FittingWorkflow

September 12, 2017

A pipeline for Bayesian inference on seizure propagation seen in sEEG data

In this notebook report, we present a systematic workflow in order to fit brain model parameters (in this study, a 2D reduction of coupled Epileptors) to the functional data such as stereotactic EEG (sEEG).

To achieve this goal, we need to define:

a) the brain network model including the parameters of interest i.e., the generative model.

b) the fitting target such as the sEEG time series i.e, the observation.

c) Bayesian inference framework to fit the brain model against the patient's empirical data.

0.1 Bayesian inference:

Bayesian approach offers a framework to deal with parameters and model uncertainty. It offers much more than a single best fit obtained within Frequentist approach.

In current study, our aim is to obtain the estimates of the excitability parameters of the brain model defined by network of reduced Epileptor model (Proix et al., 2014), given the individualized patient's empirical sEEG, within a Bayesian framework.

For the current perpose, we have a large set of emperical data, and a complex mathematical model compressing neural mass models on each network node connected via patient's connectome. Consequently, we need a complex inference framework to achieve reliable estimates. This task arises this critical technical question that how we can tackle down such a complicated challenge in parameter estimation problems (known as inverse problem)? In other words, how we can implement a Parameter Estimation Framework in order to fit large-scale brain network models against signals from the imaging modality such as sEEG? Which programming software and which algorithm are required to address this challenge? The answer is Probabilistic Programming (automated inference). We first define the generative model as well as the fitting target, and then we run a Bayesian algorithm to estimate the parameters of interest. It is important to note that deriving Bayesian inference algorithms requires tedious model-specific calculations: choosing a proper setup for the algorithm, tune its parameter values to improve the algorithm efficiency, initializing, computational efficiency of the code, and more importantly a generic framework which allows us to automatically fit personalized brain network models derived from non-invasive structural data of individual patients.

0.1.1 Algorithms for Bayesian inference:

According to our investigations for some case studies, the following recently developed algorithms are able to solve complex inference probability problems: - Hamiltonian Mote Calro (HMC) - Variational Inference (VI).

HMC takes advantage of gradient information from the likelihood to achieve much faster convergence than traditional sampling methods (such as Metropolis and Gibbs sampling), especially for larger data set. The performance of HMC depends strongly on choosing suitable values for: step size and total integration time. A poor choice of either of these parameters will result in a dramatic drop in HMC's efficiency. To remedy this shortcoming, a recently developed algorithm known as No-U-Turn Sampler (NUTS) have been used (Hoffman and Gelman 2014). This algorithm is an extension to HMC, with no hand-tuning at all!

The other way to estimate parameters within Bayesian framework is Variational Inference (VI). VI approximates the posterior with a simpler distribution. We posit a family of distributions and find the member closest to the posterior (the member of the family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior). This turns the task of computing a posterior into an optimization problem.

Traditionally, using a variational inference algorithm requires the painstaking work of: developing and implementing a custom optimization routine, specifying a variational family appropriate to the model, computing the corresponding objective function, taking derivatives, and running a gradient-based or coordinate-ascent optimization. Atomatic Differentiation Variational Inference (ADVI) solves these problems systematically. The user specifies the model, expressed as a program code, and ADVI automatically generates a corresponding variational algorithm. Note that ADVI supports a broad class of models, and enables fast inference with a large data set compared to HMC.

0.1.2 Softwares for Bayesian inference

Both Bayesian algorithms described above are provided in Satn (Open-source framework for Bayesian inference) without the hand-tuning the algorithm's parametrs for different case studies. Stan resolves the computational bottleneck of the probabilistic modeling cycle. This software is a flexible probabilistic programming system which allows of Bayesian statistical models in code (Probabilistic Programming). Stan describes a high-level language to define probabilistic models as well as a model compiler, a library of transformations, and an efficient automatic differentiation toolbox. Stan uses two step complication process: the user writes a model in pure Stan code which is then translated to C++ by Stanc compiler.

Note that there exist many other Probabilistic Programming (PP) allowing flexible specification of Bayesian statistical models in code- PYMC, PYMC3, Edward, WinBUGS to name a few. For instance, PyMC3 is a new open source probabilistic programming framework written in Python that uses Theano to compute gradients via automatic differentiation (https://pymc-devs.github.io/pymc3/notebooks/getting_started.html). Contrary to other probabilistic programming languages, PyMC3 allows model specification directly in Python code. Although there is "No free lunch" and choosing one of the Bayesian Software comes with the trade-off between model coding language, the level of documentation and the provided libraries, Stan is the major open-source framework for Bayesian inference within statisticians community. For a more detailed discussion on tradeoff between mentioned softwares please see http://andrewgelman.com/2015/10/15/whats-the-one-thing-you-have-to-know-about-pystan-and-pymc-click-here-to-find-out/.

According to our experiences, we have chosen Stan as an Open-source framework to implement NUTS (an extension to HMC) and ADVI (automatic variational inference) Bayesian inference. Certainly, one can use PYMC3 or the other Probabilistic Programming Softwares though we recommend use of Stan due its several advantages against the other alternatives.

