Dynamic Adaptive Streaming based on Deep Reinforcement Learning

Jian Chen\(^1\), JianYing Wu\(^1\),

\(^1\) Information Department, Beijing University of Technology, Beijing, China

\(^*\)Corresponding author e-mail: s201641002@emails.bjut.edu.cn, s201641019@emails.bjut.edu.cn,

Abstract. Dynamic adaptive streaming over HTTP (DASH) is the dominant technology of multimedia delivery over the Internet. In DASH system, adaptive bitrate (ABR) algorithms running on client-side video player are the key to improve user quality of experience (QoE). However, most existing ABR algorithms employ fixed control rules to make bitrate decisions based on throughput, playback buffer size, or a combination of the two. As a result, their performance in the complicated and fluctuant network environment is incompetent. In this paper, we propose QRL, a bitrate adaptation approach based on deep reinforcement learning. QRL uses double Q-Learning, an enhanced Q-Learning method. After training the neural network model, the algorithm can select proper bitrates for future video segments based on all the information collected by client during the video playback process. Simulation results show that QRL achieves better performance than other algorithms.

1. Introduction

In recent years, multimedia delivery represents one of the most important applications, which is mainly due to the increase in traffic of video streaming. At the same time, Dynamic Adaptive Streaming over HTTP (DASH) has become the most important technology of video delivery.

Figure 1 illustrates the DASH system architecture. In DASH, videos restored on the server are divided into a series of chunks. Each chunk which stands for 2-10 seconds segment of the entire video is encoded into a series of different bitrates, which means different resolution and frame rate level. Based on the network condition of the client, video player sends requests to the server to download chunks at a proper bitrate by using an adaptive bitrate (ABR) algorithm. ABR algorithms use varieties of different input signals (e.g., throughput measurements, video playback buffer occupancy, etc.) to make bitrate choices to optimize the Quality of Experience (QoE).

ABR algorithms is the core of DASH since it is intended to provide the user with the video at the highest quality. Therefore, the policies adopted by ABR algorithms heavily affect the performance of video streaming. To design ABR algorithms is challenging for the following reasons: (1) QoE is decided by a series of metrics [1], such as video rate levels, buffering events and video rate smoothness. It's hard to find a solution to improve the performance of all these metrics at the same time since they are mutually affected; (2)Network conditions vary greatly in different environment and fluctuate with time, but the artificially designed algorithms depend on the accuracy of the input signal; (3) Bitrate selection for a given chunk will affect the following performance of the algorithm, so the algorithm must take the far-reaching impacts into account. As a result, their performance in the complicated and fluctuant network environment is incompetent.
Reinforcement learning has gained great success in recent years and it also provides a feasible solution for the ABR algorithm. We propose QRL, a bitrate adaptation approach using double Q-Learning, an enhanced deep Q-Learning method.

2. The overall framework
QRL generates ABR algorithm instead of approaches that use policy based on fixed observation. The basic idea of reinforcement learning is to interact with the environment by using an agent. At time $t$ for selecting the bitrate, the agent obtains information from the environment to generate state $s_t$, then the agent takes an action $a_t$. The agent will get a reward $r_t$ while the state of the environment changes from $s_t$ to $s_{t+1}$.

The main purpose of reinforcement learning is to get the maximum accumulative expected reward: $E[\sum_{t=0}^{\infty} r_t]$, where $\gamma \in [0,1]$ is the discount for the expected future reward. The agent observes a series of signals and sends these signals to the internal neural network. The neural network outputs the action, which means the bitrate selection of the next video segment. The QoE of this selection will be subsequently passed to the ABR agent as a reward. The reward information will then be used by the agent to update the parameters of the neural network.

