University of Colorado Boulder
Department of Aerospace Engineering Sciences
Boulder, Colorado 80309

Memorandum

To: BrainStim Research group

From: Rachel M. Rise

Date: July 2, 2020

Re: Use of the MATLAB script “Logistic_Regression_Wrapper_Updated.m”

Introduction

The script “Logistic_Regression_Wrapper_Updated.m” is a script written in MATLAB for the purpose of simulating and classifying subjects that exhibit improvement with Stochastic Resonance (SR). It was developed to improve upon previous work and consolidate the efforts of team members. The script uses a logistic regression machine learning paradigm to identify which simulated subjects exhibit SR.

The program is based on a model of SR developed by SR researchers and augmented by the research team. In this model, SR is characterized by a distinct inverted U shape representing task performance as a function of applied SR noise level. Subjects are simulated according to expected performance on a task. In this case, the task is a two-interval forced choice task.

All the code and supporting scripts that I have as of now can be found on my GitLab branch, available as a zipped folder.

This document contains information about how to use the code, as well as locations in the code where edits can be made. Please forgive me if any of the line numbers are wrong

List of Script Sections

The program is broken up into the sections described in Table 1. Items marked with an asterisk (*) denote sections that contain variables or sections that the user may edit. Edits may be made in these sections only, otherwise the code will not run properly. The variables that can be changed are discussed later.

Table 1: Program Sections and Descriptions

Section Name	**Lines**	**Section Description**
BRAINSTIM LOGISTIC REGRESSION	1-10	Title, setup, global variables
BEGIN SETUP		
USER INPUTS: Simulation Setup*	11-45	User preferences for simulation setup.
Task Setup*	46-226	PEST procedure setup, analysis matrix definition
Guesses for curve fits	227-241	Initial guesses for SR curve fitting procedures
Upper and lower bounds on curve fit parameters*	242-279	Constraints on curve fit parameters
Curve fit options*	280-288	Change display options
END SETUP | BEGIN SIMULATION		
Begin simulating subjects*	291-295	Set up and simulate the subject
Simulate the subject	296-332	Continue simulating the subject
Curve fitting	333-383	Fit the characteristic SR curve to the simulated data
Calculate the fitted curve	384-399	Use the curve fit parameters to define the SR curve
Example retest procedures*	400-414	Details how to simulate retest procedures

Plot this subject's information, if desired	415-462	Plots underlying, fitted, and measured data
Concatenate all measurements	463-481	Add this subject's data to matrices for later analysis
Additional information about this subject*	482-504	Calculates values that are used in multiple features
Subject complete! Define features here	505-507	Signify end of simulation and beginning of features
END SIMULATION | BEGIN FEATURE DEFINITION		
[Multiple sections]: Feature definitions*	508-539	Define different features
[USER EDIT]: The features matrix*	540-558	Put the feature matrix together and name the features
END FEATURE DEFINITION | BEGIN ANALYSIS		
Plot feature histograms, if desired	560-594	Plots histograms of feature values each subject
Split data into training and test sets	595-611	Divides the data into training and test sets for ML
Fit model to training set	612-616	Generates the logistic regression model
Predict probabilities for test set	617-620	Classifies SR in test set
Confusion matrices	621-625	Reports script accuracy and probabilities
Chi-squared and P-Values	626-630	Performs Chi-squared test and returns p-values
Correlation Coefficients	631-634	Correlation analysis between features
END ANALYSIS | END PROGRAM		

** Contains user inputs/edits*

As a side note, if anyone knows how to make a 'super-section' in MATLAB that doesn't involve a line of equal signs, please let me know and I'll integrate it.

Supporting Scripts and Functions

The program uses the external and supporting functions described in Table 2. Each of these functions must be copied to the same working directory as the main logistic regression script, in their most up-to-date version that I have supplied.

Table 2: Supporting Functions

Function Name	Description
simulateSubject_RR.m	Simulates subject thresholds given underlying curve and SR noise levels
pest_mod_2int.m	The PEST procedure of trials to simulate a subject's threshold
pest_mod_2int_Audio.m	The PEST procedure for auditory tasks
two_int_fit_simp.m	Simple psychometric curve fit
SR_curve_fit_withmask.m	Curve fitting objective function where masking is expected
SR_curve_fit_nomask.m	Curve fitting objective function where masking is not expected
SR_minWidth_maxDepth_Updated.m	Nonlinear constraints on curve fits
confusionMatrix_Updated.m	Calculates confusion matrix about accuracy
chiSquared.m	Chi squared test on subject data
correlationCoefficients_RR.m	Calculates correlation coefficients between features

All these supporting functions may be found on my branch in the GitLab repository.

List of User Inputs and Edits

All variables and inputs that may be changed are detailed in Table 3. These variables can be broken down into three categories: experimental design (E), subject simulation (S), and features (F).

Variables that deal with experimental design only impact the type of task performed and the accuracy of performance estimation. Examples of variables that deal with experimental design are the SR type applied, the task type, and the number of trials.

Variables that deal with subject simulation impact the underlying performance of the subjects and how they get simulated. Examples of variables that deal with subject simulation are the number of simulations, the subject profile to use (this influences where the SR dip is and represents our ‘traditional’ and ‘alternative’ subjects), and whether to plot the curve fits.

Finally, variables that deal with features influence how the features get used. These variables are basically the number of features, which features are used (both their values and names), and whether they get plotted.

