TicTacToe
Double Deep Q-Learning Network
Mehak Gupta

Introduction:

In this project I have implemented TicTacToe using Double Deep Q-Learning Network. It contains of two players Player X and Player O. The state set for each player will contain all the boards when it was their turn to play. The action set contains all the options that player will have when it is his turn. Actions will be all the empty places in board at the time when it is agent's turn to play. "LeaglMoves" method is used to generate all the valid states for each action that can be taken by the agent. Reward will be +10 if the current player wins, -10 if the other player wins, +5 if it Draws, and 0 otherwise. There is also toss functionality implemented to decide which player will take first move. However, for learning purpose toss selects each player alternatively so each player has equal number of first turns.

Experiment Design:

The experiments are conducted for three different game size – 3, 5, and 7. For each game size experiment is conducted for three different agent settings. First setting has both players playing randomly. Second setting has Player X playing intelligently by learning through DDQN model and Player O playing randomly. Third setting, has both players learn through their own DDQNs and play intelligently. Second and third settings are also called modes – Easy and Hard respectively in the application. An intelligent agent chooses the optimal move by maximizing the minimum score of its next state. That is, for each state given to the agent, it finds all the possible actions available to him and then also looks for all the actions that an opponent can take after each of his action. For all the sets of opponent actions it remembers the minimum score. Then selects the action that gives maximum minimum score state for the agent's next move.

For each game size, the network contains 2 hidden dense layers, the first contains (game size)^2 * 3 number of neurons with a relu activation function and the second contains (game size)^2 * 2 number of neurons with a relu activation function. The last layer of the network is a dense layer with 1 output neuron with a linear activation function and loss function is mean absolute error. Learning rate is 0.1, Gamma is 0.5 and epsilon is 0.1.

In order to train the network, for every state a target value will be calculated:

$$target = Q(s, a) + \alpha\,(R + \Upsilon Q(s', argmax_{a'} a') - Q(s, a))$$

where $q(s,a)$ and $q(s',a')$ are calculated from the neural network itself.

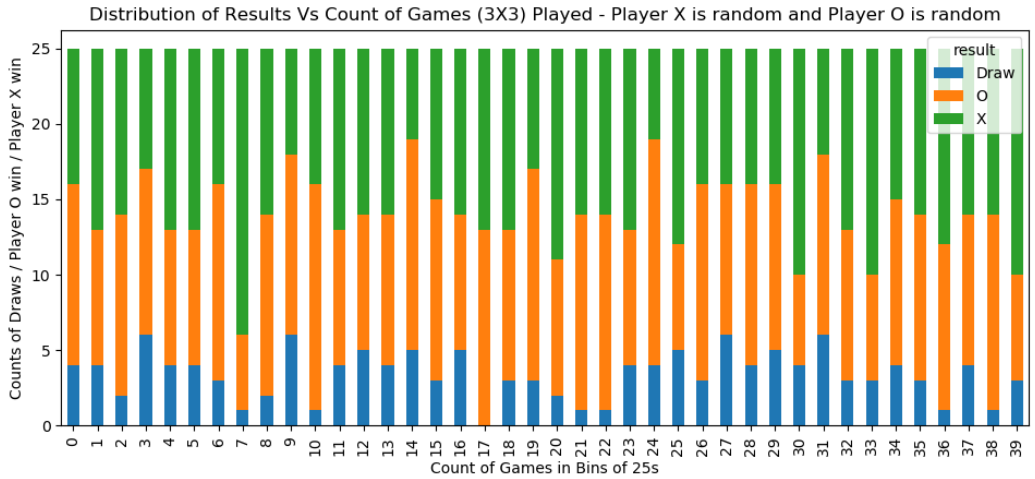
In DDQN, two different networks are used to compute the target value. For Q(s′, $argmax_{a'}$ a′), a′ – best action to be taken in next state, is computed using the learning model DQN and Q(s′, $argmax_{a'}$ a′)– Q value of taking that action in next state, is computed using target network. The target network is similar to actual DQN network but it helps to keep the Q(s′, $argmax_{a'}$ a′)stable for some time and after certain number of iterations all the weights learned by DQN network are copied to the target network. Otherwise the q value and target value both will change with every iteration and it will be like chasing a moving target. So, to stabilize the value of target for few iteration we use DDQN model introduce by Deep Mind. After each target calculation, one iteration of stochastic gradient descent is executed.

