2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)

Federated Reinforcement Learning For Fast Personalization

Chetan Nadiger, Anil Kumar, Sherine Abdelhak
Intel Corporation
e-mail: {chetan.hemanth.nadiger, anil1.kumar, sherine.abdelhak}@intel.com

***Abstract*—.**

Understanding user behavior and adapting to it has been an important focus area for applications. That adaptation is commonly called Personalization. Personalization has been sought after in gaming, personal assistants, dialogue managers, and other popular application categories. One of the challenges of personalization methods is the time they take to adapt to the user behavior or reactions. This sometimes is detrimental to user experience. The contribution of this work is twofold: (1) showing the applicability of granular (per user) personalization through the use of reinforcement learning, and (2) proposing a novel mitigation strategy to decrease the personalization time, through federated learning. To our knowledge, this paper is among the first to present an overall architecture for federated reinforcement learning (FRL), which includes the grouping policy, the learning policy, and the federation policy. We demonstrate the efficacy of the proposed architecture on a non-player character in the Atari game Pong, and scale the implementation across 3, 4, and 5 users. We demonstrate the success of the proposal through achieving a median improvement of ~17% on the personalization time.

Keywords-*federated learning, reinforcement learning; computer games; artificial intelligence; game personalization, edge computing*

I. INTRODUCTION

Personalization in the context of gaming, personal assistants, and dialogue managers has been an important area of research. Reinforcement learning (RL), among other Artificial Intelligence (AI) based techniques have recently been employed to achieve personalization like the authors show in a previous work [6]. RL in [6] was used for personalization of a non-player character (NPC) to the specific actions & reactions of the human player. Moreover, the personalization method was not limited to the human's clear and expressible actions, but potentially to the human's subtle emotions and engagements with the game.

While personalization to individual users using RL [6] is achievable, it suffers from a challenge which is the time it takes to personalize. The authors in this work present a federated reinforcement technique with the main goal of improving personalization time. Federated learning [1] applied on top of reinforcement learning techniques is one instantiation of hierarchical learning which enables agents at the lower level of the hierarchy to communicate their findings. Those agents whose environments are *similar* have higher chances of 'learning from each other' effectively. This paper describes a grouping policy which takes into consideration the similarity metrics among the agents' environments. It also describes the learning policy by which each agent evolves over time and adapts to its environment, and the federation policy.

This paper is organized as follows: section II describes the background and discusses the related works; section III explains the proposed approach; section IV presents the results and analysis; and section V concludes the paper with a summary.

II. BACKGROUND AND RELATED WORK

This work describes personalization techniques through federated reinforcement learning, which is analogous to distributed learning. Hence, the authors review existing state-of-the-art literature on distributed learning, and personalization techniques based on online learning. Since, personalization involves aspects of privacy and communication costs, the authors also specifically review federated distributed learning literature.

Federated learning is a relatively new concept popularized by Google in 2016 [1]. It was mainly developed to address privacy concerns on mobile devices. In [1] federation happens over supervised machine learning techniques. In [9], federated learning is used in a commercial, global-scale setting to train, evaluate and deploy a model to improve virtual keyboard search suggestion quality without direct access to the underlying user data [9]. However, in this paper, the authors build federation over nodes employing reinforcement learning, instead of supervised learning. More applications of Federated learning for supervised classification tasks can be found in [10], [11] and [12]. Various distributed version of stochastic gradient descent (SGD) algorithm have been proposed in [4, 5]. These works deal mainly with building unbiased estimators of population gradients in a distributed manner. However, in this work, we assert the assumption that similar clients lie on a close manifold that allows for relaxation of estimator bias.

Personalization using online learning has been used in gaming. Many of the existing works pertain to training Non-Player Characters (NPC) to play a particular game. The authors in [2] train a NPC to play Atari games in using Distributed Deep Reinforcement learning. They maintain multiple parameter servers in both synchronous and

978-1-7281-1488-0/19/$31.00 ©2019 IEEE
DOI 10.1109/AIKE.2019.00031

asynchronous mode. In this paper, however, the authors maintain only one parameter server per client grouping. This is detailed in section III. The authors of [3] propose the concept of Joint Action Learners. In this, each agent learns the value of its own actions in conjunction with those of other agents via integration of reinforcement learning (RL) with equilibrium learning [3]. However, the authors of the current paper propose a grouping policy where the agents are clustered based on similarity criteria for faster learning. The key novel contribution of this paper is the use of Deep Q Network (DQN) [7] reinforcement learning algorithm in a federated setting to achieve faster personalization.

