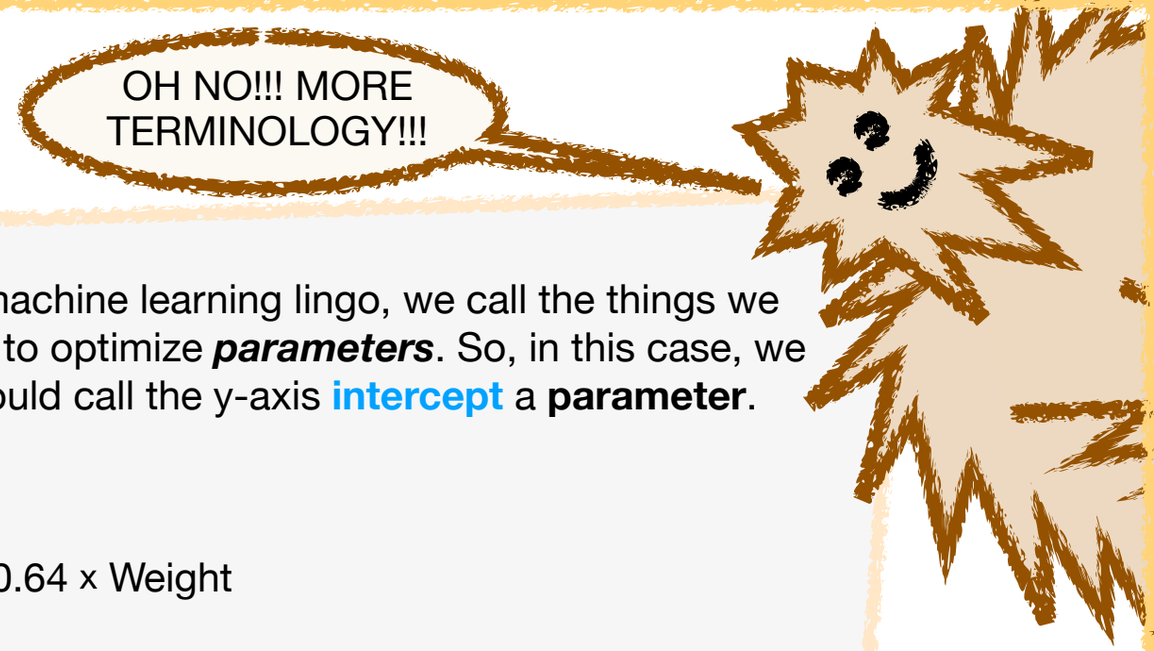
$$\frac{d\,SSR}{d\,intercept} = -2 \times \mathbf{(}\ Height - (intercept + 0.64 \times Weight)\ \mathbf{)} + -2 \times \mathbf{(}\ Height - (intercept + 0.64 \times Weight)\ \mathbf{)} + -2 \times \mathbf{(}\ Height - (intercept + 0.64 \times Weight)\ \mathbf{)}$$

Gentle Reminder: Because we're using **Linear Regression** as our example, we don't actually ***need*** to use **Gradient Descent** to find the optimal value for the intercept. Instead, we could just set the derivative equal to **0** and solve for the **intercept**. This would be an analytical solution. However, by applying **Gradient Descent** to this problem, we can compare the optimal value that it gives us to the analytical solution and evaluate how well **Gradient Descent** performs. This will give us more confidence in **Gradient Descent** when we use it in situations without analytical solutions like **Logistic Regression** and **Neural Networks**.

18 Now that we have the derivative of the **SSR** for all **3** data points, we can go through, step-by-step, how **Gradient Descent** uses this derivative to find the **intercept** value that minimizes the **SSR**. However, before we get started, it's time for the dreaded **Terminology Alert!!!**



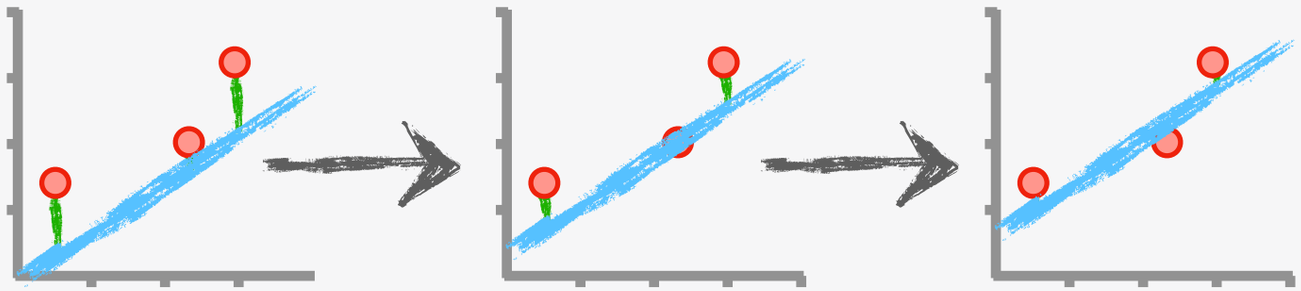
Terminology Alert!!! Parameters



1. In the current example, we're trying to optimize the y-axis **intercept**.

In machine learning lingo, we call the things we want to optimize ***parameters***. So, in this case, we would call the y-axis **intercept** a **parameter**.

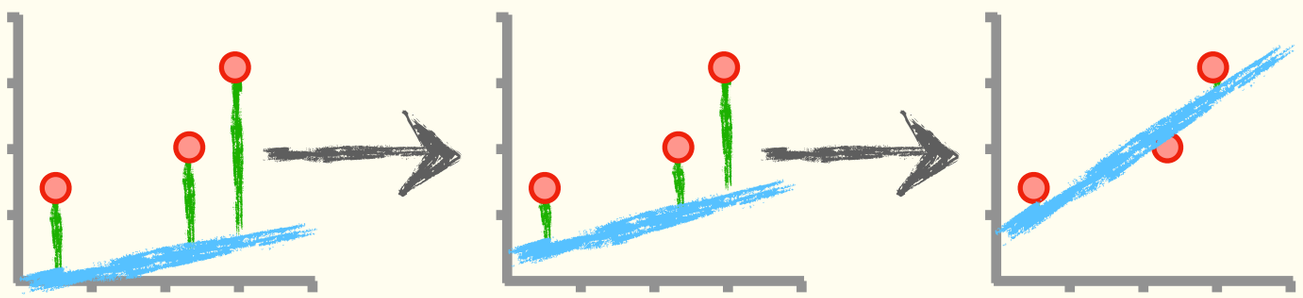
Predicted Height = **intercept** + 0.64 x Weight



2. If we wanted to optimize both the y-axis **intercept** and the **slope**, then we would need to optimize two **parameters**.

tiny bam.

Predicted Height = **intercept** + **slope** x Weight



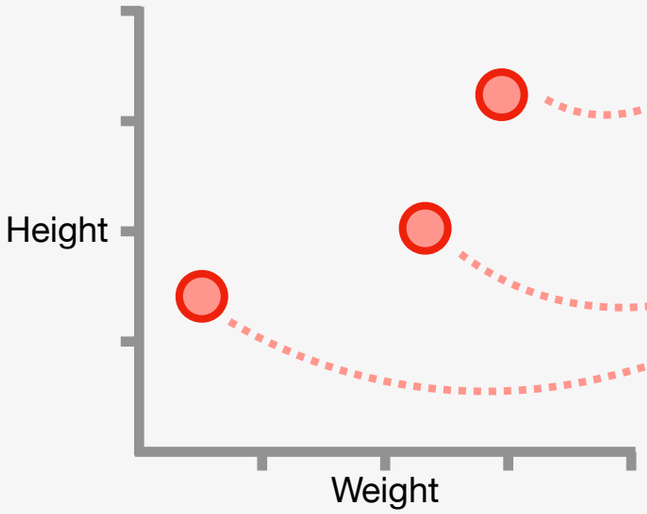
3. Now that we know what we mean when we say **parameter**, let's see how **Gradient Descent** optimizes a single **parameter**, the **intercept**, one step at a time!!!

BAM!!!

Gradient Descent for One Parameter: Step-by-Step

1 First, plug the **Observed** values into the derivative of the **Loss** or **Cost Function**. In this example, the **SSR** is the **Loss** or **Cost Function**…

…so that means plugging the **Observed Weight** and **Height** measurements into the derivative of the **SSR**.



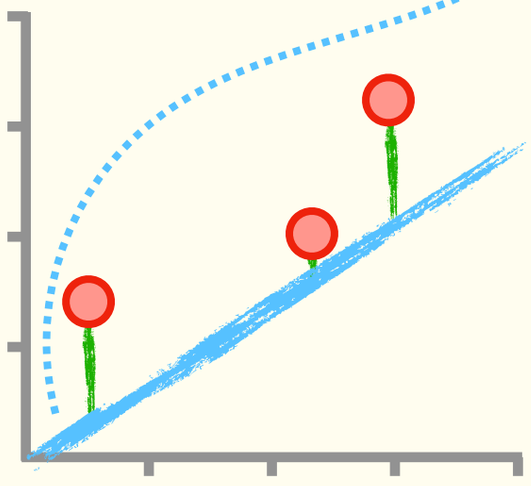
$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,Height - (intercept + 0.64 \times Weight)\,)$$
$$+ -2 \times (\,Height - (intercept + 0.64 \times Weight)\,)$$
$$+ -2 \times (\,Height - (intercept + 0.64 \times Weight)\,)$$

$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,3.2 - (intercept + 0.64 \times 2.9)\,)$$
$$+ -2 \times (\,1.9 - (intercept + 0.64 \times 2.3)\,)$$
$$+ -2 \times (\,1.4 - (intercept + 0.64 \times 0.5)\,)$$

2 Now we initialize the parameter we want to optimize with a random value. In this example, where we just want to optimize the y-axis **intercept**, we start by setting it to **0**.

Height = intercept + 0.64 x Weight

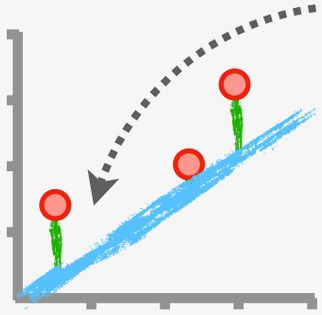
= 0 + 0.64 x Weight



$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,3.2 - (0 + 0.64 \times 2.9)\,)$$
$$+ -2 \times (\,1.9 - (0 + 0.64 \times 2.3)\,)$$
$$+ -2 \times (\,1.4 - (0 + 0.64 \times 0.5)\,)$$

Gradient Descent for One Parameter: Step-by-Step

3 Now evaluate the derivative at the current value for the **intercept**. In this case, the current value is **0**.

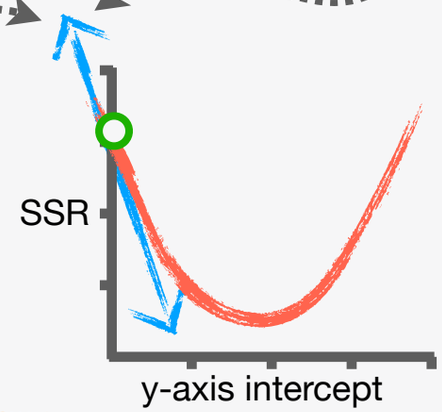


$$\frac{d\,SSR}{d\,intercept} = -2 \times (3.2 - (0 + 0.64 \times 2.9)) + -2 \times (1.9 - (0 + 0.64 \times 2.3)) + -2 \times (1.4 - (0 + 0.64 \times 0.5))$$

When we do the math, we get **-5.7**…

$$= -5.7$$

…thus, when the **intercept = 0**, the slope of this **tangent line** is **-5.7**.



4 Now calculate the **Step Size** with the following equation:

Step Size = **Derivative** x **Learning Rate**

= **-5.7** x **0.1**

= **-0.57**

Gentle Reminder: The magnitude of the derivative is proportional to how big of a step we should take toward the minimum. The sign (+/-) tells us which direction.

NOTE: The **Learning Rate** prevents us from taking steps that are too big and skipping past the lowest point in the curve. Typically, for **Gradient Descent**, the **Learning Rate** is determined automatically: it starts relatively large and gets smaller with every step taken. However, you can also use **Cross Validation** to determine a good value for the **Learning Rate**. In this case, we're setting the **Learning Rate** to **0.1**.

5 Take a step from the **current intercept** to get closer to the optimal value with the following equation:

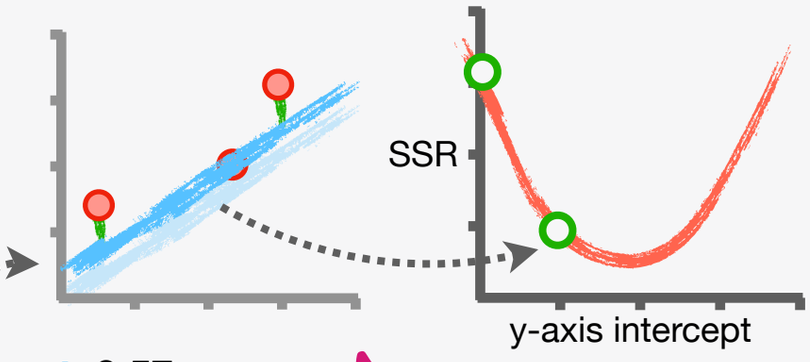
New intercept = **Current intercept** - **Step Size**

Remember, in this case, the **current intercept** is **0**.

= **0** - (**-0.57**)

= **0.57**

The **new intercept**, **0.57**, moves the line up a little closer to the data…



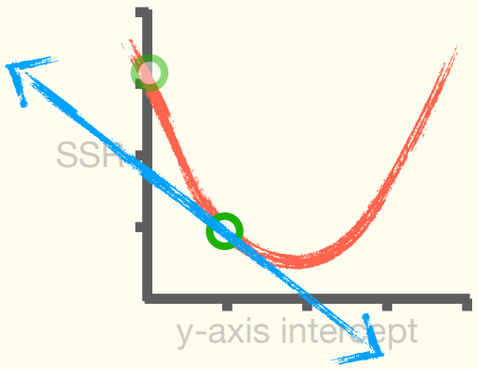
…and it results in a lower **SSR**. Bam!

