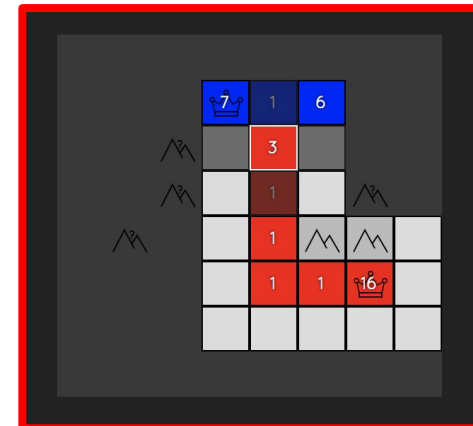


Patton: Generals.io bot with deep reinforcement learning

Yulun Li, yulun2@stanford.edu

Overview

- Problem: develop a bot to play game Generals.io



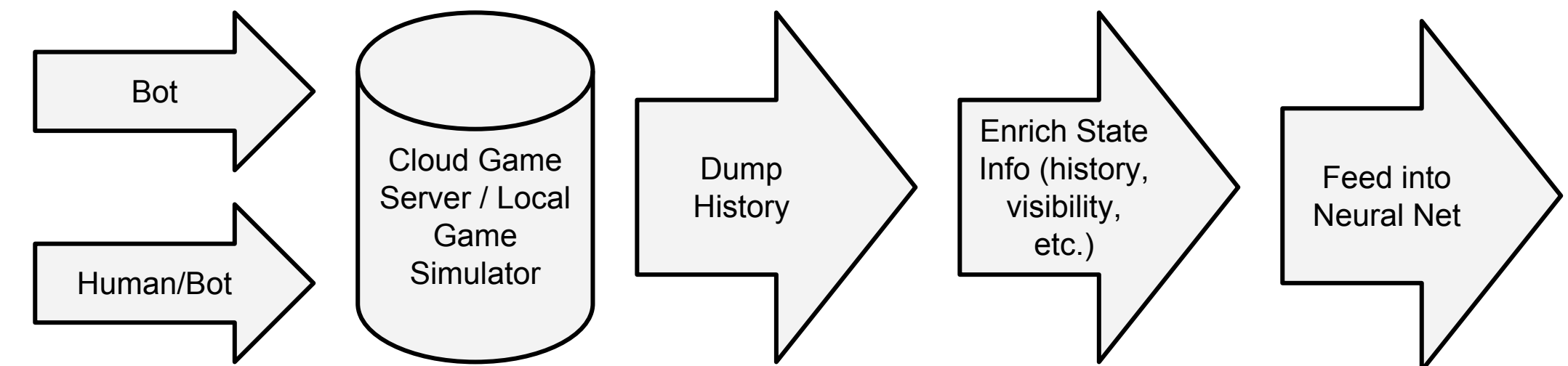
- Explore partially observable MDP
- Discover new strategies if possible

v0.1

- Only one frame is passed into the convolutional layer
- Assumes each state is Markovian, and the bot is stateless
- The hand-crafted adversary is not very efficient, which waits until its armies to be significantly larger to attack the general

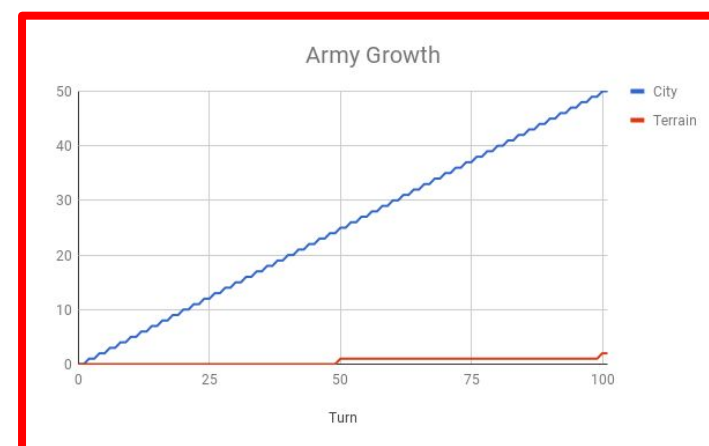
Simulator

- Online simulator connects the bot with the online game server at bot.generals.io to play against human players
- Offline simulator take 2 bots as parameters and let them play against each other offline

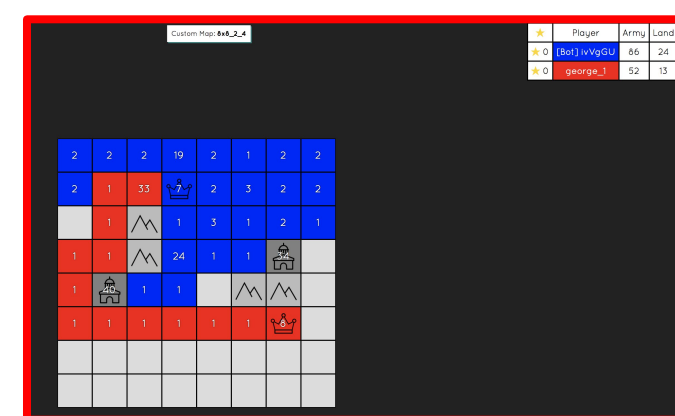


Challenges

- Difficult to train from self play
- Game state changes are highly temporal



- Enemy strength can only be inferred because even if the enemy has a large army, its location could be unfavorable
- Number of armies and terrains don't translate to strengths, as demonstrated in the following example, where the bot (blue) has significantly more armies



v0.2

- Pass the current and previous 3 frames to the convolution layer
- Keeps the state for the bot, e.g. an explored terrain will be remembered for future actions
- Added examples where human players played against the bot, this adds diversity to the dataset
- The neural network is also a lot harder to train
- Improved the hand-crafted adversary to focus on attacking the bot's general once it's discovered
- Randomized frame sequences from different games to train the model

Results

Objectives	Result
Values exploring terrains	✓
Moves army to the enemy's general after its location is discovered	✓
>50% win rate against human player (myself)	✓
Wins games without advantage in total armies	✓
Train completely by self-play	✗

Enemy	Result
Random bot	99%
Random expansion bot	89%
Random expansion bot that knows my general position	71%
Human (I drew the map and knew the bot's general location)	23%

Future

- Evaluate if a different reward scheme can improve performance
- Train/test bot with random maps
- Train/test bot on larger maps, which demands more powerful GPUs and longer training times

References

- Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search."
- Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning."
- Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps."