A Deep Learning Method for Optimal Stopping Problems

Andreu Boix Torres

FME - UPC,
Facultat de Matemàtiques i Estadística

Thesis Defense
11^{th} July 2023

Supervisor: Argimiro Arratia Quesada
Department of Computer Science & BGSMath & IMTech

Table of Contents

Table of Contents

Optimal Stopping problem: definitions

Definition 1 (Stochastic process, path and SDE)

Stochastic process: Collection of rv's $\{X_t : t \in T\}$ defined in $(\Omega, \mathcal{F}, \mathbb{P})$.
Path or trajectory: Given a stochastic process $(X_t)_t$ and fixed $\omega \in \Omega$, as time t passes, $(X_t(\omega))_t$.
SDE: Differential equation s.t. some coeffs are SPs:

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$$

We will work with Itô diffusions:

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \quad t \geq s; X_s = x.$$

More definitions

Definition 2 (Stopping time)

Consider a SP X_t on $(\Omega, \mathcal{F}, (\mathcal{M}_t)_{t\geq 0}, \mathbb{P})$. A stopping time for X_t is a non-negative integer-valued rv τ satisfying that for each $t \geq 0$, the event $\{\tau = t\}$ depends only on $\{X_0, X_1, \ldots, X_t\}$ and not on $\{X_{t+s} : s \geq 1\}$. Also, $\mathbb{P}(\tau < \infty) = 1$.

Goal: **Optimal Stopping Time problem**

We want to solve:

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

where $\mathcal{T}$ is the set of stopping times and $g : \{0, 1, \ldots, N\} \times \mathbb{R}^d \to \mathbb{R}$ is a measurable function.

Into the Optimal Stopping problem

- Integrability condition:

$$\mathbb{E}|g(n, X_n)| < \infty \quad \text{for all } n \in \{0, 1, \ldots, N\}.$$

- Confidence interval derivation condition:

$$\mathbb{E}[g(n, X_n)^2] < \infty \quad \text{for all } n \in \{0, 1, \ldots, N\}.$$

Mathematical Finance: brief introduction

Definition 3 (Options)

Call / Put option: Financial contract in which the buyer of the contract has the option of *buying / selling* the underlying asset at strike K.

European / American style: The option can be given only at maturity T or at discrete time stamps $t_1 < t_2 < \cdots < t_n = T$.

Mathematical Finance

And Optimal Stopping

A financial derivative is expressed as a function D_T such that $D_T : \Omega \to \mathbb{R}$ is measurable at $\mathcal{F}_T$. Its pricing at time $t \in [0, T]$ is:

$$D_t = e^{-r(T-t)} \cdot \mathbb{E}_q[D_T \mid \mathcal{F}_t]$$

where q is the risk-neutral measure and the exponential term is the discount factor.

For a European call option:

$$D_T = C_T = max(S_T - K, 0).$$

For an American call option, optimal stopping matters:

$$D_t = c_t = \sup_{\tau \in \mathcal{T}_t} \mathbb{E}\left[e^{-r(\tau-t)} max(S_\tau - K, 0) \mid S_t\right]$$

Underlying asset modeling

Definition 4 (Bachelier and Black-Scholes models)

Bachelier. $T > 0$. The dynamics of the underlying asset is:

$$dS_t = \mu dt + \sigma dW_t.$$

Black-Scholes. $T > 0$. The dynamics of the underlying asset is:

