Activation Function

Alternatives to the sigmoid activation

Background: Awareness in Demand Prediction

Imagine you're trying to predict whether a T-shirt will be a **top seller** based on features like:

- **Price**
- **Shipping cost**
- **Marketing effort**
- **Material quality**

Why Modeling Awareness Matters

"Awareness" plays a **crucial intermediate role**. A strong marketing campaign can increase awareness, which in turn makes it more likely that the product sells well.

Now, consider these scenarios:

Situation	Awareness Level
No marketing, hidden product	0 (No one knows about it)
Local ads, a few customers aware	Low (e.g., 0.2)
Influencer mentions it	Medium (e.g., 3.5)
Goes viral on social media	Very high (e.g., 100+)

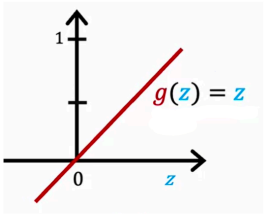
So you can see that awareness is:

- **Not binary** (not just 0 or 1)
- **Not limited to [0, 1]** like sigmoid output
- Can take on **any non-negative real number**, including very large values

Now list some common activation functions:

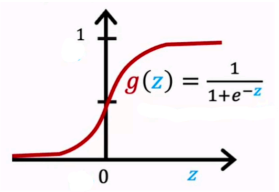
Linear

$$y \in (-\infty, \infty)$$
$$g(z) = \underbrace{\vec{w} \cdot \vec{x} + b}_{z}$$



Sigmoid

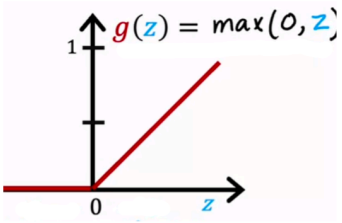
$$y \in [0, 1]$$
$$g(z) = \frac{1}{1 + e^{-z}}$$



ReLU - Rectified Linear Unit

$$y \in [0, \infty)$$
$$g(z) = max(0, z)$$

$$g(z) = \begin{cases} 0, & \text{if } z \leq 0 \\ z, & \text{if } z > 0 \end{cases}$$



Why Sigmoid is a Problem Here

The sigmoid activation function: $g(z) = \frac{1}{1+e^{-z}}$

- Squashes all values into the range $(0, 1)$
- Loses the ability to represent **high degrees of awareness**
- Compresses large values into something close to 1, making them indistinguishable from moderately large values

This makes it **poor for modeling variables** like awareness that can vary widely in magnitude.

Why ReLU is Better

Instead, using the **ReLU activation function**: $g(z) = \max(0, z)$

- Allows the model to express **awareness levels from 0 to very large values**
- Doesn't compress large values — it retains their magnitude
- Introduces **non-linearity**, while still being simple and computationally efficient

In effect, ReLU enables the model to learn that if marketing input increases awareness, and awareness crosses a certain threshold, the chance of a top seller dramatically increases.

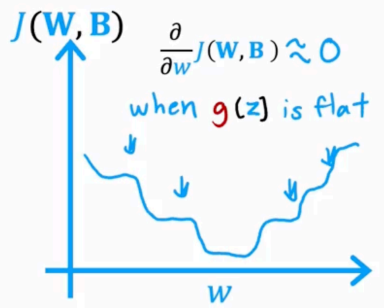
Choosing Activation functions

Output Layer

Cases	Output's Range	Preferred Activation Function
Binary Classification	$0 \text{ or } 1$	Logistic Activation
Regression	$- \text{ or } +$	Linear Activation
Regression	$0 \text{ or } +$	ReLU Activation (most common choice)

The **ReLU (Rectified Linear Unit)** activation function is the most commonly used choice for hidden layers in neural networks today. Compared to the sigmoid (logistic) function:

1. **ReLU is faster to compute** and more efficient for training.
2. **ReLU avoids flat regions** that slow down learning—unlike the sigmoid function, which has flat areas near 0 and 1 where gradients become very small.
    a. These flatter regions in the sigmoid curve can significantly reduce the speed and effectiveness of gradient descent.



b. ReLU only flattens for inputs less than or equal to 0, which allows it to **maintain stronger gradient signals** for positive values.

> 💡 **Sigmoid functions are now rarely used in hidden layers**, though they still appear in the **output layer of binary classification problems**. **Linear activation functions should not be used in hidden layers**, as stacking them results in a model no more powerful than basic linear or logistic regression.

Choosing Activation Function Summary:

The recommended activation functions are:

- Output Layer: Sigmoid for binary classification, linear for regression with both positive and negative values, and ReLU for non-negative values.
- Hidden Layers: Use ReLU as the default activation function.

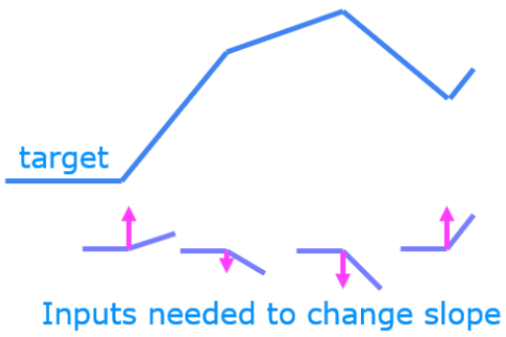
```
model = Sequential ([
    Dense(units=25, activation='relu'),
    Dense(units=15, activation='relu'),
    Dense(units=1, activation='sigmoid'),
])
```

Why do we need activation functions?

