Personalization for Web-based Services using Offline Reinforcement Learning

Pavlos A. Apostolopoulos, Zehui Wang, Hanson Wang, Chad Zhou, Kittipat Virochsiri
Norm Zhou, Igor L. Markov
Meta, Menlo Park, CA
{pavlosapost, wzehui, hansonw, yuzhoubrother, kittipat, nzhou, imarkov}@fb.com

Abstract

Large-scale Web-based services present opportunities for improving UI policies based on observed user interactions. We address challenges of learning such policies through model-free offline Reinforcement Learning (RL) with off-policy training on logged data. Deployed in a production system for user authentication in a major social network, it significantly improves long-term objectives. We articulate practical challenges, compare several ML techniques, provide insights on training and evaluation of offline RL models, and discuss generalizations.

1 Introduction

For Web-based services with numerous customers, such as social networks, UI decisions impact top-line metrics, such as user engagement, costs and revenues. Supervised Learning methods have shown promising results [24, 4] on optimizing such decisions, but they mainly focus on immediate metrics such as the click-through rate [12] and conversion rate [16]. However, long-term cumulative objectives make it challenging to label each decision for training.

As an illustration, consider online user authentication for a Web-based service, routinely performed many times per day/month. When a user mistypes or forgets her password, the service can loop back to the login prompt or offer another login channel, authorization code via SMS (Short Message Service), etc. The scale of the social network makes monetary cost significant. Some users get their password right on the second try or look it up but others give up, and this is reflected in daily/monthly user engagement. Deciding when to authenticate requires real-time personalization with consideration for long-term cost and engagement metrics (*daily/monthly active users* and monetary cost).

Reinforcement Learning (RL) [21] seeks an optimal policy to maximize a long-term reward, and thus helps drive real-time personalized decisions [22]. In RL research, it is common to assume access to direct *online* interactions with the environment for the RL agent. Here, the RL agent's *environment* is an individual online user, and the *personalized serving procedure* is modeled via sequential agent-environment interactions. To this end, training RL agents online may undermine user experience and/or incur large costs.

In this work, we use Offline RL to improve personalized authentication for a Web-based service. We avoid the pitfalls of online trial-and-error by training on prior logged experiences through *off-policy learning*. After having given careful consideration to the non-sequential formulation of the problem with Supervised Learning, we formalize the problem as a Markov decision process (MDP) [18], where the RL agent learns a personalized policy, i.e., when to send an authentication message to the user after a failed login attempt. The training process optimizes long-term user engagement and the service's authentication costs. We avoid making decisions based on a majority of action sequences, as in our application monetary costs are incurred by a minority of sequences. To ensure effective offline training, we use several training heuristics, then evaluate performance of the learned candidate

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021

policies via an *unbiased off-policy* estimator. The best learned policy is chosen, and the ϵ-greedy version of it with a non-zero exploration level for future training purposes, is evaluated in online A/B tests [7] on live data. In addition to comparisons to Supervised Learning, we share insights into modeling and offline training, and detail online evaluation to help practitioners train Reinforcement Learning agents on logged data.

2 Offline Reinforcement Learning

A common deep learning variant of Q-learning [26] called DQN [14] utilizes a replay buffer of the form $\mathcal{D} = \{(s_t, a_t, s_{t+1}, r_{t+1})\}$ for storing the agent's interaction and alternates between data collection and gradient steps with respect to the Temporal Difference loss function $\mathcal{L}_i(\theta_i)$ at training iteration i:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s_t, a_t \sim D}\left[(y_i - Q(s_t, a_t; \theta_i))^2\right] \tag{1}$$

$y_i = \mathbb{E}_{s_{t+1} \sim D}\left[r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a; \theta_i^-))\right]$ are evaluated through a target network with parameters θ_i^-. This learning procedure is termed *off-policy*, as y_i is computed without knowing the policy that generates the experiences in the replay buffer $\mathcal{D}$.

Offline RL relies on *off-policy* learning that uses state-action transitions batch-sampled from the *training set* $\mathcal{D}_{\pi_\beta}$. Although *off-policy* learning algorithms can be directly used offline, in the regular RL setting the *online* regime helps the agent refine its $Q(s_t, a_t; \theta)$ estimates, whereas in offline RL, the lack of exploration limits agent's learning [9] due to *distributional shift* phenomena.

