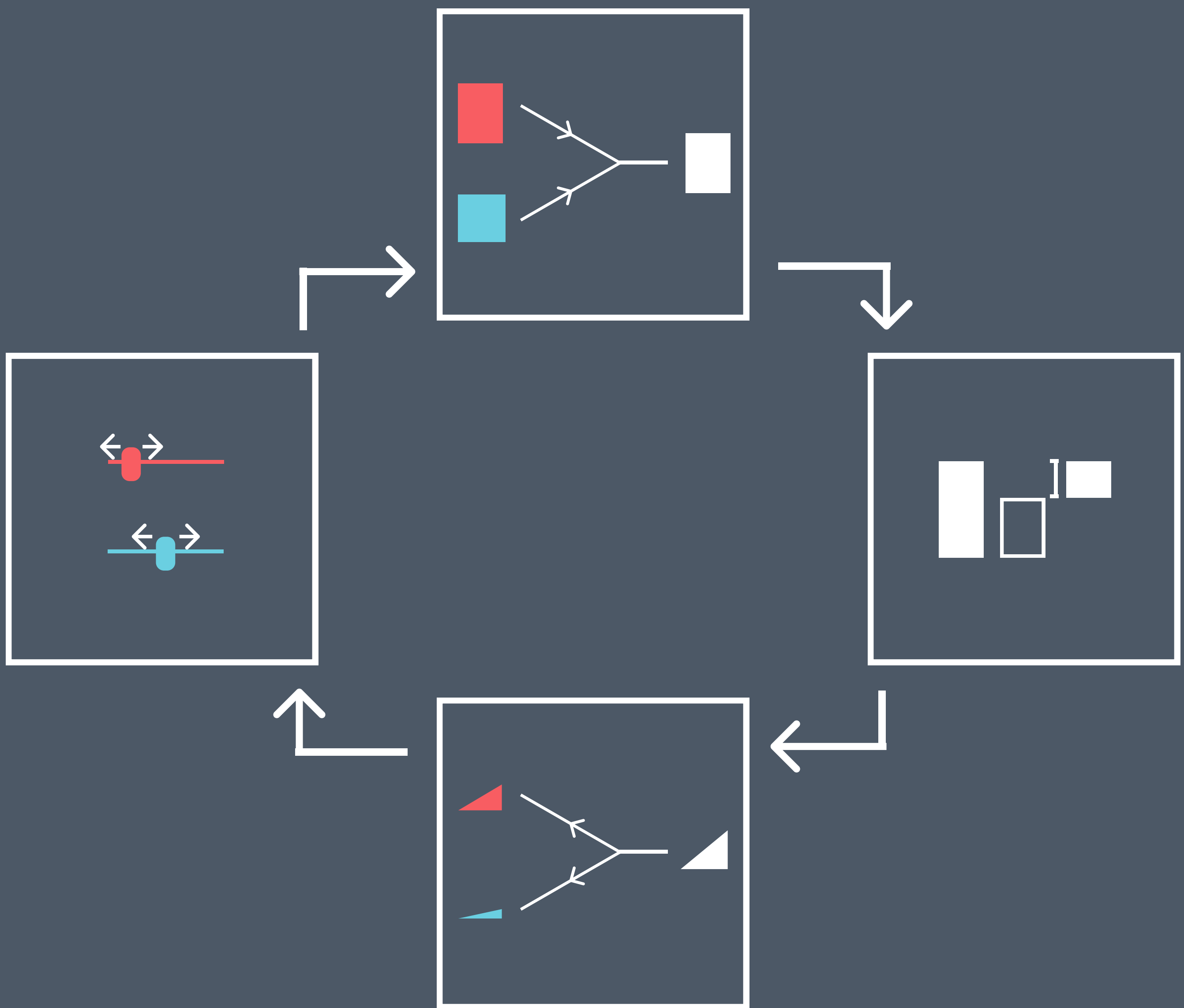


A VISUAL INTRODUCTION TO **DEEP LEARNING**



ABOUT THIS BOOK

Deep learning is the algorithm powering the current renaissance of artificial intelligence (AI). And its progress is not showing signs of slowing down. A McKinsey report estimates that by 2030, AI will potentially deliver \$13 trillion to the global economy, or 16% of the world's current GDP. This opens up exciting career opportunities in the coming decade.

But deep learning can be quite daunting to learn. With the abundance of learning resources in recent years has emerged another problem—information overload.

This book aims to compress this knowledge and make the subject approachable. By the end of this book, you will be able to build a visual intuition about deep learning and neural networks.

WHO SHOULD READ THIS BOOK

If you are new to deep learning, or machine learning in general.

If you already know some background about deep learning but want to gain further intuition.

BOOK FORMAT

This book uses a visuals-first approach. Each page of this book begins with a visual and is supported by concise text.

This book doesn't include math derivations and code examples. There are some parts where basic math is involved, but generally it is kept to a minimum.

ABOUT THE AUTHOR

My journey into AI began in 2010 after my son was born with a limb difference. I became interested in machine learning for prosthetics and did an MSc at Imperial College London majoring in neurotechnology.

I have also worked in the telecoms data analytics space, serving clients in over 15 countries.

Above all, I am passionate about education and how we learn. I am currently working on projects that explore ways to create alternative learning experiences using visuals, storytelling, and games.



email: contact@kdimensions.com

TABLE OF CONTENTS

INTRODUCTION

• PREDICTION & DECISION	4
• MACHINE LEARNING	6
• DEEP LEARNING	12
• ALGORITHM	13
• DATA	14
• COMPUTATION	15
• ROADMAP	19
• KEY CONCEPTS	20

FOUNDATIONS

• A NEURON	22
• WEIGHTED SUM	24
• WEIGHTS AND BIASES	25
• ACTIVATION	27
• DATA	30
• DATASET	34
• TRAINING	38
• TESTING	40

1 - LINEAR REGRESSION

• INTRODUCTION	43
• GOAL AND DATASET	45
• PREDICT-MEASURE-FEEDBACK-ADJUST	50
• WEIGHTED SUM AND ACTIVATION	52
• LOSS FUNCTION	58
• MEAN SQUARED ERROR	59
• MINIMIZING LOSS	64
• GRADIENT	68
• GRADIENT DESCENT	73
• LEARNING RATE	76
• EPOCH	81
• COST AND METRIC	83
• PERFORMANCE	86

2 - NON-LINEAR REGRESSION

• INTRODUCTION	93
• GOAL AND DATASET	99
• ARCHITECTURE	102
• PREDICT	108
• MEASURE	112
• FEEDBACK	114
• COMPUTATION GRAPH	117

• BACKPROPAGATION	119
• ADJUST	129
• PERFORMANCE	135
• LINEAR ACTIVATION	137
• LINEARITY	139
• NON-LINEARITY	141
• RELU ACTIVATION	142
• PERFORMANCE	145
• ACTIVATION FUNCTIONS	148

3 - BINARY CLASSIFICATION

• INTRODUCTION	150
• CLASSIFICATION VS. REGRESSION	152
• GOAL AND DATASET	153
• ARCHITECTURE	156
• SIGMOID ACTIVATION	158
• BINARY CROSS ENTROPY	166
• ACCURACY	169
• PERFORMANCE	170
• CONFUSION MATRIX	174
• PRECISION-RECALL	176
• F1 SCORE	177

4 - MULTI-CLASS CLASSIFICATION

• INTRODUCTION	180
• GOAL AND DATASET	181
• ONE-HOT ENCODING	183
• ARCHITECTURE	184
• SOFTMAX ACTIVATION	188
• CATEGORICAL CROSS ENTROPY	195
• PERFORMANCE	197
• IMPROVING PERFORMANCE	
• HYPERPARAMETERS	202
• DATA TECHNIQUES	210

THE BIGGER PICTURE

• NEURAL NETWORKS	
• FEEDFORWARD	217
• CONVOLUTIONAL	219
• RECURRENT	225
• GENERATIVE ADVERSARIAL	230
• OTHER ARCHITECTURES	232
• CONCLUSION	233
• KEY CONCEPTS REVISITED	234
• SUGGESTED RESOURCES	235



PREDICTION AND DECISION

Prediction is a key ingredient in decision-making under uncertainty. — Prediction Machines book.

Much that goes on in our lives involves some form of prediction. These predictions differ in one way, namely, how sure we are of them. In some tasks, they don't feel like predictions because we feel so sure about them. In some others, we know next to nothing about them, so they become mere guesses.

All of this depends on how simple a task is and, more importantly, how much experience we have with it.

To illustrate this, let's look at some examples.

PREDICTION → DECISION

LANGUAGE
TRANSLATION



WHAT IS THE
PERSON SAYING?

REPLY

CUSTOMER
SERVICE



WILL THE
CUSTOMER CHURN?

DISCOUNT

DRIVING



IS THAT AN
OBSTACLE?

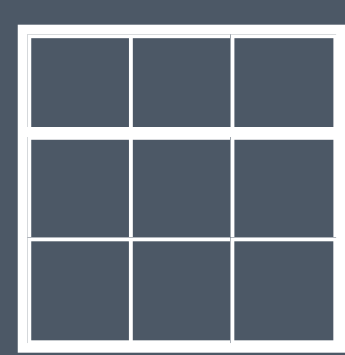
STEER

EXAMPLES

Let's take the example of language translation. As we listen to someone speaking, we are predicting what the person means. The more experience we have with this language, the better our prediction becomes, and the better our decision, that is our reply, becomes.

Take another example in a business setting. Our experience dealing with customers can help us see patterns in their behavior, so we'll notice if they are likely to churn.

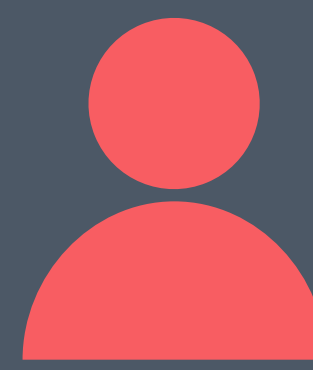
As for driving, the more miles we clock, the more skilled we become and the more adept we are at evaluating our surroundings.



DATA



PREDICTION



DECISION

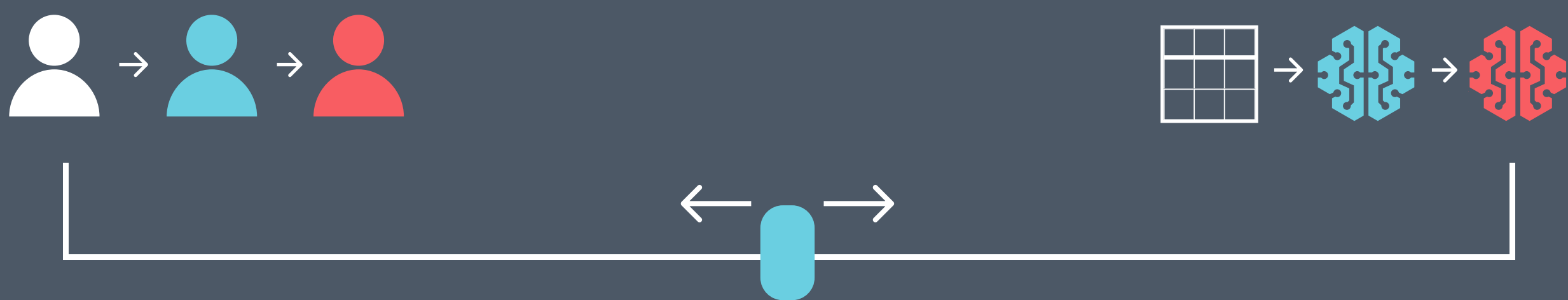
WHAT IS MACHINE LEARNING?

In many of these tasks, machine learning can handle the prediction on our behalf.

In recent years, the adoption of machine learning has accelerated. Many industries and verticals are already deploying use cases that automate predictions using machine learning.

In the machine's world, the experience comes in the form of data. Just as we learn from experience, the machine learns from data.

That is what machine learning is all about—learning from the data and turning it into predictions.



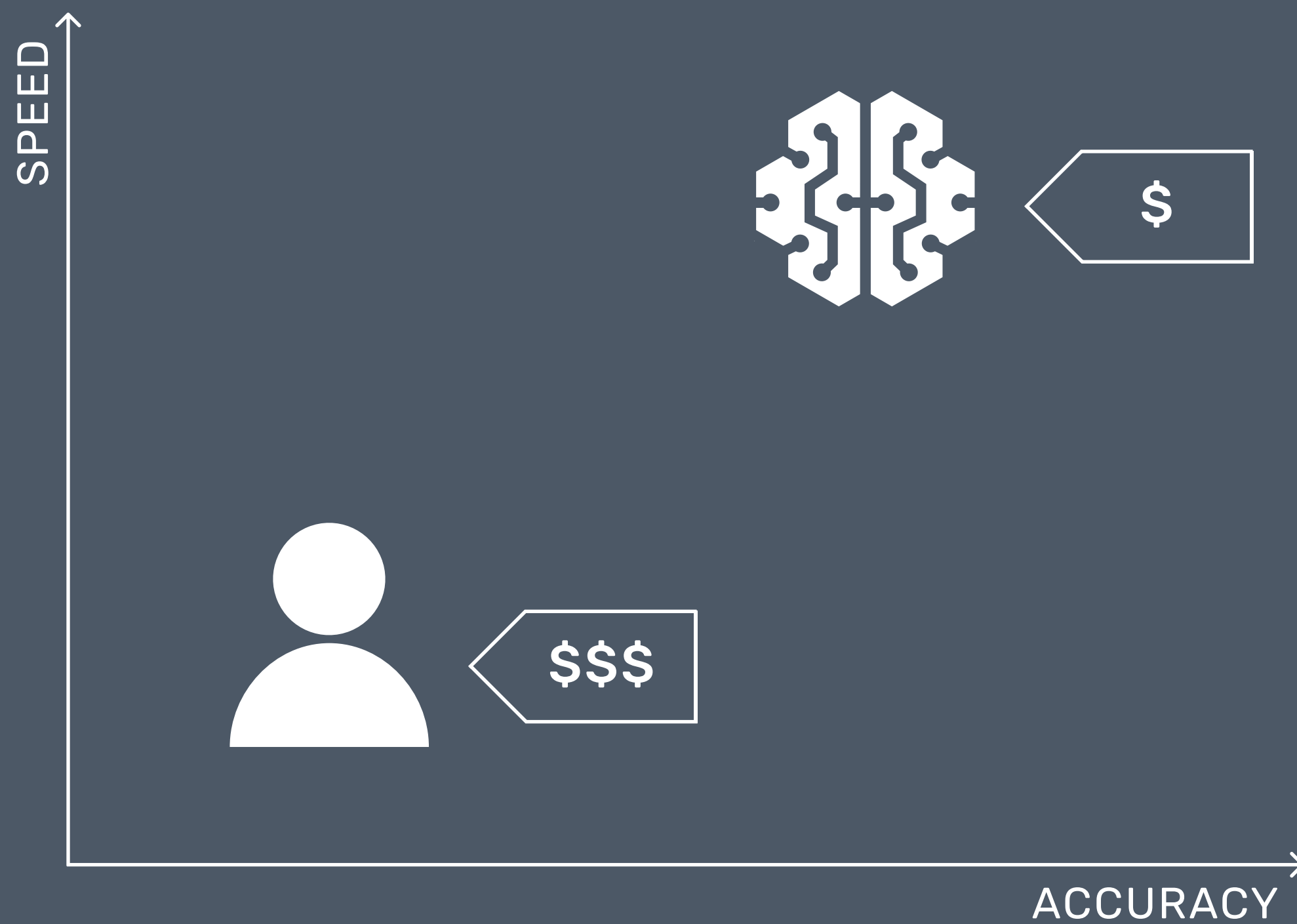
MACHINE LEARNING IN THE REAL WORLD

In fact, machine learning can even handle the decision part. In some domains, most notably self-driving cars, we are not far from seeing full automation becoming the norm.

But in most other domains, this is still far from reality. For this reason, the focus of this book is on the prediction part.

And indeed, it is upon us to ensure healthy technological progress, where people can thrive with the help of machines rather than being inhibited by them. That's the sweet spot that we are collectively trying to find.

PREDICTION



THE VALUE OF MACHINE LEARNING

So, what is the value of having machines that can make predictions on our behalf?

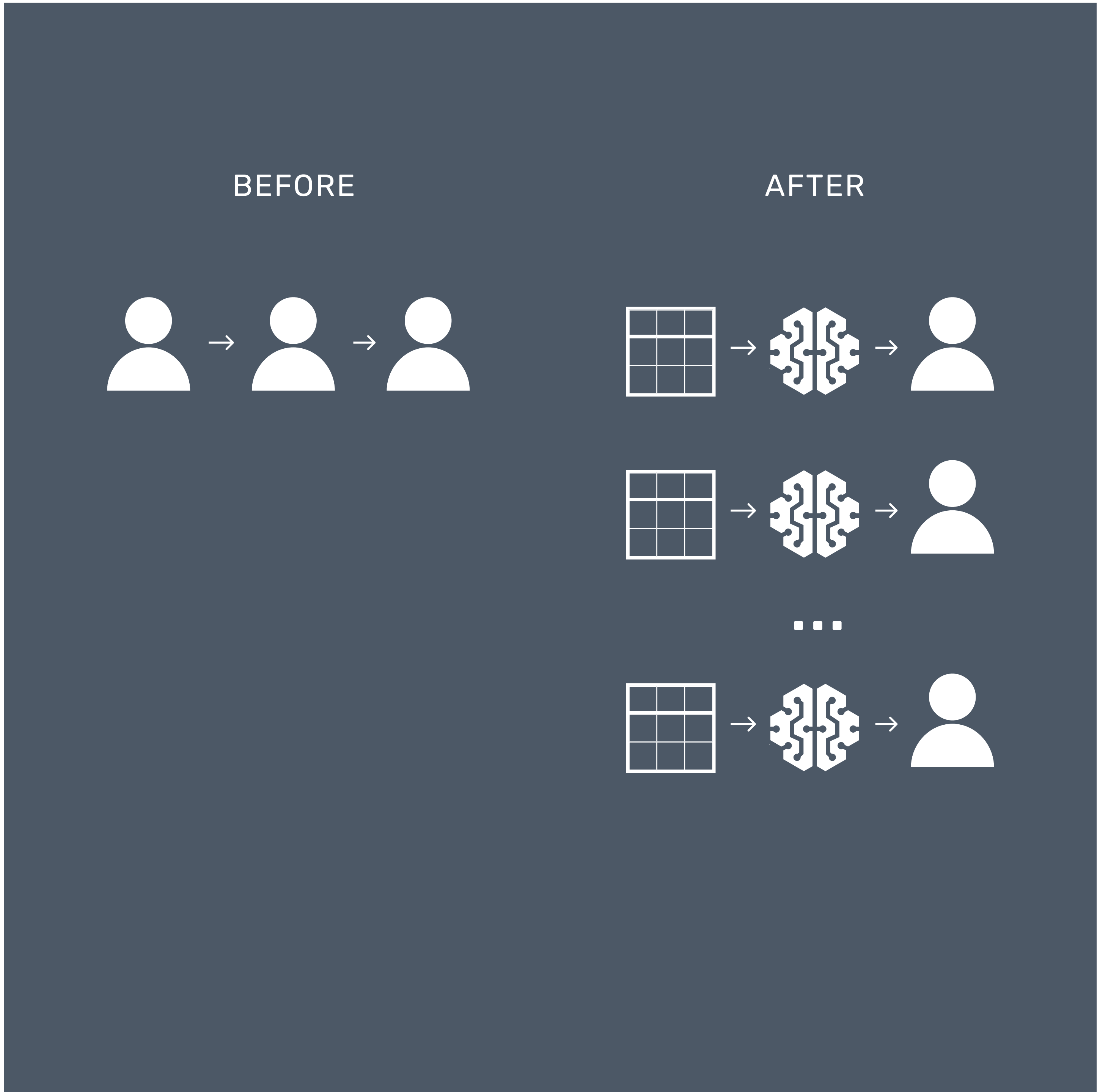
In the book *Prediction Machines*, the authors argued for a few reasons why prediction machines are so valuable, the first being that 'they can often produce better, faster, and cheaper predictions than humans can'.

PREDICTION



ACCELERATING HUMAN PROGRESS

The cheaper the cost of prediction, the more tasks we can take on. The world is full of challenges waiting to be solved. Machine learning enables us to scale our efforts in ways that have not been possible before, presenting us with the opportunity to take on these challenges.

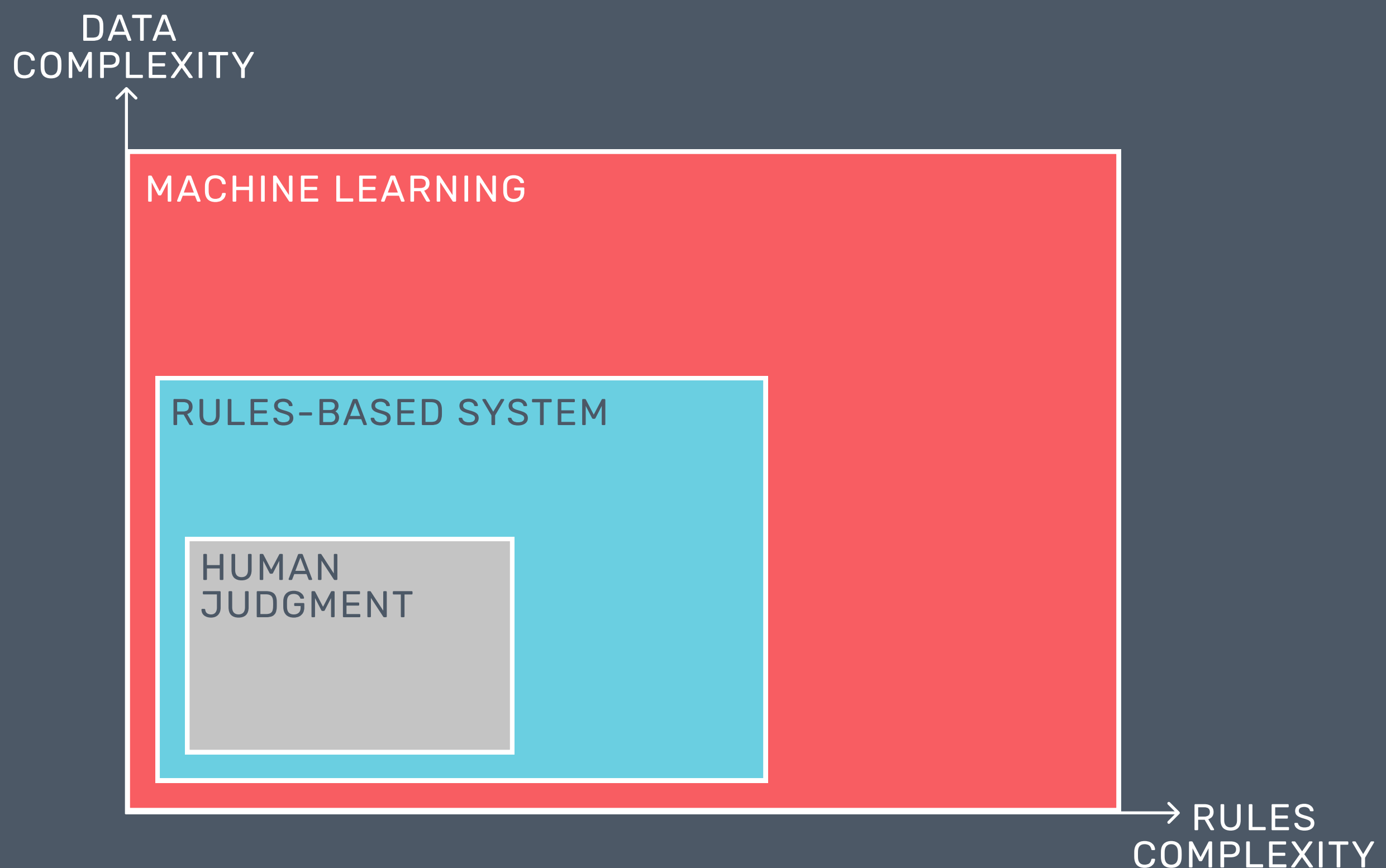


EVOLUTION IN ROLES

Some may worry that this will spell the end of most jobs, and rightly so. But looking at the bigger picture, there will in fact be even more job opportunities.

The World Economic Forum's *The Future of Jobs Report 2020* estimates that by 2025, 85 million jobs may be displaced. But on the other hand, 97 million new roles may emerge. This already takes into account the economic slowdown due to the pandemic, and still, the net effect is positive.

Job roles will evolve, and the machine's role is to serve us so we can pursue more creative and challenging endeavors.



WHEN TO USE MACHINE LEARNING?

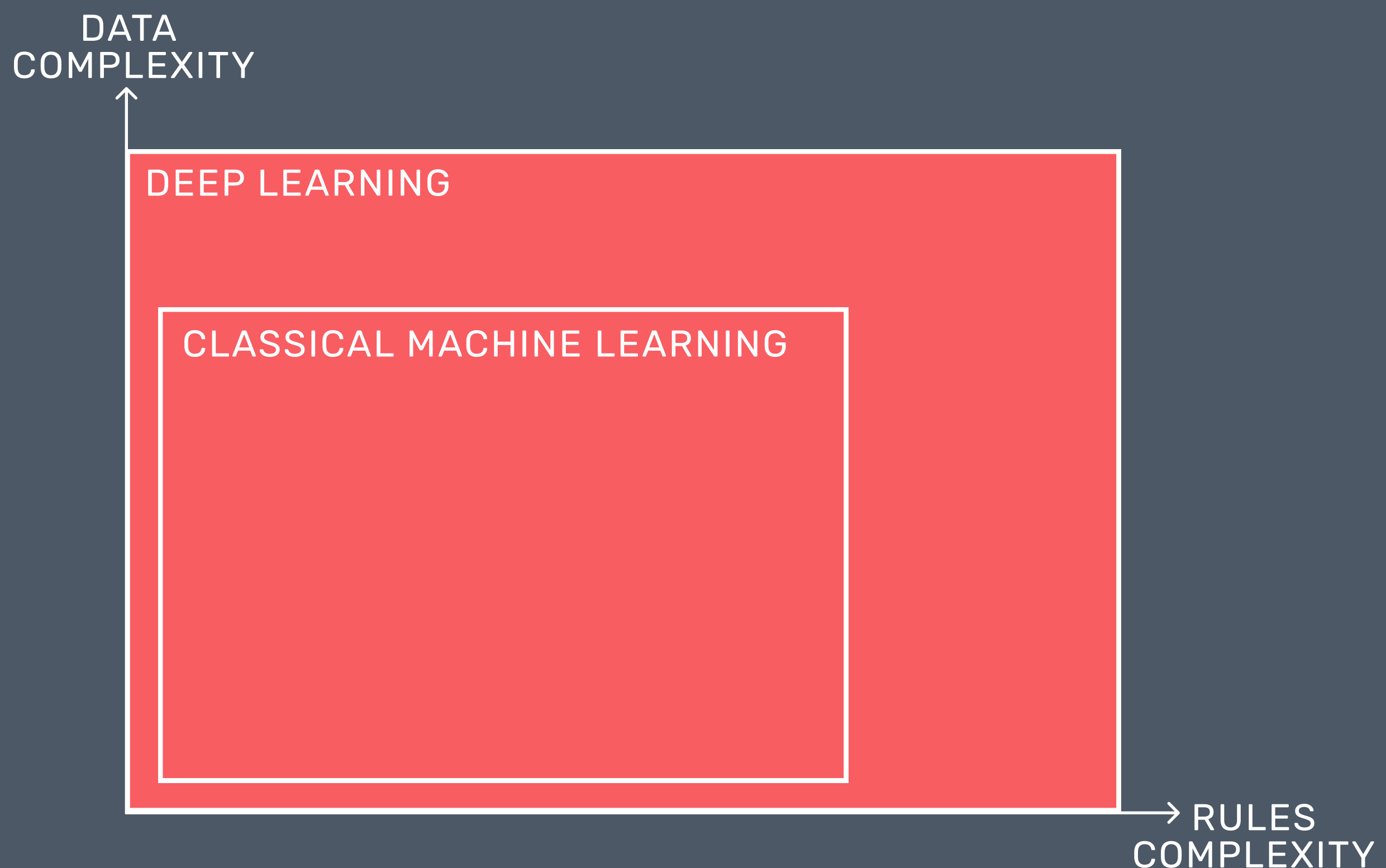
We can think of prediction automation in three phases.

The first, that is without automation, is relying on human judgment, either based on data or experience.

The second is using a rules-based system. We translate our experience into rules that software can understand and execute based on data as inputs.

The third is machine learning, which uses data to create its own rules, guided by the goal defined by humans.

As the data and rules become more complex, it makes sense to use machine learning. Otherwise, it may not be cost-effective to do so.



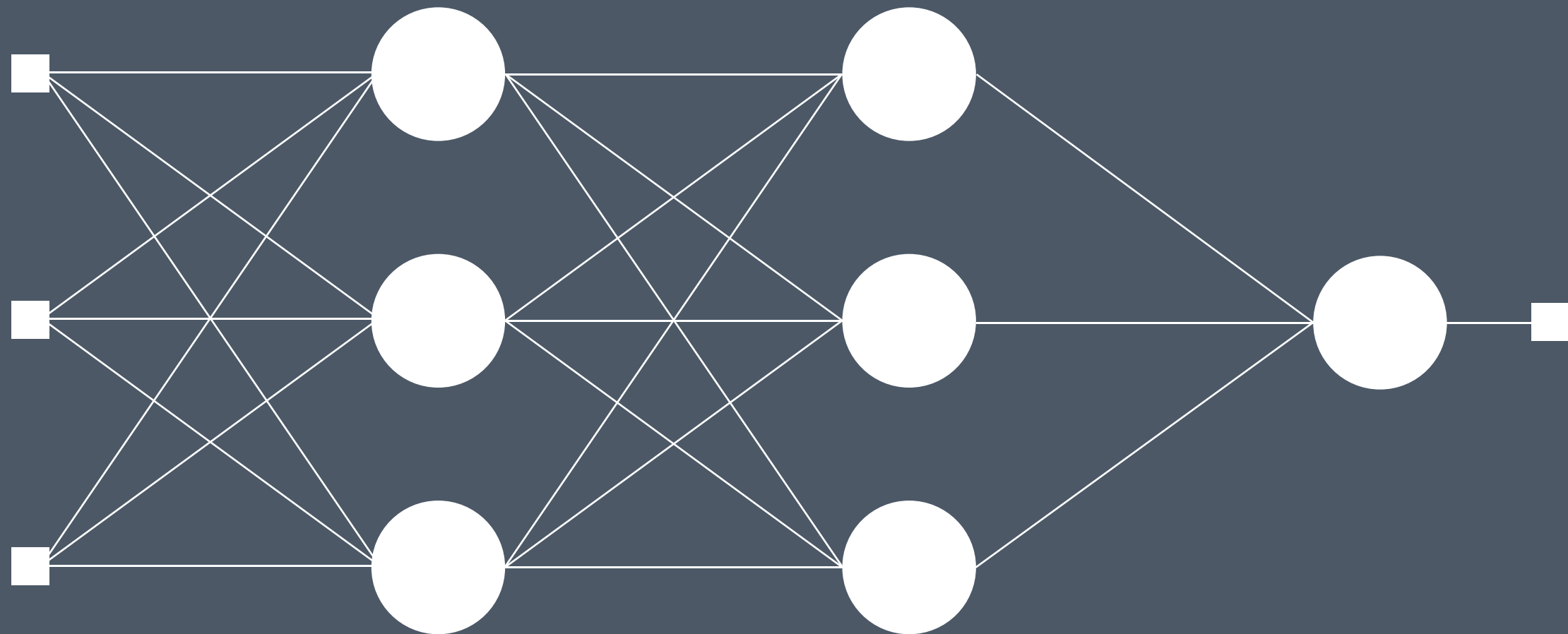
WHAT IS DEEP LEARNING?

Within machine learning, there are various types of algorithms. Think of machine learning algorithms as competing techniques to get the best out of the data. Some algorithms are better in certain aspects, but there's not one that's the best in all departments.

Deep learning is a type of algorithm that's adaptable to varying complexities of data and rules.

It's not necessarily the most accurate, but it's extremely adaptable. And this comes from its modular and flexible form, which will become evident throughout this book.

ALGORITHM



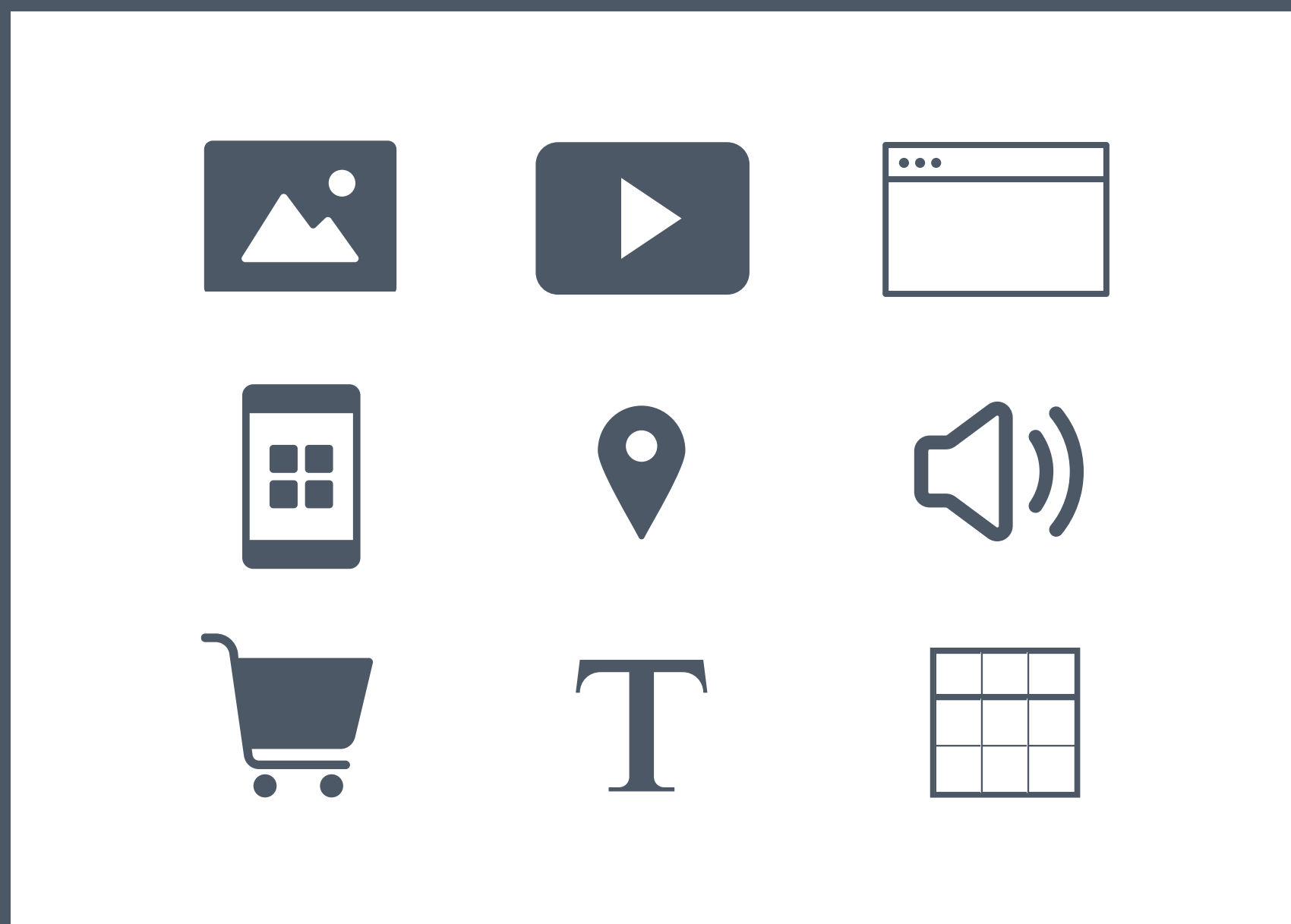
ALGORITHM

In fact, deep learning has revived the push toward Artificial Intelligence (AI) over the past decade.

