Memory-Augmented Monte Carlo Tree Search (M-MCTS)

Fan, Jett, and Audrina

Intro and background

Expectimax trees

\[ P(s_1 | s_t, a_1) * V(s_1) + P(s_2 | s_t, a_2) * V(s_2) \]
Intro and background

Expectimax trees → Simple MC search

\[ \pi(s_t) \rightarrow a_2 \]

\[ s' \sim p(s_t, a_2) \]

\[ \pi(s_1) \rightarrow a_2 \]
Intro and background

Expectimax trees → Simple MC search → Vanilla MCTS

\[ q(a_t, s_t) = \mathbb{E}_{p(s_{t+1}|a_t, s_t)}[\max_{a_{t+1}} \mathbb{E}_{p(s_{t+1}|s_t, a_{t+1})}[\max_{a_{t+2}} \ldots \max_{a_T} \mathbb{E}_{p(s_T|a_T, s_{T-1})}[r(s_{1:T})]]] \approx \frac{1}{K} \sum_{i=1}^{K} r(\hat{s}_{1:T,K}) \]
Intro and background

- Expectimax trees
- Simple MC search
- Vanilla MCTS
- M-MCTS
Main idea + contribution

Online generalization
Proving M-MCTS is better

Proof. We show that under condition (6), it can be guaranteed that $\Pr (-F_x(-c) + \tau \log M \leq \delta_x) \geq 1 - \beta$.

$$\Pr \left( -\tau \log \left( \sum_{i=1}^{M} \exp(-c_i/\tau) \right) \leq \delta_x - \tau \log M \right)$$

$$= \Pr \left( \sum_{i=1}^{M} \exp(-c_i/\tau) \geq \exp(-(\delta_x - \tau \log M)/\tau) \right)$$

$$\geq \Pr \left( \sum_{i=1}^{M} \exp(-\delta_i/\tau) \geq \exp(-(\delta_x - \varepsilon - \tau \log M)/\tau) \right)$$

$$\geq \Pr(\exists i, \exp(\delta_i/\tau) \leq \exp((\delta_x - \varepsilon - \tau \log M)/\tau)$$

$$= 1 - \prod_{i=1}^{M} \Pr(\delta_i \geq \alpha - \tau \log M)$$

$$\geq 1 - \prod_{i=1}^{M} \exp\left( - \frac{\alpha \tau \log M}{2\sigma^2} N_i \right)$$

$$= 1 - \exp\left( - \frac{\alpha \tau \log M^2 n}{2\sigma^2} \right)$$

OPT OBJ

$$\max_{w \in \Delta} \{ w \cdot (-c) \}$$

OPT OBJ (entropy regularized)

$$\max_{w \in \Delta} \{ w \cdot (-c) + \tau \cdot H(w) \}$$

MAIN POINT

$$\Pr(\text{eqn}(1) \leq \delta_x) \geq 1 - \beta$$
Query($x$)
Find top $M$ similar states in $M$, based on the distance function
$$d(\cdot, x) = -\cos(\phi(\cdot), \phi(x))$$ and compute the approximated memory value.

Add($x$)
Add new state $s$ by adding new memory entry
$$\{\phi(x), \hat{V}_x, N(x)\}$$ If the memory is full, then replace the least recently queried or updated memory entry with the new one.

Update($x$)
If $s$ is stored in the memory, only update $\hat{V}_x, N(x)$. 

$X$ \xrightarrow{\phi(x)} \phi(s), \hat{V}_s, N(s)$

$X$ \xrightarrow{\phi(x)} \phi(s), \hat{V}_s, N(s)$

$X$ \xrightarrow{\phi(x)} \phi(s), \hat{V}_s, N(s)$

$X$ \xrightarrow{\phi(x)} \phi(s), \hat{V}_s, N(s)$

$X$ \xrightarrow{\phi(x)} \phi(s), \hat{V}_s, N(s)$
Architecture - Selection & Expansion

\[
\begin{align*}
S_0 & \quad N(s_0) \quad V(s_0) \\
S_1 & \quad N(s_1) \quad V(s_1) \\
S_2 & \quad N(s_2) \quad V(s_2)
\end{align*}
\]