State distributional shift [9] affects offline RL algorithms during the agent's exposure on the real-world environment (deployment phase), as the agent's learned policy π^* can diverge from the behavior policy π_β. The latter could invoke unreasonable actions in out-of-distribution/unseen states.

Action distributional shift [27] affects *off-policy* learning algorithms during the training process as well, as the accuracy of the regression in Equation 1 depends on the estimate of the Q-function for actions that may be outside of the distribution of actions that the Q-function was ever trained.

In this work, a randomized behavioral policy π_β that enables high coverage of the state-action space, is used for a short period of time for the collection of the training set $\mathcal{D}_{\pi_\beta}$. KL-divergence is utilized for distributional stability during training (Section 6.1). The latter constitute our two essential ways for mitigating distributional shift. To guide our design decisions for the application, we firstly introduce a simplified, intuitive problem environment. We use this environment to illustrate important insights on *distributional shift* in Offline RL, and the impact of behavioral policy's exploration on the quality of trained models.

3 Application: User Authentication

User authentication starts a typical Web-based session; failures can delay successful login. Due to forgotten passwords, such failures are common when different passwords are used for each Web-based service. The authentication UI may respond to failures by offering password recovery (`Action A1`) or by prompting another login attempt (`Action A2`), see Figure 1. Password recovery verifies the user's identity by sending a code to a pre-registered mobile phone or another user-owned device. Common options include One-Time Password (OTP) sent via SMS and email, and Time-based One-Time Password (TOTP) authentication [11].

Such authentication is considered safe unless the user's device is compromised or the phone number is hijacked [20].[1] In our application, we authenticate via OTP.

The decision mechanism that chooses between available actions may affect long-term objectives, e.g., user engagement (daily/monthly active users). `Action A1` typically succeeds and improves user engagement but incurs OTP costs, whereas single or multiple invokes of `Action A2` may succeed at some point, and without such costs, while and if it fails again, `Action A1` can still be preferred over `Action A2` at any time. Thus, the decision mechanism should be optimized toward tradeoffs of user

[1] Such authentication can also support two-factor authentication (2FA) [1, 19].

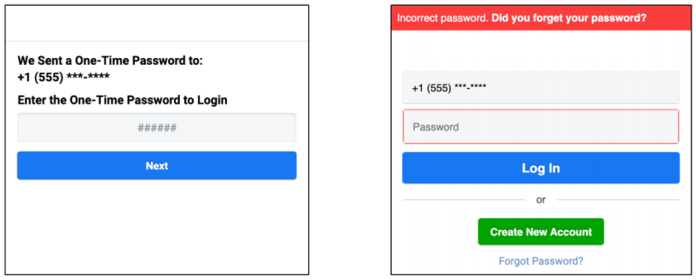


Figure 1: User authentication UI.

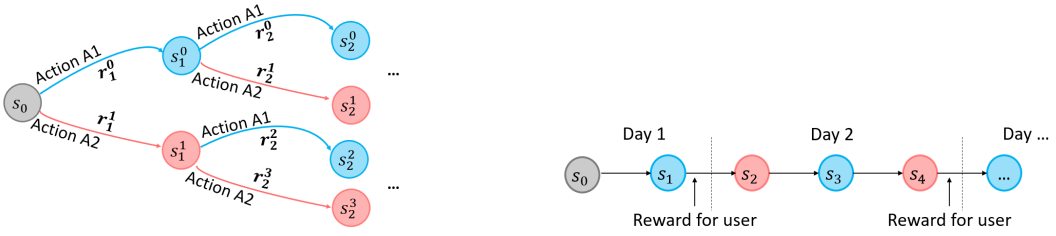
engagement and cost. The described user authentication could be appropriately formalized either as a sequential or non-sequential setting.

The former corresponds to the setting of a regular contextual bandit problem [13], where online learning may be critical to balance user engagement and OTP costs. Section 6.2 explores the non-sequential option with a reward predictor in the form of Supervised Learning, which is trained offline and drives the decision mechanism.

4 Problem formalization

Consider the motivational example of the user-authentication application with two possible actions at a time. Figure 2(a) illustrates possible system actions from the initial failed-login state and subsequent states, as well as implied state transitions, which depend entirely on the actions chosen by the system. The sequence can continue upon a future login failure, as successful login and/or day(s) without failed login attempts do not interrupt the sequence. Each sequence stops as per a defined time-out window (7days). This self-contained state-reward model can capture the most basic sequential preferences but lacks user personalization, which is key to our work.

