Using Visualizations to Support Machine Learning Education for Young People Without Programming Experience

Abigail Zimmermann-Niefield

Abstract

Machine learning (ML) is increasingly everywhere, and as it spreads, the more critical it becomes for people to understand such systems and incorporate them into their own design practices. Many existing ML classes target extremely advanced programmers or statisticians. However, a person does not need to know how to code or perform a regression in order to be able to interpret ML systems, or apply them effectively. My previous work has shown it is possible for novice users to design, quickly build and iterate on, and hypothesize about ML classifier models of their own body movement. However, the only feedback they were given was the real-time, spoken classification. This process could be much easier for users if they were provided with a visualization describing how the ML algorithm they used reached its decision. In this paper, I will present a rationale for using visualization, a design of a visualization, a brief evaluation, and potential future research directions.

Introduction

Machine learning (ML) is ubiquitous and powerful. As such, it is critical for people to learn to understand ML systems and incorporate them into their own design practices. Machine learning is different from and complementary to programming—unlike traditional Computer Science, ML involves thinking like a scientist rather than a mathematician or logician. ML practices involve data collection, data cleaning, choosing a model, and statistical testing (Shapiro, Fiebrink & Norvig, 2018). Most ML curricula are only available to advanced undergraduate or graduate students in Computer Science, but I believe people do not need to know how to code to learn how to build ML models, and begin to develop understandings of the increasingly ML-rich world.

My work focuses on developing and evaluating tools and curricula to investigate how youth with minimal programming experience can learn to apply Machine Learning in creative and meaningful contexts. My initial investigations allowed me to understand how novices could leverage their existing knowledge in athletic domains to build machine learning classifier models--with audio only feedback (Zimmermann-Niefield et al. 2019). For my final project, I extended the tool I built to include visualizations as feedback on every test movement. I exacted feedback from a range of people whose knowledge of my previous work ranged from somewhat familiar to expert.

Related Work

Learning Machine Learning

Shapiro, Fiebrink and Norvig argue that as machine learning-based interactive technologies proliferate, computing curricula must shift towards ML and research on how to do so is dearly needed (Shapiro, Fiebrink, & Norvig, 2018). Some work has described enhancements to computer science education platforms like Scratch, MIT App Inventor, and Google's Teachable Machine

that provide existing ML models people can explore, train and play with. But there has been little empirical investigation, other than my own previous work in the area.

Other previous work in machine learning education for novices has used *Interactive Machine Learning*—a creative cycle for machine learning model building that is especially suited to the development of personalized interactive technologies (Fiebrink, 2019)(Fiebrink, Cook, & Trueman, 2011). When engaging in IML, a creator iteratively engages in the following steps:

> *formulation* — identify the desired behavior,
> *synthesis* — implement a system that aims to achieve the behavior,
> *analysis* — observe actual behavior of the system,
> *evaluation* — compare observed behavior with expected behavior, and
> *reformulation* — reason about and enact ways to improve performance.

I have drawn on this framework in my previous design of tools to support novices learning to leverage ML. The visualization I have designed is intended to aid and support the *evaluation* and *reformulation* steps.

AlpacaML

In order to study young people learning machine learning, I developed AlpacaML, an iOS application that allows youth build, use, evaluate, and iteratively improve machine learning models of their own body movements (Zimmermann-Niefield et al. 2019). Use of AlpacaML involves the IML spiral described above. This reflects how novices engage in ML- repeatedly *collecting*, *labeling*, *testing*, *evaluating*, and *editing* data. To build a model, the user reviews the accelerometer data paired with video, selects their motions, and labels them to form a training set. Then, they put AlpacaML in model execution mode. AlpacaML runs a Dynamic Time Warping algorithm which produces a non-Euclidean distance between the incoming data and each training segment, and speaks the classification of the "nearest" segment. I've included two video demos at this link:
(https://drive.google.com/open?id=1yujMA8Pbd8OpAqRKIVzLCopYRs5XXLxJ,
https://drive.google.com/file/d/1BU18ipXvyqdlJAlfKQpCaPH8F7Xgm5XV/view?usp=sharing).

