Personalization of health interventions using cluster-based reinforcement learning

el Hassouni, Ali; Hoogendoorn, Mark; van Otterlo, Martijn; Barbaro, Eduardo

published in
PRIMA 2018 Principles and Practice of Multi-Agent Systems
2018

DOI (link to publisher)
10.1007/978-3-030-03098-8_31

document version
Publisher's PDF, also known as Version of record

document license
Article 25fa Dutch Copyright Act

citation for published version (APA)
el Hassouni, A., Hoogendoorn, M., van Otterlo, M., & Barbaro, E. (2018). Personalization of health interventions using cluster-based reinforcement learning. In N. Oren, Y. Sakurai, I. Noda, T. Cao Son, T. Miller, & B. T. Savarimuthu (Eds.), *PRIMA 2018 Principles and Practice of Multi-Agent Systems: 21st International Conference, Tokyo, Japan, October 29-November 2, 2018, Proceedings* (pp. 467-475). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 11224 LNAI). Springer/Verlag. https://doi.org/10.1007/978-3-030-03098-8_31

Personalization of Health Interventions Using Cluster-Based Reinforcement Learning

Ali el Hassouni[1](✉), Mark Hoogendoorn[1], Martijn van Otterlo[2], and Eduardo Barbaro[3]

[1] Department of Computer Science, Vrije Universiteit Amsterdam,
Amsterdam, The Netherlands
`{a.el.hassouni,m.hoogendoorn}@vu.nl`
[2] Department of Artificial Intelligence, Tilburg University,
Tilburg, The Netherlands
`m.vanotterlo@uvt.nl`
[3] Mobiquity Inc., Amsterdam, The Netherlands
`ebarbaro@mobiquityinc.com`

Abstract. Research has shown that personalization of health interventions can contribute to an improved effectiveness. Reinforcement learning algorithms can be used to perform such tailoring. In this paper, we present a cluster-based reinforcement learning approach which learns optimal policies for groups of users. Such an approach can speed up the learning process while still giving a level of personalization. We apply both online and batch learning to learn policies over the clusters and introduce a publicly available simulator which we have developed to evaluate the approach. The results show batch learning significantly outperforms online learning. Furthermore, near-optimal clustering is found which proves to be beneficial in learning significantly better policies compared to learning per user and learning across all users.

Keywords: Reinforcement learning · Personalization · m-Health

1 Introduction

Within the health domain, an ever increasing amount of data originating from a variety of sources is being collected about people's health state and behavior. Smart devices not only allow for the collection of data, but can also be used to provide interventions to users directly. One-size-fits-all solutions, where each user gets the same intervention, have been shown less effective compared to more personalized approaches that tailor interventions to (groups of) users (see e.g. [3]). The data collected from the users can help to establish this personalization.

A challenging aspect of intervention personalization is that success is often not immediately clear and that interventions are composed of *sequences* of actions that should act in harmony, and thus reinforcement learning (RL) (see

T. Miller et al. (Eds.): PRIMA 2018, LNAI 11224, pp. 467–475, 2018.
https://doi.org/10.1007/978-3-030-03098-8_31

e.g. [9]) arises as a very natural solution (cf. [2]). RL typically requires a substantial learning period before a suitable policy is found. In our setting, we do not have a sufficiently long learning period per user. Hence, there is a need to substantially shorten the learning period. To establish this, we can either: (1) start with an existing model (transfer learning, see e.g. [7]) or (2) pool data from multiple users who are, in some way, similar to learn policies (cf. [10]).

In this paper, we present a cluster-based RL algorithm which builds on top of the work done by [10] and test it for a complex health setting using a dedicated simulator we have built. We use K-Medoids clustering with Dynamic Time Warping (DTW) [1] as the distance function to find suitable clusters. We learn policies over the clusters using both an online RL algorithm (Q-learning, cf. [8]) and a batch-algorithm (LSPI. cf. [4]). We compare the cluster-based approach to learning a single policy across all users and learning completely individualized policies. The aforementioned simulation environment generates realistic user data for a health setting. Here, the aim is to coach users towards a more active lifestyle. In comparison with [10], our approach relies on a more sophisticated and complex simulation environment where several types of users are simulated with each their own behavioral profile and personal preferences which allows for highly personalized policies. Furthermore, we apply clustering using a state-of-the-art distance metric to learn optimal policies for clusters of users. Also, the stochasticity in the behavior of users makes the simulation environment a very robust testbed for RL algorithms.