- If you use **linear activation functions** (i.e., $g(z) = z$) in **all layers** of a neural network, the entire network becomes mathematically equivalent to **linear regression**.

- This is because a linear function of a linear function is still just a **linear function**—so stacking layers adds no extra power or complexity.
- A simple example with one hidden layer and one output layer shows that with linear activation, the final output is just a linear expression in x: $a_2 = w_2(w_1x + b_1) + b_2 = wx + b$
- Therefore, using only linear activations defeats the purpose of building deep networks, which are meant to model **non-linear patterns**.
- If you keep the hidden layers linear but apply a **sigmoid (logistic)** function only at the output, the entire network behaves like **logistic regression**.
- To learn complex, non-linear relationships, you need **non-linear activation functions** (e.g., **ReLU**) in the **hidden layers**.
- ReLU is typically recommended for hidden layers because it's simple, fast, and avoids the limitations of linear functions.

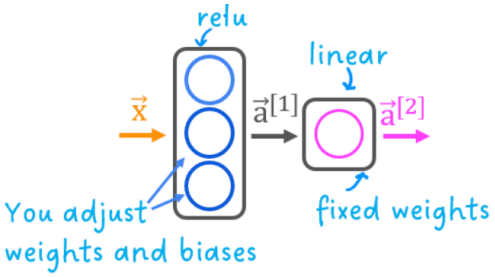
Why Non-Linear Activations?



source: C2_W2

The function shown is composed of linear pieces (piecewise linear). The slope is consistent during the linear portion and then changes abruptly at transition points. At transition points, a new linear function is added which, when added to the existing function, will produce the new slope. The new function is added at transition point but does not contribute to the output prior to that point. The non-linear activation function is responsible for disabling the input prior to and sometimes after the transition points. The following exercise provides a more tangible example.

Using the network below in a **regression** problem where you must model a **piecewise linear target** :



$$a_0^{[1]} = relu(x_0 \cdot (-2) + 2)$$
$$a_1^{[1]} = relu(x_0 \cdot w_1^{[1]} + b_1^{[1]})$$
$$a_2^{[1]} = relu(x_0 \cdot w_2^{[1]} + b_2^{[1]})$$

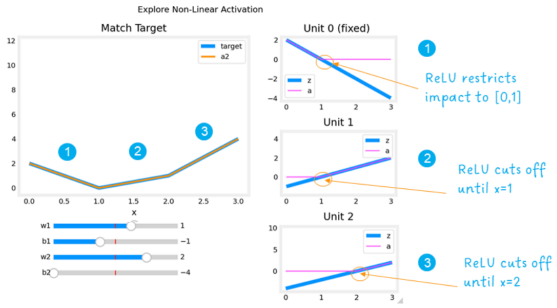
adjust

$$a_0^{[2]} = a_0^{[1]} + a_1^{[1]} + a_2^{[1]} + 0)$$

fixed weights

The network has 3 units in the first layer. Each is required to form the target. Unit 0 is pre-programmed and fixed to map the first segment. You will modify weights and biases in unit 1 and 2 to model the 2nd and 3rd segment. The output unit is also fixed and simply sums the outputs of the first layer.

Using the sliders below, modify weights and bias to match the target. Hints: Start with `w1` and `b1` and leave `w2` and `b2` zero until you match the 2nd segment.
The goal of this exercise is to appreciate how the ReLU's non-linear behavior provides the needed ability to turn functions off until they are needed. Let's see how this worked in this example.
The plots on the right contain the output of the units in the first layer.



Starting at the top, unit 0 is responsible for the first segment marked with a 1. Both the linear function z and the function following the ReLU a are shown. You can see that the ReLU cuts off the function after the interval [0,1]. This is important as it prevents Unit 0 from interfering with the following segment.

Unit 1 is responsible for the 2nd segment. Here the ReLU kept this unit quiet until after x is 1. Since the first unit is not contributing, the slope for unit 1, $w^{[1]}_1$, is just the slope of the target line. The bias must be adjusted to keep the output negative until x has reached 1. Note how the contribution of Unit 1 extends to the 3rd segment as well.

Unit 2 is responsible for the 3rd segment. The ReLU again zeros the output until x reaches the right value.The slope of the unit, $w^{[1]}_2$, must be set so that the sum of unit 1 and 2 have the desired slope. The bias is again adjusted to keep the output negative until x has reached 2.

The "off" or disable feature of the ReLU activation enables models to stitch together linear segments to model complex non-linear functions.

❓ **QUES:** True/False? A neural network with many layers but no activation function (in the hidden layers) is not effective; that's why we should instead use the linear activation function in every hidden layer.

- True
- False 👍🏻👍🏻

Explain: Yes! A neural network with many layers but no activation function is not effective. A linear activation is the same as **"no activation function"**.