The S function and its geophysical applications

Brian Russell[1]

[1] Hampson-Russell, A CGG Company, Calgary, Alberta, brian.russell@cgg.com

ABSTRACT

In a previous CREWES talk, which was also published in The Leading Edge (Russell et al., 2002), we applied the multi-layer perceptron (MLP) neural network to the solution of a straightforward AVO classification problem. In that paper, we pointed out that the basis of the MLP is the sigmoidal, or *S*, function that is used in the training. The *S* function is one of the most useful functions in mathematics and physics, and has many applications besides its use in neural networks. The *S* function comes in various guises, such as the logistic function, the tanh function, the Cole-Cole equation, the population function, etc. In this article I will first describe the function itself and its origin and derivation. I will then discuss the various applications of this function, many of which I have talked about in several previous CREWES talks, notably the talk on solving an AVO problem using a neural network. These applications range from population growth through paramagnetism and heavy oil modeling to neural networks, making the *S* function one of the most ubiquitous functions used in geophysics.

INTRODUCTION

This article will present an overview of the many applications of the sigmoidal, or *S*, function. I will start by deriving the simplest form of the *S* function, the logistic function, using population dynamics. I will then show a related function, the hyperbolic tangent, and give several examples of its use. The first is from statistical physics and involves both paramagnetism and ferrimagnetism. The second example is from the modeling of heavy oil viscosity using the Cole-Cole equation.

I will finish with a discussion of a neural network can solve a simple AVO problem. In doing so, I will shed light on several important questions: why are some neural networks only able to solve linear problems, whereas others can solve nonlinear problems, and how can neural networks be trained to do these tasks? The AVO problem that we will train the networks to solve is the recognition of a class 3 anomaly on an AVO attribute crossplot. As we shall see, training a computer to perform tasks that are simple for a human being (that is, an interpreter) can often be quite difficult. The fundamental concept of a neural network is that a number of input values are passed through a set of linear weights, in a similar fashion to multi-linear regression. However, what makes the neural network unique is that after the weights are applied, a nonlinear activation function is also applied. Although many functions could be used as the nonlinear function, the *S* function is the most popular one, due to its ability to turn the input "on" and "off" and also the unique property of its derivative, which makes the training phase of the neural network computationally tractable.

THE LOGISTIC EQUATION AND POPULATION DYNAMICS

The S-function first arose in two separate problems which seem unrelated: population dynamics and the statistical physics of paramagnetism. Although the latter problem came earlier, I will start by looking at population dynamics. You will recall that the model for exponential growth comes from the differential equation given by

$$\frac{dx(t)}{dt} = ax(t), \tag{1}$$

with solution:

$$x(t) = ke^{at}. \tag{2}$$

Equation 2 gives us a way to measure population growth or decline, and is also used for radiocarbon dating. As shown in Figure 1, for radiocarbon dating the negative exponent (a = -1) lets the curve decay to zero, as we would like. But for population growth, the positive exponent (a = -1) makes it climb to infinity.

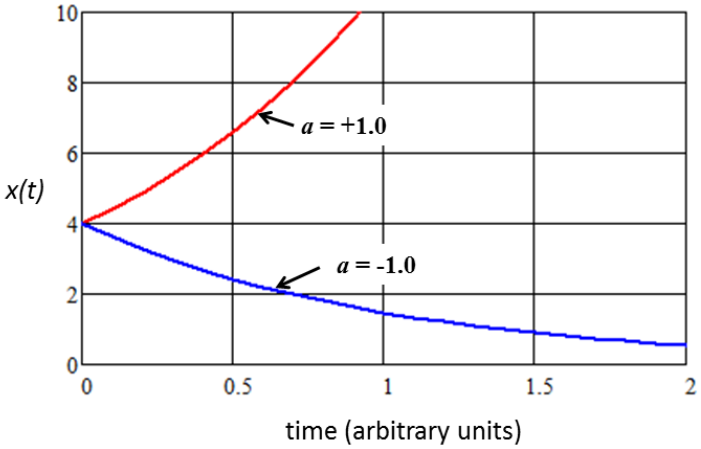


FIG. 1. The computation of the population growth model given in equation (2) for a value of k = 4 and both positive and negative values of a. Negative growth using this model fits radiocarbon dating quite well but positive growth with the exponential model is unrealistic because it rapidly goes to infinity.

Thus, to create a stable population model researchers developed the logistic growth equation (Strogatz, 1994), given by

$$\frac{dx(t)}{dt} = ax(t)(1 - x(t)). \tag{3}$$

The solution to equation (3) is the logistic equation, written in one of the two following forms,

$$x(t) = \frac{ke^{at}}{1 + ke^{at}} = \frac{1}{1 + ce^{-at}}, \tag{4}$$

where $c = 1/k$. Note that for $k = 1$, equation 4 has limits given by $x(-\infty) = 0$, $x(0) = 0.5$ and $x(\infty) = 1$. Figure 2 shows a graph of the logistic function for three different values of a, where $k = 1$. Note that the curve displays a sigmoidal, or S-type shape, hence the name of this function. As the value of a decreases, the logisitic flattens out and, conversely, as the value of a increases the logistic function approaches a step function which goes to zero for t values less than zero and to one for t values greater than zero. We will show how the step function version of the logistic function is useful for predicting the effects of neural networks later in this article.

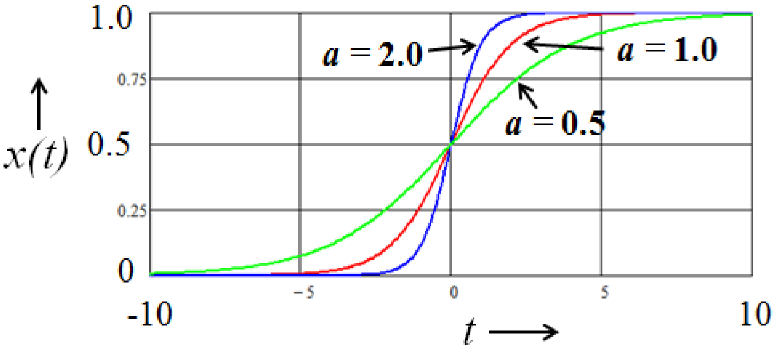


FIG. 2. A graph of the logistic function (equation 4) for various positive values of a, with the value of k =1.

