This is the English translation of the paper. Please cite as:

Das stromlose Perzeptron – Stift-und-Papier-Einführung eines einfachen Algorithmus des maschinellen Lernens

Thomas Kraska, MNU Journal, Iss. 6, 491-498, (2024)

The unplugged perceptron – pen and paper introduction of a simple machine learning algorithm

Thomas Kraska, Institute for Physical Chemistry, Faculty of Mathematics and Natural Sciences, University Cologne, Greinstraße 4-6, 50939 Köln, Germany

A haptic algorithm for linear classification is presented for the introduction of a learning algorithm in secondary education. The approach leads to a program code in a few steps. It utilizes the fact that the learning parameters represent the elements of the normal vector of the dividing line between two types of objects. Based on the parameters, the dividing line can be constructed with a triangle ruler (Geodreieck) and the classification error can directly be read off. The method was developed and applied in a year 10 STEM course.

1 Introduction

Machine learning is an important part of artificial intelligence and has attracted a great deal of attention in recent years, particularly among schoolchildren, thanks to a number of everyday applications. Even before the introduction of ChatGPT, pupils were already using translation tools or spelling correction available on the internet. Individual suggestions, for example on streaming platforms, are also based on machine learning. However, the basic functionality remains hidden from the pupils. Understanding the functionality to a certain extent is on the other hand a prerequisite for assessing the capabilities of machine learning. Is such a program really intelligent in the human sense?

The principle of machine learning can be introduced with basic models. This allows to clarify questions such as how learning takes place, where the learned content is stored and on what the quality of the learned content depends. For such a simple model, a haptic algorithm for classifying two types of objects is proposed here, which can be converted into code by the pupils. Figure 1 shows a schematic example for such a system. The objects have two properties x and y. During the learning process, a dividing line is sought that separates the two types of objects in the coordinate system of the properties. Here, we restrict ourselves to dividing lines that run through the origin. In this case, the learning process optimizes just the slope of the dividing line.

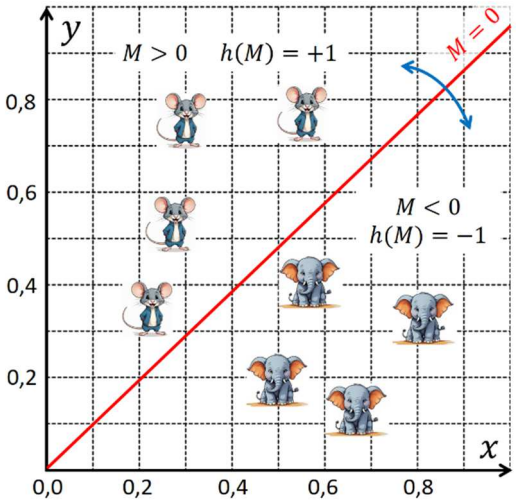


Fig. 1 A simple classification of two types of objects based on two predictor features x and y, i.e. object properties.

There are a number of studies that deal with the teaching of machine learning in school. MÜHLING and GROßE-BÖLTING (2023) show that pupils' perceptions of machine learning can be improved through unplugged activities. They conclude that pupils have no significant misconceptions about whether a computer is an intelligent being and conclude that it is not necessary to open the black box. While this may be sufficient with regard to such misconceptions and also ethical and social implications, it remains unclear how a computer actually learns. This only becomes clear if, in addition to an application-oriented approach, pupils are given the opportunity to work out a corresponding algorithm on their own as far as possible. Accordingly, other authors (STRECKER & MODROW 2019) advocate demystifying artificial intelligence (AI) through programming with Scratch. Issue 02/2024 of this journal (MNU Journal) contains a series of papers on application-oriented aspects of the use of AI and one paper on teaching the basic understanding of AI (MODROW 2024).

