Homework assignment 1

Optimal control and dynamic programming (4SC000), TU/e, 2018-2019

Lecturer: Duarte Antunes

Submission: The solutions of Assignments 1, 2, 4 should be submitted via Matlab grader and via canvas (.m files) and the solution of Assignment 3 should be submitted only via canvas. For Assignment 3, use the solution template available on Canvas.
Grade: The overall assignment will be graded on a scale 0-10 according to the following distribution

Assignment	1	2	3	4	Total
Max score	3	2	3	2	10

There are 6 (hidden and visible) tests for each of the assignments 1,2,4. For each of these assignments, the grade will be given by

$$\text{grade assignment} = y(x) \times \text{Max score}$$

where $x \in \{0, 1, \dots, 6\}$ denotes the number of tests (either hidden or visible) passed and $y(x)$ is specified in the next table

x	0	1	2	3	4	5	6
$y(x)$	0	0.2	0.4	0.4	0.5	0.6	1

.

Plagiarism: The matlab code will be checked with a plagiarism tool. In case of fraud, no points will be given for the entire assignment.
Deadline: December 4th, 23h45, 2018.

---

1. Dynamic programming (Viterbi) algorithm for angle estimation

Suppose that we wish to estimate the angle of a spinning wheel with a camera (see Figure 1). Different colors with continuously varying values of hue according to the HSV color representation are placed on the wheel and the camera can only detect a limited range of the wheel. The angle is quantized into N values. Due to changes in brightness, the camera processing algorithm can provide wrong estimates of the hue value, but these are typically close to the correct value. The wheels angle, denoted by θ, evolves according to

$$\dot{\theta}(t) = \omega + d(t)$$

where ω is constant, and $d(t)$ models stochastic disturbances. Let

$$\theta_k \in \mathcal{A} := \{0, \frac{2\pi}{N}, \ldots, \frac{2\pi(N-1)}{N}\}$$

be the quantized angle at time $k\tau$, where τ denotes the sampling period and $k \in \mathbb{N}_0$. Then, we can write

$$\theta_{k+1} = \text{mod}(\theta_k + a + w_k, 2\pi)$$

where $a = \omega\tau$ is a constant assumed to belong to $\mathcal{A}$ and w_k models disturbances resulting from $d(t)$. The disturbances are assumed to be independent and identically distributed and the initial value of the angle follows a known probability distribution. These distributions are denoted by

$$\text{Prob}[\theta_0 = i\frac{2\pi}{N}] = \alpha_i, \qquad \text{Prob}[w_k = i\frac{2\pi}{N}] = \beta_i, \quad \forall k \text{ and } i \in \mathcal{I} := \{0, 1, \ldots, N-1\},$$

for $\alpha_i \in [0,1]$, $\beta_i \in [0,1]$. The measurement at time k obtained from the algorithm processing camera data is denoted by $y_k \in \mathcal{A}$ and follows the model

$$y_k = \text{mod}(\theta_k + v_k, 2\pi)$$

where $\text{Prob}[v_k = i\frac{2\pi}{N}] = \gamma_i, \quad \forall k$ and $i \in \mathcal{I}$. Let $I_k = \{y_0, \ldots, y_k\}$ be the information available up to time k. The goal of this exercise is to compute the most likely sequence of angles given I_k, i.e.,

$$\arg \max_{\theta_0, \theta_1, \ldots, \theta_k} \text{Prob}[\theta_0, \theta_1, \ldots, \theta_k | I_k]. \tag{1}$$

We can see this as a state estimation problem for a hidden Markov model ([1, Sec. 2.2.2]). Such problems are discussed in the appendix and can be formulated and solved with dynamic programming (Viterbi algorithm).

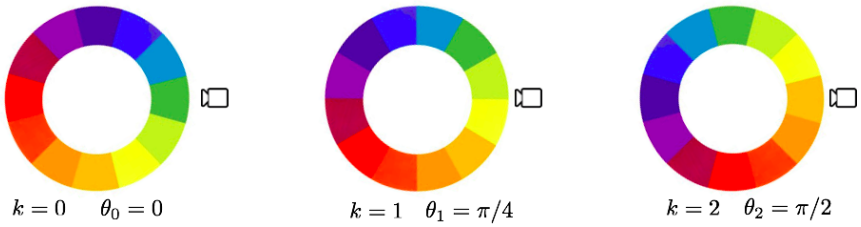

Figure 1: Illustration of spinning wheel and camera sensor

Assignment 1 Program a matlab function

`[thetaest] = viterbiangleestimation(aint,yint,alpha,beta,gamma)`

that computes the solution to (1). The input arguments are:

- An integer corresponding to the value aint= $\frac{N}{2\pi}a$.
- Column vector with vector of integers corresponding to the scaled measurements

$$\text{yint} = \frac{N}{2\pi}\begin{bmatrix} y_0 & \dots & y_k \end{bmatrix}^\intercal$$

- Column vectors with the probability distributions alpha= $\alpha = \begin{bmatrix} \alpha_0 & \dots & \alpha_{N-1}, \end{bmatrix}^\intercal$, beta= $\beta = \begin{bmatrix} \beta_0 & \dots & \beta_{N-1} \end{bmatrix}^\intercal$, gamma = $\gamma = \begin{bmatrix} \gamma_0 & \dots & \gamma_{N-1} \end{bmatrix}^\intercal$.

