**Logistic Regression**

Table of Contents

[Sigmoid Function 3](#_Toc100150394)

[Decision Boundary 4](#_Toc100150395)

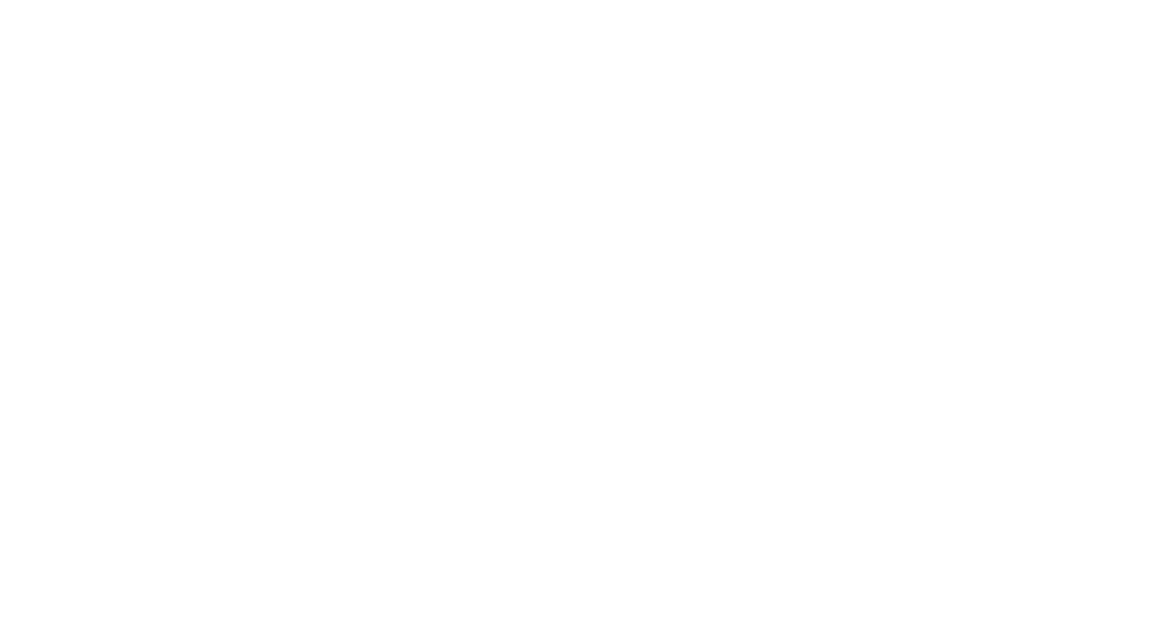
[Cost Function 6](#_Toc100150396)

[Gradient Descent 9](#_Toc100150397)

[Advanced Optimization 10](#_Toc100150398)

