UKRAINIAN CATHOLIC UNIVERSITY

FACULTY OF APPLIED SCIENCES

DATA SCIENCE MASTER PROGRAM

Agent adaptation to a changing environment

Machine Learning Project

Authors:
Oleg DATS
Dmitri GLUSO

May 18, 2019

Український католицький університет

APPLIED
SCIENCES
FACULTY

Abstract

In this project [2], we analyze the ability of the Reinforcement Learning (RL) [9] agent to generalize experience that he receives from the environment.

Research from the "Assessing Generalization in Deep Reinforcement Learning" paper [8] was used as a base for this project.

First, the Q-learning [4] was implemented and applied to FrozenLake environment from OpenAI Gym [6] to learn and experiment with RL approaches. Then Deep Q-Network (DQN) [3] from OpenAI Baselines GitHub [5] was used to make experiments on the CartPole environments of 3 types: Default (Deterministic), Random and Extreme. CartPole environment was used from Sunblaze environments GitHub [10]. Finally, DQN was used to inspect generalization abilities on Pong environment from OpenAI Baselines Atari Wrappers. Noise and pepper were manually added to the environment to simulate Random and Extreme Environments. DQN was learned on pixel features from the image.

Experiments showed that learning on the Random type of environment increases the learning time, but also increases the performance on Random and Extreme environments improving the ability to generalize.

1 Introduction

Nowadays Machine Learning achieved a lot of great not only theoretical but also practical results. Image Classification, Recommender Systems, various NLP applications, time series predictions and etc. And it is still far from the limit. A special place in the Machine Learning tasks takes the goal to explore and build the Artificial General Intelligence (AGI) [1]. Currently, AGI is not achieved yet. Reinforcement Learning (RL) [9] is considered as the closest approach to achieve AGI nowadays. But there is a lot of different problems in the RL to achieve AGI. One of the big problems is that the RL agent tends to overfit.

2 Reinforcement Learning

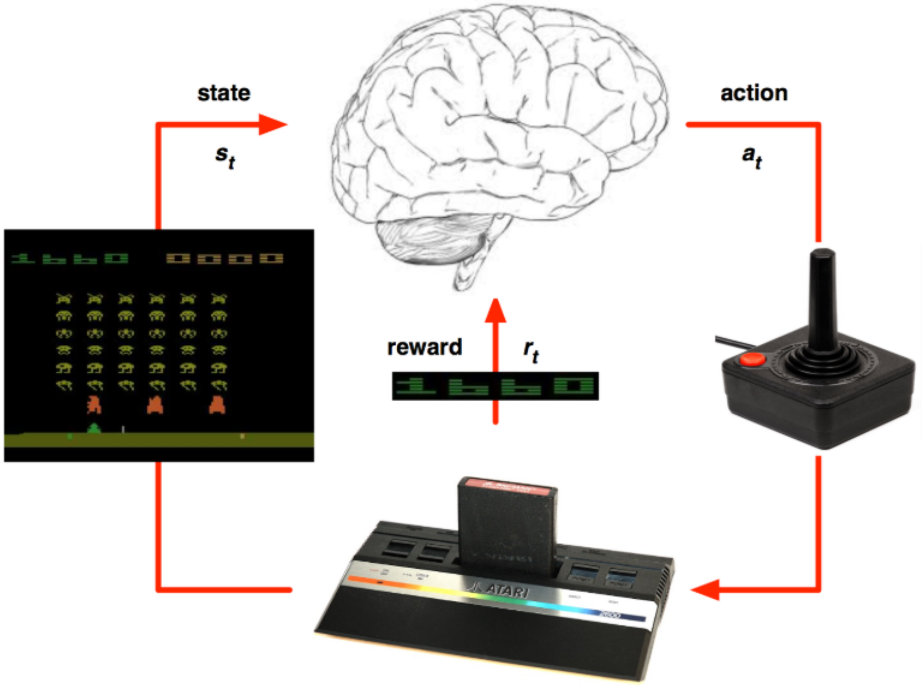


Figure 1: Agent interaction. Used from DeepMind Youtube video course.

In the RL environment agent interacts with the environment and tries to maximize the cumulative reward. Agent receives the state and the reward from the environment and interacts with it using actions.

