V4E2 - Numerical Simulation

Sommersemester 2018
Prof. Dr. J. Garcke
Teaching assistant: Biagio Paparella
Tutor: Marko Rajković
(marko.rajkovic@uni-bonn.de)

UNIVERSITÄT BONN

Exercise sheet 11.

To be handed in on **Tuesday, 10.07.2018**.

We started the last part of this course with the finite-horizon problem in optimal control. If v is the value function, the strategy for obtaining a numerical approximation has been: first, discretize the time and get a corresponding DPP for the discretized problem, whose solution we write with $v^{\Delta t}$ (Thm 36); then, choose a space discretization and define $v_n^{\Delta t}$, again with a corresponding DPP. Here we chosen a grid setting with Lagrange interpolation for values between the nodes (Thm 37), but in principle other choices are possible. We obtained a backward-in-time scheme that allowed to completely solve the finite-horizon problem.

Exercise 1. (Solving a finite-horizon problem)

Let's consider the following state dynamics (with notations as done in class):

$$\begin{cases} \dot{x}(t) = t\alpha(t) & t \in (0,1] \\ x(0) = x_0 & t = 0 \end{cases}$$

To further you go, the more you spend...

$$l(z,a) = az$$

...but if you go in the right place, you receive a good compensation:

$$\psi(x) = \begin{cases} 0 & x \leq \frac{e}{2} \\ x - \frac{e}{2} & otherwise \end{cases}$$

The set $\mathcal{A}$ of allowed controls is defined as follows: divide the time interval $[0,1]$ in 10 parts. Then $\alpha(t) : [0,1] \to \mathbb{R}$ belongs to $\mathcal{A}$ iff it is piecewise constantly 0 or 1 on every of that subdivision (you can think of $\alpha = 0$ as stopping, the other case as moving).
Consider it as a finite-horizon optimal control problem, where you want to study the cost of every dynamic depending on the possible starting point. Plot your numerical solution of the value function.

(6 Punkte)

Other absolutely important topics covered in class have been the Infinite Horizon variant, and then its formulation in the context of Reinforcement Learning. A recap will be done in the next exercise sheet, rather we conclude this problem set with a property concerning the Howards algorithm about policy iteration and a (hopefully) enjoyable Python example.
Let $\mathcal{M}$ denote the set of real valued $N \times N$ matrices. We denote by α^a the vector

$$\alpha^a = (a, \dots, a)^\top.$$

Let $\mathcal{A}^N$ be a compact set of vectors $\alpha = (\alpha_1, \dots, \alpha_n)^\top$. We define the set of minimizers associated with $F(x)$ by

$$\mathcal{A}_x := \{\alpha \in \mathcal{A}^N : \ B(\alpha)x - c(\alpha) = F(x)\}$$

where

$$F(x) := \min_{\alpha \in \mathcal{A}^N} (B(\alpha)x - c(\alpha))$$

and it's component wise set

$$\mathcal{A}_{x,i} := \{a \in \mathcal{A} : \ [B(\alpha^a)x - c(\alpha^a)]_i = [F(x)]_i\}$$

where

$$F_i(x) = \min_{\alpha \in \mathcal{A}^N} [B(\alpha)x - c(\alpha)]_i.$$

The assumptions of Theorem 45 are:

(H1) For every $\alpha \in \mathcal{A}^n$, the matrix $B(\alpha)$ is monotone (B is invertible and B^{-1} is component wise ≥ 0)

(H2) When $\mathcal{A}$ is an infinite compact set, the functions $\alpha \in \mathcal{A}^N \to B(\alpha) \in \mathcal{M}$ and $\alpha \in \mathcal{A}^N \to c(\alpha) \in \mathbb{R}^N$ are continuous.

We assume for $\alpha = (\alpha_1, \ldots, \alpha_N) \in \mathcal{A}^n$ that $B_{i,j}(\alpha)$ and $c_i(\alpha)$ depend only on α_i.

Exercise 2. (Consider the setup of Theorem 45)

(i) Show under the assumptions from Theorem 45 it holds, that for every $v \in \mathbb{R}^N$ and

$$d(\alpha_i^{v+w}, A_{v,i}) \to 0, \quad w \in \mathbb{R}^N, \ \|w\| \to 0$$

with $\alpha_i^{v+w} \in A_{v+w,i}$. Hint: Prove by contradiction. Define the set

$$K_\delta := \{a \in \mathcal{A} : \ d(a, A_{v,i}) \geq \delta > 0\}$$

and consider properties of

$$m_\delta := \inf_{a \in K_\delta} [B(a)v - c(a)]_i$$

(ii) Show that using (i) one can find $\alpha^{k,*} \in A_{v^*}$ such that for $\alpha^{k+1} = \alpha^{v^k}$

$$B(\alpha^{k+1}) - B(\alpha^{k,*}) \to 0$$

for $k \to \infty$. Hint: use $\alpha^{v^k} = \alpha^{v+h_k}$, $h_k := v^k - v^*$.

(4+4 Punkte)

Exercise 3. (Bonus exercise)

An introductory example playing Tic-Tac-Toe is presented in

Sutton, R., Barto, A. Reinforcement Learning, MIT Press, 1998.

beginning from page 10. Code that allows simulations is provided here: `https://github.com/ShangtongZhang/reinforcement-learning-an-introduction/blob/master/chapter01/TicTacToe.py` Discuss the following issues stated there. You can try to modify the code to obtain ideas.

- Self-Play: Suppose, instead of playing against a random opponent, the reinforcement learning algorithm described above played against itself, with both sides learning. What do you think would happen in this case? Would it learn a different policy for selecting moves?

- Symmetries: Many Tic-Tac-Toe positions appear different but are really the same because of symmetries. How might we amend the learning process described above to take advantage of this? In what ways would this change improve the learning process? Now think again. Suppose the opponent did not take advantage of symmetries. In that case, should we? Is it true, then, that symmetrically equivalent positions should necessarily have the same value?

- Greedy Play: Suppose the reinforcement learning player was greedy, that is, it always played the move that brought it to the position that it rated the best. Might it learn to play better, or worse, than a non-greedy player? What problems might occur?

The book is available via `http://incompleteideas.net/book/the-book-2nd.html`

(5* Punkte)