بسم الله الرحمن الرحیم

دانشکده ریاضی و علوم کامپیوتر

حل مساله حمل و نقل بهینه و مسائل مربوط به پوشش ریسک از طریق جریمه‌سازی و شبکه‌های عصبی عمیق

اساتید:

دکتر مرتضی فتوحی – دکتر هیربد آسا

ارائه‌دهندگان:

فرزانه حسینی – محمد سوری

تابستان ۱۴۰۲

فهرست مطالب

در این ارائه، ابتدا مساله‌ی حمل و نقل و کاربردهای آن در مسائل احتمالاتی و مالی را مختصر توضیحی خواهیم داد. سپس به ادعاهای مقاله مورد بررسی برای نمایش همگرایی روش عددی به مساله دوگان این مساله خواهیم پرداخت. سپس نشان خواهیم داد این روش عددی را می‌توان با شبکه‌های عصبی عمیق مدلسازی کرد. در پایان به عنوان نمونه، یک مساله از بهینه‌سازی سبد سهام تحت عدم قطعیت وابستگی دارایی‌ها و نتایج و شبیه‌سازی‌های حاصل از پیاده‌سازی آن را خواهیم دید و در پایان به مسائل دیگری که به همین روش قابل حل خواهند بود اشاره مختصری می‌کنیم.

مساله حمل و نقل

$$\phi(f) = \sup_{\nu \in \mathbb{Q}} \int f \, d\nu$$

دوگان

$$\phi(f) = \inf_{\substack{h \in \mathcal{H}: \\ h \geq f}} \int h d\mu_0.$$

قضیه دنیل استون غیرخطی

که در آن $\mu_0 \in \mathcal{Q}$ و $\mathcal{H}$ مجموعه‌ای از توابع پیوسته و کران دار $h : \mathcal{X} \to \mathbb{R}$ است.

فرم دوگان همان تابعک سوپرهجینگ
یا پوشش ریسک کمینه از بالاست

رویکرد مقاله برای حل مساله دوگان

$$\phi^m(f) = \inf_{\substack{h \in \mathcal{H}^m: \\ h \geq f}} \int h \, d\mu_0$$

$(\mathcal{H}^m)_{m\in\mathbb{N}}$

$\mathcal{H}^1 \subseteq \mathcal{H}^2 \subseteq \ldots \subseteq \mathcal{H}$

$\mathcal{H}^\infty := \bigcup_{m\in\mathbb{N}} \mathcal{H}^m$

یک شبکه عصبی با ساختاری مشخص و ثابت و تعداد نورون‌های لایه‌های پنهان برابر با m

قرار است بنابر تعریفی در H چگال باشد

جریمه‌سازی برای به روزرسانی پله‌ای در برقراری قید

$\beta : \mathbb{R} \to \mathbb{R}_+$

تابعی مشتق‌پذیر و
غیر نزولی

$\beta(f-h)$

min

تضمین برقراری
$h \geq f$

$$\phi^m_{\theta,\beta}(f) = \inf_{h \in \mathcal{H}^m} \left\{ \int h \, d\mu_0 + \int \beta(f-h) \, d\theta \right\}.$$

$$\phi_{\theta,\beta}(f) = \inf_{h \in \mathcal{H}} \left\{ \int h \, d\mu_0 + \int \beta(f-h) \, d\theta \right\}.$$

یک اندازه احتمال
نمونه‌گیری برای
سنجش برقراری قید

تعریف نوتیشن‌ها

$\mathcal{X}$

یک فضای لهستانی

$\mathcal{P}(\mathcal{X})$

مجموعه همه اندازه احتمال‌های بورل

$C_b(\mathcal{X})$

فضای خطی تمام توابع پیوسته و کراندار به فرم

$f : \mathcal{X} \to \mathbb{R}$

$f \in C_b(\mathcal{X})$

$\mu_0 \in \mathcal{P}(\mathcal{X})$ یک اندازه قیمت‌گذاری

$\mathcal{H} \subseteq C_b(\mathcal{X})$ یک فضای خطی تابعی شامل ثوابت

$$\phi(f) := \inf\left\{ \int h\, d\mu_0 : h \geq f \text{ for some } h \in \mathcal{H} \right\}$$

پارامتری‌سازی جریمه

بتا تابعی غیرنزولی، محدب
و مشتق‌پذیر است که

$$\lim_{x\to\infty} \beta(x)/x = \infty$$

$$\psi_{\theta,\gamma}(f) := \int \beta_\gamma(f)\, d\theta \text{ for a } \textit{sampling measure } \theta \in \mathcal{P}(\mathcal{X}).$$

$$\beta_\gamma(x) := \tfrac{1}{\gamma}\beta(\gamma x)$$

$$\gamma > 0.$$

$$\beta\colon \mathbb{R} \to \mathbb{R}_+$$

مزدوج دوگان تابع جریمه

$$\beta_\gamma^*(y) := \sup_{x\in\mathbb{R}}\{xy - \beta_\gamma(x)\} \quad \text{for all } y \in \mathbb{R}_+,$$

ادعا:

$$\beta_\gamma^*(y) = \frac{\beta^*(y)}{\gamma}$$

$$\beta_\gamma^*(y) := \sup_{x\in\mathbb{R}}\{xy - \beta_\gamma(x)\}$$

$$= \sup_{x\in\mathbb{R}}\left\{\frac{\gamma}{\gamma}(xy) - \frac{1}{\gamma}\beta(\gamma x)\right\}$$

$$= \sup_{x\in\mathbb{R}}\left\{\frac{1}{\gamma}((\gamma xy) - \beta(\gamma x))\right\}$$

$$= \sup_{x'\in\mathbb{R}}\left\{\frac{1}{\gamma}(x'y) - \beta(x')\right\}$$

$$\beta_\gamma^*(y) = \frac{\beta^*(y)}{\gamma}$$

ساختار تابع‌های پوشش و جریمه مورد استفاده در شبیه‌سازی‌ها

$$\begin{aligned}\mathcal{Q} =&\Pi(\mu_1,\dots,\mu_d)\\ \mathcal{H} =&\{h \in C_b(\mathbb{R}^d) : h(x_1,\dots,x_d) = h_1(x_1)+\cdots+h_d(x_d) \text{ برای همه‌ی } (x_1,\dots,x_d)\\ &\in \mathbb{R}^d \text{ و برخی از } h_i \in C_b(\mathbb{R})\}\end{aligned}$$

تابع جریمه L^p، $\beta(x) = \frac{1}{p}(\max\{0,x\})^p$ با مزدوج $\beta^*(y) = \frac{1}{q}y^q$ که در آن $q = \frac{p}{p-1}$

$p > 1$

روند کلی مقاله

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{matrix}\phi(f)\\ \textit{(initial problem)}\end{matrix} & \begin{matrix}\phi_{\theta,\gamma}(f)\\ \textit{(penalized version of } \phi(f))\end{matrix} \arrow[l, "\text{(Thm. 2.2)}"', "\gamma\to\infty"] \\
\begin{matrix}\phi^m(f)\\ \textit{(finite-dim. version of } \phi(f))\end{matrix} \arrow[u, "m\to\infty", "\begin{matrix}\text{(Prop. 2.3)}\\ \text{(Lemma 3.3)}\end{matrix}"'] & \begin{matrix}\phi^m_{\theta,\gamma}(f)\\ \textit{(numerically solved problem)}\end{matrix} \arrow[l, "\text{(Prop. 3.7)}"', "\gamma\to\infty"] \arrow[u, "m\to\infty", "\text{(Remark 3.5)}"'] \arrow[ul, "\text{(Prop. 2.4, Remark 3.5)}"', "\min\{\gamma,m\}\to\infty"]
\end{tikzcd}$$

قضیه همگرایی نسبت به شاخص جریمه

فرض کنید $f \in C_b(\mathcal{X})$ باشد. فرض کنید $\pi \in \mathcal{Q}$ وجود داشته باشد به طوری که $\pi \ll \theta$ و $\int \beta^* \left(\frac{d\pi}{d\theta}\right) d\theta < \infty$. در این صورت، داریم:

$$\phi_{\theta,\gamma}(f) = \max_{\mu \in \mathcal{Q}} \left\{ \int f d\mu - \frac{1}{\gamma} \int \beta^* \left(\frac{d\mu}{d\theta}\right) d\theta \right\}.$$

علاوه بر این، داریم:

$$\phi_{\theta,\gamma}(f) - \frac{\beta(0)}{\gamma} \leq \phi(f) \leq \phi_{\theta,\gamma}(f) + \frac{1}{\gamma} \int \beta^* \left(\frac{d\mu_\varepsilon}{d\theta}\right) d\theta + \varepsilon$$

هر گاه $\mu_\varepsilon \in \mathcal{Q}$ یک ε–بهینه‌ساز (۲) باشد به طوری که $\mu_\varepsilon \ll \theta$ و $\int \beta^*_\gamma \left(\frac{d\mu_\varepsilon}{d\theta}\right) d\theta < \infty$ باشد.
اگر $\hat{h} \in \mathcal{H}$ یک کمینه‌کننده (۳) باشد، آنگاه $\hat{\mu} \in \mathcal{P}(\mathcal{X})$ که به صورت زیر تعریف می‌شود،
یک بیشینه‌کننده (۶) است:

$$\frac{d\hat{\mu}}{d\theta} := \beta'_\gamma(f - \hat{h})$$

نوتیشن برای همگرایی نسبت به m

$$\phi^m(f) := \inf\Big\{\int h\,d\mu_0 : h \geq f \text{ for some } h \in \mathcal{H}^m\Big\}.$$

$$\phi_{\theta,\beta}(f) = \inf_{h\in\mathcal{H}}\Big\{\int h\,d\mu_0 + \int \beta(f-h)\,d\theta\Big\}.$$

$$\mathcal{H}^\infty := \bigcup_{m\in\mathbb{N}} \mathcal{H}^m$$

شرط (D): برای هر $\varepsilon > 0$ و $\mu \in \mathcal{P}(\mathcal{X})$ اعمال می‌شود:
(a) برای هر $h \in \mathcal{H}$، $h' \in \mathcal{H}^\infty$ موجود است به طوری که $\int |h-h'|\,d\mu \leq \varepsilon$.
(b) $h'' \in \mathcal{H}^\infty$ موجود است به طوری که $1_{K^c} \leq h''$ و $\int h'' d\mu \leq \varepsilon$ برای برخی از زیرمجموعه‌های فشرده K از $\mathcal{X}$.

قضیه همگرایی نسبت به تعداد نورون‌های لایه پنهان شبکه عصبی

گزاره ۳.۲: فرض کنید $\mathcal{H}^{\infty}$ یک فضای تابعی خطی است که حاوی توابع ثابت می‌باشد. با فرض شرط (D) داریم:

$$\lim_{m\to\infty} \phi^m(f) = \phi^{\infty}(f) = \phi(f)$$

برای همه $f \in C_b(\mathcal{X})$.

قضیه همگرایی نهایی مقاله

گزاره ۴.۲: فرض کنید $\mathcal{H}^{\infty}$ شرط (D) را برآورده می‌کند و برای هر $\varepsilon > 0$ یک بهینه‌ساز ε-به نام μ_ε از (۶) وجود دارد که $\mu_\varepsilon \ll \theta$ و $\int \beta^*\left(\frac{d\mu_\varepsilon}{d\theta}\right) d\theta < +\infty$. سپس می‌خواهیم ثابت کنیم که برای هر تابع $f \in C_b(\mathcal{X})$، $\phi^m_{\theta,\gamma}(f)$ به $\phi(f)$ میل می‌کند هنگامی که $\min\{m,\gamma\} \to \infty$.

