Experience Diversity in Deep Q Networks

Kyriakos Chiotis (35278597)

***Abstract*—This research paper is focusing on Deep Q Learning. The purpose was to analyse the experience replay of a Deep Q Network(DQN) and to interpret its effects on Q learning. The experimentation and analysis of the various methods were applied on the environment of Cartpole. These methods consist of a basic implementation of a DQN and therefore some adjustments and variations are examined. These variations concern the way an agent structures his memory and samples his experiences. Specifically, it was observed that after some high score achievements the agent was performing bad. That is, the agent has overestimated his Q-values and the memory was filled with experiences regarding the overestimated observation space. Because of random sampling, rich experiences that occur rarely have practically no chance to be selected. Prioritized Experience Replay(PER) was introduced in 2015 by Tom Schaul to encounter the aforementioned problem. PER is trying to change the sampling distribution by using a criterion to define the priority of each tuple of experience. Based on this idea, a new sampling method was applied. That is, for each feature of observation space we sample the quantiles. This way, the training batch contains experiences across the entire stored observation space and the network fits with experience diversity. However, this experience diversity is limited because memory is filled with common experiences.**

To deal with this problem, the memory structure of the agent is changed. Specifically, the quantiles are converted in clusters. That is, experiences are distributed to clusters according to the distance among them. Then, the agent samples each cluster randomly and equally. In addition, this procedure is modified for further experimentation. That is, the agent does not sample the memory randomly, but he samples according to the weighted average distance between his current state and the states on the center of the clusters.

These modifications derived after many experiments, not only on the memory structure, but also on the way that an agent explores his environment. Finally, after a deep study on these techniques, it is discussed a Multi-agents system that is adjusted on this new memory structure.

I. Introduction

Reinforcement Learning (RL) Agents can be quite fickle. This is because the environment for an Agent is different than that of Supervised and Unsupervised algorithms.

Supervised / Unsupervised	Reinforcement Learning
Data are gathered.	Data are simulated.
Training data cover many situations.	Sparse data depend on exploring and exploiting.
The environment is assumed static.	The environment is changing in response to the agent.

These differences lead to the main challenges of a successful RL agent. Hyper-parameter tuning is considered crucial as it not only impacts the training of the agent's neural network, but it also impacts how the data are gathered through simulation. However, more generalized methods were developed to encounter these challenges. Some of these methods regarding DQNs are: Double Deep Q network(DDQN)[1], Dueling Double Deep Q network(D3QN)[2] and Prioritized Experience Replay(PER)[3]. Therefore, keeping the hyper-parameters constant as much as possible, we can apply and analyze different methods and their combinations. Following a deep study of these methods, the main problems of a DQN are discovered.

This research paper does not include an experimental analysis of the aforementioned methods. However, it is pursued a more creative approach and custom methods are developed and analysed. These methods are trying to enrich the memory with some diversity for the observation space. The most crucial components for this task are the trade-off between exploration and exploitation and the memory sampling. The most famous algorithm that deals with exploring and exploiting is Epsilon-Greedy. Even if it is incredibly simple, often outperforms more sophisticated algorithms such as UCB ("upper confidence bound")[4]. Inspired from the simplicity of Epsilon-Greedy, it was tested a modified algorithm. Specifically, an agent starts operating according to Epsilon-Greedy algorithm. However, when he reaches his average score, he starts increasing linearly his epsilon until a maximum. One of the pros in exploring, is that experiences in a memory cover higher diversity of the stored observation space(i.e cover many situations). This diversity helps a neural network to better fit the data. This sequential logic led to a new research question. How an agent could sample his memory taking into consideration the entire observation space and not creating bias? Consequently, we take quantiles for each feature and and we sample each quantile uniformly. Even then, in some occasions quantiles cannot be created because some features have very small diversity or there is a big gap between the feature values. To deal with the aforementioned problems quantiles are converted in clusters. This means that experiences are categorized based on their distance and random sampling is applied on clusters. Furthermore, it is tested weighted sampling according to the weighted average distance between agent's current state and the states on the center of the clusters.