Figure 3 shows the effect of k on the logistic function, with the value of a =1. For k = 1, which is the normal default and the value used in Figure 2, the curve goes through 0.5 at t = 0. When $k > 1$, the curve slides up the vertical axis (or, equivalently, left on the horizontal axis). When $k < 1$, the curve slides down the vertical axis (or, equivalently, right on the horizontal axis).

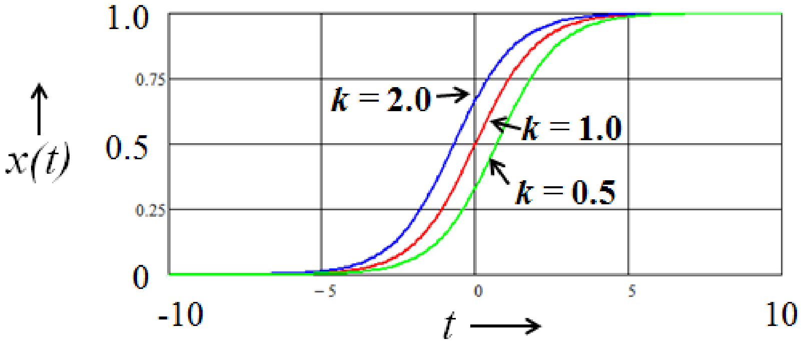


FIG. 3. A graph of the logistic function (equation 4) for various values of k, with the value of a =1.

From a population point of view the logistic function tells us that most populations will flatten off to a stable value after some time. Note that we would start at zero time, so only the upward part of the curve in Figure 3 would be displayed. Also, the value of k would have to be greater than one. The amount of time that it takes to flatten off with depend on the value of a. An example of this would be the population of some prey species like deer or rabbits, which would grow dramatically until some predator species, like wolves or coyotes, discover them. As the predator species increases by the abundance of prey, the death rate of the prey goes up until the two species strike a symbiotic balance.

THE TANH FUNCTION AND STATISTICAL PHYSICS

Another type of sigmoidal function is the hyperbolic tangent function, tanh(t), defined as

$$\tanh(at) = \frac{e^{at} - e^{-at}}{e^{at} + e^{-at}}. \tag{5}$$

Note that equation 5 has limits given by $x(-\infty) = -1$, $x(0) = 0.0$ and $x(\infty) = 1$, unlike the logistic function, which goes from 0 to 1. Figure 4 shows a plot of the tanh function for various values of a, and the shape looks very much like the logistic equation.

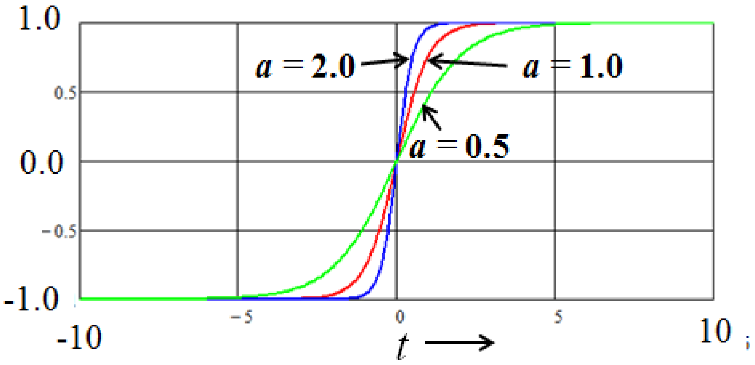


FIG. 4. A graph of the hyperbolic tangent function (equation 5) for three different values of a. Note that this function is similar to the logistic function of equation (4) and Figure 2 except that its limits are -1 and +1 rather than 0 and 1.

Indeed, we can show mathematically (by multiplying equation 5 by e^{-at}/e^{-at} and then simplifying) that the hyperbolic tangent function can be written as

$$\tanh(at) = 2\left[\frac{1}{1+e^{-2at}}\right] - 1. \tag{6}$$

That is, the hyperbolic tangent function is simply the logistic function with both its exponent and overall value multiplied by 2 and then shifted down by a value of 1. This is intuitively obvious by comparing Figures 4 and 2, since the hyperbolic tangent has a steeper slope for the same value of *a* and also goes from -1 to +1 instead of from 0 to 1, as noted earlier.

But where do we find the hyperbolic tangent in mathematics or physics? The tanh function is found in many statistical physics applications (Reif, 1965). For example, for a paramagnetic substance whose particles have two ½ spin states (+/-) aligned parallel and antiparallel to a magnetic field B, their Boltzmann probabilities

are $P_+ = C\exp[+\beta\mu B]$ and $P_- = C\exp[-\beta\mu B]$, where β = $(kT)^{-1}$, μ = the magnetic moment, k = the Boltzmann constant and T = temperature. This gives us a mean magnetic moment of

$$\overline{\mu}_B = \frac{\mu P_+ + (-\mu)P_-}{P_+ + P_-} = \mu\frac{\exp[+\beta\mu B] - \exp[-\beta\mu B]}{\exp[+\beta\mu B] - \exp[-\beta\mu B]} = \mu\tanh\left(\frac{\mu B}{kT}\right). \quad (7)$$

Thus, as shown in equation 7, paramagnetic materials display a behavior governed by the hyperbolic tangent function.

Ferrimagnetic materials also display similar behavior. As shown in Figure 6, from Ali and Potter (2012), a clean sandstone containing ferrimagnetic minerals shows a hysteresis curve that clearly follows a hyperbolic tangent dependence at a range of different temperatures.

