TRANSCRIPT - B.SC. PRESENTATION

SHENGDI CHEN

Date: 2022-08-09.

1. Synopsis

(1) [NEW SLIDE] Welcome: Presentation: B.Sc. Thesis
  (a) Three papers: on screen
    (i) Details of authorship in appendix of report
  (b) Setup of this talk
    (i) 3 copies of Thesis: refer for details
    (ii) on slides: truncation; simplification on screen
    (iii) refer report for details

Definition 1. Structure of report and talk
(1) Formulation of the game
  (a) MDP, `gym`
(2) 3 types of agents
  (a) training
  (b) testing framework; results

Solution 1. Throw-backs
(1) Three classic implementations
  (a) NES, Gameboy
  (b) 1989-1998
(2) diff. interfaces, but Commonality for abstraction
  (a) large rectangular field
  (b) falling pieces of different shapes
(3) First step
  (a) math. representation of field and piece

2. Setup

2.1. Representation.

2.1.1. *Field.*

Solution 2. Begin with the field

(1) LEFT:
  (a) segment of the field
(2) RIGHT: equivalent representation
  (a) bunch of 0 and 1; glorified `2D` matrix of `bool`

Solution 3. `2D` matrix: natural extension: the coordinates

(1) LEFT: indexes of the field
  (a) convention of `numpy` and other libraries
    (i) first index: downwards
    (ii) second index: rightwards
  (b) RIGHT: standard shape

2.1.2. *Piece.*

Solution 4. Now the pieces

(1) UPPER: all the 7 pieces
  (a) each with 4 blocks
  (b) LOWER: internal representation
    (i) variable: `pid`; range of 0 to 6

Solution 5. Pieces can be rotated

(1) UPPER: rotation-pattern of `"T"` and `"I"`, the bar
(2) NOTE: 4 rotation values
  (a) clockwise and CCW sense
  (b) same direction 4 times, resets
  (c) math. convention: CCW is positive rotation

Solution 6. Apply the coordinates to the pieces

(1) PREVIOUSLY: every piece in 4×4 box
  (a) this box is the `[R-COORD]`-matrix
  (b) every entry: relative to RED $\star$: upper-left
(2) WHY: $\star$ in upper-left?
  (a) indexing convention: all positive

Solution 7. Some color-coded examples

(1) Order: conforming to index
  (a) go down; then left

Solution 8. Ready for absolute

(1) Add `pos` ($\star$) to `R-COORD`
  (a) member-wise (math); trivial (with `numpy`)
(2) low transport content
  (a) `R-COORD`: constant for every piece!
  (b) only need `pos` ($\star$)
  (c) Reconstruct `ABS-COORD`

Solution 9. Full piece specified with 3 info

2.2. MDP.

Solution 10. Ready for Formal Formulation of dynamic system: MDP

1. Quick walk-through
2. Process: one time-step $\star \implies$ to next $\star + 1$
   - (a) state y; apply a
   - (b) evolve with f; finally rewarded

Solution 11. Formulate Tetris as MDP

1. $y :=$ Field $+$ Piece
   - (a) big `bool`-matrix; `pid` and coordinates
2. Action: applied every new piece
   - (a) PREVIOUSLY: `rot`: max. 4
   - (b) GREEN `pos1`: horizontal: field-width
   - (c) GREY: No vertical: just fall

Solution 12. Formulate Tetris as MDP

1. f: as we know it
   - (a) generate; fall; clear-lines
   - (b) GREY: intermediate piece
2. Finally: r: two-part structure
   - (a) game-over: -2 random, just negative
   - (b) #clears: RIGHT: big-formula
     - (i) $\propto$ to k-func [LEFT]
     - (ii) $\uparrow$#clears $\iff \uparrow$ score per-line
     - (iii) Tetris-Guideline

2.3. `gym`-adaptation.