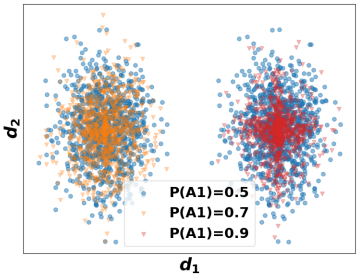
A user can be characterized by a number of personalized features, and the semantics of them are not important for our motivational example. For each user we represent the initial state s_0 (the first failed login attempt) with a state vector drew from a multivariate Gaussian distribution with mean and covariance selected randomly per user. Subsequent user states s_t with $t = 1, \ldots$ that correspond to future failed login attempts are generated randomly and as part of state transitions. Specifically, a base (user) feature vector is drawn from a multivariate Gaussian that is defined per user, per discrete time step t, and per action a_t that was prompted by the authentication UI on user state s_{t-1}. This base vector and the vector representing user state s_{t-1} are then aggregated to construct the vector for the next user state s_t. The user engagement and corresponding action costs on each user transition are represented by a single normalized scalar value-reward r_{t+1}. The latter is drawn from a normal distribution with randomly selected mean and variance, defined per user, per time step, per action. The simulated environment is used as a testbed in Section 5.



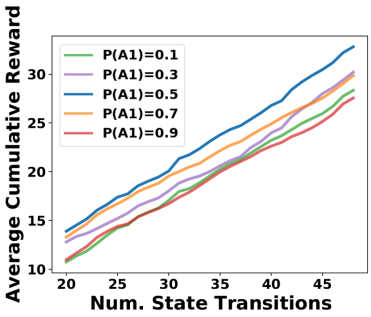
(a) State transitions indexed by depth (subscripts) and actions (superscripts).

(b) Rewards are set daily per user, to the last action.

Figure 2



(a) State features are projected to two principal components via PCA. The blue pointset is cloned.



(b) Average cumulative rewards over trajectories of different length.

Figure 3: Exploration diversity impacts state coverage and training. Between `Actions A1` and `A2`, the former is chosen with probability P(`A1`). The greatest state coverage, and best results are observed for balanced exploration P(`A1`)=0.5.

4.1 Our practical state model and rewards

The problem formalization we use in practice differs from the motivational example in the following two aspects.

User personalized features include *real-time (contextual)*, and *historical* features. For example, on-device real-time features capture the app platform, local time of day, day of the week, login time, time since last login, etc. *Historical* features highlight the impact of the decision mechanism on the sequential depth of the problem e.g., password recoveries sent in the past, the number of authentication attempts over 30 days and their outcomes, how many times each of `Action A1` and `Action A2` were used. In principle, a user's MDP sequence can have an infinite horizon as they interact with our agent (beyond a successful login), but in practice we collect data in limited time windows. Hence, all sequences are cut into pieces up to seven days.

Rewards are formulated to help optimize long-term objectives that are affected by the authentication decision mechanism, i.e., *user engagement* and *monetary cost*. In our Web-based service, user engagement is reported and expressed as a binary value that determines if the user was active during the day. Monetary cost represents the amount of money spent on password recovery after failed login attempts, including `Action A1` and password recovery options invoked by users when they give up on mainline UI. Different personalized policies of the authentication UI affect user engagement and monetary cost, e.g., sequences of `Action A2` may postpone a successful login while avoiding OTP costs. Our goal is to help users login and thus improve user engagement, while reducing costs. Hence, we formulate the reward as a linear combination of user engagement (UE) and monetary cost (Cost):

$$r = w_u \cdot \text{UE} - w_c \cdot \text{Cost}, \text{ where } w_u, w_c \in \mathbb{R} \tag{2}$$

Varying the weights helps explore the Pareto curve of multi-objective tradeoffs for the application. We assign rewards to the last action on each day (Figure 2(b)) because system metrics for user engagement are tallied at the end of each day. For example, on a given day, a nonzero *user engagement* reward is assigned to a user who was exposed to the authentication UI and logged in successfully. Many users recall or recover their password quickly, thus most action sequences are short and/or without OTP costs. Costs can only be improved for longer sequences, a small fraction of all the sequences.

