TU WIEN Informatics

Machine Learning Unplugged

Training Decision Trees and Artificial Neural Networks with Children

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Lukas Lehner, BSc
Matrikelnummer 01126793

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Gerald Futschek

Wien, 4. Mai 2023

Lukas Lehner	Gerald Futschek

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Machine Learning Unplugged

Training Decision Trees and Artificial Neural Networks with Children

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Lukas Lehner, BSc
Registration Number 01126793

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Gerald Futschek

Vienna, 4th May, 2023

Lukas Lehner	Gerald Futschek

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Lukas Lehner, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. Mai 2023

Lukas Lehner

Acknowledgements

Thank you to Gerald Futschek for taking me under his wing a year ago and inspiring my newly found passion for informatics didactics.

Thank you, Martina, for always having an open ear, good advice and coffee.

Thank you to my parents, for their unending love and support has gotten me this far.

And thank you, Alex, for baking cake, cutting cards and loving me back!

Kurzfassung

Künstliche Intelligenz (KI) und insbesondere Systeme die auf maschinellem Lernen (ML) basieren beeinflussen das Leben vieler junger Menschen - oftmals ohne deren Wissen. Sie interagieren mit diesen Systemen und spüren deren Auswirkungen, sei es bei der Nutzung von sozialen Medien, Internetsuche oder beim Schreiben ihrer Hausaufgaben. Jedoch, wie diese Systeme genau funktionieren - und warum manchmal auch nicht - wird von denen, die diese Systeme anbieten nur selten auf eine Art und Weise erklärt, wie sie Kinder oder Nicht-Expert_innen verstehen würden. Eine Technologie, die unseren Alltag prägt, nicht zu verstehen kann zwar Neugierde auslösen, im schlimmsten Fall aber auch Unbehagen und Angst.

Wege zu finden, wie man grundlegende Prinzipien einer komplexen Materie wie dem maschinellen Lernen auf ansprechende Weise vermitteln kann - insbesondere gegenüber Kindern und Nicht-Expert_innen -, kann dazu beitragen, Missverständnisse aufzulösen und KI-Kompetenz zu fördern. "Unplugged"-Aktivitäten, also Lernaktivitäten, die nicht auf Computer angewiesen sind, sind zu diesem Zweck gut geeignet. Eine Vielfalt an Unplugged-Aktivitäten zu ML wurden in den letzten Jahren präsentiert. Viele davon gehen jedoch nicht darauf ein, wie ein ML-Modell "aus Beispielen lernt", also trainiert wird. Einige behandeln das Thema zwar, aber bieten dabei eine so vereinfachte Sichtweise auf das Konzept, dass Schüler_innen falsche Schlüsse darüber ziehen könnten, wie eine Maschine tatsächlich lernt. Zum Beispiel kann das Zuweisen einer speziellen und komplexen Aufgabe an ein einzelnes Neuron in einem künstlichen neuronalen Netzwerk dazu führen, dass Schüler_innen glauben, die Neuronen seien intelligenter als sie tatsächlich sind. Und, dass der Trainings-Prozess eines Netzwerks darin besteht, dass ein Mensch die Funktionsweise jedes Neurons von Hand verliest (z. B. "ein Lächeln erkennen"). Diese Art des Vereinfachens mag hilfreich sein, um gewisse komplexe Konzepte zu vermitteln (z. B. was ein *Generator* und *Discriminator* in einem *Generative Adversarial Network* zu tun haben), aber sie schafft es nicht, Schüler_innen erkennen zu lassen, wie die Kombination aus einfachen Regeln und Rechenoperationen intelligent erscheinendes Verhalten hervorbringen kann.

Diese Arbeit untersucht daher, wie der Trainings-Prozess eines ML-Modells mit Hilfe einer Unplugged-Aktivität verständlich gemacht werden kann, ohne dieses Konzept so stark zu vereinfachen, dass Schüler_innen einen falschen Eindruck davon bekommen, wie eine Maschine tatsächlich aus Beispielen lernt. Dazu wurden zwei Unplugged-Aktivitäten

mit geringem Abstraktionsgrad zu den Trainingsaspekten von künstlichen neuronalen Netzwerken und Entscheidungsbäumen entworfen, entwickelt und erprobt und werden in dieser Arbeit vorgestellt. Diese sollen es Lehrpersonen ermöglichen jungen Menschen einen wichtigen Einblick in eine Technologie zu geben, die ihr tägliches Leben prägt.

Abstract

Artificial intelligence (AI) and specifically machine learning (ML) based systems influence many modern young peoples' lifes, often without their knowledge. They interact with and experience the effects of these systems when using social media, web search or writing their homework. However, how these systems work - and what can make them fail - is rarely explained by the providers of these systems in a way a child or a non-expert may understand. Not understanding a technology that shapes our everyday lives may spark wonder and curiosity, in the worst case, however, may cause resentment and fear.

Finding engaging ways to convey the basic principles of a complex subject matter such as machine learning - especially to children and non-experts - may help combat this lack of understanding and foster AI literacy. "Unplugged" activities, i.e. learning activities that do not rely on the use of a computer, are well suited to this task. Several such unplugged activities on ML have been proposed in recent years. Many of them, however, do not discuss how a ML model "learns from examples", i.e. is trained, at all. Some cover training but provide such a simplified view on the concept that students may draw wrong conclusions about how a model actually learns. For example, assigning specific and complex tasks to individual neurons in an artificial neural network may lead students to believe neurons are "smarter" than they actually are and that the process of training a network involves humans hand picking every neuron's job (e.g., "detect a smile"). This strategy of simplification may work to convey high level concepts (e.g., the roles of the generator and discriminator in a generative adversarial network), but does little to make student realise how a combination of simple rules and mathematical operations can produce intelligent seeming behaviour.

Thus, this thesis investigates how the process of training a machine learning model can be taught using an unplugged activity, without oversimplifying the concept to a point of abstraction that may leave students under a false impression of how a machine learns from examples. To this end, two low-abstraction hands-on unplugged activities on the training aspects of artificial neural networks and decision trees have been designed, developed and piloted and are presented in this thesis. They are intended to be used by educators to give young people insight into a technology that shapes their everyday lives.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Objectives

Whether they use artificial intelligence (AI) consciously (e.g., to write their homework) or are influenced by an AIs decision without being aware of it (e.g., while using a social media platform), AI based systems seem to be part of many modern young peoples' daily life. When the idea for this thesis was conceived in May of 2022, *ChatGPT*[1] had not yet been introduced to the public. Since then the prevalence of artificial intelligence in the public mindset seems to have multiplied manifold. It seems a discussion on how this newly available technology will influence society, but especially class rooms both in K12 education and academia is being had in many institutions, with teachers fearing that all their future homework assignments will be done by AI.

Bringing this subject into classrooms in a way that is understandable seems now even more necessary than before and could have a positive impact on such conversations. Unplugged activities can facilitate such an impact. They provide a low level of entry to complex subject matters by requiring little to no prior knowledge (of both instructor and student) about the subject they cover. Such activities may be provided as a ready made product to be used by a teacher in school, such as the seminal work *Computer Science Unplugged: Off-line Activities and Games for All Ages* by Bell, Witten and Fellows [BWF98], which has shaped its field since its release in 1998. Alternatively, workshops utilising unplugged activities may be offered by third party providers and outreach programs such as the one run by the *TU Wien Informatics Didactics eduLAB*[2]. Working for *eduLAB* over the past year let the author experience first hand how unplugged activities allow topics such as *sweep line algorithms* or *binary encoding* to be taught in a fun and engaging way - even to elementary school children.

[1] https://chat.openai.com/
[2] https://informatics-didactics.ifs.tuwien.ac.at/

As will be discussed in Chapter 2, many available unplugged activities on various machine learning (ML) concepts fall short of delivering a clear cut view of how exactly a machine can "learn on its own". In the worst case, this may leave students under a false impression regarding how these techniques work and, especially, what can make these systems fail.

For these reasons the aim of this thesis is to answer the following research question: *How can the process of training a machine learning model be taught using an unplugged activity, without oversimplifying the concept to a point of abstraction that may leave students under a false impression of how a machine learns from examples?*

The aim is to answer this question by designing, developing and piloting two unplugged activities on two different machine learning topics: artificial neural networks (ANN) and decision tree learning (DTL).

The objectives for *each* machine learning topic are as follows:

1. Determine how training the ML model can be simplified enough to be feasible in an unplugged setting without abstracting the process so much, that students may draw false conclusions as to what the training process actually looks like.
2. Based on the previous findings, design an unplugged activity that - without requiring any prior knowledge on AI - teaches how an ML model is trained.
3. Develop all needed materials and instructions for the activity.
4. Pilot the activity with students to determine whether all instructions and materials are understood and can be used as intended.

1.2 Outline

This thesis is structured as follows: Chapter 2 discusses existing approaches, i.e., unplugged activities on machine learning subjects. In Chapter 3 and 4 the design and development process of two unplugged activities are discussed and the activities are presented. Chapter 5 evaluates the activities' unplugged "distinctive features", discusses their piloting, drawbacks and future work. Chapter 6 summarises the results.

CHAPTER 2

State of the Art

Unplugged activities have been used in computer science education to teach various complex subject matters. With the rise in popularity of the subject, several unplugged activities on various concepts of artificial intelligence have also been proposed in recent years.

Clarke [Cla19] presents an extensive "alternate curriculum unit" on AI providing student activities (mainly discussions, reflections and worksheets) and resources (presentations, articles and videos) on a wide variety of AI related subjects. The curriculum includes an unplugged activity on forward propagation in artificial neural networks including video instructions[1]. The ANN in this activity can classify input images into "happy face", "sad face", and "no face". The network's neurons are played by students who are arranged in layers. Each student of the first layer is told to activate (shout "bingo") when seeing a specific feature in the input image (e.g. a smiling mouth or a nose). Students of following layers are instructed to activate on specific combinations of features activated in the previous layer. While such an activity may demonstrate the flow of information through a neural network well, it gives students the false impression that a neuron in an artificial (or biological) neural network has the ability to do a complex task such as detecting a "sad mouth". The author goes as far as calling the neurons "feature detectors", which further suggests an individual neuron to be more capable than it is in a biological neural network or even an ANN on a computer. Additionally, students may come to the conclusion that the neural networks they hear about on the news are networks such as the one presented to them during the activity, where each neuron is handed a specific job by a human instructor or programmer. Such a conclusion would not just be wrong, but also inhibit a students ability to understand why ANNs are considered "black boxes" and why wrong decisions made by an ANN can often not be explained easily. The image recognition activity by Lindner et al. (Activity 2 in [LSR19])

[1] https://www.youtube.com/watch?v=q8GwXSQ-QWQ

takes a similar approach, but the features whose presence the neurons (i.e., students) detect are less complex ("rectangular shape", "triangular shape" or "round shape").

Another unplugged activity on forward propagation is *Brain-in-a-bag* by McOwan et al. [MC14]. This activity also has students act as neurons. They are arranged in three layers (input, hidden, and output) which are connected using ropes with an empty toilet paper roll on them. Thus, the neurons can communicate by lifting the rope and sending the roll to the other end. The instructions given to each neuron are less complex than in [Cla19]: The input neurons fire when detecting one of two colors of a card, the others fire when receiving one or two signals (i.e., toilet paper rolls) from the previous layer. The network is setup to play a simple game which is then played through several times. After the activity students are told that this network is an abstraction of how a biological neural network works and that in reality the rules for when to fire are learned by the network itself, not given to it by a human like in the activity. Thus, while the activity itself uses abstraction to teach various concepts, students are not left under any false impression about how a neural network functions. This activity is discussed further in Chapter 4 as is serves as basis for the ANN activity presented in this thesis.

While [Cla19] and [MC14] do not include training ANNs in their activity, the GANs Unplugged Activity by Patrick Virtue [Vir21] does. Virtue has students act as generators and discriminators of a generative adversarial network. Over the course of the activity the GAN learns to detect "fake" drawings from "real" ones. Fake drawings in this context are drawings that do not depict a previously agreed upon topic (e.g., "Dinosaurs"). This activity shows how a GAN works by demonstrating the function of the generator and discriminator. However, as a stand-alone activity and without students having prior knowledge on ANNs, students will be left to wonder how the generator and discriminator themselves work, as it seems there is no discussion during or after the activity.

Regarding decision tree learning Lindner et al. provide an introduction to the basic concepts involved in decision tree classification with their *The Good-Monkey-Bad-Monkey Game* unplugged activity [LSR19]. The activity provides students with illustrations of monkeys' faces labeled either "biting" or "non-biting". In a first step students get to discover for themselves what facial features (e.g., "bared teeth", "eyes wide open") differentiates one monkey face from another. Students then form groups to "develop criteria for distinguishing biting from non-biting monkeys" which are (optionally) represented as a decision tree and are subsequently tested with new, unseen monkey faces. The activity teaches the basic elements of a decision tree, how to construct one from training data and test it with unseen data. Since the decision making during training is left to the students, there is no machine or system learning from examples itself, but - again - humans making these decision, possibly leaving students under the impression that that is how machine learning works.

A recent contribution by Ma et al. [MSM+23] teaches decision tree classification using pasta. Several types of pasta are provided and again students get to discover what questions (i.e., which features of the pasta) will efficiently separate these types. But again, the learning is done by humans. Thus, the activity fails to convey how a machine

would learn from the available data. As the authors write themselves "optimization" using "calculations such as entropy or Gini index" is "beyond our scope and does not align with the goal for this activity".

Ossovski et al. [OB19] present a "workshop about machine learning" in which students get to train a linear classifier. The classifier is represented using a pin board and a stick pinned to the board. The data points (sizes of a bounding boxes around images of screws) are represented by pins on the board, with the color of a pin corresponding to one of two classes. The classifier is trained by subsequently adding pins to the board and adjusting the stick that serves as dividing line between the two classes.

The MENACE, Donald Michie's 1961 tic-tac-toe playing AI made from matchboxes, inspired McOwan and Curzon's *The Sweet Learning Computer* unplugged activity on reinforcement learning [MC16], which was adapted by Lindner et al. for their Activities 3 and 4 in [LSR19]. In these activities a system learns to play a game by repeatedly playing against a human. After each loss or win the moves leading to the outcome are either reinforced or discouraged by adjusting how many colored tokens are associated to each move.

Both, the linear classifier [OB19] and reinforcement learning activity [MC16], teach machine learning concepts without omitting important concepts such as training or abstracting them to a point where students might be left under a false impression about how machines learn. This thesis aims at providing activities at this low level of abstraction but on the subjects of artificial neural networks and decision tree learning.

CHAPTER 3

Decision Tree Learning Unplugged

This chapter discusses the design and development of an unplugged activity on the subject of decision tree learning and presents this activity in the last section.

3.1 Unplugging

Nishida et al. [NKI+09] argue there are "distinctive features" to computer science (CS) unplugged activities. They will only be briefly discussed here, but later used to evaluate the activities presented in this thesis (see Chapter 5).

1. "No computers": At the heart an in the name of any unplugged activity is the lack of use of computers. This lowers the barrier of entrance, both by requiring less prior knowledge and fewer or less expensive resources. (This holds true for activities about AI, albeit it may seem more challenging.)
2. "Games" or "Challenges": activities are playful to spark "interest, curiosity and motivation"
3. "Kinaesthetic": physical, tangible objects are used (and allowed to be touched, manipulated and played with)
4. "Student directed": activities rely on group work and discovery based learning
5. "Easy implementation": materials are inexpensive and readily available
6. "Growing body of ideas": activities are shared and adapted by educators
7. "Sense of story": activities are embedded in a story to further engage students

Using these features as guidelines or even requirements helped navigate the design process of the activities presented in this thesis.