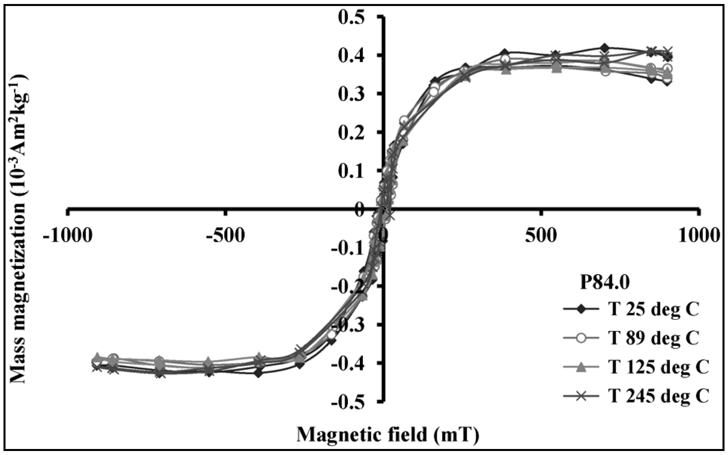


FIG. 5. The extracted ferrimagnetic hysteresis curves at varioustemperatures for sample P84.0 (from Ali and Potter, 2012).

THE COLE-COLE EQUATION IN HEAVY OIL MODELLING

As a further example of the hyperbolic tangent function found in physics, Batzle et al. (2006), Behura et al. (2007) and Gurevich et al. (2008) use the Cole-Cole equation to model the complex shear modulus of heavy oil as a function of frequency and viscosity. The Cole-Cole equation (Cole and Cole, 1941) is written:

$$\mu(\omega) = \mu_0 + \frac{\mu_\infty - \mu_0}{1 + (i\omega\tau)^{-\alpha}}, \quad (8)$$

$\tau = \dfrac{\eta}{\mu_\infty - \mu_0}$ = relaxation time, $i = \sqrt{-1}$, ω = angular frequency, η = viscosity, μ_∞ is the real shear modulus at infinite frequency, μ_0 is the real shear modulus at zero frequency and α is an experimentally determined factor between 0 and 1. Although the Cole-Cole equation does not look like an S function, we can show that the real component of the shear modulus can be written as

$$\mu_{\mathrm{Re}} = \mu_0 + \frac{1}{2}(\mu_\infty - \mu_0)\left\{1 + \frac{\sinh[\alpha \ln(\omega\tau)]}{\cosh[\alpha \ln(\omega\tau)] + \cos(\alpha\pi/2)}\right\}. \tag{9}$$

Note that if $\alpha = 1$ in equation 9, the cosine term equals zero and we therefore get the following function

$$\mu_{\mathrm{Re}} = \mu_0 + \frac{1}{2}(\mu_\infty - \mu_0)\{1 + \tanh[\alpha \ln(\omega\tau)]\}, \tag{10}$$

which is a shifted hyperbolic tangent function that is similar to the logistic function. The Cole-Cole equation therefore produces a result that goes from μ_0 to μ_∞ in a sigmoidal way for the real shear. The position on the curve depends on the value of α ln(ωt), which in turn is a function of frequency. This gives the result shown schematically in Figure 6.

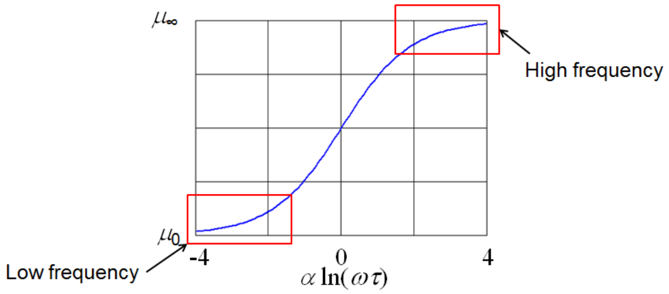


FIG. 6. A schematic interpretation of equation 10 for the behavious of the real shear modulus μ as a function of frequency in heavy oil.

Figure 7, from Batzle et al. (2006) shows how well equation 10 fits data from a measured heavy oil at 20$^\mathrm{o}$ C.

.

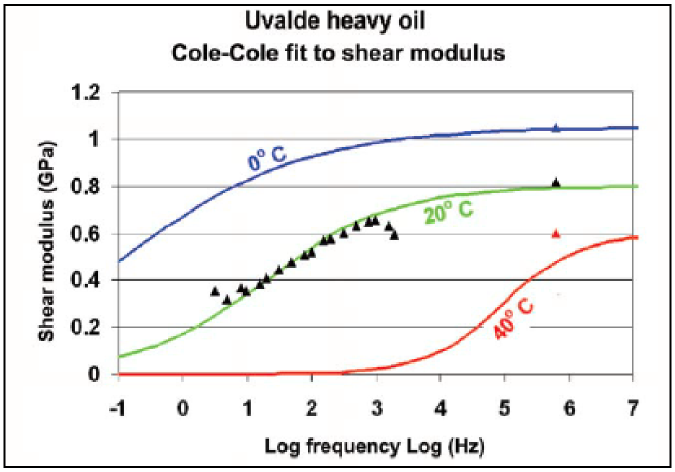


FIG. 7. Measured (triangles) and calculated (lines) shear modulus, μ, in Uvalde heavy oil (API = -5) using a viscoelastic liquid (Cole-Cole) model (Batzle et al., 2006).

The high-frequency shear modulus in Figure 7 comes from ultrasonic measurements of heavy oil from the Uvalde formation in Texas.

THE *S* FUNCTION, NEURAL NETWORKS AND AVO

In this last section I will show how the *S* function is used in the training of a neural network. Specifically, I will apply the neural network to an AVO problem that would be straightforward for a human interpreter to solve, but is difficult for a computer to solve. The basic AVO interpretation problem that we will study is differentiating between the AVO responses of the two reservoirs shown in Figure 8. Figure 8(a) shows a wet sand encased between two shale layers, and Figure 8(b) shows a gas sand encased between the same two shales. The P-wave velocity (V_P), S-wave velocity (V_S), and density (ρ) for each layer are shown in each figure.

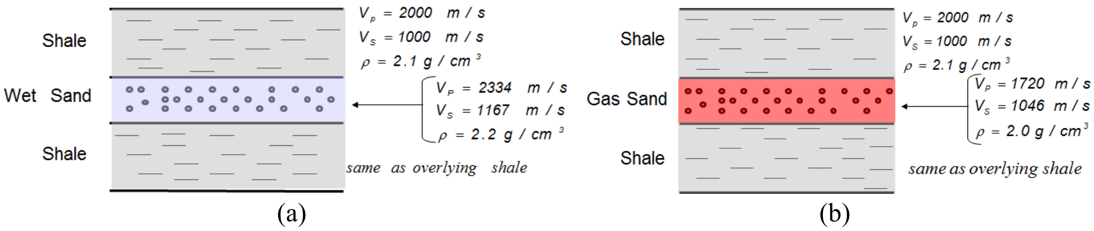


FIG. 8. Two simple geological models where (a) shows a wet sand between two shale layers and (b) shows a gas sand between the same two shales.

