insight.tech
POWERING IoT AND EMBEDDED INNOVATION

Shrinking Machine Learning Down to IoT Size

Nicholas Cravotta

Despite the potential of machine learning in embedded applications, the technology is struggling to gain adoption due to its complexity. To date, the field of machine learning has been driven by small populations of domain experts with advanced backgrounds in mathematics and computer science.

As a result, typical end users have been left to their own devices when attempting to leverage machine learning in their own applications.

One major challenge facing the lay developer is a dearth of data. In a typical workflow, training data is fed into a "learning algorithm," which in turn produces a "hypothesis function" that attempts to replicate the desired behavior **(Figure 1)**. Test data is then run through the hypothesis function, and the results are used to optimize the function's performance.

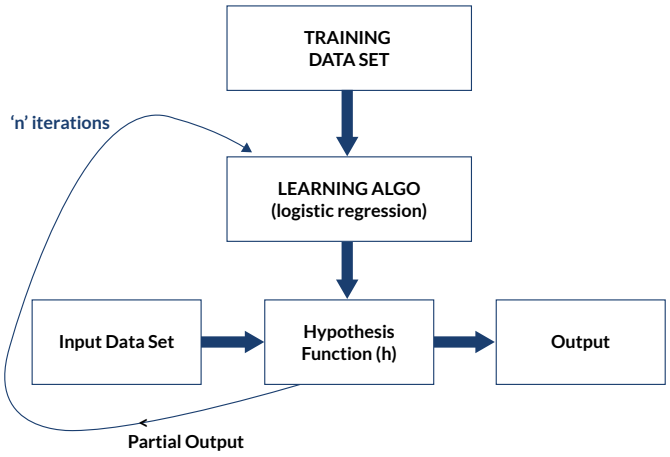

Figure 1. Machine learning development requires data for training the initial model and tuning the resulting algorithm over time. (Source: Skymind.ai)

Training data is usually constructed by domain experts, who often must make do with whatever data they can get their hands on. It is often difficult to obtain data from the field—and even more difficult to test the hypothesis function with real-time data from the target systems during development. This has prevented machine learning algorithms from reaching their maximum potential efficiency.

A related hurdle is that the modeling portion of machine learning development has historically required data center-class computing, and the output algorithms have not been tuned for embedded processors. Fortunately, development tools and hardware platforms are beginning to surface that tailor machine learning to embedded systems.

Machine Learning Prototyping and Principles

SECO, for example, has integrated components for machine learning model construction and algorithm training into its UDOO App Inventor (UAPPI) tool suite. UAPPI is a web-based integrated development environment (IDE) based on MIT's open-source App Inventor 2 platform

UAPPI allows users to design systems and applications without writing any traditional source code **(Figure 2)**. Rather, functionality in the UAPPI environment is developed and implemented using visual building blocks for each logical capability. The result is fast development and prototyping of core functionality, including graphical user interfaces (GUIs), network connectivity, and database storage, as well as machine learning modeling and algorithm training.