This is termed as Federated Reinforcement Learning (FRL). Recently, FRL has been studied in [13] and [14]. In [13], the authors consider client models as well as the shared model as one big Q network and is optimized using Bellman equation. But, in the current work, there is separate Q learning on each of the clients and a federation policy decides the shared model parameters. In [14], the authors use generative networks based knowledge fusion algorithm while in the current work a federation policy determines the model fusion method.

The effectiveness of proposed FRL policies is demonstrated by personalizing an NPC in the game of Pong. With suitable domain specific modifications, the FRL can be applied in other domains as well and is not limited to gaming scenarios.

III. Proposed Approach

This section details, how FRL is used to adapt an NPC in the game of Pongto the skill level of a human-player. In the actual implementation, a 'fixed' neural network is used as a proxy for the human-player. These proxy neural networks are built, using DQN RL algorithm [6], to a pre-fixed skill level as explained in section III. Figure 1 shows a frame-grab of the Pong game.

Figure 1: Pong game visualization

The NPC is a DQN whose weights are updated using FRL. The human-proxy is a neural network whose weights are 'frozen' and works in inference mode during the game-play. The NPC aims to adapt its weights so as to personalize itself to the skill level of human-proxy player. The feature vector, reward function used to train the NPC as well as human-proxy player is same as in the authors' previous work in [6].

A. *Personalization metric*

In gaming, personalization can be game-specific or game-independent. Typically, game-independent metrics are based around measuring physiological signals to determine the arousal levels [8]. The current work tries to measure degree of personalization through in-game metrics. The authors opine that length of a rally in the Pong game is indicative of personalization. The return of the ball by NPC/Human-proxy player is considered as a rally. The length of a rally is the number of times rallies happen consecutively without either player losing a point. Higher the rally length, higher is the personalization / user-engagement. The measure used in this implementation is as follows.

$$PM = \frac{N_{r4}}{N_r}$$

PM: Personalization measure over a game episode

N_{r4}: Number of rallies of length >= 4 over a game episode (Threshold chosen empirically)

N_r: Total number of rallies of all lengths over a game episode

B. *Player grouping policy*

The basis of FRL is to share experiences across players so that it is possible to collect more samples in a shorter time. But, this runs the risk of adding irrelevant data samples and potentially increasing the personalization time and reducing the model quality as well. To mitigate this risk, it is important to share data samples only among players similar to each other. In this work, the global model (refer to section III) is built per player group as shown in Figure 2.

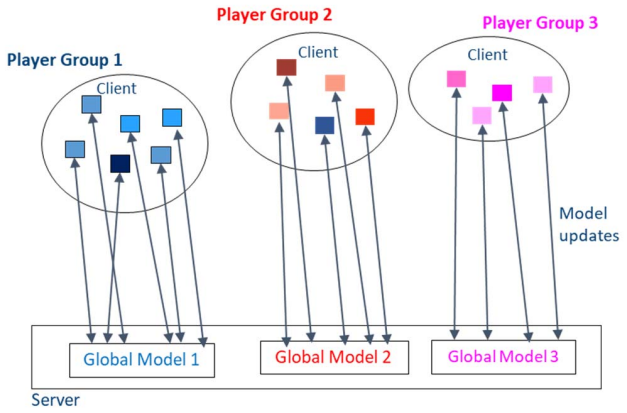

Figure 2: Different global model for each player group

Figure 2 shows a conceptual architecture of how one global model is built in server for each group of similar players (clients). Depending on the application needs, there could be different ways in which players' similarity could be

measured. The grouping policy used in this work is explained as follows. To reduce experimentation time, the authors trained few models with DQN algorithm to play the Pong game at varied skill levels as proxy for human players. This was necessary because it was impractical to actually have real human players while simulating the FRL experiments. The methodology to build and measure skill levels of these models are similar to work presented in [6]. The human-proxy players are then grouped based on their skill levels. It is to be noted that the definition of 'similar players' would change with the problem for which FRL would be applied. The next section details the experimentation setup and simulations.

C. *Federated learning (FL) setup*

In federated learning, multiple nodes participate and share model weights to a server node for aggregation [1]. The authors adopt the same architecture as shown in figure 3.