R_t - **reward**

A_t - **action**

O_t - **observation**

H_t - **history**. $H_t = A_1, O_1, R_1, \ldots, A_t, O_t, R_t$

S_t - **state** - information used to determine what happens next. $S_t = f(H_t)$

The environment is described as a Markov Decision Process (MDP), that is defined by the $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where:

1. $\mathcal{S}$ is a (finite) set of states
2. $\mathcal{A}$ is a (finite) set of actions
3. $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$
4. $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = E[R_{t+1} | S_t = s, A_t = a]$
5. γ is a discount factor, $\gamma \in [0, 1]$

G_t - **return** - is the total discounted reward from time-step t. $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

$v_\pi(s)$ - **state value function** of MDP - $v_\pi(s) = E_\pi[G_t | S_t = s]$

The state value function represents how good is a state for an agent to be in. It is equal to the expected total reward for an agent starting from state s.

$q_\pi(s,a)$ - **action value function** of MDP - $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$

The action value function is an indication of how good it is for an agent to pick action a while being in state s.

$v_*(s)$ - **optimal state-value function** - $v_*(s) = \max_\pi v_\pi(s)$

$q_*(s,a)$ - **optimal action-value function** - $q_*(s,a) = \max_\pi q_\pi(s,a)$

Bellman Optimality Equation for MDPs:

$v_*(s) = \max_a q_*(s,a)$

$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$

Combining 2 parts:

$v_*(s) = \max_a q_*(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s'))$

$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$

3 Problem Setting

Overfitting problem is well-known to the Machine Learning world. The problem of overfitting can be explained in the way that the model perfectly predicts the model it was trained on and because of that loses the ability to predict properly on the data the model has not see. Another way to observe the overfitting is when the model is sensitive to small perturbations.

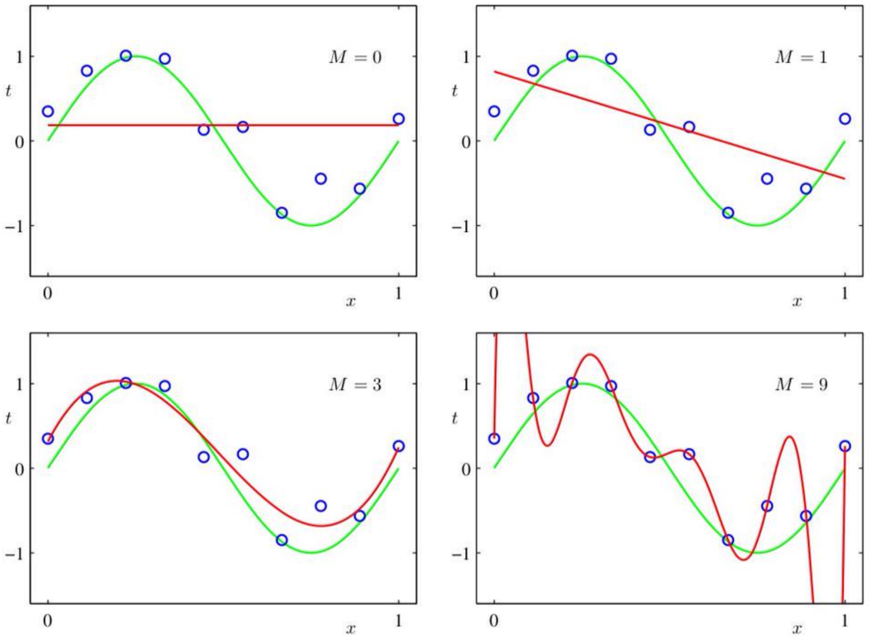


Figure 2: Overfitting picture from Christopher M. Bishop "Pattern Recognition and Machine Learning"

The main reason for the overfitting is that the model is too complex for the dataset it is trained on. And there are a lot of different solutions to overcome this problem:

- Simplify the model
- Greatly increase the dataset

- Data augmentation
- Cross-validation
- Add regularization
- Dropout
- Other techniques that improve predictions on the test or unobserved data

In the terms of the RL agent, if the agent overfits that means that he cannot perform on the states that he did not see during training. Usually, agents are trained on the same environments on which they are applied further, it means that they train on the test set. And if the environment will change or if we want to use the agent's knowledge on a similar environment (with some differences) then usually the agent will fail. A good example is the maze environment: agent can learn how to go through the maze to the finish, but when the maze will change then the agent will fail.

