Playing Like a Gambler

George Lesica

CSCI 555 - FA 2012

Instroduction

Monte Carlo Tree Search is a method for making probabilistic decisions within a game tree without building the entire tree, which is computationally infeasible in many domains.

The basic algorithm works by intelligently sampling the tree and weighting subtrees based on the estimated probability that each will result in a favorable outcome. The basic algorithm is described below:

```
function MCTSSEARCH(s0)
    create root node v0 with state s0
    while within computational budget do
        vl = TREEPOLICY(v0)
        d = DEFAULTPOLICY(s(vl))
        BACKUP(vl,d)
        return a(BESTCHILD(v0))
```

The steps within the basic algorithm are described below.

Since MCTS is a statistical anytime algorithm (in other words, the longer it is run, the better the outcome is expected to be), it can be terminated at any time. Usually, the amount of time the algorithm is allowed to run is determined by a pre-set number of iterations or some temporal or computational budget [1].

Starting with the root of the tree (which represents the current state of the game or world) a tree policy is used to choose a node further down the tree. In the naive case, this could be simply choosing a random child of the root. In the case of the Upper Confidence Bound for Trees (UCT) variant this step involves a partial breadth-first traversal in which exploration is balanced with exploitation of suspected "good" paths using weights.

Once a node has been selected by the tree policy step, the default policy is used to simulate the game to its conclusion. This usually means making random

decisions until a terminal state is reached. This is where the magic happens, so to speak. Once a terminal state is reached, its value to the player is calculated and used to inform the eventual decision.

The backup step involves propagating the simulation value from the last step back up the tree. Generally, the algorithm tracks how many times it has explored "through" a particular state (visited that state during the tree policy step). It also tracks the total value of the outcomes that resulted from these instances. During the backup step, then, these values are adjusted.

UCT Variant

The actual implementation uses the Upper Confidence Bound for Trees algorithm, which is part of the MCTS family. Pseudo code and a more thorough description follow. The pseudo code is based on Browne [1].

```
function TREEPOLICY(v)
    while v is nonterminal do
        if v is not fully expanded then
            return EXPAND(v)
        else
            v = BESTCHILD(v, Cp)
    return v

function EXPAND(v)
    a = an untried action, valid at v
    v' = result of applying a to v
    return v'

function BESTCHILD(v, c)
    return argmax of the children of v, based on weight (see text)

function DEFAULTPOLICY(s)
    while s is nonterminal do
        choose a valid action based on s, uniformly at random
        s = result of applying the action to s
    return reward for state s

function BACKUP(v, d)
    while v is not null do
        increment visit count of v
        update value of v based on d
        v = parent of v
```

Connect Four

The rules of Connect Four are summarized below. The aspects of the game that make it a candidate for MCTS are then summarized.

Connect Four is played on a rectangular, usually vertically aligned, board. The board has channels in it, into which game pieces are dropped. The object of the game is to line up four pieces in a row, either vertically, horizontally, or diagonally.

Since each piece falls to the bottom of its channel, the number of potential moves on a given turn is, at most, the width of the game board. This also affects strategy. For instance, gaps are possible in the horizontal and diagonal directions, but not in the vertical direction.

Applying MCTS

MCTS is most-easily applied to games with an inherent tree structure. The simulation conducted during the default policy step would be much more computationally complex if cycle-checking or other such acrobatics were required. For tree-shaped games, simple random choices are sufficient.

Connect Four has a tree shape. There can be no cycles since there is only room for 42 pieces (on a standard board), one piece is added each turn, and pieces are never removed.

In order to determine its next move, the computer applies UCT to the set of available actions (columns with open spaces). It runs for a pre-set number of iterations, this essentially determines the difficulty of the computer AI opponent. Since the search space it quite large, even a fairly large number of iterations does not always result in a "good" move.

Interestingly, naive appliation of the algorithm seems to be adequate to identify "good" moves, but is less adept at identifying moves that, if not taken, will quickly result in a loss. This is likely due to the aheuristic nature of the algorithm.

References

[1] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, pp. 1–43, mar 2012.