ATAI-720-2020 Assignment 3

Summary of results and general remarks

Intro to ONA

We have a few different examples, they will be uploaded to the cadia wiki after this session. They include:

A home, a coffee, a car, a screwdriver and some animals.

Plain Vanilla

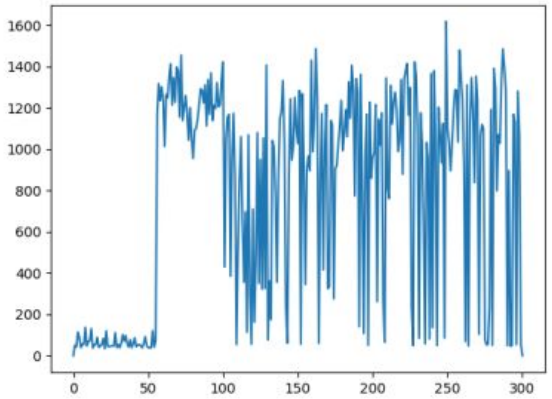
ONA learns differently than the AC.

At first no success, then sudden increase to over 1000 iterations per episode.

For all of you this graph looked identical

-> deterministic

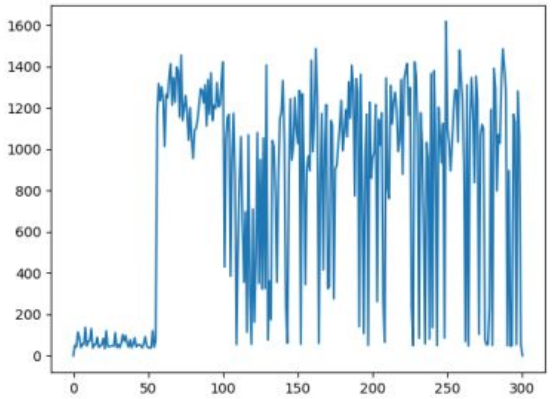
Why is ONA deterministic - Or rather why isn't the AC?



Plain Vanilla

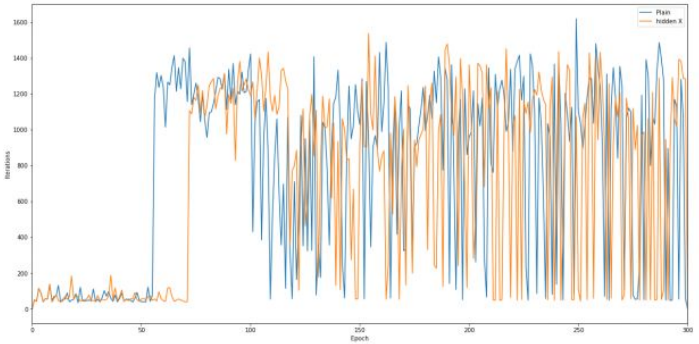
No exploration-exploitation dilemma, ONA does reasoning instead of trial-and-error learning!

