Maximum Entropy Monte-Carlo Planning

Chenjun Xiao, Jincheng Mei, Ruitong Huang, Dale Schuurmans, Martin Müller

Presented by Sourav Bhattacharjee and Andrew Jung

STA 4273 Winter 2021 - Minimizing Expectations

March 25, 2021

Xiao, C., Huang, R., Mei, J., Schuurmans, D., and Müller, M. Maximum entropy monte-carlo planning. In Advances in Neural Information Processing Systems, pp. 9516–9524, 2019.

Main Idea

Augment Monte Carlo Tree Search (MCTS) with maximum entropy policy optimization to improve the worst case efficiency of UCT

Online Planning in Markov Decision Process (MDP)

Online planning problem: finding the optimal policy at a given state (s_0)

- An MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p(\cdot|s,a), r(s,a) \rangle$
- Generative model of $\mathcal{M}$ used to simulate state-action trajectories within a sampling budget

Online Planning in Markov Decision Process (MDP)

Online planning problem: finding the optimal policy at a given state (s_0)

- An MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p(\cdot|s, a), r(s, a) \rangle$
- Generative model of $\mathcal{M}$ used to simulate state-action trajectories within a sampling budget
- From the simulation, an optimal action is proposed for the input state, s_0

Online Planning in Markov Decision Process (MDP)

Online planning problem: finding the optimal policy at a given state (s_0)

- An MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p(\cdot|s,a), r(s,a) \rangle$
- Generative model of $\mathcal{M}$ used to simulate state-action trajectories within a sampling budget
- From the simulation, an optimal action is proposed for the input state, s_0
- The paper considers episodic and deterministic MDP for simplicity

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree to propose an optimal action for the root node

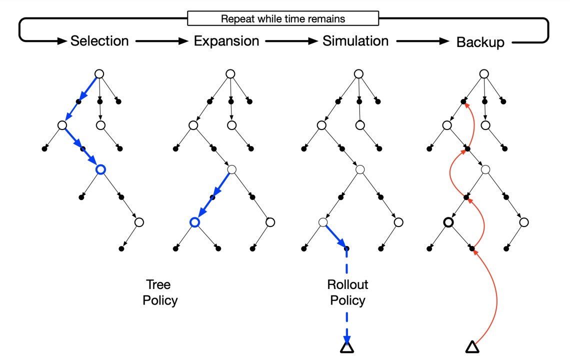


Figure 1: Each MCTS iteration consists of four phases. From Sutton, R. S. and Barto A. G., Reinforcement learning: An introduction. MIT press, 2018.

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree to propose an optimal action for the root node

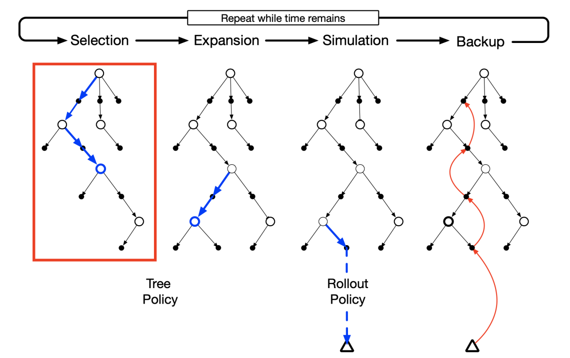


Figure 1: Each MCTS iteration consists of four phases. From Sutton, R. S. and Barto A. G., Reinforcement learning: An introduction. MIT press, 2018.

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree to propose an optimal action for the root node

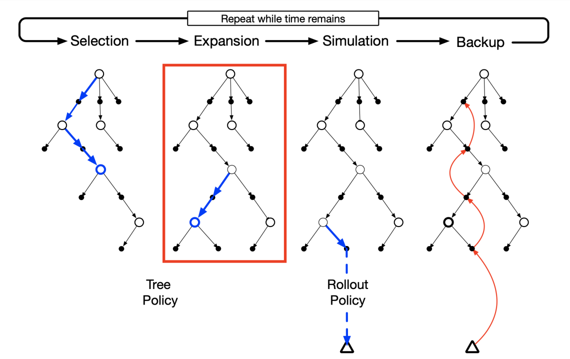


Figure 1: Each MCTS iteration consists of four phases. From Sutton, R. S. and Barto A. G., Reinforcement learning: An introduction. MIT press, 2018.