Given the "no computers" requirement it is helpful to reflect on what mathematical operations upper secondary students can perform sufficiently quickly during an activity. Given pen and paper we assume students are able to perform the following operations:

- $x + y$, with $x, y \in \mathbb{N}, x + y < 100$
- $x + y$, with $x, y \in \mathbb{Z}$, $max(|x|, |y|) < 50$
- $x * y$, with $x, y \in \mathbb{Z}, |x * y| < 100$
- x/y, with $x, y \in \mathbb{Z}, x/y \in Z, max(|x|, |y|) < 100$
- $max(x_i)$, with $x_i \in \mathbb{R}, x_i$ less than 5 decimal places, $i \in [1, .., 20], x_i < 10^3$
- $min(x_i)$, with $x_i \in \mathbb{R}, x_i$ less than 5 decimal places, $i \in [1, .., 20], x_i < 10^3$
- brackets ( ) and nested brackets (( )( ))
- x^2 with $x \in \mathbb{N}, x <= 10$
- combinations of the above, e.g. $(x + y) * (a + b)$

3.2 Simplifying Training

The task at the core of the training procedure for a decision tree is determining how to best split the data at each internal node. Several metrics for what it means to be the "best" split have been proposed. Finding a metric suitable for an unplugged activity is the main goal of this section. For information on decision trees see [LLJ14].

Before discussing these metrics a first simplification step is to decide what restrictions should be placed on the type of decision tree and data to be used: We will only consider binary decision trees and a data set with two classes. We consider up to four features, either numerical (integers only), categorical or binary. Size of the training data set should not exceed 40 data points. Additionally we simplify nomenclature to facilitate understanding and use for students as follows:

- Labels will be referred to by their names, e.g. "dangerous" or "friendly".
- A data point is referred to as "card" and data as a "stack of cards".
- A feature is referred to as "characteristic" and single features referred to by their names.

- Given a threshold or value x a split can be formulated as a natural language yes-or-no question. For numerical features in the form "f_i is x or more?" and for categorical features simply "x?", given every value is unique to a feature. A split and the associated question will be referred to as "branch", since this is the physical object students will interact with when placing a best split during the construction of the tree.

- After splitting a stack of cards based on a "branch", cards for which the splitting question was answered with "yes" are put into the yes-stack, the others into the no-stack.

- The number of occurrences of each class in the yes- and no-stacks after a split are given variable names (see also Figure 3.16):

  - a: number of cards in yes-stack of first class
  - b: number of cards in yes-stack of second class
  - c: number of cards in no-stack of first class
  - d: number of cards in no-stack of second class

We used two approaches for finding a suitable metric for the quality of a split:

1. Bottom-up: Defining a new metric based on the estimation above of what mathematical operations a student can reasonably be expected to perform

2. Top-down: Simplifying an existing metric to the point were a student can reasonably be expected to calculate it

The bottom-up approach delivered several new metrics to be subsequently evaluated, such as:

1. *addition*: $(a + b) + (c + d)$

2. *multiplication*: $(a * b) + (c * d)$

3. *subtraction*: $(a - b) + (c - d)$

4. *minimum*: $min(a + b) + min(c + d)$

5. *maximum*: $max(a + b) + max(c + d)$

6. $minimum^2$: $(min(a + b))^2 + (min(c + d))^2$

None of these performed well during testing: either the resulting tree had too many branches, relied on features designed to be unimportant or did not terminate at all (producing empty children).

The top-down approach led to the evaluation of several proposed metrics such as *Information Gain, Error Rate, Gini Impurity* and *DKM Criterion*. We were able to simplify the *Gini Impurity* metric for our purposes, without changing the order in which it ranks possible splits, the following way (original definitions adapted from [LLJ14]):

Let $X = \{x_1, ..., x_n\}$ be the set of all training data and $S = \{1, ..., k\}$ a splitting rule, then S divides X into a set of subsets $\{X_1, ..., X_k\} =: X_S$, where k is the number of child nodes of an internal node. Let $Y = \{y_1, ..., y_m\}$ be the set of class names, i.e. labels, then $|X_{ij}|$ is the number of occurrences of class j in subset X_i. Let $p_{ij} = \frac{|X_{ij}|}{|X_i|}$ be the ratio of members of class j to the whole subset. Then the *Gini Criterion* of a subset can be defined as

$$Gini(X_i) := \sum_{y \in Y} p_{iy}(1 - p_{iy}) \tag{3.1}$$

The *Gini Impurity* of a splitting rule S is defined as

$$F_{Gini}(S) := \sum_{i \in S} \frac{|X_i|}{|X|} Gini(X_i) \tag{3.2}$$

We substitute $k = 2$ (we decided on a binary tree) and $m = 2$ (we decided on a data set with only two classes). This gives us $X = X_1 \cup X_2$, $X_1 = X_{11} \cup X_{12}$ and $X_2 = X_{21} \cup X_{22}$. Let $a := |X_{11}|$, $b := |X_{12}|$, $c := |X_{21}|$, and $d := |X_{22}|$ then

$$\begin{aligned} Gini(X_1) &= \frac{a}{a+b}(1 - \frac{a}{a+b}) + \frac{b}{a+b}(1 - \frac{b}{a+b}) \\ &= \frac{ab}{(a+b)^2} + \frac{ba}{(a+b)^2} \\ &= \frac{2ab}{(a+b)^2} \end{aligned} \tag{3.3}$$

and similarly $Gini(X_2) = \frac{2cd}{(c+d)^2}$.

Thus the *Gini Impurity* for a splitting rule S can be simplified to

$$\begin{aligned} F_{Gini}(S) &= \frac{|X_1|}{|X|} \cdot Gini(X_1) + \frac{|X_2|}{|X|} \cdot Gini(X_2) \\ &= \frac{a+b}{n} \cdot \frac{2ab}{(a+b)^2} + \frac{c+d}{n} \cdot \frac{2cd}{(c+d)^2} \\ &= \frac{2}{n} \cdot \Big(\frac{ab}{a+b} + \frac{cd}{c+d}\Big). \end{aligned} \tag{3.4}$$

Given that the leading fraction $\frac{2}{n}$ is the same for all splits it will not change their ranking (i.e., the best split will stay the best split) and can thus be omitted, giving us the *Simplified Gini Impurity*

$$F_{SimpleGini}(S) := \frac{ab}{a+b} + \frac{cd}{c+d}. \tag{3.5}$$

Recalling the mathematical operations students can perform, *Simplified Gini Impurity* those not quite qualify. However, since it is still rather simple while performing just as the original, it was decided to use this metric but allow students the use of a calculator during the activity. Students can now calculate the quality of a split and "grow" a tree that matches one that would "grow" on a computer (see Figure 3.1 for a comparison).

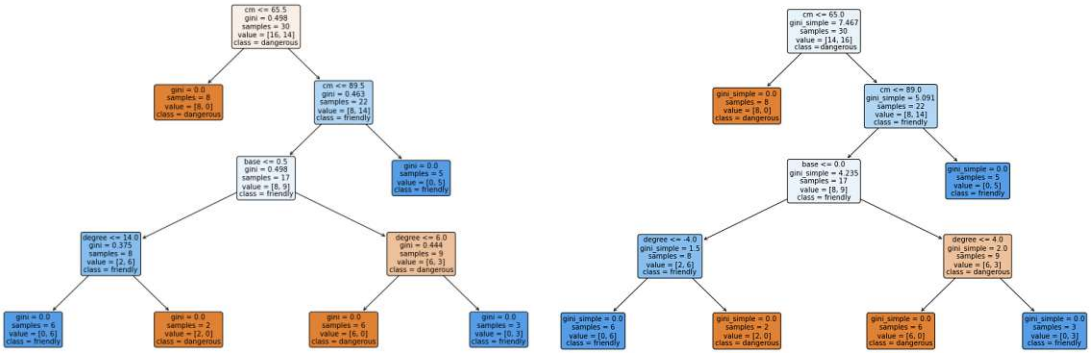


Figure 3.1: Left: Decision tree trained on the *Aliens* training data using *Gini Impurity*. Right: Decision tree trained on the same data using the *Simplified Gini Impurity* metric.

3.3 Designing the Activity

3.3.1 The Introductory Phase - Covering the Basics

Before any training aspect of a decision tree classifier can be introduced, students must first be familiarized with the basics of such a model by letting them discover the answers to the following questions:

- What elements is a decision tree classifier composed of?
- How can I assemble these elements into a functioning decision tree?
- What data is used in this activity?
- How is data represented in this activity?
- How can I classify a data point using a decision tree?
- How can I assemble a functioning decision tree that classifies all training data correctly?

Elements of a Decision Tree Classifier

The decision tree for this activity is composed of a root node, internal and leaf nodes, and directed edges between those nodes. Several methods for representing these elements unplugged were tested: Having students use pen and paper to draw a decision tree was quickly ruled out as it made discovering the right configuration of a tree by trial and error (see the third task of the activity) tiresome, since for every change an eraser has to be used or even the entire tree redrawn. Using paper-cutouts for each element improved this aspect, but lacked durability: One wrong move and the entire tree would be swooped away and all progress lost.

These experiences led to the decision to use 3D printed objects to represent each element (Figure 3.4):

- For the edges ("twigs") materials already existing at the *eduLAB* are repurposed: "Steckstäbchen" are $100mm$ by $20mm$ by $5mm$ 3D printed sticks with two holes that fit one other stick each. At *eduLAB* these are used for a variety of unplugged activities covering subjects such as binary codes and pattern recognition. They were readily available and thus the remaining decision tree elements were designed around them. Twigs of two different colors are used for a tree and students are told (via instruction sheet) which color represents a "yes"-twig, an which a "no"-twig.
- The internal nodes and root node ("branches") do not differ in design. They are triangular table-like objects with one leg in the center and legs under the tips that can each fit into a twigs hole. Their printed using brown filament suggesting they belong to the "wooden" part of a tree, i.e. a branch. The triangular design pushes students towards building binary tree. In a branch's center there is space for a $24mm$ diameter sticker to print on a value or a feature plus a threshold formulated as a yes-or-no question (e.g. "Green?" or "More than 5 seeds?", Figures 3.8 and 3.15). A sticker's background color matches their value's or feature's color on the data cards (Figures 3.6 and 3.12).
- Leaf nodes ("leaves") are leaf-shaped objects printed in green. Their single central leg fits into a twig's hole. They also have space for a sticker with a label (e.g. "Apple") printed on.
- To anchor the tree a tree stump shaped object with two holes is printed and glued onto a foiled A3 piece of paper. Any branch fits into this tree stump making it the root node. A garden is printed onto the A3 paper to further support the "a decision tree grows like an actual tree" analogy. See Figures 3.9 and 3.18 in the background.

Assembling a Decision Tree Classifier

The tree can be assembled by sticking a branch into the tree stump, putting a "yes"- and a "no"-twig under the open ends of the branch, adding a leaf or a branch to the open

ID	color	seed count	growth type	peel edible	label
1	red	5	tree	yes	apple
2	green	6	tree	yes	apple
3	yellow	5	tree	yes	apple
4	orange	9	tree	no	orange
5	yellow	0	herb	no	banana
6	green	0	vine	yes	grape
7	green	391	vine	no	watermelon
8	red	206	herb	yes	strawberry
9	green	472	vine	no	watermelon
10	brown	0	herb	no	banana
11	red	1	tree	yes	cherry
12	orange	9	tree	no	clementine

Table 3.1: The *Fruits* data set. IDs 1-8 are training data, IDs 9-12 test data.

ends of the twigs and repeating the process until no open twigs are left. The students receive written instructions (Figure 3.5) to learn this process.

The Data Set

The data set for this introductory phase must be designed to facilitate the question *How can I assemble a functioning decision tree that classifies all training data correctly?* Since at this point students will not yet know how they can assemble such a tree it helps if they can easily classify the data themselves and are thus able to double check their tree's decisions. To this end, data was chosen that any student will be able to classify: fruits.

The *Fruits* data set (3.1) consists of 12 data points of 9 different classes. There are two categorical, one binary and one numerical (integers from 0 to 472) feature. The high classes to data points ratio was chosen to facilitate the manual building of the decision tree: For the training set classes were chosen that can easily be distinguished using a maximum of two features. There are three "apple" in the set to keep students from thinking each data point *has* to have a different class. The "growth type" feature was chosen because their values might not be immediately familiar to all students (e.g. a "banana" grows on a "herb" not a "tree"), thus forcing students to actually read the data and not dismiss it because they know it by heart.

Each test data point serves a specific purpose in order to spawn a conversation on different aspects of training data quality at the end of the introductory phase (see *Fourth Intervention and Plenary Discussion.* in Section 3.4.4):

- The watermelon (ID 9) will be correctly classified by any tree that classifies ID 7 correctly, assuring at least one positive outcome.

- The brown banana (ID 10) can be correctly classified by *some* trees, depending on whether the color "yellow" was used to split data to on the path to the "banana" leaf. This is be used to demonstrate that the quality of a decision tree model depends on the diversity of its training data: If varieties of "banana" are missing during training they may later be misclassified. Thus, solely relying on a feature (in this case "color") that does not have the same value for all members of the class, should be avoided.

- The cherry (ID 11) can not be classified correctly by *any* tree based on the training data, since this class is not present there. This is used to argue that generalised claims such as "This decision tree can classify *all* kinds of fruit" should be avoided. Students learn to be wary of such generalised claims about an AI models' abilities, when encountering them in their daily lives.

- The clementine (ID 12) matches the orange's (ID 4) values exactly and can thus not be classified correctly by any tree, even if the clementine was present during training. This is used to discuss that adding more examples to the training set does not always suffice to correctly classify all data points, but rather more *features* may be needed. Students are then asked to think of features that would let their decision tree tell apart an orange from a clementine.

Data Representation

Decisions regarding data representation are dominated by what is needed to facilitate the training phase and are thus discussed below. During the introductory phase students are merely familiarised with the design of the data cards (Figure 3.6) so that they are quickly able to use the *Aliens* data cards (Figure 3.12) during the training phase.

Classifying Data using the Decision Tree

A data point is classified similarly to how it would be in regular decision tree: starting at the branch in the tree stump (root node), answering the yes-or-no question using the information provided on the card and moving along the "yes"- or the "no"-twig accordingly. Then the process is repeated until a leaf is reached. Students students are again provided with written instructions (Figure 3.7) to learn this process.

Assembling an Accurate Decision Tree Classifier

How to construct a tree that can classify all training data correctly is entirely left to the students. They will use their own *human* intelligence to solve the problem which contrasts the *artificial* intelligence that will "grow" the tree during the next phase. There are many possible solutions to this task. Students may construct sub optimal trees (more than five branches), which can be used to start a conversation about the efficiency and in general the quality of a decision tree.

3.3.2 Training Phase - Introducing Machine Learning

Now that the basics are covered, the concept of machine learning and training a model can be introduced to the students.

The Data Set

The data set (Table 3.2) for this phase will be used to train a decision tree using Gini Impurity. As such the training data is handcrafted to achieve the following goals:

- The resulting tree should have a manageable size (again five branches as in the introductory phase).
- The first two best splits should immediately "remove" data (i.e., have a leaf nodes as a child) to reduce the complexity of the subsequent calculations (e.g., 8 training data points are immediately classified "dangerous" after the first split, see Table 3.2).
- One feature should be completely random and never be used for a split to enable a discussion on feature importance.
- The other features should all be used at least once in the resulting tree.

See Figure 3.18 for an image of the resulting tree.