$$dS_t = \mu S_t dt + \sigma S_t dW_t.$$

Underlying asset modeling

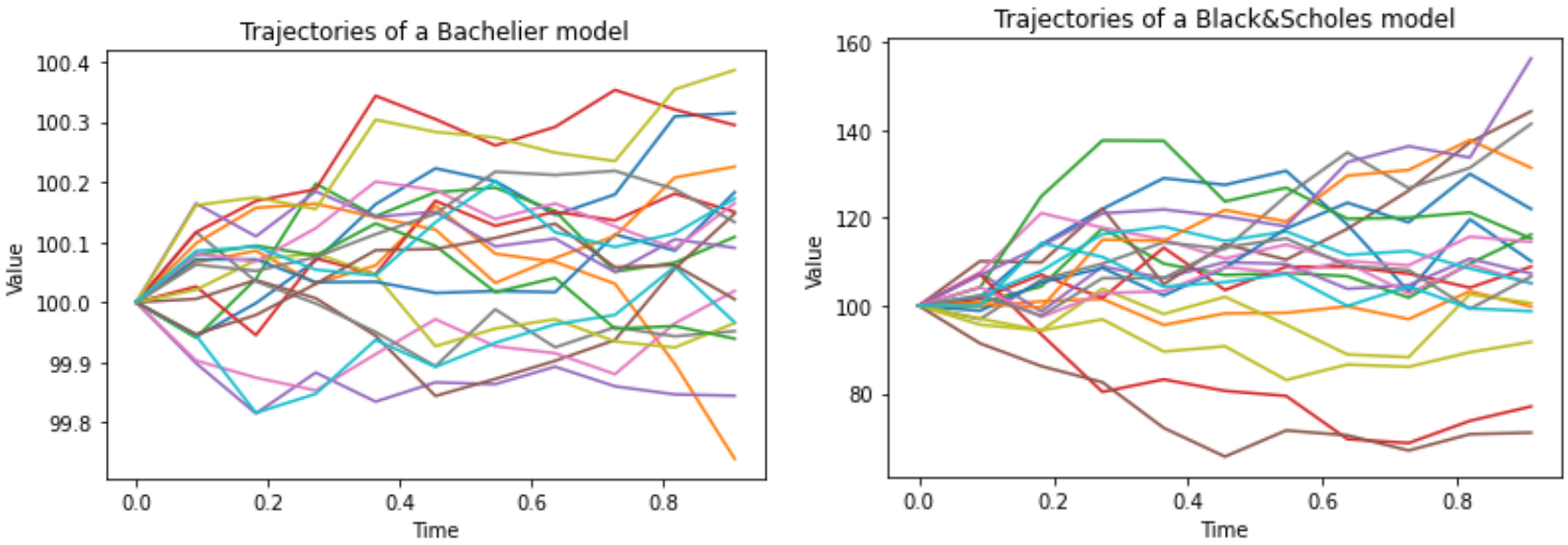


Figure 1: $T = 1, N = 10, Nsim = 20, \mu = 0.05, \sigma = 0.20, S_0 = 100$

Adding jumps

Poisson process: Let $(\tau_i)_{i\geq 1} \sim Exp(\lambda)$. Let $T_n = \sum_{i=1}^{n} \tau_i$. Then, $N_t = \sum_{n=1}^{\infty} \mathbb{1}_{\{t\geq T_n\}}$.

Definition 5 (Kou model)

Kou. $T > 0$. Let $Y_t = \sum_{i=1}^{N_t} D_i$ with $N_t \sim Poiss(\lambda t)$ and D_i has the jump distribution:

$$f_{D_i}(z) = p\lambda_+ e^{\lambda_+ z}\mathbb{1}_{\{z\geq 0\}} + (1-p)\lambda_- e^{-|\lambda_z| z}\mathbb{1}_{z<0}.$$

Having V_i as a sequence of positive iid rv such that $D_i = log(V_i)$, the dynamics of the underlying asset is:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + S_t d\left(\sum_{i=1}^{N_t}(V_i - 1)\right).$$

