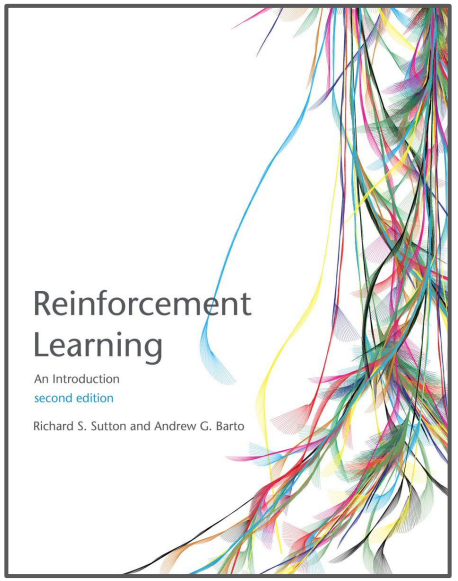
Reinforcement learning and games

Ian Marsh



Installation

Open Gym basics I

- There are two basic concepts in reinforcement learning:
  a. The **environment**
     - The outside world
  b. The **agent**
     - The algorithm we are writing

- The agent sends **actions** to the environment, and the environment replies with **observations** and **rewards** (that is, a score).

Hands on

⚙️ ianmarsh@LightMan ~/Downloads python3.9

```
>>> import gym
>>> import matplotlib
>>> env = gym.make('Acrobot-v1')
```

Install I

```
Pip install gym
```

Or clone and local install

```
git clone https://github.com/openai/gym.git
cd gym
  pip install -e .
```

Install II

Other packages:

MuJoCo -> bullet

https://gerardmaggiolino.medium.com/creating-openai-gym-environments-with-pybullet-part-1-13895a622b24

OpenGL

Atari games

Open Gym principles

1. Environments, not agents.
    a. Two core concepts are the agent and the environment
2. Emphasize sample complexity, not just final performance
3. Encourage peer review, not competition
4. Strict versioning for environments

Atari I

```
import gym
env = gym.make('SpaceInvaders-v4')
env.reset()

for _ in range(10000):
    env.render()
    env.step(env.action_space.sample())
env.close()
```

Atari II

```
import gym
env = gym.make('SpaceInvaders-v4')  /* v0 avail also */
env.reset()                         /* reset env. */

for _ in range(10000):
    env.render()
    env.step(env.action_space.sample())
env.close()
```

AI gym verses DeepMind

Tech	OpenAI	DeepMind	Notes
Goal	Deep Reinforcement Learning *environments*	Deep Reinforcement Learning *algorithms*	
Solution method	Model-free RL Tabular Q learning & Trial and error	MCTS = Monte Carlo Tree Search + Heuristics	
Access	Standardised AI		
use of a separate target network — the Q_hat part of the above equation — to stabilize training, so the TD error isn't being calculated from a constantly changing target from the training network, but rather from a stable target generated by a mostly fixed network.	Synchronous variant A2C, popularized a very successful deep learning-based approach to *actor-critic methods*.	Asynchronous Advantage Actor Critic (A3C)	

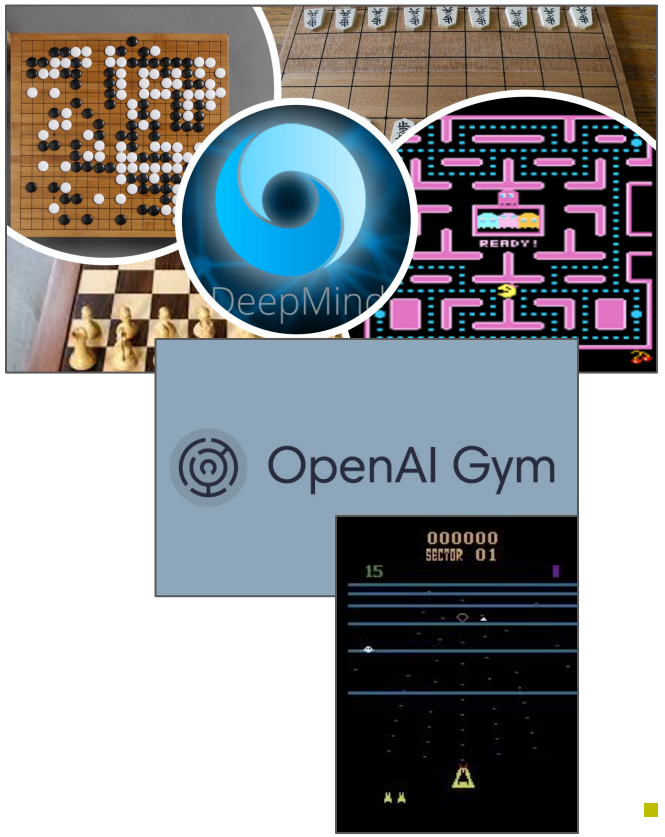
Open Gym basics II

The core gym interface is Env which is the *unified environment interface*. There is no interface for agents; that part is for us. The following are the Env methods to know:

1. reset(self): Reset the environment's state. Returns observation
2. step(self, action): Step the environment by one timestep.
    - Returns observation, reward, done, info.
3. render(self, mode='human'): Render one frame of the environment. The default mode will do something human friendly, such as pop up a window.

Implementation

1. Hand-coded
    - Will implement game
    - But not RL package
2. Use a game package
    - MuZero (Deep Mind)
    - OpenAI's Gym
    - Arcade learning ALE
3. Learning alpha, gamma, and epsilon need to be provided in a control object in package
4. Maybe visualisation large square relax mode



Theory

Model-based and non-model based RL (wikipedia)

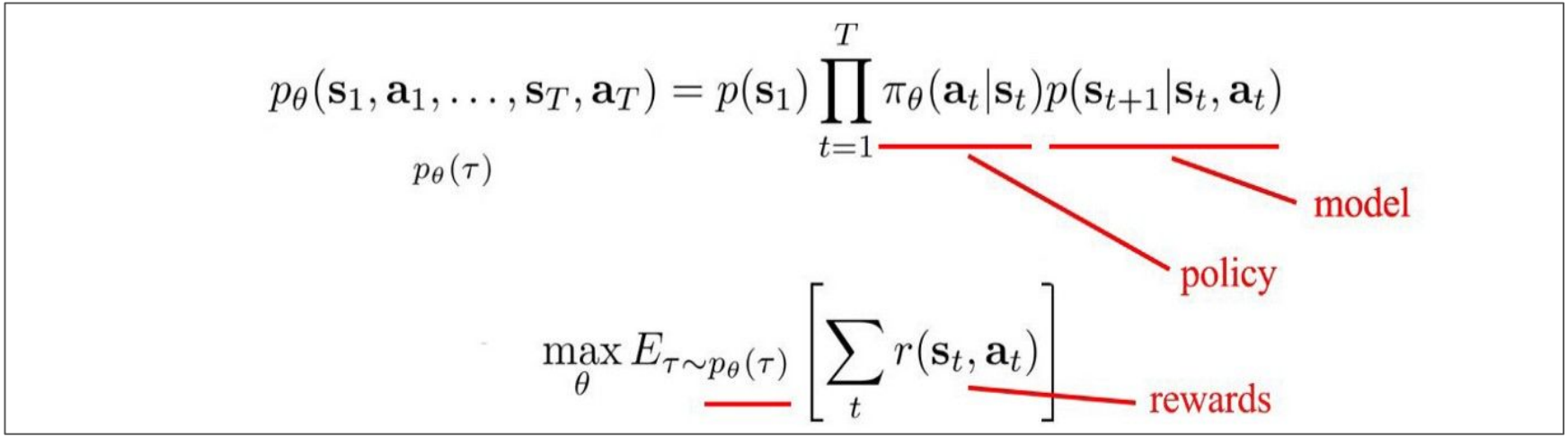
1. In RL a model-free algorithm *does not use* the *transition probability distribution* (and its reward *function*) associated with the Markov decision process which, in RL, represents the problem to be solved.

2. The transition probability distribution (or transition model) and the reward function are often collectively called the *model* of the environment (or MDP), hence the name "*model-free*".

3. A model-free RL algorithm can be thought of as an "explicit" trial-and-error algorithm. An example of a model-free algorithm is Q-learning.

Policies and system dynamics (model)

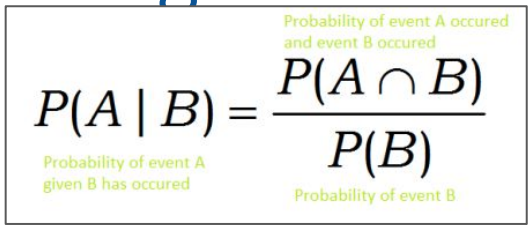
Reinforcement learning RL maximizes rewards for our actions. From the equations below, rewards depend on the policy and the system dynamics (**model**).