مدلسازی و حل مساله با شبکه‌های عصبی MLP

$$\mathbb{R}^d \ni x \mapsto A_l \circ \underbrace{\varphi \circ A_{l-1}}_{(l-1)layer} \circ \cdots \circ \underbrace{\varphi \circ A_0}_{1.layer}(x)$$

که تمام تبدیل‌های A_l آفین هستند و تمام توابع ϕ توابع غیرخطی فعال‌سازی هستند که شرایط خاصی داشته باشند. بعد ورودی شبکه d است و بعد لایه‌های پنهان m است و خروجی شبکه در نهایت یک عدد حقیقی است که همان مقدار $\phi(f)$ در مساله اصلی خواهد بود. هر تبدیل آفین به فرم یک ضرب ماتریسی از وزن‌های شبکه عصبی به علاوه یک عبارت بایاس به فرم زیر قابل نمایش است:

$$A_j(x) = M_j x + b_j$$

تمامی این وزن‌ها و بایاس‌ها پارامترهای شبکه عصبی هستند که برای یک D طبیعی خاص می‌توان آن‌ها را به عنوان عضوی از R^D در نظر گرفت.

در ادامه، تمام پارامترهای ممکن برای شبکه عصبی با ساختاری ثابت را در مجموعه‌ای مثل $\Xi \subset \mathbb{R}^D$ تعریف می‌کنیم. برای یک $\xi \in \Xi$ خاص هم یک شبکه عصبی خاص را می‌توان با l لایه و بعد ورودی d و بعد لایه‌های پنهان m به فرم زیر نمایش داد:

$$N_{l,d,m}(\xi) = A_l \circ \varphi \circ A_{l-1} \circ \cdots \circ \varphi \circ A_0$$

و همچنین تمام شبکه‌های عصبی این چنینی را در یک مجموعه بزرگ‌تر به فرم زیر می‌ریزیم:

$$\mathfrak{N}_{l,d,m}(\Xi)$$

مدلسازی و حل مساله با شبکه‌های عصبی MLP

$$\mathcal{H} = \left\{ \sum_{j=1}^{J} e_j h_j \circ \pi_j + a : h_j \in C_b\left(\mathbb{R}^{d_j}\right), a \in \mathbb{R} \right\}$$

که در آن $e_j \in C_b(\mathcal{X})$ و $\pi_j : \mathcal{X} \to \mathbb{R}^{d_j}$ به ازای تمام j های مجاز توابعی پیوسته هستند.
همچنین داریم:

$$\mathcal{H}^\infty = \left\{ \sum_{j=1}^{J} e_j h_j \circ \pi_j + a : h_j \in \mathfrak{N}_{l_j, d_j}, a \in \mathbb{R} \right\}$$

و

$$\mathcal{H}^m = \left\{ \sum_{j=1}^{J} e_j h_j \circ \pi_j + a : h_j \in \mathfrak{N}_{l_j, d_j, m}\left(\Xi_{j,m}\right), a \in \mathbb{R} \right\}$$

مدلسازی و حل مساله با شبکه‌های عصبی MLP

$$
\begin{aligned}
\phi^m_{\theta,\gamma}(f) &= \inf_{h\in\mathcal{H}^m}\left\{\int h\,d\mu_0 + \int \beta_\gamma(f-h)\,d\theta\right\}\\
&= \inf_{a\in\mathbb{R}}\ \inf_{h_j\in\mathfrak{N}_{l_j,d_j,m}(\Xi_{j,m})}\left\{\int \sum_{j=1}^{J} e_j h_j\circ\pi_j\,d\mu_0 + a\right.\\
&\qquad \left. + \int \beta_\gamma\left(f - \sum_{j=1}^{J} e_j h_j\circ\pi_j - a\right)d\theta\right\}\\
&= \inf_{a\in\mathbb{R}}\ \inf_{\xi_j\in\Xi_{j,m}}\left\{\int \sum_{j=1}^{J} e_j N_{l_j,d_j,m}(\xi_j)\circ\pi_j\,d\mu_0 + a\right.\\
&\qquad \left. + \int \beta_\gamma\left(f - \sum_{j=1}^{J} e_j N_{l_j,d_j,m}(\xi_j)\circ\pi_j - a\right)d\theta\right\}
\end{aligned}
$$

بهینه‌سازی در عدم قطعیت دارایی‌ها

$$E(Y_x) - \lambda.Var(Y_x)$$

که در آن x در واقع وزن دارایی دوم در سبد است و Y_x سبد سهام تشکیل شده از وزن‌های x و $1-x$ است. متغیر λ نیز ضریب ریسک‌گریزی سرمایه‌گذار است. واضح است که هدف هر سرمایه‌گذاری فارغ از ضریب ریسک‌گریزی، بیشیشنه‌سازی عبارت بالاست. ولی چون از کوواریانس دو دارایی مطلع نیستیم، در محاسبه صریح واریانس به مشکل می‌خوریم. پس راه حل جایگزین، در نظر گرفتن بدترین حالت ممکن برای کوواریانس دو دارایی است.

فرض کنیم دارایی اول بازده μ_1 و دارایی دوم بازده μ_2 دارد و همچنین فرض کنیم که واریانس دو دارایی به تنهایی نیز به ترتیب σ_1^2 و σ_2^2 باشد.
آنگاه به وضوح داریم:

$$E(Y_x) = (1-x).\mu_1 + x.\mu_2$$

$$Var(Y_x) = (1-x)^2.\sigma_1^2 + x^2.\sigma_2^2 + 2.Cov(K_1, K_2)$$

بهینه‌سازی معیار ریسک در سبد دو دارایی

$$min_C((1-x).\mu_1 + x.\mu_2 - \lambda.((1-x)^2.\sigma_1^2 + x^2.\sigma_2^2 + 2.x.(1-x).Cov(K_1, K_2)))$$

این مینیمم به وضوح زمانی رخ می‌دهد که مقدار C ماکزیمم باشد، که به معنی بیشینه همبستگی بین دو دارایی است. حال به عنوان مثالی خاص، فرض می‌کنیم U و V دو متغیر تصادفی از توزیع یکنواخت $(0,1)$ باشند و داشته باشیم $W = 2.V^2$ اگر فرض کنیم بازده دارایی اول از توزیع U و بازده دارایی دوم از توزیع W است آنگاه داریم:

$$Y_x = (1-x).U + 2.x.V^2$$

پس

$$E(Y_x) = 0.5 * (1-x) + \frac{2.x}{3}$$

$$Var(Y_x) = \frac{(1-x)^2}{12} + \frac{16.x^2}{45} + 2.x.(1-x).C$$

بهینه‌سازی معیار ریسک در سبد دو دارایی

ولی بیشینه C ممکن در زمان وابستگی کامل رخ می‌دهد که معادل است با زمانی که توزیع اساسی دو متغیر تصادفی یکسان باشد. یعنی U و V^2 با هم برابر باشند و یک متغیر تصادفی باشند. در چنین حالتی

$$Cov(U, 2.V^2) = 2.Cov(U, V^2)$$

که چون واریانس U برابر است با $\frac{1}{12}$ ماکزیمم این کوواریانس هم برابر واریانس هرکدام و بنابراین همین مقدار است. پس

$$maxCov(U, 2.V^2) = 2 * \frac{1}{12} = \frac{1}{6}$$

و با جایگذاری و ساده‌سازی داریم:

$$E(Y_x) - \lambda.Var(Y_x) =$$

$$0.5 + \frac{x}{6} - \lambda.(\frac{1}{12} + \frac{x}{6} + \frac{19.x^2}{180})$$

بهینه‌سازی معیار ریسک در سبد دو دارایی

ولی حالا صرفا معیاری برای بدترین حالت ممکن داریم. برای بیشینه‌سازی سبد سهام، باید در این بدترین حالت، معیار بالا را بیشینه کنیم. پس برای سبد بهین با فرض عدم فروش استقراضی داریم:

$$R = max_{x \in [0,1]}(0.5 + \frac{x}{6} - \lambda.(\frac{1}{12} + \frac{x}{6} + \frac{19.x^2}{180}))$$

برای یافتن مقدار بیشینه کافیست از طرفین نسبت به وزن دارایی دوم مشتق بگیریم و برابر صفر قرار دهیم که به معادله زیر خواهیم رسید:

$$\frac{1}{6} - \lambda.(\frac{1}{6} + \frac{19.x}{90}) = 0$$

$$x = \frac{15.(1 - \lambda)}{19.\lambda}$$

و با جایگذاری مقدار بالا در معادله اصلی هم مقدار بهین به دست خواهد آمد. در کد پیاده‌سازی شده، توابع $analyticalportfolio$ و $analyticaloptimalvalue$ همین مقادیر را پیاده‌سازی کرده‌اند.

پیاده‌سازی روش مقاله روی این مساله

حال به تشریح ارتباط روش مقاله با این مساله خواهیم پرداخت. فرض کنیم که معیار ریسک خود را با تابعی نشان می‌دهیم:

$$f_x = E(Y_x) - \lambda . Var(Y_x)$$

هدف یافتن مقدار زیر است:

$$\max_{x\in[0,1]} \left(\inf_{v\in Q}\left(\int f_x dv\right)\right)$$

اما به طور خاص

$$\inf_{v\in Q}\left(\int f_x\right) = -\sup_{v\in Q}\left(\int (-f_x dv)\right) = -\phi(-f_x)$$

پس مساله اصلی عبارت است از

$$\sup_{x\in[0,1]} \left(-\phi(-f_x)\right)$$

پیاده‌سازی روش مقاله روی این مساله

و با جایگذاری تابع f_x طبق تعریفش مساله اصلی به فرم زیر هم قابل بازنویسی است

$$\begin{aligned}
&\sup_{x\in[0,1]} -\phi\left(-f_x\right) \\
&:= \sup_{x\in[0,1]} \inf_{v\in\mathcal{Q}} \int (1-x)\xi_1 + x\xi_2 \\
&\quad - \lambda\left((1-x)\xi_1 + x\xi_2 - (1-x)\int_0^1 \zeta_1\theta_1\left(d\zeta_1\right) - x\int_0^2 \zeta_2\theta_2\left(d\zeta_2\right)\right)^2 v(d\xi).
\end{aligned}$$

که با توجه به نکات مطرح شده می‌توان عبارت بالا را به فرم زیر هم نوشت:

$$\begin{aligned}
&\sup_{x\in[0,1]} -\phi\left(-f_x\right) \\
&:= -\inf_{x\in[0,1]} \sup_{v\in\mathcal{Q}} - \int (1-x)\xi_1 + x\xi_2 \\
&\quad - \lambda\left((1-x)\xi_1 + x\xi_2 - (1-x)\int_0^1 \zeta_1\theta_1\left(d\zeta_1\right) - x\int_0^2 \zeta_2\theta_2\left(d\zeta_2\right)\right)^2 v(d\xi).
\end{aligned}$$