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree to propose an optimal action for the root node

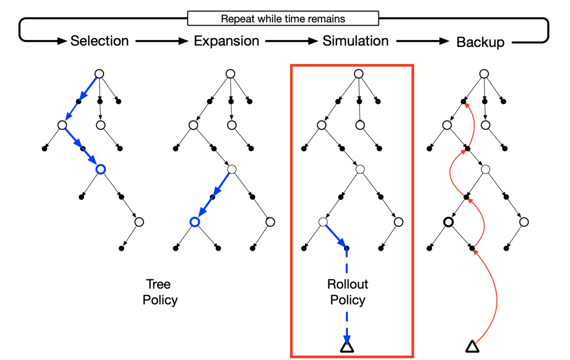


Figure 1: Each MCTS iteration consists of four phases. From Sutton, R. S. and Barto A. G., Reinforcement learning: An introduction. MIT press, 2018.

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree to propose an optimal action for the root node

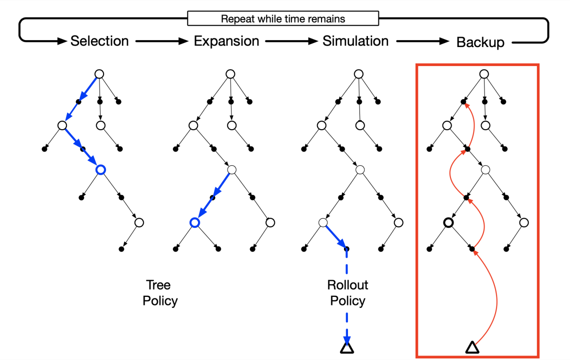


Figure 1: Each MCTS iteration consists of four phases. From Sutton, R. S. and Barto A. G., Reinforcement learning: An introduction. MIT press, 2018.

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree

- At the end of the iteration, node statistics updated from the backup (often $Q(s, a)$ and $N(s)$).

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree

- At the end of the iteration, node statistics updated from the backup (often $Q(s, a)$ and $N(s)$).
- MCTS iterations run until the budget

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree

- At the end of the iteration, node statistics updated from the backup (often $Q(s, a)$ and $N(s)$).
- MCTS iterations run until the budget
- Best action for the root node, s_0, chosen based on the node statistics

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree

- At the end of the iteration, node statistics updated from the backup (often $Q(s, a)$ and $N(s)$).
- MCTS iterations run until the budget
- Best action for the root node, s_0, chosen based on the node statistics
- Selectively sampling actions can improve the performance, especially in large search space

Monte Carlo Tree Search (MCTS)

MCTS uses simulated trajectories to incrementally build a search tree

- At the end of the iteration, node statistics updated from the backup (often $Q(s, a)$ and $N(s)$).
- MCTS iterations run until the budget
- Best action for the root node, s_0, chosen based on the node statistics
- Selectively sampling actions can improve the performance, especially in large search space
- Tree policy needs to balance exploitation with exploration

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

- Consider the choice of child node as a multi-armed bandit problem
- In K-bandits, the goal is to choose sequence of K actions to maximize the long-term expected rewards

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

- Consider the choice of child node as a multi-armed bandit problem
- In K-bandits, the goal is to choose sequence of K actions to maximize the long-term expected rewards

UCB1

At time t, choose action $a \in \{1, ..., K\}$ with the largest upper confidence bound (UCB):

$$UCB(a) = \hat{r}_t(a) + c\sqrt{\frac{\ln t}{n_{t,a}}}$$

where $\hat{r}_t(a)$ is empirical estimate of the reward from action a, $n_{t,a}$ is the number of times action a was played, and $c > 0$ is a constant.

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

- Consider the choice of child node as a multi-armed bandit problem
- In K-bandits, the goal is to choose sequence of K actions to maximize the long-term expected rewards

UCB1

At time t, choose action $a \in \{1, ..., K\}$ with the largest upper confidence bound (UCB):

$$UCB(a) = \hat{r}_t(a) + c\sqrt{\frac{\ln t}{n_{t,a}}}$$

where $\hat{r}_t(a)$ is empirical estimate of the reward from action a, $n_{t,a}$ is the number of times action a was played, and $c > 0$ is a constant.

