CSCI 4155 Machine Learning: Assignment 7
TD Learning & TicTacToe

By Derek Neil, Nathan Schucher

Summary

We applied TD-learning to the problem of TicTacToe to attempt to generate a table of values that would enable an artificial player to compete with a human. We discovered difficulties in measuring the effectiveness of learning, and learned a lot about the domain through the act of training our machines.

Optimal strategy

Looking at the sequence of play in tictactoe, the opening move has three distinct possible moves. To gain insight about those moves however, we need to look at the second players first move.

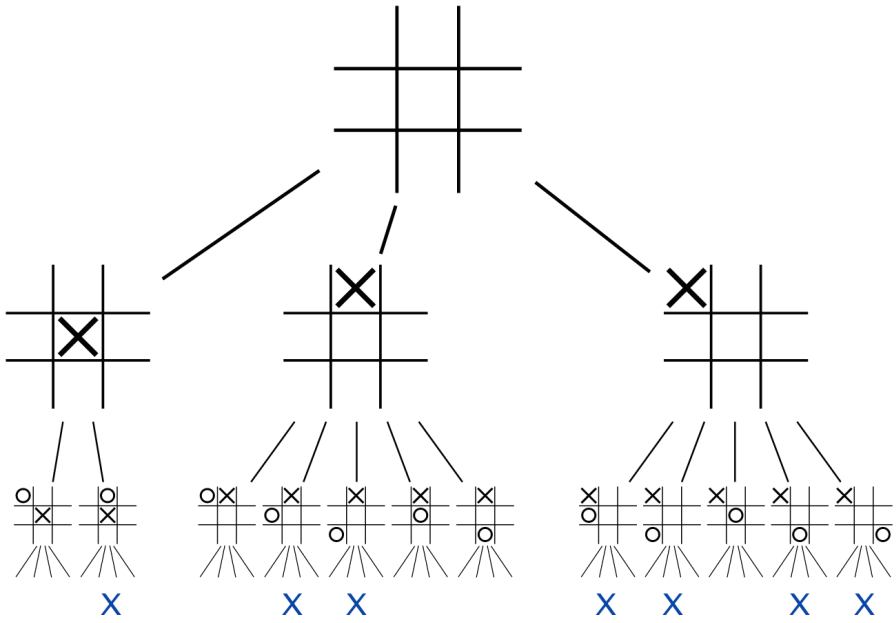

Figure 1: distinct tic tac toe opening moves for both players

Referring to figure 1 we can see of X's three opening moves, O then has 12 distinct opening moves. After O has made it's first move we can say something about how the game will progress. Of O's 12 distinct opening moves, 7 of them can always lead to X winning if X plays the optimal strategy (henceforth known as Xgames). Note that the distribution of Xgames is not uniform, it's %50 if X plays center first, %40 if X plays edge center first, and %80 if X plays a corner first. We would then hypothesize that X would favor corners and the center, over the center of edges.

In response to Xgames, we would also hypothesize that O would favour opening moves that did not lead to Xgames. Not being Xgames, O would at least draw, unfortunately,

since O always going second, there are fewer board states that lead to O winning, and if X plays an optimal game in a non Xgame, then the outcome will always be a draw. This leads us to another measure of performance. Once both players have learned optimal games, the draw rate should be %100 provided draws are rewarded higher than losses.

Symmetry

As previously discussed, X infact only has 3 distinct opening moves. All other opening moves are simply mirrors, or rotations of these three distinct moves. The same can be said about many other board states as the game is played. Taking advantage of these equivalent board states, we can speed up the "exploration" time at the beginning of the game, and assign values to all equivalent states since they lead to the same actions.

