Multi-armed Bandit Problems

Bruno Castro da Silva

Computer Science Department
University of Massachusetts at Amherst

August, 2008

Summary

- Introduction
- Classical formulation
- Properties
- Computational issues
- Extensions
- Example
- Discussion

Definition

- Multi-armed bandit (MAB) problems
  - sequential resource allocation
  - among competing (mutually exclusive) projects
- Difficulty related to conflict between
  - allocating resources that yield **good rewards**
  - trying "**not so promising**" projects
    - but maybe with better future prospects

Examples

- control theory problems
- allocating researchers to projects
- clinical trials
- sensor management

Definition

- Classical definition
  - single resource
  - allocated to one of many competing projects (bandits, arms)
  - project w/ resource can change its state
  - other projects remain frozen
  - discrete time, no switching costs

Solving

- In general this is solvable via Dynamic Programming
  - ***backwards induction***
  - $V^*(s, N) = R_N(s), \quad \forall s$
  - $V^*(s, N-1) =$
    $\max R_{N-1}(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s', N)$
  - Bellman equations
  - $V^*(s) = \max_a R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s')$
  - very general stochastic optimization problems
  - VI, PI, RL
  - Curse of dimensionality

Solving

- But MAB are simpler and allow for "index-type" solutions
  - for each bandit associate a dynamic allocation index (DAI)
  - depends only on that bandit
    - one k-armed bandit vs k single-armed bandits
  - at each time, choose the bandit with highest DAI
  - leads to optimal allocation policy
  - DAIs are also known as "Gittins Indices"

Classical formulation

- (single-armed) bandit process
  - described by pair random of sequences
    - $\{X(0), X(1), \ldots\}$
    - $\{R(X(0)), R(X(1)), \ldots\}$
  - $X(n)$: state after arm has been operated n times;
  - $R(X(n))$: reward obtained on the n-th operation
  - state evolution:
    $X(n) = f_{n-1}(X(0), \ldots, X(n-1), W(n-1))$
  - thus, arm not necessary Markov
  - $W(n)$: independent sequence of RVs; independent also from $X(0)$

Classical formulation

- multi-armed bandit process
  - k independent arms
  - one controller
  - controller operates exactly *one* arm at a time
  - machines described by time-dependent sequences:
    - $\{X_i(N_i(t)), R_i(X_i(N_i(t)))\} \quad \forall i \forall t$
    - $N_i(t)$: number of times machine i has been operated up to time t
    - $N_i(t)$ is the "local time" of machine i
  - control is $U(t) = \{U_1(t), \ldots U_k(t)\}$, ie, in the form $\{00 \ldots 1 \ldots 000\}$

Classical formulation

- System evolution
- $X_i(N_i(t+1)) =$
  - $f_{N_i(t)}(X_i(0), \ldots, X_i(N_i(t)), W_i(N_i(t)))$ if $U_i(t) = 1$
  - $X_i(N_i(t))$ if $U_i(t) = 0$
- $N_i(t+1) =$
  - $N_i(t) + 1$ if $U_i(t) = 1$
  - $N_i(t)$ if $U_i(t) = 0$
- $R_i(t) = R_i(X(N_i(t)), U_i(t)) =$
  - $R_i(X_i(N_i(t)))$ if $U_i(t) = 1$
  - 0 if $U_i(t) = 0$

Classical formulation

- Scheduling policy
- $\gamma = (\gamma_1, \gamma_2, \ldots)$
- such that $U(t) = \gamma_t(Z_1(t), \ldots Z_k(t), U(0), \ldots, U(t-1))$
- and $Z_i(t) = \{X_i(0), \ldots X_i(N_i(t))\}$
- In other words, policy might depend on full history of arms' states and previous actions

Classical formulation

- Goal is to find scheduling policy γ that maximizes

$$J^{\gamma} = E\Big(\sum_{t=0}^{\infty} \beta^t \sum_{i=1}^{k} R_i(X_i(N_i(t)), U_i(t)) \quad \Big| \quad Z(0) \Big)$$

Forward induction

- simplest policy: myopic decisions (1 look-ahead)
- not optimal, in general
- T-steps look-ahead
  - take decisions that maximize expected reward for the next T steps
- Generalization: do not fix T
  - let τ be the number of look-ahead steps
  - τ is a RV that depends at each time on how the system evolves
  - τ is considered a *stopping time*

Forward induction

- in order to maximize J^γ, we must
  - **choose a rule γ for taking a sequence of decisions**
  - **choose a value for τ**
- such that that rule, when used for τ steps, gives the max J^γ
- This extension of T-steps look-ahead works by
  - At $t = 0$, given $Z(0)$, select γ_1 and τ_1
  - Apply γ_1 for τ_1 steps
  - repeat, choosing the next γ_t, τ_t, conditioned on the new information gained
    - notice: decisions based only on current states of arms
    - “forward” because keeps deciding next policies for the future

Forward induction

- in general this is not optimal
  - route choosing example
  - problem are irrevocable decisions
  - some alternatives available at some stage are not available later
- if any decisions made are not irrevocable, forward induction is optimal
  - every arm not used is kept frozen
  - thus can deliver the same sequence of rewards later on (up to β)

Forward induction

- Gittins proved that the following index is optimal

$$v_{X_i}(x_i(0)) = \max_{\tau>0} \frac{E\Big(\sum_{t=0}^{\tau-1} \beta^t R_i(X_i(t)) \quad | \quad x_i(0)\Big)}{E\Big(\sum_{t=0}^{\tau-1} \beta^t \quad | \quad x_i(0)\Big)}$$

- suppose we are allowed to take decisions only while they're worth it,
  - then v_{X_i} gives a "retirement" value
    - ie, a value in which we are indifferent to continuing operating i or quitting
  - only quit i (and work on some j) if j has a better prospect than the retirement offered

Forward induction

- When in decision stage l, for each arm i,
- and considering information $x_i^l(\omega) = (x_i(0), \ldots, x_i(N_i(\tau_l(\omega))))$,

$$v_{X_i}(x_i^l(\omega)) = \max_{\tau > \tau_l(\omega)} \frac{E\Big(\sum_{t=\tau_l(\omega)}^{\tau-1} \beta^t R_i(X_i(N_i(\tau_l) + t - \tau_l(\omega))) \quad | \quad x_i^l(\omega) \Big)}{E\Big(\sum_{t=\tau_l(\omega)}^{\tau-1} \beta^t \quad | \quad x_i^l(\omega) \Big)}$$

- easier if arm is Markov

Computational issues

- Focus on Markov arms
- State space $S_i = \{1, 2, \ldots, \Delta_i\}$

$$v_{X_i}(x_i(t)) = \max_{\tau > t} \frac{E\Big(\sum_{t'=t}^{\tau-1} \beta^t R_i(X_i(t')) \quad | \quad x_i(t) \Big)}{E\Big(\sum_{t'=t}^{\tau-1} \beta^t \quad | \quad x_i(t) \Big)}$$

- Need to compute v for each state of each arm

Computational issues

- Offline approach: compute indices for all states, all machines
- Online approach: only index for the last used machine

- Continuation/stopping sets
  - remember, v is retirement value
  - only quit machine i if reach state from which j would be better
  - $C_i(x_i)$: all states with index higher than x_i's
  - $S_i(x_i)$: all states with index lower than x_i's

Offline calculation

- Computing $C_i(x_i)$ and $S_i(x_i)$
  - ordering on states: $l_1, l_2, \ldots, l_{\Delta_i}$ s.t.
  - $v_{X_i}(l_1) \geq v_{X_i}(l_2) \geq \ldots \geq v_{X_i}(l_{\Delta_i})$
- For machine i, set $l_1 = \arg\max_{x_i} R_i(x_i)$
  - Now consider probabilities in P^i only for transitioning to "better" states;
  - Given reward matrix R_i (reward per state);
  - For each state x_i, calculate $D^{i,n}_{x_i}$
    - expected discounted reward considering next (better) states
  - Calculate $B^{i,n}$
    - expected total "discounts", considering probabilities of transitions
  - $v_{X_i}(x_i) = \frac{D^{i,n}_{x_i}}{B^{i,n}_{x_i}}$

Online calculation

- Also uses the continuation/stopping sets approach
- Assume we are operating machine i in state a
  - now, we are given opportunity to switch to state x_i
  - maximize expected discounted reward over infinite horizon

$$V(a) = \max\left\{R_i(a)+\beta \sum_{b\in\{1,\ldots,\Delta_i\}^i} P^i_{a,b} V(b),\ R(x_i)+\beta \sum_{b\in\{1,\ldots,\Delta_i\}} P^i_{x_i,b} V(b)\right\}$$

Online calculation

- Now $C_i(x_i)$ is the set of states with expected reward larger than $V(x_i)$;
- $v_{X_i}(x_i) = (1 - \beta)V(x_i)$
- Questions:
  - Why maximize infinite horizon is equivalent?
  - Why $(1 - \beta)$ and not $\frac{1-\beta}{\beta}$?

Superprocesses

- Same as before, but now each arm i receives control input $U_i \in \{0, \ldots, M_i\}$
- $U_i = 0$ is a freezing action; rest are continuation actions
- If control policies are fixed, degenerates to regular MAB
- Otherwise, state evolution are rewards depend on current state **and** on current control input
  - Not a Markov Chain, but a Markov Process
- Scheduling policy γ controls exactly one machine

$$J^\gamma = E^\gamma\Big(\sum_{t=0}^{\infty} \beta^t \sum_{j=1}^{k} R_j(X_j(N_j(t)), U_j(t)) \quad | \quad Z(0)\Big)$$

Superprocesses

- Time evolution of arm **is** controlled
- More complex than MAB; in general, Gittins Indices not optimal
- Unless each arm (desc. by seq. X states, rewards) has a **dominating** arm

$$L(X,\mu) = max_{\tau>0}\Big(\sum_{t=0}^{\tau-1}\beta^t[R(X(t)) - \mu]\Big)$$

- X dominates Y iff
  - $L(X,\mu) \geq L(Y,\mu) \qquad \forall\mu \in R$
- μ is “retirement” value; $L(X,\mu)$ the expected gain over μ

Superprocesses

- If there is dominance, optimal because
  - No matter how big the offered retirement is (to quit i), there's always a better arm j
- In practice, this is a quite restrictive assumption

Arm-acquiring bandits

- Regular MAB, but new arms can be created
- Gittins Indices **are** optimal
  - Decisions are **not** irrevocable
  - Decisions based on indices with K_i arms consider all of them
  - But decisions prior to this *did not* have all K_i arms available;
    - no way a prior decision could be "wrong"

Switching penalties

- Regular MAB, but there is a cost c for switching arms
- Gittins Indices are **not** optimal (example in book)
- If the index is

$$v^s_{X_i}(x_j(0)) = \max_{\tau>0} \frac{E\left(\sum_{t=0}^{\tau-1} \beta^t R_j(t) - c \quad | \quad x_j(0)\right)}{E\left(\sum_{t=0}^{\tau-1} \beta^t \quad | \quad x_j(0)\right)}$$

- then only qualitative results are known [11]
- the general nature of the scheduling policies is not known
- solution usually requires full use of DP (backwards induction)

Multiple plays

- Regular *k*-processes MAB, but is *m* processors
- At each time allocate each processor to exactly one process
- No process being operated by more than one processor
- Only processes being processed generate reward
- Allocation according to *m* highest indices: **not optimal**
- Optimal if indices are sufficiently separated (C1, p.141)
  - How to guarantee this beforehand?
- For different criteria (eg: regret minimization) optimal policies are known [7,8]

Restless bandits

- k machines, m processors
- machines' states evolve over time even when not being processed
- reward of non-processed machines might be assumed to be zero
- performance criterion is

$$J^{\gamma} = E^{\gamma}\Big(\sum_{t=0}^{\infty} \beta^{t} \sum_{j=1}^{k} R_j(X_j(N_j(t)), U_j(t)) \quad | \quad Z(0) \Big)$$

- Goal is to find policy that maximizes infinite horizon expected discounted reward

Restless bandits

- In general, Gittins Indices are not optimal
- But for some other optimization criterion, indices are optimal
  - eg: infinite horizon average reward-per-time-per-machine criterion

$$\frac{1}{k}\left(\lim_{T\to\infty}\frac{1}{T}E\Big(\sum_{t=1}^{T}\sum_{i=1}^{k}R_i(X_i(t-1),U_i(t))\Big)\right)$$

Restless bandits

- Gittins indices for RB are related to “gift” values given to non-processed machines
- Argument is similar to that of the “retirement” value
  - index is a “gift” value that makes us indifferent to running or not the machine
  - it is only worth to run the machine if the expected gain is greater than the “gift” value
  - this values allow us to index all machines

Example

- find one stationary target hidden in one of k cells
- prior probability of the target in cell i is $p_i(0)$
- sensor can look into just one cell at a time
- sensor is imperfect
  - $P(\text{sensor finds target in } i \mid \text{ target is in cell } j) = \delta_{i,j} q_j$
  - where δ is the Kronecker delta function;
  - q_j (?) is probability of false positive
- reward upon completion is β^t (ie, we want to find the target ASAP)
- which sensor to activate at each time?

Example

- let $p_i(t)$ be the posterior probability of target being in cell i
- $p_i(t)$ is state of cell (arm) i at time t

Example

- For a policy γ, expected reward is

$$
\begin{aligned}
&= \sum_{t=0}^{\infty} \beta^{\tau} P(\text{target is found at } \tau \text{, analyse correct cell}) \\
&= \sum_{t=0}^{\infty} \beta^{\tau} \sum_{i=1}^{k} p_i(t) q_i P^{\gamma}(U(t) = e_i) \\
&= \sum_{t=0}^{\infty} \beta^{\tau} \sum_{i=1}^{k} R_i(p_i(t), u_i(t))
\end{aligned}
$$

- where reward is given for i iff i is activated at t ($U(t) = i$)

Example

- Unfortunately, updates in p_i affect all other probabilities (states)
- Thus, not a regular MAB
- Easy to solve if we consider unnormalized probabilities

$$
\begin{aligned}
p_i(t+1) & \\
&= p_i(t) & \text{if } u_i(t) = 0 \\
&= p_i(t)(1 - q_i) & \text{if } u_i(t) = 1
\end{aligned}
$$

Example

- We try to maximize the long-term expected reward
  - remember, $R_i(p_i(t), u_i(t)) = p_i(t)q_i$ iff $u_i(t) = 1$, zero otherwise

$$\sum_{t=0}^{\infty} \beta^t \sum_{i=1}^{k} R_i(p_i(t), u_i(t))$$

- Gittins Index of every machine is always achieved at $\tau = 1$ (?), so:
  - $v_{X_i}(p_i(t)) = p_i(t)q_i$
  - which is by the definition of GI, for one-step look-ahead
  - β can be ignored from the denominator because it is constant

Example

- If sensor operates in M modes: superprocess
- If there is cost to switch targetting area: MAB w/ switching penalties
- If there are m sensors: MAB w/ multiple plays
- If target is moving: m sensors, restless bandit

Conclusion

- Gittins indices simplify the policy calculation for a class of sequential decision problems
- MAB are very simple problems, but might be extended
  - extensions are often related with one another
  - arm-acquiring → superprocess [240]
  - switching costs → restless bandits [91]
  - Tax problem (minimization of cost of frozen machines) → MAB

Thanks

Questions?

bsilva@cs.umass.edu