Statistical Pupillometry

A detailed account of the statistical machinery used in mapping pupil dilation/contraction to value-based decision making during reinforcement learning.

Zach Wolpe

01 February 2021

Imagine if we could deconvolve the inner workings of human cognition & physiology by utilizing simple, ubiquitous, non-invasive, technology - relying only on intelligent algorithms. This article provides an in depth analysis of the application of statistical learning to study value approximation during (human) reinforcement learning. This article explains the statistical modelling framework - Bayesian hierarchical Reinforcement Learning - leveraged in the Slooten. et al. (2018) paper titled, How pupil responses track value-based decision-making during and after reinforcement learning.

Objective

Pupillometry provides a non-invasive proxy of neuro-modulation and some associated cognition expenditure. Further, it's relationship to decision making has been studied at depth: showing pupils dilate during uncertainty and after the occurrence of unexpected events. Pupil dilation has also been shown to correlate with surprise during random positive reward tasks (gambling).

Whilst a host of studies detail the neurological activity associated with pupil size, pupil activity interaction with other cognitive processes such as **attention and cognitive effort** remains ambiguous. In the Slooten, et al. paper, pupil size fluctuations during value-based learning and decision-making were investigated, using a computational Reinforcement Learning model **to identify the specific influences of value-related computations on pupil size.**

Target

The goal of the study was *to investigate which cognitive processes reflected by one's pupils during value-based decision-making*. The authors wished to examine how pupil responses are indicative of certain underlying cognitive processes, namely:

- One's tendency to explore vs exploit information.
- One's ability to learn, the rate at which value beliefs are updated.

Methodology

To achieve this the authors used a binary reinforcement learning task to study pupil responses during **value-based decisions** and subsequent **decision evaluations**.

Notable findings

They found that the pupil closely tracks reinforcement learning processes independently across trail-by-trail fluctuations in value beliefs.

- Pupil dilation predict an individual's tendency to exploit (vs explore) high value options.

- After feedback, biphasic pupil responses were observed - the amplitude of which correlated with participants' learning rates.
- Early feedback related dilation scaled with value uncertainty.
- Later pupil constriction scaled with reward prediction errors (RPE).

These findings show that pupil size fluctuations can provide detailed information about the computations underlying valued-based decisions and the subsequent updating of value beliefs.

Practical implications

As these processes are affected in a host of psychiatric disorders, the results indicate that pupillometry can be used as an accessible tool to non-invasively study the processes underlying ongoing reinforcement learning, in the clinic or elsewhere.

Define the Game

The pupil size of 34 participants was measured whilst inidividuals performed a probabilistic RL task, consisting of:

- Varying reliability of choice outcomes: $\{AB : 80 : 20; CD : 70 : 30; EF : 60 : 40\}$.
- Varying reward probabilities creating varying degrees of:
  - choice difficulty.
  - uncertainty.
  - value expectations.
- Separate learning & transfer phases.
  - Learning phase: allows one to measure how value beliefs are updated during learning.
  - Transfer phase: allows one to measure how choices were guided by previously acquired reinforcement values.

Bayesian Hierarchical Q-Learning RL Model

The model employed attempts to mimic the human behaviour - and use the data to train parameters, allowing one to reason about human cognition.

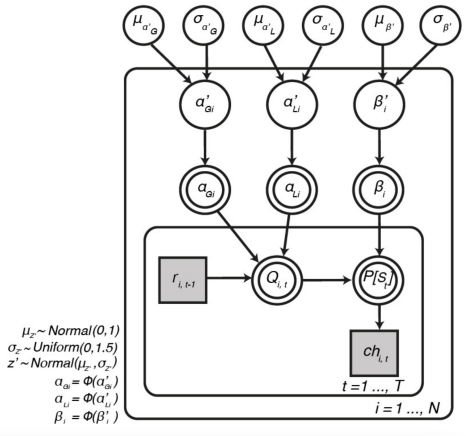

Figure 1: Graphical modeling: Bayesian Hierarchical Reinforcement Learning

The model is described in the graphical model presented above 1. It's interpretation reads:

- There are two levels to the hierarchy ($i = 1, ..., N$ $participants$ and $t = 1, ..., T$ time periods).
- Bayesian priors reside outside of any hierarchy, as they inform the model independently of this structure.
- Circular nodes: continuous variables
- Square nodes: discrete variables
- shaded nodes: observed variables
- double border nodes: deterministic variables

