RL in Spades: RL Agents for the Spades Card Game

Adam Keppler, Durga Ganesh
{akeppler, gdurga} @ stanford.edu

Introduction and Rules

- Spades is a classic trick-taking card game, where players bid the expected number of rounds (tricks) that they will win
- Each trick is composed of each player playing a card. The highest card played wins the trick
- If a player fails to reach their bid they are penalized heavily
- If a player reaches their bid they get 10 * the bid in points
- Strategic Bidding and Gameplay are integral to Spades

State-Space Representation

- **Hand:** List of each agent's set of available cards
- **Deck:** List of all previously played cards
- **Bid:** Agent's current bid (predicted trick wins)
- **Won:** Number of hands won by the agent
- **Current:** List of cards played this hand
- **Spades Broken:** Whether a Spade has been played this game
- Player agnostic representation allows the agent to learn from all players
- Hidden information and randomness lead to a very large state-space
- There are 68 unique actions in Spades drawn from (Get Hand, Bid, or Play)

Baseline and Oracle

- **Baseline:** Greedy Rules Based Agent
  - ► *Bids:* Number of Face Cards and Spades in its Hand
  - ► *Plays:* Highest Card in its Hand
  - ► *Performance:* 67% win rate in 12 games against human players
    Max Total Score - 82, Max Winning Bid - 8, Max Tricks Taken – 10
- **Oracle:** Human Performance
  - ► *Evaluation:* 4 humans played 2 games against every other player
  - ► *Performance:* Peak Win Rate - 67%, Top Score - 82,
    Max Winning Bid - 8, Max Tricks Taken - 10

Training Methodology

- We utilize a training replay buffer to ensure that old examples remain relevant
- For Q-Learning we generate a training set using the 400 most recent examples and 1,000 randomly sampled examples from the past 10,00 examples
- To train our Deep-RL model, we use the most recent 750 examples and sample 1,000 from the last 20,000 examples
- Additionally, for the Deep-RL model, we perform 10 passes over this training set to allow the model to reach a reasonable level of performance on the data

Q-Learning Model

- To prototype our environment, we developed a Q-Learning agent
- Initial results using purely the State + Action Encoding were lackluster
- Hand-Crafted featurization allowed us to greatly improve the models performance
- We introduced features that helped the model to:
  - ► Asses the strength of its hand when bidding
  - ► Better appraise the outcome of playing a specific card.
- We designed these features using human domain knowledge and strategic insights.
- Through the augmented features, the Q-Learning agent is able to achieve a 58% Win Rate against our Baseline Agent
- Given that our Baseline is comparable to a novice human, our Q-Learning agent was able to achieve basic supra-human performance.

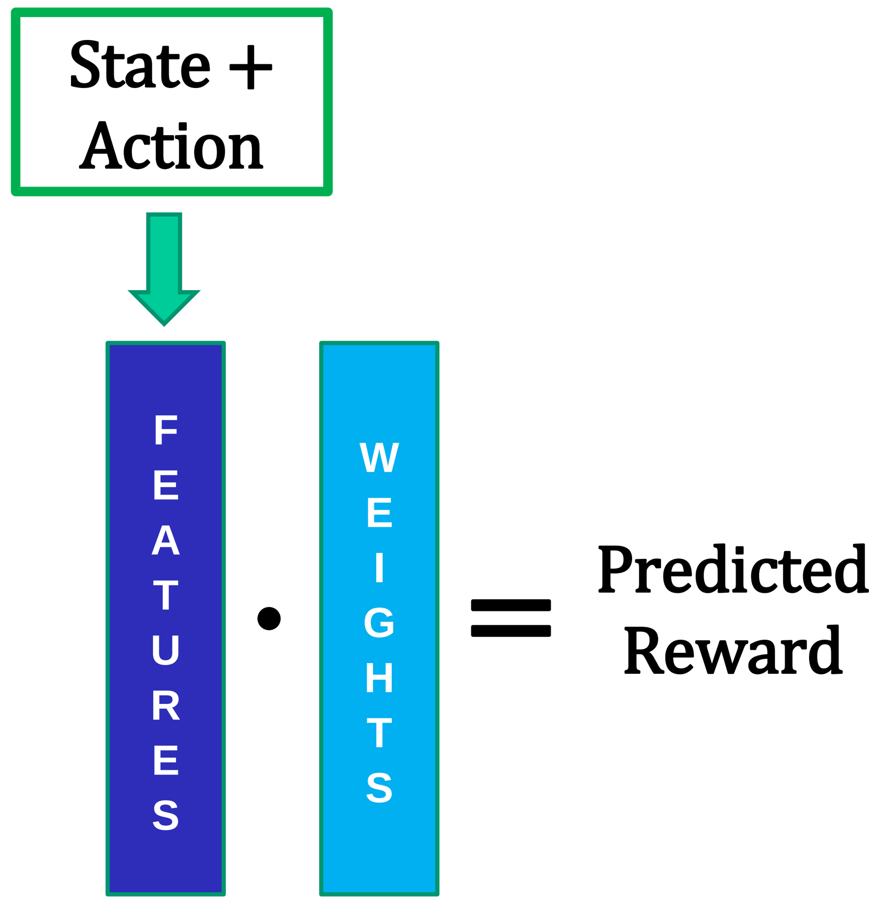

Figure 1: Q-Learning Reward Prediction via Function Approximation
Both Vectors have a dimension of (1 x (# of features))

Deep-RL Architecture

- To allow the model to learn more complex relationships, we explored a Deep-RL approach
- We enhanced our Q-Learning featurization, by using feature embeddings instead of One-Hot Encodings
- The increased number of parameters and non-linear elements provide the model with greater learning potential and power
- The added complexity also results in the need for significantly more tuning
- Due to the relatively few times Bid Actions are made, standard L1Loss fails to prioritize mastering them
- We mitigate this issue by up-sampling the number of Bid Actions and rescaling the reward/penalty for Bid Actions
- Another challenge for Deep-RL is the nature of online learning, while a model may have mastered prior games, new games present unseen challenges
- Random Exploration also proved harmful to the Deep-RL model
- Overall this model is able to defeat the Baseline Agent 58% of the time

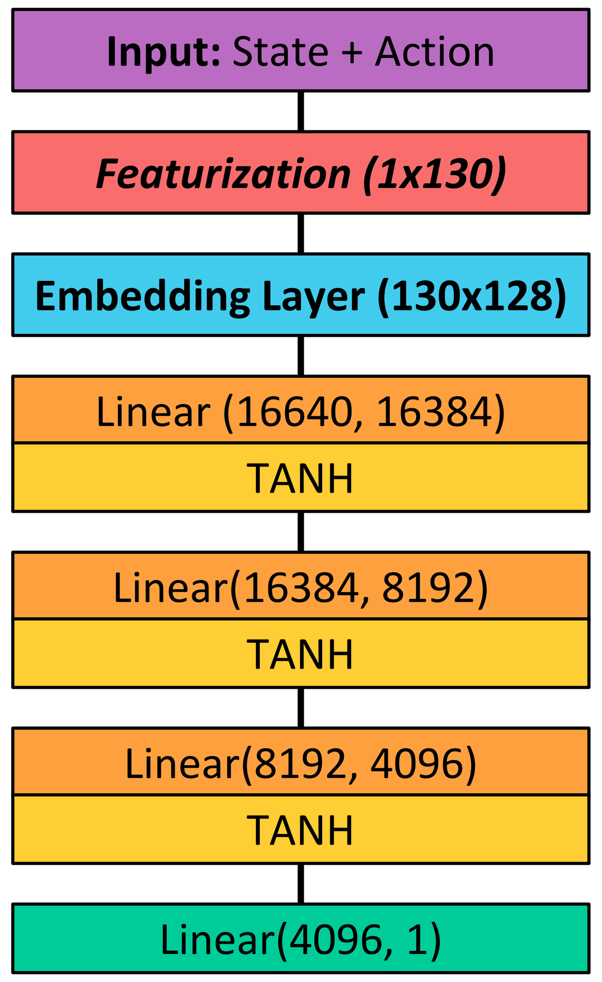

Figure 2: Deep RL Architecture
This architecture produces learns to predict the reward for each state action combination.

Results

- Both AI agents learned to:
  - ► Defeat the Baseline agent consistently at a rate better than random luck (over 50% of the time)
  - ► Assess the strength of a hand when bidding and avoid high risk bids

Frequency of Bid (%)	Bid 0	Bid 1	Bid 2	Bid 3	Bid 4	Bid 5	Bid 6	Bid 7	Bid 8	Bid 9	Bid 10	Bid 11	Bid 12	Bid 13
Q-Learning	*0.69*	*0.71*	*0.78*	*1.35*	*7.23*	20.9	48.7	15	0.9	0.67	0.69	0.75	0.8	0.78
Deep-RL	*0*	*3.23*	*0*	*3.23*	*0*	0	70.97	12.9	6.45	3.23	0	0	0	0

Challenges

- Designing informative features is very challenging
- The Hidden Information and lack of temporal relevance made standard state-of-the-art techniques, such as Monte-Carlo Tree Search or Deep LSTM memory units as used by Alpha Zero [1] and Alpha Star [2] respectively inapplicable for our task
- Many design decisions including the design of the training data buffer involve numerous hyperparameters; thus making tuning of the Deep-RL model exceedingly challenging
- The inherent randomness of the game also makes evaluation challenging

Future Exploration

- Improved hyperparameter tuning
- Exploration of model complexity through analysis of both deeper and shallower models
- Exploration of the Sigmoid Activation function and MSELoss
- Exploration of additional features and other state representations
- Exploration of the effect of the Embedding Dimension

References

[1] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. Nature, 550(7676):354, 2017.

[2] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Jun-young Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, pages 1–5, 2019.