Linear Separability as a Condition for Solving Multiple Problems by a Single Threshold Neuron

Kostadin Yotov, Emil Hadzhikolev, and Stanka Hadzhikoleva

Abstract The paper discusses the linear separability of data classes and the relationship of threshold neurons with class classifiers. The possibility of constructing a neuron that can solve two different problems without the need for an intermediate change in its parameters and architecture is shown theoretically. The idea is illustrated with a specific example of a neuron solving problems simultaneously with both Boolean functions "AND" and "OR". A conclusion has been drawn for the existence of a neuron that can solve a class of an infinite number of problems. A necessary condition for this is that the domain of the problems is linearly separable from the surface in the input data space and the existence of parallel classifiers for separability for each individual problem.

Keywords Threshold neuron · Perceptron · Linear separability

1 Introduction

Let a set of objects be given

$$A = \{A_1, A_2, \ldots A_m\}, m \in N, \text{ in which}$$

each object $A_i, i = 1 \ldots m$ is characterized by "n" different attributes of a completely random type. In the general case, when considering the linear separability, it is not necessary to define the type of the individual attributes in advance, but to draw attention to the possibility of grouping the objects into classes through these attributes.

K. Yotov · E. Hadzhikolev · S. Hadzhikoleva (✉)
University of Plovdiv "Paisii Hilendarski", Plovdiv, Bulgaria
e-mail: stankah@uni-plovdiv.bg

K. Yotov
e-mail: kostadin_yotov@uni-plovdiv.bg

E. Hadzhikolev
e-mail: hadjikolev@uni-plovdiv.bg

A. P. Pandian et al. (eds.), *Computer Networks, Big Data and IoT*, Lecture Notes on Data Engineering and Communications Technologies 117,
https://doi.org/10.1007/978-981-19-0898-9_46

One attribute can be qualitative—for example, “color” or “shape,” another—quantitative, for example, “weight” or “temperature,” and if necessary, for each qualitative characteristic we could give some quantitative expression.

Following this line of thought, if we introduce quantitative correspondences of qualitative characteristics, then we can describe each object only with quantitative characteristics. For example, if for the colors we assume green = 28, and for the shapes—polygon = 123, then the object green polygon is:

$$(\text{color} = \text{“green”}, \text{shape} = \text{“polygon”})$$

and it can be represented as:

$$(x_1 = 28, x_2 = 123).$$

To simplify the reasoning, let us assume that all the characteristics of the considered objects are quantitative values. This in turn means that each object of the set A can be considered as a point in the n-dimensional space of the attributes (Fig. 1), i.e.,

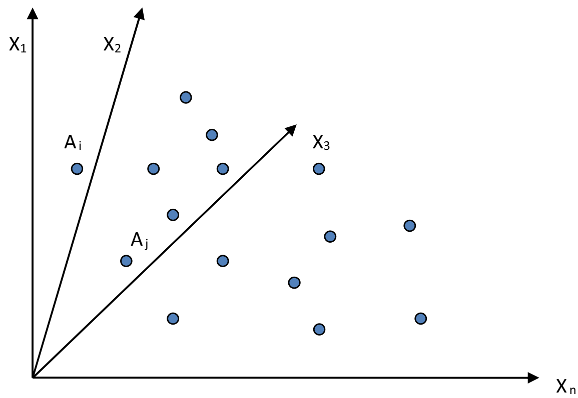
$$A_i = \{x_{i1}, x_{i2}, \ldots x_{in}\} \in E^n, \quad \forall\, A_i \in A, i = 1, 2 \ldots m$$

As for the linear separability of A, it can be represented as a union of two linearly separable classes [1, 2]:

$$\begin{aligned}
&B = \{\forall\, B_i(x_{i1}, x_{i2}, \ldots x_{in})/B_i \in A, i = 1, 2, \ldots m\} \text{ and}\\
&C = \left\{\forall\, C_j\left(x_{j1}, x_{j2}, \ldots x_{jn}\right)/C_i \in A, j = 1, 2, \ldots m\right\},\\
&\text{as } B \cap C = \emptyset,
\end{aligned}$$

if there exists a plane α with representation:

Fig. 1 Elements of the sum A, presented as points in E^n



$$\alpha : \sum_{i=1}^{n} w_i x_i + b = 0, \quad (\forall\, w_i \in R, i = 1, 2 \ldots n) \tag{1}$$

such as for $\forall\, B_i(x_{i1}, x_{i2}, \ldots x_{in}) \in B$, the inequality is fulfilled

$$\sum_{k=1}^{n} w_k x_{ik} + b < 0 \tag{2}$$

and $\forall\, C_j\left(x_{j1}, x_{j2}, \ldots x_{jn}\right) \in C$—respectively:

$$\sum_{k=1}^{n} w_k x_{jk} + b > 0. \tag{3}$$

Let us consider a two-dimensional space of attributes, assuming that all objects of the set A are characterized by only two qualities. In this case, $n = 2$ the plane (1) is reduced to its two-dimensional analogue—straight line with the equation:

$$\alpha : w_1 x_1 + w_2 x_2 + b = 0, w_i \in R, i = 1, 2\left(1'\right)$$

where x_1 and x_2 are the coordinates of an arbitrary point of α.