پیاده‌سازی روش مقاله روی این مساله

با نوشتن فرم دوگان و جریمه‌سازی به عبارت زیر می‌رسیم:

$$-inf_{x\in[0,1]\ \&\ h\in H} \int h d\mu_0 + \int \beta(-f-h)d\theta$$

$$\theta_1 \sim \mathcal{U}(0,1)$$

و اگر

$$r \sim \mathcal{U}(0,1)$$

آنگاه

$$\theta_2 = 2.r^2$$

همچنین فرض می کنیم فضای لهستانی ما به فرم زیر است:

$$\mathcal{X} = [0,1] * [0,2]$$

$$\theta^{(1)} = \theta_1 \otimes \theta_2$$

$$\theta^{(2)} = 0.5\theta^{(1)} + 0.5\left(\mathcal{U}([0,1]) \circ (\mathrm{Id}, \varphi)^{-1}\right)$$

$$h(x_1,x_2) = h_1(x_1) + h_2(x_2)$$

$$\beta(x) = \frac{1}{2}.\max(0,x)^2$$

$$\beta_\gamma(x) = \frac{1}{\gamma}.\beta(\gamma x)$$

$$\beta_\gamma(x) = \frac{1}{2\gamma}.\max(0,\gamma x)^2$$

توضیح کد

```
def generate_theta_1(size):  #reference measure theta(1) which is theta1 * theta2
  points=np.zeros([size,2])
  points[:,0]=np.random.random_sample(size)  #theta1
  points[:,1]=2*(np.random.random_sample(size)**2) #theta2

  return points

def generate_theta_2(size):  #reference measure theta(2) which is 0.5*theta(2) + 0.5* samedef gen_points_1(batch
      dataset = np.zeros([size, 2])
      dataset[:, 0] = np.random.random_sample(size)
      dataset[:round(size/2), 1] = 2 * ((np.random.random_sample(round(size/2))) ** 2)
      dataset[round(size/2):, 1] = 2 * (dataset[round(size/2):, 0] ** 2)
      return dataset
```

```
def create_model(m=64):  #Creating the uni-varaite functions h_1 and h_2 as DNN models
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(m, input_shape=(1,), activation='relu',
                              kernel_initializer=tf.random_uniform_initializer(minval=-1, maxval=1),
                              bias_initializer=tf.random_uniform_initializer(minval=-1, maxval=1)),
        tf.keras.layers.Dense(m, activation='relu',
                              kernel_initializer=tf.random_uniform_initializer(minval=-1, maxval=1),
                              bias_initializer=tf.random_uniform_initializer(minval=-1, maxval=1)),
        tf.keras.layers.Dense(m, activation='relu',
                              kernel_initializer=tf.random_uniform_initializer(minval=-1, maxval=1),
                              bias_initializer=tf.random_uniform_initializer(minval=-1, maxval=1)),
        tf.keras.layers.Dense(1,
                              kernel_initializer=tf.random_uniform_initializer(minval=-1, maxval=1))
    ])
    return model
```

توضیح کد

```python
def create_models(m=64):
    h1 = create_model(m)
    h2 = create_model(m)
    return h1, h2
```

```python
# Define the function h
def h(data,h1,h2):
  x_1=data[:,0:1]
  x_2=data[:,1:2]
  h_1= h1(x_1)
  h_2= h2(x_2)
  return h_1 + h_2
```

```python
def integral_h(data,h1,h2):
  return tf.reduce_mean(h(data,h1,h2))
```

```python
def portfolio_return(x1,x2, w):  # w is the weight for the second portfolio
  return (1-w)*x1 + w*x2
```

```python
def f(portfolio_return,landa):
  return portfolio_return-landa*variance(portfolio_return)
```

```python
def no_short_selling_penalty(w):
  return 100 * tf.nn.relu(-w) + 100 * tf.nn.relu((w-1))
```

```python
def integral_beta(gamma,landa,data,w,h1,h2):
  output= tf.reduce_mean( (1/gamma)* tf.square(gamma*tf.nn.relu(((-f(portfolio_return(data[:,0:1],data[:,1:2],w),w),landa)-h(data,h1,h2))))))
  return output
```

توضیح کد

```python
def target_function(data,size,gamma,landa,w,h1,h2,no_short_selling=True):

  temp= integral_h(data,h1,h2) + integral_beta(gamma,landa,data,w,h1,h2)
  if no_short_selling:
    temp+=no_short_selling_penalty(w)
  return temp
```

```python
def analytical_portfolio(landa, no_short_selling=True):
  if landa==0:
    return 1
  if landa==1:
    return 0

  temp = 15*(1-landa)/(19*landa)

  if temp>1 and no_short_selling:
    temp=1

  return temp
```

```python
gamma=160
size=2**13
learning_rate=0.001
beta1=0.99
beta2=0.995
epochs=10000
tol_h = 1e-5
m=32
num_experiments=1
```

توضیح کد

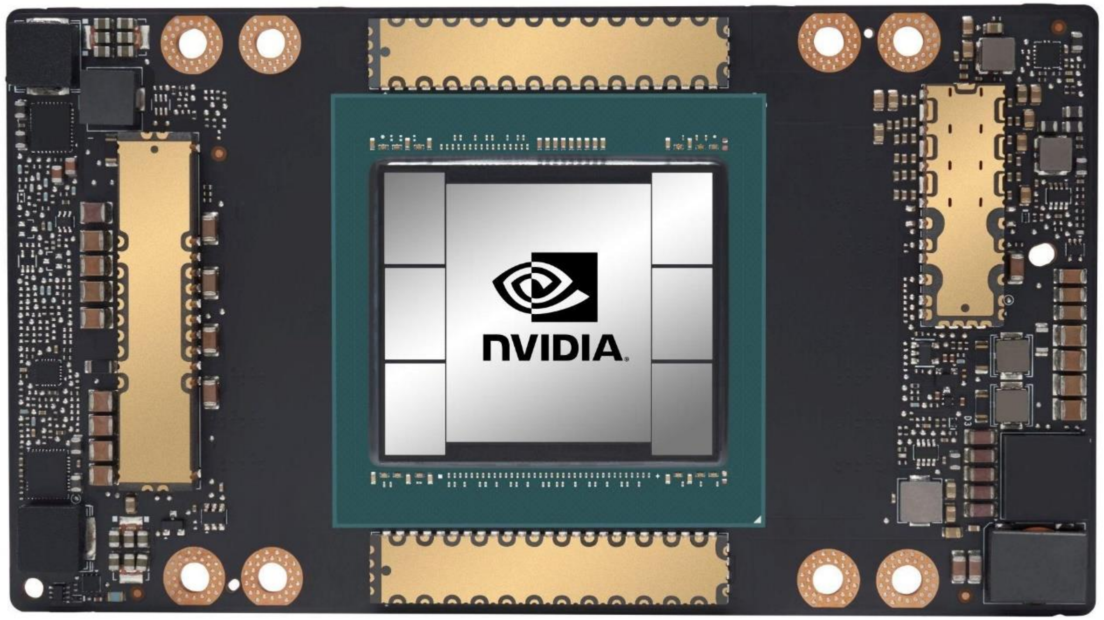
```python
def training(landa,measure):
    with tf.device(device):
        final_w_values = []
        for i in range(num_experiments):
            print("Starting experiment", i + 1)

            h1, h2 = create_models(m)
            w = tf.Variable(0.5)

            optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, beta_1=beta1, beta_2=beta2)
            if measure=="theta1":
              data=generate_theta_1(size)
            elif measure=="theta2":
              data = generate_theta_2(size)
            else:
              print("Wrong measure")

            loss_values_h = []
            recorded_loss = 1e8

            for epoch in range(epochs):
                Flag=False
                with tf.GradientTape() as tape:
                    loss = target_function(data, size, gamma, landa, w, h1, h2)

                grads = tape.gradient(loss, h1.trainable_variables + h2.trainable_variables + [w])
                optimizer.apply_gradients(zip(grads, h1.trainable_variables + h2.trainable_variables + [w]))
                loss_values_h.append(loss.numpy())

                if epoch % 1000 == 0:
```

```python
                    print('Epoch:', epoch, 'Loss for h1 and h2 and w:', loss.numpy())
                    if abs(recorded_loss-loss)<tol_h:
                      Flag=True
                    recorded_loss=loss.numpy()

                prev_w = w.numpy()

            if not Flag:
              final_w_values.append(w.numpy())
              print("Optimal w for experiment", i + 1, ":", w.numpy())
            else:
              print("This is a case a vanishing gradients")
              break

            plt.figure(figsize=(12, 6))
            plt.plot(loss_values_h)
            plt.title('Loss for h1 and h2 and w for experiment ' + str(i+1))
            plt.show()

        mean_w = np.mean(final_w_values)
        return mean_w
```

نتایج

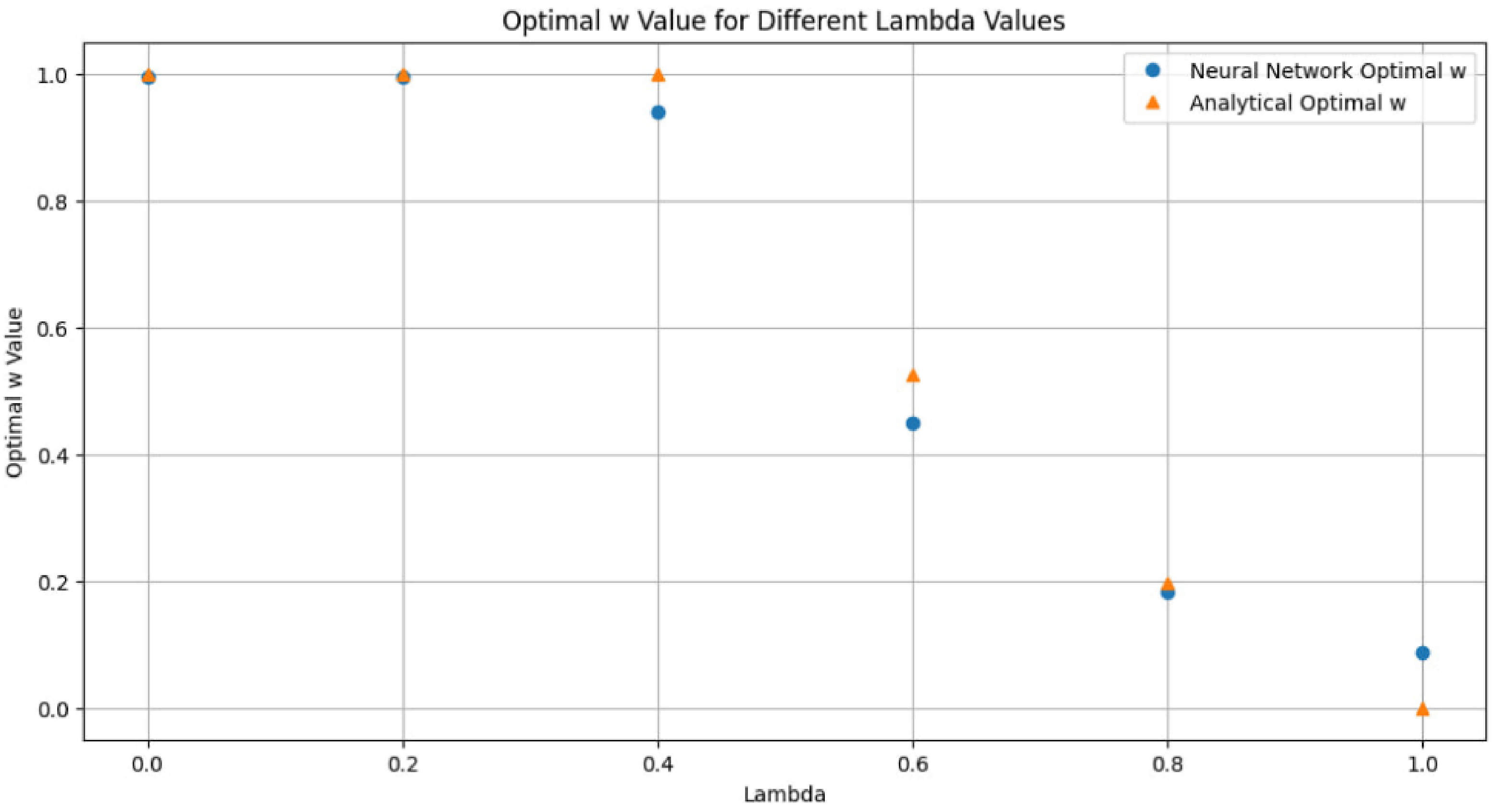
آزمایش اول: بهینه‌سازی کامل

مشکل اساسی: GPU



$$m = 32, \gamma = 160, learningrate = 0.001, \beta 1 = 0.99,$$
$$\beta 2 = 0.995, epochs = 10000, batchsize = 2^{13}$$

