How to Train Regression and Classification Models for Audio Samples Using PyAudioAnalysis

By Cassiel Moroney
B21 2018

0. Introduction

For this tutorial, you should know how to open the command line and execute commands, or how to write and execute python programs. You will also need to have python installed on your computer.

> *Tips for the very beginner:*
> Newer Macs should come with python installed. Windows does not come with python but can be downloaded directly from the python website: https://www.python.org/downloads/windows/. (I run Python 2.7 and cannot vouch for the efficacy of other versions.) You can download an exe that will help you install it. Make sure to select the "add to path" option. If you miss it, you will have to add python to your path manually. There are plenty of guides to this on the internet. You will need to choose one specific to your operating system.
>
> Pip is an install manager for python libraries. To install pip, visit https://pip.pypa.io/en/stable/installing/. Place the get-pip.py file in the python folder you installed. Open it with python by typing python get-pip.py into the command line.

All examples are taken from https://github.com/tyiannak/pyAudioAnalysis/wiki/4.-Classification-and-Regression. It has been re-formatted and my own explanations added in the hope it will be more informative for non-STEM and beginner programmers. This library is remarkably powerful, free to use, and incredibly simple to operate once you understand the basics.

1. Clone pyAudioAnalysis

If using the command line: `git clone https://github.com/tyiannak/pyAudioAnalysis.git`

Otherwise, go to the above link, download, and save to either your python path or to the folder you will be working in.

2. Install dependencies

The official documents suggest installing dependencies using pip on the command line: `pip install numpy matplotlib scipy sklearn hmmlearn simplejson eyed3 pydub`

If you do not have pip, you can try to install it, or install and use your own preferred form of command line installation. Mac users might try homebrew. Please search the internet for more complete tutorials about this. Once you have an installer, install the eight dependencies listed above.

3. Decide on your algorithm

Classification will sort a given audio sample into one of the categories you define for the algorithm. For example, it could try to label Mozart's 5th as either rock, pop, jazz, classical, or electronic.

Regression will try to assign a given audio sample a numerical value. This is useful to use as a scale of intensity of a certain audio quality. For example, it could try to give a clip of someone whispering a value between 0 and 10.

4. Prepare your audio files

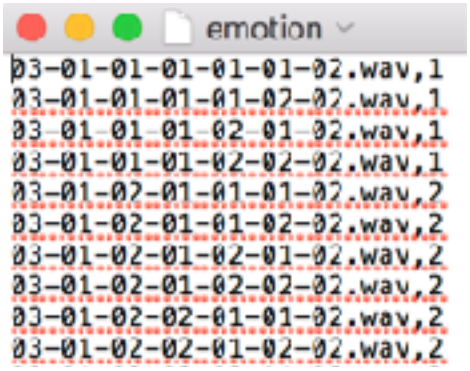
You will need many wav files to train a model. The more, the more accurate your model will be. Depending on what kind of model you want, you will need to organize them differently.

For classification

Make a folder for each category you would like the algorithm to consider and place the corresponding audio files into them. For example: tones of voice like angry, sad, and happy. Write down the paths to said folders.

For regression

Gather all audio samples in one folder. Create a .csv file with a row for each file. In each row, the first cell should be complete file name, including the .wav, and the second cell should contain the expected (correct) value of the quality being trained. Save the .csv file with the name of that quality. You can have more than one of these files in the folder if you would like to train multiple qualities at once. For example, if you have two separate files for arousal and valence, the program will produce an arousal model and a valence model.



Precise formatting is essential for this. It might be easiest to use a simple word processor such as Notepad or TextEdit to create a plain .txt file with one line per row and a comma (no space) to separate the filename from the value. Save it again, this time as a csv. Open in a spreadsheet program to double-check that it is correct.

5. Run the algorithm

The program needs 9 to 10 pieces of information for it to run:

- The name of the file running the main program: always "audioAnalysis.py" for command line and "audioTrainTest" in python
- Which method to use: classification ("trainClassifier" for command line and .featureAndTrain for python) or regression ("trainRegression" or .featureAndTrainRegression")
- The directory/ies to use: the paths to those will suffice. For command line, simply list them (ex: "/data/sad /data/happy /data/afraid") and in python, make an array of strings (ex: " ["/data/sad" , "/data/happy" , "/data/afraid"] ")

- The type of model to train: the options are “svm,” “svm_rbf,” “knn,” “extratrees,” “gradientboosting,” or “randomforest” (commas not included). In python, they must be surrounded by quotations but in command line they must not.
- Mid-term window size and step and short-term window size and step: I am not familiar with audio terms and so I have not explored these values. I found success in leaving them at the default “1.0, 1.0, aT.shortTermWindow, aT.shortTermStep” seen in the provided examples.
- The name of the new model: up to you to decide. In python, surround with quotation marks. The file the algorithm produces and that will be used to do future classifying will have this name.
- Beat: optional, only for beat extraction. A boolean (true/false).

To execute a python program, save the program as a .py file and then execute by opening it or from the command line with `python filename.py` . To execute via the command line, simply copy the desired command and enter.

