PyEEG
A Python Module for EEG Feature Extraction

Forrest Sheng Bao\textsuperscript{1,2} and Christina R. Zhang\textsuperscript{3}

\textsuperscript{1} Department of Computer Science, Texas Tech University, Lubbock, Texas
\textsuperscript{2} Department of Electrical Engineering, Texas Tech University, Lubbock, Texas
\textsuperscript{3} Department of Physiology, McGill University, Canada

Jun. 30, 2010, Scipy 2010, UT, Austin, Texas
Outline

1. Introduction
   - Feature Extraction in Neurological Signal Processing
   - EEG: The Name of The Game
   - EEG Reading in Naked Eyes
   - A Case Study: Features may be better
   - Why PyEEG?

2. PyEEG
   - Main Framework
   - EEG Pre-processing
   - Features
   - Basic Usage

3. Future work

4. Questions?
Over the past decade, computer-aided diagnosis (CAD) systems based on EEG have emerged in the early diagnosis of several neural diseases such as Alzheimer’s disease [1] and epilepsy [2].

A key component in most such CAD systems is to characterize EEG signals into certain features, a process known as feature extraction.

EEG features can come from different fields that study time series: power spectrum density from classical signal processing, fractal dimensions from computational geometry, entropies from information theory, synchrony measures from nonlinear physics, etc.

By extracting EEG features, we can use more powerful tools, such as machine learning, to analyze the signal in the next step.
EEG: The Name of the Game
EEG Reading Cards: The Limit of Our Eyes
Over 50 million people worldwide suffer from epilepsy.

Conventional epilepsy diagnosis may require long-term or repeated EEG recordings to capture seizures (ictal) or other epileptic activities. Physicians visually inspect lengthy EEG recordings - paging through 24-hr waveforms.

To help those MDs, research on picking out segments that physicians may interest has been under going for quite a while.

Recently, researchers have found it promising to use interictal (i.e., non-seizure) EEG records that do not contain particular activities. We can tell whether a EEG segment is epileptic [3] or where the seizure foci are [4], based on EEG features extracted.

Furthermore, it is possible to predict the state of the brain by tracking the values of those features. This idea is use by seizure predition researchers.
Why PyEEG?

- The computational analysis to EEG signal as described above requires a toolbox to quantify EEG patterns. Turn “The stock went rocket high” into “NASDAQ increased 5%.”
- Such a toolbox can be very useful to computational neuroscience community.
- There is no highly active project as expected out there (e.g., pbrain, ptsa).
- We are not aware of an open-source “simple” (simplifying every aspect of using) Python solution.
- “Software is like sex; it’s better when it’s free.” (Free as in GPL.)
Main Framework (cond.)

- PyEEG consists of two sets of functions, *EEG pre-processing functions*, which do not return any feature values, and *feature extraction functions* that return feature values.

- Besides standard Python functions, PyEEG only uses functions provided by Numpy/SciPy.

- PyEEG does not define any new data structure, using standard Python and NumPy ones only.

- The inputs of all functions are time series in form of a list of floating-point numbers and a set of optional feature extraction parameters. Parameters have default values.

- The output of a feature extraction function is a floating-point number if the feature is a scalar or a list of floating-point numbers if it is a vector.
EEG Pre-processing

- Pre-processing: to build new time series from given time series for further computation.
- PyEEG currently provides two pre-processing functions.
- embed_seq(): to build embedding sequence (from given lag and embedding dimension)
- first_order_diff(): to compute first-order differential sequence. One can build differential sequences of higher orders by apply first-order differential computing repeatedly.
## Features

<table>
<thead>
<tr>
<th>feature</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Intensity Ratios (RIRs) [3]</td>
<td>a 1-D vector</td>
</tr>
<tr>
<td>Petrosian Fractal Dimension (PFD) [5]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Higuchi Fractal Dimension (HFD) [6]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Hjorth mobility &amp; complexity [7]</td>
<td>a 1-by-2 vector</td>
</tr>
<tr>
<td>Spectral Entropy (entropy of RIRs)</td>
<td>a scalar</td>
</tr>
<tr>
<td>SVD Entropy [8]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Fisher Information [9]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Approximate Entropy (ApEn) [10]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Sample Entropy (SampEn) [11]</td>
<td>a scalar</td>
</tr>
<tr>
<td>Detrended Fluctuation Analysis (DFA) [12]</td>
<td>a scalar</td>
</tr>
</tbody>
</table>

More features are coming in PyEEG.
No reinvention to the wheel (skewness, etc.).
Basic Usage

SciPy is required to run PyEEG. The latest PyEEG is released as a single Python script, which includes all functions. So users only need to download and place it under a directory that is in Python module search path, such as the working directory. Example:

```python
>>> import pyeeg
>>> from numpy.random import randn
>>> for i in xrange(0,10):
...     pyeeg.dfa(randn(4096))
... 0.50473407278667271
 0.53339499445571614
 0.53034354430841246
 0.50844373446375624
 0.5162319368337136
 0.46319279647779976
 0.44515512343867669
 0.4407740703026245
 0.45894672465613884
 0.49135727073171609
```
Future work

- More features.
- More comprehensive documentations.
- Unit test for all functions.
- Faster implementation.
- File I/O.
- (Probably) Integration with nipy/pbrain
  
Questions?

Licensed under GPL v3 at http://code.google.com/p/pyeeg/