The progress is gathering pace now is because of three main reasons. The first is the algorithm, which in truth, has been around for many decades.

But that alone is not enough.

DATA



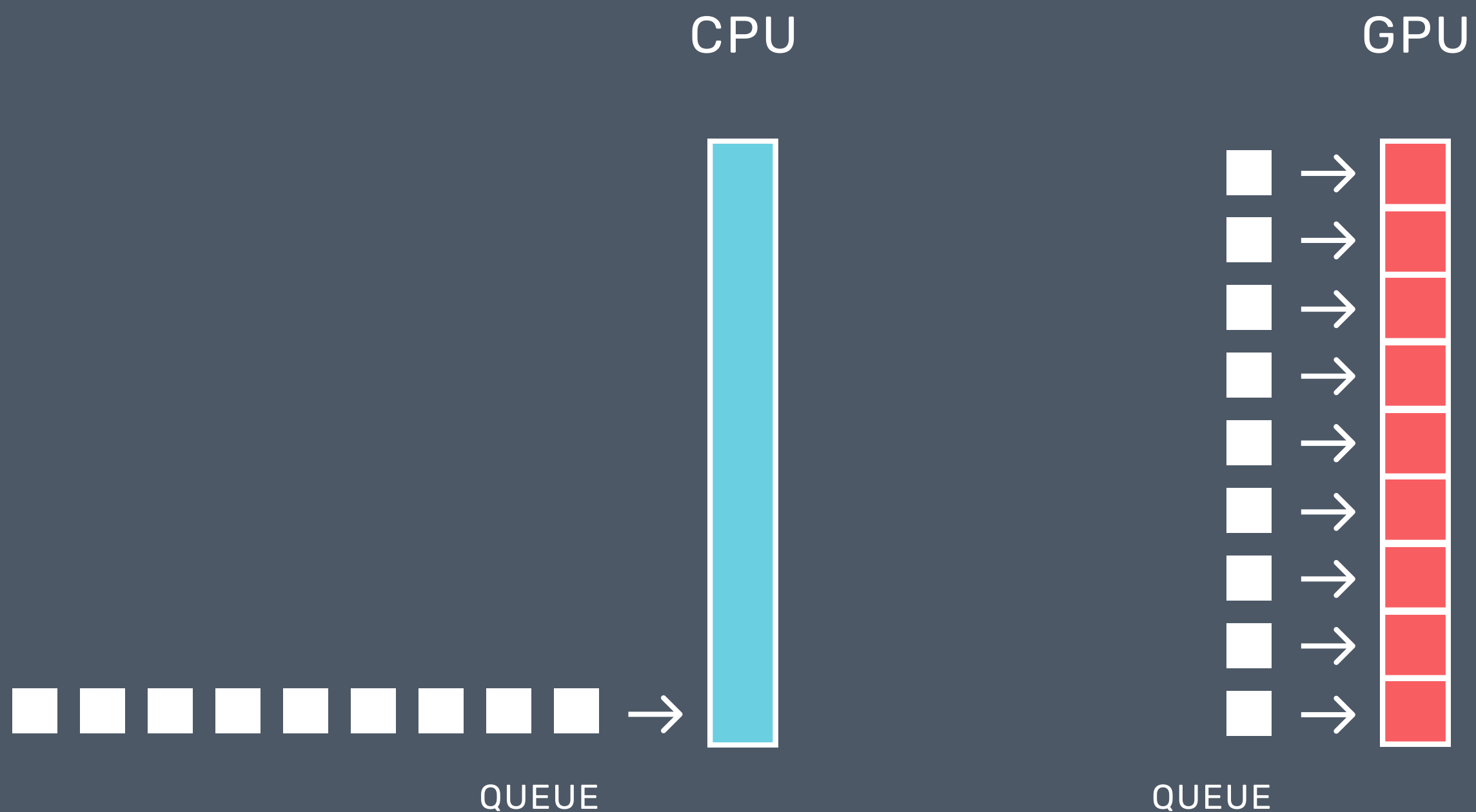
DATA

The second reason is data.

The impetus came from the Internet and followed by social media, smartphones, digital transformation, and a long list of other waves of innovation. They produce new forms of data that we've never seen before, generated in large volumes.

This data contains invaluable information that we can now extract with the help of algorithms.

COMPUTATION



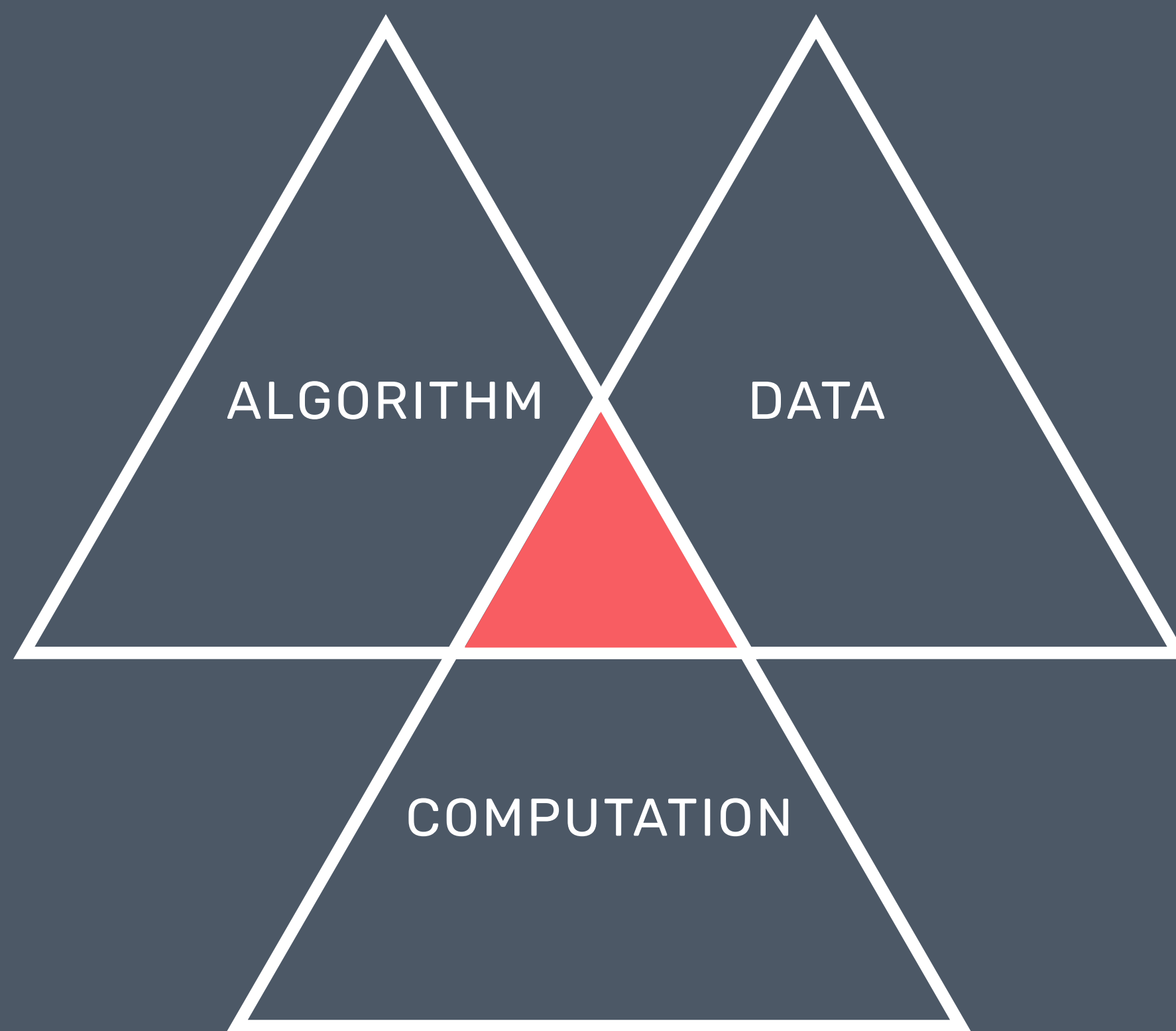
COMPUTATION

The third reason is computational power.

Machine learning involves performing a significant amount of mathematical computation on the data. In deep learning, this is multiplied many times over. The standard Central Processing Unit (CPU) architecture is not capable of handling this task efficiently.

Enter the Graphics Processing Units (GPU). Originally designed for games, it has emerged as the perfect solution for deep learning.

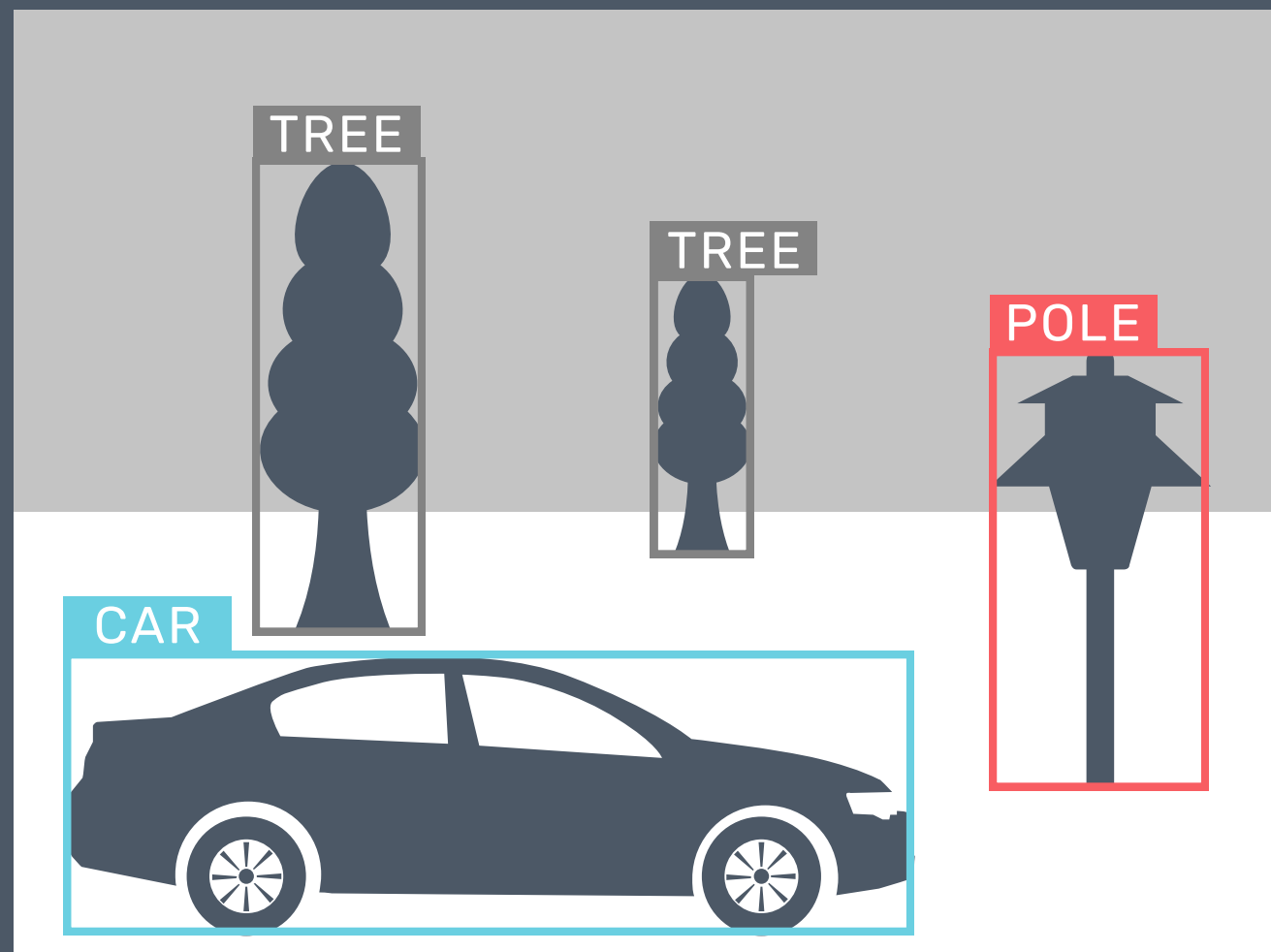
This is a hot area of research as we speak. Even more efficient hardware designs are yet to come.



THE DRIVING FORCES

Together, these three factors are the driving forces behind today's rapid advances in deep learning.

COMPUTER VISION



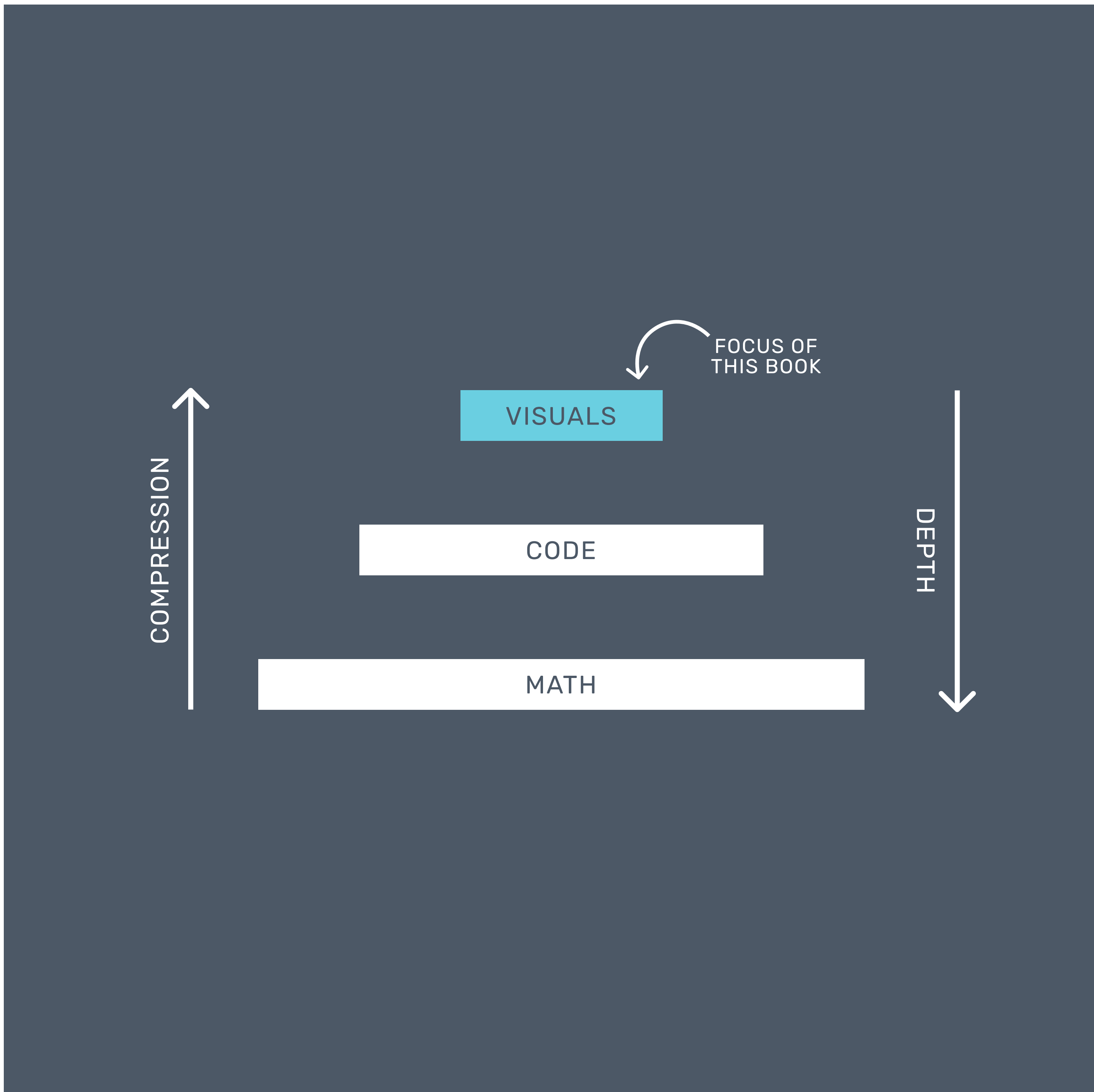
NATURAL LANGUAGE PROCESSING

SENTENCE	SENTIMENT
IT'S A GREAT DAY	POSITIVE
I DON'T LIKE MONDAYS	NEGATIVE

■ ■ ■

APPLICATIONS

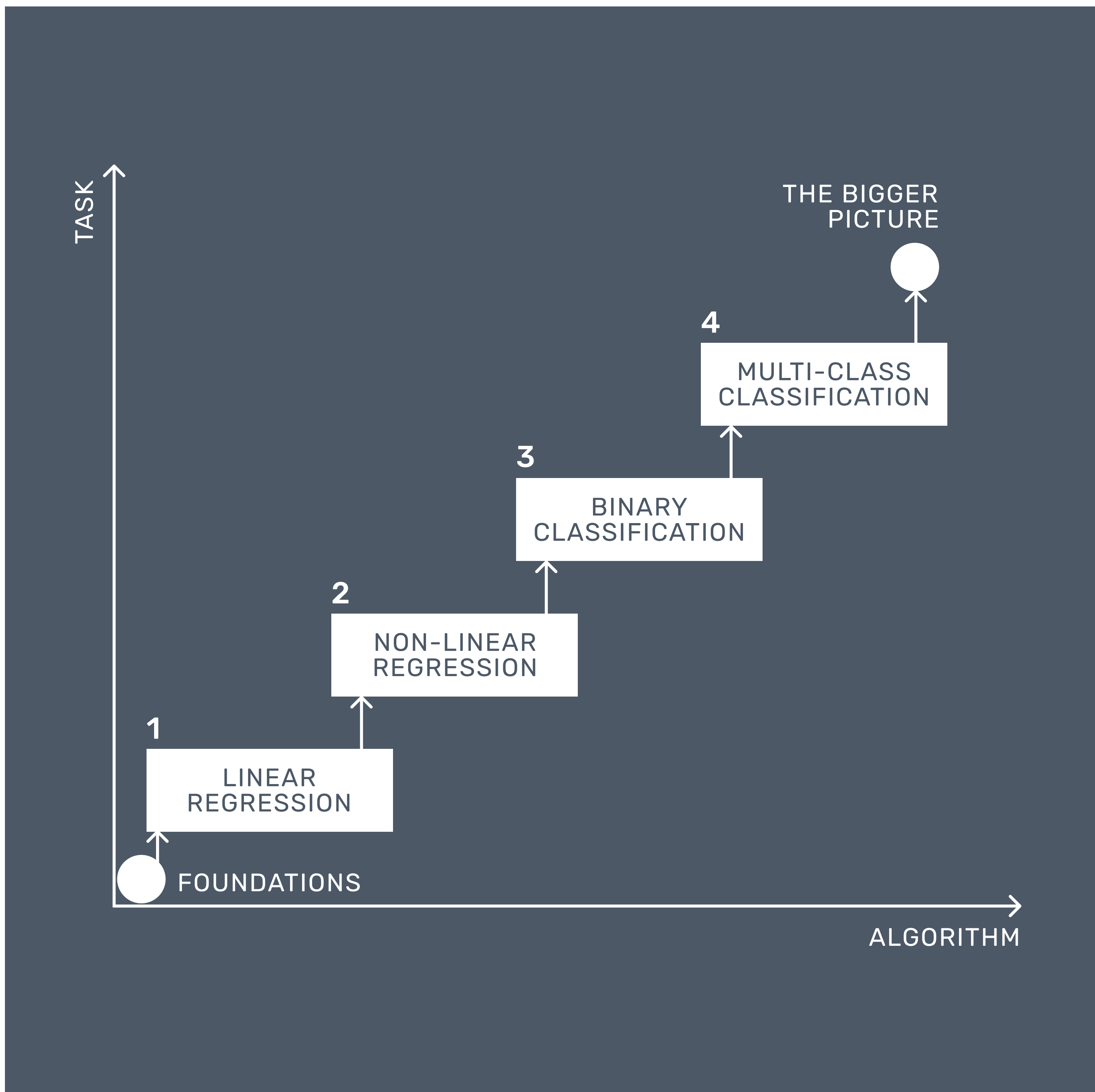
Today, there are widespread applications in computer vision, natural language processing, business automation, and beyond. And it is just the beginning.



WHAT CAN YOU EXPECT FROM THIS BOOK?

By the end of this book, you will be able to build a visual intuition about deep learning and neural networks.

This book doesn't cover mathematical proofs and code examples. As you advance your learning further, these are the domains you should progress into. They will provide you with the depth you need to be successful in this field.

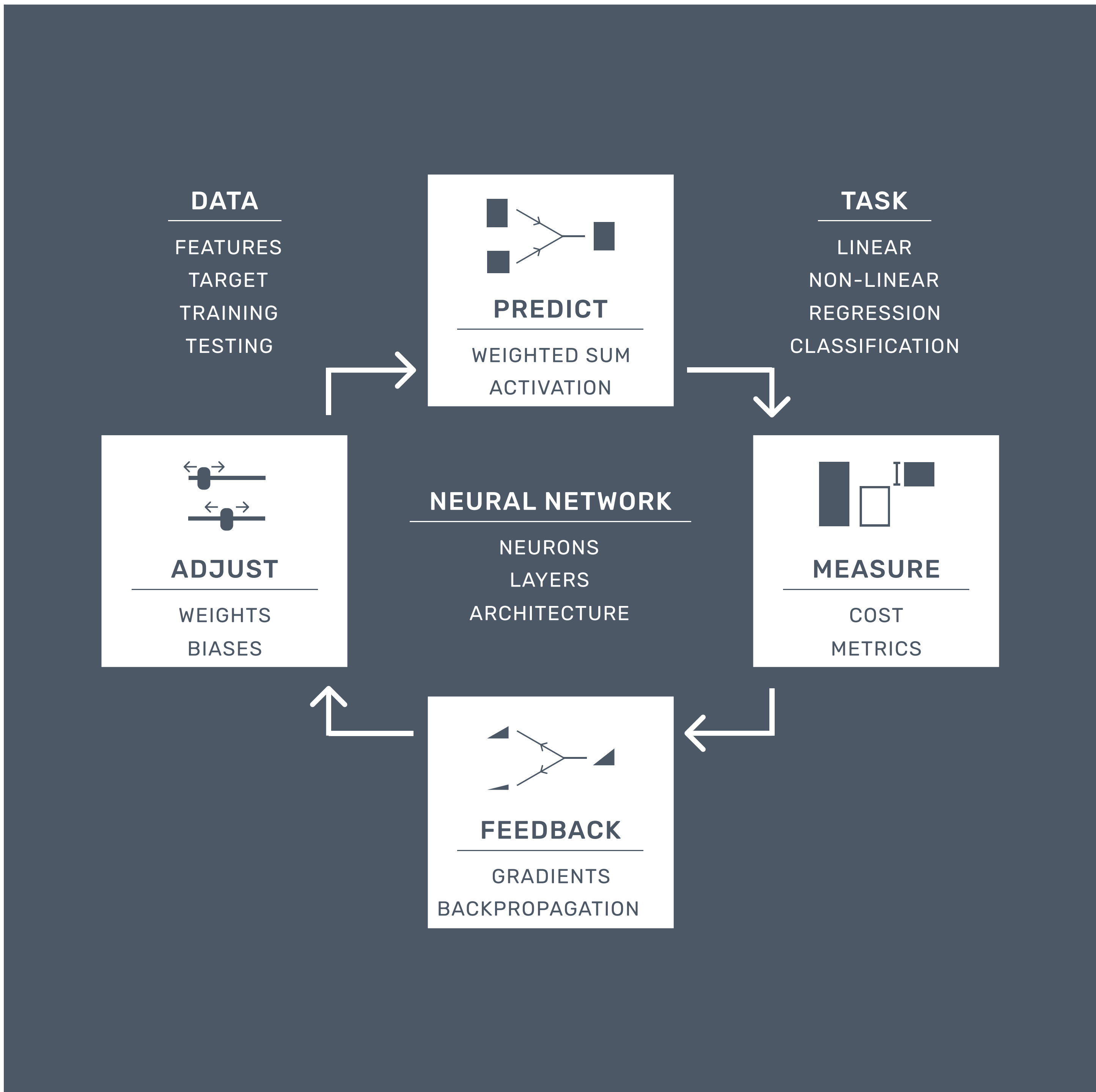


ROADMAP

We'll see how deep learning works via four tasks - linear regression, non-linear regression, binary classification, multi-class classification.

They are correspondingly split into four chapters, in which new concepts are introduced one at a time and built upon the previous ones. Therefore, it is recommended that you read the chapters in sequence.

On either side of these four chapters, we'll have a short section for foundations and a final section where we take a brief look beyond those covered in the four chapters.

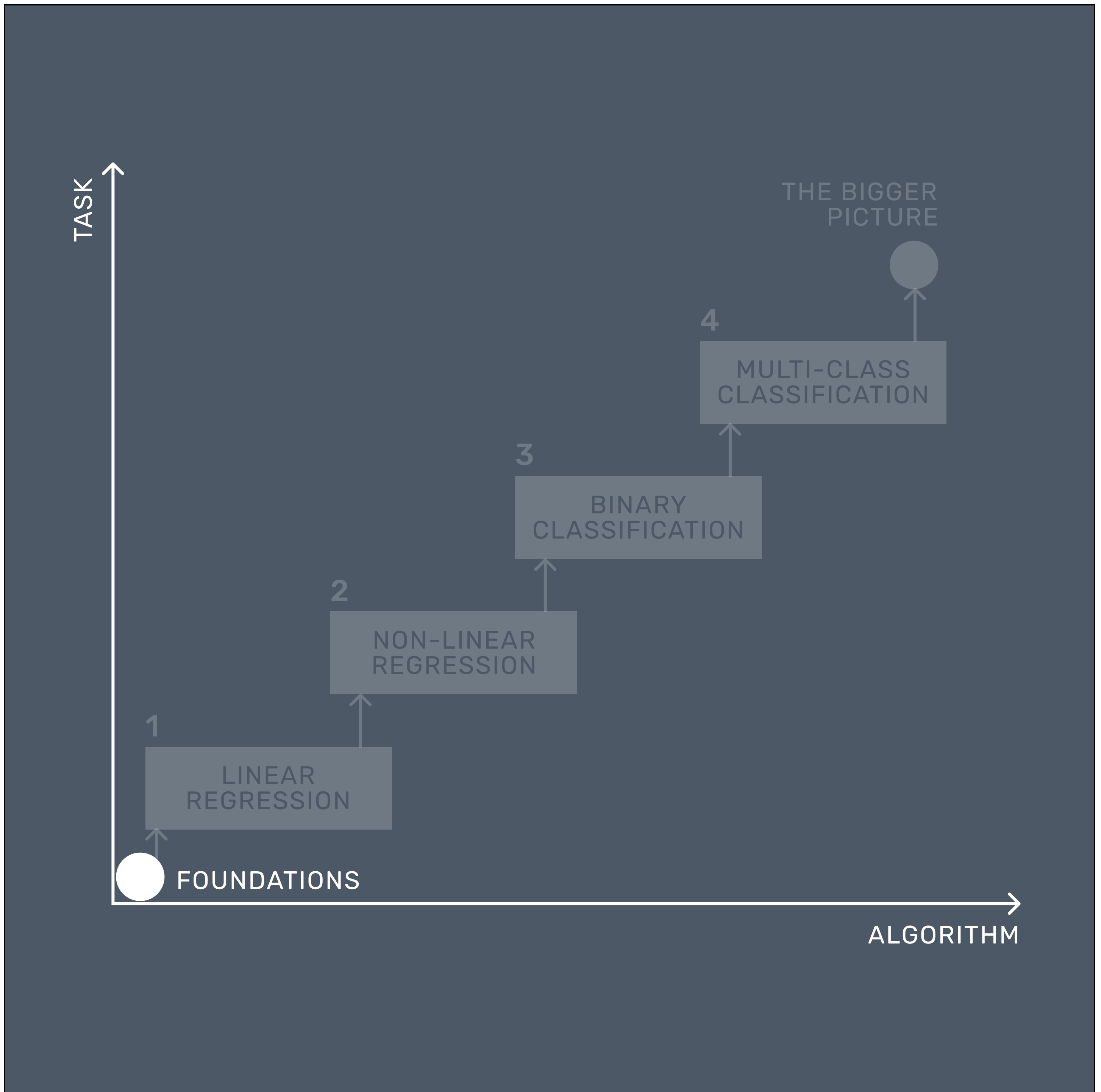


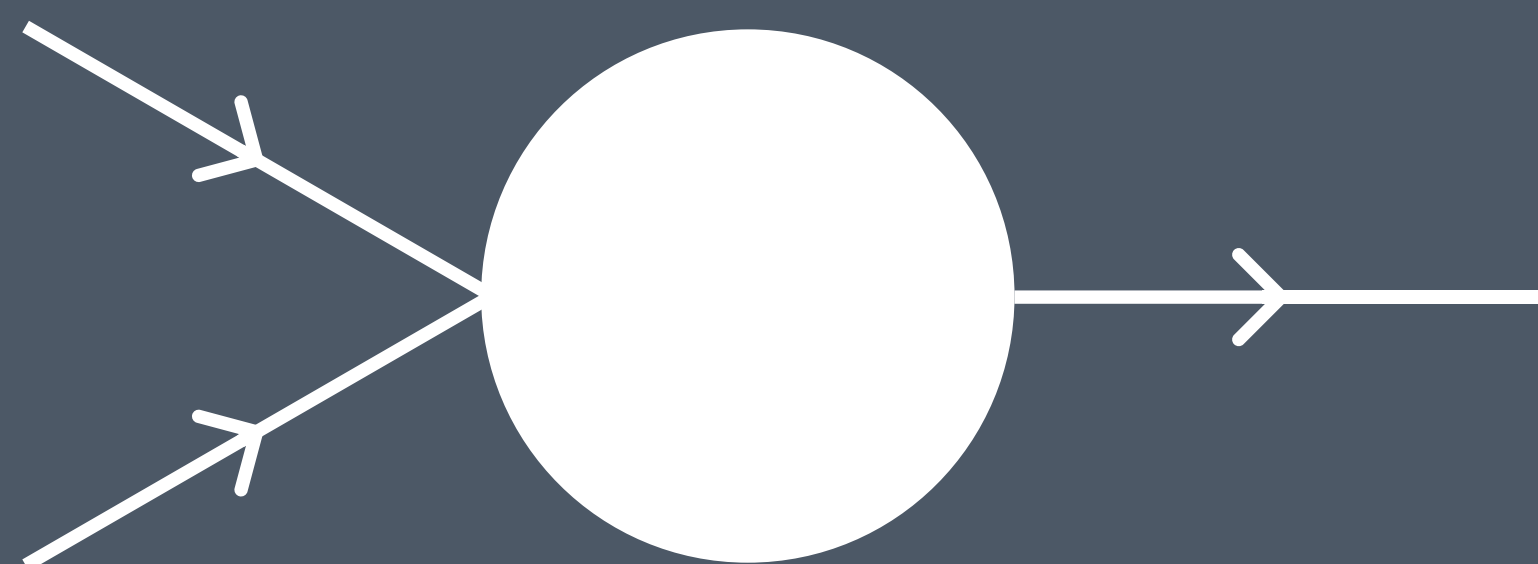
KEY CONCEPTS

Here is a summary of the key concepts that we'll explore in this book. As you go through the book, it'll be useful to return to this page from time to time to keep track of what you have learned.

Let's begin!

FOUNDATIONS



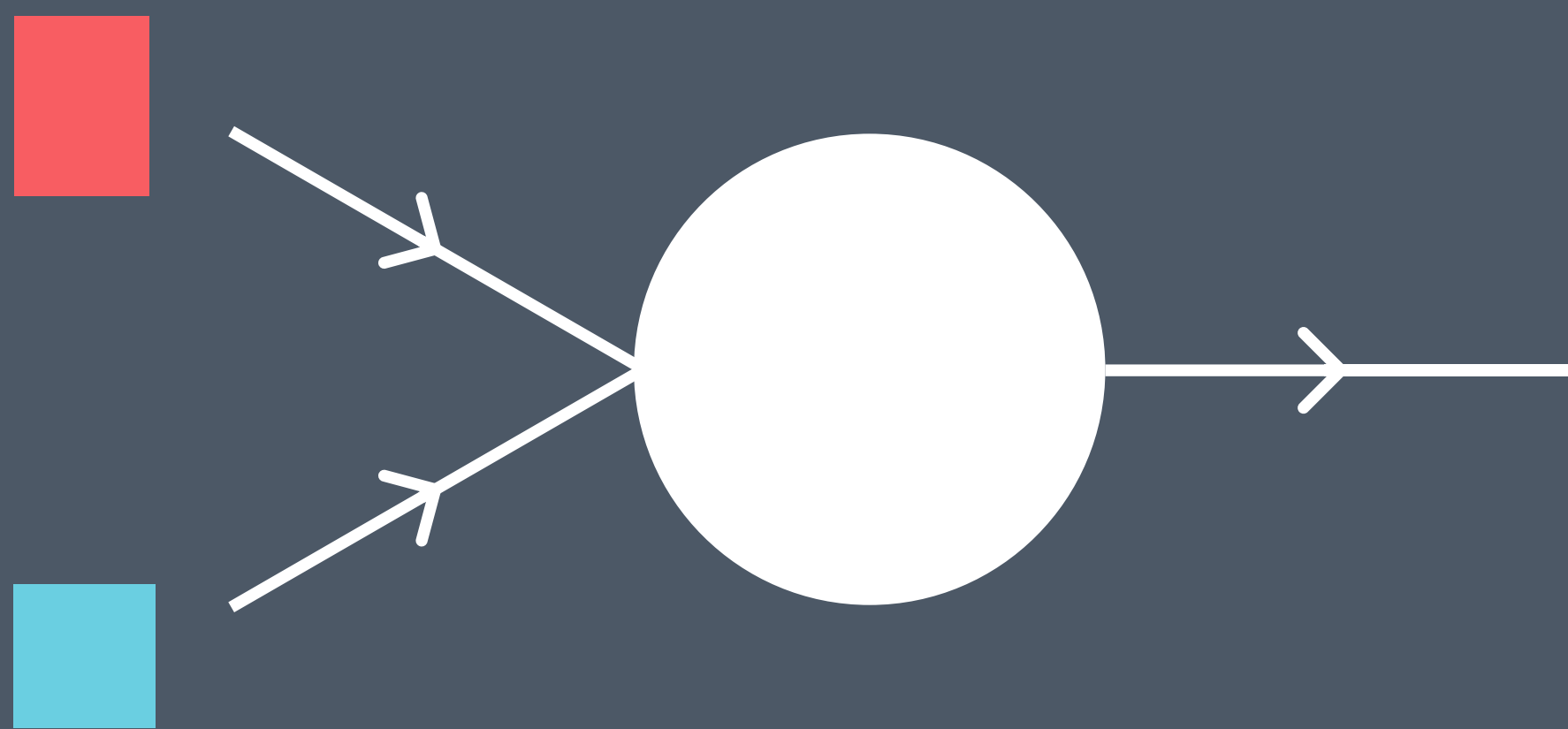


A NEURON

We have so far used the term deep learning, but from now on, we'll use *neural network* instead. These terms are used interchangeably and refer to the same thing. But as we start to go into the inner workings, neural network is a more natural term to use.