For classification

An example of a python program:

```
>> from pyAudioAnalysis import audioTrainTest as aT
>>  aT.featureAndTrain(["/home/tyiannak/Desktop/MusicGenre/
Classical/","/home/tyiannak/Desktop/MusicGenre/Electronic/","/home/
tyiannak/Desktop/MusicGenre/Jazz/"], 1.0, 1.0, aT.shortTermWindow,
aT.shortTermStep, "svm", "svmMusicGenre3", True)
```

The format for command-line execution is (replace the < > with your own data and do not include the < >):

```
python audioAnalysis.py trainClassifier -i <directory1> ...
<directoryN> --method <svm, svm_rbf, knn, extratrees, gradientboosting
or randomforest> -o <modelName> --beat (optional for beat extraction)
```

A few different examples of command line execution:

```
>> python audioAnalysis.py trainClassifier -i classifierData/speech/
classifierData/music/ --method svm -o data/svmSM
```

```
>> python audioAnalysis.py trainClassifier -i /media/tyiannak/My\
Passport/ResearchData/AUDIO/musicGenreClassificationData/ALL_DATA/
WAVs/Electronic/ /media/tyiannak/My\ Passport/ResearchData/AUDIO/
musicGenreClassificationData/ALL_DATA/WAVs/Classical/ /media/tyiannak/
My\ Passport/ResearchData/AUDIO/musicGenreClassificationData/ALL_DATA/
WAVs/Jazz/ --method svm --beat -o data/svmMusicGenre3
```

For regression

An example of a python program:

```
>> from pyAudioAnalysis import audioTrainTest as aT
>>  aT.featureAndTrainRegression("data/speechEmotion/", 1, 1,
aT.shortTermWindow, aT.shortTermStep, "svm", "data/svmSpeechEmotion",
False)
```

The format for command-line execution is:

```
>> python audioAnalysis.py trainRegression -i data/speechEmotion/
classifierData/music/ --method svm --beat -o data/svmSpeechEmotion
```

6. Use your model

You can use your new model on either a single file or on an entire folder of audio samples. It will return its estimates for the requested values.

All color coding remains conceptually the same, except now the name of the new model refers instead to the name of the model you want to use and the training files are now the files to be classified. Instead of `trainClassifier` and `trainRegression`, you will use `classifyFile` and `regressionFile`.

For Single Files

For Classification

An example of a python program:

```
>> from pyAudioAnalysis import audioTrainTest as aT
>> aT.fileClassification("TrueFaith.wav", "data/svmMusicGenre3","svm")
```

The format for command-line execution is:

```
>> python audioAnalysis.py classifyFile -i <inputFilePath> --model
<svm, svm_rbf, knn, extratrees, gradientboosting or randomforest> --
classifier <pathToClassifierModeL>
```

Some examples of command-line execution is:

```
>> python audioAnalysis.py classifyFile -i bach.wav --model svm --
classifier data/svmMusicGenre3

python audioAnalysis.py classifyFile -i 03-01-05-01-01-01-15.wav --
model svm --classifier data/svmEmotionClass2


>> python audioAnalysis.py classifyFile -i bach.wav --model knn --
classifier data/knnMusicGenre3
```

For regression

An example of a python program:

```
>> from audioAnalysis.py import audioTrainTest as aT
>> aT.fileRegression("anger1.wav", "data/svmSpeechEmotion", "svm")
```

The format for command-line execution is:

```
>> python audioAnalysis.py regressionFile -i <inputFilePath> --model
<svm, svm_rbf, knn, extratrees, gradientboosting or randomforest> --
regression <pathToRegressionModeL>
```

An example of command-line execution is:

```
>> python audioAnalysis.py regressionFile -i anger1.wav --model svm --
regression data/svmSpeechEmotion
```

The result will be the programs estimate of the numerical expression of the trained quality. If multiple qualities were trained, the result will have a form such as:
`([0.6999797228926365, -0.30015527588853286], ['arousal', 'valence'])`

For Folders

For Classification

Some examples of command-line execution:

```
>> python audioAnalysis.py classifyFolder -i testFolder/ --model svm
--classifier data/svmSM

>> python audioAnalysis.py classifyFolder -i testFolder/ --model svm
--classifier data/svmSM --details
```

For command line execution, the final argument --details is optional. It provides a full breakdown by file instead of a summary of the classes.

For regression

Some examples of command-line execution:

```
>> python audioAnalysis.py regressionFolder -i ~/ResearchData/AUDIO/
emotionSpeechData/germanSegments/Anger/ --model svm --regression data/
svmSpeechEmotion

>> python audioAnalysis.py regressionFolder -i ~/ResearchData/AUDIO/
emotionSpeechData/germanSegments/Sadness/ --model svm --regression
data/svmSpeechEmotion

>> python audioAnalysis.py regressionFolder -i ~/ResearchData/AUDIO/
emotionSpeechData/germanSegments/Happiness/ --model svm --regression
data/svmSpeechEmotion
```

The result will be a histogram based on the analysis.

7. Tips and tricks

The two most likely problems you will encounter are:

Installation problems: your python version is unable to access the required libraries or parts of pyAudioAnalysis due to a problem with your installation. Double-check your python path (this is OS-specific, so you will need to google for details) to make sure that your version of python is able to access all dependencies. Also double check where you are running your program out of and be sure you are running it from a folder that holds the pyAudioAnalysis folder (if you did not install it directly in your python path). It should also hold your training files.

Formatting problems: if you are a new programmer, please be advised that formatting is absolutely crucial. If you are receiving error messages that say your training or classification files are not found, or that some piece of information is missing, you are most likely typing something in incorrectly. If necessary, copy paste the examples directly into a word processing program and then substitute in your own values to ensure that proper formatting is maintained.