Machine Learning for Introductory Physical Computing Curricula

Ali Momeni
Carnegie Mellon University
CFA 300 - 5000 Forbes Ave.
Pittsburgh, PA 15213, USA
momeni@cmu.edu

Abstract

This position paper investigates the challenges and opportunities in providing access to machine learning (ML) tools to high school and early undergraduate students working on "Physical Computing" projects. By co-creating a library of ML tools for two popular interactive art programing IDEs (Max and PureData), and by introducing them in the classroom to early undergraduate students, the author has been developing and exploring the necessary scaffolding for creating new tools. These experiences point to several critical needs to successful integration of ML tools in classrooms with non-experts: notably, experiential examples, minimally viable software-hardware systems, and "one-click-install" cross-platform and embeddable software packages.

Author Keywords

AMachine Learning; Physical Computing; Interactive Art; Undergraduate; High School; Max; PureData

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous; See [http://acm.org/about/class/1998/]: for full list of ACM classifiers. This section is required.

Introduction

This paper is motived by the author's desire to introduce ML to introductory creative practitioners working with the area of activity commonly referred to as “Physical Computing”. As a creator of interactive systems, the author considers the notion of *mapping* as a central concept in creating engaging experiences for the user. The author is therefore interested methodologies that allow begining critical makers to enhance and expand their notion of *mapping* beyond simple arithmetic operations in order to achieve more complex mappings between sensors and actuators, vastly improved classification or recognition results in processing sensor data, more complex time-based behavior from autonomous systems, and opportunities for surprises and extrapolations.

Early results in experiences with this venture point to several critical challenges that are specific to the physical computing classroom and curriculum. These challenges include: the contrasting design of physical computing classrooms and laboratories (as compared with computer clusters or individual work spaces); the vast range and accelerated evolution of development platforms for introductory work with embedded systems; the overwhelming wealth–and deficit–of existing online examples for physical computing work that exemplifies very simplistic approaches to *mapping*; the overbearing prerequisite systems knowledge for setting up a functional work environment for utilizing ML. Initial experiences also point to a number of promising opportunities within this area: accessibility and availability of 1) Low Energy Bluetooth devices (BLE) as an alternative physical computing platform to ATMEL based microcontrollers that function within the Arduino IDE; 2) wifi-enabled physical computing platforms that can communicate with cloud-based systems for creating more complex interactions (i.e. IoT); impressive capability with introductory students to create apps on mobile platforms that communicate with embedded physical computing systems.

Background

The author has led the design and implementation of an introductory physical computing curriculum with the Integrated Design, Art and Technology (IDEATE) Program at Carnegie Mellon University, in collaboration with Professor Garth Zeglin, a colleague from the Robotics Instutute. The first iteration of the included a one-room-schoolhouse with approximatly 35 students from 12 departments pursuing project-based learning in collaborative teams. The second iteration of the course separated the students by major discipline and matched them with contrasting professors (i.e. students from engineering and sciences taught by an arts professor, and fine arts students taught by a robotics professor). The course utilized an online curriculum with refrences to a wide range of existing tools, tutorials and external resoruces for scaffolding the individualized learning through collaborative group projects. Among these resources, a set of ML tools implemented in Java, Max, PureData and OpenFrameworks were made available to students. Assignment prompts for the course include projects like: one-input-one-output interactive system, mobile robot with autonomous behavior, medium translation device (e.g. from light to sound, or from sound to movement), and an Internet of Things device. The course gives students freedom of platform, tho the wide majority work with the omnipresent Arduino microcontroller, with some students adventuring into Low Energy Bluetooth devices (BLE), embedded linux systms (e.g. Raspberry Pi, BeagleBone) and wifi-enabled microcontrollers.

