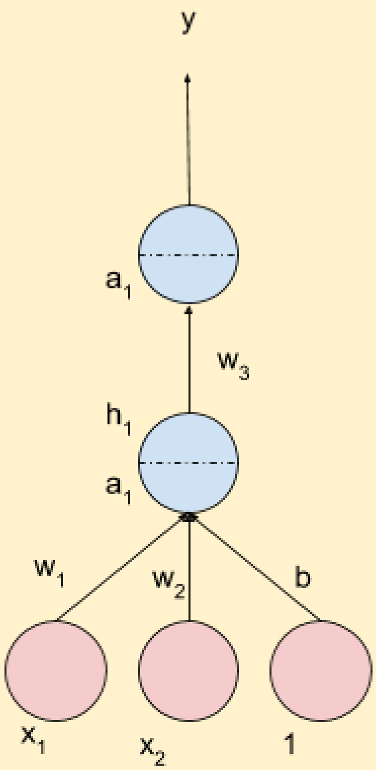
PadhAI: Activation Functions & Initialization Methods

One Fourth Labs

Tanh and ReLU Activation Functions

Is there any caveat in using ReLU?

1. Consider the following deep neural network that uses the ReLU activation function



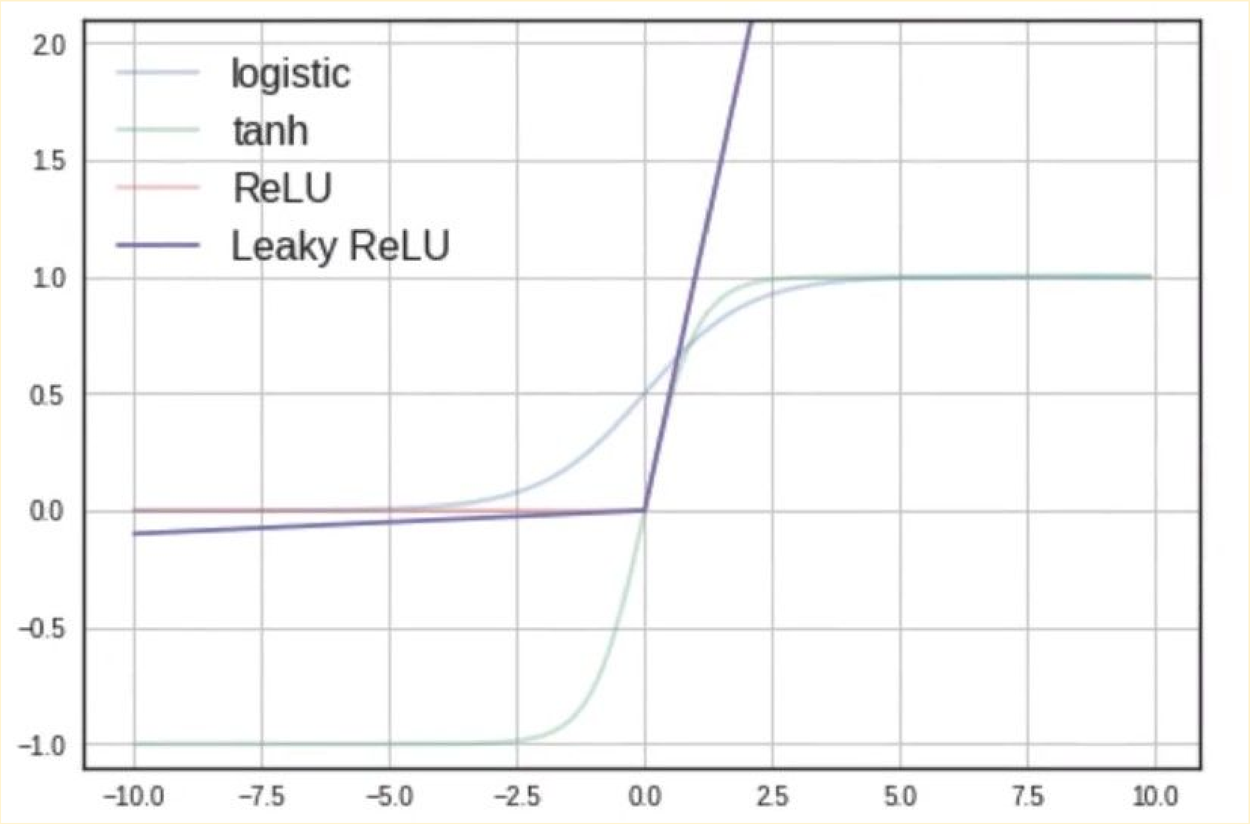
   a. $h_1 = ReLU(a_1) = max(0, a_1) = max(0, w_1x_1 + w_2x_2 + b)$
   b. What happens if b takes on a large negative value due to a large negative update ∇b at some point?
   c. $w_1x_1 + w_2x_2 + b < 0$ [if b << 0]
   d. Therefore $h_1 = 0$ [dead neuron]
   e. Which means $\frac{\partial h_1}{\partial a_1} = 0$
   f. This zero derivative is involved in the chain rule for computing the gradient w.r.t ∇w_1
   g. ∇w_1 *becomes* 0 leading to the weight not being updated, as in the case of a **saturated neuron**.
   h. This also applies to ∇w_2 and ∇b, their parameters are not updated.
   i. Here, x_1 and x_2 have been normalised, so they range between 0-1 and are therefore unable to counterbalance any large negative value b
   j. This means that once a neuron has died, it remains dead forever, as no new input would be large enough to counter the negative b value
   k. Thus, there is a very real problem of saturation of a ReLU neuron in the negative region.
   l. In practice, if there is a large number of ReLU neurons, a large fraction (up to 50%) may die during operation if the learning rate is set too high
   m. It is advised to initialise the bias to a positive value
   n. Using other variants of ReLU is recommended
2. A good alternative is the **Leaky ReLU**

PadhAI: Activation Functions & Initialization Methods

One Fourth Labs

---

a. The following figure illustrates the leaky ReLU function



b. $f(x) = max(0.01x, x)$

c. $f'(x) = \frac{\partial f(x)}{\partial x} = 0.01 \; if \; x < 0 \mid 1 \; if \; x > 0$

d. ReLU outputs the input value itself if it is positive, else it outputs a fraction of the input value, i.e. f(2) = 2, f(-2) = 0.02

e. It does not saturate in the positive or negative region

f. Will not die (0.01x ensures that at least a small gradient will flow through), this means that there isn't any 0 valued derivative, thereby ensuring that the gradients are all non-zero. Thus, the weights are always updated.

g. It is easy to compute (no expensive e^x)

h. Close to zero centered outputs