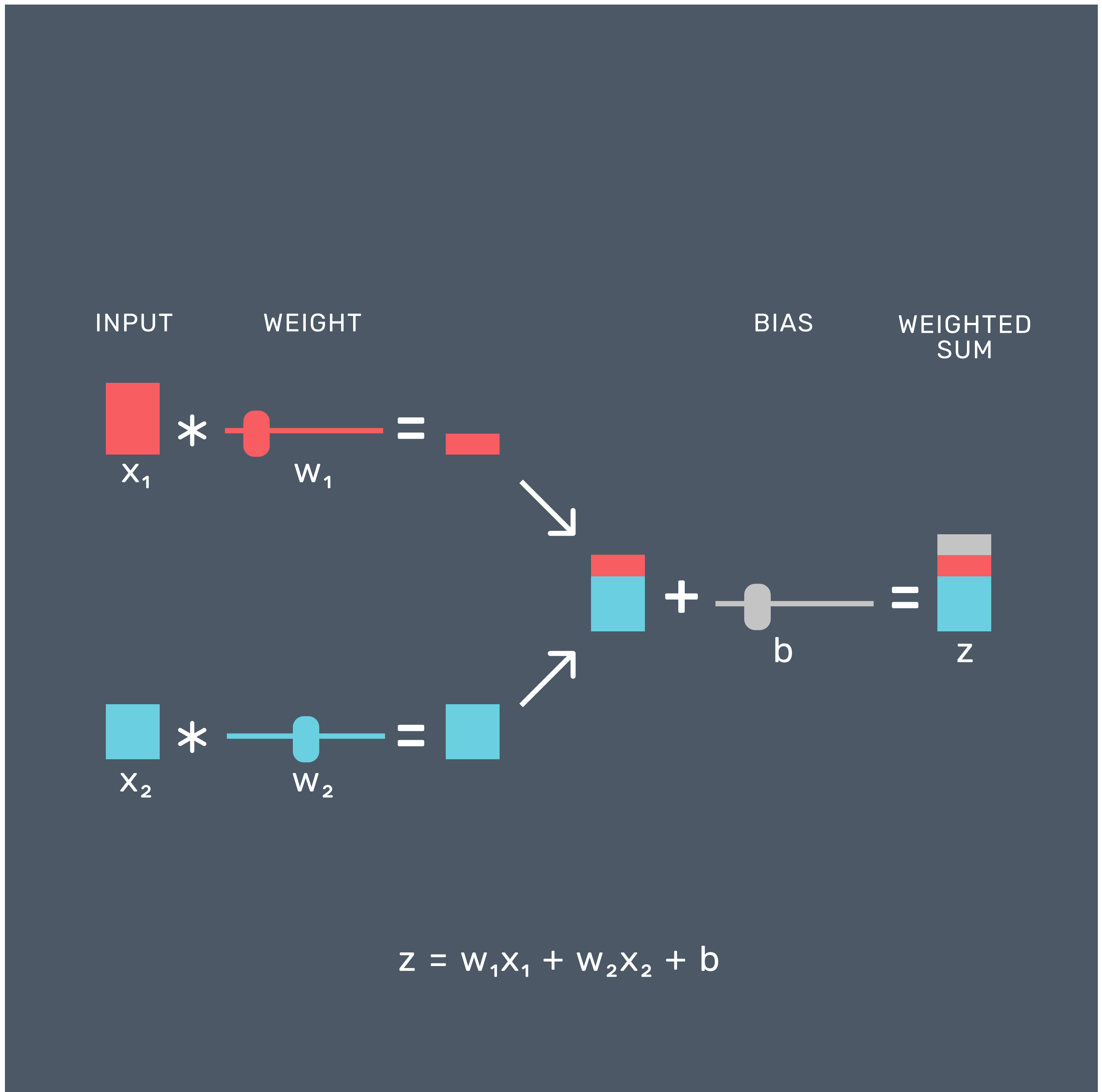
To begin our journey, let's start with a *neuron*. The neuron is the smallest unit and the building block of a neural network.

A neuron takes a set of inputs, performs some mathematical computations, and gives an output.



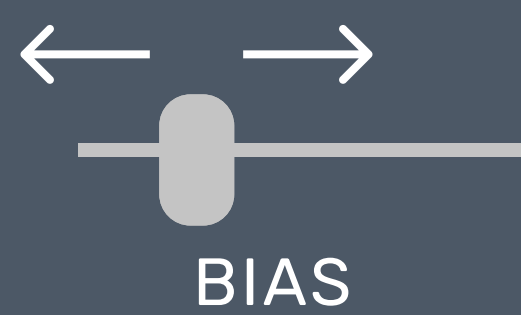
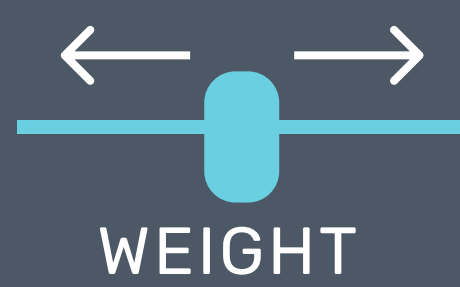
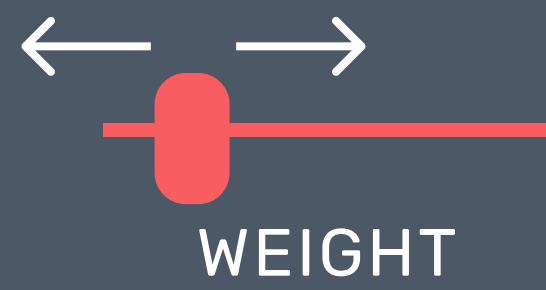
INPUTS

The inputs and outputs are numbers, either positive or negative. In this example, the neuron takes two inputs. However, there is no limit to the number of inputs a neuron can take.



WEIGHTED SUM

The first computation that a neuron performs is the *weighted sum*. It multiplies each input by its corresponding *weight*. Then all the inputs are summed and a term called *bias* is added.



WEIGHTS AND BIASES

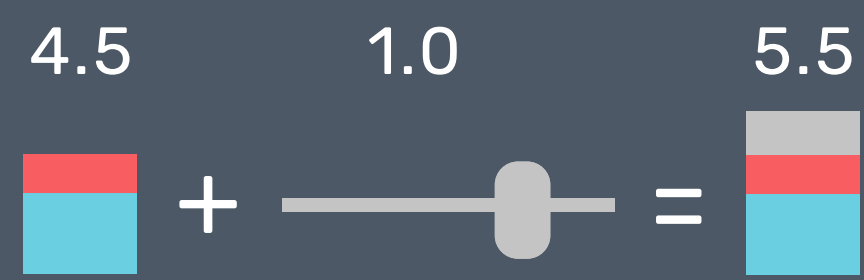
These weights and biases are called the *parameters* of the neural network.

These adjustable parameters are the medium through which a neural network learns, which we'll explore in detail in this book.

EXAMPLE #1

$$3.0 \times 0.5 = 1.5$$

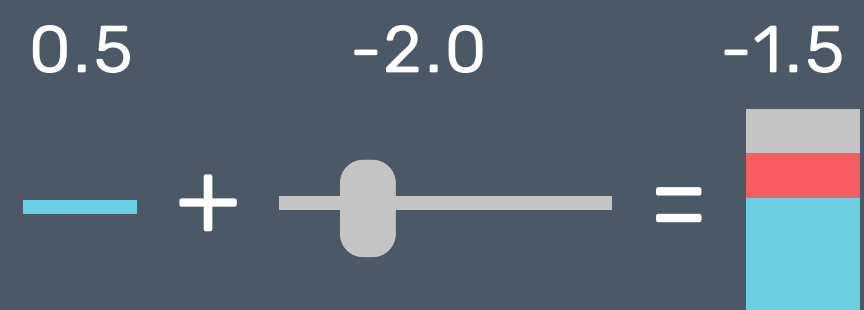

$$2.0 \times 1.5 = 3.0$$


$$4.5 + 1.0 = 5.5$$


EXAMPLE #2

$$3.0 \times -0.5 = -1.5$$

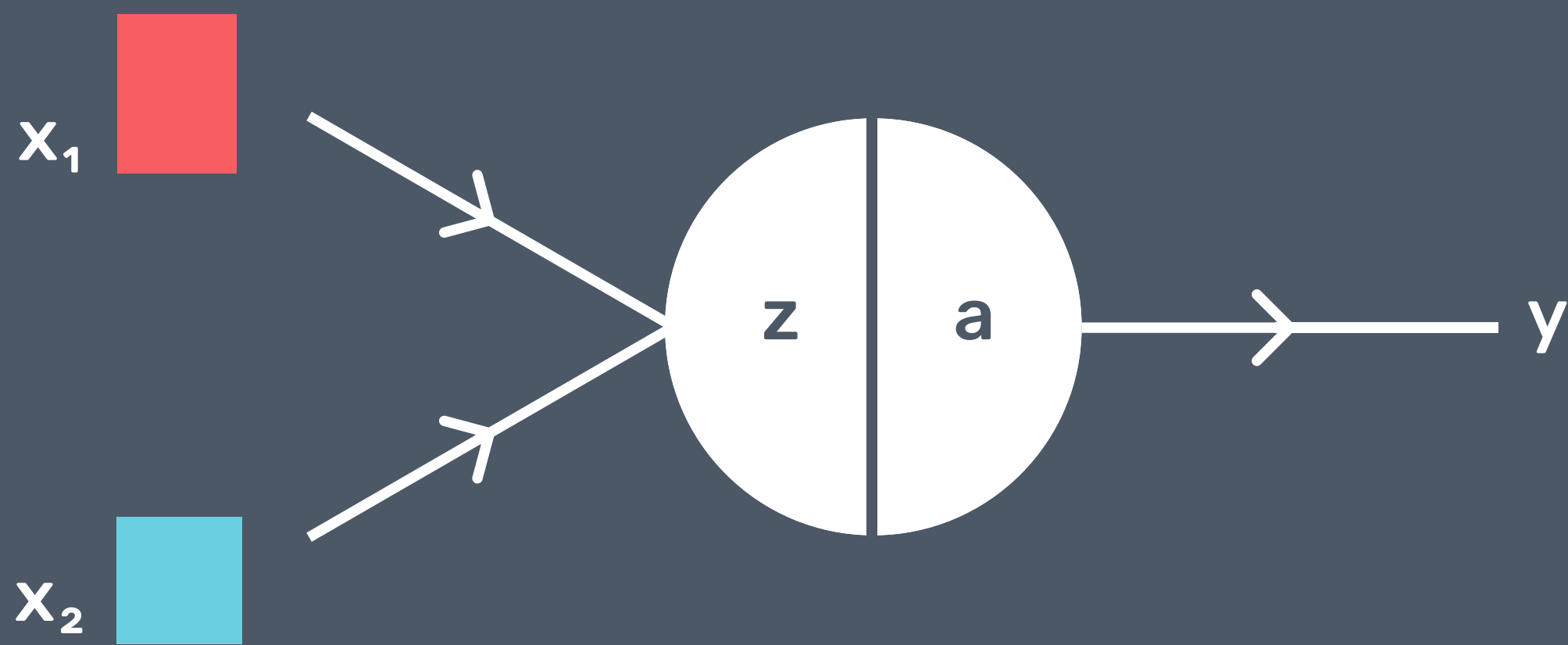

$$2.0 \times 1.0 = 2.0$$


$$0.5 + -2.0 = -1.5$$


EXAMPLE

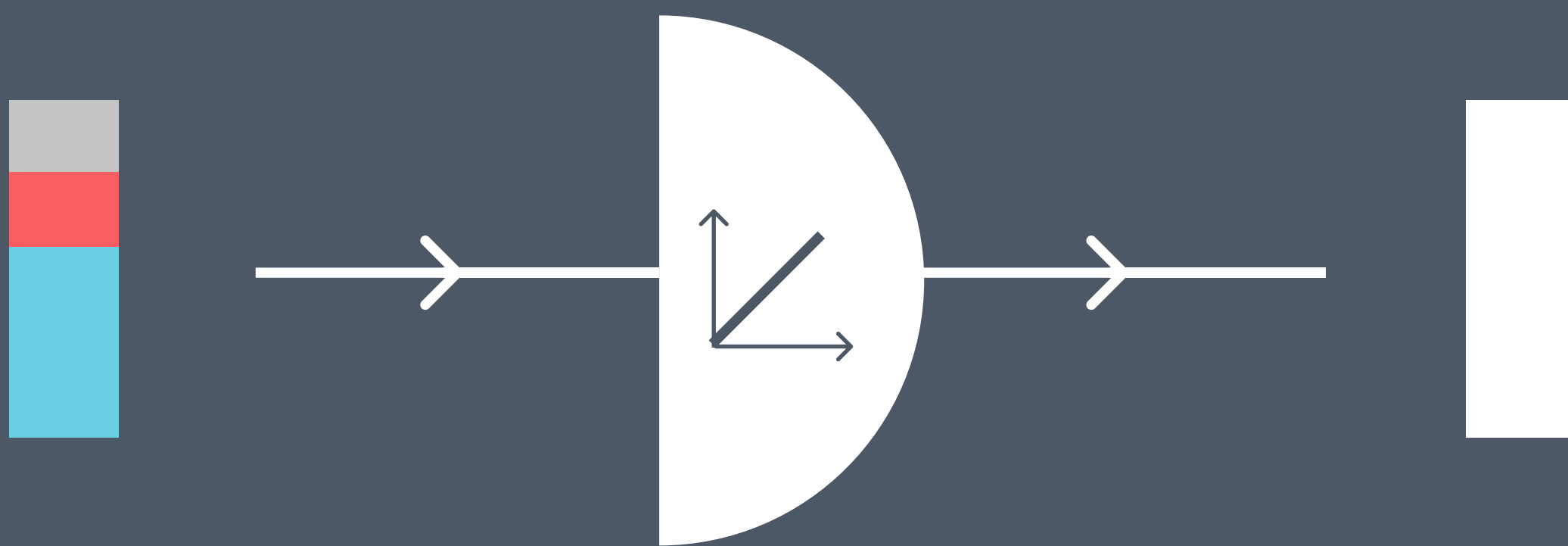
Here we have a neuron with two inputs, 3.0 and 2.0. Given different weight values, it will correspondingly output different values.

WEIGHTED SUM | ACTIVATION



ACTIVATION

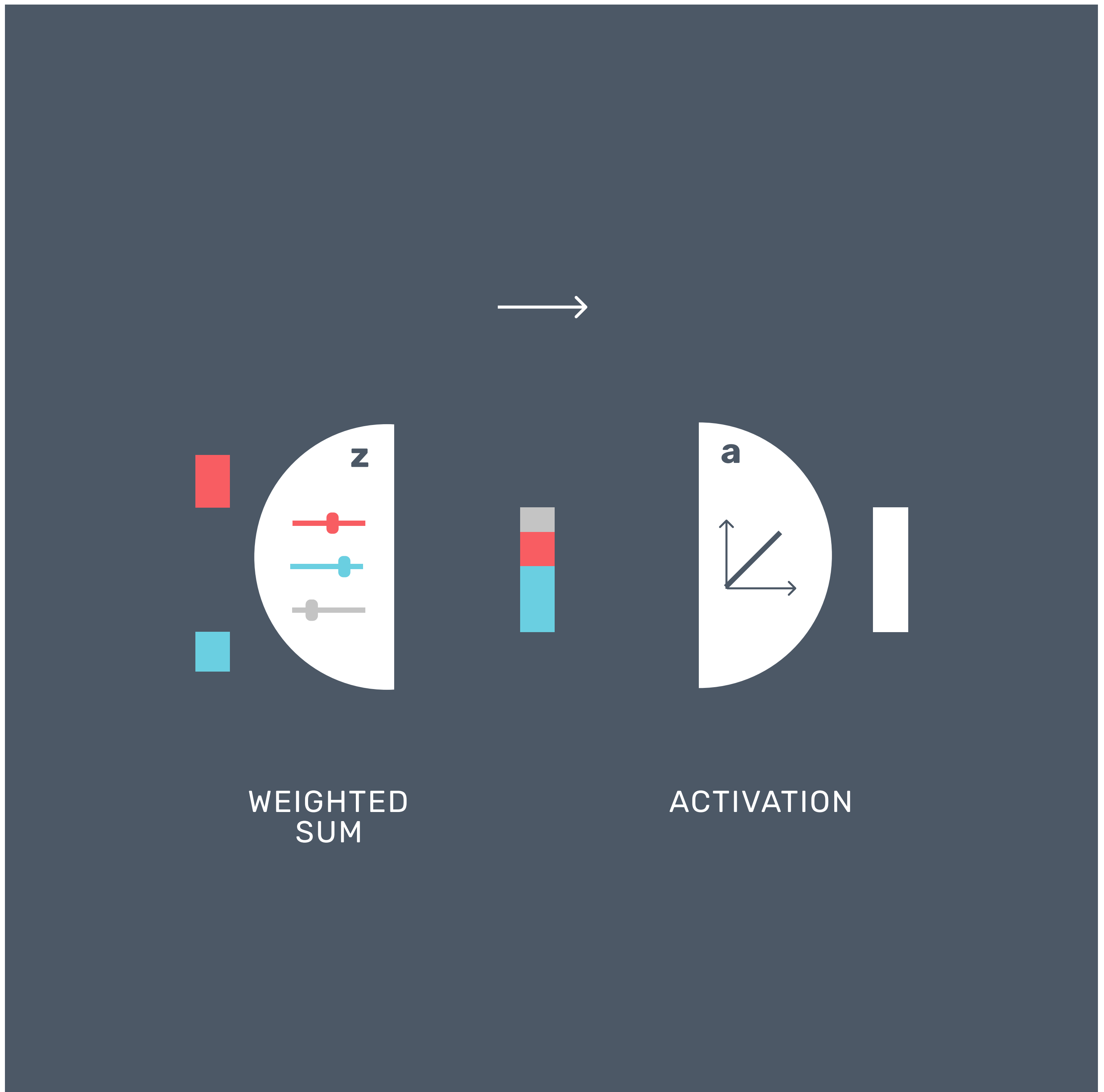
The second computation performed by the neuron is called an *activation*. This is done by taking the output of the weighted sum and passing it through an *activation function*.



LINEAR ACTIVATION

The activation function gives the neural network the ability to express itself. This will not make much sense now but will become clear by the end of this book.

There are a few common activation functions. To start with, here we have a linear activation function. It's as basic as it gets - it simply outputs the same input it receives. Plotted on a chart, it gives a straight-line relationship between the input and the output.

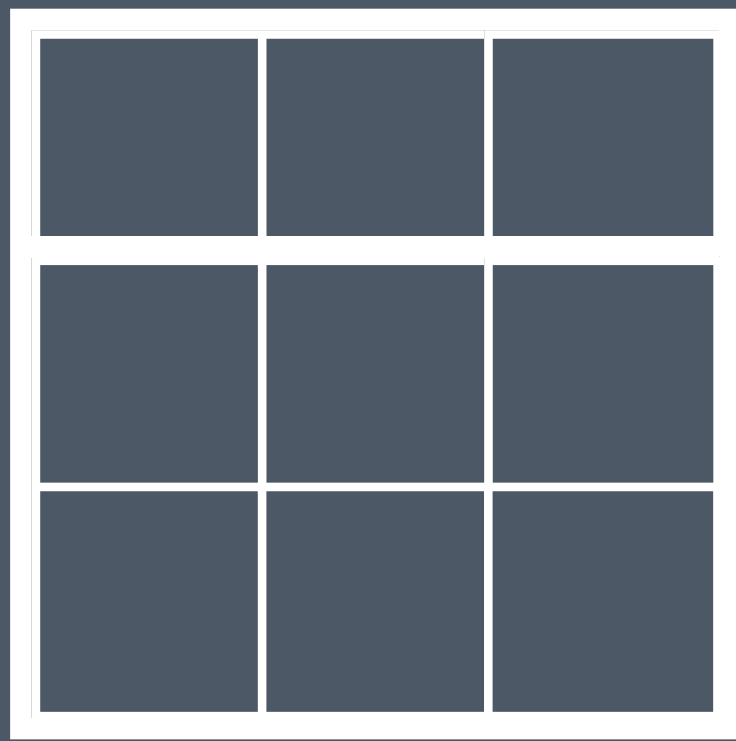


RECAP

Let's do a quick recap. When inputs are passed through a neuron, it performs a sequence of computations.

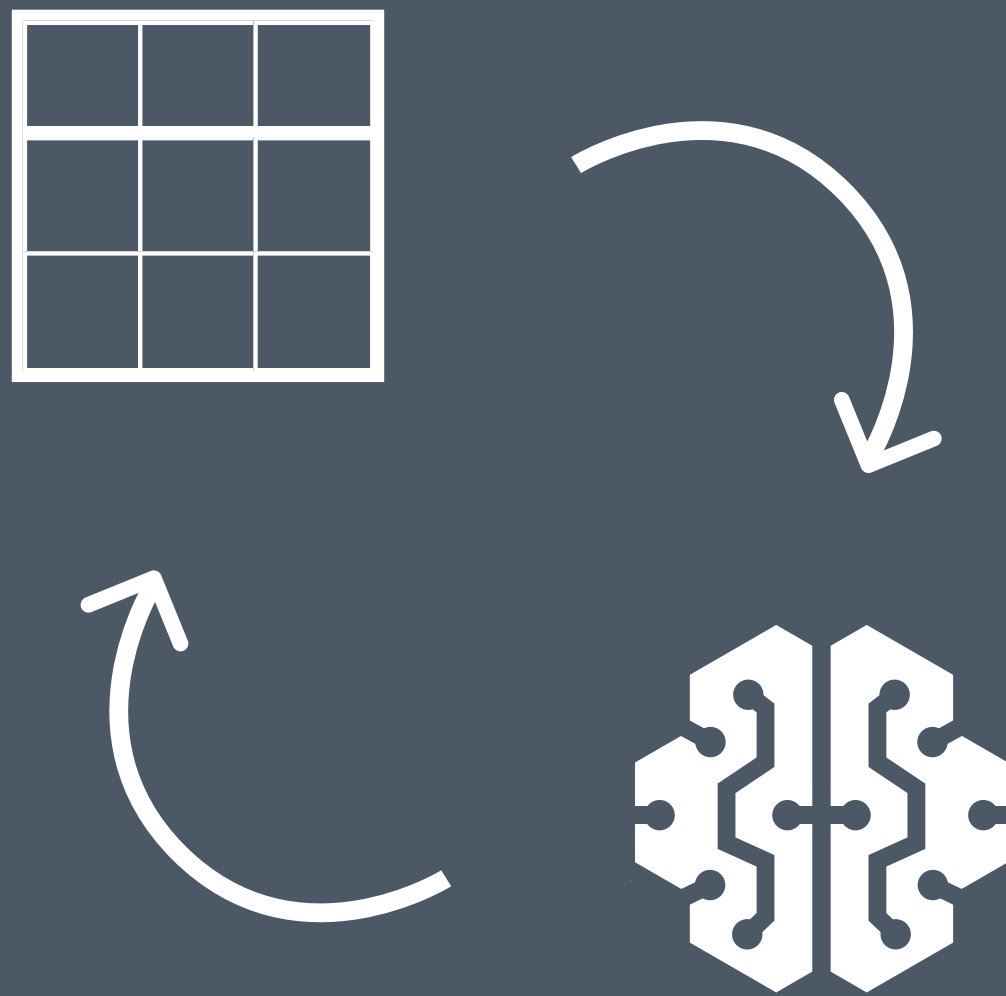
First it performs a weighted sum by multiplying each input by its corresponding weight, summing them, and finally adding a bias term.

Then it performs an activation via an activation function, which in our case, is a linear function.



DATA

Neural networks are nothing without data. Let's now turn our attention to data and what it brings.



LEARNING

Data is to the neural network as experience is to humans.

A machine learning algorithm, in this case a neural network, uses data to find useful patterns and relationships. It uses these insights to learn and update itself.

STRUCTURED DATA

XX	X	X
X	XX	X
XX	X	X

SEMI-STRUCTURED DATA

```
XX : {  
  XX : X,  
  X : XXXXX,  
  XXX : {  
    XX : XXX,  
    XXXX : XX,  
  }  
}
```

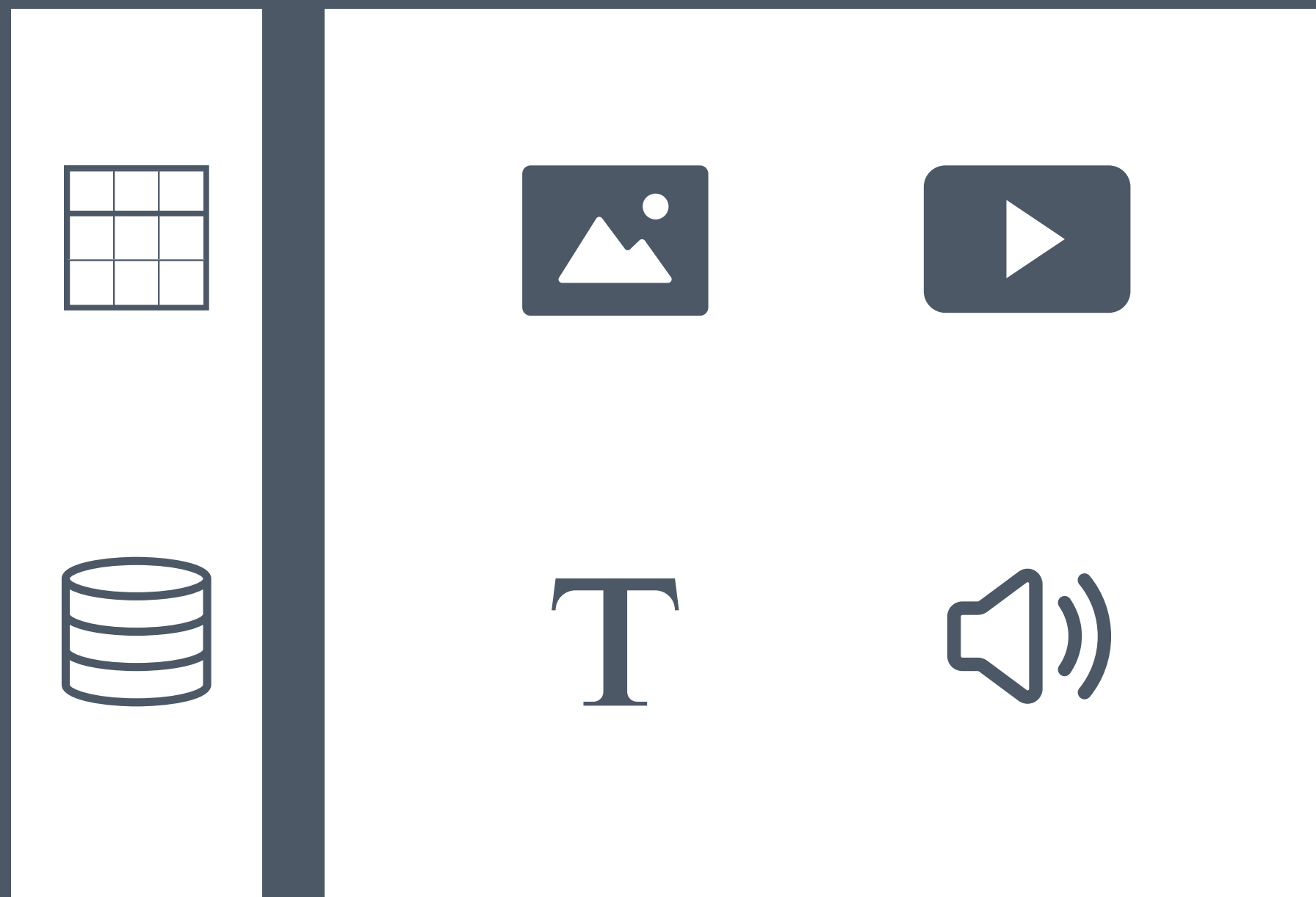
UNSTRUCTURED DATA

```
XXX XX XXXXXX X XX  
XX X XXX X XX XXXX.  
XXX, X XXX.  
  
X XXX XXXX, X XX  
XXXXX XX, XXXX X XX X.  
XXX X XX XX XXX X XX  
XX XX.
```

TYPES OF DATA

Data can come in many different forms. The most obvious form of data is the tabular format. This is an example of *structured* data, where each data point and its properties can be deciphered in a straightforward manner.

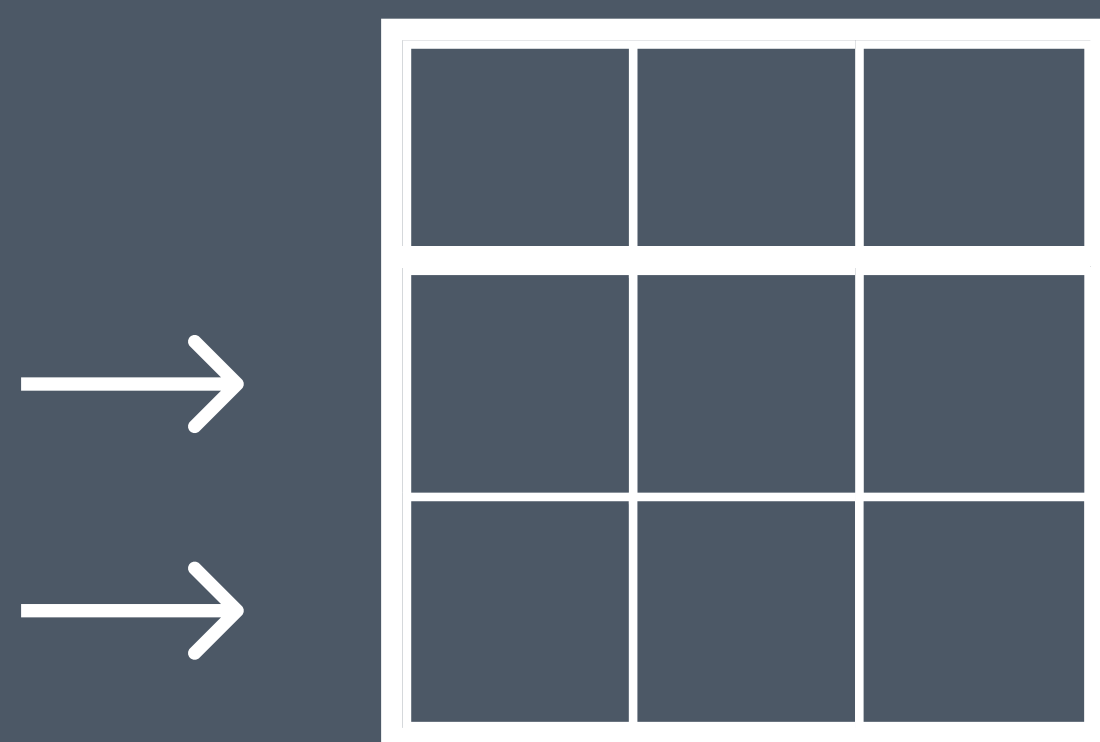
But data can come in other forms too.



SOURCES OF DATA

In fact, most of the data around us are in the *unstructured* form. According to projections from IDC, 80 percent of worldwide data will be unstructured by 2025.

And indeed, most of the exciting innovations in deep learning today come from unstructured data, such as text, images, videos, and so on.



A DATASET

Now let's look at how we should prepare a *dataset* for the neural network.

A dataset is composed of many data points. A *data point* is a single observation that we collect and record.



DISTANCE (MI)	RATING
1.5	3.6

EXAMPLE

Let's take the example of hotel room rates, a dataset we'll use throughout this book.

Each data point represents a hotel. Here we have a hotel with a distance of 1.5 miles from the city center and a guest rating of 3.6 stars.

FEATURES

DISTANCE
(MI)

RATING

FEATURES

These two pieces of information are called *features*. Features describe the properties of a data point.

Each data point in a dataset is described with the same features, of course with different values, making each of them unique.

From now on, we'll refer to these two features as *distance* and *rating* for short.

TARGET

PRICE
(\$)

TARGET

Recall that the goal of a neural network is to make predictions.

In this example, we want to predict the average daily room rate (or *price* for short) for any given hotel. This means, given the two features earlier, we want to predict how much each hotel will cost.

This is called the *target*. In other resources, you may also find the term *label* being used.

	DIST (MI)	RATING	PRICE (\$)
→	0.8	2.7	147
→	1.5	3.6	136

→	19.4	4.8	209

TRAINING

We'll give the neural network enough data points containing the features and target values, which it will learn from.

A machine learning task where we specify the target is called *supervised learning*, which will be our focus in this book.

TRAINING



TRAINING

We have just described a process called *training*.

During training, the neural network learns from the data and updates its parameters. By this point, we'll have a trained *model*.

In short, given a dataset containing features and target, we get a model.

That is why the training process is sometimes also called 'fitting the model to the data'.

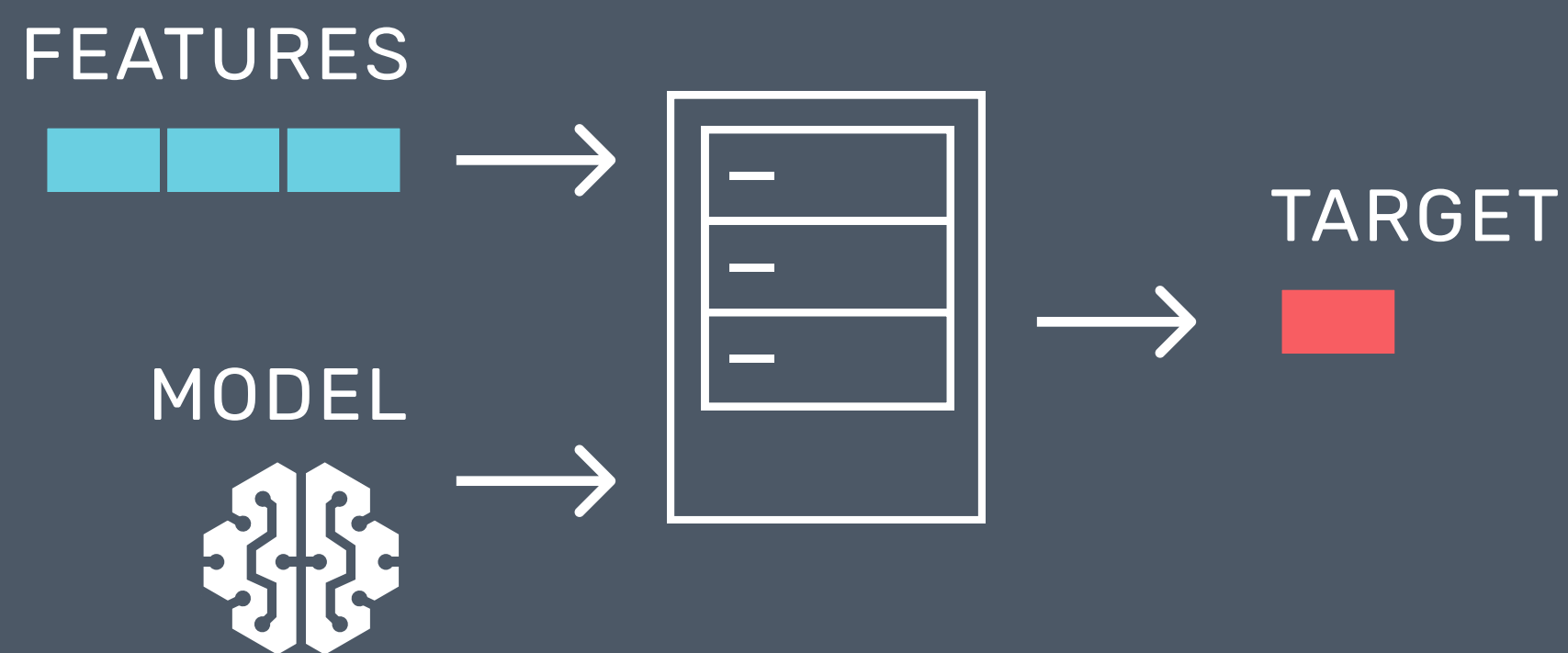


TESTING

Once the training is complete, we need a way to know how the model is performing.

For this, we'll keep a portion of the dataset for *testing*.