2.1. Reward function
The reward function is the basic guide that the agent uses to learn the optimal strategy. In video streaming, the reward function should be reflected in the measurement of the user QoE. In this paper, the quality of video streaming sessions is calculated by $QoE = \sum_{t=0}^{\infty} qoe_t$. The computations of $qoe_t$ will be expressed as:

$$QoE_N = \sum_{n=1}^{N} q(R_n) - \alpha \sum_{n=1}^{N} T_n - \beta \sum_{n=1}^{N-1} |q(R_{n-1}) - q(R_n)|$$  \hspace{1cm} (1)

where $\alpha$ and $\beta$ are non-negative weights. The three parts of $QoE_N$ are used to calculate the quality level of the video, the time of rebuffering and bitrate fluctuation. For a video divided into $N$ segments, $R_n$ represents the bitrate of chunk$_n$ and $q(R_n)$ represents the quality of bitrate, $T_n$ represents the rebuffering time of chunk$_n$, $q(R_{n-1}) - q(R_n)$ represents the change of video quality. In this paper, we set $\alpha = 1$ and $\beta = 4.3$.

2.2. Architecture
As shown in Fig. 2, the neural network model is based on the research in [2]. In this paper, we use a 1D convolution layer to connect to input layer. The 1D convolution layer contains 64 filters, whose size is 4 and stride is 1. Results from this layer will be passed to a fully connected layer including 64
neural nodes. In the output layer, the number of neurons is the same as the number that videos are encoded by the video provider.

The agent uses the state information $s_t = (\hat{x}_t, \hat{t}_t, \hat{n}_t, \hat{b}_t, \hat{c}_t, \hat{l}_t)$ to make bitrate selections. $t$ represents the decision time-point. In $s_t$, $\hat{x}_t$ is a vector that contains the throughput metrics for the past $k$ video chunks; $\hat{t}_t$ represents a vector of the downloading time of past $k$ video segments; $\hat{n}_t$ is a vector of the sizes of the next chunk that could be selected; $\hat{b}_t$ represents the level of current buffer size; $\hat{c}_t$ represents the number of left video chunks; $\hat{l}_t$ is the selected bitrate of the last chunks. In our algorithm (QRL), we set the $k = 8$. With the state information, the neural network of the agent will get the accurate description of the environment by extracting some features.

2.3. Training

The training process of reinforcement learning is usually very time consuming, especially by playing real-time video. Therefore, to accelerate this process, the training of the ABR agent is usually based on trace. We use two network trace datasets, one is a publicly available dataset provided by the US Federal Communications Commission [3], the other is provided by the Hunan Telecom which contains a collection of 1100 bandwidth traces of 240 seconds. In our method, we set $\gamma = 0.99$ in the reward function.

2.4. Policy gradient

As shown in Fig. 2, in the basic Q-value neural network, the training process of the value network parameters $\theta$ in deep Q-Learning will be expressed as:

$$
\theta \leftarrow \theta - \alpha \cdot \sum_{t} \nabla_{\theta} \left[ r_t + \gamma Q(s_{t+1}, \pi_{\theta}(s_{t+1})) - Q(s_t, a_t) \right]
$$

(2)

the optimal policy will be expressed as:

$$
\pi_{\theta}(s_t) = \arg \max_a Q(s_t, a)
$$

(3)

In this paper, we use double Q-Learning, an enhanced deep Q-learning method, which use two networks with same structure to overcome the overestimation bias [4]. In double Q-Learning method, the action $a_t$ which maximizes the expected value for state $s_{t+1}$ is chosen by the online network. The corresponding target value is calculated by the target network. In double Q-Learning method, the training process of the online network parameters $\theta$ will be then expressed as:
\[ \theta \leftarrow \theta - \alpha \cdot \sum_t \nabla \theta [r_t + \gamma \hat{Q}(s_{t+1}, \pi_\theta(s_{t+1})) - Q(s_t, a_t)] \]  

(4)

where \( \alpha \) is the learning rate, \( Q \) represents the online network and \( \hat{Q} \) represents the target network.

In the DASH bitrate adaptation scenario, double Q-Learning makes the estimated target value more balanced when epsilon-greedy policy is used to guide the selection bitrate of the next video chunk. Through double Q-learning, the trained online network can achieve better performance in approximating the real state-action value.