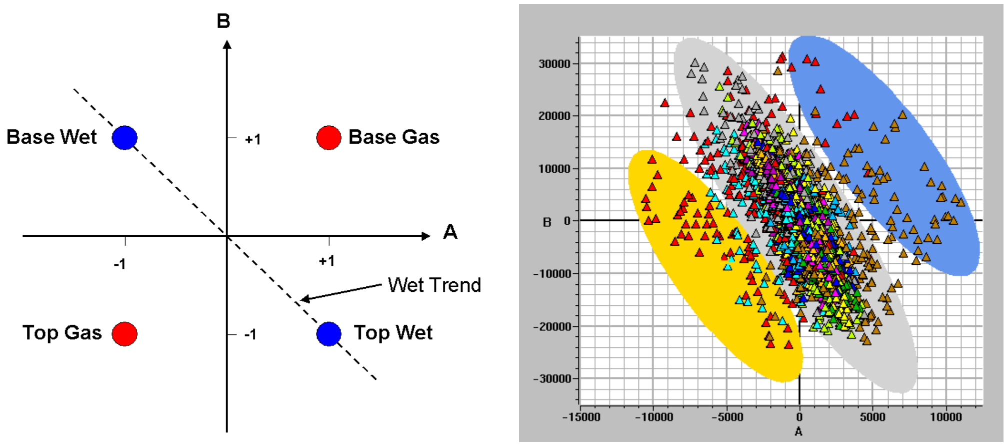
We will assume that the far angle of incidence is small enough (i.e. approximately 30°) that we can ignore the third term in the Aki-Richards equation (see Appendix A) and write the reflectivity as a function of angle of incidence θ as

$$R(\theta) = A + B\sin^2\theta\,, \tag{11}$$

where A is the AVO intercept, and B is the AVO gradient. Using the values for V_P, V_S, and density ρ shown in Figure 8, we can compute the values for the AVO intercept and gradient for the wet and gas sands. Let's start with the wet sand, noting that the V_P/V_S ratio in both the sand and shale layer is equal to 2. As shown in Appendix A, this leads to the simplification that $B = -A$ for both the top and base of the layer. Using the parameters shown in Figure 8a gives $A_{TOP\_WET} = B_{BASE\_WET} = +0.1$ and $A_{BASE\_WET} = B_{TOP\_WET}$ = -0.1. For the gas sand of Figure 8b, the V_P/V_S ratio is equal to 1.65, and the intercept does not simplify as it did the wet sand. However, the calculation is still straightforward, and leads to $A_{TOP\_GAS} = B_{TOP\_GAS}$ = -0.1 and $A_{BASE\_GAS} = B_{BASE\_GAS}$ = +0.1. Note that, for the gas case, A=B for both the top and base of the layer.

After scaling each of the values of A and B by a factor of 10 (to give values of +1 and -1) they have been put on an A-B crossplot, as shown in Figure 9(a). In our example, the wet points (shown as solid blue circles) establish the wet sand-shale trend, and the top and base gas (shown as solid red circles) plot in the other two quadrants of the A-B crossplot. This is a typical class 3 AVO anomaly (Rutherford and Williams, 1989).

Despite the simplicity of the model, the plot shows us what is expected in a noise-free AVO crossplot. For comparison, Figure 9(b) is an interpreted AVO A-B crossplot for a class 3 AVO response in the Gulf of Mexico (Ross, 2000). The centre grey ellipse encompasses all of the wet sand-shale AVO points while the gold and blue ellipses outlying the grey "wet trend" points are associated with the top and base of the pay sand, respectively. Identifying the wet trend and the outlying two points in Figure9(a) is a trivial problem for the eye to interpret. However, as we will see in the next section, the early neural networks could not properly address situations as simple as this, and technology improvements (often through trial and error) were required to eventually solve the problem.



(a) (b)

FIG. 9. Intercept versus gradient crossplots, where (a) shows the crossplot of the A and B values from the wet and gas models of Figure 8, crossplotted after being scaled by a factor of 10, and (b) shows a Gulf of Mexico real data example, where the grey ellipse shows the "wet trend" and the gold and blue ellipses show the top and base of the pay sand, respectively.

THE PERCEPTRON

The classic model of the neuron is called the perceptron (McCulloch and Pitts, 1943) and is illustrated in Figure 10. The perceptron accepts N inputs $a_1, a_2, \ldots, a_N$, and produces a single output. Mathematically, the perceptron has two separate stages. First, the inputs are weighted and summed according to the equation:

$$x = w_1 a_1 + w_2 a_2 + \ldots + w_N a_N + b \tag{12}$$

The last weight, b, is called the bias, and is often written as w_0. Next, a threshold function f is applied to the intermediate output x to product the final output y, or

$$y = f(x) \tag{13}$$

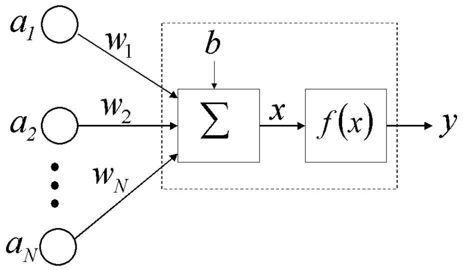


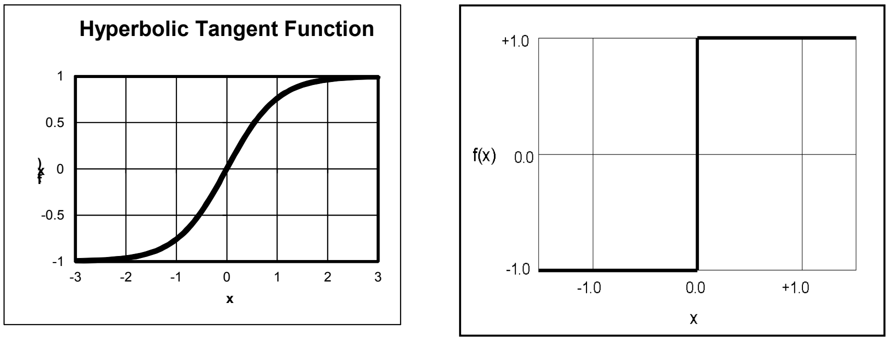
FIG. 10. The figure above shows the perceptron neural network for *N* inputs and a single output. The two boxes enclosed by the dashed line constitute the perceptron.