Further, the model parameters are interpretable as:

$$
\begin{aligned}
Priors &: \{\mu_{\alpha`G}, \sigma_{\alpha`G}, \mu_{\alpha`L}, \sigma_{\alpha`L}, \mu_{\beta`L}, \sigma_{\beta`L}\} \\
\alpha_i &: learning\ rate: \ Value\ belief\ updating\ (per\ participant\ i) \\
\alpha_{Gi} &: Gain\ (value\ update) \\
\alpha_{Li} &: Loss\ (value\ update) \\
&\quad higher\ learning\ rate = more\ rapid\ (and\ volatile)\ belief\ updating \\
\beta_i &: sensitivity\ to\ option\ value\ differences: \ \Delta Q - Value\ (per\ participant). \\
&\quad higher \beta = more\ exploitatory\ decisions\ for\ option\ with\ higher\ expected\ reward, \\
&\quad greater\ sensitivity\ to \Delta Q - Value.
\end{aligned}
\tag{1}
$$

Model interpretation

These parameters allow one to reason about how *inter-individual* differences in *value-based learning & decision-making* relate to pupil responses. Secondly, by simulating the learning process we could investigate how pupil size depends on trial-to-trial fluctuations in underlying computational variables such as: $\{$*value beliefs, uncertainty & RPE (Reward Prediction Error)*$\}$. Put otherwise, the experiment allows us to *map pupil responses onto separate computational components* both across participants and trails.

Computational Model

Q-Learning

For each option, the model estimates its expected value (Q-value) on the basis of individual sequences of choices and outcomes. All Q-values were initialized at 0.5, and updated by the RPE ($r_i(t) - Q_i(t)$). he update rule is then:

$$Q_i(t+1) = Q_i(t) + \begin{cases} \alpha_{Gain}[r_i(t) - Q_i(t)] & if\ r = 1 \\ \\ \alpha_{Loss}[r_i(t) - Q_i(t)] & if\ r = 0 \end{cases} \tag{2}$$

Where $0 \leq \alpha_{Gain}, \alpha_{Loss} \leq 1$ denote positive & negative learning rates - the magnitude by which value beliefs are updated.

The separation is due to different striatal subpopulations are involved in positive and negative feedback learning - individuals tend to learn more from positive feedback.

Probability of Choice

Given the Q-values, the probability of selecting option A over B is then described by a softmax/Boltzmann distribution:

$$P_A(t) = \frac{exp(\beta Q_A(t))}{exp(\beta Q_B(t)) + exp(\beta Q_A(t))}$$

Where $0 \leq \beta \leq 100$ capture the exploitation/exploration parameter. Large β tends to greater sensitivity to change, thus more exploitative tendencies.

Results

Behaviour & model performance

Participants learned the stimulus-reward contingencies well, as they correctly learned to select the higher reward probability option in all three pairs. Learning decreased significantly with choice reliability. The Q-learning model was able to adequately simulate behaviour by the learned parameter estimates ($\alpha's\ and\ \beta's$).

Pupil responses predict individual differences in value-based decision making

Pupil responses evoked by choice & feedback differentially predict the underlying processes supporting value-based decision in the learning phase.

- β : the tendency to exploit predicts stronger pupil dilation leading up to a value-driven choice.
- α_{Gain} predicts amplified feedback related response (negatively correlated).
- This is consistent with the theoretical tenets of the Q-learning model: the explore-exploit parameter β determines the outcome of a value-driven choice and learning rates α affect value belief updating.

Pupil dilation reflects the value of the upcoming choice during, but not after, reinforcement learning

In the learning phase pupil dilation uniquely reflected the value belief determining the upcoming choice. There were no significant differences between groups, meaning pupils tracked differences in value beliefs but did not reflect uncertainty or cognitive effort.

Feedback-related pupil responses reflect value uncertainty and reward prediction errors (RPE)

Biphasic feedback related pupil responses were observed. Early pupil dilation was modulated by uncertainty about the value of options - choices between similar valued options increased dilation the most. Late pupil constriction was modulated by the violation of current value beliefs - worse than expected outcomes elicited stronger pupil constriction.

Fitting the Model

The model was fit using a Bayesian hierarchical fitting procedure where individual parameter estimates were drawn from group-level parameter distributions that constrained the range of possible individual parameter estimates - in a Bayesian sense, once can view the group as a prior on the individual parameters.