2 Approach

Generally, we want to learn an *intervention strategy* for many types of *users*, without knowing beforehand which types of users exist, how they differ behaviorally, and how they react differently to interventions. We employ RL to optimize our system by *experimenting* with different intervention strategies.

Users and Interventions. Let U be the set of *users*. We see each user $u \in U$ as a *control problem* modeled as a *Markov decision process* [9] $M_u = \langle S_u, I, T_u, R_u \rangle$, where S_u is a finite set of *states* the user u can be in, I is the set of possible interventions (*actions*) for u, $T_u :: S_u \times I \times S_u \rightarrow [0,1]$ is a probabilistic *transition function* over u's states S_u, and $R_u :: S_u \times I \rightarrow \mathbb{R}$ is a *reward function* assigning reward $r = R_u(s_u, i)$ to each state $s_u \in S_u$ and action $i \in I$.

The user's state set S_u consists of the *observable* features of the user state. In general, we cannot observe *all* relevant features of the true underlying user state $s_{\texttt{true}}$ and S_u is therefore restricted to measurable aspects, modeled through a set of *basis functions* over a state $s_u \in S_u$. That is, we use the feature vector representation $\boldsymbol{\phi}(s_u) = \langle \phi_1(s_u), \phi_2(s_u), \ldots, \phi_n(s_u) \rangle^\top$ of the state $s_u \in S$ of user u as representation. If there is no confusion we will use s_u instead of $\boldsymbol{\phi}(s_u)$. The transition function T_u, which determines how a user $u \in U$ moves from state $s_u \in S_u$ to $s'_u \in S_u$ due to action $i \in I$, is not accessible from the viewpoint of the RL algorithm, a natural assumption when dealing with real human users. In

Sect. 3, we do show how we have implemented it for the artificial users in our simulator. The granularity of modeling T_u can be set based on the case at hand, ranging from seconds to hours, denoted by Δt. Finally, the reward function R_u determines the goal of optimization and is explained in more detail in 4.

Every time point a user u is in some state $s_u \in S$, the system chooses an intervention $i \in I$, upon which the user enters a new state s'_u, receiving a reward r. Note that for both the transition and reward function it is unknown whether they can be considered *Markov*, and thus whether the user can be controlled as an MDP. Nevertheless, we assume it is close enough such that we can employ standard RL algorithms. With a state that is Markov we can make predictions of future states using only the current state. Note also that all users share the same state representation, but can differ in R_u and T_u. An alternative strategy would be to *learn* the dynamics of T_u and R_u from experience as in *model-based* RL (e.g. see [6]), but here we focus on learning them *implicitly* by clustering users who are similar in their behavior (and thus T_u and R_u).

Evaluating and Learning Interventions. The goal is to learn intervention strategies, or *policies*, for all users. For any user $u \in U$, $\pi :: S_u \rightarrow I$ specifies the intervention for user u in state s_u. The intervention $i = \pi(s_u)$ will cause user u to transition to a new state s'_u and a reward $r = R_u(s_u, i)$ is obtained, resulting in the *experience* $\langle s_u, i, r, s'_u \rangle$. A sequence of experiences for user u can be compactly represented as $\langle s_u, i, r, s'_u, i', r', s''_u, i'', r'', \ldots \rangle$ and is called a *trace* for user u. For the sake of simplicity we will drop the user subscript if possible. To compare policies, we look at the *expected reward* they receive in the long run, represented by so-called Q-functions. For Q-learning, we optimize the policy using the standard formulation of a Q-learning approach (see e.g. [6]). Note that for all users U together one Q-function is learned. In addition, we use variants of *experience replay* [5] which amounts to performing additional updates by "replaying" experienced traces backwards to propagate rewards quicker. In our setting, we sample the experience pairs in chronological order instead of random. Using disjoint experience pairs would have been the better alternative if the set of traces we learn from was larger.

In our second method, LSPI, we employ the basis function representation $\boldsymbol{\phi}(s)$ of a state and compute a *linear function approximation* of the Q-function, $\hat{Q} = \sum_{j=1}^{k} \phi(s) w_k$, from a batch of experiences E. Here, $\boldsymbol{w} = \langle w_1, \ldots, w_k \rangle$ consists of tunable *weights*. LSPI implements an approximate version of standard *policy iteration* (cf. [6]) by alternating a *policy evaluation step* and a *policy improvement step*. However, due to the linear approximation, the evaluation step can be computed by representing the batch of experiences in matrix form and using them to find an optimal weight vector $\boldsymbol{w}$.