Adding jumps

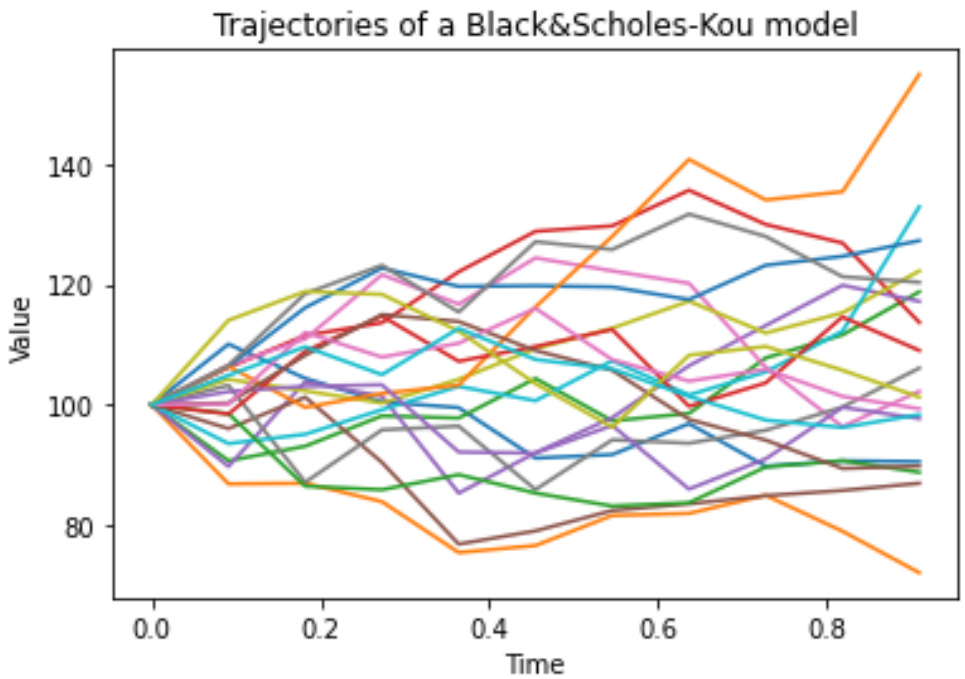


Figure 2: $T = 1, N = 10, Nsim = 20, \mu = 0.05, \sigma = 0.20, S_0 = 100, \lambda = 3, \lambda_+ = \lambda_- = 1/25$

Adding jumps

Poisson process: Let $(\tau_i)_{i\geq 1} \sim Exp(\lambda)$. Let $T_n = \sum_{i=1}^{n} \tau_i$. Then, $N_t = \sum_{n=1}^{\infty} \mathbb{1}_{\{t \geq T_n\}}$.

Definition 6 (Merton model)

Merton. $T > 0$. Let $Y_t = \sum_{i=1}^{N_t} D_i$ and $D_i \sim N(\mu_J, \sigma_J^2)$. Having V_i as a sequence of positive iid rv such that $D_i = log(V_i)$, the dynamics of the underlying asset is:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + S_t d\left(\sum_{i=1}^{N_t} (V_i - 1)\right).$$

Adding jumps

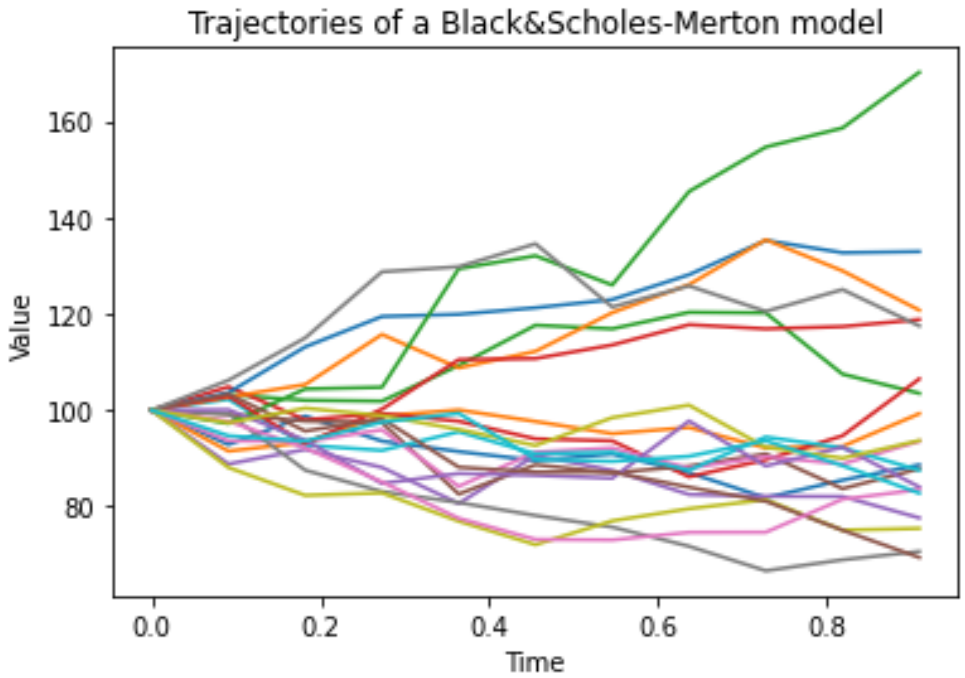


Figure 3:
$T = 1, N = 10, Nsim = 20, \mu = 0.05, \sigma = 0.20, S_0 = 100, \lambda = 3, \mu_J = 0, \sigma_J = 1$

