**Batuhan Erden - S004345**

# CS 445/545 – Playing Checkers with DP Value Iteration

## Markov Decision Process

Suppose that we are at state **s**. We take our own possible actions and move to our own potential spots. Then opponent takes over the player turn and from our own potential spots and it takes its own possible actions and move to its own potential spots. At the moment, **s → our possible actions → our potential spots → opponent's possible actions → opponent's potential spots**. Now, we will choose an action **a** from **our possible actions** which maximizes the **sum** of the **values (obtained using value function)** of all **opponent's potential spots** that can be reached from **s** by taking the action **a**. Then we will assign this **maximum sum** to **max gain**. After that, we will calculate the **probability** of **each spot** in **opponent's potential spots** by **1/opponent's potential spots**. Each spot should have the same probability. In this MDP, probability can also be though as the mean. **Probability and Max Gain** will be used in the value iteration algorithm.

## The RL Algorithm

States are the board spots. In the value iteration algorithm, the algorithm iterates through all of the states in the state space. There are currently 117622 possible states in the state space (It's the states.json file). I generated this states.json file with Emir Arditi and Cihan Eran.

The aim is to **converge**. The outer **while loop** never ends until **convergence** is achieved (The absolute value of the difference between old value function and new value function is smaller than **Epsilon=0.0001**). When it **converges**, it can be said that the **dynamic programming** did its job and all of the values are iterated.

There is an inner **for each loop** inside of the outer **while loop** that iterates through all states in state space**.** Values of the states are calculated here. The value of a state is calculated by looking one step ahead. Each state has a reward that is calculated by the reward function. On the other hand, we also have a non-futuristic **discount factor**. In this case, **discount factor**, **Gamma**, is set to 1/10.

1. **Probability, Max Gain ← MDP(state)**
2. **Value[state] ← Reward[state] + Gamma * Probability * Max Gain**

## Reward Function

It simply takes the state and calculates the reward. Normally, the reward of a state is calculated with this formula: ***1.5 * (player1's pieces + 3.33 * player1's kings - player2's pieces - 3.33 * player2's kings)***. However, if a state is a terminal state, the rewards are ***+100, -100, 0*** for winnings states, losing states and draw respectively.