A number of ML libraries designed for creative practitioners have found been developed and gained popularity in the past few years. The author’s recent publication on his co-authored contribution to this domain [xx ml.lib] provides an extensive literature review of ML tools for artists. The list includes many valuable contributions by the organizers of this workshop. These packages attempt to lower the entry fee for working with ML through a number of strategies: some packages create stand-alone systems that communicate with software platforms of one's choosing (e.g. [xxx wekanator]); others attempt to bring ML into programming environments that are the preferred choices for artists and creative practitioners (e.g. [xxx ml.lib] [xx M-to-n] [xx OfxLearn] [xx OfxGRT]); another approach is to place the computational intensive and complex ML work in the cloud and allow the end-user to user HTML technologies to tap into the magic [xxx deep learning on the web]. Several researchers and educators have created very helpful courses and syllabi for working doing creative work with ML [xxx Gene Kogan]; such courses prove to be invaluable resources for students as they also provide salient creative examples of ML at work in creating artistic projects. In the author's assessment, the bulk of these resources are designed for practitioners active in machine vision, information, advanced audio-visual synthesis, generative art; all areas that require the computational horsepower of a personal computer–or more–as opposed to a embedded platforms used in typical physical computing applications.

ACM Copyrights & Permission

How ML Can Help Physical Computing

These are exciting times for Physical Computing: the Arduino revolution has made physical computing a house hold phrase, with instructables and DIY sample projects available by the thousands online. The cost of embedded systems has dropped significantly, new platforms are introduced practically each week, an the cost of input/output components (e.g. sensors, actuator) and their accessibility (with help from the break-out board culture speer-headed by vendors like SparkFun and Adafruit) makes it possible to create faily complex sysems with affordable investments in time and money. Nonetheless, the author notes that the notion of *mapping* remains limited to the Arduino *scale* function for the wide majority of projects. In instances when more complex decision making is necessary (e.g. an autonomous robot), code created by students is dominate by *if... then* or *case* statements.

Orientation Classification of a Smart Object
Consider the following ever-recurring introductory physical computing project: A student wishes to create a "Magic 8-Ball" that can report a reading of the future based on the orientation in which the user holds the object. The student grabs an I2C 3-axis accelerometer modules (now available for under 2, with matching arduino libraries readily downloadable from a variety of online resources), successfully connects to an Arduino Uno within minutes and is ready to *map*. The student spends the following several hours struggline with noisy sensor data, experimentally finding appropriate ranges for using a combination of *if... then* with numerous *scale* functions. Eventually, the student produces a more-or-less working prototype. During the critique, if the reviewer suggests a different set of pre-defined orientations to be more natural, the student must repeat the entire process once again. An ML approach would replace the system of *if... then* and *scale* functions a Support Vector Machine, thereby rendering training, recognition and re-training trivial, while vastly improving the overall success rate of orientation classifications.

Gesture Recognition of a Smart Object
Consider yet another ever-recurring introductory physical computing project: a "mart glove" that recognizes gestures and lights various LED's accordingly (think LED powered hand turn signals for a biker). Once again the student successfully connects a cheap multi-axis accelerometer to an Arduino and set about making a creating a simple continuous gesture classifier that can distinguish between a right hand turn and a left hand turn signal. This proves to be rather difficult due to gestural variation among different users or among iterations of a gesture performed by the same user. An ML approach would render the task trivial with a simple algorithm like Dynamic-Time-Warping (DTW).

Expressive Time Varying Behavior
Consider another ever-recurring introductory physical computing project: A student wishes to create a "An Autonomous Mobile Robot" that interacts with users holding a flashlight in ways that exhibit some hint of intelligence or surprise. The professor introduces the student to the ever-giving concept of Braightenberg Vehicles [xxx], and suggests that the student begins with this approach from decades back and improves upon it. The student grabs an existing two-wheel robotic platform (e.g. a Pololu 3Pi or an Arduino Robot), successfully connects two photo-resistors to the ATMEL chips ADC's, and creates a simple mapping between left-sensor-to-right-wheel, and right-sensor-to-left-wheel, based on two transfer-functions between sensor values and the speed of each wheel. The system's impressive responsiveness gets the student excited, however, taking this further is difficult. An ML approach would introduce a mapping approach that can be generative and change over time, e.g. an Hidden-Markov-Model (HMM) or Multi-Layer-Perceptron (MLP), thereby dramatically expanding the range of expressive movements by the robot.

