Optimal stopping & switching by approximating the indicator function with neural networks

Claudia Viaro

June 28, 2022

Outline

1. Optimal stopping problems
2. Approximation methods
3. Approximating the indicator function
   - Define stopping decisions
   - Approximating stopping decisions
4. Optimal switching problems

Optimal stopping problems

Definition

Let $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n\geq 0}, \mathbb{P})$ be a filtered probability space and $X = (X_n)_{n\geq 0}^N$ an $\mathbb{R}^d$-valued discrete-time stochastic process on it, describing the price of d stock prices under the risk-neutral measure $\mathbb{P}$.

For a finite time horizon $T \in \mathbb{N}$, denote by $\mathfrak{M}^T$ the class of all stopping times τ of the filtration $(\mathcal{F}_n)_{n\geq 0}$: the random variables $0 \leq \tau \leq N$ are finite a.s. and $\{\tau = n\} \in \mathcal{F}_n$ for all $n \in \{0, 1, \dots, N\}$.

The optimal stopping problem

$$V = \sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau) \tag{1}$$

consists in finding the quantity V and a stopping time $\tau^* \in \mathfrak{M}^T$ at which the supremum is attained (if it exists), where $g : [0, N] \times \mathbb{R}^d \to \mathbb{R}$ is a discounted payoff function.

An exact solution

Optimal stopping problems can be solved exactly:

- V^* is the Snell envelope

$$\begin{aligned} U_N &:= g(X_N), \\ U_n &:= \max(g(X_n), \mathbb{E}[\alpha U_{n+1}|X_n]), \end{aligned} \tag{2}$$

  where α is a discount factor. Equivalently, U_n can be expressed as the optimal stopping problem:

$$U_n = \sup_{\tau \in_n} \mathbf{E}[\alpha^{\tau - n} g(X_\tau|X_n] \tag{3}$$

  where τ_n is the set of all stopping times $\tau \leq n$

- τ^* is the first time the immediate reward dominates the continuation value

$$\begin{aligned} \tau_N &:= N, \\ \tau_n &:= \begin{cases} n, & \text{if } g(X_n) \geq \mathbb{E}[\alpha U_{n+1}|X_n] \\ \tau_{n+1}, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Approximation methods with NN

$\rightarrow$ numerical methods suffer from the curse of dimensionality (d assets)

A more recent line of research use backward recursion and stochastic gradient based methods to tackle high dimensional stopping problems.

Consider m realizations from a d-dimensional Markovian stochastic process X under $\mathbb{Q}$, where the i-th realization is $x_0, x_1^i, \dots, x_N^i$, with fixed spot price x_0. For each m the price, under stopping strategy 1, is:

$$\begin{aligned} p_N^i &:= g(x_N^i), \\ p_n^i &:= \begin{cases} g(x_n^i), & \text{if } g(x_n^i) \geq c_n(x_n^i), \\ \alpha p_{n+1}^i, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

where $c_n(x_n^i)$ is the continuation value

Stochastic gradient based methods are used to optimize the parameters θ_n of a neural network that approximates either:

- the continuation value (Kohler et al. 2010, Lapeyre & Lelong 2021, Becker et al. 2020)

$$f_\theta(x_n^i) \approx c_\theta(x_n^i) \tag{6}$$

  where $c_{\theta_n}(x) = \mathbb{E}[\alpha U_{n+1} | X_n]$. The parameters are found by minimizing the loss function obtained by backward recursive Monte Carlo approximation

$$\psi_n(\theta_n) = \sum_{i=1}^{m} (c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i)^2 \tag{7}$$

- the indicator function (Becker et al. 2019)

$$f_\theta(x_n^i) \approx \mathbb{1}_{\{g(x_n^i) \geq c(x_n^i)\}} \tag{8}$$

  The optimization involves directly the option price $\psi_n(\theta_n) = \dfrac{1}{m} \sum_{i=1}^{m} \alpha p_n^i$

These methods do not provide theoretical convergence guarantees due to the use of stochastic gradient methods with non-convex loss functions.