The aforementioned modifications were tested on Cartpole environment. Even if the maximum episode steps were increased at 4000, this environment is not the best choice because of the limited outputs of the game. However, the quantiles and clusters sampling methods required high computational resources and further experimentation could be done in future research.

II. Related Work

A. Deep Q Learning

Richard Bellman landed research on Markov Decision Processes(MDP) and his first breakthrough is known as Value Iteration[5]. The goal is to find the value of a state (V), given the rewards that occur when following an action in a particular state (R). That is the Bellman equation:

$$V(s_t) = \max_{a_t}(R(a_t, s_t) + \gamma V(s_{t+1}))$$

Bellman equation says that the value of agent's current state is the discounted(γ) value of the next state plus the rewards picked up along the way, given that agent acts to maximize this. However, in real world problems, an agent does not know how many potential states exist in his environment neither the outcome of his actions. This is solved with Q-Learning[6] and Q equation:

$$Q^{new}(s_t, a_t) \leftarrow (1-a)Q(s_t, a_t) + a(r_t + \gamma \max_a Q(s_{t+1}, a))$$

This equation creates a Q-table which stores the values of states given the actions. The learning rate is how much the agent wants to change his Q values during simulation. This approach is useful in deterministic problems. However, in stochastic problems with multidimensional states an agent should predict the value of the next state given the current state and action. In this situation, deep learning and neural networks are considered useful and Q-Learning is changed to Deep Q Learning.

B. Prioritized Experience Replay

In 2015, Tom Schaul introduced Priority Experience Replay(PER)[3]. The central idea is that some experiences occur rarely and they are important for fitting the network. Therefore, each experience has a priority which is estimated according to the absolute difference between the predicted and actual Q-value(i.e. TD error).

$$p_t = |\delta_t| + e$$

However, prioritized experiences create bias and the probability to replay an experience is normalized by all the priority values in memory.

$$P(i) = \frac{p_i^a}{\sum_k p_k^a}$$

PER is considered very effective in DQL but its effectiveness is related with memory size. This means that a limited memory size may not include these rare valuable experiences.

III. Methodology

The experimentation was implemented in Python programming language. The environment was established by OpenAI Gym and the network was built with Tensorflow 1.13. It must be noted that all the following methods were applied in order to increase the diversity of experiences in the memory despite its size. Also, experimentation indicated that some methods are not working so well. However, each method derived from the previous one and I would like to mention all of them.

A. Cartpole Environment

This environment corresponds to the version of the cartpole problem described by Barto, Sutton, and Anderson[7]. Cartpole is a cart with a pole on its roof top. The original goal of this 'game' is to balance the pole for 500 frames by increasing and reducing the cart's velocity. However, this goal was modified for the purposes of this project and it was increased at 4000. This game could be described as a Markov Chain Process as follows:

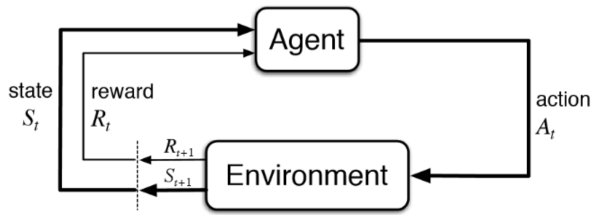
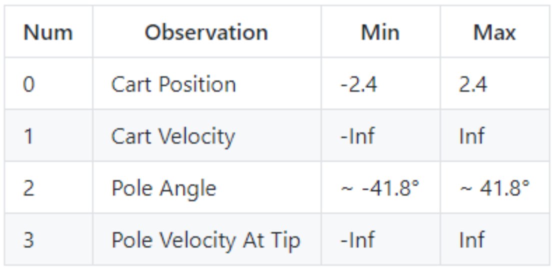


Fig. 1. Markov Chain Process.