TESTING

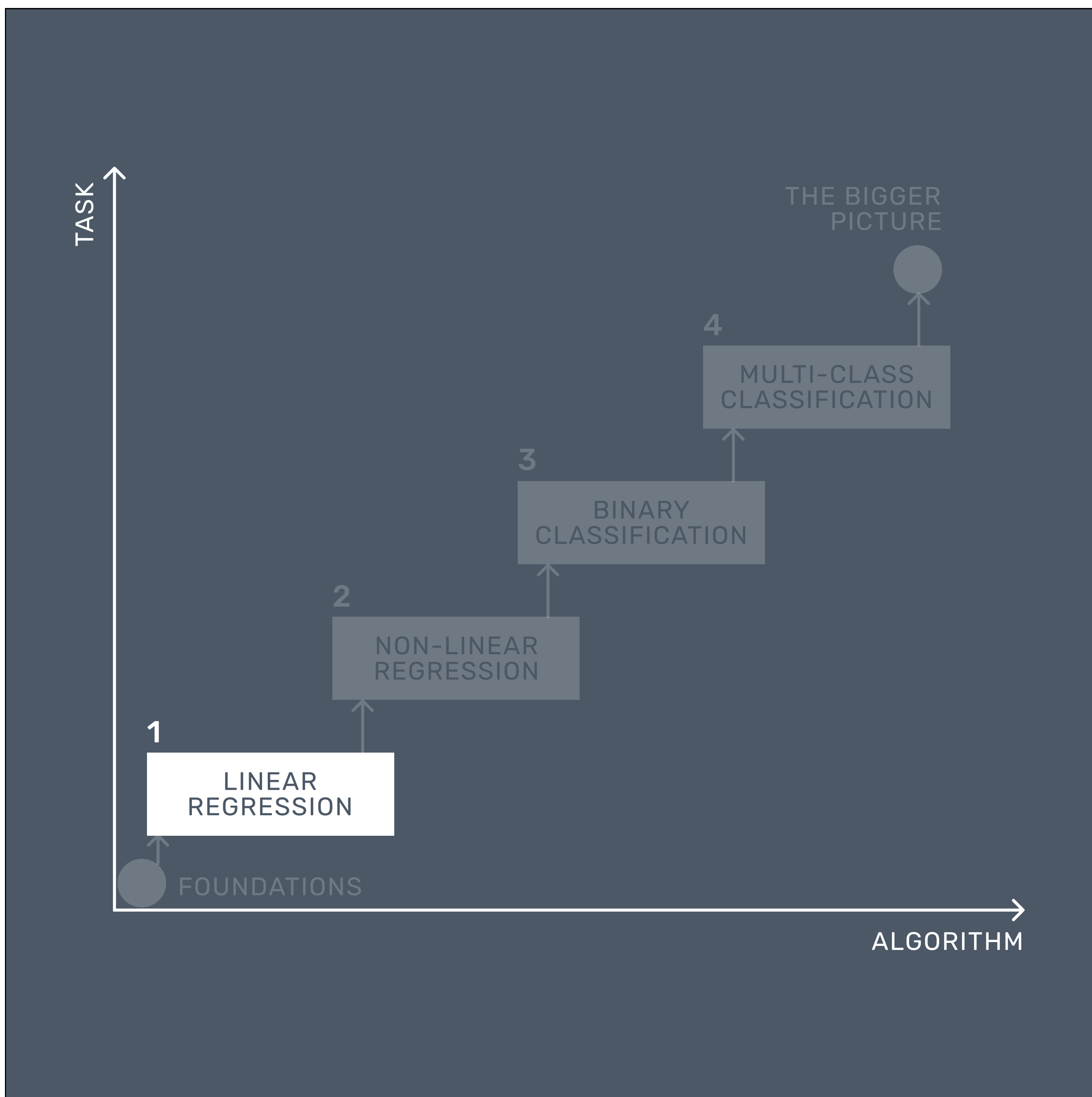


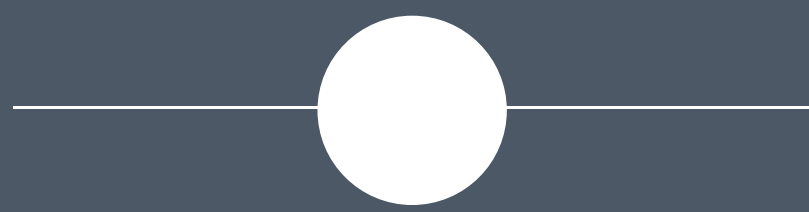
TESTING

During testing, we'll provide the neural network with just the features, without the target. Now that it's already trained, it's job is to predict the target values.

In the coming four chapters, we'll revisit these training and testing steps.

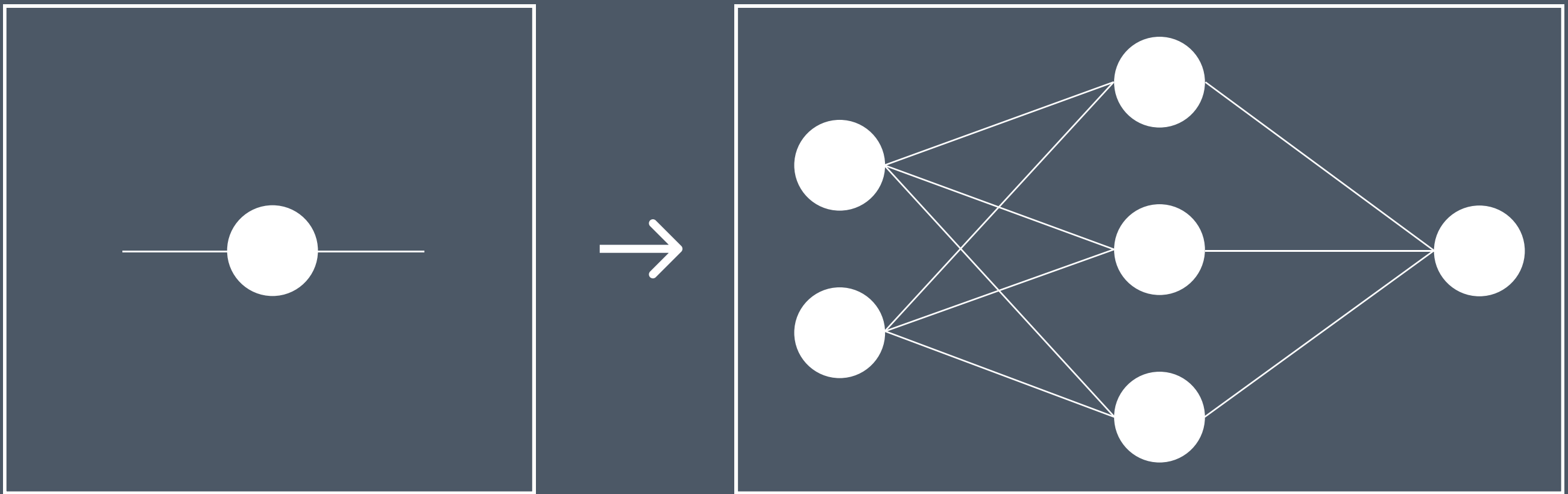
1 - LINEAR REGRESSION





A SINGLE-NEURON NEURAL NETWORK

Now let's look at how the neural network works. We'll start with its simplest possible version—a network with only one neuron and one input!



THE PLAN

We'll lay the necessary foundation in this chapter and use that in the subsequent chapters when we start building larger neural networks.



THE GOAL

Let's revisit the dataset from the previous chapter, which contains a list of hotels in a city.

Our goal is to predict the average daily room rate for a given hotel (i.e. *price*) based on the features.

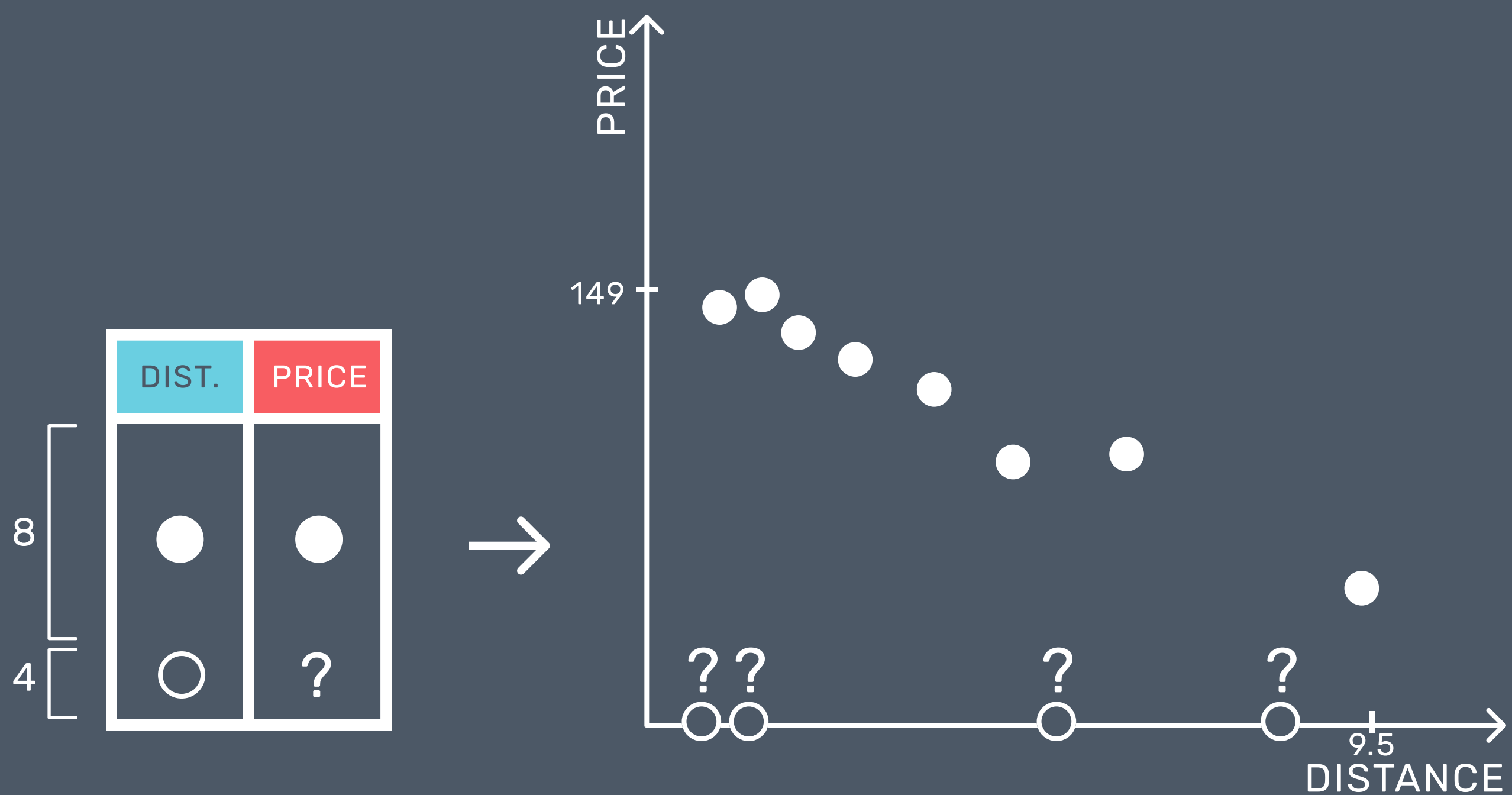
In this chapter, we'll use only one of the features—the distance from the city center (i.e. *distance*).

DISTANCE (MI)	PRICE (\$)
0.5	146.00
1.1	149.00
1.6	140.00
2.4	134.00
3.5	127.00
4.6	110.00
6.2	112.00
9.5	81.00
0.3	156.00
0.7	168.00
4.9	116.00
8.5	99.00

THE DATASET

This is what the dataset looks like. It contains twelve data points, one feature, and one target.

The distance and price values are *continuous values*—numeric values that can take any value within a range.

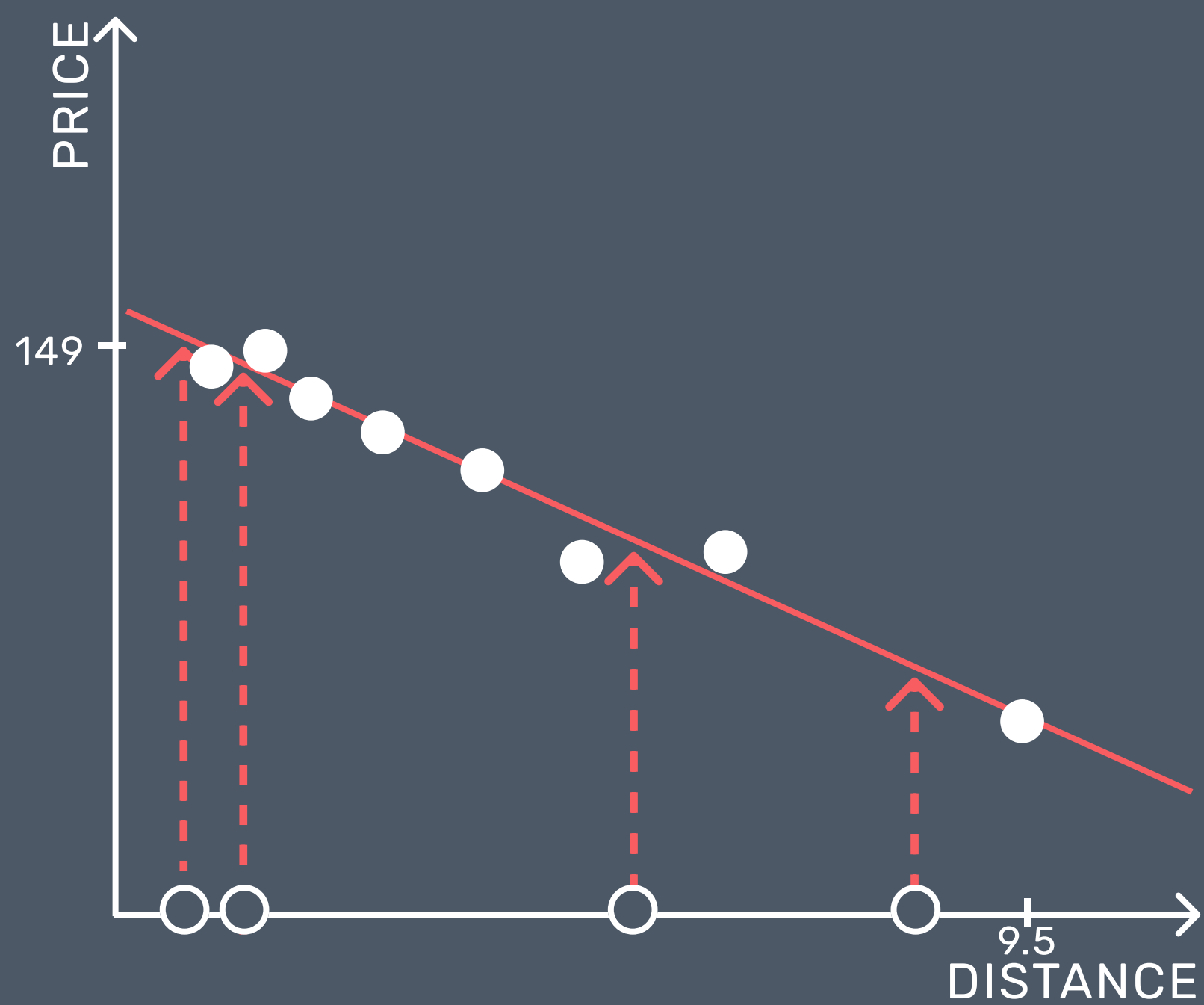


REGRESSION

This is a type of task called *regression*. In regression tasks, the target value to be predicted is a continuous value.

We'll split the dataset into the training and testing datasets and train the model using the training data.

Ultimately, we want the model to give good predictions for the four test data points.



LEARNING

By visual inspection, it's clear that there is a straight-line relationship between the feature and the target. This is called *linear regression*.

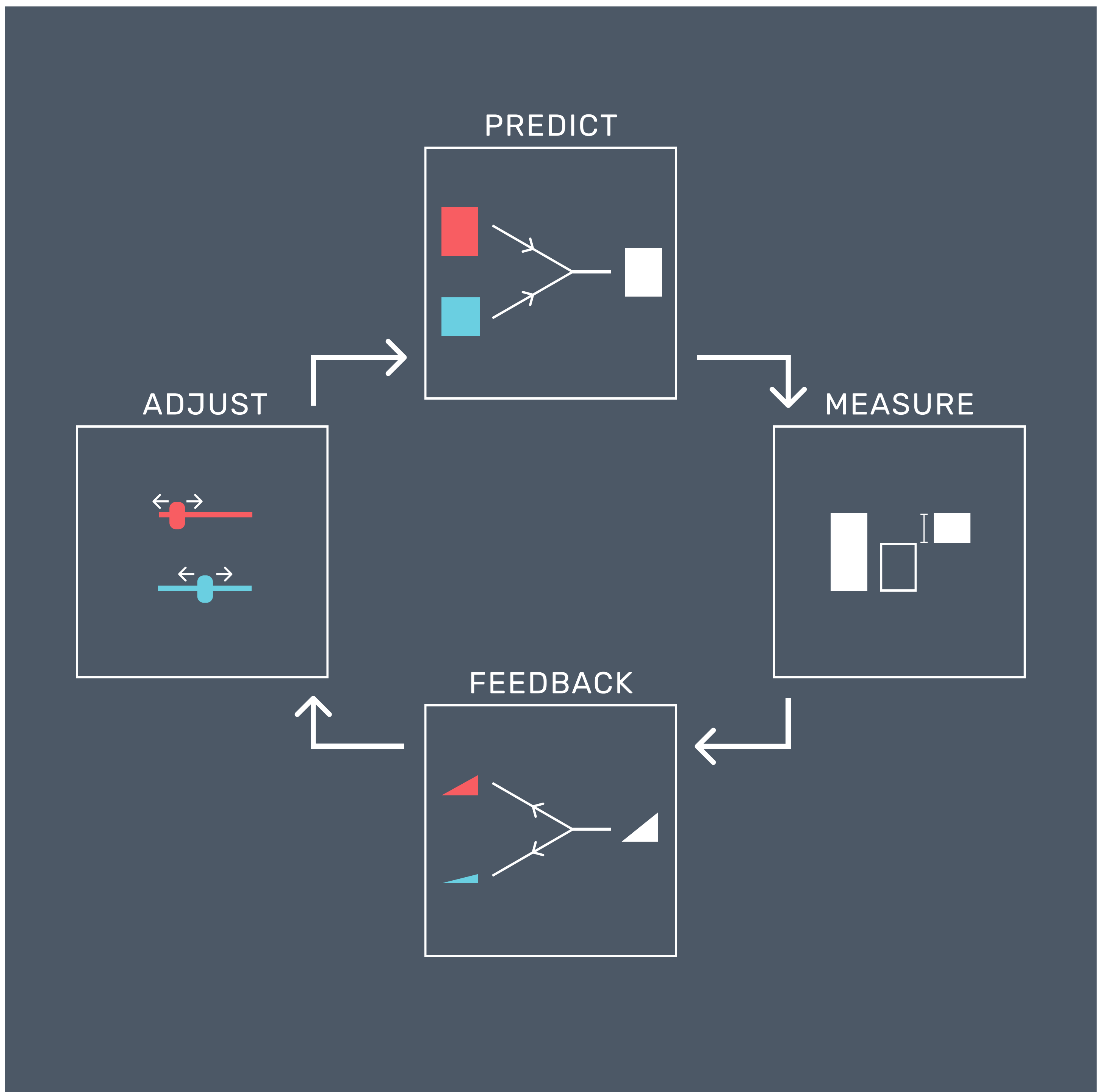
This is the relationship that we want our single-neuron network to capture. Let's see if it can do that.

TRAINING
DATA

DISTANCE (MI)	PRICE (\$)
0.5	146.00
1.1	149.00
1.6	140.00
2.4	134.00
3.5	127.00
4.6	110.00
6.2	112.00
9.5	81.00
0.3	156.00
0.7	168.00
4.9	116.00
8.5	99.00

TRAINING DATA

We'll take the first eight data points as training data and leave the other four for testing later.



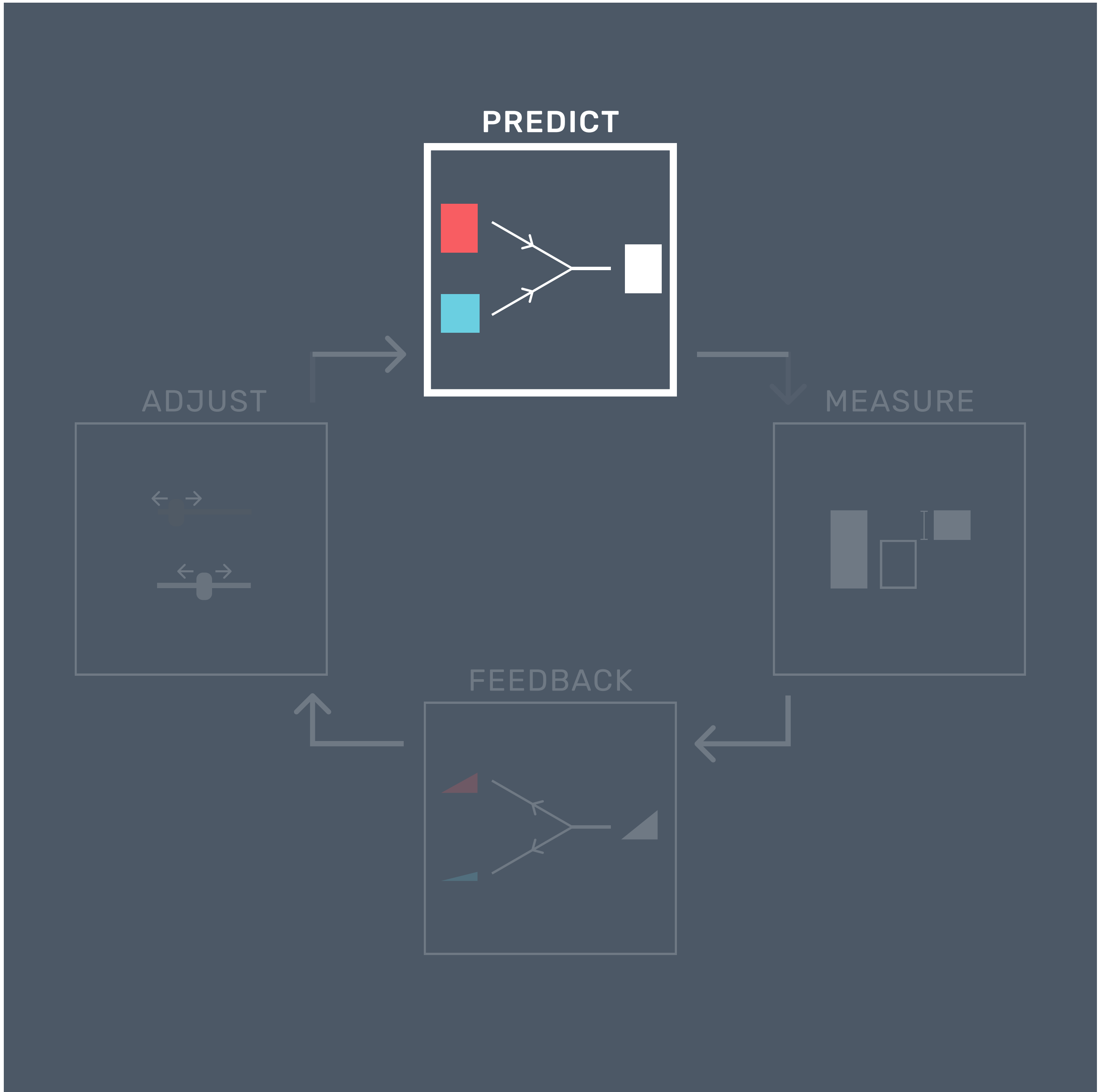
THE FOUR STEPS OF TRAINING

We can think of training a neural network as a four-step cycle: *Predict - Measure - Feedback - Adjust*.

One cycle represents one training round. This is repeated for a number of rounds until the neural network becomes good at the task it's training for.

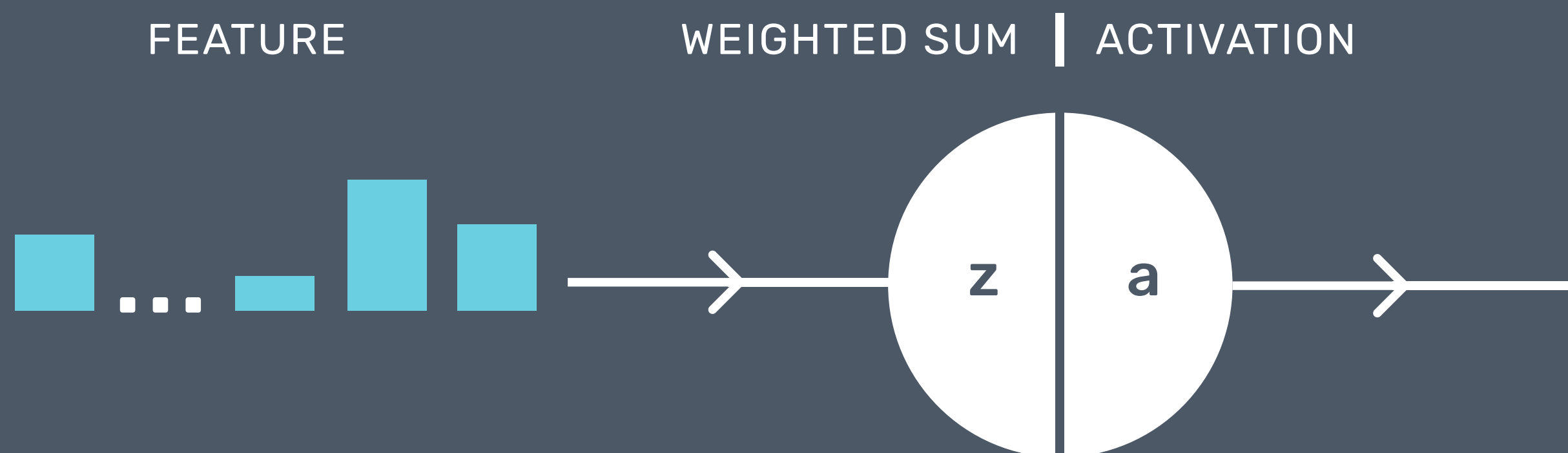
None of this will make sense to you yet, but that's exactly what we'll learn about next!

Also note that these four terms were chosen for this book to make it easy for someone new to deep learning. In other resources, you will find other terms used (e.g. *forward* instead of *predict*, *backward* instead of *feedback*, *update* instead of *adjust*). They refer to the same concepts.



PREDICT

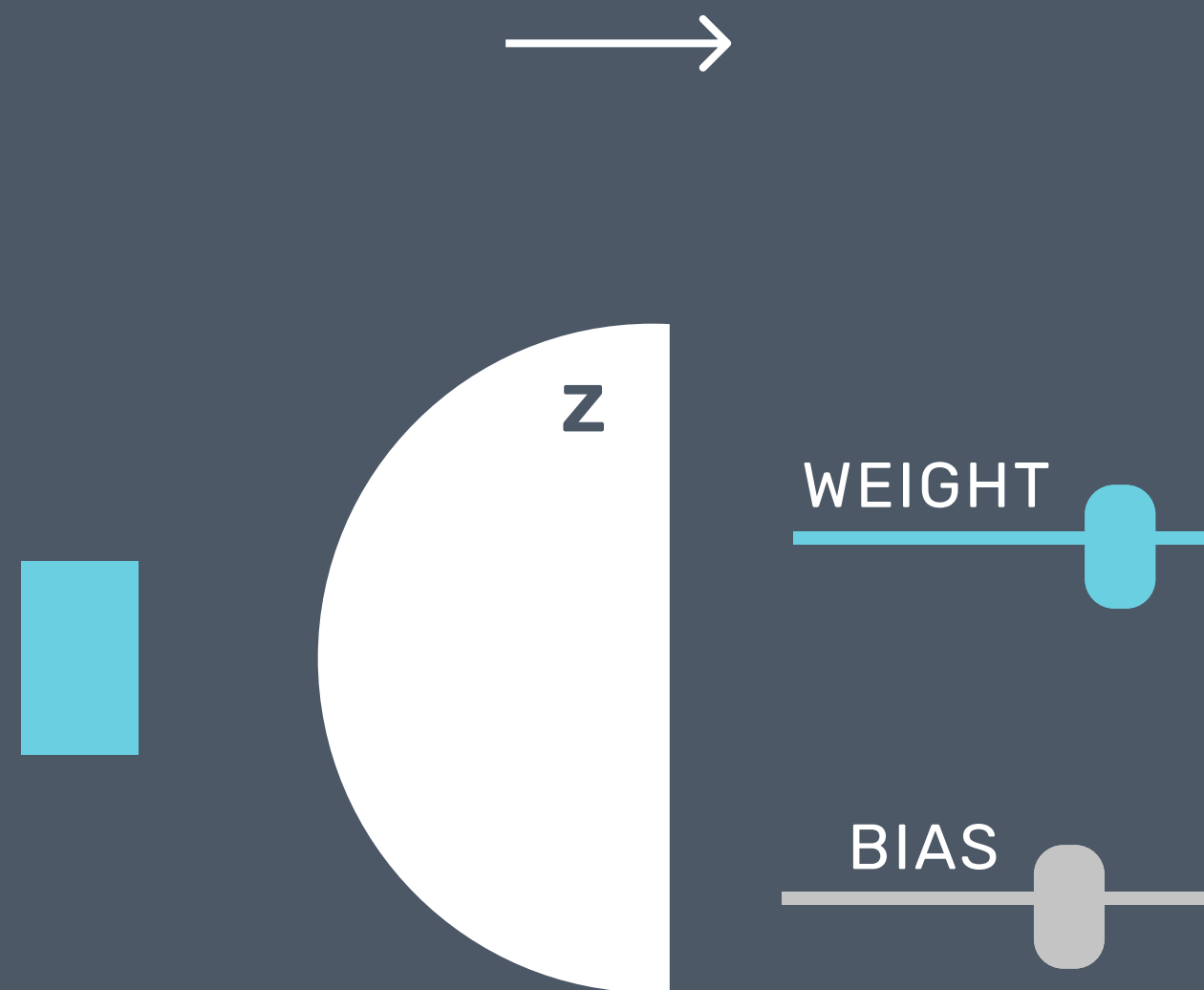
In the first step of the cycle, *predict*, we'll pass the training data through the neuron.



NEURON COMPUTATIONS

Recall that this means going through two steps of computations - weighted sum and activation, one data point at a time.

WEIGHTED SUM



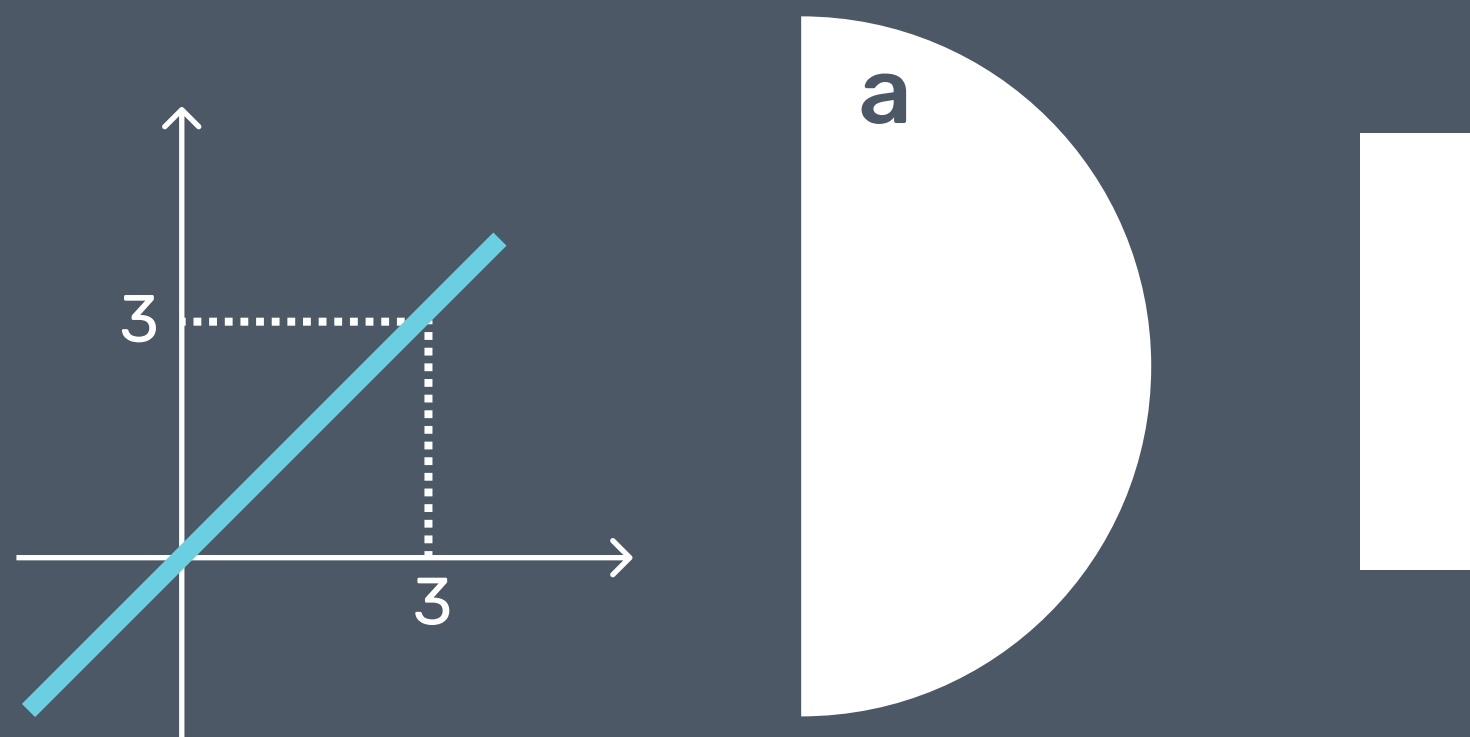
PARAMETER COUNT

We've already seen that the number of weights of a neuron is equal to the number of inputs. The inputs are the dataset's features. And since we have only one feature, there is going to be only one input, and hence, one weight.

We also saw that on top of that, a neuron has one bias value.

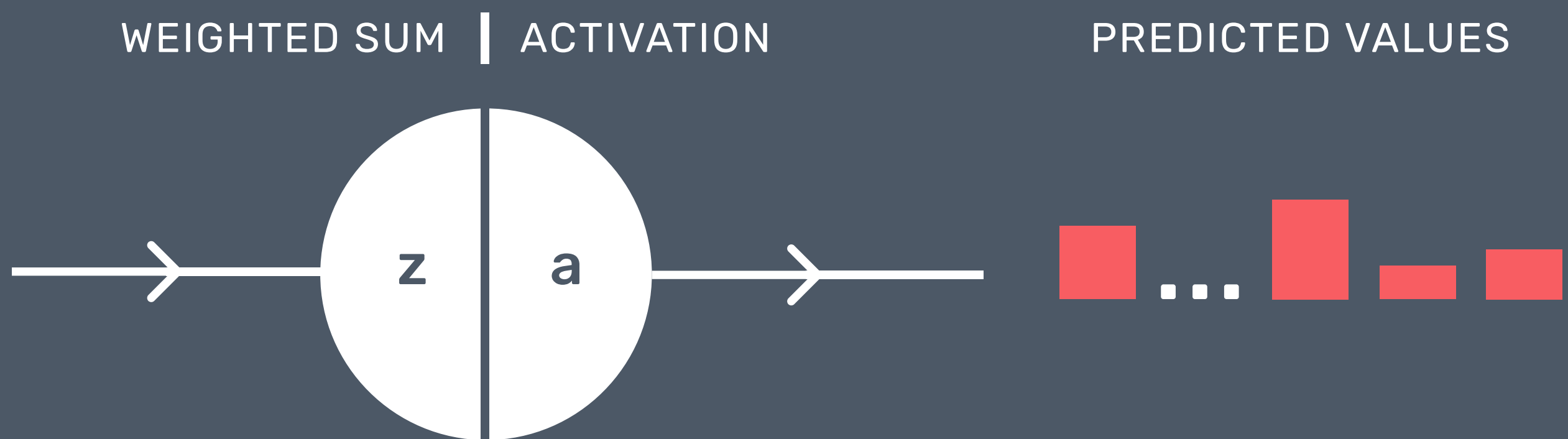
We'll assign initial values for these parameters, in which there are a number of initialization techniques we can choose from. These techniques help the neural network learn more effectively compared to simply assigning random initial values. However, this book doesn't cover this topic as it is quite mathematically involved.