Transferred to E^2 (Fig. 2), based on the two-dimensional variant of conditions (2) and (3), classes B and C are linearly separable if for $\forall\, B_i(x_{i1}, x_{i2}) \in B$ the equation is fulfilled:

$$w_1 x_{i1} + w_2 x_{i2} + b < 0 \ (2')$$

and $\forall\, C_j\left(x_{j1}, x_{j2}, \ldots x_{jn}\right) \in C :$

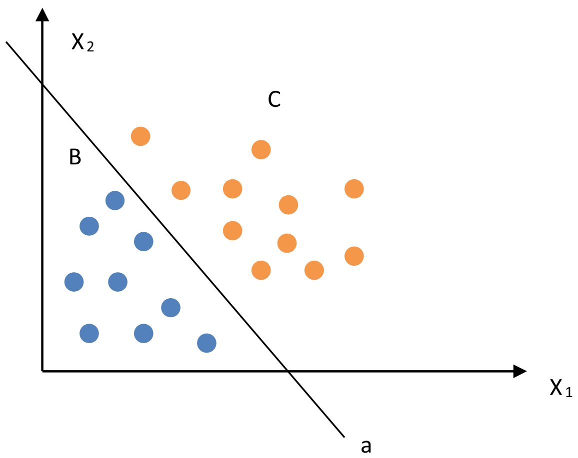


Fig. 2 Elements of the set A, presented as points in E^2

$$w_1 x_{j1} + w_2 x_{j2} + b > 0 \; (3')$$

Since the conditions (2′) and (3′), classifying the belonging of an object to one of the two classes are based on equation (1′), the line α is a linear classifier for B and C.

There are many optimization algorithms for finding a suitable classifier that can effectively separate classes [3, 4] and some testing methods for linear separability [5]. In our case, however, we are interested not so much in finding the most efficient separation, but in finding suitable parallel classifiers. Let us pay attention to the fact that if two classes are bounded and linearly separable, an infinite number of linear classifiers can be indicated (Fig. 3), which separate them in the way described by (1)–(3).

Let us look at the two-dimensional Boolean function "AND" (Table 1). Its domain consists of 4 points:

$$D = \{(0, 0), (0, 1), (1, 0)(1, 1)\}$$

Let us introduce the class of points B, containing the elements of D, for which $X_1 \wedge X_2 = 0$, and the class C, containing the points from D.O, for which $X_1 \wedge X_2 = 1$. Given the essence of the Boolean function "AND", we look for a classifier type (1′), for which the following system is implemented:

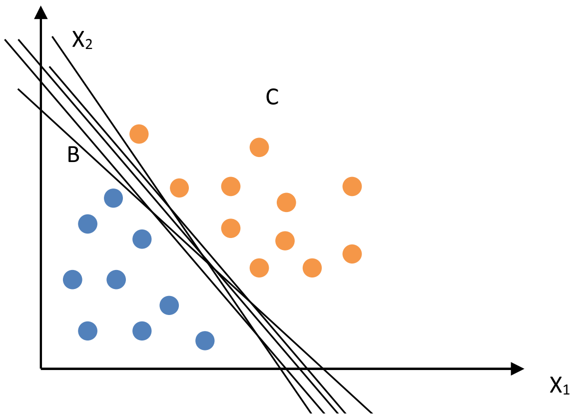


Fig. 3 Multiple classifiers separating the two classes B and C linearly

Table 1 Truth table for Boolean functions "AND" and "OR"

X_1	X_2	$X_1 \wedge X_2$	$X_1 \vee X_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

$$\left| \begin{array}{l} w_1 0 + w_2 0 + b < 0 \\ w_1 0 + w_2 1 + b < 0 \\ w_1 1 + w_2 0 + b < 0 \\ w_1 1 + w_2 1 + b > 0 \end{array} \right.$$

It is obvious that the free member b must meet the conditions:

$$\left| \begin{array}{c} b < 0 \\ b < \min\{-w_{1,} - w_2\} \\ b > -(w_1 + w_2) \end{array} \right. \tag{4}$$

Then for every two specific positive numbers w_1 and $w_{2,}$ we can find a corresponding value of b, by which we can define a classifying straight line of the type (1′), which will be only one of an infinite number of members of the same family of classifiers. If, for example, $w_1 = 0.3$ and $w_2 = 0.7$, then according to the system (4) we can choose arbitrarily b:

$$-1 < b < -0.7$$

One possible solution is $b = -0.8$. Thus, this particular representative of the whole possible class is given by the equation:

$$\alpha_1 : 0.3x_1 + 0.7x_2 - 0.8 = 0$$

For the coordinates of each point $(x_1, x_2) \in \alpha_1$ is met

$$x_2 = -\frac{0.3}{0.7}x_1 + \frac{0.8}{0.7}$$

This is a straight line passing through point (0, 1.143) and having an angular coefficient k = −(0.3/0.7) = −0.428. Figure 4 shows an illustration of other possible solutions:

$\alpha_2 : 0.3x_1 + 0.7x_2 - 0.9 = 0$,

which has the same angular coefficient as that of α_1, and

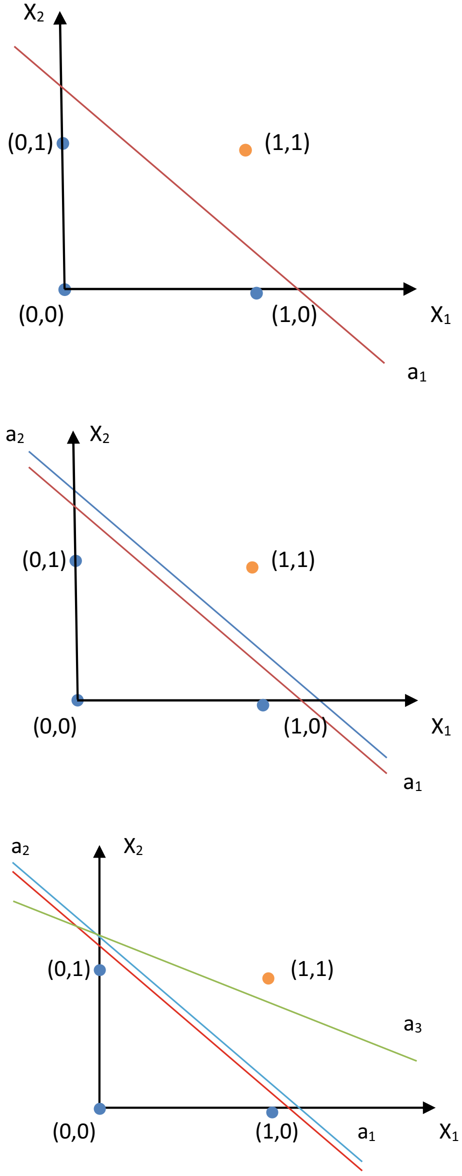
$\alpha_3 : 2x_1 + 9x_2 - 10 = 0$,

with angular coefficient k = −0.222.

2 Threshold Neuron and Its Relationship with Linear Classifiers

Let us consider a threshold neuron with n inputs which are activated through the following step function:

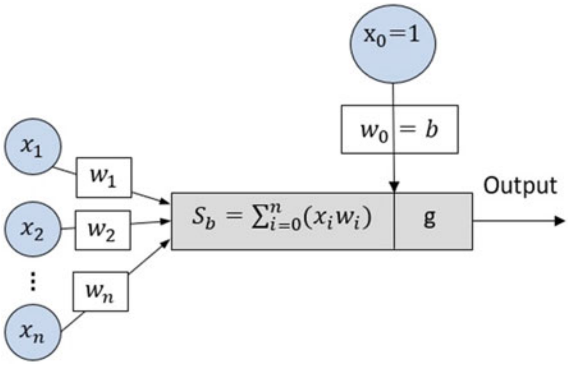
Fig. 4 Three straight lines classifying the domain of the Boolean function "AND"



$$g(S) = \begin{cases} 0, \text{ at } S < 0 \\ 1, \text{ at } S \geq 0 \end{cases} \tag{5}$$

If the weight vector $\overrightarrow{W}$ has coordinates $(w_1, w_2, \ldots, w_n)$, at input stimuli $(x_1, x_2, \ldots x_n)$ along the axon of the neuron, a signal will propagate with the following

Fig. 5 Threshold neuron



value:

$$\text{Output} = g\left[\sum_{i=1}^{n} w_i x_i + b\right],$$

where b is the threshold of the neuron [6, 7]. Given the peculiarities of the activating function, it is clear that this signal has a binary character—it will be "0" or "1". The neuron thus defined is called the threshold logic or TLU (Fig. 5), or just threshold neuron [8]. The term perceptron is often used as a synonym for threshold logic unit, although the perceptron is generally much more than a simple threshold logic unit [9].

Based on the way the TLU is constructed, it is clear that if the stimuli $(x_1, x_2, \ldots x_n)$ appear in its dendritic tree, at fixed weights $(w_1, w_2, \ldots, w_n)$, only the following results are possible:

$$g\left[\sum_{i=1}^{n} w_i x_i + b\right] = \begin{cases} 0 \\ 1 \end{cases}$$

Thus, unlike other neurons, TLU easily realizes the concept of "all or nothing," which makes it especially suitable for solving logical problems [10].

We will consider a special case when using threshold neurons, namely the one in which the set of input data is linearly divisible by a given attribute.