The impact of machine learning on classical computational thinking is discussed by SHAMIR and LEVIN (2022). The authors emphasize that machine learning does not follow the usual strategies for problem solving, but that the algorithms learn from data. This means that the data flows in machine learning and deep learning cannot be followed transparently as with rule based algorithms. Rule based algorithms convert a scientific problem or a model into a deterministic code, i.e. the data flow is traceable in the code. The authors conclude that the concept of computational thinking must be extended to include the handling and evaluation of data.

MICHEUZ (2020) also argues that AI and, in this context, data science should be part of modern school education, as AI is often accompanied by misleading statements that range from “utopian enthusiasm to dystopian fears”. To counteract this, MICHEUZ calls for a well-founded explanation of how AI works, including a background in programming language.

YIM and SU (2024) come to a similar conclusion in a review of 46 studies on the question of how AI should be taught at secondary school level. They conclude that, in addition to dealing with application programs, programming with Scratch or Python can be used to demystify AI principles.

STRECKER (2020) has already proposed an approach for implementing small neural networks in school that focuses on programming. STRECKERs intention is to avoid a black box and let pupils program themselves. Among other things, the aim is to show that these are algorithms that do not develop consciousness.

The work here is limited to an algorithm for a single neuron. A method is presented with which pupils can playfully develop a learning algorithm in the context of computational thinking and create a compact program code for it. This makes it at a very basic level clear how machine learning works, where the information is stored and to what extent this has to do with intelligence in the human sense. One aim is to allow pupils in other subjects to develop and apply a learning algorithm for their subject-specific contexts and thus to convey a fundamental impression of machine learning in the context of these applications. For this reason, the core of the algorithm is not embedded in higher-level structures such as object orientation, as this represents an avoidable overhead especially if applied in other subjects.

2 The perceptron

The simplest system for machine learning is the perceptron (ROSENBLATT 1958), which represents a single neuron. Figure 2 shows the data processing for a perceptron. The forward calculation is used to determine the type of an object. This requires the properties of the objects x and y as well as parameters a and b, which are the weights of these properties for the prediction of the type of object. In addition, a so-called bias or threshold value is required. It has a fixed input value of 1 and an adjustable weight c. The variables of the properties are then multiplied by their weights and added up together with the bias. In analogy to biological neurons, this is referred to here as the membrane potential M.

$$M(x_i, y_i) = a \cdot x_i + b \cdot y_i + c \cdot 1$$

The result for M is a continuous number that is converted into a discrete result by an activation function $h(M)$:

$$r = h(M) = \begin{cases} -1, & M < 0 \\ +1, & M \geq 0 \end{cases}$$

For $r = -1$ one type of object is present (elephant in Figure 1), for $r = +1$ the other type is present (mouse in Figure 1). Consequently, $M = 0$ applies to the dividing line. The parameters must now be selected in such a way that the type of object can be correctly predicted for known properties x, y. To do this, it is necessary to have a data set available to train the system. This is known as *supervised learning*, in which the system learns on the basis of known

classifications. A learning algorithm is based on a loss function that reflects the deviation between the calculated and the actual classifications of the training data set. A frequently used loss function is the quadratic deviation, which, strictly speaking, can only be applied to normally distributed data. If the activation function has a non-zero slope and is differentiable, the parameter corrections can be obtained using the gradient method. As these conditions are not met with the perceptron, the so-called delta rules to correct the learning parameters are introduced directly:

$$a \leftarrow a - L\, e_i\, x_i \qquad b \leftarrow b - L\, e_i\, y_i \qquad c \leftarrow c - L\, e_i$$

Here e_i is the error of a data point $e_i = r_i - s_i$ with the calculated $r_i = h(M(x_i, y_i)) = \pm 1$ and the actual value $s_i = \pm 1$ for the type of an object. In addition, a learning rate L is introduced. The value for L is between zero and one and is intended to prevent the parameters from making too large jumps and possibly jumping over a solution.