Main reference

”Deep optimal stopping” by Becker et al. (2019)

Problem formulation

Suppose the SDE describing the stochastic process is:

$$dX_t = \mu(X_t)dt + \sigma(X_t)dW_t, \qquad X_0 = x_0, \quad t \in [0, T]$$

of which we consider the Euler-Maruyama discretization scheme: let $\mathcal{X} = (\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(d)})$ $\{0, 1, \dots, N\} \times \Omega \to \mathbb{R}^d$ which satisfies for all $n \in \{0, 1, \dots, N-1\}$ that $\mathcal{X}_0 = \xi$ and

$$\mathcal{X}_{n+1} = \mathcal{X}_n + \mu(\mathcal{X}_n)(t_{n+1} - t_n) + \sigma(\mathcal{X}_n)(W_{t_{n+1}} - W_{t_n}) \tag{9}$$

where x_0 is the spot price, $W : [0, T] \times \Omega \to \mathbb{R}^d$ is a standard Brownian motion, μ and σ are the drift and volatility of the process.

Stopping decisions

Task 1

Let the n stopping problem be:

$$V_n = \sup_{\tau \in \mathcal{T}_n} \mathbb{E} g(\tau_n, X_{\tau_n}) \tag{10}$$

Show that the decision to stop the process can be made according to a sequence of functions $\{f_n(X_n)\}_{n=0}^{N}$, $f_n : \mathbb{R}^d \to \{0,1\}$. We aim to express τ_n as a function of the $0-1$ stopping decisions f_n

- at $n = N$, the process stops, this implies
  - ➢ $f_N \equiv 1$
  - ➢ time N stopping time $\tau_N \equiv N$, or $\tau_N \equiv N f_N(X_N)$
- for $n < N$, write τ_n as a function of $\{f_k(X_k)\}_{k=n}^{N}$

$$\tau_n = \sum_{m=n+1}^{N} m f_m(X_m) \prod_{j=n+1}^{m-1} (1 - f_j(X_j)) \quad \in \mathcal{T}_n \tag{11}$$

We need to show that the stopping time in 11 is the n optimal stopping time

Theorem Proof

For a given $n \in \{0, 1, \dots, N-1\}$, let τ_{n+1} be a stopping time in $\mathcal{T}_{n+1}$ of the form

$$\tau_{n+1} = \sum_{m=n+1}^{N} m f_m(X_m) \prod_{j=n+1}^{m-1} (1 - f_j(X_j)) \tag{12}$$

for measurable functions $f_{n+1}, \dots, f_N : \mathbb{R}^d \to \{0, 1\}$ with $f_N \equiv 1$. Then, there exists a measurable function $f_n : \mathbb{R}^d \to \{0, 1\}$ such that $\tau_n \in \mathcal{T}_n$ satisfies

$$\mathbb{R}g(\tau_n, X_{\tau_n}) \geq V_n - (V_{n+1} - \mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}}) \tag{13}$$

where V_n and V_{n+1} satisfy 3.

It follows that the overall optimal stopping time for 1 is

$$\tau = \sum_{n=1}^{N} n f_n(X_n) \prod_{k=0}^{n-1} (1 - f_k(X_k)) \tag{14}$$

How to use stopping decisions

Task 2

Becker et al. (2019) use a neural network to approximate the stopping decision functions. The series $\{(f_n)_{n=0}^N\}$ is approximated by a sequence of NN of the form $f^{\theta_n} : \mathbb{R}^d \to \{0,1\}$ with parameters $\theta_n \in \mathbb{R}^q$.

