Milo: A visual programming environment for Data Science Education

Arjun Rao*, Ayush Bihani†, Mydhili Nair‡
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India
Email: *mailarjunrao@gmail.com, †bihani37@gmail.com, ‡mydhili.nair@msrit.edu,

***Abstract*—Most courses on Data Science offered at universities or online require students to have familiarity with at least one programming language. In this paper, we present, "Milo", a web-based visual programming environment for Data Science Education, designed as a pedagogical tool that can be used by students without prior-programming experience. To that end, Milo uses interactive graphical blocks as abstractions of language specific implementations of Data Science and Machine Learning(ML) concepts along with creation of interactive visualizations. Using block definitions created by a user, Milo generates equivalent source code in JavaScript to run entirely in the browser. Based on a preliminary user study with a focus group of undergraduate computer science students, Milo succeeds as an effective tool for novice learners in the field of Data Science.**

I. Introduction

Over the last four years, we have seen a lot of growth in the use of Data Science in modern applications. According to LinkedIn's 2017 report [1], the top ranked emerging jobs in the U.S. are for Machine Learning Engineers, Data Scientists, and Big Data Engineers. The report also highlights that while the number of roles in the Data Science domain has risen many fold since 2012, the supply of candidates for these positions is not meeting the demand.

The most common path taken towards understanding Data Science is still through university programs, online courses and workplace training. We surveyed popular online courses in the domain using Class Central [2] and found that most courses either require prior programming experience in Python or use tools like MATLAB, Octave, R, Weka, Apache Spark, etc. which can be intimidating to non-computer science majors.

In the general field of computer science education, there have been many efforts for introducing fundamental concepts of programming to beginners through visual tools. Examples include those on Code.org or MIT's Scratch project [3]. However there have been fewer efforts in building tools for introducing concepts in Data Science and Machine Learning to non-programmers.

In this paper, we present "Milo", a web based visual programming environment for Data Science Education. Our primary aim when designing Milo, was to build a platform that is approachable to non-computer science majors, and allows students to self-learn concepts of Data Science and Machine Learning. To support these goals, we built Milo to work in the browser, and use a block-based programming paradigm that is suitable for novices and non-programmers. The main interface of Milo is shown in Figure 1, and consists of graphical blocks which abstract implementations of various concepts covered in typical Data Science courses. Supported concepts include basic statistics, linear algebra, probability distributions, ML algorithms, and more. The workspace is built using Blockly [1] and the blocks have a similar look and feel to that of Scratch [3].

Our target audience for Milo, is two fold. On one hand we target students from high-school to undergraduate students in non-computer science fields. For students who are not familiar with programming but have an understanding of basic concepts in linear algebra, and statistics, we feel that Milo is a good avenue for getting hands on exposure to using these concepts in solving practical problems, and getting exposure to the world of programming in an intuitive and visually rich manner. On the other hand we target faculty and educators, who design introductory courses for non-programmers in the fields of Data Science, Machine Learning and Linear Algebra.

The rest of this paper is organized as follows. Section II highlights related work in this domain, particularly visual programming environments and tools that we referred to. In Section III, we talk about Milo's programming model and compare this with other popular approaches. This is followed by details of our implementation in Section IV. Section V summarizes a preliminary evaluation of Milo via a user study. We then note a few limitations of our work in Section VI, and present a road map for the future (Section VII) and our conclusions (Section VIII).

II. Related Work

Visual programming environments are alternatives to text based programming, having logic constructs expressed using graphical operators and elements. This is not a new concept, as prior work on such forms of programming date back to the 90s. Work done by [4], [5], and [6], have influenced many projects from the 90s to present times, showing that visual programming environments are commonly employed in practice.

Scratch [3] is a visual programming language and online community targeted primarily at novice users, and is used