5 Applying Reinforcement Learning

The impact of the exploration level of behavioral policies in offline RL is illustrated in Figure 3 that is produced for the motivational example from Section 4. Here we compare behavioral exploration policies defined by the complementary probabilities of `Actions A1` and `A2`. We train an offline RL

model (DQN) on the induced trajectories for each behavioral policy. We observe in our simulation that balanced exploration with equal probabilities provide better state coverage. In the Figure 3(a), we project state vectors from the feature space onto two principal components via PCA. The yellow (70:30) and red (90:10) point clouds are narrower than the blue (50:50) point cloud.

Figure 3(b) shows that exploration diversity in the behavioral policy helps train a better model, as we can expect higher perceived returns (cumulative rewards) for the RL model trained offline with balanced exploration. Such evaluation is performed on users not seen in the training dataset and by using the trained RL model as a decision mechanism that drives the state transitions. The impact of behavioral policies' exploration is related to the *distributional shift* phenomenon discussed in Section 2. The higher the state coverage in the training dataset, the less significant the state distributional shift during evaluation and action distributional shift during training for an Offline RL model.

Modeling *forward sequential depth* (i.e., the future impact of current actions) is difficult for supervised learning (SL). Modeling *backward sequential depth* (the impact of past actions) is somewhat easier, using additional features that summarize past states and actions.

5.1 RL decision models

Given the large state space in our application, we use DQN and perform offline training on the static dataset $\mathcal{D}_{\pi_\beta}$ Our implementation (Section 6) makes it easy to try more sophisticated RL models. We have tried several improvements to DQN (such as DDQN, Dueling DQN) and concluded that DQN does not leave much room for improvement despite being simpler. However, more techniques that parameterize both the policy and the Q-function are generally worth trying. Whether or not this improves long-term objectives is unclear *a priori*, and we therefore implement a recent technique in this category.

Critic-Regularized Regression (CRR) [25] is an *off-policy* method that discourages sampling low-quality actions from the training dataset. As it is typical for actor-critic algorithms, CRR parameterizes both the policy and the Q-function [8]. It additionally transforms Q-values in the objective with a monotonically increasing function, such as $\exp(\cdot)$, to emphasize higher Q-values. This way, during policy update, π more often samples high-quality actions within the training distribution. The critic is trained using distributional Q-learning [25, 2]. The distributional representation of the returns translates well into stochastic behavioral policies.

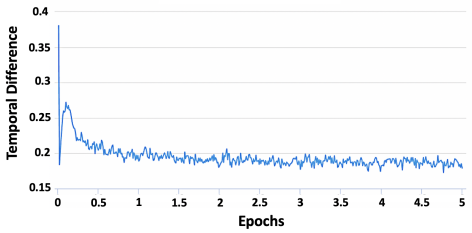
5.2 Preparing data for training

Each row of our training data includes user features, actual system actions, and rewards. Motivated by the impact of the exploration level of behavioral policies in Offline RL (Section 5) we choose equal probabilities for `Actions A1` and `A2`. Thus, upon a failed login attempt, an action is chosen at random, and another handler starts extracting and computing user features simultaneously. The two asynchronous events are tagged with the same unique ID and joined by a stream processing system for logging in the training table. The user engagement metric and the sum of OTP fees are computed by the end of the day. These rewards are joined with the training table based on an anonymized user ID column, where rewards are attributed to the last login failure event of the day (see Figure 2(b)). We logged data using the described exploratory behavioral policy for a time period of three weeks.

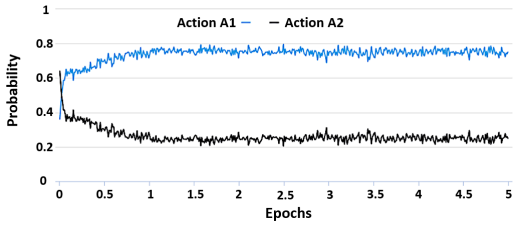
Following the standard Markov Decision Process (MDP) framework, RL models are trained on consecutive pairs of state/action tuples that correspond to state transitions in user sequences (Figure 2(a)). We use the open-source applied RL platform *ReAgent* (Horizon) [3] to prepare logged time series data.