Gradient Descent for One Parameter: Step-by-Step

6 Now repeat the previous three steps, updating the **intercept** after each iteration until the **Step Size** is close to **0** or we take the maximum number of steps, which is often set to **1,000** iterations.

a Evaluate the derivative at the current value for the intercept…

$$\frac{d\,SSR}{d\,intercept} = -2 \times (3.2 - (0.57 + 0.64 \times 2.9)) + -2 \times (1.9 - (0.57 + 0.64 \times 2.3)) + -2 \times (1.4 - (0.57 + 0.64 \times 0.5)) = -2.3$$



b Calculate the **Step Size**…

Step Size = **Derivative** x **Learning Rate**

= **-2.3** x **0.1**

= **-0.23**

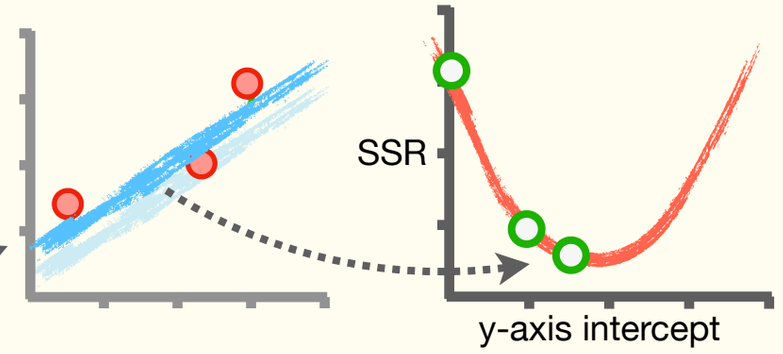
NOTE: The **Step Size** is smaller than before because the slope of the **tangent line** is not as steep as before. The smaller slope means we're getting closer to the optimal value.

c Calculate the **new intercept** value…

New intercept = **Current intercept** - **Step Size**

= **0.57** - (**-0.23**)

= **0.8**



The **new intercept**, **0.8**, moves the line up a little closer to the data…

…and it results in a lower **SSR**. **Double Bam!**

Gradient Descent for One Parameter: Step-by-Step

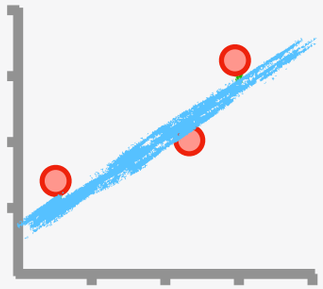
7 After **7** iterations…

a Evaluate the derivative at the current value…

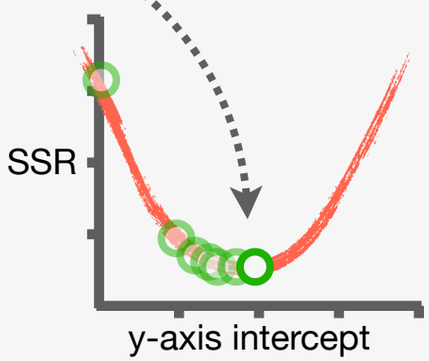
b Calculate the **Step Size**…

c Calculate the new value…

…the **Step Size** was very close to **0,** so we stopped with the **current intercept** = **0.95**…



…and we made it to the lowest **SSR**.



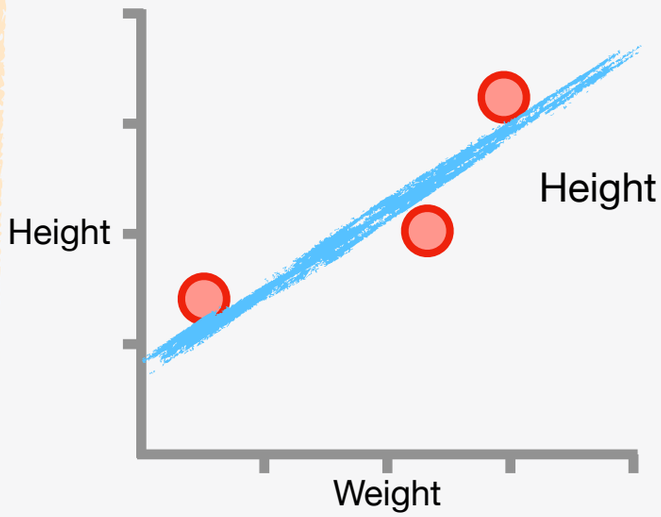
8 If, earlier on, instead of using **Gradient Descent**, we simply set the derivative to **0** and solved for the **intercept**, we would have gotten **0.95**, which is the same value that **Gradient Descent** gave us. Thus, **Gradient Descent** did a decent job.

BAM???

Not yet! Now let's see how well **Gradient Descent** optimizes the **intercept** *and* the **slope**!

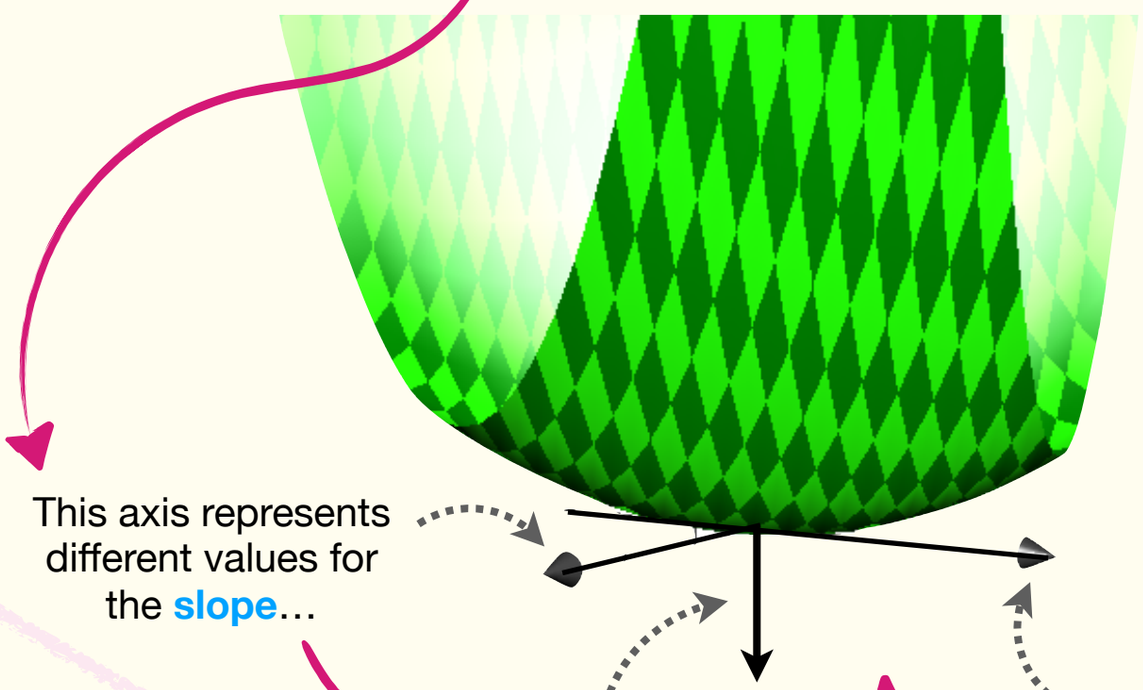
Optimizing Two or More Parameters: Details

1 Now that we know how to optimize the **intercept** of the line that minimizes the **SSR**, let's optimize both the **intercept** and the **slope**.

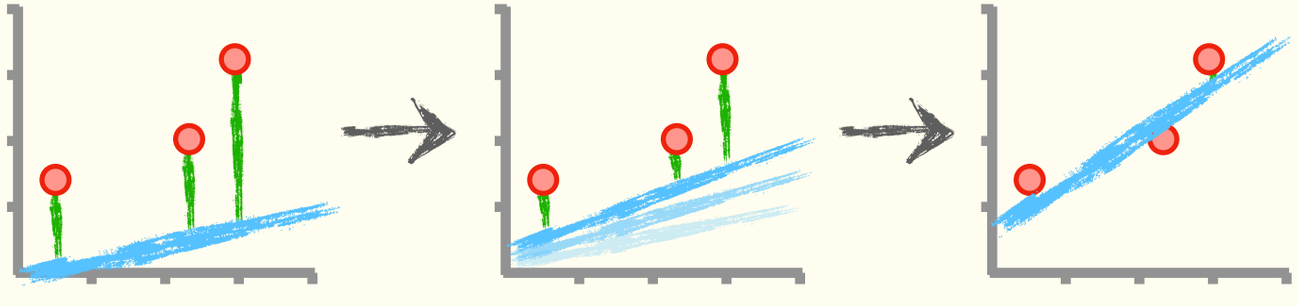


Height = **intercept** + **slope** x Weight

2 When we optimize two parameters, we get a 3-dimensional graph of the **SSR**.



3 Just like before, the goal is to find the parameter values that give us the lowest **SSR**. And just like before, **Gradient Descent** initializes the parameters with random values and then uses derivatives to update those parameters, one step at a time, until they're optimal.



4 So, now let's learn how to take derivatives of the **SSR** with respect to both the **intercept** and the **slope**.

SSR = (Height - (**intercept** + **slope** x Weight))^2

Taking Multiple (Partial) Derivatives of the SSR: Part 1

1 The good news is that taking the derivative of the **SSR** with respect to the **intercept** is exactly the same as before.

We can use **The Chain Rule** to tell us how the **SSR** changes with respect to the **intercept**.

$$SSR = (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight}))^2$$

Step 1: Create a *link* between the **intercept** and the **SSR** by rewriting the **SSR** as the function of the **Residual**.

$$SSR = (\textbf{Residual})^2 \qquad \textbf{Residual} = \text{Observed Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight})$$

Step 2: Because the **Residual** links the **intercept** to the **SSR**, **The Chain Rule** tells us that the derivative of the **SSR** with respect to the **intercept** is…

$$\frac{d\,SSR}{d\,intercept} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,intercept}$$

Because of the subtraction, we can remove the parentheses by multiplying everything inside by **-1**.

Step 3: Use **The Power Rule** to solve for the two derivatives.

$$\frac{d\,SSR}{d\,Residual} = \frac{d}{d\,Residual}(\textbf{Residual})^2 = 2 \times \textbf{Residual}$$

$$\frac{d\,Residual}{d\,intercept} = \frac{d}{d\,intercept}\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight})$$

$$= \frac{d}{d\,intercept}\text{Height} - \textbf{intercept} - \textbf{slope} \times \text{Weight}$$

$$= 0 - \textbf{1} - 0 = -1$$

Because the first and last terms do not include the **intercept**, their derivatives, with respect to the **intercept**, are both **0**. However, the second term is the negative **intercept**, so its derivative is **-1**.

Step 4: Plug the derivatives into **The Chain Rule** to get the final derivative of the **SSR** with respect to the **intercept**.

$$\frac{d\,SSR}{d\,intercept} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,intercept} = 2 \times \textbf{Residual} \times -1$$

$$= 2 \times (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight})) \times -1$$

Multiply this **-1** on the right by the **2** on the left to get **-2**.

$$= -2 \times (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight}))$$

BAM!!!

Taking Multiple (Partial) Derivatives of the SSR: Part 2

2 The other good news is that taking the derivative of the **SSR** with respect to the **slope** is very similar to what we just did for the **intercept**.

We can use **The Chain Rule** to tell us how the **SSR** changes with respect to the **slope**.

$$SSR = (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight}))^2$$

NOTE: A collection of derivatives of the same function but with respect to different parameters is called a **Gradient**, so this is where **Gradient Descent** gets its name from. We'll use the *gradient* to *descend* to the lowest **SSR**.

Step 1: Create a *link* between the **slope** and the **SSR** by rewriting the **SSR** as the function of the **Residual**.

$$SSR = (\textbf{Residual})^2 \qquad \textbf{Residual} = \text{Observed Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight})$$

Step 2: Because the **Residual** links the **slope** to the **SSR**, **The Chain Rule** tells us that the derivative of the **SSR** with respect to the **slope** is…

$$\frac{d\,SSR}{d\,slope} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,slope}$$

Because of the subtraction, we can remove the parentheses by multiplying everything inside by **-1**.

Step 3: Use **The Power Rule** to solve for the two derivatives.

$$\frac{d\,SSR}{d\,Residual} = \frac{d}{d\,Residual}(\textbf{Residual})^2 = 2 \times \textbf{Residual}$$

$$\frac{d\,Residual}{d\,slope} = \frac{d}{d\,slope}(\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight}))$$

$$= \frac{d}{d\,slope}(\text{Height} - \textbf{intercept} - \textbf{slope} \times \text{Weight})$$

$$= 0 - 0 - \text{Weight} = -\text{Weight}$$

Because the first and second terms do not include the **slope**, their derivatives, with respect to the **slope**, are both **0**. However, the last term is the negative **slope** times **Weight**, so its derivative is **-Weight**.

Step 4: Plug the derivatives into **The Chain Rule** to get the final derivative of the **SSR** with respect to the **slope**.

$$\frac{d\,SSR}{d\,slope} = \frac{d\,SSR}{d\,Residual} \times \frac{d\,Residual}{d\,slope} = 2 \times \textbf{Residual} \times -\text{Weight}$$

$$= 2 \times (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight})) \times -\text{Weight}$$

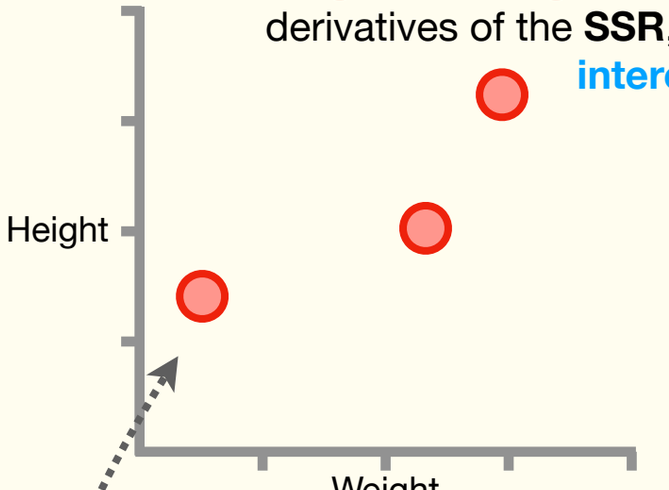
Multiply this **-Weight** on the right by the **2** on the left to get **-2 x Weight**.

$$= -2 \times \text{Weight} \times (\text{Height} - (\textbf{intercept} + \textbf{slope} \times \text{Weight}))$$

DOUBLE BAM!!!