- $r_i(t-1)$ and $ch_i(t)$ are obtained from the data. The probit transformation is used to model binary response data.
- A normal prior was applied to group level means μ_z, $N(1, 0)$ and a uniform prior to the group level standard deviation δ_z, $U(1, 1.5)$.
- The Bayesian hierarchical model was implemented in STAN - with multiple chains generated to ensure convergence.
- Averaged simulated choice behaviour appears to closely mimic the actual response data.

STAN: High Performance Probabilistic Statistical Computation

The hierarchical Bayesian model was fit using the STAN software package. STANS allows for one to conduct full Bayesian statistical inference with MCMC sampling, as well as approximate Bayesian inference with variational inference (ADVI). The model was fit via full MCMC sampling over multiple chains.

Pupillometry

Pupillometry is concerned with pupil size & reactivity - playing a key part in clinical neurological examination for patients with a number of neurological injuries as well as being used in psychological research. The pupil responses in the study were deconvolved in attempt to derive some explanatory power - in attempt to infer causality.

Design Matrix

Two design matrices were computed to aid the deconvolution analysis, the first, characterising the **learning phase** constituted:

- Time course: the onset of choice (start of decision interval).
- Time course: choice.
- Time course: feedback.
- Stick regressor: onset of the fixation dot (on screen).
- Stick regressor: offset of the options on screen.
- $\Delta Q - value$: difference in option values - within the decision/choice interval.
- $\Delta Q - value$: difference in option values - within the feedback interval.
- single trail RPE estimate.

The design matrix for the **transfer phase** is identical with two amendments:

- The time course for feedback was omitted - as no feedback is given.
- A stick regressor was included to account for choice conflict.

Pupil size deconvolution

Deconvolution was conducted by fitting a ridge regression that models the pupil size as a function of the (above mentioned) design matrices, namely:

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

Cross validation was employed to optimize/tune λ. Nonparametric cluster-based permutation t-tests was employed to test significance of regressors & to correct for multiple comparisons overtime.

The resulting deconvolution was then used to explain pupil size fluctuations.

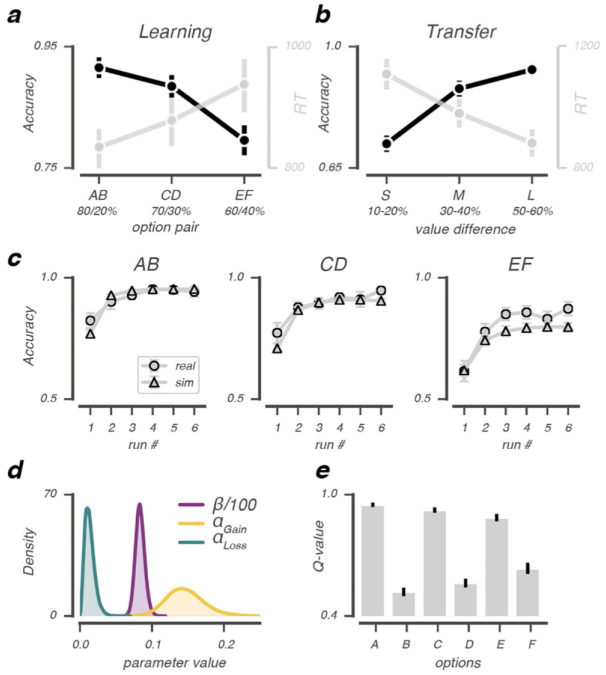

Figure 2: Parameter Analysis: **a.** Average accuracy across uncertainty groups - in the learning phase. **b.** Average accuracy per uncertainty group - in the transfer learning phase. **c.** Simulated vs actual (average) learning performance - not the near perfect, smoother, mapping. **d.** Posterior parameter distributions. **e.** Q-values over choices.

