Optional Lab - ReLU activation

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
plt.style.use('./deeplearning.mplstyle')
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LeakyReLU
from tensorflow.keras.activations import linear, relu, sigmoid
%matplotlib widget
from matplotlib.widgets import Slider
from lab_utils_common import dlc
from autils import plt_act_trio
from lab_utils_relu import *
import warnings
warnings.simplefilter(action='ignore', category=UserWarning)
```

2 - ReLU Activation

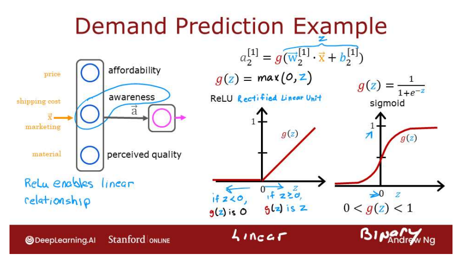
This week, a new activation was introduced, the Rectified Linear Unit (ReLU).

$$a = max(0, z) \quad \text{\# ReLU function}$$

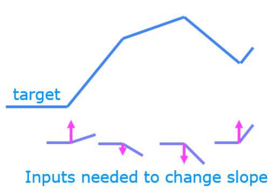
In [2]:
```python
plt_act_trio()
```

```
Canvas(toolbar=Toolbar(toolitems=[('Home', 'Reset original view', 'home', 'home'), ('Back', 'Back to previous …
```

The example from the lecture on the right shows an application of the ReLU. In this example, the derived "awareness" feature is not binary but has a continuous range of values. The sigmoid is best for on/off or binary situations. The ReLU provides a continuous linear relationship. Additionally it has an 'off' range where the output is zero. The "off" feature makes the ReLU a Non-Linear activation. Why is this needed? Let's examine this below.

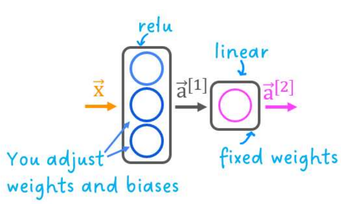


Why Non-Linear Activations?



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.

The exercise will use the network below in a regression problem where you must model a piecewise linear target :



$$a_0^{[1]} = relu(x_0 \cdot (-2) + 2)$$
$$a_1^{[1]} = relu(x_0 \cdot \boxed{w_1^{[1]} + b_1^{[1]}})$$
$$a_2^{[1]} = relu(x_0 \cdot \boxed{w_2^{[1]} + b_2^{[1]}})$$
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. Clicking rather than sliding is quicker. If you have trouble, don't worry, the text below will describe this in more detail.