Entropy-Based Prioritized Sampling in Deep Q-Learning

Mirza Ramicic
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
e-mail: mirza.ramicic@polimi.it

Andrea Bonarini
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Milan, Italy
e-mail: andrea.bonarini@polimi.it

***Abstract*—Online reinforcement agents take advantage of experience replay memory that allows them to reuse experiences from the past to re-learn, thus improving the overall efficiency of the learning process. Prioritizing on specific transitions during the sampling and replay increased the performance of learning even more, but in previous approaches the priority of the transitions was determined only by its TD error property. In this work, we introduce a novel criterion for evaluating the importance of the transition which is based on the Shannon's entropy of the agents perceived state space. Furthermore, we compare the performance of different criteria for prioritizing on one of the simulation environments included in REinforcejs framework. Experimental results show that DQ-ETD which uses a combination of entropy and TD error criterion outperforms the approaches based on the TD error criterion only such as DQ-TD.**

Keywords-reinforcement learning; neural networks; Markov decision processes; entropy;

I. INTRODUCTION

Implementation of approximation techniques widely used in supervised and unsupervised learning, namely artificial neural network architectures [1], enabled Reinforcement Learning (RL) to cope with very large state spaces. This opened a possibility of applying RL techniques to more complex problems and gave rise to successful implementations, such as playing Atari games and Go amongst others [2], [3], [4] , [5] which used a Deep Neural Network to approximate the reward function.

Online agents learn from a stream of experiences: after each transition the *Temporal Difference* (TD) error is back-propagated through the neural network so that the previous approximation is updated. However, the sequence of experiences in RL can contain highly correlated samples that break the *Independent and Identically Distributed* assumption of artificial neural network architectures [6] . To reduce the temporal correlation between experiences and improve the speed of learning, a technique called *Experience Replay* [1], [2] is used to allow an agent to reuse past experiences, therefore obtaining a more stable training of the neural network. The transitions are uniformly sampled and stored in a sliding window memory; after each transition a batch of the stored experiences are used to train the neural network.

Since some transitions are more valuable for learning than others, especially in the early stages, prioritizing on experience transitions was introduced in order to improve the general performance of learning.

Successful approaches dealt with prioritized experience replay [6] and prioritized experience sampling [7] , but their prioritization criterion was limited to one property of the transition: TD error, which is mostly conditioned by the reinforcement that the transition gave rise to. This inference was exploited in approach of prioritized experience sampling [7] that prioritized on a specific property of the transition that in general yeilds higher absolute values of TD error: its immediate reward value being non-zero.

In this work, we propose an additional criterion for prioritizing which takes into account the specific property of the sensed state space, measured by Shannon's entropy. We further show that prioritizing on transitions that include the state with higher entropy values can additionally improve the performance in some learning scenarios.

II. THEORETICAL BACKGROUND

A. Reinforcement Learning

A reinforcement learning process involves an agent learning from interactions with its environment in discrete time steps in order to update its mapping between the perceived state and a probability of selecting possible actions (policy). The agent performs a sequence of transitions of a Markov decision process represented by a tuple (s_t, a_t, r_t, s_{t+1}) and at each step updates its policy π_t in order to maximize the total amount of cumulative reward over the long run [9]. For this reason the optimal action-value function $Q^*(s,a)$ is defined as the maximum expected return following the policy π:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi] \quad (1)$$

After each transition it is possible to update the estimation of the action-value function using Bellman equation as an iterative update in order to converge to the optimal action-value function:

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s, a\right] \quad (2)$$

Equation (2) guarantees the convergence as $i \rightarrow \propto$, but it is impractical to use without any generalization and approximation, when facing high dimensional state spaces. Instead, most practical approaches use function approximators to estimate the action-value function, which range from simple linear perceptrons to non-linear approximators such as neural networks.

B. Approximation

In a function approximation with neural networks, at each iteration, the weights Θ are updated by performing a gradient descent on the loss functions $L_i(\Theta_i)$ according to Equation (3) therefore improving the previous estimate of the optimal action-value function $Q(s,a;\Theta) \approx Q^*(s,a)$.

$$\nabla_{\Theta_i} L_i(\Theta_i) = (y_i - Q(s,a;\Theta_i))\nabla_{\Theta_i} Q(s,a;\Theta_i) \qquad (3)$$

where $y_i = r + \gamma \max_{a'} Q(s',a';\Theta_{i-1})$ is the target for iteration.