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

- Consider the choice of child node as a multi-armed bandit problem
- In K-bandits, the goal is to choose sequence of K actions to maximize the long-term expected rewards

UCB1

At time t, choose action $a \in \{1, ..., K\}$ with the largest upper confidence bound (UCB):

$$UCB(a) = \hat{r}_t(a) + c\sqrt{\frac{\ln t}{n_{t,a}}}$$

where $\hat{r}_t(a)$ is empirical estimate of the reward from action a, $n_{t,a}$ is the number of times action a was played, and $c > 0$ is a constant.

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

- Consider the choice of child node as a multi-armed bandit problem
- In K-bandits, the goal is to choose sequence of K actions to maximize the long-term expected rewards

UCB1

At time t, choose action $a \in \{1, ..., K\}$ with the largest upper confidence bound (UCB):

$$UCB(a) = \hat{r}_t(a) + c\sqrt{\frac{\ln t}{n_{t,a}}}$$

where $\hat{r}_t(a)$ is empirical estimate of the reward from action a, $n_{t,a}$ is the number of times action a was played, and $c > 0$ is a constant.

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

UCT

At node s, choose action a with the largest $UCB(s, a)$:

$$UCB(s, a) = Q(s, a) + c\sqrt{\frac{\ln N(s)}{N(s, a)}}$$

where $Q(s, a)$, $N(s)$, and $N(s, a)$ are the node statistics for action-value and visitation counts.

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

UCT

At node s, choose action a with the largest $UCB(s, a)$:

$$UCB(s, a) = Q(s, a) + c\sqrt{\frac{\ln N(s)}{N(s, a)}}$$

where $Q(s, a)$, $N(s)$, and $N(s, a)$ are the node statistics for action-value and visitation counts.

- Asymptotically optimal: $Q(s, a) \xrightarrow{p} Q^*(s, a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$

Upper Confidence Bound (UCB) Applied to Trees (UCT)

UCT uses UCB1 from the bandits literature as a tree policy

UCT

At node s, choose action a with the largest $UCB(s, a)$:

$$UCB(s, a) = Q(s, a) + c\sqrt{\frac{\ln N(s)}{N(s, a)}}$$

where $Q(s, a)$, $N(s)$, and $N(s, a)$ are the node statistics for action-value and visitation counts.

- Asymptotically optimal: $Q(s, a) \xrightarrow{p} Q^*(s, a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$
- Probability of proposing a suboptimal action at the root after t iterations, $P(a_t \neq a^*)$, converges to zero at $O\left(\frac{1}{t}\right)$

Maximum Entropy Policy Optimization

Entropy regularization in RL to encourage exploration:

$$V^* = \max_{\pi} \{\pi \cdot \boldsymbol{r}\}$$

$$\pi^* = \arg\max_{\pi} \{\pi \cdot \boldsymbol{r}\}$$

where $\boldsymbol{r} \in \mathbb{R}^K$ is reward vector for K actions and $\tau \geq 0$

Maximum Entropy Policy Optimization

Entropy regularization in RL to encourage exploration:

$$V^* = \max_\pi \{\pi \cdot \boldsymbol{r}\}$$

$$V^*_{sft} = \max_\pi \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\}$$

$$\pi^* = \arg\max_\pi \{\pi \cdot \boldsymbol{r}\}$$

$$\pi^*_{sft} = \arg\max_\pi \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\}$$

where $\boldsymbol{r} \in \mathbb{R}^K$ is reward vector for K actions and $\tau \geq 0$

Maximum Entropy Policy Optimization

Entropy regularization in RL to encourage exploration:

$$V^* = \max_\pi \{\pi \cdot \boldsymbol{r}\}$$

$$V^*_{sft} = \max_\pi \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\}$$

$$\pi^* = \arg\max_\pi \{\pi \cdot \boldsymbol{r}\}$$

$$\pi^*_{sft} = \arg\max_\pi \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\}$$

where $\boldsymbol{r} \in \mathbb{R}^K$ is reward vector for K actions and $\tau \geq 0$