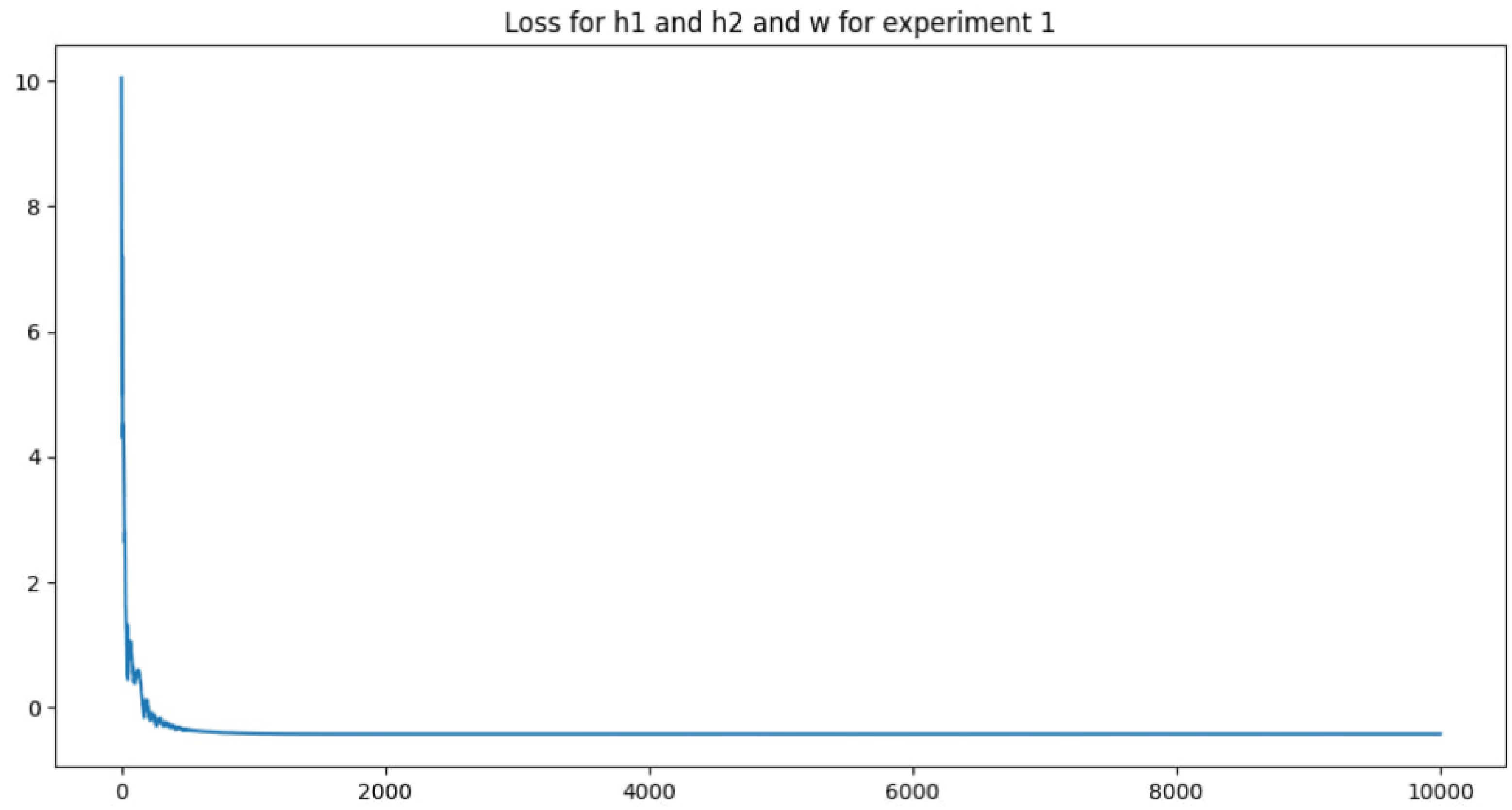
نتایج بهینه‌سازی کامل



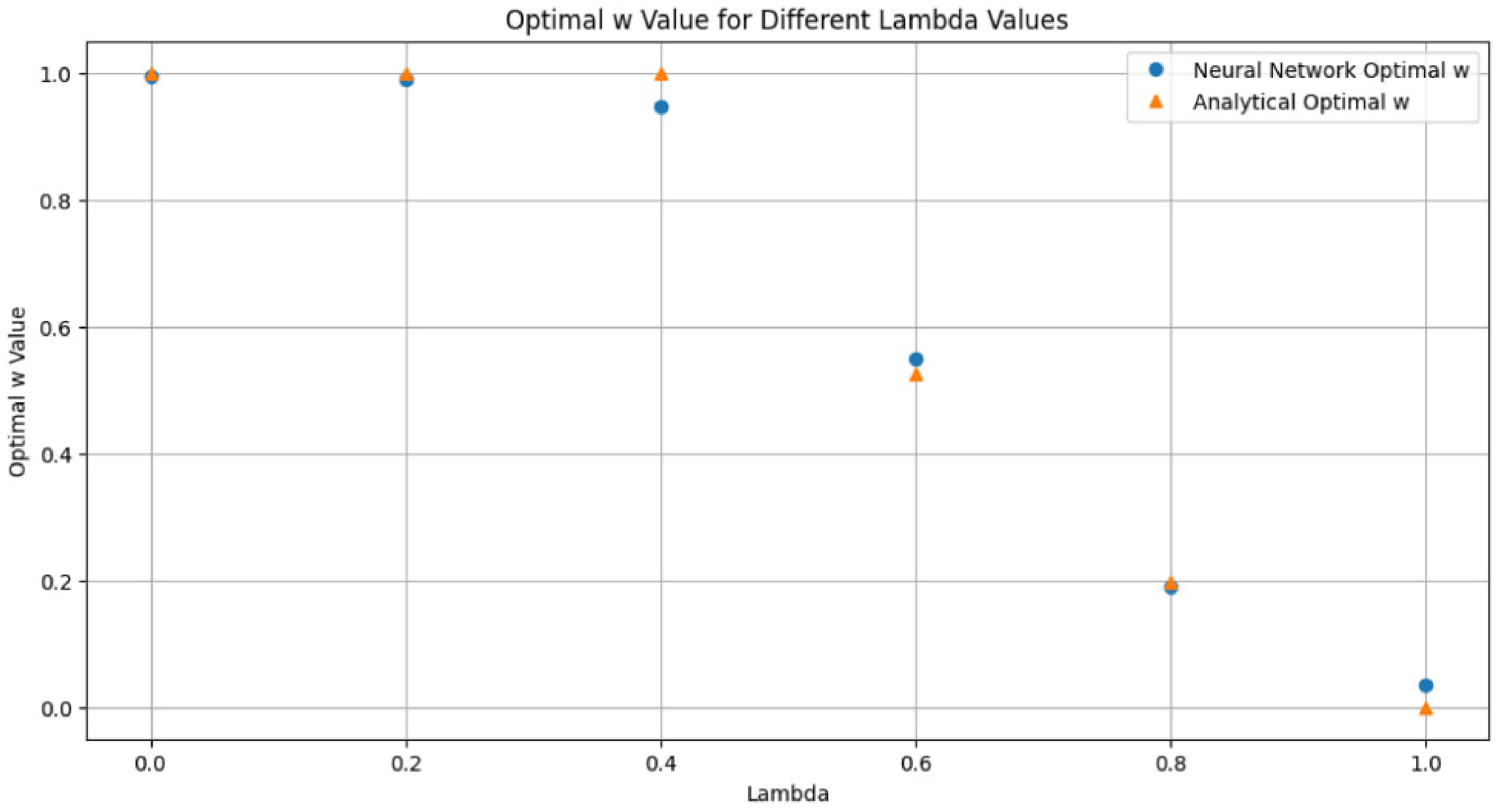
شکل ۴: مقادیر بهینه w برای مقادیر مختلف لاندا در حل آنالیتیکی و حل با توزیع $\theta^{(1)}$

