Learning exposure profiles for portfolios of exotic derivatives

Kristoffer Andersson

Centrum Wiskunde & Informatica

Machine Learning in Quantitative Finance and Risk Management,
July 2, 2020

Table of contents

Financial exposure

The exposure of an investment can be defined as:

"The amount an investor stands to lose if the counyterparty defaults."

We are interested in the exposure of a portfolio of derivatives

$$E_t^{\text{Net}} = \max\left\{\sum_{j=1}^{J} V_j(t, X_t), 0\right\}, \quad E_t = \sum_{j=1}^{J} \max\{V_j(t, X_t), 0\}.$$

The distributions of $E^{\text{Net}} = (E_t^{\text{Net}})_{t\in[0,T]}$ and $E = (E_t)_{t\in[0,T]}$ are referred to as ***exposure profiles***.

Why exposures?

The exposure is an important building block in computations of ***Valuation Adjustments*** (XVAs).

$$
\begin{aligned}
\text{Credit Valuation Adjustment:} \quad & \mathrm{CVA} = \mathbb{E}\left[D_{0,\tau^C}\,\mathrm{LGD}^C_{\tau^C}\,E_{\tau^C}\right], \\
\text{Debit Valuation Adjustment:} \quad & \mathrm{DVA} = \mathbb{E}\left[D_{0,\tau^B}\,\mathrm{LGD}^B_{\tau^B}\,E_{\tau^B}\right], \\
\text{Funding Valuation Adjustment:} \quad & \mathrm{FVA} = \mathbb{E}\left[\int_0^{\tau^B} D_{0,t}\,\mathrm{FS}^B_t\,E_t\,\mathrm{d}t\right], \\
\text{Capital Valuation Adjustment:} \quad & \mathrm{KVA} = \mathbb{E}\left[\int_0^{T} D_{0,t}\,K_t\,\mathrm{RWA}_t\left(E\right)\,\mathrm{d}t\right], \\
\vdots \quad & \vdots = \quad \vdots
\end{aligned}
$$

Exposure profiles

- In general, no access to the distribution of the exposure.
- In special cases, possible to draw random samples $(E_t(\omega))_{t\in[0,T]}$, distributed exactly as E.
- In most cases, the best we can do is to draw samples from a distribution, which in some sense, is close to E.

The distribution of E is approximated with its empirical counterpart. Common measures are:

$$\text{Expected exposure:}\quad \mathrm{EE}(t) = \mathbb{E}\left[E_t\right],$$
$$\text{Potential future exposure:}\quad \mathrm{PFE}_\alpha(t) = \inf\left\{a \in \mathbb{R} \,\middle|\, \mathbb{Q}\left(\mathrm{E}_t \leq a\right) \geq \alpha\right\}.$$

Example - European call option under Black–Scholes dynamics

Consider the single European call option, *i.e.*, $J = 1$,

$$S_t = s_0 e^{(r - \frac{\sigma^2}{2})(T-t) + \sigma W_t}, \quad g(S_T) = \max\{S_T - K, 0\}.$$

Option value, given market state (t, S_t):

$$V(t, S_t) = \mathbb{E}\left[e^{-r(T-t)} g(S_T) \,\middle|\, S_t\right] = N(d_1) S_t + N(d_2) K e^{-r(T-t)},$$

where $d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[\log \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$ and $d_2 = d_1 - \sigma\sqrt{T-t}$.
We can sample exactly from E^{BS} by

$$\left(E_t^{\mathrm{BS}}(\omega)\right)_{t\in[0,T]} = \left(\max\{V(t, S_t(\omega)), 0\}\right)_{t\in[0,T]} = \left(V(t, S_t(\omega))\right)_{t\in[0,T]}.$$

How to sample from an approximate distribution?

Approximations are needed *e.g.*, when we have a less idealized asset process than GBM, early exercise features, etc.

Classical methods for sampling from the exposure process:

- Approximate the valuation PDE with *e.g.*, finite elements/differences, and evaluate the solution along stochastic paths of the underlying asset/risk-factor process,
- Approximate conditional expectations with least squares regression,
- Approximate the valuation BSDE pathwise with *e.g.*, monte–carlo based regression,
- Approximate the valuation function with Fourier-based expansion and evaluate the solution along stochastic paths of the underlying asset/risk-factor process.

