Exploration-Exploitation Dilemma and Algorithmic Solutions

Todd W. Neller

Exploration-Exploitation Dilemma

- When actions gain information about what is best, there is a fundamental tension between:
  - Trying something new and possibly discovering a better action (exploration), or
  - Choosing the action believed to be best from previous experience (exploitation)
- Restaurant example: Do I
  - Order something I haven't tried before off the menu and possibly find a new favorite (exploration), or
  - Order my favorite entrée from previous dining (exploitation)?
- How to balance exploring new actions vs. exploiting prior knowledge?

N-Armed Bandit Problem

- “One-armed bandit” – slang for slot machine
- Stochastic outcomes
- N-armed bandit = *n* one-armed bandits, *n* stochastic outcome choices
- Often n Gaussian (normal) distributions with different means and same standard deviations
- Given *t* turns (pulls), how would you balance exploration and exploitation?
- NArmedBanditGame.java demonstration: seed 1234, normal dist., 10 arms, 100 pulls
- **Regret** of not having chosen an action = the difference between the utility of that action and the utility of the action we actually chose

Flat Monte Carlo Selection (Exploration Only)

- For each action a in A, choose a for $t/|A|$ trials and estimate Q(a) as average rewards for a, where t is the total number of trials (actions).

Epsilon-Greedy Selection

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

From section 2.4 of Sutton and Barto, Reinforcement Learning An Introduction, 2nd ed.

UCB1 (Upper Confidence Bounds) Selection

- For each a in A, choose a for 1 trial and initialize Q(a) as experience
- For (t - $|A|$) trials,
  - Let n be the number of the trial iteration $|A| + 1, \ldots, t$, with action having been chosen n_a times.
  - Choose the action a maximizing Q(a) + c $\sqrt{(\ln(n)/n_a)}$ where c = $\sqrt{2}$ (but often higher to encourage more exploration within time constraints)
  - Update Q(a) estimate as average of experiences.

- Downside: Assumes stationary, normal arm distributions. Upside: optimal regret minimization under assumptions

Softmax Selection

- For each *a* in *A*, choose a for 1 trial and initialize Q(*a*) as experience.
- Choose τ temperature parameter
- For (*t* - |*A*|) trials,
  - Choose the action a with probability: $\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$,
  - Update Q(*a*) estimate as average of experiences.

- Note that when $\tau \rightarrow 0$, greedy; when $\tau \rightarrow \infty$, flat; in between, τ varies discerning sampling (like simulated annealing step acceptance! Note similarity!)
- Downside: more complicated. Upside: doesn't assume normal distribution of bandit outcomes.

Gradient Preference Updates

- In 2nd ed., section 2.8, this is expressed with preferences $H_t(a)$:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

- Gradient preference updates:

There is a natural learning algorithm for this setting based on the idea of stochastic gradient ascent. On each step, after selecting action A_t and receiving the reward R_t, the action preferences are updated by:

$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), && \text{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), && \text{for all } a \neq A_t, \end{aligned} \tag{2.12}$$

MCTS Phases per Iteration

- From Sutton and Barto:
  1. **Selection**. Starting at the root node, a tree policy based on the action values attached to the edges of the tree traverses the tree to select a leaf node.
  2. **Expansion**. On some iterations (depending on details of the application), the tree is expanded from the selected leaf node by adding one or more child nodes reached from the selected node via unexplored actions.
  3. **Simulation**. From the selected node, or from one of its newly-added child nodes (if any), simulation of a complete episode is run with actions selected by the rollout policy. The result is a Monte Carlo trial with actions selected first by the tree policy and beyond the tree by the rollout policy.
  4. **Backup**. The return generated by the simulated episode is backed up to update, or to initialize, the action values attached to the edges of the tree traversed by the tree policy in this iteration of MCTS. No values are saved for the states and actions visited by the rollout policy beyond the tree.