Let $n = N-1, \dots, 0$, and assume the parameters $\theta_{n+1}, \dots, \theta_N \in \mathbb{R}^q$ have been obtained. We can then generate a stopping time via:

$$\tau_{n+1} = \sum_{m=n+1}^{N} m f^{\theta_m}(X_m) \prod_{j=n+1}^{m-1} (1 - f^{\theta_j}(X_j)) \tag{15}$$

- τ_{n+1} gives an expected value $\mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}})$ close to the optimum V_{n+1}
- if $n = N-1$, we have $\tau_{n+1} \equiv N$
- τ_{n+1} is computed at time n because it is required to compute the continuation value at time n, V_{n+1}
- at $n = 0$, the stopping time $\hat{\tau}$ satisfies $\mathbb{E}g(\hat{\tau}, X_{\hat{\tau}}) \geq \sup_{\tau \in \mathcal{T}} \mathbb{E}g(\tau, X_\tau) - \epsilon$, for any $\epsilon > 0$

Neural Network

We employ a sequence of feed-forward neural networks $f^{\theta_n} : \mathbb{R}^d \to \{0, 1\}$ with parameters $\theta_n \in \mathbb{R}^q$, to approximate the stopping decisions f_n in 15 for $n \leq N - 1$, using $f_N \equiv 1$.

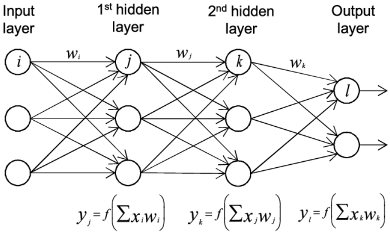
Input layer
1st hidden layer
2nd hidden layer
Output layer
i
j
k
l
w_i
w_j
w_k
$y_j = f\left(\sum x_i w_i\right)$
$y_k = f\left(\sum x_j w_j\right)$
$y_l = f\left(\sum x_k w_k\right)$

Figure: NN architecture

- NNs comprise multiple node layers through which data is passed
- each layer of nodes trains on a distinct set of features based on the previous layer's output

For $\theta \in \{\theta_0, \dots, \theta_N\}$, introduce the neural network $F^\theta : \mathbb{R}^d \to (0,1)$:

$$F^\theta = \psi \circ a_3^\theta \circ \phi_{q_2} \circ a_2^\theta \circ \phi_{q_1} \circ a_1^\theta \tag{16}$$

where

- q_1 and q_2 are the number of nodes in the hidden layers
- $a_1^\theta : \mathbb{R}^d \to \mathbb{R}^{q_1}, a_2^\theta : \mathbb{R}^{q_1} \to \mathbb{R}^{q_2}$ are affine functions : $a_i^\theta(x) = W_i x + b_i$
- $\phi_{q_1} : \mathbb{R}^{q_1} \to \mathbb{R}^{q_1}$ is the ReLU activation function: $\phi_{q_1}(x_i, \dots, x_{q_i}) = (x_i^+, \dots, x_{q_i}^+)$
- $\psi : \mathbb{R} \to \mathbb{R}$ is the logistic sigmoid function: $\psi(x) = 1/(1 + e^{-x})$

F^θ outputs stopping probabilities $\in (0,1)$ as opposed to stopping decisions f^θ because the gradient-based optimization requires continuous output.

Parameter space

- the parameter space $\theta \in \mathbb{R}^q$ of F^θ is made of $\theta = \{W_1, W_2, b_1, b_2\}$, where $W_1 \in \mathbb{R}^{q_1 \times d}, W_2 \in \mathbb{R}^{q_2 \times q_1}, W_3 \in \mathbb{R}^{q_2 \times 1}$ and $b_1 \in \mathbb{R}^{q_1}, b_2 \in \mathbb{R}^{q_2}, b_3 \in \mathbb{R}^1$ are the matrices and vectors of the affine functions

$$a_i^\theta(x) = W_i x + b_i, \quad i = 1, 2 \tag{17}$$

- the dimension of the parameter space is:

$$q = 1 + q_1 + q_2 + dq_1 + dq_2 + q_1 q_2$$

  for a NN depth of 2

Stochastic Gradient Descent - digression

at every time step $n = N-1, \dots, 1$, we have:

- **Training dataset:** simulate sample paths $(x_n^m)_{n=1}^N$ for $m = 1, \dots, M$
- **Loss function:** minimize a loss function (e.g. mean squared-error) $C(\theta)$ with respect to θ and set the model parameters using the training data
- **Gradient:** $\nabla_\theta C(\theta^t)$ is computed via backpropagation
- **Update:** the optimal parameters are found by minimizing $C(\theta)$ via a gradient descent algorithm. e.g. with learning rate η

  $$\theta_{t+1} \leftarrow \theta^t - \eta \cdot \nabla_\theta C(\theta^t) \tag{18}$$

  This is equivalent to maximizing the reward function
- **Testing dataset:** simulate new sample paths $(y_n^m)_{n=1}^N$ and run the algorithm using the optimized parameters from 18,

Reward function $C(\theta)$

Objective

By backward recursion, find the optimal stopping time τ^* in 1 that maximizes the expected future reward.

- At $n = N$ we set $f_N \equiv 1$ and the reward is $g(N, X_N)$
- At each timestep n we can
  - stop $(f_n(X_n) = 1)$ and receive reward of $g(n, X_n)$
  - continue $(f_n(X_n) = 0)$ and then proceed behaving optimally and receive a reward of $g(\tau_{n+1}, X_{\tau_{n+1}})$

This translates into finding function $f : \mathbb{R}^d \to \{0, 1\}$ that maximizes

$$\mathbb{E}[g(n, X_n)f(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - f(X_n))] \tag{19}$$

The stochastic gradient ascent optimization algorithm computes the NN parameters θ that approximates the objective function:

$$\sup_{\theta \in \mathbb{R}^q} \mathbb{E}[g(n, X_n) F^{\theta}(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - F^{\theta}(X_n))] \tag{20}$$

by iteratively maximizing the objective function for $n = N-1, \ldots, 1$ with $F^{\theta_N} \equiv 1$:

$$\mathbb{E}[g(n, X_n) F^{\theta_n}(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - F^{\theta_n}(X_n))] \tag{21}$$

which produces a value close to 20

Once $\theta_n \in \mathbb{R}^q$ are obtained, F^{θ_n} is transformed into a stopping decision $f^{\theta_n} \in \mathbb{R}^q \to \{0,1\}$

An application

Optimal switching

Let the stochastic system that operate in 2 regimes $\mathbb{I} = \{0, 1\}$.

- regimes can be switched at a sequence of stopping times over a finite horizon $[0, \dots, T]$
- there is a payoff rate per unit of time when the system is in mode $i \in \mathbb{I}$ at time t as a mapping $\Psi_i(t, X_t) : \Omega \times [0, T] \to \mathbb{R}$
- there is a cost for switching from regime i to j given by the function $\gamma_{i,j} : \Omega \times [0, T] \to \mathbb{R}$ to cover for the extra costs due to the change of the regime

Management strategy

A strategy α for the power plant is a combination of two sequences:

- non decreasing sequence of $\mathbb{F}$-stopping times $(\tau_n)_{n\geq 1}$, $n \in \mathbb{N}\backslash\{0\}$, where at τ_n the production is switched from the current mode i to j. Assume: $\tau_0 = t$ and $\tau_n \leq \tau_{n+1}$.
- a sequence of indicators $(\iota)_{n\geq 1}$, $n \in \mathbb{N}\backslash\{0\}$, $\mathcal{F}_{\tau_n}$- measurable valued in $\mathbb{I}_m$. At time $t = \tau_n$ the system is switched from the current regime ι_{n-1} to ι_n, with $\iota_0 = i$.

Denote by $\mathcal{A}_{t,i}$ the set of admissible strategies to switch at time τ_n, $n \geq 1$, from the current regime ι_{n-1} to ι_n.

