Prototyping the Useless Butler: Machine Learning for IoT Designers

Péter Kun & Kars Alfrink
ThingsCon Amsterdam 2017

Introductions

— KARS ALFRINK

Leapfrog

— PÉTER KUN

TU Delft, Industrial Design
Engineering

> … and you?
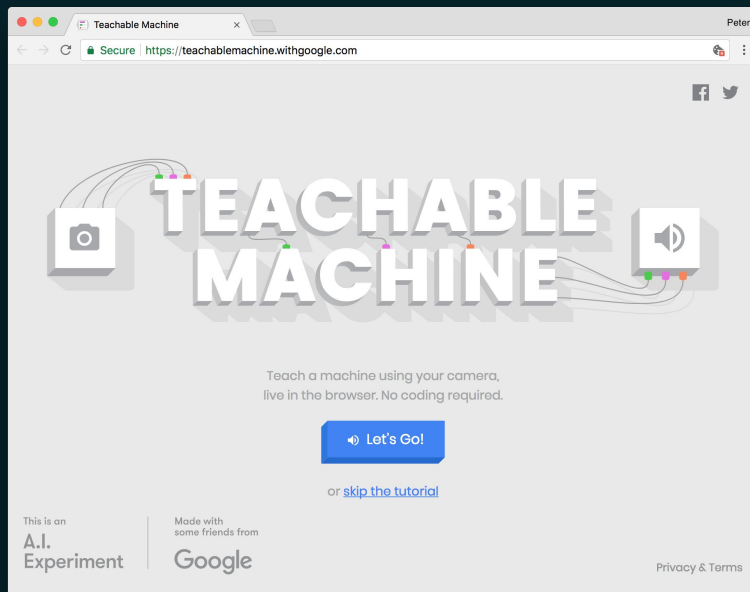*Experience with Arduino, ML, Wekinator?*

Overview

1. Brief introduction to machine learning
2. Overview of the toolchain: Wekinator, MKR1000, OSC
3. Exercises: regression, classification, dynamic time warping
4. Playtime
5. Discussion and close-out

Machine learning

Machine learning

https://teachablemachine.withgoogle.com/

Machine learning

if
then
else
|
train
learn
run
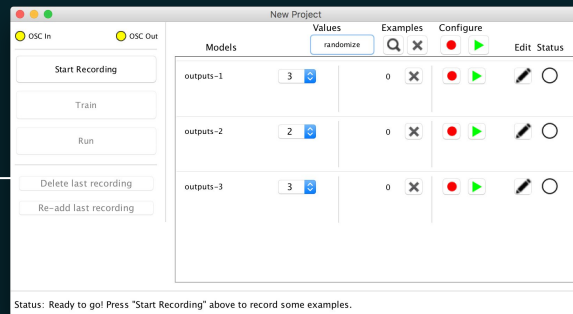
Machine learning

All these are just interfaces (abstractions) for ML algorithms.
The Wekinator as well.

Toolchain

Toolchain

sensors

actuators

Arduino
MKR1000

OpenSoundControl
Through UDP

Wekinator

Toolchain – Rationale

- Get first-hand feel for ML
- Few moving parts
- Prototype interactive ML products
- Automate wizard of oz
- Modules and components with embedded ML on the horizon

