
بسم الله الرحمن الرحیم

Optimal Stopping Of Exotic American Option By Deep Learning

«ارائه ریاضی مالی»

نفیسه ابراهیمی
محمدمعین شیرزاد

اساتید درس:

دکتر فتوحی، دکتر آسا، دکتر صلواتی و دکتر کاظمی

صورت مسئله

یک مسأله‌ی توقف بهینه (optimal stopping) شامل استفاده از متغیرهای تصادفی که به صورت متوالی مشاهده کرده‌ایم برای انتخاب یک زمان، به منظور اقدام برای انجام یک حرکت برای ماکسیمم کردن پاداش یا به صورت معادل، مینیمم کردن ضرر می باشد.
در این مقاله، این دنباله از متغیرهای تصادفی را با $X = (X_n)_{n=0}^{N}$ نشان می‌دهیم که یک فرآیند مارکف زمان گسسته که هر حالت آن، خود عضوی از $\mathbb{R}^d$ مقدار روی فضای احتمالاتی پالایش شده $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n=0}^{N}, \mathbb{P})$ است.

هدف

هدف ما توسعه دادن یک روش یادگیری عمیق می باشد به صورتی که بتواند به صورت مناسب و کارایی، سیاست بهینه (optimal policy) را، برای مسائل توقف به صورت زیر

$$\sup_{\tau \in T} \mathbb{E}[g(\tau, X_\tau)]$$

که در آن، $g : \{0, 1, \ldots, N\} \times \mathbb{R}^d \to \mathbb{R}$ توابع اندازه پذیر هستند و $\mathcal{T}$ نشان دهنده ی مجموعه ی تمام $X - stoppingtime$ها می باشد. برای این که اطمینان داشته باشیم که مسئله ی مذکور خوش تعریف است و دارای جواب بهینه می باشد، فرض می کنیم که تابع g انتگرال پذیر است و در شرط زیر (شرط اتگرال پذیری) صدق می کند.

$$\mathbb{E}[g(n, X_n)] < \infty \qquad \text{برای همه ی} \qquad n \in \{0, 1, \ldots, N\}$$

زمان توقف

تعریف زمان توقف

متغیر تصادفی $\tau : \Omega \to \{0, 1, \ldots, N\}$ را که مقادیر $\{0, 1, \ldots, N-1\}$ را به خود می گیرد، یک زمان توقف می گویم اگر برای هر $n \in \{0, 1, \ldots, N\}$، داشته باشیم $\{\tau = n\} \in \mathcal{F}_n$.

زمان توقف

تعریف زمان توقف

اگر بخواهیم در این چارچوب به صورت ریاضی بیان کنیم، می گوییم:
τ یک $X - stoppingtime$ است اگر
$\{\tau = n\} \in \mathcal{F}_n$، $\forall n \in \{0, 1, \ldots, N\}$.
و مفهوم پاداشی که به دنبال ماکسیمم کردن آن هستیم، به صورت زیر می باشد:

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

که در آن T مجموعه ی تمام $X - stoppingtime$ می باشد
و $g : \{0, 1, \ldots, N\} \times \mathbb{R}^d \to \mathbb{R}$ تابعی اندازه پذیر و انتگرال پذیر می باشد.

صورت بندی ریاضی مسأله توقف بهینه

۱. فرآیند تصادفی: فرآیند تصادفی مارکف X که زمان گسسته با افق متناهی می باشد به صورتی که

$$X = (X_n)_{n=0}^{N} \,, \qquad X_n \in \mathbb{R}^d \,, \qquad \forall n \in \{0, 1, \dots, N\}$$

۲. تابع پاداش یا عایدی (pay off) :

$$g(t, X_1, \dots, X_t) \xrightarrow{\text{با توجه به مارکف بودن فرآیند}} g(\tau, X_\tau)$$

۳. هدف (objective) :

$$\sup_{\tau \in T} \mathbb{E}[g(\tau, X_\tau)]$$

۴. استراتژی های مجاز: در استراتژی های مجاز، نباید از اطلاعات آینده برای تصمیم گیری استفاده کرد.

