Nurture: Notifying Users at the Right Time Using Reinforcement Learning

Bo-Jhang Ho
University of California, Los Angeles
Los Angeles, CA 90095, USA
bojhang@ucla.edu

Mehmet Koseoglu
University of California, Los Angeles
Los Angeles, CA 90095, USA
mkoseoglu@ucla.edu

Bharathan Balaji
University of California, Los Angeles
Los Angeles, CA 90095, USA
bbalaji@ucla.edu

Mani Srivastava
University of California, Los Angeles
Los Angeles, CA 90095, USA
mbs@ucla.edu

UbiComp/ISWC'18 Adjunct, October 8–12, 2018, Singapore, Singapore © 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5966-5/18/10 $15.00.
https://doi.org/10.1145/3267305.3274107

Abstract
User interaction is an essential part of many mobile devices such as smartphones and wrist bands. Only by interacting with the user can these devices deliver services, enable proper configurations, and learn user preferences. Push notifications are the primary method used to attract user attention in modern devices. However, these notifications can be ineffective and even irritating if they prompt the user at an inappropriate time. The discontent is exacerbated by the large number of applications that target limited user attention. We propose a reinforcement learning-based personalization technique, called *Nurture*, which automatically identifies the appropriate time to send notifications for a given user context. Through simulations with the crowdsourcing platform Amazon Mechanical Turk, we show that our approach successfully learns user preferences and significantly improves the rate of notification responses.

Author Keywords
Interruptibility; Push notification; User interaction; Reinforcement learning

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction

Many mobile devices depend upon human interaction for numerous functionalities. Crowd-sourcing apps such as Waze[1] and Yelp[2] bank on user input to deliver services; intervention-based apps such as Apple Watch Activity app[3] prompt users to exercise; and apps such as Google Now[4], Medisafe[5] send useful context-based reminders. User engagement with mobile devices is leveraged to learn preferences, label activities and configure devices.

The usefulness of mobile devices diminishes significantly if users do not want to interact with them [9]. For instance, a medical therapy may not work if the patients do not read the intervention messages. Users may not have the time to react if they ignore warnings and alerts preceding a natural disaster. Notifications can also lead to negative effects if they distract the user at the wrong time. For example, interacting with a handheld device while driving increases the crash risk 3.6 times [2].

Most mobile applications currently employ a simplistic interaction model that assumes the user is always available to engage with the device. In reality, user attention is a limited cognitive resource [5]. Users get over 60 notifications a day [12], so it is natural to skip notifications when feeling overwhelmed and to dismiss all communications when disturbances are undesirable[6]. Nevertheless, there is no systematic way to infer user availability yet, and identifying the ideal time for human interaction remains a challenging problem as user engagement depends on a wide range of variables such as context [10], environment [16], hardware status [11], and content of messages [8].

Figure 1: The proposed reinforcement learning agent interacts with the user to learn the right time to send notifications. Nurture aims to empower different applications. In this paper, the user is simulated by underlying models.

Our project is open sourced: https://github.com/nesl/Nurture-UbiTtention18

Here we approach notification timing as a reinforcement learning (RL) problem, and propose a notification time selection technique, *Nurture*, which learns user preferences through interacting with users over time without any prior knowledge. RL has the following advantages over supervised learning methods: (1) RL is an online learning process and does not require a separate data collection phase for model training. (2) RL implicitly tracks the state changes of the user to maximize long term rewards. (3) It proactively explores different actions when unseen observations occur.

Figure 1 illustrates the learning flow. Nurture obtains the user state via sensors, decides if it should notify the user, and observes user reaction after sending the notification. By considering accepted notifications as positive signals and dismissed notifications as negative signals, the agent learns the appropriate times to notify the user. We evaluated Nurture with both a synthetic and an online interactive crowdsourcing-based simulation. Our simulation results show that reinforcement learning improves user response rate compared to supervised learning methods.

Related Work

For mobile applications, push notifications are a convenient way to interact with users. However, it has been shown that push notifications can reduce work performance and increase stress especially if they arrive at inappropriate times. Hence, there is a growing interest on how to make notification systems more intelligent [7].