Results:

3 X 3

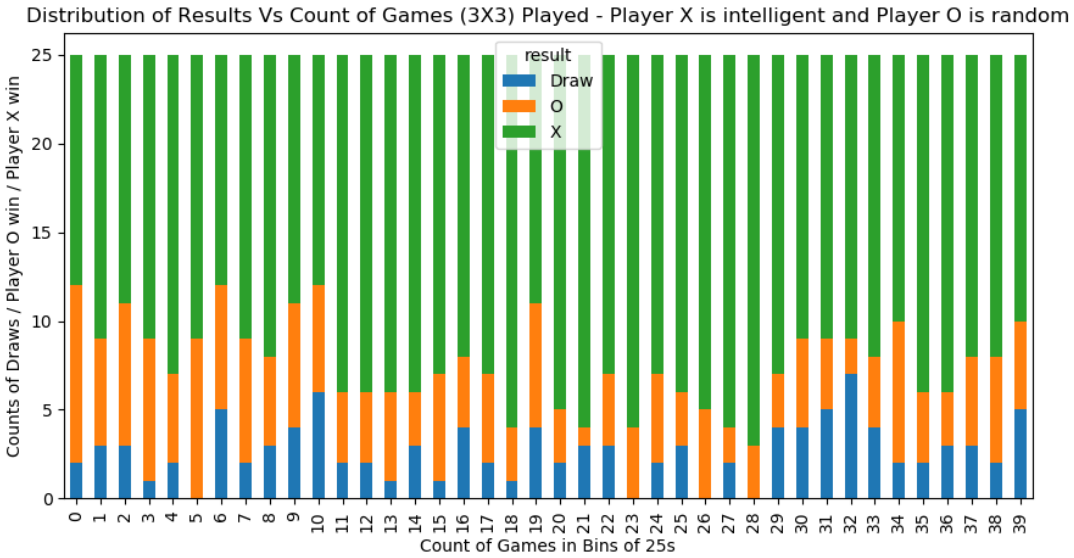
Baseline - Both Players are random

Due to small size of the game, random players win and loss probabilities are higher than draw probabilities.



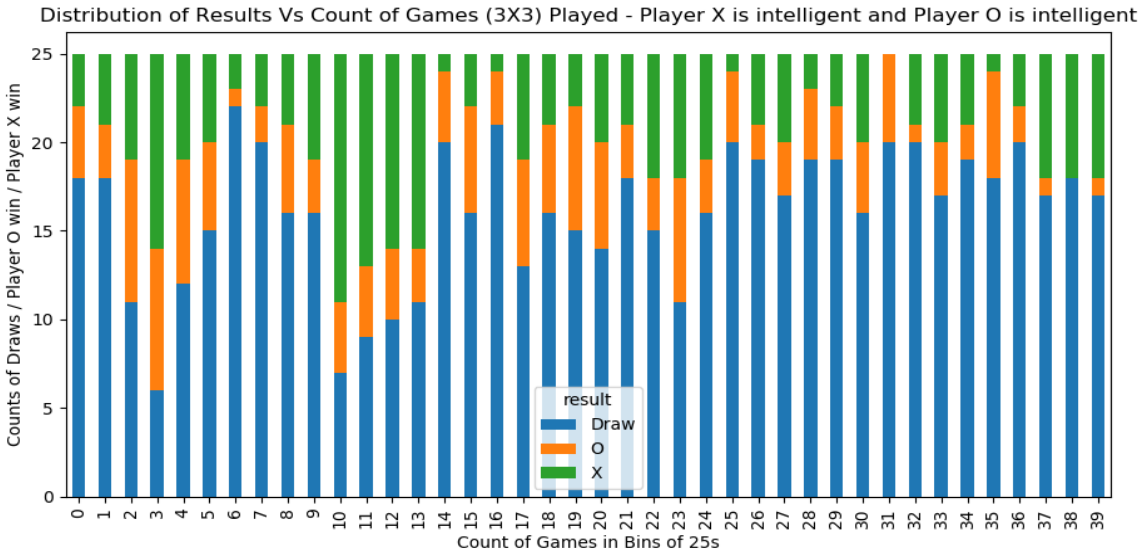
Easy Mode - Only Player X is intelligent

Player X has increased number of win as compared to Player O.