6 Empirical evaluation

To evaluate ML techniques from Section 5, we first assess the quality of the RL models trained offline, and tune their hyperparameters based on offline metrics. The best seen RL model is deployed online on live data, and compared to a baseline SL production system.



(a) Temporal Difference loss with respect to learning epochs for the DQN RL model (Section 5.1)



(b) Action distributions for `Actions A1, A2` over learning epochs for the DQN RL model (Section 5.1)

Figure 4: Off-policy learning

6.1 Offline training and evaluation for RL

To support decision models from Section 5.1, we train our DQN model using the ReAgent (Horizon) platform [3] and its default NN architecture. Equation 2 is used for the reward function.

Offline RL training minimizes *Temporal Difference* (TD) loss (Equation 1) over the provided static dataset of transitions (Section 5.2). Figure 4(a) shows how loss values for the DQN RL model (Section 5.1) change over training epochs that consist of mini-batches (iterations).

To avoid overfitting, we split the static dataset of transitions into a training and test set, then compare the average Q-values of the same `Actions A1, A2` between the training and test sets, and keep the differences below 10%. To check for distributional shifts (Section 2) and poor training, we focus on training quality metrics.

- *The TD Ratio* addresses a pitfall in using raw TD loss (Equation 1), that is sensitive to the magnitude of Q-values. A poor model will exhibit low TD values when Q-values are low. Instead, we evaluate $\frac{\text{TD}}{\min\{\bar{Q}_{\texttt{A1}}, \bar{Q}_{\texttt{A2}}\}}$, where $\bar{Q}_{\texttt{A1}}, \bar{Q}_{\texttt{A2}}$ are the average Q-values of `Actions A1, A2` over the training dataset.
- *Distributional stability* during training is vital to effective learning. The action distribution of the trained offline RL model should not shift too far from the behavioral policy, which we measure as the KL-divergence [5]. The action distribution of our DQN model in Figure 4(b) stabilizes over training. The change of the action distribution over a time window (25 training iterations) is a good proxy for the distributional stability for offline RL models.

Offline evaluation avoids the dangers of poorly trained policies in production. Counterfactual Policy Evaluation (CPE) with respect to *perceived accumulated reward* is the essential tier of our offline evaluation methodology. CPE is performed *offline* and *off-policy* because it uses a static dataset of transitions (test set), logged by the behavioral policy π_β. Additionally, CPE is performed on the same policy that is deployed online, which adds ϵ-greedy exploration to learned offline policy to enable improved future offline training. CPE offers the safety of offline ranking of RL models before testing them in production. Off-policy evaluation can be based on ordinary importance sampling [17], but at a risk of high variance which grows exponentially with state transitions. Advanced methods [23] provide controlled bias-variance tradeoffs by utilizing additional information, e.g., the learned Q-function. In our case, relatively short MDP sequences help using the *sequential doubly robust estimator* [6] to produce unbiased performance estimates for policies trained offline.

Hyperparameter optimization for Offline RL is challenging [15], as there is no access to the environment. Here, it is performed toward the described metrics on a validation set from the static dataset $\mathcal{D}_{\pi_\beta}$ and the best seen models are evaluated online. Our DQN model (Figures 4(a) and 4(b)) exhibited the best results in offline evaluation, was deployed online and evaluated vs. SL baseline production system (Section 6.2).

6.2 Empirical comparisons

Our baseline using supervised learning (SL) is a prior production model with a tree-based meta-learner that estimates the Conditional Average Treatment Effects [10], and by considering the same

Table 1: SL and RL policies are compared online to fixed policies that always perform `Action A1` or `Action A2`. RL provides the most attractive tradeoffs, shown in bold.

	USER ENGAGEMENT		OTP COST
POLICY	DAILY	MONTHLY	
A2	-	-	-
A1	+1.49% $\pm$ 0.273%	+2.81% $\pm$ 0.279%	+120% $\pm$ 4.16%
SL	+1.35% $\pm$ 0.212%	+2.58% $\pm$ 0.116%	+81.1% $\pm$ 3.13%
RL	**+1.50% $\pm$ 0.213%**	**+2.55% $\pm$ 0.183%**	**+70.3% $\pm$ 2.96%**

Table 2: Action distributions (% of `Action A1`) for behavioral policies trained with SL and RL by user cohort.