This is a big problem for AGI creation, as compared to the human, a human can easily apply experience received from the one task one many other different tasks because a human can generalize. He can understand the main patterns that are important for the specific setup and then reuse this pattern in similar situations with some differences. That is why this project is focused on analyzing the ability of the agent to generalize.

4 Related Work

We have analyzed the current progress of generalization in Deep Reinforcement Learning field. There is a paper [8] by Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krhenbhl, Vladlen Koltun, Dawn Song from Berkeley that presented a good foundation to measure and analyze generalization. They have made an empirical study of generalization in Deep Learning. They implemented deterministic, random and extremely random environment setup to train on different environment setup and compare the performance using different algorithms. They compared vanilla deep RL algorithms with algorithms specialized for generalization task and empirically approved that training vanilla deep RL algorithm with environmental stochasticity may be more effective for generalization than specialized algorithms. We decided to continue their work and follow the presented approach. Train on Deterministic and Stochastic versions and test on Deterministic, Stochastic and Extremely Stochastic versions of the same environments. Generalization is a good performance of Stochastic and Extremely Stochastic versions.

5 Solution

As mentioned before, the agent will fail if the environment will look a little bit different from the one that the agent was trained on. So, first of all, we want to prove this statement. Then we will try to overcome this problem by making environment features representation more random, for example, sample the characteristics of the environment's objects from some distribution, and try to train the agent on such environment. Our hypothesis is that in such case the agent will not have the possibility to overfit as much as with the stable environment and because of that agent will generalize and perform better on such a random environment.

We follow convention from the mentioned paper [8]:

- D - Default (Deterministic) environment. It's a stable environment, parameters are fixed at the default values. There is no noise.

- R - Random (Stochastic) environment. The parameters are changing every time the environment is reset.

- E - Extreme. Same as R, but with a much higher variance of the parameters or the noise.

- DR - trained on D, validated on R

Due to the hierarchical structure of our work we have decided to break the project on the next pipeline:

- Study RL and develop a Q-learning: model-free and the most basic value based learning algorithm. Test on a simple environment like Frozen Lake from OpenAI Gym. 'Q-learning FrozenLake.ipynb' file [2].

- Use DQN from OpenAI Baselines Github to train the model on the CartPole environment. It is not possible to use Q-learning for such environment as it has continuous state/action space. Train on Default and test on Default - DD.

- Use the environment with random and extremely random parameters (sunblaze environments [10]) to validate the poor performance of the trained DQN - DR, DE.

- Retrain the agent on the R environment and validate on other types - RD, RR, RE. 'DQN_for_Sunblaze_CartPole_(DRE),_openai_baseline.ipynb' file [2].

- CartPole uses only 4 features as the input, that is why we will go further and apply the same approach to the Pong (Atari game) using the state represented by pixels.

- Modify Pong environment so the noise can be added to the picture, modify DQN so it can be trained on the new environment with the noise.

- Apply the same approach of train-test as for CartPole: first DD, DR, second RD, RR. 'Pong_game_play.ipynb' file [2].

For training the DQN on more complex environment represented by pixels we have moved our training to the Cloud (AWS) to train using modern GPUs.

Also, the first implementation of adding noise to the environment took around 0.01 seconds to add noise to one image and it resulted to about 3 additional hours of training of the DQN as training on the simple Atari environment requires at least 1 million frames. That is why we improved the algorithm to add noise 30x times faster to not affect training time dramatically.