The choice of the threshold function *f* is important and depends on the problem being solved. If *f(x) = x*, the perceptron reduces to a linear sum of the inputs. In many applications, *f(x)* is set to the smoothly varying sigmoidal function such as the hyperbolic tangent function, which was defined in equation 5. A graph of the hyperbolic tangent function is shown in Figure 11(a).

For a two class problem, such as the one we are discussing here, we often use the step function, which is given mathematically by the equation:

$$f(x)=\begin{cases}+1, & x \geq 0\\ -1, & x<0\end{cases} \tag{14}$$

A graph of the symmetric step function is shown in Figure 11(b). Note that the symmetric step function can be thought of as the tanh function as *a* approaches infinity.



(a) (b)

FIG. 11. This figure shows a graph of (a) the hyperbolic tangent function of equation 5, and (b) the symmetric step function of equation 14.

Now, let's consider how to adapt the perceptron to our AVO problem, as shown by the neural network graph in Figure 12. We have reduced the problem to two inputs, the intercept (*A)* and gradient (*B*), and are using is the symmetric step function to compute the final output. The step function is a logical and convenient choice because A and B have scaled values of ±1. However, the interpretation of the output values will be different than the input. A value of +1 will indicate the presence of a gas sand and a value of –1 will indicate the presence of a wet sand.

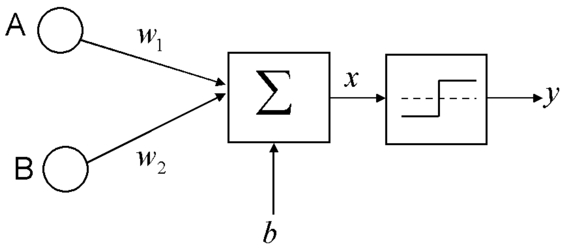


FIG. 12. The perceptron adapted to the AVO problem of Figure 3, where the inputs are the intercept (A) and gradient (B) and the function shown schematically is the symmetrical step function.

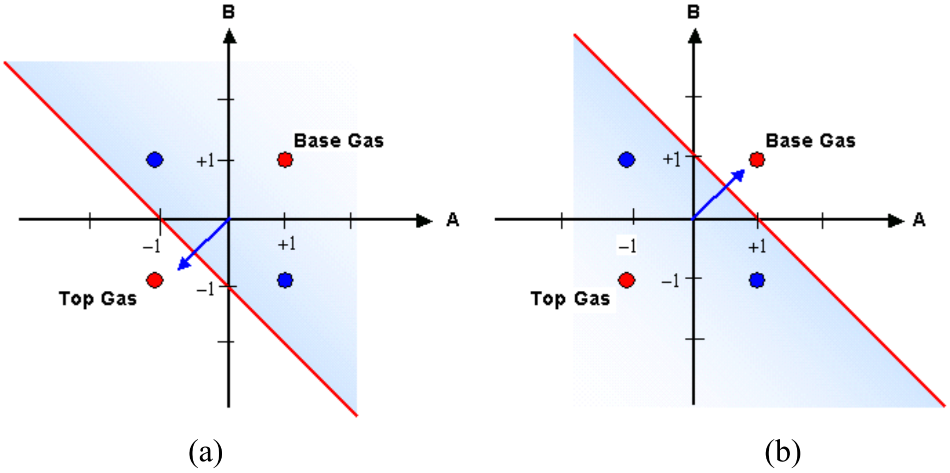
Notice that the equation for intermediate output x is now given as

$$x = w_1 A + w_2 B + b \tag{15}$$

The key question is: how do we determine the weights w_1, w_2, and b? To answer this, let us take an intuitive look at what these weights mean. From equation (6), it is obvious we are interested in the separation between $x < 0$ and $x > 0$, which occurs when $x = 0$. This is called the decision boundary. To find out where this boundary crosses the A and B axes, we simply need to successively set A and B to zero, to give

$$B = \frac{-b}{w_2}, \text{ and } A = \frac{-b}{w_1}.$$

So, let's now revisit the AVO problem of Figure 9(a), as shown in Figure 13.



(a) (b)

FIG. 13. The AVO problem from Figure 2 with decision boundaries, where (a) shows separation of the base of the gas sand and (b) shows separation of the top of the gas sand.

First of all, notice that we will not be able to separate both the top and base of the gas sand from the wet trend with a single decision plane, since this is a nonlinear separation problem. Thus, we must choose to separate either the top of the gas sand, as shown in

Figure 13(a), or the base of the gas sand, as shown in Figure 13(b). To solve for the weights for the top of the gas sand, notice that

$$A = B = -\frac{b}{w_1} = -\frac{b}{w_2} = -1 .$$

Although b, w_1 and w_2 can be scaled by any value, it is best to choose the simplest values, which are $w_1 = w_2 = -1$, and therefore $b = -1$. The perceptron diagram for this is shown in Figure 14(a).

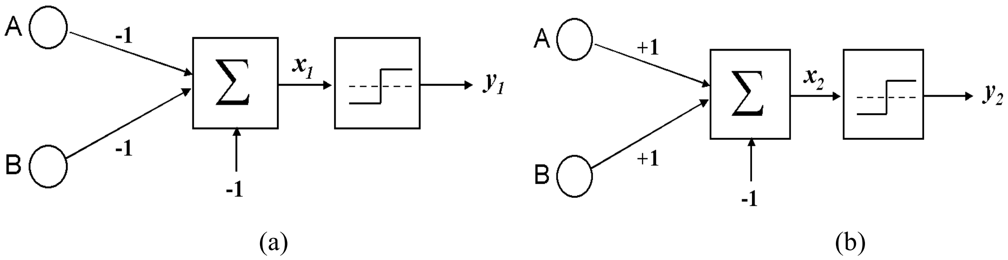


FIG. 14. Perceptron implementations of separating (a) the top of gas (Perceptron 1), and (b) the base of gas (Perceptron 2).