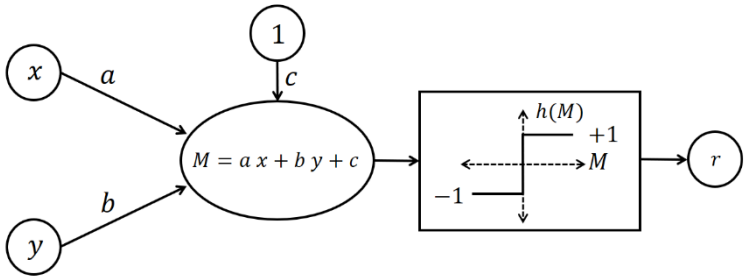


Fig. 2 Schematic representation of a perceptron.

3 Method

A detailed implementation in class and all necessary graphics are provided in the online supplements. The pupils first create their own data set of two types of objects with two properties that can be specified using numbers. The properties are divided by their maximum values to obtain numbers between 0 and 1 for both properties. The data are plotted using GeoGebra (GeoGebra classic 2024) to obtain a diagram similar to Figure 1. The learning algorithm is then introduced using a haptic method with a sheet of paper, a pencil and a triangle ruler (Geodreieck). The perceptron is simplified by omitting the bias. For this purpose, the weight c in the calculation of M and the corresponding input are deleted in Figure 2. It follows that only data sets whose dividing line runs through the origin can be classified. The equation for the membrane potential can be understood as the coordinate form of a straight line in a two-dimensional coordinate system, which can be written as a dot product:

$$M = a \cdot x + b \cdot y = \begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \vec{n} \cdot \vec{u}$$

The condition for the dividing line is $M = 0$. It follows that $\vec{n} = \begin{pmatrix} a \\ b \end{pmatrix}$ is a normal vector to $\vec{u} = \begin{pmatrix} x \\ y \end{pmatrix}$. As the bias is neglected, the position vector of the dividing line is the zero vector and the dividing line lies on the direction vector $\vec{u}$. The normal vector $\vec{n}$ is perpendicular to $\vec{u}$. It is placed at the origin. However, the straight line $M = 0$ can also be rewritten as a linear function with the slope $m = -a/b$. The meaning of the slope is known to pupils and they can use it to directly deduce the course of the dividing line. For the haptic implementation of the learning algorithm, a two-part coordinate system is required as shown in Figure 3. The two objects at their coordinates are entered in the right-hand part of the diagram. The values of the learning parameters are plotted in the left-hand part of the diagram. The connection between the two sub-diagrams results from the above-mentioned orthogonality and can be established practically with a triangle ruler (Geodreieck). The right-angled tip of the triangle ruler is placed at the origin. If the left side of the triangle ruler is placed at the point for the parameter set, one can draw the corresponding dividing line on the other side of the triangle ruler and read off the error directly there. While this mathematical background can be discussed as an application example in upper secondary school mathematics lessons, it was not discussed in a year 10 STEM course.