The action space consists of two discrete actions, left and right movement(0 or 1). The observation space contains four continuous features assigned a uniform random value between ±0.05. Their description is given below:



Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Fig. 2. Cartpole observation space.

The reward is +1 for every step taken, including the final step. However, in this implementation the reward of the final step was changed to zero. Finally, the episode terminates when the pole is more or less than ±12°or the cart position is more or less than ±2.4 or the episode length is greater than 4000(changed from 500).

A DQL agent easily solves this problem with all the modifications that were applied. However, additional challenges could be enumerated as follows:

1) The consecutive successful trials.
2) The stable learning process of the algorithm(i.e the variance of the episodes score).
3) The average reward across the episodes.

The actual challenge for the agent was to intervene his average score that was fluctuated between 200 and 250. The visual replay of the agent shows that he learns balancing quite fast but the problem is that he overrides the terrain limits. Observing the memory cells indicated that the memory was filled mostly with experiences of low diversity. Therefore, the samples that were used for fitting the network did not contain situations covering the extreme values of the observation space. Of course one way to deal with this, was to increase the memory size and implement hyper-parameter tuning to our network. However, as it was mentioned, the challenge is to use more generalise methods. These are the modified exploring-exploiting method, the quantiles sampling and the clusters sampling method.

B. *The Network*

Our network is a simple neural network. It takes state as an input and outputs the actions. Even if the focus of this research is not the deep learning, it is essential to mention the hyper-parameters of the implemented neural network. It consists of 2 hidden layers with 64 hidden neurons each. All layers use the ReLU activation function and the estimator is the mean squared error(MSE). Finally, it uses the Adam optimizer because it adapts the learning rate and we do not want to get occupied with tuning the parameters. The main difference is that it uses a specialized loss function which is the derivative of the Bellman Equation. However, the Bellman equation is not actually in the network. Consequently, the output is not the discrete action space but the Q values of the states given the actions. This peculiarity makes the network a Deep Q Network.

C. *Exploring-Exploiting*

It is well-known that in RL, the environment is changing in response of the agent. This means that the network is not trained with static data that cover the environment's observation space. Therefore, it is crucial for an agent to explore his environment and then exploits his experience according to the predicted Q values. At the beginning, the agent starts only exploring until he has filled his memory with some experiences. The question that follows, is what would be the the trade off between exploration and exploitation after the first stored experiences. The usual approach is the aforementioned Epsilon-Greedy algorithm. However, it is tested an algorithmic modification which could be represented as:

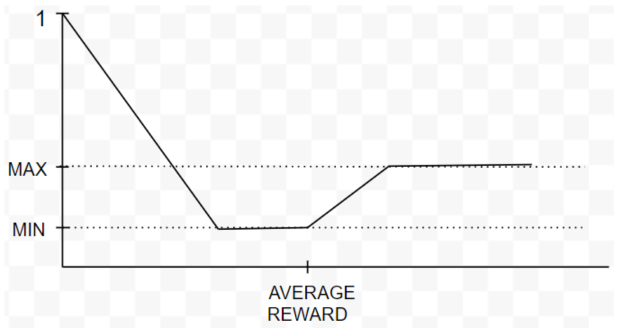


Fig. 3. Trade off between exploration and exploitation.

Figure 3 shows how the probability of choosing a random action(epsilon) is changing across the episode scores. Initially, it decays until a minimum and when the agent overrides his average score it increases until a maximum. If the agent succeeds his average score before the epsilon reaches the minimum, it remains constant or it increases until the maximum. This implementation has four parameters: The minimum epsilon, the maximum epsilon, the decay rate and the increment rate.

D. *Quantiles Sampling*

Till now, it was implemented the network(”brain”) and the exploration-exploitation trend of the agent. The last component is the memory. The memory stores experiences in a tuple structure such as (state, action, next state, reward, done). Just like the other neural networks, it is essential to randomly sample, so that the agent can learn with low bias considering the whole observation space. A famous and computationally efficient way to sample the memory is the uniform random sampling. However, it was observed that a limited size memory is filled with similar experiences and therefore random sampling does not cover many situations. Consequently, it was applied a different algorithmic approach in sampling method. That is, for each feature in observation space, cut the memory in evenly spaced bins. The size of each bin is the batch size divided by the total number of features. Then, for each bin that corresponds to a feature, the agent randomly sample one experience. During the experimentation, the memory had a storage size of 2000 experiences and a batch size of 64 experiences. Each observation in Cartpole has four features and therefore the agent will sample bins of 16 experiences. It was observed that sampling quantiles in a multidimensional space requires high computing resources. Dimensionality reduction methods such as PCA, would be considered a good approach for experimentation in more complex environments.
During execution, error handling caught that bins could not be created because there was no diversity between the experiences or there was a big gap between the feature values. This could be considered a positive sign that the cart stays totally still. But this is not a permanent solution because there is some randomness and the network probably ”forgot” what is happening in these situations. Of course one solution is to increase the memory size but the point is to keep it constant.

E. *Clusters Sampling*

The problems that were arisen during the above methods led to a new idea, clustering the memory. Initially, we set the number of clusters. When the agent starts his training, he places each experience in a new cluster of a deque[8] structure until we have the maximum number of clusters. Each of these first experiences is the center of the cluster. Therefore, whenever the agent adds a new experience in his memory, he tests the Euclidean distance between the new experience and the center of each cluster and he adds the experience in the closer cluster. If the cluster is full, the deque structure pops an experience and the following experience in the deque becomes the new center of the cluster. This means that clusters are changing dynamically. A visual representation could be as follows:

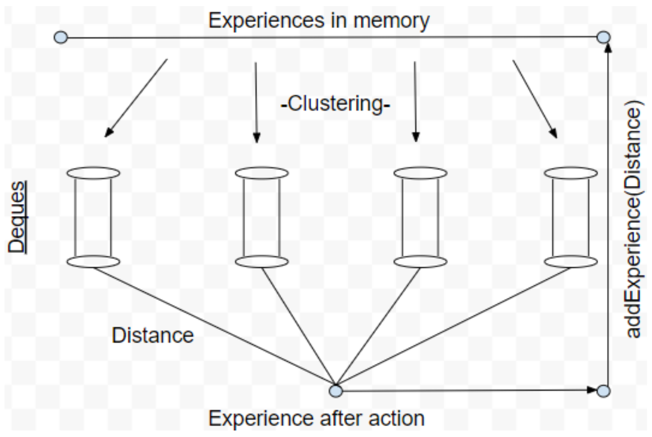


Fig. 4. Clustering the experiences.

Afterwards, the sampling procedure is the usual. The agent samples equally and uniformly each cluster according to the batch size. For example, if there are 4 clusters and the agent wants to sample a batch size of 64 experiences, he samples 16 experiences from each cluster.

F. Weighted Sampling in Clusters

The final approach was to change the uniform sampling method. That is, the agent is sampling experiences according to his current state. Specifically, he measures the Euclidean distance between his current state and the state of each cluster center. Then, he samples each cluster according to the normalised reversed weighted average for the distances or the reciprocal of the weighted average for the distances.

$$1 - Weight_c = 1 - \frac{\sqrt{\sum_{i=1}^{n}(s_i^c - s_i)^2}}{\sum_{c=1}^{k}\sqrt{\sum_{i=1}^{n}(s_i^c - s_i)^2}}$$

and the weights are calculated as:

$$Weight_{norm}^c = (1 - Weight_c) \times \frac{1}{\sum_{c=1}^{k}(1 - Weight_c)}$$

$$Weight_{rec}^c = \frac{\sqrt{\sum_{i=1}^{n}(s_i^c - s_i)^2}^{-1}}{\sum_{c=1}^{k}\sqrt{\sum_{i=1}^{n}(s_i^c - s_i)^2}}$$

The former calculates a normalised ratio for the distances between the current state and the states on the center of the clusters. The latter calculates the exact ratio but there is a risk of bias. However, it could be considered controversial the use of Euclidean distance as a metric between the current state and the center of the cluster. That is because an experience, which includes the state, has more dimensions than a single state.
Weighted sampling in clusters means that the agent tries to remember experiences that are correlated to his current state and then act accordingly. Finally, a Multi-agent system could be applied but this is analysed in section V.

