Contextual Preference Ranking and Reinforcement Learning for Gin Rummy

Mykola Zaitsev
k12147393

2025W: 351.072

Overview

- Gin Rummy is a card game where the player constantly has to evaluate whether a new card fits well into the hand she is holding
- Contextual preference ranking can learn such information using Siamese neural networks
- Reinforcement learning can be readily applied as the games are comparably short and there is a clear feedback signal

Task:

- Realize a Gin Rummy player based on a combination of contextual preference ranking and reinforcement learning

Content

Gin Rummy

Objective: score more points

- Run: sequences of three or more cards in the same suit e.g. **3♥ 4♥ 5♥**
- Set: three or four cards sharing the same rank e.g. **8♥ 8♦ 8♠**

Standard deck, King high, Ace low

Discarding card until all cards in hand are part of runs or sets

Players receive point based on opponents unmatched cards (deadwood)

Score is the difference between value of unmatched cards + bonuses (e.g. bonus for having no deadwood)

Q-Learning with Siamese Neural Network

Integrating Q-learning with a Siamese Neural Network allows reinforcement learning agent to evaluate card similarity and make better decisions.

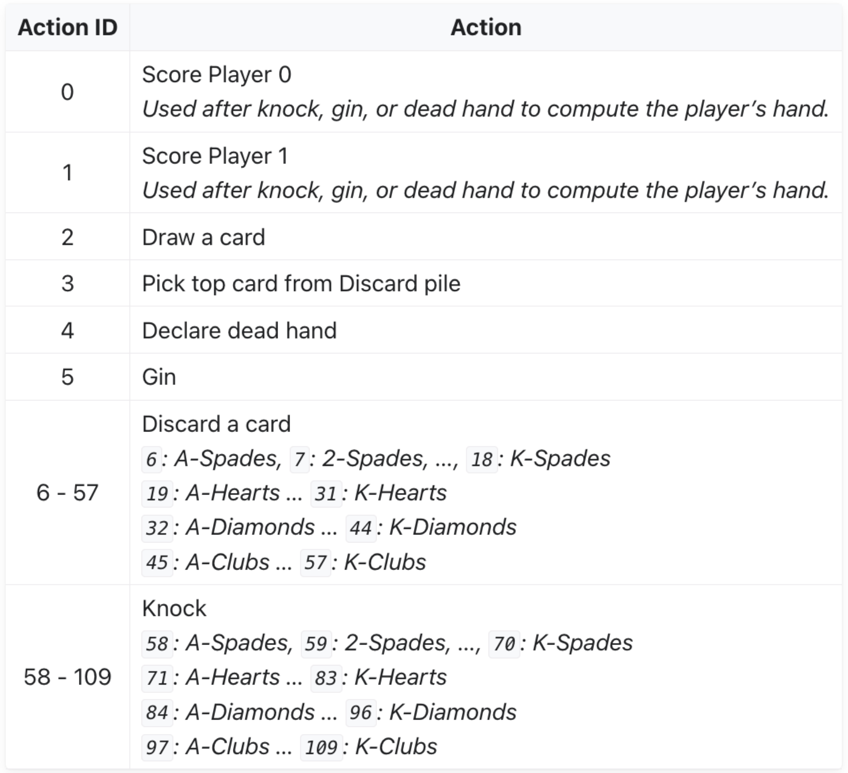
Training process:

- Convert the hand into an SNN state embedding
- Select an action based on the Q-table
- Update the Q-table

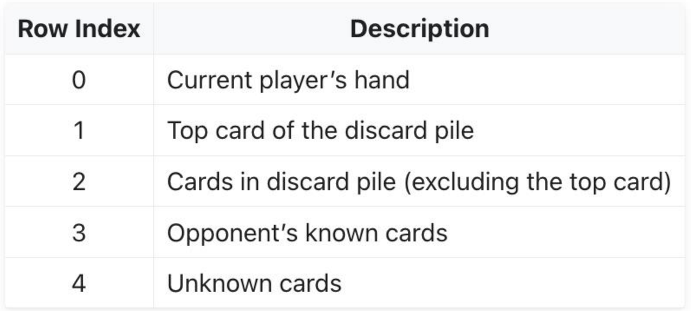
Review of the code and methods

Environment: gin_rummy_v4 environment from pettingzoo library

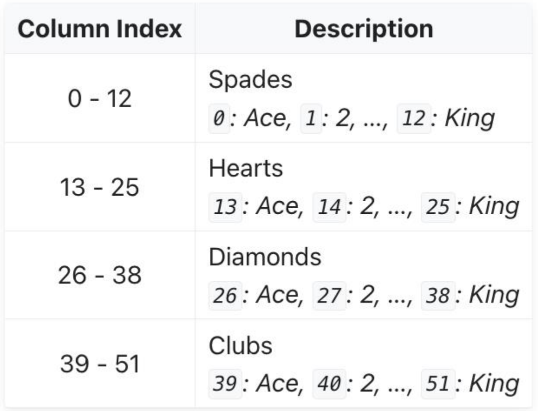
```
from pettingzoo.classic import gin_rummy_v4
```



