Why Sigmoid Transformation Helps Incorporate Logic Into Deep Learning: A Theoretical Explanation

1st Chitta Baral
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406, USA
chitta@asu.edu

2nd Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@utep.edu

***Abstract*—Traditional neural networks start from the data, they cannot easily handle prior knowledge – this is one of the reasons why they often take very long to train. It is desirable to incorporate prior knowledge into deep learning. For the case when this knowledge consists of propositional statements, a successful way to incorporate this knowledge was proposed in a recent paper by van Krieken et al. That paper uses the fact that a neural network does not directly return a truth value, it returns a real value – in effect, the degree of confidence in the corresponding statement – from which we extract the truth value by fixing a threshold. Thus, the authors of the paper used formulas for transforming degrees of confidence in individual statements into a reasonable estimate for the degree of confidence in their logical combinations, formulas developed and studied under the name of fuzzy logic. However, it turns out the direct use of these formula often leads to very slow training. That paper showed that we can get effective training if instead of directly using the resulting degree of confidence we first apply a sigmoid-related transformation. In our paper, we provide a theoretical explanation of this semi-empirical idea: specifically, we show that under reasonable conditions, the optimal nonlinear transformation is either a sigmoid or an (arc)tangent or an appropriate combination of sigmoids, (arc)tangents, and their limit cases (such as linear functions).**

***Index Terms*—deep learning, fuzzy logic, sigmoid transformation**

I. Formulation of the Problem

Standard neural networks with a binary output: reminder. One of the main applications of machine learning techniques is to decide, based on the input, whether a certain property P is satisfied or not, for example, whether an animal in the picture is a dog; see, e.g., [2]. In a neural network, we usually do not directly get the truth value of this property. Instead, we get a real number that describes, crudely speaking, the degree of confidence that this property is true. We can then set a threshold and conclude that the desired property is true if the degree of confidence is larger than or equal to this threshold.

Usually, to train this neural network, we use the gradient-descent-based backpropagation algorithm.

Need to take logic into account. In many practical situations, we are not just interested in one single property, we are interested in checking whether a logical combination of such properties is true. For example, when we are driving a car, our decision to slow down or to speed up depends not only on the presence or absence of other cars, but also on the weather, on our own tiredness, etc. In such cases, the desired property P is a logical combination of other properties $P_1, \ldots, P_n$, e.g., a combination obtained by using the usual logical connectives such as "and", "or", and "not".

In principle, we can train n neural networks to check whether each the properties $P_1, \ldots, P_n$ is true, and then use the usual logical operations to come up with the truth value of the composite property P. The limitations of this approach is related to the fact that in general, in machine learning, the more examples we have for training, the more accurate and reliable the results. If we only use neural networks to check properties $P_1, \ldots, P_n$, we will not be able to use the cases when we only know whether P was true or not – while having no information whether the properties P_i were true or not.

Alternatively, we could simply train a single neural network to check whether the property P is true – but this way, we miss examples where we only know some of the properties P_i but not whether P was true or not. From this viewpoint, it is desirable to somehow incorporate the logical relation between P and P_i in the neural network.

The incorporation of logic into neural networks. Such an incorporation has indeed been proposed and successfully used; see, e.g., [10]. This incorporation is based on the following idea. Instead of simply applying thresholds to n separate neural networks for checking n properties:

- we feed the original real-valued outcomes into a special logical layer, where our degrees of confidence in state-

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology. It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

ment P_i are transformed into a degree of confidence in the composite statement P, and
- only after that, we apply the threshold to come up with a definite answer.

This way:
- if we have examples when we only know some values P_i, we can train the i-th part of the resulting network, and
- if we only know P, we can also train all the parts.

When all the logical operations are differentiable, we can use backpropagation for this training.

In computer science, operations that transform our degree of confidence in two statements A and B into an estimate for the degree of certainty in logical combinations $A \,\&\, B$ and $A \vee B$ are known – they form what is known as *fuzzy logic*; see, e.g., [1], [4], [5], [7], [8], [12]. The motivation for fuzzy logic was different: it came from the need to use experts' degrees of certainty in individual statements to estimate the degrees of certainty corresponding to their logical combinations – but mathematics there is general, and many of the proposed operations are differentiable.