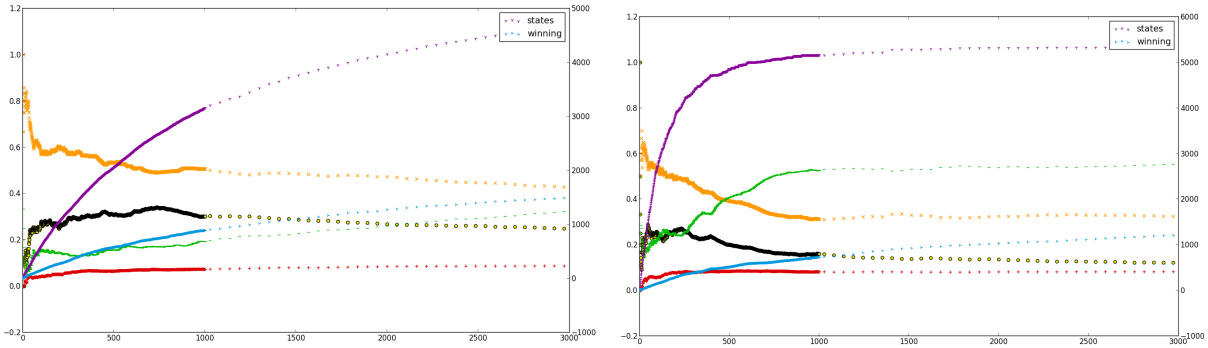

Figure 2: difference in state discovery (purple) and draw rate (green) for 3000 games of TD vs TD, first played without symmetry(left), then played with symmetry (right)

Examining the differences between the left and right of figure 2, we see significant decrease in the number of games it takes to discover all the playable board states. Using draw rate as a measure of optimal game play by two intelligent players, we see the draw rate also increase significantly faster when taking advantage of symmetry.

TD Learning

Our implementation focused on TD learning, with a learning rate between %15 (figures 2....) to %30 and a random move selection between %5 and %10. Rewards were awarded as +5 for a win, +1 or +4 for a draw, -5 for a loss. Varying the draw reward did not seem to have a significant impact on time to converge on optimal games by either player, however, taking advantage of symmetry meant we had to do batch updating of values to prevent inflating or deflating changes in values that were on symmetric boards.

TD vs Random

One of our experiments was to train a TD-learner on an entirely random player without taking advantage of symmetry. This should give chance to see how well TD-learning can generalize without “cheating” or giving it advantages. In Figure 3 below, can see how a TD-learner going first can very quickly learn an optimal strategy against a random player.

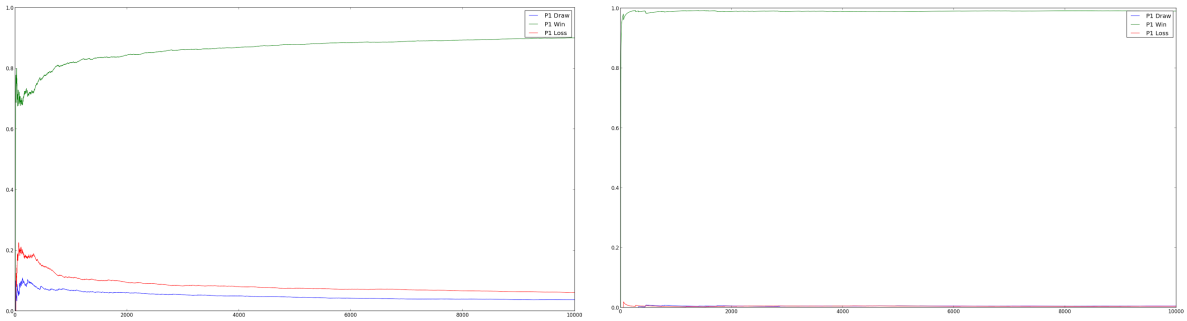

Figure 3: To left, win rate against random with a TD-learner; to right, 10,000 evaluation games after training

Notice that it starts off already winning, as first player has a higher chance of winning but it fairly quickly learns optimal opening moves. We tested the accuracy by playing entirely on-policy (ie. setting epsilon to 0) for 10,000 evaluation runs, after 10,000 training runs. Immediately the win-rate jumps up to near 100%, with the only losses being in games that hadn't been visited yet.

