Software Engineering Assignment Report

Ajayrama Kumaraswamy (ajkumaraswamy@tutamail.com) May 13, 2021

Introduction

Recent advances in Computational Psychiatry allows the quantification of deficits in goal-directed control, where subjects tend to form rigid habits instead of flexibly adapting their decision-making towards intended goals. This deficit has been associated with obsessive-compulsive disorder and addiction. Daw et al. (2011) illustrated the quantification of goal-directed control using parameter fits of a reinforcement learning model to data from a novel multi-step decision task. Gillan et al. (2016) used this quantification paradigm to find an association between deficits in goal-directed control and a psychiatric symptom dimension comprising compulsive behavior and intrusive thought.

The goal of the current assignment was to implement the model used by Gillan et al. (2016) and estimate model parameters based on data provided. This model has also been used in Otto et al. (2013) and Huys et al. (2011), both of which provide more implementational details. Both studies use a two-level Bayesian Hierarchical model, where the data of each subject is modeled using different sets of parameters, which are in turn drawn from group-level distributions. Huys et al. (2011) estimate group-level distributions by pooling data across subjects, which are then used as priors for Maximum-A-Posteriori (MAP) estimation of individual parameters using an Expectation-Maximization (EM) approach. Otto et al. (2013) use a Markov Chain Monte Carlo (MCMC) method based on No-U-Turn Sampler (NUTS) to estimate posterior distributions of model parameters.

Methods

Python has a number of frameworks for building Bayesian Hierarchical models. PyMC3 (Salvatier et al., 2016) was used for this assignment as it has excellent documentation and is rich in community resources like blogs and forums. PyMC3 provides both MAP as well as NUTS sampling methods for parameter estimation. While MAP method is faster, it could converge to a local maximum. While MAP only provides a point estimate of model parameters, sampling methods provide estimations of parameter distributions, which could, for example, be used for quantifying their uncertainty. Both estimation methods were used in this assignment.

Model hyperparameters and priors were taken from Otto et al. (2013) (see supplementary information), except for α, which was parameterized as in Equation-group1. An exponential distribution is used for the shape parameter σ_α for its lighter tails, which can lead to better sampling. This was suggested by a tutorial of PyMC3 (The PyMC Development Team, 2021).

$$\alpha \sim \text{Beta}\{A, B\} \text{ where} \tag{1a}$$
$$A = \mu_\alpha \sigma_\alpha \text{ and } B = (1 - \mu_\alpha)\sigma_\alpha \text{ where} \tag{1b}$$
$$\mu_\alpha \sim \text{Uniform}\{0, 1\} \text{ and } \sigma_\alpha = \exp\{\sigma_{\alpha\_log}\}, \text{ where} \tag{1c}$$
$$\sigma_{\alpha\_log} \sim \text{Exponential}\{1.5\} \tag{1d}$$

Models were fit on a laptop with a 4-core Intel i5 CPU and 16GiB of memory with three configurations as shown in TableS1.

Results

Number of trials strongly affected memory demand

Parameter estimations using data from more than 25 trials failed as they ran out of memory even though data from only two subjects were used (data not shown). However, parameter estimation using data from 25 trials and 15 subjects was successful. This strongly indicated that the number of trials used had a stronger effect on memory consumption than the number of subjects used.

Runtimes and convergence

Data from a larger number of trials and a larger number of subjects lead to longer run times for both MAP and sampling estimations (see TableS1). For sampling estimations, although drawing a larger number of samples led to much higher estimation run times, they achieved better convergence as quantified by the Gelman-Rubin diagnostic (see TableS2-4).

MAP vs Sampling Estimates

MAP estimates were closer to initial estimates than NUTS sampling estimates (see TableS4). Choosing better initial estimates could result in better MAP estimates.

Further Work

The model implementation presented can be deployed on a powerful computer with 128 or 256GB of RAM to estimate parameter distributions for all 250 subjects of the dataset and using all 200 trials. PyMC3 has an experimental yet highly promising implementation where both the model and the sampler are Just-in-time (JIT) compiled. This implementation can use available GPUs and TPUs as well and can lead to much faster parameter estimation (Wiecki, 2020).