For example, one of the simplest "and"-operations is the product $a \cdot b$, when we estimate our degree of confidence in a statement $A \,\&\, B$ as the product of our degrees of confidence a and b in the component statements A and B. Similarly, we can estimate our degree of confidence in $A \vee B$ as $a + b - a \cdot b$. Both these functions $a \cdot b$ and $a + b - a \cdot b$ are perfectly differentiable.

Need for a sigmoid transformation. If we use differentiable fuzzy logic operations, then, in principle, we can utilize the general backpropagation ideas to train the resulting combined neural network. However, it turned out that in many cases, the resulting training is too slow to be practical: the gradient is close to 0, so backpropagation (when the changes in the weights are proportional to this gradient) is too slow.

For example, when we initially think that both A and B are false, with degrees of confidence a and b close to 0, and we have new evidence that they are probably both true, the gradient of the expression $a \cdot b$ is equal to (b, a), i.e., is indeed close to 0.

Interestingly, the authors of [10] came up with an effective solution: instead of the degree of confidence x obtained by applying the corresponding fuzzy logic operations, we first apply a non-linear sigmoid transformation to this value, resulting in $y = 1/(1 + \exp(-x))$. To be more precise, they:
- first perform some linear transformation $x \mapsto a + b \cdot x$,
- then they apply the sigmoid, and
- then they again apply some linear transformation.

A natural question and what we do in this paper. A natural question is: why namely this logistic transformation works and not any other non-linear function? In this paper, we provide a possible answer to this question.

Namely, we show that, under some reasonable assumptions, the optimal non-linear transformation is either the sigmoid function (or one of its limit cases) or the arctangent, or the inverse of the sigmoid function or of the arctangent, or a combination of these two functions and their inverses.

II. Analysis of the Problem

Let us consider a general problem. Instead of focusing on the above specific problem – why the sigmoid transformation turned to be successful when incorporating logic into neural networks – let us consider this problem from the most general viewpoint: which nonlinear transformations are optimal in some reasonable sense?

To answer this question, let us analyze what is usually meant by optimal.

What is optimal: analysis. In applied mathematics, when we talk about optimality, usually we mean that we have an objective function $F(\alpha)$ that we want to maximize (or minimize), and the optimal alternative $\alpha_{\rm opt}$ is the one that leads to the largest possible value of this objective function. For example:
- the optimal plan for a company is the one that leads to the largest profit,
- the optimal algorithm for solving a large system of equations is the one that provides, on average, the most accurate solution, etc.

However, in practice, optimization can mean something more complex. For example, if we have several alternatives with the same largest possible value of the original objective function, then we can use this non-uniqueness to optimize something else. For example, if two algorithms have the same average accuracy, we can select, among them, the one that has the smallest worst-case approximation error. In this case, the relation between two alternatives α and β is no longer simply the comparison of the values $F(\alpha)$ and $F(\beta)$, we need to also known the value of the auxiliary objective function. If this does not lead to the unique selection, this means that our optimality criterion is not yet final, we can use this non-uniqueness to optimize something else.

How can we describe all such possible very complex situations? What is important is that:
- for some alternatives α and β, we can tell that α is better than β; we will denote it by $\alpha > \beta$, and
- for some alternatives α and β, we can tell that α is of the same value as β; we will denote it by $\alpha \sim \beta$.

Of course, these relations must be consistent: e.g., if α is better than β and β is better than γ, then we should have $\alpha > \gamma$. Thus, we arrive at the following definitions.

Definition 1. *Let A be a set; its elements will be called* alternatives. *By an* optimality criterion *on the set A, we mean a pair of relations $(>, \sim)$ that satisfy the following properties for all alternatives α, β, and γ:*
- *if $\alpha > \beta$ and $\beta > \gamma$, then $\alpha > \gamma$;*
- *if $\alpha > \beta$ and $\beta \sim \gamma$, then $\alpha > \gamma$;*
- *if $\alpha \sim \beta$ and $\beta > \gamma$, then $\alpha > \gamma$;*
- *if $\alpha \sim \beta$ and $\beta \sim \gamma$, then $\alpha \sim \gamma$;*

- $\alpha \sim \alpha$;
- *if* $\alpha \sim \beta$, *then* $\beta \sim \alpha$;
- *if* $\alpha > \beta$, *then we cannot gave* $\alpha \sim \beta$.

Definition 2.

