An Reinforcement Learning Approach for Effective Survey Plans: Applications to Alcohol Use Disorder

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Introduction

There are in total $N = 151$ participants.

2 Preliminaries

We model the prediction data as sequences of received surveys; for each participant, X_i the participant's $i^{\mathtt{th}}$ *received* response (including the response time). We also create a binary prediction label y_i within the next 24-hour window forecast at the $i^{\mathtt{th}}$ prediction point as in [3].

Environment Specifics. We formulate the survey design problem as offline reinforcement learning (RL). Specifically, we model the interaction with each participant as an *episode*. Our goal is to design a survey plan that maximizes the *expected* cumulative rewards (and minimize penalties) over all participants during the entire episodes. That is, we consider each participant as a new environment instantiated out of N environments; each environment is simulated based on the offline survey data collected in [3].

In each episode with a participant, at each $t^{\mathtt{th}}$ time-step, we either ask (1) whether or not to reveal the survey, or (2) to predict the next 24-hour window lapse. When we get the next survey X_t from the participant (the former), we choose a binary action a_t deciding whether to reveal the survey or not; that is, we do not observe the survey $o_t = 0$ without penalty if $a_t = 0$, and observe the received survey $o_t = X_t$ if $a_t = 1$, with a penalty depending on penalty levels. This penalty level is a control factor that balances the survey frequency and the prediction accuracy.

If the environment asks to predict the next 24-hour window lapse event y_t (the latter), an agent does not get any observation, only getting a hidden reward/penalty depending on whether the prediction is correct. The exact reward and penalty values can be found in Table 1.

Reinforcement Learning. Our goal is to learn a survey policy π that maximizes the cumulative returns – the sum of all rewards and penalties – averaged over all participants. The policy π takes all k-step previous observations, which we call a historical context or simply *history*, and decides an action to take. We let $k = 25$ in our experiments. More details can be found in Appendix A.

3 Method

We take a two-phase training approach where we separate two tasks: (1) learn compact representations of truncated historical contexts to predict the lapse outcome and (2) improve the optimal survey

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

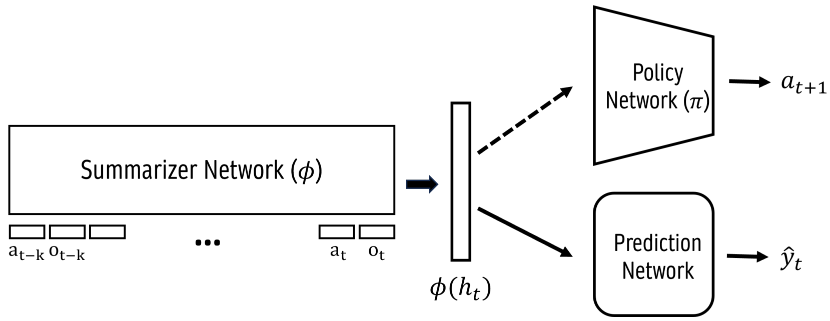

Figure 1: *Architecture for two-phase training*

strategy through reinforcement learning. More specifically, our training pipeline alternates between the following two tasks:

1. *(Prediction Learning)* The truncated k-step history $h_t = (o_{t-k}, a_{t-k}, ..., o_t)$ is input into the history summarization model $\phi(\cdot)$, yielding summary statistics $\phi(h_t)$. Then, this summary statistics is fed into the prediction network to predict the next lapse label. The sequence model ϕ is updated to minimize the prediction loss in this phase.
2. *(Reinforcement Learning)* In this phase, new trajectories are generated with ϕ and the current policy π, which takes a summary statistics $\phi(h_t)$ as an input and outputs the next action (see Figure 1). π is updated to maximize the long-term returns of the system.

Architecture. Our default configuration employs transformer architectures for the summarizing model. For implementing the transformer model, we adhere to the minimal implementation of NanoGPT's standard framework[1]. We use an `Embedding` layer for discrete actions. For the policy network, we employ the soft actor-critic method for discrete actions (SACD) [2]. We use the base two-layer fully-connected architecture for both actor and critic networks. The lapse prediction network uses a single Gated Recurrent Unit (GRU) network [1]. Additional details can be found in Appendix B.

