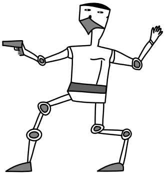
RL Virtual School

Multi-Armed Bandits



Monte Carlo Tree Search

Tor Lattimore

DeepMind

Menu

Part 1: Bandits (10am – noon)

- Finite-armed bandits
- Exploration/exploitation dilemma
- Optimism in the face of uncertainty
- Demonstration
- Contextual bandits

Part 2: Monte-Carlo tree search (2pm – 4pm)

- Monte Carlo tree search
- Practical

Please ask questions and I or one of the TA’s will try to answer!

Bandit problems

- Reinforcement learning without (controlled) state
- Simplicity buys depth and practicality
- Many many (potential) applications
  - A/B testing
  - On-line advertising
  - Education delivery
  - Clinical trials
  - Tree search (see this afternoon)
  - Dynamic pricing
  - Network routing
  - Ranking
- Very active research topic

Interaction protocol

Bernoulli bandit model

- **Finite horizon:** Interaction lasts n rounds
- **Finitely many actions:** k actions
- **Binary rewards:** The reward when playing action a in round t is $R_{t,a}$ and has a Bernoulli distribution with **unknown** mean $\mu_a \in [0, 1]$
- **Optimal action:** $a^* = \operatorname{argmax}_a \mu_a$
- **Mean payoff of optimal action:** $\mu^* = \max_a \mu_a$

Demo

A/B TESTING

EXPLORATION/EXPLOITATION DILEMMA

- **Two arms** $k = 2$
- **Small horizon** $n = 10$
- You have played action one 5 times and received 3 wins
- You have played action two 2 times and received 1 win
- Looks like action one is better, but maybe not. Should you explore (play action two) or exploit (play action one)?

Optimism in the face of uncertainty

Act *as if the* ***world*** *were as nice as* ***plausibly possible***

- In bandits, the only unknown is the mean reward for each action
- The **nicest plausible** bandit would have the largest means **consistent** with the data
- To make this formal we need to make rigorous the notions **plausible/consistent**

A diversion on estimation (CLT)

A diversion on estimation (Hoeffding's)

The Upper Confidence Bound Algorithm

```
# play each action once
for a in range(bandit.arms()):
  bandit.play(a)

# iterate over remaining rounds
while bandit.rounds() < n:
  # compute vector of indices
  index = bandit.mean() + sqrt(0.5 / bandit.T() * log(bandit.rounds()))

  # find maximising index
  a = argmax(index)

  # play arm with largest index
  bandit.play(a)
```

$$A_t = \operatorname{argmax}_a \hat{\mu}_a(t-1) + \sqrt{\frac{\log(t)}{2T_a(t-1)}}$$

Regret

Regret bounds for UCB

Regret bounds for UCB

Regret bounds for UCB

Regret bounds for UCB

Regret bounds for UCB

Demo

QUIZ

Consider a bandit with means $(1/2, 1/2 - \Delta)$. What do you think the regret will look like as a function of $\Delta \in [0, 1/2]$?

(a) The regret will increase as Δ gets large

(b) The regret will decrease as Δ gets large

(c) The regret will increase and then decrease as a function of Δ

(d) The regret will decrease and then increase as a function of Δ

Demo

What is going on?

Minimax bounds

Minimax bounds

Minimax bounds

THOMPSON SAMPLING

- A Bayesian approach
- Prior on the unknown mean of each arm
- Convenient choice for Bernoulli bandits is a $\mathrm{B}(\alpha, \beta)$ prior
- Posterior for mean of arm a in round t is

$$B_a(t-1) = \mathrm{B}\left(\alpha + \underbrace{\sum_{s=1}^{t-1} \mathbf{1}_{A_t=a} R_t}_{\text{number of wins from } a}, \beta + \underbrace{\sum_{s=1}^{t-1} \mathbf{1}_{A_t=a}(1-R_t)}_{\text{number of losses from } a}\right)$$

- TS samples $\tilde{\mu}_a(t-1)$ from $B_a(t-1)$
- Plays $A_t = \operatorname{argmax}_a \tilde{\mu}_a(t-1)$

Thompson sampling in pictures

Demo

BAYESIAN OPTIMALITY

- Thompson sampling uses the posterior to quantify uncertainty and introduce exploration
- **Bayesian optimal** policy maximises

$$\mathbb{E}\left[\sum_{t=1}^{n} R_{t,A_t}\right]$$

- Believed to be computationally hopeless but...
- When $\gamma \in (0,1)$ the policy maximising

$$\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R_{t,A_t}\right]$$

  can be approximated in polynomial time

- Keyword is **Gittins' index**
- **Beautiful, empirically superb**, **hard to analyse and a little brittle**