6 Evaluation

6.1 CartPole

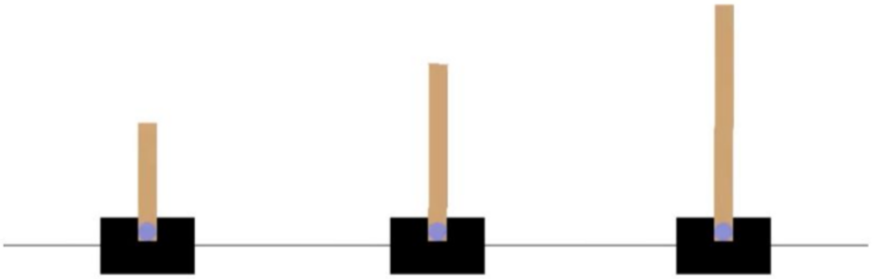
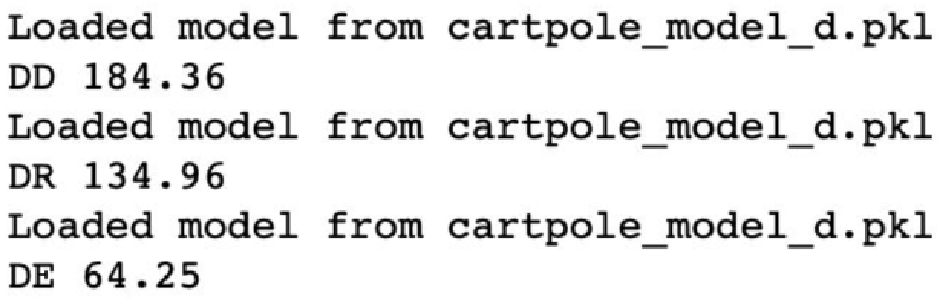


Figure 3: CartPole

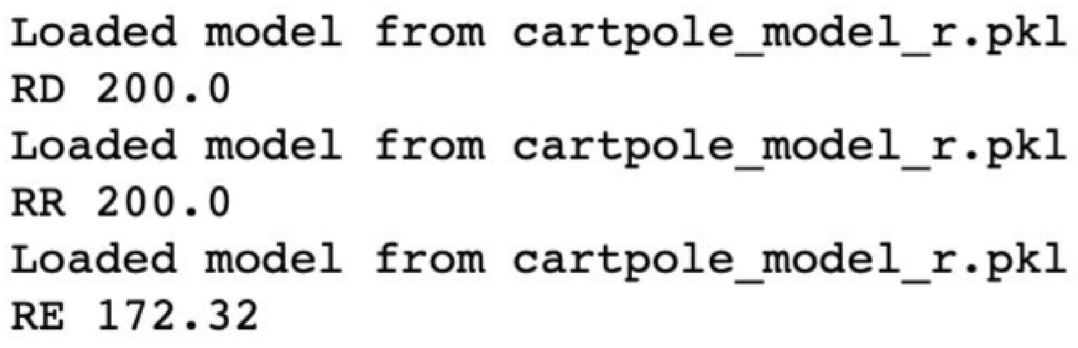
As was expected DQN trained in the D environment has bad performance on the R and E environment. The model was training on CPU for around 45 minutes.



```
Loaded model from cartpole_model_d.pkl
DD 184.36
Loaded model from cartpole_model_d.pkl
DR 134.96
Loaded model from cartpole_model_d.pkl
DE 64.25
```

Figure 4: CartPole DD, DR, DE showing poor results on the stochastic environment

After we trained an agent on the R environment, the agent performed much better on the R and E environment. The model was training on CPU for around 45 minutes.



```
Loaded model from cartpole_model_r.pkl
RD 200.0
Loaded model from cartpole_model_r.pkl
RR 200.0
Loaded model from cartpole_model_r.pkl
RE 172.32
```

Figure 5: CartPole RD, RR, RE showing good results on the stochastic environment

6.2 Pong

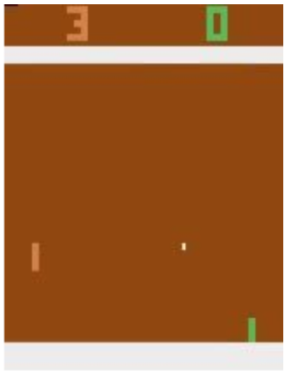


Figure 6: Pong

We added a function that adds stochasticity to the environment.

