Analyzing Strategies for Voronoi Area Game

Samir Dileep & Sandipan Samanta

- **Idea:** Predict optimal moves for victory in the Voronoi diagram area game and analyse resulting patterns.
- **Midway Goals:**
  1. Developing the game interface.
  2. Generating dataset containing complete instances of the game.
  3. Implementation of ML algorithms on said dataset.

- **Development of Interface:**

```python
def Voronoi(bgset, red_markers, blue_markers, quarrantine_distance=5):

    # Calculating closest distance from red and blue markers
    red_dist = (np.array([np.sum((bgset-pt)**2, axis=1) for pt in red_markers]).min(axis=0))**(0.5)
    blue_dist = (np.array([np.sum((bgset-pt)**2, axis=1) for pt in blue_markers]).min(axis=0))**(0.5)

    # Classifying points in the region as red, blue or black
    # on the basis that it's closest to red or blue or equidistant from both
    red_pt = np.array([bgset[i] for i in np.where(red_dist<blue_dist)[0]])
    blue_pt = np.array([bgset[i] for i in np.where(blue_dist<red_dist)[0]])
    black_pt = np.array([bgset[i] for i in np.where(red_dist==blue_dist)[0]])

    # not_available_points = np.array([bgset[i] for i in np.where(red_dist <= quarrantine_distance)[0]]
    # available_points = np.array([x for x in bgset if x not in not_available_points])

    return red_pt, blue_pt, black_pt
```

```python
def voronoi_plot(voronoi_cells, red_markers, blue_markers, show_black=False):
    red_pt, blue_pt, black_pt = voronoi_cells
    r, b = percent(voronoi_cells)
    fig, ax = plt.subplots()
    plt.plot(red_pt[:, 0], red_pt[:, 1], color=(1,0,0,0.15), marker="o", linestyle="")
    plt.plot(blue_pt[:, 0], blue_pt[:, 1], color=(0,0,1,0.15), marker="o", linestyle="")
    if show_black:
        plt.plot(black_pt[:, 0], black_pt[:, 1], color=(0,0,0), marker=".", linestyle="")
    plt.plot(red_markers[:, 0], red_markers[:, 1],"ko")
    plt.plot(blue_markers[:, 0], blue_markers[:, 1], "ko")
    ax.set_aspect("equal")
    plt.title("Red: "+str(r)+"%    Blue: "+str(b)+"%")
    plt.show
```

Link to the full code in Google Colab

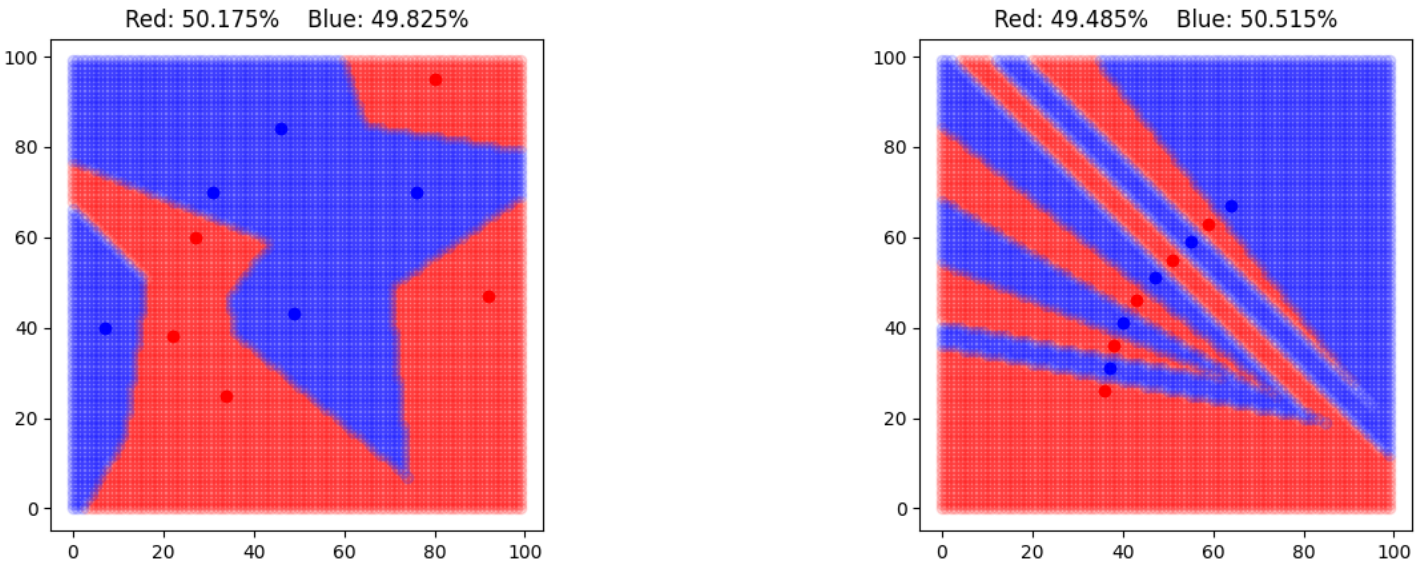

Figure: (Left to right) Results from the interface for random and greedy playstyles for the Voronoi game in a 100×100 grid with a qurantine distance of 5 units.

- **Implemetation of ML:**
  - Initially, we planned to employ Q-learning, a reinforcement learning technique, for training the model to play the Voronoi Area Game optimally.
  - Q-learning involves maintaining a Q-matrix, which stores the expected future rewards for each state-action pair.
  - In the Voronoi Area Game, the state representation needs to encode the placements of points by both players in all previous turns. As a result, the number of possible states grows exponentially with each turn, leading to an impractically large Q-matrix, even for a relatively small 100 x 100 grid.

- **Endterm Goals:**
  1. Finding a compact and efficient state representation to utilise during the learning process (Currently experimenting with training CNNs for each player).
  2. Complete training of both models and note the observations.