SCDAA Coursework 2020-21*

David Šiška, School of Mathematics, University of Edinburgh

Submit via Learn by Friday 9th April 2021 at 12:01pm

Overview

Your overall task will be to implement a numerical algorithm for solving stochastic control problems using the Method of Successive Approximation (MSA).

This will be broken into steps which a) give you credit for completion and b) allow you to check your progress.

1. Create a Python method for estimating conditional expectations from data, see Section 1.

2. Use this method for estimating conditional expectations to solve a linear BSDE arising in Black–Scholes pricing and hedging and check that you are getting the correct answers, see Section 2.

3. Implement the MSA algorithm for deterministic linear quadratic control problem, see Section 3.

4. Implement the MSA algorithm for stochastic linear quadratic control problem, see Section 4.

You will be expected to submit a Jupyter notebook which a) clearly describes what you're doing[1] and why and b) can be run from beginning to the end in under 3 minutes producing output (text / plots) demonstrating sucessful completion of each of the tasks. Use `numpy`, `scipy` and `matplotlib` and all the functions they contain but do not use other libraries.[2]

You will be working in groups of two[3] and the first cell in the Jupyter notebook should state your name, student number and individual contribution percentage.[4] You cannot split the tasks (saying one does 1. and 2. and the other does 3. and 4.) because they depend sequentially on succesful completion of the previous ones. You need to work together on each of the tasks.

There are 22 marks in total on this assignment but it is *marked out of* 20. Think of the extra two marks as a bonus. The Exercise 4.2 may be hard but you can still

*Last updated 5th April 2021.

[1]It is possible to write LaTeX in formulae in the Markdown fields.

[2]If you wish to use another library seek permission from the course organiser first.

[3]Self-organise. One person from each group should send the course organiser an email (Cc'ing the other member) with the subject `SCDAA CW group` stating names and student numbers of group members.

[4]The default is that you contributed equally i.e. 50% and 50%. If one of you feels they contributed more then they have to agree with the other about how you split this (e.g. 65% and 35%). If you cannot agree speak with the course organiser.

score 80% without tackling it. **Please note that a submission that doesn't demonstrate why things work (e.g. test examples, convergence plots etc.) will get very few marks even if you have code that correctly solves the exercises.**

We will assume throughout that we have a suitable probability space $(\Omega, \mathcal{F}, \mathbb{P})$ on which our random variables exist.

1 Empirical estimation of conditional expectation

You may wish to review what you know about conditional expectations e.g. in [4, Appendix A.3]. We will use the notation from there. Let X, Y be random variables in $L^2(\mathcal{F})$. We know that

$$\|Y - \mathbb{E}[Y|X]\|_2^2 = \inf_{Z \in L^2(\sigma(X))} \|Y - Z\|_2^2 .$$

So among all the square integrable random variables measurable w.r.t. $\sigma(X)$ the conditional expectation $\mathbb{E}[Y|X]$ is the one which minimizes the L^2 norm.

Moreover the Doob–Dynkin Lemma tells us that there is a measurable function h s.t. for every $Z \in L^2(\sigma(X))$ we have $Z = h(X)$. So we can equivalently say that the conditional expectation must satisfy

$$\|Y - \mathbb{E}[Y|X]\|_2^2 = \inf_{h \text{ meas. and } h(X) \in L^2(\sigma(X))} \|Y - h(X)\|_2^2 . \tag{1}$$

Let $\psi_1, \dots, \psi_M$ be fixed "basis functions" and let $\beta_1, \dots, \beta_M \in \mathbb{R}$. Consider functions of the form

$$h(x) = \sum_{r=1}^{M} \beta_r \psi_r(x) .$$

If, instead of minimizing over all measurable functions like in (1), we would do the minimization over $(\beta_r)_{r=1}^M$ then we could get an approximation of the conditional expectation. The quality of this approximation will depend on our basis functions and on how many we take.[5] This is

$$\|Y - \mathbb{E}[Y|X]\|_2^2 \approx \inf_{\beta \in \mathbb{R}^M} \left\| Y - \sum_{r=1}^{M} \beta_r \psi_r(X) \right\|_2^2 \tag{2}$$

and if $\hat{\beta} \in \mathbb{R}^M$ are the parameters which minimize the above expression then

$$\mathbb{E}[Y|X] \approx \sum_{r=1}^{M} \hat{\beta}_r \psi_r(X) .$$

Now imagine you have N independent samples $(x_i, y_i)_{i=1}^N$ from the (joint) distribution of (X, Y). Since $\|\cdot\|_2^2 = \mathbb{E}[|\cdot|^2]$ we can use Monte Carlo estimators for the minimization problem (2). So our problem becomes: find $\hat{\beta}$ (which depend on N and the choice of your basis functions) which minimize, over all $\beta \in \mathbb{R}^M$ the expression

$$\frac{1}{N^2} \sum_{i=1}^{N} \left| y_i - \sum_{r=1}^{M} \beta_r \psi_r(x_i) \right|^2 .$$

[5] We know from Taylor's theorem that any smooth function is an (infinite) sum of polynomials, we know from Fourier analysis that every L^2 function is an (infinite) sum of e^{-ikx} with integer k etc.