3. Evaluations

In this section, we compare our approach based on reinforcement learning against existing approaches implemented in a JavaScript video streaming simulator. The simulator has a playback buffer whose capacity is 30 seconds.

We use a video taken from DASH dataset [5]. Each segment has a size of 2 seconds and encoded at bitrates in \{300, 750, 1200, 1850, 2850, 4300\} kbps. We use the datasets provided in [3] as a test set.

We can compare our approach with the following bitrate adaption algorithms:

1. Buffer-Based(BB): We implement the buffer-based bitrate adaption which approach proposed by Huang et al. [6]. In this paper, we set the buffer reservoir to 5 seconds and cushion to 15 seconds. The player will select the highest bitrate when the buffer occupancy is more than 15 seconds and hold the playback buffer more than 5 seconds.

2. Rate-Based(RB): We implement the rate-based bitrate adaption by calculating the mean throughput of the previous 5 video segments. The predicted bitrate of next video segment is the maximum bitrate value that is smaller than mean throughput.

3. robustMPC [7]: robustMPC algorithm chooses the video bitrate with buffer occupancy signals and throughput predictions. robustMPC algorithm tries to maximize the given QoE metrics by accounting for errors between predicted throughputs and observed throughputs of the past 5 chunks.

4. QRL: Our proposed approach that based on double Q-Learning. In order to make the whole model converge as much as possible, we generate a training dataset with 500 tracks that obeys the Gauss distribution, which have different mean and variance.

From the CDF of the 4 algorithms shown in fig. 3, we can observe from the picture that the RL algorithm gets the highest total reward. It means the proposed algorithm balances the quality level of the video, the time of rebuffering and bitrate fluctuation.

According to fig. 4, we can see that the RL algorithm gets the highest bitrate selection score and has the lowest bitrate fluctuation score. The BB algorithm get the poorest performance because of its frequent fluctuations of bitrate and conservative bitrate selection. Though robustMPC gets a higher score of video bitrate, its highest quality fluctuation score reduce the total reward. It indicates that QRL
algorithm has learned how to use the throughput and buffer occupancy observations to make decisions according to its long-term consideration.

4. Conclusion
In this paper, we propose QRL, a bitrate adaption algorithm based on reinforcement learning. We train QRL by simulating the interaction of the client and the video server network. The experiment results show QRL has significant advantages in maintaining stable video bitrate switching under bandwidth fluctuation conditions. QRL has a comprehensive consideration of the bitrate level of the video, the rebuffering event and bitrate fluctuation, thus improves quality, fluency and stability in video streaming.

References
[1] Balachandran, A., Sekar, V., Akella, A., Seshan, S., Stoica, I., & Hui, Z. (2013) Developing a predictive model of quality of experience for internet video. Acm Sigcomm Computer Communication Review, 43(4), 339-350.
[2] Gadaleta, M., Chiariotti, F., Rossi, M., & Zanella, A. (2017) D-dash: a deep q-learning framework for dash video streaming. IEEE Transactions on Cognitive Communications and Networking, 3(4), 703-718.
[3] Federal Communications Commission. (2016) Raw Data - Measuring Broadband America. https://www.fcc.gov/reports-research/reports/
[4] Van Hasselt, H., Guez, A., & Silver, D. (2015) Deep reinforcement learning with double q-learning. Computer Science.
[5] Lederer, S., Mueller, C., Timmerer, C., Concolato, C., Feuvre, J. L., & Fliegel, K. (2013) Distributed DASH dataset. Acm Mmsys.
[6] Huang, T. Y., Johari, R., Mckeown, N., Trunnell, M., & Watson, M. (2014) A buffer-based approach to rate adaptation: evidence from a large video streaming service. Acm Conference on Sigcomm.
[7] Yin, X., Jindal, A., Sekar, V., & Sinopoli, B. (2015) A control-theoretic approach for dynamic adaptive video streaming over http. ACM SIGCOMM Computer Communication Review, 45(5), 325-338.