Action ID	Action
0	Score Player 0 *Used after knock, gin, or dead hand to compute the player's hand.*
1	Score Player 1 *Used after knock, gin, or dead hand to compute the player's hand.*
2	Draw a card
3	Pick top card from Discard pile
4	Declare dead hand
5	Gin
6 - 57	Discard a card `6` *: A-Spades,* `7` *: 2-Spades, ...,* `18` *: K-Spades* `19` *: A-Hearts ...* `31` *: K-Hearts* `32` *: A-Diamonds ...* `44` *: K-Diamonds* `45` *: A-Clubs ...* `57` *: K-Clubs*
58 - 109	Knock `58` *: A-Spades,* `59` *: 2-Spades, ...,* `70` *: K-Spades* `71` *: A-Hearts ...* `83` *: K-Hearts* `84` *: A-Diamonds ...* `96` *: K-Diamonds* `97` *: A-Clubs ...* `109` *: K-Clubs*



Row Index	Description
0	Current player's hand
1	Top card of the discard pile
2	Cards in discard pile (excluding the top card)
3	Opponent's known cards
4	Unknown cards



Column Index	Description
0 - 12	Spades `0` *: Ace,* `1` *: 2, ...,* `12` *: King*
13 - 25	Hearts `13` *: Ace,* `14` *: 2, ...,* `25` *: King*
26 - 38	Diamonds `26` *: Ace,* `27` *: 2, ...,* `38` *: King*
39 - 51	Clubs `39` *: Ace,* `40` *: 2, ...,* `51` *: King*

Review of the code and methods

SNN: Three layers with first layer 52 input size for 52 cards in deck

```
class SNN(nn.Module):
    def __init__(self):
        super(SNN, self).__init__()
        self.fc1 = nn.Linear(52, 64)
        self.fc2 = nn.Linear(64, 32)
        self.fc3 = nn.Linear(32, 16)

    def forward_one(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.relu(self.fc3(x))
        return x

    def forward(self, hand, card):
        hand_embedded = self.forward_one(hand)
        card_embedded = self.forward_one(card)
        distance = torch.abs(hand_embedded - card_embedded)
        similarity_score = torch.sigmoid(-torch.norm(distance, dim=1))
        return similarity_score
```

Review of the code and methods

Q-learning: Epsilon-greedy action selection with Siamese network to evaluate card fit

```
class RLAgentWithRanking:
    def __init__(self, action_space, observation_space, siamese_model, epsilon=0.1,
alpha=0.1, gamma=0.99):

    def get_state(self, observation):…

    def choose_action(self, state, action_mask):…

    def update_q_table(self, state, action, reward, next_state, done, action_mask):…

    def train_siamese(self, current_hand, new_card, target):…
```

Review of the code and methods

Reward Computing:

- Based on deadwood reduction
- Based on melds over reducing deadwood

Training:

- Player 0 uses Q-Learning with Siamese Neural Network to evaluate card usefulness
- Player 1 has random policy
- 1000 Episodes

