Beads task modelling

Matteo Lisi

2024-06-20

Optimal model, general case (unequal prior probabilities)

To find $p(G \mid n_d, n_g)$ when $p(G) \neq p(B)$, we use Bayes' theorem:

$$p(G \mid n_d, n_g) = \frac{p(n_d, n_g \mid G) \cdot p(G)}{p(n_d, n_g)}$$

Where:

- $p(G)$ is the prior probability of the green urn.
- $p(B) = 1 - p(G)$ is the prior probability of the other urn.
- $p(n_d, n_g \mid G)$ is the likelihood of observing n_d draws with n_g green beads given the green urn.

Likelihood

1. **Green Urn (G)**:
   - Probability of drawing n_g green beads out of n_d total draws when drawing from the green urn:

$$p(n_d, n_g \mid G) = \binom{n_d}{n_g} q^{n_g} (1-q)^{n_d - n_g}$$

2. **Other Urn (B)**:
   - Probability of drawing n_g green beads out of n_d total draws when drawing from the other urn:

$$p(n_d, n_g \mid B) = \binom{n_d}{n_g} (1-q)^{n_g} q^{n_d - n_g}$$

where q is the probability of drawing a beads of the majority color.

Marginal likelihood

$$p(n_d, n_g) = p(n_d, n_g \mid G) \cdot p(G) + p(n_d, n_g \mid B) \cdot p(B)$$

$$p(n_d, n_g) = \binom{n_d}{n_g} \left[q^{n_g} (1-q)^{n_d - n_g} \cdot p(G) + (1-q)^{n_g} q^{n_d - n_g} \cdot p(B) \right]$$

Applying Bayes

$$
\begin{aligned}
p(G \mid n_d, n_g) &= \frac{p(n_d, n_g \mid G) \cdot p(G)}{p(n_d, n_g)} \\
&= \frac{\binom{n_d}{n_g} q^{n_g}(1-q)^{n_d-n_g} \cdot p(G)}{\binom{n_d}{n_g}\left[q^{n_g}(1-q)^{n_d-n_g} \cdot p(G) + (1-q)^{n_g} q^{n_d-n_g} \cdot p(B)\right]} \\
&= \frac{q^{n_g}(1-q)^{n_d-n_g} \cdot p(G)}{q^{n_g}(1-q)^{n_d-n_g} \cdot p(G) + (1-q)^{n_g} q^{n_d-n_g} \cdot p(B)} \\
&= \frac{p(G)}{p(G) + \left(\frac{1-q}{q}\right)^{n_g}\left(\frac{q}{1-q}\right)^{n_d-n_g} \cdot p(B)} \\
&= \frac{p(G)}{p(G) + \left(\frac{q}{1-q}\right)^{n_d-2n_g} \cdot (1-p(G))}
\end{aligned}
$$

This formula shows the conditional probability $p(G \mid n_d, n_g)$ when the prior probabilities $p(G)$ and $p(B)$ are not equal.

In the special case where $p(G) = p(B) = (1-p(G)) = \frac{1}{2}$ this simplifies in the formula in the Furl & Averbeck (2011) paper (equation 1):

$$
p(G \mid n_d, n_g) = \frac{1}{1 + \left(\frac{q}{1-q}\right)^{n_d-2n_g}}
$$

Code

In R, the posterior probability can be computed as

```
prob_g <- function(g, n, q=0.7, prior=0.5){
  prior / (prior + (1-prior)*(q/(1-q))^(n-2*g))
}
```

