The Random Walk of Distraction

Andrew T. Smith

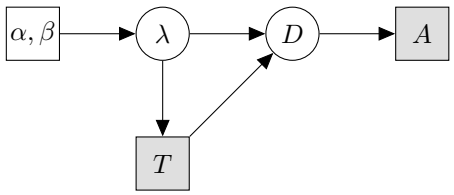
May 20, 2022

1 Introduction

The events that distract us seem to occur randomly, including both external stimuli and internal thoughts, such as like a song stuck in one's head. This can be modeled as a Poisson process, but we cannot observe the exact time a user becomes distracted, only periodically inquire about what they are experiencing. Second, the "sticky" nature of distraction means that a user's estimate of how many recent events they experienced that *could* have pulled them away from their task is only reliable if it is 0 or 1. Truly distracting events are assumed drive the user into a terminal state, in this sense. One can be distracted *from* other distractions but returning to work is a dicier prospect, and for this paper, assumed not to happen.

2 Distraction Model

The distraction model learns from data generated by a sequence of N trials. Let the observation of a trial consists of the pair (A, T), where T is the trial duration and A is the user's binary response at the end. During each trial the user will work on other tasks, and during this time they will be confronted with an unknown number D of irresistible distractions until time T passes. Then they are asked if they are either in a distracted, off-task state, or have remained working. Their answer, A (where $A = 1$ means "yes, distracted", etc.) is used to update the model parameters α and β, which define the (gamma) distribution of the latent random variable λ, the expected number of distractions during time T. D has a Poisson distribution with rate parameter λ/T.



Param.	Distribution	Description
α, β		(model parameters)
λ	$\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}$	rate of distraction, gamma distribution $\Gamma_{\alpha(i),\beta(i)}(\lambda)$ for trial i
T		duration of trial, calculated from user settings and λ
D	$\frac{(\lambda T)^D}{D!}e^{-\lambda T}$	number of distractions during period T, i.e. the Poisson distribution $\Pr_{\lambda(i)}(D)$
A		user response $A = \mathbb{1}_{D>0}(i)$, i.e. 1 if distracted, otherwise 0

Distraction model: observed variables are shaded, latent unshaded. Random variables are in round nodes and deterministic values / model parameters in rectangles. (Deterministic variables have no distribution.)

Trial durations $T(i)$ are set to the expected wait time for the probability that the first distraction has occurred to exceed a user-set threshold p given the current *maximum a posteriori* estimate of the rate parameter λ, i.e. $P(D \geq 1|\lambda) \geq p$. Since D has a Poisson distribution, the expected wait times between distracting events has an exponential distribution with the same parameter λ:

$$\begin{aligned} P(t < T) = F_\lambda(t) &= 1 - e^{-\lambda t} \\ T(i) &:= F_\lambda^{-1}(p) \end{aligned} \tag{1}$$

where F_λ is the exponential CDF.
The user can also decide to end the time period earlier, resulting in an (A, T) observation with a shorter T.

2.1 Learning with Monte Carlo

2.1.1 Factoring the joint distribution

The graphical model encodes conditional independence relationships that simplify factoring the joint distribution (first step) and using the fact that the deterministic variables have $p = 1$ simplifies it further (2nd step):

$$\begin{aligned} p(L, D, T, \lambda, \alpha, \beta) &= p(L|D, T, \lambda, \alpha, \beta)p(D|T, \lambda, \alpha, \beta)p(T|\lambda, \alpha, \beta)p(|\lambda|\alpha, \beta) \\ &= p(T|P)p(\lambda|\alpha, \beta)p(D|\lambda, T)p(L|D) \\ &= p(\lambda|\alpha, \beta)p(D|\lambda, T), \end{aligned} \tag{2}$$

The generative process starts with λ being sampled from the gamma distribution defined by the current values of α and β. Then the trial duration T is calculated from λ and the user setting p. D is sampled from the Poisson distribution ??? with rate λ/T. Finally, the user reports their answer A, whether $D > 0$ or not.
BROKEN AFTER HERE
Given the rate of user distractions λ, the number of distractions $D(i)$ is a random variable with a Poisson distribution

$$P(D = k|\lambda, t) = \frac{(t\lambda)^k e^{-t\lambda}}{k!} \tag{3}$$

where F_λ is the exponential CDF with rate parameter λ. Therefore the time required for the probability of the next distraction to exceed $p = P(t; \lambda)$ is the inverse CDF:

$$\begin{aligned} t_\lambda(p) &= F_\lambda^{-1}(p) \\ &= \frac{\ln(\lambda) - \ln P(t; \lambda)}{\lambda} \end{aligned} \tag{4}$$

3 Interrupting a distracted user

This is an outline of an app to model a user's distraction process by using timed alarms to prompt the user to input their mental state.