An important thing to keep in mind is that players can appear to be playing intelligently, when in reality, the statistics of the game make it such that they win an deceivingly high amount of times. Our software had critical bugs that prevented TD-learning from taking place in any measurable way, but appeared to still being performing "decently" (ie. 60% wins). After considering the domain, and the fact that an intelligent player should *never* lose, we fixed these bugs, and the learning actually began.

TD vs TD

With the opening piece X being fixed by the game, a TD learner can effectively play itself since the board states for X and O are duals of each other. Said another way, training a TD learner to play as X then, seeing how well it plays as O means that it would not have any values for O on it's first game playing as O and would be no better than a random player.

Random Moves & Restarting Game Scores

Using a version of TD learning that had an element of randomness to it allowed for further exploration of states that might have otherwise been cut off by some other combination of previous games. Using a figure such as %10 random moves, when there is more than one move available seemed like a good round number. Show in figure 2 and 4 we can see the red + showing running average number of random moves as a fraction of total moves played.

Referring back to figure 2, as early as 1000 games (with symmetry), we see draw rates starting to level off at %55. But if both players have explored all the states, shouldn't they be playing optimal games? Eventually we realized that the random moves were only helpful for the initial learning and for a short time after all the states had been discovered. One solution was to scale down the percentage of random moves proportional to the number of wins for a given player. To test our idea in a more abrupt manner, we picked 10k games as

the cut off, and turned off random move selection for both players. We also added a new draw games statistic (red -) that reports the draw rate for the last 1000 games played. Shown in figure 4 Immediately after 10k games, and random moves are disabled for both players our new draw game marker immediately shows nearly %100 draw rate, and the average draw rate for all games (green -) shows a sharp change.

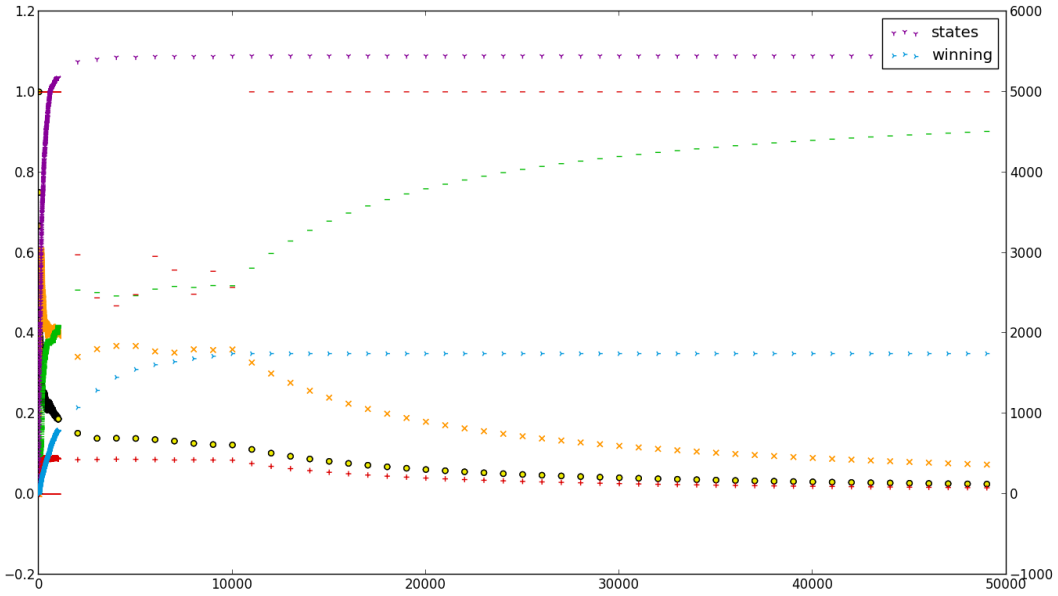

Figure 4: After 10k games, random moves are turned off for both TD players, draw rate for last 1000 games (red bar) immediately hits %100.

To confirm our suspicions as to whether a mere %10 random move selection was indeed contributing to the lack of draw games we then modified our program further. After 10k games, instead of disabling random moves for both players, we only disable it for one player for 10k games, then switch to the other for the next 10k games and so on, finally turning random play off for both players at 70k. Added to figure … are red x and o showing wins for that player over the last 1000 games.