Monte Carlo Tree Search (MCTS)

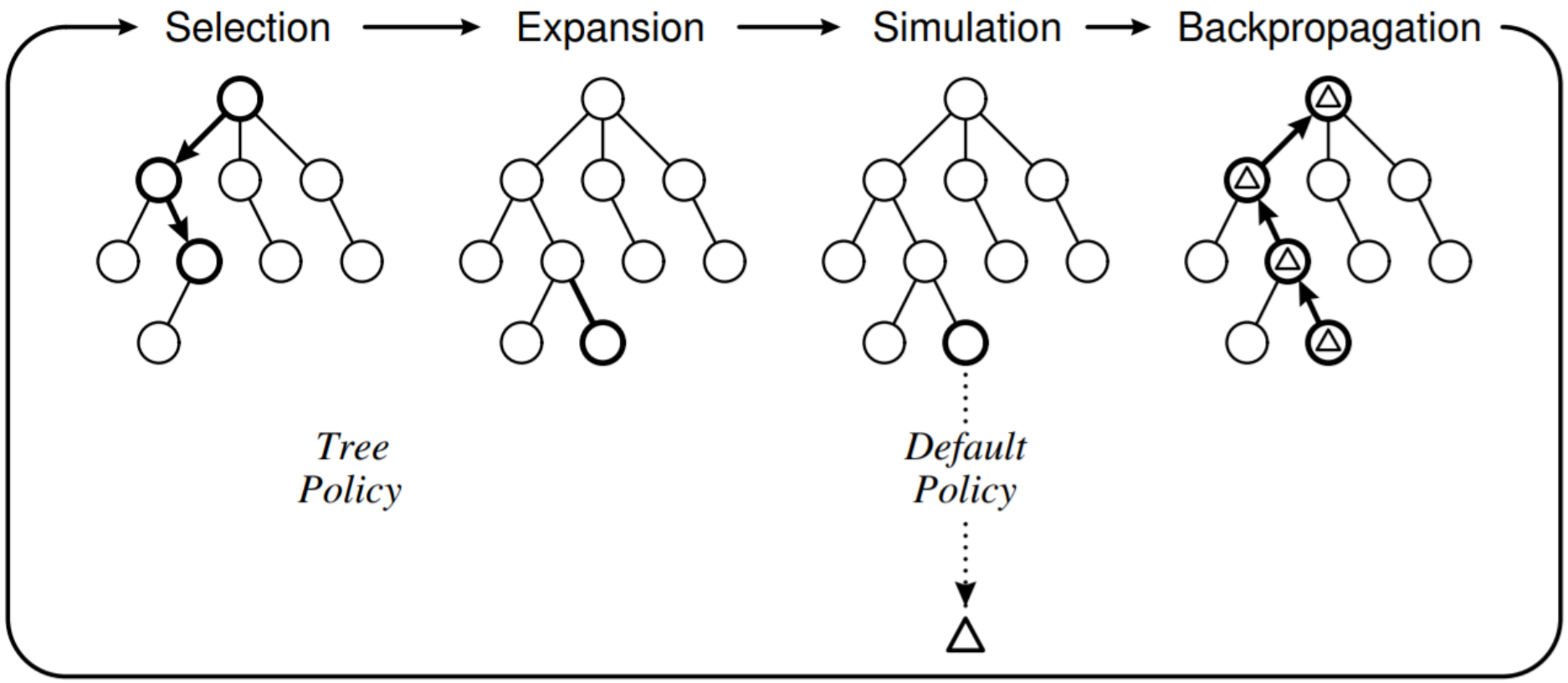

Fig. 2. One iteration of the general MCTS approach.

From Browne et al. A Survey of Monte Carlo Tree Search Methods, 2012.

UCT – “Vanilla” MCTS with UCB1 Selection

- Selection: UCB1 at each node
- Expansion: Add node if untried action
- Simulation: Random playout (uniform random action selection)
- Backup: Update average with terminal node evaluation for each arm chosen

- Criticism:
  - UCB1 assumes a normal distribution. This is violated in most cases, e.g. 1/0/-1 win/draw/loss distributions.
  - UCB1 assumes a stationary distribution. This is violated whenever we find a new, better line of play and the returns from an arm change significantly.
- Thus, there is room for improvement!