Maximum Entropy Policy Optimization

Entropy regularization in RL to encourage exploration:

$$
\begin{aligned}
V^* &= \max_{\pi} \{\pi \cdot \boldsymbol{r}\} \\
V^*_{sft} &= \max_{\pi} \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\} = \tau \log \sum_{a} \exp\{r(a)/\tau\} \\
&= \text{(smooth approximation to max)} \\
\pi^* &= \arg\max_{\pi} \{\pi \cdot \boldsymbol{r}\} \\
\pi^*_{sft} &= \arg\max_{\pi} \{\pi \cdot \boldsymbol{r} + \tau \mathcal{H}(\pi)\} = \exp\{(\boldsymbol{r} - V^*_{sft})/\tau\} \\
&= \text{(smooth approximation to argmax)}
\end{aligned}
$$

where $\boldsymbol{r} \in \mathbb{R}^K$ is reward vector for K actions and $\tau \geq 0$

Maximum Entropy Policy Optimization

Simplify the notation by introducing $\mathcal{F}_\tau(\boldsymbol{r}) = \tau \log \sum_a \exp\left(r(a)/\tau\right)$ and $\boldsymbol{f}_\tau(\boldsymbol{r}) = \exp\left\{\left(\boldsymbol{r} - \mathcal{F}_\tau(\boldsymbol{r})\right)/\tau\right\}$:

Solving the regularized problem:

$$V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r}),$$
$$\pi^*_{sft} = \boldsymbol{f}_\tau(\boldsymbol{r})$$

where $\boldsymbol{r} \in \mathbb{R}^K$ is reward vector for K actions and $\tau \geq 0$

Main Idea

Augment MCTS with maximum entropy policy optimization to improve the worst case efficiency of UCT

- Apply entropy regularization to bandit problem
- Apply regularized bandit to MCTS

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t
- Propose an optimal algorithm for softmax bandit problem and show this is optimal by two theorems: 1) there is a lower bound for $\mathcal{E}_t$ and 2) the proposed algorithm achieves the lower bound asymptotically

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t
- Propose an optimal algorithm for softmax bandit problem and show this is optimal by two theorems: 1) there is a lower bound for $\mathcal{E}_t$ and 2) the proposed algorithm achieves the lower bound asymptotically

Empirical Exponential Weight (E2W)

$$\pi_t(a) = (1 - \lambda_t)\,\boldsymbol{f}_\tau(\hat{\boldsymbol{r}}_t)(a) + \lambda_t \frac{1}{|\mathcal{A}|},$$

where $\lambda_t = \epsilon\,|\mathcal{A}|\,/\log(t+1)$ is decay rate for exploration and $\epsilon > 0$

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t
- Propose an optimal algorithm for softmax bandit problem and show this is optimal by two theorems: 1) there is a lower bound for $\mathcal{E}_t$ and 2) the proposed algorithm achieves the lower bound asymptotically

Empirical Exponential Weight (E2W)

$$\pi_t(a) = (1 - \lambda_t)\, \boldsymbol{f}_\tau(\hat{\boldsymbol{r}}_t)(a) + \lambda_t \frac{1}{|\mathcal{A}|},$$

where $\lambda_t = \epsilon\,|\mathcal{A}|\,/\log(t+1)$ is decay rate for exploration and $\epsilon > 0$

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t
- Propose an optimal algorithm for softmax bandit problem and show this is optimal by two theorems: 1) there is a lower bound for $\mathcal{E}_t$ and 2) the proposed algorithm achieves the lower bound asymptotically

Empirical Exponential Weight (E2W)

$$\pi_t(a) = (1 - \lambda_t)\,\boldsymbol{f}_\tau(\hat{\boldsymbol{r}}_t)(a) + \lambda_t \frac{1}{|\mathcal{A}|},$$

where $\lambda_t = \epsilon\,|\mathcal{A}|\,/\log(t+1)$ is decay rate for exploration and $\epsilon > 0$

Softmax Value Estimation in Bandit

Apply maximum entropy regularization to bandit problem (softmax bandit)