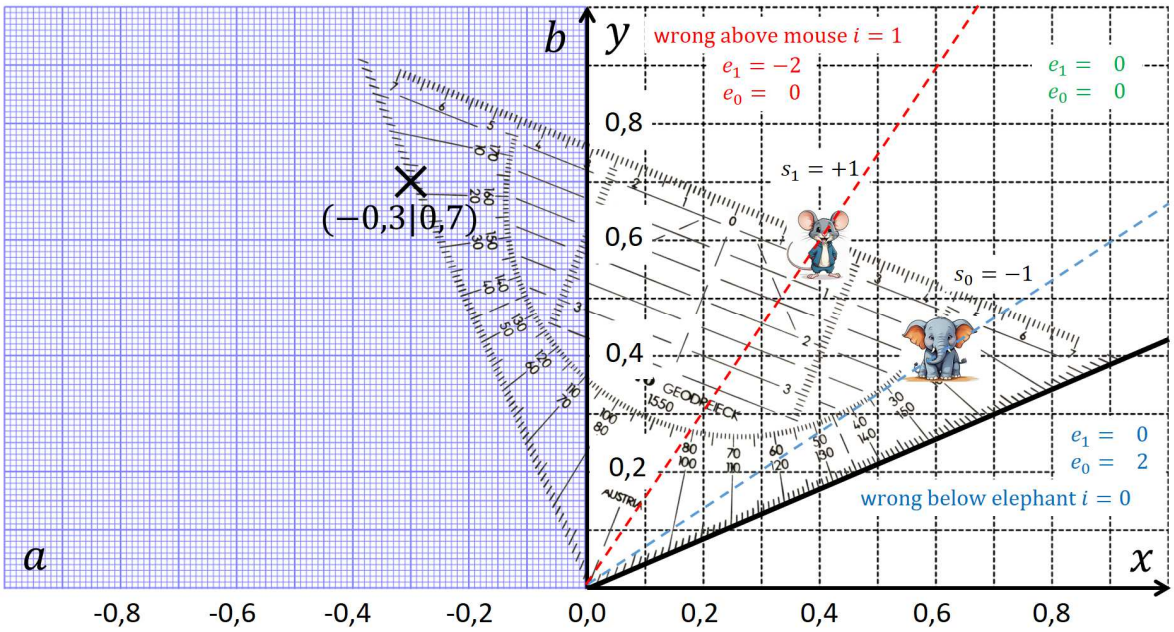


Fig. 3: Section of the worksheet with the dividing line (bold) on the right and the learning parameters on the left (cross), here at $a = -0.3$ and $b = 0.7$. The triangle ruler (Geodreieck) establishes the connection. If the dividing line lies in the area above the red line, then the errors written in red apply. If the dividing line lies below the blue straight line, the errors written in blue apply.

4 Implementation

In order to introduce the delta rule empirically, we first simply subtract the errors from the parameters without addressing the weighting by the input data x_i and y_i and the learning rate L. The sequence of the haptic learning algorithm is then as follows:

A) The starting values $a = 0$, $b = 1$ are entered as coordinate points in the left-hand diagram. The tip of the right angle of the triangle ruler (Fig. 3) is placed on the origin. Here it follows that the dividing line lies exactly on the x-axis in the positive range.

B) In the diagram on the right, the relative position between the object (elephant) and the dividing line is checked and the error is read off. As it lies wrongly under the elephant, the error is $e_0 = 2$.

C) In the **test field** of learning protocol 1 (Fig. 4) "$e_0 = 2$" and "line wrong below elephant" are entered.

D) The new values for a and b as well as the slope $m = -a/b$ are calculated in the right-hand column **parameters**. The error e_i is transferred to the brackets in the right-hand column and the new values for a and b are calculated from their current values.

E) The same procedure is carried out with the new parameters a and b. Step A) now follows for the mouse. In the following, the calculation is carried out alternating for the objects (Fig. 4).

However, the first run in Figure 4 does not converge and the dividing line alternates between two positions that do not represent a solution. Based on this knowledge, the weighting of the error e_i with the input data x_i and y_i can be worked out. If, for example, the input value x_i is small, then the correction term must also be correspondingly small. Instead of $a \leftarrow a - e_i$, we therefore use $a \leftarrow a - e_i \cdot x_i$ and $b \leftarrow b - e_i \cdot y_i$. The procedure is the same as described above, but the errors e_i have to be multiplied by x_i or y_i. The values are entered in learning protocol 2 (Figure 5).

The question may arise as to why the error e_i is subtracted and not added when correcting the weights. If the dividing line (Fig. 3) lies wrongly below the elephant, then its slope $m = -a/b$ must be increased, i.e. the negative a must become more negative and/or the positive b must become smaller. If the dividing line is below the elephant, then it is $e_0 = +2$ and a reduction of b ($b > 0$) is only possible if e_0 is subtracted. Since a is negative, subtracting e_0 leads to an even more negative value ($b \leftarrow b - e_0, a \leftarrow a - e_0$). In both cases, subtracting e_0 leads to an increase in the slope of the separation line, which is necessary to get closer to a correct classification. The analogous consideration for a separation line that lies above the mouse also leads to the conclusion that the error $e_1 = -2$ must be subtracted in order to obtain improved parameters.

