Gradient boosting-based numerical methods for high-dimensional backward stochastic differential equations

Long Teng

MATHEMATICAL MODELLING, ANALYSIS AND COMPUTATIONAL MATHEMATICS

BERGISCHE UNIVERSITÄT WUPPERTAL

Outline

Introduction

Gradient boosting-based approaches for solving BSDEs
- Semidiscrete θ-scheme
- Gradient boosting-based approaches
- Error estimates

Numerical examples

Outline

Introduction

Gradient boosting-based approaches for solving BSDEs

- Semidiscrete θ-scheme
- Gradient boosting-based approaches
- Error estimates

Numerical examples

Introduction to Forward-Backward SDE I

The general form of (decoupled) FBSDE

$$\begin{cases} dX_t = a(t, X_t)\,dt + b(t, X_t)\,dW_t,\ X_0 = x_0, \\ -dY_t = f(t, X_t, Y_t, Z_t)\,dt - Z_t\,dW_t, \\ Y_T = \xi = g(X_T), \end{cases}$$

For $X_t = W_t$:

$$\begin{cases} -dY_t = f(t, Y_t, Z_t)\,dt - Z_t\,dW_t, \\ Y_T = \xi = g(W_T), \end{cases}$$

which is called standard BSDE.

- $f(t, X_t, Y_t, Z_t) : [0, T] \times \Omega \ \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{m \times d} \to \mathbb{R}^m$ is the driver function, ξ is the **square-integrable** terminal condition
- The solution (Y, Z) (a pair of **adapted processes**) exists uniquely provided that a, b, f, g are Lipschitz in all variables
  [E. Pardoux and S. Peng, 1990]
- Well-known result: There are functions $y(t, x)$ and $z(t, x)$ such that $Y_t = y(t, X_t)$ and $Z_t = z(t, X_t)$

Introduction to Forward-Backward SDE II

- Relation to PDE:
  $X_T^{t,x}$: solution of X_T starting from x at time t
  Y_T : terminal value which is equal to $g(X_T^{t,x})$
  The solution $(Y_t^{t,x}, Z_t^{t,x})$ of Forward-Backward-SDEs can be represented as

$$Y_t^{t,x} = u(t,x), \quad Z_t^{t,x} = (\nabla u(t,x))b(t,x) \quad \forall t \in [0,T),$$

  which is the solution of the **semilinear parabolic** PDE of the form

$$\frac{\partial u}{\partial t} + \sum_{i}^{n} a_i \partial_i u + \frac{1}{2} \sum_{i,j}^{n} (bb^T)_{i,j} \partial_{i,j}^2 u + f(t,x,u,(\nabla u)b) = 0$$

  with the terminal condition $u(T,x) = g(x)$

Motivation

Numerical methods for the high dimensional BSDEs:

(1) Fully history recursive multilevel Picard method: *[E et al. 2019, Hutzenthaler et al. 2020]* and the following references therein

(2) Deep-learning based numerical methods

- Deep BSDE algorithm *[E et al. 2017, Han et al.2018]*: global loss function
  Problem: (sometimes) no convergence or stuck in a local minimum
- Backward resolution algorithm *[Huré et al. 2020]*: local loss function
  Problem: not for high-dimensional BSDEs with solution in a complex structure
- Picard algorithm *[Chassagneux et al. 2021]*: a sequence of linear-quadratic optimization problems
  Problem: not for high-dimensional BSDEs with solution in a very complex structure
- (Stochastic) control based algorithms
  [Andersson et al. 2022, Ji et al. 2020a, 2020b, 2021]: relationship between (F)BSDEs and control problems
  Problem: only for the (F)BSDEs stemming from the control problems
- many others...

Two-step procedure of backward schemes:

(1) Time discretisation: *one-step θ-method [Zhao et al. 2012]; multi-step schemes [Chassagneux 2014, Teng et al. 2020, Teng et al. 2021, Zhao et al. 2010, Zhao et al. 2014]*

(2) Approximation of the resulting conditional expectations: *e.g., (Least-squares) Monte Carlo, cubature method, regression tree, Fourier cosine method, spatial approximation,* **Gradient boosting**

Outline

Introduction

Gradient boosting-based approaches for solving BSDEs

- Semidiscrete θ-scheme
- Gradient boosting-based approaches
- Error estimates

Numerical examples

Outline

Reference equation of Y

Consider one-dimension: $m = n = d = 1$ and the **uniform** time partition

$$\Delta_t = \{t_i | t_i \in [0, T], i = 0, 1, \cdots, N_T, t_i < t_{i+1}, t_0 = 0, t_{N_T} = T\},$$

$\Delta t := h = \frac{T}{N_T}, t_i = t_0 + ih, i = 0, 1, \cdots, N_T$
Let (Y_t, Z_t) be the adapted solution

$$Y_t = Y_T + \int_t^T f(s, \mathbb{X}_s)\, ds - \int_t^T Z_s\, dW_s, \quad \mathbb{X}_s = (X_s, Y_s, Z_s),$$

we thus have

$$Y_i = Y_{i+1} + \int_{t_i}^{t_{i+1}} f(s, \mathbb{X}_s)\, ds - \int_{t_i}^{t_{i+1}} Z_s\, dW_s, \quad t \in [0, T)$$