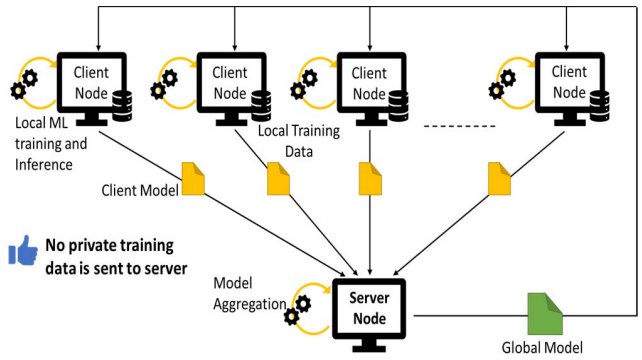

Figure 3: Client-Server architecture

The client nodes hold the local training data to train client models. As determined by a communication policy, the client model weights are shared to a server node where model aggregation takes place in tune with a federation policy to build a global model. The global model is then sent to the client nodes (again in accordance to a communication policy). The client nodes use the global model to update their local model weights based on a model replacement policy. Following sections describe the communication, federation and model replacement policies used in this work.

1) *Communication policy*

In each of the client node, a human-proxy player plays the Pong game with the NPC (DQN model). The NPC DQN model weights are updated to personalize it for the human-proxy player. Whenever any player reaches to a score of 15, the game session ends and this is termed as one episode. Figure 4 shows the flowchart of one client and its communication policy with the server.

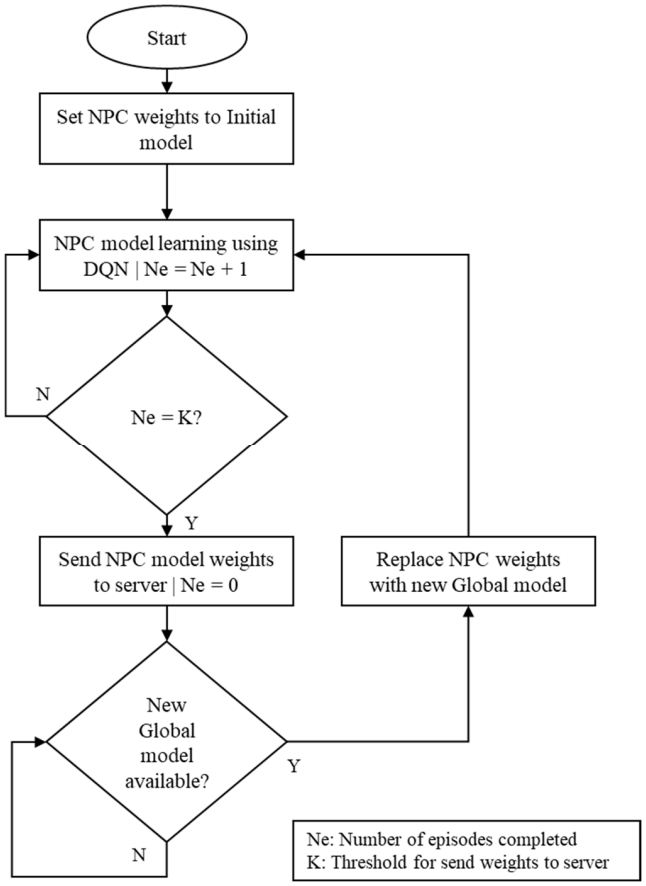

Figure 4: Communication policy

The client-server interaction happens after certain 'K' number of episodes of the game on the client. In this work, K is set to be 25 episodes. Once the client sends its NPC model weights, it then waits till a new global model is available. Once the new global model is available, it updates the NPC weights as per the model replacement policy. Then, it resumes play with human-proxy player for next K episodes. The global model is generated on the server through a federation policy.

2) *Federation policy*

The federation policy determines the strategy of aggregation of the client models.

