Learning Treatment Policies for Mobile Health Using Randomized Least-Squares Value Iteration

Celine Liang, Serena Yeung, Peng Liao, Susan Murphy

Harvard University, Department of Statistics, Statistical Reinforcement Learning Lab

Introduction

- This work investigates the use of Randomized Least-Squares Value Iteration (RLSVI) [1] for learning treatment policies in mobile health.
- We show that RLSVI learns a better policy and performs more robustly than an algorithm employing least-squares value iteration (LSVI) which selects actions in an ε-greedy manner.
- Eventual goal is to develop an online algorithm for mobile health that can learn and update the treatment policy efficiently and continuously.
- Continuing task adaptation of RLSVI is useful for tackling two of the main challenges in mobile health: the importance of *efficient learning* and *online learning as a continuing task*.
- Testbeds simulate the task of balancing reward and burden.

Problem Formulation

Motivation

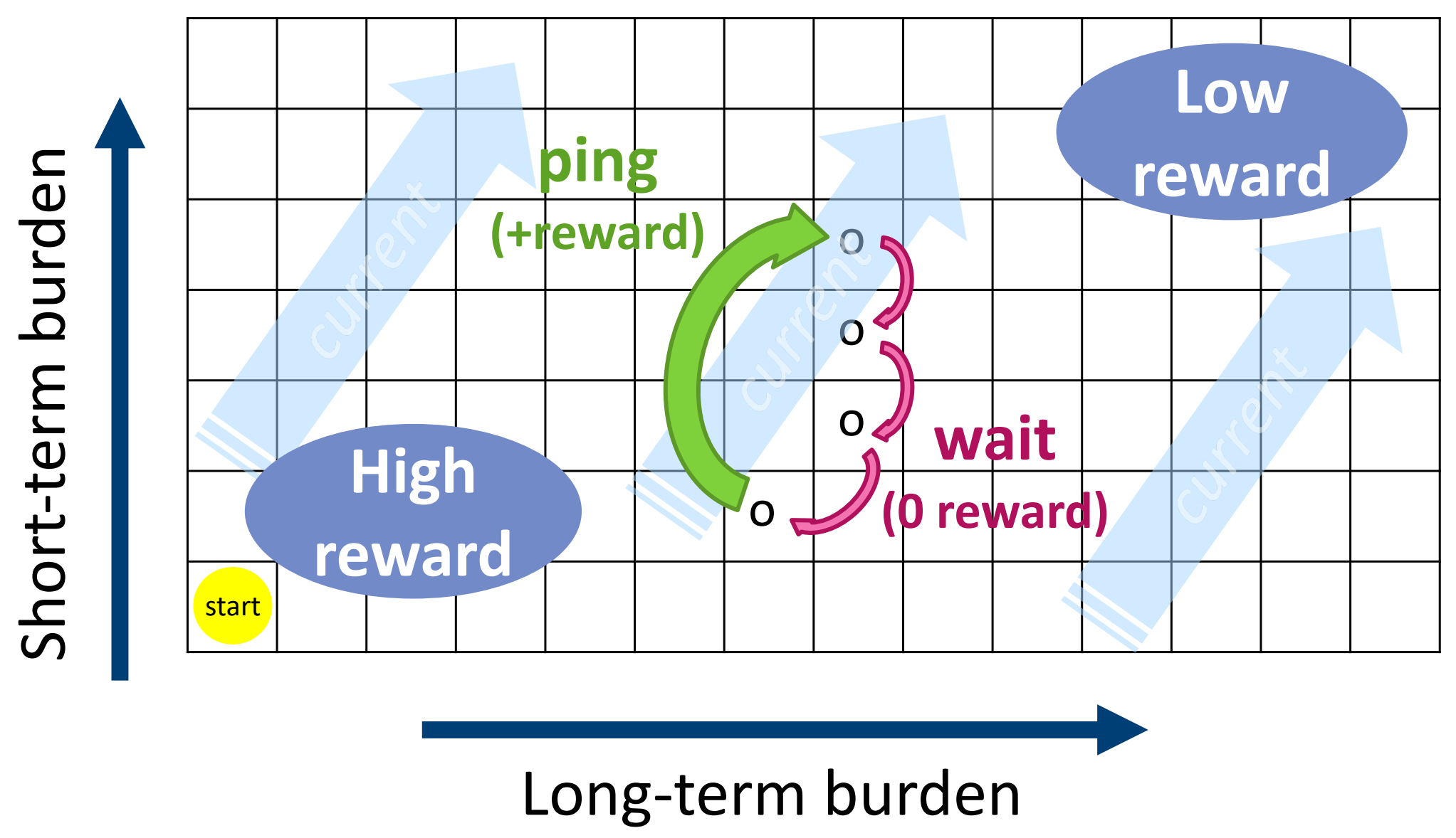
- Want to model a simplified mobile health problem of determining when to send mobile notifications to a user.
- To do this, we define an agent on a two dimensional state space where the horizontal and vertical displacement at time t is $s_t^{(0)}$ and $s_t^{(1)}$.
- At each time step *t,* two possible action choices: pinging (delivering treatment to user) or waiting (no treatment).
- Pinging increases quantity of short-term burden by the *burden size,* d, causes agent to receive lower immediate reward, and contributes to long-term burden.
- Waiting decreases short-term burden by small amount but causes agent to receive no reward.
- Greater reward is obtained by pinging in low-burden states than in high-burden states.