[1] https://developers.google.com/blockly/

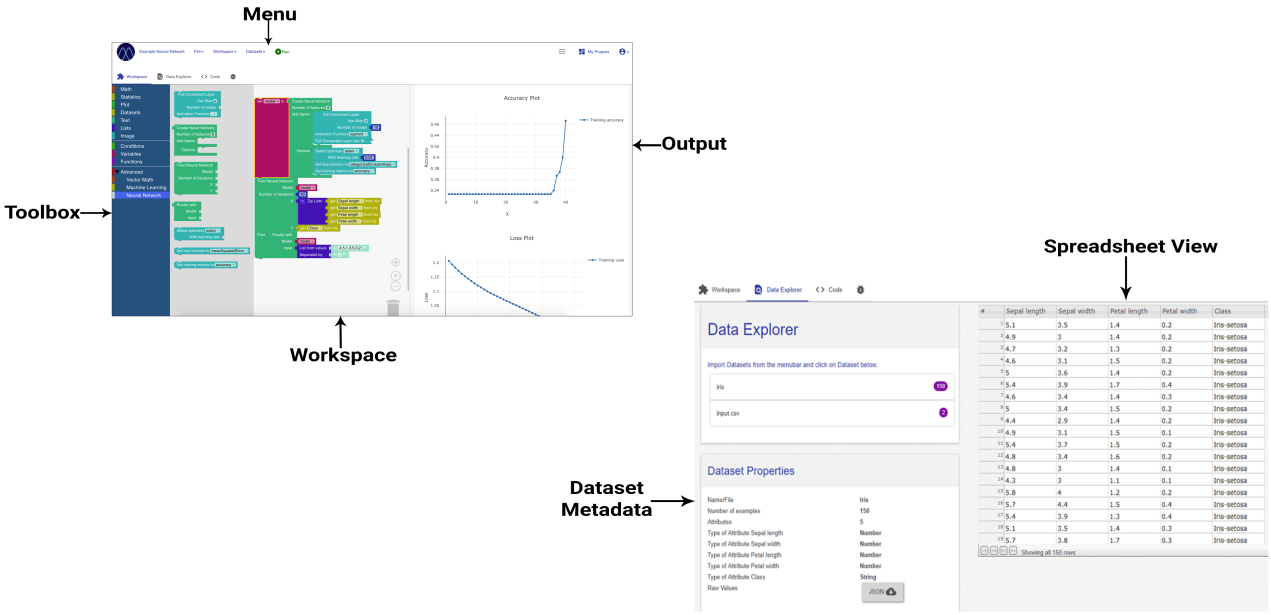


Fig. 1. The top left screenshot shows the Milo IDE Interface consists of the top menu, a toolbox that holds blocks organized by category, the workspace for building block based programs, and the output pane. The bottom right screenshot shows the data explorer with the Iris dataset loaded in a spreadsheet like view.

as an introductory language to delve into the field of programming through blocks. Our main motivation for choosing a block-based design for Milo was Scratch, due to it's proven track record for being a popular introductory tool for non-programmers to get involved with computer science. The user interface of Scratch follows a single-window, multi-pane design to ensure that key components are always visible. The success of this approach motivated us to build a single window IDE in Milo, that allows execution of programs along side the workspace used to create them. This prevents distraction for users and presents all the important aspects of the IDE in one place.

According to Zhang et al. [6], a single environment for researchers to manage data and perform analysis, without having to learn about multiple software systems and their integration, is highly effective. Thus Milo borrows these ideas and includes a Data Explorer for viewing and understanding datasets in a spreadsheet like format, along with the main block workspace that is used to perform operations on data or train ML models. (See Fig 1)

BlockPy [7] is a web based python environment built using Blockly with a focus on introductory programming and data science. This is primarily done by integrating a host of real-world datasets and block based operations to create simple plots of data. However we found that BlockPy is primarily suited to give a gentle introduction to Python and falls short of the requirements of a full-fledged Data Science course.

Another popular tool used for teaching Data Science is Jupyter Notebooks[2]. While it is a great tool for exploratory data analysis and quick prototyping, we found that Jupyter notebooks are more suited for Computer Science majors, and those who are familiar with concepts of programming in general, and more specifically those of Python or Julia.

III. Comparisons between programming models

Milo uses a block based programming model. This approach to programming is unlike that of popular visual tools for Machine Learning like Rapid Miner[3] or Orange[4], as they follow a dataflow approach to programming. There are significant differences between Milo and Scratch in terms of design and computational aspects. We shall highlight these points of distinction in this section.

When compared with Scratch, Milo's programming model may seem very similar in terms of look and feel. This is because they are both rendered using Blockly, however, the styles, blocks, and their connections are designed for different use cases, and hence the language vocabulary and block patterns are different. Unlike Scratch, Milo does not use an event driven model. This is because, we do not have sprites, or a stage with graphical objects that interact with one another. Instead, Milo uses a sequential approach to programming, where blocks are executed from top to bottom in the workspace, ie. blocks placed above others will be executed first. Additionally, Milo generates syntactically correct source code from the block definitions, and this is presented in the code tab of the UI. During our prototyping stages, we found that using such a model makes the transition to real