The subject "Aliens" was chosen to justify the use of nonsensical feature names. The features are made up, have no meaning and can thus not be "understood" by students, mirroring how a ML system would also not "understand" the data it is trained on.

Data Representation

Data representation plays a crucial role in facilitating the otherwise tedious task of calculating the impurity of every possible split.

A first prototype relied on multiple printed out copies of a spreadsheet containing the data. In each copy the data was sorted by a different feature. This facilitated the calculations for the first branch. After choosing the first split however, the spreadsheets are turned useless, since for all but the feature of the chosen split, the sheets now contained data from each stack mixed together.

Thus, a representation was needed that could easily be divided into separate stacks after each split. A stack of cards seemed the obvious solution. However, the issue with regular playing card style data cards is that they would need to be sorted according to each feature (primarily numerical features) for every branch to efficiently count up each needed amount. Using the *Aliens* data set this would mean sorting 30 cards for the first branch, 22 for the second, then 17, 9, and finally 8 cards - each time twice. Adding to that the counting for the variables and the impurity calculation, this seemed

#	cm	degree	base	poly	label
20	94	-8	C	square	friendly
33	93	16	C	square	friendly
8	92	0	C	triangle	friendly
5	91	10	Si	circle	friendly
22	90	24	Si	triangle	friendly
7	89	-24	Si	triangle	dangerous
14	86	-4	C	circle	friendly
19	85	-30	Si	square	dangerous
24	84	-34	C	triangle	friendly
26	83	8	Si	circle	friendly
6	82	32	C	circle	dangerous
4	81	-20	Si	triangle	dangerous
15	80	-32	C	circle	friendly
13	78	-26	Si	circle	dangerous
11	77	26	Si	triangle	friendly
12	76	-28	C	triangle	friendly
32	75	-38	C	triangle	friendly
23	73	36	Si	square	friendly
30	72	38	C	triangle	dangerous
2	70	-36	Si	triangle	dangerous
27	69	4	Si	square	dangerous
21	68	-2	Si	square	dangerous
25	67	28	Si	square	friendly
17	66	-12	C	circle	friendly
3	65	34	C	triangle	dangerous
16	64	20	C	circle	dangerous
34	63	-18	C	square	friendly
1	62	-14	Si	circle	dangerous
9	61	-16	Si	circle	dangerous
31	59	12	Si	square	dangerous
10	58	2	C	circle	dangerous
28	57	22	C	circle	dangerous
29	56	6	C	triangle	dangerous
18	55	-6	Si	square	dangerous

Table 3.2: The *Aliens* data set, sorted in descending order by "cm". Highlighted rows (#11, #21, #28, and #34) are test data. The dashed line below #17 indicates where the first best split separates the data and consequently causes the misclassification of "alien #34". The dashed line below #7 indicates the best split after the first.

to extensive a workload. Clearly, a data representation that does not require resorting after each split was needed.

This requirement inspired the design of the "data cards" (Figures 3.2 and 3.12) for the *Aliens* data set. They are 66*mm* by 132*mm* rectangular cards. Each side corresponds to one of the four features of an *Aliens* data point. The value for each feature is indicated by a tag. Each of a data point's values are indicated twice on the card: once by text (or image) printed on a tag and once by that tag's position.

Figure 3.2: Stacked *Alien Data Cards*. Two images to the left: Picture of an unsorted stack of printed, foiled and cut out data cards containing a card for each *Aliens* training data point, and the digital version of that stack. Two images to the right: resulting stacks of data cards after applying the first best split (using *Gini Impurity* as metric): feature "cm" at threshold 66.

This design allows students to count the class frequencies for each possible split based on the "cm" or "degree" feature without needing to sort the stack of cards. Figure 3.2 shows the stacked *Aliens* training data cards before and after applying the first best split. Additionally, the tags can be used to literally split a stack of cards according to the chosen best split, by pushing all tags below the threshold to one side, and those above to the other, and the separating the cards.

For the two categorical features the stack is best sorted into each of the two or three categories. Alternatively one can try to split the stack by pushing the tags or to determine the class frequencies directly by looking at the stack sideways or tilting it to one side in order to count the tags.

During the activity, students are free to discover these mechanics on their own, but are given assistance as needed.

How a Decision Tree Learns

What remains is to teach the students how a decision tree learns, i.e. how they can train the machine learning model. To this end they are provided with two algorithms (Figures

3.13 and 3.14): One teaches them how to find the best split given a stack of cards by calculating the "error" (the *Simplified Gini Impurity*) for each possible split of the stack. The other, how to use that information to construct a decision tree.

Students are provided with empty spreadsheets (Figure 3.16) to note down their results. To lighten the workload and serve as instructive examples, a partially filled out spreadsheet (Figure 3.17) is provided, in which the impurity of some splits of the first branch are already calculated.

3.4 Presenting: *The Tree Nursery*

3.4.1 Learning Objectives

When having completed the activity *The Tree Nursery*, students will be able to ...

1. Describe and demonstrate how a decision tree classifies a data point.
2. Construct a classification decision tree based on a small data set ($\sim$ 10 data points, 4 features).
3. Use a decision tree to classify data.
4. Discuss and explain why a data point is misclassified by a decision tree.
5. Discuss quality aspects of training data.
6. Adapt a decision tree to fit new data.
7. Understand that some features may be more important that others when classifying data and discuss which features may be important when classifying everyday objects.
8. Execute an algorithm (top-down induction of a decision tree) to train a machine learning model (classification tree) based on a small data set ($\sim$ 30 data points, 4 features).
9. Explain how a decision tree learns from data.
10. Explain why outliers may be misclassified by a decision tree.
11. Name examples of domains in which decision trees are used and for what purpose.

3.4.2 Student Prerequisites

Students do not require prior knowledge on decision trees. The first set of tasks (*The decision tree*, Figure 3.3) introduces all information and skills on that subject necessary to complete the entire activity. Prior knowledge on AI or ML is not required either. Basic (German) reading, writing, speaking and listening skills are required. Students need to

be able to read and execute step-by-step instructions that include if-then-else statements, goto statements (redirecting students to previous or following steps) and one for-each loop. These instructions, however, are presented in natural language and do not require knowledge of algorithms or programming. See Figure 3.5 for an example. For the second task sheet (*The Self-Learn-Tree*) students need to be able to use a pocket calculator to repeatedly calculate a formula consisting of two integer additions, multiplications, and divisions as well as a rational addition (Figure [refgini formula). In order to calculate this formula students need a basic understanding of variables in mathematical formulas and be able to insert values for these variables. A help sheet is provided to the students to facilitate this process (Figure 3.16).

3.4.3 Target Setting and Audience

The activity presumes a classroom setting where students are split into groups, each with a desk and chairs. The ideal group size is two to four students. Each group receives a copy of the materials and students are supposed to work together within their group to solve the tasks. The sixth task can be done with all groups working together cooperatively or by each group individually. The teacher leads plenary discussions, distributes materials and acts as assistant during group work. Given the prerequisites the target audience are students in upper secondary school. The first task sheet on its own however can be used with a younger audience (lower secondary school students). The target duration for the activity is 90 minutes.

3.4.4 Activity Rundown

This activity *The Tree Nursery* (German original: *Die Baumschule*) is a mix of group work and plenary discussions. First, students are introduced to what a decision tree is, after which they learn how to train a tree using the *Simplified Gini Impurity* metric for finding the best split. The activity has four main parts:

1. Introduction and Plenary Discussion
2. The Decision Tree
3. The Self-Learn-Tree
4. Closing Plenary Discussion

Introduction and Plenary Discussion

The activity starts with a short, general introduction and plenary discussion between the teacher and students of all groups on the topic of AI and ML. This discussion may go as follows:

TEACHER: Has anyone heard the term "artificial intelligence" - in short "AI" - before?

STUDENT: Yes!

TEACHER: Can anyone describe what AI is?

[*Students answer and their answers are discussed. If nobody answers:*]

TEACHER: Can you name examples for AI?

[*If still nobody answers:*]

TEACHER: Where have you heard that term before?

[*Students answer.* Teacher *resolves the question with a short introduction to artificial intelligence.*]

TEACHER: Has anyone heard the term "machine learning" before?

STUDENT: Yes!

TEACHER: Can you describe what "machine learning" is? [*If nobody answers*] Or where you've heard it? [*Students answer and their answers are discussed.* Teacher *resolves the question:*]

TEACHER: A machine learning system is a special kind of AI system, that "learns" to solve a task from examples. One could say we "teach" or "train" the system using examples. There are many different types of such ML systems, one of which you will get to train today yourself! It is called a "decision tree", has anyone heard of that before? [Teacher *lets students answer and concludes by introducing the first task sheet. A definition or description of decision trees should not be provided at this point, since the activity serves to let students discover this on their own.*]

First Task Sheet: *The Decision Tree*

Each group is handed a copy of the first task sheet *The Decision Tree* (Figure 3.3) which consists of four tasks. Instructions for tasks are each confided to a rectangle on the task sheet.

First Task: Wild Growth. Each group is instructed to read the first task/rectangle *Wild Growth* (*Wildwuchs*). Then a copy of the following materials are handed out to each group:

1. A small bag containing several 3D printed "tree parts" (Figure 3.4):
   a) 9 "yes"-twigs and 9 "no"-twigs
   b) 12 branches
   c) 9 leaves

Figure 3.3: The first task sheet *The Decision Tree* (*Der Entscheidungs-Baum*).

2. *The Garden*: A foiled A3 paper with a garden and a tree stump printed on it. Additionally, there is a 3D printed part of the tree stump glued onto the paper. This tree stump part has holes such that a branch can fit into it and can thus be used for the first branch and starting point of the decision tree.

3. *Manual 1 "How a tree grows"* (Figure 3.5): An algorithm (referred to as a "manual") for constructing a decision tree from the materials listed above.

4. *Manual 2 "How a tree recognises a fruit"* (Figure 3.7): A second manual showing how a decision tree "recognises", i.e. classifies, fruit. It is needed for the second task, but printed on the back of *Manual 1* and thus handed out with the first task.

The instructions for the first task read:

We want to plant a tree that can recognise fruit. Such a "decision tree" consists of: leaves, branches, "yes"-twigs and "no"-twigs. Grow a decision tree in the garden! Take a look at Manual 1 "How a tree grows"! This time use a maximum of five branches!

Manual 1 "How a tree grows" reads:

1. Stick any branch into the tree trunk.

A – branch with question sticker
B – leaf with label sticker
C – "no"-twigs
D – "yes"-twigs
E – tree stump glued onto *The Garden*
F – bag for materials

Figure 3.4: "Tree parts": 3D printed materials students receive for the first task of *The Decision Tree*.

Figure 3.5: *Manual 1 "How a tree grows"*. An algorithm for assembling a decision tree from 3D printed "branches", "twigs", and "leaves".

2. Stick a "yes"-twig and a "no"-twig to the two free ends of the new branch.
3. Choose any free twig in the tree and either stick a branch in and go to step 2 or stick a leaf in and go to step 4.
4. If there are still free twigs in the tree, go to step 3, otherwise you are done.

Students are now left to solve the first task on their own.

First Intervention. When a group finishes the first task they will either call upon the teacher right away or start reading the second task at which point they will call upon the teacher to receive the "fruit". In either case, the teacher can now interact with the group to do the following:

1. Check whether a correct tree was assembled. If not, point out that fact but provide as little assistance as needed, e.g. only suggest having another look at *Manual 1* or directly discuss steps in the algorithm that might have been misread. Let the group try again and check back later. Otherwise continue.
2. Foster reflection: Let a student explain what the group has done so far.
3. Instruct the group to proceed to the second task and provide them with the eight *Fruit Training Data Cards* (Figure 3.6 and IDs 1-8 in Table 3.1).

Second Task: Wild Harvest. The second task *Wild Harvest* (*Wilde Ernte*) reads:

Finished growing? Try out your tree! Ask the workshop leader for the fruit and have a look at Manual 2 "How a tree recognises a fruit" (back of Manual 1)! Everything correct? Discuss why fruits are recognised correctly or incorrectly!

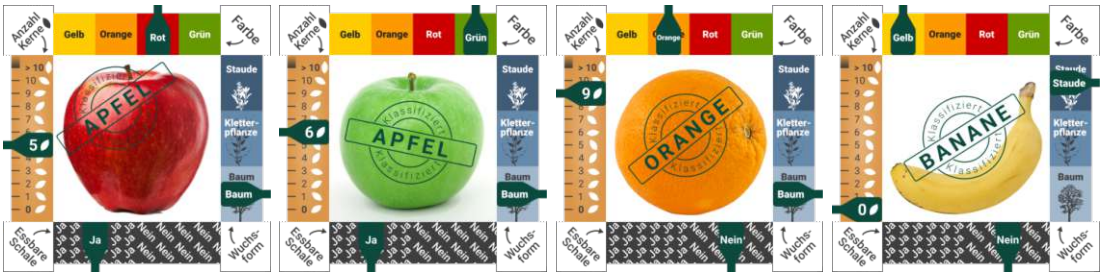


Figure 3.6: *Fruit Data Cards* for four of the training data points in the *Fruits* data set (IDs 1, 2, 4, and 5).

Manual 2 "How a tree recognises a fruit" reads:

1. Take a card.
2. Put the card next to the branch in the tree stump.

Figure 3.7: *Manual 2 "How a tree recognises a fruit".* An algorithm for using a decision tree to classify a fruit.

3. Answer the question written on the branch (with the help of the card):
   Answer is yes: Put the card at the end of the "yes"-twig that is stuck in the branch.
   Answer is no: Put the card at the end of the "no"-twig that is stuck in the branch.

4. At the end of the branch there is either another branch or a leaf:
   Branch: Go to step 3.
   Leaf: Done! The tree has recognised the card as the fruit that is written on the leaf.

The group is again left to solve the task by themselves.

Second Intervention. A full intervention at this point is not required for every group. Some groups may move on to the next task and start fixing their tree immediately after realising it does not work as intended. Others may be stumped as to why some cards are not classified correctly or may not understand how *Manual 2* works at all. The latter groups should be addressed and engaged in a conversation. First the teacher should check whether students executed *Manual 2* correctly. If not, some assistance may be provided. If yes, a conversation on what goes wrong in their tree during the execution of *Manual 2* and how it may be fixed, should be had. In any case, each group should be handed a set of *Fruit Stickers* allowing them to re-label branches and leaves during the next tasks.

Third Task: Cultivated Plant. The third task *Cultivated Plant* (*Kulturpflanze*) reads:

Previously, the tree grew randomly, i.e. wild growth. This leads to fruits being wrongly recognised. Let us now take a more cultivated approach: Plant a new tree. It must recognise all the fruit cards correctly! Think about which branches and leaves belong where. Use stickers to re-label branches. Use as many branches as you want and as many as you have.

Finished growing? Let all the fruits be recognised! Everything recognised correctly? Great! The tree is ready. Use it for the next task! Fruits recognised incorrectly? Change the tree until it makes no more mistakes!