بیان کردن زمان های توقف به صورت دنباله ای از تصمیمات توقف

اولین کاری که باید انجام بدهیم این است که نشان دهیم تصمیم به توقف فرآیند مارکف را $(X_n)_{n=0}^{N}$ می توان بر اساس دنباله ای از توابع $f_n : \mathbb{R}^d \to \{0, 1\}$, $\{f_n(X_n)\}_{n=0}^{N}$ گرفت. برا ی انجام این کار، در ابتدا، مسأله ی کلی توقف بهینه ی خود در (۱) را به دنباله ای از مسائل توقف تقسیم می کنیم. به طور خاص، برای زمان n، مسئله ی توقف را به صورت زیر در نظر بگیرید:

$$V_n = \sup_{\tau \in T_n} \mathbb{E}[g(\tau, X_\tau)] \qquad (۱)$$

که در آن، T_n مجموعه ی تمام $X - stoppingtime$ ها به صورتی که $N \leq \tau \leq n$ می باشد، است.

بیان کردن زمان های توقف به صورت دنباله ای از تصمیمات توقف

به وضوح، از آن جایی که فرآیند باید در زمان N توقف کند، $\mathcal{T}_N = \{N\}$ و بنابراین زمان N، زمان توقف بهینه می باشد و $\tau_N = N$. علاوه بر این، ازآن جایی که $\{f_n(X_n)\}_{n=0}^{N}$ دنباله ی صفر و یکی تصمیمات توقف ما می باشد، باید ذکر شود که $f_N \equiv 1$ و لذا می توان به این صورت $\tau_N = Nf_N(X_N)$ نوشت.

بیان کردن زمان های توقف به صورت دنباله ای از تصمیمات توقف

حال که زمان توقف زمان N را به صورت تابعی از تصمیمات توقف صفر و یکی نوشتیم، قصد داریم این کار را برایn هایی که $0 \leq n \leq N-1$ نیز انجام دهیم. به این معنی که قصد داریم زمان توقف n، τ_n، را به صورت تابعی از $\{f_k(X_k)\}_{k=n}^{N}$ بنویسیم. قصد داریم با استفاده از معادله زیر انجام دهیم:

$$\tau_n = \sum_{k=n}^{N} k f_k(X_k) \prod_{j=n}^{k-1} (1 - f_j(X_j)) \qquad (۲)$$

که در آن $f_N \equiv 1$ می باشد. معادله فوق یک زمان توقف درτ_n را تعیین می کند.

لم Doob-Dynkin

در اثبات قضیه ی یک از یک لم استفاده می شود که در ابتدا به بیان آن لم می پردازیم و سپس به سراغ اثبات قضیه ی یک می رویم. نام لم پایین، لم Doob-Dynkin می باشد که دارای نسخه های متعدد متفاوتی می باشد.
ایده ی اصلی لم، به طور کلی، لم Doob-Dynkin ، شرایطی روی دو تابع X و Y در نظر می گیرد به صورتی که این شرایط، وجود یک تابع مانند ϕ به صورتی که $X = \phi \circ Y$ است را تضمین می کند.

$$\begin{tikzcd}
 & \Omega_X \\
\Omega \arrow[ur, "X"] \arrow[dr, "Y"'] & \\
 & \Omega_Y \arrow[uu, "\phi"']
\end{tikzcd}$$

شکل: شمای کلی لم Doob-Dynkin

قضیه (۱)

برای یک $n \in \{0, \dots, N-1\}$، فرض کنید τ_{n+1} یک زمان توقف در $\mathcal{T}_{n+1}$ $(\tau_{n+1} \in \mathcal{T}_{n+1})$ به فرم زیر:

$$\tau_{n+1} = \sum_{k=n+1}^{N} k f_k(X_k) \prod_{j=n+1}^{k-1} (1 - f_j(X_j)) \qquad (۳)$$

برای توابع اندازه پذیر $f_{n+1}, \dots, f_N : \mathbb{R}^d \to \{0, 1\}$ با $f_N \equiv 1$ باشد. آن گاه یک تابع اندازه پذیر $f_n : \mathbb{R}^d \to \{0, 1\}$ به صورتی که زمان توقف داده شده به فرم (۳) $\tau_n \in \mathcal{T}_n$ وجود دارد که شرط زیر را ارضا می کند:

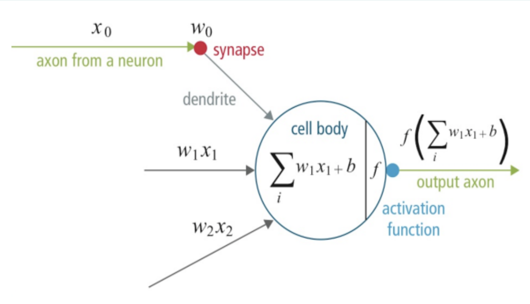
$$\mathbb{E}[g(\tau_n, X_n)] \geq V_n - (V_{n+1} - \mathbb{E}[g(\tau_{n+1}, X_{n+1})]), \qquad (۴)$$

که در آن V_n و V_{n+1} مقادیر بهینه تعریف شده در (۲) می باشد.