Next, to solve for the weights for the base of gas sand, notice that:

$$A = B = -\frac{b}{w_1} = -\frac{b}{w_2} = +1$$

Therefore, the weights are $w_1 = w_2 = +1$ and $b = -1$. This is shown in Figure 14(b).

Table 1 shows that these values do indeed solve the problem for the four possible cases. Although we have solved for the top and bottom of the gas sand individually we have still not solved the complete problem, which is to separate the gas sand responses from the wet sand responses. This requires a multi-layer perceptron.

Table 1. Outputs from the perceptron models of Figure 9.

Inputs			**Perceptron 1**		**Perceptron 2**	
Sand	**A**	**B**	x_1	y_1	x_2	y_2
Top Gas	-1	-1	+1	+1	-3	-1
Base Wet	-1	+1	-1	-1	-1	-1
Top Wet	+1	-1	-1	-1	-1	-1
Base Gas	+1	+1	-3	-1	+1	+1

THE MULTI-LAYER PERCEPTRON

The problem that we encountered in the last section, that perceptrons can only solve linearly separable problems, was also encountered by the early neural network researchers. The solution is to add a second layer of perceptrons. In Figure 15 we have shown a two-layer perceptron with *N* inputs into *M* perceptrons or neurons. The sum and function boxes of our earlier figures have been replaced by a single box, but that the perceptrons still has the same internal workings as previously described. Notice that the first set of weights now have two subscripts and a superscript, written as $w^{(k)}{}_{ij}$, where i represents the perceptron number, j represents the input number and the superscript k in brackets indicates the layer number.

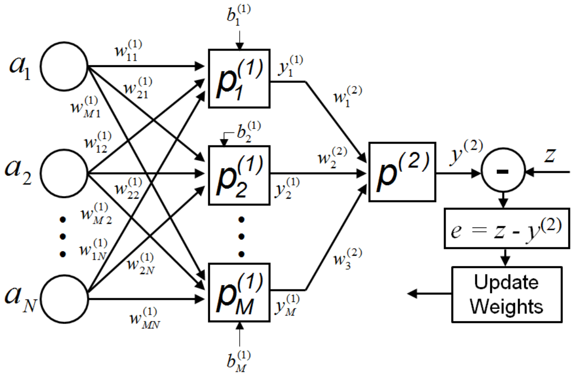


FIG. 15. A multi-layer perceptron with *N* inputs, *M* perceptrons, and a single output.

In Figure 15, the the input consists of the N attributes a_i and the output ($y^{(2)}$, where the superscript 2 refers to the second layer of weights) is compared by subtraction to the desired output value z. The error is then used to update the weights

using a technique called backpropagation, which will be discussed later. Mathematically, the two-layer perceptron can be written as the following matrix equation.

$$\boldsymbol{y}^{(2)} = f^{(2)}(W^{(2)} f^{(1)}(W^{(1)}\boldsymbol{a} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}), \tag{16}$$

where

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}, \; W^{(j)} = \begin{bmatrix} w_{11}^{(j)} & w_{12}^{(j)} & \cdots & w_{1N}^{(j)} \\ w_{21}^{(j)} & w_{22}^{(j)} & \cdots & w_{2N}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M^{(j)}1}^{(j)} & w_{M^{(j)}2}^{(j)} & \cdots & w_{M^{(j)}N}^{(j)} \end{bmatrix}, \boldsymbol{b}^{(j)} = \begin{bmatrix} b_1^{(j)} \\ b_2^{(j)} \\ \vdots \\ b_{M^{(j)}}^{(j)} \end{bmatrix},$$

and the superscript refers to the layer number. Note that the above scheme allows us to have different functions in each layer, but we usually use the same function.

To understand why the multi-layer perceptron can solve our AVO problem, we need to recast it as as a multi-layer perceptron, as shown in Figure 16. In this neural network, the inputs are still the intercept (A) and gradient (B), but now they are both interconnected, via the weights, to the two perceptrons.

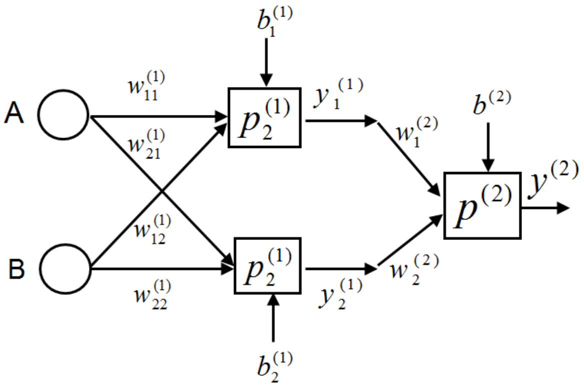


FIG. 16. The multi-layer perceptron for the gas-water sand model of Figure 9.

Thus, in a multi-layer network we simply connect two perceptrons to produce two separate outputs. These outputs now represent the input to a new perceptron. If we use the two perceptrons from Figure 14 as our two first layer perceptrons, we can crossplot their outputs $y_1^{(1)}$ and $y_2^{(1)}$, and design a new perceptron, $p^{(2)}$, to separate these outputs. This is shown in Figure 17, using the output values from Table 1. The base gas and top gas points have moved to what was the wet trend on the input points, but both of the wet sands have moved to the point (-1, -1), which was the top gas point for the input. This new linearly separable problem is graphically shown by the decision boundary on Figure 17. Thus, we can derive the weights for perceptron $p^{(2)}$ in Figure 12, which are:

$$-\frac{b^{(2)}}{w_1^{(2)}} = -\frac{b^{(2)}}{w_2^{(2)}} = -1, \text{ or } w_1^{(2)} = w_2^{(2)} = b^{(2)} = +1.$$

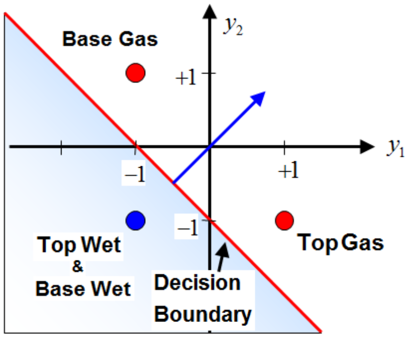


FIG. 17. A crossplot of the outputs of the two perceptrons of Figure 9.