Prior work on identifying the right moment for notifying users can be categorized to two types:
(i) Show notifications when the user is transitioning between activities. An activity transition usually indicates

[1] Waze - https://www.waze.com/
[2] Yelp - https://www.yelp.com/
[3] Use the Activity app on your Apple Watch - https://support.apple.com/en-us/HT204517
[4] Google Now - https://en.wikipedia.org/wiki/Google_Now
[5] https://medisafe.com/
[6] https://www.wired.com/story/turn-off-your-push-notifications/

RL Agent: Our system (i.e., *Nurture*)

Environment: The edge device user

State: User context (e.g., time, location, activity)

Action: *Send a notification* or *remain silent*

Reward: User's reaction of handling notifications

Figure 2: The entities in our reinforcement learning setup.

Category	Values
Time of the day	*morning, afternoon, evening*
Day of the week	*weekday, weekend*
Location	*home, work, others*
Motion activity	*stationary, walking, running, driving*
Last notification	*within / beyond 1 hour*

Table 1: The state categorical values. Each state is presented by the combination of these five categories, so there are 144 different states in total.

that the former activity is completed, hence, it is a good time to interrupt users before they start another task. Oasis [4] exploits breakpoints between computer tasks to interrupt users. Attelia [10] detects physical activity transitions using smartphones and wristbands, and identifies which transitions indicate user availability. The downside of this approach is that the changes in sensor readings do not always reflect activity transitions, hence, high false positive rate becomes a concern.

(ii) Infer user interruptibility from context. Sarker et al. [13] show that location, activity type, stress, time, and day of the week affect user participation in medical interventions. Goyal et al. [3] show that users are likely to pay attention to the notifications at times of increasing arousal detected from the electro-dermal activity. Aminikhanghahi et al. [1] proposed a combined activity recognition and intelligent notification framework based on supervised learning which improves notification response rate. PrefMiner [6] correlates the phone context and notification responses, and makes recommendations to users for managing their notifications using learned rules.

Prior works have primarily focused on supervised learning and manual labeling to understand the relationship between the user state and notification response. Here we approach the problem as a reinforcement learning problem where the agent learns by *interacting* with the user and observing the context of the interaction.

Design and Implementation

Problem Setup

In reinforcement learning (RL), an *agent* interacts with an *environment* to achieve certain goals. At each step, the environment is in a certain *state* and the agent takes an *action*. The environment reacts to the action by transitioning into another state and returns a *reward* to the agent. The strategy with which the agent selects its actions is called its *policy*. The RL agent tries to find a good policy by trying different actions which maximizes the cumulative reward.

Figure 2 summarizes the mapping between our setup and RL terminology. **Nurture** is the RL *agent* running in a mobile device, and the *environment* is the **user**. Periodically (e.g. 10 minutes), Nurture senses the user *state* based on **sensor readings**. The user context we have considered are time, location, physical activity, and last notification usage, summarized in Table 1. Nurture performs one of the following *actions* - **remain silent** or **send a notification** to the user. The *reward* is based on the reaction of the user: a positive reward is received if the user responds to the notification, a negative reward is received if the notification is dismissed, and zero reward is received if the user ignores it. We consider a relatively small state space as a proof of concept. However, the learning agent can be scaled up by introducing more dimensions, e.g., notification type.

Contextual Bandits

Bandits are a special case of RL where the agent does not keep track of sequence of states visited. In a bandit problem, the agent can choose from a set of actions with unknown rewards. Similar to RL, the agent maximizes cumulative reward over successive iterations. The bandit agent explores the rewards associated with each action (i.e, arm) and aims to learn the best action. Contextual bandit is a type of bandit where the agent receives observations as *context*, and the context is used to learn the state in which the environment is in. In our setup, the two arms of the bandit are (i) to send a notification, or (ii) to remain silent. The context is same as the state listed above. The reward from the first arm depends on the response from the user and the second arm always gets zero rewards.

Sample survey question:
It is 10:30 AM on Saturday. You're sitting in a shopping mall. You responded to a notification 1 hours 15 minutes ago. Now you receive a notification from our app to complete a 10-second task. What action will you take?

Available responses:
a) Dismiss this notification
b) Leave the notification and answer it later
c) Take ten seconds to respond the notification

Figure 3: A sample of survey question and available response choices.