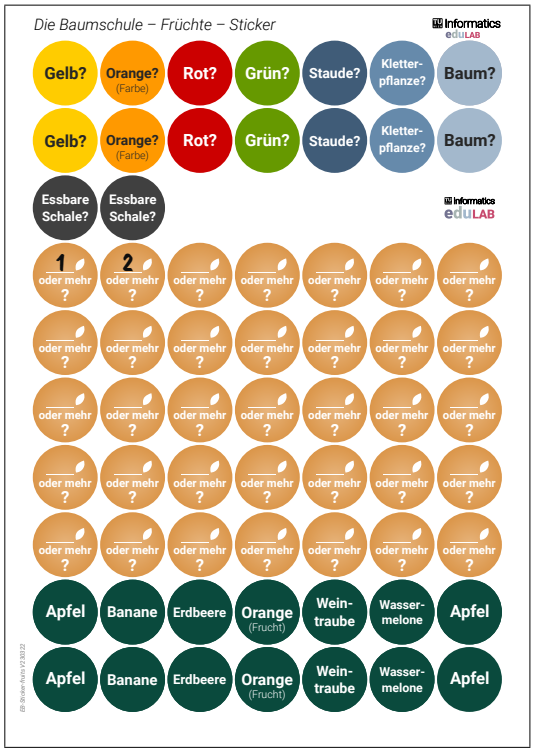


Figure 3.8: Set of *Fruit Stickers* used to label *branches* and *leaves* during the third and fourth task.

After being handed a set of *Fruit Stickers*, a group is left to fix or rebuild their tree on their own.

Third Intervention. Similarly to the first intervention, students will either call upon the teacher after having completed the third task or after having read the first line of the fourth task. In either case the teacher should engage the group at this point and do the following:

1. Check whether the tree is a correct decision tree and whether it classifies the training data correctly. If not, point out that there are mistakes and tell the group

Figure 3.9: A student group's solution to the third task *Cultivated Plant*.

to find and fix them.

2. Check whether the tree is as small as possible. Note that there are many smallest solutions but each has exactly five branches. If more than five branches were used, ask the group whether they think their tree could be improved. If the group struggles to find ways to improve their tree, some assistance may be given (e.g. by taking a data card that during classification is moved along a sub-optimal path in the tree, classifying the card in front of the group and asking whether each branch is really necessary). After fixing the tree a short conversation about the benefits and drawbacks of making a tree as small as possible can be had.

3. Segue into the next task by handing out the test data cards (Figure 3.10) and telling students to put their tree to the *real* test with the next task.

Fourth Task: Harvest Time! The third task *Harvest Time!* (*Erntezeit!*) reads:

Ask the workshop leader for new fruits to harvest! Let your tree recognise the new fruits! Everything correct? Discuss why some fruits are recognised incorrectly! Notify the workshop leader once you are done.

Each test data point serves a specific purpose (as described during the upcoming fourth intervention and discussed in *The Data Set.* in Section 3.3.1), which students should discover during this task. Some students may start fixing their tree again to make it classify the (brown) "banana" and/or "cherry" test data card correctly. This should be allowed but discussed during the following intervention. Some students may call upon

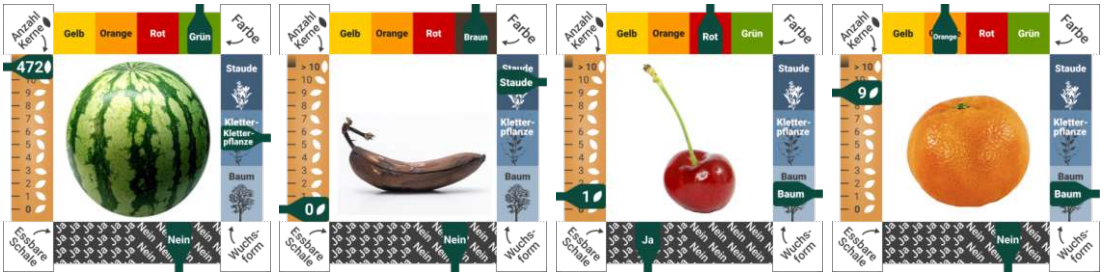


Figure 3.10: *Fruit Data Cards* for all the test data of the *Fruits* data set.

the teacher because they failed to make the tree recognise the "clementine" data card correctly, at which point the students should be reminded that the task is to simply let the tree recognise the new fruits and discuss why some fruits are recognised incorrectly - not necessarily fix the tree.

Fourth Intervention and Plenary Discussion. This final intervention closing off the first task sheet is best held as a plenary discussion with all groups. (Note: If a group finishes a lot sooner than the others and cannot be asked to wait for the rest to finish, they can already be posed some of the questions from the conversation below until the other groups have also finished off the first task sheet.)

During the fourth task students should have discovered and discussed within their groups why each of the test data cards was correctly or incorrectly classified. These findings can now be shared and discussed with the entire class and used as jumping off points for discussions on various subjects. The line of questioning may go as shown in the conversation below. Note that the answers given by students below as well as in the conversation to follow represent a best case scenario and it is unlikely that every answer will be given as shown. Whenever no such answer is given the teacher should steer the students towards that answer by asking more questions, giving hints, or finally resolving the question by answering it themselves.

TEACHER: So you've all tested your decision trees with new fruits that your tree hasn't seen before. [*Teacher holds up the watermelon test data card (ID 9)*] Show of hands, whose tree recognised this watermelon correctly? [*All hands should be raised.*] Great! Why do you think that is?

STUDENT: Because we're great!

ANOTHER STUDENT: Because it's like the other watermelon!

TEACHER: Right, it's definitely similar, but is it exactly the same as the other watermelon?

STUDENT: No, it has more seeds!

TEACHER: Right, and still your tree recognised the new watermelon! That's great!

TEACHER: Let's move on. [*Teacher holds up the brown banana data card (ID 10)*] Whose tree recognised this banana correctly? [*None to all hands may be raised. Ideally at least one group did and one group did not classify it correctly.*]

TEACHER: Why do you think (some of) your trees did not recognise this banana?

STUDENT: It is brown not yellow.

TEACHER: OK, but it is still a banana, right? So, how can we grow a tree that recognises both bananas?

STUDENT: Do not ask whether it's yellow.

ANOTHER STUDENT: Do not ask about the color.

TEACHER: Good idea! Bananas can be yellow, brown, or even green! So asking for the color in order to recognise a banana may go very wrong! What else could we do to avoid this problem?

STUDENT: More data cards!

TEACHER: Right, but not just any data cards. Which data cards should we add?

STUDENT: Cards with green and brown bananas.

TEACHER: Yes! Because We need more diverse data cards! Adding more yellow bananas to the set of data cards we use to grow the tree won't change a thing. We need green and brown banana cards to make sure we avoid this kind of false recognition. Do you think your trees will recognise all bananas correctly once we added green and brown ones to the data cards used to grow the trees?

STUDENT: Yes!!

TEACHER: But there are red and pink bananas too! [*If there is disbelieve Teacher may show a picture of a red banana.*]

STUDENT: Ooh!

TEACHER: So the point here is: It is really hard to be certain that your tree will always be correct. Especially when used on data cards that it has never "seen" before, that is, data cards that were not used while growing the tree. Which is why it is important to use diverse data for growing, and also keep in mind that no matter how well trained, errors can happen!

[*Teacher holds up the cherry data card (ID 11)*]

TEACHER: Whose tree recognised this card correctly?

[*We differentiate two cases, Case 1 (most likely case): No hands are raised.*]

TEACHER: Ah yes, this was a tough one. Why do you think none of your trees got this one right?

STUDENT: It can't because there is no leaf for "Cherry".

ANOTHER STUDENT: Because there was no cherry card with the data cards while growing the tree!

TEACHER: Right, there was just no way your tree could have gotten this one right.

[*Case 2: At least one group raises their hands:* TEACHER *addresses the groups that raised their hands.*]

TEACHER: Wow, great! How did you manage that?

STUDENT: We wrote "cherry" on one of the empty stickers and put it on a leaf. We added a "2 seeds or more" branch to the tree to distinguish the cherry from the red apple!

TEACHER: That's great! So you fixed the problem many others had by adapting your tree to the cherry data card as if it was there from the beginning! But what if you didn't know it was a cherry? What would have happened then?

STUDENT: The tree would have recognised it as an apple and we would not have known any better.

TEACHER: Right!

[*Continue here in either case.*]

TEACHER: So how can we avoid such a situation?

STUDENT: More data cards!

TEACHER: Yes! But not just any data cards! Which ones do we need? [TEACHER *lets the students name several fruits that may be added.*] Great, so data cards with new fruits on them. But what if [insert fruit that was not named] comes along, would the tree recognise it correctly?

STUDENT: No!

TEACHER: Right, so maybe just adding more data will not fix the issue entirely. Like before with the colors of the banana we cannot guarantee that our tree can recognise *every* fruit correctly unless we trained it on *every* fruit there is, including all their variations, colors, seed numbers etc. Do you think that is possible?

[*Students guess as to whether that is realistically possible. Their answers are discussed. The conclusion to be steered towards is that our claim "our tree can recognise every fruit" is best revised to "our tree can recognise some fruits" and that we should keep in mind that in any case, errors can happen. Then* Teacher *continues to the next test data card.*]

TEACHER: Did any of you change your tree to fit a new data card?

STUDENT: Yes!

TEACHER: Did that work for all cards?

STUDENT: No, not for the clementine.

TEACHER: Why do you think that is?

STUDENT: It's the exact same as the orange.

TEACHER: Right! And why is that a problem?

STUDENT: We can't tell the orange apart from the clementine.

TEACHER: So how do we fix this?

STUDENT: More data cards!

TEACHER: Not quite, maybe the number of seeds changes on some cards but that won't help us distinguish between this orange and this clementine. [*Teacher holds up orange and clementine data cards.*] But there are of course differences between the two fruits, can you name some?

STUDENT: Size! Weight!

ANOTHER STUDENT: Taste! Peel thickness!

TEACHER: Right! But how can we use that to improve our trees?

STUDENT: Add that information to the data cards.

TEACHER: Yes! Right now a data card has four sides and each side shows one information about the fruit, like color, number of seeds and so on. We could add a fifth side that shows the fruits weight for example. And then, what do we do?

STUDENT: Add a branch to the tree that asks about the weight of the fruit!

TEACHER: Correct! So what does this mean for us: With our original data cards we simply could not distinguish between an orange and a clementine. And just adding more data cards would not have helped like before. We had to change what *kind* of information we show on a card to make a difference!

[*Optional: A discussion about feature importance may be had at this point, if time allows.*]

TEACHER: So sometimes there is not the right kind of data available to distinguish between two fruits. Let's discuss a different example, let's say we want to tell a dog apart from a cat. What information could we use to recognise one from the other? "Number of legs" for example?

STUDENT: No! [*Students call out features while* TEACHER *interrupts with bad examples ("number of eyes", "has fur", "has tail", etc.). After some back an forth* TEACHER *continues.*]

TEACHER: So we discovered some information is more helpful than other when trying to tell two things apart. Why do you think you should care about what information is important and what not when growing a decision tree?

STUDENT: Because we could waste branches on questions that don't have an impact on the result.

Second Task Sheet: *The Self-Learn-Tree*

The second task sheet *The Self-Learn-Tree* (Figure 3.11) consists of three tasks, the first of which only sets up the story. Depending on how fast the groups have solved the tasks so far the second task sheet may be solved by all groups working together in parallel to lighten the workload or by each group individually. Students are allowed to use a pocket calculator for this task sheet. In either case the teacher hands out a copy of the second task sheet to each group.

Introduction and Fifth Task: ALIENS!! The fifth task *ALIENS!!* (first task on the second task sheet) reads:

UFOs have been sighted! The Secret Service needs your help: Four characteristics can be used to tell whether an alien is friendly or dangerous. Some such "unknown life forms" have already been recognised as friendly or dangerous. However, some of these life forms could not be recognised so far - and the time is running out, because the UFOs could land at any moment!

This passage may be read out loud to the entire class by a student or the teacher. Ideally with some theatrics to motivate students for the next task.

Sixth Task: Your Mission. The second task *Your Mission* (*Euer Auftrag*) reads:

Plant a Self-Learn-Tree that can recognise whether an alien is friendly or dangerous. A Self-Learn-Tree recognises things just like a decision tree would, but it grows by itself! For this you will need: The Alien Data Cards, Manual 3 "How a Self-Learn-Tree grows", and Manual 4 "Finding the best branch".

Done growing? Let the Self-Learn-Tree recognise a few of the alien cards! If they were all identified correctly, it's time to save humanity!

Each group is now handed a copy of the following materials:

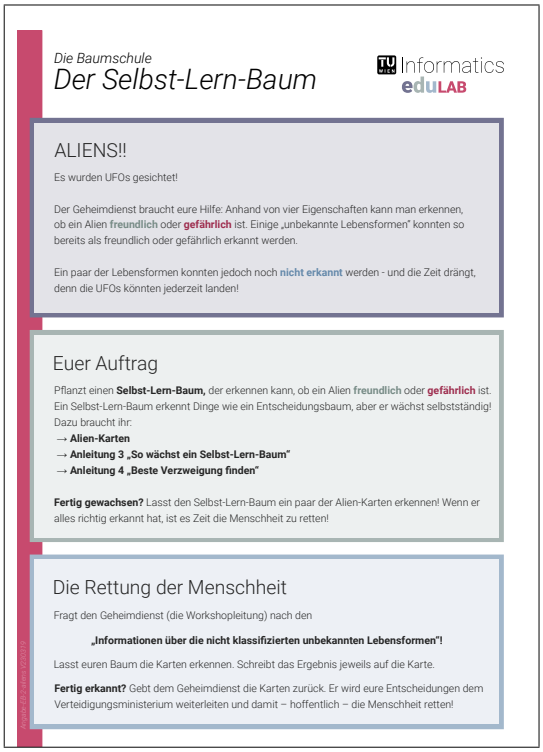


Figure 3.11: The second task sheet *The Self-Learn-Tree* (*Der Selbst-Lern-Baum*).

1. A stack of 30 *Alien Data Cards* (Figure 3.12)
2. *Manual 3 "How a Self-Learn-Tree grows"* (Figure 3.13)
3. *Manual 4 "Finding the best branch"* (Figure 3.14)
4. A set of *Alien Stickers* (Figure 3.15)
5. 4 empty *Spreadsheets* (Figure 3.16)
6. An *ABCD-Example* (Figure 3.16)
7. A partially prefilled *Spreadsheet* (Figure 3.17)

Manual 3 reads:

1. Put all the alien cards in a stack above the tree stump.
2. Find the best branch (see Manual 4 "Finding the best branch").
   Pin the best branch.
   Pin a "yes"-twig and a "no"-twig to the two free ends of the new branch.
   Divide the stack based on the new branch.
   Place the two new stacks above the free ends of the corresponding twigs.

3. Take any card stack from a twig. Distinguish:
   The aliens of the stack are ...
   - ... all "Friendly": Stick a "Friendly" leaf to the twig and set the stack of cards aside.
   - ... all "Dangerous": Stick a "Dangerous" leaf on the twig and set the stack of cards aside.
   - ... mixed "Friendly" and "Dangerous": Go to step 2.
4. If at least one stack of cards lies at a twig: Go to step 3.
5. Otherwise: Stop, the tree has finished growing.

Manual 4 reads:

1. Take a stack of cards and an empty Spreadsheet.
2. Take a new branch and do the following:
   - Count how many cards answer the branching question with ...
     - A: "Yes" and are "Friendly".
     - B: "Yes" and are "Dangerous".
     - C: "No" and are "Friendly".
     - D: "No" and are "Dangerous".
   - Fill in A, B, C, D on the Spreadsheet.
   - Calculate the "Error" of the branch and enter it on the Spreadsheet.
3. Repeat step 2 for all possible branches.
4. Error of all possible branches calculated? Find the branch with the smallest error. It is the best branch.

At this point students can be left to fend for them selves since all necessary information is provided or can be discovered. Assistance should be provided as needed. If groups are supposed to work cooperatively on this task, some coordination from the teacher will be necessary, since cooperation between groups is not indicated on the task sheet. One possible way to divide the workload is to assign every group one of the ordinal features starting either from the top or from the bottom and have them report their findings to each other. This way, four groups can efficiently work in parallel to complete the partially prefilled spreadsheet.

