Stochastic Simulation: Lecture 16

Christoph Reisinger

Oxford University Mathematical Institute

Modified from earlier slides by Prof. Mike Giles.

Simulation and deep learning

In this lecture, we give an overview of neural networks enhancing Monte Carlo methods.

We give some general methodology and three case studies from finance:

- policy gradient methods, e.g. in optimal allocation problems;
- deep optimal stopping;
- deep BSDE solver, e.g. for valuation adjustments.

Need following ingredients (see also "Theories of Deep Learning"):

- (dynamic programming and BSDEs;)
- neural network architectures;
- (stochastic) gradient descent optimisation.

Stochastic control

Consider

$$dX_t \quad = \quad b(t, X_t; \alpha_t)\, dt + \sigma(t, X_t; \alpha_t)\, dW_t, \qquad X_0 = x,$$

where $(\alpha_t)_t$ is a suitable admissible control process.

The control is chosen such that

$$\mathbb{E}\left[\int_0^T f(X_t, \alpha_t)\, dt \,+\, g(X_T)\right] \quad \longrightarrow \quad \min_{\alpha}.$$

Can be formulated as

- (Hamilton–Jacobi–Bellman) PDE via dynamic programming;
- FBSDE via stochastic maximum principle.

Policy gradient methods

Can also write the control in feedback form, $\alpha_t = a(t, X_t)$. Then

- parametrize as $a(t, X_t; \rho)$;
- discretize X by Euler–Maruyama,

$$\widehat{X}_{n+1}^{\rho} = \widehat{X}_n^{\rho} + b(t_n, \widehat{X}_n^{\rho}; a(t_n, \widehat{X}_n^{\rho}; \rho)))\, \Delta t + \sigma(t_n, \widehat{X}_n^{\rho}; a(t_n, \widehat{X}_n^{\rho}; \rho))\, \Delta W_n;$$

- generate M samples $\widehat{X}_n^{\rho,(m)}$ and solve

$$\frac{1}{M} \sum_{m=1}^{M} \sum_{n=0}^{N-1} f(\widehat{X}_n^{\rho,(m)}, a(t_n, \widehat{X}_n^{\rho,(m)}; \rho))\Delta t + g(\widehat{X}_N^{\rho,(m)}) \quad \rightarrow \quad \min_{\rho}.$$

Multiperiod optimal investment

Reference: *A Data Driven Neural Network Approach to Optimal Asset Allocation for Target Based Defined Contribution Pension Plans*, Yuying Li and Peter Forsyth (2019).

Consider:

- M risky and risk-free assets, with (Markovian) price process $S(t) = (S_m(t))_{1 \le m \le M}$.
- Intervention times $\mathcal{T} = \{0 = t_0 < t_1 < \ldots < t_N = T\}$.
- Returns $R(t_n) = (R_m(t_n))_{1 \le m \le M}$.
- A fraction ρ_n^m invested in the m-th asset in (t_n, t_{n+1}).
- The total wealth $W(t_n)$.
- Cash injections $q(t_n)$ at time t_n.

Model and objective

Then we have, for $n = 0, 1, \dots, N-1$:

$$\begin{aligned} W(t_n^+) &= W(t_n^-) + q(t_n) \\ W(t_{n+1}^-) &= \rho_n^T R(t_n) W(t_n^+) \end{aligned}$$

The investor aims to solve the minimisation problem

$$\min_{\{\rho_0,\dots,\rho_{N-1}\}} \quad g(W(T)) = \mathbb{E}\left[\min(W(T) - W^*, 0)^2\right]$$

$$\text{subject to} \quad \begin{aligned} &0 \le \rho_n \le 1, \quad n = 0, 1, \dots, N-1 \\ &\mathbf{1}^T \rho_n = 1, \quad n = 0, 1, \dots, N-1 \end{aligned}$$

for a target W^*.

- Related to mean-variance optimisation problem.
- Could allow short-selling, leverage constraints, etc.