- The new entropy regularized objective: estimate the optimal softmax value $V^*_{sft} = \mathcal{F}_\tau(\boldsymbol{r})$ for some $\tau > 0$
- To achieve this, find a sequential sampling algorithm to minimize mean squared error $\mathcal{E}_t = \mathbb{E}\left[(U^* - U_t)^2\right]$ where $U^* = \sum_a \exp\{r(a)/\tau\} = e^{V^*_{sft}/\tau}$, $U_t = \sum_a \exp\{\hat{r}_t(a)/\tau\} = e^{(V_{sft})_t/\tau}$, and $\hat{\boldsymbol{r}}_t$ is the empirical estimate of $\boldsymbol{r}$ at time t
- Propose an optimal algorithm for softmax bandit problem and show this is optimal by two theorems: 1) there is a lower bound for $\mathcal{E}_t$ and 2) the proposed algorithm achieves the lower bound asymptotically

Empirical Exponential Weight (E2W)

$$\pi_t(a) = (1 - \lambda_t)\, \boldsymbol{f}_\tau(\hat{\boldsymbol{r}}_t)(a) + \lambda_t \frac{1}{|\mathcal{A}|},$$

where $\lambda_t = \epsilon\,|\mathcal{A}|\,/\log(t+1)$ is decay rate for exploration and $\epsilon > 0$

Optimal Sequential Sampling Strategy: E2W

Show E2W is optimal by demonstrating E2W achieves the lower bound of $\mathcal{E}_t$ asymptotically

Optimal Sequential Sampling Strategy: E2W

Show E2W is optimal by demonstrating E2W achieves the lower bound of $\mathcal{E}_t$ asymptotically

Theorem 1: lower bound on $\mathcal{E}_t$

In the stochastic softmax bandit problem, for any algorithm that achieves $\mathcal{E}_t = O\left(\frac{1}{t}\right)$, there exists a problem setting such that

$$\lim_{t \to \infty} t\mathcal{E}_t \geq \frac{\sigma^2}{\tau^2} \left(\sum_a \exp\left(r(a)/\tau\right) \right)^2$$

assuming all reward distributions are σ^2-subgaussian

Optimal Sequential Sampling Strategy: E2W

Show E2W is optimal by demonstrating E2W achieves the lower bound of $\mathcal{E}_t$ asymptotically

Theorem 1: lower bound on $\mathcal{E}_t$

In the stochastic softmax bandit problem, for any algorithm that achieves $\mathcal{E}_t = O\left(\frac{1}{t}\right)$, there exists a problem setting such that

$$\lim_{t\to\infty} t\mathcal{E}_t \geq \frac{\sigma^2}{\tau^2}\left(\sum_a \exp\left(r(a)/\tau\right)\right)^2$$

assuming all reward distributions are σ^2-subgaussian

Theorem 2: gauranteed convergence of E2W to the lower bound

For the softmax stochastic bandit problem, E2W can guarantee,

$$\lim_{t\to\infty} t\mathcal{E}_t = \frac{\sigma^2}{\tau^2}\left(\sum_a \exp\left(r(a)/\tau\right)\right)^2$$

Maximum Entropy for Tree Search (MENTS)

Maximum Entropy for Tree Search (MENTS) applies maximum entropy policy optimization to MCTS

- Building out a tree $\mathcal{T}$ online
- Each node $n(s) \in \mathcal{T}$ corresponds to a state s
- Each node has a softmax value estimate $Q(s, a)$ and visit count $N(s, a)$ associated with it for each action a
- $\boldsymbol{Q_{sft}}(s)$ denotes $|A|$-dimensional vector of components $Q_{sft}(s, a)$

Maximum Entropy for Tree Search (MENTS)

Maximum Entropy for Tree Search (MENTS) applies maximum entropy policy optimization to MCTS

1. Use E2W as tree policy

$$\pi_t\left(a|s\right) = \left(1-\lambda_s\right) \boldsymbol{f}_\tau\left(\boldsymbol{Q}_{sft}(s)\right)(a) + \lambda_s \frac{1}{|\mathcal{A}|},$$

where $\lambda_s = \epsilon\,|\mathcal{A}|\,/\log\left(N(s)+1\right)$.

Maximum Entropy for Tree Search (MENTS)

Maximum Entropy for Tree Search (MENTS) applies maximum entropy policy optimization to MCTS

1. Use E2W as tree policy

$$\pi_t(a|s) = (1 - \lambda_s)\, \boldsymbol{f}_\tau\left(\boldsymbol{Q}_{sft}(s)\right)(a) + \lambda_s \frac{1}{|\mathcal{A}|},$$

   where $\lambda_s = \epsilon\, |\mathcal{A}| \,/\, \log\left(N(s) + 1\right)$.