Taking conditional expectation $E_i[\cdot](= E[\cdot|\mathcal{F}_{t_i}])$ yields

$$Y_i = E_i[Y_{i+1}] + \int_{t_i}^{t_{i+1}} E_i[f(s, \mathbb{X}_s)]\, ds.$$

Applying θ-method *[Zhao et al. 2012]* gives

$$Y_i = E_i[Y_{i+1}] + h\theta_1 f(t_i, \mathbb{X}_i) + h(1 - \theta_1) E_i[f(t_{i+1}, \mathbb{X}_{i+1})] + R_\theta^{Y_i}, \quad \theta_1 \in [0, 1]$$

which is **implicit** (Newton's method or Picard iteration)

Reference equation of Z

Recall

$$Y_i = Y_{i+1} + \int_{t_i}^{t_{i+1}} f(s, \mathbb{X}_s)\, ds - \int_{t_i}^{t_{i+1}} Z_s\, dW_s, \quad t \in [0, T),$$

Multiplying $\Delta W_{i+1} := W_{t_{i+1}} - W_{t_i}$ and taking the conditional expectations we obtain

$$-E_i[Y_{i+1}\Delta W_{i+1}] = \int_{t_i}^{t_{i+1}} E_i[f(s, \mathbb{X}_s)\Delta W_s]\, ds - \int_{t_i}^{t_{i+1}} E_i[Z_s]\, ds,$$

Applying the θ-method gives

$$\begin{aligned} -E_i[Y_{i+1}\Delta W_{i+1}] &= h(1-\theta_2)E_i[f(t_{i+1}, \mathbb{X}_{i+1})\Delta W_{i+1}] - h\theta_3 Z_i, \\ &\quad - h(1-\theta_3)E_i[Z_{i+1}] + R_\theta^{Z_i}, \quad \theta_2, \theta_3 \in [0,1] \end{aligned}$$

which is explicit.

The semi-discretisation in time

$(Y_i^{\Delta_t}, Z_i^{\Delta_t})$: the approximation to (Y_i, Z_i)
Given $Y_{N_T}^{\Delta_t}$ and $Z_{N_T}^{\Delta_t}$, then $Y_i^{\Delta_t}$ and $Z_i^{\Delta_t}$ can be computed for $i = N_T - 1, \cdots, 0$

For $i = N_T - 1, \cdots, 0:$

$$
\begin{aligned}
Z_i^{\Delta_t} &= \frac{\theta_3^{-1}}{h} E_i[Y_{i+1}^{\Delta_t} \Delta W_{i+1}] + \theta_3^{-1}(1-\theta_2) E_i[f(t_{i+1}, \mathbb{X}_{i+1}^{\Delta_t}) \Delta W_{i+1}] \\
&\quad - \theta_3^{-1}(1-\theta_3) E_i[Z_{i+1}^{\Delta_t}], \\
Y_i^{\Delta_t} &= E_i[Y_{i+1}^{\Delta_t}] + h\theta_1 f(t_i, \mathbb{X}_i^{\Delta_t}) + h(1-\theta_1) E_i[f(t_{i+1}, \mathbb{X}_{i+1}^{\Delta_t})].
\end{aligned}
$$

Different schemes by choosing different values for $\theta_k,\ k = 1, 2, 3,$

- $\theta_1 = \theta_2 = \theta_3 = \frac{1}{2}$: second-order provided that g is continuously differentiable
- $\theta_1 = \theta_2 = \theta_3 = 1$: first-order and Z_{N_T} is not needed

  [Li et al., 2017, Zhao et al. 2012, Zhao et al. 2013]

Outline

Introduction

Gradient boosting-based approaches for solving BSDEs

- Semidiscrete θ-scheme
- Gradient boosting-based approaches
- Error estimates

Numerical examples

Non-parametric regression I

Consider non-parametric regression model

$$Y = \eta(X) + \epsilon,$$

where ϵ has a zero expectation and a constant variance. It is well-known that

$$E[Y|X = x] = \eta(x).$$

The estimator, $\hat{\eta}(x)$ is represented by an XGBRegressor model.

Given a dataset (samples), $(\hat{x}_{\mathcal{M}}, \hat{y}_{\mathcal{M}})$, $\mathcal{M} = 1, \cdots, M$, an XGBRegressor model can be **fitted on the data and reused** to determine (predict) $E[Y|X = x]$ for an arbitrary x.

Example: $\theta_2 = 1, \theta_3 = 1$
Consider

$$Z_i^{\Delta_t} = E\left[\frac{1}{h} Y_{i+1}^{\Delta_t} \Delta W_{i+1} | X_i^{\Delta_t}\right], \quad i = N_T - 1, \cdots, 0.$$

Non-parametric regression II

Aim is to find the deterministic functions $z_i^{\Delta t}(x)$ represented by XGBRegressors such that

$$Z_i^{\Delta t} = z_i^{\Delta t}(X_i^{\Delta t}) \approx \hat{R}_i^z(X_i^{\Delta t})$$

Starting from T with samples $(\hat{x}_{N_T-1,\mathcal{M}}, \frac{1}{h}\hat{y}_{N_T,\mathcal{M}}\Delta\hat{w}_{N_T,\mathcal{M}})$, the XGBRegressor $\hat{R}^z_{N_T-1}$ is fitted for $Z^{\Delta t}_{N_T-1}$, i.e., the function

$$z^{\Delta t}_{N_T-1}(x) = E\left[\frac{1}{h}Y^{\Delta t}_{N_T}\Delta W_{N_T}|X^{\Delta t}_{N_T-1} = x\right],$$

is **estimated and presented** by the XGBRegressor $\hat{R}^z_{N_T-1}$.

