Rectified Linear Unit (ReLU)

The rectified linear unit or ReLU activation function is illustrated in Figure 29-9 and works by setting the activation to 0 for values, x, less than 0 and a linear slope of 1 when values, x, are greater than 0.

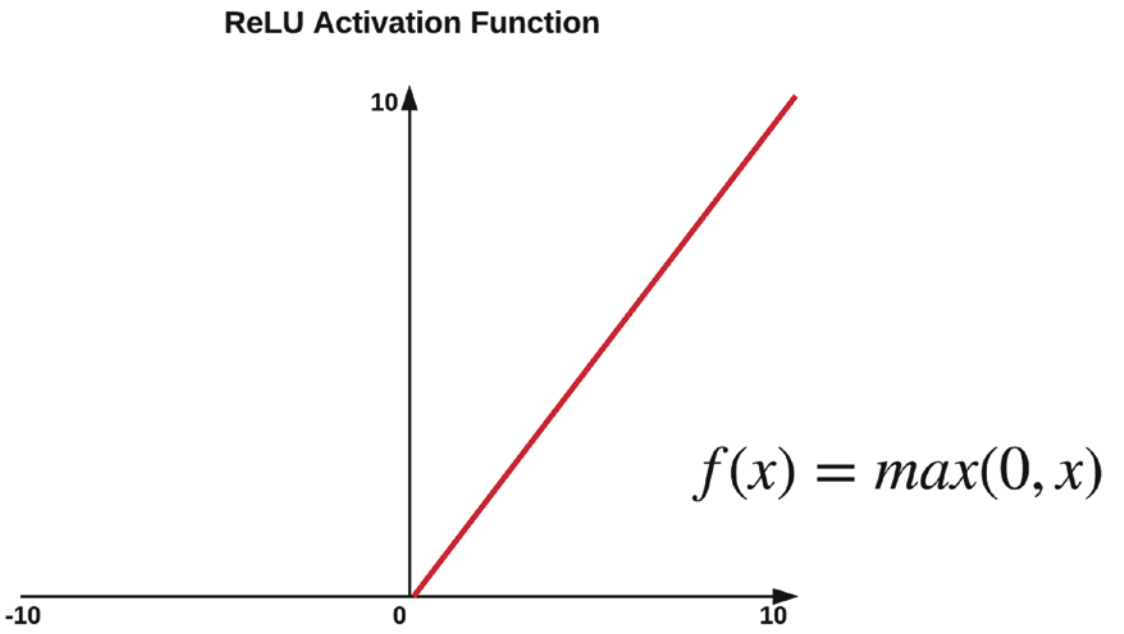
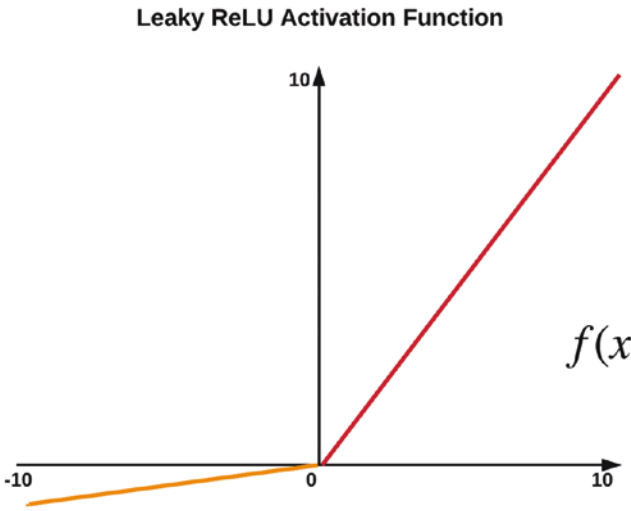


Figure 29-9. *ReLU activation function*

ReLU offers a vast improvement on the tanh and sigmoid activation functions by greatly mitigating the vanishing and exploding gradient problem. However, some gradients can still die out during backpropagation with a large learning rate. However, with a well-defined learning rate, we should not have a problem.

Leaky ReLU

Leaky ReLU is another activation function that is proposed to solve the case of some neurons completely dying out in ReLU by avoiding zero gradients. Leaky ReLU is illustrated in Figure 29-10. The function works by setting the activation to a small negative slope when the value $x < 0$.



$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Figure 29-10. *Leaky ReLU activation function*

Maxout

The Maxout activation function generalizes the ReLU and leaky ReLU functions and hence takes advantage of the efficiency of ReLU while avoiding its pitfalls of some neurons dying out. In any case, a trade-off needs to be made, because Maxout increases the parameter size of each neuron during training.

As a rule of thumb, different types of activation functions are not mixed in the same network. Also, ReLU is typically used for the hidden layers, and the softmax activation is used for classification problems at the output layer since this layer returns a probability of membership of a particular class.

This chapter provided an overview on how to train a predictive model using neural networks. This chapter ends Part 5 on introducing deep learning. The chapters in Part 6 will cover deep learning algorithms and their implementation with TensorFlow and Keras.