- *We say that the alternative* $\alpha_{\rm opt}$ *is* optimal *with respect to the optimality criterion* $(>, \sim)$ *if for every* $\alpha \in A$, *we have either* $\alpha_{\rm opt} > \alpha$ *or* $\alpha_{\rm opt} \sim \alpha$.
- *We say that the optimality criterion is* final *if there exists exactly one optimal alternative.*

We need to look for a family of transformations. We are interested in transformations between physical quantities. However, in a computer, we only deal with numbers, and what number is assigned to a quantity depends on what measuring unit and what starting point we choose. For example, we can measure time in years or in days, we can start measuring time at year 0 in our usual calendar or at the moment when recording started, etc.

In general, if we use a measuring unit which is λ times smaller, all numerical values are multiplied by λ: $x \mapsto \lambda \cdot x$. For example, 2 meters becomes $100 \cdot 2 = 200$ centimeters. If we select a starting point which is x_0 units earlier than before, then this value x_0 is added to all the numerical values: $x \mapsto x + x_0$. By changing both, we get generic linear transformations $x \mapsto \lambda \cdot x + x_0$.

Such transformations make sense not only for physical quantities like time, but also for degrees of confidence. Indeed, one way to estimate the degree of confidence in a statement is to poll experts. If M out of N experts believe that this statement is true, we take the ratio M/N as the desired degree x. To get the most accurate estimate, we should ask N top experts in the field. As usual with statistics-like estimates, if we want to get a more accurate estimate of the degree of certainty, we can ask more experts. The problem is that these additional m experts may be intimidates by opinion of top N experts, so they either be reluctant to express any opinion at all, or they will simply copy the opinion of the top-expert majority. In the first case, we get $M/(N+m)$, i.e., $\lambda \cdot x$, where $\lambda = N/(N+m)$. In the second case, when most top experts were positive, we get $(M+m)/(N+m)$, i.e., $\lambda \cdot x + x_0$, where $x_0 = m/(N+m)$.

In general, because of the possibility of such a re-scaling, the same transformation $y = f(x)$ can change its form if we use different units for measuring x or y. Which transformation is optimal should not depend on which unit we use, so the transformation $y = f(x)$ and, e.g., $y = f(x) + a$ should be equally good. Thus, we cannot say that a single transformation function is optimal, we should consider *families* of transformation functions.

In view of the fact that we are interested in using back-propagation, i.e., using derivatives, all transformations must be smooth (i.e., differentiable).

Definition 3. *Let* r *be a natural number.*

- *By an* smooth r-parametric family of functions*, we mean a smooth mapping* F *that assigns, to every real value* x *and to every point* $(c_1, \ldots, c_r)$ *from some open subset* U *of the set* $\mathbb{R}^r$ *of all* r*-tuples of real numbers, a value* $F(x, c_1, \ldots, c_r)$.
- *We say that a function* $f(x)$ belongs *to the family* F *if there exists values* $c_1, \ldots, c_r$ *for which, for every* x, *we have* $f(x) = F(x, c_1, \ldots, c_r)$.

Comment. In general, it is desirable to select a solution which is as simple as possible. For families of functions, this means, in particular, that we should consider families with small number of parameters r.

Invariance. We have already mentioned that the relative quality of two families of transformation functions should not depend on what measuring unit and what starting point we use for measuring x and/or y. In addition to such linear re-scalings, we may have different measurement scales that are related to each other by a non-linear transformation. What are natural non-linear transformations?

If we have a natural transformation from scale A to scale B, then an inverse transformation should also be natural. Similarly, if we have a natural transformation from A to B and another natural transformation from scale B to scale C, then their composition should also be a natural transformation from A to C. Thus, the class of all natural transformations should be closed under inverse and under composition. Such classes are known as *transformation groups*.

We also want to make sure that all natural transformation be implementable in a computer, and in a computer, at any given moment of time, we can only store finitely many numbers. Thus, the class of all natural transformations should depend on finitely many parameters, i.e., in mathematical terms, it should be *finite-dimensional.* So, natural transformations form a finite-dimensional transformation group that contains all linear transformations. The description of all such transformation groups was first proposed – without proof – in [11]; in [3], [9], this description was proved to be correct. In our 1-D case, of transformations from real numbers to real numbers, the corresponding result states that all such natural transformations are fractional linear, i.e., have the form

$$x \mapsto \frac{\lambda \cdot x + x_0}{1 + c \cdot x};$$

see also [6]. Because of this result, in the following text, we will call such transformations *natural*.

As we have mentioned, it makes sense to require that the comparison between two families of transformation functions $y = f(x)$ does not change if we apply a natural transformation either to x or to y.