Table 3: Variables the user may edit

Variable Name	**Line**	**Type**	**Description**
numberOfFeatures (F)	14	int	Number of features to examine with the ML algorithm
numberOfSimulations (S)	17	int	Number of subjects to simulate
probabilityOfSR (S)	20	float	Probability that a simulated subject will exhibit SR
subjectProfile (S)	23	int	Simulate subjects with improvement at different SR noise levels
plotSRCurveFits (S)	26	bool	Logical variable to determine whether to plot the curve fits
plotFeatureHistograms (F)	29	bool	Logical variable to determine whether to plot features
SRType (E)	32	int	Determines the type of SR being applied. 1 = ASR, 2 = VSR
TestType (E)	35	int	The type of test simulated. 1 = visual, 2 = auditory, 3 = tactile
maskingExpected (E)	38	bool	Whether masking is expected. Here, expected for ASR auditory
method (S)	42	str	Determines how width of the curve is defined in curve fitting
trainTestRatio (S)	45	float	The proportion of data used to train the ML algorithm
subjectsWithSR (S)	68	bool	Vector that determines which subjects have SR.
options (S)	284	str	Determines the display options for the curve fits
warning (S)	286	str	Change warning settings
thisSubjectSRNoiseLevels (E)	317	float	Determine the SR noise levels for this subject
numberOfTrials (E)	322	int	Determine how many trials are done for the PEST task
features (F)	532	float	Matrix of feature values by subject and feature
featureNames (F)	538	char	Character cell array of feature names for plotting

Subject Simulation

To simulate a subject, the following information must be defined:

- The subject’s underlying SR curve, defined by the parameters A0, λ, ω0, m, s, f, and B. These parameters are currently pre-defined in lines 83-216 and should NOT be changed.
- The SR type and test type. These are pre-defined. Simply modify the values of these variables to suit the test type you are examining.
- The subject profile. There are two possible subject profiles. Profile 1 has a dip at a lower SR level, and profile 2 has a dip at a higher SR level. *Please validate your code with both subject profiles, like we were doing before.*
- The SR noise levels to apply, found in line 317. The ‘default’ noise levels for ASR visual/tactile, ASR auditory, and VSR are in the comments. Uncomment the appropriate one and comment out the others, or define your own. Just make sure to change it when you run each SR/test type.
- The number of trials, in line 322. The function “simulateSubject_RR.m” will accept either a single number of trials and apply them to all the SR noise levels, or a vector of trial numbers that is equivalent in size to the vector of SR noise levels.

- Whether or not the subject has SR

To simulate any set of L measured thresholds for a given subject, simply use the function "simulateSubject_RR.m." This function will work with any set of SR noise levels (i.e. initial sweep of measurements, retest levels, etc). The function takes the following inputs:

- thisSubjectSRNoiseLevels: the 1xL vector of SR noise levels to use
- thisSubjectUnderlyingCurveParameters: 1x7 vector containing A0, λ, ω0, m, s, f, and B that defines the subject's underlying curve
- numberOfTrials: either a single integer or 1xL vector of the number of trials to use
- subjectHasSR: logical value indicating whether the subject has SR

The function "simulateSubject_RR.m" returns two 1xL vectors:

- thisSubjectUnderlyingThresholds: the actual underlying thresholds exhibited by this subject at the query SR noise levels. *Note: the underlying thresholds are NOT to be used as inputs to features. We cannot know the underlying thresholds in real life, so our ML algorithm cannot use them.*
- thisSubjectMeasuredThresholds: the measured thresholds produced by the PEST procedure and psychometric curve fitting. *Note: ONLY use these measured thresholds when calculating feature values.*

From the measured thresholds, features may be defined. The measured thresholds are also used to perform the SR curve fitting.

Curve Fits

The curve fits should run automatically. The variable plotSRCurveFits controls whether the curve fits get plotted. If the plots are on, only the first 10 subjects will be plotted. If masking is expected, the program will use the equation that incorporates masking. The curve fits take the following input arguments:

- SR_curve_fit_withmask(curveFitParameters,thisSubjectSRNoiseLevels,thisSubjectMeasuredThresholds)
- curveFitGuesses (previously defined)
- 4 empty braces for linear constraints ([],[],[],[])
- LowerBounds (previously defined)
- UpperBounds(previously defined)
- SR_minWidth_maxDepth nonlinear constraint function
- Options

The curve fit outputs the following:

- 1x5 (no masking) or 1x7 (masking) vector of fitted curve parameters A0, λ, w0, m, s, f, and B. This is automatically reformatted for use in plotting, storage, and analysis.
- Cost function value J
- Optimization function exit flag

The raw parameters are used to calculate two new vectors, X and SRCurveFit, that define the curve over 1000 data points.

Feature Definition

To define a feature, use the following process:

1) Create a new section near the other features in the program
2) Calculate the feature's value for *this subject only, using only the measured information available about this subject. Do not index into variables like allMeasuredThresholds, simply use* thisSubjectMeasuredThresholds *or similar*. If there are a lot of interim steps, you are more than welcome to write an external function that calculates the feature's value.
3) Please name the feature's variable in UpperCamelCase (i.e. ThresholdsAboveSham)
4) Update the features matrix
5) Update the feature names cell array so you can make pretty histograms!
6) Update the number of features in line 14

Presentation Expectations

This week, we expect to see:

- A plot of at least 1 working characteristic SR curve fit (doesn't have to be a 'good' fit, we just want to make sure it works)
- A plot of the histogram associated with each feature you define to demonstrate its impact
- Coefficient analysis using beta and p-values

Conclusion

I hope this document has been informative and helps you understand this new architecture, and that the new code is much easier to work with than our previous version. I did my best to make it easy to understand, well-commented, and relatively effort-free. I expect that you do most of your debugging yourself, but please let me know if you run into any weird edge cases or head-scratching errors that you can't figure out after a solid debug attempt.

Happy 4th!

Rachel

[illegible]