Temporal difference learning combined with a deep neural network for approximation of action-value function is called *Deep Q-Learning*, or DQL [2] .

III. State Space Entropy Prioritization

A. Going Beyond TD Error

An agent performs the learning process on a single transition (s_t, a_t, r_t, s_{t+1}) by first predicting its previous estimate of the Q-value for being in a state s_t and taking an action a_t. This process performs a forward pass on the neural network approximator with s_t on input, after which we select the predicted Q-value on the output a_t. TD error represents the discrepancy between the previous estimate and the expected target Q-value after the transition which is given by its newly discovered reinforcement value r_t and the discounted maximum Q-value of the next state s_{t+1}. The learning process represents an update on the estimate of the function approximator by using backpropagation rule with perceived state space features s_t on the input and TD error difference on the a_t output.

Previous prioritization approaches consider that the magnitude of TD error which a specified transition generates can directly influence the speed of the learning process given the nature of the learning update. Transitions with high magnitude of TD error that are usually produced by non-zero reinforcement reward are especially valuable in the early stages of learning because they carry more training information and thus can make the learning process faster.

Because the backpropagation update takes into account not only the TD error at the output but a state space vector at the input while performing a gradient descent we can analogously postulate that the nature of the state space vector can also have an effect on the learning process itself. Some of the state space vectors are potentially carrying more training information and can be favored by prioritizing, in order to improve the learning performance.

For example: if an agent perceives a predictable or simple environment during the transition, the state space vector will potentially carry less information for training than when the transition is made in a dynamic and highly unpredictable environment.

We can thus say that the learning utility of the transition not only depends on the transition TD error, but also on the potential information that is carried by the perceived state space vector.

B. Quantifying the Unpredictability of the State Space

In order to quantify the amount of uncertainty and possible information gain that a state space vector can carry we have applied Shannon's entropy as a measure of diversity, also called Shannon's index. The state space vector is represented by a number M of variables that are in most cases continuous and normalized in

$$0..1$$

In order to measure the entropy, each of the M state space variables are discretized into N bins and calculated using Equation (4), where p_i is the frequency of values belonging to the ith bin.

$$H(s_t) = -\sum_{i=1}^{M} p_i log_2 p_i \qquad (4)$$

C. Model Architecture and Learning Algorithm

Previous prioritization algorithms [7] used a stochastic sampling method that falls between uniform sampling and greedy sampling based on the TD error.

In our approach, we introduce an additional criterion based on the diversity of the state space that can further prioritize on the uniform sampling part of the algorithm.

For the purpose of prioritizing we extend the experience description tuple with an entropy value $H(s_t)$ of the state space that is calculated using Equation 4, so that our stored transition takes the form of $e_t = (s_t, a_t, r_t, H(s_t), s_{t+1})$.

Instead of greedy sampling on $H(s_t)$ values which can make the system prone to over-fitting because of the lack of diversity [6] , we define a stochastic prioritization based on the entropy criterion $H(s_t)$ where the probability of sampling the $P(i)$ transition from the sliding window experience memory D is determined from Equation 5. $H(s_t)$ in this case represents the priority of the transition and the β parameter determines how much prioritization is used; in the uniform case $\beta = 0$.

$$P(i) = \frac{H(s_t)_i^{\beta}}{\sum_{j=1}^{j=size(E)} H(s_t)_j^{\beta}} \qquad (5)$$

To alleviate the selection of the values for the β parameter, which would need to be tweaked for the specific application, we introduce a more general prioritization technique based on the descriptive statistical property of quartiles that can be used in a broader sense with no additional adjustments.

In order to sample basing on the $H(s_t)$ criterion, in Algorithm 1, instead of the stochastic approach given by Equation (5) we use a descriptive statistic approach which

takes into account the upper interquartile mean of the data stream or the third quartile value ($Q3$) of the H_t values of agents experiences stored in a sliding window memory E of capacity n. This is computed by Equation (6).

$$H(s_t)_{Q3} = \frac{3(n+1)}{4} thH(s_t) \quad (6)$$

Given this, we sample only the transitions with $H(s_t)$ higher than the upper interquartile mean $H(s_t)_{Q3}$ of the entropy experience memory E as shown in Algorithm 1.