Definitions

- Short-term burden (vertical axis, $s_t^{(1)}$) is caused by a single ping that affects the user's immediate reward.
- Long-term burden (horizontal axis, $s_t^{(0)}$) is caused by the total treatment that the user has undergone; decreases effect of future treatments.
- Burden size (d) measures the user's inclination to feel burden; increase in short-term burden accumulated upon receiving a ping.
- Current (c) represents the many problems that users confront that push them to poor states (this is motivated by *RiverSwim* in [2])

Parameters

- Burden size d that short-term burden increases by if user is pinged
- Memory window T of previous actions
- Lookahead horizon H on which to perform value iteration
- Current c is the action-error probability that a wait action fails and instead causes the agent to remain in the same place; ping actions are not affected by current and proceed as usual



Problem Formulation (cont.)

At time step t:

- State is $\left(s_t^{(0)}, s_t^{(1)}\right) \in \{0, \dots, d(T+1)\} \times \{0, \dots, T\}$.
- Action is $a_t \in \{0,1\}$ where 1 corresponds to pinging user and 0 corresponds to waiting.
- Transition function is $T(s,a) = s'$ such that

$$T\left(\left(s_t^{(0)}, s_t^{(1)}\right), 1\right) = \left(\min\left(s_t^{(0)} + d, d(T+1)\right), p_t\right)$$

$T\left(\left(s_t^{(0)}, s_t^{(1)}\right), 0\right) = \left(\max(0, s_t^{(0)} - 1), p_t\right)$ with probability $1 - c$, and $\left(s_t^{(0)}, s_t^{(1)}\right)$ otherwise, where p_t is the number of pings in last T time steps.

- Reward observed from taking action a_t at state s_t is

$$r_t = R\left(\left(s_t^{(0)}, s_t^{(1)}\right), a_t\right) + \epsilon_t$$

where $\varepsilon_t \sim N(0, \sigma_{true}^2)$ is random noise and our reward function is

$$R\left(\left(s^{(0)}, s^{(1)}\right), 1\right) = r_0^{-s^{(0)}} r_1^{-s^{(1)}}$$

$$R\left(\left(s^{(0)}, s^{(1)}\right), 0\right) = 0$$

where $r_0 \geq 1, r_1 \geq 1$ are fixed constants.