- The dataset $\hat{z}_{N_T-1,\mathcal{M}}$, $\mathcal{M} = 1, \cdots, M$ of $Z^{\Delta t}_{N_T-1}$ can be predicted via the XGBRegressor $\hat{R}^z_{N_T-1}$ with the dataset $\hat{x}_{N_T-1,\mathcal{M}}$
- In the same manner, the dataset $\hat{z}_{N_T-1,\mathcal{M}}$ are used to construct the XGBRegressor $\hat{R}^z_{N_T-2}$ and generate the dataset $\hat{z}_{N_T-2,\mathcal{M}}$ of $Z^{\Delta t}_{N_T-2}$ at the time t_{N_T-2} and so on
- At $t = 0$, the initial value x_0 is known. Based on the XGBRegressor $\hat{R}^z_0$ we obtain the solution $Z_0^{\Delta t} = z_0^{\Delta t}(x_0)$

XGBoost regression I

Regularized learning objective

Consider $d = 1$ and omit the index of the time step, e.g., $\hat{x}_{\mathcal{M}} := \hat{x}_{i,\mathcal{M}}$.

Using a given dataset with M samples

$$\mathcal{D} = \left\{ (\hat{x}_{\mathcal{M}}, \hat{\mathcal{Z}}_{\mathcal{M}}) | \, |\mathcal{D}| = M, \, \hat{x}_{\mathcal{M}}, \, \hat{\mathcal{Z}}_{\mathcal{M}} \in \mathbb{R} \right\}$$

a tree ensemble model consists of K regression trees can be constructed to predict the output

$$\hat{z}_{\mathcal{M}} = \hat{\eta}(\hat{x}_{\mathcal{M}}) = \sum_{k=1}^{K} \tilde{f}_k(\hat{x}_{\mathcal{M}}), \quad \tilde{f}_k \in \mathcal{S},$$

where $\mathcal{S} = \{\tilde{f}(x) = \omega_{q(x)}, \, q : \mathbb{R} \to \hat{T}, \, \omega \in \mathbb{R}^{\hat{T}}\}$ is the space of regression trees.
q : the tree structure that maps an example to the corresponding leaf index
ω_j : score on the j-th leaf, $\hat{T}$: the number of leaves

Train the model by optimizing the mean squared error (MSE)

$$L(\hat{z}_{\mathcal{M}}, \hat{\mathcal{Z}}_{\mathcal{M}}) = \frac{1}{M} \sum_{\mathcal{M}=1}^{M} (\hat{z}_{\mathcal{M}} - \hat{\mathcal{Z}}_{\mathcal{M}})^2.$$

with the regularization term *[Chen and Guestrin 2016]*

$$\Omega(\tilde{f}) = \gamma \hat{T} + \frac{1}{2} \lambda \, \|w\|^2 = \gamma \hat{T} + \frac{1}{2} \lambda \sum_{j=1}^{\hat{T}} w_j^2$$

which controls the model complexity, γ, λ are positive regularization parameters

XGBoost regression II

The regularized objective (loss function) is thus given by

$$\mathcal{L}(\eta) = \sum_{\mathcal{M}=1}^{M} L(\hat{z}_{\mathcal{M}}, \hat{\mathcal{Z}}_{\mathcal{M}}) + \sum_{k=1}^{K} \Omega(\tilde{f}_k). \tag{1}$$

Gradient Tree Boosting *[Chen and Guestrin 2016]*
The gradient descent is used to minimize the loss function (1) iteratively by greedily adding $\tilde{f}_k$

$$\begin{aligned}\mathcal{L}^{(k)} &= \sum_{\mathcal{M}=1}^{M} L(\hat{\mathcal{Z}}_{\mathcal{M}}, \hat{z}_{\mathcal{M}}^{k}) + \sum_{j=1}^{k} \Omega(\tilde{f}_j) \\ &= \sum_{\mathcal{M}=1}^{M} L(\hat{\mathcal{Z}}_{\mathcal{M}}, \hat{z}_{\mathcal{M}}^{(k-1)} + \tilde{f}_k(\hat{x}_{\mathcal{Z}})) + \sum_{j=1}^{k} \Omega(\tilde{f}_j),\end{aligned}$$

where $\hat{z}_{\mathcal{M}}^{k} = \sum_{j=1}^{k} \tilde{f}_j(\mathbf{x}_{\mathcal{M}})$, and $k = 1, \cdots, K$.
Taking the second-order approximation one obtains

$$\mathcal{L}^{(k)} \approx \sum_{\mathcal{M}=1}^{M} \left(L(\hat{\mathcal{Z}}_{\mathcal{M}}, \hat{z}_{\mathcal{M}}^{(k-1)}) + g_{\mathcal{M}} \tilde{f}_k(\hat{x}_{\mathcal{M}}) + \frac{1}{2} h_{\mathcal{M}} \tilde{f}_k^2(\hat{x}_{\mathcal{M}}) \right) + \sum_{j=1}^{k} \Omega(\tilde{f}_j),$$

where $g_{\mathcal{M}} = \partial_{\hat{z}^{(k-1)}} L(\hat{\mathcal{Z}}_{\mathcal{M}}, \hat{z}^{(k-1)})$ and $h_{\mathcal{M}} = \partial^2_{\hat{z}^{(k-1)}} L(\hat{\mathcal{Z}}_{\mathcal{M}}, \hat{z}^{(k-1)})$ are first and second order gradients, respectively.

Tree building algorithm I

By removing the constant terms one obtains the objective at k-th step

$$\tilde{\mathcal{L}}^{(k)} = \sum_{\mathcal{M}=1}^{M} \left(g_{\mathcal{M}} \tilde{f}_k(\hat{x}_{\mathcal{M}}) + \frac{1}{2} h_{\mathcal{M}} \tilde{f}_k^2(\hat{x}_{\mathcal{M}}) \right) + \Omega(\tilde{f}_k),$$

which needs to be optimized by finding a $\tilde{f}_k$. Define the index set

$$I_j = \{\mathcal{M} | q(\hat{x}_{\mathcal{M}}) = j\}$$

which contains the indices of data points mapped to the j-th leaf. Further reformulation reads

$$\begin{aligned}\tilde{\mathcal{L}}^{(k)} &= \sum_{\mathcal{M}=1}^{M} \left(g_{\mathcal{M}} \tilde{f}_k(\hat{x}_{\mathcal{M}}) + \frac{1}{2} h_{\mathcal{M}} \tilde{f}_k^2(\hat{x}_{\mathcal{M}}) \right) + \gamma \hat{T} + \frac{1}{2} \lambda \sum_{j=1}^{\hat{T}} w_j^2 \\ &= \sum_{j=1}^{\hat{T}} \left(\left(\sum_{\mathcal{M} \in I_j} g_{\mathcal{M}} \right) w_j + \frac{1}{2} \left(\sum_{\mathcal{M} \in I_j} h_{\mathcal{M}} + \lambda \right) w_j^2 \right) + \gamma \hat{T}.\end{aligned}$$

For a fixed tree structure $q(\hat{x})$, one can easily compute the optimal w_j of leaf j as

$$w_j^* = -\frac{\sum_{\mathcal{M} \in I_j} g_{\mathcal{M}}}{\sum_{\mathcal{M} \in I_j} h_{\mathcal{M}} + \lambda}.$$

Tree building algorithm II

The optimal value of the objective reads thus

$$\tilde{\mathcal{L}}^{(k)}(q) = -\frac{1}{2}\sum_{j=1}^{\hat{T}} \frac{\left(\sum_{\mathcal{M}\in I_j} g_{\mathcal{M}}\right)^2}{\sum_{\mathcal{M}\in I_j} h_{\mathcal{M}} + \lambda} + \gamma\hat{T},$$

which can be used as a scoring function to measure the quality of q.
A greedy algorithm

- Start with the root (depth 0)
- Add a split for each leaf node, the change of objective reads

$$\mathcal{L}_{\text{gain}} = \frac{1}{2}\left(\frac{\left(\sum_{\mathcal{M}\in I_L} g_{\mathcal{M}}\right)^2}{\sum_{\mathcal{M}\in I_L} h_{\mathcal{M}} + \lambda} + \frac{\left(\sum_{\mathcal{M}\in I_R} g_{\mathcal{M}}\right)^2}{\sum_{\mathcal{M}\in I_R} h_{\mathcal{M}} + \lambda} - \frac{\left(\sum_{\mathcal{M}\in I} g_{\mathcal{M}}\right)^2}{\sum_{\mathcal{M}\in I} h_{\mathcal{M}} + \lambda}\right) - \gamma$$

  where $I = I_L \cup I_R$, I_L and I_R denote index sets of left and right nodes after splitting, respectively.
- Comparisons over index set $\Longrightarrow$ The best split along the feature

Trade-off between simplicity and predictiveness

- The best split have negative gain $\Longrightarrow$ stop
- Grow a tree to maximum depth, and prune all the splits with negative gain

Outline

Error estimates I

XGBoost regression error: R_{xgb} Error of iterative method: R_{impl}

$$
\begin{aligned}
&\hat{y}_{N_T,\mathcal{M}} = g(\hat{x}_{N_T,\mathcal{M}}),\ \hat{z}_{N_T,\mathcal{M}} = g_x(\hat{x}_{N_T,\mathcal{M}}),\\
&\quad \textbf{For } i = N_T - 1, \cdots, 0,\ \mathcal{M} = 1, \cdots, M:\\
&z_i^{\Delta t}(\hat{x}_{i,\mathcal{M}}) = E_i^{\hat{x}_{i,\mathcal{M}}}\left[\frac{\theta_3^{-1}}{h} Y_{i+1}\Delta W_{i+1} + \theta_3^{-1}(1-\theta_2) f(t_{i+1}, \mathbb{X}_{i+1})\Delta W_{i+1} - \theta_3^{-1}(1-\theta_3) Z_{i+1}\right] + R_{\text{xgb}}^{Z_i}\\
&y_i^{\Delta t}(\hat{x}_{i,\mathcal{M}}) = E_i^{\hat{x}_{i,\mathcal{M}}}\left[Y_{i+1} + h(1-\theta_1) f(t_{i+1}, \mathbb{X}_{i+1})\right] + h\theta_1 \hat{f}_{i,\mathcal{M}}(t_i, \mathbb{X}_i) + R_{\text{impl}}^{Y_i} + R_{\text{xgb}}^{Y_i},
\end{aligned}
$$

