trainGMM, v.1.0

Title: README_trainGMM

Release date: January 6, 2020

This code was developed by Brenton Blair and Andrew Glenn at Lawrence Livermore National Laboratory. THIS CODE IS COVERED BY THE MIT SOFTWARE LICENSE.

ABOUT THE trainGMM CODE

trainGMM performs Gaussian mixture model (GMM) analysis over a user-provided data set of sampled waveforms to compute vectors suitable for discriminating between kinds of pulse shapes, produces graphics illustrating the results, and writes output data to a file. The output data can be uploaded to special FPGA firmware that runs on the Struck SIS3316 digitizer. The parameters of the GMM and the data it operates on are restricted to match several hard-coded parameters expected by the Struck FPGA code. The code has a few user-settable parameters.

trainGMM was developed in python and is run using python. The instructions below instruct you how to acquire and install python and run trainGMM. You can find some of the internal details in the following journal article:

Simms L.M., Blair B., Ruz J., Wurtz R., Kaplan A.D., Glenn A., 2018, "Pulse discrimination with a Gaussian mixture model on an FPGA", NIMPA, 900, 1
https://doi.org/10.1016/j.nima.2018.05.039

HOW TO INSTALL AND USE PYTHON

It is recommended that you install the Anaconda package, which includes Python and all dependent codes that are required for running trainGMM. You can download Anaconda from the following URL:

https://www.anaconda.com/distribution/

Select the Python 3.0 version, and follow the instructions therein for installing Anaconda.

It is also possible to install python separately, but in that case it will also be necessary to install the packages that trainGMM requires. This can be done using the pip installer as follows:

python3 -m pip install --upgrade pip

python3 -m pip install sklearn

python3 -m pip install scikit-learn

python3 -m pip install numpy

python3 -m pip install matplotlib

python3 -m pip install tk

INPUT

The input file requires a mixture of digitized scintillator pulses from neutron and gamma events of varying energy. The script recognizes two kinds of input data formats: binary and text. The binary filename must end in .dat. The text filename must end in .txt. The binary file format follows the Struck format. See Struck documentation of the SIS3316 for more detail. trainGMM is capable of dealing with a file with or without a 16-byte main header. The main header is followed by records for every pulse consisting of a pulse header and a raw waveform. The pulse records as detailed in the Struck document are flexible and are self-described in the pulse header; trainGMM is capable of parsing the flexible Struck data format. The text file format is at least 180 space-separated columns, typically 258 total columns. The first two columns are unused, the last 178 or more columns are the samples of the pulses.

The user interface also requires five input values: the input filename, the number of events to use, the minimum integral of counts in a pulse used, the number of the channel (for binary data only) that the code is computing the templates for, and the start of the tail in the post-baseline pulse for visualizations relying on simple charge ratio. All but the filename reverts to a default value.

The samples of the pulse are dealt with in the following manner:
The first 42 samples are unused.
The next 8 samples are used to compute a baseline.
The next 128 samples are used for training and testing.
Any trailing samples after the first 178 samples are unused and need not be present in the either the .dat or .txt file.

OUTPUT

Each time the “Train” button is pressed, the program produces a diagnostic window on your screen. The window contains four graphs that appear in a two-by-two grid. The top row shows the results of the training. The upper left corner shows the two templates. The upper right corner shows the integral baseline-subtracted counts versus simple charge ratio for all events

used for training colored by their likelihood score with respect to the template with the lower first-sample value (indicative of neutrons in liquid scintillator). The bottom row shows the results of test. If the test file is not specified, the training events are used for test. The lower left corner shows the integral baseline-subtracted counts versus simple charge ratio for all events used for test. The lower right corner shows the integral baseline-subtracted counts versus a particle score formed from the difference of the two likelihoods divided by their sum.

When you press “Save Templates”, the program outputs a new text file containing 385 rows of one value each. The first 128 rows are the first of the two templates, the second 128 rows are the second of two templates, the third 128 rows are the covariance diagonal, and the last value is currently un-used. The templates are intended to match certain specific requirements hard-coded into the Struck firmware.

HOW TO RUN trainGMM

In a terminal window in your working directory, type:

python trainGMM.py

from the command line. This will instantiate a python session and bring up the trainer GUI tool in a window on your screen. The boxes and one button in the interface allow you to modify the input parameters to trainGMM. A summary of the interface is as follows:

1. Choose File button

Clicking on this button brings up a session to select an input file. Navigate to the directory containing your input file and select it. The extension “.dat” causes the program to treat the input file as a binary file, the extension “.txt” causes the program to treat the input file as a text file.

2. Max Events box. Default = 10000

This limits the total number of events used to obtain the output templates. If the input file is very long, processing can take a very long time. It has been found that generally no more than 10,000 events (and often a lot fewer) are necessary for producing templates. If the kinds of events are mixed throughout the file, and not segregated nor sorted by particle type or integral counts (energy), then use this option to build templates from the first few events. Events below the integral threshold are included in the counted events, but not used in the training set.

3. Min Integral box. Default = 0

This limits the trainer to events of no less than a user-selected amount of integral counts. Low-energy events tend to be noisier and less-useful for training the templates. We recommend you

start with a “min integral” of 0. If it is apparent from the graphical output that the training did not successfully separate the dataset into neutrons and gammas due to noise at low energies, try to use a “min integral” setting at value of the integral in the charge comparison plots where the populations appear to begin separating. Once you’ve found a successful minimum, you may search for the lowest value where training is successful.

4. Tail Start box. Default = 17

This parameter helps you perform a simple charge ratio for visual comparison of the familiar energy versus charge ratio graph. There is currently no automation to determine the tail region of the tail-to-total feedback plots. “Tail start” is the number of the sample inside the 128 samples of the pulse used for the templates, so a setting of 17 is the 67^{th} sample of the entire 256. For many applications, a setting of peak sample plus 10-20 will provide good separation in tail-to-total to confirm the model has separated the populations into neutrons and gammas.

5. Channel (bin only) box. For binary datafiles only. Default = 0

The current version of the trainer produces the three templates for one channel at a time. For input data consisting of multiple channels, you must run the trainer once for each channel. This box allows you to choose which channel you are training for.

Running the code:

trainGMM has three actions: “Train”, “Save Templates”, and “Test”.

1. Train

A run of 10,000 events takes less than 10 seconds to run on a 4-core 2.8 GHz Mac.

2. Save Templates

Once you are satisfied with the graphs of the outcome of the training session, you can save the result of the training session using the Save Templates button. You will be prompted for a filename, and the templates will be saved into a file whose format is described in the output section above.

3. Test

Each time the “Test” button is pressed, the bottom two windows are refreshed according to values selected by the user. It is generally advisable to train and test on two different datasets. A dataset can be split into two files, one for train and one for test. It is also advisable to test on gamma source data to get an idea of the thresholds for separating the populations for analysis.

Happy computing! Be sure to convey any problems running trainGMM to the developers, and feel free to make suggestions for improvement.

RELEASE

LLNL-CODE-800564