Neuroevolution of self interpretable agents

October 11, 2022

Andrea Morelli 1845525

1. Abstract

This project aims to investigate the benefits of attention in reinforcement learning to see that also having a reduced view of the environment and simpler models we can create agents able to play procgen games(Karl Cobbe, 2020). The code of the project can be found at this link: https://github.com/bbooss97/attentionrl. A demo of the agent playing the starpilot game can be found here: https://youtu.be/RvH9VFQngG4.

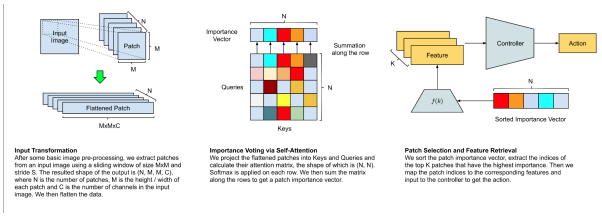
2. Introduction

A lot of parameters : Most reinforcement learning methods used to create agents that acts on environments are based on big models with a lot of parameters to be trained. What if we could reduce those number of parameters to optimize and still be able to perform the task?

Bottleneck : The idea is to divide the input in patches and select just the most important ones that have to be taken into account to take decisions to act on the environment

3. Structure of the network

The structure of the network that is used is divided in 3 parts: a patch selection phase in which we select the most important patches, a patch features extraction phase in which from those patches we get features that will be used by the controller , and a controller that is the part of the network that will produce an action starting from the features of the previous step.(Yujin Tang, 2020)



Email: Andrea Morelli 1845525 <morelli.1845525@studenti.uniroma1.it>.

Deep Learning and Applied AI 2022, Sapienza University of Rome, 2nd semester a.y. 2021/2022.

4. Patch selection

In order to select the most important patches to attend we resolve to the attention mechanism: The attention mechanism allows to learn coefficients for every input in order to see which of those are the most importants.The inputs i and j are firstly transformed with linear transformations with learnable paramethers.Every transformed input i of the attention mechanism is then dot producted with all the transformed inputs j and the result is the attention that i gives to j.This attention map is normalized,scaled and summed across the second dimension.The most important patches are passed to the next phase.This is the normalization and scaling applied to generate the attention map.

$$A = \text{softmax}\left(\frac{1}{\sqrt{d_{in}}}\left(XW_k\right)\left(XW_q\right)^\top\right) \qquad (1)$$

From these summing across the second dimension we can get the patches that are ”the most voted” by the others.

5. Patch features extraction

The patch features extraction phase allows to extract features from the most important patches to be used by the controller to select an action to execute. In the original paper the normalized position of the patches is used. I also used the mean of the channel's colors and the features extracted from a module with learnable parameters (neural network).A combination of the above can and has been used.

6. Controller

The controller is the module that is in charge of selecting an action to be performed in the environment given the features extracted from the most important patches.Different types of controllers with different structures and dimensions were tested. I tried a simple multilayer perceptron,an lstm controller,and an attention controller.The lstm perceptron uses 15 neurons as hidden state that is used to keep track of what the agent was doing before the current moment, while the attention controller uses multihead attention to learn the attention coefficients to give different importance to different parts of the features given at input.

7. Benefits of attention

The attention mechanism(Ashish Vaswani, 2017) is useful in this project because it allows to reduce the number of parameters to train. Having the self attention map we can derive from it the patches that are the most ”voted” by all the others and thus we can infer that those will be the most important ones to perform an action,discarding all the other features that are not important for the task.

8. Training the network

The problem with using attention to retrieve the best patches(sorted) is that the network is not differentiable so we have to resort to genetic algorithms. The genetic algorithm used is called cma-es covariance matrix adaptation evolution strategy(Hansen, 2016).

9. Idea confired sperimentally

Sperientally i tested the network on different games of the procgen environment and i focused especially in training it on the starpilot game even though it performs with good performances in other games.The network is able to play the game and without taking too much time it learns how to perform actions in the environment to maximize the reward.It starts by learning that the first thing to do is to move backwards to diminish the probability of being hit and than learns to move up and down on the left edge to destroy as many objects as possible.If left training it eventually starts to dodge some bullets but never in a human level.It could eventually reach human or superhuman level but i don’t know if the time it would take is long or if the network is not big enough to learn to play in that way.

10. Improvement on the network and the acting

Different improvements were made with respect to the original code for example the vectorized environment has been used to improve the performances.The network to match this difference has been batched in all it’s components and produces in parallel using cuda an action for every game environment generated. The network has as it’s reward after some number of steps the average reward of all the games that have been played in the vectorizd environment.A loss to the deaths has been also added to make the training faster since the agents learns to penalize actions that leads to death. Using this vectorized environment i’m able to play different games in parallel and not in series so significantly improving the training. Other improvements were made with the attention agent that allows to use multihead attention on the features given in input to the controller. I also improved the way features were derived from the patches: over the simple position of the patches i inserted the mean of the color channels and a small neural network that allows to automatically extract from the patches automatic features useful to perform a good action to maximize the cumulative expected reward.

11. Different ideas in this new direction

Different things could be improved in this project in order to make the network perform better. One of the things that could be added is the possibility of using patches with different dimensions.Using patches of different dimension in the self attention mechanism could be useful in case small patches dimensions do not allow to catch important details of the image.In order to make everything works different keys and queries matrices would have to be used. Other improvement could be done in the area of the features that the controller is able to see to perform actions,for example using different neural network to extract automatic features.

12. Performances

The network is not trained in parallel using kuberneetes containers in the cloud as in the paper’s code but i used the batched network with the vectorized environment to mitigate these performances problems. The performances were monitored using weight and biases and all the runs created could be seen in the following link with as artifacts the network weights and as name of the run the parameters of the network: `https://wandb.ai/bbooss97/attentionAgent?workspace=user-bbooss97`. The following are the general performances and differences i’ve experienced while training the networks. The lstm controller in general can’t be too big because the cma-es algorithm is not good in training networks with a lot of parameters, on the contrary the attention controller has a lower number of parameters and performs alright in the starpilot game , training a lot faster than the lstm controller. The mlp is always outperformed by the other 2 architectures. Using the color features slows down the training , meanwhile the automatically extracted features works better than the color features.Using just one or 2 automatic features doesn’t slow the training an achieves the best performances from what i’ve witnessed.Using just the positions of the patches performs also alright but adding automatic features works the best. The number of first best features also determines the time to train and the performances: using 10 first best features takes more time to train than 5 but the performances are in general better. The cma-es starts from a variance of 1 and the mean that are the parameters initialized with pytorch. Using a big population slows a lot the training time so i left the algorithm optimize this value.The initial initialization determines usually how fast we train the network.

Bibliography. Those are the bibliographic references:

References

Ashish Vaswani, Noam Shazeer, N. P. J. U. L. J. A. N. G. L. K. I. P. Attention is all you need. 2017.

Hansen, N. The cma evolution strategy: A tutorial. 2016.

Karl Cobbe, Christopher Hesse, J. H. J. S. Leveraging procedural generation to benchmark reinforcement learning. 2020.

Yujin Tang, Duong Nguyen, D. H. Neuroevolution of self-interpretable agents. 2020.