Online Learning under Delayed Feedback

Pooria Joulani, András György, Csaba Szepesvári

In ICML 2013

3 January, 2014

Outline

Up Front

- Focusing on their algorithm and regret bound for stochastic (non-adversarial) setting.
- Their bound is interesting; they show a meta-algorithm which lets you create parallel algorithms which are asymptotically no worse than the sequential algorithm.
- Their algorithm appears dissatisfying for application.

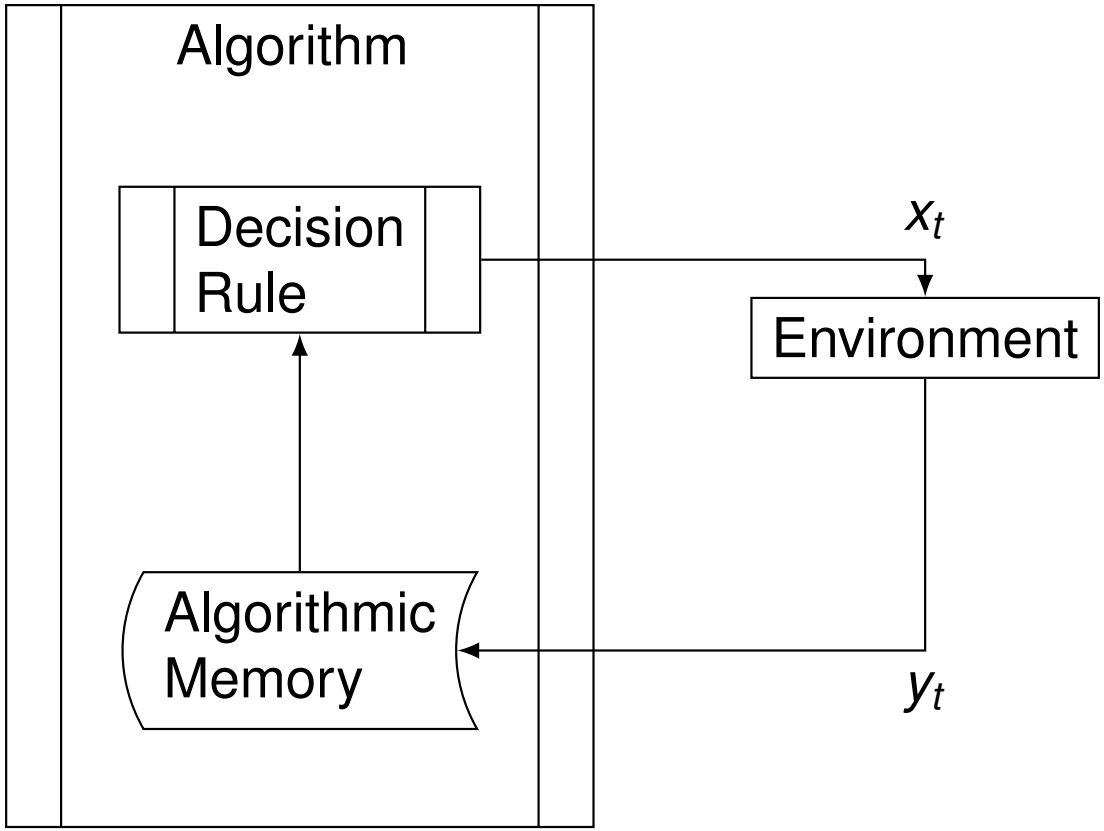
Bandit Algorithms

- The bandit setting describes an experimental agent interacting with its environment.
  - The algorithm is attempting to maximize its rewards $f(x_t)$. f: unknown reward function, x_t: action selected in round t.
  - At each round, the algorithm selects x_t from a decision set D and observes a noisy version of the corresponding reward, $y_t = f(x_t) + \epsilon_t$ (e.g., ϵ_t are Gaussian white noise).
  - No other observations are available; the algorithm must then make action selections to both *exploit* its current knowledge of the reward function to earn high reward and *explore* the reward function further to help itself make good decisions in future rounds.

Regret

- Often discuss regret $r_t = f(x^*) - f(x_t)$, the difference between the reward of an optimal fixed action x^* and that of the action we chose.
- Bounding the growth of the cumulative regret $R_T = \sum_{t=1}^{T} r_t$ tells us about the performance of the algorithm.
- Ideally, we would like R_T to be strictly sub-linear, which (roughly speaking) implies $r_t \to 0$.

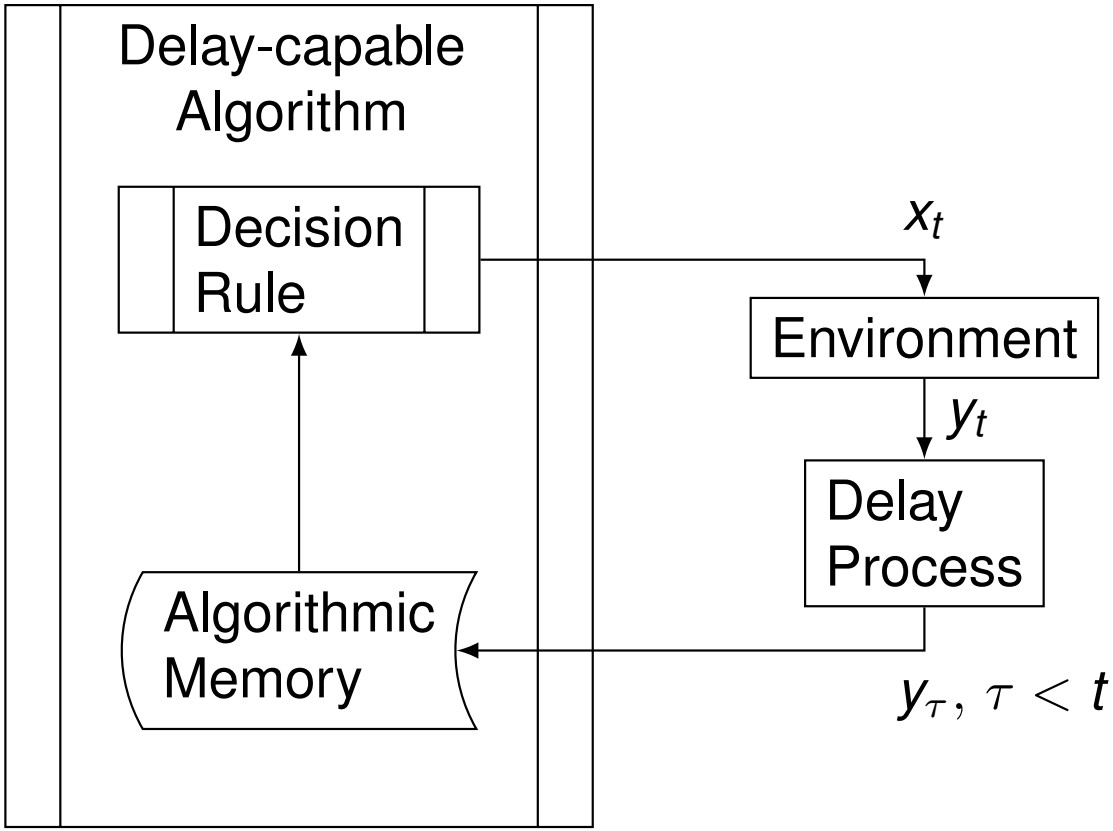
Diagramatically



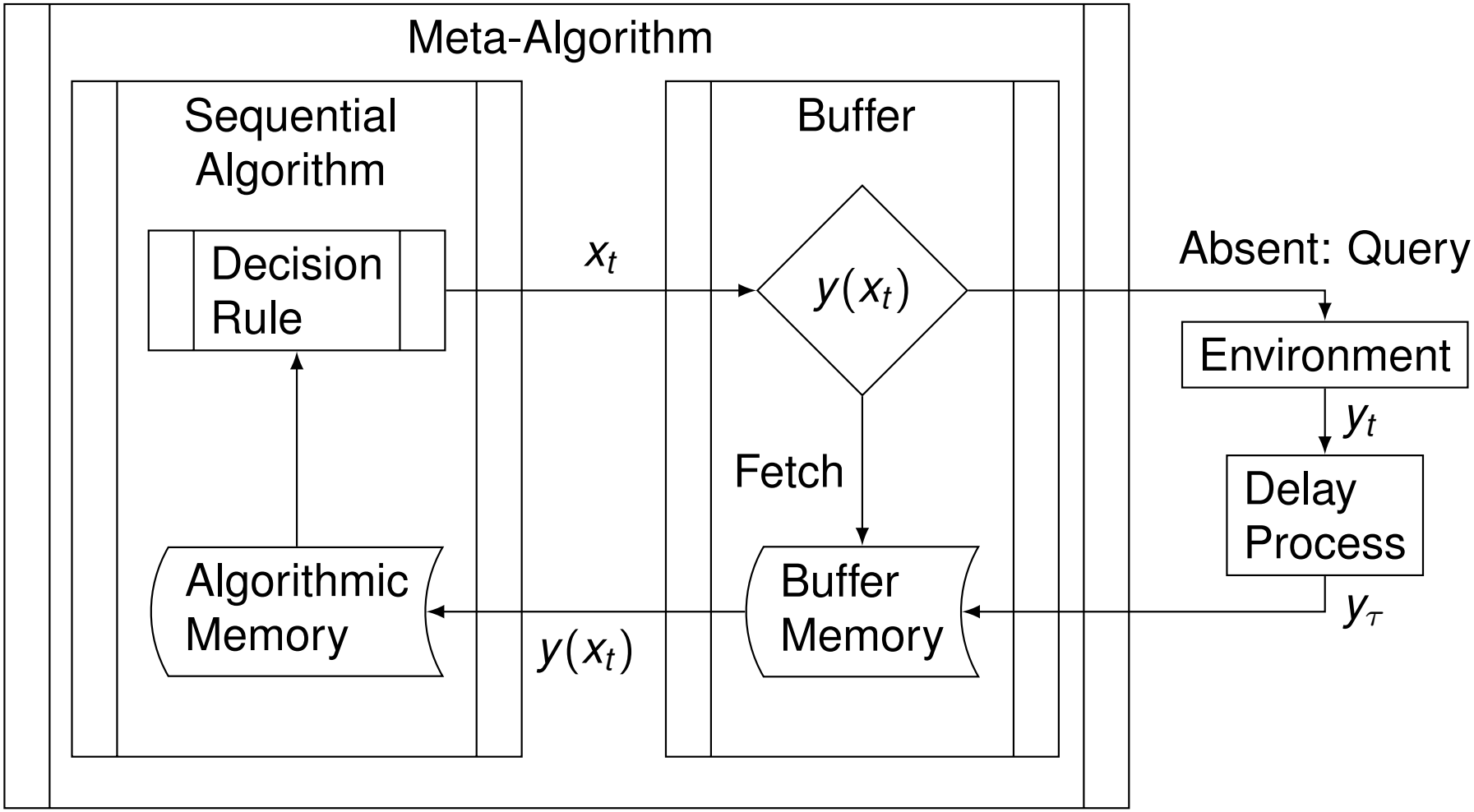
Dealing with Delays

- What if we have a delay in our system, i.e., if y_t may not be available before we have to select x_{t+1}?
- Lots of approaches have been tried.
  - My horse in the race: UCB algorithms which pretend they have received the observations they asked for.
    - This reduces redundancy, but makes life complicated.
  - Joulani et al. propose introducing a buffer between the sequential algorithm and the environment, such that the sequential algorithm doesn't "see" the delay.
    - Handily, this is a general strategy with easy (and quite strong) proofs.

Delays


Delay-capable Algorithm
Decision Rule
Algorithmic Memory
x_t
Environment
y_t
Delay Process
$y_\tau, \tau < t$

Meta-Algorithm of Joulani et al.



Key Points

- Sequential algorithm only ever interacts with the buffer.
- At each time step of sequential algorithm, it asks the buffer for an observation.
  - Present: fetch from the buffer and continue to the next internal time step. *No elapsed external time*
  - Absent: query the environment and *wait*.

Key Points 2

- Crucially, this means queries to the environment (i.e., real-world elapsed time) only occur when the buffer is empty.
- For lack of anything better to do, the meta-algorithm repeatedly submits the same query at every intervening real-world timestep until the buffer is non-empty.
- The sequential algorithm only sees the *one* query it has been waiting for, until it requests another one, at which point it should be present in the buffer.
- It is thus buffered from the delay, and only ever acts as a fully-sequential algorithm.