Thus, let the domain of the input variables be linearly divisible with respect to the ordered n-tuple $(x_1, x_2, \ldots x_n)$ leading to the appearance of "1" at the output of the neuron, and those $(x_1, x_2, \ldots x_n)$, which generate the end result "0". From the assumed linear separability, it follows that there is a plane:

$$\alpha : \sum_{i=1}^{n} w_i x_i + b = 0 (\forall\ w_i \in R, i = 1, 2 \ldots n), \text{ such that}$$

for $\forall(x_{01}, x_{02}, \ldots x_{0n})$, for which at the output we have

$$g\left[\sum_{i=1}^{n} w_i x_{0i} + b\right] = 0$$

the inequality is fulfilled

$$\sum_{i=1}^{n} w_i x_{0i} + b < 0,$$

and for $\forall(x_{11}, x_{12}, \ldots x_{1n})$, for which at the output we have

$$g\left[\sum_{i=1}^{n} w_i x_{1i} + b\right] = 1, \text{ we have: } \sum_{i=1}^{n} w_i x_{1i} + b > 0$$

The presented information about the linear classifiers outlines their connection with the threshold neurons. If we look at the total signal in the body of the artificial neuron, we will notice the obvious classification form:

$$S = \sum_{i=1}^{n} w_i x_i + b,$$

while the classification itself is done through the activating function (5). Thus, the action of the trained threshold neuron could be considered as an act of classification of the linearly separable domain of its input variables. This essential relationship between TLU and linear classifiers allows us to draw two very important conclusions:

(1) Based on each mathematically constructed linear classifier, we can form a threshold neuron that is genetically prepared to solve the classifier problem. If we consider, for example, one of the classifiers for the Boolean function "AND", which we built above:

$$\alpha_1 : 0.3x_1 + 0.7x_2 - 0.8 = 0$$

It is clear that we could form the neuron as shown in Fig. 6. Along the axon, we have a signal which, given the weights and thresholds, fully corresponds to the Boolean function "AND".

(2) We mentioned that if two classes are linearly separable, then there are infinitely many linear classifiers separating the objects in each of the classes. And since the classifier is uniquely determined by the coefficients $w_1, w_2, \ldots, w_n, b$, with which we could subsequently form a corresponding threshold neuron, this means something very important to us, namely ***For each specific problem***

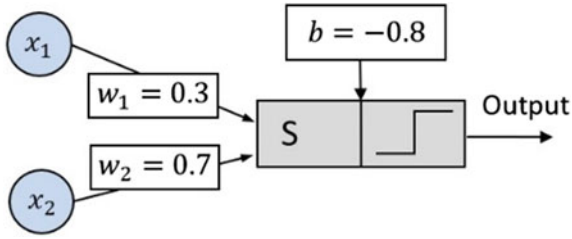


Fig. 6 Threshold neuron prepared to solve the Boolean function "AND"

with a linear separable compact domain of the input variables, there are an infinite number of threshold neurons that are able to solve it.

On the other hand, from the existence of infinitely many, but let us emphasize now, ***parallel*** linear classifiers, follows the existence of neurons:

$$H_1 : Out_1 = g\left[\sum_{i=1}^{n} w_{1i}x_i + b_1\right]$$

$$H_2 : Out_2 = g\left[\sum_{i=1}^{n} w_{2i}x_i + b_2\right]$$

....

$$H_r : Out_r = g\left[\sum_{i=1}^{n} w_{ri}x_i + b_r\right]$$

....

with weights for which

$$\frac{w_{1p}}{w_{1q}} = \frac{w_{2p}}{w_{2q}} = \cdots = \frac{w_{rp}}{w_{rq}} = \cdots, p \neq q;\ p, q = 1, 2, \ldots n$$

and associated with planes

$$S_j = 0, \text{ where}$$

$$S_j = \sum_{i=1}^{n} w_{ji}x_i + b_j,\ j \in N.$$

3 Solving Multiple Problems from a Single Threshold Neuron

An interesting question is about the possibility of the same neural network to solve a set of several tasks. There are various researches on this issue. For example, Kirkpatricka and team apply a special type of neural network regularization, which is associated with sequential training of the network to solve two tasks—A and B [11]. In the second task B, the weights required for the first task A are retained and the gradient descent continues. On the other hand, Yang and team train single network models to perform 20 cognitive tasks that depend on working memory, decision making, categorization, and inhibitory control [12]. The authors find that after training, recurrent units can be grouped into clusters that are functionally specialized for different cognitive processes and introduce a simple but effective measure to quantify relationships between single-unit neural representations of tasks. In the present study, we focus on the possibility of a separate threshold neuron to solve two logical functions simultaneously, without the need for sequential training for both tasks, as in the networks of Kirkpatricka and team.