<table>
<tr><th colspan="4">learning protocol 1</th></tr>
<tr><td colspan="3"></td><td>parameters
$a - e \rightarrow a$
$b - e \rightarrow b$
$m = -\frac{a}{b}$</td></tr>
<tr><td colspan="2">epochs and data</td><td>test field, is the location of the dividing line correct?</td><td>initial values
$\boldsymbol{a = 0} \quad \boldsymbol{b = 1}$
$m = 0$</td></tr>
<tr><td rowspan="2">0</td><td>  </td><td>$i = 0$
$s_0 = -1$
$x_0 = 0{,}6$
$y_0 = 0{,}4$</td><td>$e_0 = 2$
line wrong below elephant</td><td>$0 - (2) \rightarrow \boldsymbol{a = -2}$
$1 - (2) \rightarrow \boldsymbol{b = -1}$
$m = -2$</td></tr>
<tr><td>  </td><td>$i = 1$
$s_1 = +1$
$x_1 = 0{,}4$
$y_1 = 0{,}6$</td><td>$e_1 = -2$
line wrong above mouse</td><td>$-2 - (-2) \rightarrow a = 0$
$-1 - (-2) \rightarrow b = 1$
$m = 0$</td></tr>
</table>

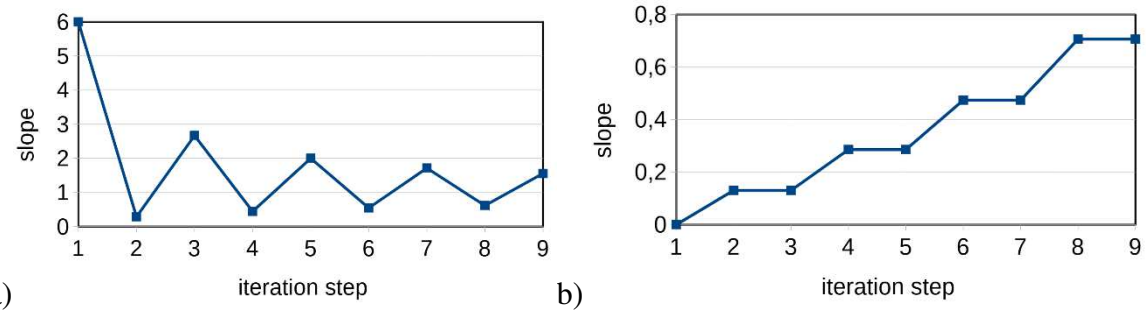
Fig. 4: Completed learning protocol for a learning algorithm without weighting with the input data x_i, y_i. A format DIN A3 diagram and an unfilled learning protocol are available in the online supplements. In this example, the initial values are returned after one run. An epoch is a single run through a data set, in this case once elephant and once mouse.

In the third execution of the learning algorithm, the learning rate L can be included. In learning protocol 3 (online supplements), $L = 0.1$ is used. The multiple execution of this pen and paper algorithm serves the purpose of repetition. Also, the first version is greatly simplified so that pupils can recognize the principle more easily than if they were to start with the last version.

Figure 6 shows the development of the slope of the separation line for $L = 1$ and $L = 0.1$. For $L = 1$, the jumps in the parameters are so large that the separation line approaches the solution alternately. For $L = 0.1$, the separation line approaches the solution from one side in small steps. Figure 6b also shows that every second step does not lead to any change, as in this case the mouse is always on the correct side of the separation line.

