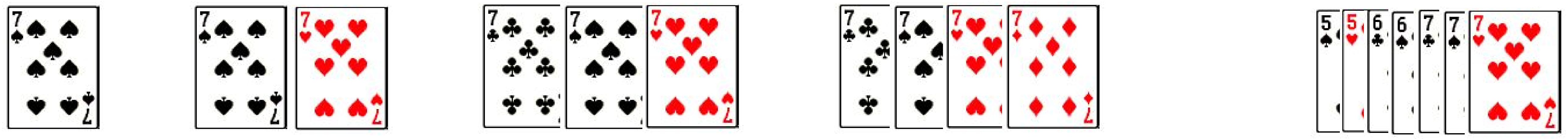
“Competing Upstream” Against YOU!

Leon Lin, Wei-ting Hsu, Hua Feng

Game Rules

ZhengShangYou, or “Competition Upstream” is a Chinese card game that is part strategy, part luck. We implemented an agent to play a slightly simplified version of this game with two players: 1 human, 1 computer, through reinforcement learning. Each player is dealt a random hand of cards and tries to get rid of all the cards in one’s hand. The player starting the has several options for cards to play:



Single *Double* *Triple* *Bomb:* 4 of a kind *Chain:* consecutive cards with at least two of each link

The next player must match the pattern but with higher cards, play a “bomb”, or pass. Once every player has passed, the last player to play some cards wins the round, and starts the next round with any options above. The value of the cards increases from 3 to K, then A, 2, Black Joker, Red Joker

Approach 1

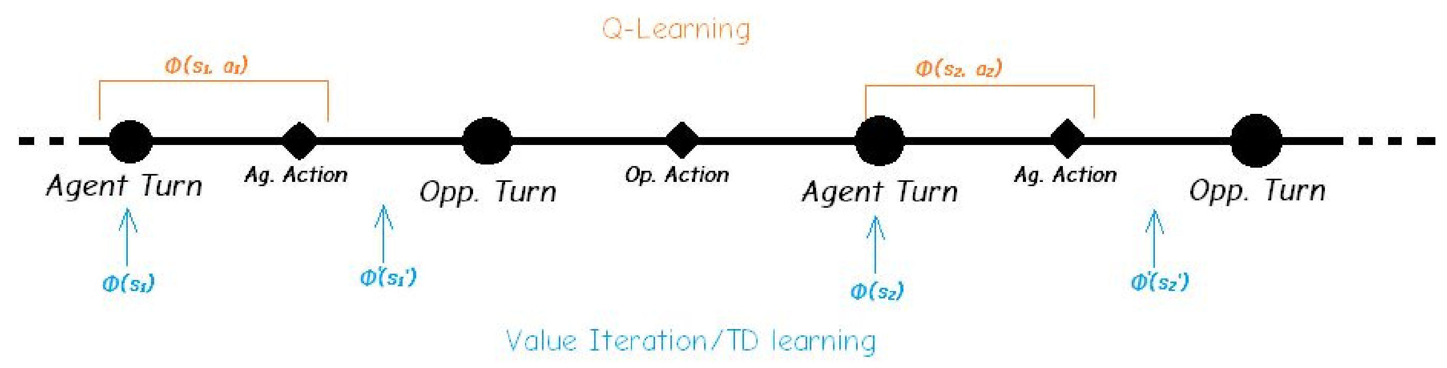
Q-Learning with Value Approximation

The game’s state space is very large. With 18 random cards dealt to each player, no exact set of starting hands will ever be seen more than once. At each turn, there can be dozens of possible moves. Thus, we had to use value approximation with a feature extractor.

- Features: The state is the beginning of each player's turn, we extract features such as the number of doubles, triples, and pattern cards, ratio between number of cards, whether it is before or after half game, etc. Each feature is coupled with the action taken.
- Learning: simulate 5000 games against the random agent with 20% probability of exploration. The weights are updated after both players take a move and the new state and reward are known.
- Adjusting Bias: We reduced the step size for high cards features to counter the higher frequency that high cards are played

The Q-Learning agent won some games against the baseline (‘Greedy Agent’) with a reduced deck size, but could not win with the full deck. It could not beat a human on any deck size.

