Backup plans:

If this Zoom meeting ends prematurely, five-minute break, check Piazza
Other announcements:

Please fill in survey on Canvas
Other announcements:

First quiz: Thu Apr 9 12:00pm
Topic: Uninformed search
24 hour window for submission
Further details out shortly after lecture
Next Karma Lecture

Thursday, at 11:40am
https://cornell.zoom.us/j/276979194

“Efficient Machine Learning via Data Summarization”
Baharan Mirzasoleiman, Stanford University
Multi-Armed Bandit

What strategy do I use to pick a sequence of $a_i$?
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

• Pick the arm with largest UCB($M_i$) instead of $\hat{R}_i$

$$UCB(M_i) = \hat{R}_i + \frac{g(N)}{\sqrt{N_i}}$$

where

$\hat{R}_i$ = average reward for $i$ so far
$N$ = total number of pulls made so far
$N_i$ = total number of pulls of $M_i$ so far

$$g(N) = c\sqrt{\ln N} \quad g(N) = \sqrt{2 \log (1 + N \log^2 N)}$$

$g(N)$ should go up more slowly than $\sqrt{N_i}$
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

- Pick the arm with largest UCB($M_i$) instead of $\hat{R}_i$

$$\text{UCB}(M_i) = \hat{R}_i + \frac{g(N)}{\sqrt{N_i}}$$

where

- $\hat{R}_i$ = average reward for $i$ so far = $\frac{\text{Sum}_i}{N_i}$
- $N$ = total number of pulls made so far
- $N_i$ = total number of pulls of $M_i$ so far

$$g(N) = c\sqrt{\ln N} \quad g(N) = \sqrt{2 \log (1 + N \log^2 N)}$$

$g(N)$ should go up more slowly than $\sqrt{N_i}$
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Algorithm:

For i ← 1 to n { Sum_i ← R(arm_i); N_i ← 1 }; N ← n  /* Initialization */

Loop Forever

best ← argmax \[ \frac{\text{Sum}_i}{N_i} + \frac{g(N)}{\sqrt{N_i}} \] \[1 \leq i \leq n\]

pull arm a_{best} and get reward r

\text{Sum}_{\text{best}} ← \text{Sum}_{\text{best}} + r; \quad N_{\text{best}} ← N_{\text{best}} + 1; \quad N ← N+1
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Algorithm:

For $i \leftarrow 1$ to $n$  
{ $\text{Sum}_i \leftarrow R(\text{arm}_i)$;  $N_i \leftarrow 1$};  $N \leftarrow n$  /* Initialization */

Loop Forever

$\text{best} \leftarrow \text{argmax}_{1 \leq i \leq n} \left[ \frac{\text{Sum}_i}{N_i} + \frac{g(N)}{\sqrt{N_i}} \right]$

pull arm $a_{\text{best}}$ and get reward $r$

$\text{Sum}_{\text{best}} \leftarrow \text{Sum}_{\text{best}} + r$;  $N_{\text{best}} \leftarrow N_{\text{best}} + 1$;  $N \leftarrow N+1$
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Algorithm:

For i ← 1 to n  { Sum_i ← R(arm_i);  N_i ← 1 };  N ← n  /* Initialization */

Loop Forever

best ← argmax \[ \frac{\text{Sum}_i}{N_i} \] \( g(N) \) \( \sqrt{\frac{1}{N_i}} \) \( \hat{R}_i \)

pull arm \( a_{\text{best}} \) and get reward \( r \)

\( \text{Sum}_{\text{best}} \leftarrow \text{Sum}_{\text{best}} + r; \quad N_{\text{best}} \leftarrow N_{\text{best}} + 1; \quad N \leftarrow N + 1 \)
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Algorithm:

Pull each arm once

For i ← 1 to n  
{ Sum$_i$ ← R(arm$_i$);  
N$_i$ ← 1 };  
N ← n  /* Initialization */

Loop Forever

best ← argmax$_{1 \leq i \leq n}$ \[ \frac{\text{Sum}_i}{\text{N}_i} \left( \frac{\text{g}(N)}{\sqrt{\text{N}_i}} \right) \] \[ \hat{R}_i \]

pull arm a$_{\text{best}}$ and get reward r

Sum$_{\text{best}}$ ← Sum$_{\text{best}}$ + r;  
N$_{\text{best}}$ ← N$_{\text{best}}$ + 1;  
N ← N+1
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Reminder: this is a form of MDP
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Reminder: this is a form of MDP

\[ \Pi^{\text{UCB}}(s) = \underset{1 \leq i \leq n}{\text{argmax}} \left[ \frac{\text{Sum}_i}{N_i} + \frac{g(N)}{\sqrt{N_i}} \right] \]
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Instead of cumulative discounted reward, average reward for $N$ steps
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why \( g(N) = \sqrt{2 \log (1 + N \log^2 N)} \) or \( c\sqrt{\ln(N)} \)?

Instead of cumulative discounted reward, average reward for \( N \) steps

Expected average reward for a policy \( \Pi \): \( \mu^\Pi_N = E_\Pi \left[ \frac{\sum_{i=1}^n \text{Sum}_i}{N} \right] \)
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Instead of cumulative discounted reward, average reward for $N$ steps

Expected average reward for a policy $\Pi$: $\mu^\Pi_N = \mathbb{E}_\Pi \left[ \frac{\sum_{i=1}^n \text{Sum}_i}{N} \right]$

Expected average reward for always picking optimal arm: $\mu^{\text{best}} = \max_i R_i$
Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Expected average reward for a policy $\Pi$: $\mu^\Pi_N$

Expected average reward for optimal arm: $\mu^{\text{best}}$

Regret for a policy: $\text{regret}^\Pi_N = N\mu^{\text{best}} - N\mu^\Pi_N$
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Expected average reward for a policy $\Pi$: $\mu^\Pi_N$
Expected average reward for optimal arm: $\mu^{\text{best}}$

Regret for a policy: $\text{regret}_N^\Pi = N\mu^{\text{best}} - N\mu^\Pi_N$

How much exploration costs you
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log(1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Regret for a policy: $\text{regret}_N = N\mu_{\text{best}} - N\mu_N$
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why \( g(N) = \sqrt{2 \log (1 + N \log^2 N)} \) or \( c\sqrt{\ln(N)} \)?