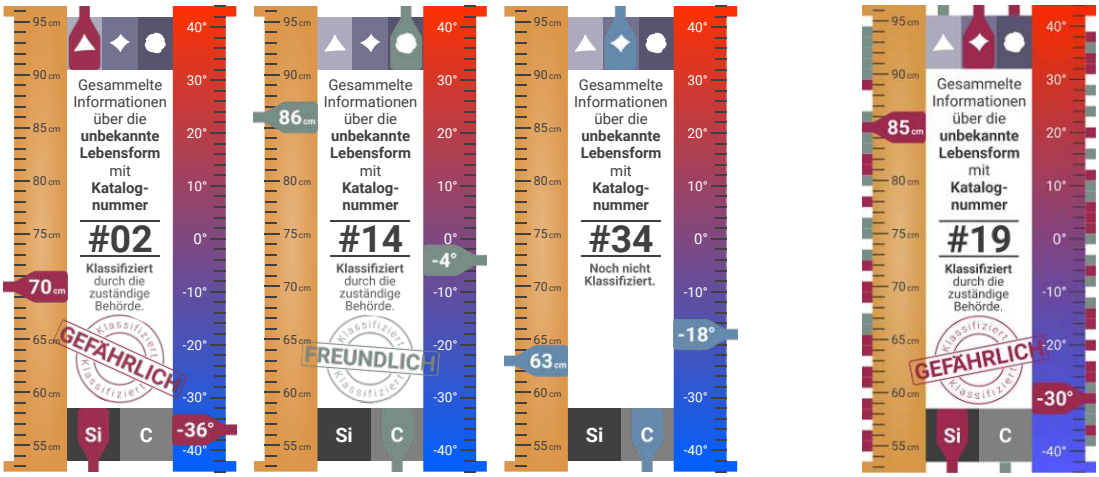


Figure 3.12: Left: Three *Alien Data Cards*: two representing two training data points and one representing a test data point (the outlier that will later be misclassified) from the *Aliens* data set. Right: All 30 training *Alien Data Cards* stacked on top of each other.

Figure 3.13: *Manual 3 "How a Self-Learn-Tree grows"*. An algorithm for how to make a decision tree "grow" by learning from data cards.

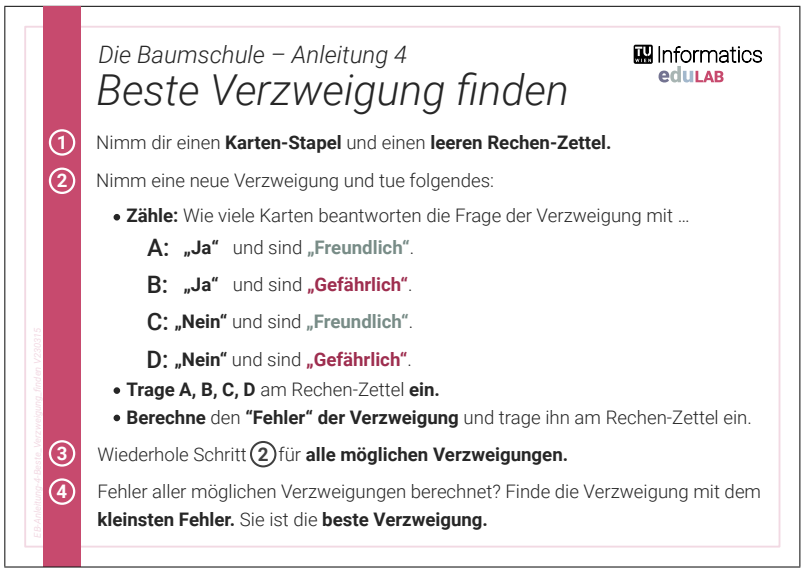


Figure 3.14: *Manual 4 "Finding the best branch".* An algorithm for finding the branch with the lowest "error", i.e. calculating the best split.

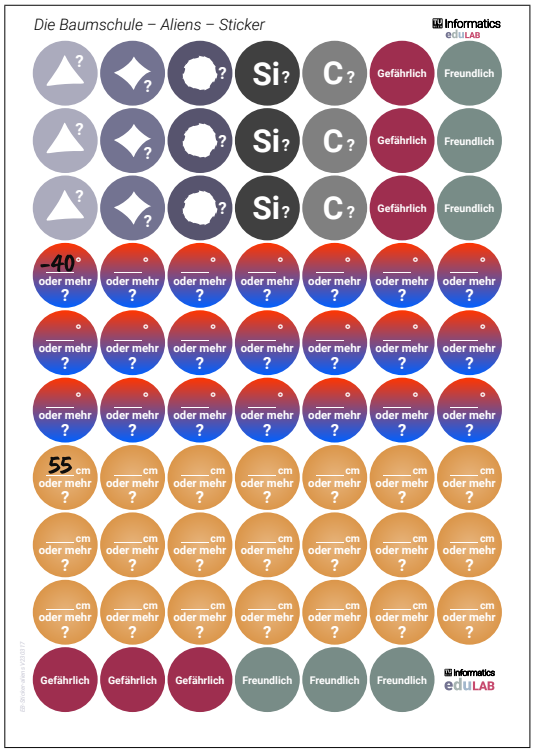


Figure 3.15: Set of *Aliens Stickers* used to label *branches* and *leaves* during the sixth task.

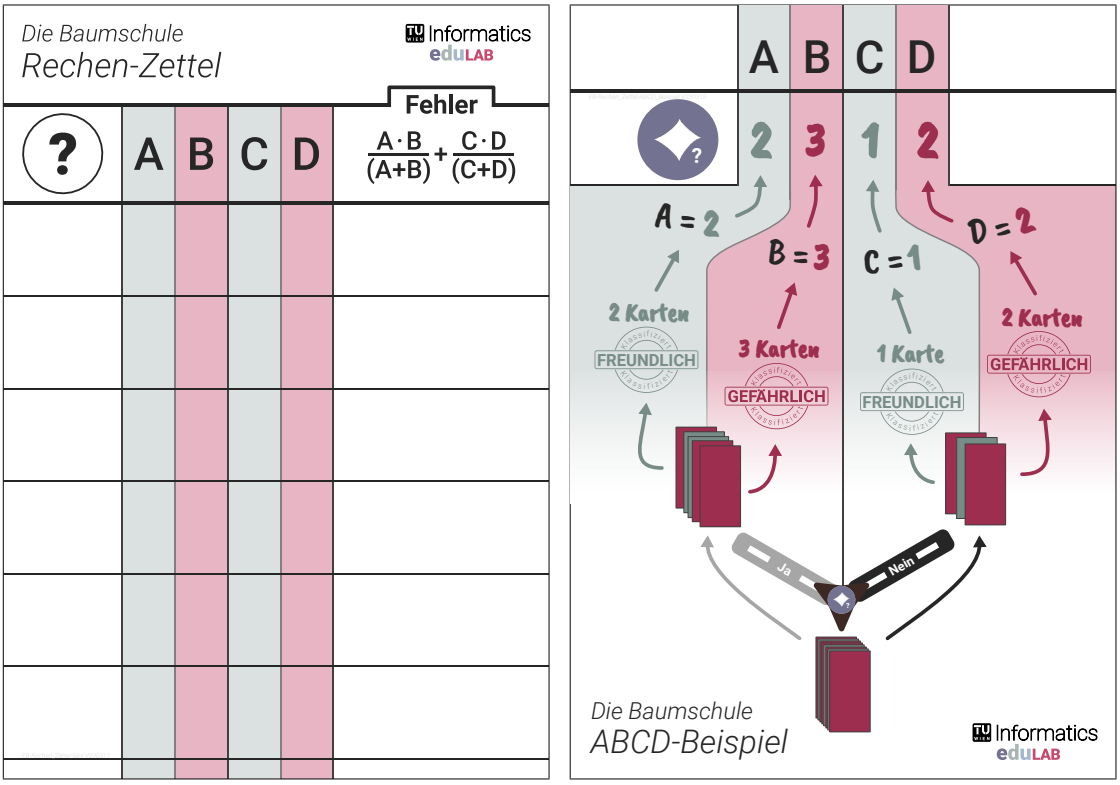


Figure 3.16: Left: An empty *Spreadsheet* used to note down and calculate the *Simplified Gini Impurity* of several possible splits. In the top right corner the formula for calculating the "Error" ("Fehler") is displayed. Right: An *ABCD-Example* which illustrates how values for variables A, B, C, and D (used to calculate the *Simplified Gini Impurity*) are determined given a stack of *Alien Data Cards*.

Final Intervention. Once again students will reach out to the teacher once they are done with the second task or right after starting the seventh task. At this point the teacher should make sure the groups tree was trained "correctly", i.e. matches the sample solution. If not, the groups calculations must be checked or other possible issues resolved. Otherwise the group can proceed to the seventh and final task by receiving the test data cards. If the sixth task was solved cooperatively all groups should have the same solution and the intervention can be done as a plenum discussion.

Seventh Task: Saving Humanity. The seventh and final task *Saving Humanity* (*Die Rettung der Menschheit*) reads:

Ask the Secret Service (the workshop leader) for the "information on the unclassified unknown life forms"! Let your tree recognise the cards. Write the result on each card.

Finished recognising? Give the cards back to the Secret Service. They will pass on your decisions to the Ministry of Defence and - hopefully - save humanity!

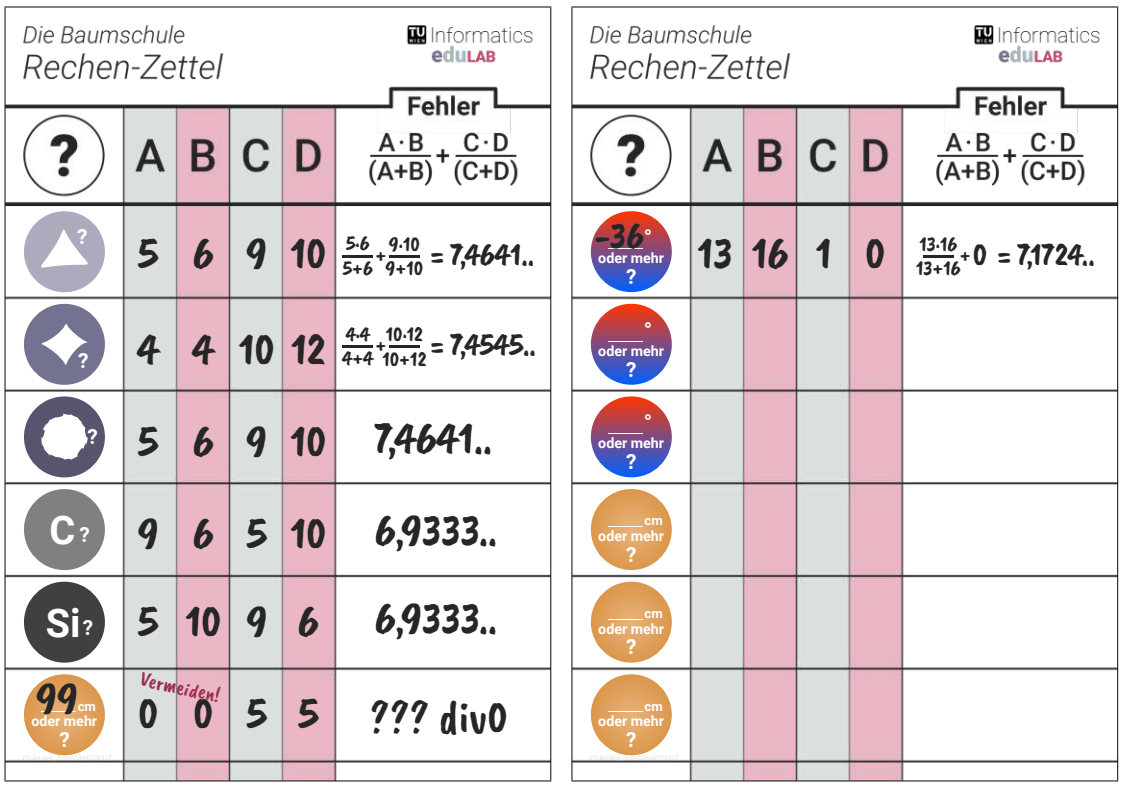


Figure 3.17: The front side (left) and back side (right) of a partially prefilled *Spreadsheet*.

Students are to simply test the four test data cards, write the results on them and return them to the teacher. The teacher waits for all groups to finish and hand in their classified test data cards before continuing.

Solution and Plenary Discussion. After having collected all groups' classified test data cards the teacher reveals to the class the class of each unknown (test) alien. A discussion on why the alien with catalogue number 34 was misclassified by every tree follows:

TEACHER: So, why do you think that one alien was misclassified by all of your trees?

[*Various answers come up and are discussed. The answer to be steered towards is that alien #34 is a bit on an outlier that happens to have the lowest cm value of any friendly alien - 63 cm- which is also lower than two dangerous aliens. This is why it got misclassified right away with the first split "66 cm or more".*]

[*TEACHER instructs all groups to mix all test data cards in with the training data cards and repeat the first branch/split to see how alien 43 gets misclassified.*]

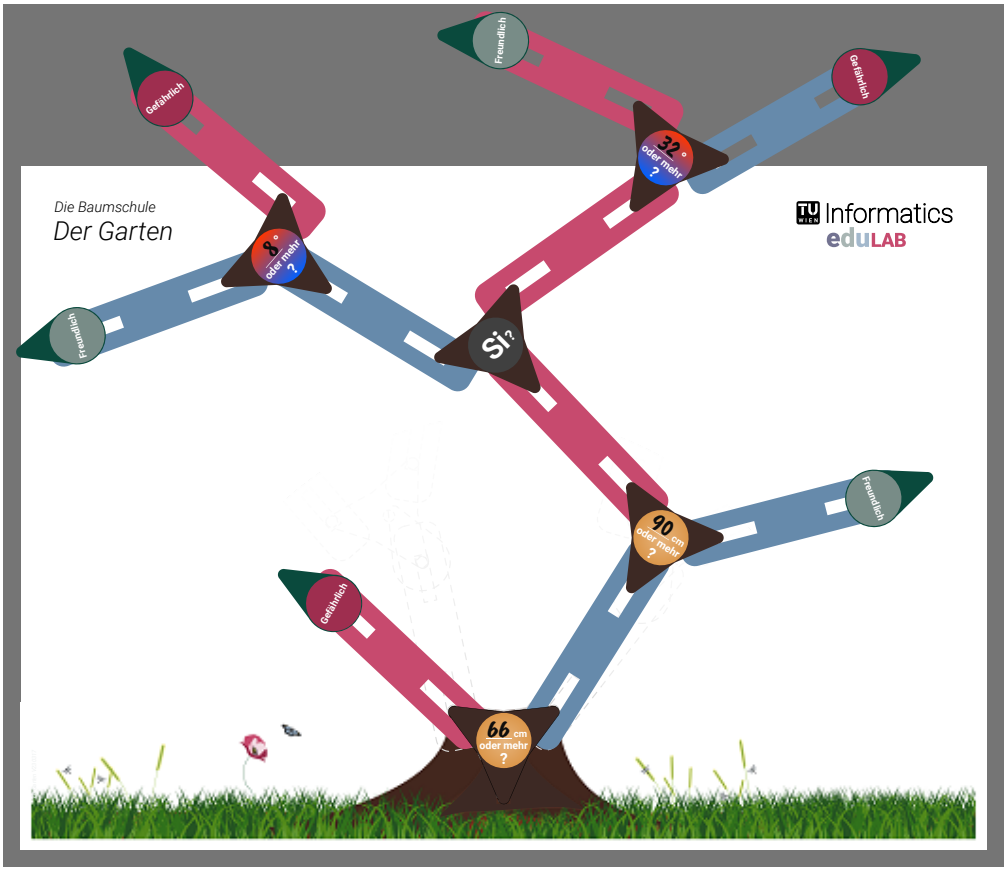


Figure 3.18: A mock-up of a possible solution to the sixth task *Your Mission*.