ACTIVATION



ACTIVATION

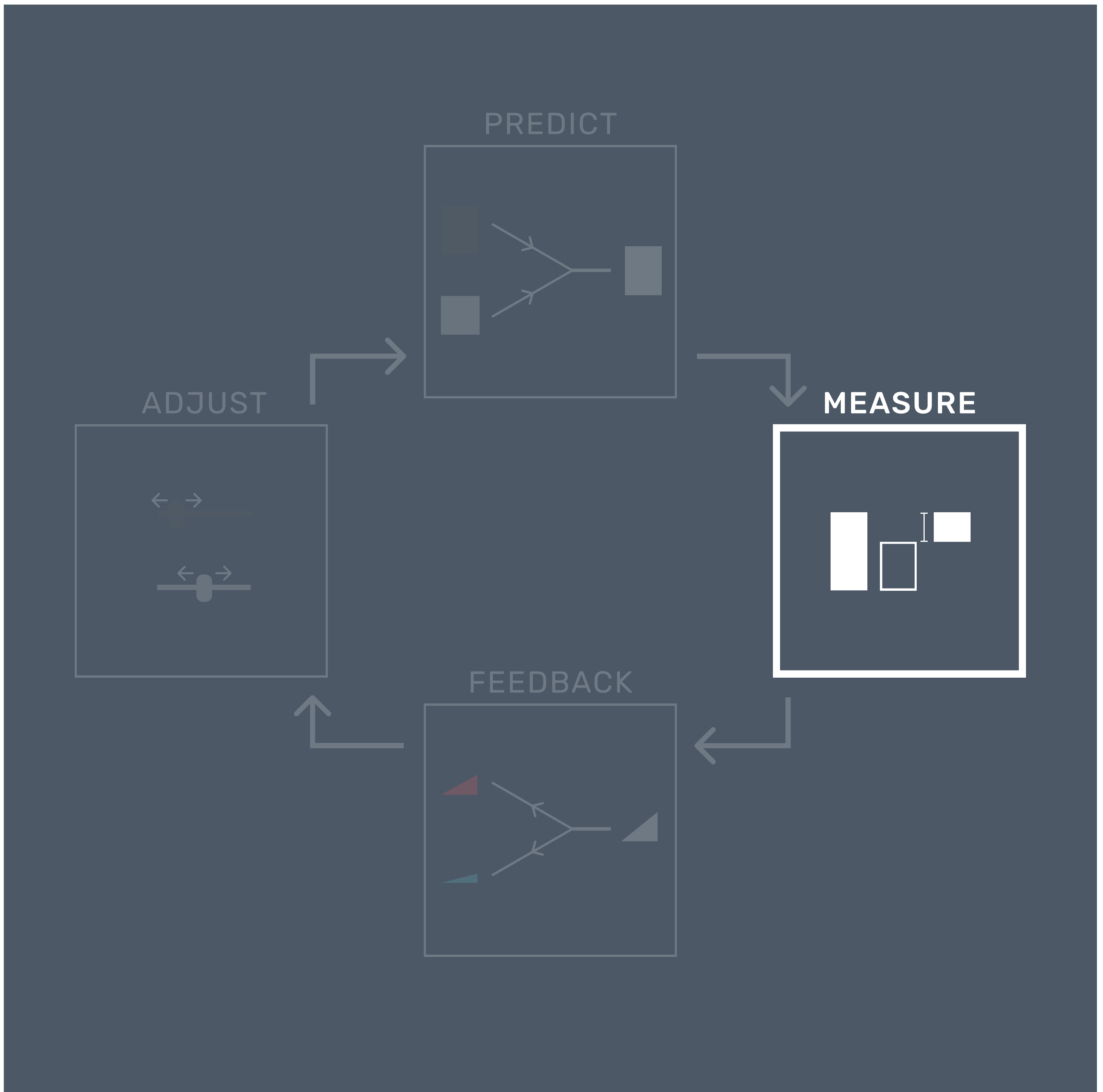
For this task, we'll stick to the linear activation function.



OUTPUT

By now, we will have the neuron successfully outputting eight values. They represent the prices that the neuron predicted.

The problem, however, is that the neuron hasn't learned anything yet. As a result, its predictions will be completely wide of the mark.



MEASURE

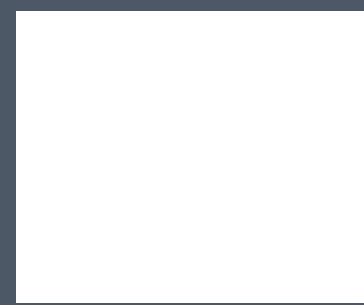
But how do we actually know if the neuron's prediction is good or bad?

This is when we move to the second step, *measure*, where we'll quantify its performance.

PREDICTED
VALUE

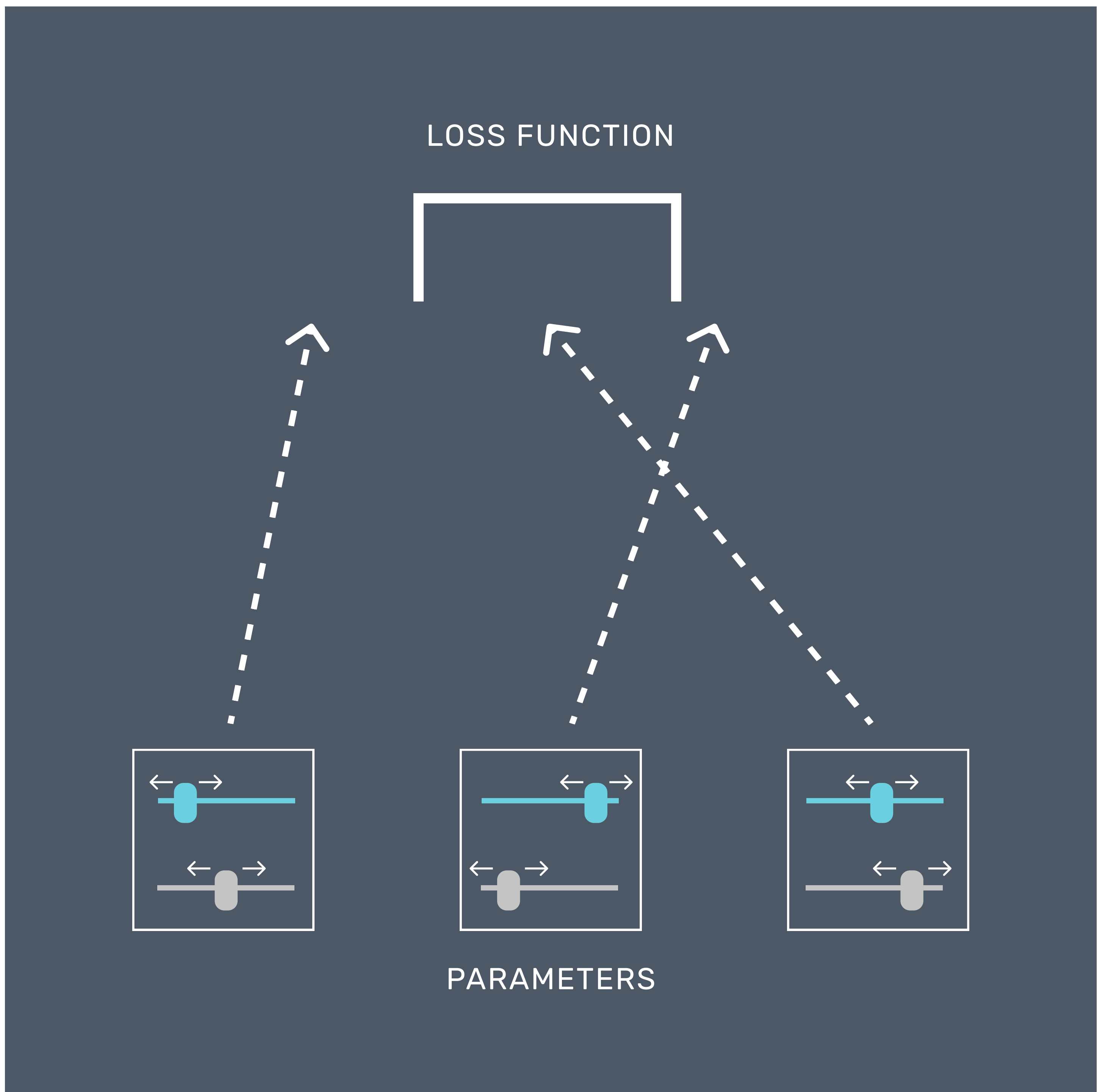
ACTUAL
VALUE

ERROR



ERROR VALUE

Since we know the actual value of the target, we can quantify the performance by computing the difference between the predicted and actual prices. This is called the *error value*.

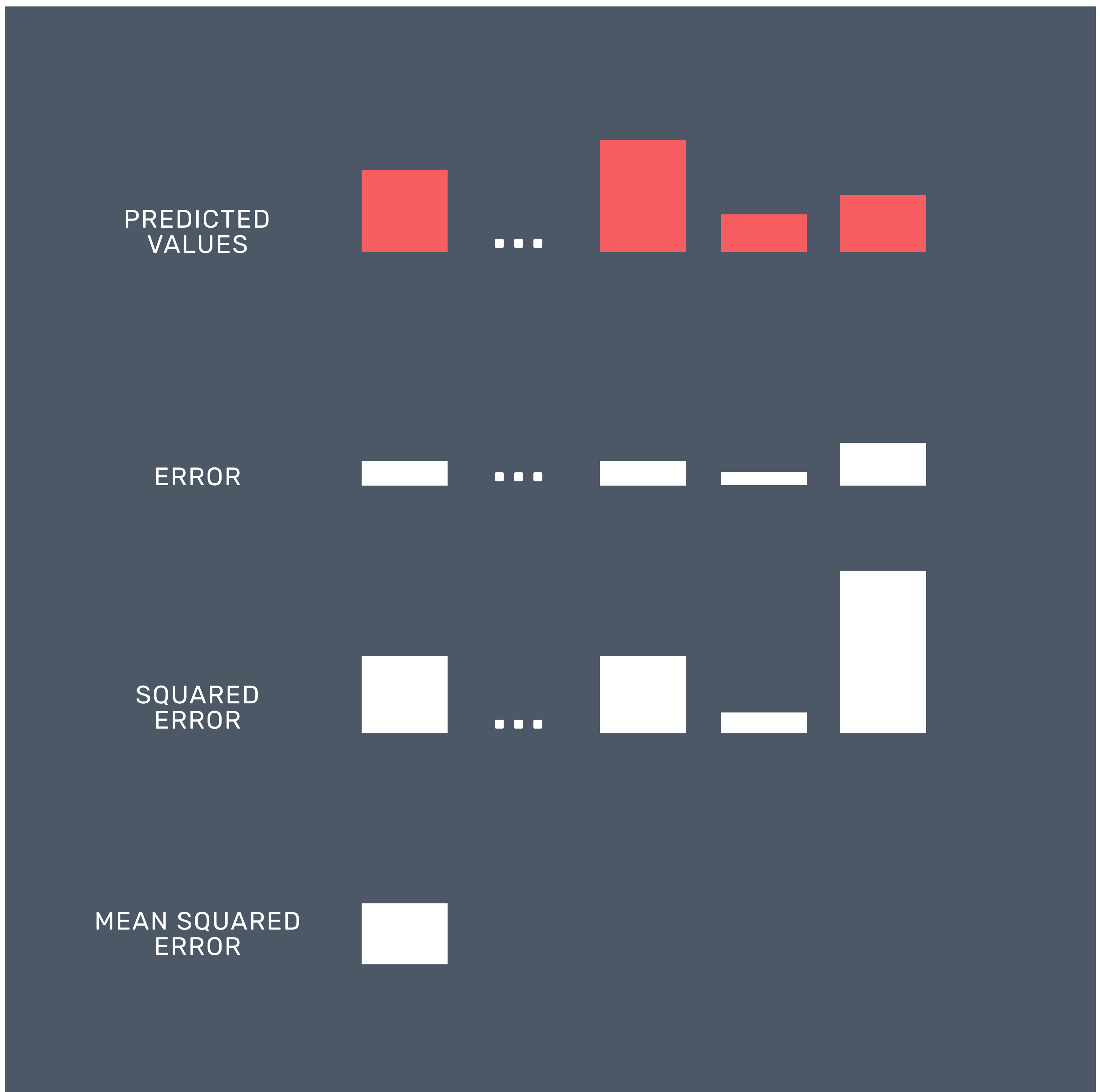


LOSS FUNCTION

This brings us to one of the most crucial parts of designing a neural network—choosing its *loss function*.

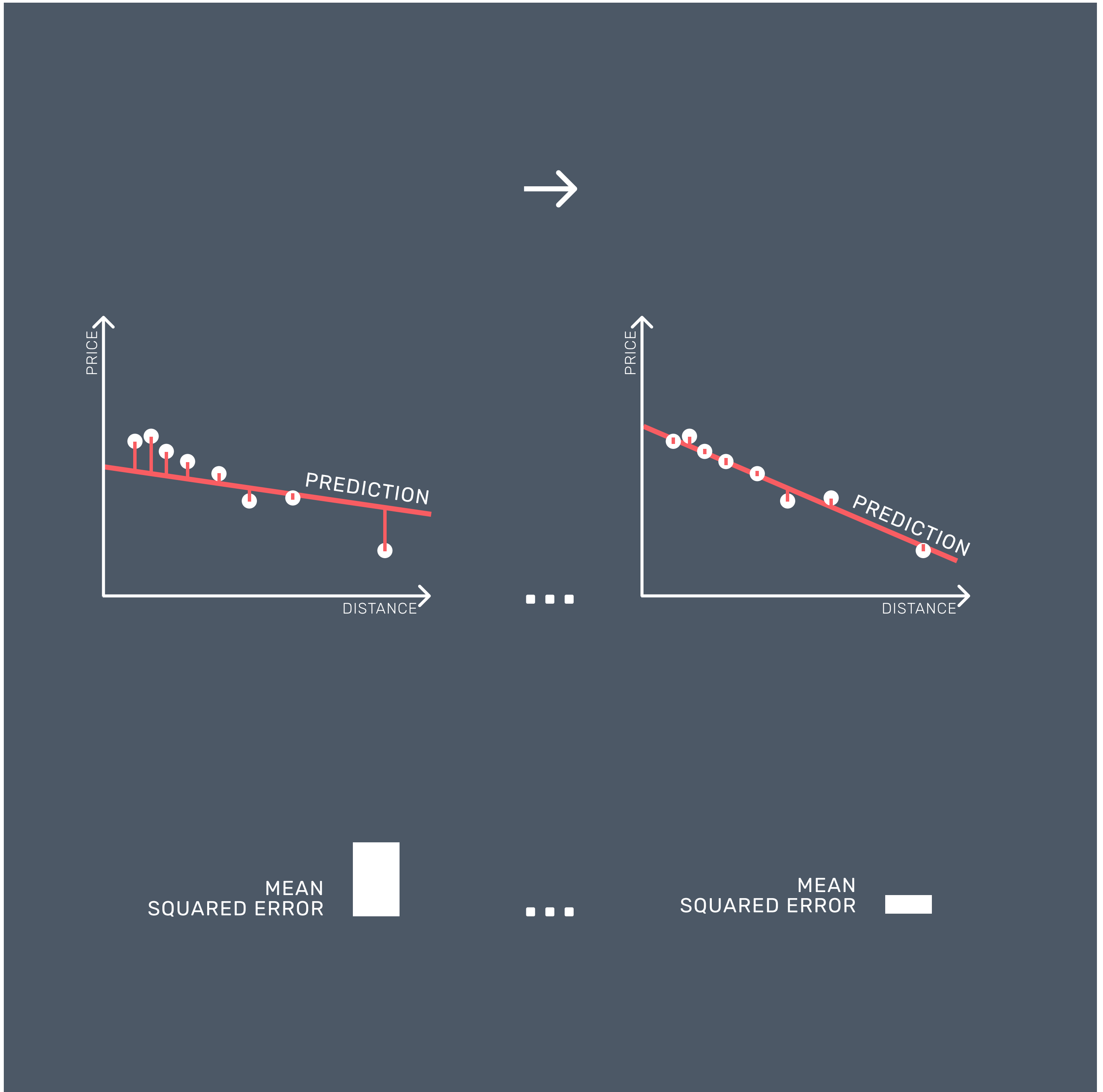
While the parameters are the dials that the network adjusts to reach its goal, the loss function is the goal itself.

The loss function comes in various forms and it all depends on the nature of the task. This will become clearer in Chapters 3 and 4, where we'll use other kinds of loss functions.



MEAN SQUARED ERROR

The loss function we'll use for this task is called the mean squared error, or MSE for short. Each of the eight error values is squared to get the squared error. Then they are averaged to get the MSE.

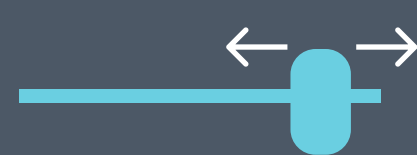
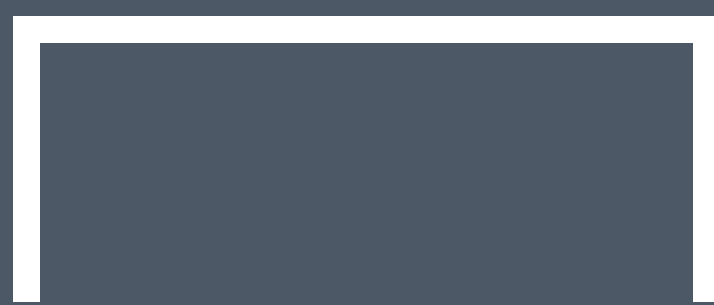


MINIMIZE LOSS

The MSE is a measure of error. That means the smaller the MSE, the better the network is doing.

In other words, the neuron's goal is to minimize its loss over many training rounds.

LOSS FUNCTION

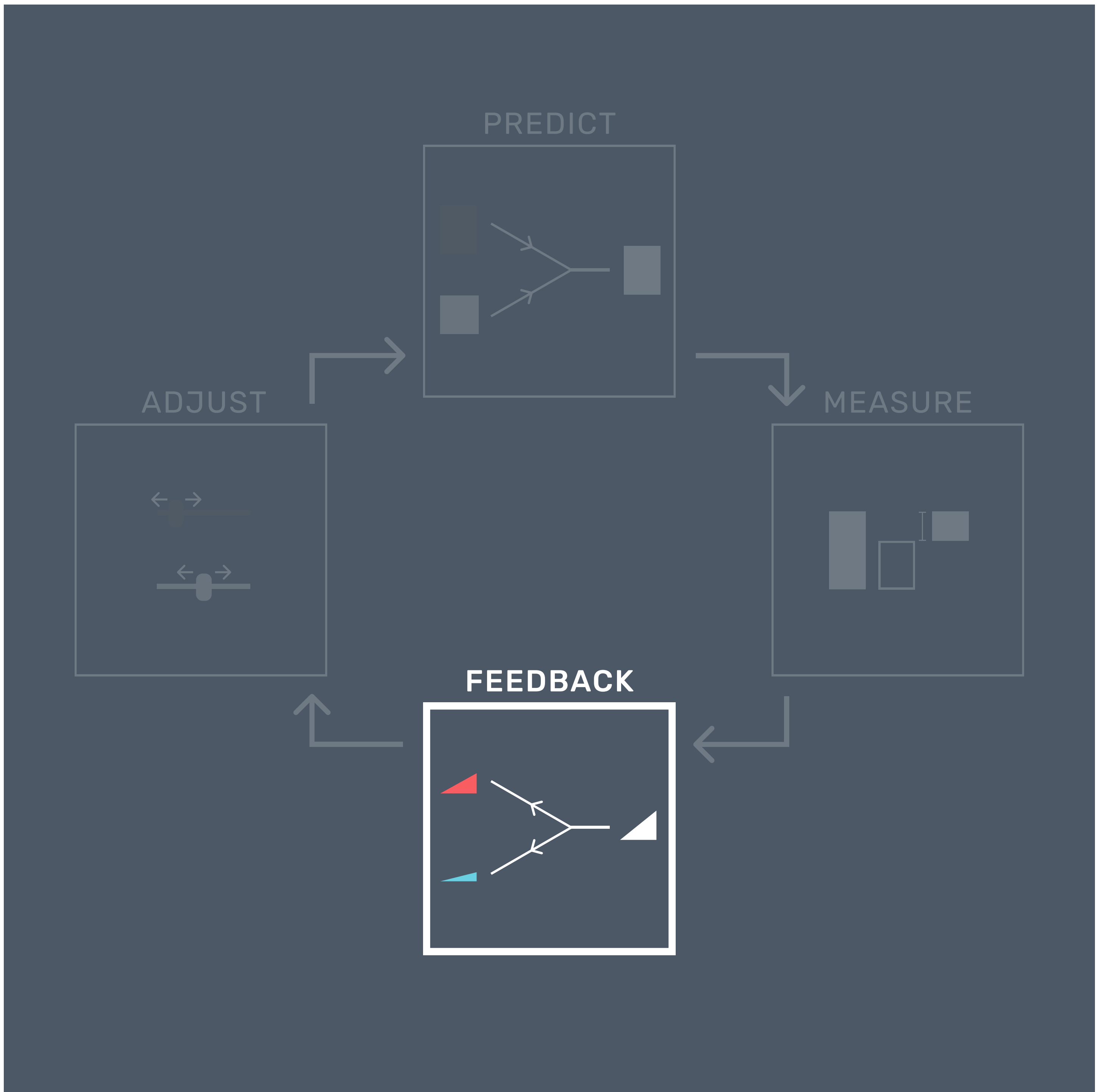


WEIGHT

WEIGHT VS. LOSS

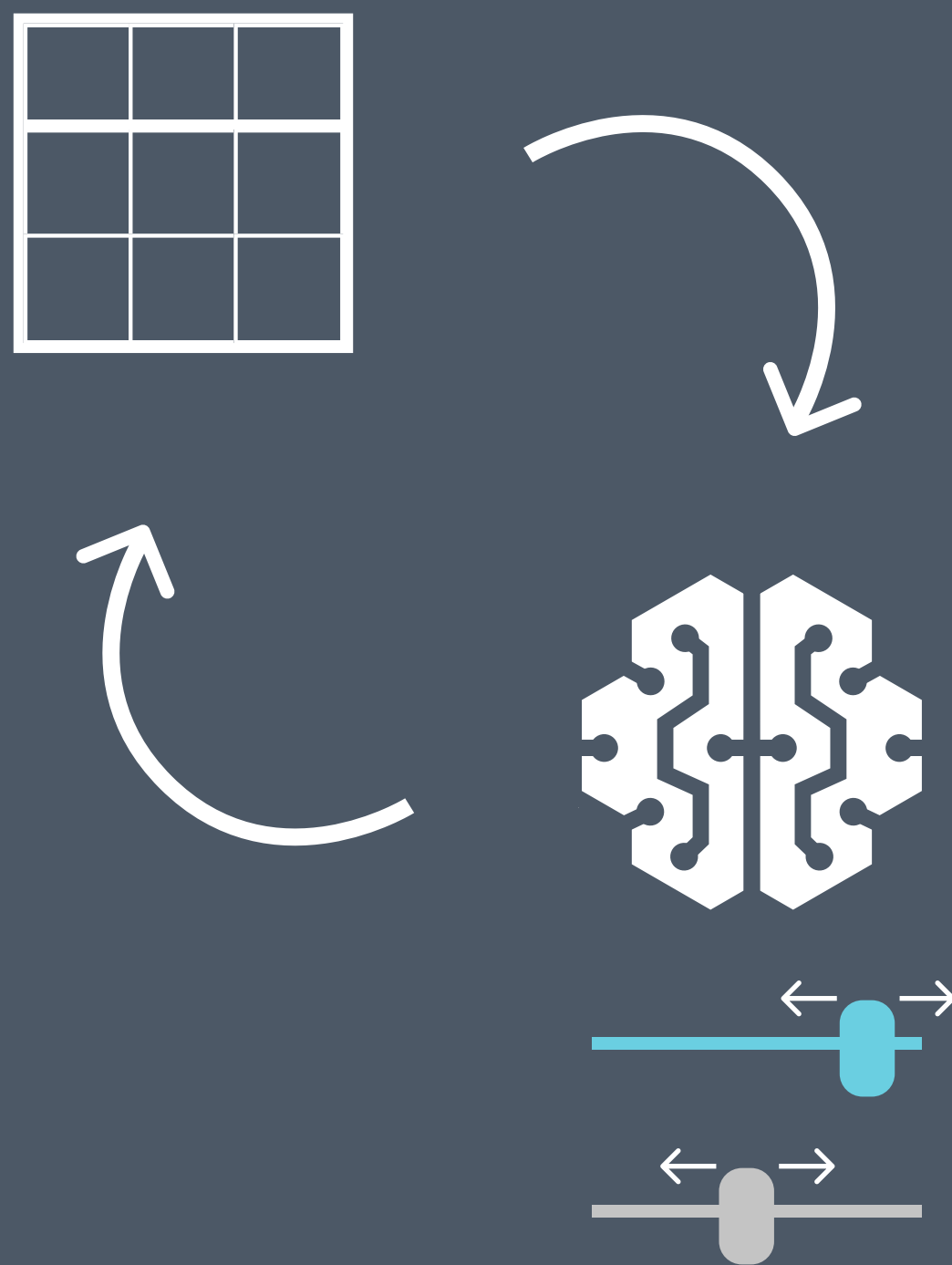
Recall that a neural network learns by adjusting its parameters - weights and biases. Let's first focus on weights since this is where most of the learning takes place. We'll come back to biases later.

We want to find out how changing the weights affects the loss.



FEEDBACK

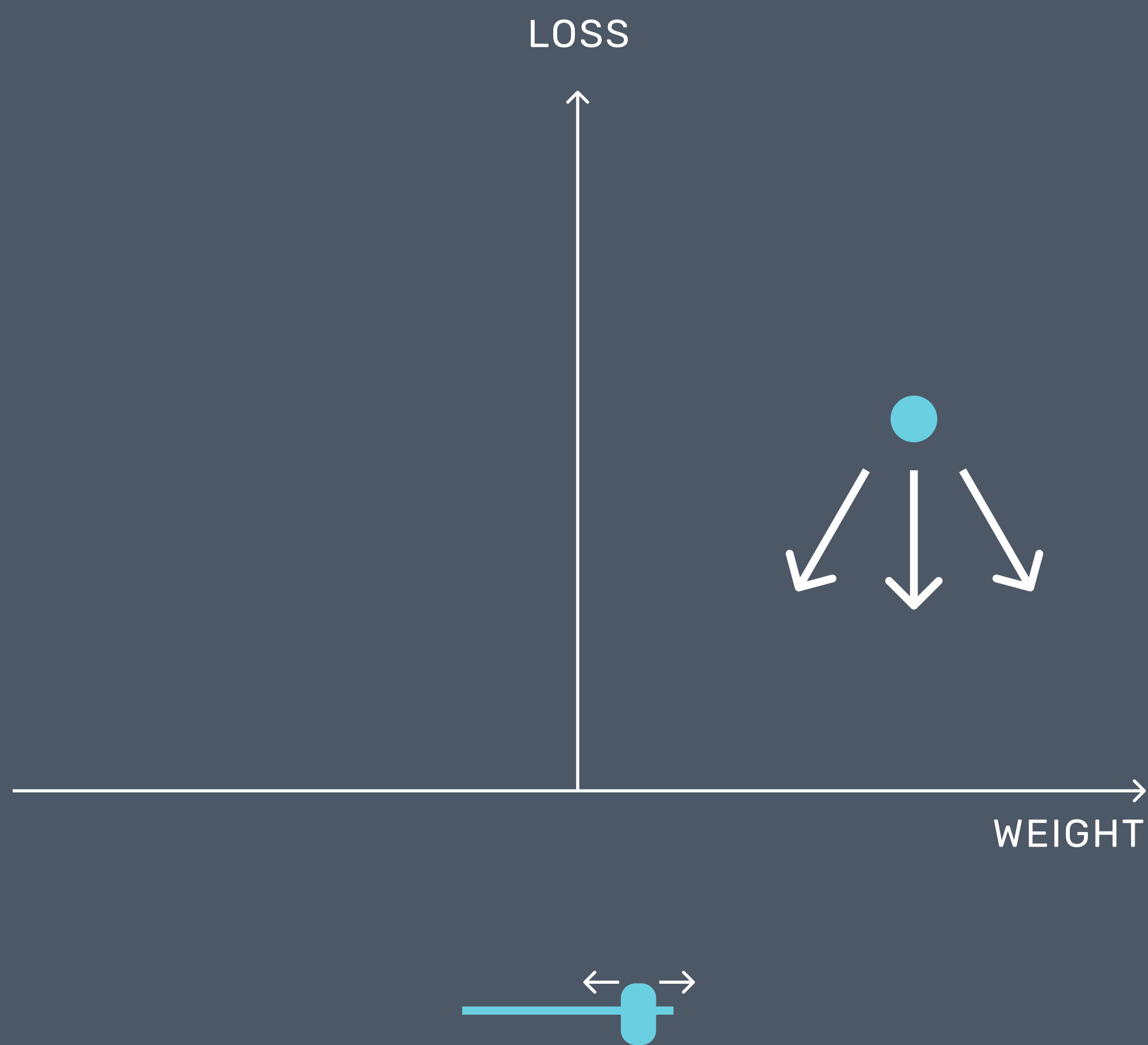
At this point, the neuron hasn't learned anything yet. And learning is exactly what is going to happen in the third step, *feedback*.



LEARNING

We have established that the neuron's goal is to minimize the training loss by adjusting its parameters.

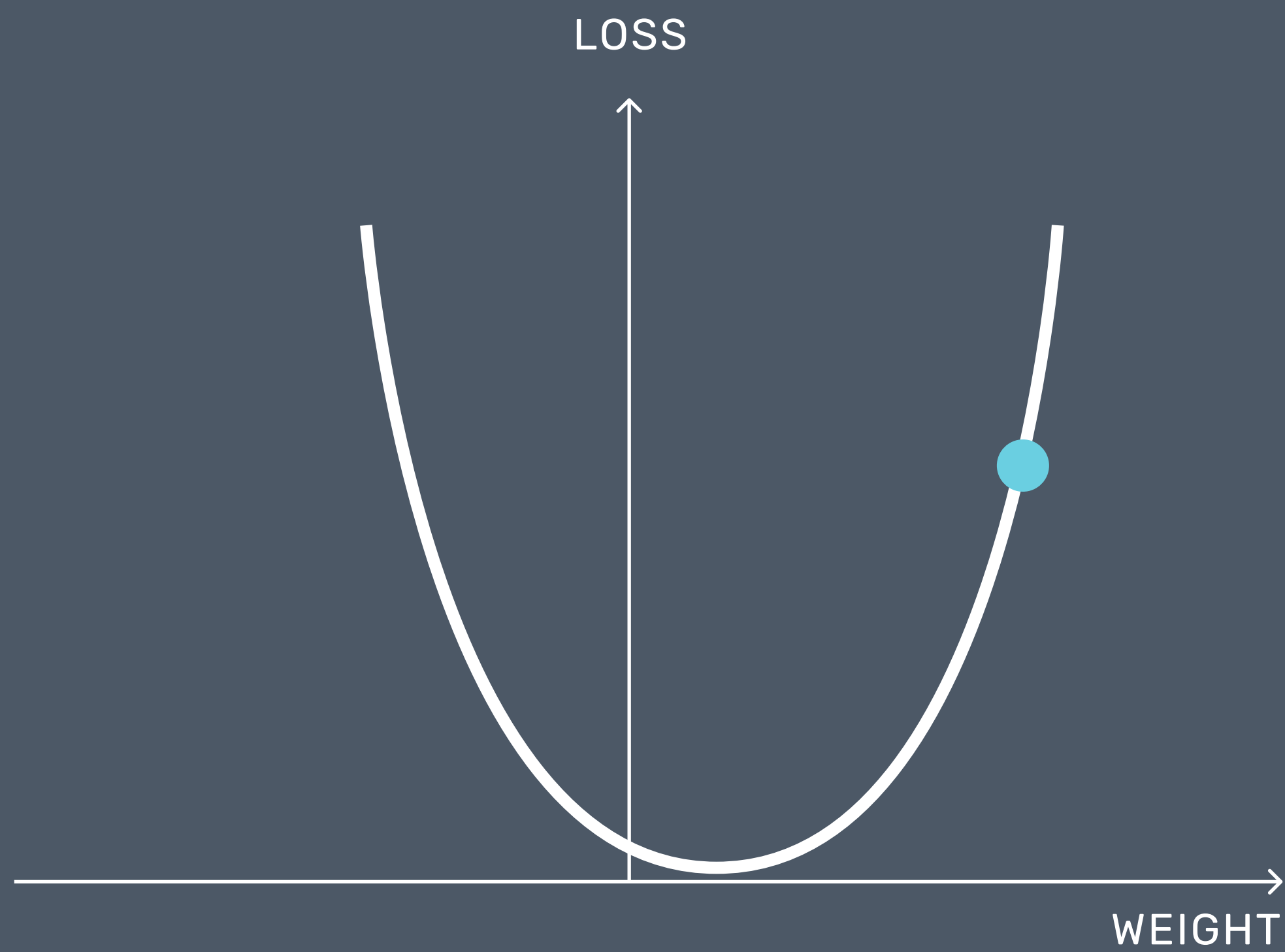
This is the essence of learning in a neural network. Let's find out how this works.



MINIMIZING LOSS

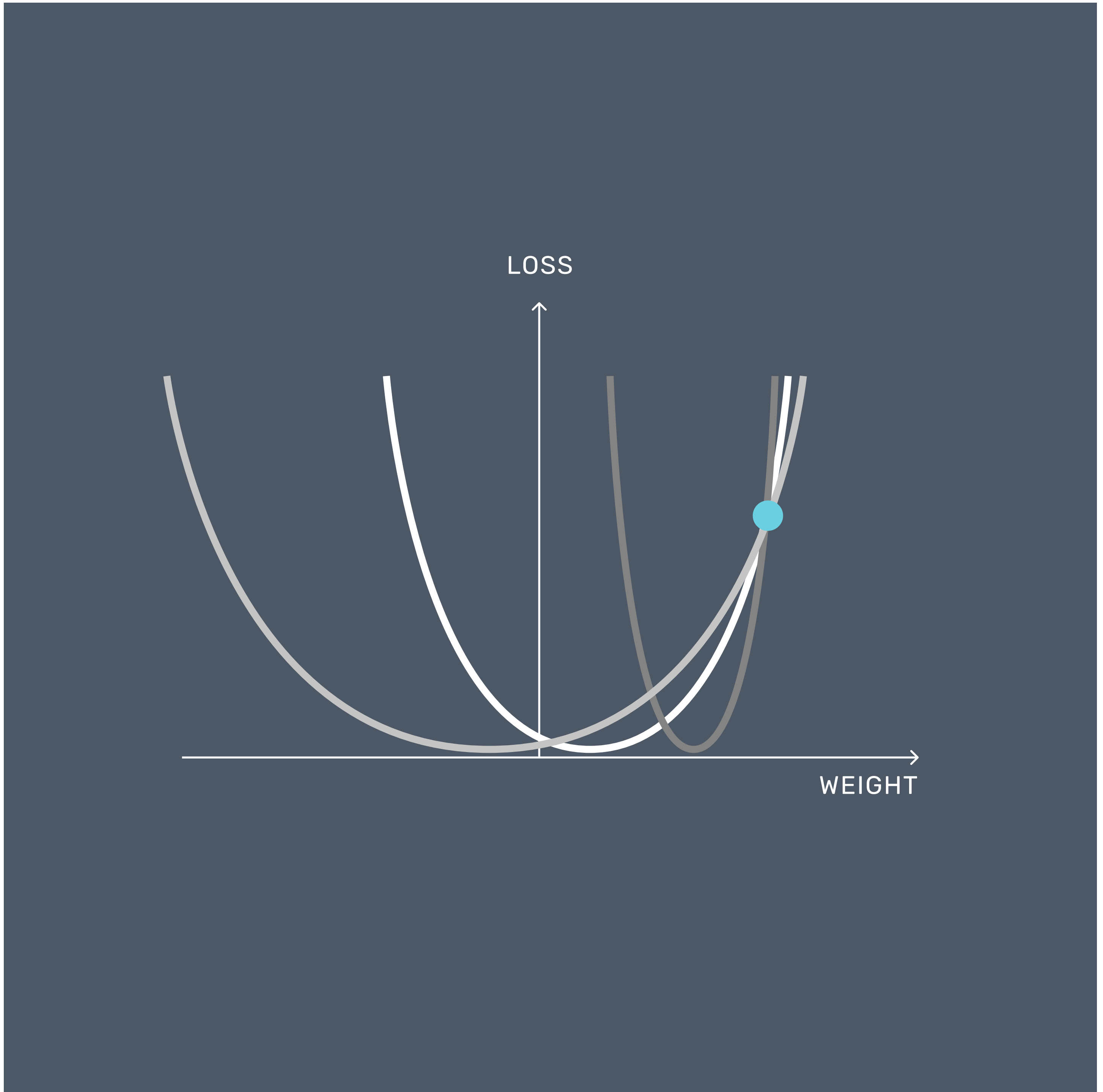
Let's start with one training cycle and plot the loss (i.e. MSE) on a chart.