شبکه عصبی

انگیزه ی اولیه برای استفاده از رویکرد ماشین لرنینگ در مسائل توقف بهینه، توانایی آن ها برای (curse of dimensionality) غلبه بر نفرین بعد می باشد که روش های عددی مرسوم با آن مواجه بودند. شبکه ی عصبی یک سیستم محاسباتی می باشد که به صورت ساده سازی شده بر اساس مغز انسان طراحی شده است. بدون عمیق شدن در جزئیات زیستی، ایده اصلی این است که یک نرون ورودی را از نرون های دیگر یا از یک منبع خارجی دریافت می کند و از آن برای تولید خروجی استفاده می کند. این خروجی، می تواند به عنوان ورودی به نرون بعدی داده شود و یا به عنوان خروجی کل سیستم در نظر گرفته شود. در شبکه عصبی محاسباتی، به این نرون ها، نودهای شبکه عصبی گفته می شود.
روشی که در آن یک نود ورودی را دریافت می کند و آن را به خروجی تبدیل می کند، در شکل (۱) نشان داده شده است.

شبکه عصبی



شکل: یک نود در شبکه عصبی

معرفی یک شبکه عصبی

حال نشان دادیم که زمان توقف بهینه را می توان به صورت تابعی از دنباله ی تصمیمات توقف صفر و یکی، $\{f_n\}_{n=0}^{N}$، محاسبه کرد، نیاز دارم تا راهی برای تقریب زدن تابع نام برده بیابیم. روشی که می خواهیم از آن استفاده کنیم، با به کار گیری از یک شبکه ی عصبی می باشد. به طور خاص، دنباله ای از شبکه های عصبی به فرم $f^{\theta_n} : \mathbb{R}^d \to \{0, 1\}$ با پارامتر $\theta_n \in \mathbb{R}^q$ برای تقریب زدن f_n می سازیم. سپس، به سادگی می توانیم τ_{n+1} را با

$$\sum_{k=n}^{N} k f^{\theta_k}(X_k) \prod_{j=n}^{k-1} \left(1 - f^{\theta_j}(X_j)\right) \tag{۵}$$

تقریب بزنیم.

معرفی یک شبکه عصبی

پس اگر بخواهیم به صورت دقیق تر بیان کنیم:
روش عددی ما برای حل مسئله ی توقف بهینه بر پایه به صورت تکرار شونده (itereatively) تقریب زدن تصمیمات توقف بهینه $f_n : \mathbb{R}^d \to \{0, 1\}, n = 0, 1, \dots, N-1$ به وسیله ی یک شبکه عصبی $f^\theta : \mathbb{R}^d \to \{0, 1\}$ با پارامتر $\theta \in \mathbb{R}^q$ می باشد. این کار را به این صورت انجام می دهیم که تصمیم توقف نهایی را به صورت $f_N \equiv 1$ در نظر می گیریم و با اعمال استقرای بازگشتی (backward induction) شروع می کنیم.

معرفی یک شبکه عصبی

به طور دقیق تر، قرار دهید $n \in \{0, 1, \ldots, N-1\}$ و فرض کنید مقادیر پارامتر ها $\theta_{n+1}, \theta_{n+2}, \ldots, \theta_N \in \mathbb{R}^q$ به صورتی در نظر گرفته شده اند که $f^{\theta_N} \equiv 1$ و زمان توقف

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

یک امید ریاضی $\mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})]$ نزدیک به مقدار بهینه V_{n+1} تولید می کند.

معرفی یک شبکه عصبی

اولین مشکلی که با آن مواجه هستیم، این است که خروجی تابع f^{θ_n}، فقط دارای مقادیری در $\{0, 1\}$ می باشد. و در نتیجه با توجه به θ_n، پیوسته نمی باشد. این یک مشکل جدی می باشد چرا که در آموزش پارامترها برای یک شبکه عصبی، ما اغلب روش های بهینه سازی مبتنی بر گرادیان (gradient-based) را برای ماکسیمم کردن تابع پاداش (یا معادلاً مینیمم کردن ضرر) را به کار می بریم. همان طور که گفتیم، از آن جایی که f^{θ} مقادیری از $\{0, 1\}$ اخذ می کند، پس نمی توانیم به صورت مستقیم از روش های بهینه سازی میتنی بر گرادیان استفاده کنیم.