\[
UCB = V(s_i) + c\sqrt{(\ln(N)/N(s_i))}
\]

\[
\begin{align*}
S_0 & \quad N(s_0), N_M(s_0) \quad \hat{V}(s_0), \hat{V}_M(s_0) \\
S_1 & \quad N(s_1), N_M(s_1) \quad \hat{V}(s_1), \hat{V}_M(s_1) \\
S_2 & \quad N(s_2), N_M(s_2) \quad \hat{V}(s_2), \hat{V}_M(s_2)
\end{align*}
\]

\[
UCB_M = (1 - \lambda_{s_i})\hat{V}(s_i) + \lambda_{s_i}\hat{V}_M(s_i)
\]
All the trajectory of visited states are obtained in each stimulation.

Compute the memory value by \( \text{query}(s_1) \) operation of the memory \( M \).
Update the in-memory statistics by performing the \textit{update}(s_1) or \textit{add}(s_1) operation in the memory $M$. 
Architecture - Back-propagation

\[ N(s) \leftarrow N(s) + 1 \]
\[ \hat{V}(s) \leftarrow \hat{V}(s) + \frac{R - \hat{V}(s)}{N(s)} \]

\[ N_M(s) \leftarrow N_M(s) + X \]
\[ \hat{V}_M(s) \leftarrow \hat{V}_M(s) + \frac{R - \hat{V}_M(s) \ast X}{N_M(s)} \]
To not be time-consuming, Only compute the memory value at the leaf node
# The Overall structure of M-MCTS and MCTS is similar except adding a memory data structure

**Class node:**
V, V_m, N, N_m

**Class Memory:**
\{\phi(x), V, N\} * size of Memory
Query()
Update()
Add()

def selection(node):
    while node is fully expanded:
        node = ucb_M(node)
    return pick unvisited node.children or node

def rollout(node):
    feature = h(\zeta(node))
    M = closest states in memory by d(\cdot, node)
    V_m = \sum_{i=1}^{M} w_i \hat{v}(i)
    while node does not return a result:
        node = find actions that max the value
    V = result(node)
    Update()
    return V and V_m

def backpropagate(node, result):
    Update N and N_m
    Update V and V_m
    node.stats = (N, N_m, V, V_m)
    if is_root(node) return
    backpropagate(node.parent)
Figure 2: Experimental results. Figure (a)-(c) shows the results of testing different value of $M$. Figure (d) shows the results of testing different size of memory. In all figures, x-axis is the number of simulations per move, y-axis means the winning rate against the baseline.

Related Work

• Utilizing information
  
  • Kawano, Y. 1996. Using similar positions to search game trees
    
    • Uses the priority scheme to extend nodes. Similar positions have same priority (i.e., static) score.
    
    • MCTS is better -> rollout policy can utilize offline training results.

• Memory Architectures
  
  • Pritzel, A. 2017. Neural episodic control
    
    • A buffer of past experience containing slowly changing state representations and rapidly updated estimates of the value function (memory framework).
    
    • Both M-MCTS and NEC stored more information in memory.
    
    • M-MCTS & NEC have shown experiment results better than MCTS.
• Generalization

  • Childs, B. E. 2008. Transpositions and move groups in Monte Carlo tree search

    • Nodes in the same state share the same simulation statistics (Transposition table $\tau \approx \tau \approx 0$ in M-MCTS).

    • M-MCTS with $\tau > 0$ the memory can provide more generalization.

  • Srinivasan, S. 2015. Improving exploration in UCT using local manifolds

    • Uses kernel regression to approximate a state value function.

    • Equivalent to M-MCTS's addressing scheme using $w = f_\tau(-c)$

    • M-MCTS provides theoretical justifications but their work did not.
Limitation

- Online Value Approximation
  - might not generalized feature representation (offline)
  - compute and lookup time can be expensive

- Memory augmented
  - might face scalability issues (exploration vs exploitation)

- Lack of incorporating feature representation learning with M-MCTS in an end-to-end fashion.
Thanks!