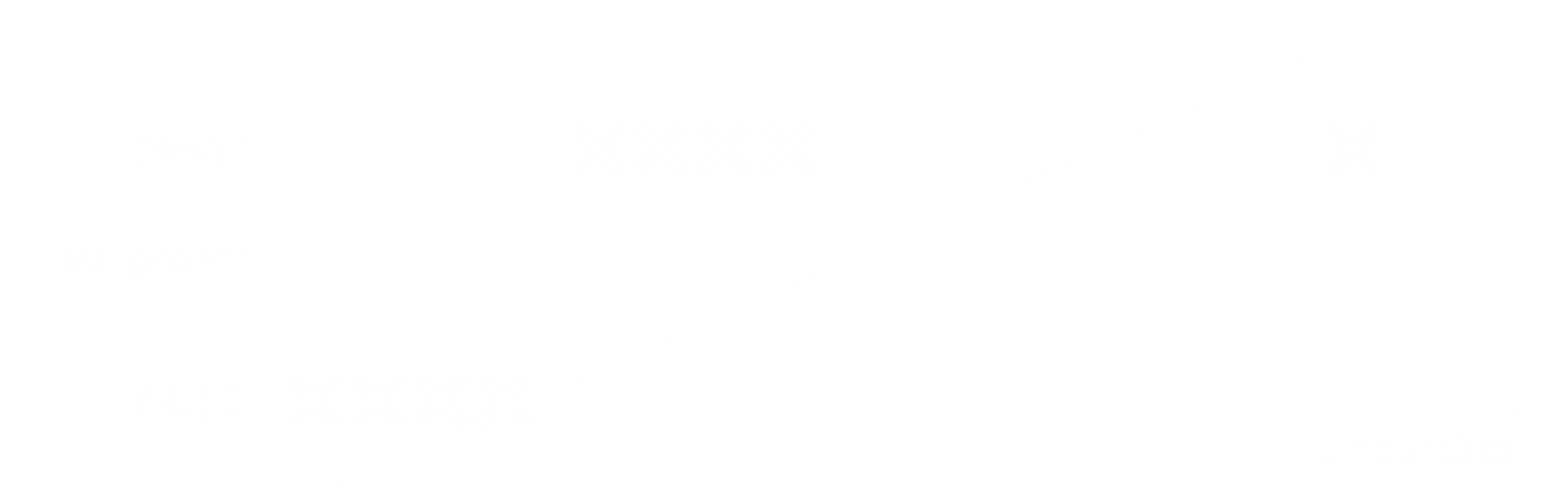
**Logistic Regression** is used with **classification** problems. To start off, we will be dealing with **binary classification**, meaning the output can be either or .

Say we have a situation where we want to predict whether a tumour is harmless or cancerous. The parameter we are considering is the size of the tumour. We could apply **Gradient Descent** to this problem.

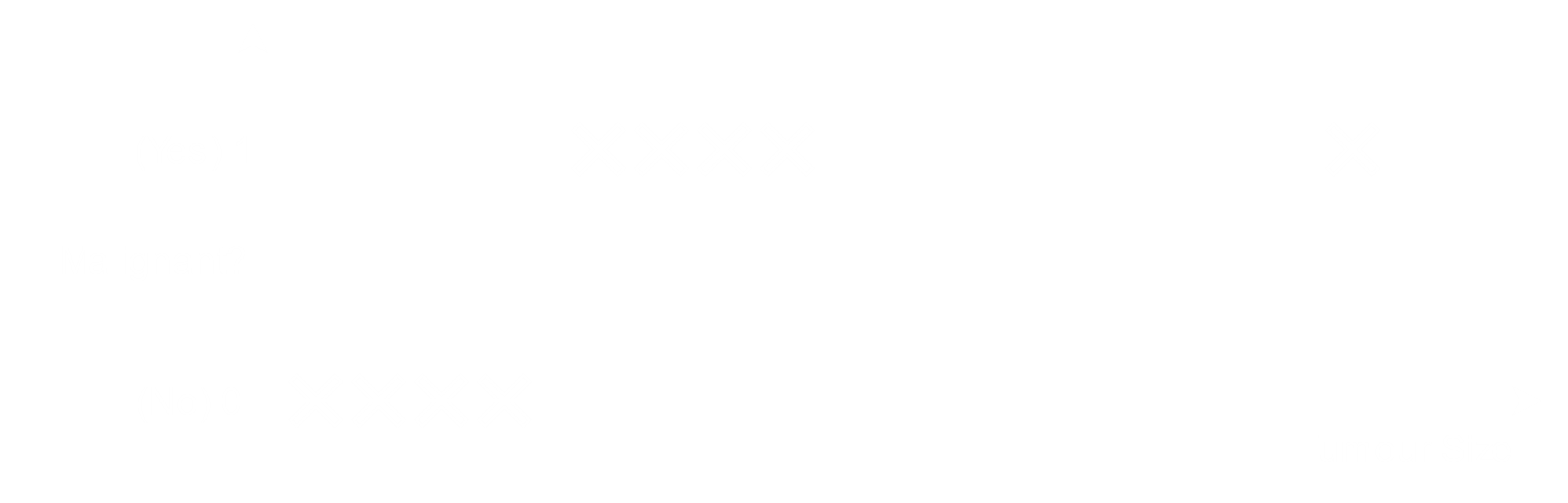


Using Gradient Descent, we get some line for . If , then . Problem solved right? Not quite.

