Looking Beyond Supervised Classification and Image Recognition – Unsupervised Learning with Snap*!*

Tilman Michaeli, *tilman.michaeli@fau.de*
Computing Education Research Group, FAU Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen

Stefan Seegerer, *stefan.seegerer@fau.de*
Computing Education Research Group, FAU Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen

Sven Jatzlau, *sven.jatzlau@fau.de*
Computing Education Research Group, FAU Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen

Ralf Romeike, *ralf.romeike@fu-berlin.de*
Computing Education Research Group, FU Berlin, Königin-Luise-Str. 24-26, 14195 Berlin

Abstract

The possibilities of machine learning and its influence on our everyday lives are expanding rapidly. Computer science education must react to these developments. To this end, many new tools and approaches have been developed recently, particularly for novices. Whether "unplugged" or "plugged", most of these approaches focus on supervised learning as one of the three paradigms of machine learning – while the potential of the other two paradigms remains untapped. Furthermore, these approaches focus on mere usage of machine learning models by using external libraries for the actual learning. While learners are able to use machine learning in existing artifacts, they are not able to create their own or gain a deeper understanding of them. As the learning process itself is not addressed or visible for the learners, machine learning remains a black box. To address this problem, we developed SnAIp, a project aimed at enabling machine learning within the block-based language Snap*!*. We outline design principles guided by constructionist learning theory with the goal to break open the black box. Building upon these principles, we present a framework and curriculum for unsupervised learning in the classroom enabling learners to create and understand machine learning artifacts.

SnAIp Part B – unsupervised learning in Snap!

Keywords

AI, artificial intelligence, computer science education, Snap*!*, block-based, programming, machine learning

Introduction

Artificial intelligence (AI) is increasingly becoming the subject of attention due to its societal relevance in many sectors. Primarily driven by advances in machine learning (ML), it is now also discussed in pedagogical contexts (e.g. Touretzky et al., 2019; Heinze, Haase, & Higgins, 2010). Consequently, AI is also featured in international curricula (e.g. CSTA, 2017; Yu & Chen, 2018) with machine learning being a prominent aspect.

Novices are the target audience for many existing pedagogical frameworks. Typically, these frameworks show simple and intuitive ways for learners to use ML concepts – without the hurdles introduced by programming (Kahn et al., 2019). Many of these approaches share a common feature: they focus on supervised learning. Often, this results in the focus being shifted toward pattern recognition and supervised learning being equated to ML, while the potential offered by often-overlooked ML algorithms remains untapped. We argue that the potential of unsupervised learning (UL) is that it highlights the simplicity of ML algorithms particularly clearly (as ultimately all ML algorithms aim to make a simple machine appear intelligent). Furthermore, it helps in providing a broader view on how machines actually can “learn”.

Another common trait of existing approaches is that the underlying AI framework remains a black box. In many cases, this is due to the actual computing being done by external API calls or servers - which severely limits the ability of the user to “look inside”. This limitation makes it difficult for learners to gain a deeper understanding of the actual machine learning process. According to the constructionist learning theory (Papert & Harel, 1991), knowledge is constructed through the interaction of creating an artifact and understanding it. To allow constructionist learning in the context of ML, students need to go beyond the mere usage of ML; they also need to understand what they build. By incorporating both the creation of ML artifacts (rather than simply using them) and supporting their understanding, learners construct knowledge about ML. This deep, fundamental understanding is necessary if learners are expected to analyze the effects of AI on our society in a competent way.

In this paper, instead of focusing on supervised learning, we present approaches to another area of ML suitable for classrooms. The *SnAIp framework* for unsupervised learning is based on an implementation entirely in the Snap*!*-environment to enable constructionist learning by “breaking open the black box”. After outlining the framework and its design principles, we also present parts of a corresponding curriculum to incorporate these topics into teaching.