معرفی یک شبکه عصبی

برای حل این مشکل، پارامتر ها را با یک شبکه عصبی چند لایه پیشخور آموزش می دهیم که خروجی های آن به صورت احتمال هایی که در بازه ی $(0,1)$ می باشند. سپس بعد از آموزش، می توانیم آن ها را به تصمیمات توقف صفر و یکی تبدیل کنیم. یعنی، برای $\theta \in \{\theta_0, \ldots, \theta_N\}$ ، $F^\theta : \mathbb{R}^d \to (0,1)$ را معرفی می کنیم که یک شبکه عصبی به فرم زیر می باشد:

$$F^\theta = \psi \circ a_I^\theta \circ \varphi_{q_{I-1}} \circ a_{I-1}^\theta \circ \cdots \circ \varphi_{q_1} \circ a_1^\theta$$

معرفی یک شبکه عصبی

$$F^{\theta} = \psi \circ a_l^{\theta} \circ \varphi_{q_{l-1}} \circ a_{l-1}^{\theta} \circ \cdots \circ \varphi_{q_1} \circ a_1^{\theta}$$

که در آن

- $l, q_1, q_2, \ldots, q_{l-1}$ اعداد صحیح مثبت هستند و عمق شبکه ی عصبی و تعداد نود های موجود در هر لایه ی نهان شبکه عصبی را نشان می دهند. (در صورت وجود لایه نهان)
- $a_l^{\theta} : \mathbb{R}^{q_{l-1}} \to \mathbb{R}$ و $a_1^{\theta} : \mathbb{R}^{d} \to \mathbb{R}^{q_1}, \ldots, a_{l-1}^{\theta} : \mathbb{R}^{q_{l-2}} \to \mathbb{R}^{q_{l-1}}$ توابع آفین (affine funtions) یا همان توبع تبدیلات خطی هستند.

معرفی یک شبکه عصبی

- برای $j \in \mathbb{N}$، $\varphi_j : \mathbb{R}^j \to \mathbb{R}^j$ یک تابع فعال ساز ReLU می باشد که به صورت جزء به جزء (component-wise) اعمال می شود و به صورت زیر است:

$$\varphi_j (x_1, \ldots, x_j) = \left(x_1^+, \ldots, x_j^+\right)$$

- $\psi : \mathbb{R} \to (0, 1)$ تابع لوجستیک استاندارد می باشد و به صورت زیر است:

$$\psi(x) = e^x / (1 + e^x) = 1 / \left(1 + e^{-x}\right)$$

معرفی یک شبکه عصبی

هدف ما تعیین کردن $\theta_n \in \mathbb{R}^q$ می باشد به صورتی که

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

مقداری نزدیک به
$\sup_{\theta \in \mathbb{R}^q} \mathbb{E}\left[g\left(n, X_n\right) F^{\theta}\left(X_n\right) + g\left(\tau_{n+1}, X_{\tau_{n+1}}\right)\left(1 - F^{\theta}\left(X_n\right)\right)\right]$
باشد. وقتی که به این هدف رسیدیم، تابع $f^{\theta_n} : \mathbb{R}^d \to \{0, 1\}$ را به صورت زیر تعریف می کنیم:

$$f^{\theta_n} = 1_{[0,\infty)} \circ a_I^{\theta_n} \circ \varphi_{q_{I-1}} \circ a_{I-1}^{\theta_n} \circ \cdots \circ \varphi_{q_1} \circ a_1^{\theta_n}, \qquad (۶)$$

که در آن، $1_{[0,\infty)} : \mathbb{R} \to \{0, 1\}$ تابع مشخصه (indicator function) از $[0, \infty)$ می باشد.

معرفی یک شبکه عصبی

توجه شود که تنها تفاوتی که F^{θ_n} با f^{θ_n} دارد، غیرخطی بودن نهایی است.
توجه: توابع F^{θ_n} و f^{θ_n}:

$$F^{\theta} = \psi \circ a_l^{\theta} \circ \varphi_{q_{l-1}} \circ a_{l-1}^{\theta} \circ \cdots \circ \varphi_{q_1} \circ a_1^{\theta}$$
$$f^{\theta_n} = 1_{[0,\infty)} \circ a_l^{\theta_n} \circ \varphi_{q_{l-1}} \circ a_{l-1}^{\theta_n} \circ \cdots \circ \varphi_{q_1} \circ a_1^{\theta_n}$$

F^{θ_n} یک احتمال توقف در بازه ی $(0,1)$ تولید می کند در صورتی که f^{θ_n}، یک تصمیم توقف مشخص دقیق صفر یا یکی با توجه به این که f^{θ_n} دارای مقدار زیر $1/2$ باشد یا بالای آن، به ما می دهد.