Consider that we have a different situation where another data point is much further to the right.



This time if we say if then , we would be wrong, because clearly for values of as well. What we actually needed is something like this:



This is a **threshold**. Unfortunately, Gradient Descent cannot give us a line like this.

Another issue is that the range of values of ranges from to , but we literally need just two values, and .

## Sigmoid Function

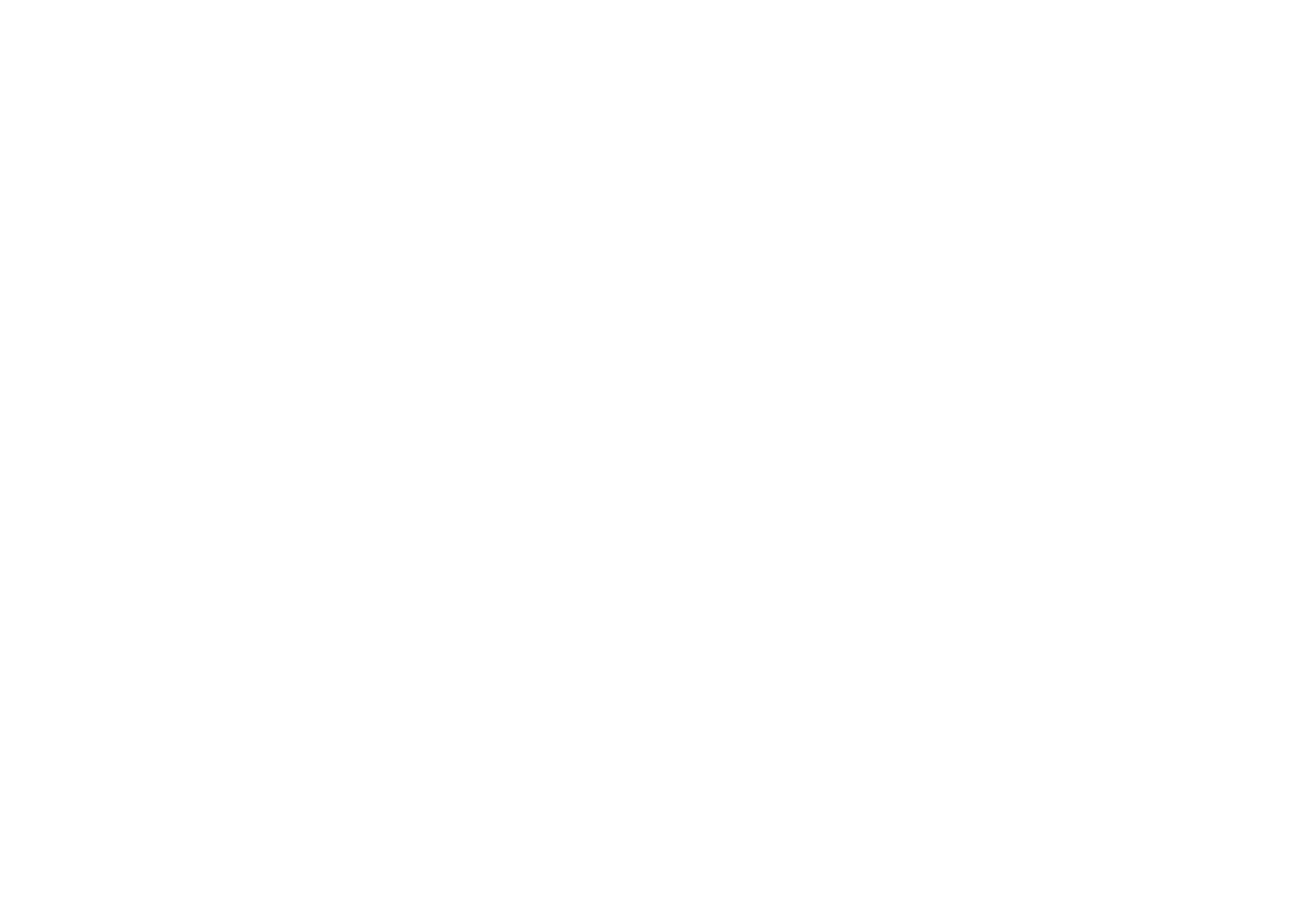
Instead of using Gradient Descent, we will be using the following function:

where

This is called the **Sigmoid Function** or the **Logistic Function**. Essentially, it gives us the **probability** that the output will be . More formally, it gives us the probability that given parameterized by , . Thus, if , there is a probability that .