Definition 7 (Heston model)

ρ correlation between Brownian motions, $\sigma(t) = \sqrt{y(t)}$, ξ vol-of-vol, θ mean-reversing term, κ speed of mean-reversion.

$$\begin{cases} dS_t = \hat{\mu} S_t dt + \sigma(t, S_t) dW_t^S \\ dy_t = \kappa(\theta - y_t)dt + \xi\sqrt{y_t} dW_t^y \end{cases}$$

where:

$$dW_t^S \cdot dW_t^y = \rho dt$$

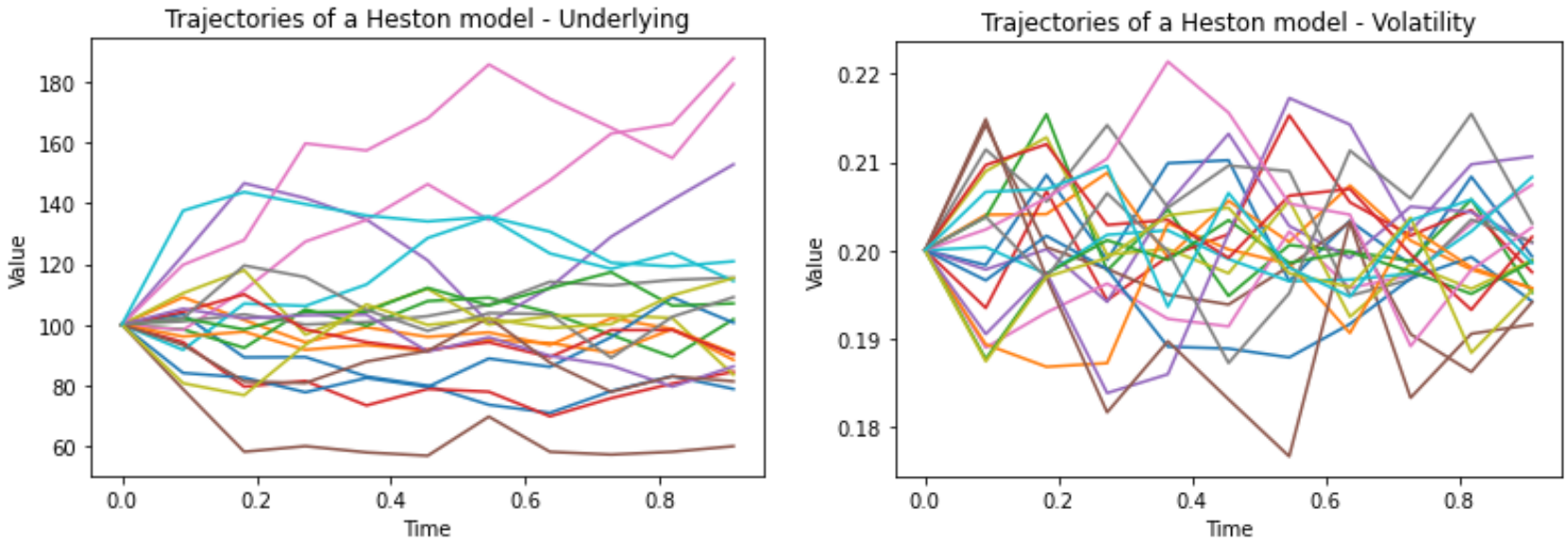
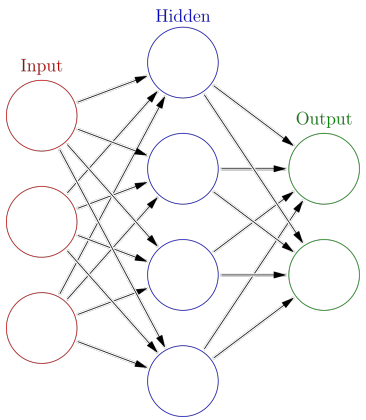


Figure 4:
$T = 1, N = 10, Nsim = 20, \mu = 0.05, \sigma = 0.20, S_0 = 100, \lambda = 3, \mu_J = 0, \sigma_J = 1$

Neural Networks

Definition 8 (Feed-forward neural network)