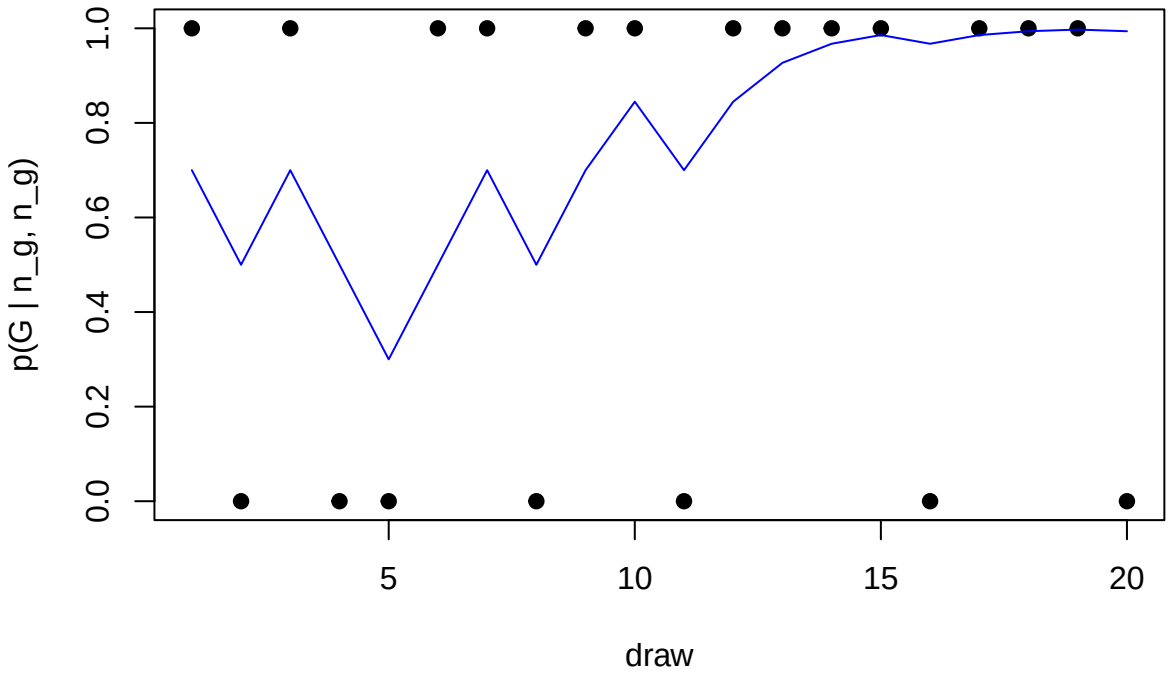
Which can be used to iteratively compute beliefs of the ideal observer after each draw of a sequence:

```
# simulate 10 draws with split 0.7
set.seed(123)
q <- 0.7
n_draws <- 20
draws_g <- rbinom(n_draws, 1, prob=q)

# compute the probability
estimated_prob_g <- rep(NA, n_draws)
for(i in 1:n_draws){
  estimated_prob_g[i] <- prob_g(g=sum(draws_g[1:i]),
                                n=i,
                                q=0.7,
                                prior=0.5)
}

# plot
```

```
plot(1:n_draws, draws_g, xlab="draw", ylab="p(G | n_g, n_g)", pch=19)
lines(1:n_draws, estimated_prob_g, col="blue")
```



Delta rule updating model

An alternative, perhaps descriptively more plausible model of human observers could use a form of delta-rule updating. This model is not optimal anymore, but mimic an observer with limited memory, by updating the probability incrementally, in a way that .

Parameters of the model:

- $p(G)$ is the prior probability of the green urn.
- α learning rate
- β "inverse temperature" parameters of a softmax function that transform the estimated fraction of green beads after current draw to the probability that the urn we are sampling from is the one with majority of green beads.

The model essentially after each draw update the probability estimate $\hat{q}$ using the new information and a weighted combination of the previous probability estimate.

Let's define d_t as an indicator variable indicating the outcome of the t-th draw (1 if green, 0 if the other color).

The probability updating is then

$$\hat{q}_t = \hat{q}_{t-1} + \alpha \left(d_t - \hat{q}_{t-1} \right)$$

And the probability $p_t(G)$, corresponding to the probability that the draws are coming from green urn after t draws is:

$$p_t(G) = \frac{e^{\beta \hat{q}_t}}{e^{\beta \hat{q}_t} + e^{\beta (1-\hat{q}_t)}}$$

Code

In R, the delta rule model is

```
prob_g_delta <- function(draws, alpha = 0.1, beta=1, prior=0.5){

  q_hat <- prior
  q_hat_list <- numeric(length(draws))
  q_hat_list[1] <- q_hat

  prob_g_list <- numeric(length(draws))

  for (t in 1:length(draws)) {
    q_hat <- q_hat + alpha * (draws[t] - q_hat)
    q_hat_list[t] <- q_hat
    prob_g_list[t] <- exp(beta*q_hat)/(exp(beta*q_hat) + exp(beta*(1-q_hat)))
  }

  return(list(q=q_hat_list, g=prob_g_list))
}
```

This model is very flexible, and can reproduce effect such as over-adjusting the probability relative to the optimal (for example by setting a larger β - see below).

```r
# compute the probability
alternative_prob_g <- prob_g_delta(draws_g[1:i], alpha = 0.1, beta=12, prior=0.5)

# plot
plot(1:n_draws, draws_g, xlab="draw", ylab="p(G)", pch=19)
lines(1:n_draws, estimated_prob_g, col="blue")
lines(1:n_draws,alternative_prob_g$g, col="red")
legend(x=14,y=0.4,legend=c("optimal","delta rule"),lwd=1, col=c("blue","red"), bty="n")
```