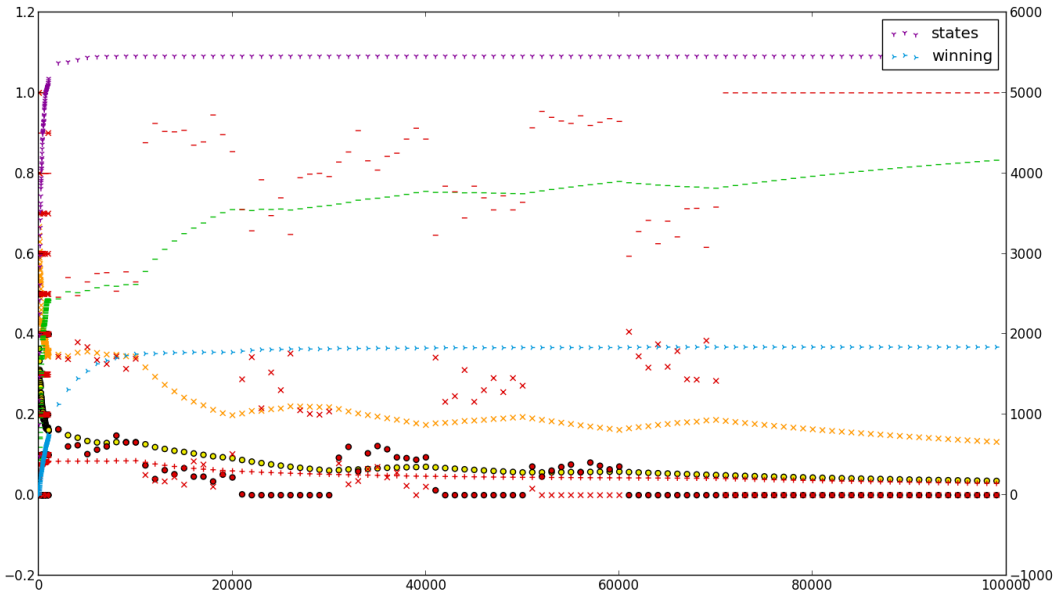

Figure 5: player with randomness turned on loses after 10k games, at 70k games draw rate for last 1000 games (red bar) jump, and stay at %100.

The result, as we predicted, the randomness had a significant impact on the outcomes of the games, even at %10, the player that was playing with randomness immediately began to lose more often compared to when both players were playing randomly. After 70k games we finally set both players back to playing without any random move selection and again we see the same red - draw rate for the last 1000 games shoot up to %100.

Human vs TD

In order to fully test our TD-learner we thought it would be best to play it against a human player. After ten thousand iterations the TD-learner continued to make severely sub-optimal moves. After 100,000 iterations the TD-learner plays much better, performing with some amount of strategy, but still loses sometimes. Some moves that are clearly optimal are missed by the learner, e.g., moves that are both win for O and block a winning line from X are sometimes missed (clearly not intelligent). With training in the millions of iterations it is conceivable that

Conclusion

One of our primary takeaways was the importance of random move selection to the learning process for TD learning, but also that it needs to be managed properly as the learner grows. Additionally, short cuts such as taking advantage of game symmetry worked very well for a simple game like tic tac toe, and would scale in a reducing manner to larger board with binary player pieces, however for game such as chess this would not have as big an impact beyond endgame and startgame pattern reduction.

Measuring the effectiveness of a TD-learner is tricky: it can be doing very well against random players and other TD-learners but still fail against a human player. An AI can be

playing completely randomly and still appear to be performing intelligently. It's important to have a number of metrics for measuring performance. Some that we used were measuring the values of opening moves, percentage win/loss/draw, adapting learning parameters, switching the opponent, reducing randomness of play to 0. Examining the results of this array of metrics can give better insight into whether or not learning is actually happening. This is harder, but more important, for domains where the optimal strategies are not known, or more ambiguous.