The output is:

- a column vector of integers

$$\text{thetaest} = \frac{N}{2\pi}\begin{bmatrix} \theta_0 & \dots & \theta_k \end{bmatrix}^\intercal,$$

  where $\begin{bmatrix} \theta_0 & \dots & \theta_k \end{bmatrix}^\intercal$ is the optimal solution to (1). If there exists more than one optimal solution, output the solution $\begin{bmatrix} \theta_0^* & \dots & \theta_k^* \end{bmatrix}^\intercal$ with the following property: if $\begin{bmatrix} \bar{\theta}_0 & \dots & \bar{\theta}_k \end{bmatrix}^\intercal$ is another optimal solution, then either $\theta_0^* < \bar{\theta}_0$ or if $\theta_i^* = \bar{\theta}_i$ for $0 \leq i < j$ and for a maximal j such that $1 \leq j \leq k-1$, then $\theta_j^* < \bar{\theta}_j$.

2. Optimal stopping: the secretary problem

The secretary problem is a famous optimal stopping time problem, which can be explained as follows. Consider an administrator who wishes to hire the best secretary out of n applicants. The applicants are interviewed one by one in random order. The decision to hire or not a given applicant must be made immediately after the interview and cannot be revoked. To make this decision, the administrator can rank all applicants interviewed so far, but the quality of the applicants yet to be interviewed is unknown to the administrator. The challenge for the administrator is to determine when to hire a secretary and stop interviewing in order to maximize the probability of selecting the best applicant.

The secretary problem can be solved via dynamic programming (see [2]) and it can be proved that the optimal policy is the following: reject the first $r-1$ applicants and then accept the first applicant who is ranked higher than every applicant interviewed so far (if the best candidate was one of the first $r-1$ interviewed the last interviewed candidate is selected).

Assignment 2

Program a Matlab function

```
[r] = secretaryproblem(n);
```

that computes r given n.

3. Implementing A* with the Dijkstra's algorithm

Consider a weighted graph with n nodes, and weights $w_{ij} \geq 0$ representing the cost of the path (arrow) from node i to node j. Moreover, considered the graph search problem of finding the path from a given initial node i_0 to a final node i_f minimizing the sum of weights associated to the path. For this shortest path problem, consider the Dijkstra's algorithm and the A* algorithm with heuristic $h(i)$, $i \in \{1, \ldots, n\}$, as described in slides 19 and 25 of lecture 3, respectively, and with the following convention: if in step 1 of either algorithm, at a given iteration, two or more neighbours (nodes) result in the same minimum value of the cost-to-come, then the node with the smallest label i is selected in both algorithms. The heuristic is such that, for each node i, $h(i)$ is less than or equal to the cost of the optimal path from node i to node i_f and satisfies

$$h(i) \leq h(j) + w_{ij}.$$

Under this assumption, the following weights are non-negative:

$$\bar{w}_{ij} = w_{ij} + h(j) - h(i).$$

Assignment 3 Prove that the A* algorithm for the orignal graph with weights w_{ij} and heuristic $h(i)$ is equivalent to the Dijktra's algorithm for a graph with weights $\bar{w}_{ij}$ in the sense that both generate the same list of OPEN nodes at every iteration and consequently generate the same optimal path.

4. Bayes filter for angle estimation

The goal of this exercise is to compute, in the scope of the angle estimation problem 1,

$$\text{Prob}[\theta_k = c|I_k].$$

Assignment 4

Program a matlab function

```
[p] = bayesangleestimation(aint,yint,alpha,beta,gamma)
```

that computes $\text{Prob}[\theta_\ell = \frac{2\pi}{N} i|I_\ell]$ for $\ell \in \{0,\dots,k\}$. The input arguments are as in problem 1 and the output is a $N \times (k+1)$ matrix with entries

$$p_{i\ell} = \text{Prob}[\theta_{\ell-1} = \frac{2\pi}{N}(i-1)|I_{\ell-1}], \quad \ell \in \{0,\dots,k\}, i \in \{1,\dots,N\}.$$