Two Learning Phases. For any given set of users we define two optimization phases. In the first phase (*warm-up*) we employ a default policy $\pi_{\mathtt{def}}$ (see the experimental section for details) to generate traces for each user, and use all experiences of all users to compute $Q^{\pi_{\mathrm{def}}}$. By maximization we obtain a better policy π' that is used at the start of the second phase (*learning*). During this phase we iteratively apply the policy to obtain experiences and update our Q-function (and policy) using either Q-learning or LSPI. In this phase some exploration is used, reducing the amount of exploration ϵ over time.

Cluster-Based Policy Improvement. So far, we have assumed all users belong to one group. Our main hypothesis is that since users have different (but unknown) transition and reward functions, learning one general policy for all users will not be optimal. To remedy this, we add a clustering step after the warm-up phase. We employ K-Medoids clustering and by employing DTW instead of a default Euclidean distance, we can also measure similarity when two users are out of phase. The traces that are used here contain the states and rewards. Matching two traces needs to satisfy constraints: (1) every data point from the trace of the first user has to be matched with at least one data point from the trace of the second user and vice versa, (2) the first (and last) data point from the trace of the first user has to be matched with that of the second user, and (3) the mapping of the data points from the trace of the first user to those of the second user must increase monotonically. We split user traces by day and deploy DTW to calculate the optimal match.

Let U be the set of users targeted in the warm-up phase, and Σ^U the set of all traces generated. Let $\Sigma^{u_{i,m}}$ be user i's experiences during day m, excluding the interventions. The similarity between users u_1 and u_2 is defined:

$$S_{DTW}(u_1, u_2) = \sum_{m=0}^{M} DTW(\Sigma^{u_{1,m}}, \Sigma^{u_{2,m}}) \tag{1}$$

Let the number of resulting clusters be k and $\Sigma_1^U, \ldots, \Sigma_k^U$ be the partitioning of Σ^U, and let $U_1, \ldots U_k$ be the partitioning of U. Instead of utilizing all experiences of U for one Q-function, we now induce a separate Q-function $Q_{\Sigma_i^U}$ (and corresponding policy $\pi_{\Sigma_i^U}$) for each user set U_i based on the traces in Σ_i^U and continue with learning and performance phases for each subgroup individually.

3 Simulator

In our health setting applying RL directly to real users would be prohibited by the number of interaction samples required to learn good strategies. We therefore built a simulator to experiment with algorithmic settings first.

3.1 Schedules

We assume that we have n users in our simulator: $\{u_1, \ldots, u_n\}$, originating from the set U as defined before. Each of these users can conduct one of m activities at each time point ($\{\varphi_1, \ldots, \varphi_m\}$). Time points in our simulator have a discrete step size δt. Let Φ denote the possible values of the activity. Example activities are working, sleeping, working out, and eating breakfast. Each user has a unique activity that is being conducted at a time point $t \in T$ ($activity : A \times T \rightarrow \Phi$). Note that this activity can also be *none*. For each user, a *template* schedule can be specified, which expresses for each activity φ_i: (i) an early and late start time ($early\_start(\varphi_i)$ and $late\_start(\varphi_i)$), (ii) a minimum and maximum duration of the activity ($min\_duration(\varphi_i)$ and $max\_duration(\varphi_i)$), (iii) a standard deviation of the duration of the activity ($sd\_duration(\varphi_i)$), (iv) a probability per day of performing the activity ($p(\varphi_i, day)$), and (v) priorities of other activities over this activity. Using these template schedules, a complete schedule is derived which instantiates activities at each time point, on a per day basis.

3.2 Interventions and Rewards

Besides performing activities during a day, interventions can also be sent to users. In our system, the set of interventions I contains a binary action as {`yes`, `no`}, representing at each decision moment whether the system sends an intervention or not. Acceptance of a message is determined by conditions in the user's profile. If a message is sent at the right time and a gap in the schedule is between t_{plan_min} and $t_{plan\_min} + t_{plan\_duration}$ from the time the message is sent, the activity will be performed. These parameters define a time window in the schedule into which the users will try to fit the desired activity.