where $E_i^{\hat{x}_{i,\mathcal{M}}}[\mathcal{Y}]$ denotes calculated conditional expectation $E[\mathcal{Y}|X = \hat{x}_{i,\mathcal{M}}]$ using the constructed XGBRegressor models with the samples of $\mathcal{Y}$.
It can be shown that

$$R_{\text{xgb}}^{Z_i} \leq 2(Var_i^z + \hat{\mathcal{L}}_{\min}(\hat{q}^{z_i})) \text{ and } R_{\text{xgb}}^{Y_i} \leq 2(Var_i^y + \hat{\mathcal{L}}_{\min}(\hat{q}^{y_i})),$$

where Var_i^z and Var_i^y are the constant variances of ϵ_i^z and ϵ_i^y, respectively.

Time complexity analysis It can be shown that

$$\mathcal{O}(K\tilde{d}MdN_T + MdN_T \log B)$$

where $\tilde{d}$: maximum depth, B : maximum number of rows in each block (memory units in which data stored)

Error estimates II

Time-discretization errors:

$$\begin{aligned}
\epsilon^{Y_i,\theta} &:= Y_i - Y_i^{\Delta_t},\\
\epsilon^{Z_i,\theta} &:= Z_i - Z_i^{\Delta_t},\\
\epsilon^{f_i,\theta} &:= f(t_i, \mathbb{X}_i) - f(t_i, \mathbb{X}_i^{\Delta_t}).
\end{aligned}$$

The deterministic functions $Z_i^{\Delta_t} = z_i^{\Delta_t}(X_i^{\Delta_t})$ and $Y_i^{\Delta_t} = y_i^{\Delta_t}(X_i^{\Delta_t})$ are approximated by the XGBRegressors, resulting in the approximations $\hat{y}_i^{\Delta_t}, \hat{z}_i^{\Delta_t}$ with

$$\hat{Y}_i^{\Delta_t} = \hat{y}_i^{\Delta_t}(X_i^{\Delta_t}) \text{ and } \hat{Z}_i^{\Delta_t} = \hat{z}_i^{\Delta_t}(X_i^{\Delta_t}),$$

respectively.

Thus, global errors read:

$$\begin{aligned}
\epsilon^{Y_i} &:= Y_i - \hat{Y}_i^{\Delta_t},\\
\epsilon^{Z_i} &:= Z_i - \hat{Z}_i^{\Delta_t},\\
\epsilon^{f_i} &:= f(t_i, \mathbb{X}_i) - f(t_i, \hat{\mathbb{X}}_i^{\Delta_t}).
\end{aligned}$$

Error estimates III

Assumption 1: some non-degeneracy conditions on a and b

- The local truncation errors $R_{\theta}^{Y_i}$ and $R_{\theta}^{Z_i}$ are bounded by $C(\Delta t_i)^3$ when $\theta_i = \frac{1}{2}$, $i = 1, 2, 3$, and otherwise by $C(\Delta t_i)^2$ *[Li et al., 2017, Zhao et al. 2012, Zhao et al. 2013]*
- Assume that $X_i = X_i^{\Delta t}$
- Picard iterations which converges for any initial guess when Δt_i is small enough provided that the Lipschitz assumptions on the driver

Theorem

Under Assumption 1, if $f \in C_b^{2,4,4,4}$, $g \in C_b^{4+\alpha}$ for some $\alpha \in (0,1)$, a and b are bounded, $a, b \in C_b^{2,4}$, and given

$$E_{N_T-1}^{x_{N_T-1}}[|\epsilon^{Z_{N_T}}|^2] \sim \mathcal{O}((\Delta t)^2), \quad E_{N_T-1}^{x_{N_T-1}}[|\epsilon^{Y_{N_T}}|^2] \sim \mathcal{O}((\Delta t)^2),$$

It holds then

$$E_0^{x_0}\left[|\epsilon^{Y_i}|^2 + \frac{(8\theta_3^2(\theta_2-1)^2 + (1-\theta_3)^2\theta_2^2)\Delta t}{2(1-\theta_3)^2 + 2\theta_3^2}|\epsilon^{Z_i}|^2\right] \le Q(\Delta t)^2 + \tilde{Q}\sum_{i+1}^{N_T}\left(\frac{N_T(\mathit{Var}_j^{\mathcal{Y}})^2}{T} + \frac{T(\mathit{Var}_j^{\mathcal{Z}})^2}{N_T}\right),$$

$0 \le i \le N_T - 1$, *where Q is a constant which only depend on T, x_0 and the bounds of f, g and a, b, $\tilde{Q}$ is a constant depending on T, x_0 and L, and $\mathit{Var}_i^{\mathcal{Y}}$ and $\mathit{Var}_i^{\mathcal{Z}}$ are the bounded constants, and M is the number of samples. [Teng 2022]*