IV. Results

Before analysing the results, some clarifications need to be pointed. Hyper-parameter tuning was skipped because of computational resources. This means that a method may be more or less effective than an another if the parameters were tuned. For example, in the case of clusters sampling, bigger memory size may increase the performance exponentially. Moreover, testing in more complex environments than Cartpole may also indicate interesting results. However, this paper is focusing more in methodology and further experimentation could be done in future research.
Also, the experimentation regards only the training process because every applied method achieves a perfect model in Cartpole environment.

A. Configuration of the experiments

Initially, some abbreviation and parameters are provided to facilitate the presentation of the results:

	Original	Modified Epsilon-Greedy	Quantiles Sampling	Reciprocal Weighted Clusters	Normalised Weighted Clusters
Abbre	cl1	modE	quant	rec4 rec8	norm4 norm8
Memory Size	2000	2000	2000	2000	2000
Batch Size	64	64	64	64	64
Clusters	1	1	1	4, 8	4, 8
Epsilon Decay	0.001	0.001	0.001	0.001	0.001
Epsilon Min	0.001	0.001	0.001	0.001	0.001
Epsilon Max	None	0.1	None	None	None
γ	0.95	0.95	0.95	0.95	0.95
Optimizer	Adam	Adam	Adam	Adam	Adam
Layers	2	2	2	2	2
Neurons	64	64	64	64	64

Table 1: Configuration

B. Analysis

As it was mentioned all the methods were developed in a sequential way for further research on experience diversity. The last method, clusters sampling was the most successful one according to the results. For this reason, the following graphs are focusing more on this method.

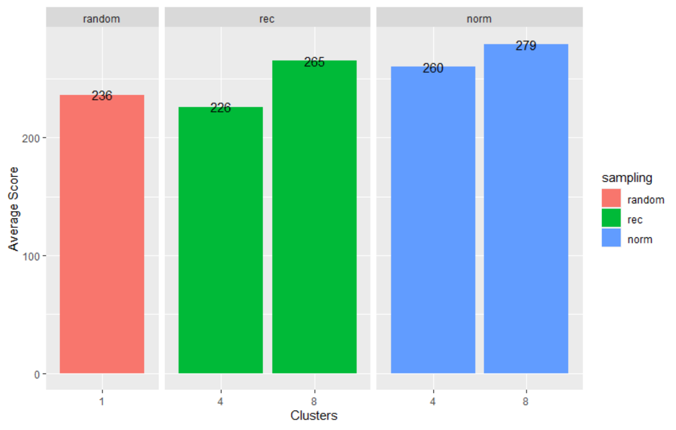


Fig. 5. Bar plot of the average scores.

Figure 5 indicates that clusters sampling achieves a better average score than the original DQN. Specifically, the reciprocal weighted average method with 4 clusters seems to be worse. Probably, this happens because of the bias. However, increasing the clusters to 8, the method outperforms the original. The best results were observed by normalised weighted average with 8 clusters.

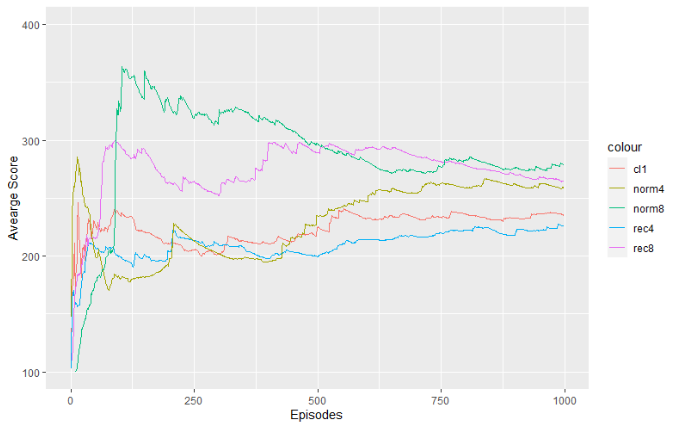


Fig. 6. Average score through the episodes.