4 Experimental Setup

As said, we focus on a health setting where learning a policy as fast as possible (i.e. based on limited experiences) is essential. Within this paper, we aim to answer the following questions: **RQ1** *How do batch and online learning in our simulator setting differ, and how can generalization be employed to speed up learning?*, **RQ2** *Can a cluster-based RL algorithm learn faster compared to (1) learning per individual user or (2) learning across all users at once?*, and **RQ3** *Can we effectively cluster users based on traces of their states and rewards?*

Simulator Setup. In our simulator setup, we aim to improve the amount of physical activity of users. We include several types of users. More specifically, we employ three *prototypical users*, referred to as the *workaholic*, the *sporter* (an avid athlete), and the *retiree*. The simulator itself runs on a fine-grained time scale (δt is one second) while we model T_u at a coarser granularity (Δt is one hour). We include the following activities: *sleep*, *breakfast*, *lunch*, *dinner*, *work*, *workout*. We use three profiles with $n = 33$ agents each from which daily activities are spawned with some level of variability per agent.

The goal of the scenario is to make sure the total work out time meets the guideline for the amount of daily physical activity (30 min per day). Messages can be sent to the user to start working out. The acceptance of the message is dependent on the planning horizon of the user and whether it fits into the schedule where the workaholic needs to know long in advance, the retiree works with a short advance notice and the sporter is right in the middle. In addition, acceptance windows are defined (during lunch for workers (with probability 0.7), outside of lunch for retiree (0.5) and anytime for sporters (0.9)). How long the workout activity will be performed is defined in the profile of the user *Fatigue* plays a role here. Fatigue can build up when working out across multiple days. The value of fatigue is the number of times a user worked out in total during a consecutive number of days where at least one workout per day occurred.

Algorithm Setup. As features for the state (i.e. $\phi(s_u)$) we use: (i) the current time (hours), (ii) the current week day (0-6), (iii) whether the user has already worked out today (binary), (iv) fatigue level (numerical), and (v) which activities were performed in the last hour (six binary features). All these features are realistically observable through sensor information, or inferable.

The reward function R_u consists of three components. If an intervention is sent and the user accepts it, the immediate reward is +1 (otherwise −1). A second reward component is obtained while the user is exercising, where the exact reward value is scaled relative to the length of the exercise (+0 per Δt) and when the user finishes exercising (+10). A third component is related to the fatigue level of the agent at each hour of the day: higher levels result in a small negative reward (−0.1 per unit of fatigue per hour) which *shape* the intervention strategy such that it does not overstimulate the user with exercises.

The first part of a simulation run is a *warm-up* phase of seven days where interventions are driven by a *default policy* which sends one intervention per day to each user at random between 9:00 h and 21:00 h. This allows us to perform exploration and to generate traces for clustering.

The second part of a simulation run is the *learning phase* that lasts for 100 days. Immediately after the start of this phase we update the Q-table and learn LSPI policies using the traces generated during the warm-up phase. During the *learning phase* we perform updates to Q-table once every hour and update the LSPI policies at the end of each day. For Q learning and LSPI we use $\gamma = 0.95$, and $\epsilon = 0.05$ and 0.01 resp. and the learning rate α for Q-learning decreases from an initial 0.2 with 1% every day. These parameters have been set using grid search for γ between 0.85 and 0.95 with step size 0.05, ϵ between 0 and 0.05 with step size 0.05 and α was fixed at 0.2 with a 1% decrease rate every day. For LSPI the maximum number of iterations was set at 20 with a threshold of the change in policy weights as a stopping criterion of 0.00001 and we use a *first win* tie breaking strategy. We initialize the Q-values with a random value between 0 and 1 if the action of the state-action pair is 0 otherwise we initialize the Q-values with a random number between −1 and 0, all to encourage exploration. To speed up the learning we use experience replay. We store the last 250 experiences and use these to update the Q-values.

Setup of Runs. To answer our research questions, we run several simulations. First, we vary the type of RL algorithm: online (Q-learning) and batch learning

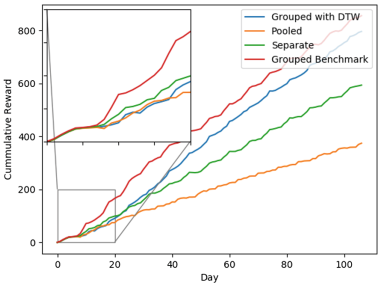

Fig. 1. Cumulative reward LSPI

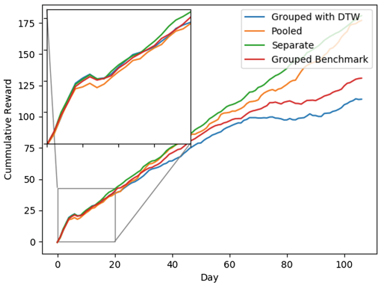

Fig. 2. Cumulative reward Q Learning

(LSPI); this enables us to answer *RQ1*. For each type of algorithm, we compare runs where we learn a single policy across all users (*pooled approach*) to a *cluster-based* approach and learning a completely individualized policy for each user (*separate approach*). This variation reflects *RQ2*. For each algorithm we do two simulation runs for the cluster-based approach; one simulation run using K-Medoids clustering with the DTW distance (*cluster-based approach*) and a second simulation run using three homogeneous clusters, one for each type of agent (*grouped benchmark approach*). The latter provides us with a benchmark to evaluate the cluster quality (i.e. *RQ3*). Hence, in total we perform eight runs.