$$w = [0.99412537, 0.99498355, 0.93941647, 0.44836196, 0.18348294, 0.08750755]$$

نتایج بهینه‌سازی کامل

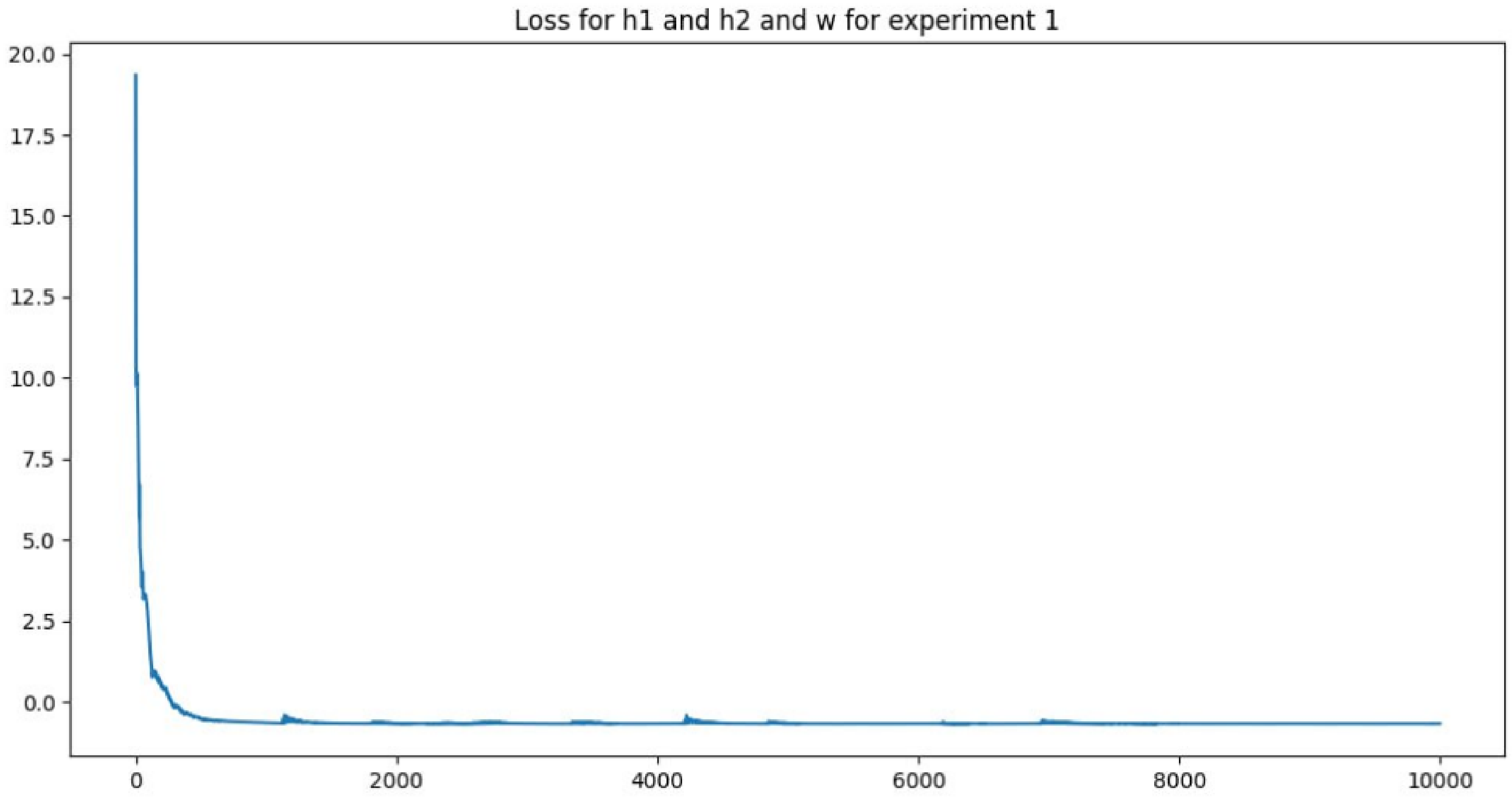


نتایج بهینه‌سازی کامل



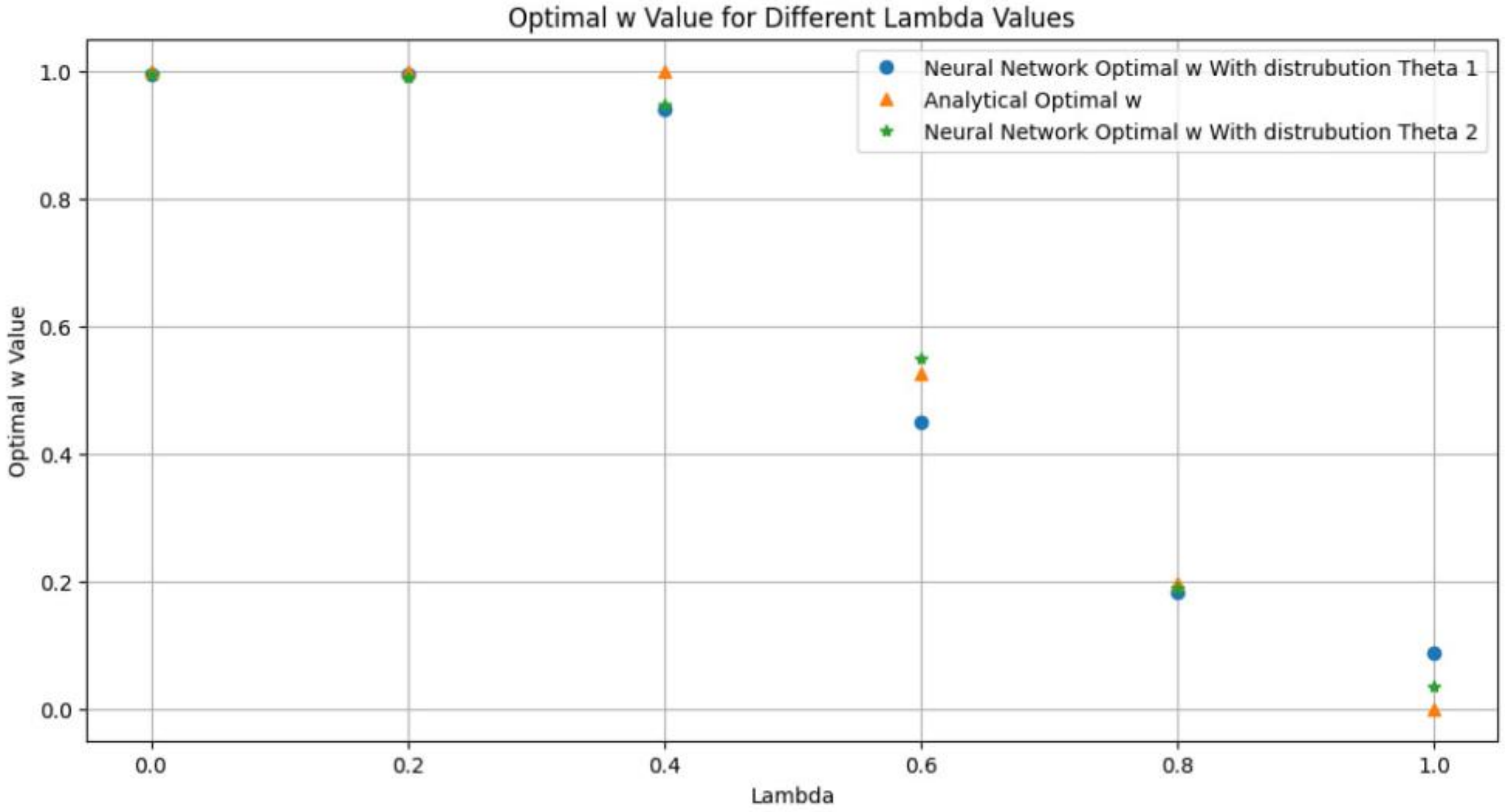
شکل ۷: مقادیر بهینه w برای مقادیر مختلف لاندا در حل آنالیتیکی مسئله و حل با توزیع $\theta^{(2)}$
$w = [0.99536186, 0.99050164, 0.947735, 0.55005884, 0.18909983, 0.03486183 3]$