Figure 6 shows how the average score fluctuated through the episodes. It is noteworthy that norm8 achieved also the max average score during the training process. This means that norm8 is our best method. In the following graphs we observe the effectiveness of norm8 against the original DQN.

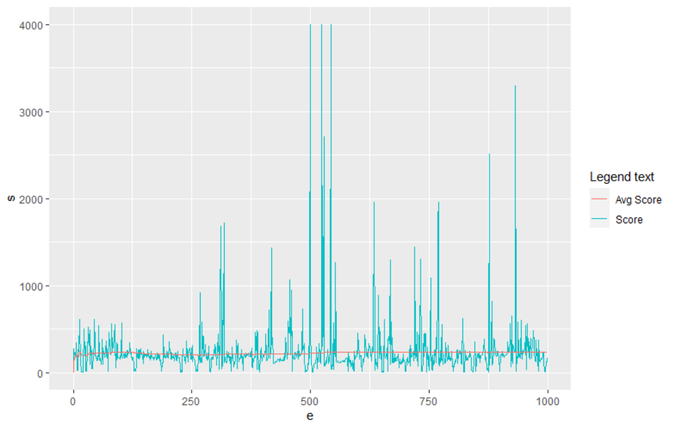


Fig. 7. Overview of the original DQN.

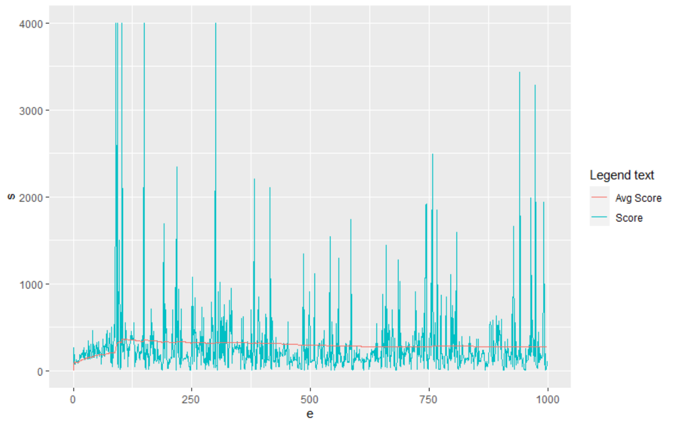


Fig. 8. Overview of the norm8.

Finally, for typical reasons we provide the bar plots of modified E-Greedy and quantiles sampling methods. These experiments were delivered in a different way and they cannot be compared with clusters sampling. None of them was considered better than the original.

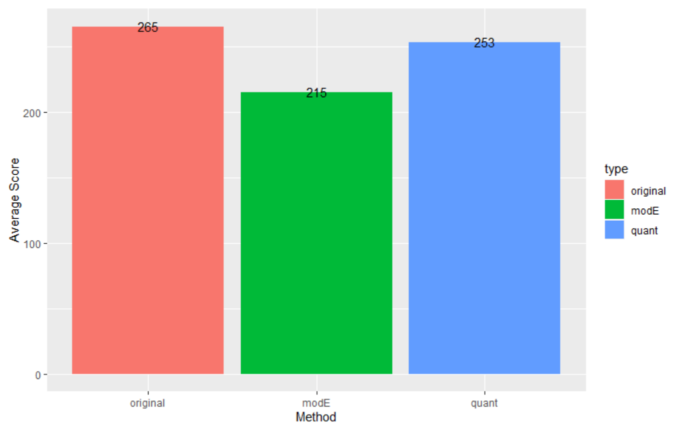


Fig. 9. Modified E-Greedy and Quantiles Sampling.