[2]https://jupyter.org

[3]https://rapidminer.com/

[4]https://orange.biolab.si/

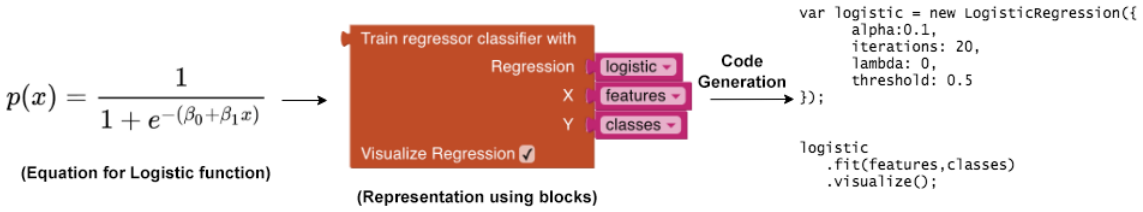


Fig. 2. Shows how a machine learning concept such as a classifier using logistic regression is represented using blocks on Milo, along with the code it generates.

programming languages after Milo, fairly intuitive. The block constructs and their respective translations in JavaScript or Python are easily comparable and the sequential flow of execution is preserved after the translation.

When it comes to dataflow paradigms, we feel that while such paradigms are intuitive in understanding the transformations from input to output, it is less useful in understanding the internal steps of this transformation. It makes Machine Learning algorithms seem like black boxes to novice students, obscuring implementation details. In Milo, students can drag a single block, such as the one shown in Fig. 2, that trains a logistic regression based classifier using the given input, and produces an animated visualization of the training steps, which is similar to the black box like approach of dataflow programming. However they can also go a step further and build this block themselves by using primitive blocks for manipulating input vectors, and math operators like exponential functions or logarithms. Thus novice students would first learn concepts using built-in high-level blocks, and then figure out how to build these algorithms themselves using primitive blocks that they assemble from scratch.

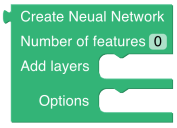
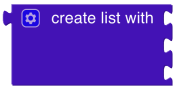
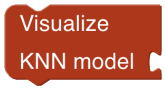
IV. Implementation

We implemented Milo's block language using Google's Blockly library, which is used to build visual programming editors. The main user interface of Milo, as shown in Figure 1, consists of the workspace where block based programs are assembled, the output pane, a menu bar that lets users switch between the workspace, the data explorer, which is a space for viewing datasets in a spreadsheet like format, and a tab that shows generated code. The tool also includes a few popular datasets that are used in introductory ML courses.

In Milo, all programming constructs and implementations of various Data Science concepts are represented as interconnecting drag and drop blocks. They are the basic primitives for building any program on the platform. Connections between blocks are constrained such that incompatible blocks cannot be connected together. This allows us to generate syntactically correct code from block representations and prevent logical errors. Figure 2 is an example of how a machine learning concept like Logistic regression is represented through blocks and translated to source code.

The blocks result in generation of syntactically correct Javascript code, which is used for execution on the web.

TABLE I
Types of Blocks in Milo

	**Chained Input Blocks** have space for chaining a number of supplementary input blocks (such as the Add Layers input in the block on the left), and are used to create dynamically defined objects. Examples include creating different neural network architectures by chaining neural network layer definitions one below the other or for creating multiple plots by chaining plot definitions.
	**Compute blocks** are those that have a notch on the left that represents a return value connection. These Blocks optionally take inputs and always return a value that is the result of some computation. The blocks may have additional options for advanced usage that is indicated by presence of a gear icon, on the block.
	**Operation blocks** are those that represent a single operation/function that does not return any value. These blocks can take input and additionally may transform their input but they do not return any value. The notches above and below the block are used to chain operations to execute in a particular order or to act as inputs to **Chained Input Blocks**

We used tensorflow.js [5] for implementing low-level math operations like matrix multiplications, vector manipulation, etc. Table I illustrates the various types of blocks available in Milo.

The Milo Platform consists of a NodeJS[6] based web server, which acts as the backend, and the frontend for the platform is written in AngularJS[7]. The projects created using Milo, are stored as XML documents in a MongoDB database [8]. As part of the client side code, we include an execution library, which exposes high-level APIs for implementations of various Machine Learning algorithms, similar to what scikit-learn[9] does in Python. The pre-made blocks for algorithms like KMeans, KNN, etc are translated to calls to functions in this execution library, and these in-turn are implemented

[5]https://js.tensorflow.org
[6]https://nodejs.org
[7]https://angularjs.org
[8]https://www.mongodb.com
[9]http://scikit-learn.org

using tensorflow.js and custom javascript code. These pre-made ML algorithm blocks also come with corresponding blocks to generate visualizations that show models being updated with each iteration of training, such as animated decision boundaries, or clustering data points in real time with iterations of training. Additionally, for some algorithms, like KNN, there are interactive visualization blocks that allow users to place a new test point on the 2d plane showing a scatter plot of the training dataset and see in real time what class the model might assign to this point, and what neighbours were considered. These interactive visualizations and other plotting functions in Milo are implemented using D3.js[10].

