LEARNING TEKKEN 7

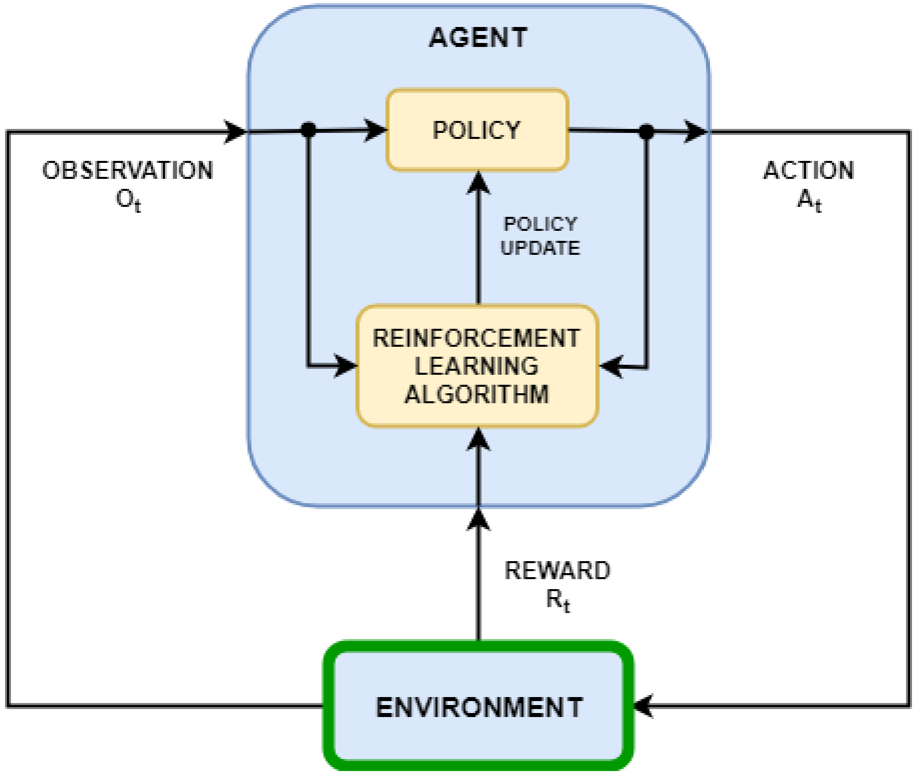
BY ZACHARY SAFIR

WHAT IS TEKKEN?

- A three-dimensional fighting game developed by Bandai Namco Studios
- Has a staggering 50-character roster
- Each character has over 100 unique moves
- Objective is to reduce your opponent's health points down to 0 in a 60 second round
- First player to win three rounds wins the match

WHAT IS REINFORCEMENT LEARNING?

- A type of machine learning that trains an agent to act intelligently in an interactive environment
- Using trial and error feedback shaped by numerical reward and punishment signals
- The goal is to find the correct actions that maximize that numerical value
- Continues to learn using past experiences
- Different from both supervised and unsupervised learning, a third category



Source: https://www.mathworks.com/help/reinforcement-learning/ug/create-matlab-environments-for-reinforcement-learning.html

REINFORCEMENT LEARNING PROCESS

KEY CONCEPT

- The curse of dimensionality, the more complex the environment the more difficult it becomes to model
- Using deep learning, much of this problem is alleviated
- Termed Deep Q-Network
- Became able to make very skilled models for Atari video games
- Next breakthrough, found that the policy network can also be made into a deep network
- A policy gradient actor-critic algorithm called Deep Deterministic Policy Gradients

SELF PLAY AND REWARD SHAPING

- Self play is a multi-agent learning approach that deploys an algorithm against copies of itself
- Does not require expert data to teach the model how to act
- Reward shaping is the process of adding and applying different kinds of reward signals to the learning agent
- In combination with self play, can train agents to handle a variety different play styles

IMPORTANT TERMINOLOGY

Frames: A measure of time in a fighting game, each frame is $1/60^{th}$ of a second, determines how a player can act a given time

Block Stun: Frame count after having an attack blocked

Hit Stun: Frame count after landing an attack

Whiffing: Attempting to attack and missing entirely

Back dashing: A more advanced form of backwards movement

Observation Space: The information pertaining to what an agent can observe at a given time

Action Space: The actions available to the agent

SIMPLIFYING THE ACTION SPACE

- Using prior knowledge, able to simplify the learning process down
- Rather than supplying potentially confusing numerical signals, can shape the action space
- For example, if the agents frame count is currently negative, remove the ability for it to continue attacking
- At the start of the round, prevent the agent from attacking blindly
- By considering and accounting for various known situations, can guide the learning process along smoothly

master TekkenBot / TekkenGameState.py / <> Jump to

WAZAAAAA0 CPU fix by skipping error loop ... Latest commit 0be4745

5 contributors

1423 lines (1102 sloc) | 58.5 KB

```
"""
This module's classes are responsible for reading and interpreting the memory of a Tekken7.exe proecess.

TekkenGameReader reads the memory of Tekken7.exe, extracts information about the state of the game, then saves a
'snapshot' of each frame.

Each GameSnapshot has 2 BotSnapshots, together encapsulating the information of both players and shared data for a single game frame.

TekkenGameState saves these snapshots and provides an api that abstracts away the difference
between questions that query one player (is player 1 currently attacking?), both players (what is the expected frame
advantage when player 2 emerges from block), or multiple game states over time (did player 1 just begin to block this
frame?, what was the last move player 2 did?).

"""
```