Algorithm

```
Algorithm 1: Continuing task RLSVI
Input: Features $\Phi(s_i, a_i), r_i : i < t, \lambda > 0, \sigma > 0, H > 0$
Output: $\tilde{\theta}_{t,0}, \dots \tilde{\theta}_{t,H-1}$
for $h = H-1, \dots, 1, 0$ do
    Generate regression problem $A \in \mathbb{R}^{(t-H+1)\times K}, b \in \mathbb{R}^{t-H+1}$:
    for $i = 0, 1, \dots, t-H$ do
        $A_i \leftarrow \Phi(s_{h+i}, a_{h+i})$
        $b_i \leftarrow \begin{cases} r_{h+i} + \max_\alpha \left(\Phi\tilde{\theta}_{t,h+1}\right)(s_{h+i+1}, \alpha) & \text{if } h < H-1 \\ r_{h+i} & \text{if } h = H-1 \end{cases}$
    end
    Bayesian linear regression for the value function
        $\bar{\theta}_{t,h} \leftarrow \sigma^{-2}\left(\sigma^{-2}A^T A + \lambda I\right)^{-1} A^T b$
        $\Sigma_{t,h} \leftarrow \left(\sigma^{-2}A^T A + \lambda I\right)^{-1}$
    Sample $\tilde{\theta}_{t,h} \sim \mathcal{N}(\bar{\theta}_{t,h}, \Sigma_{t,h})$ from Gaussian posterior
end
```

```
Algorithm 2: Policy learning procedure for continuing tasks
Input: Features $\Phi$; $\sigma > 0, \lambda > 0, H$
for $t = 0, 1, 2, \dots$ do
    if $t \leq H-1$ then
        Use random $\tilde{\theta}_{t,0}, \tilde{\theta}_{t,1}, \dots, \tilde{\theta}_{t,H-1} \sim \mathcal{N}(0, 1/\lambda)$
    else
        Obtain $\tilde{\theta}_{t,0}, \dots, \tilde{\theta}_{t,H-1}$ from Algorithm 1
    end
    Sample $a_t \in \arg\max_{\alpha \in \mathcal{A}} \left(\Phi\tilde{\theta}_{t,0}\right)(s_t, \alpha)$
    Observe $r_t$ and $s_{t+1}$
end
```

- This algorithm is an adaption of the original RLSVI algorithm of [1] to a continuing task setting where the horizon is not finite.
- Approximates a continuing task by computing the optimal policy assuming a finite lookahead horizon.
- Motivated by model predictive control which performs iterative, finite-horizon planning over short time scales in complex systems.

At each time step, Algorithm 1 learns H policies corresponding to the next H time steps, but Algorithm 2 only uses the policy corresponding to the subsequent time step to choose the next action. Note that the first H time steps of Algorithm 2 use random policies to choose actions due to lack of observed data.

Experiments

Baseline

- Compare with Least-Squares Value Iteration (LSVI) with ε-greedy exploration as our baseline in these experiments