V. Preliminary Evaluation

In order to evaluate our implementation, we conducted a user study, with a focus group of 20 undergraduate computer science students.

A. Study Setup

Participants were selected using a convenience sampling approach from a class of students who were taking their first introductory course in Machine Learning. The class follows the book Introduction to Machine Learning by Ethem Alpaydin [8]. Prior to the study we asked participants, to report their familiarity with various ML concepts, and their programming experience. We found that only 10% of participants reported that they would consider themselves more-comfortable with programming, while 55% considered themselves less-comfortable, with 25% of participants reporting that they had never programmed in Python/R or Julia before. In order to evaluate the usefulness of Milo in a course, we asked participants of the study to take a model class on Machine Learning that uses Milo as part of the pedagogy via a flipped classroom approach [9]. During the class participants used Milo, to perform clustering using K-Means on the Iris dataset [10].

After the class we administered a post-study questionnaire. The questionnaire asked students to rate various features of Milo that they tried, in terms of usefulness and ease of use, along with their perceived level of understanding K-Means clustering after the flipped-classroom activity. They were also asked open-ended questions that prompted feedback about various activities done as part of the class.

B. Study Results

- As the participants were taking a course on machine learning which followed a traditional classroom model, their experience with a flipped classroom model using Milo lead them to have highly positive sentiments.
- 90% of participants reported that visualizations were very easy to create using Milo and supplemented their understanding of the concepts.
- The study was mainly preliminary in nature, to evaluate the tool, in terms of usefulness and whether or not it met the requirements of students learning Machine Learning concepts for the first time, and based on the survey responses, we found that 70% of students felt the tool would be very useful for novice learners.

[10]https://d3js.org/

VI. Limitations

Due to the preliminary nature of our user study, our focus group size was limited. The students in the study had some level of prior-programming experience as they had taken at least one formal programming course. As a preliminary study, we had computer science students as our participants, as they would be in a position to evaluate the merits of the tool in terms of what works, and what is missing. However a real test for the tool will only be when we conduct a user study with non-computer science students. As the main focus is education, Milo is not intended to be used for training and developing production ML models. The platform does not support advanced neural networks such as LSTM, GRU or using convolutional layers etc. or large datasets that are typical for Deep Learning problems.

VII. Future Work

Our goal with Milo is to help learners understand complex concepts using a simple visual approach. Concepts such as neural networks, and multivariate distributions need to be explained in an intuitive way to new learners. Keeping this in mind, the next iteration of Milo will include interactive visualization for neural networks, support for multinomial distributions, multivariate gaussians, etc. to make these concepts more approachable to beginners. To improve code reusability, the platform will let users download generated code and results which can then be embedded in external blogs or other websites. While our current security model prevents, to a large extent, execution of code that may be harmful or malicious in nature, we are working on enhancing security by including a more robust execution sandbox for code that runs in the browser.

VIII. Conclusion

In this paper, we present Milo, a novel visual language targeting new learners in the field of Data Science and Machine Learning. Through our preliminary user study, we show that a visual programming environment is an effective platform for introductory courses on Data Science. Additionally, we establish a direction for future work in improving Milo, to allow effective transitions from blocks to real world programming through our proposed work on code generation using Julia.

Functional Prototype

To facilitate research and further evaluation of our work, we have released our code on GitHub under an open source license, and can be found at https://miloide.github.io/.

REFERENCES

[1] L. E. G. Team, "Linkedin's 2017 u.s. emerging jobs report," December 2017.

[2] Class central: A popular online course aggregator. [Online]. Available: https://www.class-central.com/

[3] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment." *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–15, Nov 2011.

[4] A. A. Disessa and H. Abelson, "Boxer: A reconstructible computational medium." *Communications of the ACM*, vol. 29, no. 9, pp. 859–868, Sept 1986.

[5] G. EP., "Visual programming environments: Paradigms and systems," 1990.

[6] Y. Zhang, M. Ward, N. Hachem, and M. Gennert, "A visual programming environment for supporting scientific data analysis." *Proceedings 1993 IEEE Symposium on Visual Languages.*, 1993.

[7] A. C. Bart, J. Tibau, E. Tilevich, C. A. Shaffer, and D. Kafura, "Blockpy: An open access data-science environment for introductory programmers," *Computer*, vol. 50, no. 5, pp. 18–26, May 2017. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MC.2017.132

[8] E. Alpaydin, *Introduction to machine learning*. The MIT Press, 2010.

[9] M. B. Gilboy, S. Heinerichs, and G. Pazzaglia, "Enhancing student engagement using the flipped classroom," *Journal of nutrition education and behavior*, vol. 47, no. 1, pp. 109–114, 2015.

[10] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.