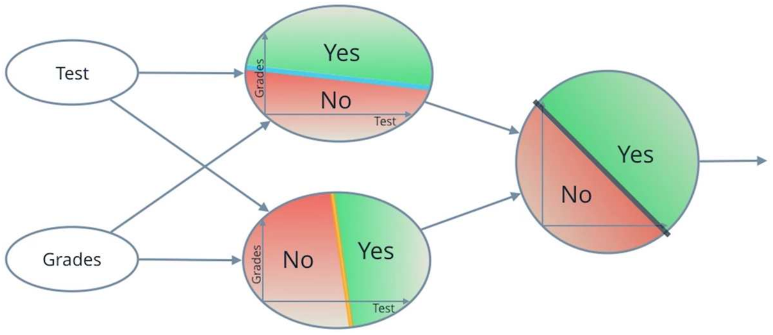
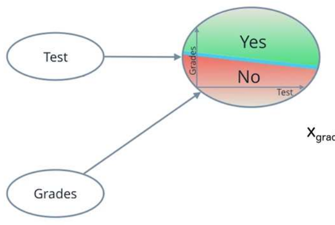


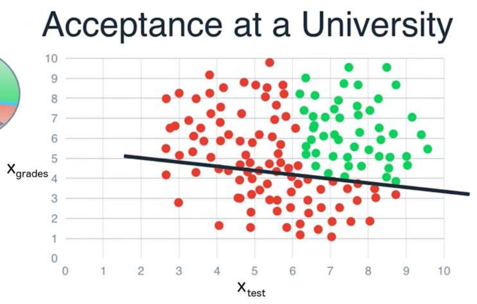
Neural Network

Perceptron

Now you've seen how a simple neural network makes decisions: by taking in input data, processing that information, and finally, producing an output in the form of a decision! Let's take a deeper dive into the university admission example and learn more about how this input data is processed.

Data, like test scores and grades, is fed into a network of interconnected nodes. These individual nodes are called **perceptrons** or neurons, and they are the basic unit of a neural network. *Each one looks at input data and decides how to categorize that data.* In the example above, the input either passes a threshold for grades and test scores or doesn't, and so the two categories are: yes (passed the threshold) and no (didn't pass the threshold). These categories then combine to form a decision -- for example, if both nodes produce a "yes" output, then this student gains admission into the university.





Let's zoom in even further and look at how a single perceptron processes input data.

The perceptron above is one of the two perceptrons from the video that help determine whether or not a student is accepted to a university. It decides whether a student's grades are high enough to be accepted to the university. You might be wondering: "How does it know whether grades or test scores are more important in making this acceptance decision?" Well, when we initialize a neural network, we don't know what information will be most important in making a decision. It's up to the neural network to *learn for itself* which data is most important and adjust how it considers that data. It does this with something called **weights**.

Weights

When input data comes into a perceptron, it gets multiplied by a weight value that is assigned to this particular input. For example, the perceptron above have two inputs, `tests` for test scores and `grades`, so it has two associated weights that can be adjusted individually. These weights start out as random values, and as the neural network network learns more about what kind of input data

leads to a student being accepted into a university, the network adjusts the weights based on any errors in categorization that the previous weights resulted in. This is called **training** the neural network.
A higher weight means the neural network considers that input more important than other inputs, and lower weight means that the data is considered less important. An extreme example would be if test scores had no affect at all on university acceptance; then the weight of the test score input data would be zero and it would have no affect on the output of the perceptron.

Summing the Input Data

So, each input to a perceptron has an associated weight that represents its importance and these weights are determined during the learning process of a neural network, called training. In the next step, the weighted input data is summed up to produce a single value, that will help determine the final output - whether a student is accepted to a university or not. Let's see a concrete example of this.

We'll use w_{grades} for the weight of `grades` and w_{test} for the weight of `test`. For the image above, let's say that the weights are: $w_{grades} = -1, w_{test} = -0.2$. You don't have to be concerned with the actual values, but their relative values are important. w_{grades} is 5 times larger than w_{test}, which means the neural network considers `grades` input 5 times more important than `test` in determining whether a student will be accepted into a university. After applying these weights to the input data, the perceptron then sums these numbers to get a value we'll call the **linear combination**, which is pictured below. In the equation, x_{grades} represents `grades` and x_{test} represents `test` scores.
The linear combination represents how much the perceptron believes a student will be accepted into a university.

$$w_{grades} * x_{grades} + w_{test} * x_{test} = \sum_{i=1}^{m} w_i * x_i$$

Linear Combination

Calculating the Output with an Activation Function

Finally, the result of the perceptron's summation is turned into an output signal! This is done by feeding the linear combination into an **activation function**.
One of the simplest activation functions is the **Heaviside step function**. This function returns a **0** if the linear combination is less than 0. It returns a **1** if the linear combination is positive or equal to zero. The **Heaviside step function** is shown below, where h is the calculated linear combination:

$$f(h) = \begin{cases} 0 & \text{if } h < 0 \\ 1 & \text{otherwise} \end{cases}$$

Heaviside Step Function

In the university acceptance example above, we used the weights $w_{grades} = -1, w_{test} = -0.2$.
Since $w_{grades}$and w_{test} are negative values, the activation function will only return a 1 if grades and test are 0! This is because the range of values from the linear combination using these weights and inputs are $(-\infty, 0]$.

We want more than one set of inputs to return a 1. We want a range of scores and grades that will be acceptable for the university. You can solve this problem by adding a single number to the linear combination called a **bias**, b. Just like the weights, the bias is also updated and changed by the neural network during training. Now with this value, we have a complete perceptron formula:

$$f(x_1, x_2, ..., x_m) = \begin{cases} 0 & \text{if } b + \sum w_i * x_i < 0 \\ 1 & \text{otherwise} \end{cases}$$

Perceptron Formula

This formula returns 1 if the input $(x_1,x_2,...,x_m)$ belongs to the accepted-to-university category or returns 0if it doesn't. The input is made up of one or more **real numbers**, each one represented by x_i, where m is the number of inputs.

Then the neural network starts to learn! Initially, the weights (w_i) and bias (b) are assigned a random value, and then they are updated using a learning algorithm like gradient descent. The weights and biases change so that the next training example is more accurately categorized, and patterns in data are "learned" by the neural network.

Now that you have a good understanding of perceptions, let's put that knowledge to use. In the next section, you'll create the AND perceptron from the *Neural Networks* video by setting the values for weights and bias.