Parametrization

- For small M, can solve HJB (Markovian case).
- Here, optimise directly over ρ by simulation.
- [F&L (19)] use $\rho_n = p(F(t_n))$, $F(t)$ a d-vector of features;
- satisfy the constraints by construction:

$$p_m(F(t_n)) = \frac{\mathrm{e}^{\sum_k x_{km} h_k(F(t_n))}}{\sum_i \mathrm{e}^{\sum_k x_{ki} h_k(F(t_n))}}, \qquad m = 1, \ldots, M,$$

- where

$$h_j(F(t_n)) = \sigma\left(\sum_i F_i(t_n) z_{ij}\right), \qquad \sigma(u) = \frac{1}{1 + \mathrm{e}^u},$$

- and $z \in \mathbb{R}^{d \times l}$, $x \in \mathbb{R}^{l \times M}$ are the weights of the output and input layer, respectively.

Optimisation

The optimisation problem becomes

$$\min_{z \in \mathbb{R}^{d\times l},\, x \in \mathbb{R}^{l\times M}} \mathbb{E}\left[\min(W(T) - W^*, 0))^2\right]$$

where W is determined from z and x, and F, as above.

- Estimate expectation with L sample paths of S, F, W;
- features can be S itself;
- cost of gradient: $O(l(d+M)NL)$; cost of Hessian: $O(l^2(d+M)^2NL)$ (see [F&L (19)]);
- in the [F&L (19)] application, $l(d+M)$ small and trust region method feasible;
- otherwise SGD.

Optimal stopping

Key reference: *Deep optimal stopping*: Sebastian Becker, Patrick Cheridito, Arnulf Jentzen (2020).

Consider:

- a discrete-time Markov process $(X_n)_{n=1\ldots N}$ in $\mathbb{R}^d$;
- an optimal stopping problem
$$\sup_{\tau \in \mathcal{T}} \mathbb{E}[g(\tau, X_\tau)];$$
- auxiliary problems
$$\sup_{\tau \in \mathcal{T}_n} \mathbb{E}[g(\tau, X_\tau)],$$
where $\mathcal{T}_n = \{\tau \in \mathcal{T} : \tau \geq n\}$.

NN approximation

- Define functions $f_m : \mathbb{R}^d \to \{0, 1\}$ and
- candidate stopping times

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

- Approximation with trial functions f^θ,

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

- where $f^\theta = \Psi \circ \psi^\theta$, $\Psi(x) = 1/(1 + \exp(-x))$ and ψ^θ a NN parametrised by θ.
- Optimise recursively over θ.

FBSDEs (again)

Recall the FBSDE

$$
\begin{aligned}
dX_t &= b(t, X_t)\,dt + \sigma(t, X_t)\,dW_t, \qquad X_0 = x; \\
dY_t &= f(t, X_t, Y_t, Z_t)\,dt + Z_t\,dW_t, \qquad Y_T = h(X_T).
\end{aligned}
$$

Discretize (forward):

$$
\begin{aligned}
\widehat{X}_{n+1} &= \widehat{X}_n + b(t_n, \widehat{X}_n)\,\Delta t + \sigma(t_n, \widehat{X}_n)\,\Delta W_n, \\
\widehat{Y}_n &= \widehat{Y}_n + f(\widehat{X}_n, \widehat{Y}_n, \widehat{Z}_n)\,\Delta t + \widehat{Z}_n\,\Delta W_n.
\end{aligned}
$$

Use a "shooting method" to optimise over Z for Y to "hit" h at T.