2. Use *softmax backup* to update the Q-values along the nodes in a trajectory

$$Q_{sft}(s_t, a_t) = \begin{cases} r(s_t, a_t) + R & t = T - 1 \\ r(s_t, a_t) + \mathcal{F}_\tau\left(\boldsymbol{Q}_{sft}(s_{t+1})\right) & t < T - 1 \end{cases}$$

Maximum Entropy for Tree Search (MENTS)

Maximum Entropy for Tree Search (MENTS) applies maximum entropy policy optimization to MCTS

1. Use E2W as tree policy

$$\pi_t(a|s) = (1 - \lambda_s)\, \boldsymbol{f}_\tau(\boldsymbol{Q}_{sft}(s))(a) + \lambda_s \frac{1}{|\mathcal{A}|},$$

   where $\lambda_s = \epsilon\, |\mathcal{A}| \,/\, \log(N(s) + 1)$.

2. Use *softmax backup* to update the Q-values along the nodes in a trajectory

$$Q_{sft}(s_t, a_t) = \begin{cases} r(s_t, a_t) + R & t = T - 1 \\ r(s_t, a_t) + \mathcal{F}_\tau(\boldsymbol{Q}_{sft}(s_{t+1})) & t < T - 1 \end{cases}$$

3. At the end of the iterations, propose $a = \arg\max_a Q_{sft}(s, a)$

Convergence Property

Theorem 5

Let a_t be the action returned by MENTS at iteration t. Then for large enough t with some constant C,

$$P(a_t \neq a*) \leq Ct \exp\left\{-\frac{t}{(\log t)^3}\right\}$$

Convergence Property

Theorem 5

Let a_t be the action returned by MENTS at iteration t. Then for large enough t with some constant C,

$$P(a_t \neq a*) \leq Ct \exp\left\{-\frac{t}{(\log t)^3}\right\}$$

Convergence Property

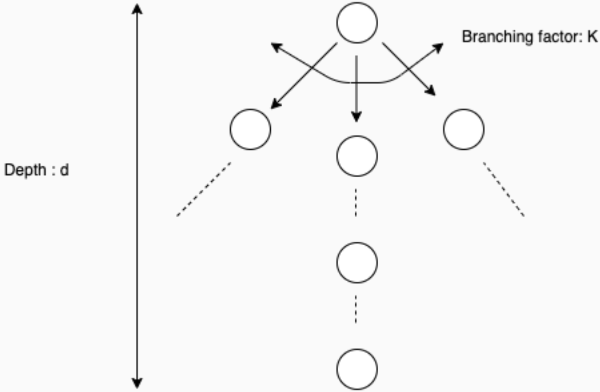
Theorem 5

Let a_t be the action returned by MENTS at iteration t. Then for large enough t with some constant C,

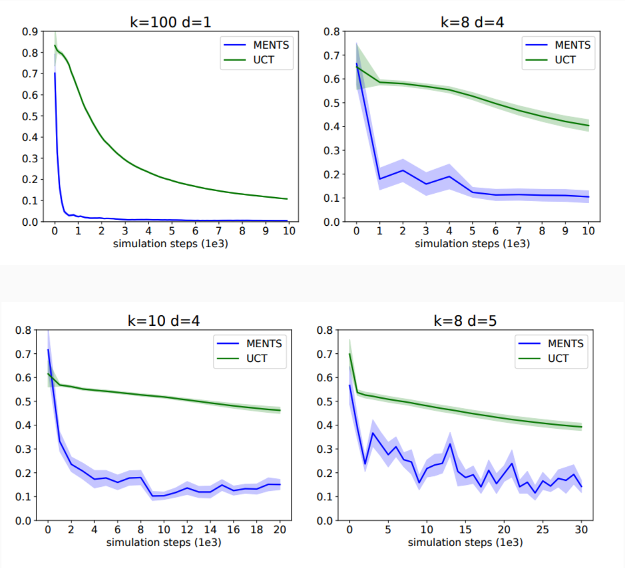
$$P(a_t \neq a*) \leq Ct \exp\left\{-\frac{t}{(\log t)^3}\right\}$$