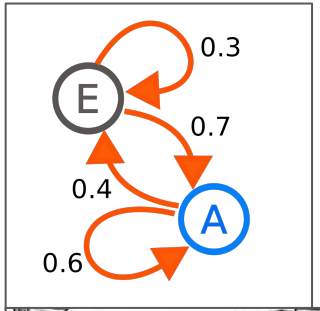


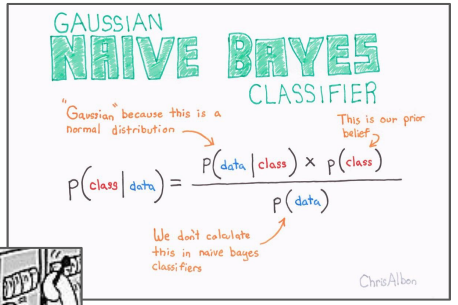
https://jonathan-hui.medium.com/rl-model-based-reinforcement-learning-3c2b6f0aa323

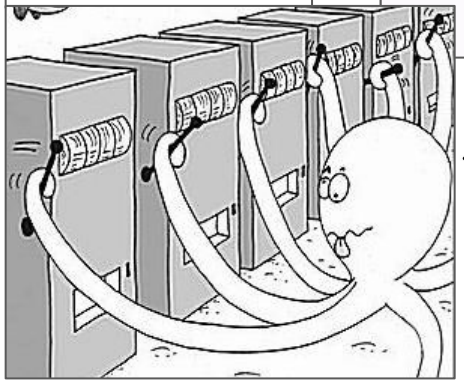
Pre Reinforcement learning

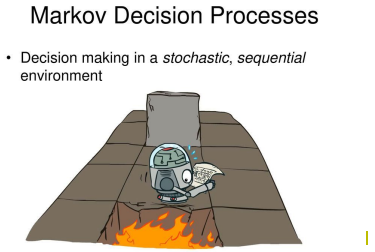
1. A rich history from stochastic processes
    a. Conditional probability
    b. Bayesian
    c. Markov processes
    d. Markov Decision Processes
    e. Multi-arm bandits