-> No random actions chosen

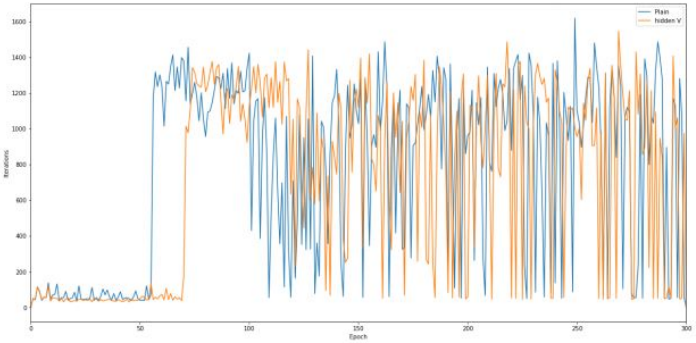


Hidden variables

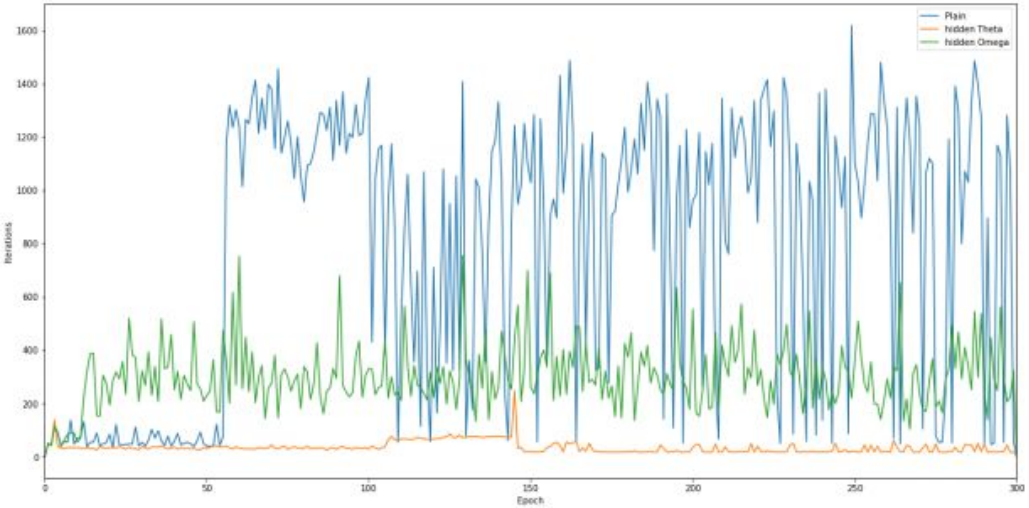
a. Hidden X :



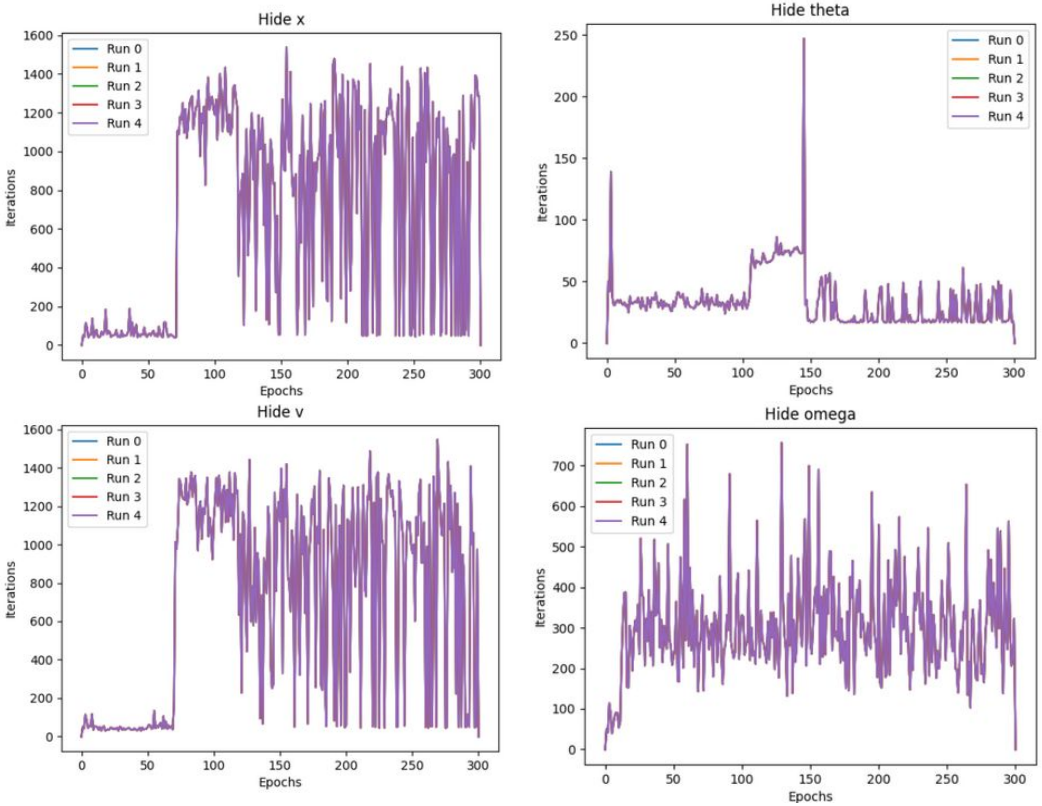
b. Hidden V:



c. Hidden Theta & Omega:



Hidden variables



Again all runs deterministic

With x or v hidden it just takes longer to learn

With theta hidden close to no learning

With omega hidden still averaging around 300 iterations per episode (Better than the win condition of 200 iterations over 100 episodes)

Hidden variables

Which variables are most important for learning the task for ONA?