Models



Approach 2

Value iteration (TD Learning)

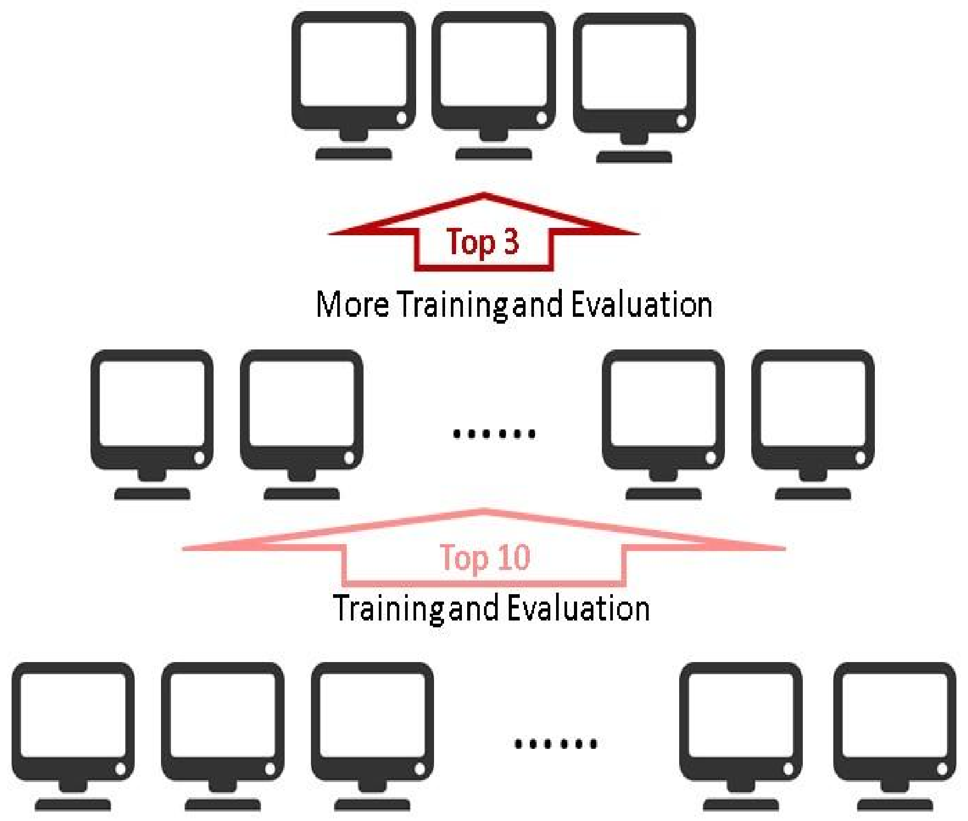
We have many hypotheses as to why the first approach failed, which we tried to address with this approach: a value iteration algorithm that is a modified version of TD learning.

- Features: We extracted similar features as before, but only over the states. However, since the opponent's hand is unknown, the transition probabilities are unknown, so it is not possible evaluate successor states. We instead created a hybrid solution. Each time it's agent's turn, we consider 2 states, s and s': s is before an action is taken and s' is after an action is taken but before the opponent takes an action. We extract the same features from both, but the features are convoluted over whether it’s from s or from s'. The policy is the action who’s s' has the highest value.
- Learning: Simulate x games against itself with 20% exploration. The history of features are stored until a game terminates, then the weights are updated with the histories before a new game.

This feature extractor, although it works over twice the number of states, has a much smaller feature space than that of Q-learning: it’s dimension is *numFeatures* * 2 instead of *numFeatures* * *numActions*. TD simulates a full game before updating the weights instead of updating the weights after each turn; we made this change because the rewards only come at the end of the game. The last big difference is that we changed the step size from 1/sqrt(iteration) to 50/sqrt(iteration + 100) so that the step size decreased more gradually.

This algorithm produced better results, as seen on the right.