Attributes/Routings	User1	User2	User3	User4
Staying workplace	2.81	2.96	9.74	3.89
Staying outdoor	2.76	7.29	3.80	6.96
Walking+running	1.32	0.54	1.57	1.34
Driving	0.37	1.20	0.43	2.05

Table 2: Statistics for four users, showing the number of hours each user spends at each location and activity

Reinforcement Learning
We also consider our problem as a Markov Decision Process (MDP) which we solve using Q-learning. Different from the contextual bandit, the Q-learning implicitly learns transitions between states.

The expected long term cumulative reward for taking an action from a given state under a specific policy is referred to as *Q-value*. In Q-learning [15], the agent updates the Q-value based on rewards received as it visits different states using a random exploration policy. Over time, the Q-value converges to its optimum value, and we get the policy by picking the actions with the highest Q-value at each state.

Experimental Setup
We conducted a simulation-based experiment using crowdsourced data as a proof-of-concept study. We do not claim that the crowdsourced data can represent real user data. Instead, we consider the crowdsourced data as a basis to compare our proposed algorithms against supervised learning algorithms. Our experimental procedure is approved by UCLA IRB. The simulation process is depicted in Figure 1. In our simulator, we imitate a user in her daily routine. The mobile application senses the user context through her mobile phone and wearables, and Nurture decides whether to send a notification or to remain silent. Once our agent sends a notification, the simulated user can respond to the notification, explicitly dismiss the notification, or perform no action to skip it. Nurture then obtains the reward according to the user's reaction and adjusts the strategy.

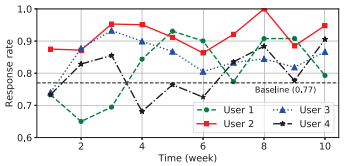
A simulated user is driven by a *behavior model* and a *response model*. The behavior model reflects the daily routine of the user, i.e., the location and the activity of the user during a week. We derived the behavior model from the ExtraSensory dataset [14], which includes daily traces of 60 participants for improving context recognition in-the-wild. The dataset consists of 29 locations and 15 activity contextual labels, and the participants marked the applicable labels at a granularity of one minute. We group these labels to fit our simulation settings and pick four user traces which have distinct lifestyles. Important statistics regarding the weekly routines of these users are summarized in Table 2. The response model simulates how a user *responds* to a notification at a given context, and the user context is determined by the behavior model.

Each question in the mTurk survey describes a scenario that specifies the time of the day, day of the week, location and activity of the user, and when the last notification was responded to. We provide a sample survey question and the possible response choices in Figure 3.

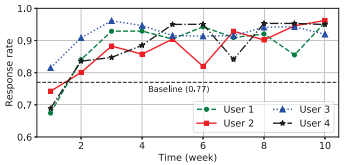
We performed two different types of simulations to imitate human responses: a synthetic simulation and an online interactive simulation. In the synthetic simulation, we deployed a survey on the Amazon Mechanical Turk[7] (MTurk) prior to running Nurture to collect data on how users will respond to notifications given different context. The user response is determined by looking up the survey response describing the same situation.

In the online interactive simulation, we do not collect data a priori. Nurture interactively sends notifications in the form of an MTurk survey to model a "pseudo-user". Depending on the responses collected in an iteration, Nurture updates the notification scheduling policy either using the contextual bandit or the Q-learning algorithm. The updated policy is used to generate a new mTurk survey so as to maximize the likelihood of positive response from the workers. We create two pseudo-user profiles where each user has a different daily routine, and we collect each of their responses

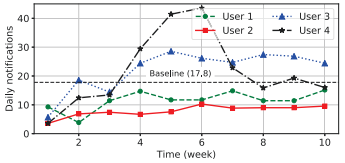
[7] Amazon Mechanical Turk - https://www.mturk.com/



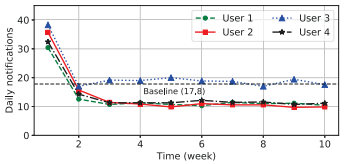
(a) Response rate of contextual bandit



(b) Response rate of Q-learning



(c) Number of daily notifications of contextual bandit



(d) Number of daily notifications of Q-learning

Figure 4: The performance of contextual bandit and Q-learning algorithms over weeks, compared with SVM as our baseline.

from a different region, e.g., the U.S. vs India.

Evaluation

We first performed synthetic simulation in which the human responses are approximated. This gives us the advantage of repeatedly and systematically iterating over our algorithms. We then tested our algorithm on online interactive simulation where the user responses are collected in the wild over a crowdsourced platform. This gives us insight on whether our agents can adapt to user preferences over time.