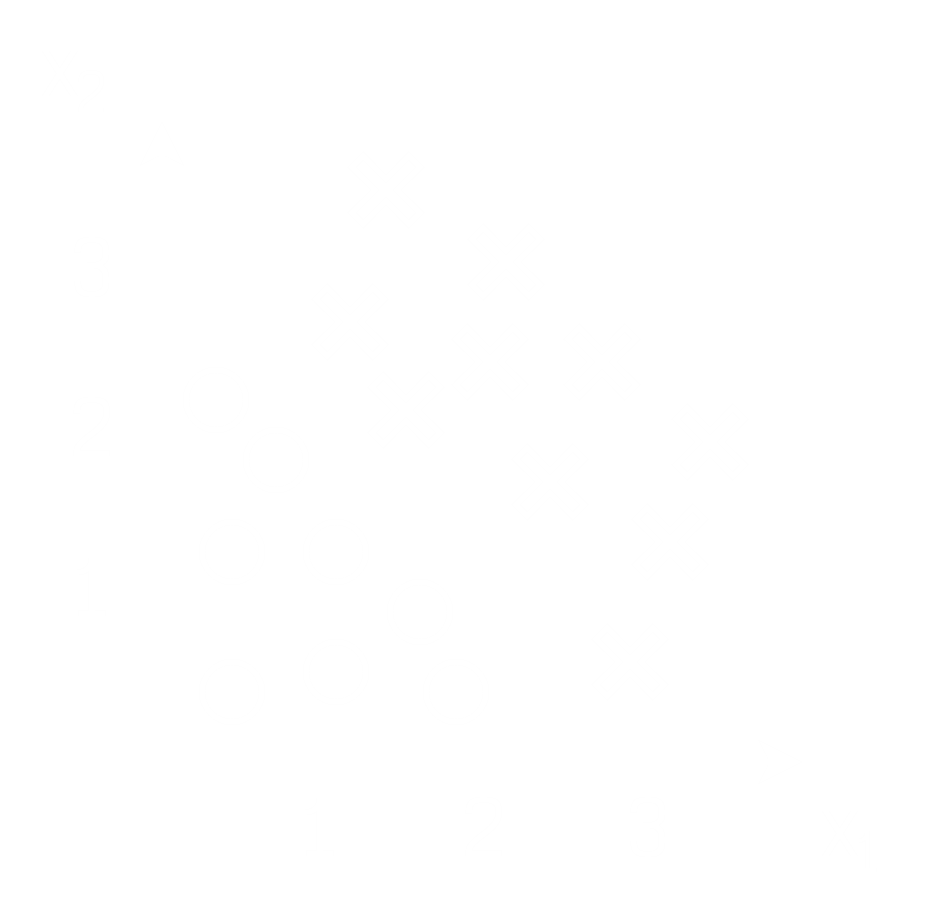
This function is also sometimes broken down as , where .