4 Experiments

All experiments were conducted on an NVIDIA GeForce RTX 3090.

- Our baseline is XGBoost with minimla feature engineering.

- We compare four penalty levels and see the effect of penalty design and accuracy.

A Environment Details

X_t: how each quantity is represented? (e.g., survey questions, quantities)

y_t: how is this calculated?

Penalty Level	Penalty
Lv 1	-0.02
Lv 2	-0.05
Lv 3	-0.08
Lv 4	-0.12

Lapse/Action	Correct	Wrong
Not Lapsed	0.03	-1.2
Lapsed	0.05	-0.2

Table 1: Penalty/Reward Table

[1] `https://github.com/karpathy/nanoGPT`

Algorithm 1 Decoupled Representation Learning and Reinforcement Learning (`DRL`^{2})

```
Inputs: step sizes α, β, tunable inner steps T_tr, T_gen
for i ≥ 1 do
    # Alternative training lapse prediction and reinforcement learning
    for j ∈ [T_tr] do
        φ ← φ − α ∇̂_φ PSRLoss(φ; D)
        π ← π − β ∇̂_π RLLoss(π; φ, D)
    end for
    # Generate new (simulated) samples
    for j ∈ [T_gen] do
        add a new trajectory sample to D with π
    end for
end for
```

Partially Observed Markov Decision Processes. An environment consists of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P)$, with the underlying (unobservable) state space $\mathcal{S}$; action space $\mathcal{A} = \{0, 1\}$; observation space $\mathcal{O}$; and transition kernels $P(s', o|s, a)$, where only o and a are observable quantities. $\mathbb{P}(\cdot)$ is the probability measure in this environment. Let a sequence of arbitrary length t of observations $o_{1:t} := (o_1, ..., o_t)$ and a sequence of actions $a_{1:t-1} := (a_1, ..., a_{t-1})$. An (unobserved) reward r_t is decided by a tuple (s_t, a_t). For simplicity, we assume that the historical contexts of the previous $k = 25$ steps form sufficient statistics of the true states of the system, *i.e.,* $(o_{t-k:t}, a_{t-k:t})$ with $k = 25$ is sufficient statistics of any given history $h_t := (o_{1:t}, a_{1:t})$. The numerical reward values are tuned with extensive hyperparameter search (see Table 1).

Objective Formulation. The optimization objective is to minimize the prediction loss:

$$\texttt{PLoss}(\phi; \mathcal{D}) := \mathbb{E}_{(h_t, y_t) \sim \mathcal{D}} \left[\ell(y_t; \phi(h_t)) \right], \tag{1}$$

where $l(\cdot; \cdot)$ is a cross-entropy (CE) loss function, and $\mathcal{D}$ is a trajectory dataset generated during the off-policy learning procedure.

The goal of reinforcement learning is to maximize the following cumulative objective

$$\texttt{RLoss}(\pi; \phi) := \mathbb{E} \left[\sum_{t=1}^{H} r_t \; \middle| a_t \sim \pi(\cdot | \phi(h_t)) \right], \tag{2}$$

where r_t collectively represents the penalty $(r_t < 0)$ and rewards $(r_t \geq 0)$. $\mathbb{E}$ is expectation over all participants with a running policy π.

B Experimental Details

Following the base minimal implementation, for all Actor-Critic and lapse prediction networks, we use the 2-layer fully-connected neural network architecture.

Hyperparameters. The tuned-hyperparameters can be summarized as the following:

- For SAC agent learning
  1. learning rate for the actor model: 0.0001
  2. learning rate for the critic model: 0.0002
  3. SAC update steps (T_{tr}) per phase: 500
  4. data generation episodes (T_{gen}) per phase: 50
  5. entropy regularizer: 0.03
- For backbone transformer architecture
  1. embedding dimension: 128
  2. number of layers: 1
  3. number of heads: 4

- For prediction network:
    1. network width of the first layer: 256
    2. network width of the second layer: 128
- For training:
    1. learning rate for the prediction model: 3e-5
    2. batch size: 512

References

[1] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] P. Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[3] K. Wyant, S. J. Sant'Ana, G. E. Fronk, and J. J. Curtin. Machine learning models for temporally precise lapse prediction in alcohol use disorder. *Journal of psychopathology and clinical science*, 133(7):527, 2024.