An AI for Dominion using Deep Reinforcement Learning

Eric Yang (iameric@stanford.edu), Yu-Chi Kuo (yuchikuo@stanford.edu)

Introduction (Problem statement)

Rules for the game Dominion

- Dominion is a deck building card game for 2-4 players. Player with the most victory points at the end wins
- Victory point cards useless during the game
- Sequential planning for deck building
- more info on http://wiki.dominionstrategy.com/

Challenges

- Stochastic nature of this game. This card game involves shuffling cards, so outcomes of a given action is not deterministic.
- Dynamic action set. Different cards are affordable at each step.

(credit: Dominion online)

Goal

- Build a AI that decides what cards to buy at the buy phase of the game
- Find the algorithm and hyperparameters for deep reinforcement learning that can achieve good performance in a 2 player game with a fix set of cards
  - Kingdom cards include: village, cellar, smithy, festival, market, laboratory, chapel, warehouse, council room, militia, moat, witch
- Achieve good win rate against strong heuristic-based AI

Approach: Deep Reinforcement Learning

- Reinforcement Learning is the framework for learning sequential decision making.
- Deep learning provides neural network as universal function approximator.
- Deep RL uses neural network to learn functions to improve decision making.
- Tried 3 model-free RL algorithms
  - **Monte Carlo Learning** aggregates rewards over episodes

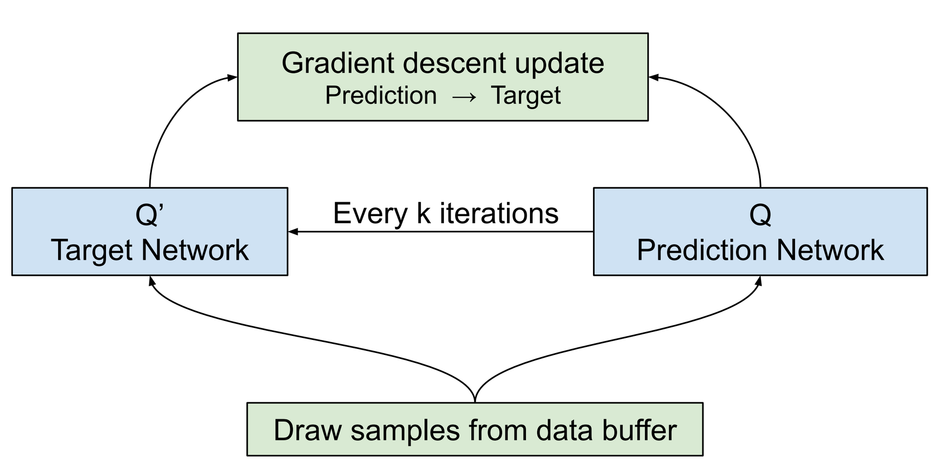
$$G(s_t,a_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-t} r_T$$
$$Q(s,a) \leftarrow Q(s,a) + \alpha(G(s,a) - Q(s,a))$$