The connection of the threshold neurons with the linear classifiers creates preconditions for searching for ways to apply linear separation in neural networks. We have already mentioned that for each specific problem with a bounded, closed and linearly separable area of the input data, an infinite number of neurons can be constructed, which are genetically prepared to solve it. Let us look at things from a different perspective and ask ourselves the question—***is it possible for a threshold neuron to be constructed in a way that allows it to solve several different problems?***

Let two problems be given related to finding the binary solutions of different functions of n-tuples: $f_1 = f_1(x_1, x_2, \ldots x_n)$ и $f_2 = f_2(x_1, x_2, \ldots x_n)$. Let the domains $D.O_1$ and $D.O_2$ of the two functions be compact and linearly separable with respect to the n-tuples $(x_1, x_2, \ldots x_n)$, for which f_1 and f_2 return "0" and, respectively, "1". Under these conditions, it follows that there are threshold neurons H_1 and H_2, which successfully solve the two problems (Fig. 7) in the following way:

$$H_1 : f_1(x_1, x_2, \ldots, x_n) = g\left[\sum_{i=1}^{n} w_{1i}x_i + b_1\right]$$

$$H_2 : f_2(x_1, x_2, \ldots, x_n) = g\left[\sum_{i=1}^{n} w_{2i}x_i + b_2\right]$$

However, it is there a vector $(w_1, w_2, \ldots, w_n)$, and a value for b, with which both problems can be solved by a single neuron? What would such a threshold neuron look like? If we submit only the variables $(x_1, x_2, \ldots, x_n)$ at the input, will the sought TLU know what to do with them? Should we use this data to calculate the function $f_1(x_1, x_2, \ldots, x_n)$, or should we use the inputs provided to calculate the value on $f_2(x_1, x_2, \ldots, x_n)$? Obviously, along with the input data, the neuron needs

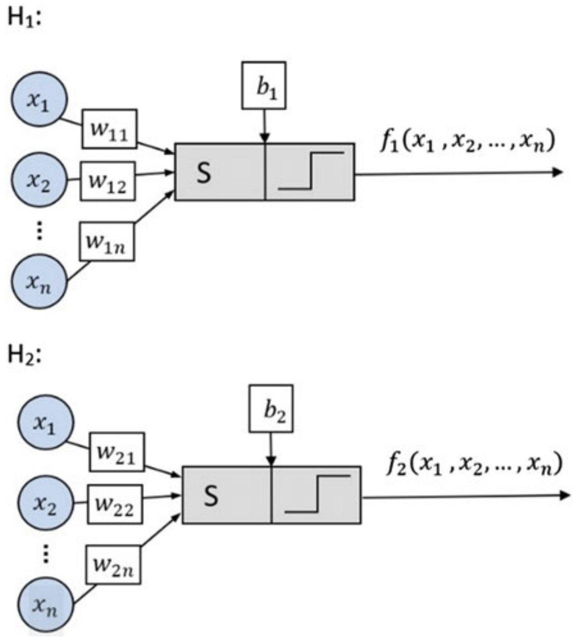


Fig. 7 Threshold neurons H_1 and H_2, solving two problems

another input to get a question: "How much is $f_1 = f_1(x_1, x_2, \ldots, x_n)$?" or the task "Calculate $f_2 = f_2(x_1, x_2, \ldots, x_n)$."

The purpose at first glance is to find the weights $\{w_i\}_{i=1}^n$ and the threshold b, for which:

$$g\left[\sum_{i=1}^{n} w_i x_i + b\right] = \begin{cases} f_1(x_1, x_2, \ldots, x_n), \text{Question: "How much is } f_1(x_1, x_2, \ldots, x_n)\text{?"} \\ f_2(x_1, x_2, \ldots, x_n), \text{Question: "How much is } f_2(x_1, x_2, \ldots, x_n)\text{?"} \end{cases}$$

Now we have another input for the neuron, and that is "Question" (Fig. 8). It is also a variable input value to the sought neuron and is fed to its body through the

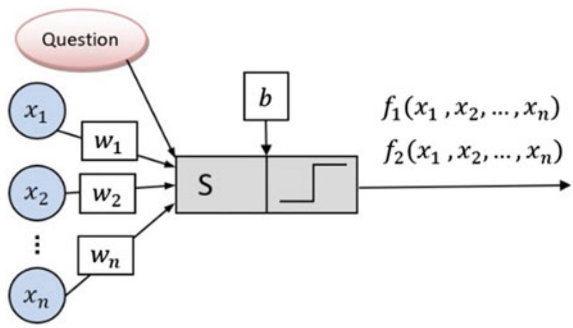


Fig. 8 Structure of a neuron solving both problems simultaneously

dendritic tree, along with the other variables. Thus, with the emergence of the need for a question, the input vector now takes the shape $\overrightarrow{X}(x_1, x_2, \ldots, x_n, x_{n+1} = \text{question})$, and the weight—$\overrightarrow{W}(w_1, w_2, \ldots, w_{n,} w_{n+1})$. That is why we are actually looking for

$$g\left[\sum_{i=1}^{n+1} w_i x_i + b\right] = \begin{cases} f_1(x_1, x_2, \ldots, x_n), \ x_{n+1} = \text{“How much is } f_1(\mathrm{x}_1,\mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n})\text{?”} \\ f_2(x_1, x_2, \ldots, x_n), \ x_{n+1} = \text{“How much is } f_2(\mathrm{x}_1,\mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n})\text{?”} \end{cases}$$

We have already discussed that the qualitative characteristics of objects can be represented by quantitative values. And since at the input of the neuron there is no way to ask the question x_{n+1} directly by using some linguistic construction, we need a quantitative interpretation that the neuron can process correctly. So, let

$x_{n+1} = 0$, If we want to ask the question “How much is $f_1(x_1, x_2, \ldots, x_n)$?”

and

$x_{n+1} = 1$, if the question is “How much is $f_2(x_1, x_2, \ldots, x_n)$?”

