ELOQUENT ARDUINO

SHOW MENU

How to deploy an Arduino Machine learning classifier in 4 easy steps

11 NOVEMBER 2019 / SIMONE

Are you getting started with **Machine learning on Arduino boards**? Do you want to run the model you trained in Python into **any C++ project**, be it Arduino, STM32, ESP32?

In this tutorial I'll show you how easy it is: we'll go from start to end in just **4 easy steps!**

Step 1. Load the data

To train a classifier, we need some data. If you're starting from zero and don't have already a preferred folder structure, I suggest you to create a folder that will hold the data you collect.

Inside this folder, create a dedicated file (`.csv`) for each of the classes you want to class putting a sample on each line. If doing so, you can use the next function to load this data.

```
import numpy as np
from glob import glob
from os.path import basename

def load_features(folder):
    dataset = None
    classmap = {}
    for class_idx, filename in enumerate(glob('%s/*.csv' % folder)):
        class_name = basename(filename)[:-4]
        classmap[class_idx] = class_name
        samples = np.loadtxt(filename, dtype=float, delimiter=',')
        labels = np.ones((len(samples), 1)) * class_idx
        samples = np.hstack((samples, labels))
        dataset = samples if dataset is None else np.vstack((dataset, samples))

    return dataset, classmap
```

Step 2. Train the classifier

Once you have the data, it's time to train the classifier.

The `micromlgen` package (the package that can port Machine learning classifiers to plain C) supports the following classes:

- Decision Tree
- Random Forest)
- XGBoost
- Gaussian NB
- Support Vector Machines
- Relevance Vector Machines
- SEFR

I really like Random Forest, but you can swap it with any of the other ones without changing the rest of the code.

```
from sklearn.ensemble import RandomForestClassifier

def get_classifier(features):
    X, y = features[:, :-1], features[:, -1]
    return RandomForestClassifier(20, max_depth=10).fit(X, y)
```

Step 3. Export to plain C

Now you can convert the trained classifier to plain C code using the `micromlgen` package.

```
pip install micromlgen
```

```
from micromlgen import port

if __name__ == '__main__':
    features, classmap = load_features('your-data-folder')
    classifier = get_model(features)
    c_code = port(classifier, classmap=classmap)
    print(c_code)
```

This is the code you need to import in your Arduino project. To follow along with the tutorials on this blog, save it in a file called `model.h`.

Step 4. Use in your project

Now we have the code we need to run Machine learning directly on our microcontroller.

```
// put the code you got in Step 3 into this file
#include "model.h"

// this class will be different if you used another type of classifier, just check the
model.h file
Eloquent::ML::Port::RandomForest classifier;

void classify() {
    float x_sample[] = { /* fill this vector with sample values */ };

    Serial.print("Predicted class: ");
    Serial.println(classifier.predictLabel(x_sample));
}
```

That's it: if everything went fine, your microcontroller is running Machine learning!