Hard Mode – Both players are intelligent

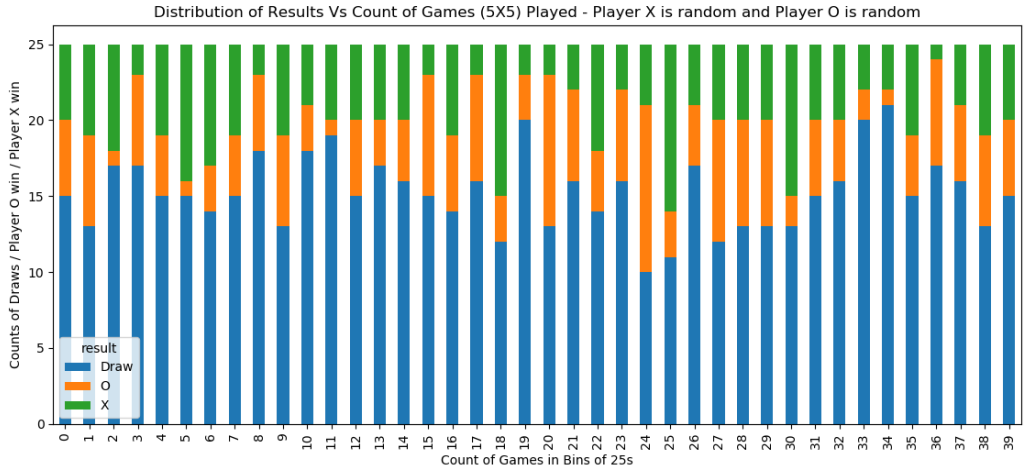
Since both players are intelligent, the number of draws increases.



5 X 5

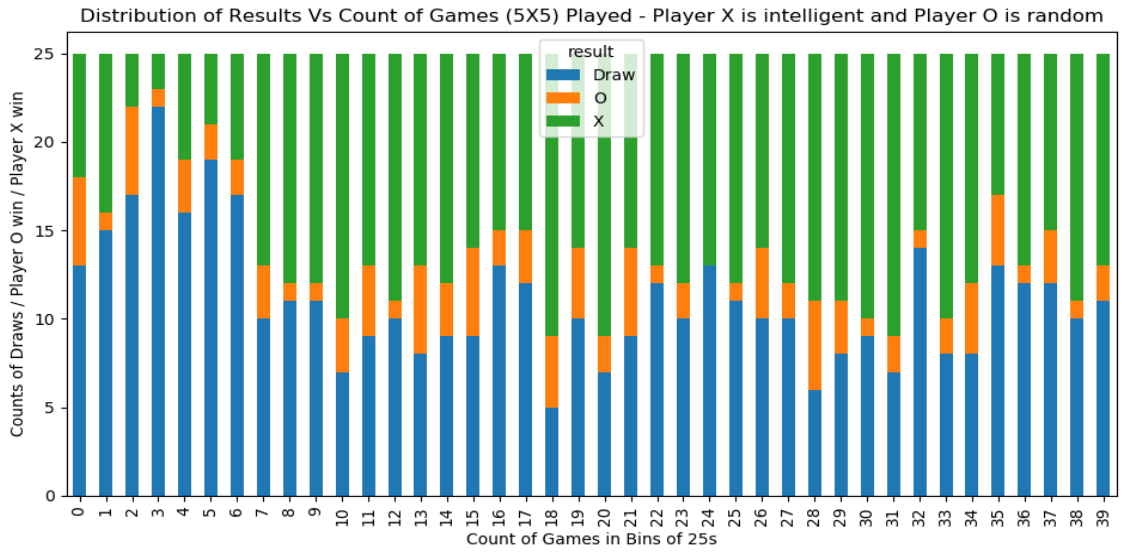
Baseline - Both Players are random

Due to large size of the game, random players win and loss probabilities are lower than the draw probabilities.



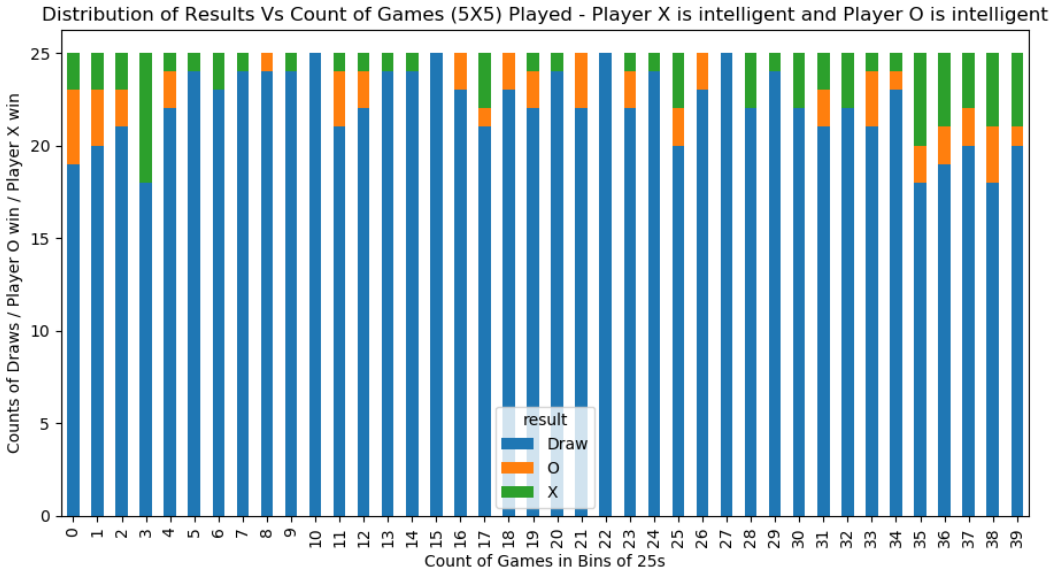
Easy Mode - Only Player X is intelligent

Player X has increased number of win as compared to Player O.