Approximate nonlinear functions using hidden layers and activation functions.
Deep Neural Networks have multiple hidden layers.
Neurons activate based on weighted input combinations, producing signals. The output is generated by combining these signals with a bias and applying an activation function.



$$o = f\left(\sum_i \phi_i x_i + \sum_j \theta_j \mathcal{L}\left(\sum_i \omega_{i,j} x_i + \alpha_j\right) + \varepsilon\right)$$

where x_i input values, o output, $\mathcal{L}$ activation function for each hidden node, α_j thresholds, $\omega_{i,j}$ weights for i-j, θ_j and ϕ_i weights for the linear combinations, ε bias and f output node's activation function.

Net's architecture

$$F^\theta = \psi \circ a_l^\theta \circ \phi_{q_l-1} \circ a_{l-1}^\theta \circ \cdots \circ \phi_{q_1} \circ a_1^\theta,$$

where

- $l, q_1, q_2, \ldots, q_{l-1}$ are positive integers that specify the depth of the network and the number of nodes in the hidden layers.
- $a_1^\theta : \mathbb{R}^d \to \mathbb{R}^{\theta_{q_1}}, \ldots, a_{l-1}^\theta : \mathbb{R}^{q_{l-2}} \to \mathbb{R}^{q_{l-1}}$ and $a_{q_{l-1}}^\theta : \mathbb{R}^{q_{l-1}} \to \mathbb{R}$ are affine functions.
- Let $j \in \mathbb{N}$, $\phi_j : \mathbb{R}^j \to \mathbb{R}^j$ is the component-wise ReLU activation function given by $\phi_j(x_1, \ldots, x_j) = (x_1^+, \ldots, x_j^+)$.
- $\psi : \mathbb{R} \to (0, 1)$ is the logistic function $\psi(z) = e^x/(1 + e^x) = 1/(1 + e^{-x})$.
- θ in f^θ refers to the parameters of the model that include the weights and the biases.

Table of Contents

Representation

Goal

The decision to stop the Markov process $(X_n)_{n=0}^N$ can be made according to a sequence of functions $\{f_n(X_n)\}_{n=0}^N : \mathbb{R}^d \to \{0,1\}$.

Consider

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

with $\mathcal{T}_n$ set of all stopping times s.t. $n \leq \tau \leq N$.

Notice that for $n = N$, $\mathcal{T}_N = \{N\} \implies \tau_N = N = N \cdot f_N(X_N)$ with $f_N \equiv 1$.

Representation

For $0 \leq n \leq N-1$:

$$\tau_n = \sum_{k=n}^{N} k f_k(X_k) \prod_{j=n}^{k-1} (1 - f_j(X_j))$$

$\tau_n \in \mathcal{T}_n$? Let $0 \leq n < q \leq N$. Then:

$$\{\tau_n = q\} \iff \{f_q(X_q) \prod_{k=n}^{q-1} (1 - f_k(X_k)) = 1\} \iff$$

$$\iff f_q(X_q) = 1 \text{ and } f_k(X_k) = 0 \ \forall k \text{ s.t. } n \leq k \leq q-1.$$

Representation

Theorem 9 (Backward recursion method)

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

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

for $f_{n+1}, \ldots, f_N : \mathbb{R}^d \to \{0, 1\}$, with $f_N \equiv 1$.

Then, there exists $f_n : \mathbb{R}^d \to \{0, 1\}$ such that $\tau_n \in \mathcal{T}_n$ satisfies:

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

Representation

Hence,

Remark

The overall optimal stopping time corresponding to

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

is given by:

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

Now... how to select $\{f_i\}_i$?

Optimality

Goal

Iteratively approximate optimal stopping decisions $\{f_n\}_{n=0}^N$ with a sequence of neural networks $f^{\theta_n} : \mathbb{R}^d \to \{0,1\}$ with parameters $\theta_n \in \mathbb{R}^q$.

Let $n \in \{0,1,\ldots,N-1\}$ and assume $\theta_{n+1}, \theta_{n+2}, \ldots, \theta_N \in \mathbb{R}^q$ have been found such that $f^{\theta_N} \equiv 1$ and the stopping time:

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

produces $\mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}})$ close to the optimum V_{n+1}.

Optimality

Remark (Gradient-based optimization method condition)

$f^\theta \in \{0,1\} \implies F^\theta \in (0,1)$.