Related Work

Paradigms of Machine Learning

The subject area of machine learning deals with algorithms that improve through experience over time (Mitchell, 1997). Machine Learning can be divided into the three paradigms *supervised*, *unsupervised* and *reinforcement learning*. Some authors further diversify this categorization, e.g. by adding *semi-supervised learning*, a combination of supervised and unsupervised approaches (Ayodele, 2010), or *evolutionary learning* (Marsland, 2014).

In supervised learning, the algorithm is given a set of sample inputs and corresponding outputs. It learns of the connection between input and output data independently and can then generalize it to any input data (Marsland, 2014).

The situation is different with unsupervised learning: the algorithm is also given a number of inputs, but there is no corresponding output, yet. Hence, the algorithm needs to identify similarities in data inputs in order to classify the data and assign an output label.

Reinforcement learning is a paradigm of machine learning inspired by psychology: The agent learns through reward and punishment. It learns to master a certain task autonomously through interaction with its environment by attempting to maximize the total reward. As is the case with unsupervised learning, the learning process is not supervised.

Machine Learning in the Classroom

Recently, numerous tools have been introduced to allow learners to use machine learning in programming. A large number of these tools use block-based languages.

The hands-on toolkit PopBots focuses on Pre-K and Kindergarten children (Williams et al., 2019). The toolkit utilizes the same blocks as ScratchJr, providing teaching materials for knowledge-based systems, supervised learning, and generative music. Due to the target audience, the toolkit material does not explore concepts of ML in-depth, or the underlying algorithms.

Cognimates is a framework that assists with building games, programming robots, and training supervised learning models (Druga, 2018). The framework focuses on social robots. The learning process, however, is not visualized to the user, instead taking place on servers.

Machine Learning for Kids (Lane, 2018) is an online platform that supports supervised learning, allowing the Computer to recognize text, images, sets of numbers, or sounds within Scratch projects. Users provide training data, however, as before, the learning process is done on external servers.

Ecraft2learn is a framework for AI programming in Snap*!* (Kahn et al. 2018). It provides a library with several AI-related blocks. Using these blocks, which are merely an interface for the underlying API, the user can experiment with a broad range of AI concepts within the Snap*!*-environment. Internally, APIs or server-based services handle the actual computation involved.

AlpacaML (Zimmermann-Niefield et al., 2019) is an app that allows users to collect data, create and test models with supervised machine learning. AlpacaML is intended to be used in sports. It utilizes the sensor data of a wearable computing device. After data collection, the user labels the data, e.g. good or bad passes in soccer and applies the model to new data.

An analysis of these existing approaches shows that most of these educational tools and approaches focus on supervised learning as one of the three paradigms of machine learning algorithms. This is typically done due to the paradigm's widespread distribution and its accessibility with common examples, such as image recognition. However, this limits the perceived scope of ML to supervised learning, and therefore to a very narrow view on (machine) learning. Furthermore, many of these frameworks do not provide the opportunity to explore how the actual learning processes take place. The machine learning remains hidden to the user, e.g. with an API call or a JavaScript function. As a result, ML remains a *magical mystery*, limiting constructionist learning.

Design Principles for Teaching ML with Block-Based Languages

To help students explore the underlying principles of ML, we previously developed a collection of activities and an underlying framework called SnAIp for Reinforcement Learning (Jatzlau et al., 2019). These activities make up Part A of SnAIp. After building a foundation of knowledge through unplugged activities, we explore the concept of reinforcement learning through an algorithm named Q-learning in Snap*!*. In order to enable constructionist learning in which learners reach understanding of ML, we introduced the following underlying design principles:

Looking behind the scenes: Instead of only using pre-trained models, applying libraries or making API calls, users should be able to look behind the scenes and be able to improve or make adaptations to the ML algorithms.

Everything inside the tool: Other systems utilize extra services to train models, or rely on background API calls. This tends to reinforce the notion of block-based languages being "learning languages", and if the goal was instead to do "real programming", they are not sufficient. To combat this misconception, we emphasize that everything should work from within the framework.

Useful projects over learning projects: When introducing difficult topics such as ML, we often choose "learning projects", artificial problems and examples, as these support the visualization of certain concepts. In this framework, however, we want learners to be able to create their own, personally-meaningful projects according to the constructionist learning theory (Harel & Papert, 1991).

