Pricing American Options with Reinforcement Learning

Ashwin Rao

ICME, Stanford University

February 21, 2020

Overview

Stopping Time

- Stopping time τ is a "random time" (random variable) interpreted as time at which a given stochastic process exhibits certain behavior
- Stopping time often defined by a "stopping policy" to decide whether to continue/stop a process based on present position and past events
- Random variable τ such that $Pr[\tau \leq t]$ is in σ-algebra $\mathcal{F}_t$, for all t
- Deciding whether $\tau \leq t$ only depends on information up to time t
- Hitting time of a Borel set A for a process X_t is the first time X_t takes a value within the set A
- Hitting time is an example of stopping time. Formally,

$$T_{X,A} = \min\{t \in \mathbb{R} | X_t \in A\}$$

eg: Hitting time of a process to exceed a certain fixed level

Optimal Stopping Problem

- Optimal Stopping problem for Stochastic Process X_t:

$$W(x) = \max_{\tau} \mathbb{E}[H(X_\tau)|X_0 = x]$$

  where τ is a set of stopping times of X_t, $W(\cdot)$ is called the Value function, and H is the Reward function.
- Note that sometimes we can have several stopping times that maximize $\mathbb{E}[H(X_\tau)]$ and we say that the optimal stopping time is the smallest stopping time achieving the maximum value.
- Example of Optimal Stopping: Optimal Exercise of American Options
  - X_t is risk-neutral process for underlying security's price
  - x is underlying security's current price
  - τ is set of exercise times corresponding to various stopping policies
  - $W(\cdot)$ is American option price as function of underlying's current price
  - $H(\cdot)$ is the option payoff function, adjusted for time-discounting

Optimal Stopping Problems as Markov Decision Processes

- We formulate Stopping Time problems as Markov Decision Processes
- *State* is X_t
- *Action* is Boolean: Stop or Continue
- *Reward* always 0, except upon Stopping (when it is $= H(X_\tau)$)
- *State*-transitions governed by the Stochastic Process X_t
- For discrete time steps, the Bellman Optimality Equation is:

$$V^*(X_t) = \max(H(X_t), \mathbb{E}[V^*(X_{t+1})|X_t])$$

- For finite number of time steps, we can do a simple backward induction algorithm from final time step back to time step 0

Mainstream approaches to American Option Pricing

- American Option Pricing is Optimal Stopping, and hence an MDP
- So can be tackled with Dynamic Programming or RL algorithms
- But let us first review the mainstream approaches
- For some American options, just price the European, eg: vanilla call
- When payoff is not path-dependent and state dimension is not large, we can do backward induction on a binomial/trinomial tree/grid
- Otherwise, the standard approach is Longstaff-Schwartz algorithm
- Longstaff-Schwartz algorithm combines 3 ideas:
    - Valuation based on Monte-Carlo simulation
    - Function approximation of continuation value for in-the-money states
    - Backward-recursive determination of early exercise states

Ingredients of Longstaff-Schwartz Algorithm

- m Monte-Carlo paths indexed $i = 0, 1, \ldots, m-1$
- $n+1$ time steps indexed $j = n, n-1, \ldots, 1, 0$ (we move back in time)
- Infinitesimal Risk-free rate at time t_j denoted r_{t_j}
- Simulation paths (based on risk-neutral process) of underlying security prices as input 2-dim array $SP[i, j]$
- At each time step, $CF[i]$ is PV of current+future cashflow for path i
- $s_{i,j}$ denotes state for $(i, j) :=$ (time t_j, price history $SP[i, :(j+1)]$)
- $Payoff(s_{i,j})$ denotes Option Payoff at (i, j)
- $\phi_0(s_{i,j}), \ldots, \phi_{r-1}(s_{i,j})$ represent feature functions (of state $s_{i,j}$)
- $w_0, \ldots, w_{r-1}$ are the regression weights
- Regression function $f(s_{i,j}) = w^T \cdot \phi(s_{i,j}) = \sum_{l=0}^{r-1} w_l \cdot \phi_l(s_{i,j})$
- $f(\cdot)$ is estimate of continuation value for in-the-money states