The final perceptron, with weights, is shown in Figure 18. Notice that the weights are all equal to +1 or –1, which are the simplest set of weights but that there are many other sets of weights that would solve the problem.

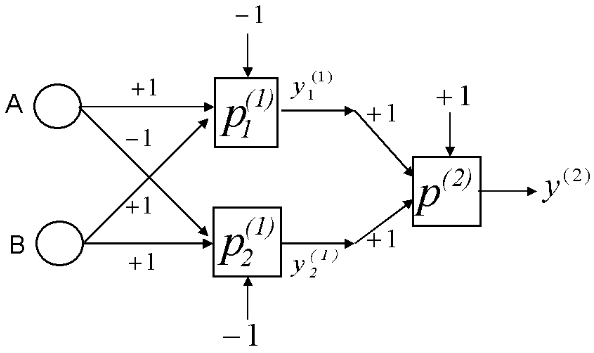


FIG. 18. The final multi-layer perceptron weights.

To verify that this indeed is the solution, Table 2 then shows the outputs for all possible inputs. As we can see in Table 2, the correct output is given for all four input cases. Thus, we have solved the problem of how to separate a simple Class 3 gas sand from its equivalent wet sand.

Table 2. The computed output values from the multi-layer perceptron in Figure 13 for the four input models of Figure 3, where $y^{(2)}$ shows that the gas sand values (+1) have been separated from the wet sand values (-1).

	Inputs		**Perceptron** $p_1^{(1)}$	**Perceptron** $p_2^{(1)}$	**Perceptron** $p^{(2)}$
Sand	**A**	**B**	$y_1^{(1)}$	$y_2^{(1)}$	$y^{(2)}$
Top Gas	-1	-1	+1	-1	+1
Base Wet	-1	+1	-1	-1	-1
Top Wet	+1	-1	-1	-1	-1
Base Gas	+1	+1	-1	+1	+1

Finally, let us see how a real neural network would solve this problem, which will also shed light on why the *S* function is such a good choice for the nonlinear function used in the multi-linear perceptron. Figure 19 again shows the two layer perceptron neural network for the AVO classification problem but now shows the error term that was shown in Figure 15, and indicates that we need some way of updating the weights based on the error.

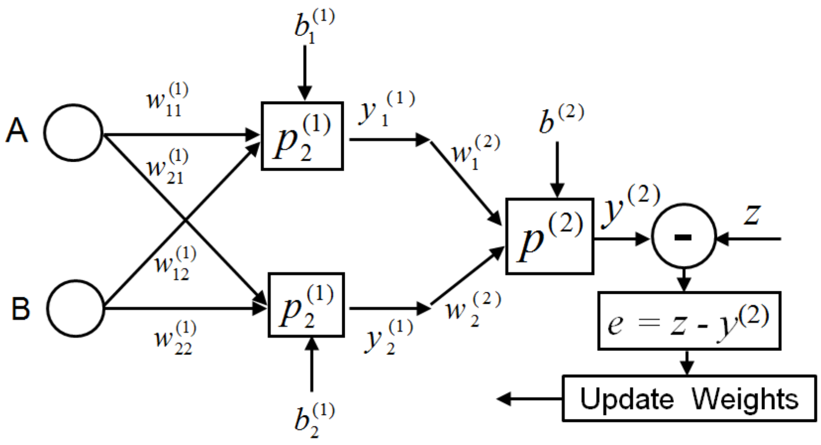


FIG. 19. The two layer perceptron for the AVO classification problem where we have indicated that the weights are to be updated by backpropagation of the error.

The method for doing this was developed by McClelland and Rumelhart (1981) and is called error backpropagation. The idea is to minimize the squared error between the computed and desired outputs, which can be written as a function of the weights as

$$E(\boldsymbol{w}) = \frac{1}{2}(z - y^{(2)})^2 , \tag{17}$$

by updating the weight values iteratively. To do this, we differentiate the error term with respect to the weights, giving

$$\Delta \boldsymbol{w} = -\eta \frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} , \tag{18}$$

where η is a scaling value between 0 and 1. The weights are updated iteratively by the equation

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \Delta \boldsymbol{w}(n) . \tag{19}$$

The details of this calculation are given in Russell (2002) but the final solution for the second set of weights is given by:

$$\Delta w_{kj}^{(2)} = \eta \delta_j^{(2)} y_{kj}^{(1)} , \tag{20}$$

and for the first set of weights is given by

$$\Delta w_{ki}^{(1)} = \eta \delta_j^{(2)} w_{kj}^{(2)} f'(x_{kj}^{(1)}) a_{ij} , \tag{21}$$

where $\delta_j^{(2)} = (z_j - y_j^{(2)}) f'(x_j^{(2)})$. Notice that both of equations 20 and 21 require the computation of the derivative of the nonlinear function used in the neural network, which in our case is the tanh function. It can easily be shown that the derivatives of the hyperbolic tangent is given by

$$\frac{d\tanh(x)}{dx} = 1 - \tanh^2(x) \Rightarrow f'(x) = 1 - f(x)^2 . \qquad (22)$$

The simplicity of this derivative therefore makes the tanh function a very good choice for the neural network. To start the process, an estimate of the initial layer weights is done using random numbers. In our case, the values for the first and second layer were

$$W_{(0)}^{(1)} = \begin{bmatrix} b_1^{(1)} & b_2^{(1)} \\ w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.9501 & 0.4860 \\ 0.2311 & 0.8913 \\ 0.6068 & 0.7621 \end{bmatrix}, \text{ and } \boldsymbol{w}_{(0)}^{(2)} = \begin{bmatrix} b^{(2)} \\ w_1^{(2)} \\ w_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0.4565 \\ 0.0185 \\ 0.8214 \end{bmatrix}.$$

This gives an output value of

$$\boldsymbol{y}_{(0)}^{(2)} = [-0.2145 \quad 0.7240 \quad 0.6374 \quad 0.8546],$$

with an RMS error of E = 2.6738.