Now, we want to bring this MSE down to be as close to zero as possible. What we need is to find the weight value that gets us there. But how do we do this?



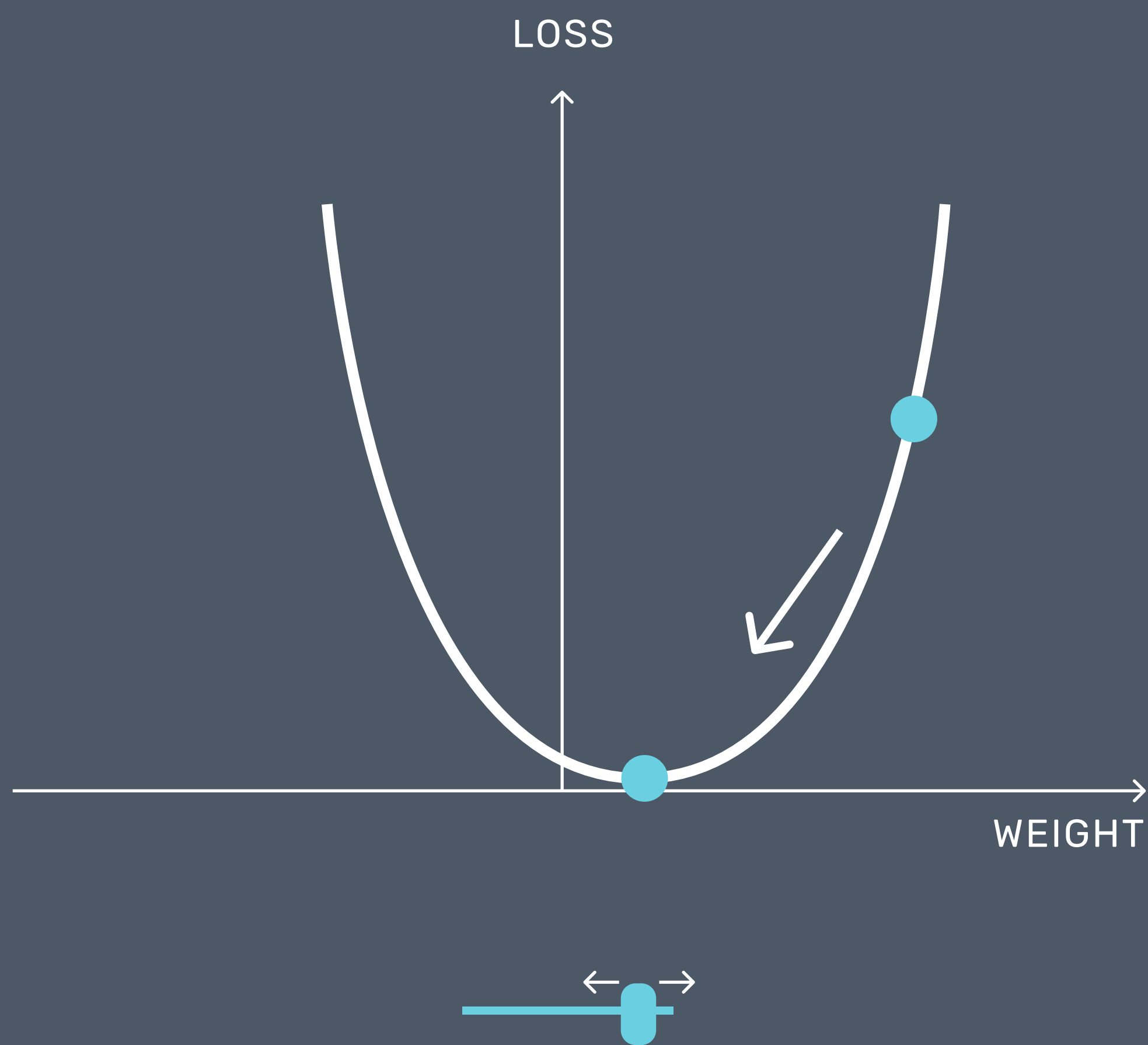
THE LOSS CURVE

As a loss function, the MSE gives us a very desirable property. If we tried plotting all possible weight values and the corresponding MSEs, we'd get a U-shaped curve. This comes from the squaring effect of the MSE.



MINIMUM POINT

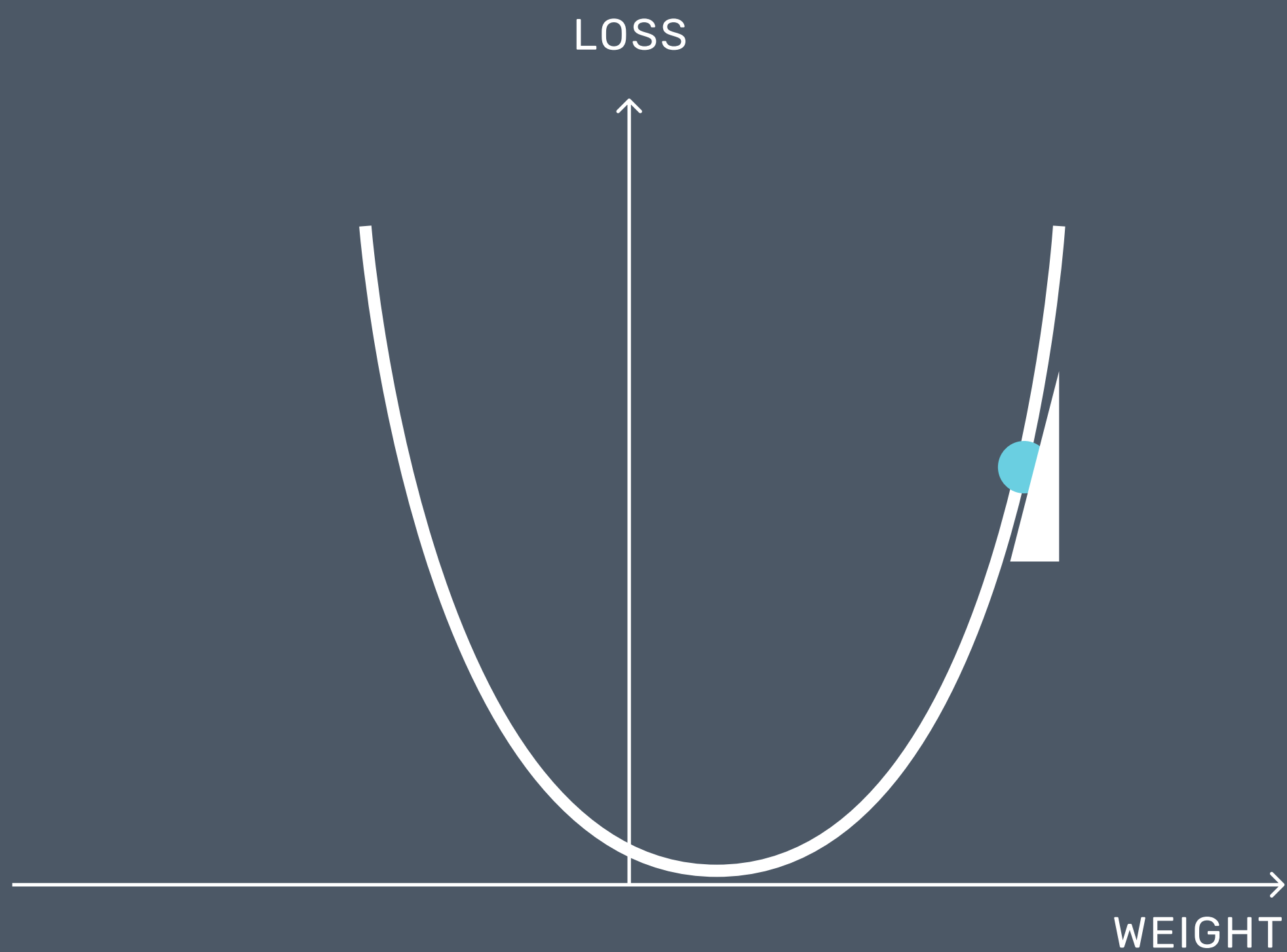
Its width and position may vary, but its shape will always be the same - there is a single point where the curve reaches its minimum. And that is what we are after!



GOAL

And that is our goal - to get the neuron to find the weight value that will bring the MSE to its minimum.

In practice, we won't be able to get exactly to the lowest point. But we can get very close, and that's good enough for most tasks.

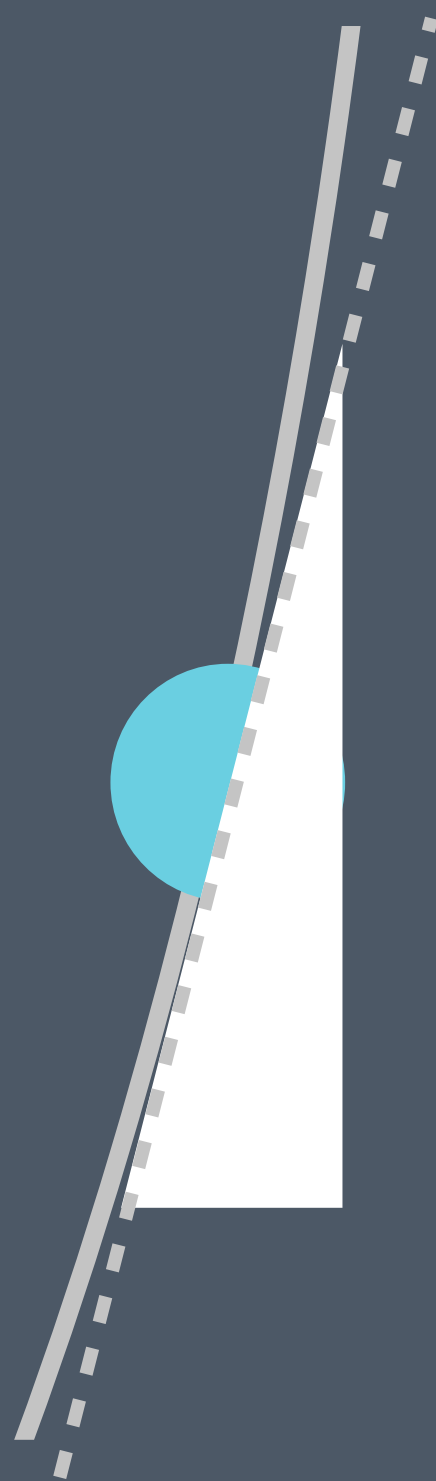


GRADIENT

The next question then is, how does the neuron know by how much to adjust its weight? The answer is to find the *weight gradient*.

A gradient is the derivative of an output with respect to its input. In our case, the output is the loss function, while the input is the weight. In Chapter 2, we'll find gradients of the loss function with respect to other types of inputs, so it's worth keeping this in mind.

We won't go deep into the math, but let's understand why this is useful for the neural network.



STEEPNESS

The gradient is a measure of the steepness of the curve of the loss function. And where we are now, it's very steep. The steeper the curve, the greater the gradient.

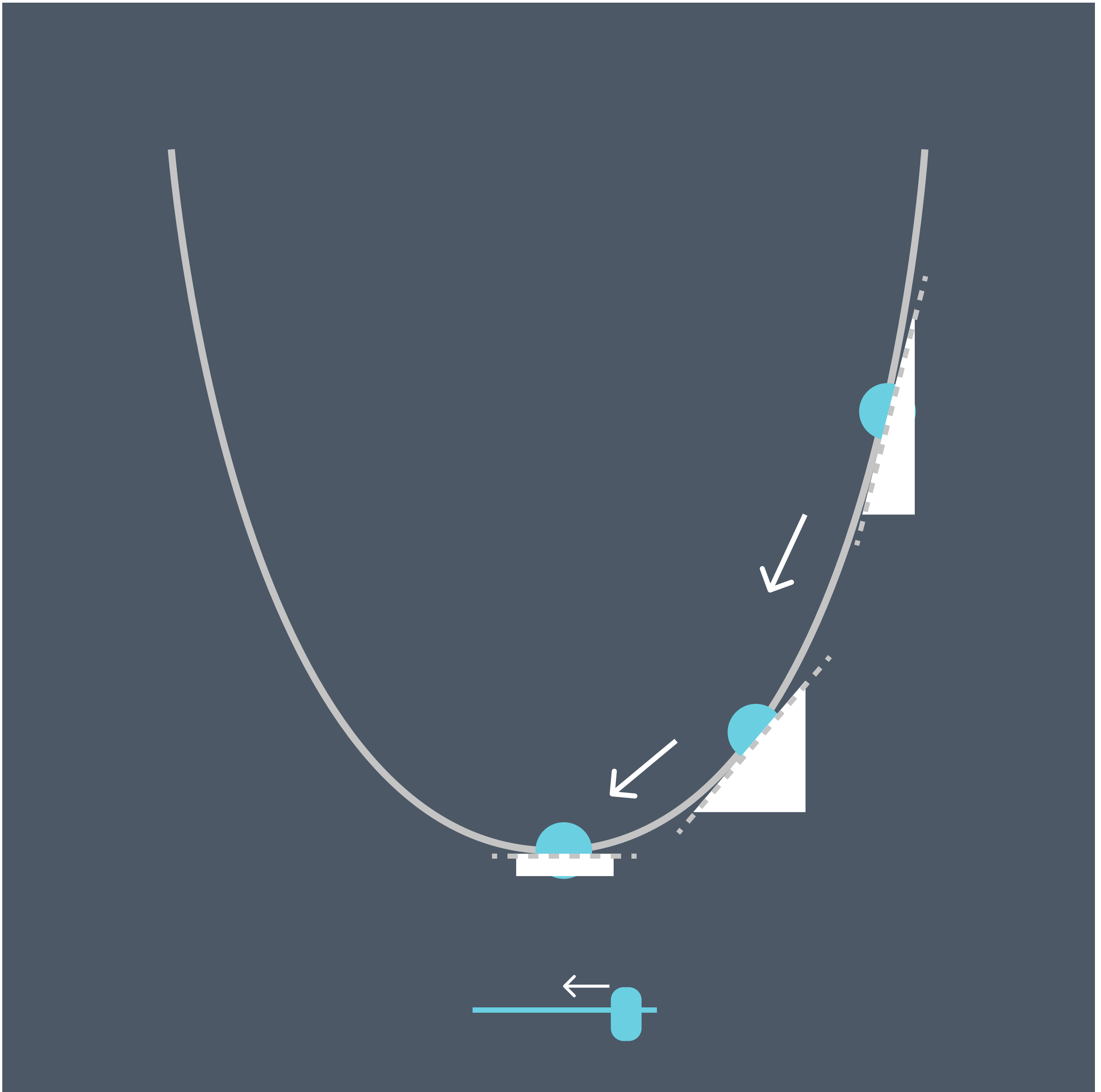
A large gradient indicates that the weight is still far from the optimal value, so we'll need to adjust it by some amount.



MINIMUM GRADIENT

But why is this so? To better understand, let's pretend that we've succeeded in finding the ideal weight value that brings the loss to the bottom of the curve.

The gradient here is zero, which means that we no longer need to adjust the weight.

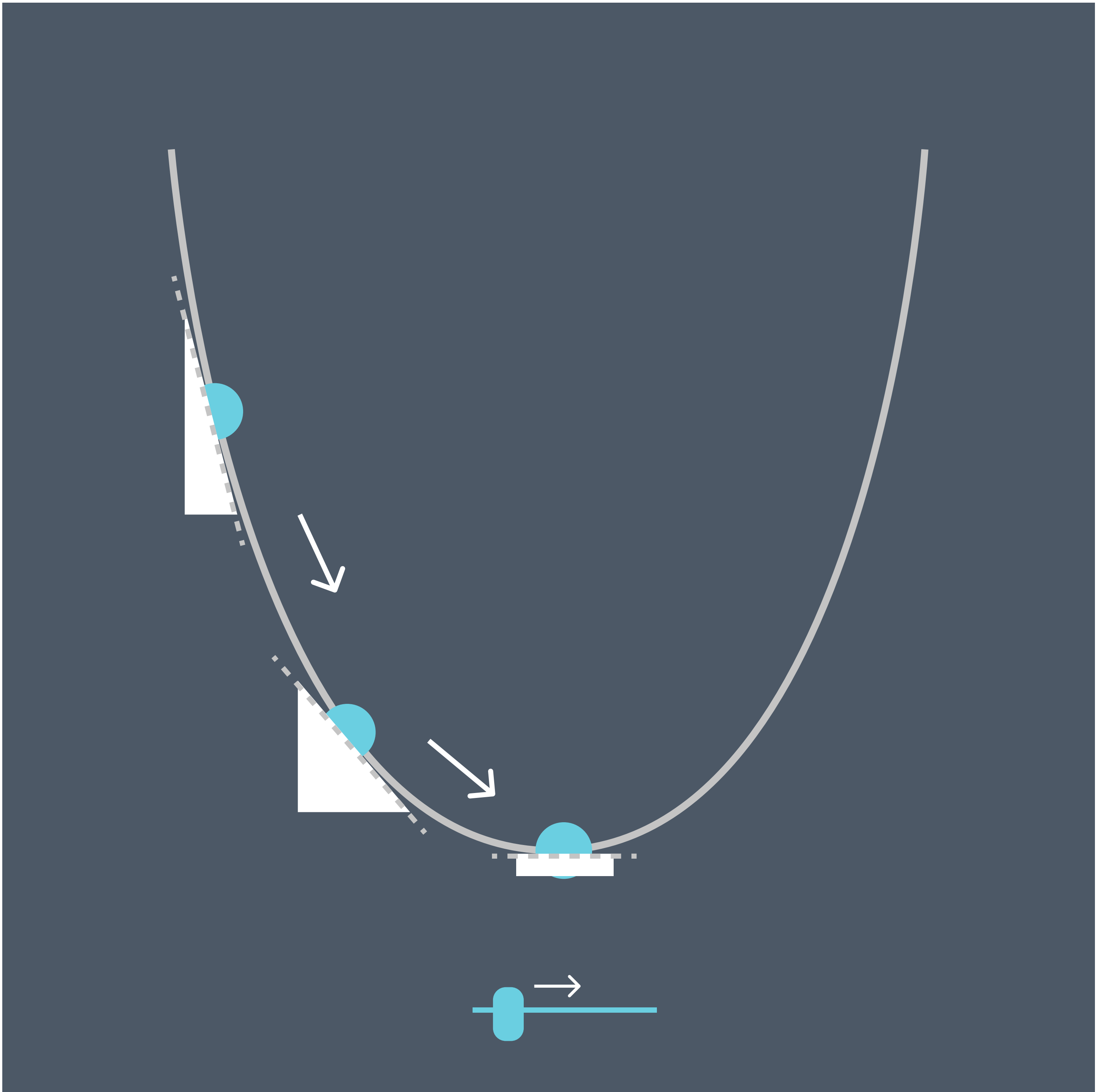


MAGNITUDE

Notice that as we decrease the weight from its initial position to the bottom of the curve, the gradient continues to decrease until it reaches zero.

This is the first property of the weight gradient - its *magnitude*.

The magnitude of the gradient informs the neuron how far its prediction is from the actual. And by the same token, it also informs how much the neuron needs to adjust its weights.



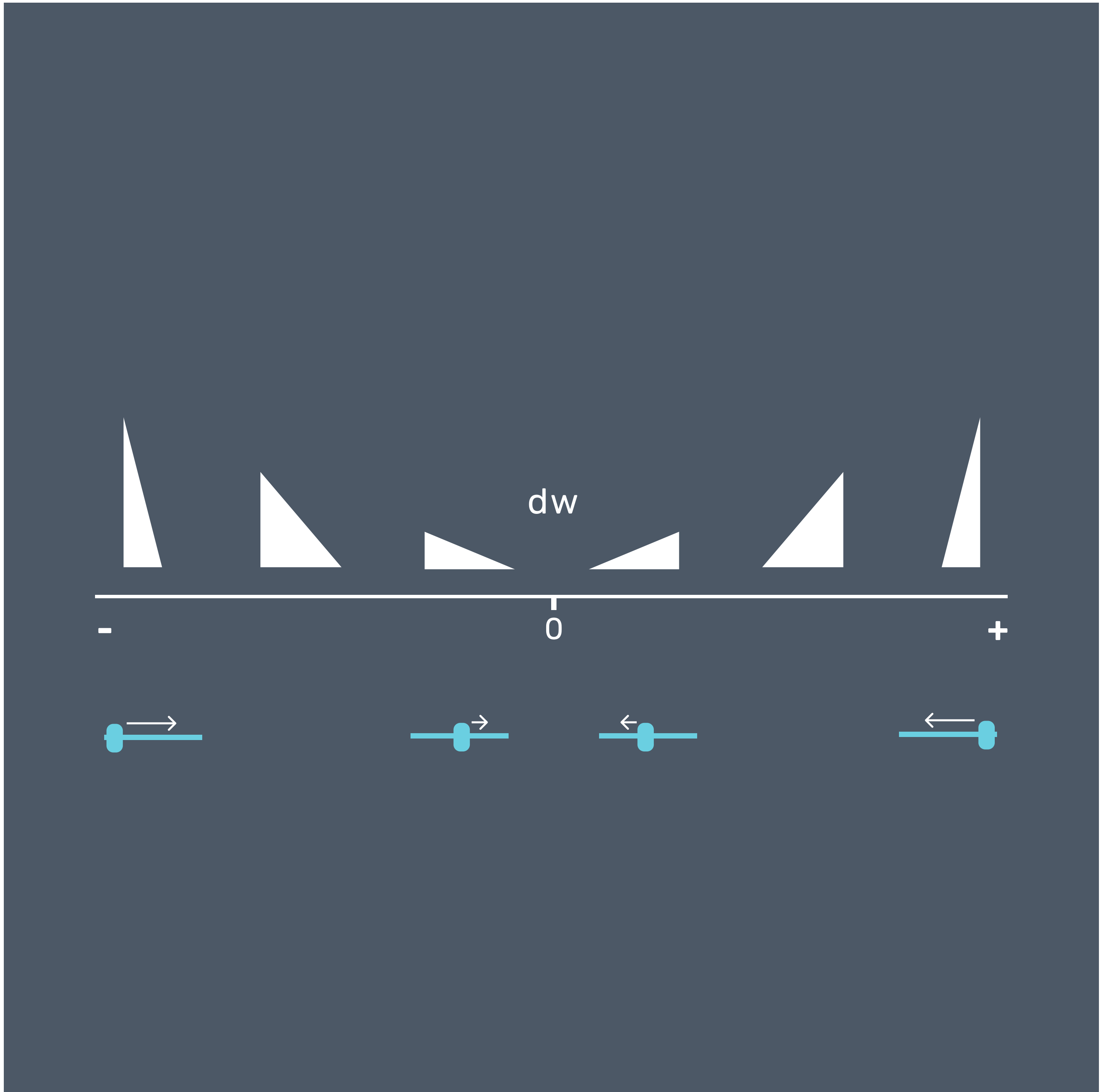
DIRECTION

The second property of the weight gradient is its *direction*.

Suppose the starting weight is on the other side of the curve. This causes the sign of the gradient to become negative.

This indicates that the gradient is too small rather than high. Instead of decreasing, we'll need to increase the weight to reach the bottom of the curve.

Therefore, the sign of the weight gradient informs the neuron about the direction of weight adjustment.



GRADIENT DESCENT

The magnitude and direction of the weight gradient are the two types of feedback returned to the network.

As the network goes through multiple training cycles, we want the weight to move down the curve toward its minimum point.

For this reason, this method is called *gradient descent*.

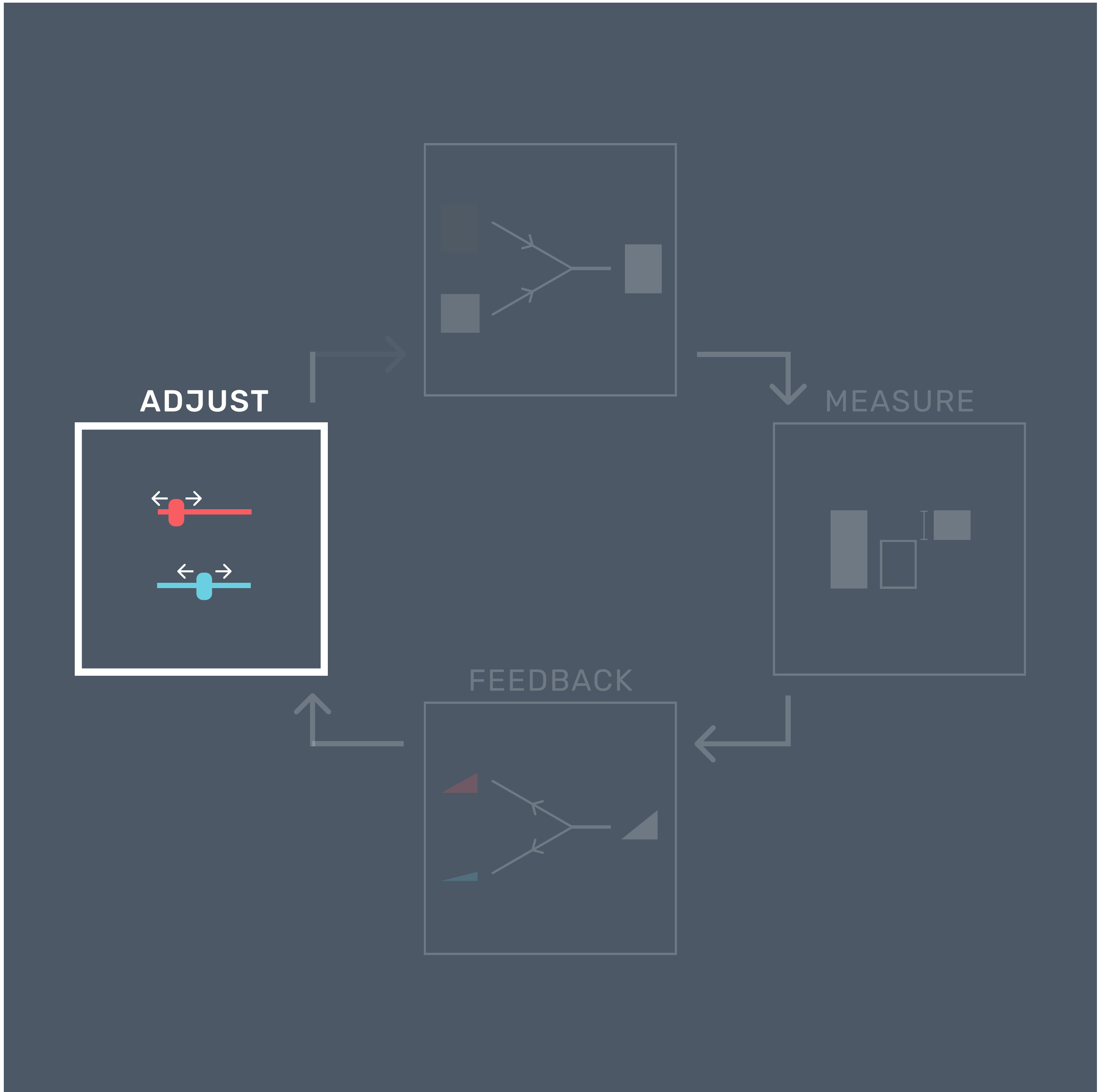
$$dw = \text{input} * \text{error}$$



WEIGHT GRADIENT

We've seen that the weight gradient is the derivative of the loss function with respect to the weight. We won't go into the mathematical proof, but the result is the input value multiplied by the error. We'll represent the weight gradient as dw for short.

The final gradient to be passed back to the network is the average gradient from all the training data points.



ADJUST

We have now reached the fourth and final step of the training cycle, *adjust*. In this step, the neuron will adjust its weight according to the gradient that it receives.

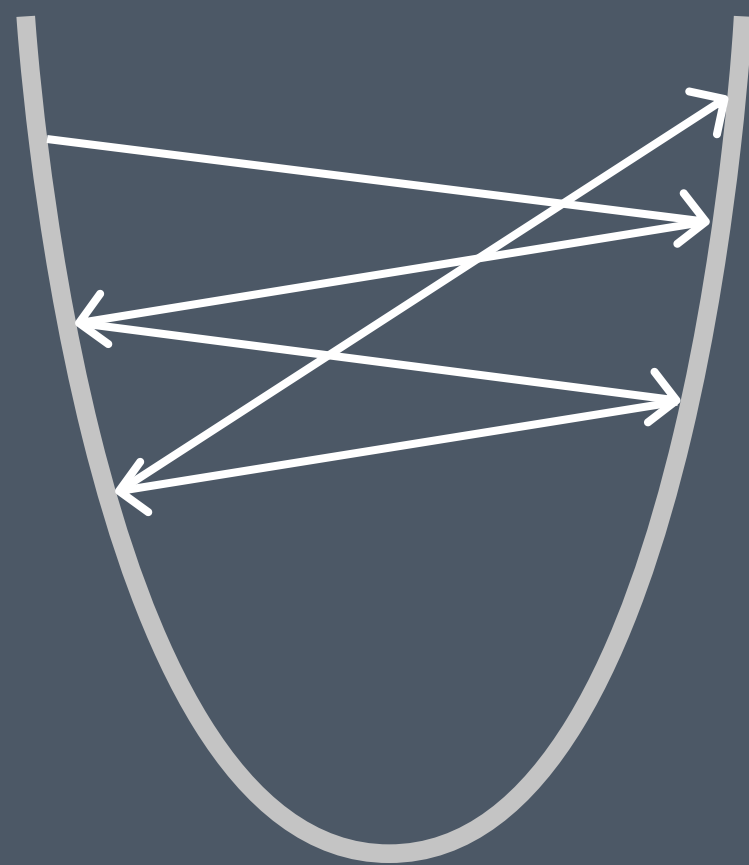
$$W_{\text{new}} = W_{\text{previous}} - \text{alpha} * dw$$



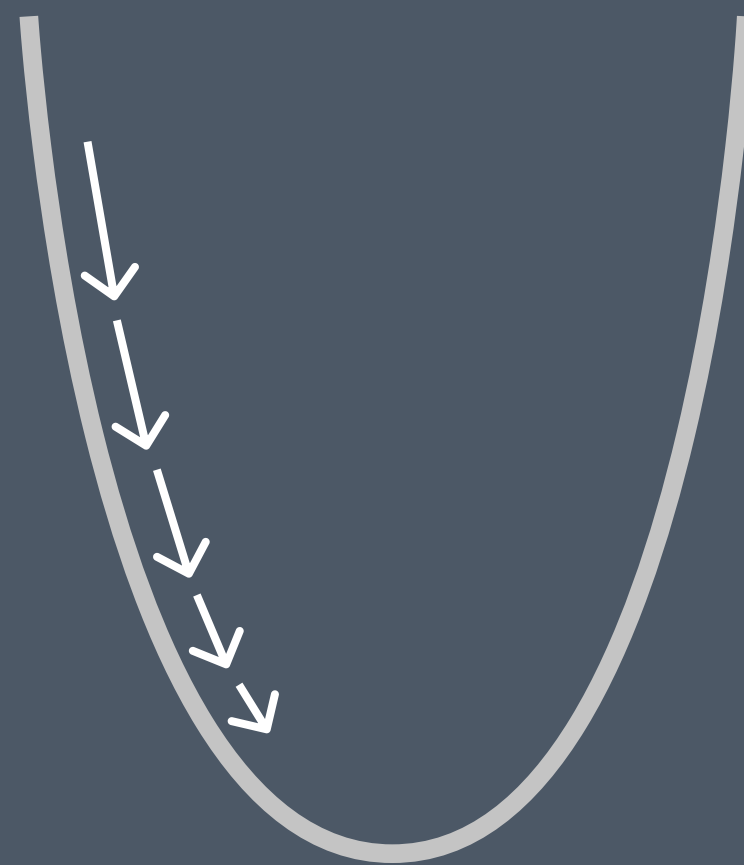
LEARNING RATE

If the gradient is a positive value, the previous weight is reduced commensurate to the magnitude. On the other hand, if the gradient is negative, the previous weight is increased.

Here we introduce another term called the *learning rate*, represented by *alpha* for short. It is a value multiplied by the gradient before making the weight adjustment.



WITHOUT
LEARNING RATE



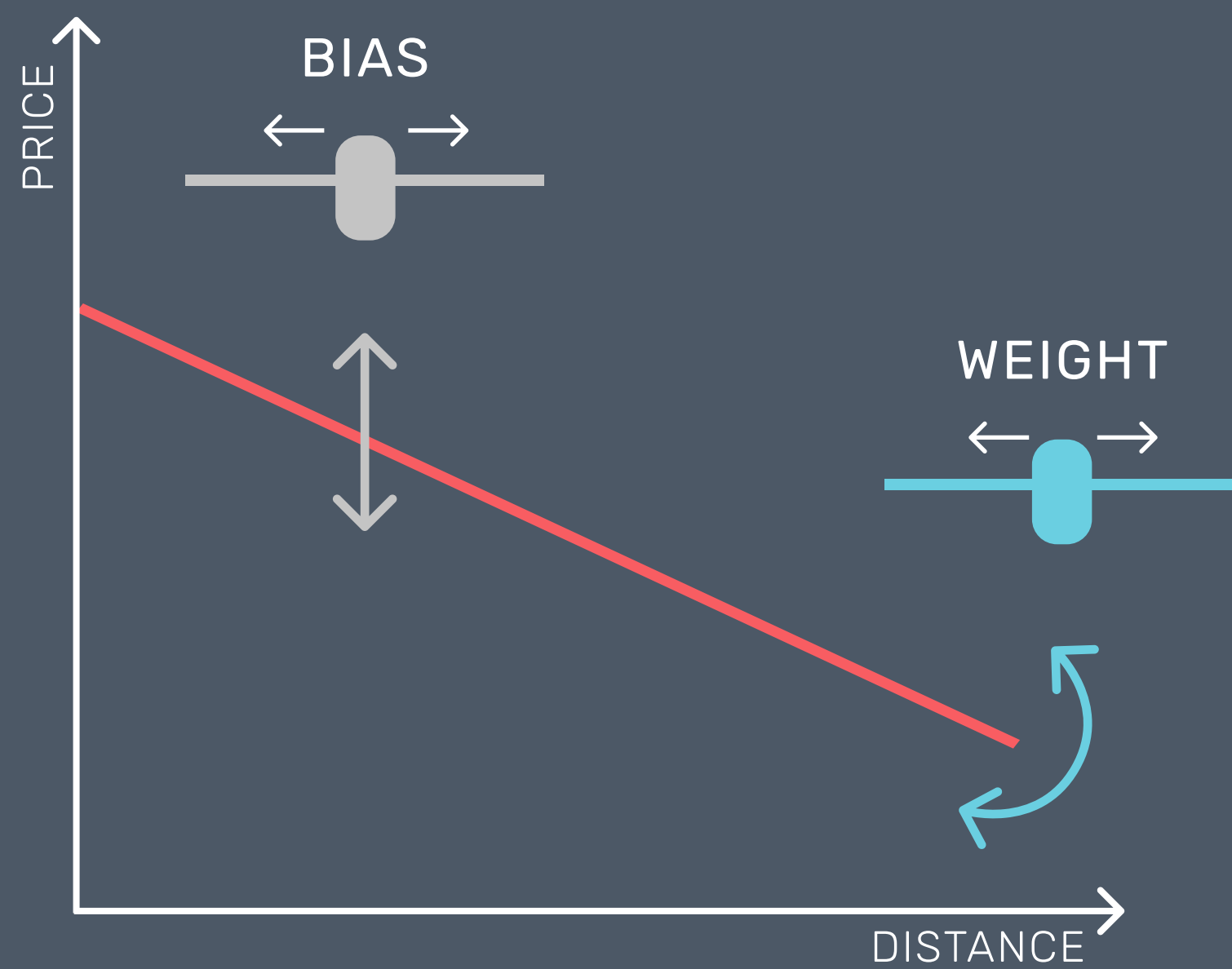
WITH
LEARNING RATE

THE ROLE OF LEARNING RATE

We want the gradient descent process to be a smooth one. And to ensure that, we need to apply the learning rate. It is usually a very small number that scales the gradient down. In our task, we'll use 0.08.

