Monte Carlo Tree Search

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057

1 January 1970
Overview

- MCTS consists of four main steps (Browne et al., 2012)
  1. Selection: Starting at the root, select the best action until reaching a node that has not been fully explored (i.e., a node with untried and therefore unevaluated actions).
  2. Expansion: Choose an action, and expand the tree by adding a child node.
  3. Simulation: From the newly added child, uniformly randomly select actions until reaching a leaf node and receiving a reward (e.g., +1 for winning, −1 for losing).
  4. Backpropagation: Starting at the new child node, propagate the reward to the root by adjusting the visit count $N(v)$ and the simulation reward $Q(v)$ of the nodes along the path.
Fig. 2. One iteration of the general MCTS approach.
Upper-confidence Bound for Trees (UCT)

1: function uctSearch(s₀)
2:   create a root node v₀ with state s₀
3:   while within computational budget do
4:     vᵢ ← treePolicy(v₀)
5:     Δ ← defaultPolicy(s(vᵢ))
6:     backup(vᵢ, Δ)
7:   end while
8:   return a(bestChild(v₀, 0))
9: end function
Tree Policy

1: function treePolicy(ν) 
2:   while ν is non-terminal do 
3:     if ν not fully expanded then 
4:       return expand(ν) 
5:     else 
6:       ν ← bestChild(ν, Cp) 
7:     end if 
8:   end while 
9: return ν 
10: end function
function expand(ν)
2: choose \( a \in \text{untried actions from } A(s(ν)) \)
3: add a new child \( ν' \) to \( ν \) with \( s(ν') = f(s(ν), a) \) and \( a(ν') = a \)
4: return \( ν' \)
5: end function
Best Child

1: function bestChild(v, c)
2: return argmax_{v' \in \text{Children}(v)} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}
3: end function
Default Policy

1: function defaultPolicy(s)
2:   while s is non-terminal do
3:     choose $a \in A(s)$ uniformly at random
4:     $s \leftarrow f(s, a)$
5:   end while
6:   return reward for state $s$
7: end function
Backup

1: function backup(v, Δ)
2:    while s is not null do
3:       N(v) ← N(v) + 1
4:       Q(v) ← Q(v) + Δ(v, p)  ▷ p is player
5:       v ← parent of v
6:    end while
7: end function
Backup Negamax

1: function backupNegamax(v, ∆)
2: while s is not null do
3:   N(v) ← N(v) + 1
4:   Q(v) ← Q(v) + ∆
5:   ∆ ← −∆
6:   v ← parent of v
7: end while
8: end function
Fig. 3. Asymmetric tree growth [68].
Monte Carlo Tree Search

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC  20057

1 January 1970