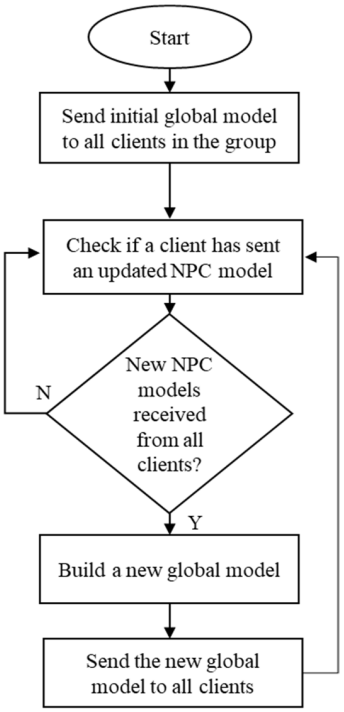

Figure 5: Illustration of the federation policy

Referring to figure 5, the server sends a base global model to all the clients to start the NPC learning. This gives each of the client a kind of 'warm-start'. The base global model was built offline using the method described in [6]. The clients then update the NPC model weights as per the local RL algorithm (refer to section III). The server waits till it receives NPC models from all clients of the group. Then, a global model built using the following equation.

$$W_{t+1}^{g} = \propto W_{t}^{g} + (1-\propto)\sum_{i=1}^{N}\frac{p_i}{P}W_{t}^{c}$$

$$\text{Where, } P = \sum_{i=1}^{N} p_i$$

W^{g}: Global model

W^{c}: Client model

$\propto$: Global model regularization factor

p_i: Percentage of rallies of length ≥ 4 on client i

N: Number of clients

Basically, the global model is built by making a weighted average of all received client models. Weightage is based on client personalization measure.

3) Model replacement policy

The model replacement policy determines how the client NPC weights are updated whenever a new global model is received. The NPC in this work is a DQN model. The DQN training involves two neural networks namely Q and Target Q networks [7]. The Q network weights are updated using a first-order optimization algorithm over a loss function defined as per the Bellman equation [7]. The Target Q network weights are updating by copying the Q network weights at certain frequency.

In this work, when a new global model is received, the authors update the Target Q network using a weighted combination of previous Q network weights and current global model weights. This is illustrated in figure 6.

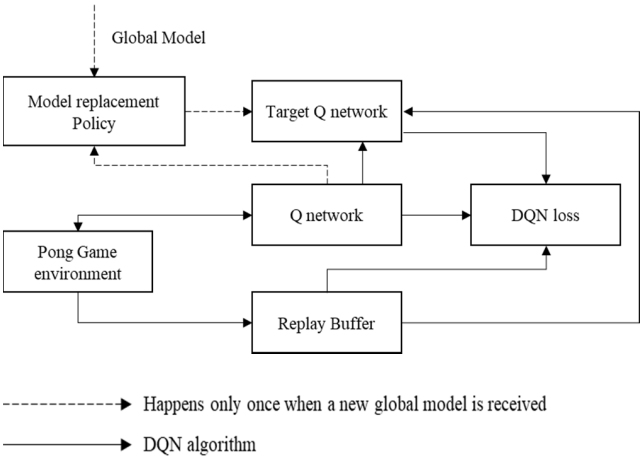

Figure 6: Illustration of the target Q update

The weightage factors for combining the models, is based upon their contribution to increase/decrease in personalization measure. This is illustrated in detail in algorithm 1. Once the target Q network is updated with the new global model, the NPC training resumes as per the original DQN algorithm [7].

Algorithm 1 Model Replacement

Input: Previous Q Model Q_{i-1}, New Global Model G_i, Previous Reward R_{i-1}, Previous Gain Gn_{i-1}, Previous Global Model Factor Gmf_{i-1}

Output: TargetQ Model TQ_i

set $Gn_i = R_i - R_{i-1}$
set $\alpha_1, \alpha_2, \alpha_3$ with random values $1 > \alpha_1 > \alpha_2 > \alpha_3 > 0$
set Rt_{min} with minimum reward threshold value
if $R_i > Rt_{min}$
  if $Gn_i > 0$ and $Gn_{i-1} > 0$ # *consistent gains*
    $Gmf_i = Gmf_{i-1}$ # *unchanged global model factor*
  else if $Gn_i < 0$ and $Gn_{i-1} < 0$ # *falling gains*
    if $Gmf_{i-1} == \alpha_1$ # *due to global model factor*
      $Gmf_i = \alpha_3$ # *reduce global factor value*
    else # *due to previous Q model*
      $Gmf_i = \alpha_1$ # *increase global factor value*

12: **else** *# inconsistent gains*
13: **if** $Gmf_{i-1} == \alpha_1$ or $Gmf_{i-1} == \alpha_2$
14: $Gmf_i = \alpha_3$
15: **else**
16: $Gmf_i = \alpha_1$
17: **else** *# current reward lower than threshold*
18: **if** $Gmf_{i-1} == \alpha_1$ or $Gmf_{i-1} == \alpha_2$
19: $Gmf_i = \alpha_3$
20: **else**
21: $Gmf_i = \alpha_1$
22: $TQ_i = Gmf_i \times G_i + ((1 - Gmf_i) \times Q_{i-1})$

Algorithm 1 presents the model replacement function at each client upon receiving the new global model. Note this logic is executed only once and later the DQN algorithm is used to continue updating the Target Q network. Empirical values are chosen for α_1, α_2, α_3 and Rt_{min}.