معرفی یک شبکه عصبی

حال که یک نسخه ی هموار از تصمیمات توقف تعریف کرده ایم و یک روش نیز، برای تبدیل خروجی های آن به تصمیمات توقف صفر و یکی معرفی کرده ایم، باید تابع پاداش (reward function) یا ضرر (loss function) برای میزان (درست) کردن (tune) پارامتر های θ_n تعریف کنیم. در هر گام، هدف ما ماکسیمم کردن امید ریاضی پاداش آینده می باشد. ما در زمان n می دانیم که اگر توقف کنیم $(f_n(X_n) = 1)$و پاداشی به مقدار $g(n, X_n)$ دریافت می کنیم و اگر ادامه دهیم $(f_n(X_n) = 0)$ و پس از آن نیز به رفتار بهینه ی خود ادامه دهیم، در نهایت پاداشی به مقدار $g\left(\tau_{n+1}, X_{\tau_{n+1}}\right)$ دریافت خواهیم کرد. (با توجه به قضیه ی (۱)) لذا در هر گام زمانی، به دنبال پیدا کردن تابع $f: \mathbb{R}^d \to \{0, 1\}$ می باشیم که عبارت زیر را ماکسیمم کند:

$$\mathbb{E}\left[g(n, X_n) f(X_n) + g\left(\tau_{n+1}, X_{\tau_{n+1}}\right)(1 - f(X_n))\right] \qquad (۷)$$

قضیه (۲)

قضیه ای که در ادامه خواهیم دید، اثبات می کند که ما می توانیم f در معادله ی (۱۰) را با f^θ جایگزین کنیم. این کار به ما این اجازه را می دهد که به جای پیدا کردن تابع بهینه $f : \mathbb{R}^d \to \{0, 1\}$، معادله ی (۱۰) را با توجه به $\theta \in \mathbb{R}^q$ ماکسیمم کنیم.

قضیه (۲)

قرار دهید $n \in \{0, 1, \dots, N-1\}$ و یک زمان توقف مانند $\tau_{n+1} \in \mathcal{T}_{n+1}$ را فیکس در نظر بگیرید. آن گاه، برای هر عمق $I \geq 2$ و ثابت $\varepsilon > 0$، اعداد صحیح مثبت $q_1, \dots, q_{I-1}$ وجود دارند به صورتی که

$$\begin{aligned} &\sup_{\theta \in \mathbb{R}^q} \mathbb{E}\left[g(n, X_n)\, f^\theta(X_n) + g\left(\tau_{n+1}, X_{\tau_{n+1}}\right)\left(1 - f^\theta(X_n)\right)\right] \\ &\geq \sup_{f \in \mathcal{D}} \mathbb{E}\left[g(n, X_n)\, f(X_n) + g\left(\tau_{n+1}, X_{\tau_{n+1}}\right)\left(1 - f(X_n)\right)\right] - \varepsilon. \end{aligned} \qquad (۸)$$

که در آن، $\mathcal{D}$ مجموعه ی تمام توابع اندازه پذیر $f : \mathbb{R}^d \to \{0, 1\}$ می باشد.

قرارداد های اختیار متعارف و غیرمتعارف

قرارداد های اختیار نامتعارف(exotic option) :
قرارداد های اختیار نامتعارف، دسته ای از قرارداد های اختیار هستند که با قرارداد های اختیار مرسوم (قرارداد های استاندارد اروپایی و آمریکایی)، در شرایط فعال سازی، ساختار عایدی، تاریخ انقضاء و قیمت توافقی تفاوت دارند. به بیان دیگر، قرارداد های اختیار نامتعارف، همان قرارداد های اختیار اروپایی و آمریکایی هستند اما در آن ها شرط یا شرایط جدیدی لحاظ می شود که موجب رونق بیشتر این قرارداد های اختیار در خرید و فروش ها می شود. به سادگی، یک قرارداد اختیار نامتعارف، هر نوع قرارداد اختیاری غیر از قرارداد های اختیار متعارف در مبادلات عمده است. قرارداد های اختیار آسیایی و برمودا از این دسته قرارداد های اختیار هستند.