Our aim is to introduce a method, capable of approximating the exposure profiles for a portfolio of (potentially) high-dimensional and exotic, derivatives.

- J derivatives and a risk-factor process, X, taking on values in $\mathbb{R}^d$,
- Denote by $\boldsymbol{\mathcal{T}}(t) = \{\mathcal{T}_1(t), \mathcal{T}_2(t) \ldots, \mathcal{T}_J(t)\}$ the space of all $X-$stopping times vectors taking on values in $\mathbb{T}(t) = \{\mathbb{T}_1(t), \mathbb{T}_2(t), \ldots, \mathbb{T}_J(t)\} \subseteq [0, T]^J$.

Then, the valuation function of the portfolio is given by

$$\Pi(t,x) = \sup_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}(t)} \sum_{j=1}^{J} \mathbb{E}_{t,x}\left[D_{t,\tau_j} g_j(X_{\tau_j})\right].$$

and the exposures at market state, (t, X_t), are given by

$$\mathrm{E}_t^{\mathrm{Net}} = \max\left\{\sum_{j=1}^{J} V_j(t, X_t)\mathbb{I}_{\{\tau^*_{\mathbf{0},j} > t\}}, 0\right\}, \quad \mathrm{E}_t = \sum_{j=1}^{J} \max\left\{V_j(t, X_t)\mathbb{I}_{\{\tau^*_{\mathbf{0},j} > t\}}, 0\right\},$$

where $\tau^*_{0,j}$ is the stopping strategy that satisfies the expression above. Note that E is not a Markov process but $\left(E_t, \mathbb{I}_{\{\tau^*_{\mathbf{0},\mathbf{1}} > t\}}, \ldots, \mathbb{I}_{\{\tau^*_{\mathbf{0},J} > t\}}\right)_{t \in [0,T]}$ is.

High level picture of algorithm

Divide problem into two sub-problems and solve each sub-problem separately:

Phase I: Use neural networks to learn the optimal stopping rule from Monte-Carlo samples of the underlying risk factors.

Phase II: Apply the stopping rule from Phase I on Monte-Carlo samples from the underlying risk factors to generate cashflow-paths. Use neural networks to learn the mapping from the underlying risk factors to the exposure/portfolio value by using the cashflows as "labels" (minimize the MSE between our approximation and the cashflow-paths).

References

Phase I is a generalization of the so-called ***Deep Optimal Stopping*** algorithm (Becker et al. *"Deep Optimal Stopping."* Journal of Machine Learning Research 20.74 (2019): 1-25.).

Presentation is based on:
K. Andersson, & C. W. Oosterlee. *"A deep learning approach for computations of exposure profiles for high-dimensional Bermudan options."* arXiv preprint arXiv:2003.01977, (2020).

K. Andersson, & C. W. Oosterlee. *"Learning exposure profiles for portfolios of exotic derivatives."* working paper, (2020).

Outline

Continuation/exercise regions and decision functions

For derivative $j \in \{1, 2, \ldots, J\}$,

Exercise region: $\mathcal{E}_j(t) = \Big\{x \in \mathbb{R}^d \mid V_j(t,x) = g_j(t,x) \ \text{and} \ t \in \mathbb{T}_j(0)\Big\}$,

Continuation region: $\mathcal{C}_j(t) = \Big\{x \in \mathbb{R}^d \mid V_j(t,x) > g_j(t,x) \ \text{or} \ t \notin \mathbb{T}_j(0)\Big\}$,

Optimal decision function: $f_j^*(t,x) = \mathbb{I}_{\{x \in \mathcal{E}_j(t)\}}$.