Findings and results

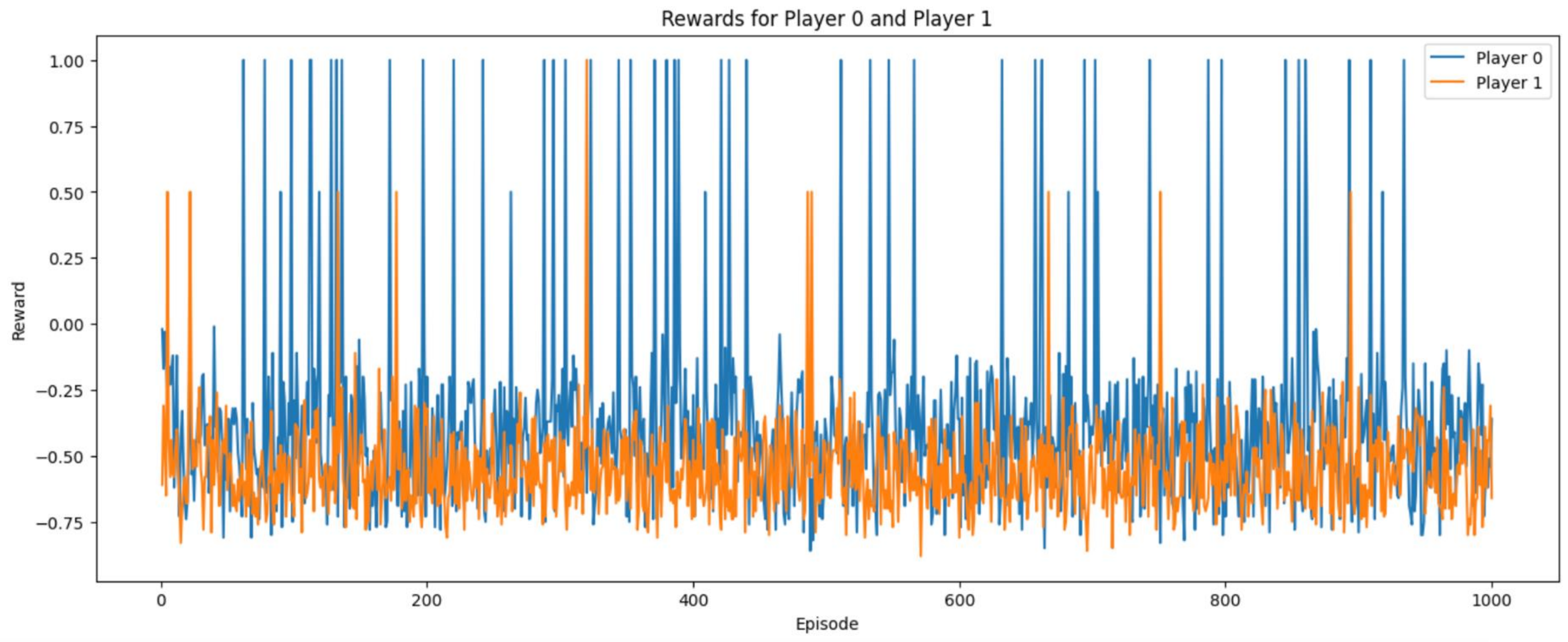
Based on deadwood reduction

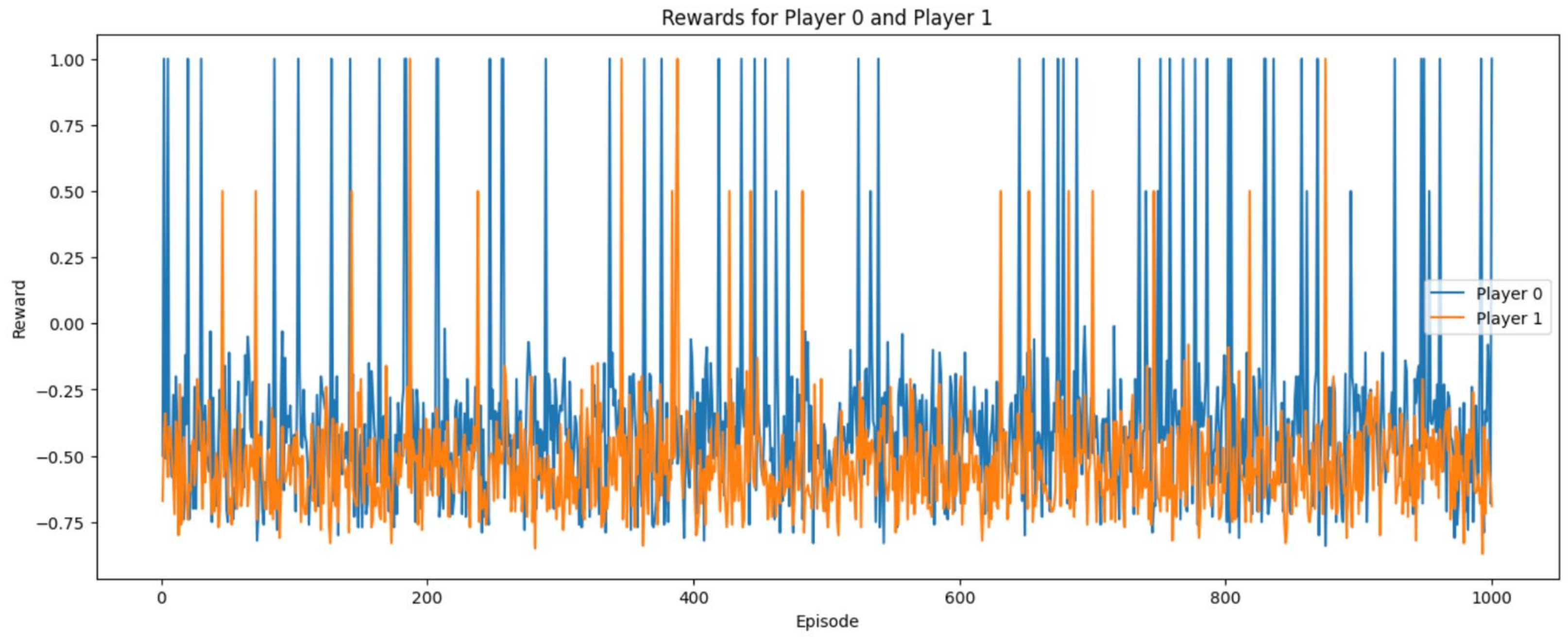
- Player 0 went Gin 43 times
- Win percentage: 70.0%

Based on melds over reducing deadwood

- Player 0 went Gin 51 times
- Win percentage: 68.89%

Much more times went Gin however the overall win percentage is a little worse





2025W: 351.072

Conclusions

- Gin Rummy is a good game to work with as there is not much actions and all the work is focused on evaluating the cards in the hand
- SNN helps Q-learning with generalization and better decision-making.
- Deadwood reduction policy has a higher win rate however policy based on melds over reducing deadwood has a higher chance of earning the maximum reward

For future work, we can consider using Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) in combination with SNN