قرارداد های اختیار برمودا و آسیایی

تعریف قرارداد اختیار برمودا (Bermudan Max-Call Option) :

قرارداد اختیار برمودا که در زمان T اجرا می شود، قرارداد اختیاری می باشد که روی d دارایی نوشته می شود، $\{X^i\}_{i=1}^d$، که به دارانده ی آن این اجازه را می دهد تا یکی از این دارایی ها را با قیمت توافقی K در هر مقطع زمانی در $T = t_N < \ldots < t_1 < t_0 = 0$ خریداری کند. ما فرض می کنیم که در یک بازار با مدل بلک–شولز هستیم. عایدی این نوع آپشن به صورت زیر می باشد:

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

که در آن، سوپریمم روی تمام $S - stoppingtime$ها که مقادیر آن ها $t_0, t_1, \ldots, t_N$ می باشد.

قرارداد های اختیار برمودا و آسیایی

تعریف قرارداد اختیار آسیایی (Asian option) :

قرارداد اختیار آسیایی، اولین بار در سال ۱۹۸۷ در (Tokyo) معامله شد و به خصوص، در تجارت کالا ها بسیار محبوبیت دارد. در قرارداد اختیار آسیایی، (برخلاف قرارداد های اختیار اروپایی و آمریکایی که عایدی فقط وابسته به قیمت دارایی پایه در یک مقطع زمانی خاص می باشد) عایدی وابسته به میانگین قیمت دارایی پایه در یک بازه ی زمانی خاص می باشد.

دو نوع قرارداد اختیار آسیایی داریم: قیمت توافقی ثابت (fixed strike):در این نوع آپشن، قیمت میانگین به جای قیمت دارایی پایه استفاده می شود.

قیمت ثابت (fixed price):در این نوع آپشن، میانگین قیمت به جای قیمت توافقی استفاده می شود.

قرارداد اختیار آسیایی

عایدی حاصل از قرارداد فروش آسیایی قیمت توافقی ثابت به صورت زیر می باشد:

$$C(T) = \max(A(0, T) - K, 0)$$

کدها

+ Code + Text Connect

```python
# -*- coding: utf-8 -*-
"""BS_PO_NN_LB.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1nQiRpU6zKZ0YuebE4nzPz8EARZFdtKeh
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
from math import ceil, sqrt
import copy

"""## Black & Scholes model"""

class MultiDimensionalBS:
    def __init__(self, initial_prices, interest_rate, dividends, volatilities, correlation_matrix):
        self.initial_prices = initial_prices
        self.interest_rate = interest_rate
        self.dividends = dividends
        self.volatilities = volatilities
        self.correlation_matrix = correlation_matrix
        self.dimension = len(initial_prices)
        self.var_cov = np.linalg.cholesky(np.matmul(np.diag(self.volatilities), np.matmul(self.correlation_matrix, np.diag(self.volatilities))))

        assert self.dimension == len(self.dividends) and self.dimension == len(self.volatilities) and self.dimension == len(self.var_cov[0]) and self.dimension == le

    def Simulate(self, expiry, nb_time_discr, number_paths):
        """ Simulate stock prices under BS model until expiry and with constant timestep """
```

کدها

+ Code + Text

Connect

```python
    def Simulate(self, expiry, nb_time_discr, number_paths):
        """ Simulate stock prices under BS model until expiry and with constant timestep """
        diffusion_list = np.linspace(0, expiry, nb_time_discr + 1)
        last_index = len(diffusion_list) - 1
        delta_time = expiry / last_index
        dW = np.sqrt(delta_time) * np.random.randn(number_paths, last_index, self.dimension)
        dW = np.append(np.zeros((number_paths, 1, self.dimension)), dW, axis=1)
        cst = self.interest_rate - self.dividends - 0.5 * (self.volatilities ** 2)
        W = np.cumsum(np.matmul(dW, self.var_cov), axis = 1)
        S = self.initial_prices * np.exp(cst * diffusion_list[np.newaxis, :, np.newaxis] + W)

        return S

    def Simulate_Next_State(self, current_price, current_time, diffusion_times, nb_paths):
        """ Simulate conditionnal stock prices under BS model until expiry and with constant timestep """
        delta_time = diffusion_times[0] - current_time
        dW = np.sqrt(delta_time) * np.random.randn(nb_paths, len(diffusion_times), self.dimension)
        cst = self.interest_rate - self.dividends - 0.5 * (self.volatilities ** 2)
        W = np.cumsum(np.matmul(dW, self.var_cov), axis = 1)
        S = current_price[:, np.newaxis, np.newaxis, :] * np.exp(cst * (diffusion_times[np.newaxis, :, np.newaxis] - current_time) + W)

        return S

"""## Payoffs Management"""

class Payoff:
    def __init__(self, strike, option_type):
        self.strike = strike
        self.option_type = option_type

    def value(self, asset_prices, interest_rate, dates, k=0):
        if (self.option_type == 0):
            # max call
```