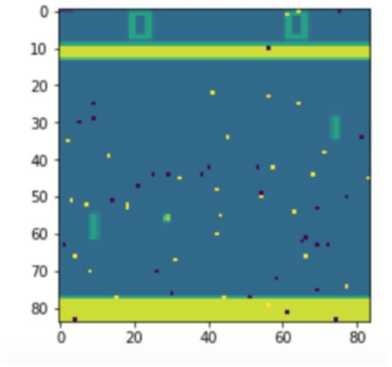
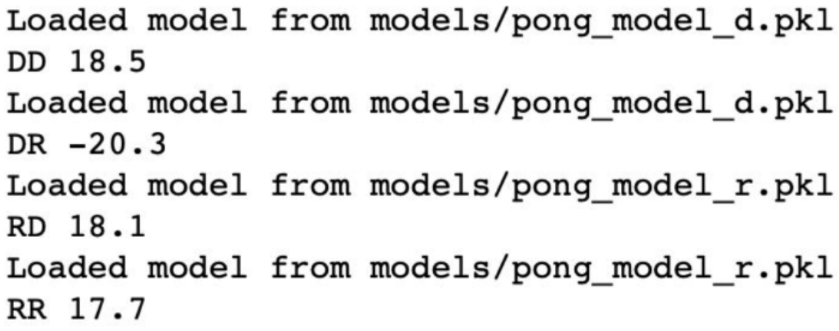


Figure 7: Pong environment with stochasticity

Training agent on the D environment took 2 hours on the NVidia Tesla P100 (AWS instance). The agent performed well on D and very bad on R. After that we trained an agent on the R environment and it took around 5 hours. The second agent performed very well on D and R environments.



```
Loaded model from models/pong_model_d.pkl
DD 18.5
Loaded model from models/pong_model_d.pkl
DR -20.3
Loaded model from models/pong_model_r.pkl
RD 18.1
Loaded model from models/pong_model_r.pkl
RR 17.7
```

Figure 8: Pong results

7 Conclusions

Our hypothesis was that an agent that is trained on a stable environment will have poor performance on the environment with stochastic objects and noise. We empirically approved our hypothesis on two environments with different interaction approach: we obtained the same results on CartPole environment with 4 features and Pong environment with 84x84 pixels features. Also, the same result was approved in "Assessing Generalization in Deep Reinforcement Learning" paper [8]. The training procedure is much more important than complicated algorithms. Creating a stochastic environment with noise improves the ability of the agent to generalize and perform better in such environments.

8 Future Work

Despite that fact that we received good results improving generalization of the agent, there is still a lot of work to be done in the generalization direction to achieve a human level. A lot of more studies can be made using the current result and improving it:

- Try to learn in the deterministic environment and continue to learn in the stochastic environment to reduce training duration
- Try different stochasticity adding not only noise but making changes in some environment rules leaving the same best policy to force an agent to learn behavior patterns instead of overfitting on some states and try to apply it on CoinRun OpenAI environment for generalization quantification [7]
- Add stochasticity to the self-driving car virtual environment to improve agent performance when it will be transferred to the real environment
- Reuse agent's knowledge and experience from one environment in another having similar behavior patterns in similar situations

References

[1] "Artificial general intelligence". In: (). URL: https://en.wikipedia.org/wiki/Artificial_general_intelligence.

[2] Oleg Dats and Glushko Dmitrii. "Agent adaptation to a changing environment". In: 2019. URL: https://github.com/odats/rl-generalization.

[3] Jonathan Hui. "RL - DQN Deep Q-network". In: (2018). URL: https://medium.com/@jonathan_hui/rl-dqn-deep-q-network-e207751f7ae4.

[4] Vaibhav Kumar. "Reinforcement learning: Temporal-Difference, SARSA, Q-Learning Expected SARSA in python". In: (2019). URL: https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e.

[5] OpenAI. "OpenAI Baselines GitHub". In: (). URL: https://github.com/openai/baselines/tree/master/baselines/deepq.

[6] OpenAI. "OpenAI Gym". In: (). URL: https://gym.openai.com/.

[7] OpenAI. "Quantifying Generalization in Reinforcement Learning". In: (). URL: https://github.com/openai/coinrun.

[8] Charles Packer et al. "Assessing Generalization in Deep Reinforcement Learning". In: (2018). URL: https://arxiv.org/abs/1810.12282.

[9] "Reinforcement learning". In: (). URL: https://en.wikipedia.org/wiki/Reinforcement_learning.

[10] "RL environments for evaluating generalization". In: (). URL: https://github.com/sunblaze-ucb/rl-generalization.