This way, we look for weights $\{w_i\}_{i=1}^{n+1}$ of a neuron with $n + 1$ inputs $x_1, x_2, \ldots, x_n, x_{n+1}$ and a threshold b, for which:

$$g\left[\sum_{i=1}^{n+1} w_i x_i + b\right] = \begin{cases} f_1(x_1, x_2, \ldots, x_n), \ x_{n+1} = 0 \\ f_2(x_1, x_2, \ldots, x_n), \ x_{n+1} = 1 \end{cases}$$

The ability of the neuron H_1 to calculate correct values for the function $f_1(x_1, x_2, \ldots, x_n)$ means that there is a linear classifier of the domain D.O$_1$ of the input variables of the first problem, represented by a plane

$S_1 : \sum_{i=1}^{n} w_{1i} x_i + b_1 = 0,$

where x_i are the coordinates of any point $M \in S_1$.

Adding the question x_{n+1} requires a transition to $(n + 1)$—tuple dimensional space of attributes and a requirement for linear separation of D.O$_1$ through

$$S_1 : \sum_{i=1}^{n+1} w_{1i} x_i + b_1 = 0, \quad \text{for } x_{n+1} = 0. \tag{6}$$

In a similar way for the plane $S_{2,}$ which divides linearly D.O$_2$ in the $(n + 1)$—tuple dimensional space of the attributes, we have:

$$S_2 : \sum_{i=1}^{n+1} w_{2i} x_i + b_2 = 0, \quad \text{for } x_{n+1} = 1 \tag{7}$$

where x_i are the coordinates of any point $M \in S_2$.

The domain for the threshold neuron we are looking for is:

$$\mathrm{D.O} = \mathrm{D.O}_1 \cup \mathrm{D.O}_2 \tag{8}$$

The neuron H that solves the two problems would exist only if this common D.O is linearly separable. However, given that D.O consists of points in the coordinate plane $Ox_1x_2 \ldots x_n x_{n+1}, as\ x_{n+1} = 0$ and points in its parallel $Ox_1x_2 \ldots x_n x_{n+1}, as\ x_{n+1} = 1$, from (6)–(8) it follows that for neurons H_1 and H_2 a very important condition must be imposed: The existence of parallel linear classifiers of the type (6) and (7), through which to build the sought plane dividing the common D.O. This condition means that it is necessary to have surfaces S_1 and $\mathrm{S}_{2,}$ with the representation (6) and (7), respectively, for which the condition of parallelism is fulfilled:

$$\frac{w_{11}}{w_{21}} = \frac{w_{12}}{w_{22}} = \ldots = \frac{w_{1n}}{w_{2n}}, b_1 \neq b_2 \tag{9}$$

allowing us to construct new planes separating D.O of the same class of parallel planes

$$S : \sum_{i=1}^{n+1} w_i x_i + b = 0, x_{n+1} = \lambda, \lambda \in R$$

In other words, from the existence of parallel surfaces S_1 and S_2, with the representation (6) and (7) follows the possibility of constructing a family of parallel planes containing S_1 and S_2 ($\lambda = 0\ or\ 1$) and linearly separating D.O.

Let us look at a specific example—the Boolean functions "AND" and "OR" with the truth tables given in Table 1. The domains of the two functions are linearly separable with respect to the pairs (X_1, X_2) returning "0" or "1". Let S_1 be one of the infinite sets separating $\mathrm{D.O}_1$ lines:

$$S_1 : 0.3\,x_1 + 0.2\,x_2 - 0.4 = 0.$$

Then there is a threshold neuron H_1, solving the function $f_1(x_1, x_2) = x_1 \wedge x_2$, with weights and threshold indicated by the coefficients of D_1 (Fig. 9). Similarly, if we look at the Boolean "OR" function and use the classifier:

$$S_2 : 0.75x_1 + 0.9x_2 - 0.3 = 0$$

Through it we can form a neuron $H_{2,}$ solving the function $f_2(x_1, x_2) = x_1 \vee x_2$.

So, we have two problems

$$f_1(x_1, x_2) = x_1 \wedge x_2 \text{ and } f_2(x_1, x_2) = x_1 \vee x_2,$$

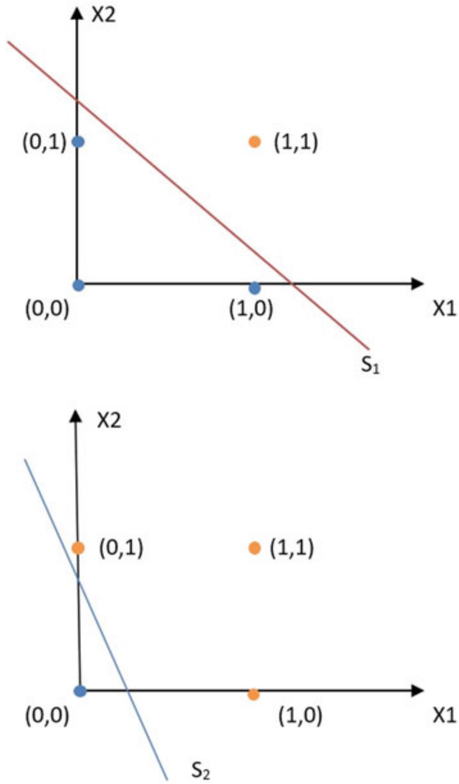


Fig. 9 Linear separability of D.O of Boolean functions "AND" and "OR"

which we can solve quite precisely through two different neurons. Our goal is to build a neuron H, which is capable of solving both problems (Fig. 10).