	SL	RL
ALL USERS	55.88%	22.75%
SINGLE-LOGIN USERS	60.16%	36.41%
MULTIPLE-LOGIN USERS	49.40%	11.19%

user personalized features as the RL model (Section 4.1). The model acts as a reward predictor for `Actions A1, A2`, and unlike the RL model, it addresses the user authentication problem under a non-sequential setting. The treatment effects are represented by a linear combination of the rewards defined in Equation 2. The meta-learner decomposes the decision process into two steps by first estimating the conditional reward expectations, $U = \mathbb{E}[\text{UE} \mid a_t, s = s_t]$, $C = \mathbb{E}[\text{Cost} \mid a_t, s = s_t]$, where $a_t \in A$. U and C are computed using Gradient Boosted Decision Trees whose parameters, including learning rate, tree depth and tree leaves, are swept in a large range to minimize mean squared errors. Then the learner takes the differences between the estimates and chooses the action that gives the highest reward:

$$a_t = \arg\max_{a_t \in A} \mathbb{E}[r_t | a_t, s = s_t] \tag{3}$$

After deriving r_t via Equation 2, the final action is given by

$$a_t = \arg\max_{a_t \in A} (w_u \cdot U - w_c \cdot C) \tag{4}$$

The reward weights w_u and w_c are chosen through online experiments to ensure compelling multiobjective tradeoffs.

Online evaluation is performed using 30 days' data. Table 1 demonstrates the first round of results at a 95% confidence level, where our RL model is trained on randomized data as per Section 5.2. The RL model is compared to fixed policies (some action repeated always) and a prior production model based on SL. Overall, the RL model greatly reduces OTP costs vs SL, with minimal impact on daily and monthly user engagement. We estimate *Return On Investment* (ROI) by dividing total cost by total monthly engagement. RL outperforms SL by $5.93\% \pm 0.782\%$. To collect data for *recurrent training*, we deploy our RL agent by extending the deterministic policy to an ϵ-greedy policy with $\epsilon = 0.1$.

Table 3: Cost per state (10^{-2}) for multiple-login users. Engagement Sequence 11 represents being active in two consecutive states, while 00 being inactive in both states.

ENGAGEMENT SEQUENCE	SL	RL	% COST REDUCTION
11	0.178	**0.141**	20.8%
01	0.675	**0.543**	19.6%
00	1.44	**1.30**	9.72%
10	0.511	**0.402**	21.3 %

Table 4: Recurrent training results compared with the initial RL agent trained on randomized data (the "Rand" policy issues `Action A1` and `Action A2` with equal probability).

	User engagement		OTP cost
Policy	Daily	Monthly	
DQN	+0.002% ±0.106%	+0.058%±0.084%	-0.17% ±0.44%
CRR	+0.116% ±0.240%	+0.107%±0.140%	+1.27% ±2.87%
Rand	-0.336% ±0.266%	-0.693%±0.162%	+3.78% ±3.29%

When ML is applied to practical problems, it commonly optimizes *surrogate objectives.* In our case, one such metric counts *notification disavow events* (NDEs) for password reset, where a user turns off notifications, perhaps because there were too many. Comparing to SL, we observe that RL reduces NDEs by 50%. To this end, Table 2 shows that the RL agent trades off `Action A1` for `Action A2` and thus reduces authentication messages, while maintaining neutral engagement metrics. We also group users into cohorts, as multiple-login users are more likely to enter their password correctly without help from our authentication messages. From Table 2, we see that both RL and SL make use of this user attribute but RL exploits it more efficiently. We also augment SL data into MDP sequences to calculate the average cost per state (total cost of a sequence divided by its length). Focusing on sequences with under 3 steps (95% of all the sequences), we found that RL and SL imply very similar costs per state for single-login users (within 3%). But for multiple-login users RL reduces cost by 20% on sequences that have at least one successful engagement, as reported in Table 3. We permute the engagement result per state on MDP sequences of length 2 and use binary 0 and 1 to represent user activeness of that state.