  - **SARSA** bootstrapping with additional target network

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t))$$

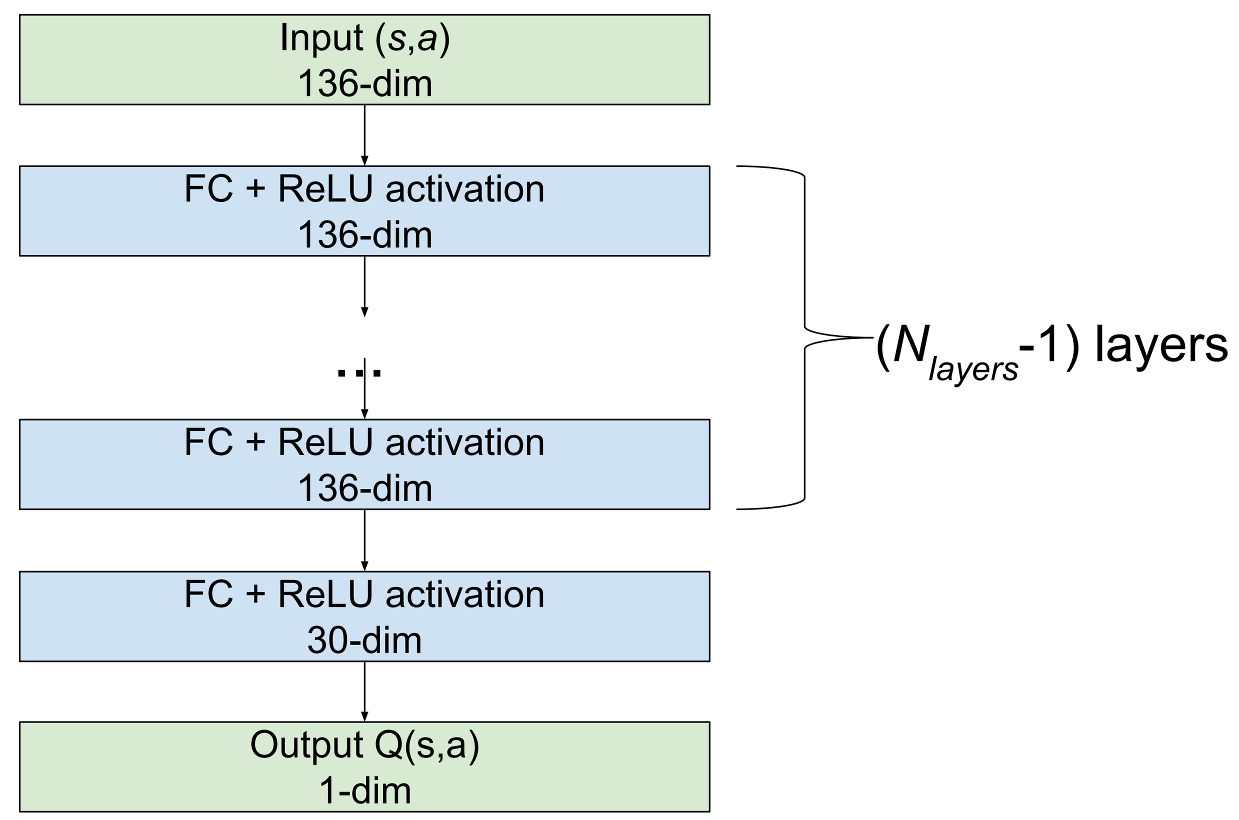
  - **Deep Q-Learning** off-policy+bootstrapping

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a' \in a'_{opt}} Q(s',a') - Q(s,a))$$



Q-function Neural Network Structure

- **Q(*s*,*a*)** is the expected reward at state ***s*** if action ***a*** is taken.
  - ***s*** (game state) has is a vector of length 117
  - ***a*** (card bought) is a vector of length 19
- Use a neural network (Q-network) to learn Q-function.



Hyperparameters and RL reward

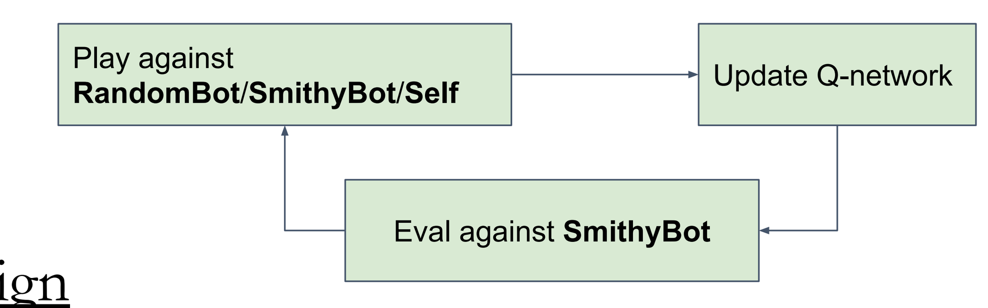
- Epsilon-greedy exploration for RL agents

$$\pi(a|s) = \begin{cases} (1-\epsilon) + \epsilon/|A|, & \text{if } a = \arg\max_{a \in A} Q(s,a) \\ \epsilon/|A|, & \text{otherwise} \end{cases}$$

- A decaying epsilon balance between exploration and exploitation

$$\epsilon = \begin{cases} \epsilon_0, & \text{if } \epsilon\text{-decay} = \text{False} \\ 10 \times \epsilon_0 / N_{iter}, & \text{if } \epsilon\text{-decay} = \text{True} \end{cases}$$

- generate data against heuristics AI/RL agent itself
- Evaluate performance against heuristics AI



Reward Design

- Victory points bought as reward at each step
- A terminal win reward Rw: true reward but might be too sparse
- A terminal points per turn reward Rp: motivates winning fast when the agent is stronger than opponent

$$r(a_t,s_t) = \begin{cases} \Delta P_t, & \text{if } s_t \neq s_T \\ \Delta P_t + R_w + R_p \times P_T/N_{\text{turn}}, & \text{if } s_t = s_T \text{ \&win} \\ \Delta P_t - R_w + R_p \times P_T/N_{\text{turn}}, & \text{if } s_t = s_T \text{ \&lose} \end{cases}$$

