CS 520 : Project 4 - The Imitation Game

16:198:520

Due: Dec 15 by 11:55 PM, Submitted via Canvas

The purpose of this project is to train an agent (rather than implementing a specific decision agent) to imitate the agents we've build previously. For each of Project 1 and Project 2, take an agent of your choice, and build an ML agent to mimic it in the following way:

- Take the input data to be state descriptions of the current knowledge of the gridworld.
- Take the output data to be the corresponding actions the agent takes in the input states.
- The ML agent should consist of a neural network that maps from the input space to the output space (or, as necessary, modified, i.e. for softmax/probability regression).
- Train your model on data generated by the agent of your choice, over many episodes and many gridworlds.

After sufficient training, the ML agent can run by simulating the appropriate gridworld and using the trained network to make action selections given the current state. Repeatedly simulating the original agent and the ML agent on identical gridworlds will allow a comparison of performance between the two.

For this project, you're welcome to use Tensorflow or PyTorch or whatever machine learning framework you see fit, or roll your own if you are inclined. All code must be your own, and commented sufficiently well as to indicate you understand what the underlying framework is doing. Half this project is building an agent to mimic an agent from Project 1, half this project is building an agent to mimic an agent from Project 2.

For each project, I'd like you to experiment with two kinds of architectures:

- Architecture 1: Only Full Dense Layers
- Architecture 2: At Least One Convolutional Neural Layer

For each of these, you'll need to experiment (number of layers, nodes per layer, activation functions, window sizes, strides, filters, etc) in order to find the optimal structure.

For each project, for each architecture, you'll need to address the following:

1) How should the state space (current information) and action space (action selected) be represented for the model? How does it capture the relevant information, in a relevant way, for your model space? *One thing to consider here is local vs global information.*
2) How are you defining your loss function when training your model?
3) In training, how many episodes on how many different gridworlds were necessary to get good performance of your model on the training data?
4) How did you avoid overfitting? Since you want the ML agent to mimic the original agent, should you avoid overfitting?
5) How did you explore the architecture space, and test the different possibilities to find the best architecture?
6) Do you think increasing the size or complexity of your model would offer any improvements? Why or why not?
7) Does good performance on test data correlate with good performance in practice? Simulate the performance of your ML agent on new gridworlds to evaluate this.

8) For your best model structure, for each architecture, plot a) performance on test data as a function of training rounds, and b) average performance in practice on new gridworlds. How do your ML agents stack up against the original agents? Do either ML agents offer an advantage in terms of training time?

Note that for (7) and (8) above, you'll need to be able to handle that the ML agent may not ever successfully reach the target - how should you fairly include this data? Don't just discard it.

General Rubric

- Specification of input/output/model space, loss function, and architecture exploration. **(10 points)**
- Training. **(10 points)**
- Evaluation of models structures. **(20 points)**
- Evaluation of final agents. **(60 points)**

Bonuses:

1) **(20 points)** Research different kinds of layers and architectures, and build a third kind of model to mimic a Project 2 agent using these novel structures. Repeat the above analysis and be thorough. Do these structures provide any kind of performance gain? Why or why not.
2) **(30 points)** Build and train an agent to imitate one of your Project 3 agents (repeating the same analysis as above).
3) **(40 points)** Train an ML agent that outperforms your best agent from Project 1, Project 2, or Project 3. Be very clear as to your methodology, results, and evaluation.