The Sigmoid Function looks like this:

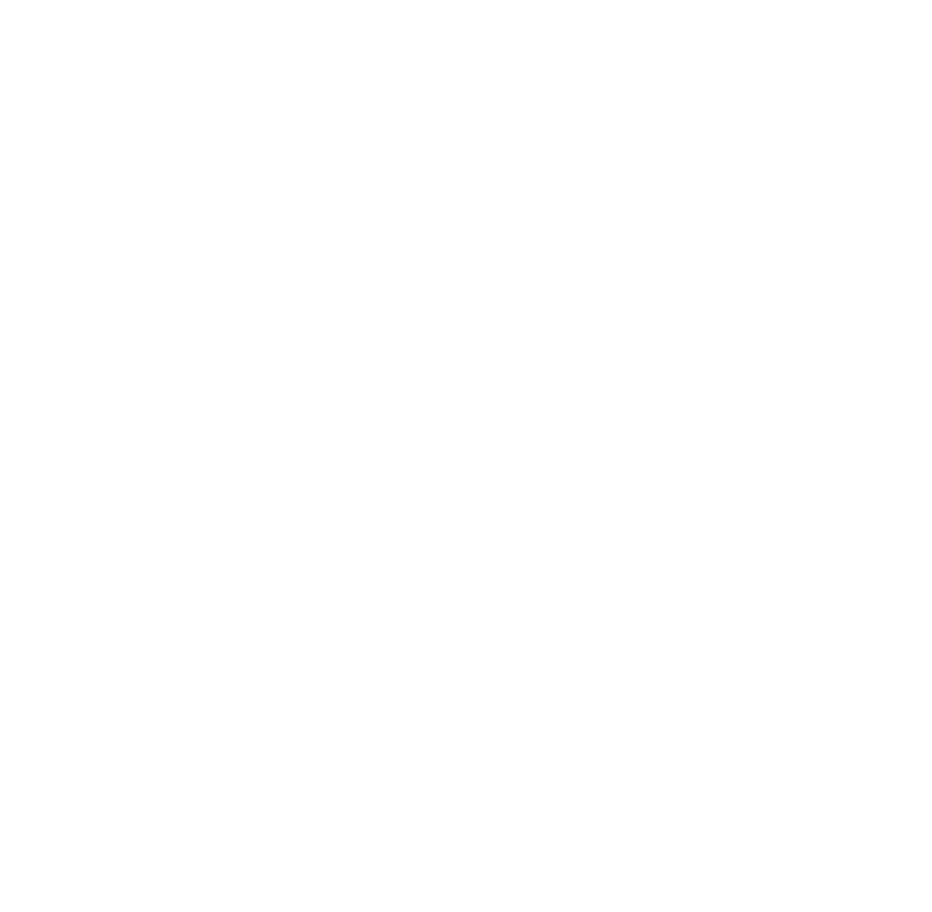


Thus, the range of is also reduced to .

## Decision Boundary

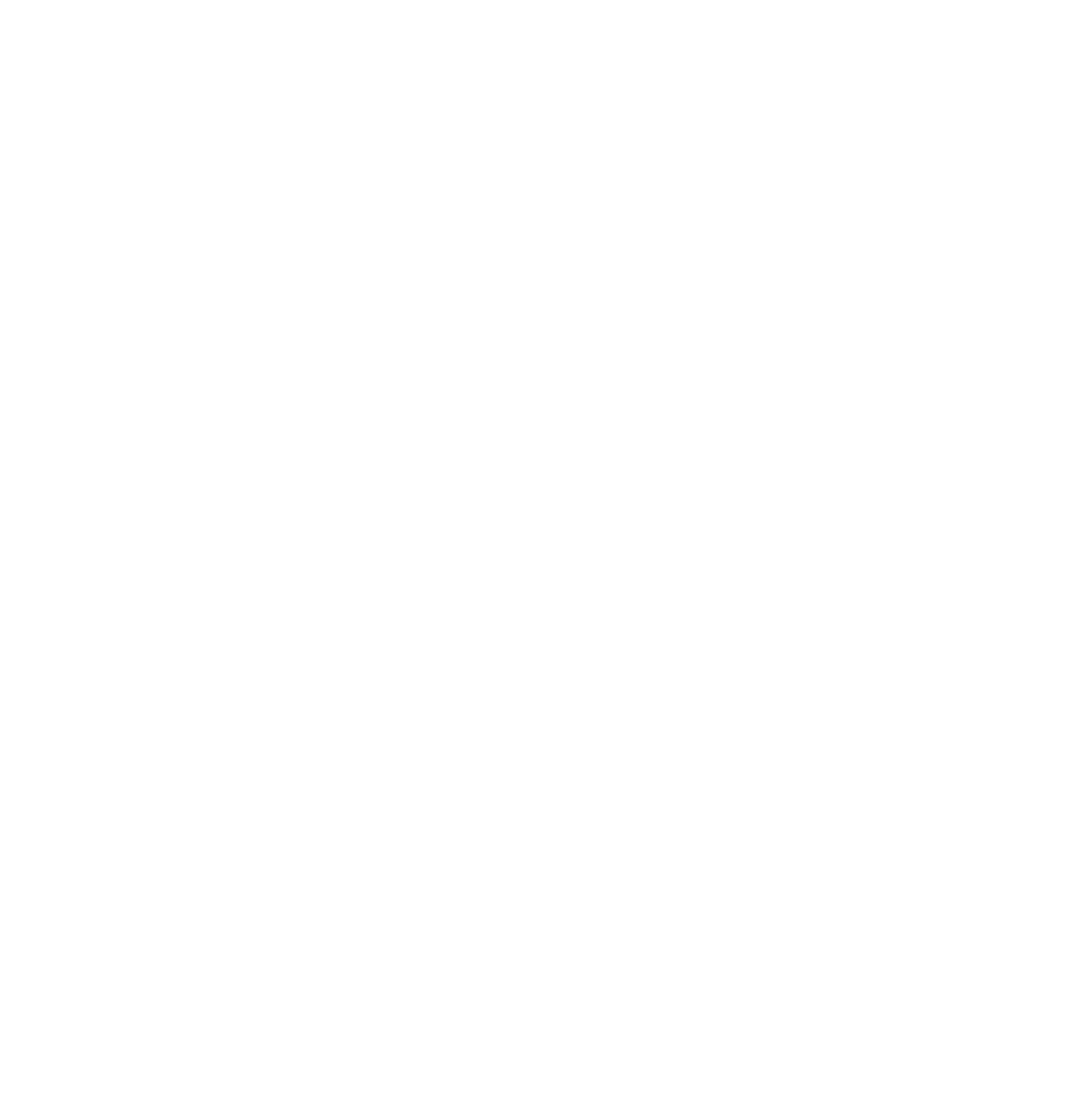


Consider that we have the above data, for which we form a hypothesis . We will look into how to calculate the parameters for logistic regression later on, but for now, know that the correct values in this case are , and respectively. Thus, if , we will predict . This equation gives us the following line:

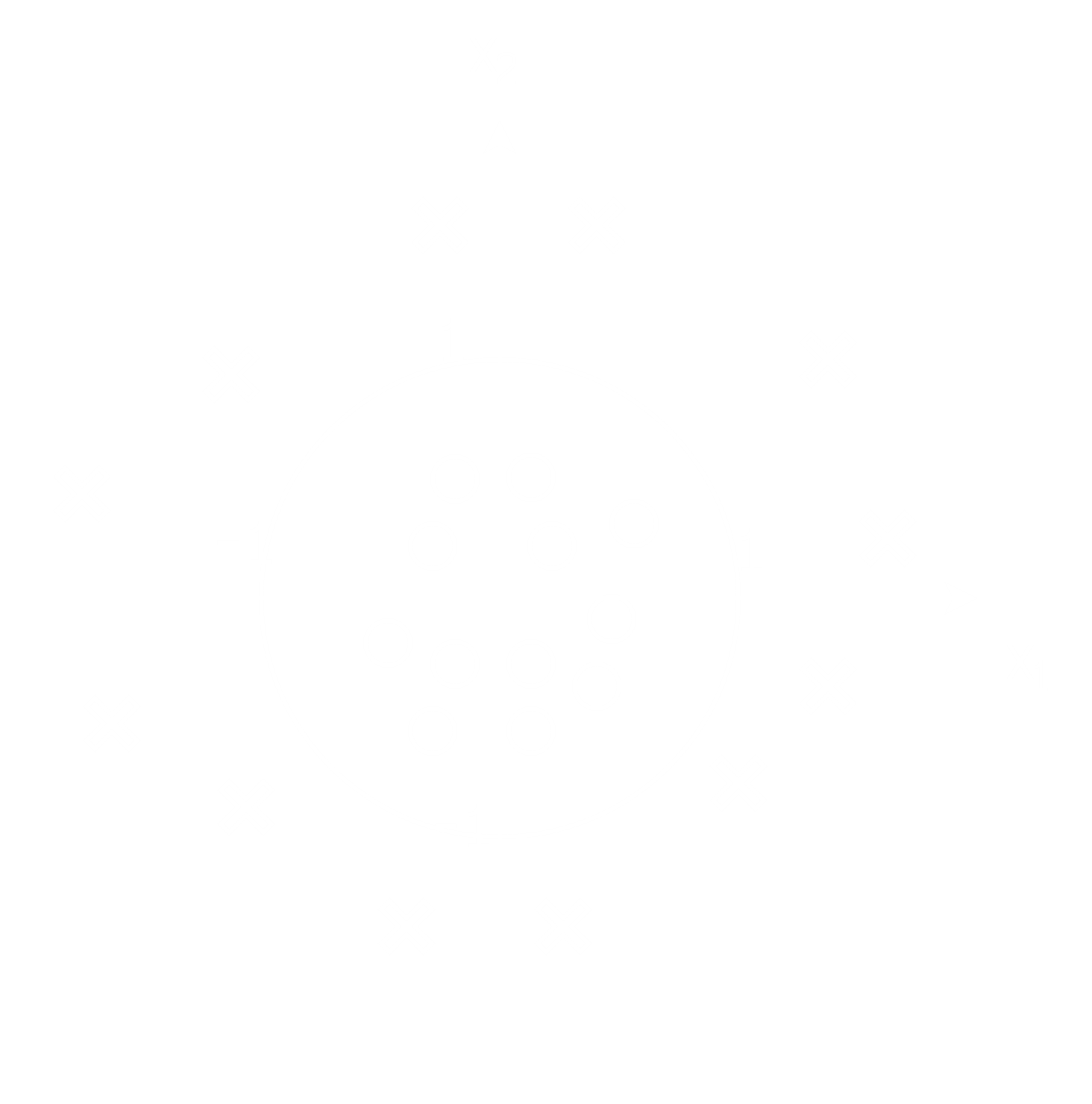


This line is called the **decision boundary**. Note that this decision boundary is a property of the **hypothesis**, not the data set. If we form a different, incorrect hypothesis, it will have a different decision boundary. Thus, the decision boundary for is .

We know that the hypothesis could use a **non-linear** function as well. This means that the decision boundary can also be non-linear.



Suppose for the above data we have a hypothesis , for which we find that correct parameter values are , , , and respectively. This gives us the decision boundary .



Using this, extremely complicated decision boundaries can be formed.

## Cost Function

For linear regression, we saw that the **cost function** could be written as:

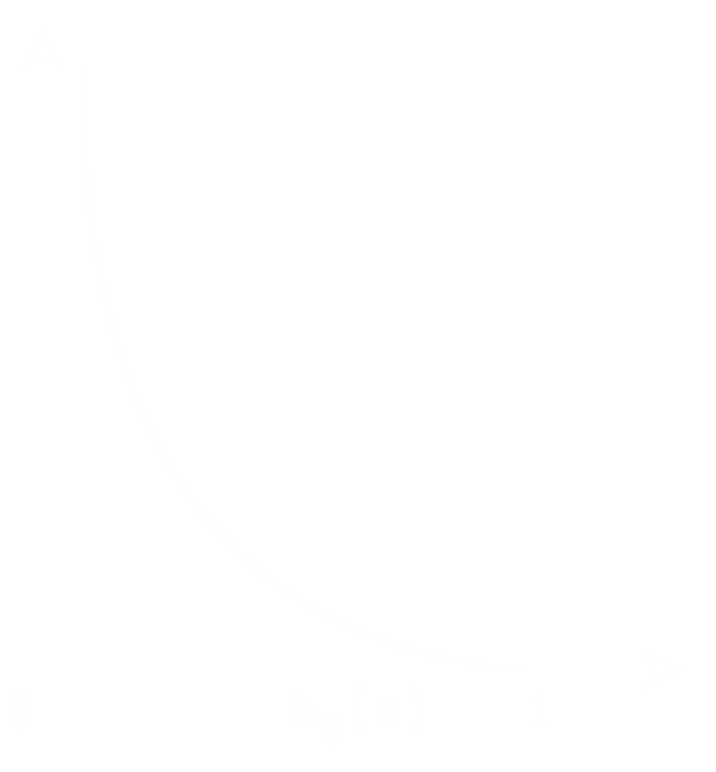
For a single prediction, let’s suppose that the error is represented as