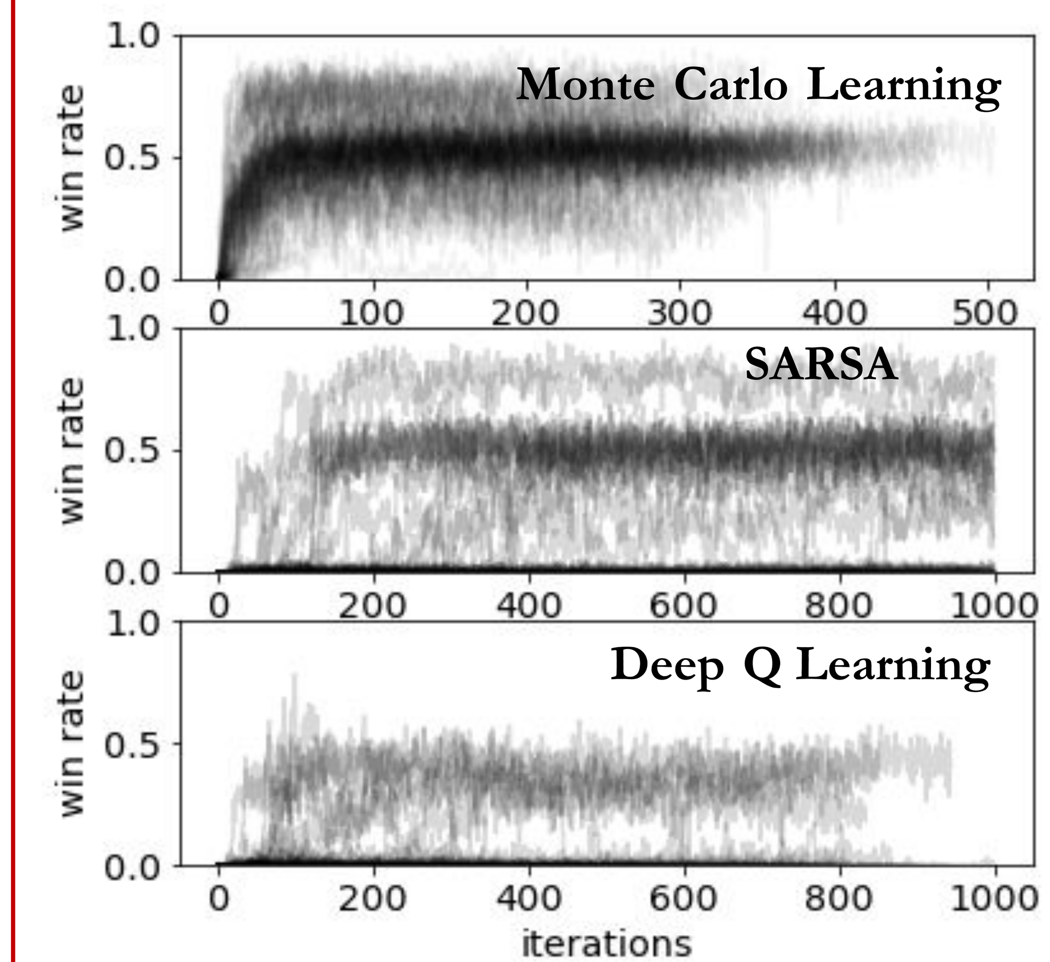
Summary

hyperparam	category	min	max	distribution
N_{layers}	network structure (Fig.2)	2	5	uniform (discrete)
Dropout	regularization (Fig.2)	0.0	0.5	uniform
ϵ_0	exploration (Eq.7)	0.1	0.01	log uniform
ϵ-decay	exploration (Eq.7)	False	True	Bernoulli $p = 0.5$
γ	reward calculation (Eq.2)	0.9	0.99	(1- γ) log uniform
R_w	reward design (Eq.10)	0.01	100	log uniform
R_p	reward design (Eq.10)	0.01	10	log uniform