Solution 13. `gym`: Agent-framework; 2016

1. [LEFT]: [RED] extra observation x
2. Agent:
   - (a) LOWER: has access
   - (b) UPPER: hidden internals

2.3.1. *Field-based Obs.*

Solution 14. Driven by Q.: how to tell

1. Good VS Bad: 3 things
   - (a) Abs-height
   - (b) relative-height
   - (c) DEMO: hole
2. Math. ranges; in `python`
   - (a) elevation: 9 values: every 2 neighbors $\implies$ 1 value
   - (b) Total: 29 values to track
   - (c) every column
3. Too many? Sum over all
   - (a) [RED] upper-script C: Compact-mode: 3

Solution 15. Two more observations

(1) #clears; old-friend
(2) Some other research: `pid`

Solution 16. How to pick the correct combination?

(1) Que: full VS compact field?
(2) 4 modes
(3) NOTE: Size of OBS-space
  (a) [GREEN] least $\iff$ smallest; [RED] biggest
  (b) BACK-LATER

3. Agents

3.1. `sb3`.

Solution 17. Agent specified by `gym`; NOW: train

(1) Established library: `sb3`
(2) Space-struct of Tetris $\Longrightarrow$ 3 algs
(3) Except: DQN
  (a) Action-Sp. is `1D`-discrete
  (b) re-inflate

3.2. Tuned-DQN.

Solution 18. DQN: Also: Self-implemented

(1) First: Q-Learning
  (a) [GREEN] Return $:=$ sum of discounted reward
  (b) [BLUE] Q-Value $:=$ $\mathbb{E}$ of Return
  (c) Policy $:=$ max'ze Q-Value
(2) 1992; breakthrough: 2013
  (a) Exact Q-value: non-convex; complicated
  (b) Neural-Net
(3) NOTE: size of 1^{st}-Layer
  (a) Min. VS Max. Obs-modes

Solution 19. Also: Action-Space

(1) PREVIOUSLY: general Act-space
  (a) accommodate: for every-piece $\iff$ worst-case
  (b) SUBTITLE: invariant
(2) Tune!
  (a) Taylor for piece, e.g., ROT
  (b) HALF; Quarter
(3) Result
  (a) piece-dependent; SMALLER

4. Results

4.1. Setup.

Definition 2. Before the results: Setup the Env

1. Reproduce `Python`: `pipenv`: package management
2. constraint: Every Alg. + OBS
3. gauge the perf.:
   - (a) invite: mental exercise

4.2. Results.

Solution 20. `sb3`: General, Raw Act-sp

1. 12 Agents: 3 Algs + 4 modes
2. Generally: ~50; BEST in red
   - (a) recall our threshold
3. [SUBTITLE] when BEST?
   - (a) HORIZ: PPO performs best (ADV-func)
   - (b) VERT: MAX obs

Solution 21. All lost? Look @ Tuned-DQN

1. Tailored Action-Sp
   - (a) Much better! VS `sb3`
   - (b) BEST: 6 times the threshold
2. The big Q: Which Mode of OBS?
   - (a) DQN: tuned Action-space: Observes LEAST $\iff$ great results
   - (b) `sb3`: general, raw ACT: Observes MOST $\iff$ much worse
3. Graphs logged by `TensorBoard`: BEST-variant

Solution 22. Finally, Real-time testing

1. go back to `pid`: underlying generator
   - (a) every consecutive $7 \implies$ a »bag«
   - (b) within a bag $\implies$ permutation
2. [UNIQUE] Disregarding the Guidelines:
   - (a) Bag-content & Randomness
   - (b) `{"O", "I", "T"}` Symmetric-pieces: EASY
   - (c) full-bag: [RANDOM] STANDARD
     - (i) Non-RANDOM: (easy-reproducible + Non-trivial)
3. Result: Coloring: stock-market
   - (a) Non-Random: failing at `"O"` (not covered in training)
   - (b) Random: Full-7 (real test) Double the threshold
   - (c) TITLE: successful (MOST TIME)