Algorithm 1 selectively stores the transitions after each update step based on two criteria. The first one is based on the TD error, and simply stores the transitions that result in a reinforcement r_t different from null. The second criterion stores the transitions that have the entropy state space value $H(s_t)$ higher than the upper interquartile mean of the n latest entropy samples from E given by the $H(s_t) > H(s_t)_{Q3}$ conditional.

After each transition a random batch of the previous transitions is selected from the replay memory D in order to perform additional training on the approximator.

Algorithm 1 DQL with entropy-based prioritization

Initialize replay memory D with capacity N and entropy experience memory E
Initialize action-value function Q with random weights
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    Calculate the entropy value $H(s_t)$ of the state space based on Equation 4 and add it to the sliding window memory E
    if $r_t ! = 0$ **then**
      Store transition $(s_t, a_t, r_t, H(s_t), s_{t+1})$ in D
    end if
    Calculate upper interquartile mean $H(s_t)_{Q3}$ of the last n samples from E using Equation 6
    if $H(s_t) > H(s_t)_{Q3}$ **then**
      Store transition $(s_t, a_t, r_t, H(s_t), s_{t+1})$ in D
    end if
    Sample random batch of transitions $(s_t, a_t, r_t, H(s_t), s_{t+1})$ from D
    set $y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{nonterminal} \end{cases}$
    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 3
  end for
end for

IV. Experimental Setup

To evaluate the proposed model we have adopted a learning environment that consists of moving good/bad food pieces [9] . Food pieces are generated at a random position with random speed and direction, and move in a constrained environment by bouncing on the walls. Agents can move in the same environment and should learn to touch (eat) good food pieces and to avoid bad food pieces. The goal of each agent is to consume as much good food pieces as possible, while, in turn, try to avoid the bad food sources. After being consumed, new food pieces of the same type of the consumed ones are re-generated with a random position, speed, and direction, thus keeping the distribution of food constant. Agents receive reinforcement +1 for consuming good food pieces and -1 for consuming bad ones.

The state space is continuous and intentionally high-dimensional for the purpose of increasing the entropy and consequently the diversity of possible experience transitions. Each agent has 30 directional sensors and each of them can perceive 5 continuous variables: distance of sensed object (good food, bad food, wall), the first two of which have the two additional attributes: speed in x direction and speed in y direction; this gives a total of 150 state space inputs for each agent.

As a function approximator we are using a deep neural network with weights Θ to approximate $Q(s, a; \Theta) \approx Q^*(s, a)$. To reduce the computational complexity of having multiple forward steps each time, we want to find an action that maximizes the state-action function $\text{argmax}_a Q(s, a)$; the network takes the state vector s as an input and predicts $Q(s, a)$ for each possible action.

We have adopted the original Q-learning update fourmula with a learning rate α set to a low value (0.05) because of the nature of the approximator, and discount factor $\gamma = 0.9$. The default capacity of the replay memory buffer D included 7000 experiences and the entropy experience memory capacity n was set to 500.

A. Entropy Criterion Comparisons

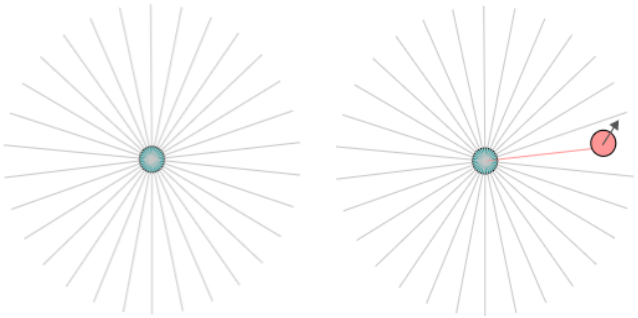
In order to evaluate how does entropy value of the state space vector $H(s_t)$ relate to the diversity of the agents' perception and further to its learning potential we are comparing state examples from the experimental setup grouped in low, medium and high entropy levels. For the purpose of evaluation each of the detected objects is depicted with its speed vector that represents the composition of its x and y speed components as described in the state space.

Figure 1 showcases some of the low entropy states, ranging from $H(s_t)$ 1.0 to 1.4. The center circle represents the agent, the lines its 30 detectors, the other circles food pieces. From Figure 1a we can see that if an agent is not perceiving any object the entropy of the state vector has the lowest value of $H(s_t) = 1{,}0723$. This represents the transition having the least value for learning process even if the transition results in a reinforcement reward that makes the TD error potentially high.

