Independent Study

Report-7, 7th May

Submitted by: Vaibhav Garg(20171005), Kartik Gupta(20171018)

Extending to vector outputs

One extension needed for this environment, over previous environments we tested on, was the fact that the agent has to provide a vector output. So far our implementation had assumed only scalar outputs. To handle this, we extended our library to provide for a more robust system, wherein the user can specify how each element of a vector is composed from various program elements. These values can then be wrapped in a concatenate production. The implementation of this production can either be taken from our existing templates or the user can implement a simple concatenate function on his/her own.

Performance & Analysis on Bipedal-Walker

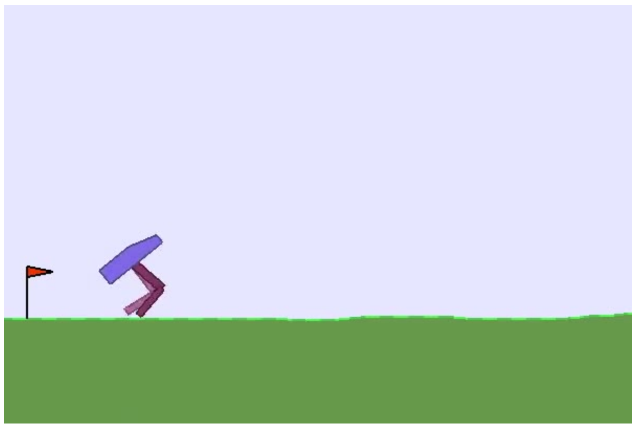

Fig. Bipedal-Walker v2 environment

The Bipedal-Walker environment was one of the complex environments that required a 4-length real vector as output, so we decided to test our modified framework on this. We started off with the following sketch where <if1> represents the root:

```
<if1>;
<if1> -> if leg1_touches_ground then <if2> else <if2>;
<if2> -> if leg2_touches_ground then <ret> else <ret>;
<ret> -> <o1>, <o2>, <o3>, <o4>;
```

<o1> -> c0*hullAng + c1*hullAngVel + c2*hip1Ang + c3*hip1Speed + c4 + c20*velX;
<o2> -> c5*hullAng + c6*hullAngVel + c7*knee1Ang + c8*knee1Speed + c9 + c21*velX;
<o3> -> c10*hullAng + c11*hullAngVel + c12*hip2Ang + c13*hip2Speed + c14 + c22*velX;
<o4> -> c15*hullAng + c16*hullAngVel + c17*knee2Ang + c18*knee2Speed + c19 + c23*velX;

The results using this sketch weren't very promising as the learnt program struggled to even take a complete single step. The curve below shows the decrease in the MSE loss between the program's and oracle's outputs, even though the loss has decreased steadily the reward function didn't really increase for the program. We think the reason for the particularly bad performance is the lack of richness in the program sketch, maybe it is just too simple compared to the neural oracle for the problem at hand and the loss simply hasn't decreased enough.

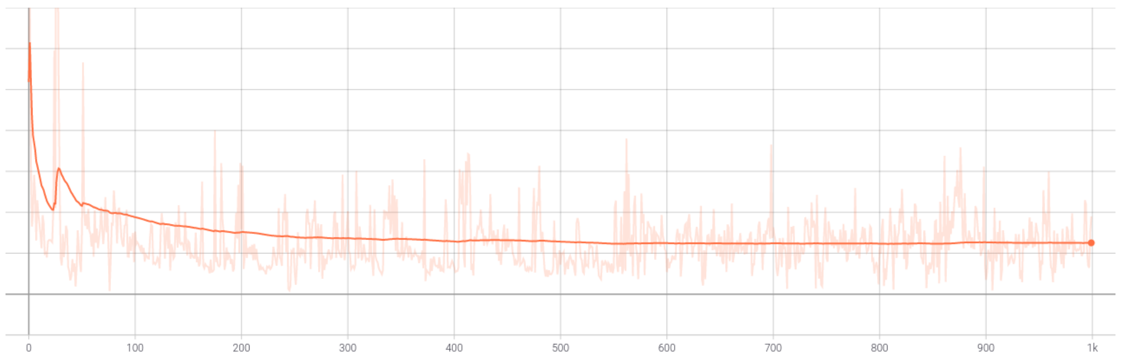

Fig. Decrease in loss

Future Scope

Our written framework is now well capable of generating and optimizing on any given program sketch, however as we saw above, the problem still remains the finding of a suitable sketch which is both interpretable and rich enough to imitate the neural oracle at the same time. An important aspect of the sketch is prior environment knowledge and human expertise, making it difficult to automatically generate a sketch for a given RL problem. Once a decent sketch is found, the results seem to be fairly good, hence one could explore further by coming up with sketches for various different environments.

This sketch is not deep enough to allow for any meaningful neighbourhood search, so the subsequent research can be done to find sketches which also have sufficient recursive depth. If we have a sufficiently rich sketch, we can then also look into techniques like observation buffers, to make the training more stable.

Currently the user has to specify the corresponding tensorflow mapping for each production in the sketch (or he can use an existing default mapping if the production does not represent an active operation). We plan on providing the user with a preexisting library of functions and guidelines on how to write productions to directly use those functions.