For **logistic regression**, we simply change this error to this:

Unfortunately, this is not a **convex function**. This will prevent gradient descent from working, so instead, we use this:

Let’s take a look at why this works.

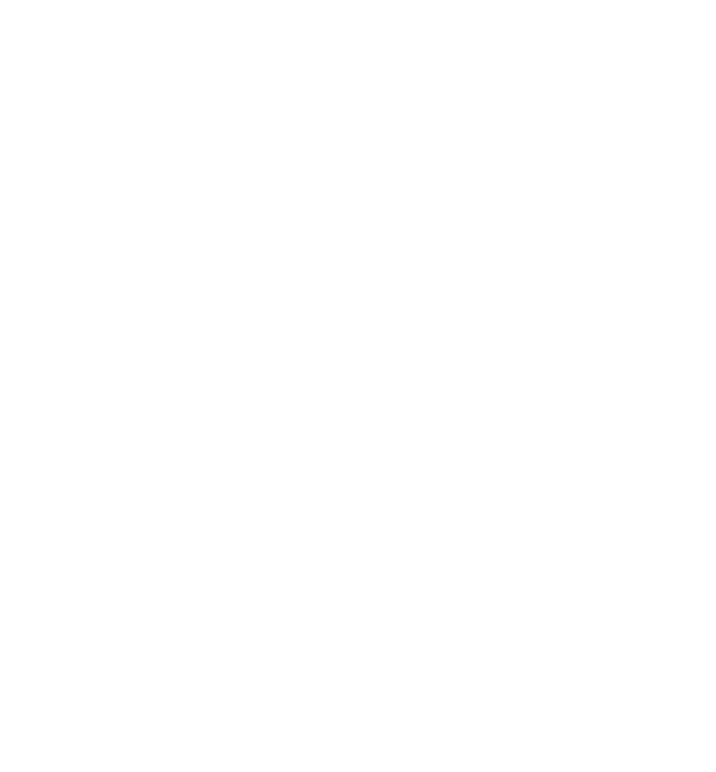
For , we have the following graph for :



Thus, if and our prediction is , we have a  **error**. On the other hand, the closer we get to making a prediction of , the direct opposite of the true answer, the **higher our error**.

Here, it is important to note that the **actual prediction** of our model will still be either or . Thus, if we get a value of , we will say that our prediction is (which will be correct in this case). However, we will still have an **error** of , which will contribute to helping us decide which parameters gives the least overall error.

Similarly, for , we get the following graph for :



This also works in a similar way. If and , we get a error, while the closer we get to , the higher our error.

These two equations can be combined together cleverly into one line:

Notice that for and , the respective parts of the equation ‘activate’, making this equation exactly the same as the two-part equation we originally saw.

Thus, for logistic regression:

where .

## Gradient Descent

Now we know that, for logistic regression,

We need to apply gradient descent on this to find the optimal parameter values.

For linear regression, we found the parameters as follows:

For logistic regression, we just need to make one change, since has changed.

In fact, if we write it in terms of , it will seem as though there is no change at all.

## Advanced Optimization

There are other techniques we can use, such as Conjugate Gradient, BFGS, L-BFGS, etc., which are much faster than Gradient Descent and we often don’t even need to pick a value for , but they are also far more complicated. For such techniques, it is best to use library functions instead of trying to write them out ourselves.