نتایج بهینه‌سازی کامل

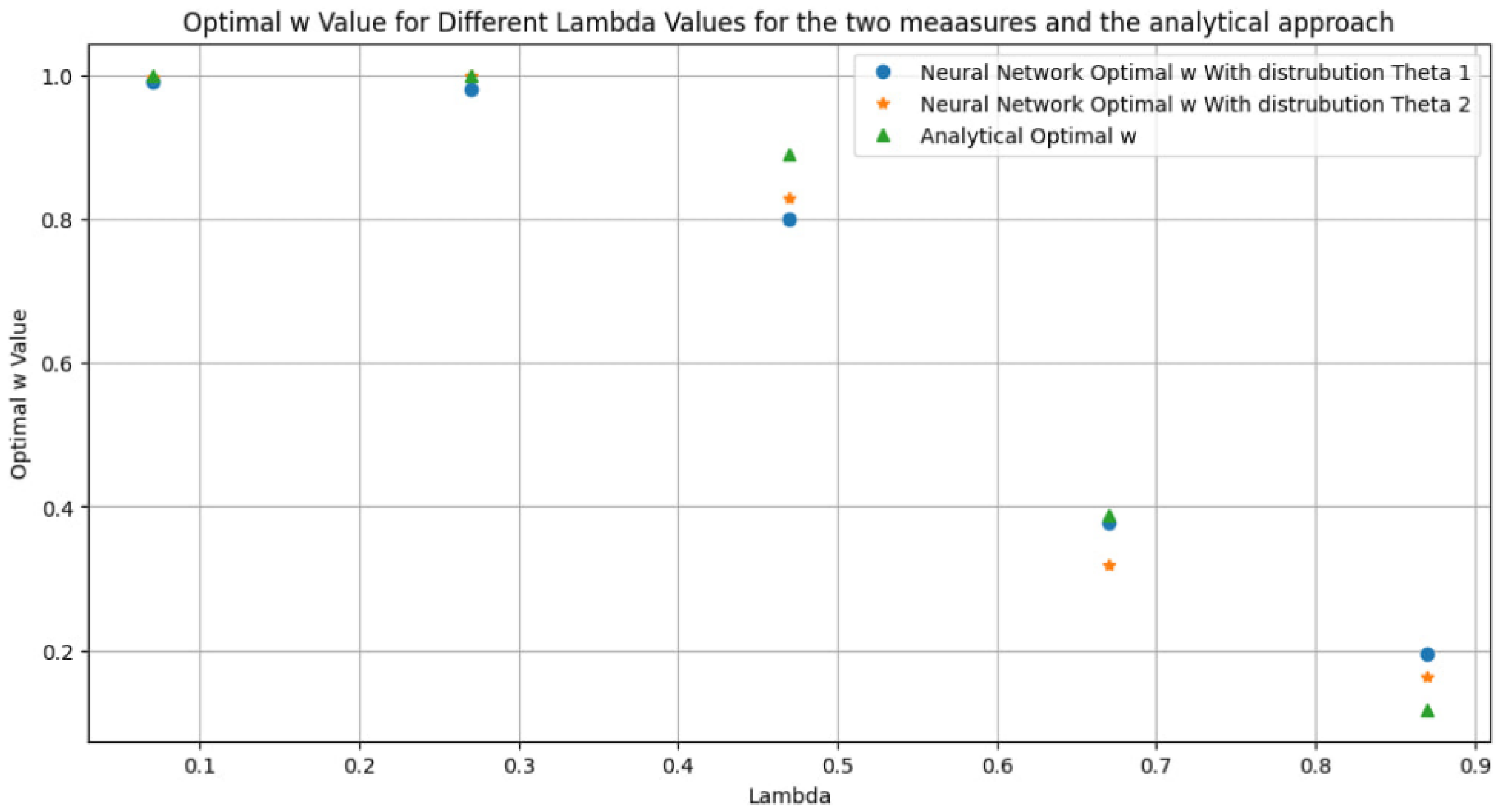


شکل ۶: نمونه تابع $loss$ برای $\theta^{(2)}$

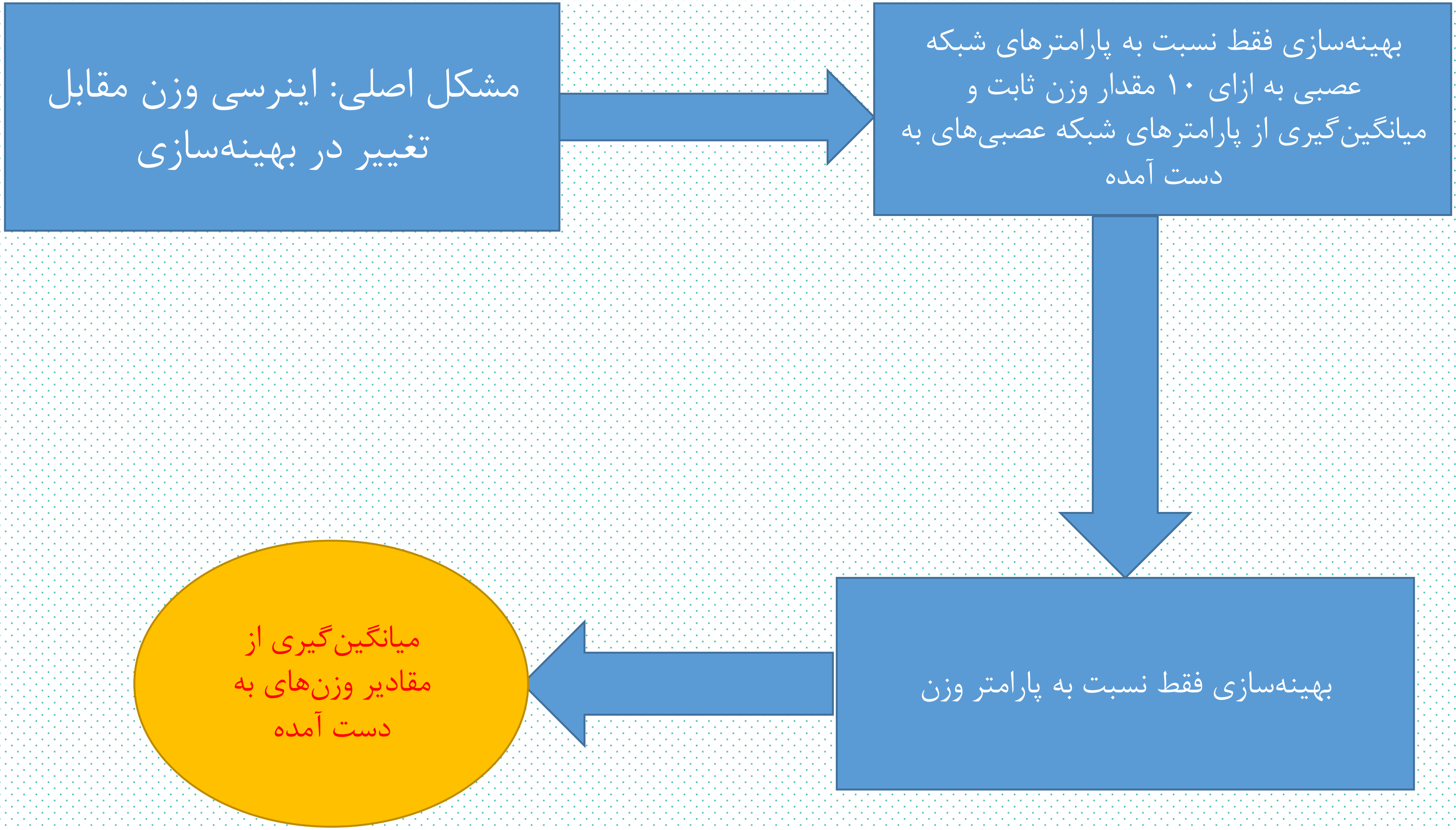
نتایج بهینه‌سازی کامل



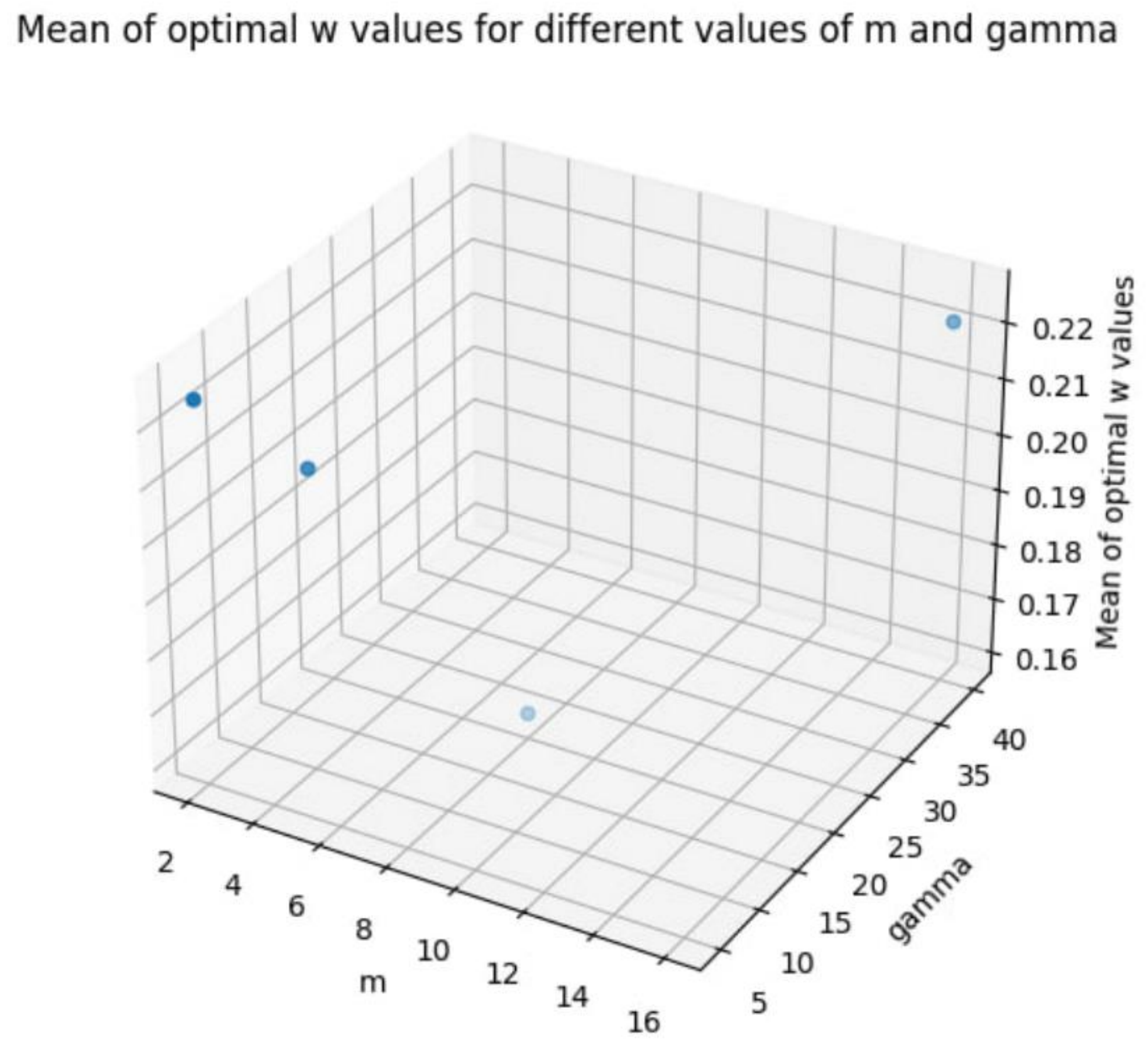
نتایج بهینه‌سازی کامل



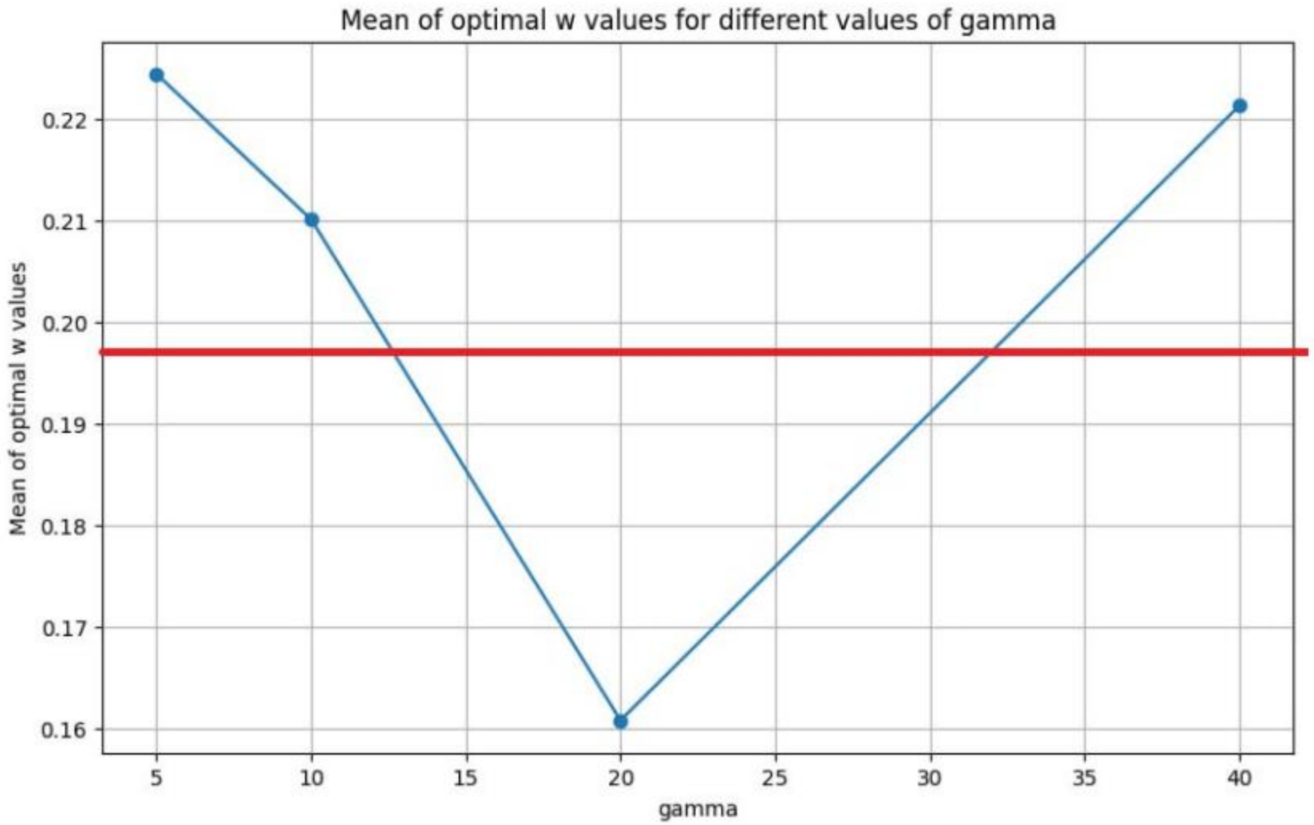
بهینه‌سازی هیوریستیکی برای مقایسه ضرایب مختلف جریمه و تعداد نورون‌ها