Gradient Descent for Two Parameters: Step-by-Step

1 Plug the **Observed** values into the derivatives of the **Loss** or **Cost Function**. In this example, the **SSR** is the **Loss** or **Cost Function**, so we'll plug the **Observed Weight** and **Height** measurements into the two derivatives of the **SSR**, one with respect to the **intercept**…



….and one with respect to the **slope**.

$$\frac{d\,SSR}{d\,intercept} = -2 \times (\text{Height}_1 - (\text{intercept} + \text{slope} \times \text{Weight}_1)) + -2 \times (\text{Height}_2 - (\text{intercept} + \text{slope} \times \text{Weight}_2)) + -2 \times (\text{Height}_3 - (\text{intercept} + \text{slope} \times \text{Weight}_3))$$

$$\frac{d\,SSR}{d\,intercept} = -2 \times (3.2 - (\text{intercept} + \text{slope} \times 2.9)) + -2 \times (1.9 - (\text{intercept} + \text{slope} \times 2.3)) + -2 \times (1.4 - (\text{intercept} + \text{slope} \times 0.5))$$

$$\frac{d\,SSR}{d\,slope} = -2 \times \text{Weight}_1 \times (\text{Height}_1 - (\text{intercept} + \text{slope} \times \text{Weight}_1)) + -2 \times \text{Weight}_2 \times (\text{Height}_2 - (\text{intercept} + \text{slope} \times \text{Weight}_2)) + -2 \times \text{Weight}_3 \times (\text{Height}_3 - (\text{intercept} + \text{slope} \times \text{Weight}_3))$$

$$\frac{d\,SSR}{d\,slope} = -2 \times 2.9 \times (3.2 - (\text{intercept} + \text{slope} \times 2.9)) + -2 \times 2.3 \times (1.9 - (\text{intercept} + \text{slope} \times 2.3)) + -2 \times 0.5 \times (1.4 - (\text{intercept} + \text{slope} \times 0.5))$$

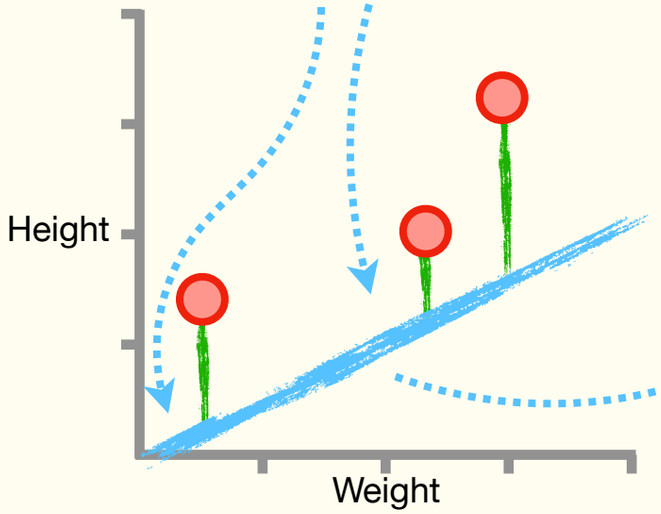
Gentle Reminder: The Weight and Height values that we're plugging into the derivatives come from the raw data in the graph.

Gradient Descent for Two Parameters: Step-by-Step

2 Now initialize the parameter, or parameters, that we want to optimize with random values. In this example, we'll set the **intercept** to **0** and the **slope** to **0.5**.

Height = **intercept** + **slope** x Weight

Height = **0** + **0.5** x Weight



$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,3.2 - (\text{intercept} + \text{slope} \times 2.9)\,)$$
$$+ -2 \times (\,1.9 - (\text{intercept} + \text{slope} \times 2.3)\,)$$
$$+ -2 \times (\,1.4 - (\text{intercept} + \text{slope} \times 0.5)\,)$$

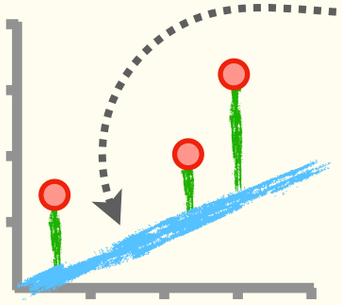
$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,3.2 - (0 + 0.5 \times 2.9)\,)$$
$$+ -2 \times (\,1.9 - (0 + 0.5 \times 2.3)\,)$$
$$+ -2 \times (\,1.4 - (0 + 0.5 \times 0.5)\,)$$

$$\frac{d\,SSR}{d\,slope} = -2 \times 2.9 \times (\,3.2 - (\text{intercept} + \text{slope} \times 2.9)\,)$$
$$+ -2 \times 2.3 \times (\,1.9 - (\text{intercept} + \text{slope} \times 2.3)\,)$$
$$+ -2 \times 0.5 \times (\,1.4 - (\text{intercept} + \text{slope} \times 0.5)\,)$$

$$\frac{d\,SSR}{d\,slope} = -2 \times 2.9 \times (\,3.2 - (0 + 0.5 \times 2.9)\,)$$
$$+ -2 \times 2.3 \times (\,1.9 - (0 + 0.5 \times 2.3)\,)$$
$$+ -2 \times 0.5 \times (\,1.4 - (0 + 0.5 \times 0.5)\,)$$

Gradient Descent for Two Parameters: Step-by-Step

3 Evaluate the derivatives at the current values for the **intercept**, **0**, and **slope**, **0.5**.



$$\frac{d\,SSR}{d\,intercept} = -2 \times (3.2 - (0 + 0.5 \times 2.9)) + -2 \times (1.9 - (0 + 0.5 \times 2.3)) + -2 \times (1.4 - (0 + 0.5 \times 0.5)) = -7.3$$

$$\frac{d\,SSR}{d\,slope} = -2 \times 2.9 \times (3.2 - (0 + 0.5 \times 2.9)) + -2 \times 1.9 \times (2.3 - (0 + 0.5 \times 1.9)) + -2 \times 0.5 \times (1.4 - (0 + 0.5 \times 0.5)) = -14.8$$

4 Calculate the **Step Sizes**: one for the **intercept**…

…and one for the **slope**.

Step Size$_{Intercept}$ = **Derivative** x **Learning Rate**

= **-7.3** x **0.01**

= **-0.073**

Step Size$_{Slope}$ = **Derivative** x **Learning Rate**

= **-14.8** x **0.01**

= **-0.148**

NOTE: We're using a smaller **Learning Rate** now (**0.01**) than before (**0.1**) because **Gradient Descent** can be very sensitive to it. However, as we said earlier, usually the **Learning Rate** is determined automatically.

5 Take a step from the **current intercept**, **0**, and **slope**, **0.5**, to get closer to the optimal values…

New intercept = **Current intercept** - **Step Size**$_{Intercept}$

= **0** - (**-0.073**)

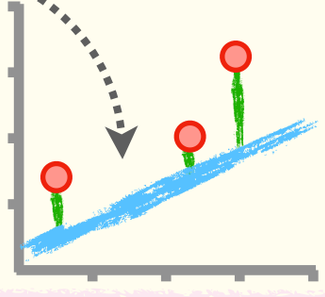
= **0.073**

…and the **intercept** increases from **0** to **0.073**, the **slope** increases from **0.5** to **0.648**, and the **SSR** decreases. **BAM**!

New slope = **Current slope** - **Step Size**$_{Slope}$

= **0.5** - (**-0.148**)

= **0.648**



Gradient Descent for Two Parameters: Step-by-Step

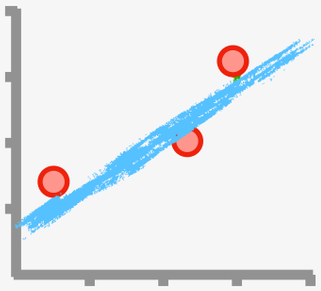
6 And after **475** iterations…

a Evaluate the derivatives at their current values…

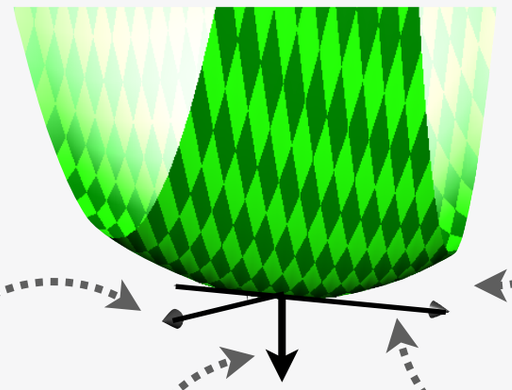
b Calculate the **Step Sizes**…

c Calculate the new values…

…the **Step Size** was very close to **0**, so we stopped with the **current intercept** = **0.95** and the **current slope** = **0.64**…



…and we made it to the lowest **SSR**.



This axis represents different values for the **slope**…

…this axis is for the **SSR**…

…and this axis represents different values for the **intercept**.

7 If, earlier on, instead of using **Gradient Descent**, we simply set the derivatives to **0** and solved for the **intercept** and **slope**, we would have gotten **0.95** and **0.64**, which are the same values **Gradient Descent** gave us. Thus, **Gradient Descent** did a great job, and we can confidently use it in situations where there are no analytical solutions, like **Logistic Regression** and **Neural Networks**.

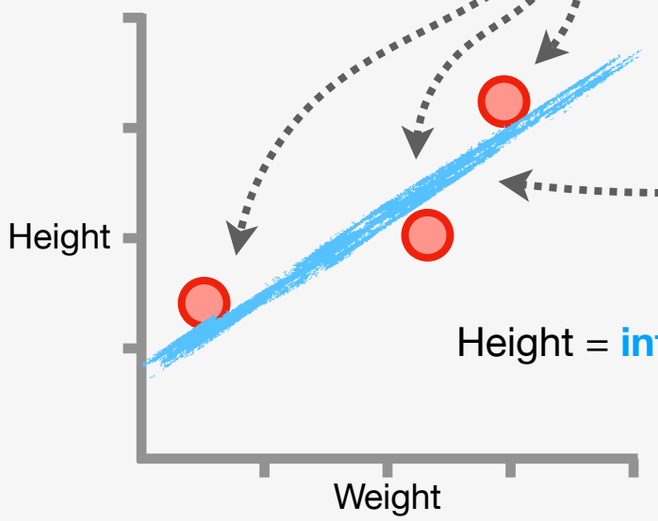
TRIPLE BAM!!!

Gradient Descent is awesome, but when we have a lot of data or a lot of parameters, it can be slow. Is there any way to make it faster?

YES! Read on to learn about **Stochastic Gradient Descent**.

Stochastic Gradient Descent: Main Ideas

1 So far, things have been pretty simple. We started out with a tiny dataset that only had **3** points…



…and we fit a straight **line**, which only has **2** parameters, the **intercept** and the **slope**.

Height = **intercept** + **slope** x Weight

2 Because there were only **2** parameters, we only had **2** derivatives that we had to calculate in each step…

$$\frac{d\,SSR}{d\,intercept} \qquad \frac{d\,SSR}{d\,slope}$$

…and because we only had **3** data points, each derivative only needed to compute **3** terms per derivative.

$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,3.2 - (intercept + slope \times 2.9)\,) + -2 \times (\,1.9 - (intercept + slope \times 2.3)\,) + -2 \times (\,1.4 - (intercept + slope \times 0.5)\,)$$

$$\frac{d\,SSR}{d\,slope} = -2 \times 2.9 \times (\,3.2 - (intercept + slope \times 2.9)\,) + -2 \times 2.3 \times (\,1.9 - (intercept + slope \times 2.3)\,) + -2 \times 0.5 \times (\,1.4 - (intercept + slope \times 0.5)\,)$$

3 However, what if we had **1,000,000** data points? Then we would have to compute **1,000,000** terms per derivative.

Ugh!

And what if we had a more complicated model with **10,000** parameters? Then we would have **10,000** derivatives to compute.

Double Ugh!

Taking **10,000** derivatives, each with **1,000,000** terms to compute, is a lot of work, and all of that work only gets us one step into the process that can take **1,000s** of steps!!!

TRIPLE UGH!

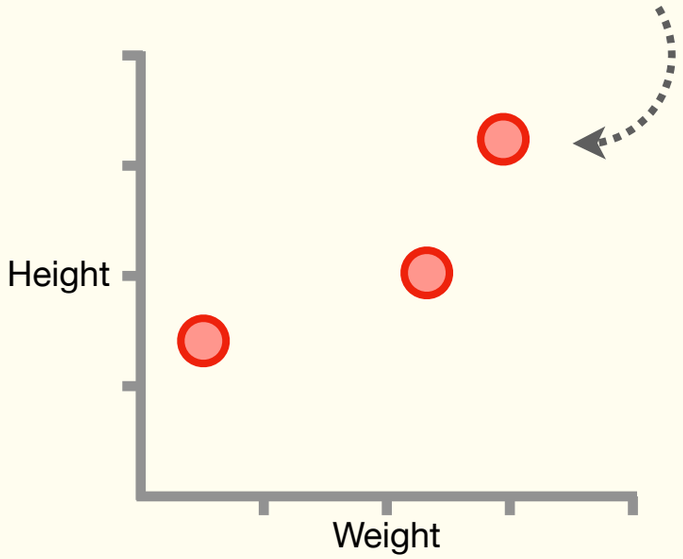
Thus, for **BIG DATA**, **Gradient Descent** requires a lot of computation and can be slow.

4 The good news is that **Stochastic Gradient Descent** can drastically reduce the amount of computation required to optimize parameters. Although it sounds fancy, the word **Stochastic** just means **Randomly Determined** and all **Stochastic Gradient Descent** does is randomly select one data point per step. So, regardless of how large your dataset is, only one term is computed per derivative for each iteration.

