General information

Author: Scott Albert
Email: salbert8@jhu.edu
Institution: Johns Hopkins University
Lab: Laboratory for Computational Motor Control
Advisor: Reza Shadmehr
Location: Baltimore, MD 21211
Date: January 18th, 2017
Version: 1.1

Summary

This package contains MATLAB files that use Expectation Maximization (EM) to fit a two state model of motor learning to behavioral data. EM is a statistical algorithm used for parameter estimation. It uses a Kalman Filter and Maximum Likelihood Estimation to perform system identification. It identifies a parameter set that maximizes the likelihood function for a system that has noise both in the production of a movement and the planning of a movement. A less realistic model of learning, one with noise only in the production of a movement, can be fit using a least-mean-squared-error (LMSE) algorithm.

More information concerning EM, LMSE, and two state models of learning is available in:

> Albert ST, Shadmehr R (2017) Estimating properties of the fast and slow adaptive processes during sensorimotor adaptation. *J Neurophsyiol*.

The article is provide for your convenience in this package.

Version

This version fits data using EM. This version is compatible with paradigms that include/exclude error-clamp trials but not set breaks. To model paradigms that contain set breaks, use Version 1.2.

Contents

This package includes the following files:

generalized_expectation_maximization.m
This function coordinates the EM algorithm. It links the E- and M-steps of the algorithm.

kalman_smoother.m
This function performs Kalman smoothing used in the E-step of the EM algorithm.

m_step.m
This function coordinates the M-step of the EM algorithm. It calls *fmincon* to maximize the expected complete log-likelihood function in a constrained parameter space. This likelihood can be computed by either a MATLAB m-file, or a mex function as described below.

expected_complete_log_likelihood.m
This function is called during the M-step to compute the expected complete log-likelihood function. It is the objective function used by *fmincon*. This m-file is also implemented as an executable mex function, described below. This package has the option to use the m-file or the mex-file. See the section entitled MEX Function, below, for more information.

expected_complete_log_likelihood_mex.cpp
This function is called during the M-step to compute the expected complete log-likelihood function. It is the objective function used by *fmincon*. It is the analogue of *expected_complete_log_likelihood.m*. It is a C++ file that must be compiled. See the section entitled MEX Function, below, for more information.

incomplete_log_likelihood.m
This function computes the incomplete log-likelihood function. We monitor increases in this function across EM iterations to determine if the algorithm has converged.

tutorial.m
This file is a MATLAB script that illustrates how to fit a two state model to a data set using EM. It is intended to provide an overview of how to use this package. The script simulates data according to a two state model. Next, this script calls functions that coordinate the EM algorithm to fit the two state model to the simulated data. Finally, the script provides some output of the EM results.

two_state_simulation_with_noise.m
This function is referenced in *tutorial.m*. It simulates a two state model with noise.

two_state_simulation_without_noise.m
This function is referenced in *tutorial.m*. It simulates a two state model without noise.

Albert_Shadmehr_JNP_2018.pdf
This is a copy of the manuscript that describes the EM algorithm and compares it to LMSE for the fitting of behavioral data with a two-state model. All model details and technical details about EM and LMSE can be found here.

MEX Function
Our EM algorithm uses a numerical search for the M-step. This search is coordinated by the MATLAB function *fmincon*. In an attempt to speed the performance of the M-step, we implemented a mex function that computes the expected complete log-likelihood function. The name of this function is *expected_complete_log_likelihood_mex.cpp*. Its output is identical to the m-file alternative *expected_complete_log_likelihood.m*.

Using a mex function to compute the likelihood used by fmincon will dramatically speed the M-step of the algorithm. Therefore, using the mex function implementation over the m-file implementation is highly recommended. We provided the m-file so that a general user can easily modify, read, and understand the code should they want to make edits or changes.

There is an input option to *generalized_expectation_maximization.m* which determines if the M-step should use the m-file version, or the mex function.

To run the mex function, you must compile it first.

From the MATLAB Command Window, to build the executable enter the following:
mex *expected_complete_log_likelihood_mex.cpp*

To do this, MATLAB must have access to a C/C++ compiler on your machine. We have hard-coded essentially all of the mathematical operations into the C++ code, so that as long as you have a C++ compiler, you should not need to link to special libraries.

Getting started

To use EM for your data, the best way to get started and familiar with this MATLAB software is to delve into *tutorial.m*. The comments within this script should hopefully be sufficient to show the general user how to interface with the EM software.

For your data, you would likely take the following steps to use the EM algorithm. Use the *tutorial.m* script as a template and change the following components.

1. Change the perturbation schedule to match your paradigm (r parameter)
2. Change the error-clamp schedule to match your paradigm (EC parameter)
3. Specify the values of the clamped errors to match your paradigm (EC_value parameter)
4. Change the set break schedule to match your paradigm (SB parameters)
5. Replace the simulated data with your own behavioral data (y parameter)
6. Specify the number of EM iterations you want to perform (num_iterations parameter)
7. Specify if a mex function is going to be used (use_mex parameter)
   Note: to use the mex function, compile it first according to the previous section.
8. Modify the parameter boundaries for the EM fit (search_space parameter)
9. Modify the constraints relating the two-state model parameters (constraints parameter)
10. Specify an initial parameter guess for the algorithm (IC_EM parameter)