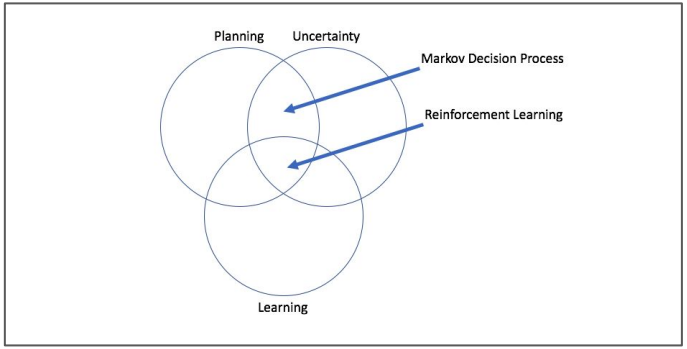






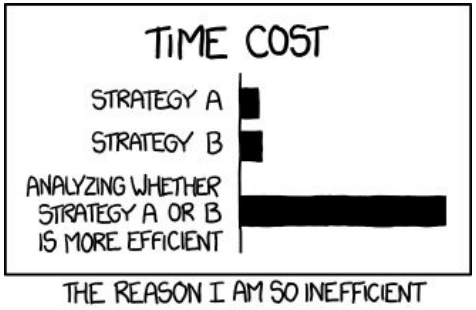


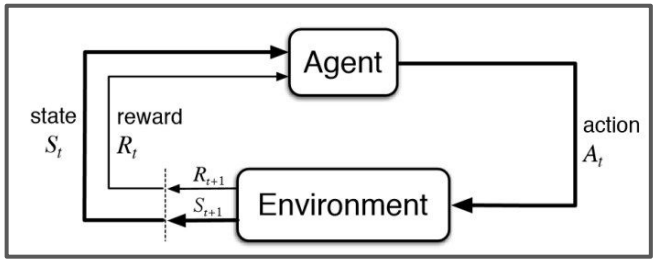




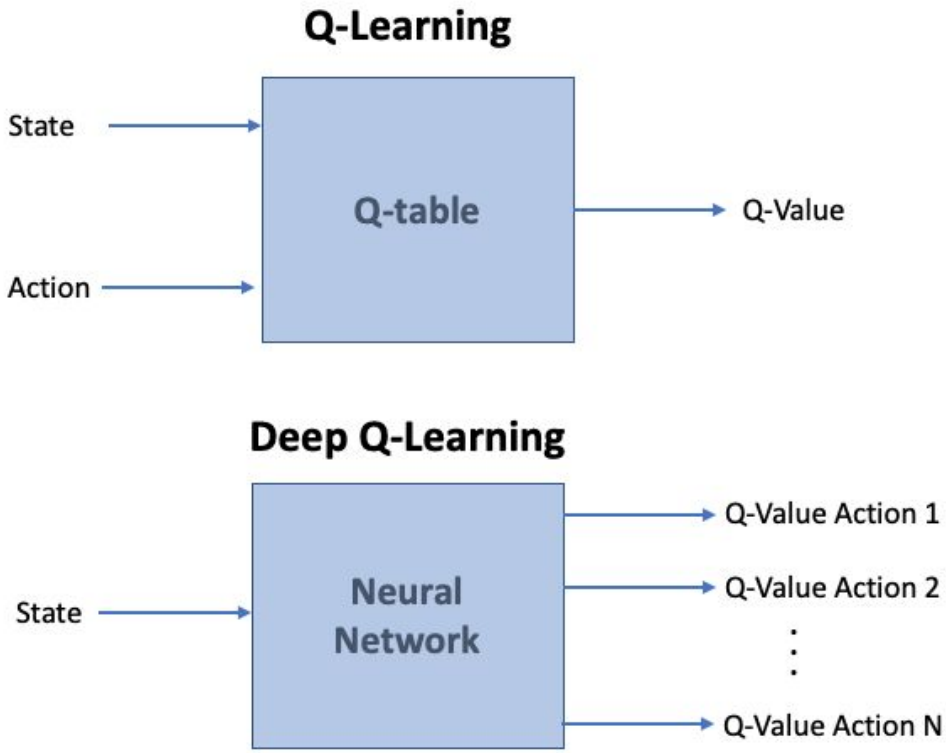
Reinforcement learning

1. Agent is us, algorithm or ‘learning machine’
2. Policy learning
   - Stores local knowlege
   - Q($State_i$, $Action_i$)
   - Learn the function policy (called Q)
3. Feedback really comes via a reward (R_i)
   - Training at each step
     - (S_i, A_i, R_{i+1}, S_{i+1}) tuple
4. Environment
   - Randomly place or learn from humans



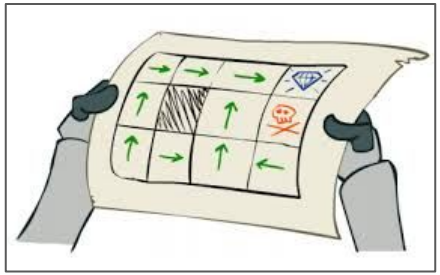


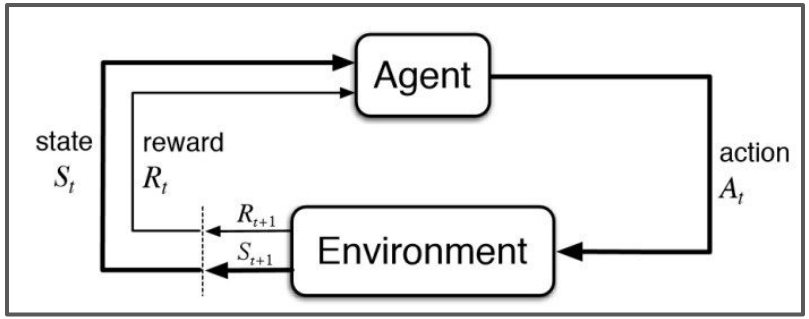
Q-learning and Deep Q-learning



Learning from predefined observations

Useful when input data is pre-determined or we want to train an agent that replicates past behaviour. Here one just inputs a tabular data structure with past observations into the RL package. Could be from an external source, and doesn't need further interaction with the environment.