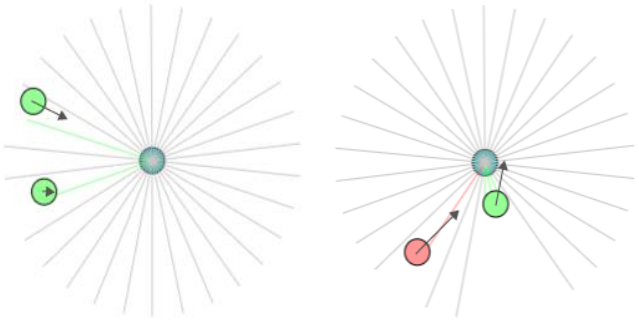
Figure 1 also shows that the $H(s_t)$ rises with the number of perceived objects but this is not always the case; both Figure 1c and Figure 1d have the same number of objects in the range but in Figure 1d we see more differences in distance from the objects and in their respective speed vectors, which account for higher diversity and consequently higher entropy values.

Medium entropy states shown in Figure 2 confirm the previous observation, but also include the food sources that are triggering more detectors because they are closer to the agent and this results in even higher entropy values. The previous assumption of the diversity in distance and speed vector is especially evident in the entropy difference between the situations respectively depicted in Figure 2a and Figure 2b.

Figure 3 shows the states with the highest entropy and diversity. We can notice that these states involve a high number of objects with diverse distances and speed vectors, which have greater potential for the learning process because they inherently carry more information.



(a) $H(s_t) = 1{,}0723$ (b) $H(s_t) = 1{,}1556$



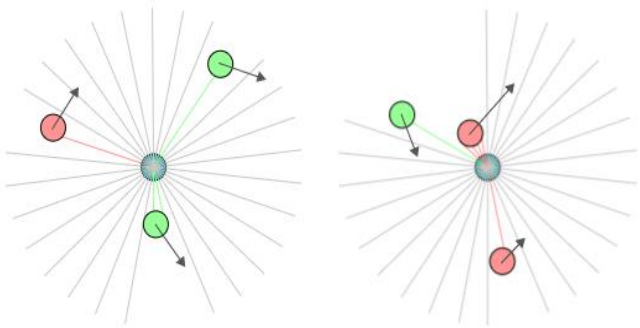
(c) $H(s_t) = 1{,}2035$ (d) $H(s_t) = 1{,}3711$

Figure 1. State spaces with low entropy.

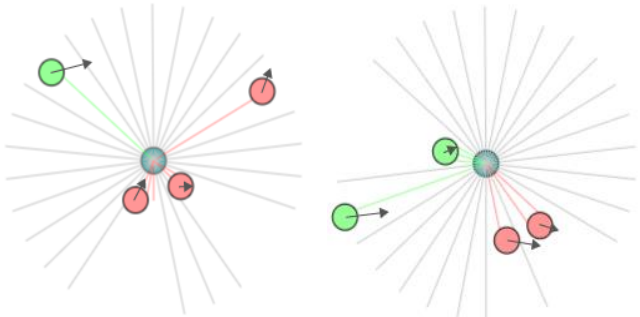
V. Experimental Results

In the experiments, we have compared two types of prioritized sampling algorithms with the baseline one, which uses only uniform sampling *DQ-U*. First prioritized sampling algorithm *DQ-TD* combined uniform sampling and prioritization based on the immediate reinforcement component of TD error value being non-zero, while the *DQ-ETD* combined two criteria for prioritization: entropy of the state space vector $H(s_t)$ and the reinforcement value one.

Figure 4 shows the comparison between the three different algorithms applied to our experimental setup; First algorithm *DQ-U* represents the baseline as it utilizes only uniform sampling, with no prioritization.