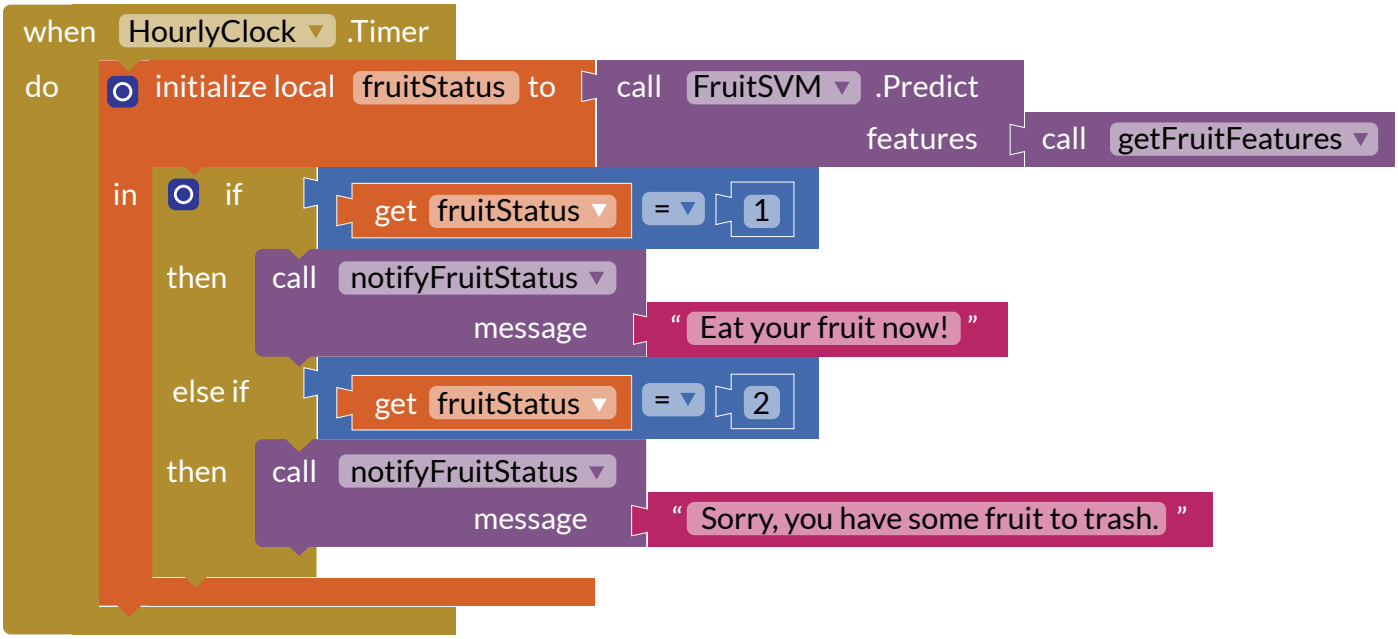

Figure 2. The UDOO App Inventor (UAPPI) is a web-based IDE. (Source: SECO)

The core machine learning component within UAPPI is the UdooSvm, a supervised learning model and set of associated data classification algorithms based on support vector machines (SVMs). When provided a training data set that includes feature vectors and tags, UdooSvm uses a mapping function to classify unlabeled features as it encounters them.

For a better understanding of how UdooSvm generates machine learning algorithms that are suitable for use in embedded systems, let's explore an example.

Better Bananas with Machine Learning

Consider a system that determines when a banana is ripe by measuring gases released by bacteria. The hypothetical system consists of a banana in a container that has been punctured with air holes, as well as a companion gas sensor array that measures air composition every five minutes.

The kinds and levels of gases associated with "ripeness" are not known to the system when it is first being trained.

Therefore, data collected by the gas sensor array is initially tagged as "Good Fruit."

As the days pass, the air composition changes while the banana completes its ripening, and data samples are tagged as "Going to Rot." The banana eventually becomes black, releases a different mixture of gases, and the data is tagged "Rotten Fruit."

With this data set, learning models can be created using a variety of SVM kernels, including a Linear, Polynomial, or radial-based function (RBF) kernel. To determine their accuracy, test data can be extracted from the original training data set and compared with the algorithm produced by each model **(Figure 3)**.

Once the most accurate learning model has been selected, it can be used to generate a UdooSvm algorithm for the "ripeness" system.

UdooSvm algorithms are optimized to run on embedded processor targets but are also unaware of what specific

SVM Kernel Used	nSV*	Accuracy
Linear	247	94.75%
Polynomial (d=3, g=1)	126	99.07%
RBF (g=1)	188	98.45%

Figure 3. Different kernels produce models with varying degrees of accuracy. (Source: SECO) *nSV = number of support vectors

feature vectors represent or even how they are generated. This means that feature vectors and tags based on data from components such as an embedded sensor can be folded back into the algorithm so that it can evolve autonomously over time.

Adding Intelligence, Simply

UAPPI and the UdooSvm algorithm are optimized to work with SECO UDOO development boards, including the UDOO x86 Arduino 101 boards **(Figure 4)**. Because UDOO x86 offers full compatibility with Arduino libraries and shields, developers can quickly source a range of sensors from the Arduino ecosystem to supply UdooSvm algorithms with real-time input data.

Figure 4. The UDOO x86 is an Arduino 101 development board. (Source: SECO)

UDOO x86 is available with processors including an Intel Atom® processor X5-E8000, Intel® Celeron® processor N3160, or Intel® Pentium® processor N3710. This scalable set of processors available supports everything from algorithms with only a few vectors to complex applications with multiple data sources.

Machines Going Their Own Way

UAPPI, UdooSvm, and UDOO x86 are not the only machine learning prototyping tools available, but they do offer a level of abstraction that makes machine learning more accessible to developers of all backgrounds. And with access to the Arduino ecosystem, more sophisticated machine learning systems can be developed quickly and easily.

It isn't hard to imagine a not-so-distant future in which machine learning systems are deployed with an initial factory algorithm and then improve themselves on their own based on environmental data they capture in the field. At a time when automation has become the only way to keep pace with the fast-moving world of technology, systems that can learn independently will be too hard to resist.