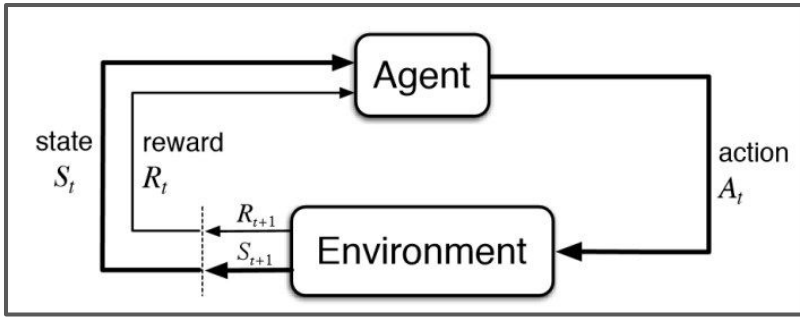




Learning from an interactive environment

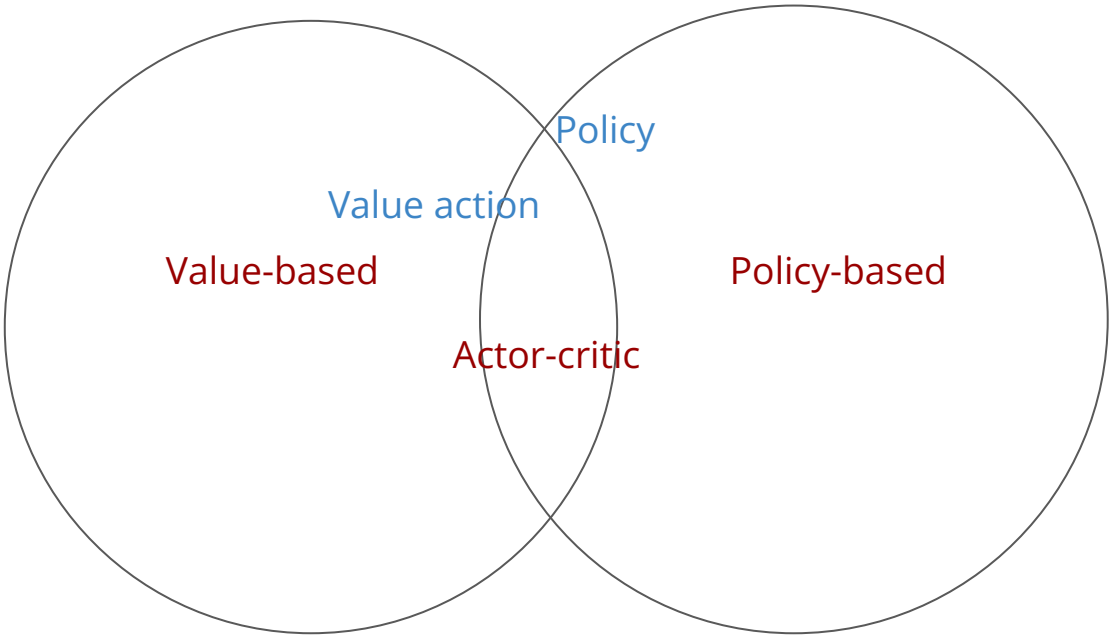
Mimics the behaviour of the environment. Agent samples experience from this function. Takes state-action pair as input, returns a list of the name of next state and reward. Collect random sequences from it, e.g. 1000.

```
environment <- function(state, action)
{
  ...
  return(list("NextState" = newState,
              "Reward" = reward))
}
```