We then define the (optimal) decision vector, consisting of J, (optimal) decision functions

$$\boldsymbol{f}^*(t,x) = (f_1^*(t,x), f_2^*(t,x), \ldots, f_J^*(t,x))^T.$$

Stopping times in terms of decision functions

Denote the set of exercise dates (the dates where at least one of the derivatives may be exercised) by $\mathbb{T}^{\Pi}(t) = \bigcup_{j=1}^{J} \mathbb{T}_j(t)$ and the number of exercise dates by $N = |T^{\Pi}(0)|$. We use the simplified notation

$$\mathbb{T}^{\Pi}(0) = \{T_1, T_2, \ldots, T_N\}.$$

The optimal exercise strategy (at portfolio level) is then given by the $X-$stopping time

$$\boldsymbol{\tau}[\boldsymbol{f}^*](X) = \sum_{k=1}^{N} T_k \boldsymbol{f}^*(T_k, X_{T_k}) \odot \prod_{m=1}^{k-1} (\mathbf{1}_J - \boldsymbol{f}^*(T_m, X_{T_m})),$$

or written component-wise

$$\tau_j[f_j^*](X) = (\boldsymbol{\tau}[\boldsymbol{f}^*](X))_j = \sum_{k=1}^{N} T_k f_j^*(T_k, X_{T_k}) \prod_{m=1}^{k-1} \left(1 - f_j^*(T_m, X_{T_m})\right).$$

Valuation function in terms of the decision vector

Let $t \in (T_{n-1}, T_n]$, and introduce short-hand notation

$$\boldsymbol{\tau}_n^* = \boldsymbol{\tau}[\boldsymbol{f}^*](X^{t,x}), \quad \text{and} \quad \tau_{n,j}^* = \left(\boldsymbol{\tau}[\boldsymbol{f}^*](X^{t,x})\right)_j,$$

and the portfolio value can be written as

$$\Pi(t,x) = \sum_{j=1}^{J} \mathbb{E}_{t,x}\left[D_{t,\tau_{n,j}^*} g_j(\tau_{n,j}^*, X_{\tau_{n,j}^*})\right].$$

Represent decision functions by neural networks and introduce loss function

For $n \in \{1, 2, \dots, N\}$, we replace the optimal decision function $\boldsymbol{f}^*(T_n, \cdot)$ with a neural network $\boldsymbol{f}_n^{\theta_n} : \mathbb{R}^d \to \{0, 1\}^J$.

Want to find set of parameters $\Theta_1 = \{\theta_1, \theta_2, \dots, \theta_N\}$, such that

$$(\boldsymbol{f}^*(T_1, \cdot), \boldsymbol{f}^*(T_2, \cdot), \dots, \boldsymbol{f}^*(T_N, \cdot))^T \approx (\boldsymbol{f}_1^{\theta_1}, \boldsymbol{f}_2^{\theta_2}, \dots, \boldsymbol{f}_N^{\theta_N})^T = \boldsymbol{f}_1^{\Theta_1}.$$

Want to find θ_n, such the loss function is minimized

$$-\mathbb{E}_{T_n}\left[\sum_{j=1}^{J} \left(\boldsymbol{f}_n^{\theta_n}(X_{T_n})\right)_j g_j(T_n, X_{T_n}) + \left(1 - \left(\boldsymbol{f}_n^{\theta_n}(X_{T_n})\right)_j\right) D_{T_n, \tau^*_{n+1,j}} g_j\left(\tau^*_{n+1,j}, X_{\tau^*_{n+1,j}}\right)\right].$$

Problems:

1. In general, we have no access to the expected value above.
   Solution: Approximate with sample mean.
2. $\boldsymbol{f}_n^{\theta_n}$ is discontinuous, which makes a gradient decent type algorithm unsuitable.
3. We have no access to $\boldsymbol{\tau}^*_{n+1}$.

Loss function

2. **Solution:** While optimizing, replace the discontinuous function $\boldsymbol{f}_n^{\theta_n}$, with $\boldsymbol{F}_n^{\theta_n} : \mathbb{R}^d \to (0,1)^J$. After optimization, set

$$\boldsymbol{f}_n^{\theta_n} = \mathfrak{a} \circ \boldsymbol{F}_n^{\theta_n},$$

   where $\mathfrak{a}$ is the component-wise round-off function $(\mathfrak{a}(x))_j = \mathbb{I}_{\{x_j \geq \frac{1}{2}\}}$.

3. **Solution:** If derivative j has maturity at some $n \in \{1, 2, \dots, N\}$, we know that

$$f_{j,n}^{\theta_n}(\,\cdot\,) \equiv \mathbb{I}_{\{g_j(T_n,\,\cdot\,)>0\}}, \quad \text{and} \quad f_{j,k}^{\theta_k}(\,\cdot\,) \equiv 0, \text{ for } k > n.$$

   Therefore, at maturity of the portfolio, the optimal decision function is known and we can set $\boldsymbol{f}_N^{\theta_N} = \boldsymbol{f}^*(T_N, \cdot)$. The optimization can then be carried out, recursively, backwards in time.

Comments on the structure of the neural networks

- We use 1-3 hidden layers, with 15-30 nodes in each hidden layer,
- We use the (component-wise) ReLU activation function in the hidden layers and the (component-wise) sigmoid function in the output layer to guarantee that all component are mapped to $(0, 1)$,
- The Adam optimizer is used to optimize the trainable parameters,
- More details can be found in references.

Algorithm

Sample M_{train} training samples $(x_t^{\text{train}}(m))_{t\in[0,T]}$, distributed as X. For $m \in \{1,2,\dots,M_{\text{train}}\}$, and $j \in \{1,2,\dots,J\}$, set $\mathrm{CF}_{N,j}(m) = g_j(T_N, x_{t_N}^{\text{train}}(m))$

For $n = N-1, N-2, \dots, 1$, do the following:

1. Find a $\hat{\theta}_n \in \mathbb{R}^{q_n}$ which approximates

$$\hat{\theta}_n^* \in \underset{\theta\in\mathbb{R}^{q_n}}{\arg\max} \left(\frac{1}{M_{\text{train}}} \sum_{m=1}^{M_{\text{train}}} \sum_{j=1}^{J} \left(\boldsymbol{F}_n^{\theta} \left(x_{T_n}^{\text{train}}(m) \right) \right)_j g_j\left(T_n, x_{T_n}^{\text{train}}(m)\right) \right.$$
$$\left. + \left(1 - \left(\boldsymbol{F}_n^{\theta} \left(x_{T_n}^{\text{train}}(m) \right) \right)_j \right) \mathrm{CF}_{n+1,j}(m) \right).$$

2. For all j and m, update $CF_{n,j}(m)$:

$$\mathrm{CF}_{n,j}(m) = \left(\boldsymbol{f}_n^{\hat{\theta}_n} \left(x_{T_n}^{\text{train}}(m) \right) \right)_j g_j\left(T_n, x_{T_n}^{\text{train}}(m)\right)$$
$$+ \left(1 - \left(\boldsymbol{f}_n^{\hat{\theta}_n} \left(x_{T_n}^{\text{train}}(m) \right) \right)_j \right) D_{T_n, T_{n+1}} \mathrm{CF}_{n+1,j}(m).$$

Portfolio valuation

Sample M_{val} valuation samples, $\left(x_t^{\text{val}}(m)\right)_{t\in[0,T]}$, distributed as X. Denote the vector of optimized decision functions by

$$\boldsymbol{f}_n^{\hat{\Theta}_n} = \left(\boldsymbol{f}_n^{\hat{\theta}_n},\, \boldsymbol{f}_{n+1}^{\hat{\theta}_{n+1}}, \ldots, \boldsymbol{f}_{N-1}^{\hat{\theta}_{N-1}}\right),$$

and $\boldsymbol{f}^{\hat{\Theta}} = \boldsymbol{f}_1^{\hat{\Theta}_1}$. We then obtain for sample m, *i.e.*, $x^{\text{val}}(m)$, the following stopping rule

$$\tau_{0,j}^{\hat{\Theta}}(m) = \left(\boldsymbol{\tau}\left[\boldsymbol{f}^{\hat{\Theta}}\right]\left(x^{\text{val}}(m)\right)\right)_j = \sum_{k=n}^{N} T_k \left(\boldsymbol{f}_k^{\hat{\theta}_k}\left(x_{T_k}^{\text{val}}(m)\right)\right)_j \prod_{\ell=1}^{k-1}\left(1-\left(\boldsymbol{f}_\ell^{\hat{\theta}_\ell}\left(x_{T_\ell}^{\text{val}}(m)\right)\right)_j\right).$$

The estimated portfolio value at $t=0$ is then given by

$$\hat{\Pi}(0, x_0) = \frac{1}{M_{\text{val}}}\sum_{m=1}^{M_{\text{val}}}\sum_{j=1}^{J}\frac{g_j\left(\tau_{0,j}^{\hat{\Theta}}(m),\, x_{\tau_{0,j}^{\hat{\Theta}}(m)}^{\text{val}}(m)\right)}{B_{\tau_{0,j}^{\hat{\Theta}}(m)}}.$$