Outline

Introduction

Gradient boosting-based approaches for solving BSDEs

- Semidiscrete θ-scheme
- Gradient boosting-based approaches
- Error estimates

Numerical examples

Option pricing with different interest rate (100d) I

$$dS_{t,d} = \mu S_{t,d}\,dt + \sigma S_{t,d} dW_{t,d}, \quad d = 1, \cdots D,$$

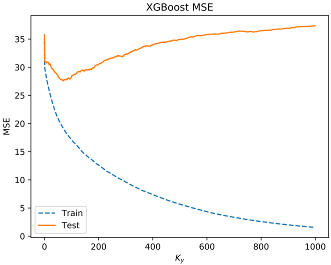
where $\sigma > 0$ and $\mu \in \mathbb{R}$, $W_{t,d}$ are independent.

$$\begin{cases} -dY_t = f(t, X_t, Y_t, Z_t)\,dt - \mathbf{Z}_t\,d\mathbf{W}_t \\ Y_T = \max\left(\max_{d=1,\cdots,D}(S_{T,d}) - K_1, 0\right) - 2\max\left(\max_{d=1,\cdots,D}(S_{T,d}) - K_2, 0\right) \end{cases}$$

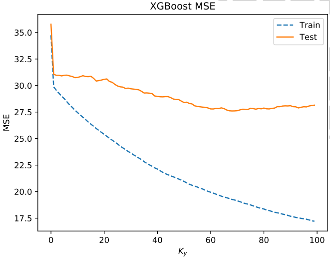
with

$$f(t, x, y, z) = -R^l y - \frac{\mu - R^l}{\sigma}\sum_{d=1}^{D} z_d + (R^b - R^l)\max\left(0, \frac{1}{\sigma}\sum_{d=1}^{D} z_d - y\right)$$

$T = 0.5$, $S_0 = 100$, $\mu = 0.6$, $\sigma = 0.02$, $R^l = 0.04$, $R^b = 0.06$, $K_1 = 120$, $K_2 = 150$, the reference price $Y_0 = 21.2988$ *[E et al. 2019]*



The XGBoost model for Y until $K_y = 1000$.



The enlargement of the learning curves on the left hand side until $K_y = 100$.

The MSEs of the XGBoost models for different numbers of trees, $N_T = 10$, $M = 10000$ and the learning rate is 0.9

Option pricing with different interest rate (100d) II

The error $error_y := \frac{1}{10}\sum_{k=1}^{10}|Y_0 - Y_{0,k}^{\Delta t}|$

The standard deviation $\sqrt{\frac{1}{9}\sum_{k=1}^{10}|Y_{0,k}^{\Delta t} - Y_0^{\Delta t}|^2}$ with $Y_0^{\Delta t} = \frac{1}{10}\sum_{k=1}^{10} Y_{0,k}^{\Delta t}$

$d = 100$ $T = 0.5$ / N_T	Ref. price *[E et al. 2019]* Y_0	$M = 10000$ $error_y$ (Std. dev.) avg. runtime	$M = 20000$ $error_y$ (Std. dev.) avg. runtime	$M = 50000$ $error_y$ (Std. dev.) avg. runtime	$M = 100000$ $error_y$ (Std. dev.) avg. runtime
10	21.2988	0.13725(0.13335) 9.10	0.13911(0.09088) 19.37	0.12952(0.06001) 54.06	0.18669(0.04351) 131.17
20	21.2988	0.20207(0.21176) 25.03	0.14609(0.16716) 51.73	0.05542(0.03960) 139.14	0.08281(0.01219) 324.49
30	21.2988	0.33619(0.43693) 40.94	0.14689(0.15127) 84.17	0.04741(0.05735) 224.13	0.04096(0.05090) 519.47

- The relative error of 0.0039 in 566 seconds with the Deep BSDE in *[E et al. 2017]*
- The relative errors 0.00222 and 0.000192 can be achieved in runtime 224.13 and 519.47, respectively

The Allen-Cahn equation

$$\begin{cases} dX_t &= \sqrt{2}\, dW_t, \\ -dY_t &= \left(Y_t - Y_t^3\right) dt - Z_t\, dW_t, \\ Y_T &= \arctan\left(\max_{\hat{d}\in\{1,2,\cdots,d\}} X_T^{\hat{d}}\right). \end{cases}$$

[Beck et al. 2021]

$T = 0.3,\ N_T = 10$ d	Ref. value *[E et al. 2019]*	$M = 2000$ $error_y$(Std. dev.) avg. runtime	$M = 5000$ $error_y$(Std. dev.) avg. runtime
10	0.89060	0.00279(0.00342) 0.19	0.00175(0.00233) 0.33
50	1.01830	0.00141(0.00187) 0.45	0.00076(0.00076) 1.18
100	1.04510	0.00265(0.00147) 0.85	0.00098(0.00113) 2.21
200	1.06220	0.00101(0.00130) 1.69	0.00074(0.00097) 4.31
300	1.07217	0.00247(0.00171) 2.53	0.00075(0.00044) 6.74
500	1.08124	0.00134(0.00110) 4.37	0.00071(0.00034) 11.79
1000	1.09100	0.00111(0.00142) 9.25	0.00051(0.00103) 25.33
5000	1.10691	0.00162(0.00086) 69.51	0.00174(0.00012) 129.90
10000	1.11402	0.00049(0.00087) 151.89	0.00037(0.00017) 670.24

