**DARPA-RAM documentation  
Overview**

This code contains a set of classes and methods, mainly to perform two tasks:

1. To provide an easy-to-use interface for reading the DARPA RAM’s publicly available data.
2. To implement the classifier explained in Ezzyat et. al. (2017).

Classes:

The following classes have been implemented using MATLAB:

1. ***subject***
2. ***experiment***
3. ***FR1*** and ***catFR1*** (which are inherited from *experiment* class)
4. ***session***
5. ***sessionCollection***

**subject**

This class represents a subject and will contain the data related to the performed experiments (FR1, catFR1, etc.).

To create a new *subject* object:

subj1 = subject(<r1.json file path> , <patient id>);

To load an experiment:

subj1.loadexperiment(<experiment type, e.g. 'FR1' or 'FR2', etc.>)

Note that in order to load an experiment the corresponding class should be implemented. FR1 and catFR1 are currently implemented.

To see available experiment for the subject:

subj1.getexperimenttypes

**experiment**

By calling *subject.loadexperiment* a new subclass of *experiment* is created (e.g. FR1, catFR1, etc.) and the session data are loaded in. Bipolar EEG data is loaded according to the pairs indicated in the RAM’s provided meta data.

After loading, each experiment’s session data can be accessed:

subj1.experiments.<experiment type>.sessions.<session id>

For example, to get the EEG data of session x0x30\_ of subj1’s FR1 experiment:

subj1.experiments.FR1.sessions.x0x30\_.eegData

**session**

This class represents a session. Once loaded, it contains the eeg, and events data, and after the preprocessing step, it will contain the training data for use in training a classifier.

The session data fields are:

* sessionID: session id
* taskEvevnts: a cell array containing the task events
* eegData: This session’s nbpolar sEEG data
* wordEvetsEEG: a 3D matrix of eeg data for “word events” (number of word events X signal length X number of channels)
* trainingData: processed data ready to be used in training
* trainingLabels: event’s “recall” label (0 or 1)
* allEvents: a cell array containing all events
* mathEvents: a cell array containing math events
* sampleRate: sampling rate for current session (500, 1000, or 1600 Hz)
* nSamples: session length

**Experiment specific classes (FR1, catFR1, etc.)**

These *experiment* subclasses are implementing the required pre-processing steps, specific to each experiment type. Since FR1 and catFR1 are very similar, and identical for our current requirements, the two classes have identical implementations. For future use, in case other kinds of preprocessing was needed, only these classes need to be modified or implemented.

In both *FR1* and *catFR1* classes the method *preptrainingdata()* need to be called after loading the data in order to prepare and save the trainingData and trainingLables inside the session

subj1.experiments.<experiment type>.preptrainingdata(session ID , [negative offset , positive offset] , buffer , frequency range , number of wavelet scales , resampling rate , trimming length);

(see sample\_run.m for usage)

The following steps are performed in *preptrainingdata* for FR1 and catFR1:

1. Load the EEG signals
2. Make Bipolar by subtracting channel pairs
3. Extract EEG for word presentation events from -500ms to 1600ms adding 1500ms buffer on both sides
4. Morlet wavelet transform on each event’s EEG channels (wave number=5, 50 log spaced frequencies from 1 to 200Hz)
5. Log transform
6. Resample at 1/10 rate (original sampling rate was 500 Hz -> resampled to 50 Hz)
7. Discard the additional buffer for eliminating edge effect of the convolution
8. Discard an additional 500 ms from both sides as mentioned in “Multivariate Classification” of the SupplementalData document (At this stage the training data is 0-1600ms relative to word event onset).
9. Zscored within frequency band and channel (using mean and std across all events within session)
10. For each channel, took the mean at each frequency, across all time points in the event

**sessionCollection**

This class represents a collection of sessions to be used in training a logistic regression. Sessions can be from different experiments but should have consistent dimensionality and size. This class will be used after finishing all the preprocessing steps, having *session* object ready with their trainingData and trainingLabels.

To use we first create a *sessionCollection* object, then add the desired session objects with *addsessionobject* method:

sc = sessionCollection;

sc.addsessionobject(subj1.experiments.<experiment type>.sessions.<session ID>)

There are two types of training in Ezzyat et. al.: in-session and multi-session

For in-session, call *sc.trainis()* and for multi-session call *sc.trainms()*

A sample usage is provided in *sample\_run.m*