We define the neural networks

$$F^{\theta_n} := \psi \circ \phi \circ a_2^{\theta_n} \circ \phi \circ a_1^{\theta_n}$$

where a_i is an affine function for the layer i, $\phi(X) = max(x, 0)$ and $\psi(x)$ is the sigmoid or logistic function.

$$f^{\theta_n} = (F^{\theta_n} > 0.5) = \mathbb{1}_{(0,\infty)} \circ \phi \circ a_2^{\theta_n} \circ \phi \circ a_1^{\theta_n}. \quad (1)$$

Criterion

Maximize over f measurable functions:

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

Theorem 10 (Neural network approximation theorem)

Let $n \in \{0, 1, \ldots, N-1\}$ and fix a stopping time $\tau_{n+1} \in \mathcal{T}_{n+1}$. Then, for every constant $\varepsilon > 0$, there exist positive integers q_1, q_2 such that:

$$\sup_{\theta \in \mathbb{R}^q} \mathbb{E}\left[g(n, X_n) f^{\theta}(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - f^{\theta}(X_n))\right] \geq$$

$$\geq \sup_{f \in \mathcal{D}} \mathbb{E}\left[g(n, X_n) f(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - f(X_n))\right] - \varepsilon.$$

Corollary 11

For any $\varepsilon > 0$ and $\sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau)$ there exist $q_1, q_2 \in \mathbb{N}$ and neural networks of the form (1) s.t. $f^{\theta_N} \equiv 1$ and the stopping time

$$\hat{\tau} = \sum_{n+1}^{N} n f^{\theta_n}(X_n) \prod_{k=0}^{n-1} (1 - f^{\theta_k}(X_k))$$

satisfies $\mathbb{E} g(\hat{\tau}, X_{\hat{\tau}}) \geq \sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau) - \varepsilon$.

Table of Contents

Lower bound

Step 1. Simulate M independent paths of the Markov process $(X_n)_{n=0}^N$: $(x_n^m)_{n=0}^N$ for $m = 1, \dots, M$.

Step 2. Assume we are at time step n. Then

$$\hat{\tau}_{n+1}^m = \sum_{k=n+1}^{N} k f^{\theta_k}(x_n^m) \prod_{j=n+1}^{k-1} (1 - f^{\theta_j}(x_j^m)).$$

Step 3. Compute the rest of the $\{\theta_n\}_n$ recursively backwards for $n = N-1, \dots, 0$ using as a loss function the Monte Carlo estimator for the Criterion shown before, using

$$r_n^m(\theta_n) = g(n, x_n^m) F^{\theta_n}(x_n^m) + g(\hat{\tau}_{n+1}^m, x_{\hat{\tau}_{n+1}^m}^m)(1 - F^{\theta_n}(x_n^m)),$$

which is the virtual reward along the path m. That is,

$$r_n^{MC} = \frac{1}{M} \sum_{m=1}^{M} r_n^m(\theta_n)$$

Lower bound
Testing section

Step 4. Simulate M independent paths of the Markov process $(X_n)_{n=0}^N$: $(y_n^m)_{n=0}^N$ for $m = 1, \dots, M$.

Step 5. Using the $\{\theta_n\}_n$ trained with our $(x_n^m)_{n,m}$, compute:

$$\hat{\tau}_{n+1}^m = \sum_{k=n+1}^{N} k f^{\theta_k}(y_n^m) \prod_{j=n+1}^{k-1} (1 - f^{\theta_j}(y_j^m)).$$

Step 6. The lower bound of the estimator for $\mathbb{E}g(\hat{\tau}, X_{\hat{\tau}})$ is:

$$\hat{L} = \hat{V} = \frac{1}{M} \sum_{m=1}^{M} g(\hat{\tau}_m, y_{\hat{\tau}_m}^m)$$

Upper bound

Idea

Defining a dual function $F(t, \pi)$ for an arbitrary supermartingale π_t:

$$F(t,\pi) := \mathbb{E}\left[\max_{s\in[t,N]\cap\mathcal{T}}(g(s,X_s)-\pi_s)\right]+\pi_t,$$

the dual problem is to minimize the dual function at time 0 for all supermartingales π_t. The optimal value is:

$$U = \inf_{\pi} F(0,\pi) = \inf_{\pi}\mathbb{E}\left[\max_{s\in\mathcal{T}} g(s,X_s)-\pi_s\right]+\pi_0.$$

Theorem 12 (Duality Relation)

The optimal values of the primal problem and the dual problem are equal: $V = U$.