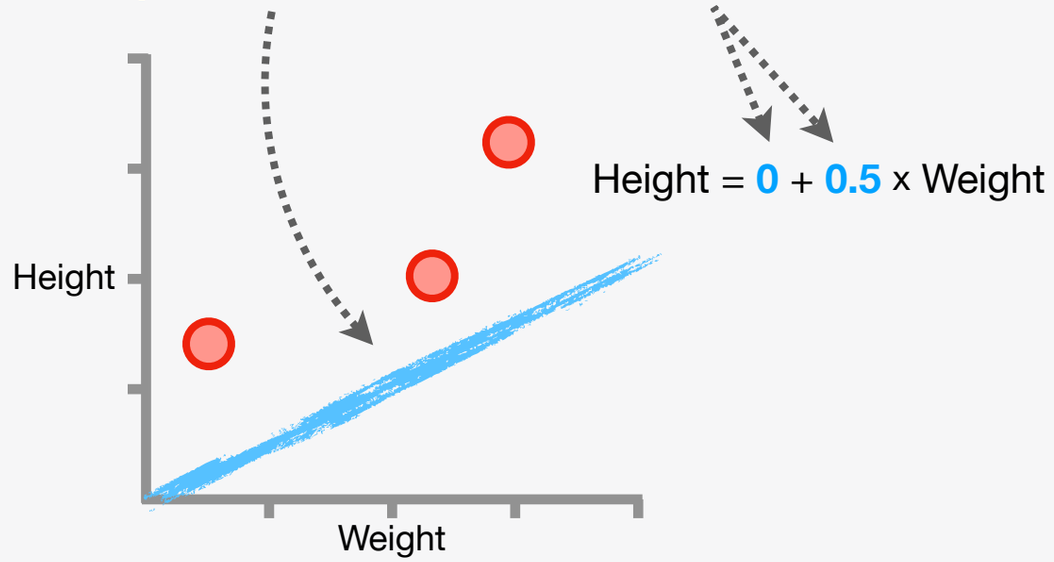
BAM!

Stochastic Gradient Descent: Details Part 1

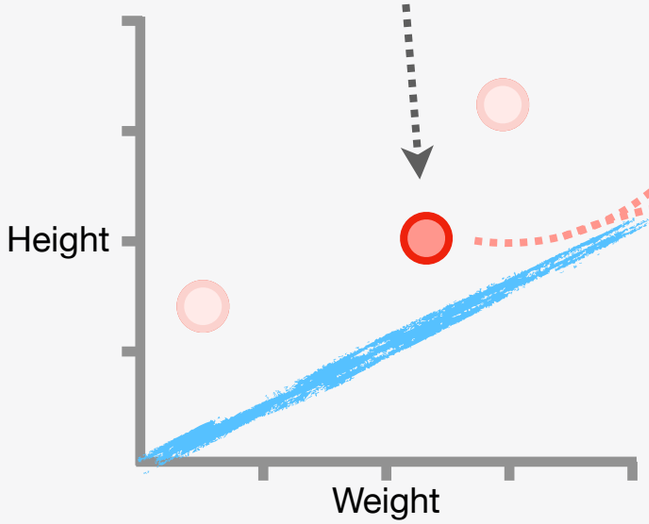
1 To see how **Stochastic Gradient Descent** works, let's go back to our simple example, where we want to fit a line to **3** data points.



2 And just like with normal **Gradient Descent**, we start by initializing the **intercept** and **slope** of the line with random values.



3 Now we randomly pick one point. In this case, we'll pick this one in the middle.



4 Then, we evaluate the derivatives using just that single point…

$$\frac{d\,SSR}{d\,intercept} = -2 \times (\,Height - (intercept + slope \times Weight)\,)$$

$$\frac{d\,SSR}{d\,slope} = -2 \times Weight \times (\,Height - (intercept + slope \times Weight)\,)$$

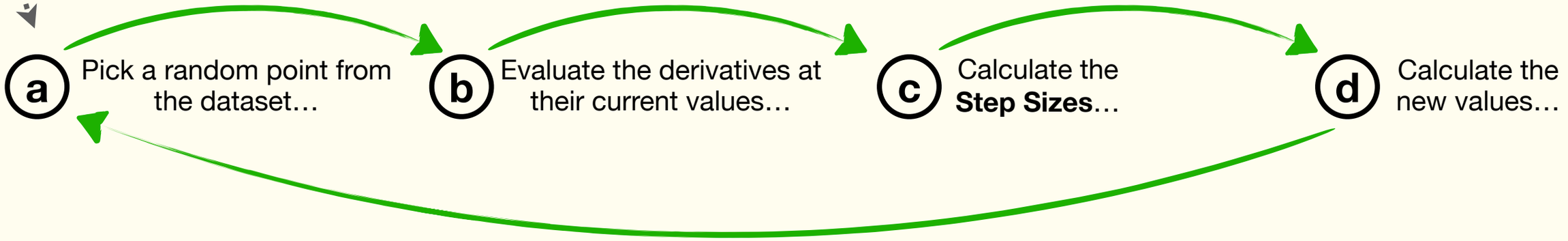
5 …and then we calculate the **Step Sizes**…

6 …and then we calculate the new values.

Stochastic Gradient Descent: Details Part 2

7 Then we just repeat the last **4** steps until the **Step Sizes** are super small, which suggests we've optimized the parameters, or until we reach a maximum number of steps.

BAM!

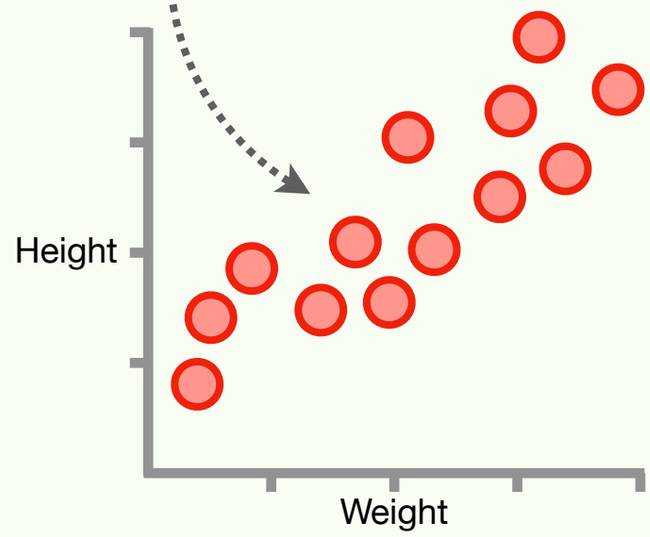


8

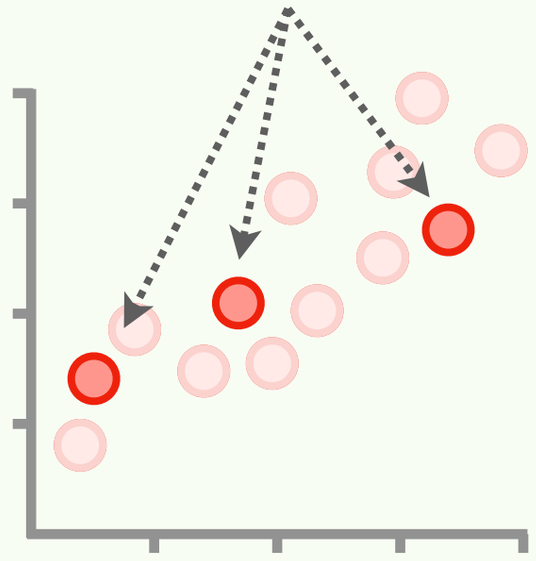
TERMINOLOGY ALERT!!!

Although a strict definition of **Stochastic Gradient Descent** says that we only select a single point per iteration, it's much more common to randomly select a small subset of the observations. This is called **Mini-Batch Stochastic Gradient Descent**. Using a small subset, rather than a single point, usually converges on the optimal values in fewer steps and takes much less time than using all of the data.

9 For example, if we had these data and wanted to use **Mini-Batch Stochastic Gradient Descent**…



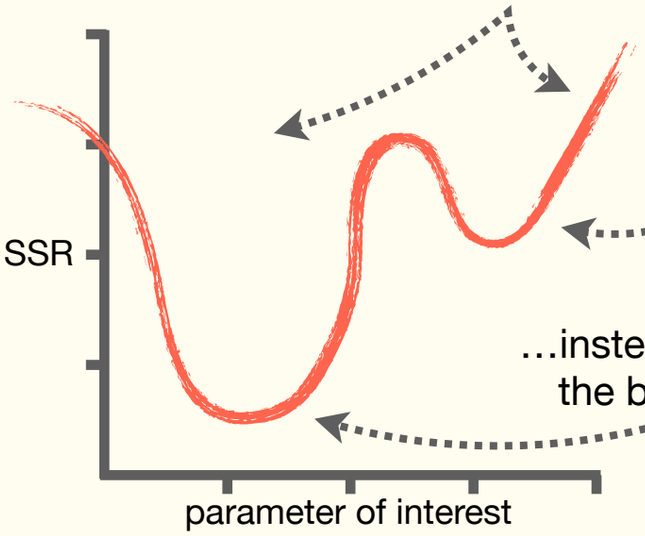
…then instead of randomly selecting one point per iteration, we might select **3** points.



Gradient Descent: FAQ

Will Gradient Descent always find the best parameter values?

Unfortunately, **Gradient Descent** does not always find the best parameter values. For example, if the graph of the **SSR** looked like this….



…then it's possible that we might get stuck at the bottom of this **local minimum**…

…instead of finding our way to the bottom and the **global minimum**.

When this happens (when we get stuck in a local minimum instead of finding the global minimum), it's a bummer. Even worse, usually it's not possible to graph the **SSR**, so we might not even know we're in one, of potentially many, local minimums. However, there are a few things we can do about it. We can:

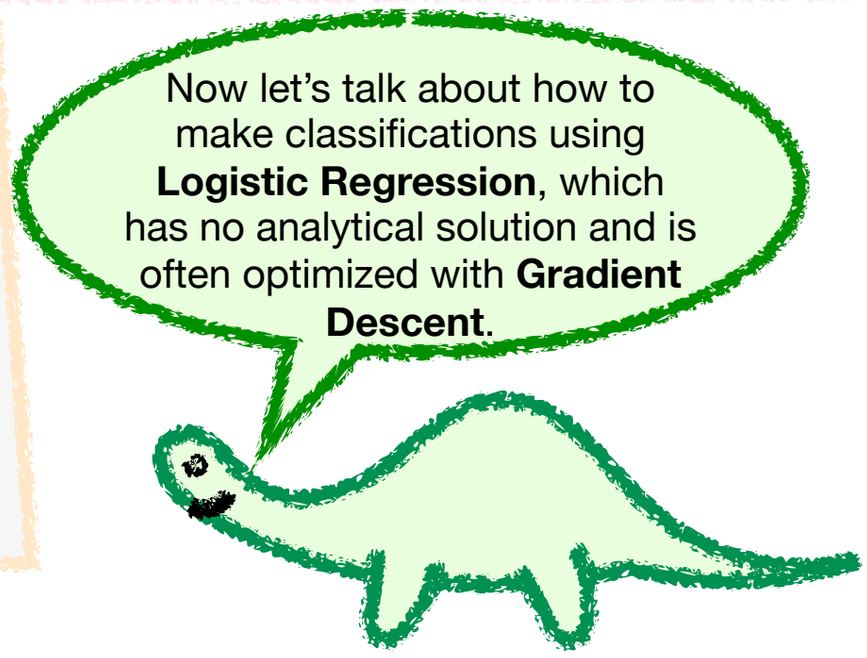
1) Try again using different random numbers to initialize the parameters that we want to optimize. Starting with different values may avoid a local minimum.

2) Fiddle around with the **Step Size**. Making it a little larger may help avoid getting stuck in a local minimum.

3) Use **Stochastic Gradient Descent**, because the extra randomness helps avoid getting trapped in a local minimum.

How do you choose the size of a Mini-Batch for Stochastic Gradient Descent?

The answer to this question really depends on the computer hardware you're using to train (optimize) your model. For example, because one of the main reasons we use **Mini-Batch Stochastic Gradient Descent** is to train our model as quickly as possible, one major consideration is how much high-speed memory we have access to. The more high-speed memory we have, the larger the **Mini-Batch** can be.

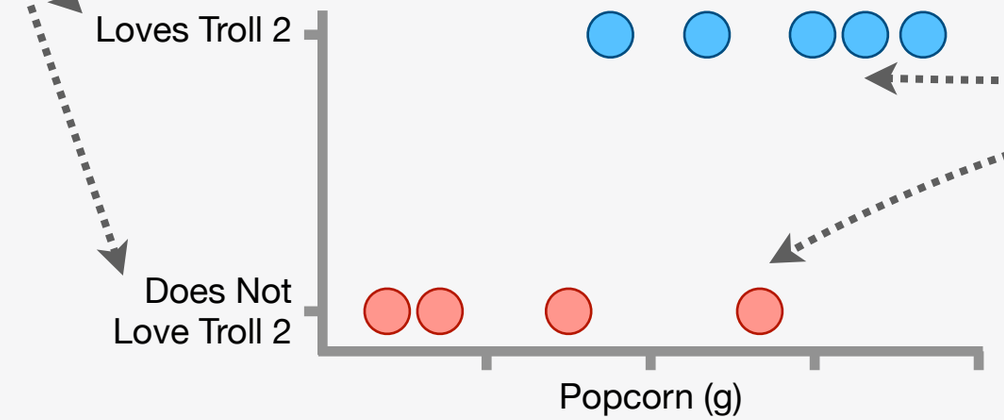


Chapter 06

Logistic Regression!!!

Logistic Regression: Main Ideas Part 1

1 **The Problem: Linear Regression** and **Linear Models** are great when we want to predict something that is ***continuous***, like **Height**, but what if we want to classify something ***discrete*** that only has two possibilities, like whether or not someone loves the movie Troll 2?