Outline

Using stopping rule from Phase I

For $t \in (T_{n-1}, T_n]$ denote the vector-valued discounting process and pay-off function by

$$\boldsymbol{D}(t, \boldsymbol{\tau}_n^*) = \begin{pmatrix} D(t, \tau_{n,1}^*) \\ \vdots \\ D(t, \tau_{n,J}^*) \end{pmatrix}, \quad \boldsymbol{g}(\boldsymbol{\tau}_n^*, X^{t,x}) = \begin{pmatrix} g_1(\tau_{n,1}^*, X_{\tau_{n,1}^*}) \\ \vdots \\ g_J(\tau_{n,J}^*, X_{\tau_{n,J}^*}) \end{pmatrix},$$

and the vector-valued process of discounted cashflows by

$$\boldsymbol{Y}_t = \boldsymbol{D}_{t,\boldsymbol{\tau}_n^*} \odot \boldsymbol{g}(\boldsymbol{\tau}_n^*, X_{\boldsymbol{\tau}_n^*}), \quad Y_{t,j} = (\boldsymbol{Y}_t)_j.$$

Furthermore, define the exercise process

$$\mathbb{I}_t = \begin{pmatrix} \mathbb{I}_{\{\tau_{\mathbf{0},1}^* < t\}} \\ \vdots \\ \mathbb{I}_{\{\tau_{\mathbf{0},J}^* < t\}} \end{pmatrix}.$$

Note! Neither $\boldsymbol{Y}_t$ nor $\mathbb{I}_t$ are $\mathcal{F}_t-$measurable.

Regression function

Recall that for $t, s \in [0, T]$, with $t \leq s$ and for $Z\colon [0, T] \times \Omega \to \mathbb{R}^a$, square integrable, a function m which satisfies

$$m(t, \cdot) \in \underset{\boldsymbol{h} \in \mathcal{D}(\mathbb{R}^a; \mathbb{R}^b)}{\arg\min} \; \mathbb{E}_t \left[\| \boldsymbol{h}(X_t) - Z_s] \|_2^2 \right],$$

is given by the conditional expectation (assuming X is a Markov process)

$$m(t, x) = \mathbb{E}_{t,x}[Z_s].$$

Regression functions in our context

Recall the equations for the exposures

$$E_t^{\text{Net}} = \max\left\{\sum_{j=1}^{J} V_j(t, X_t)\mathbb{I}_{\{\tau^*_{\mathbf{0},j} > t\}},\, 0\right\}, \quad E_t = \sum_{j=1}^{J} \max\left\{V_j(t, X_t)\mathbb{I}_{\{\tau^*_{\mathbf{0},j} > t\}},\, 0\right\}.$$

For $t \in (T_{n-1}, T_n]$, denote $\boldsymbol{V}(t,\,\cdot\,) = (V_1(t,\,\cdot\,), \ldots V_J(t,\,\cdot\,))^T$. The essential parts of E^{Net} and E can the expressed as regression functions

$$\boldsymbol{V}(t,\cdot) \in \underset{\boldsymbol{h}\in\mathcal{D}(\mathbb{R}^d;\mathbb{R}^J)}{\arg\min}\ \mathbb{E}_t\left[\|\boldsymbol{h}(X_t) - \boldsymbol{Y}_t\|_2^2\right],$$

$$\sum_{j=1}^{J} V_j(t,\cdot)\mathbb{I}_{\{\tau^*_{\mathbf{0},j} > t\}} \in \underset{h\in\mathcal{D}(\mathbb{R}^d\times\{0,1\}^J;\mathbb{R})}{\arg\min}\ \mathbb{E}_t\left[\left|h(X_t, \mathbb{I}_t) - \sum_{j=1}^{J} Y_{t,j}\mathbb{I}_{\{\tau^*_{\mathbf{0},j} < t\}}\right|^2\right].$$