Note that N^{-2} is just a constant in this minimization and can be dropped. What you're looking at now is a classical ordinary least squares problem.

Exercise 1.1 (Empirical estimation of conditional expectation, 5 marks)**.**

i) Write a method which will estimate the optimal coefficient $\hat{\beta}$ for estimating conditional expectation of $\mathbb{E}[Y|X]$ from samples $(x_i, y_i)_{i=1}^N$. Use the method signature

```
def coeffsForCondiExp(X,Y,hermiteOrder):
```

ii) Write a method which will estimate the conditional expectation given an input and the optimal coefficients. Use the method signature

```
def approxCondExp(X,beta):
```

Use probabilist's Hermite polynomials as basis functions ψ_r so that $\psi_0(x) = 1$, $\psi_1(x) = x$, $\psi_2(x) = x^2 - 1$ etc. These are implemented in `numpy.polynomial.hermite_e`. You can use the built-in least-squares solver: `numpy.linalg.lstsq` or a built-in function to fit the hermite polynomials: `numpy.polynomial.hermite_e.hermefit`.

2 Solving a BSDE numerically

It will help to read [4, Section 5]. Consider a BSDE of the form

$$dY_t = g_t(Y_t, Z_t)\, dt + Z_t\, dW_t\,, t \in [0, T]\,,\ \ Y_T = \xi$$

with appropriate "driver" g and terminal condition ξ. Consider a time grid Π on $[0, T]$ i.e.

$$\Pi := \{(t_i)_{i=0}^N : 0 = t_0 < t_1 < \cdots < t_N = T\}\,. \tag{3}$$

Clearly

$$Y_{t_i} = Y_{t_{i+1}} - \int_{t_i}^{t_{i+1}} g_t(Y_t, Z_t)\, dt - \int_{t_i}^{t_{i+1}} Z_t\, dW_t\,,\ \ i = 0, \ldots, N-1\,,\ \ Y_{t_N} = \xi\,. \tag{4}$$

Writing $Y_i := Y_{t_i}$, $Z_i := Z_{t_i}$, $g_{t_i} := g_i$, $\Delta t_{i+1} := t_{i+1} - t_i$ and $\Delta W_{i+1} := W_{t_{i+1}} - W_{t_i}$ we can approximate (4) by the explicit Euler approximation

$$Y_i \approx Y_{i+1} - g_i(Y_{i+1}, Z_i)\, \Delta t_{i+1} - Z_i \Delta W_{i+1}\,,\ \ i = 0, 1, \ldots, N-1\,,\ \ Y_N = \xi\,. \tag{5}$$

Taking conditional expectation $\mathbb{E}_{t_i}[\cdot] := \mathbb{E}_{t_i}[\cdot|\mathcal{F}_{t_i}]$ in (5) we get

$$Y_i \approx \mathbb{E}_{t_i}\Big[Y_{i+1} - g_i(Y_{i+1}, Z_i)\, \Delta t_{i+1}\Big]\,,\ \ i = 0, 1, \ldots, N-1\,, Y_N = \xi\,. \tag{6}$$

To evaluate this recurrence we still need Z_i. To that end we approximate (4) using implicit Euler scheme obtaining

$$Y_i \approx Y_{i+1} - g_i(Y_i, Z_i)\, \Delta t_{i+1} - Z_i \Delta W_{i+1}\,,\ \ i = 0, 1, \ldots, N-1\,,\ \ Y_N = \xi\,. \tag{7}$$

Multiplying (7) by ΔW_{i+1} and taking the same conditional expectation we obtain:

$$\mathbb{E}_{t_i}\Big[Y_i \Delta W_{i+1}\Big] \approx \mathbb{E}_{t_i}\Big[Y_{i+1}\Delta W_{i+1} - g_i(Y_i, Z_i)\Delta W_{i+1}\, \Delta t_{i+1} - Z_i(\Delta W_{t+1})^2\Big]\,.$$

This is

$$0 \approx \mathbb{E}_{t_i}\Big[Y_{i+1}\Delta W_{i+1}\Big] - Z_i \Delta t_{i+1}\,,\ \ i = 0, 1, \ldots, N-1\,.$$

Together with (6) we thus have the numerical scheme:

$$\begin{aligned} Y_i &\approx \mathbb{E}_{t_i}\Big[Y_{i+1} - g_i(Y_{i+1}, Z_i)\,\Delta t_{i+1}\Big]\,, i = 0, 1, \ldots, N-1\,, Y_N = \xi\,, \\ Z_i &\approx \frac{1}{\Delta t_{i+1}}\mathbb{E}_{t_i}\Big[Y_{i+1}\Delta W_{i+1}\Big]\,. \end{aligned} \tag{8}$$

You will, in particular notice that you need to calculate two conditional expectations at each time step.

Exercise 2.1 (Numerical solution of a BSDE, 4 marks)**.** We will use a BSDE for which we know what the explicit solution is so that you can check that your method works. To that end fix $\mu, r \in \mathbb{R}$, $\sigma \in (0, \infty)$ constants and consider

$$dY_t = \big[rY_t + \sigma^{-1}(\mu - r)Z_t\big]\,dt + Z_t\,dW_t\,, t \in [0, T]\,, Y_T = \xi\,,$$

where $\xi = [S_T - K]_+$ for $K > 0$ fixed and $dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$.

Implement a numerical algorithm based on (8) and your method for estimating conditional expectations.[6] Test its convergence against the exact solution. Plot[7] how it depends on number of Monte Carlo samples N_{MC}, number of time steps N and the number of basis functions M.

3 Deterministic MSA

In the case of deterministic control the Hamiltonian is:

$$H(t, x, y, a) = b(t, x, a) \cdot y + f(t, x, a)\,.$$

Any optimal control, together with the optimal forward and backward equations must satify the ordinary differential equation (ODE) system

$$\begin{cases} \alpha_t \in \arg\max\limits_{a \in A} H(t, X_t^\alpha, Y_t^\alpha, a)\,, \\ dX_t = b(t, X_t^\alpha, \alpha_t)\,dt\,, \;\; t \in [0, T]\,, \;\; X_0 = x\,, \\ dY_t = -(\partial_x H)(t, X_t^\alpha, Y_t^\alpha, \alpha_t)\,dt\,, \;\; t \in [0, T]\,, \;\; Y_T^\alpha = (\partial_x g)(X_T^\alpha)\,. \end{cases} \tag{9}$$

The method of succesive approximation (MSA) is the following iterative method. Fix a stopping criteria $\varepsilon > 0$. Start with a guess for the control $\alpha^{(0)} = (\alpha_t^{(0)})_{t \in [0,T]}$. At each j-th step of the algorithm:

i) solve

$$\begin{aligned} dX_t^{(j)} &= b(t, X_t^{(j)}, \alpha_t^{(j-1)})\,dt\,, \;\; t \in [0, T]\,, \;\; X_0 = x\,, \\ dY_t^{(j)} &= -(\partial_x H)(t, X_t^{(j)}, Y_t^{(j)}, \alpha_t^{(j)})\,dt\,, \;\; t \in [0, T]\,, \;\; Y_T^{(j)} = (\partial_x g)(X_T^{(j)})\,. \end{aligned} \tag{10}$$

ii) find (approximately is sufficient)

$$\alpha_t^{(j)} \in \arg\max_{a \in A} H(t, X_t^{(j)}, Y_t^{(j)}, a)\,. \tag{11}$$

[6] Note that you *must* use the same increments $(\Delta W_i)_{i=1,\ldots N}$ to generate samples from S_T and in (8) otherwise you will not see good results.

[7] You probably want log-log plots to see what's happening.

iii) Aproximate $J^{\alpha^{(j)}} = \int_0^T f(t, X_t^{(j)}, \alpha_t^{(j)})\, dt + g(X_T^{(j)})$. If $J^{\alpha^{(j)}} < J^{\alpha^{(j-1)}}$ then stop: the method failed due to some numerical instability (you can prove that the objective functional should be increasing with j). If $J^{\alpha^{(j-1)}} + \varepsilon > J^{\alpha^{(j)}}$ then stop: the method converged within your stopping criteria. You found the correct (approximate) solution. Otherwise continue to the next iteration.

Of course the ODEs in (10) have to be solved numerically on some grid Π, see (3). Then for each $t_i \in \Pi$ you need to approximate (11) so you can update the control. In [2] they suggest approximating it using 10 iterations of limited memory BFGS method (L-BFGS). This exists in Python, see `scipy.optimize.minimize` and use specifically `method='L-BFGS-B'`. On the other hand, it is often enough to fix some $\delta > 0$ (small) and instead of (11) do

$$\alpha_t^{(j)} = \alpha_t^{(j-1)} + \delta(\nabla_a H)(t, X_t^{(j)}, Y_t^{(j)}, \alpha_t^{(j-1)})$$

i.e. do one step of gradient ascent towards the maximum.