The classification lines S_1 and S_2 have angular coefficients, respectively, $k_1 = -\left(\frac{w_{11}}{w_{12}}\right) = -1.5$, and $k_2 = -\left(\frac{w_{21}}{w_{22}}\right) = -0.8333$. Obviously, they are not parallel. We cannot build a plane through them that divides in an appropriate way

$$\mathrm{D.O} = \mathrm{D.O}_1 \cup \ \mathrm{D.O}_2$$

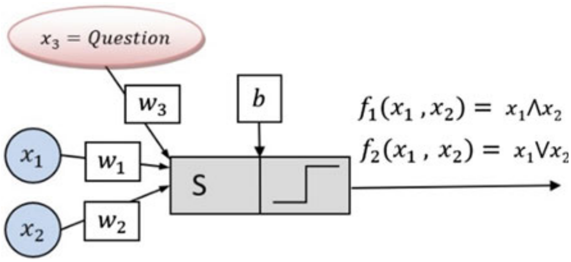


Fig. 10 Structure of a neuron that can solve two problems related to Boolean functions "AND" and "OR"

One possible solution is to keep looking for other linear classifiers for the two functions until we find parallel ones. Another solution is to use the condition of parallelism (9), which in our case has the following representation:

$$\frac{w_{11}}{w_{21}} = \frac{w_{12}}{w_{22}}, b_1 \neq b_2$$

Therefore, we can choose the linear classifier S_2, in such a way that

$$w_{22} = w_{21}\left(\frac{w_{12}}{w_{11}}\right) \tag{10}$$

So with the correction (10) of w_{22}, we have:

$$S_2 : 0.75\,x_1 + 0.5\,x_2 - 0.3 = 0$$

We can easily see that this line is still a linear classifier for the Boolean "OR" function, with an angular coefficient $k_2 = -\left(\frac{w_{21}}{w_{22}}\right) = -1.5$, i.e., $S_1 \| S_2$.

Let point M_1 and point $M_2 \in S_1$, while point M_3 and point $M_4 \in S_2$. The choice of these points does not matter much. It is enough to concretize them clearly, so that belonging to the two lines, through them to form the vectors with which we will construct a plane containing S_1 and S_2. Let:

$$point\ M_1\left(x_1 = 0.5, x_2 = \left((-1)\frac{\mathrm{w}_{11}}{\mathrm{w}_{12}}\right)\mathrm{x}_1 - \frac{\mathrm{b}_1}{\mathrm{w}_{12}} = 1.25, x_3 = 0\right)$$
$$point\ M_2\left(x_1 = -0.3, x_2 = \left((-1)\frac{\mathrm{w}_{11}}{\mathrm{w}_{12}}\right)\mathrm{x}_1 - \frac{\mathrm{b}_1}{\mathrm{w}_{12}} = 2.45, x_3 = 0\right)$$
$$point\ M_3\left(x_1 = 0.1, x_2 = \left((-1)\frac{\mathrm{w}_{21}}{\mathrm{w}_{22}}\right)\mathrm{x}_1 - \frac{\mathrm{b}_2}{\mathrm{w}_{22}} = 0.45, x_3 = 1\right)$$
$$point\ M_4\left(x_1 = 0.6, x_2 = \left((-1)\frac{\mathrm{w}_{21}}{\mathrm{w}_{22}}\right)\mathrm{x}_1 - \frac{\mathrm{b}_2}{\mathrm{w}_{22}} = -0.3, x_3 = 1\right)$$

Let us now form the vectors $\vec{p} = \overrightarrow{M_3M_1}$ and $\vec{q} = \overrightarrow{M_4M_2}$. We have

$$\vec{p}(p_1 = -0.4, p_2 = -0.8, p_3 = 1), \text{ and}$$
$$\vec{q}(q_1 = 0.9, q_2 = -2.75, q_3 = 1).$$

Then, if the plane S is defined by the vectors $\vec{p}$ and $\vec{q}$, and the point M_4, then

$$S : Ax_1 + Bx_2 + Cx_3 + D = 0,$$

as

$$A = p_2q_3 - p_3q_2, B = p_3q_1 - p_1q_3, C = p_1q_2 - p_2.$$

For the free member D, we have:

$$D = -Ax_1 - Bx_2 - Cx_3,$$

where (x_1, x_2, x_3) are the coordinates of point M_4.