How does this compare to the AC?

How does this compare to the human learner?

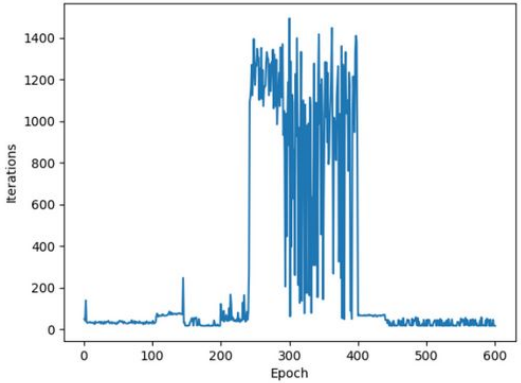
Hidden variables

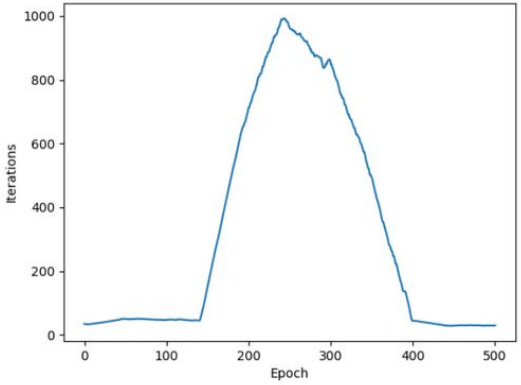
Apparently Theta more important than Omega

For the AC Omega was more important

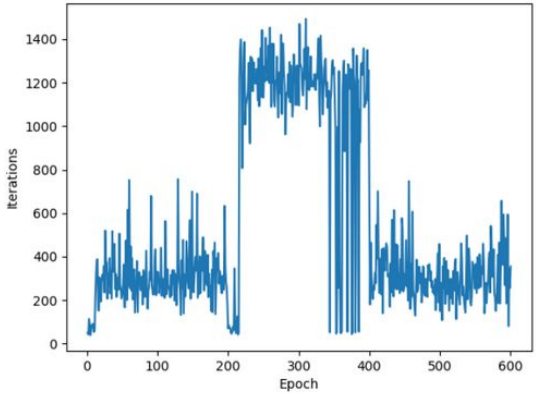
Most humans (most of you) intuitively look for theta - rather than omega

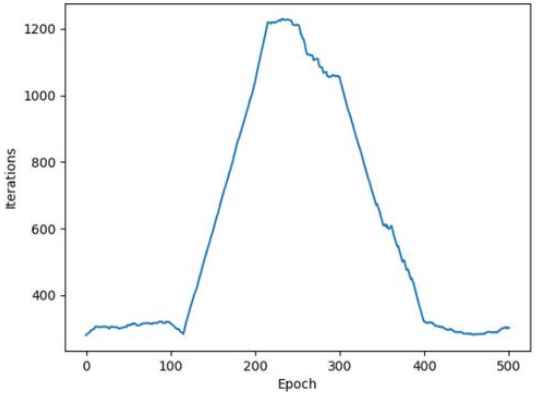
Sudden availability





Theta





Omega

Sudden availability

X and V have close to no impact on the performance.

After being exposed to theta and omega it takes a few episodes before it reaches the vanilla performance.

After disappearance it drops back to previous performance.

Why can’t ONA use acquired “policy” after exposed variables are hidden again?

Did you have a look at the code? How is information passed to ONA?