Synthetic Simulation Results

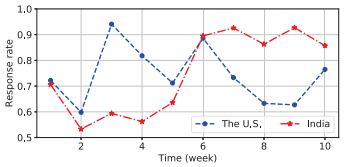
We collected 3,019 survey responses from MTurk across 123 workers. 44.1% notifications were accepted, 26.6% were dismissed, and the rest were marked as 'answer it later'.

We use the *response rate* and *notification volume* as performance metrics. Response rate is defined as the fraction of accepted notifications over notifications sent without considering those marked as skipped. High response rate implies our agent can accurately identify when to approach users. However, an agent may increase the response rate by avoiding interaction with users. Thus, we use number of notifications to balance this effect. A well-behaved agent should keep a high response rate while maintain a high notification volume.

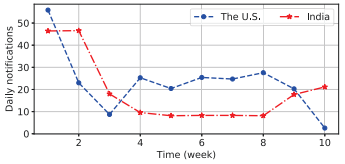
We consider the following two supervised algorithms as our baseline: Support Vector Machine (SVM) with Radial Basis Function kernel, and 2-layer Neural Network (NN) with 32 neurons in each layer. The models apply one-hot encoding to represent contextual user data (i.e. the user state shown in Table 1) as a feature vector, and an anticipated action (i.e. user response) as the predicted label. We created training and testing schedules that last 4 weeks and 6 weeks, respectively. During the training phase, each algorithm randomly sends 15 notifications a day, and obtains the responses from the simulated user. During the testing phase, both algorithms predict an action every 10 minutes to determine whether to send a notification or not. We apply the baseline algorithms to the four users weekly routines as described in Expermental Setup Section. SVM and NN achieve a mean response rate of 77.4% and 77.1% each, and send an average of 17.8 and 6.7 notifications per day, respectively. We choose SVM as our baseline for the rest of the paper.

Figure 4 shows the performance of Nurture. We apply contextual bandit and Q-learning algorithms on the four different weekly routings. We found that these two algorithms take opposite strategies: Contextual bandit starts conservatively, whereas Q-learning sends more notifications in the first two weeks. Both algorithms are able to achieve higher response rate over time, and converge at 89.6% and 93.8% response rate, respectively. Moreover, both algorithms demonstrate that they can achieve better response rate compared against the baseline in less than 4 weeks, which is the length of the training period of our baseline algorithms. Q-learning is superior because it considers the state transition, whereas contextual bandit does not consider pursuing future rewards.

It should be noted that Q-learning converged earlier than the end of the training periods of the benchmark supervised learning algorithms and achieved better performance. Since random notifications are sent during the training period of the supervised learning algorithms, the user experience is subpar during training. On the other hand, Q-learning exploits what it had learn while it continues to explore the state space. Hence the user experience improves as soon



(a) Response rate



(b) Number of daily notifications

Figure 5: Performance of interacting with MTurk workers using Q-learning.

as the Q-learning algorithm sufficiently learns the user preferences.

Moreover, the duration of the training period of supervised learning benchmarks has to be decided a priori for all users. However, a global selection for all users may not be optimal for each user. On the other hand, online learning approaches adapts for each user.

Online Interactive Simulation Results

To demonstrate Nurture can interact with real users and adapt to their preferences, we update our model online based on the crowdsourced responses from MTurk. We choose Q-learning algorithm because it performed better in the offline case. We started two simulations which obtain notifications from the U.S. and India. We removed the responses from abusive workers that take unreasonably short amount of time to finish the survey or only enter a single option. At the end, we recruited 223 workers from the U.S. and 67 workers from India that collectively completed this experiment.

The simulation result is shown in Figure 5. In the first two weeks, Q-learning aggressively sends notifications to explore user preferences under different contexts. The development of the two simulations deviate there onward. In the simulation of India, since Nurture cannot find the opportune moments to interrupt users, it becomes conservative but continues to learn, and finally improves the response rate after week 6. In contrast, in the simulation of the U.S., Nurture converged to a high response rate on week 3, and reaches out to the user more often. The agent then realizes the user starts to show disagreement with the notification schedule, and adjusts the strategy to carefully choose when to approach the user. As we can see, the response rate increases in the end after learning from its mistakes.