بهینه‌سازی هیوریستیکی برای مقایسه ضرایب مختلف جریمه و تعداد نورون‌ها

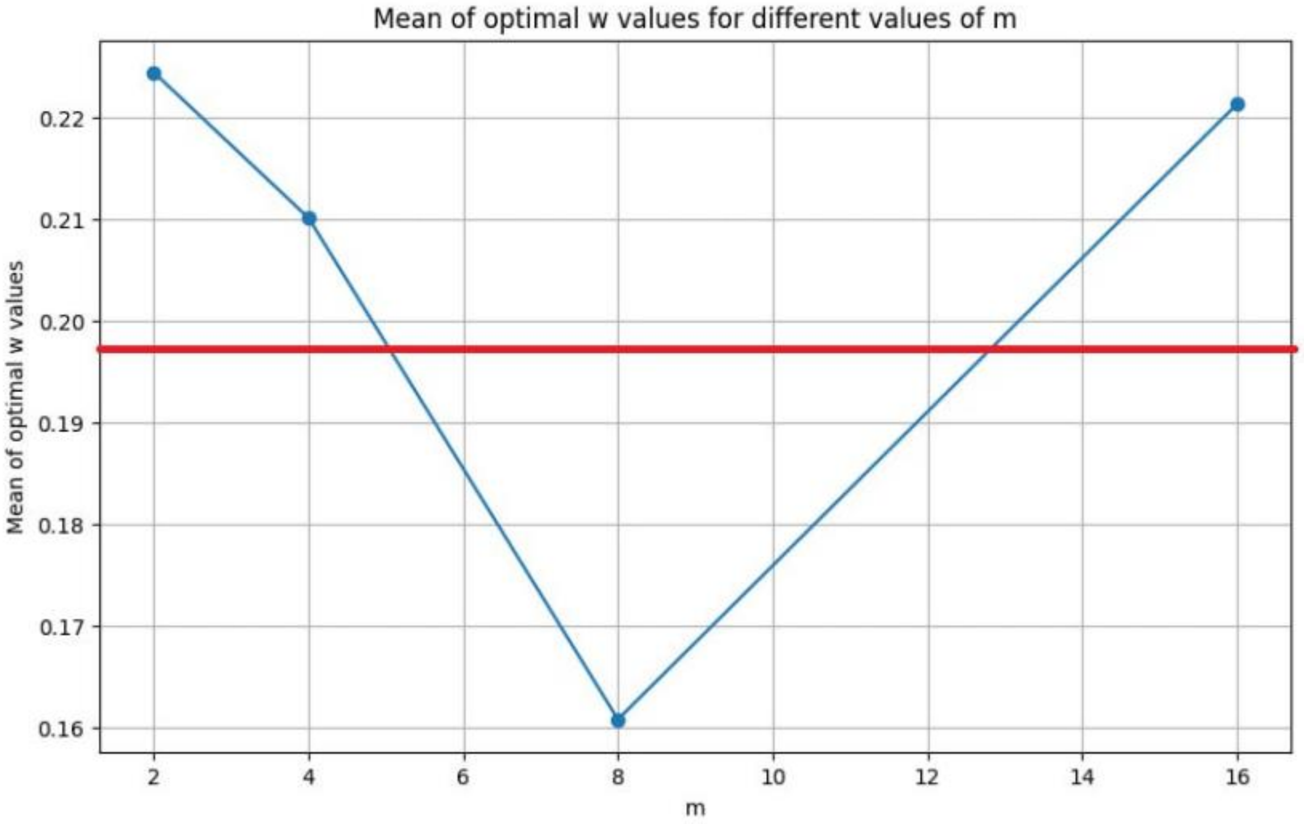


بهینه‌سازی هیوریستیکی برای مقایسه ضرایب مختلف جریمه و تعداد نورون‌ها



شکل ۱۱: میانگین وزن‌های به دست آمده برای ۴ مقدار متفاوت گاما به ازای $\lambda = 0.8$.
خط قرمز، جواب بهین واقعی است.

بهینه‌سازی هیوریستیکی برای مقایسه ضرایب مختلف جریمه و تعداد نورون‌ها



پیاده‌سازی اختیار خرید اروپایی

$$Payoff_{\,call} = \max(0, S_T - K)$$

$$C(S_t, t) = N(d_1)S_t - N(d_2)PV(K)$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

$$PV(K) = Ke^{-r(T-t)}$$

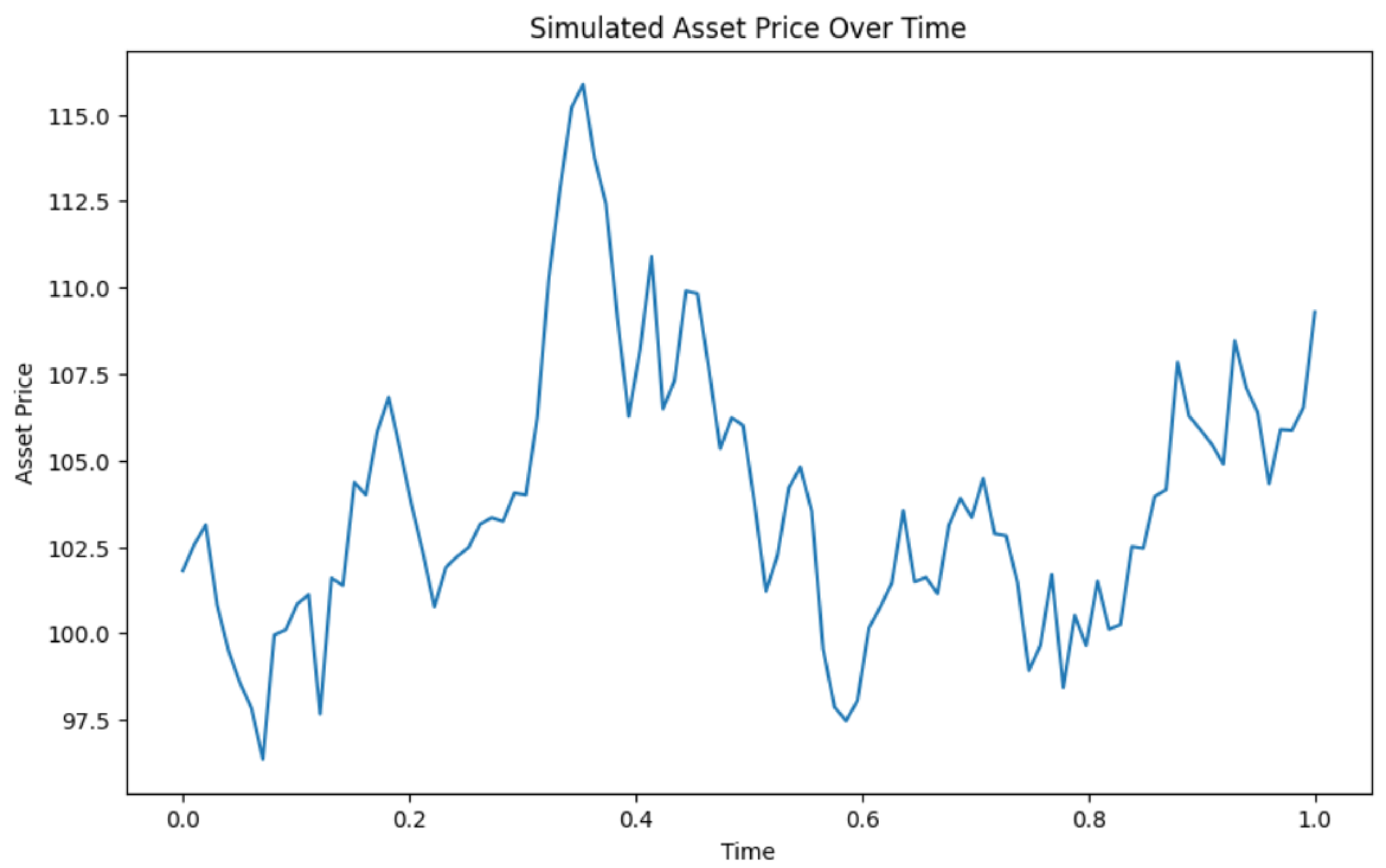
```
# Black-Scholes formula
def black_scholes_call(S0, K, T, r, sigma):
    d1 = (np.log(S0 / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = (np.log(S0 / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    call_price = (S0 * norm.cdf(d1, 0.0, 1.0) - K * np.exp(-r * T) * norm.cdf(d2, 0.0, 1.0))
    return call_price

# Parameters
S0 = 100       # initial asset price
K = 100        # strike price
T = 1.0        # time to maturity
r = 0.05       # risk-free rate
sigma = 0.2    # volatility
```

پیاده‌سازی اختیار خرید اروپایی

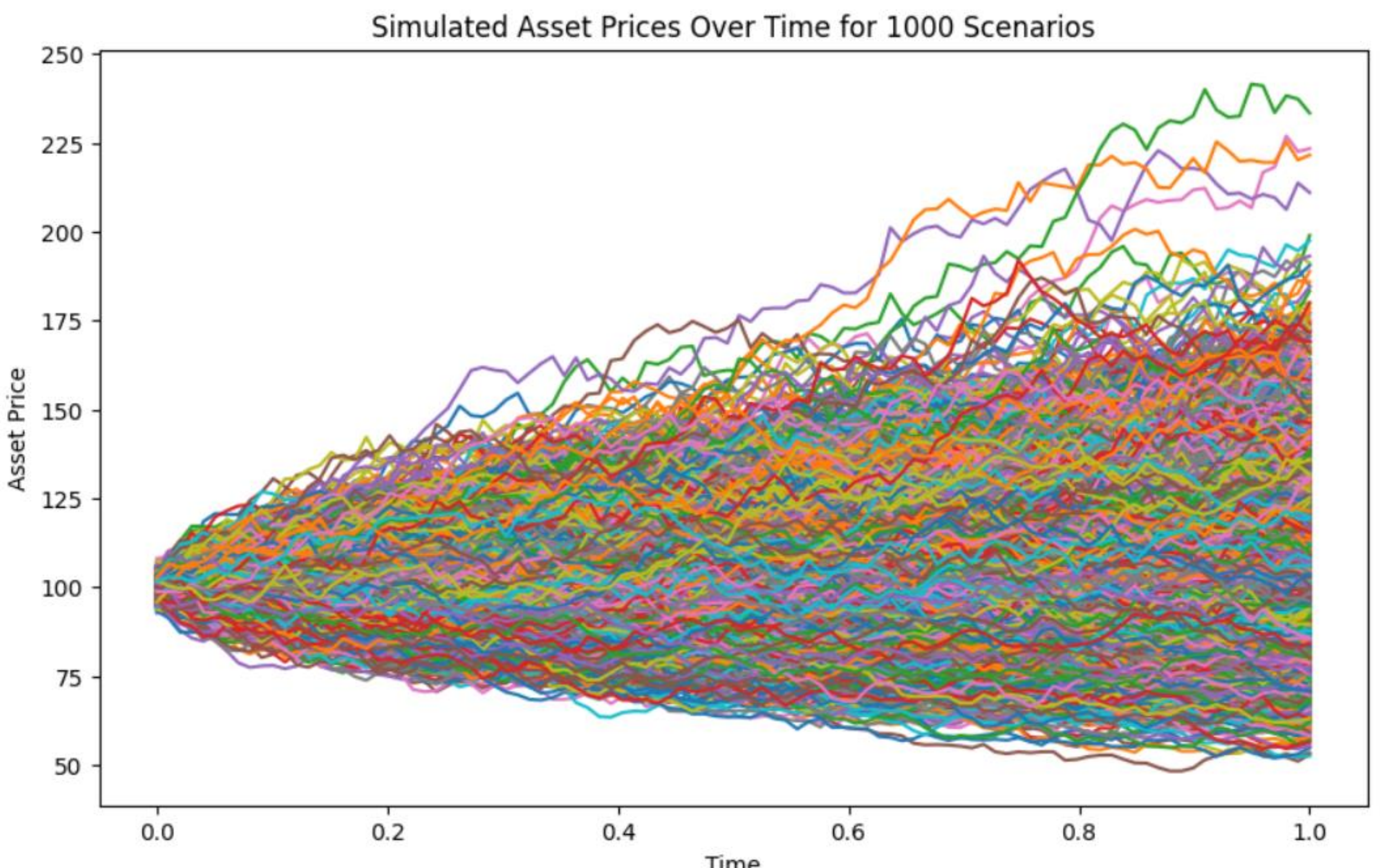
```python
# Function to simulate the asset price using Geometric Brownian Motion
def simulate_asset_price(S0, mu, sigma, T, dt, num_simulations):
    N = round(T/dt)
    t = np.linspace(0, T, N)
    S = np.zeros((num_simulations, N))
    S[:, 0] = S0
    for i in range(num_simulations):
        W = np.random.standard_normal(size = N)
        W = np.cumsum(W)*np.sqrt(dt) # standard brownian motion
        X = (mu-0.5*sigma**2)*t + sigma*W
        S[i] = S0*np.exp(X) # geometric brownian motion
    return S, S[:,-1], t  # Return the last column of S which represents the price at strike time

# Parameters
S0 = 100      # initial asset price
mu = 0.05     # drift
sigma = 0.2   # volatility
T = 1.0       # time to maturity
dt = 0.01     # time step
num_simulations = 5000 # number of simulations

# Simulate asset price
S, data, t = simulate_asset_price(S0, mu, sigma, T, dt, num_simulations)

# Print the price at strike time for all scenarios
print('The price at strike time for all scenarios is:', data)

# Plot the simulated asset prices
plt.figure(figsize=(10,6))
for i in range(num_simulations):
    plt.plot(t, S[i])
plt.title('Simulated Asset Prices Over Time for 1000 Scenarios')
plt.xlabel('Time')
```