GETTING GAME VALUES

Source https://github.com/WAZAAAAA0/TekkenBot/blob/master/TekkenGameState.py

ENVIRONMENT TEMPLATE

```python
import gym
from gym import spaces

class CustomEnv(gym.Env):
  """Custom Environment that follows gym interface"""
  metadata = {'render.modes': ['human']}

  def __init__(self, arg1, arg2, ...):
    super(CustomEnv, self).__init__()
    # Define action and observation space
    # They must be gym.spaces objects
    # Example when using discrete actions:
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)
    # Example for using image as input:
    self.observation_space = spaces.Box(low=0, high=255,
                                        shape=(HEIGHT, WIDTH, N_CHANNELS), dtype=np.uint8)

  def step(self, action):
    ...
    return observation, reward, done, info
  def reset(self):
    ...
    return observation  # reward, done, info can't be included
  def render(self, mode='human'):
    ...
  def close (self):
    ...
```

MY ACTION SPACE

```
Movements = { 'simple': np.array(['Block','SSL','SSR','BackDash','Duck']),
              'full': np.array(['Block','SSL','SSR','BackDash','Duck','Forward' ])

            }
```

```
['1' '1,1' '2' '2,1' '3' '4' 'f+1' 'f+2' 'f+3' 'f+4' 'd/f+1' 'd/f+2'
 'd/f+3' 'd/f+4' 'd+1' 'd+2' 'd+3' 'd+4' 'd/b+1' 'd/b+2' 'd/b+3' 'd/b+4'
 'b+1' 'b+2' 'b+3' 'b+4']
```

```
['d/f', '1']
```

MY OBSERVATION SPACE

```
self.observation_space = spaces.Dict({
'Round Timer': spaces.Box(low=0, high=60, shape=(1,), dtype=np.int32),

    'Health': spaces.Tuple( (
        spaces.Box(low=0, high=175, shape=(1,), dtype=np.int32),
        spaces.Box(low=0, high=175, shape=(1,), dtype=np.int32),

    )),

    'Frames':spaces.Tuple( (
        spaces.Box(low=-20, high=20, shape=(1,), dtype=np.int32),
        spaces.Box(low=-20, high=20, shape=(1,), dtype=np.int32),

    )),
    'Opponet': spaces.Discrete(50),
    'State': spaces.Tuple((
        spaces.Discrete(4),
        spaces.Discrete(4)
    )),
    'whiff': spaces.Tuple((
        spaces.Discrete(2),
        spaces.Discrete(2)
    ))

})
```

CURRENT STATE OF MY PROJECT

https://www.youtube.com/watch?v=rUj4CODZXWI

WHAT COMES NEXT

- Finding and fixing the improperly functioning memory address values
- Once everything runs properly, modeling
- Finding the best model for playing the game
- Devising different learning strategies

REINFORCEMENT LEARNING ALGORITHMS READY TO USE

CI passing build passing docs passing Codacy Badge Codacy Badge

Stable Baselines

Stable Baselines is a set of improved implementations of reinforcement learning algorithms based on OpenAI Baselines.

You can read a detailed presentation of Stable Baselines in the Medium article.

These algorithms will make it easier for the research community and industry to replicate, refine, and identify new ideas, and will create good baselines to build projects on top of. We expect these tools will be used as a base around which new ideas can be added, and as a tool for comparing a new approach against existing ones. We also hope that the simplicity of these tools will allow beginners to experiment with a more advanced toolset, without being buried in implementation details.

Note: despite its simplicity of use, Stable Baselines (SB) assumes you have some knowledge about Reinforcement Learning (RL). You should not utilize this library without some practice. To that extent, we provide good resources in the documentation to get started with RL.

Source: https://github.com/hill-a/stable-baselines

THANK YOU

REFERENCES

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- MIT Press.Ng, A. Y., Harada, D., and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In ICML Vol. 99: 278-287
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J.,Bellemare, M. G., et al. 2015. Human-level control through deep einforcement learning. Nature, 518(7540), 529.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. 2016. Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484
- Oh, I., Rho, S., Moon, S., Son, S., Lee, H., & Chung, J. (2020, January 31). *Creating pro-level AI for a real-time fighting game using Deep Reinforcement Learning*. arXiv.org.
- Yoshida, S., Ishihara, M., Miyazaki, T., Nakagawa, Y., Harada, T.,and Thawonmas, R. 2016. Application of Monte-Carlo tree search in a fighting game AI. In IEEE 5th Global Conference on Consumer Electronics:1-2