Results and Discussions

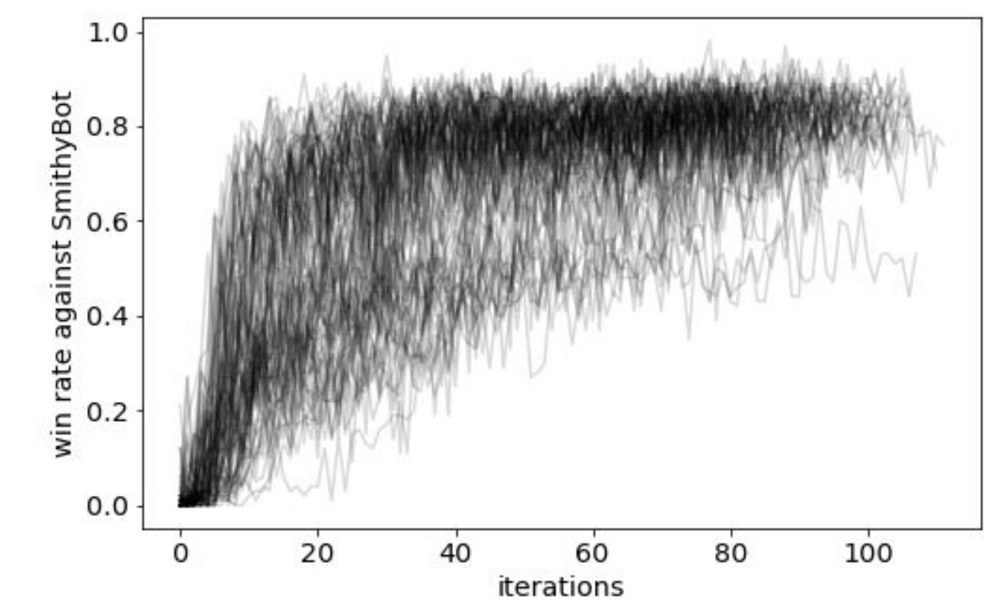
Comparison of RL algorithms

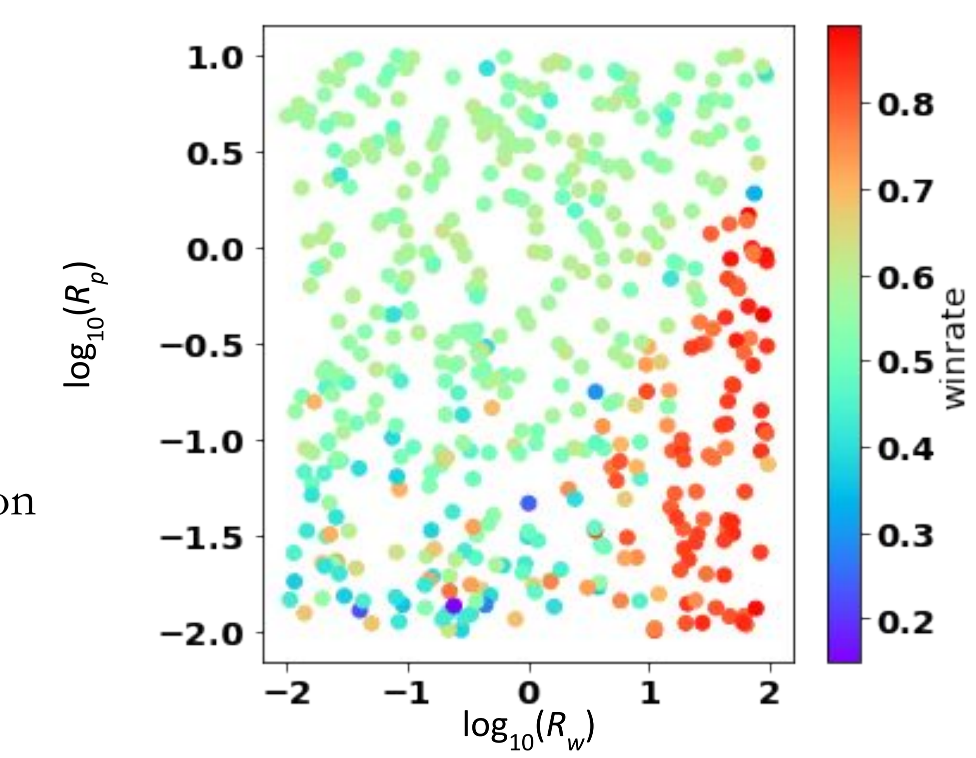
- Evaluated win rate against strong heuristic-based AI: **SmithyBot**
- Monte Carlo Learning performs the best