Models as artifacts: Artifacts or *objects-to-think with* are central in contructionist learning (Ackermann, 2004). With block-based environments that allow users to store and save variables locally, models can become these tangible artifacts. This enables several new and interesting activities in the classroom, i. e. a competition for students to compare their models, discuss, and learn from each other.

In this paper, we present the part B of the SnAIp framework, which deals with unsupervised learning and is developed according to the same design principles.

SnAIp: Unsupervised Learning in Snap*!*

Unsupervised Learning and Linear Vector Quantization

Unsupervised learning is a paradigm of machine learning heavily influenced by statistics. Often, the task is to group or categorize unlabeled data, e.g. segmenting customers into groups with similar purchasing habits. This can be done to tailor ads to specific subgroups. An illustration of the underlying principle can be found in fig. 1.

For SnAIp, we use linear vector quantization (LVQ), an algorithm that finds such clusters in data sets. *Learning* takes the form of a fixed number of prototypes, which move in the multidimensional feature space determined by the different attributes of the underlying data. For each data point, the closest prototype (according to a chosen distance measurement) is selected and its coordinates are updated.

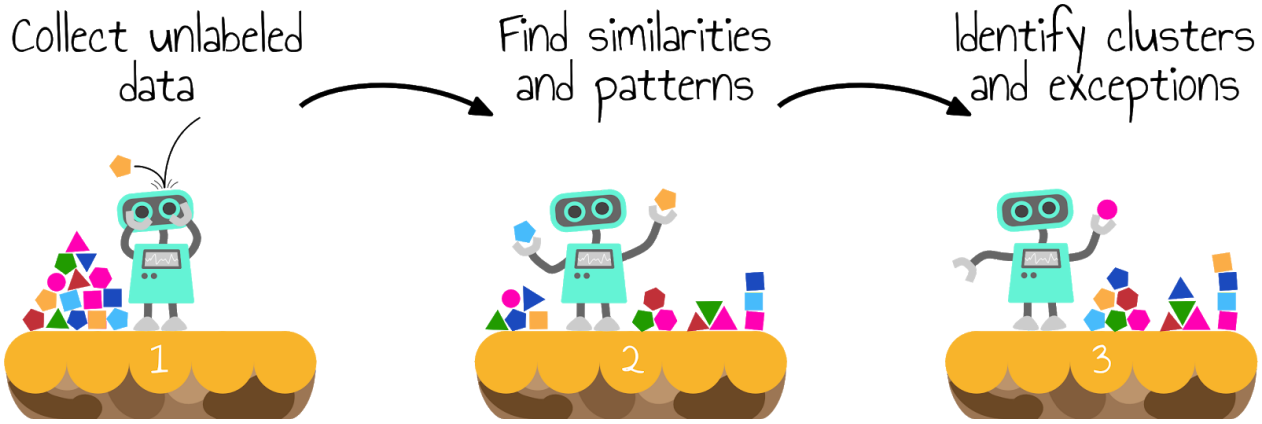

Figure 1. Unsupervised learning.

The SnAIp-framework

Using the visual programming language Snap*!*, SnAIp allows for the visualization of unlabeled data on a stage in two dimensions. Thus, students can visually explore the process of learning. Snap*!* as a tool and a programming environment provides a great deal of visualization and enables the implementation of complex projects. Compared to many similar programming environments and languages, it also supports efficient data processing. We chose Snap*!* for this framework in order to lower entry barriers to programming and to make ML graspable for students, following the ideals of the constructionist learning theory.

In the following section, we will illustrate how the design principles for SnAIp, which were laid out in the previous section, implement these goals for UL. With visualization being one of the primary factors, a number of blocks are required, which also aid the look behind the scenes.