Sudden availability

```
ot = "ot"
vx = "vx"
self.observables =
    list(self.env.state_space.observables.keys())
if "omega" in self.observables:
   pos = self.observables.index("omega")
   if abs(state[0][pos] < 0.3):
       ot += '1'
   elif state[0][pos] < 0:
       ot += '2'
   elif state[0][pos] > 0:
       ot += '3'
```

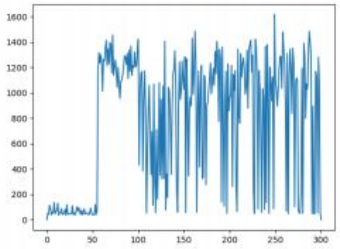
```
if "theta" in self.observables:
   pos = self.observables.index("theta")
   if 0 < state[0][pos] < 0.3:
       ot += '1'
   elif -0.3 < state[0][pos] < 0:
       ot += '2'
   elif state[0][pos] < 0:
       ot += '3'
   elif state[0][pos] > 0:
       ot += '4'
```

```
if "v" in self.observables:
   pos = self.observables.index("v")
   if abs(state[0][pos]) < 0.3:
       vx += '1'
   elif state[0][pos] < 0:
       vx += '2'
   elif state[0][pos] > 0:
       vx += '3'
```

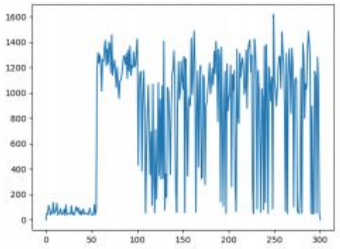
```
if "x" in self.observables:
   pos = self.observables.index("x")
   if abs(state[0][pos]) < 0.1:
       vx += '1'
   elif state[0][pos] < 0:
       vx += '2'
   elif state[0][pos] > 0:
       vx += '3'
return ot, vx
```

The value passed actually changes, ONA sees this as strings, not as values therefore no connection between the two different settings

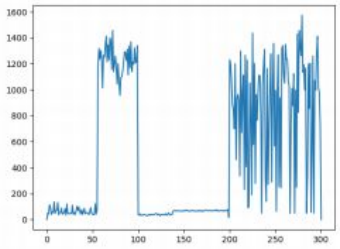
Sudden disappearance



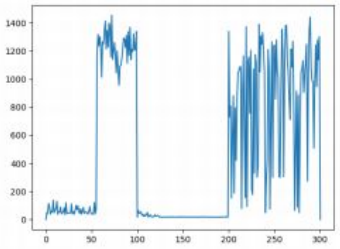
1. x
Hi: 1619
Avg: 775.666
Med: 1019



2. v
Hi: 1019
Avg: 775.666
Med: 1019



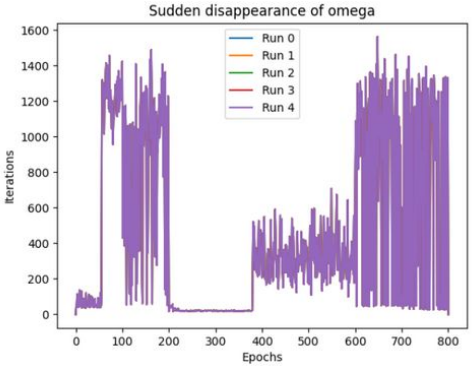
3. theta
Hi: 1575
Avg: 502.090
Med: 84



4. omega
Hi: 1456
Avg: 489.485
Med: 88

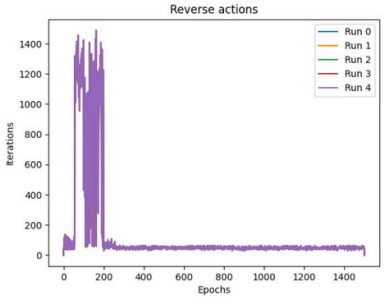
Same as sudden availability, just inverted.

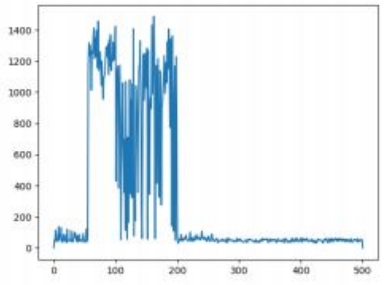
Some of you had different results for this - Was this due to misunderstandings? Or is ONA not deterministic in this case? Really wondering!