5. Closer

Definition 3. Closing section

1. 12 weeks; `14GB` data and 300 hours on `Euler`
2. Trained: 16 agents
   1. (a) 4 obs-modes
   2. (b) 4 algorithms

Definition 4. Insights gained

1. LEFT: SELF-DQN: generally great performance
   - (a) tuned Act-space + min. Obs-space
     - (i) during training: $6\times$ benchmark
2. RIGHT: RAW `sb3`
   - (a) raw Act-space + max. Obs
   - (b) insufficient perf
3. Summary:
   - (a) With tuned act-space: (we observe less + achieve more)
   - (b) Without tuning: (OBS more + achieve less)

Definition 5. Into the Future

1. Modify internals of Engine
   - (a) current in control theory
   - (b) previewing algorithms, dynamic regret-structure
2. Testing process
   - (a) standard test-sequence
   - (b) machine-against machine
3. The algorithms themselvs
   - (a) implement PPO, A2C
   - (b) Ray/Rlib; Tianshou
4. Self-promotion:
   - (a) if opening available
   - (b) love to do semester-thesis

Solution 23. Wrap up

1. Bachelor-Thesis; ISE
   - (a) supervisor; advisors
2. Math. modeling; MPD
   - (a) concept of obs
3. General Act-Sp.
   - (a) easy; invariant
   - (b) lackluster perf.
4. Tuned Act-Sp.
   - (a) needs less information
   - (b) great perf.
   - (c) opening for future

Definition 6. Final words

1. Thanks
   - (a) Prof. Buhmann:

            (i) Accepting the initial sketch; Locate in ISE
      (b) Advisors
            (i) Organize; Admin
            (ii) Talks; Inputs
(2) I myself
      (a) deepen in ML (RL in particular)
      (b) coding; reward/observation engineering
      (c) TETRIS itself

6. DEMO

Definition 7. DEMO-generals

(1) Free-play

```
// 1. free-play: pycharm.game
$ tk.py
[PRESS] e

// 2. changing bag: pycharm.engine.py -> _get_generator()

shetris -> True True 5
shetris01 -> True False 4
02 -> False True 31
03 -> False False 30
```

(2) Procedure:
      (a) free-play (prove engine works)
            (i) Non-Random, `"I"`; `{"O", "I", "T"}`
      (b) DQN (go to the slide 63)
            (i) Non-Random, `"I"` (no animation, but is real; works; loops! $\Rightarrow \infty$ n_P-value!)
            (ii) Non-Random, `{"O", "I", "T"}` (confirm: value on the slide)
            (iii) Random, `{"O", "I", "T"}` (change to random; hope do not hit the green)
            (iv) Random, 7-bag (the big test)

Definition 8. Backups

(1) DQN

(a) Run default

```
// no GUI: test.py
$ test.py


// GUI: entry.py -> Entry().def __init__
=> self._actor = DQN

$ entry.py
PRESS [r]
HOLD [s]
```

(b) variation

```
// util.py -> Loader() -> __init__:
=> script_path = ... + "/result/shetris<CHANGE>"

// reporter.py -> Reporter() -> __init__
=> self.obs_factory = ObsStandard(
      use_compact_field=<CHANGE>,
      use_pid=<CHANGE>

// network.py
   def __init__(self):
      super().__init__()
      self.conv1 = ...nn.Linear(<CHANGE>, 64),...
```

(2) Ref

```
// 1. no GUI
$ genetic.py

// 2. with GUI: entry.py->Entry.__init__
self._actor = self.make_actor_genetic()
```

(3) Sb3: algorithm (PPO or A2C or DQN)

```
// info.py -> ShetrisInfo.get_algpol
=> pick the pair

// info.py -> ShetrisInfo.get_rel_dirs()
pick the pair

// shenv.py
def __init__():
   flatten_action = TRUE (if DQN); False (if PPO or A2C)
```