An Interesting Parallel

A machine learning algorithm called Q-learning (Karunakaran, 2020) is based on the reinforcement learning model used in this assignment. Given an environment that provides stochastic rewards to actions, an artificially intelligent agent can use Q-learning to select an optimal decision-making policy that maximizes the expected reward. Implementations of Q-learning using deep neural networks exist (Wang, 2020). It might be possible to use a two-level Q-learning algorithm to select the optimal decision-making policy for the task used in Gillan et al. (2016) and compare it with the policies used by test subjects.

Another machine learning approach related to the reinforcement learning model used in this assignment is the problem of hyperparameter selection for offline reinforcement learning (Paine et al., 2020). Given a dataset of actions performed by an agent and associated rewards provided by an environment, this approach aims to first train a set of policies using different sets of hyperparameters using only the dataset provided and not interacting further with the environment, and then select the best among these policies. This approach is similar to the model-fitting approach of this assignment, where the task is to identify the best set of hyperparameters with which the reinforcement learning model produces the set of actions selected by test subjects given stimuli and rewards. The arguments and discussions in Paine

et al. (2020) can help in evaluating the model fits of the reinforcement learning model of this assignment as well as draw parallels and device new methods for parameter fitting.

Supplementary Files

All supplementary files are available on GitHub at `https://github.com/ajkswamy/gillan-model-assignment-report.git`.

Code

Python code used in this assignment is available in the folder 'code' of the repository. Usage instructions are available in the file `Readme.md`.

Tables

Please note that the CSV files mentioned below are best visualized in a spreadsheet program such as Microsoft Excel or LibreOffice Calc. Sorting rows based on different columns can help compare estimates of a parameter across subjects or estimates of all parameters for a subject.

1. **TableS1:** Details of the configurations used for model fitting. See `test_configurations_runtimes.csv` in the folder 'Results'.
2. **TableS2:** Summary of estimation results using configuration 1. See `summary_multilevel_2subjects_20trials.csv` in the folder 'Results'.
3. **TableS3:** Summary of estimation results using configuration 2. See `summary_multilevel_5subjects_20trials.csv` in the folder 'Results'.
4. **TableS4:** Summary of estimation results using configuration 3. See `summary_multilevel_15subjects_25trials.csv` in the folder 'Results'.

References

Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., & Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, *69*(6), 1204–1215. URL `https://doi.org/10.1016/j.neuron.2011.02.027`

Gillan, C. M., Kosinski, M., Whelan, R., Phelps, E. A., & Daw, N. D. (2016). Characterizing a psychiatric symptom dimension related to deficits in goal-directed control. *eLife*, *5*. URL `https://doi.org/10.7554/elife.11305`

Huys, Q. J. M., Cools, R., Gölzer, M., Friedel, E., Heinz, A., Dolan, R. J., & Dayan, P. (2011). Disentangling the roles of approach, activation and valence in instrumental and pavlovian responding. *PLoS Computational Biology*, *7*(4), e1002028. URL `https://doi.org/10.1371/journal.pcbi.1002028`

Karunakaran, D. (2020). Q-learning: a value-based reinforcement learning algorithm. URL `https://medium.com/intro-to-artificial-intelligence/q-learning-a-value-based-reinforcement-learning-algorithm-272706d835cf`

Otto, A. R., Raio, C. M., Chiang, A., Phelps, E. A., & Daw, N. D. (2013). Working-memory capacity protects model-based learning from stress. *Proceedings of the National Academy of Sciences*, *110*(52), 20941–20946.
URL https://doi.org/10.1073/pnas.1312011110

Paine, T. L., Paduraru, C., Michi, A., Gülçehre, Ç., Zolna, K., Novikov, A., Wang, Z., & de Freitas, N. (2020). Hyperparameter selection for offline reinforcement learning. *CoRR*, *abs/2007.09055*.
URL https://arxiv.org/abs/2007.09055

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, *2*, e55.
URL https://doi.org/10.7717/peerj-cs.55

The PyMC Development Team (2021). Hierarchical partial pooling.
URL https://docs.pymc.io/pymc-examples/examples/case_studies/hierarchical_partial_pooling.html

Wang, M. (2020). Deep q-learning tutorial: mindqn.
URL https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc

Wiecki, T. (2020). Using jax for fasting sampling.
URL https://docs.pymc.io/notebooks/GLM-hierarchical-jax.html