Definition 4.

- *Let* α *be a family of functions, and let* T *be a natural transformation. Then we denote*

  $$T_x(\alpha) = \{f(T(x)) : f \in \alpha\} \text{ and } T_y(\alpha) = \{T(f(x)) : f \in \alpha\}.$$

- *We say that the optimality criterion on the set all* r*-parametric families is* scale-invariant *if the following two*

conditions hold for or every two families α and β and for every natural transformation T:

- *if $\alpha > \beta$, then $T_x(\alpha) > T_x(\beta)$ and $T_y(\alpha) > T_y(\beta)$; and*
- *if $\alpha \sim \beta$, then $T_x(\alpha) \sim T_x(\beta)$ and $T_y(\alpha) \sim T_y(\beta)$.*

Now, we are ready to formulate our main result.

III. Main Result

Proposition. *For $r \le 5$, for every scale-invariant final optimality criterion of the set of all r-parametric families, each function from the optimal family $\alpha_{\rm opt}$ has the form $f(x) = g^{-1}(\text{const} - h(x))$, where g^{-1} means the inverse function, and each of the two functions $g(x)$ and $h(x)$ has one of the following forms:*

- *the function can be linear, in which case its inverse is also linear;*
- *the function can have the form $g(x) = a \cdot \ln(b \cdot x + c) + d$, in which case the inverse function takes the form*

$$g^{-1}(x) = A + B \cdot \exp(C \cdot x);$$

- *the function can be fractional linear*

$$g(x) = \frac{a + b \cdot x}{1 + c \cdot x},$$

in which case the inverse function is also fractional linear;

- *the function can have the form*

$$g(y) = a \cdot \arctan(b + c \cdot x) + d,$$

in which case the inverse function takes the form

$$g^{-1}(x) = A \cdot \tan(B + C \cdot x) + D;$$

and

- *the function can have the form*

$$g(y) = a_0 \cdot \ln\left(\frac{a + b \cdot x}{1 + c \cdot x}\right) + b_0,$$

in which case the inverse function takes the form

$$g^{-1}(x) = A \cdot \sigma(B + C \cdot x) + D, \text{ where}$$

$$\sigma(x) \stackrel{\rm def}{=} \frac{1}{1 + \exp(-x)}.$$

Comments.

- One can show that the first three cases are, in effect, limit cases of the fourth (arctangent) case and of the fifth (sigmoid) case.
- When $h(x) = -x$ and $g^{-1}(x)$ is a sigmoid, we get exactly the transformation that we want to explain.

Proof.

1°. Let us first prove that the optimal family $\alpha_{\rm opt}$ is itself invariance with respect to transformations T_x and T_y, i.e., we have $T_x(\alpha_{\rm opt}) = T_y(\alpha_{\rm opt}) = \alpha_{\rm opt}$.

Indeed, the fact that the family $\alpha_{\rm opt}$ is optimal means that for every family α, we have

- either $\alpha_{\rm opt} > \alpha$
- or $\alpha_{\rm opt} \sim \alpha$.

In particular, this means that for every family $T_x^{-1}(\alpha)$, where T_x^{-1} means an inverse transformation, we have either $\alpha_{\rm opt} > T_x^{-1}(\alpha)$ or $\alpha_{\rm opt} \sim T_x^{-1}(\alpha)$. Due to scale-invariance, this implies that either $T_x(\alpha_{\rm opt}) > \alpha$ or $T_x(\alpha_{\rm opt}) \sim \alpha$. This is true for every alternative α. By definition of an optimal alternative, this means that the alternative $T_x(\alpha_{\rm opt})$ is optimal. However, our optimality criterion is final, which means that there is only one optimal alternative. Thus, $T_x(\alpha_{\rm opt}) = \alpha_{\rm opt}$.

Similarly, we can prove that the optimal family is invariant with respect to transformations T_y.

2°. The fact that the optimal family is scale-invariant means that for every function $f(x)$ from this family and for every two fractional-linear transformations T and U, the function $T(f(U(x)))$ also belongs to this family. We have a 3-parametric family of transformations T and a 3-parametric family of transformations U. Thus, if all the functions $T(f(U(x)))$ were different, we would have a 6-parametric family, but we know that the family is at most 5-parametric. Hence, there exists at least a 1-dimensional family of pairs of transformations (T_a, U_a) depending on the parameter a for which, for all x and for all values a and b of the corresponding parameter, we have $T_a(f(U_a(x))) = T_b(f(U_b(x)))$.