Regret for a policy: \( \text{regret}_N \Pi = N\mu_{\text{best}} - N\mu_{\Pi} \)

Known Result: \( \text{regret}_N \Pi = \Omega(\log(N)) \)
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why \( g(N) = \sqrt{2 \log (1 + N \log^2 N)} \) or \( c\sqrt{\ln(N)} \)？

Regret for a policy: \( \text{regret}_N = N\mu_{\text{best}} - N\mu_N \)

Known Result: \( \text{regret}_N = \Omega(\log(N)) \)

Your expected regret will grow at least logarithmically with \( N \).
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why $g(N) = \sqrt{2 \log (1 + N \log^2 N)}$ or $c\sqrt{\ln(N)}$?

Regret for a policy: $\text{regret}_N = N\mu_{\text{best}} - N\mu_{\Pi}$

Known Result: $\text{regret}_N = \Omega(\log(N))$

$\text{regret}_N^{\text{UCB}(g(N))} = O(\log(N))$

UCB with these $g(N)$ functions has regret that grows at worst logarithmically with $N$.

Your expected regret will grow at least logarithmically with $N$.
Multi-Armed Bandits: Upper Confidence Bound (UCB) Heuristic

Why \( g(N) = \sqrt{2 \log (1 + N \log^2 N)} \) or \( c \sqrt{\ln(N)} \)?

Regret for a policy: \( \text{regret}_N = N \mu_{\text{best}} - N \mu_N \)

Known Result: \( \text{regret}_N = \Omega(\log(N)) \)

\( \text{regret}_N^{\text{UCB}(g(N))} = \mathcal{O}(\log(N)) \)

UCB with these \( g(N) \) functions has regret that grows at worst logarithmically with \( N \)

They are “optimal”

Your expected regret will grow at least logarithmically with \( N \)
Monte Carlo Tree Search (MCTS)
Application of multi-armed bandits
Monte Carlo Tree Search (MCTS)
Application of multi-armed bandits

(Section 5.4)
# Timeline of Key Ideas in Game Tree Search

<table>
<thead>
<tr>
<th>Year</th>
<th>Person</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1948</td>
<td>Alan Turing</td>
<td>Look ahead and use an evaluation function</td>
</tr>
<tr>
<td>1950</td>
<td>Claude Shannon</td>
<td>Game tree search</td>
</tr>
<tr>
<td>1956</td>
<td>John McCarthy</td>
<td>Alpha-beta pruning</td>
</tr>
<tr>
<td>1959</td>
<td>Arthur Samuel</td>
<td>Learn evaluation function (Reinforcement learning)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
1997: Deep Blue defeats Gary Kasparov (3½–2½)

Game tree search with alpha-beta pruning plus lots of enhancements
Go?

Branching factor is in the 100s

Evaluation function is difficult because payoff may be very far away

Needed new ideas
<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Bruce Ballard</td>
<td>Lookahead for probabilistic moves</td>
</tr>
<tr>
<td>1987</td>
<td>Bruce Abramson</td>
<td>Evaluation by expected outcome (repeated simulation)</td>
</tr>
<tr>
<td>1992</td>
<td>Gerald Tesauro</td>
<td>TD-Gammon (reinforcement learning, self-play)</td>
</tr>
<tr>
<td>1992</td>
<td>Bernd Brugmann</td>
<td>Monte Carlo Go (simulated annealing)</td>
</tr>
<tr>
<td>1999</td>
<td>U of Alberta</td>
<td>Simulation in Poker</td>
</tr>
<tr>
<td>1999</td>
<td>Matt Ginsberg</td>
<td>Simulation in Bridge</td>
</tr>
<tr>
<td>2002</td>
<td>Brian Sheppard</td>
<td>Simulation in Scrabble</td>
</tr>
<tr>
<td>2006</td>
<td>Levente Kocsis and Csaba Szepesvár</td>
<td>Multi-armed bandits for Monte-Carlo tree search</td>
</tr>
</tbody>
</table>
2016: AlphaGo defeats Lee Sedol (4-1)
Key ideas of Monte Carlo Tree Search:

1. View move selection as a multi-armed bandit problem
Multi-Armed Bandit

\[
\begin{align*}
R_1 & \quad R_2 & \quad R_3 & \quad R_4 & \quad \ldots & \quad R_n \\
M_1 & \quad M_2 & \quad M_3 & \quad M_4 & \quad \ldots & \quad M_n
\end{align*}
\]
Multi-Armed Bandit for Game Tree Search

\[ a_1, a_2, a_3, a_4, \ldots, a_n \]

\[ M_1, M_2, M_3, M_4, \ldots, M_n \]

\[ R_1, R_2, R_3, R_4, R_n \]
Multi-Armed Bandit for Game Tree Search

What move should I try?
Key ideas of Monte Carlo Tree Search:

1. View move selection as a multi-armed bandit problem
2. Evaluate moves by simulating games
Multi-Armed Bandit for Game Tree Search

What move should I try on each simulated game?