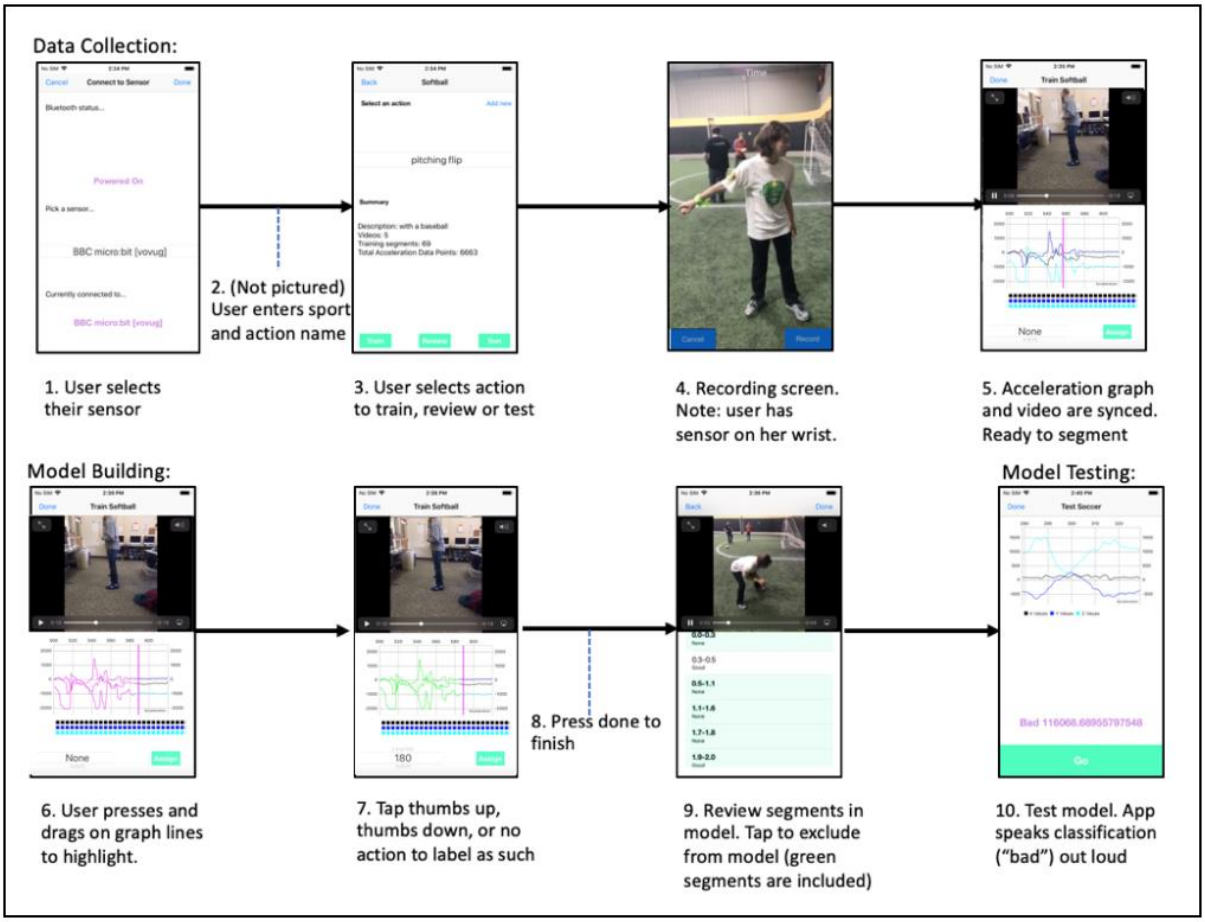


Fig 1. The previous UI of AlpacaML before my Final Project modifications

I previously conducted a pilot study and semi-structured focus group with five participants. I found that they were able to leverage their prior athletic knowledge to collect data, build models, test, and evaluate models using AlpacaML. They were also able to conduct rapid iterations to test their hypotheses about how well their models worked, and make adjustments to their models to make them better (Zimmermann-Niefield et al. 2019). The work I present here continues both these veins of inquiry- what do users *do* when they're building models, and what do they *understand* when they are building models. This time, instead of just providing audio classification, AlpacaML provides a visualization of the closeness of every incoming classified movement to each training example. The resulting bar chart is meant to aid in the development of hypotheses and the design of iterations.

Visualizing ML

Visualization is frequently used with deep learning and neural nets because their internal processes are opaque and they are frequently applied to important problems (Hohman, F., 2019). Researchers and storytellers have also used visualizations describe more intuitive artificially intelligent algorithms like decision trees (Yee & Chu). Notably, Giorgino used an R package to visualize DTW, the algorithm AlpacaML uses. However, an R user likely has more advanced knowledge of both programming and statistics than the users I target with AlpacaML, who have none. The

visualizations presented by Giorgino require a great deal of subject matter knowledge. Mine will use much simpler bar charts. In general, my purposes are slightly different than the ones listed above. I intend to use visualization as a prompt to learners who are in the process of *evaluating* and *iterating* on models.