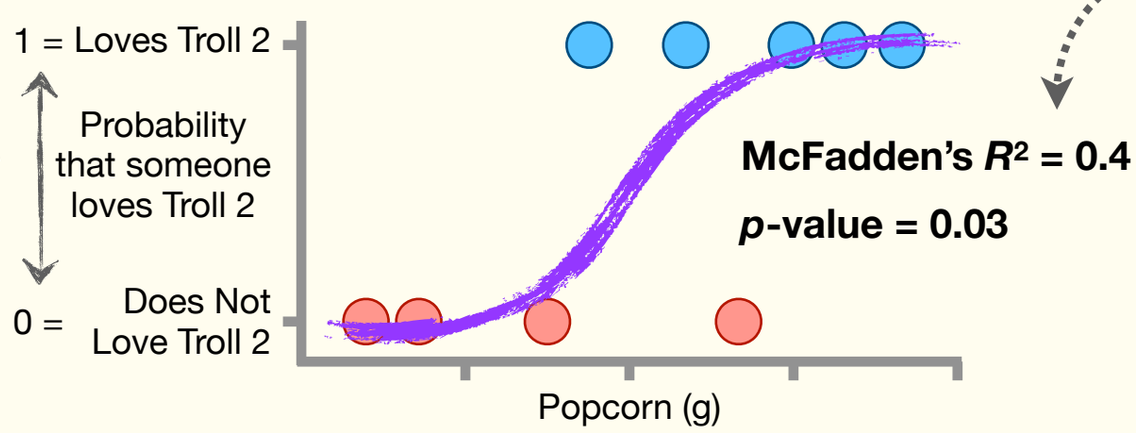
In this example, we measured the amount of Popcorn a bunch of people ate (in grams), which is ***continuous***, and whether they **Love Troll 2** or **Do Not Love Troll 2**, which is ***discrete***.

= Loves Troll 2

= Does Not Love Troll 2

The goal is to make a ***classifier*** that uses the amount of Popcorn someone eats to classify whether or not they love Troll 2.

2 **A Solution: Logistic Regression**, which probably should have been named **Logistic Classification** since it's used to classify things, fits a **squiggle** to data that tells us the predicted probability (between **0** and **1**) for ***discrete*** variables, like whether or not someone loves Troll 2.



Like **Linear Regression**, **Logistic Regression** has metrics that are similar to R^2 to give us a sense of how accurate our predictions will be, and it also calculates ***p*-values**.

Even better, all of the tricks we can do with **Linear Models** also apply to **Logistic Regression**, so we can mix and match ***discrete*** and ***continuous*** features to make ***discrete*** classifications.

BAM!!!

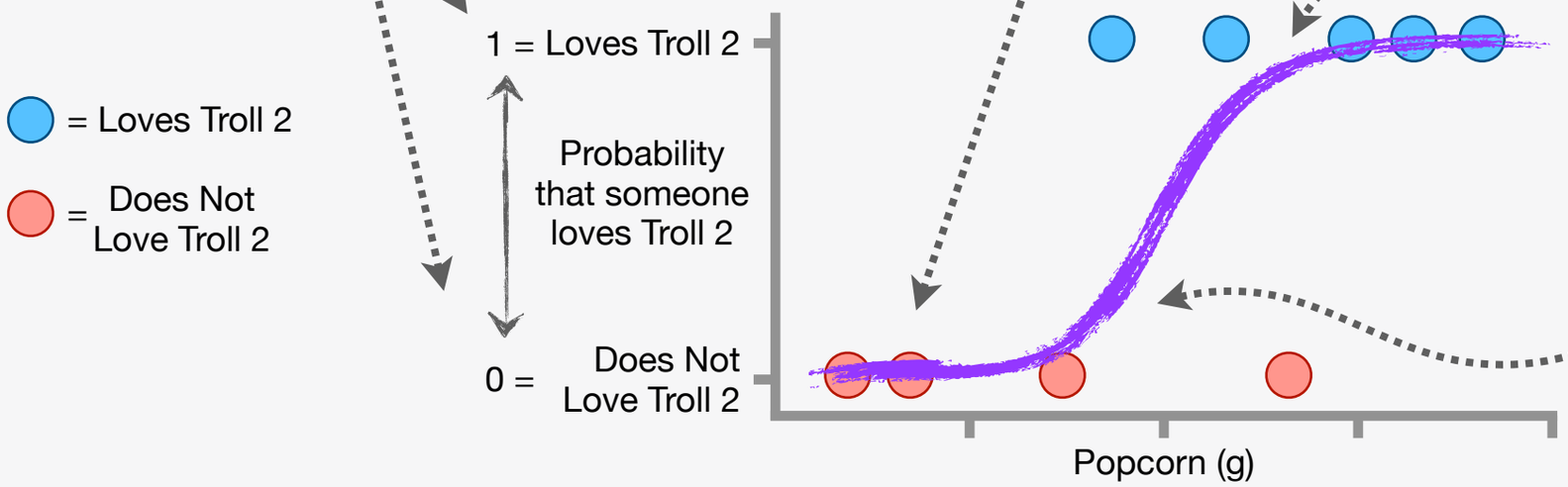
Logistic Regression: Main Ideas Part 2

3 The y-axis on a **Logistic Regression** graph represents probability and goes from **0** to **1**. In this case, it's the probability that someone loves Troll 2.

The colored dots are the **Training Data**, which we used to fit the squiggle. The data consist of **4** people who **Do Not Love Troll 2** and **5** people who **Love Troll 2**.

The squiggle tells us the predicted probability that someone loves Troll 2, which means that when the squiggle is close to the top of the graph, there's a high probability (a probability close to **1**) that someone will love Troll 2…

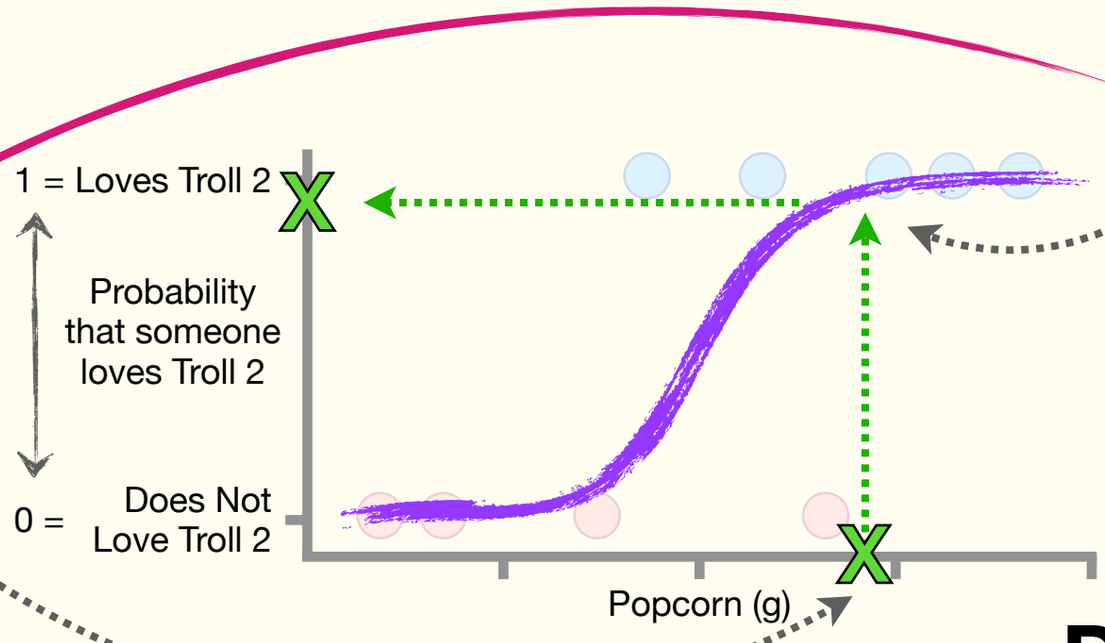
…and when the squiggle is close to the bottom of the graph, there's a low probability (a probability close to **0**) that someone will love Troll 2.



4 If someone new comes along and tells us that they ate this much Popcorn…

…then the squiggle tells us that there's a relatively high probability that that person will love Troll 2.

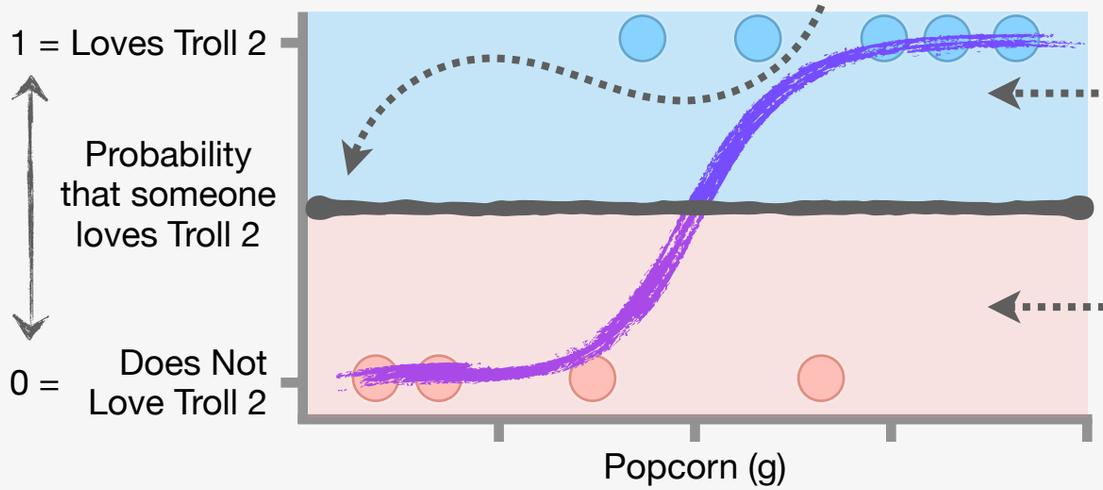
Specifically, the corresponding y-axis value on the squiggle tells us that the probability that this person loves Troll 2 is **0.96**.



DOUBLE BAM!!!

Logistic Regression: Main Ideas Part 3

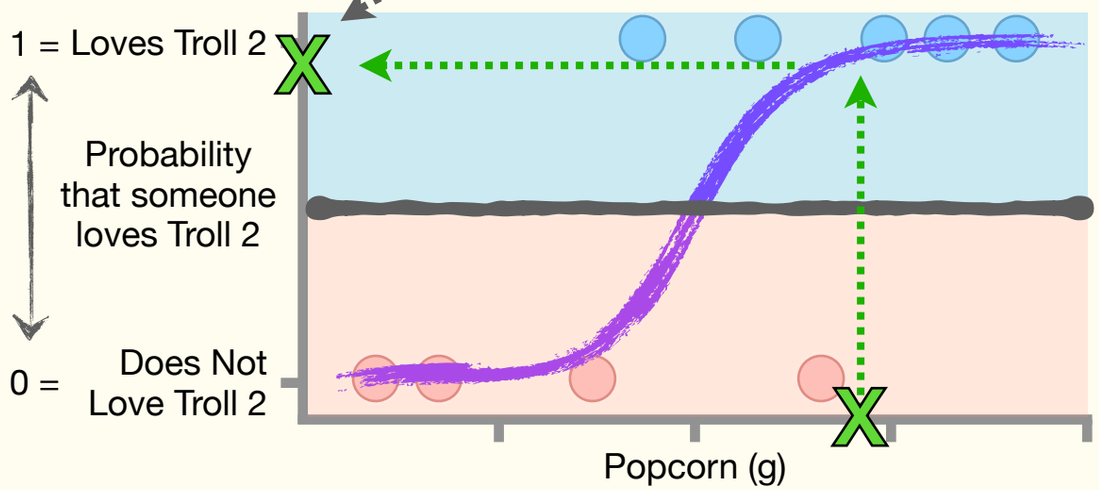
5 Now that we know the *probability* that this person will love Troll 2, we can *classify* them as someone who either **Loves Troll 2** or **Does Not Love Troll 2**. Usually the threshold for classification is **0.5**…



…and in this case, that means anyone with a probability of loving Troll 2 **> 0.5** will be classified as someone who **Loves Troll 2**…

….and anyone with a probability **≤ 0.5** will be classified as someone who **Does Not Love Troll 2**.

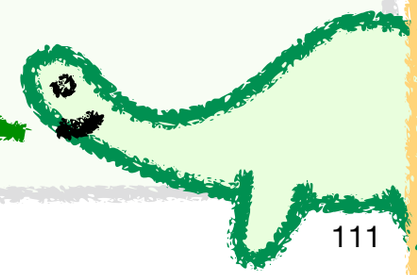
6 Thus, in this example, since **0.96 > 0.5**, we'll classify this person as someone who **Loves Troll 2**.



7 One last thing before we go: In this example, the classification threshold was **50%**. However, when we talk about **Receiver Operator Curves** (**ROCs**) in **Chapter 8**, we'll see examples that use different classification thresholds. So get excited!!!

TRIPLE BAM!!!

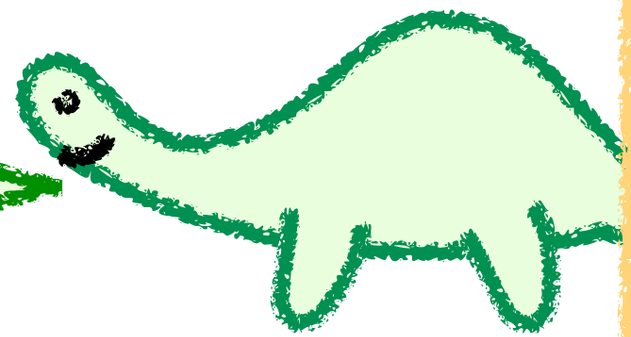
In a few pages, we'll talk about how we fit a **squiggle** to **Training Data**. However, before we do that, we need to learn some fancy terminology.