The Longstaff-Schwartz Algorithm

Algorithm 2.1: LONGSTAFFSCHWARTZ($SP[0:m, 0:n+1]$)

comment: $s_{i,j}$ is shorthand for state at $(i,j) := (t_j, SP[i, :(j+1)])$

$CF[0:m] \leftarrow [Payoff(s_{i,n})$ for i in range$(m)]$

for $j \leftarrow n-1$ **downto** 1

do
- $CF[0:m] \leftarrow CF[0:m] \cdot e^{-r_{t_j}(t_{j+1}-t_j)}$
- $X \leftarrow [\phi(s_{i,j})$ for i in range(m) if $Payoff(s_{i,j}) > 0]$
- $Y \leftarrow [CF[i]$ for i in range(m) if $Payoff(s_{i,j}) > 0]$
- $w \leftarrow (X^T \cdot X)^{-1} \cdot X^T \cdot Y$
- **comment:** Above regression gives estimate of continuation value
- **for** $i \leftarrow 0$ **to** $m-1$
  - **do** $CF[i] \leftarrow Payoff(s_{i,j})$ **if** $Payoff(s_{i,j}) > w^T \cdot \phi(s_{i,j})$

$exercise \leftarrow Payoff(s_{0,0})$

$continue \leftarrow e^{-r_0(t_1-t_0)} \cdot mean(CF[0:m])$

return $(\max(exercise, continue))$

RL as an alternative to Longstaff-Schwartz

- RL is straightforward if we clearly define the MDP
- *State* is [Current Time, History of Underlying Security Prices]
- *Action* is Boolean: Exercise (i.e., Stop) or Continue
- *Reward* always 0, except upon Exercise (= Payoff)
- *State*-transitions based on Underlying Security's Risk-Neutral Process
- Key is function approximation of state-conditioned continuation value
- Continuation Value $\Rightarrow$ Optimal Stopping $\Rightarrow$ Option Price
- We outline two RL Algorithms:
    - Least Squares Policy Iteration (LSPI)
    - Fitted Q-Iteration (FQI)
- Both Algorithms are batch methods solving a linear system
- Reference: Li, Szepesvari, Schuurmans paper

Review of Least Squares Policy Iteration (LSPI)

- LSPI Algorithm performs a Least Squares Temporal Difference (LSTD) for each batch of episodes
- LSTD (for fixed policy π in a batch) builds matrix A and vector B
- $x(\cdot, \cdot)$ be a set of feature functions of state and action
- Update for A at each time step is: $x(s,a) \cdot (x(s,a) - \gamma \cdot x(s', \pi(s')))^T$
- Update for B at each time step is: $r \cdot x(s,a)$
- Sample (s,a,r,s') is randomly picked from stored past experiences (possibly generated from policies other than the batch target policy π)
- At end of batch:
  - Solve square linear system $Aw = B$
  - Update linear-approx Action-Value Function $Q(s,a;w) = w^T \cdot x(s,a)$
  - Improve policy as $\pi'(s) = \text{argmax}_a\, Q(s,a;w)$

LSPI customized for American Options Pricing

- a is e (exercise) or c (continue), s is $s_{i,j}$, s' is $s_{i,j+1}$
- r is $\gamma \cdot Payoff(s_{i,j+1})$ if $\pi(s_{i,j+1}) = e$ and $r = 0$ if $\pi(s_{i,j+1}) = c$
- We set $Q(s_{i,j}, e) = Payoff(s_{i,j})$ (not to be learnt)
- We set $Q(s_{i,j}, c; w) = w^T \cdot \phi(s_{i,j})$ (to be learnt)
- This requires us to set: $x(s_{i,j}, c) = \phi(s_{i,j})$ and $x(s_{i,j}, e) = 0$
- When $\pi(s_{i,j+1}) = c$, i.e., when $w^T \cdot \phi(s_{i,j+1}) \geq Payoff(s_{i,j+1})$
    - A update is: $\phi(s_{i,j}) \cdot (\phi(s_{i,j}) - \gamma \cdot \phi(s_{i,j+1}))^T$
    - B update is: 0
- When $\pi(s_{i,j+1}) = e$, i.e., when $w^T \cdot \phi(s_{i,j+1}) < Payoff(s_{i,j+1})$
    - A update is: $\phi(s_{i,j}) \cdot (\phi(s_{i,j}) - \gamma \cdot 0)^T$
    - B update is: $\gamma \cdot Payoff(s_{i,j+1}) \cdot \phi(s_{i,j})$