Translation Among Senses
Finally, consider a last ever-recurring *mapping* exercise that presents itself again and again in the classroom: A student wishes to make a system that translates light (vision) into sound (audition). The student grabs a photoresistor from the bin, successfully connects it to an Arduino Uno within minutes, and maps the sensor values to the frequency of an oscillator produced an Arduino audio library [xxx]. The resulting system is functional (more light, higher frequency, less light, lower) but aesthetically disappointing even to the proud maker. A ML approach would allow the student to explore *generative melodies* by introducing transition probabilities among notes (i.e. a Markov Chain); alternatively, an ML approach would enable the student to use a much larger array of sensors, map them to a much larger array oscillators (this time produced by a computer in order to overcome the Arduino's limitations) that are tuned to pre-defined frequencies thus introducing a notion of *harmony* using MLP.

Challenges
The physical computing classroom and work flow presents special challenges to successful integration of ML tools into curriculum. The most fundamental challenge is one of hardware platforms: the majority of physical computing curricula taught across universities today being with the Arduino platform; many of them continue with that platform into more advanced work as well. While some attempts at bringing ML to the ATMEL based platform are documented online (primarily in implementing Decision Trees[xxx]), the computational limitations of the platform present a major barrier. Needless to say, most multi-platform open source ML libraries do not run on embedded microcontrollers.

Another challenge in integrating ML into the physical computing platform relates to the physical design of physical computing laboratories. Physical computing consists of a great deal of hardware work, thereby require a laboratory to be equipped with soldering stations, bins of electronic components, break-out boards and mechanical parts, small to medium size fabrication tools (e.g. saws, laser cutters, 3d printers) along with open work surfaces for assembling projects of varying sizes and shapes. These requirements tend to preclude the inclusion of a large set array of workstations that can have software pre-installed and configured in order to scaffold student work with more complex software packages. In our implementation of a campus-wide physical computing laboratory at Carnegie Mellon, we attempt to address this issue in two ways: 1) an array of laptops available to students at a lending desk that come with preinstalled software, 2) a Puppet system for embedded linux systems (namely for the Raspberry Pi [xxx] and for the Udoo board [xxx]) that allows students to install a huge list of debian packages with a few commands. Both appraoches have had limited success. As for the former, students increasingly tend to feel ownership of their own digital life and workspace and therefore tend to prefer working with their own laptops; this ethos is also consistent with the DIY spirit that surrounds physical computing culture documented throughout the net. As for the latter, despite the efficacy of a Puppet system for embedded linux boards, the initial barrier of getting a Raspberry Pi up and running (get a power supply, connect keyboard and mouse, find a screen and a clean work space, register the ethernet MAC address in order to have internet or make a wireless dongle work, get the Puppet up and running, wait for 4+ hours for the installations to complete) appears to all but the most ambitious or already-experienced students against the platform, especially in light of the global dominance of Arduinos as synonymous with physical computing itself.

The immense popularity of the Arduino and Raspberry Pi paltforms and the googles of accompanying documentation and sample projects online allow educators to attempt a flipped classroom within this complex medium. Our experience has show that students are consistently successful at getting a new or unknown component working through internet existing online instructions. However, our experience as also shows that the vast majority of physical computing projects documented online exhibit very simplistic approaches to *mapping*; namely, the vast majority are limited to using linear mappings between inputs and outputs. While recent advanced research in experimental sensing has contributed very successful and impressive examples of ML at work (e.g. [xxx touche] [xxx touch and activate] [xxx botanicus interacticus]), these systems tend to require more complex hardware as well as software running on a personal computer.

Opportunities

Initial experiences also point to a number of promising opportunities within this area: accessibility and availability of 1) Low Energy Bluetooth devices (BLE) as an alternative physical computing platform to ATMEL based microcontrollers that function within the Arduino IDE; 2) wifi-enabled physical computing platforms that can communicate with cloud-based systems for creating more complex interactions (i.e. IoT); impressive capability with introductory students to create apps on mobile platforms that communicate with embedded physical computing systems.