TDNS

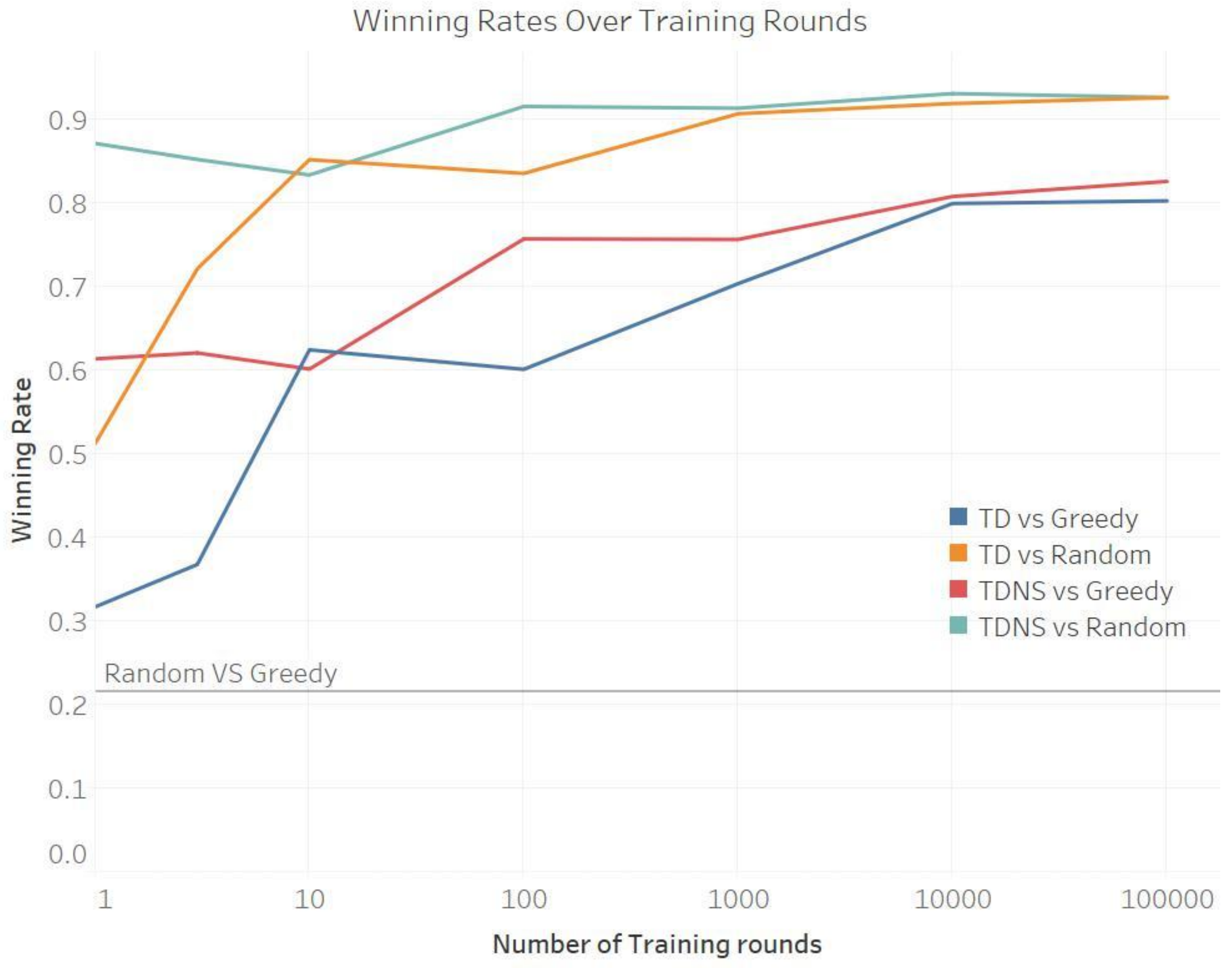


We noticed that different instances of the TD agent would win different percentages of games, even with the same number of training instances. As this may be due to local optima, we attempt a method of multiple instantiations we called ‘TDNS’ (for TD natural selection), which consists of 3 rounds:

The first round initializes 50 TD agents, trains each for a number of games, then tests each against the baseline agent. The best 10 move to the next round, where they are trained and pruned again, leading to 3 algorithms ranked by effectiveness after the final round.

Result and Analysis

The TD agents yield significantly better results than Q-Learning, so we graphed the TD results. They beat our baseline (‘Greedy’) ~82% of the time, and humans ~40% of the time.



player\score	TDNS vs Human Player
Player1	8:12
Player2	8:12
Player3	6:14