Hard Mode – Both players are intelligent

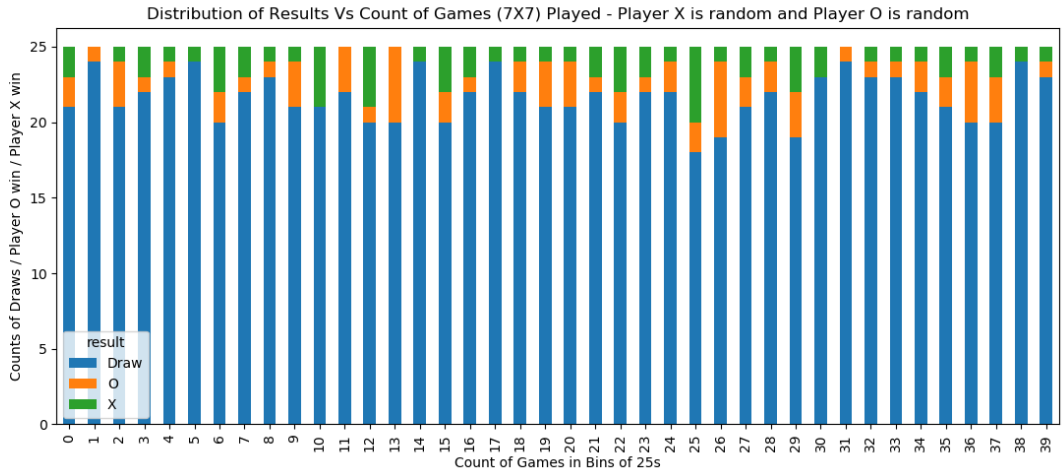
Since both players are intelligent, the number of draws increases. But since game size is large so there is not much difference from the baseline because in large space random players tend to draw more than the win or loss.



7 X 7

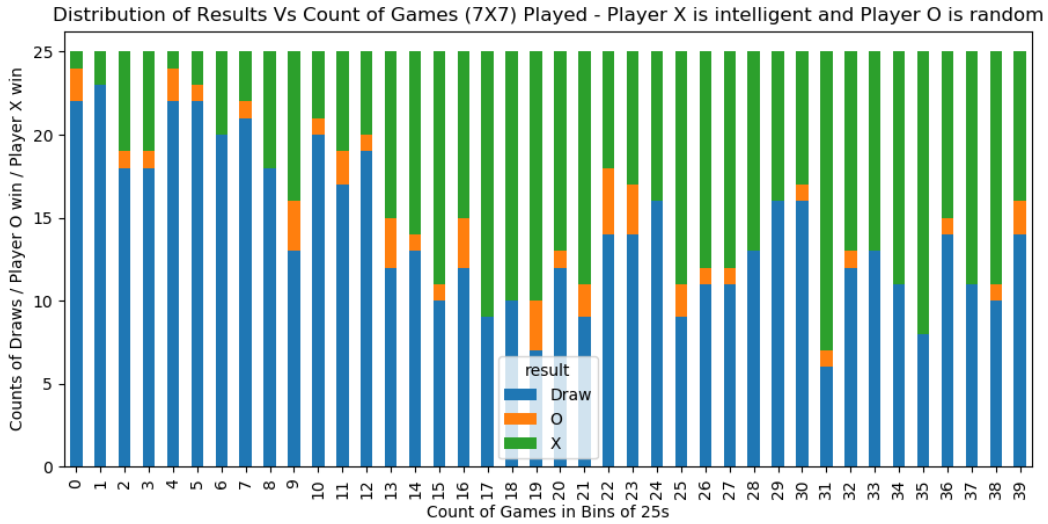
Baseline - Both Players are random

Due to large size of the game, random players win and loss probabilities are lower than the draw probabilities.



Easy Mode - Only Player X is intelligent

Player X has increased number of win as compared to Player O



Hard Mode – Both players are intelligent

Since both players are intelligent, the number of draws increases. But since game size is large so there is not much difference from the baseline because in large space random players tend to draw more than the win or loss.