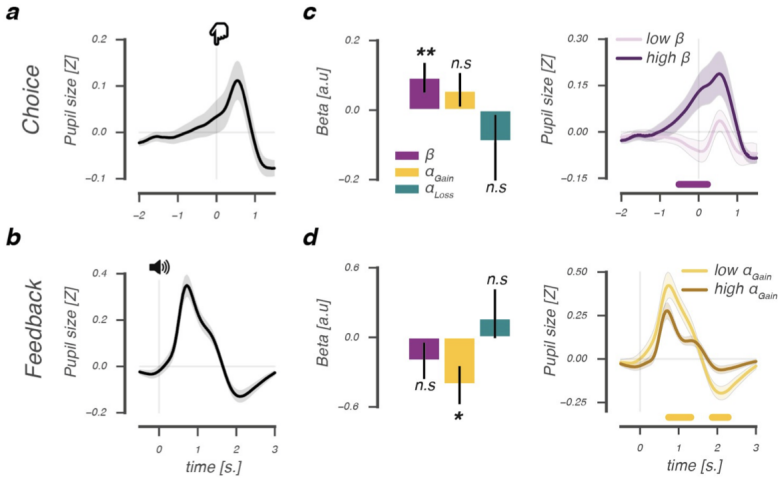

Figure 3: Deconvolution Analysis: Ridge regression modelling pupil size. **a.** & **b.** Modelled (predicted average) response pupil size at the time of choice and after feedback respectively. **c.** & **d.** *left panels* Parameter distributions over β and *right panels* the modelled response varying in magnitude of β's and α's - over choice & feedback stages respectively.

STAN Instance

Whilst a wealth of statistical techniques are explored, the core model is fit using STAN - a statistical inference framework for probabilistic programming.

Probabilistic Programming

Probabilistic Programming is a statistical computational paradigm that employs a full Bayesian approach to model estimation. Whilst these models can fail to scale to extremely large datasets, the Bayesian paradigm offers a host of powerful advantages:

- Full credibility intervals over the parameter space.
- An explicit formulation of capturing uncertainty.
- Thorough statistical theory allows for explicit inference on the model, without assuming we're working in the limit (taking expectations) as in the frequentist setting.
- Incorporation of prior information - and inversely, explicit specification of uncertainty (visiable uncertaintly).
- The ability to specify explicit, intuitive model structure - like this hierarchical instance.

The drawback of these methods is the computational cost - standard optimization procedures do not translate well as one needs to sample from the approximate posterior distribution to conduct inference - and approximating gradients can be intractable. Bayesian approximation is concerned with sample from high variance parameters such to generate a dataset (sample trace) that maximizes variation (information). Methods like AVDI are able to scale models - where traditional Monte Carlo procedures break - however scalability does warrant concern. Gradient approximators can are also utilized to speed computation, to sample efficiently.

STAN

The primary implementation of the paper was implemented in STAN - a state of the art high-performance statistical computational software. STAN offers a both Python and R wrappers, making it very accessible. The model specification is very intuitive, first one specifies that *data* - enclosed brackets (</ >) placing lower & upper bounds on the raw value of the variable, whilst square brackets denote an index (the number of variables). For example, the data variable $int < lower = 1, upper = 5 > \ Choice[n_t]$ denotes the individual's choice on trail t (there are n_t observations), each of which is limited to $(1 \leq x \leq 5)$.

Thereafter one specifies the trainable *parameters*, including both informative and uninformative priors. The same bounding & indexing structure applies - note which parameters form the individual level parameters (they are indexed). All (and exclusively) our priors are specified in this section (with the suffix: var_{pr}). The subsequent *transformed parameters*, allow for parameter transformations - & thus our model's probit transformation.

The *model* section then specifies the sequence of parameter calls. Loops allow one to traverse through all group level parameters - though a vector form is also possible. Note that the priors are specified as exact parameterized distributions. Also note that all trainable variables are defined as probability distributions and - and not static values.

The model samples through subjects, over each trail, iteratively specifying parameter estimates per trail. Ultimately computing the probability of choice, and alpha reinforcement learning parameters (Q-learning update) per subject, per trail.