Limitations and Future Work

In our experiments, we have used crowdsourced data to emulate user response to approximate what may happen in a real deployment. Although we have only considered user context that can be inferred from mobile phone sensors, our proposal can make use of other sensing modalities. For example, acoustic sensors such as Alexa can be a rich channel to infer user emotion, or electro-dermal activity from a watch can be used to detect arousal or stress [3], which are closely related to user availability. Additionally, instead of treating all notifications uniformly, Nurture can consider the importance of the message by manipulating the reward associated with each notification, e.g., a large reward for a high priority message. Building upon these preliminary experiments, we plan to do a human subject study to evaluate the performance of the proposed algorithms.

Conclusion

We have designed a reinforcement learning based algorithm to improve notification response rate, which in turn increases the quality of interactions from mobile devices and reduces disturbance. Our algorithm considers user context to optimize user engagement via push notifications. We have conducted experiments using crowdsourced data to evaluate the performance of our algorithm. The experimental results show that our algorithm improves the notification response rate significantly with respect to a supervised learning benchmark, while balancing the amount of notifications. Therefore, our proposed approach improves the quality of interaction between the user and the applications running on mobile devices by providing more timely notifications.

Acknowledgement:
Mehmet Koseoglu is an assistant professor in the Department of Computer Engineering, Hacettepe University, Turkey and his visit to UCLA is supported by the Fulbright Program with grant number FY-2017-TR-PD-02. This research is funded in part by the National Science Foundation under awards CNS-1640813, and by the National Institutes of Health under awards #U154EB020404. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, or the U.S. Government.

REFERENCES

1. S. Aminikhanghahi, R. Fallahzadeh, M. Sawyer, D. J. Cook, and L. B. Holder. 2017. Thyme: Improving Smartphone Prompt Timing Through Activity Awareness. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 315–322.
2. Thomas A. Dingus, Feng Guo, Suzie Lee, Jonathan F. Antin, Miguel Perez, Mindy Buchanan-King, and Jonathan Hankey. 2016. Driver crash risk factors and prevalence evaluation using naturalistic driving data. *Proceedings of the National Academy of Sciences* 113, 10 (2016), 2636–2641.
3. Nitesh Goyal and Susan R Fussell. 2017. Intelligent Interruption Management using Electro Dermal Activity based Physiological Sensor for Collaborative Sensemaking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 52.
4. Shamsi T Iqbal and Brian P Bailey. 2010. Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks. *ACM Transactions on Computer-Human Interaction (TOCHI)* 17, 4 (2010), 15.
5. Kyungmin Lee, Jason Flinn, and Brian Noble. 2015. The case for operating system management of user attention. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 111–116.
6. Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. 2016. PrefMiner: mining user's preferences for intelligent mobile notification management. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1223–1234.
7. Abhinav Mehrotra and Mirco Musolesi. 2017. Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges. *CoRR* abs/1711.10171 (2017).
8. Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. 2015. Designing content-driven intelligent notification mechanisms for mobile applications. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 813–824.
9. Daniel Moldovan, Georgiana Copil, and Schahram Dustdar. 2018. Elastic systems: Towards cyber-physical ecosystems of people, processes, and things. *Computer Standards & Interfaces* 57 (2018), 76 – 82.
10. Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K Dey, and Hideyuki Tokuda. 2015. Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 475–486.
11. Tadashi Okoshi, Kota Tsubouchi, Masaya Taji, Takanori Ichikawa, and Hideyuki Tokuda. 2017. Attention and engagement-awareness in the wild: A large-scale study with adaptive notifications. In *Pervasive Computing and Communications (PerCom), 2017 IEEE International Conference on*. IEEE, 100–110.
12. Martin Pielot, Karen Church, and Rodrigo De Oliveira. 2014. An in-situ study of mobile phone notifications. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 233–242.

13. Hillol Sarker, Moushumi Sharmin, Amin Ahsan Ali, Md Mahbubur Rahman, Rummana Bari, Syed Monowar Hossain, and Santosh Kumar. 2014. Assessing the availability of users to engage in just-in-time intervention in the natural environment. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 909–920.
14. Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. 2017. Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches. *IEEE Pervasive Computing* 16, 4 (2017), 62–74.
15. Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (01 May 1992), 279–292.
16. Dominik Weber, Alexandra Voit, Philipp Kratzer, and Niels Henze. 2016. In-situ investigation of notifications in multi-device environments. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1259–1264.