learning protocol 2				
				parameters $\boldsymbol{a = 0 \quad b = 1}$ $m = 0$
epochs and data			**test field**, is the location of the dividing line correct?	$a - e \cdot x_i \rightarrow a$ $b - e \cdot y_i \rightarrow b$ $m = -a/b$
0		$i = 0$ $\boldsymbol{s_0 = -1}$ $x_0 = 0{,}6$ $y_0 = 0{,}4$	$\boldsymbol{e_0 = 2}$ line wrong below elephant	$0 - (\mathbf{2}) \cdot 0{,}6 \rightarrow \boldsymbol{a = -1{,}2}$ $1 - (\mathbf{2}) \cdot 0{,}4 \rightarrow \boldsymbol{b = 0{,}2}$ $m = 6$
		$i = 1$ $\boldsymbol{s_1 = +1}$ $x_1 = 0{,}4$ $y_1 = 0{,}6$	$e_1 = -2$ line wrong above mouse	$-1{,}2 - (-2) \cdot 0{,}4 \rightarrow \boldsymbol{a = -0{,}4}$ $0{,}2 - (-2) \cdot 0{,}6 \rightarrow \boldsymbol{b = 1{,}4}$ $m = 0{,}286$
1		$i = 0$ $s_0 = -1$ $x_0 = 0{,}6$ $y_0 = 0{,}4$	$e_0 = 2$ line wrong below elephant	$-0{,}4 - 2 \cdot 0{,}6 \rightarrow \boldsymbol{a = -1{,}6}$ $1{,}4 - 2 \cdot 0{,}4 \rightarrow \boldsymbol{b = 0{,}6}$ $m = 2{,}667$
		$i = 1$ $s_1 = +1$ $x_1 = 0{,}4$ $y_1 = 0{,}6$	$e_1 = -2$ line wrong above mouse	$-1{,}6 - (-2) \cdot 0{,}4 \rightarrow \boldsymbol{a = -0{,}8}$ $0{,}6 - (-2) \cdot 0{,}6 \rightarrow \boldsymbol{b = 1{,}8}$ $m = 0{,}444$
	…	…	…	…

Fig. 5: Completed learning protocol with weighting with the input data for the first two epochs. In the online supplements, a fully completed and an unfilled version are available.



a) b)

Fig. 6: Progression of the slope of the dividing line according to learning protocol 2 with a) $L = 1$ and b) $L = 0.1$.

After running through the algorithm several times on paper, the pupils can create a pseudo-code, i.e. a list of steps in a generally understandable language. It can be structured or formulated as text. Structure diagrams can be used as an alternative or as an additional step. While the loops in the data flow diagram in Figure 7a are implemented using conditional statements, the Nassi-Shneidermann diagram in Figure 7b provides a somewhat clearer representation of the loops. The last step is the implementation in a specific program code. Figure 8 shows an example for the TigerJython interpreter suitable in school (ARNOLD et al. 2024). This program and additional variants with graphical output are available in the online supplements.