Robustness of Monte Carlo Learning

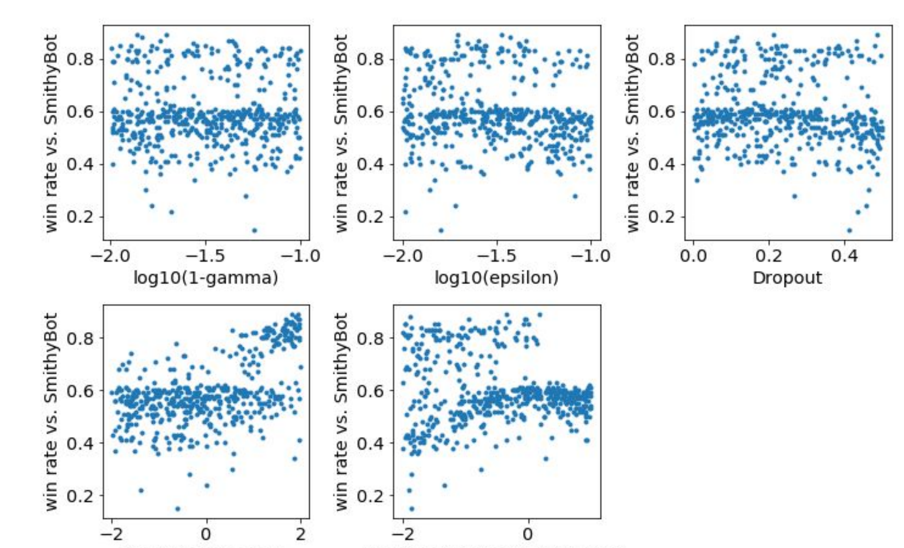
- 100 agents with the same hyperparameters
- Most agents converge to high win rate within 100 iterations

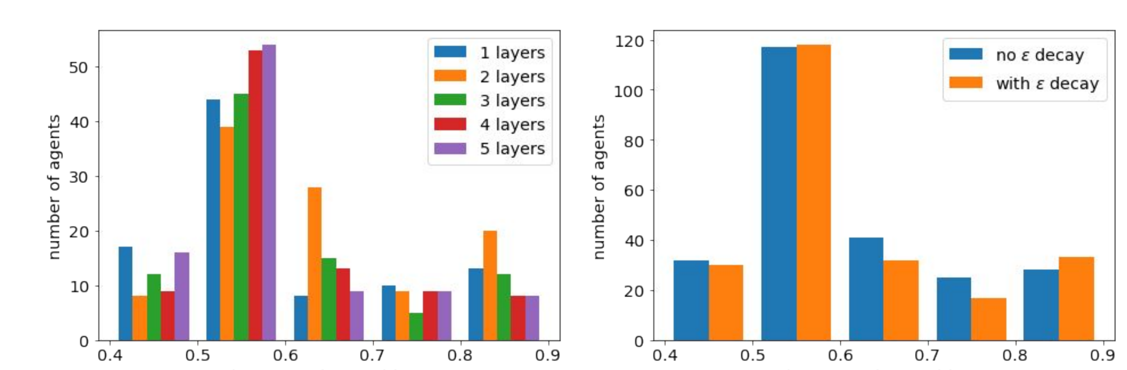




Hyperparameters

- Among the 7 hyperparameters explored, only the ones regarding reward (Rw and Rp) has correlation with the performance
- Rw Guides RL agent to aim for winning.





Future Work

- Generalize to more cards. The complete dominion game consists of 12 expansions and >300 different kingdom cards whereas this project is only limited to 12 basic cards.
- Generalize the AI beyond 2 player games into multiple player game and see how the optimal strategy changes.
- Use RL approach in action phase of the game.
- Build an AI that can play well with different combination of kingdom cards including combinations that it has not seen before.

References

- Sutton, Richard S.., and Andrew G.. Barto. Reinforcement Learning: an Introduction. The MIT Press., 2018.
- 2019,https://github.com/rspeer/dominiate-python,[Online; accessed 4-November-2019]
- Fynbo, R. B. 2010, Developing an agent for Dominion using modernAI-approaches,http://gameprogrammer.dk/data/Developing_an_Agent_for_Dominion_using_modern_AI-approaches.pdf, [Online;accessed 10-October-2019]
- Tollisen, R., Jansen, J. V., Goodwin, M., & Glimsdal, S. 2015, in CurrentApproaches in Applied Artificial Intelligence, ed. M. Ali, Y. S. Kwon,C.-H. Lee, J. Kim, & Y. Kim (Cham: Springer International Publishing),43
- Wiki contributors. 2019, Dominion Strategy Wiki,http://wiki.dominionstrategy.com/index.php/Main_Page,[Online; accessed 5-December-2019]
- Wikipedia contributors. 2019, Dominion (card game) — Wikipedia, TheFree Encyclopedia,https://en.wikipedia.org/w/index.php?title=Dominion_(card_game)&oldid=918940837,[Online; accessed 10-October-2019]