Regret Bound Sketch

- At any time, the union of the observations in the sequential algorithm's memory and those in the buffer is the set of all observations (rewards) obtained.
- If we know the size of the decision set D and a bound on the maximal delay τ_{max}, then the buffer contains at most $D\tau_{max}$ observations, and so the regret of the meta-algorithm is at most the regret of the sequential algorithm, plus $2||f||_\infty D\tau_{max}$.

Analysis

- This is a neat result; we've shown that what we pay for delay is at most some constant additive cost beyond the sequential algorithm's regret.
- Thus, we can potentially design algorithms which have no worse asymptotic scaling than the fully sequential algorithm, despite dealing with some fixed delay.

Practical Problems and Possible Improvements

- The buffer sweeps too much under the rug.
- The buffer cost is non-negligible for large $|D|$.
- The query procedure is too slow; since we don't switch until we've obtained at least one observation, we often submit far more queries than we are likely to need.
- One possibility for both of these: provide the algorithm with a finite, pre-constructed initialization.
- Observations in the buffer are ignored for decision-making. Wasteful when we need good results fast.
- Generalization to a temporally non-stationary f may be problematic: the buffer may be very "stale."

Summary

- Joulani et al. develop a simple meta-algorithm for adapting standard bandit algorithms to the delay setting.
- This algorithm suffers only additive regret vs. the sequential algorithms it “wraps" around.
- I have some reservations about the practicality of this algorithm.