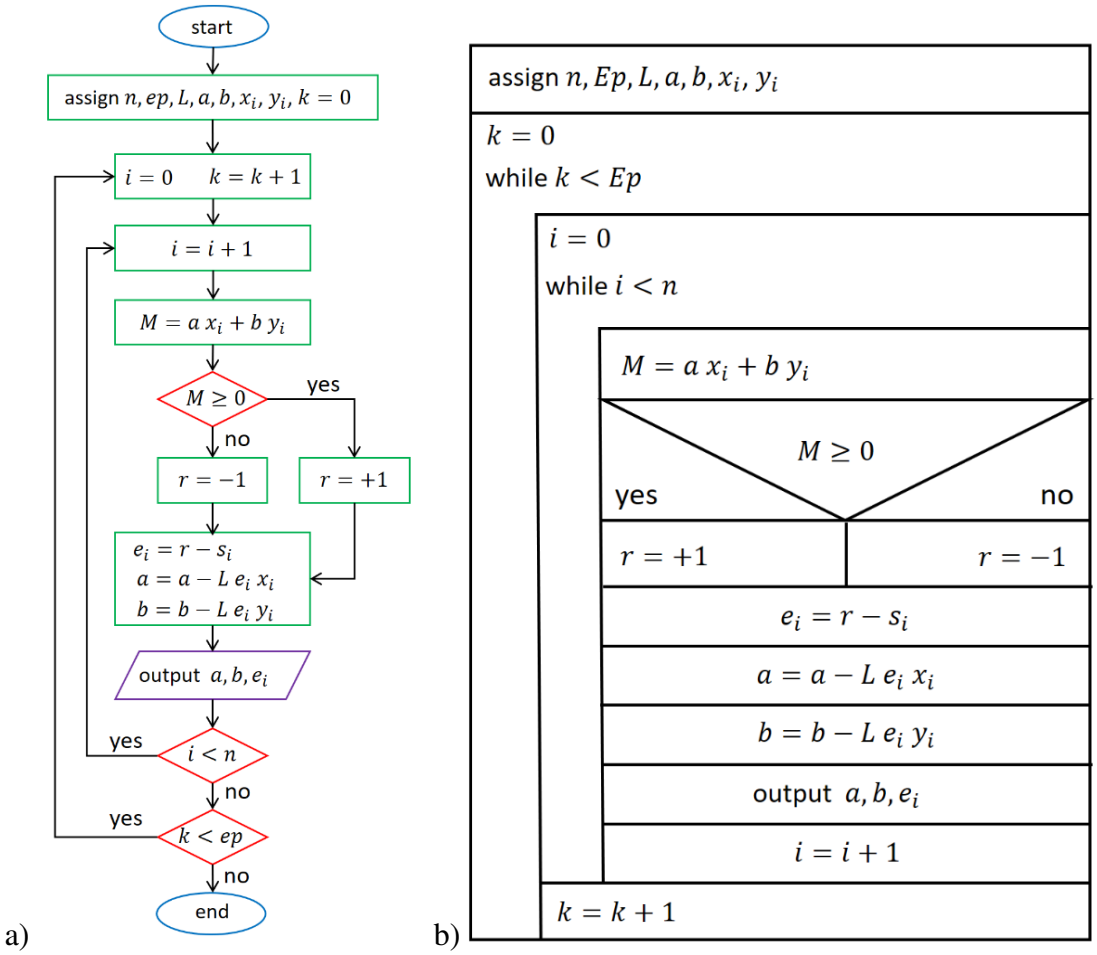


Fig. 7. a) Data flow diagram and b) Nassi-Shneidermann diagram of the learning algorithm. Ep: number of epochs, n: number of objects in the training dataset. Empty diagrams are available in the online supplements.

5 The Adaline algorithm

The method described here can also be applied to the Adaline algorithm (WIDROW & HOFF 1959). Adaline is derived from "adaptive linear neuron" and uses a linear activation function $h(M) = M$, which mathematically means that there is no activation function. The difference to the perceptron is that the output consists of continuous numbers and not two discrete values, which has an effect on the learning process. At the end, however, the types of objects are again determined by the sign of the output. In contrast to the perceptron, the iteration does not end as soon as all objects are correctly classified, but continues until a minimum in the mean square deviation between the output of M and the correct values s_i is reached. The Adaline algorithm can be implemented with a small change. To do this, delete lines 18 and 19 in the code in Figure 8 and insert the line `r=M` instead.

```
1  L      = 1
2  epochs = 20
3
4  # data
5  x = [ 0.6, 0.4]
6  y = [ 0.4, 0.6]
7  s = [  -1, 1  ]
8  ndat = len(x)
9
10 # initial values
11 a =  0.0
12 b =  1.0
13
14 # learning algorithm
15 for k in range(epochs):
16     for i in range(ndat):
17         M = a * x[i] + b * y[i]
18         if M>0: r= 1
19         else:   r=-1
20         e = r - s[i]
21         a = a - L * e * x[i]
22         b = b - L * e * y[i]
23         slope = -a/b
24         print i,k,round(a,4),round(b,4),slope,M
```

Fig. 8 Program code for the learning algorithm.

