Statistical Machine Learning
Activation Functions

Horacio Gómez-Acevedo
Department of Biomedical Informatics
University of Arkansas for Medical Sciences

March 14, 2022

What are activation functions?

Recall that in the threshold logic, once we receive the input, a decision must be made to *fire* or *suppress* the output.
Activation functions generalize the concept of activation given a specific input.

Linear Functions

These functions are used to model the firing rate of a neuron.

$$f(x) = kx \quad k \text{ is a positive constant}$$

Pros: It is continuous and differentiable.

Step Functions

These are biologically inspired type of activation.

1. Heaviside function (threshold step function). This function fires only for positive values.

$$H(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

This function is differentiable except in $x = 0$. Informally, people refers to the Dirac delta function as its *derivative*.

$$H'(x) = \delta(x) = \begin{cases} 0 & \text{if } x \neq 0, \\ \infty & \text{if } x = 0 \end{cases} \quad \text{it also satisfies} \int_{-\infty}^{\infty} \delta(x)dx = 1$$

Step Functions (cont)

2. Signum Function. This is a variation of the Heaviside function

$$S(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

Hockey stick functions

1. Rectified Linear Unit (ReLu). The activation is linear for $x \geq 0$.

$$ReLU(x) = xH(x) = \max\{x, 0\} = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases}$$

Pros:

- This function does not saturate (see below with sigmoid functions)
- Neural networks with ReLU activation functions tend to learn several times faster than similar networks with saturating activation function.

Hockey stick functions

2. Parametric Rectified Linear Unit (PReLUI). In this case the activation is piecewise linear, having different firing rates for $x < 0$ and $x > 0$

$$PReLU(\alpha, x) = \begin{cases} \alpha x & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \quad \alpha > 0.$$

Sigmoid functions

These types of activation functions have that advantage that they are smooth and can approximate a step function to any degree of accuracy.

1. Logistic function with parameter $c > 0$.

$$\sigma_c(x) = \frac{1}{1 + \exp(-cx)}$$

   The parameter $c > 0$ controls the firing rate of the neuron. Large values of c correspond to a fast change of values from 0 to 1.

Sigmoid Functions

1. Hyperbolic tangent.

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Sigmoid Functions

1. Arctangent function.

$$h(x) = \frac{2}{\pi} \arctan(x)$$

Cost Functions

In the learning process, the parameters of a neural network are subject to minimize a certain objective function which represents a measure of proximity between the prediction of the network and the associated target.
Note that these functions are also called *error functions* or *loss functions*.

Input, Output and Target

The **input** of a neural networks is obtained from given data or from sensors that perceive the environment. The input is a variable that is fed into the network. It can be one-dimensional variable x, or a vector $\mathbf{x} \in \mathbb{R}^n$, a matrix, a tensor, or a random variable X. The network acts as a function and provides an **output** which can be one-dimensional $y \in \mathbb{R}$, or a vector $\mathbf{y} \in \mathbb{R}^n$, a matrix, a tensor, or a random variable Y. We normally denote the **input-output** mapping by $f_{w,b}$. With the previous notations $f_{w,b}(x) = y$, $f_{w,b}(\mathbf{x}) = \mathbf{y}$, and $f_{w,b}(X) = Y$.
The **target function** is the desired relations which the network tries to approximate. This function is independent of the parameters w and will be denoted by $z = \phi(x)$, $\mathbf{z} = \phi(\mathbf{x})$ or $Z = \phi(X)$.

Goal

The neural network will tune the parameters (w, b) until the output variable y will be in a proximity of the target variable z. The proximity will be determined by the cost function $C(w, b) = \text{distance}(y, z)$ The optimal parameters of the network are given by

$$(w^*, b^*) = \text{arg min}_{w,b} C(w, b)$$

The process by which the parameters (w, b) are tuned into (w^*, b^*) is called **learning**.

Supremum Cost Function

Let's assume that a neural network takes inputs in $x \in [0, 1]$, is supposed to learn a given continuous function $\phi\colon [0, 1] \to \mathbb{R}$. If $f_{w,b}$ is the input-output mapping of the network the associated cost function is

$$C(w, b) = \sup_{x \in [0,1]} |f_{w,b}(x) - \phi(x)|$$

For all practical purposes, when the target function is known at n points

$$z_1 = \phi(x_1), z_2 = \phi(x_2), \dots, z_n = \phi(x_n)$$

then, the supremum cost function becomes

$$C(w, b) = \max_{1 \le i \le n} |f_{w,b}(x_i) - z_i|$$