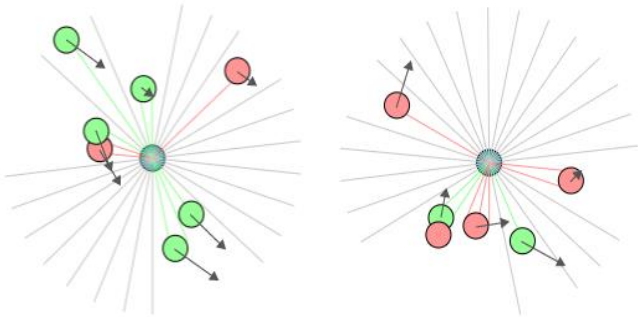


(a) $H(s_t) = 1{,}4832$ (b) $H(s_t) = 1{,}5589$

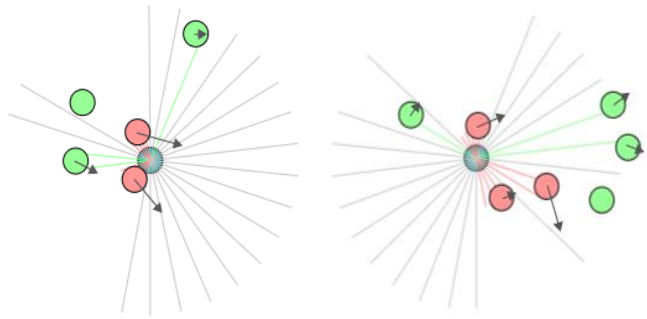


(c) $H(s_t) = 1{,}6985$ (d) $H(s_t) = 1{,}7895$

Figure 2. State spaces with medium entropy.



(a) $H(s_t) = 1{,}8633$ (b) $H(s_t) = 1{,}9608$



(c) $H(s_t) = 2{,}0610$ (d) $H(s_t) = 2{,}1244$

Figure 3. State spaces with high entropy.

Algorithm *DQ-TD* uses prioritization based on reinforcement value only, while *DQ-ETD* combines the entropy and TD error criteria based on the reinforcement value being non-zero as shown in Algorithm 1.

From Figure 4 we can see that the *DQ-ETD* method outperforms both the baseline *DQ-U* and *DQ-TD* method based on TD error prioritization only.

From these results, we can notice that adopting an additional prioritization criterion based on the state space properties can significantly improve the efficiency of the prioritization mechanism.

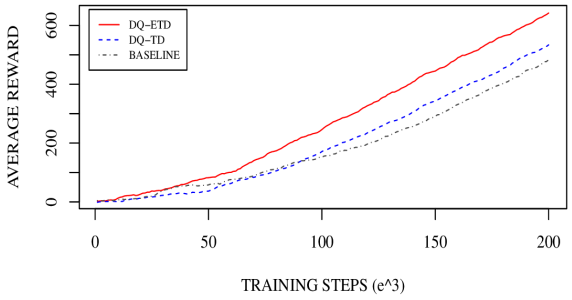


Figure 4. Comparison between average reward values in 20 learning epochs respectively with *DQ-ETD*, *DQ-TD*, and the baseline *DQ-U*, over first 200K learning steps.

VI. Conclusion

We presented a novel approach to sample replay memory combined with Deep Q-learning, which includes a new criterion for prioritization based on the entropy of the state space vector called *DQ-ETD*. Experimental results have shown that *DQ-ETD* can outperform the prioritized sampling approaches based on the TD error component criterion only such as *DQ-TD* in the early stages of the learning process.

A. Limitations

Greedy sampling on the prioritization criteria in both *DQ-TD* and *DQ-ETD* introduces a bias which is tolerable in the early stages of learning, but it may violate the convergence guarantee of the Equation 2, and therefore may prevent the agent to obtain an optimal policy π in the long run. For this reason, in our future work we intend to use adjusted annealed importance sampling [10] to compensate for the bias.

B. Future work

In this work, we have dealt only with the criteria for sampling the transitions from the agents' stream of experiences; we have not dealt with the prioritized memory replay. In our future work, we plan to combine *DQ-TD* with prioritized memory replay technique based on the amount of replay times of the transition in the memory buffer [7] .

References

[1] L.-J. Lin, "Reinforcement learning for robots using neural networks," DTIC Document, Tech. Rep., 1993.

[2] J V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., "Mastering the game of go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[5] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," arXiv preprint arXiv:1506.08941, 2015.

[6] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[7] J. Zhai, Q. Liu, Z. Zhang, S. Zhong, H. Zhu, P. Zhang, and C. Sun, "Deep q-learning with prioritized sampling," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 13–22.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[9] Karpathy, "Reinforcejs framework," https://github.com/karpathy/reinforcejs , accessed: 2016-12-04.

[10] R. M. Neal, "Annealed importance sampling," Statistics and Computing, vol. 11, no. 2, pp. 125–139, 2001.