This framework allows for experimenting with and applying unsupervised learning algorithms dealing with clustering in a two-dimensional space. To this end, we provide the following blocks that help with visualizing and processing two-dimensional data:

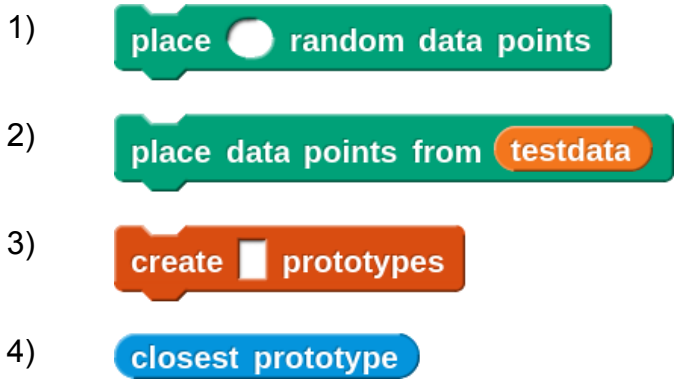

Figure 2. SnAIp framework for part B.

Initially, data points need to be created and, optionally, placed on the stage. For that, two blocks can be used (blocks 1 and 2 in figure 2). The first block creates a given number of **random** data points and clusters them on the stage. This is just a simple option to get started right away. For use cases that go beyond mere demonstration, the other block allows to use previously-imported data; After dragging-and-dropping data points from a **.csv file** into the Snap*!*-environment, this block normalizes out the **imported** data points (called “testdata” here) and spreads them out on the stage.

Both of the place-blocks are used for *visualization*. If data points do not need to be visualized on the stage, neither of the two blocks are necessary, as the algorithm relies exclusively on the prototype-blocks.

In the next step, prototypes need to be created. For this, block 3 in figure 2 is used. It creates a given number of prototypes and spreads them out randomly on a two-dimensional canvas (i. e. the stage in Snap*!* is used to visualize the data points, the prototypes, and the algorithm itself).

Finally, the closest prototype of each data point is determined and performs some type of action. The corresponding block (block 4 in figure 2) reports the closest prototype to a quantitative data point in an *n*-dimensional space and is used to determine which prototype needs to be moved.

SnAIp in the Classroom

In the following section, we will elaborate on our approach on how to introduce unsupervised learning in the classroom. The curriculum consists of three parts and is intended to be used in upper high school classrooms (ages 14 and older). Similar to part A, we start with an unplugged activity to introduce the concept of unsupervised learning in an active and motivating way. This is followed by an implementation in Snap*!*, in which the learners construct and optimize the unsupervised learning algorithm to solve problems from the ground up.

a) What is unsupervised learning

As unsupervised learning is mostly used for clustering, the introductory unplugged activity is about finding clusters in a set of two-dimensional data. In a wild west setting, the students' task is to determine the best possible digging site for three excavation teams. A good digging site is characterized as being as close as possible to many gold sources. After being introduced to the setting of the gold rush, students are given a map of the wild west (see fig. 3). They are also given 25 data points in the form of playing cards, consisting of *x* and *y* coordinates of a gold source. Furthermore, they are provided 3 coins that serve as *prototypes*, visualizing the excavation teams. The students continuously draw data point cards, consider how to process the information they contain, and put them face-down on the discard pile. Therefore, they only see one datapoint at a time, the same way a computer would, and never "see all the data at once". As they have no way of marking the different data points, students have to use and move the coins in reaction to each new datapoint. They will come up with an algorithm conceptually similar to the following LVQ-variation:

1. Place the prototypes (coins) on the map (randomly).
2. For each datapoint:
    a. Identify the prototype closest to the data point
    b. Update the prototype's position by moving it half the distance toward the coordinates of the data point

Figure 3. Unplugged activity used for the introduction of unsupervised learning.

After processing all the data, students are provided with a projector slide containing all the data points, which they can lay over their map to evaluate their final results. With the help of this slide, they can then improve their algorithm. Based on the experience the students have made in this activity, principles and characteristics (see figure 1) of unsupervised learning can be discussed and compared to other machine learning paradigms.

b) Unsupervised learning in Snap!