Upper bound

We discretize (decompose) $\mathbb{E}g(\tau, X_\tau)$ and find an upper bound:

$$U = \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M_n - \varepsilon_n)\right]$$

where M_n is the martingale part of $\mathbb{E}g(\tau, X_\tau)$ and ε_n is the sequence of integrable error terms at time n.

Upper bound

Implementation

Step 1. Generate a third set of realizations of $(X_n)_{n=0}^N$ $(z_n^k)_{n=0}^N$, $k = 1, 2, \ldots, K_U$. For every z_n^k, simulate J continuation paths $\tilde{z}_{n+1}^{k,j}$, $j = 1, \ldots, J$.

Step 2. Compute the estimation for the continuation values:

$$C_n^k = \frac{1}{J} \sum_{j=1}^{J} g\left(\tau_{n+1}^{k,j}, \tilde{z}_{\tau_{n+1}^{k,j}}^{k,j}\right), \quad n = 0, 1, \ldots, N_1.$$

Step 3. $M_n^\Theta + \varepsilon_n$ can be estimated with:

$$M_n^k = \begin{cases} 0 & \text{if } n = 0 \\ \sum_{m=1}^{n} \Delta M_m^k & \text{if } n \geq 1 \end{cases}$$

$$\Delta M_n^k = f^{\theta_n}(z_n^k) g(n, z_n^k) + (1 - f^{\theta_n}(z_n^k)) C_n^k - C_{n-1}^k$$

Step 4. We derive

$$\hat{U} = \frac{1}{K_u} \sum_{k=1}^{K_U} \max_{0 \leq n \leq N} \left(g(n, z_n^k) - M_n^k\right).$$

Point estimation

- **Estimation**.

$$V_0 = \frac{\hat{L} + \hat{U}}{2}$$

- **Confidence intervals**. Applying the central limit theorem for large K_L and K_U, the $1-\alpha$ confidence interval for V_0 is:

$$\left[\hat{L} - z_{\alpha/2}\frac{\hat{\sigma}_L}{\sqrt{K_L}}, \hat{U} + z_{\alpha/2}\frac{\hat{\sigma}_U}{\sqrt{K_U}}\right]$$

where

$$\hat{\sigma}_L^2 = \frac{1}{K_L - 1}\sum_{k=1}^{K_L}\left(g(I^k, y_{I^k}^k) - \hat{L}\right)^2$$

and

$$\hat{\sigma}_U^2 = \frac{1}{K_U - 1}\sum_{k=1}^{K_U}\left(\max_{0\leq n\leq N}\left(g(n, z_n^k) - M_n^k\right) - \hat{U}\right)^2$$

Table of Contents

Optimization techniques

Our implementation was coded in PyTorch (ML library for Python).

- Batch normalization
- Training with the payoff
- Training the nets simultaneously
- Tweaking learning rates

Batch normalization

Activations have zero mean and unit variance

Network converges faster and more consistently

```
class Neural_Net_NN(torch.nn.Module):
    def __init__(self, M,shape_input):
        super(Neural_Net_NN, self).__init__()
        self.dense1 = nn.Linear(shape_input,M)
        self.dense2 = nn.Linear(M,1)
        self.batchnorm1 = nn.BatchNorm1d(M)
        self.relu = nn.ReLU()
    def forward(self, x):
        x = self.batchnorm1(
            self.relu(self.dense1(x))
            )
        x = self.relu(self.dense2(x))
        return x
```

Training with the payoff

Training $Y_t = (X_t, g(t, X_t))$.

```
neural_net =
    Neural_Net_NN(num_neurons,d+1).to(device)
X,p_,g_tau = x, p, p[:,:,n-1]
X,p_,g_tau = X.to(device),
    p_.to(device),g_tau.to(device)
state = torch.cat((X,p_),axis = 1)
loss = np.zeros(1)
# Afterwards in the code...
net_n = neural_net(state[:,:,n])
F_n   = torch.sigmoid(net_n)
loss -= torch.mean(p_[:, :, n] * F_n + g_tau
    * (1. - F_n))
g_tau = torch.where(net_n > 0, p_[:, :, n],
    g_tau)
```