Let us fix some value b. Then, by applying the inverse transformation T_b^{-1} to both sides of this equality, we get

$$t_a(f(U_a(x))) = f(U_b(x)), \tag{1}$$

where $t_a(y) \stackrel{\rm def}{=} T_b^{-1}(T_a(y))$.

For any number X, we can take $x = U_b^{-1}(X)$, so that $U_b(x) = X$. Substituting this expression for x into the formula (1), we conclude that

$$t_a(f(u_a(X))) = f(X), \tag{2}$$

where we denoted $t_a(X) \stackrel{\rm def}{=} U_a(U_b^{-1}(X))$.

The transformations t_a and u_a are obtained from fractional-linear ones by using inverse and composition. Since fractional linear transformations form a group, the transformations t_a and u_a also belong to the same group, i.e., are also fractional linear. So, we have a 1-parametric family of pairs (t_a, u_a) of fractional linear transformations for which the equality (2) holds for all X.

3°. If this property (2) holds for two pairs of transformations (t_a, u_a), then it also holds for their composition and for their inverse. Thus, such pairs form a group, and the corresponding pairs of transformations (t_a, u_a) also form a group.

All 1-parametric groups are – at least locally – isomorphic, in particular, they are all isomorphic to the group of shifts $v \mapsto v + v_0$. Let us denote by $t^*_{v_0}$ and $u^*_{v_0}$ transformations corresponding to $v \mapsto v + v_0$. So, the equality (2) takes the form

$$t^*_{v_0}(f(u^*_{v_0}(X))) = f(X) \tag{3}$$

for all X and v_0.

Isomorphism means that there exist functions $g(y)$ and $h(X)$ – describing this isomorphism – for which

$$g(t^*_{v_0}(y)) = g(y) + v_0, \tag{4}$$

and

$$h(u^*_{v_0}(X)) = h(X) + v_0. \tag{5}$$

By applying the function $g(y)$ to both sides of the equality (3), we conclude that

$$g(t^*_{v_0}(f(u^*_{v_0}(X)))) = g(f(X)). \tag{6}$$

Due to (4), this leads to

$$g(f(u^*_{v_0}(X))) + v_0 = g(f(X)). \tag{7}$$

Instead of the original variable X, let us consider the new variable $z = h(X)$ for which $X = h^{-1}(z)$. In terms of this new variable z, the equality (7) takes the form

$$g(f(h^{-1}(h(u^*_{v_0}(X)))))+v_0 = g(f(h^{-1}(h(x))) = g(f(h^{-1}(z)). \tag{8}$$

Due to (5), this leads to

$$g(f(h^{-1}(h(X) + v_0))) + v_0 = g(f(h^{-1}(z)), \tag{9}$$

i.e., to

$$g(f(h^{-1}(z + v_0))) + v_0 = g(f(h^{-1}(z)). \tag{10}$$

Thus, for the function

$$F(z) = g(f(h^{-1}(z))), \tag{11}$$

we have

$$F(z + v_0) + v_0 = F(z). \tag{12}$$

For $z = 0$, we thus get $F(v_0) + v_0 = F(0)$, i.e.,

$$F(v_0) = \text{const} - v_0. \tag{13}$$

By applying the inverse transformation $g^{-1}(y)$ to both sides of the equality (11) and by taking $z = h(x)$, we get

$$f(x) = g^{-1}(F(h(x))),$$

i.e., due to (13):

$$f(x) = g^{-1}(\text{const} - h(x)). \tag{14}$$

Thus, to find all possible transformation functions $f(x)$ from the optimal family, it is sufficient to find all possible homomorphisms $g(y)$ and $h(X)$.

4°. Let us describe all the functions $g(y)$ that satisfy the equality (4). In this equality, the transformation $t^*_{v_0}$ is fractional linear, so it has the form

$$t^*_{v_0}(y) = \frac{a(v_0) + b(v_0) \cdot y}{1 + c(v_0) \cdot y} \tag{16}$$

for some functions $a(v_0)$, $b(v_0)$, and $c(v_0)$. Thus, the formula (4) has the form:

$$g\left(\frac{a(v_0) + b(v_0) \cdot y}{1 + c(v_0) \cdot y}\right) = g(y) + v_0. \tag{17}$$

Differentiating both sides of this formula with respect to v_0 and taking $v_0 = 0$, we conclude that

$$\frac{dg}{dy} \cdot \left(p + q \cdot y + c \cdot y^2\right) = 1 \tag{18}$$

for some values p, q, and r.