Terminology Alert!!! Probability vs. Likelihood: Part 1

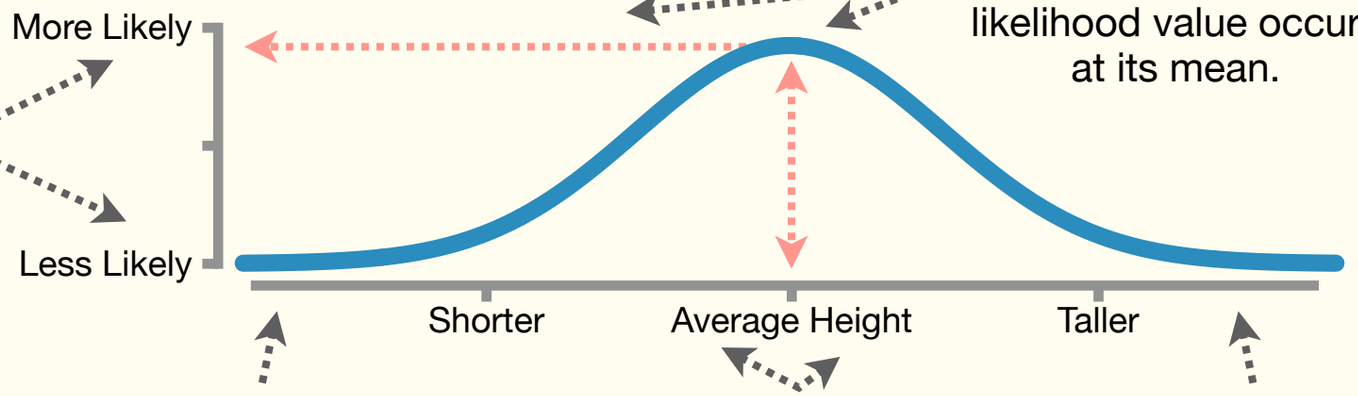
Hey **Norm**, don't the words **Probability** and **Likelihood** mean the same thing?

In casual conversations, yes, we can use **Probability** and **Likelihood** interchangeably. But unfortunately, in the context of statistics, these terms have different uses and, in some cases, completely different meanings.



1 Way back in **Chapter 3**, when we described the **Normal Distribution**, we saw that the y-axis represented **Likelihood**.

In this specific example, the y-axis represents the **Likelihood** of observing any specific Height.



The **Normal Distribution**'s maximum likelihood value occurs at its mean.

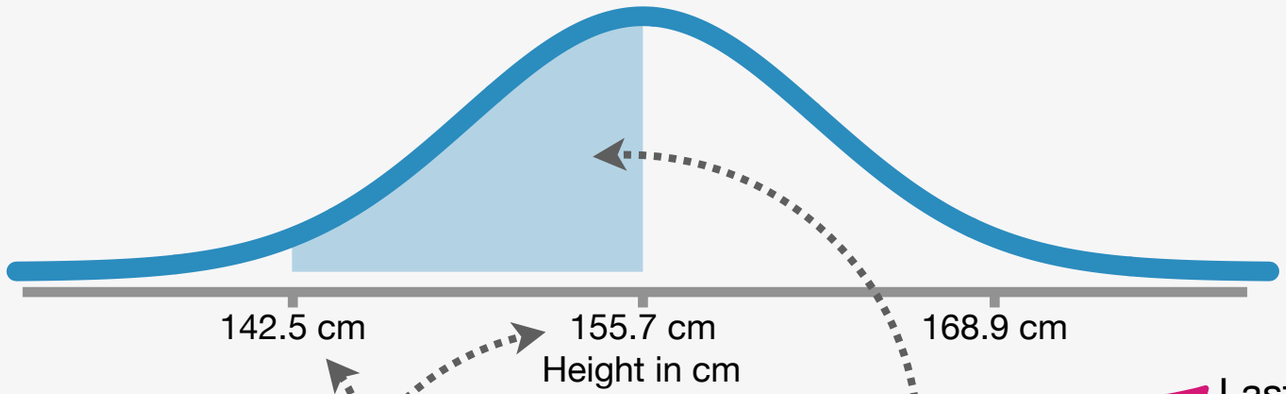
For example, it's relatively rare to see someone who is super short…

…relatively common to see someone who is close to the average height…

…and relatively rare to see someone who is super tall.

Terminology Alert!!! Probability vs. Likelihood: Part 2

2 In contrast, later in **Chapter 3**, we saw that **Probabilities** are derived from a **Normal Distribution** by calculating the **area under the curve** between two points.



For example, given this **Normal Distribution** with **mean = 155.7** and **standard deviation = 6.6**, the probability of getting a measurement between **142.5** and **155.7** cm…

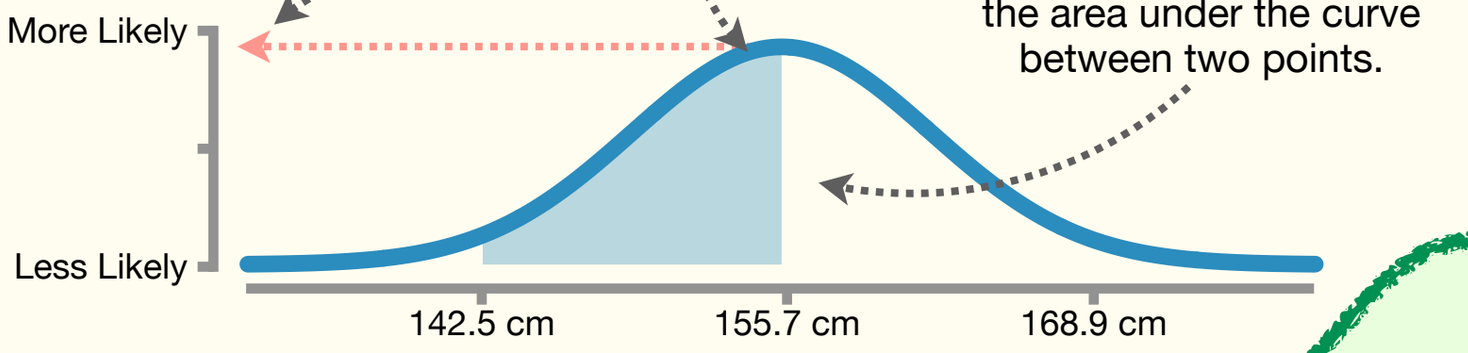
…is equal to this area under the curve, which, in this example, is **0.48**. So, the probability is **0.48** that we will measure someone in this range.

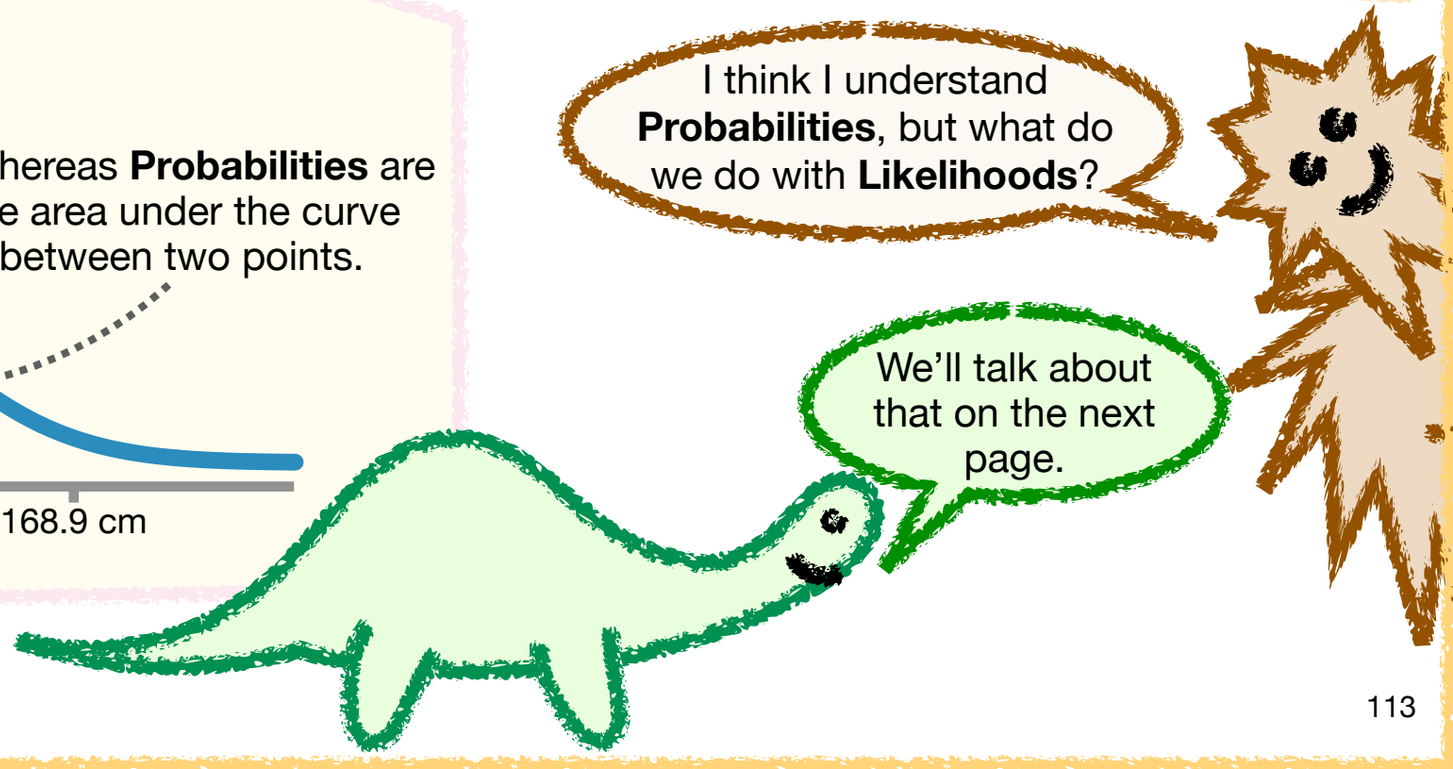
Lastly, in **Chapter 3** we mentioned that when we use a **Continuous Distribution**, like the **Normal Distribution**, the probability of getting any specific measurement is always **0** because the area of something with no width is **0**.

3 So, in the case of the **Normal Distribution**…

…**Likelihoods** are the y-axis coordinates for specific points on the curve…

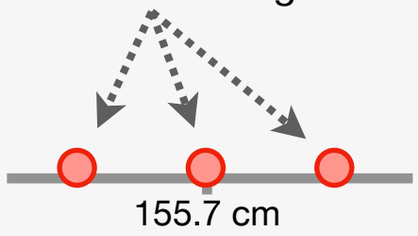
…whereas **Probabilities** are the area under the curve between two points.



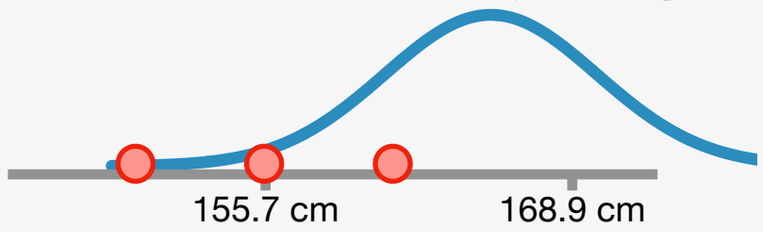


Terminology Alert!!! Probability vs. Likelihood: Part 3

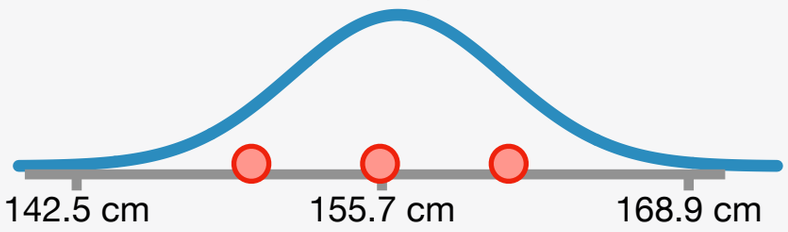
4 **Likelihoods** are often used to evaluate how well a statistical distribution fits a dataset. For example, imagine we collected these three Height measurements…



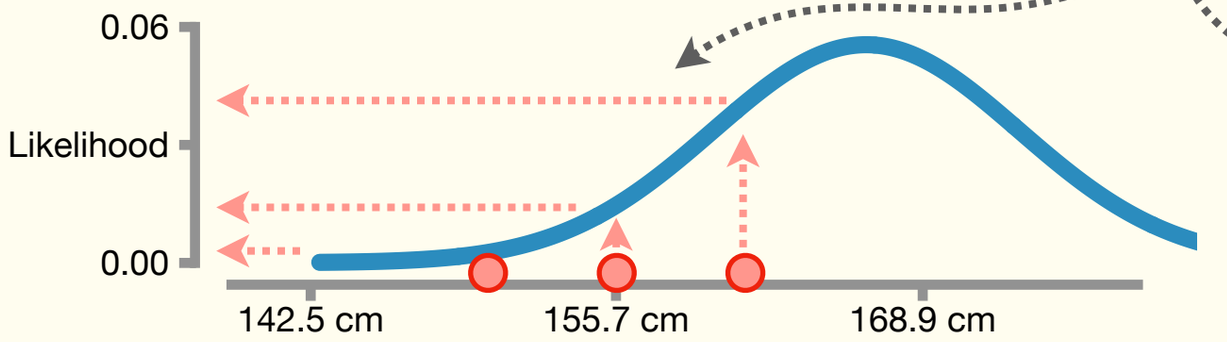
…and we wanted to compare the fit of a **Normal Curve** that has its peak to the right of the data…



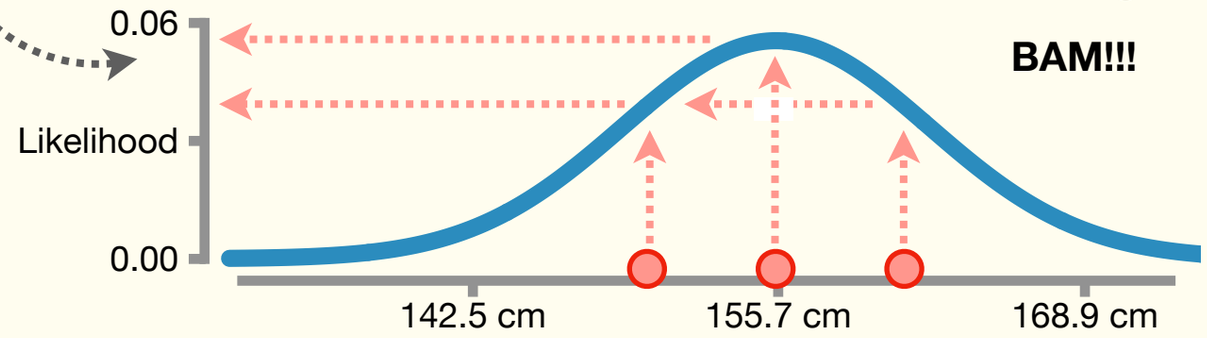
…to the fit of a **Normal Curve** that is centered over the data.