Recurrent training and stability evaluation. When user behaviors shift over time, ML models must be refreshed using the most recent data (recurrent training). The initial RL model trained on randomized data shows sizable improvements. However, continued collection of randomized data is risky in terms of direct costs and user experience. The "Rand" policy (Section 5) in Table 4 suggests that its deployment limits the benefits of RL models. Thus, we refresh the model using behavioral data collected from our RL agent, and through offline training (Section 6.1) with particular attention in the KL-divergence between the initial and refreshed RL models' learned policies. Additionally, to check if recurrent training stabilizes in the long term compared with the initial RL model, we perform a second round of online evaluation to study recurrent training using RL behavioral data and model stability in the long term.

We train a DQN and a CRR model (Section 5.1) with the same parameters and rewards as the initial RL model, but collect their training data via policies from the deployed RL agent. Results summarized in Table 4 use the same settings as before (a 95% confidence level, one-month's testing window, and an ϵ-greedy policy on DQN and CRR with 10% exploration rate). Both recurrent DQN and CRR give neutral results compared with the initial RL model, which indicates stable performance in recurrent training. Comparing the CRR model to the initial RL model by daily user engagement for a one-month testing window, we see consistent performance with no metric loss.

7 Conclusions

In this work we show how to apply reinforcement learning (RL) to personalize user authentication in a Web-based system and compare RL to a competing SL approach. Working with industry data, using *offline* RL avoids releasing poorly trained agents in production. However, this approach requires careful extraction and augmentation of training data to ensure that off-policy learning does not succumb to *distributional shift*. In practice, RL is often susceptible to high variance and high bias at several of its stages, especially when operating on large-scale live data. Fortunately, our method is sufficiently robust for production deployment using the *ReAgent* (Horizon) platform [3]. Starting from state modeling and data collection, we articulate obstacles and milestones toward model training and demonstrate practical improvement in end-to-end system-level metrics at a large-scale deployment at Meta. During development, we use a simplified problem environment to test intuition without sensitive data.

References

[1] Akshay Awasthi. Reducing Identity Theft Using One-Time Passwords and SMS. *EDPACS*, 52(5):9–19, 2015.

[2] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed Distributional Deterministic Policy Gradients, 2018.

[3] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Yuchen He, Zachary Kaden, Vivek Narayanan, and Xiaohui Ye. Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. *CoRR*, abs/1811.00260:10pp., 2018.

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based Recommendations with Recurrent Neural Networks, 2016.

[5] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog, 2019.

[6] Nan Jiang and Lihong Li. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 652–661, New York, NYA, 20–22 Jun 2016. PMLR.

[7] Ron Kohavi and Roger Longbotham. Online Controlled Experiments and A/B Testing. *Encyclopedia of Machine Learning and Data Mining*, 7(8):922–929, 2017.

[8] Vijay Konda and John Tsitsiklis. Actor-Critic Algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 1008–1014, Denver, CO, 2000. MIT Press.

[9] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 11784–11794, Virtual, 2019. Curran Associates, Inc.

[10] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019.

[11] Ricardo Margarito Ledesma. Systems and methods for one-time password authentication, October 22 2020. US Patent App. 16/918,742.

[12] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural Attentive Session-Based Recommendation. In *Proc. 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1419–1428, New York, NY, 2017. ACM.

[13] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013.

[15] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter Selection for Offline Reinforcement Learning, 2020.

[16] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié, and Frédéric Dufau-Joel. A Case Study in a Recommender System Based on Purchase Data. In *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 377–385, New York, NY, 2011. ACM.

[17] Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[18] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, Virtual, 2014.

[19] Paul Rockwell. Two factor authentication using a one-time password, June 28 2016. US Patent 9,378,356.

[20] Mohit Kr Sharma and Manisha J Nene. Two-factor authentication using biometric based quantum operations. *Security and Privacy*, 3(3):e102, 2020.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[22] Xueying Tang, Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. A reinforcement learning approach to personalized learning recommendation systems. *British Journal of Mathematical and Statistical Psychology*, 72(1):108–135, 2019.

[23] Philip Thomas and Emma Brunskill. Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2139–2148, New York, NY, 20–22 Jun 2016. PMLR.

[24] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. A Path-Constrained Framework for Discriminating Substitutable and Complementary Products in E-Commerce. In *Proc. Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 619–627, New York, NY, 2018. ACM.

[25] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic Regularized Regression, 2020.

[26] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[27] Yifan Wu, George Tucker, and Ofir Nachum. Behavior Regularized Offline Reinforcement Learning, 2019.