کدها

```
            # max call
            return np.exp(-interest_rate * dates[np.newaxis, k:]) * np.maximum(asset_prices.max(axis = 2) - self.strike, 0)

        elif (self.option_type == 1):
            # max put
            return np.exp(-interest_rate * dates[np.newaxis, k:]) * np.maximum(self.strike - asset_prices.min(axis = 2), 0)

        elif (self.option_type == 2):
            d = len(asset_prices[0][0])
            # geometric call
            np.exp(-interest_rate * dates[np.newaxis, k:]) * np.maximum(asset_prices.prod(axis = 2)**(1/d) - self.strike, 0)

"""## Neural Net Training"""

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

def Build_Neural_Network(nn_structure):

    nb_hidden_layers = len(nn_structure) - 2
    assert nb_hidden_layers >= 2, " A neural network has a least two layers : input and output "

    layers = []

    for i in range(0, nb_hidden_layers):
        layers += [nn.Linear(nn_structure[i], nn_structure[i + 1]), nn.BatchNorm1d(nn_structure[i + 1]), nn.ReLU()]

    layers += [torch.nn.Linear(nn_structure[nb_hidden_layers], nn_structure[nb_hidden_layers + 1]), torch.nn.Sigmoid()]

    return nn.Sequential(*layers)

def model_train(nn_structure, nn_hyper_params, model_params, nb_epoch_before_display, re_use_nn_params = False):

```

کدها

+ Code + Text Connect

```python
def model_train(nn_structure, nn_hyper_params, model_params, nb_epoch_before_display, re_use_nn_params = False):

    # nn_structure : list containing specification of hidden layer :
    # nn_hyper_params : dict containing nn hyperparameters
    # model_params : dict containing BS model parameters
    # re_use_nn_params : if True : we previously trained nn to initialize current nn weights

    ##################################################################################################################
    ####################################################                  MODELS INIT

    N = len(model_params['diffusion_times']) - 1
    nb_batches = ceil(nn_hyper_params['inputs_size'] / nn_hyper_params['batch_size'])

    nn_base = Build_Neural_Network(nn_structure)
    f = [copy.deepcopy(nn_base).to(device) for i in range(N - 1)]
    optimizers = [torch.optim.Adam(f[i].parameters(), lr = nn_hyper_params['lr']) for i in range(N - 1)]
    BS = MultiDimensionalBS(model_params['initial_prices'], model_params['interest_rate'], model_params['dividends'], model_params['volatilities'], model_params['cor
    losses = np.array([[0.0] * (N - 1)] * (nn_hyper_params['nb_epochs'] // nb_epoch_before_display))

    comp = 0

    ##################################################################################################################
    ####################################################                  TRAINING

    for epoch in range(nn_hyper_params['nb_epochs']):
        S_train = BS.Simulate(model_params['diffusion_times'][-1], N, nn_hyper_params['inputs_size'])
        x = torch.from_numpy(S_train).float()
        y = torch.unsqueeze(torch.from_numpy(my_payoff.value(S_train, model_params['interest_rate'], model_params['diffusion_times'])), dim=2).float()
        payoff_tau = [y[i * nn_hyper_params['batch_size'] : min((i + 1) * nn_hyper_params['batch_size'], nn_hyper_params['inputs_size']) , N].to(device) for i in rar
        this_epoch_loss = np.full(N - 1, 0.0)

        # bakward induction
```

کدها

```python
"""## Test"""

##### BS parameters
nb_assets = 2
expiry = 3
nb_ex_dates = 9
strike = 100
my_payoff = Payoff(strike, 0)

# NN parameters
nb_hidden_layers = 3
nn_units = [nb_assets + 1] + [nb_assets + 40 for i in range(nb_hidden_layers - 1)] + [1]
nb_epoch_before_display = 100

nn_hp = {'lr' : 0.001, 'nb_epochs' : 1000, 'batch_size' : 8192, 'inputs_size' : 8192*1}

BS_params = {'initial_prices' : np.full(nb_assets, 100),
             'interest_rate' : 0.05,
            'dividends' : np.full(nb_assets, 0.1),
            'volatilities' : np.full(nb_assets, 0.2),
            'diffusion_times' : np.linspace(0, expiry, nb_ex_dates + 1),
            'correlation_matrix' : np.eye(nb_assets)}

my_models, losses = model_train(nn_units, nn_hp, BS_params, nb_epoch_before_display)

epoch = np.array([i * nb_epoch_before_display for i in range(nn_hp['nb_epochs'] // nb_epoch_before_display)])
plt.figure(figsize=(15,8))
for i in range(len(losses[0])):
    plt.plot(epoch, losses[:,i])
plt.xlabel("epoch")
plt.ylabel("loss")

"""# # Lower Bound"""
```