Toolchain – Wekinator

The Wekinator is free, open source software originally created in 2009 by [Rebecca Fiebrink](). The Wekinator allows users to build new interactive systems by demonstrating human actions and computer responses, instead of writing programming code.
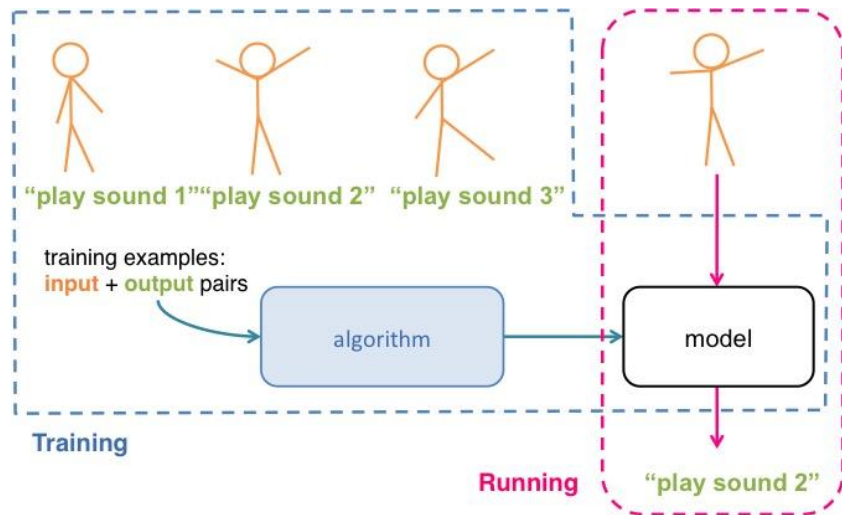
Toolchain – MKR1000

Arduino MKR1000 is a powerful board that combines the functionality of the Zero and the Wi-Fi Shield. It is the ideal solution for makers wanting to design IoT projects with minimal previous experience in networking.

Zero is a simple and powerful 32-bit extension of the platform established by the Uno. This board aims to provide a platform for innovative projects in smart IoT devices, wearable technology, high-tech automation, crazy robotics, and much more.

Toolchain – OSC

Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology.

We use an Arduino and Teensy library implementation of OSC. It was developed at CNMAT (The Center for New Music and Audio Technologies at UC Berkeley) where OSC was invented.
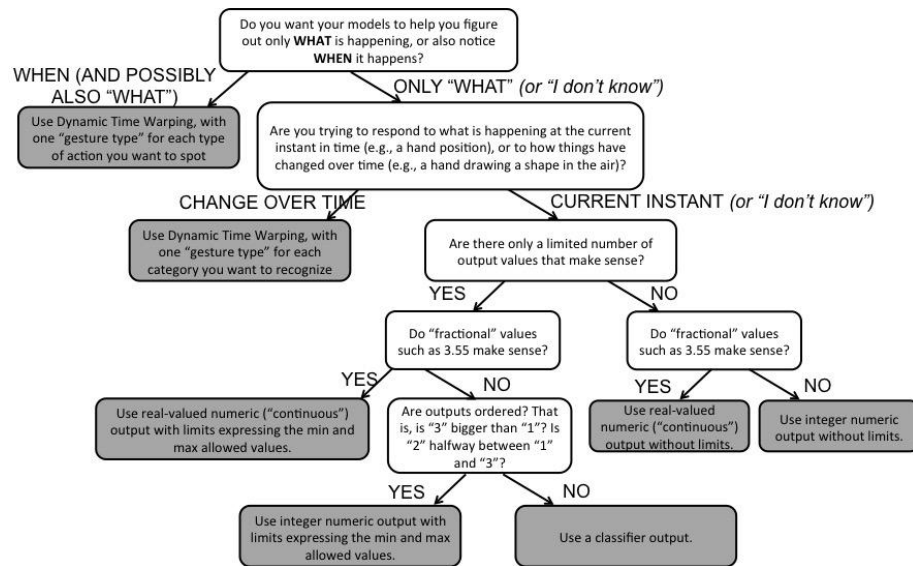
Exercises

Before we get started…

Download and install all the things!

http://bit.ly/useless-butler

Three types of output

1. Regression
2. Classification
3. Dynamic Time Warping (DTW)

Regression

Real-valued ("continuous") numeric outputs can take on any number value (possibly limited to a certain range). For example, you might want to control "audio gain" with a real-valued output limited between 0 and 1. This is the default output type in Wekinator.

Think of this as a smart slider.