3.1 Basic usage

1. The user sets an "alarm" by setting a threshold $P(t; \lambda)$ or, equivalently, the expected wait time for that threshold to be exceeded $t_\lambda(p)$.
2. After the expected wait time has elapsed, the alarm is activated.
3. The alarm should have three different buttons that deactivate and reset it:
   - Red - User was distracted and should have been interrupted sooner.
   - Yellow - User was distracted but has only recently become so.
   - Green - User was not distracted and should not have been interrupted so soon.
4. The app adjusts the alarm period based on which button was pressed, and repeats back to step 1.
5. If the user presses the buttons early, the response is the same.

3.2 App features

- The alarm should be timed so as not to be too frequent that it *itself* interrupts the user's work, while also not too infrequent that the user wastes much time in a distracted state.
- The alarm timing should be well-calibrated, in the sense that if the user selects a threshold probability p, then the proportion of times the alarm find the user in a distracted state should approach p. This is what the proportion of clicks in {Red, Green} that are Red (user is distracted) approaches until Yellow is clicked, because:
- A Yellow click indicates directly that the alarm is calibrated and the rate should not be adjusted.
- The rate should be calculated from as many of the most recent clicks as possible, but is constrained by Yellow clicks. Because the user did not desire adaptation at this point, clicking Red / Green after a Yellow means the former rate λ is no longer valid, but a reasonable prior probability $P(\lambda)$.
- Adaptation can be optimal: the Red and Green responses should allow the alarm timing to approach the user's desired rate as quickly as possible, or adapt as responsively as possible as that rate changes over time.

4 Optimal Adaptation - Bayesian updates

The probability threshold is set by the user, so the app must adjust an internal estimate of the rate parameter λ to raise the alarm at the desired time. Since the user's Yellow clicks indicate optimality, we only update the model based on data after the most recent Yellow click.
Let:

- $L(t) \in \{\text{Red}, \text{Yellow}, \text{Green}\}$ be the sequence of letters indicating the button pressed at time t for $0 \leq t \leq T$, e.g. $L = (..., R, G, Y, Y, R, G)$
- $D(T)$ be the duration of the time period preceeding the button click at time t (e.g. in seconds), and
- $A(t)$ indicate the state of the alarm $A(t) \in \{0 = \text{silence}, 1 = \text{alarm}\}$.

Given the most recent user interaction from time S to time T, $D = \{(d_t, l_t, a_t) : S \leq t \leq T, L(S) = YorS = 0\}$, what is the best estimate of the rate λ?
The Bayesian update model consists of:

$$P(\lambda|d,l) = \Gamma(x)P(d,l|\lambda)$$
$$P(\lambda|d,l) \tag{5}$$

and uses Bayes' rule to get the new distribution ofλ:

5 old

Given the User's history with the alarm clock, a sequence of obersvations, pairs of time durations and the user's responses to the alarms at the end of each one, $O_i = (d_i, r_i)$, where $r_i \in \{\text{red}, \text{yellow}, \text{green}\}$ and $0 \leq i \leq N$, the optimal rate is defined as the maximum *a posteriori* estimate of λ.
The conjugate prior of the exponential distribution with parameter λ is:

$$\pi(\lambda; k, \theta) = \frac{1}{\Gamma(k)\theta^k}\lambda^{k-1}e^{\frac{\lambda}{\theta}}$$

where $1/\theta$ is β, the "rate paramter" of the Gamma distribution.
Given new observation at time $n+1$:

$$P(\theta|O_{n+1}) = \frac{P(O_{n+1}|\theta) * P(\theta)}{P(O_{n+1})}$$

Which suggests the update rule:

$$\theta_{n+1} = \operatorname*{argmax}_{\theta} \frac{P(O_{n+1}|\theta_n) * P(\theta_n)}{P(O_{n+1})}$$
$$= \operatorname*{argmax}_{\theta} \mathrm{Exp}(O_{n+1}|\theta_n) \tag{6}$$
$$\tag{7}$$

6 Future work

The state of the alarm is not used in the update. It is determined only by the user's threshold setting and the current probability estimate (which is defined by the current rate estimate λ and time t), and its only purpose is to get the user to interact with the app. This raises two issues:

1. If user presses Red/Green when the alarm is not ringing, the durations $D(t)$ in the trainable suffix will be biased, expected to be less than the durations resulting from each button press being immediately after an alarm started. The effects of this on the learning are unclear.

2. Beyond learning the distraction rate λ, yt is also possible to learn what makes a user more / less likely to interact with the app, an interaction rate ξ. Like many active learning problems, this may lead to evil.