V. Discussion

According to the results, memory clusters seem very promising in DQN. However, the experimentation is not complete. Hyper-parameter tuning and testing in different environments is required to make the final conclusions. Finally, dynamic memory structure could be also expanded in a Multi-agent system. This is illustrated in the following figure:

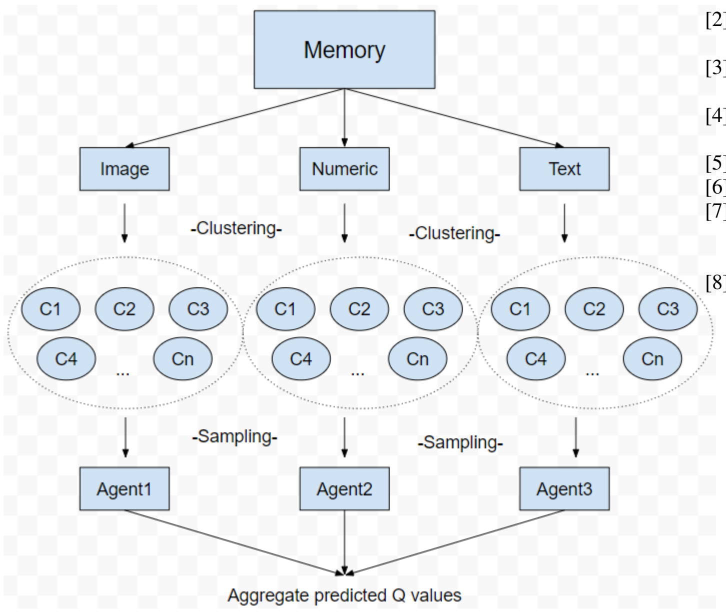


Fig. 10. Multi-agent memory structure.

The concept is that there are different mechanics analyzing the incoming information. For example, a CNN agent fits better an image. For example, let's consider an agent that plays poker and a common expression in a poker table is poker face. The agent could probably understand the bluffing or other tricks and in parallel training for the probabilistic outcomes of the game, he acts accordingly.

VI. Conclusion

This research paper in analysing how experience diversity in memory impacts the training process of a DQN agent. The two major components are the trade-off between exploration and exploitation and the memory structure. Exploration was pursued in the entire process with a modified E-Greedy algorithm. This method did not outperform an original DQN in Cartpole environment. On the other hand, modifications in memory structure led to more positive results. The first method was the quantiles sampling which operated almost the same with a DQN agent. However, it required a lot of computational resources. The second and final method in memory structure was clusters sampling. On this method the agent managed to keep stored most of the experience diversity despite the memory size. Also, the agent was perceived more effective than an original DQN. Taking everything into consideration, clusters changed the serial 2-dimensional memory buffer into a multidimensional structure. Furthermore, weighted sampling the memory simulates a human. Our interaction with the environment reminds us situations in the past that somehow are correlated with out current state. This correlation is not always obvious but plays a deterministic role in the way we act.

References

[1] Hado van Hasselt, Arthur Guez, David Silver, 2015 Deep Reinforcement Learning with Double Q-learning

[2] Google DeepMind, London, UK, 2015 Dueling Network Architectures for Deep Reinforcement Learning

[3] Tom Schaul, John Quan, Ioannis Antonoglou, David Silver, 2016 Prioritized Experience Replay

[4] Peter Auer, 2002 Using Confidence Bounds for Exploitation-Exploration Trade-offs

[5] Richard Bellman, 1957 A Markovian Decision Process

[6] CHRISTOPHER J.C.H. WATKINS, 1992 Technical Note Q-Learning

[7] Andrew G. Barto ; Richard S. Sutton ; Charles W. Anderson, 1983 Neuronlike adaptive elements that can solve difficult learning control problems

[8] https://docs.python.org/2/library/collections.htmlcollections.deque