In the next step, the previously-developed LVQ-algorithm is implemented in Snap*!*. Students receive a template that provides prototypes, and a dynamic number of randomly-created data points, both of which are already visualized on the stage. The task is to complete the actual LVQ-algorithm (see fig. 4). In doing so, students can experiment with the various parameters, such as number of prototypes, data points, etc.

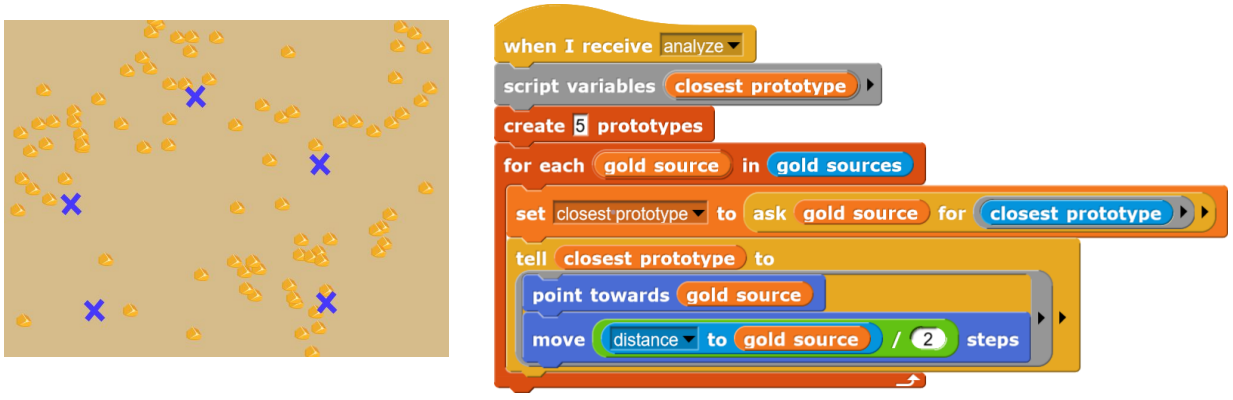

Figure 4. Gold rush clustering with unsupervised learning in Snap!

Rather than being limited to working with random data, the framework can also be applied to real-world data (which could even be gathered beforehand by/with the students), in the sense of constructionist learning theory. The framework enables the processing and clustering of real-world data by working with csv imports. One possible application of this might be to cluster residential locations to determine ideal sites for bus stops (the two dimensions in this case being longitude and latitude). In general, this algorithm can be applied to any data set. Categorical data, however, must be transformed into numerical inputs before.

c) Competition

Afterward, students are asked with optimizing the algorithm in a competition. As the algorithm presented here is only a very simplified approach, there is room for improvement and optimization. There are multiple possible approaches toward this, such as:

- Using multiple iterations to refine the final prototype locations
- Decreasing the movement range for each prototype gradually
- Increasing the priority of prototypes that have not been moved
- Optimizing the initial position of the prototypes

Similar optimisations are typically applied to unsupervised learning algorithms as well. By providing data sets that can be imported via Snap*!* and visualized with the frameworks' respective block, students' solutions can be compared and discussed in the classroom.

Discussion and Conclusion

An increased interest in machine learning creates the need for concepts, curricula and teaching material that make it possible to incorporate the different paradigms of ML into the classroom. Our approach incorporates both the creation of ML artifacts and supporting their understanding. Existing approaches fall short on providing a way to create a fundamental understanding of ML in order to construct their own knowledge, or focus on a single paradigm, supervised learning. Although many real-world applications of ML refer to supervised learning, there is a growing number of applications of unsupervised and reinforcement learning. Thus, students will interact with or experience these paradigms in apps or services they use.

The framework we outlined in this paper utilizes the potential provided by UL and fulfills our four design principles:

1. **Looking behind the scenes**: Learners are tasked with implementing the entire learning algorithm directly in Snap! itself with very little guidance. Constructing the algorithm from the ground up enables a unique look behind the scenes for learners.

2. **Everything inside the tool**: All the algorithms are implemented directly in Snap!, making it easy for learners to make adaptations and optimizations to the algorithm.

3. **Useful projects over learning projects**: According to the constructionist learning theory, learners are able to create their own, personally-meaningful projects by using and adapting the algorithms to fit their own projects. As an example, when working on an image compression project, users can identify patterns in their own csv-data and apply the concepts they learned.

4. **Models as artifacts**: With the ability to export data structures as files, learners create their own artifacts and use them for various activities, such as the competition outlined above.

In summary, in this paper we present how to look beyond supervised classification and image recognition with students in secondary education. To this end, we introduce a framework to explore machine learning in a constructionist and transparent way. Furthermore, we outline a corresponding curriculum to address unsupervised learning in the classroom. This enables learners to create and understand ML artifacts.

Acknowledgements

We want to thank Urs Lautebach for contributing the idea for the chosen unsupervised learning algorithm and unplugged activity and the Königstein group for developing the idea further.

References

Ackermann, E. (2004) Constructing Knowledge And Transforming The World, A learning zone of one's own: Sharing representations and flow in collaborative learning environments [M. Tokoro and L.Steels (Eds.). Amsterdam, Berlin, Oxford, Tokyo, Washington, DC. IOS Press. 1, 2 (pp. 15–37).

Ayodele, T. O. (2010). Types of machine learning algorithms. In: Y. Zhang. New advances in machine learning. IntechOpen.

CSTA (2017). About the CSTA K-12 computer science standards, https://www.csteachers.org/page/standards, last accessed 02/06/2020.

Druga, S. (2018). Growing up with AI: Cognimates: from coding to teaching machines (PhD thesis), Massachusetts Institute of Technology, Cambridge, USA.

Heinze, C., Haase, J. & Higgins, H. (2010). An action research report from a multi-year approach to teaching artificial intelligence at the k-6 level. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10). Atlanta, GA USA: AAAI Press (pp. 1890–1895).

Jatzlau, S., Michaeli, T., Seegerer, S., & Romeike, R. (2019). It's not Magic After All-Machine Learning in Snap! using Reinforcement Learning. In 2019 IEEE Blocks and Beyond Workshop (B&B). Memphis, USA TN: IEEE (pp. 37–41).

Kahn, K. Megasari, R., Piantari, E. & Junaeti, E. (2018). AI Programming by Children using Snap! Block Programming in a Developing Country. In EC-TEL Practitioner Proceedings 2018: 13th European Conference On Technology Enhanced Learning.

Lane, D. (2018). Explaining artificial intelligence. Hello World, 4, https://helloworld.raspberrypi.org/issues/4, last accessed 02/06/2020.

Marsland, S. (2015). Machine learning: an algorithmic perspective. Boca Raton, USA FL : CRC Press Taylor & Francis Group.

Mitchell, T. (1997) Machine Learning, New York, NY USA: McGraw-Hill Education Ltd.

Papert, S., & Harel, I. (1991). Situating constructionism. In: I. Harel, & S. Papert. Constructionism (pp. 1–13). Norwood, NJ USA: Ablex Pub. Corp.

Touretzky, D., Gardner-McCune, C., Martin, F. & Seehorn, D. (2019). Envisioning AI for K-12: What should every child know about AI? In "Blue sky talk" at the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19).

Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8(3-4). Boston, MA USA: Kluwer Academic Publishers (pp. 279–292).

Williams, R., Park, H. W., Oh, L., & Breazeal, C. (2019). Popbots: Designing an artificial intelligence curriculum for early childhood education. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 9729-9736).

Yu, Y., & Chen, Y. (2018). Design and development of high school artificial intelligence textbook based on computational thinking. Open Access Library Journal, 5(09):1.

Zimmermann-Niefield, A., Turner, M., Murphy, B., Kane, S. K., & Shapiro, R. B. (2019). Youth Learning Machine Learning through Building Models of Athletic Moves. In Proceedings of the 18th ACM International Conference on Interaction Design and Children, ACM (pp. 121-132).