Adversarial bandits

- All models are wrong
- What if rewards are not really random?
- Remarkably, you can relax **all** assumptions on the data!
- Let $(R_t)_{t=1}^n$ be an arbitrary sequence of reward vectors, $R_t \in [0,1]^k$
- Best action in hindsight is $a^* = \operatorname{argmax}_a \sum_{t=1}^n R_{t,a}$
- Regret is $\mathfrak{R}_n = \mathbb{E}\left[\sum_{t=1}^n R_{t,a^*} - R_{t,A_t}\right]$
- There exists an algorithm such that $\mathfrak{R}_n \leq \sqrt{2nk}$

Contextual bandits – Example

User's arrive at my on-line site sequentially and I want to use a bandit algorithm to choose the centerpiece product to recommend

User information

Age, gender, previous products purchased,...

Product information

Many (similar products)

Contextual bandits

- Let $\mathcal{C}$ be a set of all possible contexts
- In round t, observe $C_t \in \mathcal{C}$
- **Standard idea** design feature mapping $\varphi : [k] \times \mathcal{C} \to \mathbb{R}^d$
- Assume $R_{t,a} = \langle \varphi(a, C_t), \theta \rangle + \eta_t$ for some unknown $\theta \in \mathbb{R}^d$

Contextual bandits

- Let $\mathcal{C}$ be a set of all possible contexts
- In round t, observe $C_t \in \mathcal{C}$
- **Standard idea** design feature mapping $\varphi : [k] \times \mathcal{C} \to \mathbb{R}^d$
- Assume $R_{t,a} = \langle \varphi(a, C_t), \theta \rangle + \eta_t$ for some unknown $\theta \in \mathbb{R}^d$
- **Simplified view** The learner receives $\mathcal{A}_t = \{\varphi(a, C_t) : a \in [k]\} \subset \mathbb{R}^d$
- Plays action $A_t \in \mathcal{A}_t \subset \mathbb{R}^d$
- Reward is $\langle A_t, \theta \rangle + \eta_t$

Least squares

Concentration Bounds

UCB for contextual bandits

Regret analysis

Regret analysis

Regret analysis

WHAT DID I NOT TALK ABOUT (MUCH)?

- Adversarial model
- Fully Bayesian approaches (Gittin's index)
- Combinatorial bandits
- **Practical problems**
  - Non-stationarity
  - Delayed/anonymous rewards
  - Non-linear contextual bandits
  - Constraints
- Off-policy evaluation/optimisation
- Pure exploration
- Partial monitoring
- Scaling up to RL

Monte Carlo Tree Search

Two player zero sum extensive form full information games

- Players make moves alternately until the game ends
- The utility for the second player is the negative of the utility of the first (zero sum)
- No hidden information. The available moves/utility only depends on the moves of the players

Examples , **Connect4**, Checkers, Chess, Go

Non-examples

- Stratego (not full information)
- Poker (not full information and randomised)
- Kalah or Backgammon (randomised)

Game trees

Minimax search

AlphaBeta search

Depth-limited minimax/AlphaBeta

Why not classical search?

Why not classical search?

- They are hard(ish) to make selective
- Especially in games with many actions and long-term consequences (e.g., Go)
- ~ 400 moves in Go compared to ~ 30 in Chess
- Historically, MCTS was less effective in ‘tactical’ games like Chess
- MCTS and AlphaBeta seem about equally good in chess now

Monte Carlo Tree Search

- Builds a selective tree
- Simple to implement
- Simple to parallelise (AB is hard to parallelise)
- Simple to incorporate knowledge (or ML)

Monte Carlo Tree Search in pictures

Monte Carlo Tree Search pseudocode

```
# start with root an empty tree
while time_left():
  # traverse the current tree to a leaf
  node = find_leaf(root)
  # expand one child of that leaf
  node = expand_leaf()
  # compute a rollout until the end of the game
  result = rollout(node)
  # update the path back to the root
  while (node != root):
    node.update(result)
    node = node.parent()
return select_move(root)
```

Five components

- Finding a leaf from the root
- Expanding a leaf
- Computing a rollout
- Updating nodes on the path back to the root
- Choosing a move from the root node

Connect4

Connect4

- Connect4 is a forced win for the first player
- Solved in 1988
- Solved by a modern alpha-beta search in less than a second now
- Illustrates some of the benefits of MCTS

Theory

Extensions and modifications

Combining MCTS with ML

Thanks!