Deep BSDE solver

- Parametrize $\widehat{Z}_n = \hat{z}_n(\widehat{X}_n; \rho)$, where $\hat{z}_n$ is a parametric function of x and ρ a parameter; denote the resulting Y for given ρ and $Y_0 = \xi$ by $\widehat{Y}^{\rho,\xi}$.
- In the "deep" solver, $\hat{z}_n$ is a multi-layer, fully connected, neural network with the parameter ρ containing the weights and biases.
- Now write the (discrete) FBSDE as optimisation problem:

$$\mathbb{E}[(\widehat{Y}_N^{\rho,\xi} - h(\widehat{X}_N))^2] \quad \to \quad \min_{\rho,\xi}.$$

- In practice, generate M samples $(\widehat{X}^{(m)}, \widehat{Y}^{\rho,\xi,(m)})$ and solve

$$\frac{1}{M}\sum_{m=1}^{M}(\widehat{Y}_N^{\rho,\xi,(m)} - h(\widehat{X}_N^{(m)}))^2 \quad \to \quad \min_{\rho,\xi}.$$

Error bounds

Define a suitable continuous-time interpolant $(\widetilde{X}_t, \widetilde{Y}_t, \widetilde{Z}_t)$. Then

$$\sup_{0\leq t\leq T}\left(\mathbb{E}|X_t-\widetilde{X}_t|^2+\mathbb{E}|Y_t-\widetilde{Y}_t|^2\right)+\int_0^T\mathbb{E}|Z_t-\widetilde{Z}_t|^2\,dt$$
$$\leq C\left(\Delta t+\mathbb{E}|\widetilde{Y}_T-h(\widetilde{X}_T)|^2.\right)$$

- These bounds are "a posteriori", i.e. the r-h-s can be estimated from the numerical solution (subject to C);
- the first term on the r-h-s is the time stepping error;
- the second term includes the optimisation error;
- also hold in the coupled case with $b(t, X_t, Y_t)$, $\sigma(t, X_t, Y_t)$;
- see J. Han & J. Long, Convergence of the deep BSDE method for coupled FBSDEs, Probability, Uncertainty and Quantitative Risk, 2020.

Counterparty credit risk

References:

Financial Modeling, A Backward Stochastic Differential Equations Perspective, Stephane Crépey (2013).

Deep learning-based numerical methods for ... backward stochastic differential equations, W. E, J. Han and A. Jentzen (2017).

- Two agents: the bank (B, our perspective), the counterparty (C);
- Default times: τ^j, for $j \in \{B, C\}$ and $\tau = \min(\tau^B, \tau^C\}$;
- Risky assets: $X_t = (X_t^1, \ldots, X_t^d)$ solution of a SDE;
- Cash accounts: B_t^j, $j \in \{B, C\}$;
- Collaterals: C_t exchanged between the parties.

Valuation adjustments

Banks need to compute Credit Valuation Adjustments (CVA), Debt Valuation Adjustments (DVA), Funding Valuation Adjustments (FVA), and other adjustments (xVAs).

Consider a portfolio of M (European) contingent claims:

$$Y_t^m = \mathbb{E}^{\mathbb{Q}}\left[e^{-r_m(T_m-t)} g_m(X_{T_m}) | \mathcal{F}_t\right], \qquad m = 1, \dots, M, \quad t \in [0, T],$$

solving the following (decoupled) FBSDE:

$$\begin{cases} \mathrm{d}X_t^m = \mu(t, X_t)\,\mathrm{d}t - \sigma(t, X_t)\,\mathrm{d}W_t^{\mathbb{Q}}, \\ -\mathrm{d}Y_t^m = -r_t Y_t^m\,\mathrm{d}t - \sum_{k=1}^d Z_t^{k,m}\,\mathrm{d}W_t^{k,\mathbb{Q}}, \\ X_0 = x, \\ Y_{T^m}^m = g_m(X_{T_m}). \end{cases}$$