TEACHER: Can you think of a way to prevent such a misclassification?

STUDENT: More data!

TEACHER: Sure, that can help. But in a situation like ours, with limited information available, we don't have the luxury of being able to just add "more data"! Any other ideas?

STUDENT: Don't make such precise cut-off points when the two classes lie so closely together regarding that one feature.

TEACHER: Excellent! We could have used "61 cm or more" as first branch. We would not have taken the best split according to our formula the available training data, but we would not have misclassified alien #34.

TEACHER: And finally, if that also doesn't fix our problem, our model just might not be good enough to be used in a life or death situation. Especially when there is little data available, training a reliable machine learning model is hard or even impossible.

TEACHER: Can you think of any other situations - besides alien invasions - where such outliers exist? [*Short discussion on outliers in other domains.*]

Closing Plenary Discussion

This final plenary discussion closes out the activity:

TEACHER: What would you say, is the first kind of trees you grew today, the ones that recognised fruits, was that tree artificial intelligence?

[*Students answer both "Yes!" and "No!". TEACHER lets both camps explain why they think the fruits tree is or is not to be considered AI. After some back and forth TEACHER continues.*]

TEACHER: I'd argue that kind of tree is a type of artificial intelligence. It was surely artificial and it could do some simple problem-solving: Given some information about an unknown fruit it could decide which kind of fruit it is! But I'd also argue that this intelligence, i.e. problem-solving skill did not just appear out of nowhere. Where do you think it came from?

STUDENT: From us!

ANOTHER STUDENT: From the instructions!

TEACHER: Right! It seems like you and me, we worked together: I came up with the instructions and you decided how exactly to grow the tree. We manifested our intelligence as a physical machine, a system, that then could do simple problem-solving in a way that we could have done as well. Thus, this system just mimics our decisions, our intelligence. There is not more intelligence per se, rather our intelligence was funneled into a system that can be stored, copied, and even used by laymen - much like a book. But our system can also be automated by a computer!

TEACHER: So what do you think about the alien tree? Do you think it is artificial intelligence?

STUDENT: Yes!!

TEACHER: Do you see any differences between the fruits and the aliens tree?

STUDENT: One recognises fruits the other aliens!

ANOTHER STUDENT: The aliens tree grew by itself!

TEACHER: [*To the latter*] Yes! So how does that change things?

STUDENT: It's no longer our intelligence that is funneled into a system, but the system gathered intelligence by learning from examples.

TEACHER: Yes, I mostly agree! The system learned from examples, learned from data that we provided. After training, the system can now do things that we can not - at least not without us learning from the data as well. But is there none of our intelligence in that system?

STUDENT: None!

ANOTHER STUDENT: Some! We came up with the rules for the system. We came up with how a tree can recognise aliens and we decided how it should grow from data, i.e. which formula to use. When the system failed, we had a look and thought of ways of improving the system.

STUDENT: Correct!

TEACHER: Can anyone describe what the system actually learned?

STUDENT: Which question to ask and when.

TEACHER: Does anyone have ideas for what such a system could be used for - besides fruit recognition and alien invasions?

[*Let Students suggest areas of application. Ask what specific problems could be solved. Ask which data would be needed in those situations.*]

CHAPTER 4

Training an Artificial Neural Network Unplugged

This chapter discusses the design and development of an unplugged activity on the subject of training an artificial neural network and presents this activity in the last section.

4.1 The *Brain-in-a-Bag* activity

Peter McOwan and Paul Curzon created the activity *Brain-in-a-Bag: creating an artificial brain*[1]. This unplugged activity demonstrates the functioning of a neural network by letting students themselves become neurons and connecting them into a network that can play a simple game [MC14]. This network is made of three layers, with four neurons in the input layer, two neurons in a hidden layer, and one output neuron. The hidden layer neurons are both connected via rope to the output neuron and each to two different input neurons (meaning the network is *not* fully connected). Each rope has an empty toilet paper roll on it which enables the students/neurons to communicate with each other by holding up the rope and letting the roll slide down to the connected neuron. This mechanic inspired our activity, which adapts an extends it by letting the network be trained from examples.

The input to the network are cards of two different colors. Two red cards or two black cards being held up in front of the input neurons is considered a positive case and is supposed to trigger the output neuron after that information is propagated through the network. To that end, each neuron is given specific instruction when to fire:

[1] Available at https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-brain-in-a-bag-activity/

1. Two input neurons connected to the same hidden layer neuron are assigned a color and each a side, i.e., left or right. They fire when the card of their side shows their color.
2. The hidden neurons trigger upon receiving exactly two rolls.
3. The output neuron triggers upon receiving one roll.

Thus, the network can play a game of "Snap"[2], triggering when two of the same cards are shown.

We will now discuss how this activity was adapted and extended into our activity *Seven Little Neurons*.

4.2 Unplugging

The same distinctive features suggested by Nishida et al. [NKI+09] were used to guide the design process as in the previous chapter. Regarding students' required ability to perform mathematical operations during the activity, we must consider that this time students need to perform the operations *without* pen and paper or any other physical tool and by being given verbal instructions (i.e., equations are *not* handed out it written form). Thus we restrict the operations we consider to the following:

- $x + y$, with $x, y \in \{1, ..., 20\}$
- $x + y$, with $x, y \in \mathbb{Z}$, $max(|x|, |y|) < 10$
- $x * y$, with $x, y \in \{0, 1, ..., 10\}$
- $min(x_i)$ and $max(x_i)$, with $x_i \in \mathbb{Z}$, $i \in [1, .., 4], |x_i| < 100$
- x^2 with $x \in \mathbb{N}, x <= 10$

4.3 Simplifying Training

The task at the core of the training procedure of an artificial neural network is the search for the perfect set of values for each weight and bias. Finding a search strategy suitable for adaption in an unplugged setting is the main goal of this section. For information on ANNs see [Bie14].

The restrictions regarding mathematical operations rule out modern approaches involving *gradient descend*. A review of *low bit* [LLZJ17] and *binary neural networks* [YA22] was not fruitful. The nonlinear activation functions of a *multilayer perceptron* and again, the use of gradient descend during training, rules it out as well.

[2] https://en.wikipedia.org/wiki/Snap_(card_game)

If we are willing to accept to only partially train our network, e.g., only the weights of the connections between input and hidden layer, the learning rule of a single-layer perceptron (Algorithm 4.1, adapted from Algorithm 8.1 in [Bie14]) can be used to for our purposes: Let $X = \{x_1, ..., x_n\}$ be the training data and $y = \{y_1, ..., y_n\}$ the label vector with $x_i \in \mathbb{R}^D, y_i \in \{0, 1\}$. Let $w = (w_1, ..., w_d)$ be the set of weights. Let $\lambda > 0$ be the learning rate. Then the learning rule for a single-layer perceptron is as seen in Algorithm 4.1.

Algorithm 4.1: Single-layer perceptron learning

```
1 Randomly initialize the weights w.
2 repeat
3     for i ∈ {1, ..., n} do
4         Compute output o_i = sign(∑_{j≤d} w_j · x_ij)
5         if o_i ≠ y_i then
6             w = w + λx_i
7         end if
8     end for
9 until All training data are correctly classified or a set number of iterations are
    executed
```

Given the restrictions and setting of our targeted unplugged activity we make the following changes:

1. Weights must be positive integers (including 0), thus the learning rate must be integer. We set $\lambda = 1$.

2. All input is binary and the weights are positive, thus the prediction will always be positive, which is why the output must be changed to be either 0 or 1.

3. Output neuron(s) use the unit step function for activation (this function returns 0 until a threshold is reached, then returns 1).

4. We randomly initialize each neuron's activation function f_k with threshold t_k with $t_k \in \{2, .., 8\}$.

5. We randomly initialize the weights with integer values from 1 to 6 (by throwing a die).

With these changes a learning rule for a fully connected neural network made of four input neurons and two output neurons (the hidden neurons of our original network) can be formulated as shown in Algorithm 4.2. This way we can train the first two layers of the fully connected version of the *Brain-in-a-Bag* network.

4.4 Designing the Activity

Our activity changes and adds some materials (Figure 4.3) to facilitate the newly added training process:

- Weights are changed from toilet paper rolls to smaller, heavier tokens, so that up to 6 weights can traverse a neural connection.
- A "weight gate" is added to connections so that weights can easily be "removed" (stored behind the gate) and "added" (added back from storage) during training.
- Each connections can easily be added (and removed) from neurons using 3D printed connector clips.
- A random generator (a die in a jar) is introduced with which the weights and biases can be initiated.

Additionally, new cards are introduced: One new color to "test" the network and spark a discussion on why a network may fail. Cards with "0", "1", and "1/2" printed on to allow a discussion on how an ANN on a computer would receive data.

Our activity is designed as one long conversation between students and the teacher. During the first phase of this conversation all parts of a network are discussed and assembled into a network that can play "Snap", just like in the *Brain-in-a-Bag* activity. Then, however, students are confronted with the question *Do you think this network should be considered AI?*, which leads into a discussion about which elements ("factors") of the network are "hand-crafted" by a the teacher, i.e. a human, and how these factors can be reduced or removed. After removing three such factors the network will be in a generalized state that can no longer play the game "Snap". At this point the training process is introduced, with which the ability to play the game is relearned.

4.5 Presenting: *Seven Little Neurons*

4.5.1 Learning Objectives

For these first learning objectives, "ANN" refers to a multilayered artificial neural network with one hidden layer, a total of seven neurons, binary weights, and biases $\in \{1, 2\}$. When having completed the first part (*The Hand-Crafted Network*) of the activity *Seven Little Neurons*, students will be able to ...

1. Describe elements of an ANN and their function.
2. Demonstrate how an artificial neural network receives input to classify a data point.
3. Demonstrate how information is processed by an ANN from input to output layer.

4. Test an ANN.

For these second learning objectives, "ANN" refers to a multilayered artificial neural network with one hidden layer, a total of seven neurons, and weights and biases $\in \{1, ..., 10\}$. When having completed the second part (*The Self-Taught Network*) of the activity *Seven Little Neurons*, students will be able to ...

1. Understand how humans can influence an ANN.
2. Describe how an ANN learns from examples using a "learning rule".
3. Describe the training process for an ANN.
4. Describe what changes, i.e. is learned, during the training process on an ANN.
5. Understand that an ANN can be used for a different task by retraining the network.

4.5.2 Student Prerequisites

Students do not require prior knowledge on AI. All physical materials are language neutral, thus the language requirements are whichever language the teacher chooses to speak in. No reading or writing skills are needed, only speaking and listening. Students need to be able to execute an instruction made of one if-then-else statement. This instruction is presented in natural language and does not require knowledge of algorithms or programming. Students need to remember their assigned if-then-else statement. Students need to be able to count up to 10 objects and determine if the count is equal to or more than a fixed number (smaller than 10). Students will be engaged in an activity for approximately 70 minutes during which they are required to stand and hold, raise and lower objects connected by ropes to other students. Thus, students need to be able to stay focused and disciplined for that duration.

4.5.3 Target Setting and Audience

The activity can be held in- and outdoors with a group of 7 to 12 students (12 being the ideal group size). The teacher "directs" the entire activity by leading the continuous plenary discussion during which all other tasks happen. Open space of approximately 5 by 8 meters is required. The low technical prerequisites might indicate a possible target audience as young as primary school, but the last requirement (asking for focus and discipline over a 70 minute time span) leads to upper secondary school students as recommended target audience. The target duration for the activity is 90 minutes.

4.5.4 Materials

The following materials are required for this activity:

- Network building blocks (Figure 4.3):

  - 9 neuron sticks
  - 2 long connections
  - 8 medium connections with a weight gate at one end
  - 50 weights

- 1 portable random number generator (Figure 4.3)
- 2 data cards of each type: blue, red, green, zero, one, half (Figure 4.1)

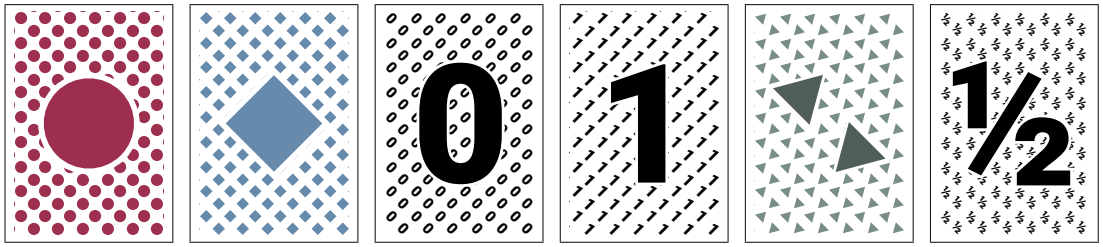


Figure 4.1: "Data cards" for the *Seven Little Neurons* activity. Each to be printed in size A5. From left to right: "red", "blue", "zero", "one", "green", and "half".

4.5.5 Activity Rundown

The entire activity *Seven Little Neurons* (German original: *Die Sieben Kleinen Neuronen*) is one continuous conversation between the teacher and the student. During this conversation students are given one of five different roles, becoming part of an artificial neural network, representing data input or acting as a referee and random number generator. This neural network is first "hand-crafted" by the teacher so that it can "play" a simple game. Later the network is generalised and human influence is reduced (by adding neural connections and randomising parameters), at which point the network will not be able to play the game anymore. Here the concept of training a neural network is introduced and the network is trained until it can play the game again. The activity has four main parts:

1. Introduction and Plenary Discussion
2. The Hand-Crafted Network
3. The Self-Taught Network
4. Closing Plenary Discussion

Introduction and Plenary Discussion

The activity starts with the same introduction and plenary discussion on AI and ML as the decision tree learning activity (*Introduction and Plenary Discussion* in Section 3.4.4), except that the last question is changed to *It is called an "artificial neural network", has anyone heard of that before?*. Note that the answers given by students in conversations shown below represent a best case scenario. It is unlikely that every answer will be given as shown. Whenever no desired answer is given the teacher should steer the students towards that answer by asking more questions, giving hints, or finally resolving the question.

The Hand-Crafted Network

Distribute Roles and Position Students. If there are 12 students available, the roles (Table 4.1) can be distributed as shown in the discussion below. Students are positioned to match the "initial setup" as shown in Figure 4.2. If fewer students are available, the dealers and referee can be played by the teacher and the hidden neurons (in) and (out) can be played by one student each, instead of two.