Objective function

For any initial condition $(x, i) \in [0, T] \times \mathbb{I}_m$, and any control $\alpha = (\tau_n, \iota_n)_{n \leq 0} \in \mathcal{A}_{t,i}$. the total expected payoff up to T for such strategy can be expressed as:

$$J_i(x, \alpha) = \mathbb{E}\Big[\sum_{s=t}^{T-1} \Psi(X_t^{x,i}, I_t^i) + \Gamma - \sum_{n \leq 1} \gamma_{\iota_{n-1}, \iota_n} \mathbf{1}_{\{\tau_n < T\}} | \mathcal{F}_n \Big] \tag{22}$$

The objective is to maximize this expected total profit for all strategies α. For this purpose, we set the value function:

$$V_i(x) = \sup_{\alpha \in \mathcal{A}} J_i(x, \alpha) \qquad \forall \alpha \in \mathcal{A}_{t,i} \ \mathbb{P} \ a.s. \tag{23}$$

Proposed approach

Following the approach given by Becker et al. (2019), we want to provide a solution to the optimal switching problem in 23. This requires

1. express stopping times as a series of stopping decisions
2. approximate these $0-1$ stopping decisions using a neural network

$$\check{Y}^i_N = \Gamma$$

$${}^{i}_{n} = \Psi_i(n) + \max_{j\in\{0,1\}}\{-\gamma_{i,j}(n) + e^{-\rho h}\check{Y}^j_{n+1}\} \qquad \text{for } n = N-1,\ldots,0 \text{(24)}$$

An application

some elements of the system:

- payoff function for the call option used is of the form $(\max_{i\in\{1,\ldots,d\}} X_t^i - K)^+$, where K is the strike price at any point in the time grid $0 = t_0 < t_1 < \ldots < t_N = T$
- the system also outputs a final reward for being in mode $i \in \mathbb{I}$ at time T given by Γ_i

Training performance

.5

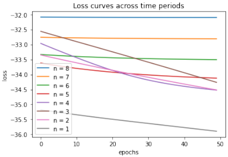

Figure: Optimal Stopping

.5

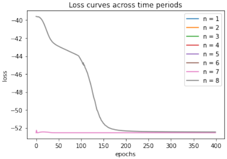

Figure: Optimal Switching

Figure: Loss curves (train) for 8 time steps

- payoff function for the call option used is of the form $(\max_{i \in \{1,\dots,d\}} X_t^i - K)^+$, where K

Longstaff & Schwartz (2001) approximate continuation value for the decision to stop or to continue most used method in the financial industry If $\mathbb{E}[U_{t_n+1}|\mathcal{F}_{t_n}]$ is known, an optimal stopping time policy can be synthesized explicitly by stopping if and only if $Z_{t_n} \geq \mathbb{E}[U_{t_n+1}|\mathcal{F}_{t_n}]$. The algorithm proposed approximates the continuation value as:

$$c_\theta \tag{25}$$

convergence guarantees do not easily scale to high dimensional problems since the number of basis functions usually grows polynomially or even exponentially (Longstaff and Schwartz, 2001, Section 2.2) in the number of stocks. One direction of research to overcome this problem is to apply dimension reduction techniques (Bayer et al. 2021)

Least Square Policy Iteration

A policy π is a behavior function that maps states to actions:

$$a = \pi(s) \tag{26}$$

If we consider a full episode (one of our complete trajectories) that starts at time t=0 and terminates at time t=T, then we define the return Râ from the starting state as:

$$R_0 = \sum_{t=0}^{T} \gamma^t r_t \tag{27}$$

Where r_t is the reward at time $= t$, and γ is a discount factor in the range $[0, 1]$, which allows us to adjust the time horizon, and ensure the return over long episodes remains finite.
As the transition from state $s \rightarrow s'$ is generally probabilistic (starting multiple episodes from the same state will generally result in different returns), we take the expectation value of return over all allowed trajectories. By doing so, we can define the expected return, called the state-value function V_π which describes the expected return starting from state s and then following policy π:

$$V_\pi = \mathbb{E}[R_t | s_t = s_t] \tag{28}$$

where R_t is the return of a single, specific full episode starting at time t. The expectation operator $\mathbb{E}[\cdot]$ averages over all possible individual episodes/trajectories starting from

In our algorithm, we approximate the action-value function $Q_\pi(s,a)$ by a linear architecture and its actual representation consists of a compact description of the basis functions and a set of parameters