Approximating the regression functions

Given M_{reg} samples, $(x_t^{\text{reg}})_{t\in[0,T]}$, distributed as X, and for $t \in (T_{n-1}, T_n]$, we define the empirical regression problems

$$\min_{\boldsymbol{h}\in\mathcal{D}} \left\{ \frac{1}{M_{\text{reg}}} \sum_{m=1}^{M_{\text{reg}}} \|\boldsymbol{h}(x_t(m)) - \boldsymbol{y}_t(m)\|_2^2 \right\}, \tag{1}$$

$$\min_{h\in\mathcal{D}} \left\{ \frac{1}{M_{\text{reg}}} \sum_{m=1}^{M_{\text{reg}}} \left| h(x_t(m), \mathbb{I}_t(m)) - \sum_{j=1}^{J} y_{t,j}(m) \mathbb{I}_{\{\tau_{n,j}^*(m)<t\}} \right|^2 \right\}. \tag{2}$$

The classical way to approximate the regression function is ***Least Squares Monte-Carlo*** (LSMC) regression. Search for functions in a smaller function class, *e.g.*, polynomials of the components of X_t up to degree 4, including cross-terms.

- 👍 Closed form solution when $\mathcal{D}$ is the class of linear combinations of polynomials,
- 👍 Overcomes the curse of dimensionality (but may run into memory issues instead),
- 👎 Not easy to choose appropriate basis functions when function surface to approximate is complicated,
- 👎 Not trivial to approximate a vector valued function in (1) or how to handle $\mathbb{I}_t(m)$ in (2) (piecewise linear regression?).

Neural network-based regression

By letting $\mathcal{D}$ be the class of functions which can be represented by a neural network (with all hyperparameters specified), we can optimize the parameters in the neural network to approximately solve (1) and (2).

- Possible to specify larger function classes (than for LSMC) without running out of memory,
- No need to specify clever basis functions, since difficulty rather lies in solving the optimization problem,
- No closed form solution for optimization problem (black box),
- More time consuming than LSMC.

Summary: In our experience, for $J = 1$, with low-dimensional X and relatively simple pay-off functions LSMC is to prefer. For $J > 1$, or/and high-dimensional X, or/and complicated pay-off functions neural network-based regression seems to be more accurate.

Neural network-based regression

For $n \in \{1, 2, \ldots, N\}$, use neural networks of the form

$$\boldsymbol{h}_n^{\Phi_n} \colon \mathbb{R}^d \to \mathbb{R}^J, \quad \text{and} \quad h_n^{\Phi_n} \colon \mathbb{R}^d \times \{0,1\}^J \to \mathbb{R}$$

with empirical loss functions given by

$$\frac{1}{M_{\text{reg}}} \sum_{m=1}^{M_{\mathbf{reg}}} \| \boldsymbol{h}_n^{\Phi_n^1}(x_t(m)) - \boldsymbol{y}_t(m) \|_2^2,$$

$$\frac{1}{M_{\text{reg}}} \sum_{m=1}^{M_{\mathbf{reg}}} \Big| h_n^{\Phi_n^2}(x_t(m), \mathbb{I}_t(m)) - \sum_{j=1}^{J} y_{t,j}(m) \mathbb{I}_{\{\tau^*_{\mathbf{0},j}(m) < t\}} \Big|^2.$$

Outline

Black–Scholes dynamics

The only risk factor is the $d-$dimensional asset process, with component $i \in \{1, 2, \ldots, d\}$ given by

$$(S_t)_i = (s_0)_i \exp\left((r - q_i - \frac{1}{2}\sigma_i^2)t + \sigma_i (W_t)_i \right),$$

with risk-free rate r, continuously paying dividend (of asset i) q_i, diffusion coefficient σ_i, and W a $d-$dimensional, correlated Brownian motion.

Max-call-option:

$$g(s) = (\max\{s_1, s_2 \ldots, s_d\} - K)^+ .$$

Arithmetic-average-option:

$$g(s) = \left(\frac{1}{d} \sum_{i=1}^{d} s_i - K \right)^+ .$$

Geometric-average-option:

$$g(s) = \left(\left(\prod_{i=1}^{d} s_i \right)^{\frac{1}{d}} - K \right)^+ .$$

Numerical results

We consider a portfolio of the put and call versions of Bermudan max-options, arithmetic-average-option, and geometric average option with 10 exercise dates.