Visualizations in Education

Visualizations as educational tools have been studied at the elementary school level (Alper, 2017). Alper et al. draw from work in visualization literacy and pedagogical strategies from the learning sciences and educational psychology. They conducted a qualitative review of visualizations found in textbooks, and surveys with 16 teachers. Using their findings, they prototyped visualizations and performed a field study in a classroom. They found that visualizations can be used to support the acquisition of abstract concepts, and that using them in classrooms can provide increased visualization literacy. My approach follows the first finding outlined by Alper et al. Using DTW as a classifier on 3D acceleration is an extremely complicated, multi-tiered abstract concept. My visualization uses a simple visualization to help ground this concept for learners.

Results and Discussion

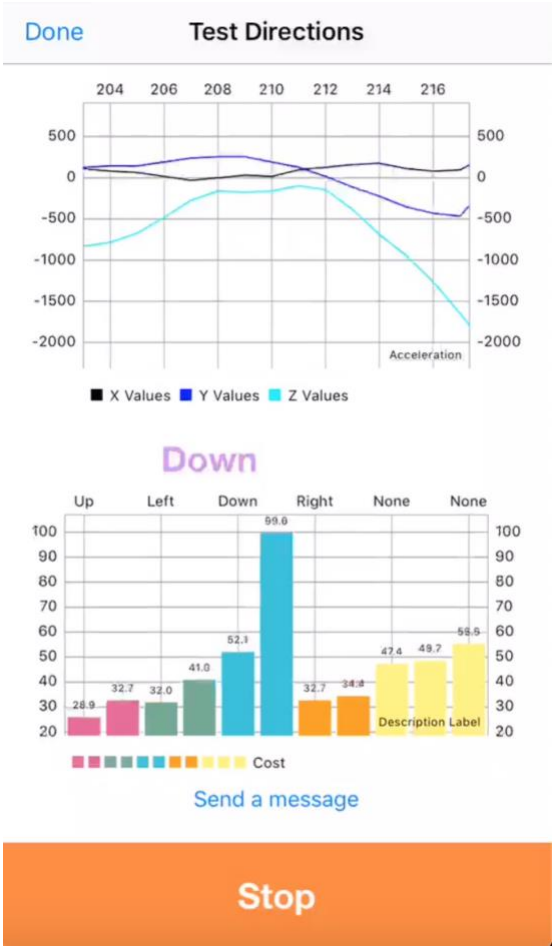
Design

I added visualization-based feedback to AlpacaML that allows the user to accomplish the following tasks:

- **High level:** to guide the user of AlpacaML to better evaluation and analysis of a model during execution.
- **Middle level:** to support the user in reading and interpreting the results of each test motion they perform during model execution.
- **Low level:** to show the user how good of a match the motion they just executed was with each of the example motions.

I created this visualization to display to the user during AlpacaML's real time model execution. Whenever AlpacaML makes a non-trivial classification, it displays a bar chart to the user that shows how close each the movement identified was to each training set. I made the following design decisions:

- **I chose to group examples with the same class by color,** so it would be easy for the user to tell which examples belonged to which class. I chose an existing palate from the iOS charts module, because I liked the way it looked, and my labmates also liked the way it looked when I showed them.
- **I chose to invert the cost function when displaying the graph.** As I mentioned above, DTW relies on a cost function- a warped distance between two timeseries. The algorithm compares an incoming example with the existing training examples, and selects the class of the example with the *lowest* warped distance. I originally used raw cost in the bar chart, but it was so confusing it even confused me (discussed below).



In the image to the left, the user (not pictured), has just moved the micro:bit down. In the top chart, we see the 3D acceleration graphed. This is not part of my new visualization. In the middle of the screen, we see the word "Down," which is also the word AlpacaML would speak out loud. At the bottom, we see a bar chart that pictures the inverse cost from the incoming "Down" movement to each example. It was closest to an example under the class "Down," so it was classified as "Down."

Cost

I originally pictured raw cost on the bar chart rather than inverse cost. For example, in the case pictured left, "Down" would have had the lowest bar. This was so confusing, it made me think I had built an incorrect model at one point during testing. Additionally, I showed the visualization to two intermediate participants at this step. One of them said,

"Wait, so is the bar graph at the bottom representing what label the app is identifying the gesture as?"

