Analyzing Learning in Behavioral Experiments

Prepared by Léo M. Walton, The Neuroscience Statistics Research Laboratory,
Massachusetts General Hospital
June 2007

Introduction:

This tutorial is meant to guide users through the proper use of the learning curve analysis software. This Matlab-based software allows researchers to quantitatively characterize subjects' learning through the analysis of behavioral responses.

Important Resources and References:

Neuroscience Statistics Research Laboratory (Neurostat): Emery Brown – Massachusetts General Hospital https://neurostat.mgh.harvard.edu/

Software for Bayesian analysis of learning: Anne Smith – Univ. of California – Davis http://www.ucdmc.ucdavis.edu/anesthesiology/research/bayes/index.html

Smith AC, Frank LM, Wirth S, Yanike M, Hu D, Kubota Y, Graybiel AM, Suzuki WA, Brown EN. Dynamic analysis of learning in behavioral experiments. Journal of Neuroscience 2004; 24(2):447-461.

Smith AC, Stefani MR, Moghaddam B, Brown EN. Analysis and design of behavioral experiments to characterize population learning. J.Neurophysiology (published online Sept. 29, 2004), 2005, 93:1776-1792.

Smith AC, Wirth A, Suzuki W, Brown EN. Bayesian analysis of interleaved learning and response bias in behavioral experiments. Journal of Neurophysiology, 2007, 97(3): 2516-2524.

Quick Overview of Programs:

The software estimates the learning curve and learning trial of a single subject or a population of subjects by analyzing behavioral data obtained in a learning experiment. The learning curve characterizes the dynamics of the learning process as a function of trial number and is plotted along with its upper and lower 95% confidence bounds. The learning trial is defined as the first trial on which there is reasonable certainty (0.95) that for the remainder of the experiment the subject performs better than chance.

For more information about these analyses, refer to the resource material.

Gibbs Sampling VS Expectation Maximization: Which Method to use?

In general, it is best to use the Gibbs Sampling method (computationally efficient and small MISE), except in the case of long datasets. Use the Expectation Maximization software for long (> 100 trials) datasets.

Table 1. MISE for the EM algorithm and the Gibbs sampling algorithm in estimating two simulated learning curves.

		Mean Integrated Square Error (MISE)	
		Delay rapid learning	Early rapid learning
Methods:	Gibbs Sampling (winBUGS)	0.4779	0.2074
	EM (smoothing)	0.5025	0.1445

For both learning curves, the MISE was computed for each method using 100 simulated learning experiments. The number of trials per experiment was 50.

Running the Gibbs Sampling (WinBUGS) algorithms:

Before the Gibbs Sampling algorithm can be run:

1. Install Winbugs from http://www.mrc-bsu.cam.ac.uk/bugs/

2. Download "matbugs.m" from
   http://www.cs.ubc.ca/~murphyk/Software/MATBUGS/matbugs.html

Analysis of a single subject:

To run the learning analysis on an individual subject, use the function *runanalysis.m* within the *IndividualAnalysisWinBUGS* folder.

To run the example, type:

>> runexample

To run your own data, call the function *runanalysis(***Responses, MaxResponses, BackgroundProb***),* where the input parameters are defined as follows:

Responses Vector of the observed response at each trial (experimental data). This data can be in binary or proportion form. *required

MaxResponses Total number that could be correct at each trial. Should be a vector of equal length to the Responses (vector of 1's for binary data). *required

BackgroundProb Probability of correct response by chance**.** The user is expected to have some knowledge about the background probability of the experiment. *required

Display the results:

To display the results of the analysis, call the function *plotresults.*

Analysis of a population of subjects:

To run the learning analysis on a population of subjects, use the function *runanalysis.m* within the *PopulationAnalysisWinBUGS* folder.

To run the example, type:

>> runexample

To run your own data, call the function *runanalysis(***Responses, BackgroundProb***)*, where the input parameters are defined as follows:

Responses Set of vectors (a matrix), each containing a subject's observed response at each trial (experimental data). This data must be in binary form (no population analysis is available for proportion data at this time). *required

BackgroundProb Probability of correct response by chance**.** The user is expected to have some knowledge about the background probability of the experiment. *required

Display the results:

To display the results of the analysis, call the function *plotresults.*

Running the Expectation Maximization (EM) algorithms:

Analysis of a single subject:

To run the learning analysis on an individual subject, use the function *runanalysis.m* within the *IndividualAnalysisEM* folder.

To run the binary data example, type:

>> runbinaryexample

To run the proportion data example, type:

>> runproportionexample

To run your own data, call the function *runanalysis(***Responses, MaxResponse, BackgroundProb, SigE, UpdaterFlag***),* where the input parameters are defined as follows:

Responses Vector of the observed response at each trial (experimental data). This data can be in binary or proportion form. *required

MaxResponse Total number that could be correct at each trial (1 for binary data). Should be the same length as **Responses**. *required

BackgroundProb Probability of correct response by chance**.** The user is expected to have some knowledge about the background probability of the experiment.*required

SigE User's estimate of the square root of the state space model's (learning process's) variance. *optional

UpdaterFlag The manner in which the initial conditions are estimated. *optional
User can choose to

- fix at the chance line (*UpdaterFlag* = 0)
- estimate initial conditions (*UpdaterFlag* = 1) assuming initial probability of a correct response is close to chance
- estimate initial conditions (*UpdaterFlag* = 2) assuming initial probability of a correct response is completely estimated

Display the results:
To display the results of the analysis, call the function *plotresults.*

Functions for further analysis:

trialtotrial - Display a surface image depicting whether the probability at trial i is different from the probability at trial j, for i, j = 1,...,N. N is the total number of trials.

findj(p, n , pcrit) - Finds the minimum number of ones in a row you need to say with certainty p<pcrit that learning has occurred for n binary trials.

Analysis of a population of subjects:
To run the learning analysis on a population of subjects, use the function *runanalysis.m* within the *PopulationAnalysisEM* folder.

To run the example, type:

>> runexample

To run your own data, call the function *runanalysis(***Responses, BackgroundProb, SigE, SigB***),* where the input parameters are defined as follows:

Responses Set of vectors, each containing a subject's observed response at each trial (experimental data). This data must be in binary form (no population analysis is available for proportion data at this time). *required

BackgroundProb Probability of correct response by chance**.** The user is expected to have some knowledge about the background probability of the experiment. *required

SigE User's estimate of the square root of the state space model's (learning process's) variance. *optional

SigB User's estimate of the square root of the state space random effects model. *optional

Display the results:

To display the results of the analysis, call the function *plotresults*.

Troubleshooting

If the EM fails to converge it may be due to:

1. poor starting value - change parameter SigE
2. too stringent convergence criterion – increase parameter CvgceCrit and check that newsigsq has converged by plot(newsigsq)
3. there is not enough data/information to estimate the hidden process and there is likely to be no learning occurring
4. need to increase number of EM iterations by increasing NumberSteps
5. Assign UpdaterFlag = 0 so that the initial condition is fixed converges in more cases than when UpdaterFlag =1

Bugs and questions to: annesmith@ucdavis.edu