شبیه‌سازی یک حرکت براونی هندسی با پارامترها

```
The price of the European call option using Black-Scholes formula is: 10.450583572185565
The simulated asset price is: 109.28907808268717
```



شبیه‌سازی ۵ هزار حرکت براونی هندسی با پارامترها



سوپرهجینگ اختیار خرید اروپایی

$f: \max(0, S_T - K)$

$$Superhedging\ Price = \inf\left\{\int h\ dP\ for\ all\ h \geq f\ a.s\right\}$$

$$\phi(f) = \inf\left\{\int h\ dP + \int \beta(f-h)\, d\mu\ for\ all\ h \in H\right\}$$

$$\phi^{m}_{\gamma,\theta}(f) = \inf\left\{\int h\ d\mu + \int \beta_\gamma\,(f-h)\, d\theta\ for\ all\ h \in H^m\right\}$$

حل مساله سوپرهجینگ با روش مقاله

```
def h_model(m):
    model = create_model(m)
    return model

def integral_h(data,model):
    return tf.reduce_mean(model(data))
```

```
def f(data,K):
  f_calculated=[]
  for d in data:
    f_calculated.append(tf.nn.relu(d-K))

  return f_calculated
```

```
def integral_beta(gamma,f,model,data):
    return tf.reduce_mean((1/(2*gamma))*(tf.square(gamma*tf.nn.relu(f-model(data)))))
```

```
def target_function(model,gamma,K,data):
    inth=integral_h(data,model)
    intbeta=integral_beta(gamma,f(data,K),model,data)
    return inth+intbeta
```

```
gamma=100
size=1000
learning_rate=0.00001
beta1=0.99
beta2=0.995
epochs=10000
tol_h = 1e-5
m=16
num_experiments=1
K=100
```

آموزش شبکه عصبی چندلایه

```
    # Reset the models and w for each experiment
    h = h_model(m)

    # Define the optimizers
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, beta_1=beta1, beta_2=beta2)
    # Generate new data for each experiment

    # Prepare lists to store the loss values for plotting
    loss_values_h = []

    for epoch in range(epochs):
        with tf.GradientTape() as tape:
            loss =  target_function(h,gamma,K,data)

        grads = tape.gradient(loss, h.trainable_variables)
        optimizer.apply_gradients(zip(grads, h.trainable_variables))
        # Store the loss value for plotting
        loss_values_h.append(loss.numpy())

        # Print the loss value every 10 epochs
        if epoch % 100 == 0:
            print('Epoch:', epoch, 'Loss for h', loss.numpy())
            recorded_loss=loss.numpy()

    # Plot the loss values
    plt.figure(figsize=(12, 6))
    plt.plot(loss_values_h)
    plt.title('Loss ' + str(i+1))
    plt.show()
# Compute and print the mean of the optimal w values
print("Super-hedging value", loss.numpy())
```

نتایج

مقدار تحلیلی: حدودا ۱۰.۴۵۰۵۸۳۵۷۲۱۸۵۵۶۵

لاگ آموزش:

```
Epoch: 0 Loss for h 851028.9
Epoch: 100 Loss for h 769594.25
Epoch: 200 Loss for h 692834.7
Epoch: 300 Loss for h 621012.8
Epoch: 400 Loss for h 554118.75
Epoch: 500 Loss for h 491832.56
Epoch: 600 Loss for h 433976.1
Epoch: 700 Loss for h 380378.62
Epoch: 800 Loss for h 330887.94
Epoch: 900 Loss for h 285373.66
Epoch: 1000 Loss for h 243741.47
Epoch: 1100 Loss for h 205892.88
Epoch: 1200 Loss for h 171753.6
Epoch: 1300 Loss for h 141250.39
Epoch: 1400 Loss for h 114313.54
Epoch: 1500 Loss for h 90866.984
Epoch: 1600 Loss for h 70826.07
Epoch: 1700 Loss for h 54091.855
Epoch: 1800 Loss for h 40543.848
Epoch: 1900 Loss for h 30029.021
Epoch: 2000 Loss for h 22344.715
Epoch: 2100 Loss for h 17170.566
Epoch: 2200 Loss for h 13489.464
Epoch: 2300 Loss for h 10672.547
Epoch: 2400 Loss for h 8467.578
Epoch: 2500 Loss for h 6726.745
Epoch: 2600 Loss for h 5351.6006
Epoch: 2700 Loss for h 4267.769
Epoch: 2800 Loss for h 3416.1604
```

```
Epoch: 2800 Loss for h 3416.1604
Epoch: 2900 Loss for h 2750.262
Epoch: 3000 Loss for h 2237.8938
Epoch: 3100 Loss for h 1830.234
Epoch: 3200 Loss for h 1504.8729
Epoch: 3300 Loss for h 1244.354
Epoch: 3400 Loss for h 1034.088
Epoch: 3500 Loss for h 863.77344
Epoch: 3600 Loss for h 725.4606
Epoch: 3700 Loss for h 612.8631
Epoch: 3800 Loss for h 521.07654
Epoch: 3900 Loss for h 446.2518
Epoch: 4000 Loss for h 385.164
Epoch: 4100 Loss for h 335.31314
Epoch: 4200 Loss for h 294.60657
Epoch: 4300 Loss for h 261.37732
Epoch: 4400 Loss for h 234.27553
Epoch: 4500 Loss for h 212.20421
Epoch: 4600 Loss for h 194.2745
Epoch: 4700 Loss for h 179.78159
Epoch: 4800 Loss for h 168.146
Epoch: 4900 Loss for h 158.90019
Epoch: 5000 Loss for h 151.65065
Epoch: 5100 Loss for h 146.07065
Epoch: 5200 Loss for h 141.87808
Epoch: 5300 Loss for h 138.83307
Epoch: 5400 Loss for h 136.72092
Epoch: 5500 Loss for h 135.34929
Epoch: 5600 Loss for h 134.54085
```

تحقیقات آینده

ما در صورت داشتن بودجه محاسباتی مناسب مایل به بررسی تزها و فرضیات زیر هستیم:

- در حالت بهینه‌سازی سبد سهام، آیا بالا بردن تدریجی m و گاما و بهینه‌سازی مدام هایپرپارامترهای شبکه عصبی واقعا منجر به نزدیک‌تر شدن جواب کد به جواب آنالیزی می‌شود؟
- در حالت سوپرهجینگ، آیا بالا بردن تدریجی دو فاکتور بالا واقعا منجر به یافتن یک بازه‌ی replication کوچک‌تر و بهتر می‌شود؟
- تعامل و trade-off بین تخمین دقیق‌تر و هزینه محاسباتی چقدر است و با توجه به کاربردهای واقعی در بازار، چه هایپرپارامترهایی برای دو مساله بالا، مناسب‌تر هستند؟
- پیچیده‌تر کردن ساختار شبکه‌های عصبی مورد استفاده و استفاده از تکنیک‌های دیگر یادگیری ماشینی از قبیل یادگیری تقویتی، استفاده از پارامترهای اشتراکی در سری‌های زمانی(RNN) و توابع فعال‌سازی دیگر، چه تأثیری در عملکرد واقعی برای مسائل دارد؟
- در تمامی حالات بالا، آیا می‌توان توجیه ریاضیاتی همگرایی به جواب آنالیزی مثل روش مقاله یافت؟

منابع

[1] Cheridito, P., Kupper, M., Tangpi, L.: Duality formulas for robust pricing and hedging in discrete time. SIAM J. Financ. Math. 8(1), 738–765 (2017)

[2] Kantorovich, L.V.: On the translocation of masses. Dokl. Akad. Nauk SSSR 37, 199–201 (1942)

[3] Bartl, D., Cheridito, P., Kupper,M., Tangpi, L.: Duality for increasing convex functionals with countably many marginal constraints. Banach J. Math. Anal. 11(1), 72–89 (2017)

[4] Vallender, S.: Calculation of the Wasserstein distance between probability distributions on the line. Theory Probab. Appl. 18(4), 784–786 (1974)

[5] Villani, C.: Optimal Transport: Old and New, vol. 338. Springer, New York (2008)

[6] Beiglböck, M., Henry-Labordère, P., Penkner, F.: Model-independent bounds for option prices: a mass transport approach. Financ. Stoch. 17(3), 477–501 (2013)

[7] Galichon, A., Henry-Labordere, P., Touzi, N.: A stochastic control approach to no-arbitrage bounds given marginals, with an application to lookback options. Ann. Appl. Probab. 24(1), 312–336 (2014)

[8] Guo, G., Obloj, J.: Computational methods for martingale optimal transport problems. arXiv preprint arXiv:1710.07911 (2017)

[9] Henry-Labordère, P.: Automated option pricing: numerical methods. Int. J. Theor. Appl. Financ. 16(08), 1350042 (2013)

منابع

[10] Bernard, C., Rüschendorf, L., Vanduffel, S., Yao, J.: How robust is the value-at-risk of credit risk portfolios? Eur. J. Financ. 23(6), 507–534 (2017)

[11] Embrechts, P., Puccetti, G., Rüschendorf, L.: Model uncertainty and VaR aggregation. J. Bank. Financ. 37(8), 2750–2764 (2013)

[12] Puccetti, G., Rüschendorf, L.: Computation of sharp bounds on the distribution of a function of dependent risks. J. Comput. Appl. Math. 236(7), 1833–1840 (2012)

[13] Bartl, D., Kupper, M., Lux, T., Papapantoleon, A.: Sharpness of improved Fréchet-Hoeffding bounds: an optimal transport approach. arXiv preprint arXiv:1709.00641 (2017)

[14] Lux, T., Papapantoleon, A.: Improved Fréchet-Hoeffding bounds on d-copulas and applications in model-free finance. Ann. Appl. Probab. 27(6), 3633–3671 (2017)

[15] Pflug, G.C., Pohl, M.: A review on ambiguity in stochastic portfolio optimization. Set-Valued Var. Anal. 23(1), 11–25 (2017)

[16] Alfonsi, A., Corbetta, J., Jourdain, B.: Sampling of probability measures in the convex order and approximation of martingale optimal transport problems (2017)

منابع

C.Merton, R. (December 1980). On estimating the expected return on the market: An exploratory investigation. *Journal of Financial Economics*, 323-361.

Dubois, M. S. (2015). Topics in Portfolio Optimisation. *The London School of Economics and Political Science.*

Higham, N. J. (2016). *The Princeton Companion to Applied Mathematics.* Princeton .

Kopp, M. J. (2014). *Portfolio Theory and Risk Management.* Cambridge.

Michaud, R. O. (1989-Volume 45,Issue 1). The Markowitz Optimization Enigma: Is 'Optimized' Optimal? *Financial Analysts Journal*, 31-42.

Olivier Ledoit, M. W. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis(88)*, 365–411.