We send our PyTorch tensors to *device*, to use CUDA:

```
device = torch.device('cuda' if
    torch.cuda.is_available() else 'cpu')
```

Training the nets simultaneously

And tweaking LRs

```
1    neural_net =
         Neural_Net_NN(num_neurons,d+1).to(device)
2    optimizer =
         torch.optim.Adam(neural_net.parameters(),
         lr=0.05)
3    scheduler =
         torch.optim.lr_scheduler.MultiStepLR(optimizer,
         milestones=lr_boundaries, gamma=0.1) # Check
         this
```

Training the nets simultaneously

And tweaking LRs

```
for train_step in range(training_steps):
    X,p_,g_tau = x, p, p[:,:,n-1]
    state = torch.cat((X,p_),axis = 1)
    loss = torch.tensor(np.zeros(1)).to(device)
    for n in range(N-2, -1, -1):
        net_n = neural_net(state[:,:,n]) # Check
            this
        F_n   = torch.sigmoid(net_n)
        loss -= torch.mean(p_[:, :, n] * F_n +
            g_tau * (1. - F_n))
        g_tau = torch.where(net_n > 0, p_[:, :,
            n], g_tau)
    px_mean_batch = torch.mean(g_tau)
    loss = torch.mean(loss)
    ...
```

Applications

Multi-dimensional Bermudan option

Driven by:

$$S_t^i = S_0^i exp([r - \delta_i - \sigma_i^2/2] \cdot t + \sigma_i W_t^i), \quad i = 1, 2, \ldots, d,$$

where S_0^i are the initial values, r is the interest rate, δ_i are the dividend yields and σ_i are the volatilities and the Brownian motions are correlated with $\rho_{ij} = \rho$.

Call payoff:

$$P = \max\left(\max_{1\leq i\leq d} S_t^i - K, 0\right) \equiv \left(\max_{1\leq i\leq d} S_t^i - K\right)^+$$

Since it can be exercised at any point $0 = t_0 < t_1 < \cdots < t_N$,

$$\sup_\tau \mathbb{E}\left[e^{-r\tau}\left(\max_{1\leq i\leq d} S_t^i - K\right)^+\right],$$

where the supremum is over all stopping times taking values in $\{t_0, t_1, \ldots, t_N\}$.

Applications

Multi-dimensional Bermudan option

Denoting $X_n^i = S_{t_n}^i$, $n = 0, 1, \ldots, N$ and $\mathcal{T}$ be the set of X-stopping times, the price can be written as $\sup_{\tau \in \mathcal{T}} \mathbb{E} g(\tau, X_\tau)$ for:

$$g(n, x) = e^{-rt_n} \left(\max_{1 \leq i \leq d} x^i - K \right)^+ .$$

Multi-dimensional Bermudan option

Symmetrical case

$S_0^i = S_0$, $\delta_i = \delta = 10\%$, $\rho = 0$, $\sigma_i = \sigma = 0.2$, $K = 100$, $T = 3$ and $N = 9$.

Summary results against Becker et al, Andersen and Brodie ($d = 2, 3$) and Broadie and Cao ($d = 5$) (binomial lattice method).

d	S_0	B's point estimate	B's 95% CI	Point estimate	95% CI
2	90	8.074	[8.060, 8.081]	8.078	[7.964, 8.193]
2	100	13.899	[13.880, 13.910]	14.215	[14.083, 14.347]
2	110	21.349	[21.336, 21.354]	20.443	[20.301, 20.584]
3	90	11.287	[11.276, 11.290]	11.075	[10.967, 11.183]
3	100	18.690	[18.673, 18.699]	19.998	[19.831, 20.165]
3	110	27.573	[27.545,27.591]	28.318	[28.158, 28.478]
5	90	16.644	[16.633, 16.648]	16.224	[16.075, 16.373]
5	100	26.159	[26.138, 26.174]	26.529	[26.351, 26.708]
5	110	36.772	[36.745, 36.789]	37.93	[37.719, 38.145]

Fast computation

For $d < 100$, our computation time is less than a minute.

For up until $d = 500$, our computation time is at maximum 278 seconds.

Q&A

Thank You!

I will now be taking questions.