Valuation adjustments

Let $\overline{Y}_t := \sum_{m=1}^{M} Y_t^m$ and $t < \tau$ (pre-default). Consider:

$$XVA_t = -CVA_t + DVA_t + FVA_t$$

where (λ the default intensities, r risk-free rates)

$$CVA_t := B_t^{\tilde{r}}\, \mathbb{E}^{\mathbb{Q}}\Big[(1 - R^C) \int_t^T \frac{1}{B_u^{\tilde{r}}} \left(\overline{Y}_u - C_u\right)^- \lambda_u^{C,\mathbb{Q}} \,\mathrm{d}u \Big| \mathcal{F}_t\Big],$$

$$DVA_t := B_t^{\tilde{r}}\, \mathbb{E}^{\mathbb{Q}}\Big[(1 - R^B) \int_t^T \frac{1}{B_u^{\tilde{r}}} \left(\overline{Y}_u - C_u\right)^+ \lambda_u^{B,\mathbb{Q}} \,\mathrm{d}u \Big| \mathcal{F}_t\Big],$$

$$FVA_t := B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\Big[\int_t^T \frac{(r_u^{f,l} - r_u)\left(\overline{Y}_u - XVA_u - C_u\right)^+}{B_u^{\tilde{r}}} \,\mathrm{d}u \Big| \mathcal{F}_t\Big]$$

$$- B_t^{\tilde{r}}\, \mathbb{E}^{\mathbb{Q}}\Big[\int_t^T \frac{(r_u^{f,b} - r_u)\left(\overline{Y}_u - XVA_u - C_u\right)^-}{B_u^{\tilde{r}}} \,\mathrm{d}u \Big| \mathcal{F}_t\Big].$$

Value adjustments

The following BSDE representation also holds:

$$\begin{cases} -\mathrm{d}XVA_t = f\left(\overline{Y}_t, XVA_t\right)\,\mathrm{d}t - \sum_{k=1}^{d} U_t^k\,\mathrm{d}W_t^{k,\mathbb{Q}}, \\ XVA_T = 0, \end{cases}$$

where

$$\begin{aligned} & f\left(\overline{Y}_t, XVA_t\right) := \\ & -\left(1-R^C\right)\left(\overline{Y}_t - C_u\right)^- \lambda_t^{C,\mathbb{Q}} \\ & +\left(1-R^B\right)\left(\overline{Y}_t - C_u\right)^+ \lambda_t^{B,\mathbb{Q}} \\ & +\left(r_t^{f,l} - r_t\right)\left(\overline{Y}_t - XVA_t - C_t\right)^+ - \left(r_t^{f,b} - r_t\right)\left(\overline{Y}_t - XVA_t - C_t\right)^- \\ & +\left(r_t^{c,l} - r_t\right)C_t^+ - \left(r_t^{c,b} - r_t\right)C_t^- . \end{aligned}$$

XVA computation

(Numerical) solution of BSDEs in possibly high dimension:

- for the exposures Y_t^m, $m = 1, \ldots, M$;
- for the XVA itself.

(See, eg: Cesari et al. ('10), Shöftner ('08) , Pham, Huré, Warin ('19), Abbas-Turki, Crépey, Diallo ('18) et al.)

References: Regression based techniques of "Longstaff-Schwartz" type (coupled with Picard iteration for recursive XVAs), nested MC simulations, PDE techniques (see, eg: Cesari et al. ('10), Shöftner ('08) , Pham, Huré, Warin ('19), Abbas-Turki, Crépey, Diallo ('18) et al.)

Can apply the **deep BSDE solver** by E and Jentzen ('17) (similar to She, Gercu ('17)).

Deep BSDE solver

Algorithm 1: Deep algorithm for exposure simulation

Set parameters: N, L (time steps and Monte Carlo paths)
Fix architecture of ANN (with parameters ρ)
Deep BSDE solver for exposure computation(N, L)
Simulate L paths $(X_n^{(\ell)})_{n=0,\ldots,N}$, $\ell = 1, \ldots, L$.
Define the neural networks $(\varphi_n^\rho)_{n=1,\ldots,N}$;
for $m = 1, \ldots, M$ **do**

minimize over ξ and ρ $\frac{1}{L}\sum_{\ell=1}^{L}\left(g_m(X_N^{(\ell)}) - \mathcal{Y}_N^{m,\rho,\xi,(\ell)}\right)^2$ (recall: $Y_{t_N}^m - g_m(X_{t_N}) = 0$)