5 Results

Batch versus Online Learning: Figures 1 and 2 demonstrate that LSPI significantly outperforms Q-learning when we compare the average daily reward over the 100 days. Significance has been tested using a Wilcoxon Signed-Rank test with a significance level of 0.05. The Q-learning experiments show that online (table-based) learning without generalizing over states is not capable of learning reasonable policies in a period of 100 days. LSPI on the other hand, generalizes over states and utilizes the relatively short amount of interaction much better. This is not a surprise, but it does confirm that generalization – over the experiences of multiple agents, but also over states – is needed to obtain reasonable policies in "human-scale" interaction time (and thus answers RQ1).

Different Learning Approaches: The grouped benchmark approach with LSPI provided us with a policy that outperformed all other policies in this setting. The grouped approach using clustering with DTW was the second best performing approach. The separate approach has the ability to match the performance of the grouped benchmark approach given enough time to learn. At the same time the grouped approach clearly outperformed the pooled approach which indicates that clustering helps us learn better policies in a shorter amount of time, by generalizing over the *groups* of agents. With the cluster-based approach we are able to speed up the learning time in comparison with the pooled approach to potentially reach better policies. The policies that were produced by

Q-learning show little variation in terms of performance resulting from the different learning approaches. On the contrary, LSPI produces policies learned using the same approaches that are significantly different among each other (Wilcoxon Signed-Rank test, 0.05 significance). Although Q-learning shows little differences across the setups, an interesting observation is that clustering using knowledge about the profiles of the users performs slightly worse in terms of average daily reward than the remaining approach while using Q-learning.

Clustering: Clustering with the K-Medoids algorithm and the DTW distance metric for LSPI is clearly near-optimal. Two users of the type *retiree* were confused as the type *sporter* and one *sporter* was put together with the *workaholics* in the same cluster. For the Q-learning case similar patterns were observed.

6 Discussion

In this paper, we have introduced steps towards a cluster-based RL approach for personalization of health interventions. Based on the results we can say that: **RQ1:** RL with batch learning and function approximation outperforms table-based RL using online learning in a significant way, thereby disqualifying the latter when interaction time is short. **RQ2:** A cluster-based RL can learn a significantly better policy within 100 days compared to learning per user and learning across all users, provided that a suitable clustering is found. **RQ3:** Learning suitable clusters using the Dynamic Time Warping distance function and K-Medoids clustering based on traces of states and rewards over 7 days shows to be very feasible, resulting in a near perfect clustering. While our simulator exhibits realistic behavior, we plan on moving more and more to a setting where the actual user is in the loop. Furthermore, from a methodological side, we aim to experiment with more powerful RL techniques, and we want to explore different clustering algorithms and more distance metrics to improve the clustering itself.

References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, pp. 359–370 (1994)
2. Hoogendoorn, M., Funk, B.: Machine Learning for the Quantified Self: On the Art of Learning from Sensory Data. Springer, New York City (2017). https://doi.org/10.1007/978-3-319-66308-1
3. Kranzler, H.R., McKay, J.R.: Personalized treatment of alcohol dependence. Curr. Psychiatry Rep. **14**(5), 486–493 (2012)
4. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. J. Mach. Learn. Res. **4**(Dec), 1107–1149 (2003)
5. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Mach. Learn. **8**(3–4), 293–321 (1992)
6. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT press, Cambridge (2017). in progress

7. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: a survey. J. Mach. Learn. Res. **10**(Jul), 1633–1685 (2009)
8. Watkins, C.J., Dayan, P.: Q-learning. Mach. learn. **8**(3–4), 279–292 (1992)
9. Wiering, M., van Otterlo, M.: Reinforcement Learning: State of the Art. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3
10. Zhu, F., Guo, J., Xu, Z., Liao, P., Huang, J.: Group-driven reinforcement learning for personalized mHealth intervention (2017). arXiv preprint arXiv:1708.04001