The Burgers-type equation

$$\begin{cases} dX_t &= \frac{d}{\sqrt{2}}\, dW_t, \\ -dY_t &= \left(Y_t - \frac{2+d}{2d}\right)\left(\sum_{\hat{d}=1}^{d} Z_t^{\hat{d}}\right) dt - Z_t\, dW_t, \end{cases}$$

with the analytical solution

$$\begin{cases} Y_t &= \dfrac{\exp\left(t+\frac{1}{d}\sum_{\hat{d}=1}^{d} X_t^{\hat{d}}\right)}{1+\exp\left(t+\frac{1}{d}\sum_{\hat{d}=1}^{d} X_t^{\hat{d}}\right)}, \\ Z_t &= \dfrac{\sigma}{d}\dfrac{\exp\left(t+\frac{1}{d}\sum_{\hat{d}=1}^{d} X_t^{\hat{d}}\right)}{\left(1+\exp\left(t+\frac{1}{d}\sum_{\hat{d}=1}^{d} X_t^{\hat{d}}\right)\right)^2}\mathbf{1}_d. \end{cases}$$

$d = 100$ $T = 0.5$ N_T	Theoretical solution Y_0 $\mathbf{Z}_0$	$M = 10000$ $error_y$(Std. dev.) $error_z$(Std. dev.) avg. runtime	$M = 20000$ $error_y$(Std. dev.) $error_z$(Std. dev.) avg. runtime	$M = 50000$ $error_y$(Std. dev.) $error_z$(Std. dev.) avg. runtime	$M = 100000$ $error_y$(Std. dev.) $error_z$(Std. dev.) avg. runtime
10	0.5 $0.17678\,\mathbf{1}_d$	0.05486(0.03434) 0.00601(0.00412) 10.66	0.05570(0.02464) 0.00529(0.00505) 22.23	0.05545(0.01359) 0.00460(0.00244) 59.58	0.05203(0.01430) 0.00454(0.00135) 152.96
20	0.5 $0.17678\,\mathbf{1}_d$	0.01625(0.00038) 0.00641(0.00881) 27.11	0.01629(0.00019) 0.00560(0.00629) 55.67	0.01650(0.00010) 0.00454(0.00381) 146.40	0.01640(0.00009) 0.00387(0.00235) 369.11
30	0.5 $0.17678\,\mathbf{1}_d$	0.00712(0.00010) 0.00785(0.00494) 43.55	0.00712(0.00005) 0.00526(0.00509) 88.39	0.00714(0.00005) 0.00519(0.00259) 234.21	0.00713(0.00003) 0.00424(0.00289) 583.45

- $d = 20$ is considered in *[E et al. 2017]* and approximations of Z are not given

A challenging problem I

$$\begin{cases} dX_t &= \frac{1}{\sqrt{d}} \mathsf{I}_d \, dW_t, \\ -dY_t &= \left(1 + \frac{T-t}{2d}\right) A(X_t) + B(X_t) + C \cos\left(\sum_{\hat{d}=1}^{d} \hat{d}\, Z^{\hat{d}}\right) dt - Z_t \, dW_t, \end{cases}$$ *[Chassagneux et al. 2021]*

with

$$A(x) = \frac{1}{d} \sum_{\hat{d}=1}^{d} \sin(x^{\hat{d}}) \mathbb{1}_{\{x^{\hat{d}}<0\}}, \quad B(x) = \frac{1}{d} \sum_{\hat{d}=1}^{d} x^{\hat{d}} \mathbb{1}_{\{x^{\hat{d}}\geq 0\}}, \quad C = \frac{(d+1)(2d+1)}{12},$$

and the analytic solution

$$Y_t = \frac{T-t}{d} \sum_{\hat{d}=1}^{d} \left(\sin(X_t^{\hat{d}}) \mathbb{1}_{\{X_t^{\hat{d}}<0\}} + X_t^{\hat{d}} \mathbb{1}_{\{X_t^{\hat{d}}\geq 0\}} \right) + \cos\left(\sum_{\hat{d}=1}^{d} \hat{d}\, Z_{\hat{d}} \right).$$

- *[E et al. 2017]* fails when $d \geq 3$
- *[Huré et al. 2020]* and *[Chassagneux et al. 2021]* fail when $d \geq 8$

A challenging problem II

$T = 1$	$M = 10000$ $error_y$(Std. dev.) avg. runtime	$M = 50000$ $error_y$(Std. dev.) avg. runtime	$M = 100000$ $error_y$(Std. dev.) avg. runtime	$M = 200000$ $error_y$(Std. dev.) avg. runtime
N_T	$d = 1, Y_0 = 1.3776, K_z = 10, K_y = 100$			
10	0.00371(0.00501) 0.72	0.00153(0.00188) 2.94	0.00096(0.00112) 5.80	0.00118(0.00169) 11.62
20	0.00565(0.00649) 1.51	0.00127(0.00121) 6.25	0.00173(0.00220) 12.36	0.00087(0.00128) 24.77
30	0.00526(0.00598) 2.31	0.00112(0.00170) 9.54	0.00159(0.00191) 18.86	0.00134(0.00141) 37.98
N_T	$d = 2, Y_0 = 0.5707, K_z = 8, K_y = 150$			
10	0.00893(0.01154) 1.10	0.00505(0.00622) 4.69	0.00278(0.00359) 9.21	0.00258(0.00337) 18.69
20	0.01156(0.01370) 2.31	0.00376(0.00437) 10.01	0.00317(0.00386) 19.81	0.00327(0.00340) 40.05
30	0.01167(0.01772) 3.52	0.00558(0.00607) 15.32	0.00325(0.00425) 30.34	0.00177(0.00252) 61.56
N_T	$d = 5, Y_0 = 0.8466, K_z = 2, K_y = 150$			
10	0.02626(0.03105) 1.68	0.01533(0.01038) 7.91	0.01191(0.00681) 16.02	0.00917(0.00545) 32.79
20	0.01854(0.02541) 3.58	0.01101(0.01310) 17.27	0.00537(0.00761) 34.72	0.00398(0.00489) 70.96
30	0.02439(0.03115) 5.48	0.00687(0.00947) 26.49	0.00718(0.01015) 53.30	0.00452(0.00437) 108.78