subject to $$\begin{cases} \mathcal{Y}_{n+1}^{m,\rho,\xi,(\ell)} = \mathcal{Y}_n^{m,\rho,\xi,(\ell)} + r_n \mathcal{Y}_n^{m,\rho,\xi,(\ell)} \Delta t + (\mathcal{Z}_n^{\rho,(\ell)})^\top \Delta W_n^{(\ell)}, \\ \mathcal{Y}_0^{m,\rho,\xi,(\ell)} = \xi, \\ \mathcal{Z}_n^{\rho,(\ell)} = \varphi_n^\rho(X_n^{(\ell)}). \end{cases}$$

Save the optimizer $(\bar{\xi}^m, \bar{\rho}^m)$.

end
end

Non-recursive adjustments

CVA and DVA can be written as $\mathbb{E}^{\mathbb{Q}}\Big[\int_t^T \Phi(u,\overline{Y}_u)\mathrm{d}u\Big|\mathcal{F}_t\Big].$

Algorithm 2: Deep method non-recursive adjustments

Set parameters: N, L, P (time steps, inner/outer paths);
Fix architecture of ANN.
Apply Algorithm 1.
Simulate $(\mathcal{Y}_n^{m,(p)})_{n=0\ldots N, p=1\ldots P}$, $m = 1,\ldots,M$,
where $\xi = \bar{\xi}^m$, $\rho = \bar{\rho}^m$
Define $\overline{\mathcal{Y}}_n^{(p)} = \sum_{m=1}^M \mathcal{Y}_n^{m,(p)}$, $n = 0,\ldots,N$, $p = 1,\ldots,P$

Compute the adjustment $$\frac{1}{P}\sum_{i=1}^{P}\left(\sum_{n=0}^{N}\eta_n\Phi(\overline{\mathcal{Y}}_n^{(p)})\right)$$

where η_n are weights of the used quadrature formula.

Deep algorithm for XVA computation

Algorithm 3: Deep algorithm for xVA simulation

Set parameters; fix architecture of ANNs.

Apply Algorithm 1

Simulate $(\mathcal{Y}_n^{m,(p)})_{n=0\ldots N, p=1\ldots P}$, $m = 1, \ldots, M$,
where $\xi = \bar{\xi}^m$, $\rho = \bar{\rho}^m$

Define $\overline{\mathcal{Y}}_n^{(p)} = \sum_{m=1}^{M} \mathcal{Y}_n^{m,(p)}$, $n = 0, \ldots, N$, $p = 1, \ldots, P$ **Deep BSDE solver for adjustment computation** (N,P)**:**

Define the neural networks $(\psi_n^\zeta)_{n=1,\ldots,N}$;

minimize over ν **and** ζ, $\quad \frac{1}{P}\sum_{p=1}^{P}\left(\mathcal{X}_N^{\zeta,\nu,(p)}\right)^2 \qquad$ (recall: $XVA_{t_N} = 0$)

subject to
$$\begin{cases} \mathcal{X}_{n+1}^{\zeta,\nu,(p)} = \mathcal{X}_n^{\zeta,\nu,(p)} - f(\overline{\mathcal{Y}}_n^{(p)}, \mathcal{X}_n^{\zeta,(p)})\Delta t + (\mathcal{U}_n^{\zeta,(p)})^\top \Delta W_n^{(p)}, \\ \mathcal{X}_0^{\zeta,\nu,(p)} = \nu, \\ \mathcal{U}_n^{\zeta,(p)} = \psi_n^\zeta. \end{cases}$$

end

Closing words

- Neural networks are effective function approximators in high dimensions.
- Can be used to approximate decision policies, value functions, or their gradients.
- Requires approximation of, and sampling from, underlying dynamics ($\rightarrow$ **Monte Carlo methods**).
- Optimisation over hyper-parameters usually by SGD.
- Requires efficient computation of gradients ($\rightarrow$ **back propagation**).
- Impressive empirical results giving "good" accuracy in high dimensions.