کدها

```
    # bakward induction

    for n in range(N - 1, 0, -1):
        inputs = torch.cat((x[:,n], y[:,n]), 1)

        if n < N - 2 and re_use_nn_params:
            f[n].parameters = f[n + 1].parameters

        for batch in range(nb_batches):
            batch_range = [i for i in range(batch * nn_hyper_params['batch_size'], min((batch + 1) * nn_hyper_params['batch_size'], nn_hyper_params['inputs_size'
            this_batch_inputs = inputs[batch_range, :].to(device)
            optimizers[n - 1].zero_grad()
            F_n = f[n - 1](this_batch_inputs)
            f_n = (F_n >= 0.5).detach().float()
            payoffs_batch = y[batch_range , n].to(device)
            payoff_tau[batch] = payoffs_batch * f_n + payoff_tau[batch] * (1 - f_n)
            z = torch.mean(torch.mul(torch.subtract(payoff_tau[batch], payoffs_batch), F_n))
            this_epoch_loss[n - 1] += z.item() / float(nb_batches)
            z.backward()
            optimizers[n - 1].step()

    if(epoch % nb_epoch_before_display == 0):
        losses[comp] = this_epoch_loss
        print('epoch {}, losses : {}'.format(epoch, losses[comp]))
        print('\n')
        comp += 1

  return f, losses

"""## Test"""

##### BS parameters
nb_assets = 2
expiry = 3
```

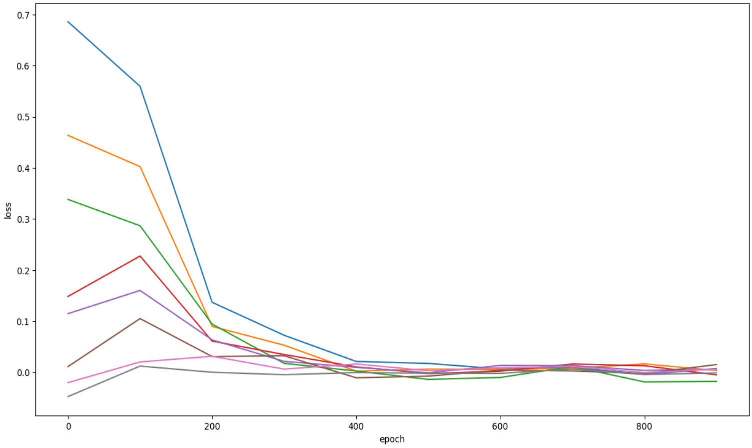
کدها

+ Code + Text

Connect

```python
"""# # Lower Bound"""

K_L = 1096000
BS_lower = MultiDimensionalBS(BS_params['initial_prices'], BS_params['interest_rate'], BS_params['dividends'], BS_params['volatilities'], BS_params['correlation_matr
S_lower = BS_lower.Simulate(expiry, nb_ex_dates, K_L)
x = torch.from_numpy(S_lower).float()
y = torch.unsqueeze(torch.from_numpy(my_payoff.value(S_lower, BS_params['interest_rate'], BS_params['diffusion_times'])), dim=2).float()

# init
tau = y[:,nb_ex_dates].new_full(y[:,nb_ex_dates].shape, nb_ex_dates).to(device)
payoff_tau = y[:,nb_ex_dates].to(device)

# backward induction
for n in range(nb_ex_dates - 1, 0, -1):
    inputs = torch.cat((x[:,n], y[:,n]), 1)
    f_n = (my_models[n - 1](inputs.to(device)) >= 0.5).detach().float()
    tau = n * f_n + tau * (1 - f_n)
    payoffs = y[:, n].to(device)
    payoff_tau = payoffs * f_n + payoff_tau * (1 - f_n)
    print(n, 'reward :', payoff_tau.mean())

# initial date
payoffs = y[:,0].to(device)
f_n = (payoffs >= (payoff_tau).mean()).detach().float()
tau *= (1 - f_n)
payoff_tau = payoffs * f_n + payoff_tau * (1 - f_n)

lower = payoff_tau
lower_bound = lower.mean().item()
print(0, 'reward :', payoff_tau.mean())

print('tau :', round(tau.mean().item(), 3), 'lower bound :', round(lower_bound, 3))
```

نتایج



«با تشکر از توجه شما»