	M-call	**M-put**	**A-call**	**A-put**	**G-call**	**G-put**
Ref	13.902	9.530	NA	NA	NA	NA
DOS	13.899	9.528	4.364	16.775	4.930	15.318
SGBM	13.921	9.535	4.366	16.781	4.941	15.319
LSMC	13.851	9.520	4.363	16.778	4.927	15.309

The above results are the average values of 5 independent runs. For LSMC and SGBM each derivative value is computed individually, for DOS all are computed in one run.

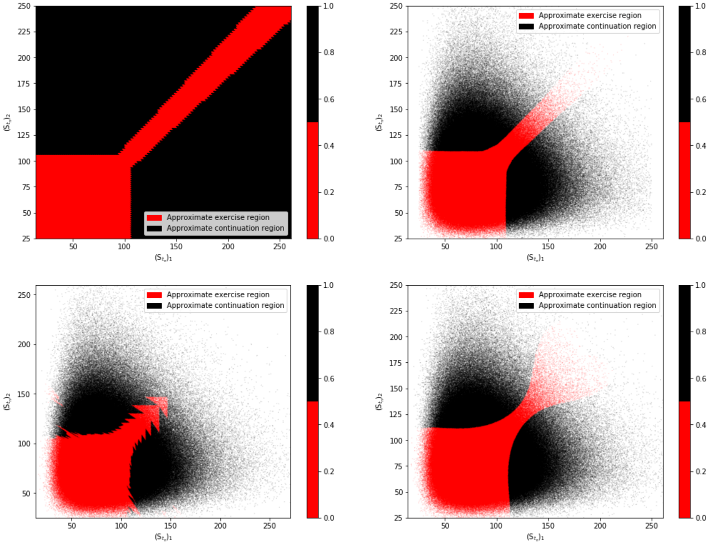

Figure: Approximate exercise boundaries for a two-dimensional max-call option at $t_8 \approx 2.67$. **From top left to bottom right:** FEM (American option), DOS, SGBM and LSMC respectively.

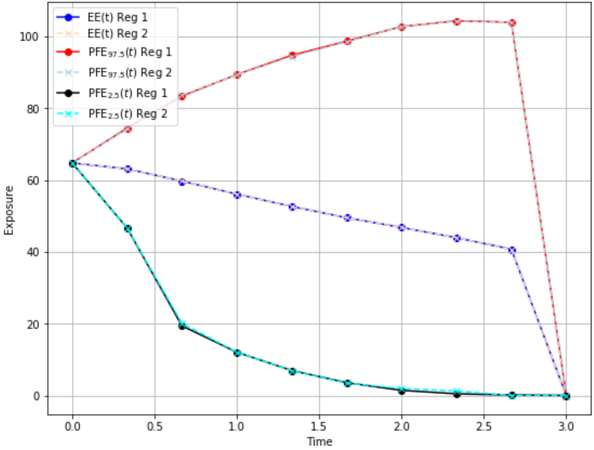

Figure: EE, $PFE_{97.5}$, and $PFE_{0.25}$ computed with neural network-based regression with outputs in $\mathbb{R}$ and $\mathbb{R}^J$, respectively.

Outline

In this example we consider a portfolio of 6 Bermudan interest rate swaptions with partially overlapping exercise dates, and different strike prices.

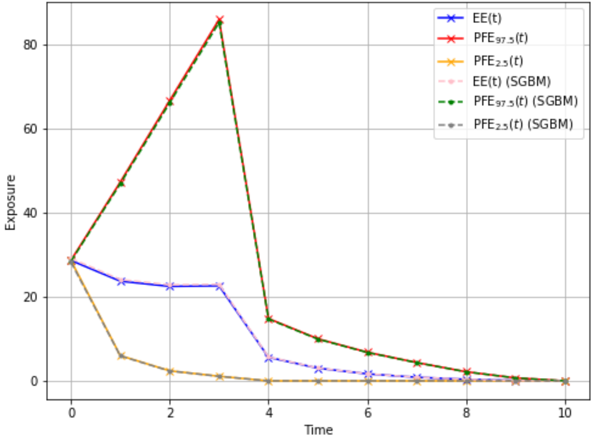

Figure: EE, $PFE_{97.5}$, and $PFE_{0.25}$ computed with neural network-based regression and with SGBM (individually for each derivative).