We then ran 1000 iterations of the backpropagation algorithm with an η value of 0.2 and got weights for the two layers given by

$$W_{(1000)}^{(1)} = \begin{bmatrix} 1.5024 & -1.3779 \\ 1.6564 & 1.5539 \\ 1.6556 & 1.5534 \end{bmatrix}, \text{ and } \boldsymbol{w}_{(1000)}^{(2)} = \begin{bmatrix} 2.3006 \\ -2.5788 \\ 2.5885 \end{bmatrix},$$

which leads to the final computed values of

$$\boldsymbol{y}_{(1000)}^{(2)} = [0.9736 \quad -0.9806 \quad -0.9806 \quad 0.9734],$$

with an RMS error that has decreased to $E = 0.0464$. Although these values are not exact to our intuitively derived values, they are very close.

CONCLUSIONS

In this article I presented an overview of the many applications of the sigmoidal, or S, function. I started by deriving the simplest form of the S function, the logistic function, using population dynamics. I then showed a related function, the hyperbolic tangent, and give several examples of its use. The first example was
from statistical physics and involved both Paramagnetism and ferrimagnetism. The second example was from the modeling of heavy oil viscosity.

I then showed how neural networks can be used to solve a simple Class 3 AVO classification problem. The simplicity of the model allowed us to derive the weights by hand for two types of neural networks, the multi-layer perceptron (MLP) and radial basis function neural network (RBFN). In doing so, we gained insight into both types of neural networks. First, a single layer perceptron can only solve a linearly separable problem. However, by adding a second layer of weights and perceptrons, as in the MLP, we can transform a nonlinear problem into a linear problem, and thus find the solution using a single output perceptron. On the other hand, the RBFN does not require a second layer of weights, and applies a nonlinear function to the initial input to perform the linear separation. In both cases, our simple AVO problem proved to be an ideal model for understanding the inner workings of the neural networks in question.

In more complex problems, the weights for the MLP were derived using a technique called back propagation, in which we try to reduce the error between the known output and the output created by a set of initial weights by backward propagation of the errors. This method illustrated once again one of the many uses of the S function

REFERENCES

Ali, A. and Potter, D.K., 2012, Temperature dependence of the magnetic properties of reservoir rocks
and minerals and implications for in situ borehole predictions of petrophysical parameters: Geophysics, 77, p WA 211 – WA 221.

Batzle, M., R. Hofmann, and D. Han, 2006, Heavy oils - seismic properties: The Leading Edge, 25, 750–756.

Behura, J., M. Batzle, R. Hofmann, and J. Dorgan, 2007, Heavy oils: Their shear story: Geophysics, 72, no. 5, E175–E183.

Ciz, R., and Shapiro, S.A., 2007, Generalization of Gassmann equations for porous media saturated with a solid material: Geophysics, 72, no. 6, A75–A79.

Cole, K.S. and Cole, R.H. (1941), Dispersion and Absorption in Dielectrics – I: Alternating Current Characteristics: Journal of Chemical Physics 9 (4), 341–52.

Gurevich, B., K. Osypov, R. Ciz, and D. Makarynska, 2008, Modeling elastic wave velocities and attenuation in rocks saturated with heavy oil: Geophysics, 73, no. 4, E115–E122.

Han, D.-H., J. Liu, and M. Batzle, 2008, Seismic properties of heavy oils—Measured data: The Leading Edge, 27, 1108–1115.

McClelland, J.L. and Rumelhart, D. E., 1981, An interactive model of context effects in letter perception: part 1: An account of the basic findings, *Psychological Review* **88**: 375-407.

McCulloch, W. S. and Pitts, W., 1943, A logical calculus of the ideas immanent in nervous activity: *Bulletin of Mathematical Biophysics*, **5**, 115-133.

Reif, F., 1965, Fundamentals of Statistical and Thermal Physics: McGraw-Hill Book Company.

Ross, C. P., 2000, Effective AVO crossplot modeling: A tutorial: Geophysics, **65,** 700-711.

Russell, B., Ross, C., and Lines, L., 2002, Neural networks and AVO: The Leading Edge, Vol. 21, No. 3, pp. 268-314.

Strogatz, S.H., 1994, Nolinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry and Engineering: Westview Press.

APPENDIX A

THE AKI-RICHARDS EQUATION

The Aki-Richards equation is given by:

$$R(\theta) = A + B\sin^2\theta + C\sin^2\tan^2\theta, \tag{A-1}$$

where

$A = \frac{1}{2}\left(\frac{\Delta V_P}{V_P} + \frac{\Delta\rho}{\rho}\right)$ is the intercept term,

$B = \frac{1}{2}\frac{\Delta V_P}{V_P} - 4\gamma\frac{\Delta V_S}{V_S} - 2\gamma\frac{\Delta\rho}{\rho}$ is the gradient term,

$C = \frac{1}{2}\frac{\Delta V_P}{V_P}$ is the curvature term,

$\gamma = \left(\frac{V_S}{V_P}\right)^2, \quad \Delta V_P = V_{P2} - V_{P1}, \quad \Delta V_S = V_{S2} - V_{S1}, \ \Delta\rho = \rho_2 - \rho_1,$

$V_P = \frac{V_{P1} + V_{P2}}{2}, \quad V_S = \frac{V_{S1} + V_{S2}}{2}, \ and \ \rho = \frac{\rho_2 + \rho_1}{2}.$

Assuming a small incident angle (< 30^o) the third term can be dropped, and, if we assume that $V_P/V_S = 2$, or $\gamma = 1/4$, then the gradient can be simplified to

$$B = \frac{1}{2}\left(\frac{\Delta V_P}{V_P} + \frac{\Delta\rho}{\rho}\right) - \left(\frac{\Delta V_S}{V_S} + \frac{\Delta\rho}{\rho}\right), \tag{A-2}$$

and we also find that

$$\frac{\Delta V_P}{V_P} = \frac{\Delta V_S}{V_S}. \tag{A-3}$$

Substituting (A-3) into (A-2), we get:

$$B = -A \tag{A-4}$$

ACKNOWLEDGEMENTS

We wish to thank our colleagues at the CREWES Project and at Hampson-Russell Software for their support and ideas, as well as the sponsors of the CREWES Project.