IV. Results and Analysis

The FRL concept presented in section III was verified by implementing it for personalizing the game of Pong. The authors haven't come across similar use-cases of FRL for faster personalization for any comparative studies. Table 1 shows the results of the experiments.

Concept	Clients	C1	C2	C3	C4	C5
Inference		12%	15%	20%	16%	15%
RL		19%	18%	19%	18%	18%
FRL	3	**22%**	18%	18%	NA	NA
FRL	4	**24%**	**21%**	**26%**	**26%**	NA
FRL	5	**21%**	**20%**	**21%**	**28%**	**30%**

Table 1: Personalization scores obtained for different experiment configurations

Row 1 in the Table 1 shows the personalization scores (refer section III-A) for clients C1 to C5 when the NPC is the base DQN model as explained in [6] working in 'Inference' mode. In other words, the weights of the NPC DQN are 'frozen' and are not updated. The personalization score is computed over 2500 game episodes of the Pong game.

Row 2 in Table 1 presents the scores when the NPC DQN model is allowed to be updated using Reinforcement Learning using training samples obtained only from the game-play with 1 client. The NPC DQN starts with some base model and keeps on 'learning' over 2500 episodes.

Rows 3-5 show the score obtained using the proposed FRL concept for 3, 4 and 5 number of clients. It can clearly be observed that in most of the cases, FRL is able to get higher personalization scores as compared to RL or Inference in same number of episodes. This indicates that FRL is able to produce higher rate of adaptation.

V. Conclusions

In conclusion, this paper presents an approach using federated reinforcement learning to speed up the personalization process of smart agents to their environment. The paper presented a grouping policy, a learning policy, and a federation policy which constitute the overall FRL architecture. The results show the efficacy of the approach by testing it on 3, 4, and 5 agents, where the speed up of the personalization time is ~ 17%. This approach is paving the road for advancements in the area of federated continuous learning for the aim of instant personalization to changing environments.

References

[1] McMahan, H. Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." arXiv preprint arXiv:1602.05629 (2016).

[2] Adamski, Igor, et al. "Distributed Deep Reinforcement Learning: Learn how to play Atari games in 21 minutes." International Conference on High Performance Computing. Springer, Cham, 2018.

[3] Claus, Caroline, and Craig Boutilier. "The dynamics of reinforcement learning in cooperative multiagent systems." AAAI/IAAI 1998 (1998): 746-752

[4] De, Soham, and Tom Goldstein. "Efficient distributed SGD with variance reduction." 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016

[5] Chen, Jianmin, et al. "Revisiting distributed synchronous SGD." arXiv preprint arXiv:1604.00981 (2016).

[6] Bodas, Anand, et al. "Reinforcement learning for game personalization on edge devices." 2018 International Conference on Information and Computer Technologies (ICICT). IEEE, 2018.

[7] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

[8] Bersak, Daniel, et al. "Intelligent biofeedback using an immersive competitive environment." Paper at the Designing Ubiquitous Computing Games Workshop at UbiComp, 2001.

[9] Yang, Timothy, et al. "Applied Federated Learning: Improving Google Keyboard Query Suggestions." arXiv preprint arXiv:1812.02903 (2018).

[10] Chen, Tianyi, et al. "LAG: Lazily aggregated gradient for communication-efficient distributed learning." Advances in Neural Information Processing Systems. 2018.

[11] Zhou, Wei, et al. "Real-Time Data Processing Architecture for Multi-Robots Based on Differential Federated Learning." 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, 2018.

[12] Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. S. (2017). Federated multi-task learning. In Advances in Neural Information Processing Systems (pp. 4424-4434).

[13] Zhuo, H. H., Feng, W., Xu, Q., Yang, Q., & Lin, Y. (2019). Federated Reinforcement Learning. arXiv preprint arXiv:1901.08277.

[14] Liu, B., Wang, L., Liu, M., & Xu, C. (2019). Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems. arXiv preprint arXiv:1901.06455.