Q_π values are approximated by a linear parametric combination of k basis functions (features):

$$\hat{Q}_\pi(s,a;w) = \sum_{j=1}^{k} \phi_j(s,a) w_j \tag{31}$$

where the $w_j$âs are the parameters. The basis functions $\phi_j(s,a)$ are fixed, but arbitrary and, in general, non-linear, functions of s and a. We require that the basis functions ϕ_j are linearly independent to ensure that there are no redundant parameters and that the matrices involved in the computations are full rank. Typical linear approximation architectures are polynomials of any degree (each basis function is a polynomial term) and radial basis functions (each basis function is a Gaussian with fixed mean and variance). Define $\phi(s,a)$ to be the column vector of size k where each entry j is the corresponding basis function ϕ_j computed at (s,a):

$$\phi(s,a) = \begin{pmatrix} \phi_1(s,a) \\ \dots \\ \phi_k(s,a) \end{pmatrix} \tag{32}$$

Now, $\hat{Q}_\pi$ can be expressed compactly as $\hat{Q}^\pi = \mathbf{\Phi} w^\pi$, where w^π is a column vector of length k with all parameters and $\mathbf{\Phi}$ is a $(|\mathcal{S}||\mathcal{A}|\tilde{A}k)$ matrix, where each row contains the value of all basis functions for a certain pair (s,a) and each column the value of a

Update rule

Consider the problem of learning the (weighted) least-squares fixed-point approximation $\hat{Q}_\pi$ to the state-action value function Q_π of a fixed policy π from samples. Assuming that there are k linearly independent basis functions in the linear architecture, this problem is equivalent to learning the parameters w^π of $\hat{Q}_\pi = \mathbf{\Phi} w^\pi$. The exact values for w^π can be computed from the model by solving the $(k\tilde{A}k)$ linear system:

$$\mathbf{A} w^\pi = b, \tag{33}$$

where

$$\mathbf{A} = \mathbf{\Phi}^\intercal \Delta_\mu (\mathbf{\Phi} \hat{a} \gamma \mathbf{P} \mathbf{\Pi}_\pi \mathbf{\Phi}) \quad \text{and} \quad b = \mathbf{\Phi}^\intercal \Delta_\mu \mathcal{R}, \tag{34}$$

and μ is a probability distribution over $S\tilde{A}A)$ that defines the weights of the projection. For the learning problem, $\mathbf{A}$ and b cannot be determined a priori, either because the matrix $\mathbf{P}$ and the vector $\mathcal{R}$ are unknown, or because they are so large that they cannot be used in any practical computation. However, $\mathbf{A}$ and b can be learned using samples; the learned linear system can then be solved to yield the learned parameters we π which, in turn, determine the learned value function.
So, given any finite set of L samples:

$$D = \Big\{ (s_i, a_i, r_i, s_i'), i = 1, \dots, L \Big\} \tag{35}$$

then we can compute $\mathbf{A}$ and b as:

Table of Contents

Supplemental content. Back to main.

References

Bayer, C., Eigel, M., Sallandt, L. & Trunschke, P. (2021), ‘Pricing high-dimensional bermudan options with hierarchical tensor formats’, *arXiv preprint arXiv:2103.01934* .

Becker, S., Cheridito, P. & Jentzen, A. (2019), ‘Deep optimal stopping’, *Journal of Machine Learning Research* **20**, 74.

Becker, S., Cheridito, P. & Jentzen, A. (2020), ‘Pricing and hedging american-style options with deep learning’, *Journal of Risk and Financial Management* **13**(7), 158.

Kohler, M., Krzyżak, A. & Todorovic, N. (2010), ‘Pricing of high-dimensional american options by neural networks’, *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* **20**(3), 383–410.

Lapeyre, B. & Lelong, J. (2021), ‘Neural network regression for bermudan option pricing’, *Monte Carlo Methods and Applications* **27**(3), 227–247.

Longstaff, F. A. & Schwartz, E. S. (2001), ‘Valuing american options by simulation: a simple least-squares approach’, *The review of financial studies* **14**(1), 113–147.