5 First, we determine the **Likelihoods**, the y-axis coordinates on the curves, for each data point…



…and, by eye, we can see that, overall, the **Likelihoods** are larger when we center the curve over the data.



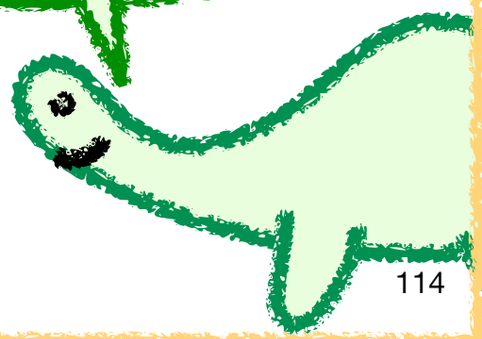
And larger likelihoods suggest that the centered curve fits the data better than the one shifted to the right.

BAM!!!

6 **NOTE:** When we try to fit a **Normal Curve** to data, we can't use **Probabilities** because, as we saw in **Chapter 3**, the probability for a specific point under a **Normal Curve** is always **0**.

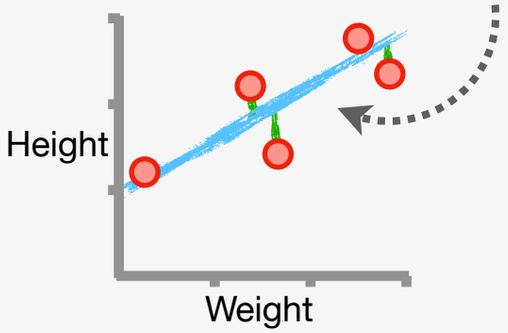
7 Lastly, as we just saw with the **Normal Distribution**, **Probabilities** and **Likelihoods** can be different. However, as we'll soon see, this is not always the case. In other words, sometimes **Probabilities** and **Likelihoods** are the same. When **Probabilities** and **Likelihoods** are the same, we could use either **Probabilities** or **Likelihoods** to fit a curve or a squiggle to data. However, to make the terminology consistent, when we're fitting a curve or squiggle to data in a statistical context, we almost always use **Likelihoods**.

Now that we know how to use **Likelihoods** to fit curves, let's talk about the main ideas of fitting a squiggle to data for **Logistic Regression**.

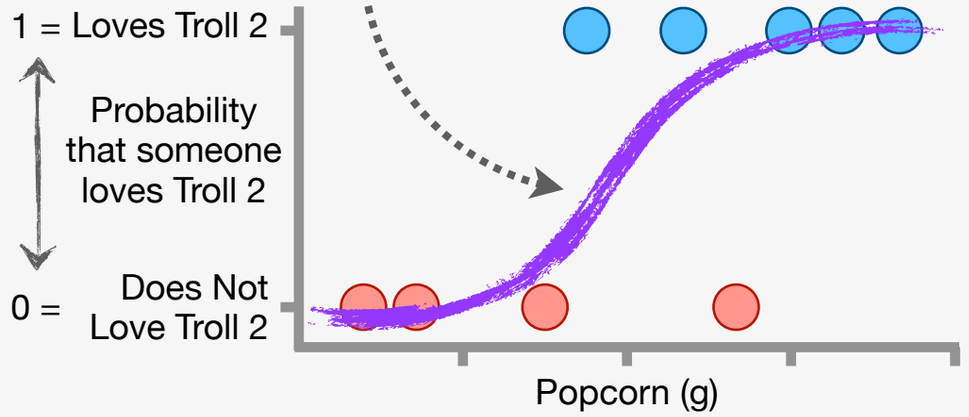


Fitting A Squiggle To Data: Main Ideas Part 1

1 When we use **Linear Regression**, we fit a **line** to the data by minimizing the **Sum of the Squared Residuals** (**SSR**).



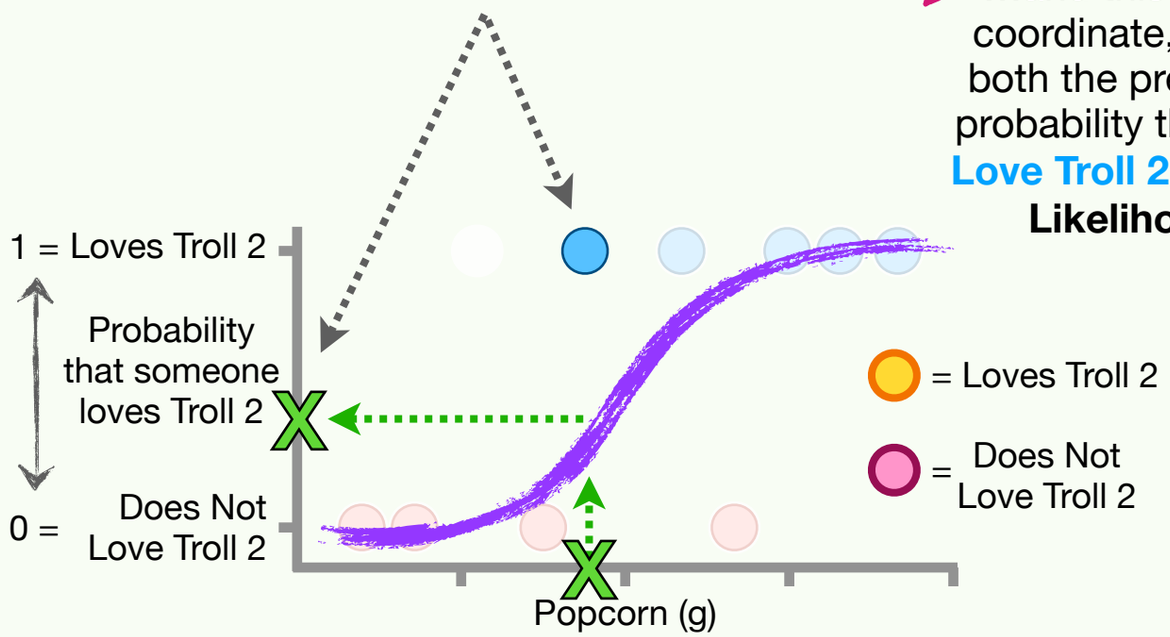
In contrast, **Logistic Regression** swaps out the **Residuals** for **Likelihoods** (y-axis coordinates) and fits a **squiggle** that represents the **Maximum Likelihood**.



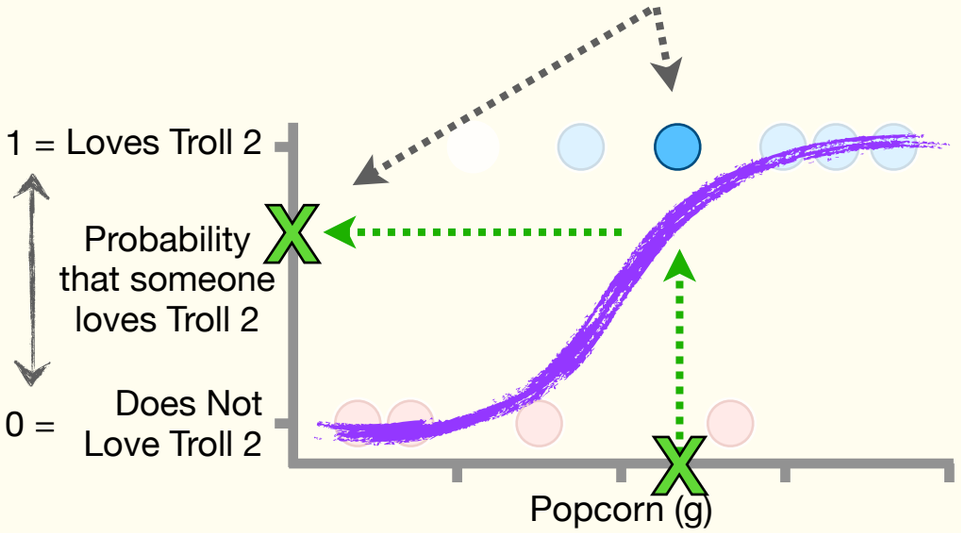
2 However, because we have two classes of people, one that **Loves Troll 2** and one that **Does Not Love Troll 2**, there are two ways to calculate **Likelihoods**, one for each class.

3 For example, to calculate the **Likelihood** for this person, who **Loves Troll 2**, we use the **squiggle** to find the y-axis coordinate that corresponds to the amount of Popcorn they ate…

…and this y-axis coordinate, **0.4**, is both the predicted probability that they **Love Troll 2** and the **Likelihood**.

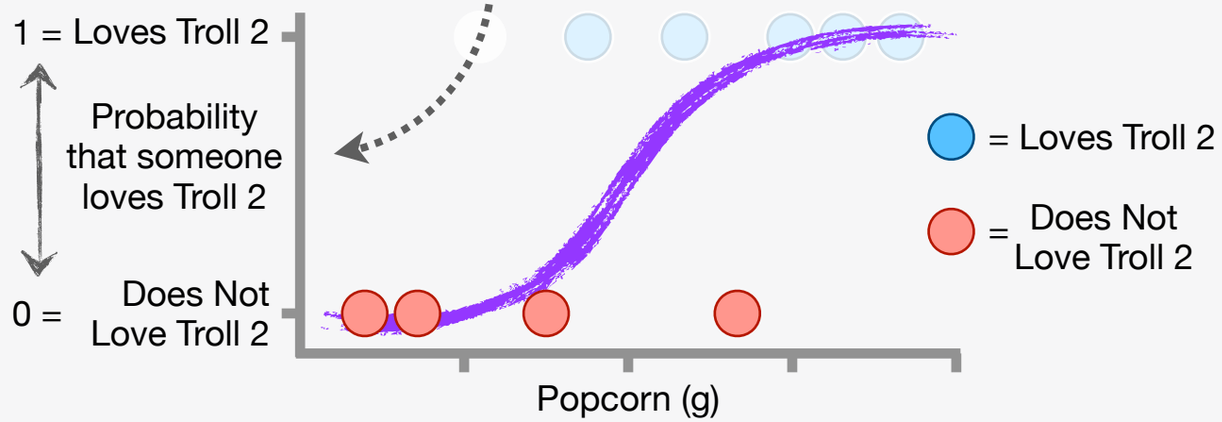


4 Likewise, the **Likelihood** for this person, who also **Loves Troll 2**, is the y-axis coordinate for the **squiggle**, **0.6**, that corresponds to the amount of Popcorn they ate.



Fitting A Squiggle To Data: Main Ideas Part 2

5 In contrast, calculating the **Likelihoods** is different for the people who **Do Not Love Troll 2** because the y-axis is the probability that they **Love Troll 2**.



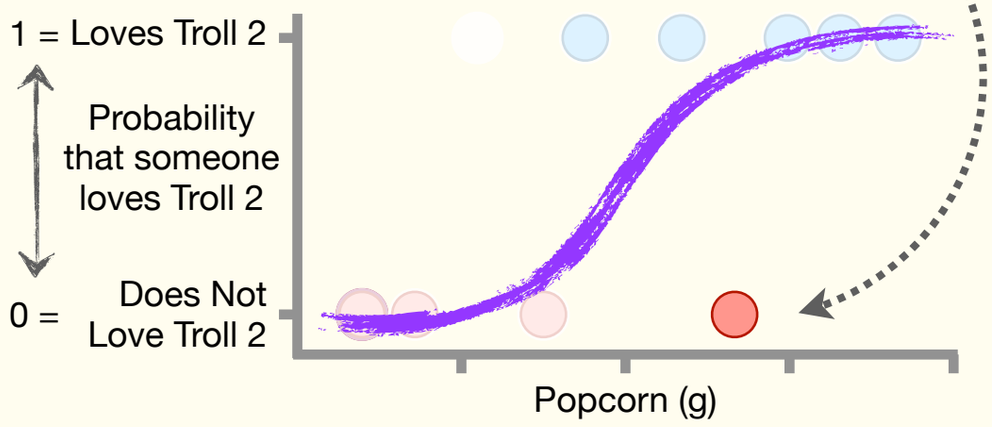
The good news is, because someone either loves Troll 2 or they don't, the probability that someone does not love Troll 2 is just **1** minus the probability that they love Troll 2…

p(**Does Not Love Troll 2**) = 1 - p(**Loves Troll 2**)

…and since the y-axis is both probability and likelihood, we can calculate the **Likelihoods** with this equation.

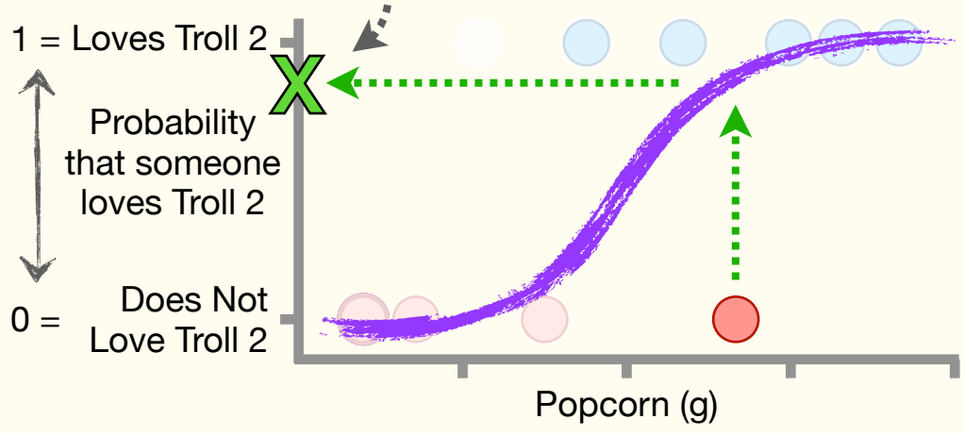
L(**Does Not Love Troll 2**) = 1 - L(**Loves Troll 2**)

6 For example, to calculate the **Likelihood** for this person, who **Does Not Love Troll 2**…



…we first calculate the **Likelihood** that they **Love Troll 2**, **0.8**…

…and then use that value to calculate the **Likelihood** that they **Do Not Love Troll 2** = **1 - 0.8 = 0.2**.