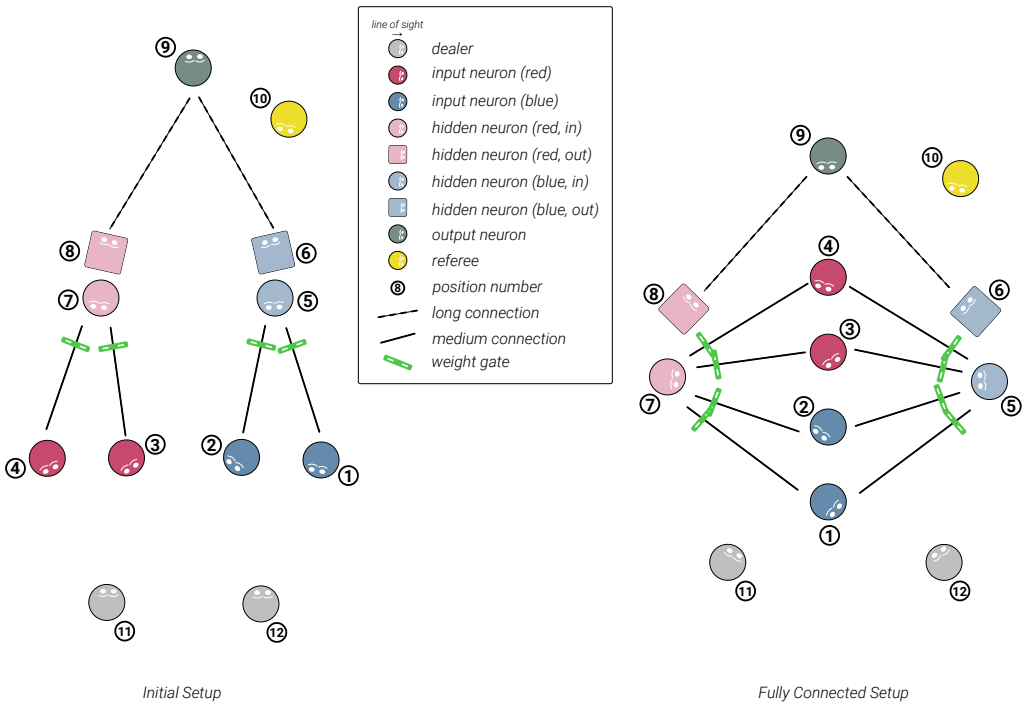


Figure 4.2: Left: Initial setup of students for *The Hand-Crafted Network*. Right: Fully connected setup after repositioning students for *The Self-Taught Network* and fully connecting the input and hidden neurons.

dealer (left)	Carries a blue and a red data card. Holds up one of the cards whenever a round of the game is started.
dealer (right)	Carries a blue and a red data card. Holds up one of the cards whenever a round of the game is started.
input neuron (right, blue)	Triggers when dealer (right) holds up a blue card.
input neuron (left, blue)	Triggers when dealer (left) holds up a blue card.
input neuron (right, red)	Triggers when dealer (right) holds up a red card.
input neuron (left, red)	Triggers when dealer (left) holds up a red card.
hidden neuron (blue, in)	Triggers (notifies hidden neuron (blue, out)) when receiving two or more weights.
hidden neuron (blue, out)	Triggers when notified by hidden neuron (blue, in).
hidden neuron (red, in)	Triggers (notifies hidden neuron (red, out)) when receiving two or more weights.
hidden neuron (red, out)	Triggers when notified by hidden neuron (red, in).
output neuron	Triggers when receiving exactly one weight.
referee	Judges the network's performance after each game, i.e. compares the input data cards to the output neuron's reaction. Generates random numbers on demand.

Table 4.1: Students' initial roles and their behaviour for the *Seven Little Neurons* activity.

The conversation between teacher and students during the setup of the hand-crafted network goes as follows.

TEACHER: So we want to build a neural network that can play a game! The game is simple: Two cards are held up. If they are the same color the player - in our case the neural network - must shout out "Same!" and otherwise stay silent. But how do we make that happen?

Well, we need to build a neural network!

I need 4 people with relatively good eyesight, please. [*Position the volunteers at positions 1-4 and hand each one a neuron stick.*] Great, thank you, you are my "input neurons".

Now I need 4 people that are not afraid to count to ten! [*Position the volunteers at positions 5-8 and hand each one a neuron stick.*] Great, thank you, you are my "hidden neurons". Actually, you are only two hidden neurons, not four, but

because you will have a tougher job than the others I split your job in two, hope that's OK.

Now I need one person that can speak. No worries, it's just going to be one word. [*Position the volunteer at position 9 (facing away from positions 11 and 12) and hand them a neuron stick.*] Thanks, you are my "output neuron".

Great, now we have all the neurons we need to play the game! But every game needs a "referee", right? So the next person I need has to be fair and impartial. Is anyone like that here? [*Position the volunteer at position 10 and hand them the random number generator.*] Great, you are my "referee".

And we can't play a game of cards without cards and someone to deal them! That leaves the two of you to be my "dealers". Have you held up a card before?

STUDENT: Yes!

TEACHER: Great, that means you're qualified for the job. [*Position the two remaining students at positions 11 and 12 and hand each two cards: one red and one blue.*]

TEACHER: Looks like we are ready to go. But before we start I have some bad news for the neurons: You are neurons. Not the complete, intelligent humans anymore! And a neuron can't really do much. What you all *can* do is stand at your position and raise and lower your arms.

Input neurons, you are additionally allowed to receive one input! That means you are allowed to look at one dealer and any card he/she is holding up, but not at the other dealer or their cards!

Hidden neurons, you can only see your immediate surroundings! That means you cannot see other neurons or the cards.

Output neuron, you can also not see any neurons or cards. But you are the only one allowed to say something during the game, to say one word, whenever you want to. Can you guess which word that is?

OUTPUT NEURON: Same!

TEACHER: Correct! Well, looks like now we're really ready to start.

TEACHER: Let's play a round! Dealers, please each hold up one of your cards!

[*Wait for dealers to hold up cards. Wait a bit for dramatic effect. Nothing should happen. Then continue.*]

[*In case of different colors*] Great! That worked perfectly. The cards were different and the network stayed silent. That's a point for the neural net! Let's try this again! Dealers, please hold up cards of the same color this time, just so we can make sure the network works in all cases.

[*In case of same colors*] Hmm, something seem to be off. We should have gotten a "Same!". The cards where the same color after all. Referee, what do you say?

REFEREE: Yeah, I guess!

[*Instruct Referee on their role.*]

TEACHER: Does anybody have an idea what we're missing here? What does the network need in order to play better?

STUDENT: The neurons need to communicate with each other!

TEACHER: Right, but how can that work? Most of you are not allowed to speak!

STUDENT: The neurons need to be connected, like in a real network.

TEACHER: Right! Let's do that.

Connect Neurons. The teacher connects the neurons according to the initial setup shown in Figure 4.2. They may call on other available educators for help to speed up the process, or ask the referee and dealers to help.

TEACHER: Great, we've connected some of you. Do you think it'll work now? Will you shout out when cards of the same color are "dealt"?

STUDENT: No!!

TEACHER: Right, you might be connected but you can't communicate with each other yet! That's why I brought these little "weights" for you.

[*Teacher hangs one weight on each connection.*]

TEACHER: Try it out!

[*Let students discover that by sending weights from one end to the other they can communicate.*]

TEACHER: So, how about now? Do you think this neural net can play the game and win?

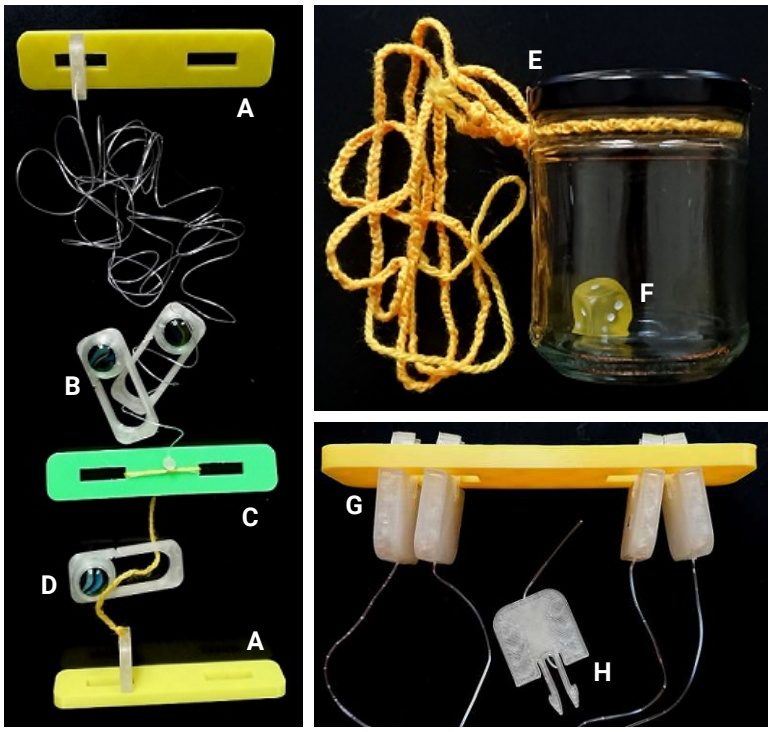
STUDENT: Yes!

ANOTHER STUDENT: No!

TEACHER: Let's try it out!

[*Initiate another round of the game like before. At least one mixed input and one equal should occur with the same outcomes as before. Let the referee judge each outcome.*]

TEACHER: That still didn't quite work like we wanted to.



A – neuron stick
B – weights
C – weight gate
D – inactive weight
E – random number generator
F – six sided die
G – neuron stick
(connected four times)
H – connector in profile

Figure 4.3: Left: A medium connection with weight gate and weight count 2 (weight "D" is "behind" the weight gate an thus does not currently count towards the weight count). Top right: A random number generator with a cord to hang around the neck. Bottom right: A neuron stick with four out of eight possible connections attached.

Determine Neuron Behaviour. Neurons have not been told their role's exact behaviour (Table 4.1). Students are free to discover these behaviours on their own, but will be given assistance as needed.

TEACHER: Anybody got any ideas what we could do to make the network work as intended?

[*Let students discuss what to do. If they figure out how to make the network work, great! Otherwise lead them towards the right setup.*]

TEACHER: It's not easy, but let's see what we can do.

Input neurons, you are the only ones who can see any cards. [*Try and lead them towards their job description.*]

Hidden neurons, you are the link between input and output. What could your job be? [*Try and lead them towards their job description.*]

Output neuron, now your job should be clear, what do you think you have to do?

OUTPUT NEURON: Shout "Same" when I receive a weight/signal!

TEACHER: Right! Great! Let's try it out. [*Play another round of the game. The neural network should now perform as intended for all possible card combinations.*]

Discussion.

TEACHER: Looks like we have a functioning artificial neural network that can play a game! That's great! Now, what do you think, should this network be considered "artificial intelligence"?

[*Students answer both "Yes!" and "No!". Teacher lets both camps explain why they think the network should or should not be considered AI. After some back and forth Teacher continues.*]

TEACHER: I'd argue that this network is a type of artificial intelligence. It is surely artificial and it can do some simple problem-solving: Given two cards it can tell whether they are of the same color or not. That seems somewhat intelligent to me! But I'd also argue that this intelligence, the problem-solving skill, did not just appear out of nowhere. Where do you think it came from?

STUDENT: From us!

ANOTHER STUDENT: From you!

TEACHER: Right! It seems like you and me, we worked together: I came up with how to position and connect the neurons and together we figured out how each of the neurons has to behave for the network to play the game! It seems like we manifested our own intelligence as a physical machine, a system, that can now do simple problem-solving in the exact way we set it up to do. Thus, this system just mimics our decisions, our intelligence. There is not more intelligence here than before, rather our intelligence was funneled into a system that can now work on its own, can even be stored, copied and even used by someone else, someone who doesn't even know how to play the game. Which is great! But let's try and go even further.

The Self-Taught Network

The goal of this task is to let student discover in which ways the previous network was influenced by a human intelligence. Three of these "human factors" will be removed one by one and replaced with a generalised or randomised version. This new network will not be able to play the game anymore, which is why a learning rule is introduced that lets the new network *learn* how to play the game.

Human Factors. In this activity a "human factor" can be any aspect of the previous neural network that was determined or influenced by the teacher and/or the students or any action performed by the neurons while playing the game that seem to require human intelligence. The following human factors may be discovered by students:

1. Which neurons are connected.
2. The number of weights per connection.
3. The conditions under which each neuron fires.
4. Input neurons see and recognise the colors on the cards.
5. The number of neurons and how they are arranged.
6. Other actions neurons perform, such as
   - raising and lowering arms,
   - shouting a word,
   - counting weights,
   - deciding whether to fire based on the number of weights.
7. The rule by which the network learns from examples.
8. The learning rate, i.e. the degree to which the weight count is altered during each round of training.

Factors 1 to 3 are supposed to be discovered by the students during the upcoming discussion and will be subsequently removed (see discussion below). If factor 4 is discovered, the teacher should agree with the students, but tell them to ignore that factor for now, as it will be dealt with in the final discussion. If factor 5 is discovered a discussion can be had on how computer scientists put a lot of effort into finding out what the right neuron count and setup for a given problem is and that this is one way how humans influence artificial intelligence: by setting the basic framework. If any of the actions in factor 6 are discovered, each can be dismissed as something a simple machine could do, and thus should not be considered a human factor. Factors 7 and 8 can only be discovered after the learning rule is introduced. Both can be handled like factor 5.

Finding and Removing Human Factors. At this point students are still in the same roles and positions as before. Students may switch roles amongst each other now, if they want to. Then the conversation continues with the teacher explaining what a "human factor" is and asking the students to spot human factors in the current neural network. If any factors besides 1, 2, or 3 are spotted, the teacher acts as described in the previous paragraph. The conversation for factors 1 to 3 is as follows:

TEACHER: Has anybody spotted a "human factor"?

[*Human factor 1 is discovered.*]

STUDENT: You chose which neurons to connect. That's a human factor!

TEACHER: Correct! So how can we reduce or remove it?

STUDENT: Connect all the neurons!

TEACHER: Yes! I mostly agree. We will connect all the input neurons with all the hidden neurons. [*Teacher starts repositioning students to match the fully connected setup (Figure 4.2)*] To do that we need to first reposition you a little bit, so that we don't get any crossing connections. That would hinder the free flow of weights. [*After repositioning, Teacher starts adding the connections, including weights.*] We don't connect the input neurons to the output neuron, because I want to keep this simple layer-by-layer setup. Input layer, hidden layer, output layer. There is no specific reason, other than I think it will work better this way. This is a smaller human factor that will remain. Computer scientists actually put a lot of effort into finding out what the right layer setup for a given problem is. And that is one way how humans influence artificial intelligence: by setting the basic framework.

[*Human factor 2 is discovered.*]

STUDENT: You chose to put exactly one weight on each connection. Why not more, or less?

TEACHER: Yes! Let's change that. What would happen if we removed the single weight from a connection?

STUDENT: The two neurons could not communicate anymore, as if they weren't connected.

TEACHER: Right, that doesn't seem like a good idea right now, but we could add more weights! How many should we add?

STUDENT: 1! 3!

ANOTHER STUDENT: A thousand!

TEACHER: Not quite! See, any number you could have told me would have had the same issue: A human came up with it! Why is that an issue?

STUDENT: We want to reduce human influence on the system.

TEACHER: Right! So what do we do?

STUDENT: Use the random number generator!

TEACHER: Yes! Let chance decide the number of weights, not humans. [*Instruct* REFEREE *to roll the die and shout out the result whenever called upon. Add weights to the connections between input and hidden layer, s.t. the weight count matches the die roll.*] We will only add weights to the connections between the first two layers. We could add weights to the output neuron's connections, and if this were an artificial neural network on a computer we would definitely do that, but here we want to keep it simple.

[*Human factor 3 is discovered.*]

STUDENT: You/We decided when each neuron fires and when not!

TEACHER: Correct! Let's change that. What can we do?

STUDENT: Use the random number generator!