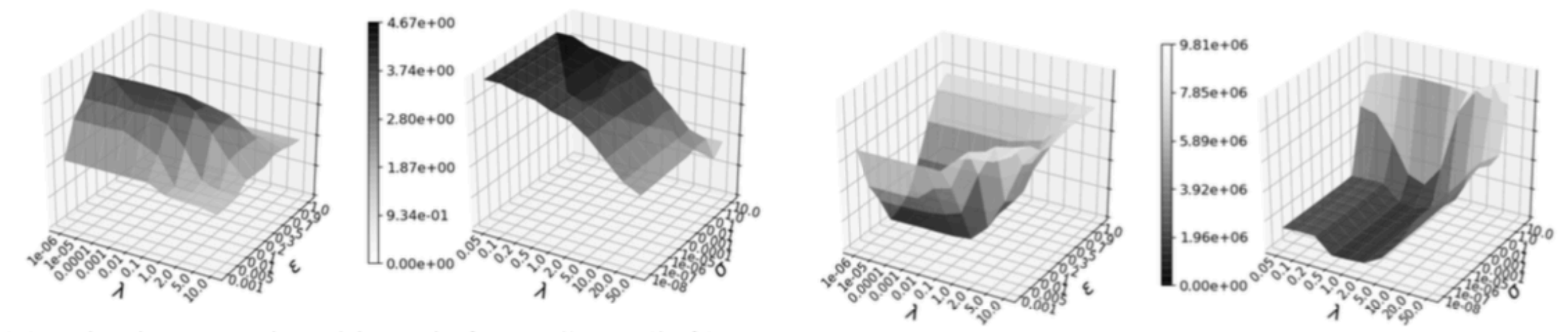
Implementation details

- One-hot vector representations of the state-action space as our feature matrix Φ
- Burden size $d = 3$, number of trials $N = 10$
- Finite horizon: time horizon $H = 16$, reward decay rates $r_0 = 1.5, r_1 = 1.0$, 1000 episodes
- Continuing task: memory window $T = 4$, lookahead horizon $H = 8$, reward decay rates $r_0 = 1.5, r_1 = 1.2$, 10000 time steps, current 0 or 0.3

Evaluation Metrics

- Final reward: the average total reward obtained in last 100 time steps
- MSE: $\left(\bar{r} - r_{opt}\right)^2 + \frac{1}{N}\sum_{n=1}^{N}(r_n - \bar{r})^2$ where N is the number of trials that were performed, r_n is the total reward obtained in the nth trial, $\bar{r}$ is the mean total reward averaged over N trials, and r_{opt} is the maximum reward achievable using the optimal policy.

Results

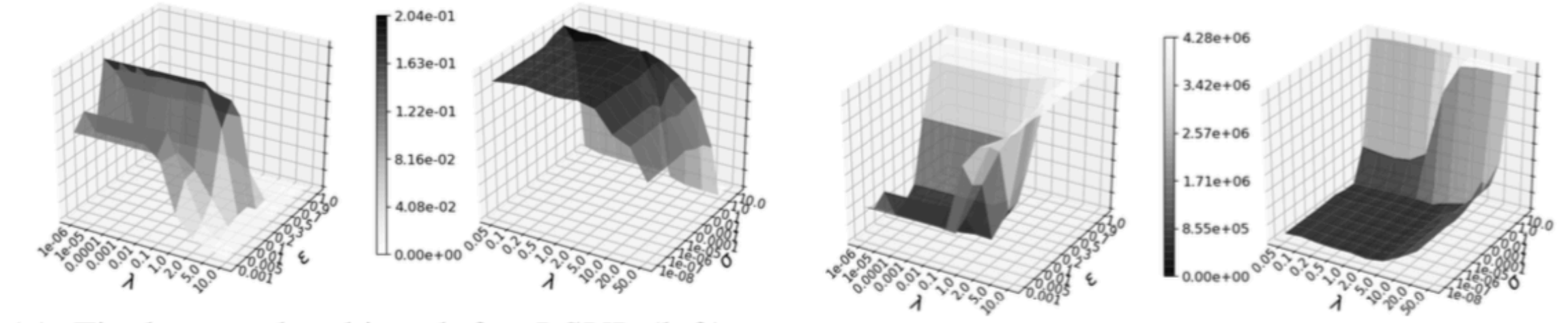


(a) Final reward achieved for LSVI (left) vs. RLSVI (right)

(b) MSE for LSVI (left) vs. RLSVI (right)

Finite horizon, current $c = 0$ (optimal final reward 4.67)

RLSVI	LSVI
Maximum final reward: 4.67	Maximum final reward: 4.28
Minimum MSE: 3.53e+05	Minimum MSE: 3.91e+05