Finally, it is worth pointing out that in none of the existed Probabilistic Programming Softwares within Bayesian framework, no solver was provided into the code to integrate the stochastic differential equations. Thus, we have to solve and integrate the equations ourselves (using Euler-Maruyama method) and then we are able to call the VI or HMC algorithm to construct the posterior distributions. Only there exists solver for detereminsitic diferential equations in Stan. This more motivated us to use Stan in our work with the aim to solve deterministic version of the model using in addition to the stochastic equations. In the following, more details on Stan software is presented.

0.1.3 Bayesian inference in Stan:

Stan® is freedom-respecting, open-source software for facilitating statistical inference, statistical modeling and high-performance statistical computation at the frontiers of applied statistics.

Stan implements both Hamiltonian Monte-Carlo and automatic variational inference algorithms for generic differential probability models (The Stan Development Team, 2015).

Using Stan within different interfaces such as Python and shell, runing on all major platforms (Linux, Mac, Windows), we are able to perform:

- Bayesian statistical inference with the gradient-based MCMC sampling known as Hamiltonian Monte Carlo (HMC): HMC takes advantage of gradient information to achieve much faster convergence than traditional sampling methods, and its trajectories do not resemble random-walks.

- approximate Bayesian inference using automatic differentiation variational inference (ADVI): an automatic method that derives variational inference algorithms for complex probabilistic models. ADVI searches over a family of simple distributions and find the member closest to the posterior. Compared to HMC, variational inference tends to be faster and easier to scale to large data.

For more details on the algorithms and Stan interfaces, see Stan homepage: mc-stan.org

In this work, we use the python inference (PyStan) and we run the code in shell (cmdstan). For this purpose, we need to install PyStan (in Python) and CmdStan (shell, command-line terminal).

To install PyStan, please see:
https://pystan.readthedocs.io/en/latest/getting_started.html
To install CmdStan, unpack zip file CmdStan 2.16.0 from:
https://github.com/stan-dev
and consult the user's guide (in directory doc/) for installation and getting started instructions.

0.2 Seizure propagation model:

Several studies have linked the interpretation of neuroimaging signals to computational brain models; the use of connectivity derived from Diffusion-weighted MRI (dMRI) to constrain large-scale brain network models.

VEP is based on personalized brain network models derived from non-invasive structural data of individual patients (Jirsa et al., NeuroImage 2016). Using patient-specific connectomes in large-scale brain networks as generative models of neuroimaging signals, we fit and validate the brain model against the patient's empirical stereotactic EEG (SEEG).

The approach to build the VEP brain model comprises :

- Structural network modeling (reconstruction of the patient's individual brain network topography)
- Functional network modeling (Epileptor are defined on each network node)
- Hypothesis formulation (first hypotheses of the location of the Epileptogenic Zone)
- Evaluation of the VEP brain model (simulation, data fitting and mathematical analysis).

After the preprosseing of the sEEG signal which is presented in this notebook, we fit the neural mass model of 2D reduction of coupled Epileptors, network connectivity & sEEG forward solution. The network dynamics follow Proix et al, Eq (2-6),

$$\dot{x}_{1,i} = 1 - f_1(x_{1,i}) - z_i + I_{1,i} \tag{1}$$

$$\tau_0 \dot{z}_i = h(x_{1,i}) - z_i - k \sum_{j=1}^{N} K_{ij}(x_{1,j} - x_{1,i}) \tag{2}$$

$$\tag{3}$$

where their evolution follows a Euler-Maruyama discretization of the corresponding stochastic differential equations from time t to $t + dt$, with linear additive normally distributed noise.

Constants and auxiliary functions are taken from Proix et al. 2014, and the work flow pursueS the paper Jirsa et al. NeuroImage 2016. The target for all data fitting is the excitability parameter x0. Obtaining such estimates of the parameters of the network model, given the available functional data is performed within a Bayesian framework, using a reduced Epileptor model and reduced functional data set for the fitting.

0.3 Required empirical data:

To fit the brain model described above to processed sEEG signals i.e., to provide posterior distributions of excitability parameters on each node, we need the following data: - Seizure time series (complex.npy) - Structural connectivity matrix (weights.txt) - Region centers (centers.txt) - xyz position of electordes and their contacts (elecs_name.txt) - ades file containing sampling rate, number of samples, and the contact labesl e.g., A'1, A'2, . . . , TP'1 (complex.ades)

In the following, we load and preprocess sEEG data, sensors and connectivity, then build a dynamical model (Reduced Epileptor model following Proix et al, 2014) linking the data, and use Hamiltonian Monte Carlo and variational inference techniques as implemented by the Stan software.

Briefly, using the work flow presented here, we can obtain the posterior estimates of excitability parameter x0 of the described large-scale brain network model from sEEG signals. This work flow generally can be applied to fit brain model parameters to the functional data.