Given the specific values of the coordinates of the vectors $\vec{p}$, $\vec{q}$ and the point M_4, it follows that we can construct a linear classifier of

$$\mathrm{D.O} = \mathrm{D.O}_1 \cup \mathrm{D.O}_2,$$

which has the representation:

$$S : 1.95x_1 + 1.3x_2 + 1.82x_3 - 2.6 = 0,$$

and the corresponding threshold neuron has the construction as shown in Fig. 11.

Let us recall that

$$\mathrm{Out} = g\left[\sum_{i=1}^{3} w_i x_i + b\right] \tag{11}$$

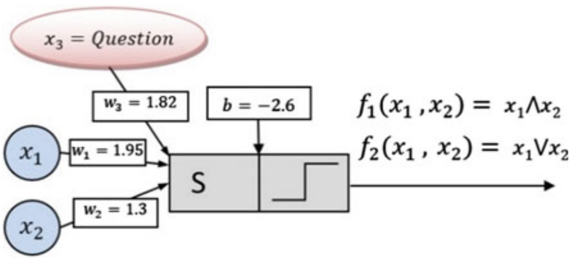
where $x_3 = 0$, if we want to ask a question to the neuron "How much is $f_1(x_1, x_2) = x_1 \wedge x_2$?", and $x_3 = 1$, to demand calculation of the value of $f_2(x_1, x_2) = x_1 \vee x_2$.

We will do a check with two specific examples. Let us pass the pair of logical variables $(x_1 = 0, x_2 = 1)$ at the input of the neuron and pose the question "How much is $f_1(x_1, x_2) = x_1 \wedge x_2$?". In the dendritic tree, we have input stimuli:

$$x_1 = 0, x_2 = 1, x_3 = 0.$$

Then, according to (11), a signal propagates along the axon of the neuron

Fig. 11 Threshold neuron corresponding to the found classifier S



$$\text{Out} = g\left[\sum_{i=1}^{3} w_i x_i + b\right] = g(-1.3) = 0$$

Now let us ask the question to calculate the value of $f_2(x_1, x_2) = x_1 \vee x_2$. At the same values of x_1 and x_2, at the input of the neuron, we have:

$$x_1 = 0, x_2 = 1, x_3 = 1.$$

and on the axon—a signal:

$$\text{Out} = g\left[\sum_{i=1}^{3} w_i x_i + b\right] = g(0.52) = 1$$

Other cases can be checked in a similar way.

4 Conclusion

Examining the linear separability of the data and the relation of the threshold neurons with the classifiers of the classes, we showed that it is possible to construct a neuron that can solve two different problems without the need for an intermediate change in its weights. But how effective and useful is a neuron that calculates several functions? On the one hand, the use of such a neuron saves the use of neural structures; therefore—memory, and on the other hand—this leads to an increase in the number of operations in the body of the neuron by two.

In conclusion, we should note that summarizing the presented ideas and results, we can find a neuron that solves a whole class of an infinitely number of problems with domains that are linearly separated from the found surface S. The only condition is that for each individual problem there is separability with linear classifiers which are parallel to each other.

Acknowledgements The work is partly funded by the MU21-FMI-004 project at the Research Fund of the University of Plovdiv "Paisii Hilendarski."

References

1. Gelig, A., Mateev, A.: Introduction to the Mathematical Theory of Learned Recognition Systems and Neural Networks. St. Petersburg State University (2014)
2. Bauckhage, C., Cremers, O.: Lecture Notes on Machine Learning: Linear Separability, University of Bonn, Bonn (2019)
3. Kowalczyk, A.: Support Vector Machines Succinctly, Syncfusion, Inc. (2017)
4. Picton, P.: Threshold logic: is there finally a solution? In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 45–51 (2016). https://doi.org/10.1109/IJCNN.2016.7727179
5. Elizondo, D.: The linear separability problem: some testing methods. IEEE Trans. Neural Netw. **17**(2) (2006)
6. El-Shahat, A.: Advanced Applications for Artificial Neural Networks. Georgia Southern University (2018)
7. Gorzałczany, M.: Essentials of Artificial Neural Networks. Physica, Heidelberg Online ISBN 978-3-7908-1801-7 (2002)
8. Gurney, K.: An Introduction to Neural Networks. Taylor & Francis e-Library, London and New York (2004)
9. Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M.: Threshold logic units. In: Computational Intelligence. Texts in Computer Science. Springer, London (2016). https://doi.org/10.1007/978-1-4471-7296-3
10. Minsky, M.L., Papert, S.A.: Perceptrons. MIT Press, Cambridge (1969)
11. Kirkpatricka, J., Pascanua, R., Rabinowitza, N., Veness, J.: Overcoming catastrophic forgetting in neural networks, DeepMind, N1C 4AG. United Kingdom, London (2017)
12. Yang, G., Joglekar, M., Song, H., Newsome, W., Wang, X.-J.: Task representations in neural networks trained to perform many cognitive tasks. Nat Neurosci **22**, 297–306 (2019)