Without the learning rate, the descent may become unstable, or worse, never reach the bottom of the curve.

Choosing the right values is an art in and of itself. Too small and the neuron learns too slowly. Too high and the neuron never finds the minimum point.



BIAS

Let's now bring the bias parameter into the discussion.

The weight's role is to adjust the shape of the prediction line or curve. Meanwhile, the bias' role is to adjust the position of the function, shifting it up or down.

$$dw = \text{input} * \text{error}$$



$$db = \text{error}$$



BIAS GRADIENT

We get the bias gradient db by taking the derivative of the loss function with respect to the bias. Without going into the mathematical proof, the result is the error value itself.

The final gradient to be passed back to the network is the average gradient from all the training data points.

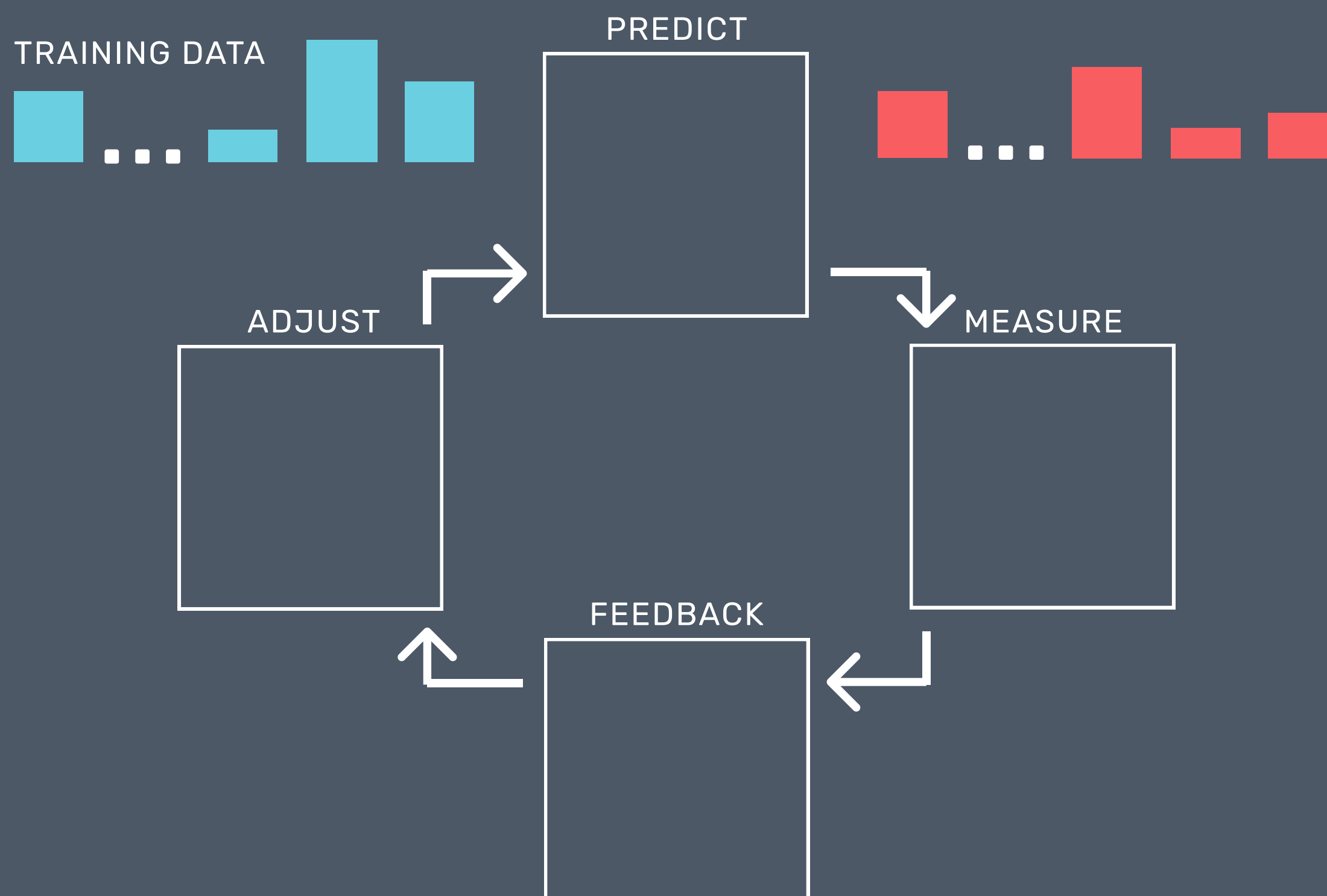
$$b_{\text{new}} = b_{\text{previous}} - \text{alpha} * db$$



ADJUST

As in the weight, the bias gradient is multiplied by the learning rate before making the bias adjustment.

TRAINING



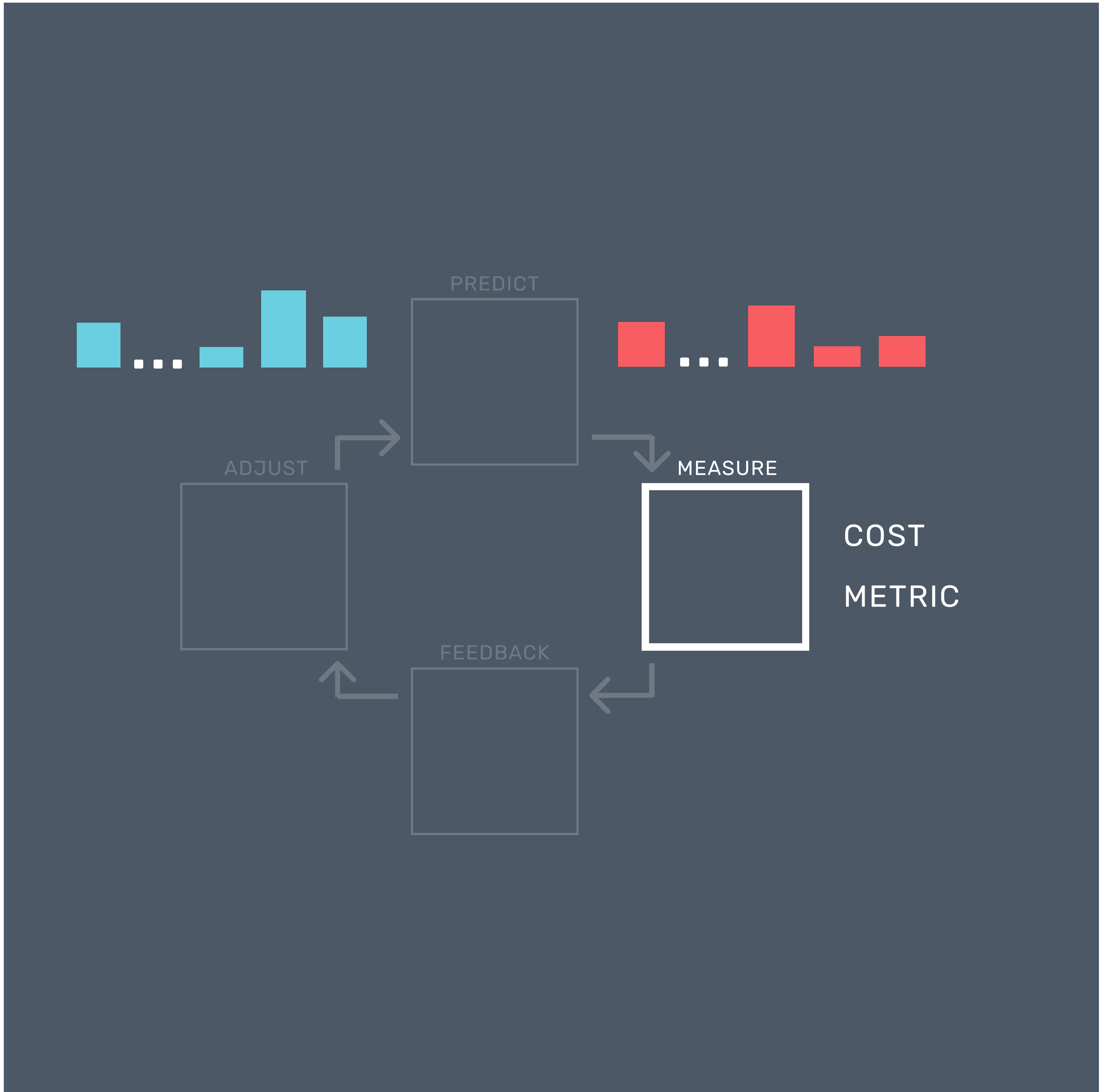
EPOCH

This completes one round of training, also called an *epoch*.



COMPLETE TRAINING

We'll repeat the four steps for 100 epochs. And once we've gone through all the epochs, training will be complete!

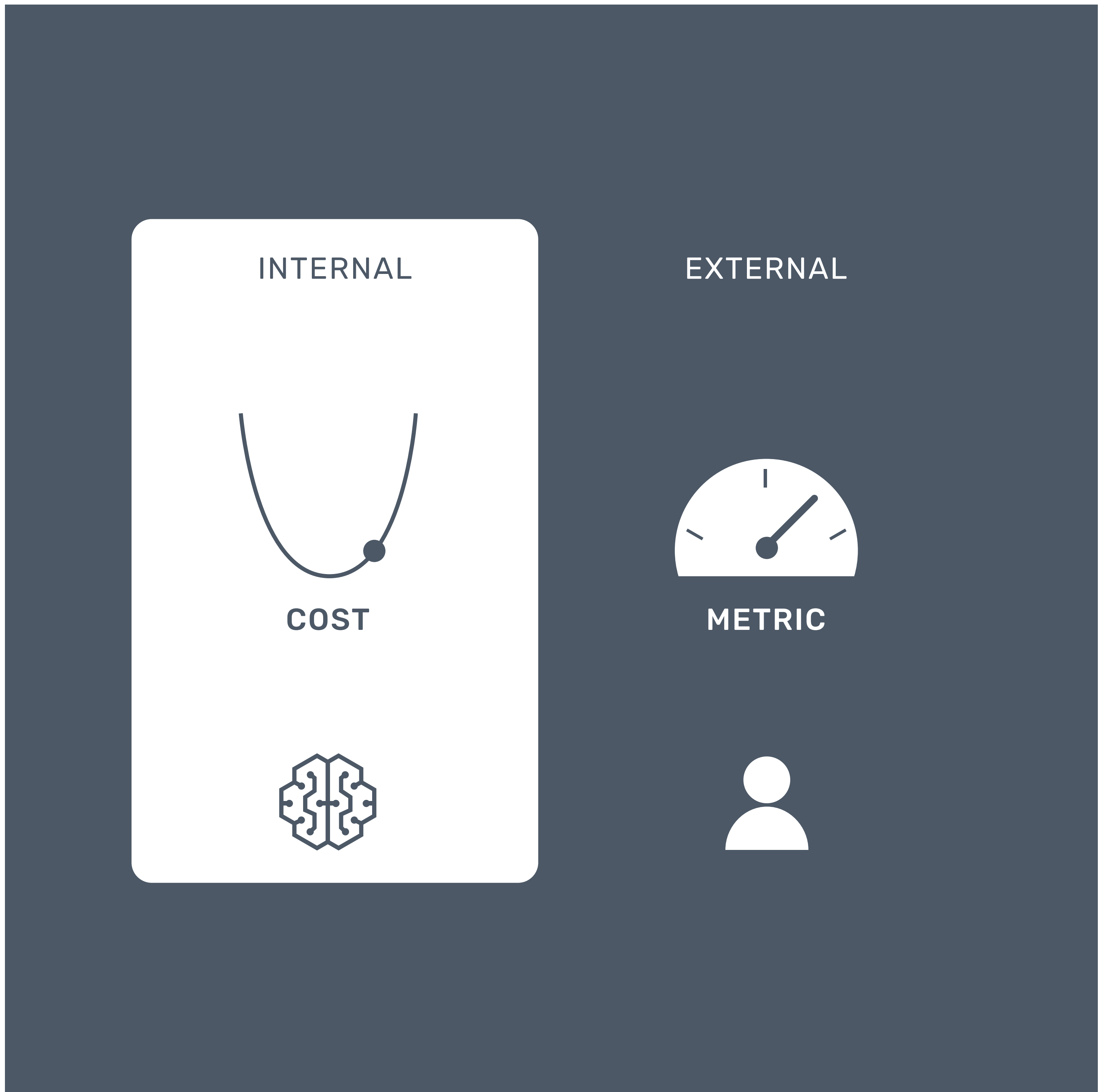


COST AND METRIC

During the *measure* step of the cycle, we actually measure two things after each epoch.

The first is *cost*. Cost is simply the average loss value over the training data points (i.e. the MSE). So far, we've only used the term *loss* for simplicity, but cost is the more precise term. In practice though, it's not uncommon to see these two terms used interchangeably.

The second is *metric*. For this task, it's also equivalent to the MSE. It may not make sense now why we need two measures for the same thing, but it will become apparent when we get to Chapters 3 and 4, where the cost and metric are different.

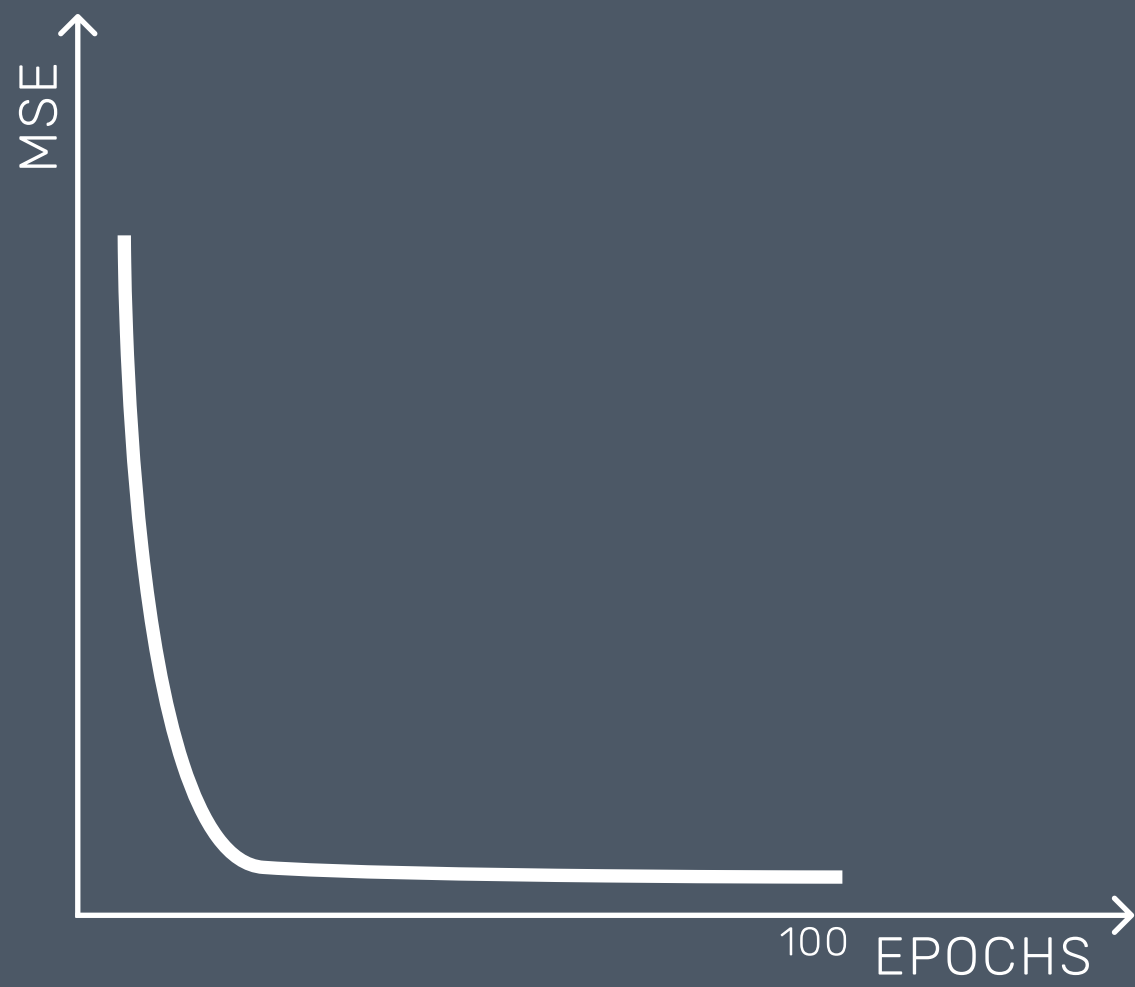


PERFORMANCE MEASURE

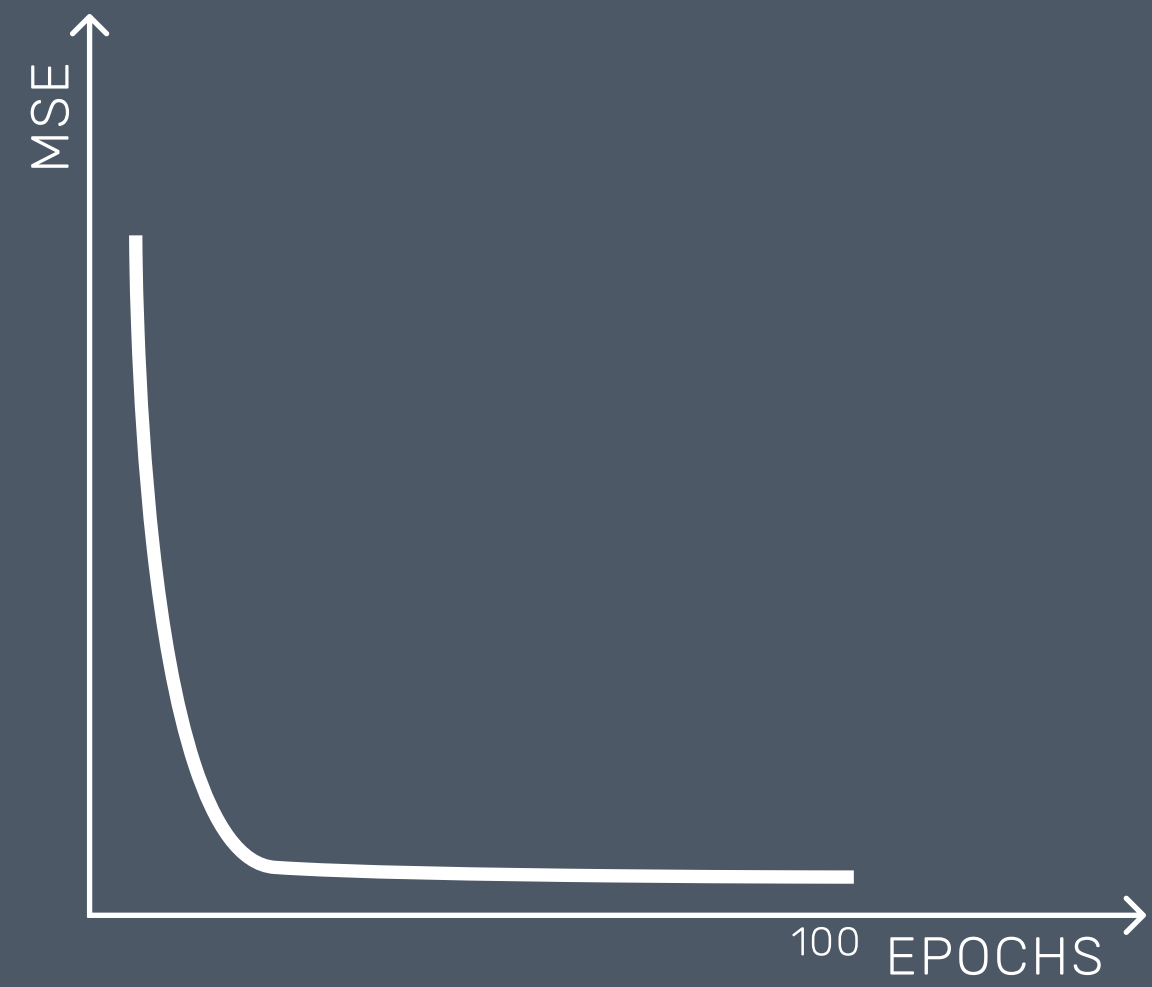
In other words, the cost is an internal performance measure - that is, for the algorithm.

Conversely, the metric is an external performance measure - that is, for the human.

COST



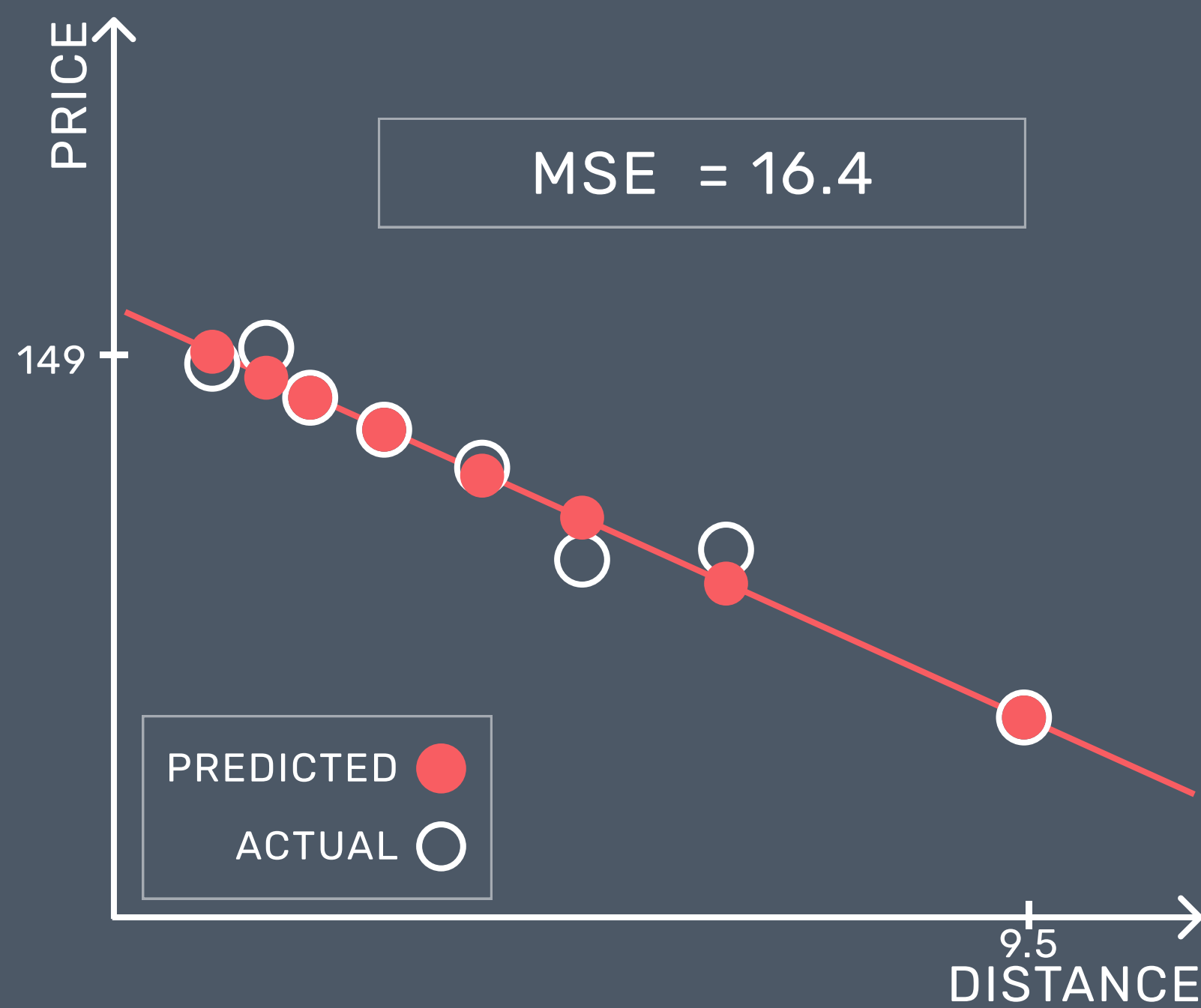
METRIC



MONITORING

Over many epochs, we can see that the MSE continues to improve and converge toward zero.

PREDICTION - TRAINING DATA



TRAINING PERFORMANCE

At the end of the 100 epochs, we have a trained model that gives a respectable MSE of 16.4.

Plotting the predicted values on a chart gives us a linear regression line that's defined by the learned parameters.

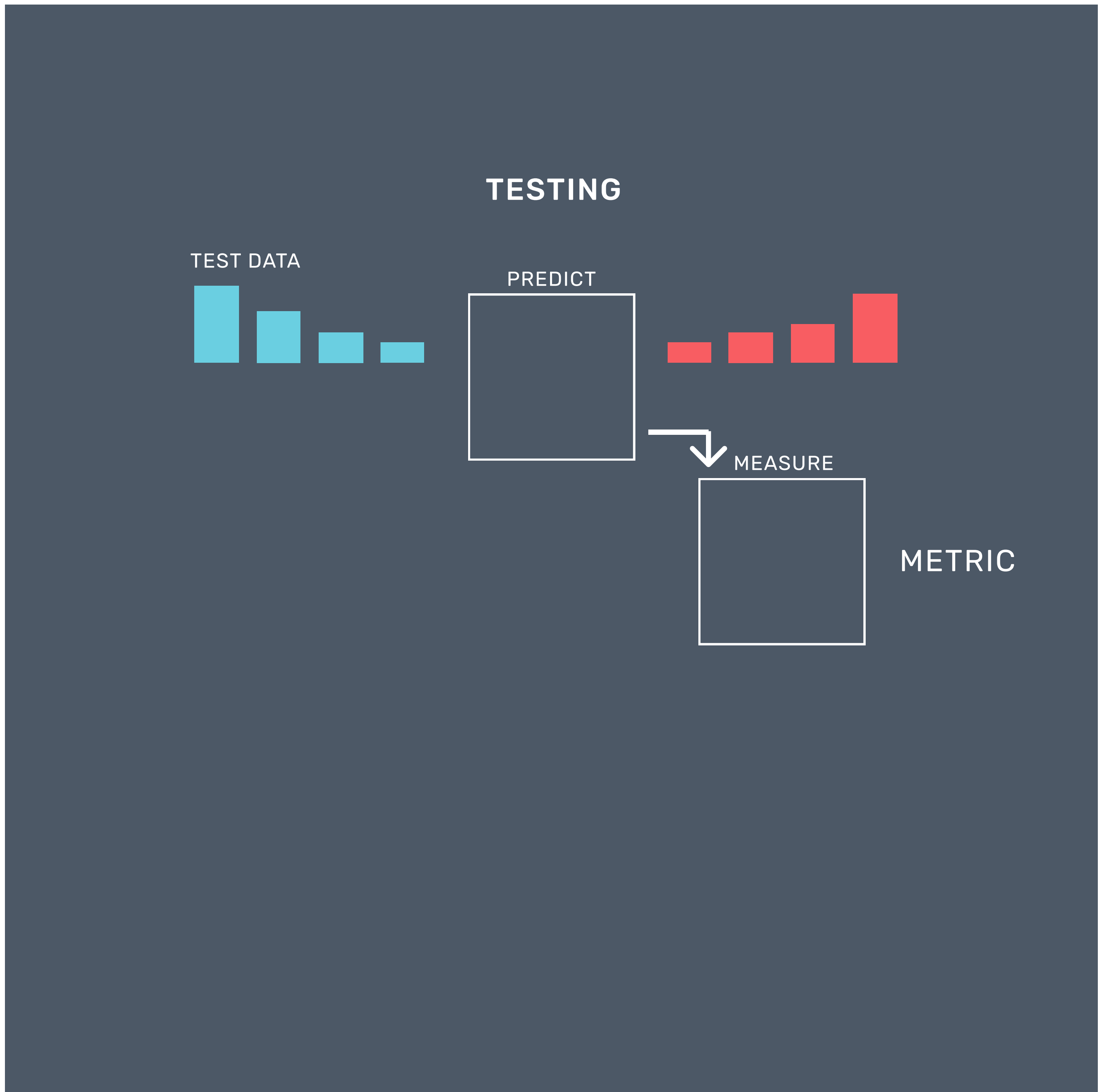
Note that the MSE can only go to zero if the actual training data are perfectly aligned along a straight line. In this case, they aren't.

	DISTANCE (MI)	PRICE (\$)
	0.5	146.00
	1.1	149.00
	1.6	140.00
	2.4	134.00
	3.5	127.00
	4.6	110.00
	6.2	112.00
	9.5	81.00
TEST DATA	0.3	156.00
	0.7	168.00
	4.9	116.00
	8.5	99.00

TESTING

Measuring the performance of a model based on training data is a good indicator of success, but it is far from the true measure. The reason is that we are measuring its performance based on the data it has seen.

We need to measure its performance based on data it has never seen. For this, we use the four test data points that we set aside.



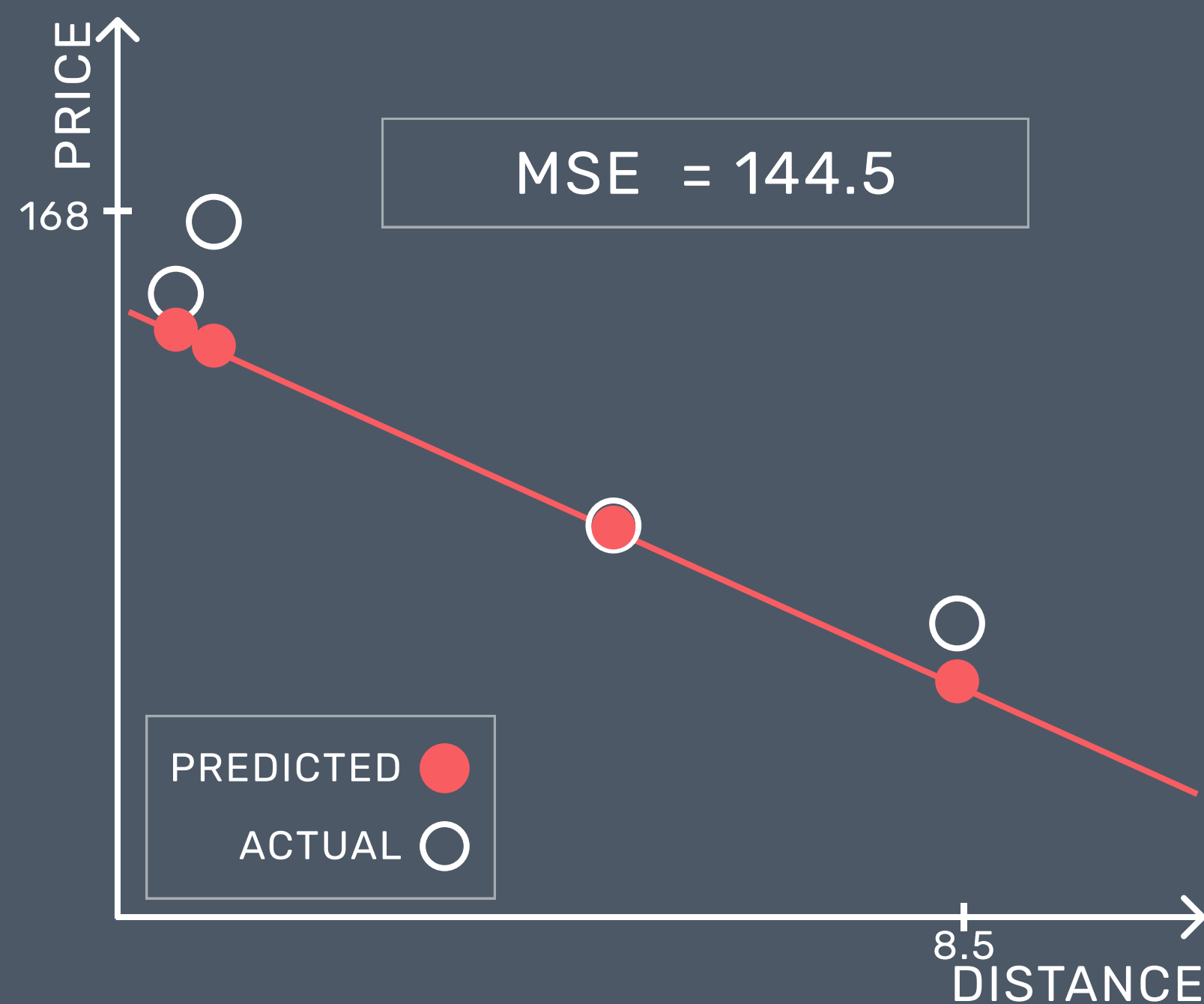
TESTING

For the test data, we don't need to go through all four steps of the cycle. We only need the *predict* and *measure* steps.

In *predict*, we pass through the features (distance) through the neural network and get the prediction (price) at the other end.

In *measure*, we compute the metric (the MSE) of the prediction. The cost is internal to the model and it's used only during training, so we won't need to consider that.