Exercise 3.1 (Deterministic Linear Quadratic control, 4 marks)**.** Implement the MSA algorithm for the deterministic linear quadratic control problem. Use explicit Euler discretization for both the forward and backward ODEs.

Compare to the exact solution (you can solve the Riccati ODE numerically or choose a simple setting where it is solvable by hand). Discuss convergence of the MSA and how it depends on problem and parameter choices.

4 MSA for stochastic control

In the case of stochastic control the Hamiltonian is:

$$H(t, x, y, z, a) = b(t, x, a) \cdot y + \operatorname{tr}(\sigma(t, x, a)^\top z) + f(t, x, a)\,.$$

Any optimal control, together with the optimal forward and backward processes must satify the stochastic differential equation (SDE) system

$$\begin{cases} \alpha_t \in \arg\max_{a \in A} H(t, X_t^\alpha, Y_t^\alpha, Z^\alpha, a)\,, \\ dX_t = b(t, X_t^\alpha, \alpha_t)\, dt + \sigma(t, X_t^\alpha, \alpha_t)\, dW_t\,, \quad t \in [0, T]\,, \quad X_0 = x\,, \\ dY_t = -(\partial_x H)(t, X_t^\alpha, Y_t^\alpha, Z_t^\alpha, \alpha_t)\, dt + Z_t^\alpha\, dW_t\,, \quad t \in [0, T]\,, \quad Y_T^\alpha = (\partial_x g)(X_T^\alpha)\,. \end{cases} \tag{12}$$

The algorithm is then identical as before but (10)-(11) now contain the stochastic components. Hence solving the backward equation amounts to solving a BSDE as in Section 2. You will need to generate MC sample paths so that you can evaluate the conditional expectations. Moreover to approximate $J^{\alpha^{(j)}} = \mathbb{E}\int_0^T f(t, X_t^{(j)}, \alpha_t^{(j)})\, dt + g(X_T^{(j)})$ you need to approximate the expectation.

Exercise 4.1 (Stochastic Linear Quadratic control, 4 marks)**.** Implement the MSA algorithm for the stochastic linear quadratic control problem. Use explicit Euler discretization for both the forward SDE and use a BSDE solver for the backward equation.

Compare to the exact solution (you can solve the Riccati ODE numerically or choose a simple setting where it is solvable by hand). Discuss convergence of the MSA and how it depends on problem and parameter choices.

If you got as far as here you have the MSA working for stochastic problems where the solution is known already - good to have but not that exciting.

Exercise 4.2 (Optimal asset allocation in Heston Model, 6 marks)**.** Assume that we have a risk-free asset with price at time t given by B_t such that $B_0 = 1$ and $dB_t = rB_t dt$. Moreover there is a risky asset with the dynamics given by the stochastic differential equation

$$dS_t = \mu S_t dt + \sqrt{V_t} S_t dW_t^{(1)}, \;\; S_0 = S \tag{13}$$

where

$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}\left[\rho dW_t^{(1)} + \sqrt{1-\rho^2}dW_t^{(2)}\right], \;\; V_0 = v. \tag{14}$$

It can be shown that (14) has a unique solution such that, if $2\kappa\theta > \sigma^2$ then $V(t) > 0$ (provided of course that the initial value is strictly positive).

Formulate and solve a final-time-utility Merton problem with no consumption for an utility function of your choice using the MSA. You will find sensible parameter for the stochastic volatility model e.g. in [1].

Hint. You'll need to modify your code for conditional expectation so that you can condition on two dimensional inputs. That is, you want to find a function $h : \mathbb{R}^2 \to \mathbb{R}$ such that $\mathbb{E}[Y|X_1, X_2] = h(X_1, X_2)$. Try e.g.

$$h(x_1, x_2) \approx \sum_{r_1, r_2 = 1}^{M} \beta_{r_1 r_2} \psi_{r_1}(x_1)\psi_{r_2}(x_2).$$

You can then condition on $(W_{t_i}^{(1)}, W_{t_i}^{(2)})$ or on (S_{t_i}, V_{t_i}) in (8).

If the algorithm fails to converge, try using the Modified MSA described in [3].

References

[1] H. Albrecher and P. Mayer and W. Schoutens and J. Tistaert, The Little Heston Trap, *Wilmott Magazine*, 2007.

[2] Q. Li, L. Chen, C.Tai, and W. E, Maximum principle based algorithms for deep learning, *J. Mach. Learn. Res.*, 18(165), 1–29, 2018.

[3] B. Kerimkulov, D. Šiška, and L. Szpruch. A modified MSA for stochastic control problems. *arXiv:2007.05209*, 2020.

[4] G. dos Reis and D. Šiška. *Stochastic Control and Dynamic Asset Allocation.* `https://www.maths.ed.ac.uk/ dsiska/LecNotesSCDAA.pdf`. 2021.