Regression – Circuit

- RGB LED on digital PWM pins 2, 3, 4
- Two photoresistors on analog pins A0, A1

Note: Anode RGB LED goes to ground instead of 5V

fritzing
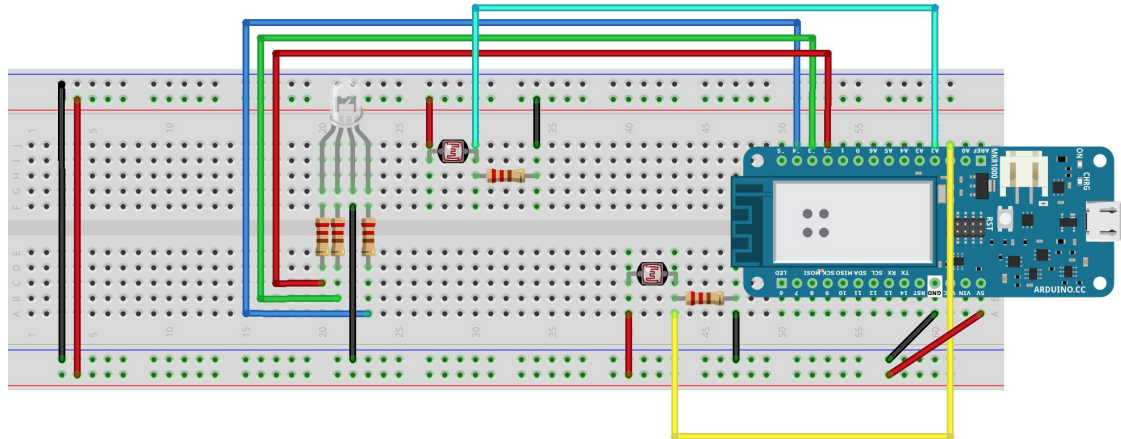
Regression – Code

1. Download example code from GitHub
2. Open 'regressionExample'
3. Copy config_sample.h and rename to config.h
4. Set SSID and password in config.h
5. Get laptop IP from network settings and set in Arduino IDE 'outIp'
6. Upload sketch to MKR1000
7. Make note of MKR1000 IP in Arduino IDE serial monitor

Regression – Wekinator

- Two inputs
- Three outputs
- All continuous (default)
- Don't forget to set your MKR1000's IP under 'host'
- Leave ports as they are
- Hit 'next' to start training

Regression – Training & Running

Classification

These are discrete categories, such as "Position 1", "Position 2," "Position 3." You'll need to tell Wekinator how many categories to use. Wekinator will send outputs as numbers, such as "1," "2," "3" for categories 1, 2, and 3. Wekinator will attempt to categorize every new input you send it.

Think of this as a smart switch.

Classification – Circuit

Same as regression

Classification – Code

1. Open 'classificationExample'
2. Copy config_sample.h and rename to config.h
3. Set SSID and password in config.h
4. Get laptop IP from network settings and set in Arduino IDE 'outIp'
5. Upload sketch to MKR1000
6. Make note of MKR1000 IP in Arduino IDE serial monitor

Classification – Wekinator

- Two inputs
- One output
- All classifiers
- Four classes
- Don't forget to set your MKR1000's IP under 'host'
- Leave ports as they are
- Hit 'next' to start training

Classification – Training & Running

Dynamic Time Warping (DTW)

Use this output type when you want Wekinator to recognize patterns over time. For instance, you might want to play one note every time you draw a circle in the air with your hand, and another note every time you draw a square. If you're not drawing either one, or if you're in the middle of drawing, you don't want anything to happen. That is, you want Wekinator to look for a particular pattern (or multiple patterns) of how the inputs are changing over time, and tell you when a pattern is spotted and which one it was.

Think of this as a smart button.

Dynamic Time Warping – Circuit

Same as previous exercises

Dynamic Time Warping – Code

1. Open 'dtwExample'
2. Copy config_sample.h and rename to config.h
3. Set SSID and password in config.h
4. Get laptop IP from network settings and set in Arduino IDE 'outIp'
5. Upload sketch to MKR1000
6. Make note of MKR1000 IP in Arduino IDE serial monitor

Dynamic Time Warping – Wekinator

- Two inputs
- One output
- All DTW
- Three gesture types
- Don't forget to set your MKR1000's IP under 'host'
- Leave ports as they are
- Hit 'next' to start training

Dynamic Time Warping – Training & Running

Rename your outputs to "output/1", "output/2" and "output/3" – underscores won't work!

Playtime

Discussion

Thank you!