Listing 1: PySTAN Model Instance

```
#!/usr/bin/env python
# encoding: utf-8

def import_stan_model_code():

        stan_model = """

        data{
          int<lower=1> n_s;                       // number of subjects
          int<lower=1> n_t;                       // number of trials per subject, plus total number of trials
          int<lower=1,upper=5> Choice[n_t];       // choice options trial n_t (choice is 1,3 of 5). (All subjects stacked)
          int<lower=1,upper=2> Correct[n_t];      // correct (=1, yes-correct)? trial n_t
          int<lower=0,upper=1> Reward[n_t];       // reward (=1, yes)? trial n_t
          int<lower=1,upper=n_s> Subject[n_t];    // subject number? n_s
          int<lower=0,upper=1> Init[n_t];         // is this first trial of a subject? Should RL be initialized?
        }                                         // end data


        parameters{
          // group level mean parameters
                real mu_b_pr;                             //inverse gain parameter
                real mu_ag_pr;                                  //alphaG
                real mu_al_pr;                                  //alphaL


          // group level standard deviation
          real<lower=0> sd_b;                             //inverse gain parameter
                real<lower=0> sd_ag;                            //alphaG
                real<lower=0> sd_al;                            //alphaL


          // individual level parameters
          real b_ind_pr[n_s];                             //inverse gain parameter
                real ag_ind_pr[n_s];                            //alphaG
                real al_ind_pr[n_s];                            //alphaL
                }//end paramters

        transformed parameters{
                // group level mean parameters
          real<lower=0,upper=100> mu_b;                                 //inverse gain parameter
                real<lower=0,upper=1> mu_ag;                            //alphaG
                real<lower=0,upper=1> mu_al;                            //alphaL


          // individual level parameters
          real<lower=0,upper=100> b_ind[n_s];                   //inverse gain parameter
                real<lower=0,upper=1> ag_ind[n_s];                      //alphaG
                real<lower=0,upper=1> al_ind[n_s];                      //alphaL


          // group level mean parameters (probit)
          mu_b  =Phi(mu_b_pr)*100;              //inverse gain parameter
                mu_ag =Phi(mu_ag_pr);                           //alphaG
                mu_al =Phi(mu_al_pr);                           //alphaL


          // individual level parameters (probit)
          for (s in 1:n_s)
          {
            b_ind[s]  = Phi(b_ind_pr[s])*100;
            ag_ind[s] = Phi(ag_ind_pr[s]);
            al_ind[s] = Phi(al_ind_pr[s]);

          }// end for loop

                }// end transformed parameters



        model{
          // define general variables needed for subject loop
          int si;
          real prQ0[6];
          real prQ[6];
          real Qchoice[2];
          real epsilon;
          int a;
          real alpha;
          vector[2] pchoice;

          // set prior on group level mean parameters
          mu_b_pr ~  normal(0,1);                       //inverse gain parameter
                mu_ag_pr ~ normal(0,1);                         //alphaG
                mu_al_pr ~ normal(0,1);                         //alphaL
```

```
  // set prior on group level standard deviations
  sd_b ~  uniform(0,1.5);                  //inverse gain parameter
        sd_ag ~ uniform(0,1.5);                                  //alphaG
        sd_al ~ uniform(0,1.5);                                  //alphaL


  // set prior for individual level parameters
  for (s in 1:n_s)
  {
    b_ind_pr[s] ~ normal(mu_b_pr,    sd_b);                 //inverse gain parameter
          ag_ind_pr[s]~ normal(mu_ag_pr,   sd_ag);                          //alphaG
          al_ind_pr[s]~ normal(mu_al_pr,   sd_al);                          //alphaL

  }



  // defineer epsilon
  epsilon = 0.00001;

  // now start looping over subjects
  for (t in 1:n_t)
  {

      // set initial values subject
      if (Init[t]==1){
            si= Subject[t];
            for (v in 1:6)
              {
                prQ0[v] = 0.5;
                prQ[v] = 0.5;

              }// end inital values loop
          // trial 1
          pchoice[1]=0.5;
          pchoice[2]=0.5;
        }


          Qchoice[1]     = prQ[Choice[t]];
          Qchoice[2]     = prQ[(Choice[t]+1)];
            pchoice[1]     = 1/(1+exp(b_ind[si]*(Qchoice[2]-Qchoice[1])));
                pchoice[2]     = 1-pchoice[1];
                pchoice[1]     = epsilon/2+(1-epsilon)*pchoice[1];
          pchoice[2]     = epsilon/2+(1-epsilon)*pchoice[2];

          Correct[t]~categorical(pchoice);
          a = Correct[t]-1; //0=correct,1=incorrect

                // reinforcement
          alpha = Reward[t]*ag_ind[si]+(1-Reward[t])*al_ind[si];
                prQ[(Choice[t]+a)] = prQ[(Choice[t]+a)] + alpha*(Reward[t]-prQ[(Choice[t]+a)]);


     }// end subject loop
}// end of model loop

"""
return stan_model
```

FIN.