I responded, *"yes,"* and mentioned that I planned to invert cost because it was confusing me.

"Yeah, I totally interpreted that as the opposite way," she said.

The best way I knew to invert cost was to divide some constant by it. The original cost numbers tended to be on the order of 1,000 (but not more than 5,000). Although these numbers are *technically* quantitative data, they are meaningless to a human other than as an ordinal measure of closeness. I decided to target numbers that seemed reasonably sized and easily interpretable (i.e. not below 10, and *certainly* not below 1, and not so large they would crowd the screen or overlap on the bars). I ended up choosing to use 10,000 as my constant.

A Design Experiment

I conducted a brief design experimentd with six people who ranged from somewhat familiar to very familiar with AlpacaML. All participants said they liked the color scheme. All participants said they liked AlpacaML better with the visualization. One participant said:

" I love that it gives more insight into how the device is determining which gesture/movement."

I also found this. As I was building models to test my visualization, I found that I understood the models better. For example, in a model with "right," "left," and "none," "right" tended to be closer

to “none” than it did to “left,” except in the case where my gesture had a lot of rebound in the opposite direction.

I asked participants what they thought the bar chart represented. The three participants who commented on this all interpreted the bar chart as a measure of goodness, i.e. the higher the bar, the better the match:

“So, bottom is approximate posterior probability? I say approximate, because I guess it’s more like inverse warped distance? Closeness?”

“In that example, is each bar one training example? So two for everything that's labeled and three that are labeled as none?”

“Is it a measure of confidence?”

Note that all three participants who answered phrased their answers as questions. The visualization was not obvious enough that they could confidently state what it represented. It is possible I could produce a more clear visualization. Perhaps not- perhaps Dynamic Time Warping distance *is* confusing, and learning how to interpret such visualizations should be a part of learning how to build models with ML. This would follow what Alper et al. suggested- visualization is not only a tool for representing abstract concepts. Abstract concepts could promote kids learning about visualization.

Conclusion and Future Work

My prototype visualization of AlpacaML showed promise. My participants said they liked it, and one in particular mentioned they thought it gave more insight into the internal processes of DTW. I found that I learned as I was using it as well, even though I was already very familiar with the process having built the app. My next step is to eliminate small bugs I found in the visualization and prepare it for use with real participants. I plan to run a small-scale design study, similar to the work described in Zimmermann-Niefield et al. (2019). I will investigate what additional practices and theories participants develop when they use AlpacaML with visualizations, and how they differ from those they develop when they use AlpacaML without visualizations.

I also plan to implement non-real time testing, so the user can review video data of both the testing and training segments along with the bar charts. I will also allow the user to tell AlpacaML the correct classifications after testing. Then, AlpacaML will visualize the amount of correct and incorrect classifications that occurred during testing. There are a wide variety of visualizations that AlpacaML’s data could easily support. They will likely give learners a great deal more insight into modeling and machine learning than AlpacaML currently gives. It will be fascinating to study how visual feedback can guide users in their learning.

References

Alper, B., Riche, N. H., Chevalier, Boy, F. J. and Sezgin, M. Visualization literacy at elementary school. In Proc. CHI ’17, pp. 5485–5497. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025877

Fiebrink, R. *Machine learning education for artists, musicians, and other creative practitioners.* ACM Transactions on Computing Education (2019).

Fiebrink, R., Cook, P. R., and Trueman, D. *Human model evaluation in interactive supervised learning.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, ACM (New York, NY, USA, 2011), 147–156.

Giorgino, T. *Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package.* In the Journal of Statistical Software, 31(7). (2009), 1-24.

Hohman, F. *Visualization in Deep Learning.* (2019, March 1). Retrieved from: https://medium.com/multiple-views-visualization-research-explained/visualization-in-deep-learning-b29f0ec4f136

Shapiro, R. B., Fiebrink, R., and Norvig, P. *How machine learning impacts the undergraduate computing curriculum.* Communications of the ACM 61, 11 (November 2018).

Yee, S. J., Chu, T. S. *A Visual Introduction to Machine Learning.* Retrieved from: http://www.r2d3.us/visual-intro-to-machine-learning-part-1/.

Zimmermann-Niefield, A., Turner, M., Murphy, B., Kane, S. K, and Shapiro, R. B. *Youth learning machine earning through building models of athletic moves.* In Proceedings of the 18th ACM International Conference on Interaction Design and Children (Boise, ID, USA, 2019). 121–132