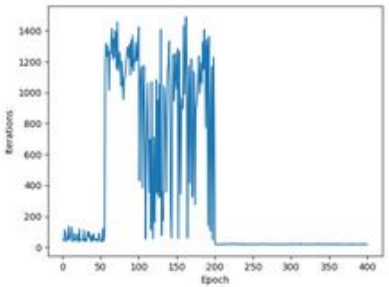


Custom tasks

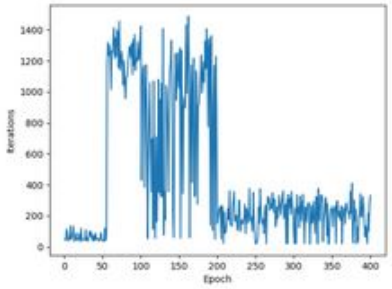
Inversion

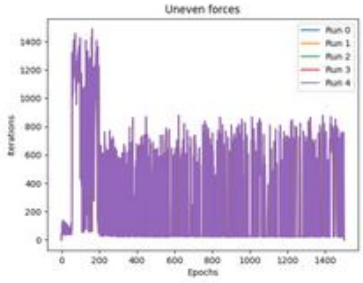






Uneven F

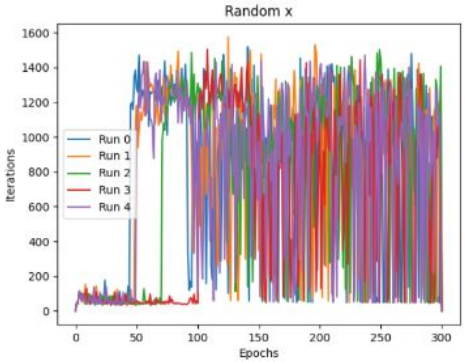


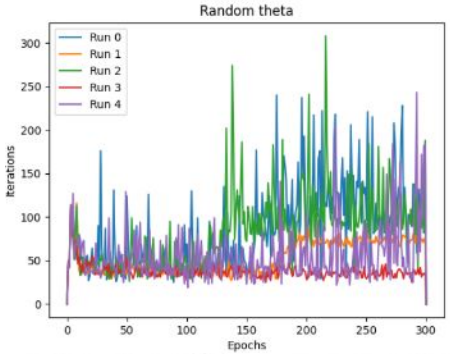


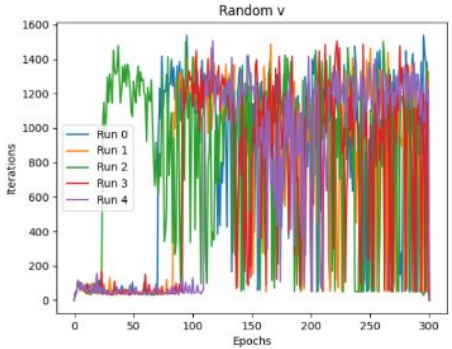
Also: Gravity increase, Pole mass increase etc. All of them with expectable outcomes.

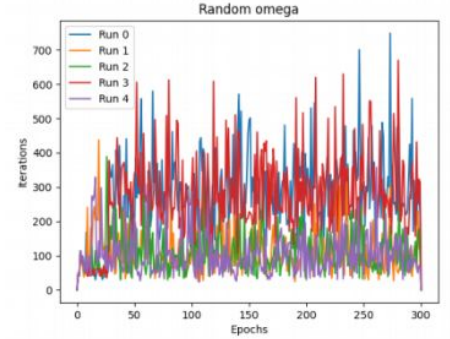
Own ideas

Noise vs hidden









ONA did not perform worse than with hidden variables for x, v, and theta, but performed worse with random omega, than with hidden omega.

And many more - like inference steps, change of “bad” reward, etc.

Conclusion

What can we conclude on in regards of learning in the AC, the human, and ONA?

Let’s discuss the following points:

Reliability, learning speed, cumulative learning, adaptability, generality, autonomy.