We can separate the variables if we divide both sides by the quadratic expression and multiply both sides by dy, then we get

$$dg = \frac{dy}{p + q \cdot y + r \cdot y^2} = dx.$$

Integrating both parts, we conclude that

$$g(y) = \int \frac{dy}{p + q \cdot y + r \cdot y^2}. \tag{19}$$

Let us consider possible expressions for this integral, expressions depending on which coefficients are equal to 0 and which are different from 0.

5°. Let us first consider the case when $q = r = 0$. In this case, (19) leads to the fact that $g(y)$ is a linear function.

6°. Let us now consider the case when $r = 0$ but $q \neq 0$. In this case, we have

$$g(y) = \int \frac{dy}{p + q \cdot y} = \int \frac{1}{q} \cdot \frac{d(p + q \cdot y)}{p + q \cdot y} = \frac{1}{q} \cdot \ln(p+q\cdot y)+C,$$

where C is the integration constant.

Here, $g(y) = v$ if and only if

$$v = \frac{1}{q} \cdot \ln(p + q \cdot y) + C,$$

so $\ln(p+q\cdot y) = q\cdot v - C$, hence $p+q\cdot y = \exp(q\cdot v - C) = \text{const}\cdot\exp(q\cdot v)$ and thus, the inverse function takes the form $y = \text{const}\exp(q\cdot v) + \text{const}$.

7°. When $r \neq 0$, then, by a linear transformation of y, to some $Y(y)$, we can reduce the quadratic form $p+q\cdot y+r\cdot y^2$ to the full-square form, i.e., either to Y^2 or to $1+Y^2$ or to $1-Y^2$.

7.1°. When the reduced quadratic form has the form Y^2, integration leads to

$$g(Y) = \int \frac{dY}{Y^2} = -\frac{1}{Y} + C.$$

Taking into account that Y is a linear function of y, we get a general fractional linear function. In this case, the inverse function is also fractional linear.

7.2°. When the reduced quadratic form has the form $1 + Y^2$, integration leads to

$$g(Y) = \int \frac{dY}{1 + Y^2} = \arctan(Y).$$

In this case, $g(y)$ is, in effect, an arctangent – modulo linear transformations before and after – and the inverse is, in effect, the tangent function.

7.3°. When the reduced quadratic form has the form $1-Y^2 = (1-Y)\cdot(1+Y)$, then

$$\frac{1}{1-Y^2} = \frac{1}{2}\cdot\left(\frac{1}{1-Y}+\frac{1}{1+Y}\right),$$

thus

$$g(Y) = \int \frac{dY}{1-Y^2} = \frac{1}{2}\cdot\left(\int \frac{dY}{1-Y} + \int \frac{dY}{1+Y}\right) =$$

$$\frac{1}{2}\cdot(-\ln(1-Y)+\ln(1+Y)) + C = \frac{1}{2}\cdot\ln\left(\frac{1+Y}{1-Y}\right) + C.$$

Taking into account that Y is a linear function of y, we get logarithm of a fractional-linear function. The inverse function in this case is linearly related to the sigmoid.

The proposition is proven.

REFERENCES

[1] R. Belohlavek, J. W. Dauben, and G. J. Klir, *Fuzzy Logic and Mathematics: A Historical Perspective*, Oxford University Press, New York, 2017.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.

[3] V. M. Guillemin and S. Sternberg, "An algebraic model of transitive differential geometry", *Bulletin of American Mathematical Society*, 1964, Vol. 70, No. 1, pp. 16–47.

[4] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[5] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, Springer, Cham, Switzerland, 2017.

[6] H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.

[7] H. T. Nguyen, C. L. Walker, and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2019.

[8] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer, Boston, Dordrecht, 1999.

[9] I. M. Singer and S. Sternberg, "Infinite groups of Lie and Cartan, Part 1", *Journal d'Analyse Mathematique*, 1965, Vol. XV, pp. 1–113.

[10] E. van Krieken, E. Acar, and F. van Harmelen, "Analyzing differentiable fuzzy logic operators", *Artificial Intelligence*, 2022, Vol. 302, Paper 103602.

[11] N. Wiener, *Cybernetics, or Control and Communication in the Animal and the Machine*, 3rd edition, MIT Press, Cambridge, Massachusetts, 1962.

[12] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.