6 Observations

When starting to create their own data sets, the pupils were committed to contributing their own areas of interest. Structuring their data with a provided table (see online supplements) is helpful especially concerning the scaling of the data by dividing by its maximum value. The diagram as shown in Figure 1 with GeoGebra (GeoGebra classic 2024) was then created without any problems.

In the discussion above, individual pupils independently came up with the idea that the parameters must be corrected with the error. It was then necessary to provide Figure 3 to explain the haptic algorithm. In addition, a diagram in which the three areas for the error e_i in Figure 3 are highlighted in color (online supplements) was made available. This enabled the pupils to draw the dividing line based on the parameters, read off the errors and calculate the new values for the parameters in the learning protocol.

During the calculations in the learning protocols, some pupils made careless mistakes when subtracting negative numbers such as $-1.2-(-2)\cdot 0.4$. Since such errors are propagated during an iteration, it makes sense to point out such errors in advance.

After the pupils recognized the alternating behavior of the parameters in the first run, they suggested reducing the error value, e.g. by multiplying it by a small number, during the discussion. In doing so, they anticipated the idea of a learning parameter L. First, however, the multiplication of the errors with the input values was taken into account. The algorithm was understandable to the pupils after completing learning protocol 2, so that protocol 3 was not necessary. Finally, they wrote down the algorithm as pseudocode. As the pupils already had experience with structure diagrams, they were able to convert the pseudo code into suitable, empty diagrams. Alternatively, it is also possible to provide such diagrams filled out and let the pupils establish the connection to the pen and paper algorithm. The final step was the conversion into specific code. This was then used to check the manual calculation for learning protocol 2. Finally, the learning rule for the bias was added, where the input value is always 1 and therefore the multiplication with the input value is practically omitted. A program with graphical output was provided for the calculation with the pupils' own data sets. The pupils entered their data in lists in this program. It can be summarized that the paper algorithm was successfully carried out for pupils in year 10. The pupils were able to understand the algorithm well and recognize how machine learning works for this example.

Acknowledgments

The author would like to thank GEOTEC Schul- und Bürowaren GmbH for providing an image file for the triangle ruler (Geodreieck). The images for the mouse and the elephant were generated with the AI image generator https://app.leonardo.ai/.

Literature

ARNOLD, J., KOHN, T., KOMM, D. & ROTH, N. (2024). Programming environment TigerJython, https://www.tigerjython.ch/

GeoGebra classic (2024). Geometry and Algebra program. https://www.geogebra.org/classic

MICHEUZ, P. (2020). Anmerkungen zur Künstlichen Intelligenz als Thema im Schulunterricht, Bildung und Digitalisierung, 1(4) 271-292.

MODROW, E. (2024). Datenkompression und Maschinelles Lernen mit SciSnap! *MNU-Journal*, 77, 111-118

MÜHLING, A. & GROßE-BÖLTING, G. (2023). Novices' conceptions of machine learning, *Computers and Education: Artificial Intelligence*, Volume 4, 100142

ROSENBLATT, F. (1958). The Perceptron: a probabilistic model for information storage and organization in the brain. *Psych. Rev., 65,* 386-408.

SHAMIR, G. & LEVIN, I. (2022). Teaching machine learning in elementary school, *Int. J. Child-Comp. Interaction, 31*, 100415.

STRECKER, K. (2020) Ein kleines Neuronales Netz selbst programmieren *MNU-Journal,* 73, 92-96

STRECKER, K. & MODROW, E. (2019). Eine Unterrichtssequenz zum Einstieg in Konzepte des maschinellen Lernens, in Arno Pasternak (Hrsg.): Informatik für alle. *Lecture Notes in Informatics (LNI)*, Gesellschaft für Informatik, Bonn 2019, S. 335-344

WIDROW, B., & HOFF, M. E. (1960). Adaptive switching circuits. *IRE WESCON convention record, 4*, 96-104.

YIM, I.H.Y. & SU, J. (2024). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. *J. Comput. Educ.* (https://doi.org/10.1007/s40692-023-00304-9)