PREDICTION - TEST DATA

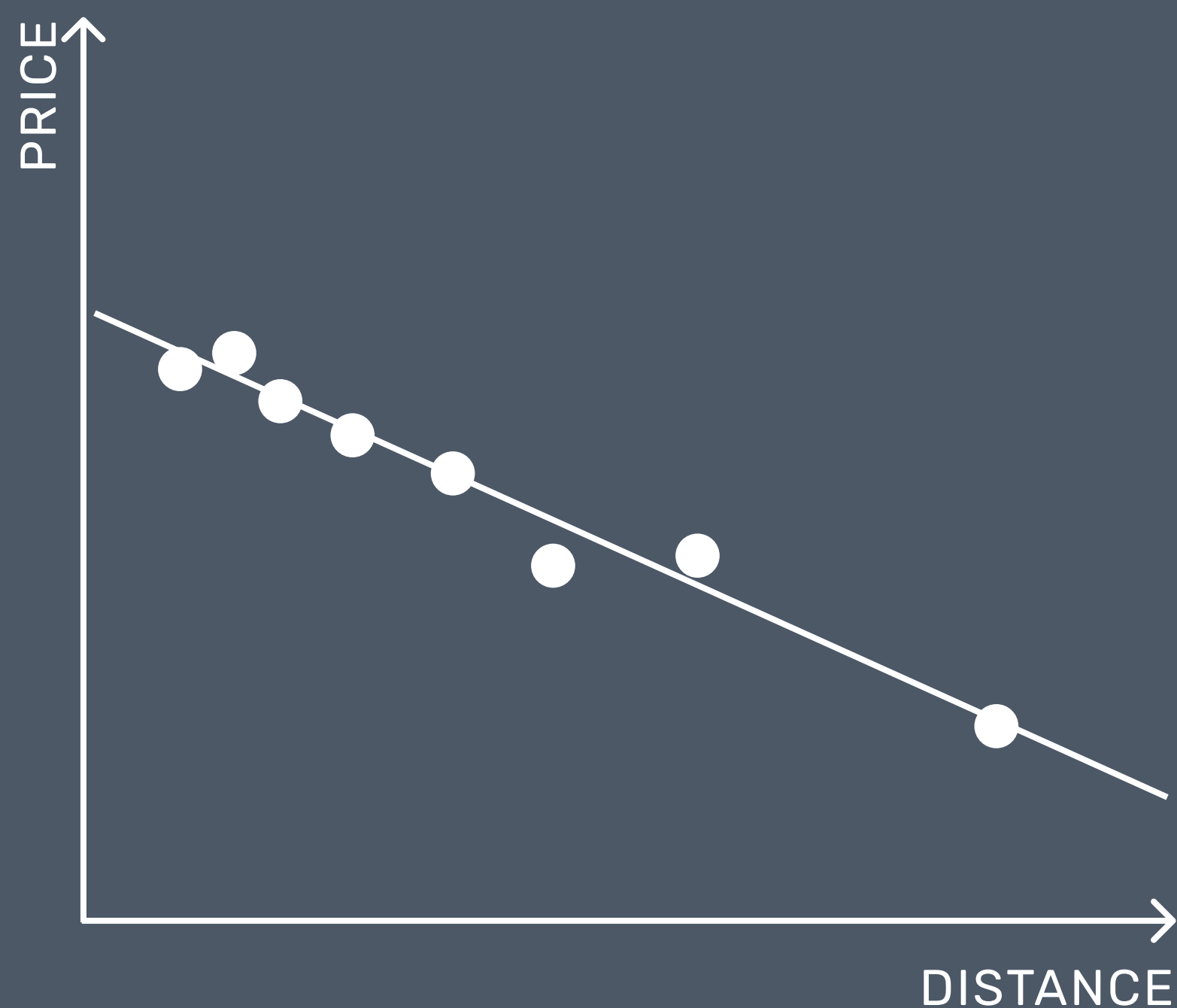


TEST PERFORMANCE

We get an MSE that is substantially worse than that of the training data. Examining this visually, this is because the test data are more sporadic compared to the training data. This indicates a slightly different distribution of the test data compared to the training data.

However, for this amount of data, we can't really confirm that, and this performance is fine.

In general, the performance with the test data will inevitably not be at the same level as with the training data. But our job is to get them as close as possible.



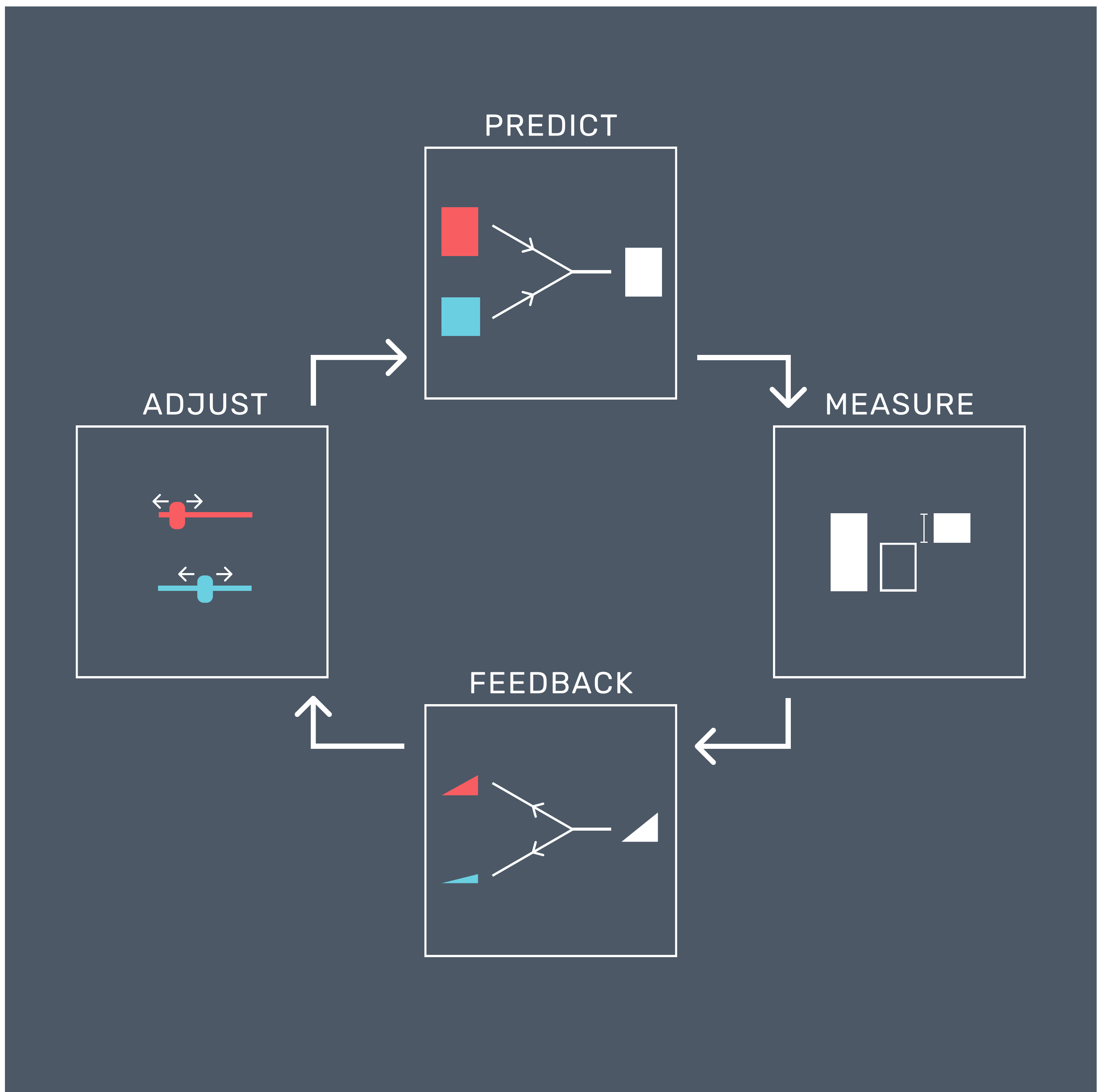
LINEAR REGRESSION

And that's the end of our first example. If you're familiar with linear regression, you might be wondering why we're taking all the trouble to use a neural network to build a linear regression model?

You are right! There are other simpler algorithms that would have achieved the same outcome.

But this is where the similarity ends. In the following chapters, we'll start to build complexities and non-linearities, which is when the neural network shines.

This chapter is all about building a foundational understanding of neural networks via a simple example. In the coming chapters, we'll continue to build on it.



RECAP

Let's do a recap of the four-step training cycle: *Predict - Measure - Feedback - Adjust*.

In *predict*, the dataset is passed through the neuron to generate predictions.

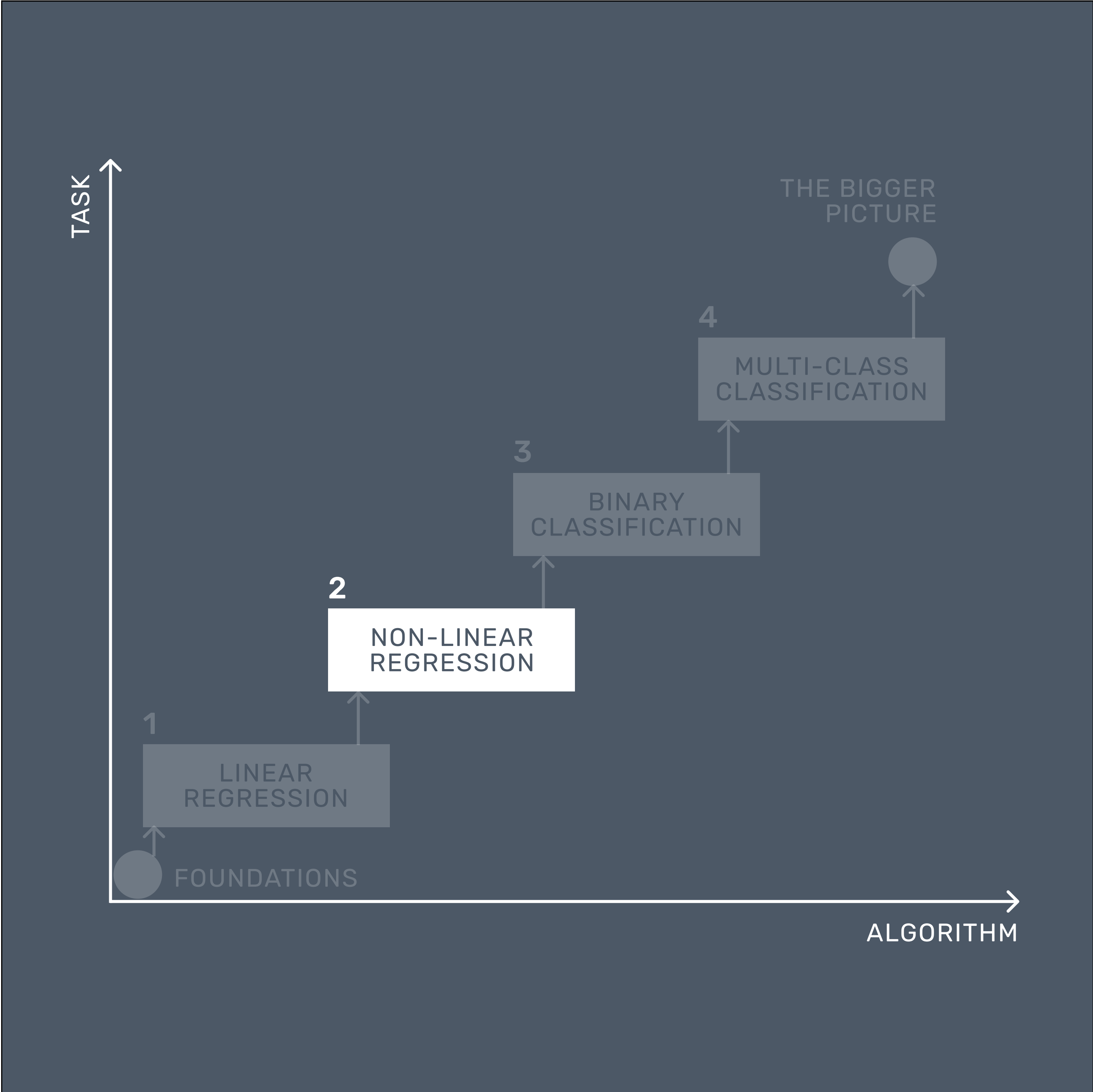
In *measure*, we measure the cost, that is how far are the predictions from the actual values.

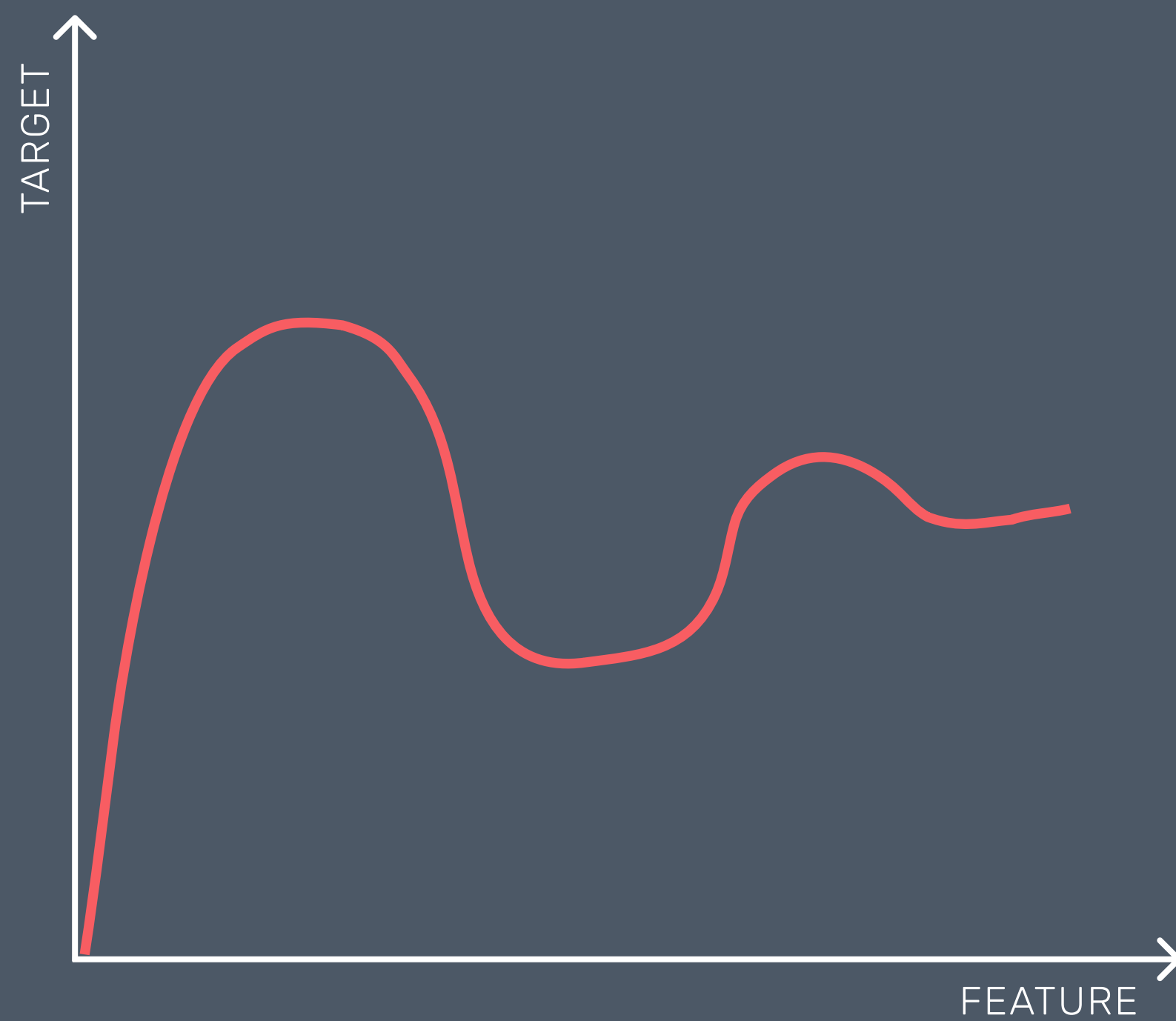
In *feedback*, we compute the parameter gradients and feed them back to the neuron.

Finally, in *adjust*, we increase or decrease the parameters based on the gradients.

Repeat through many epochs and we get a trained model.

2 - NON-LINEAR REGRESSION



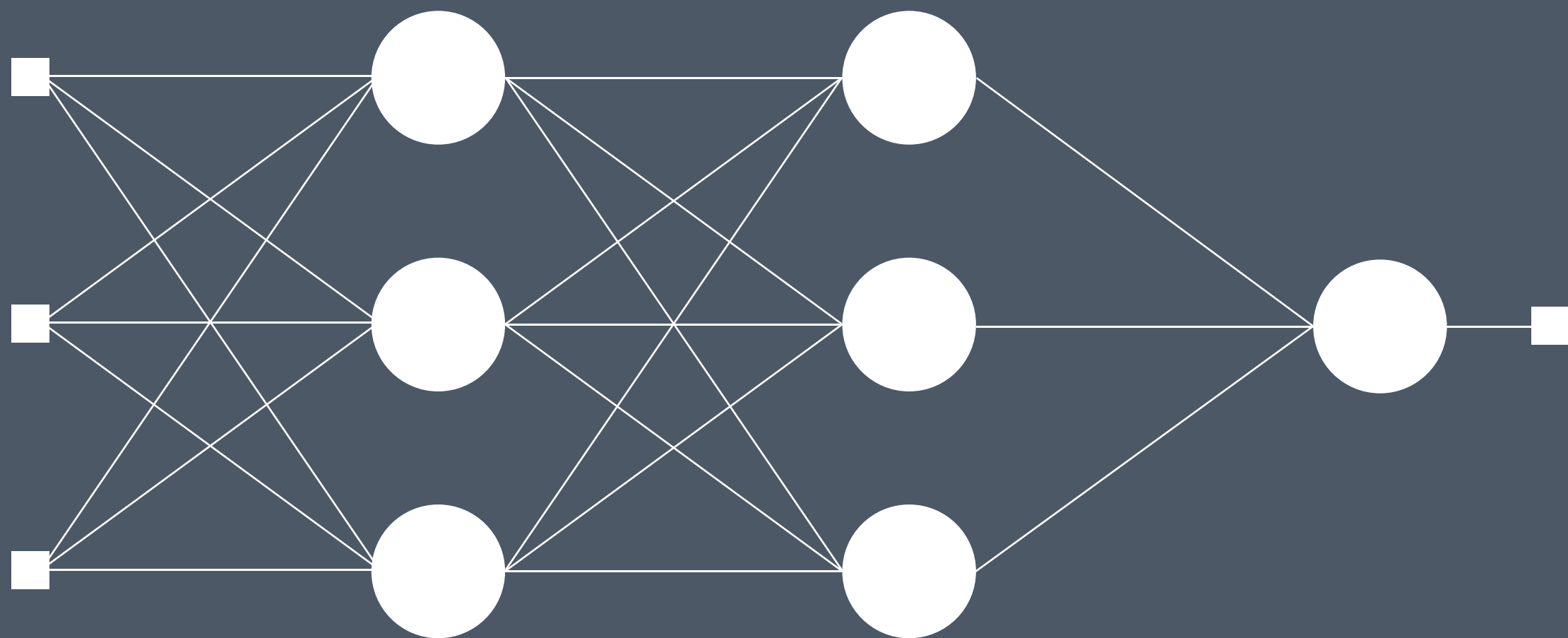


NON-LINEARITY

In the last chapter, our single-neuron neural network had the task of modeling a linear relationship.

In practice, however, we are more likely to encounter non-linear relationships. The datasets will be more complex. Instead of one, there will be many features at play.

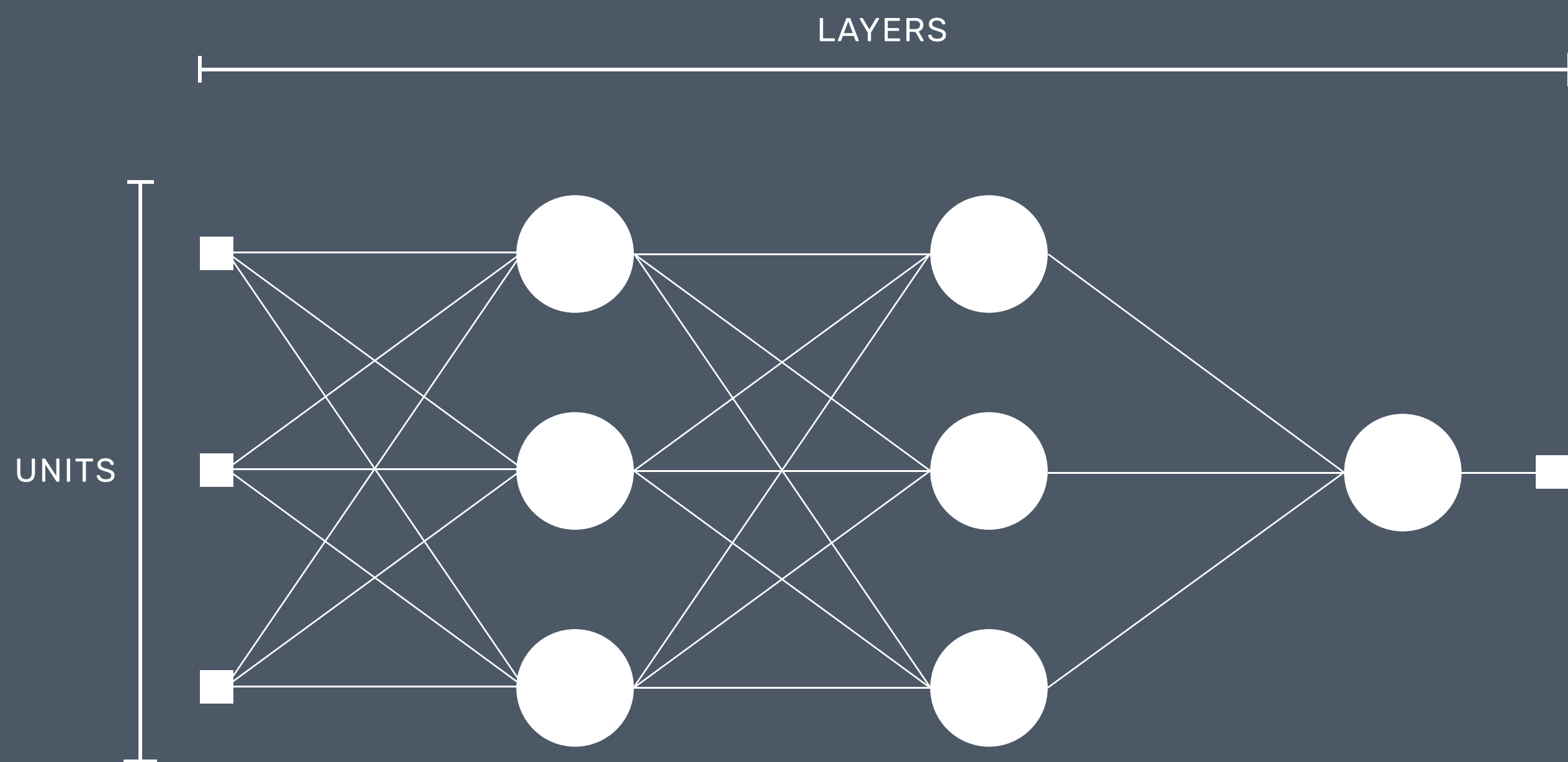
A single neuron won't be able to handle these scenarios.



NEURAL NETWORK

This is when we need a proper neural network.

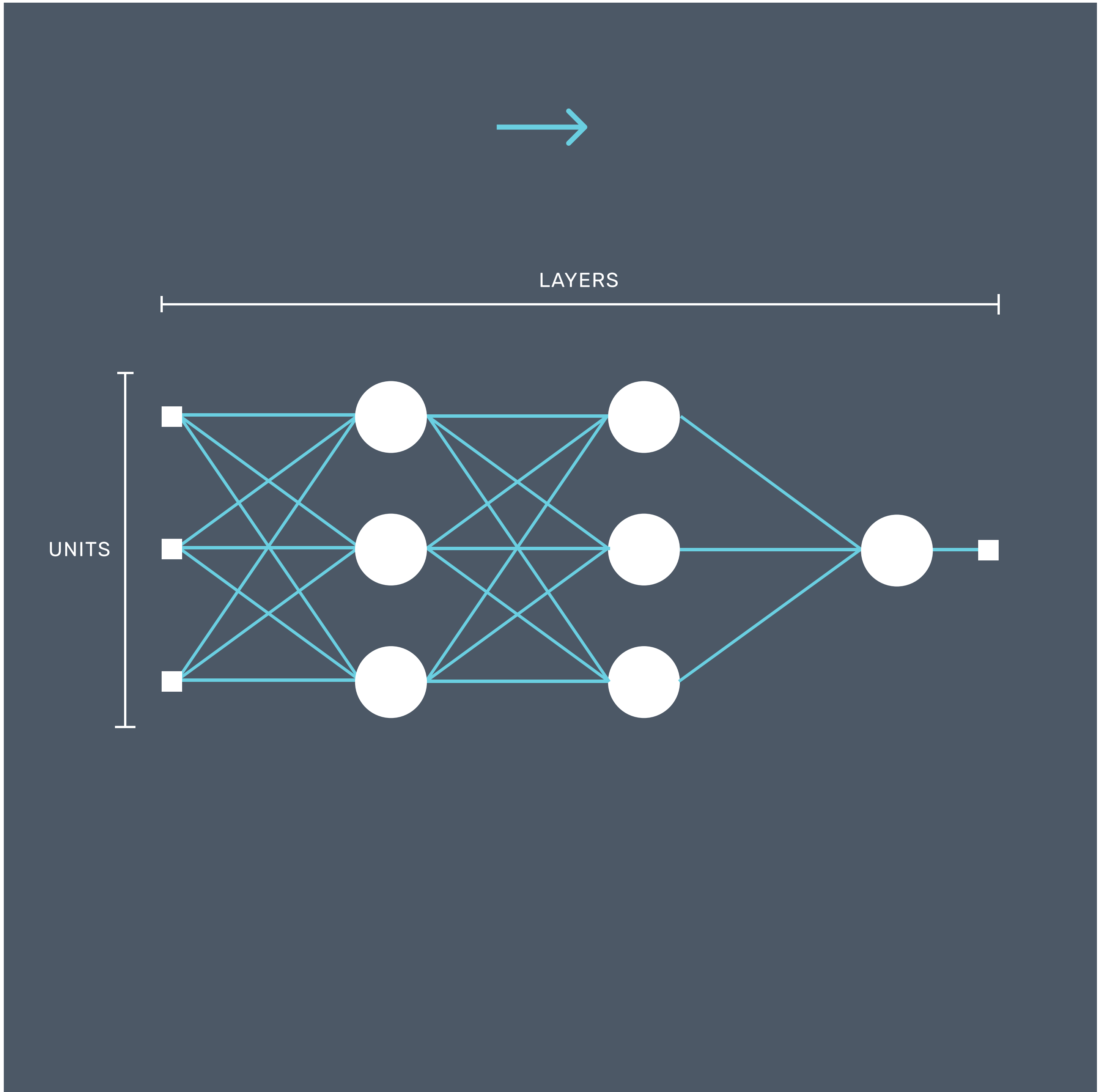
A neural network consists of its building blocks - neurons. These neurons learn uniquely at the individual level and synergistically at the collective level.



LAYERS AND UNITS

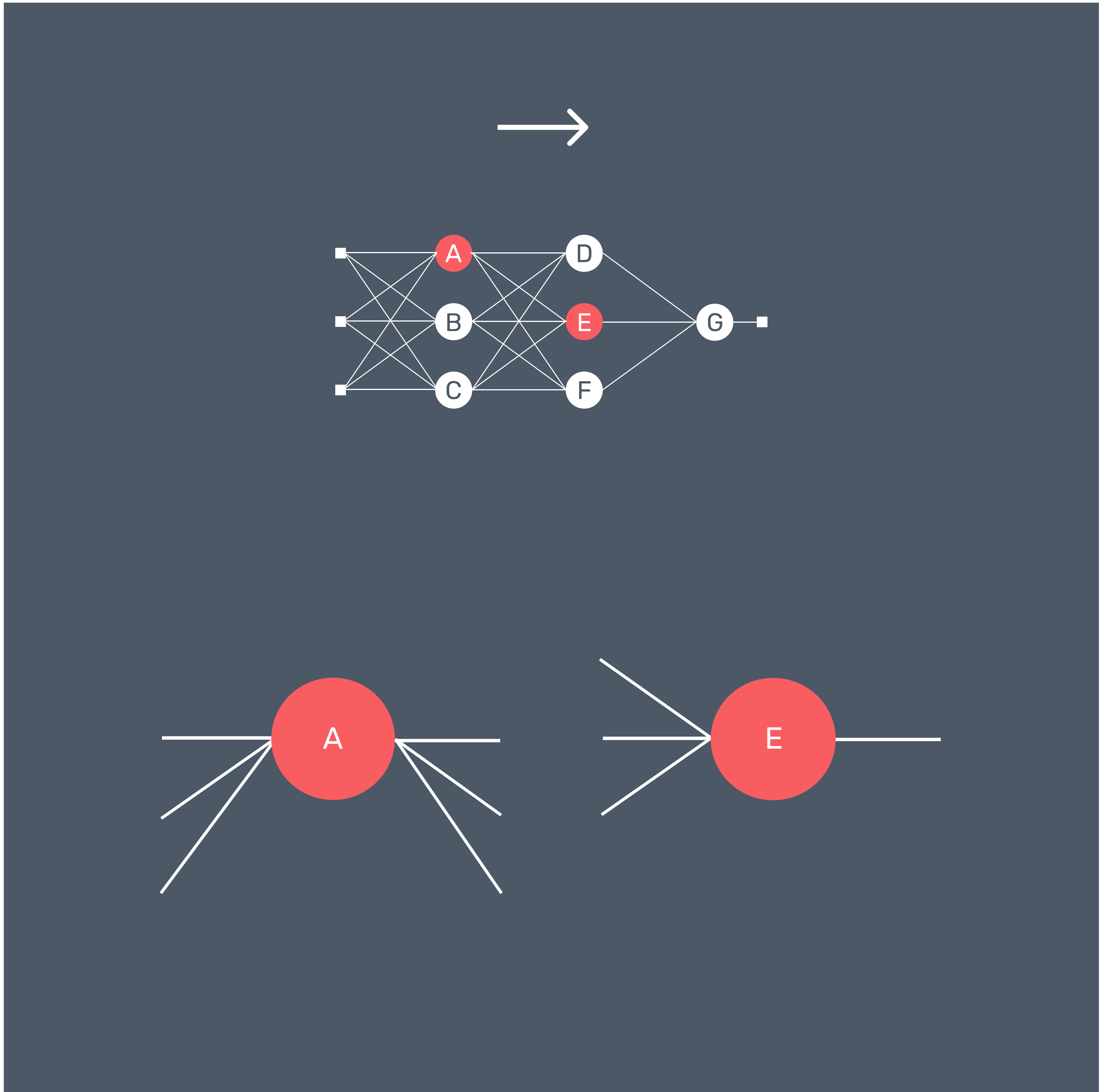
Neural networks come in various configurations called *architectures*.

In its basic architecture, a neural network consists of *layers*. Each layer has its own number of neurons, or *units*.



DATA FLOW

The lines represent the flow of data. Each neuron receives inputs from all the neurons in the preceding layers. Then it sends its output to all neurons in the next layer.

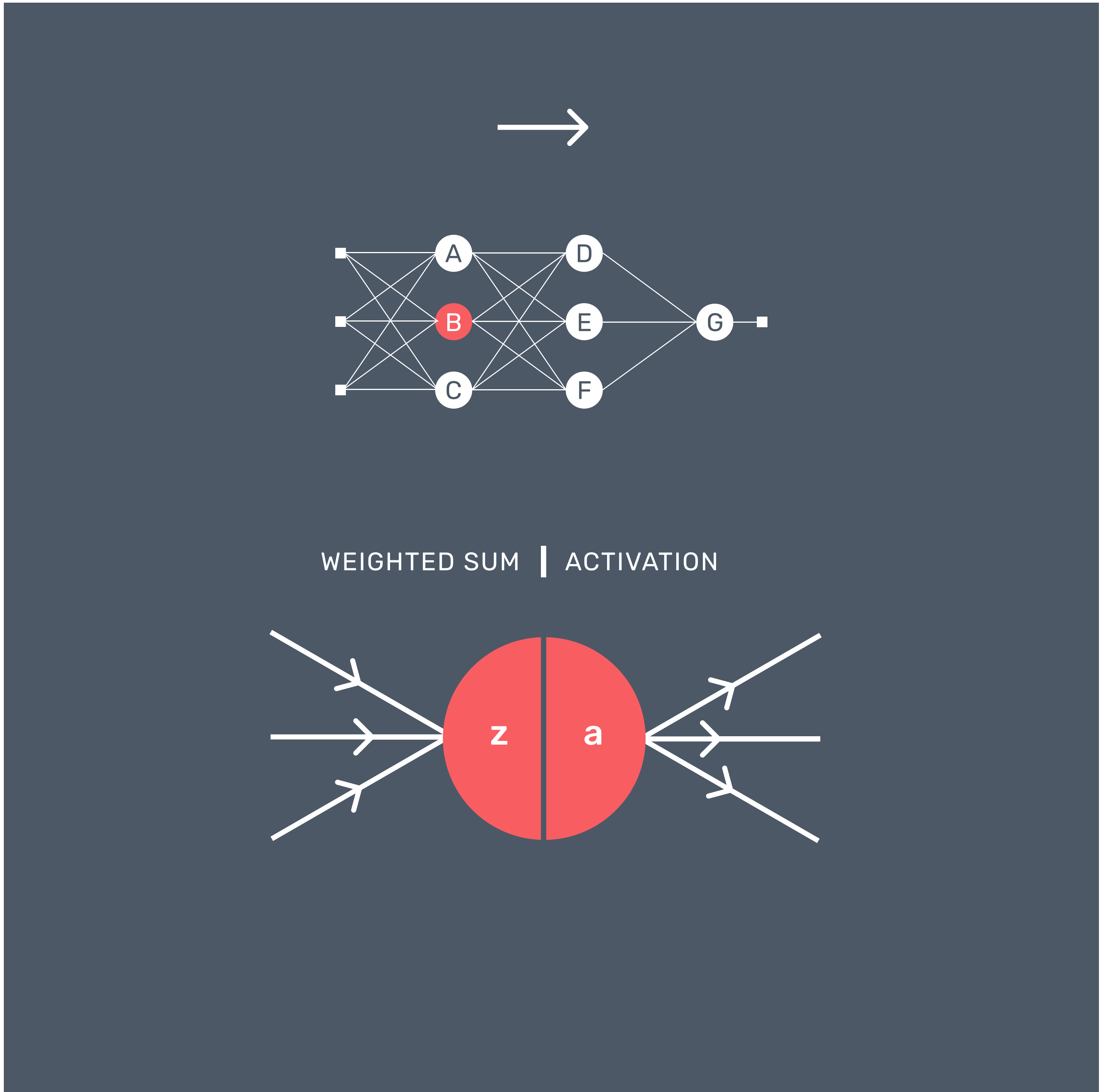


EXAMPLE

The neurons and connections can make the neural network look complicated, so let's break it down and look at a couple of examples.

Example 1: Neuron A receives three inputs directly from the data and sends its output to 3 neurons in the next layer.

Example 2: Neuron E receives three, which are the outputs of neurons A, B, and C in the previous layer, and sends its output to one neuron in the next layer.



NEURON COMPUTATIONS

Regardless of how many inputs and outputs a neuron is connected to, or in which layer the neuron is located, it goes through the same set of computations—weighted sum and activation.

Take Neuron B, for example. It takes three inputs, computes the weighted sum on these inputs, and then performs the activation. The output of the activation is then passed to three neurons in the next layer.



THE GOAL

Let's now build the neural network architecture we need for this task.

Before that, we'll define the goal for this task, which is the same as in Chapter 1—to predict a hotel's average daily room rates (i.e. *price*).

	DIST (MI)	RATING	PRICE (\$)
TRAINING DATA (24)	0.2	3.5	157.00
	0.2	4.8	155.00
	0.5	3.7	146.00
	0.7	4.3	168.00
	0.8	2.7	147.00
	1.5	3.6	136.00
	1.6	2.6	140.00
	2.4	4.7	134.00
	3.5	4.2	116.00
	3.5	3.5	127.00
	4.6	2.8	106.00
	4.6	4.2	110.00
	4.9	3.8	116.00
	6.2	3.6	112.00
	6.5	2.4	92.00
	8.5	3.1	99.00
	9.5	2.1	81.00
	9.7	3.7	92.00
	11.3	2.9	75.00
	14.6	3.8	108.00
	17.5	4.6	166.00
	18.7	3.8	188.00
	19.5	4.4	211.00
	19.8	3.6	207.00
TEST DATA (8)	0.3	4.6	156.00
	0.5	4.2	162.00
	1.1	3.5	149.00
	1.2	4.7	145.00
	2.7	2.7	123.00
	3.8	4.1	118.00
	7.3	4.6	82.00
	19.4	4.8	209.00

THE DATASET

The difference is this time, we have two features instead of one. Here, we bring back the rating feature that we left out in Chapter 1.

Another difference is the size of the dataset. We had only 12 data points in Chapter 1. For this task, we are adding 20 more, making up a total of 32 data points. We'll use 24 for training and 8 for testing.

The result is, instead of linear, our task now becomes a *non-linear regression* task. Let's see why this is so.