- MENTS enjoys fundamentally faster convergence rate than UCT
- MENTS applies the E2W as the tree policy during simulations
- Softmax values are back-propagated up the search tree which can be estimated effectively in an optimal rate for each node
- This assures that tree policy converges to the optimal softmax policy π^*_{sft} asymptotically
- Probability of sub-optimal decision at root decays exponentially

Experiments: Synthetic Tree Environment



Experiments: Synthetic Tree Environment



Experiments: Synthetic Tree Environment

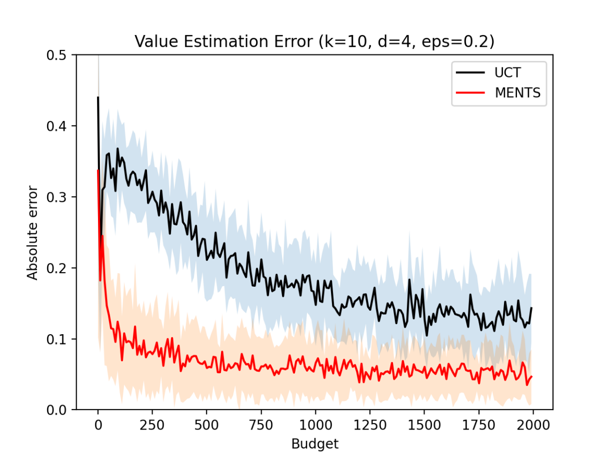


Figure 2: Value estimation error at root with depth = 4 and k = 10

Experiments: CartPole

- Two actions and reward of $+1$ until the pole falls over (end of episode)
- A single neural network to compute:
  - $P(s, a)$: prior probability on action selection
  - $V(s)$: used for leaf node evaluation, instead of MC rollout
- Instead of UCT, used its variant, PUCT:

  $$PUCT(s, a) = Q(s, a) + \epsilon P(s, a) \frac{\sqrt{N(s)}}{1 + N(s, a)},$$

  where P is a prior probability on action selection and $\epsilon > 0$
- In MENTS, prior probability used to initialize $Q_{sft}(s, a)$
- 32 MCTS iteration budget for proposing an action
- Cart can take up to 300 steps in an episode, and total reward calculated from the steps
- Samples from an episode used to train the value/policy network

Experiments: CartPole

- Need good value/policy network to perform well
- But need enough exploration to 'stumble' upon good episodes to learn from

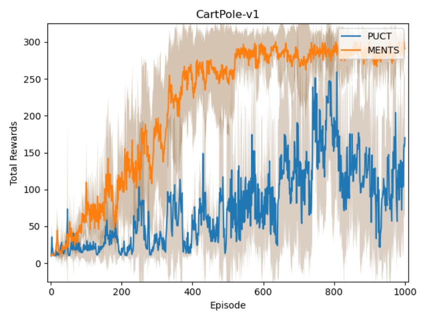


Figure 3: Total reward at each episode. Value / policy network is updated at the end of each episode.

Notebook Implementation

Notebook implementation at: `https://colab.research.google.com/drive/13KhMkjW7NHgFTIrmxGOt1ybG7re9B7L-?usp=sharing`

Conclusion

Summary

- Monte-Carlo value estimates in MCTS do not enjoy effective convergence guarantee when value is back-propagated
- MENTS augments MCTS with maximum entropy policy optimization where softmax values are back-propagated up the search tree
- MENTS enjoys exponential convergence rate to the optimal softmax policy π^*_{sft}, ie, probability of of choosing sub-optimal action at root decays exponentially

Thoughts and Open questions

- MENTS performance in our implementation was very sensitive to changes in the exploration parameter - if not chosen carefully easily degenerates to random policy at each node
- Does MENTS always perform better than UCT in all settings?
- Performance in some Atari experiments not much better than UCT - attributes constraint in simulation budget as a reason

References

Xiao et al. Maximum Entropy Monte-Carlo planning, NeurIPS 2019
Browne et al. A Survey of Monte Carlo Tree Search Methods Convex Regularization in Monte-Carlo Tree Search: https://openreview.net/pdf?id=-kfLEqppEm