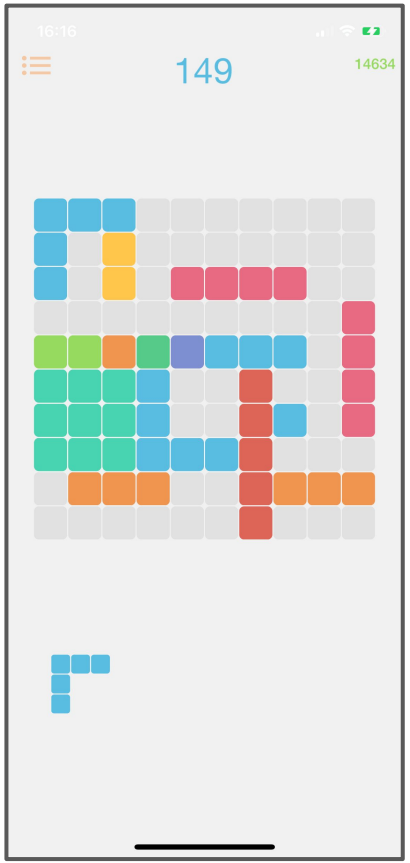
Explore Versus Exploitation

Learning policies high level

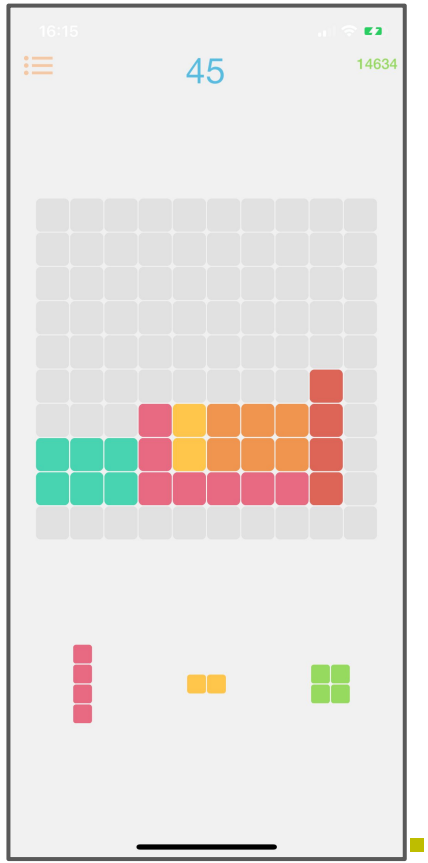


1010 Crazy!

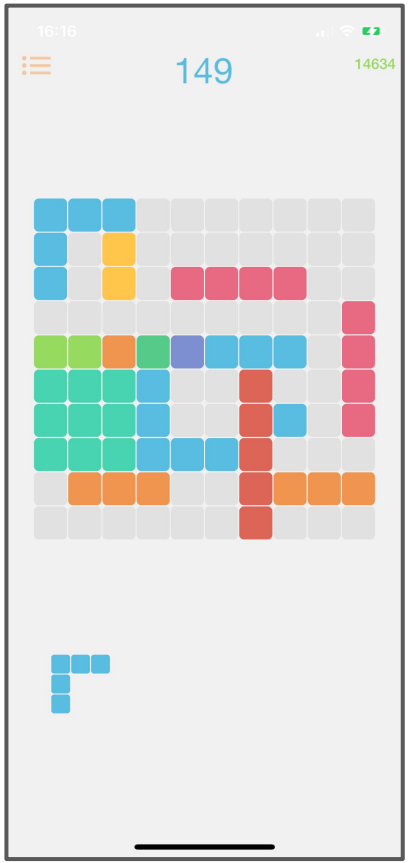
1010 Crazy!



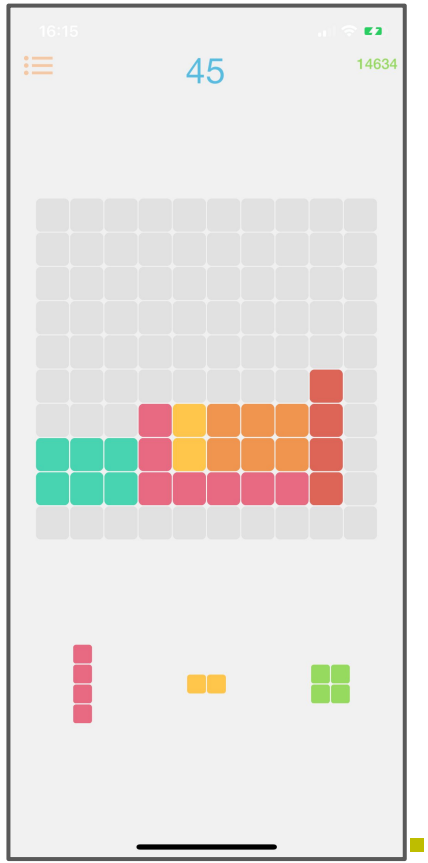
1. Addictive, beware 🤦
2. 10 x 10 board
3. Complete lines, horizontal or vertical
4. Points for #squares in each piece placed + bonus of 10 for completed lines
5. 14 random generated shape-rotation pieces
6. Human scores
    - 14634 mine, good 194727
    - Task is code RL to beat me 👻



Intuition



1. Cost function that maximises score
    - Or number of pieces placed
2. Strategy that completes lines (obv)
    - Or sizeof contiguious regions
3. Rewards at each step based on points
    - Or blocks (ready to be ‘finished’)
    - Cf. Left and right pics



Supplements

Videos

1. Policy gradients
    - Hado (deep mind) https://www.youtube.com/watch?v=bRfUxQs6xIM
    - Books
        - Reinforcement Learning : An introduction (Sutton)
        - Oreilly - Reinforcement Learning Industrial Applications of Intelligent Agents (Winder)
        - Grokking - Deep Reinforcement learning (Morales)
        - Pakt - Deep Reinforcement Learning with Python (Ravichandiran)

Papers

OpenAI Gym

https://arxiv.org/pdf/1606.01540.pdf

Reinforcement Learning, Bit by bit

https://arxiv.org/abs/2103.04047

Blogs

- https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2