Algorithm 3.1: $\text{LSPI-AMERICANPRICING}(SP[0:m, 0:n+1])$

comment: $s_{i,j}$ is shorthand for state at $(i,j) := (t_j, SP[i, :(j+1)])$

comment: A is an $r \times r$ matrix, b and w are r-length vectors

comment: $A_{i,j} \leftarrow \phi(s_{i,j}) \cdot (\phi(s_{i,j}) - \gamma \cdot \mathbb{I}_{w^T \cdot \phi(s_{i,j+1}) \geq Payoff(s_{i,j+1})} \cdot \phi(s_{i,j+1}))^T$

comment: $b_{i,j} \leftarrow \gamma \cdot \mathbb{I}_{w^T \cdot \phi(s_{i,j+1}) < Payoff(s_{i,j+1})} \cdot Payoff(s_{i,j+1}) \cdot \phi(s_{i,j})$

$A \leftarrow 0, B \leftarrow 0, w \leftarrow 0$

for $i \leftarrow 0$ **to** $m - 1$

$$\textbf{do} \begin{cases} \textbf{for } j \leftarrow 0 \textbf{ to } n-1 \\ \quad \textbf{do} \begin{cases} Q \leftarrow Payoff(s_{i,j+1}) \\ P \leftarrow \phi(s_{i,j+1}) \textbf{ if } j < n-1 \textbf{ and } Q \leq w^T \cdot \phi(s_{i,j+1}) \textbf{ else } 0 \\ R \leftarrow Q \textbf{ if } Q > w^T \cdot P \textbf{ else } 0 \\ A \leftarrow A + \phi(s_{i,j}) \cdot (\phi(s_{i,j}) - e^{-r_{t_j}(t_{j+1}-t_j)} \cdot P)^T \\ B \leftarrow B + e^{-r_{t_j}(t_{j+1}-t_j)} \cdot R \cdot \phi(s_{i,j}) \end{cases} \\ w \leftarrow A^{-1} \cdot b, A \leftarrow 0, b \leftarrow 0 \textbf{ if } (i+1)\%BatchSize == 0 \end{cases}$$

Fitted Q-Iteration for American Options Pricing

Algorithm 3.2: FQI-AMERICANPRICING($SP[0:m, 0:n+1]$)

comment: $s_{i,j}$ is shorthand for state at $(i,j) := (t_j, SP[i, :(j+1)])$

comment: A is an $r \times r$ matrix, b and w are r-length vectors

comment: $A_{i,j} \leftarrow \phi(s_{i,j}) \cdot \phi(s_{i,j})^T$

comment: $b_{i,j} \leftarrow \gamma \cdot \max(Payoff(s_{i,j+1}), w^T \cdot \phi(s_{i,j+1})) \cdot \phi(s_{i,j})$

$A \leftarrow 0, B \leftarrow 0, w \leftarrow 0$
for $i \leftarrow 0$ **to** $m-1$

$$
\textbf{do} \begin{cases}
\textbf{for } j \leftarrow 0 \textbf{ to } n-1 \\
\quad \textbf{do} \begin{cases}
Q \leftarrow Payoff(s_{i,j+1}) \\
P \leftarrow \phi(s_{i,j+1}) \textbf{ if } j < n-1 \textbf{ else } 0 \\
A \leftarrow A + \phi(s_{i,j}) \cdot \phi(s_{i,j})^T \\
B \leftarrow B + e^{-r_{t_j}(t_{j+1}-t_j)} \cdot \max(Payoff(s_{i,j+1}), w^T \cdot P) \cdot \phi(s_{i,j})
\end{cases} \\
w \leftarrow A^{-1} \cdot b, A \leftarrow 0, b \leftarrow 0 \textbf{ if } (i+1)\%BatchSize == 0
\end{cases}
$$

Feature functions

- Li, Szepesvari, Schuurmans recommend Laguerre polynomials (first 3)
- Over $S' = S_t/K$ where S_t is underlying price and K is strike
- $\phi_0(S_t) = 1, \phi_1(S_t) = e^{-\frac{S'}{2}}, \phi_2(S_t) = e^{-\frac{S'}{2}} \cdot (1 - S'), \phi_3(S_t) = e^{-\frac{S'}{2}} \cdot (1 - 2S' + S'^2/2)$
- They used these for Longstaff-Schwartz as well as for LSPI and FQI
- For LSPI and FQI, we also need feature functions for time
- They recommend $\phi_0^t(t) = sin(\frac{\pi(T-t)}{2T}), \phi_1^t(t) = \log(T - t), \phi_2^t(t) = (\frac{t}{T})^2$
- They claim LSPI and FQI perform better than Longstaff-Schwartz with this choice of features functions
- We will code up these algorithms to validate this claim ☺