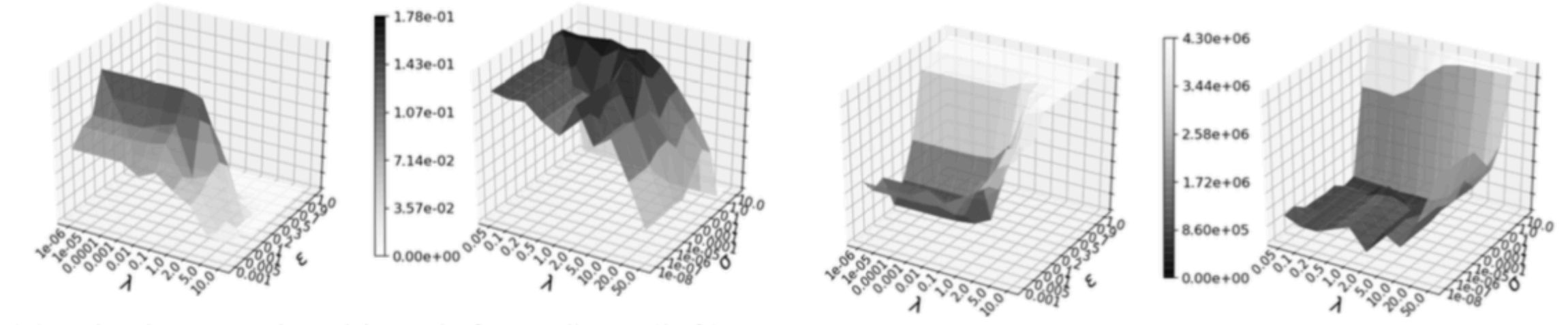


(a) Final reward achieved for LSVI (left) vs. RLSVI (right)

(b) MSE for LSVI (left) vs. RLSVI (right)

Continuing task, current $c = 0$ (optimal final reward 0.2083)

RLSVI	LSVI
Maximum final reward: 0.2041	Maximum final reward: 0.1951
Minimum MSE: 9.35e+04	Minimum MSE: 4.71e+04



(a) Final reward achieved for LSVI (left) vs. RLSVI (right)

(b) MSE for LSVI (left) vs. RLSVI (right)

Continuing task, current $c = 0.3$ (optimal final reward 0.2083)

RLSVI	LSVI
Maximum final reward: 0.1784	Maximum final reward: 0.1585
Minimum MSE: 2.30e+05	Minimum MSE: 4.06e+05

Results (cont.)

- Finite horizon simulations clearly show advantage of RLSVI over LSVI (lower MSE, higher final reward obtained).
- Both finite horizon and continuing task simulations show that RLSVI is more robust than LSVI with ε-greedy across varying hyperparameters.
- For the continuing task simulations with *zero* current, RLSVI attains higher max final reward but does not achieve lower min MSE than the LSVI alternative.
- But for continuing task simulations with *positive* current, RLSVI successfully achieves both higher max final reward and lower min MSE than LSVI alternative.

One problem that we have encountered (not shown here) is that these results generalize poorly to a larger state space. Further analysis shows that this is because, even with the current, our problem formulation somewhat penalizes extensive exploration; the optimal policy moves around near the starting state, and since RLSVI explores the entire state space extensively, this actually translates to low rewards.

- This explains why RLSVI successfully attains a min MSE lower than LSVI for continuing task simulations with *positive* current but not for simulations with *zero* current.

Conclusion

Takeaways

- Overall, these simulations show that our proposed continuing task extension of RLSVI is more effective than LSVI for the purpose of mobile health problems: it efficiently learns a near-optimal policy that achieves higher rewards, lower MSE, and better robustness over varying hyperparameters than LSVI.
- However, although the testbed was made to reflect a simplified mobile health problem, it was unsuccessful at directly testing the need to explore efficiently – this made it so that success was limited and generalized poorly to a larger state space.
- Thus our next step is to test this algorithm on a testbed that will reward efficient exploration, since our ultimate goal is to apply this algorithm to mobile health problems where efficient exploration is necessary.

Future Work

- Since we would like to further investigate our algorithm's ability to explore extensively and efficiently, it may be useful to revert to a bottom-up approach starting with simpler testbeds and sparser rewards, such as a modified chain setting from [1], and increasing in complexity.
- Explicitly trying an approach that requires deep exploration approach such as *deep-sea* in [3] (modified to reflect a continuing task in mobile health) may also be a good testbed on which to test compare our continuing task algorithm with LSVI.

References

[1] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. 2016.

[2] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[3] Ian Osband, Daniel Russo, Zheng Wen, and Benjamin Van Roy. Deep exploration via randomized value functions. *arXiv preprint arXiv:1703.07608*, 2017.