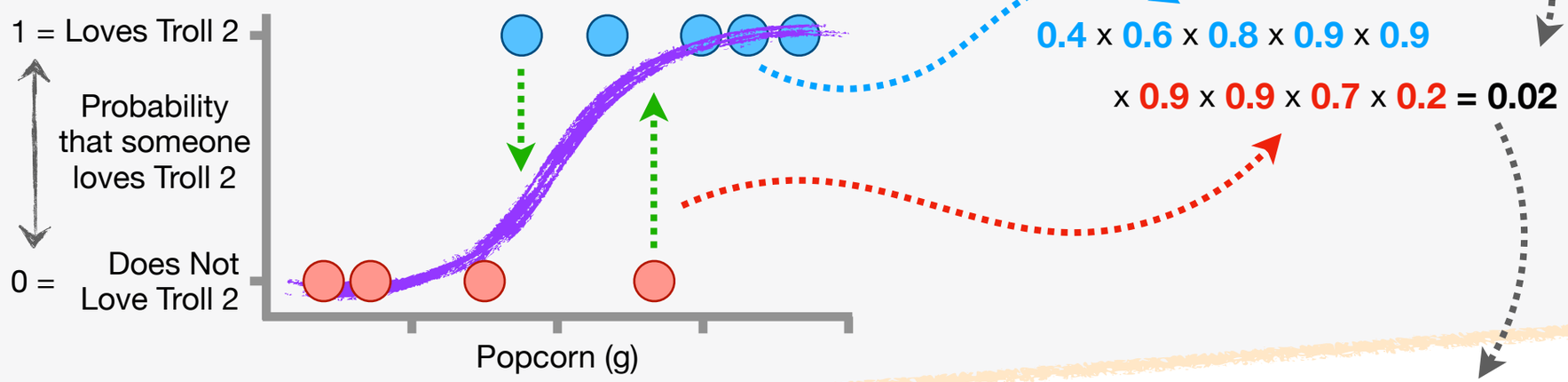


Bam!

Fitting A Squiggle To Data: Main Ideas Part 3

7 Now that we know how to calculate the **Likelihoods** for people who **Love Troll 2** and people who **Do Not Love Troll 2**, we can calculate the **Likelihood** for the entire **squiggle** by multiplying the individual **Likelihoods** together…

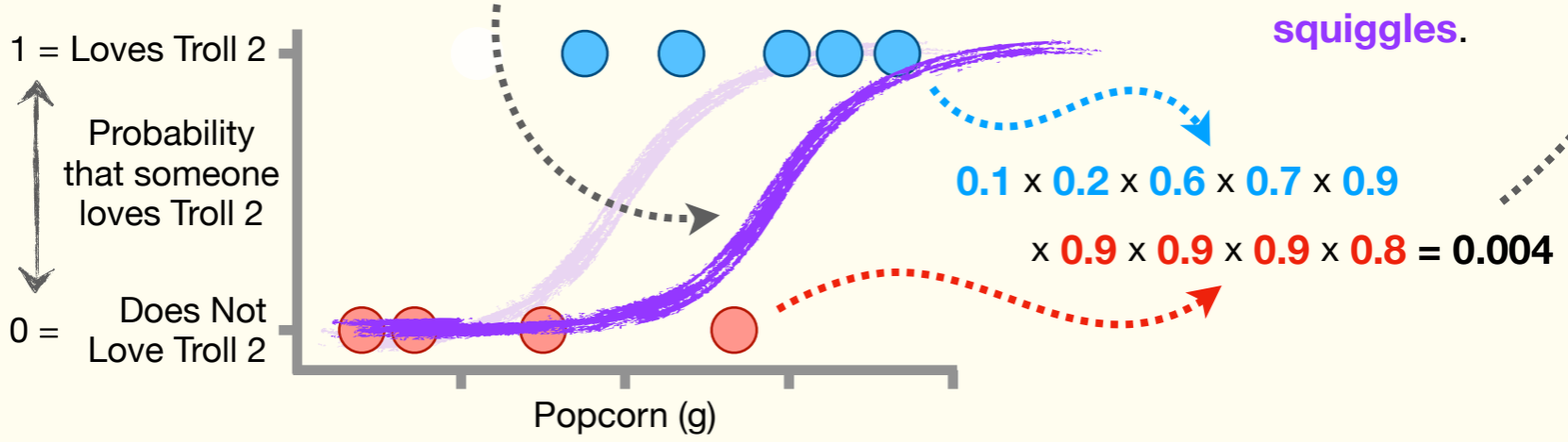
…and when we do the math, we get **0.02**.



VS.

8 Now we calculate the **Likelihood** for a different **squiggle**…

…and compare the total **Likelihoods** for both **squiggles**.



9 The goal is to find the **squiggle** with the **Maximum Likelihood**.

In practice, we usually find the optimal **squiggle** using **Gradient Descent**.

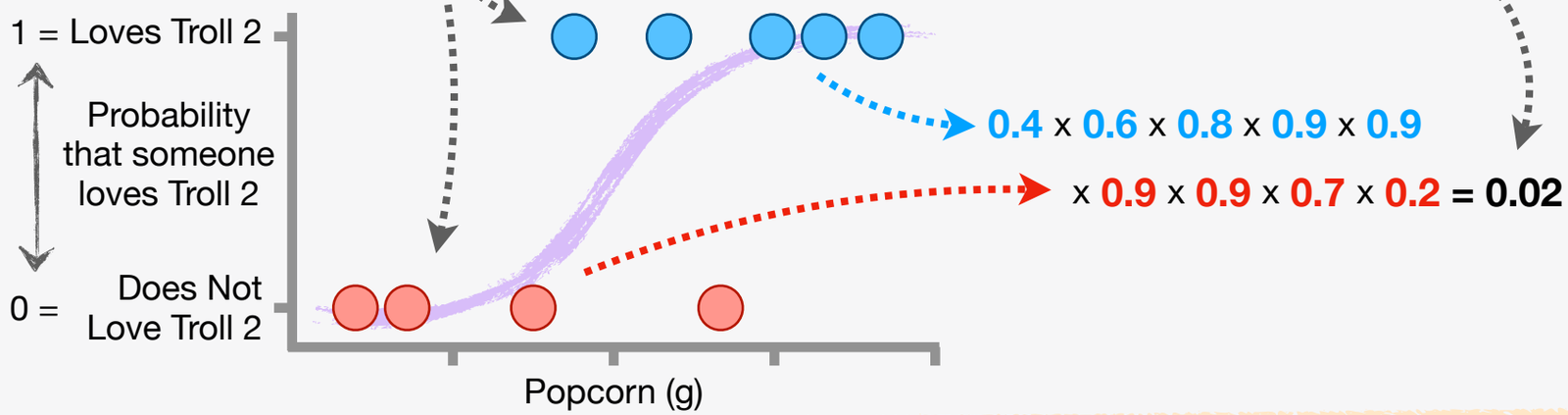
TRIPLE BAM!!!

Fitting A Squiggle To Data: Details

If you'd like to learn more details about **Logistic Regression**, scan, click, or tap this QR code to check out the **'Quests** on YouTube!!!

1 The example we've used so far has a relatively small **Training Dataset** of only **9** points total…

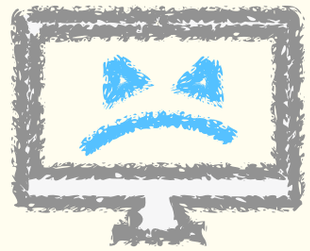
…so when we multiplied the **9** **Likelihoods** together, it was easy, and we got **0.02**.



0.4 x **0.6** x **0.8** x **0.9** x **0.9**

x **0.9** x **0.9** x **0.7** x **0.2** = **0.02**

2 However, if the **Training Dataset** was much larger, then we might run into a computational problem, called **Underflow**, that happens when you try to multiply a lot of small numbers between **0** and **1**.



Technically, **Underflow** happens when a mathematical operation, like multiplication, results in a number that's smaller than the computer is capable of storing.

Underflow can result in errors, which are bad, or it can result in weird, unpredictable results, which are worse.

3 A very common way to avoid **Underflow** errors is to just take the **log** (usually the **natural log**, or **log base *e***), which turns the multiplication into addition…

log(0.4 x 0.6 x 0.8 x 0.9 x 0.9 x 0.9 x 0.9 x 0.7 x 0.2)

= log(0.4) + log(0.6) + log(0.8) + log(0.9) + log(0.9)

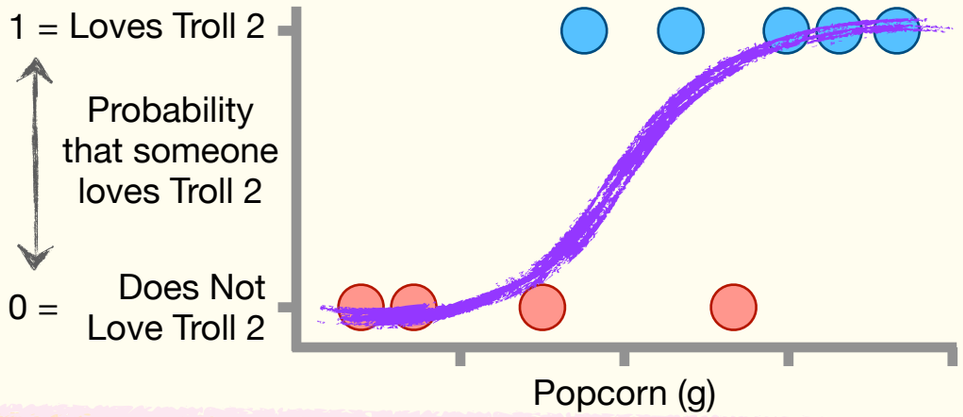
+ log(0.9) + log(0.9) + log(0.7) + log(0.2)

= -4.0

…and, ultimately, turns a number that was relatively close to **0**, like **0.02**, into a number relatively far from **0**, **-4.0**.

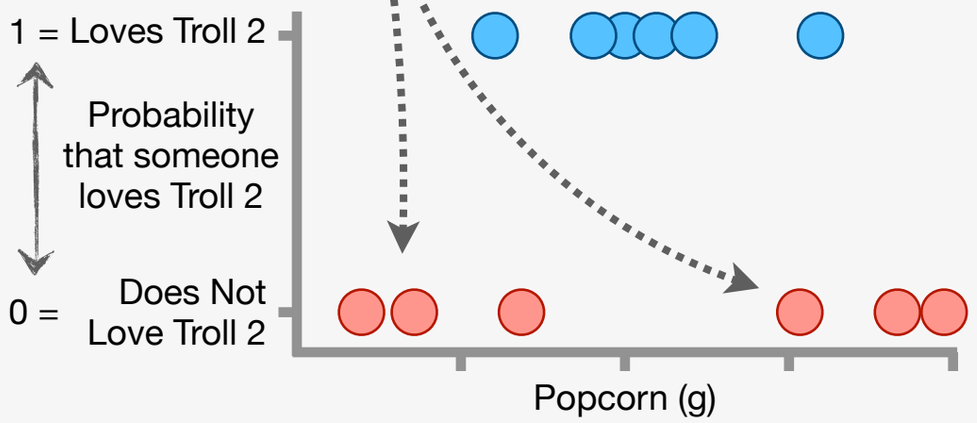
Logistic Regression: Weaknesses

1 When we use **Logistic Regression**, we assume that an **s-shaped squiggle** (or, if we have more than one independent variable, an s-shaped surface) will fit the data well. In other words, we assume that there's a relatively straightforward relationship between the Popcorn and Loves Troll 2 variables: if someone eats very little Popcorn, then there's a relatively low probability that they love Troll 2, and if someone eats a lot of Popcorn, then there's a relatively high probability that they love Troll 2.

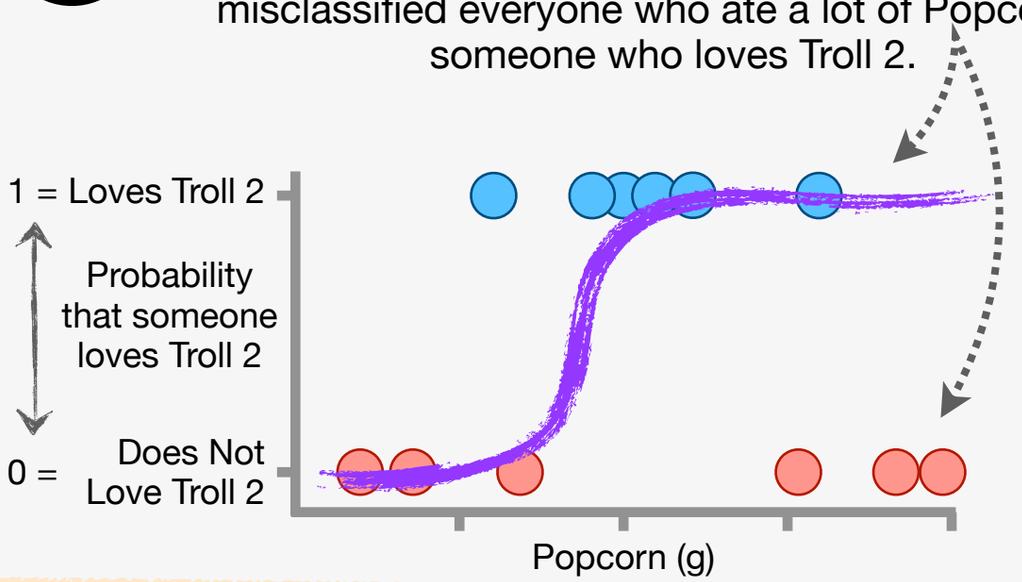


2 In contrast, if our data had people who ate a little or a lot of Popcorn and do not love Troll 2…

…and people who ate an intermediate amount did not, then we would violate the assumption that **Logistic Regression** makes about the data.



3 And if we used **Logistic Regression** to fit an **s-shaped squiggle** to the data, we would get a horrible model that misclassified everyone who ate a lot of Popcorn as someone who loves Troll 2.



4 Thus, one of the limitations of **Logistic Regression** is that it assumes that we can fit an **s-shaped squiggle** to the data. If that's not a valid assumption, then we need a **Decision Tree** (**Chapter 10**), or a **Support Vector Machine** (**Chapter 11**), or a **Neural Network** (**Chapter 12**), or some other method that can handle more complicated relationships among the data.

Chapter 07

Naive Bayes!!!