TEACHER: Yes! To keep things simple we're leaving the output neuron the way it was. But the hidden neurons, you get a new random number. Let's simply add a die roll to each of your current thresholds. [*Referee rolls the die for each hidden neuron and the value is added to their threshold.*]

If any of factors 1 to 3 are not discovered by students, the teacher must lead them towards them.

Introducing the "Learning Rule". After generalising the neural network in the previous step, it will most likely not be able to play the game anymore. This is tried out and then addressed by finding the "learning rule" (Algorithm 4.2).

Algorithm 4.2: The Learning Rule

1. Let dealers deal a random card combination.
2. Play a game with the card combination.
3. If the network behaved correctly go to step 1.
4. Else, both hidden neurons (in) adapt the weight count of their connections to the two input neurons that fired as follows: If the hidden neuron itself was supposed to fire, but didn't, then add a weight, else remove a weight. Go to step 2.

TEACHER: We have a whole new network now, with way less human influence. Let's try it out and see how well it plays the game!

[*Play a game with every card combination. The network should fail on at least one of them.*]

TEACHER: Looks like our network lost its ability to play the game right! Why do you think that is?

STUDENT: It just does random stuff.

ANOTHER STUDENT: We need to remove those weights there. [*Student points at some connection.*]

TEACHER: So yes, it behaves pretty randomly. But if we start fixing it by deciding the weights ourselves again, we would just put a human factor back into the system and go back to where we were before! We don't want that. But as we've discussed in the beginning, a machine learning system is a system that can learn. How could this system learn?

[*Students answer and their answers are discussed. If no student comes up with the learning rule we are looking for:*]

TEACHER: There's a saying about how we learn. "We learn from our ..."

STUDENT: Mistakes!

TEACHER: Right, so when the network makes a mistake, it has an opportunity to learn. What does "learning" mean in this context?

STUDENT: Changing the number of weights!

TEACHER: Yes! [*Teacher proceeds to lead students towards the learning rule.*]

Training the Neural Network. At this point students understand how the network can play the game and how it can learn from mistakes, i.e. be trained. What remains is to repeatedly apply the learning rule (Algorithm 4.2) until the network reacts correctly to every card combination.

Notes on the application of the learning rule:

1. The learning rule itself has no stopping condition, thus the teacher has to step in at some point to suggest trying all card combinations to see if all succeed.
2. The learning rule immediately repeats failing card combinations until the network reacts correctly before moving on to other combinations in order to speed up the training process.
3. Hidden neuron (red) / (blue) is "supposed to fire" when two red / blue cards are dealt.
4. A hidden neuron adds a weight to / removes a weight from a connection by moving a weight over the weight gate away from / towards itself.
5. If a connection has no more weights, but a hidden neuron has to remove a weight, no change shall occur.

Once the training process has finished the final step is initiated.

Introducing Disruptors. Several new data cards (Figure 4.1) are introduced to spawn the following discussions:

TEACHER: [*Teacher covertly hands the dealers one green data card each and takes away the red and blue ones.*] So our network has now learned to recognise whether two cards have the same color, right?

STUDENT: Yes!!

TEACHER: Great, let's try that one last time. Dealers please!

[*Dealers hold up the green cards. Nothing happens.*]

TEACHER: Referee, what do you say?

REFEREE: We should have gotten a "Same!" from the output.

TEACHER: Right, so what went wrong here?

STUDENT: The colors are wrong!

TEACHER: But they are the same on both cards and our network can detect that, right?

STUDENT: The network was not trained on green cards!

TEACHER: Right! So we should be careful what we say our network can do. It really can only recognise whether there are two red or two blue cards, but not for any other colors. Could the network *learn* to recognise two green cards in addition to red and blue? [*Let students discuss. Possible solution is adding two more neurons to the input layer and one more to the hidden layer in order to recognise a new color.*]

A short discussion on the versatility of neural networks is had.

TEACHER: So, neural networks - the ones in our head as well as artificial ones - a very versatile. We could in our case for example make the network to play the opposite game. That is, to shout out when the two cards are different! How could that work?

STUDENT: More neurons?

ANOTHER STUDENT: Just training!

TEACHER: [*To the latter.*] Right! Just training. We just need to set the new goal and let the network learn!

[*If there happens to be enough time (and motivation) left, a new training process can be started. Alternatively the network can be set up manually to demonstrate it can play the opposite game, or just explained verbally.*]

TEACHER: If fact this network can learn to "shout" for all sorts of card combinations, like two red *and* two blue at the same time, or two blue and one red, etc. All we need to do is set the new goal and let the network learn!

Now the "one" and "zero" data cards are used to discuss human factor 4 *Input neurons see and recognise the colors on the cards.*

TEACHER: I still owe you a human factor: The input neurons see and recognise the colors of the cards their dealer holds up! To me that seems a bit too intelligent for a single neuron - no offense. Can anyone think of a way to remove this human factor?

STUDENT: Use numbers instead!

TEACHER: Right, computers can handle numbers pretty well. What numbers would you suggest?

STUDENT: Zero and One!

TEACHER: Good idea! I happen to have cards with those numbers on them here. But if we use those instead of the colors, the input neurons still have to recognise the number and remember what to do in each case. Still pretty intelligent behaviour for a neuron. Can you think of a mechanism so simple "even" a computer could do it? [*Let students discuss. Intended solution is giving each input neuron one card directly (i.e. add two dealers) and using multiplication: Input neurons always fire, but the amount of weights they send is multiplied by the input value.*]

TEACHER: Great! That's how artificial neural networks on a computer also do it! Now what should an input neuron do with this card? [*Teacher holds up the "half" data card.*]

STUDENT: Send half as many weights!

TEACHER: Yes! What would we do with a quarter card? Or a fifth?

STUDENT: Send only a quarter of the weights, or a fifth!

TEACHER: Exactly! In many real world examples we do not only have zeros and ones as input, but all kinds of numbers! And we can now deal with all of them.

Closing Plenary Discussion

Students may leave the their roles behind now and return the materials to the teacher. A final discussion follows.

TEACHER: So what do you think. Should our second network be considered artificial intelligence?

STUDENT: Yes!!

TEACHER: What difference is there to the first network?

STUDENT: There was less human factors / human influence.

ANOTHER STUDENT: The second network could play the game because it learned from examples and not because we told it how to play.

TEACHER: [*To both*] Yes! So how does that change things?

STUDENT: It's no longer our intelligence that is funneled into a system, but the system gathered intelligence by learning from examples.

TEACHER: Yes, I mostly agree! The system learned from examples, learned by playing the game and receiving feedback. After training, the system can now play the game without us having directly decided every little detail. But is there none of our intelligence in that system?

STUDENT: None!

ANOTHER STUDENT: Some! We came up with the rules for the system. We came up with how the layers are set up, and how many neurons there are in each.

STUDENT: Correct!

TEACHER: Can anyone describe what the system actually learned during training?

STUDENT: How many weights to put where.

TEACHER: Does anyone have ideas for what such a system could be used for - besides playing this simple game?

[*Let students suggest areas of application and/or examples for software that is based on artificial neural networks. Discuss how this technology is in widespread use.*]

CHAPTER 5

Evaluation

5.1 Distinctive Features of Unplugged Activities

As previously mentioned, this evaluation is based on the "distinctive features" that Nishida et al. have identified in [NKI+09].

5.1.1 *The Tree Nursery*

We try and answer the question: Does the activity *The Tree Nursery* exhibit the "distinctive features" of a CS Unplugged activity?

1. "No computers": Partially: In the first phase of the activity, no computers are used. In the second phase students are allowed to use pocket calculators to calculate the *Simplified Gini Impurity* of several splits. This of course breaks with the "No computers" premise, however, there is a case to be made that pocket calculators are by comparison low-tech devices and thus do not violate this rule. Alternatives for the *Simplified Gini Impurity* may have to be investigated to completely remove the use of computers from this activity.

2. "Games" or "Challenges": Also in parts. The second phase's task can be considered a challenge: a race against time.

3. "Kinaesthetic": Yes! All elements of a decision tree are 3D printed objects and all data is represented as playing cards. These materials can be freely explored and manipulated by the students.

4. "Student directed": Yes. All tasks are group work and incorporate discovery based learning as much as possible.

5. "Easy implementation": No. In their current form, tree parts require a 3D printer and cutting out the data cards is tedious and requires several hours per set.

6. "Growing body of ideas": Yes, the activities will be used at *eduLAB*, studied, developed further and shared.

7. "Sense of story": Yes. For the first phase it is an almost zen-like story of "growing a tree in your garden" and the second phase serves a fantasy rich story line in which students get so save the world from an alien invasion.

5.1.2 *Seven Little Neurons*

Again we try and answer the question: Does the activity *Seven Little Neurons* exhibit the "distinctive features" of a CS Unplugged activity?

1. "No computers": Yes, this is completely fulfilled this time.

2. "Games" or "Challenges": Yes, while the activity itself is not a game, the network is trained to play a game, which serves a similar purpose.

3. "Kinaesthetic": Yes, every element of the neural network is made of tangible objects students get to interact with. Data is represented by big cards.

4. "Student directed": Not quite. The activity is led by the teacher over the entire duration. And while students are constantly engaged in a conversation with the teacher, who tries to have them discover concepts and ideas themselves by posing questions, they are not free to work on their own or small groups to solve tasks.

5. "Easy implementation": Yes, the materials presented here might require a 3D printer, but they can be easily replaced by more available alternatives. E.g., by simply using sticks and binding the ropes to them and using any ring-like token with a small opening, such as shower curtain hangers, for weights.

6. "Growing body of ideas": Again, yes, the activities will be used at *eduLAB*, studied, developed further and shared.

7. "Sense of story": No. We refrained from adding any distractions from the core principal that this activity wants to get across: Neurons in an ANN are small, non-intelligent calculators. Giving the neurons any roles beyond that would inhibit this goal. But otherwise a story might be added as to what the network might be used for, instead of just playing a game.

5.2 Piloting the Activities

The aim of piloting the activities was to determine whether

- students understand the instructions and execute the tasks as intended, and
- students can train the models in a workshop setting to high accuracy in an appropriate time span.

The activities were piloted during two workshops at *eduLAB*, each with a class of 9^{th} graders. Both activities were well received.

During the *The Tree Nursery* activity students understood all instructions without much added assistance and executed the tasks as intended. However, the originally planned duration of 45 minutes was exceeded: After 60 minutes the first best split was found and students were working together on the second. Then the activity had to end due to time constraints. Thus, the targeted activity time is now set at 90 minutes to have enough time to find the rest of the splits and then the plenary discussion.

The *Seven Little Neurons* activity has no written instructions but the continuous conversation between students and teacher worked as intended to lead students through the activity. The activity lasted the intended 90 minutes with enough time for a plenary discussion at the end.

5.3 Drawbacks, Limitations and Future Work

The mayor drawback of the *The Tree Nursery* activity seems to be the cost of the materials and the requirement of a calculator. The initial cost and technical know-how required to install and run a 3D-printer will prevent many educators from implementing this activity themselves. Cutting out the data cards takes time, but can be done with enough conviction. Further research may reveal a way to eliminate the use of a calculator entirely.

The *Seven Little Neurons* activities lacks student independence and self-directed work, as they have no opportunity to reflect, discuss or discover on their own, without the teacher being involved. This, however, lies in the nature of the continuous plenary discussion. Further evaluation with students should reveal how much of a drawback this is. From the technical perspective it would be desirable to train a bigger ANN and train the entire network, not just parts.

Other future work may include:

1. Further evaluate materials: Investigate how the activities impact students' understanding of ML.
2. Make more accessible version of materials: Research low-cost alternatives for the decision tree parts.
3. Create follow-up materials: Create versions of the activities' tasks in excel, s.t. teachers may use them after having completed an activity to teach "regular" excel

functionalities, but in a context students are already familiar with (e.g., classifying fruits with a decision tree) and find engaging.

4. Extend ANN activity: Investigate how larger networks can be represented and trained in an unplugged setting.

CHAPTER 6

Conclusion

Unplugged learning activities allow educators to teach complex computer science concepts in an engaging way that is appropriate for children. Several such activities have been proposed on the topic of artificial intelligence and various machine learning concepts and techniques. Many, however, do not cover the concept of training a ML model, or simplify it so much that students may fall under a false impression of what the training process actually involves.

In this thesis, we presented two unplugged activities that let students train a decision tree and an artificial neural network. These processes are simplified so that they can be executed in an unplugged setting by upper secondary school students without prior knowledge on AI. They aim, however, to provide a realistic look at and discuss how these processes would work on a computer.

To this end, we proposed using a *Simplified Gini Impurity* for finding the best split during the training of a decision tree. We further presented a "data card" design that facilitates the calculation of the best split by allowing students to count up class occurrences for the needed variables without requiring prior sorting of the data. For the training of an artificial neural network we adapted and extended an existing unplugged activity. We introduced training to this activity by using a positive integer only learning rule on the first two of three layers of the network.

During piloting our activities were well received, albeit them being longer than anticipated. Other drawbacks include the production costs of the decision tree activity and the that the neural network training procedure only covers parts of the network.

The presented activities will become part of the *TU Wien Informatics Didactics eduLAB* catalogue of available workshop activities. As such they will be developed further, studied and explored. They will be in regular use for student workshops and thus have a chance to realise the impact they were designed for.

List of Figures

List of Tables

Bibliography

[Bie14] Alain Biem. Neural Networks: A Review. In *Data classification: Algorithms and applications*. Chapman and Hall/CRC, 2014.

[BWF98] Timothy C Bell, Ian H Witten, and Mike Fellows. *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged, 1998.

[Cla19] Beverly Clarke. Alternate Unit: Artificial Intelligence, June 2019.

[LLJ14] Victor E. Lee, Lin Liu, and R. Jin. Decision Trees: Theory and Algorithms. In *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC, 2014.

[LLZJ17] Cong Leng, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM, September 2017. arXiv:1707.09870 [cs].

[LSR19] Annabel Lindner, Stefan Seegerer, and Ralf Romeike. Unplugged Activities in the Context of AI. In Sergei N. Pozdniakov and Valentina Dagien, editors, *Informatics in Schools. New Ideas in School Informatics*, Lecture Notes in Computer Science, pages 123–135, Cham, 2019. Springer International Publishing.

[MC14] Peter McOwan and Paul Curzon. The Brain-in-a-bag Activity, November 2014.

[MC16] Peter McOwan and Paul Curzon. The Sweet Learning Computer, July 2016.

[MSM+23] Ruizhe Ma, Ismaila Temitayo Sanusi, Vaishali Mahipal, Joseph E. Gonzales, and Fred G. Martin. Developing Machine Learning Algorithm Literacy with Novel Plugged and Unplugged Approaches. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, pages 298–304, New York, NY, USA, March 2023. Association for Computing Machinery.

[NKI+09] Tomohiro Nishida, Susumu Kanemune, Yukio Idosaka, Mitaro Namiki, Tim Bell, and Yasushi Kuno. A CS unplugged design pattern. *ACM SIGCSE Bulletin*, 41(1):231–235, March 2009.

[OB19] Elisaweta Ossovski and Michael Brinkmeier. Machine Learning Unplugged - Development and Evaluation of a Workshop About Machine Learning. In Sergei N. Pozdniakov and Valentina Dagien, editors, *Informatics in Schools. New Ideas in School Informatics*, Lecture Notes in Computer Science, pages 136–146, Cham, 2019. Springer International Publishing.

[Vir21] Patrick Virtue. GANs Unplugged. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):15664–15668, May 2021. Number: 17.

[YA22] Chunyu Yuan and Sos S. Agaian. A comprehensive review of Binary Neural Network, February 2022. arXiv:2110.06804 [cs].