A challenging problem III

$T = 1$ $M = 20000$ d N_T	Theoretical solution Y_0	Numerical approximation $Y_0^{\Delta t}$	$error_y$(Std. dev.)	$K_z = K_y$	avg. runtime
8 20	1.16032	1.16830	$0.01047(0.00931)$	12	5.47
10 20	-0.21489	-0.21517	$0.02435(0.03030)$	40	14.19
20 30	0.25904	0.2555	$0.02838(0.03492)$	16	32.55
50 400	-0.47055	-0.47437	$0.00667(0.00778)$	10	1805.75

Thank you for your attention!

References

K. Andersson, A. Andersson and C. W. Oosterlee, *Convergence of a robust deep FBSDE method for stochastic control*, arXiv.2201.06854, 2022

C. Beck, S. Becker, P. Cheridito, A. Jentzen and A. Neufeld, *Deep splitting method for parabolic PDEs*, SIAM J. Sci. Comput. 43(5): A3135–A3154, 2021

J. F. Chassagneux, *Linear multistep schemes for BSDEs*, SIAM J. Numer. Anal. 52: 2815–2836, 2014

J. F. Chassagneux, J. Chen, N. Frikha and C. Zhou *A learning scheme by sparse grids and Picard approximations for semilinear parabolic PDEs*, arXiv.2102.12051, 2021

T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system*, KDD 16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 785–794, 2016

W. E., J. Han and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Commun. Math. Stat. 5: 349–380, 2017

W. E., M. Hutzenthaler, A. Jentzen and T. Kruse, *On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations*, J. Sci. Comput. 19: 1534–1571, 2019

J. Han, A. Jentzen and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences 115 (34): 8505–8510, 2018

C. Huré, H. Pham and X. Warin, *Deep backward schemes for high-dimensional nonlinear PDEs*, Math. Comput. 89: 1547–1579, 2020

M. Hutzenthaler, A. Jentzen, T. Kruse, T. Nguyen and P. von Wurstemberger, *Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations*, Proceedings of the Royal Society A 476 (2244), 2020.

S. J, S. Peng, Y. Peng and X. Zhang, *Three algorithms for solving high-dimensional fully-coupled FBSDEs through deep learning*, IEEE Intelligent Systems 35(3): 71–84, 2020

S. J, S. Peng, Y. Peng and X. Zhang, *Solving stochastic optimal control problem via stochastic maximum principle with deep learning method*, arXiv.2007.02227, 2020

References

S. J, S. Peng, Y. Peng and X. Zhang, *A novel control method for solving high-dimensional Hamiltonian systems through deep neural networks*, arXiv.2111.02636, 2021

N. EL Karoui, S. Peng and M. C. Quenez, *Backward stochastic differential equations in finance*, Math. Finance, 7(1): 1–71, 1997

Y. Li, J. Yang and W. Zhao, *Convergence error estimates of the crank-Nicolson scheme for solving decoupled FBSDEs*, Sci. China. Math., 60(5): 923–948, 2017

E. Pardoux and S. Peng, *Adapted solution of a backward stochastic differential equations*, System and Control Letters 14: 55–61, 1990

L. Teng, *Gradient boosting-based numerical methods for high-dimensional backward stochastic differential equations*, Appl. Math. Comput. 426: 127119, 2022

L. Teng, *A review of tree-based approaches to solve forward-backward stochastic differential equation*, J. Comput. Finance, 25(3): 125–159, 2021

L. Teng, A. Lapitckii and M. Günther *A Multi-step Scheme based on Cubic Spline for solving Backward Stochastic Differential Equations*, Appl. Numer. Math. 150: 117-138, 2020

L. Teng and W. Zhao *High-order combined multi-step scheme for solving forward backward stochastic differential equations*, J. Sci. Comput. 87(81), 2021

W. Zhao, Y. Li and L. Ju *Error estimates of the crank-nicolson scheme for solving backward stochastic differential equations*, Int. J. Numer. Anal. Model. 10(4), pp.876–898, 2013

W, Zhao, Y. Li and G. Zhang, *A generalized θ-scheme for solving decoupled FBSDEs*, Discrete Cont. Dyn-B. 17(5), pp.1585–1603, 2012

W. Zhao, G. Zhang and L. Ju, *A stable multistep scheme for solving backward stochastic differential equations*, SIAM J. Numer. Anal. 48(4), pp.1369–1394, 2010

W. Zhao, Y. Fu and T. Zhou, *New kinds of high-order multistep schemes for coupled forward backward stochastic differential equations*, SIAM J. Sci. Comput. 36, pp.A1731–A1751, 2014