Appendix - Viterbi algorithm

Consider a dynamic system

$$x_{k+1} = f(x_k, w_k),$$

where $x_k \in \{1, \ldots, n\}$ is the state and $w_k \in \{1, \ldots, n_w\}$ is a sequence of independent and identically distributed random disturbance variables and the output equation

$$y_k = h(x_k, v_k),$$

where $y_k \in \{1, \ldots, p\}$ is the output and $v_k \in \{1, \ldots, n_v\}$ is a sequence of independent and identically distributed random noise variables. The probability distribution of the initial state is assumed to be known $\delta_i = \text{Prob}[x_0 = i]$, $i \in \{1, \ldots, n\}$. From the maps w and v one can define the matrices

$$P_{ri} = \text{Prob}[x_{k+1} = r | x_k = i], \quad R_{\ell i} = \text{Prob}[y_k = \ell | x_k = i].$$

These equations characterize a hidden Markov model with n states. Consider the problem of estimating the most likely sequence of states up to time k, $x_0, \ldots, x_k$ given the information $y_0, \ldots, y_k$, i.e.,

$$\arg \max_{x_0, \ldots, x_k} \text{Prob}[x_0, \ldots, x_k | y_0, \ldots, y_k].$$

Following similar derivations as in [1, Sec. 2.2.2], we can conclude that $\text{Prob}[x_0, \ldots, x_k | y_0, \ldots, y_k]$ is proportional to

$$\delta_{x_0} R_{y_0 x_0} \Pi_{\ell=1}^{k} P_{x_\ell x_{\ell-1}} R_{y_\ell x_\ell}.$$

Maximizing this quantity is the same as maximizing its logarithm[1] and, by multiplying the expression by -1, we can convert this problem into the following minimization problem

$$\arg \min_{x_0, \ldots, x_k} -\log(\delta_{x_0} R_{y_0 x_0}) - \sum_{\ell=1}^{k} \log(P_{x_\ell x_{\ell-1}} R_{y_\ell x_\ell})$$

or equivalently into the problem

$$\min_{x_0} \underbrace{\min_{u_0, u_1, \ldots, u_{h-1}} \sum_{\ell=0}^{h-1} g_\ell(x_\ell, u_\ell) + g_h(x_h)}_{:= J_0(x_0)}$$

subject to

$$x_{\ell+1} = u_\ell, \quad \ell \in \{0, 1, \ldots, h-1\},$$

where

$$g_\ell(x_\ell, u_\ell) = \begin{cases} -\log(\delta_{x_0} R_{y_0 x_0}) - \log(P_{u_0 x_0} R_{y_1 u_0}) \text{ if } \ell = 0, \\ -\log(P_{u_\ell x_\ell} R_{y_{\ell+1} u_\ell}) \text{ otherwise,} \end{cases}$$

and $g_h(x_h) = 0$ for every x_h and the initial condition x_0 is arbitrary. For each fixed x_0 this is a dynamic programming problem. One can then find the optimal policy $u_k = \mu_k(x_k)$, $k \in \{0, \ldots, h-1\}$, and the cost-to-go $J_0(x_0)$, solve

$$x_0^* = \text{argmin} J_0(x_0),$$

and use x_0^* as the initial condition to find the optimal path x_0^*, x_1^*, $\ldots$, x_h^*. Alternatively, one can find the shortest path in the graph of Figure 2. This follows from the connection between transition diagrams and discrete optimization problems.

[1] Consider $-\log(0) = \infty$

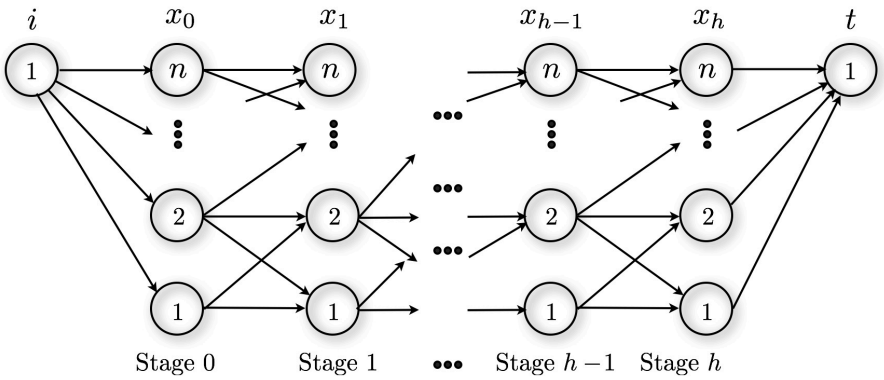

Figure 2: Graph corresponding to the dynamic programming (Viterbi) algorithm. The arrows from node i to node x_0 at stage 0 have cost $-\log(\delta_{x_0} R_{y_0 x 0})$. The arrows from a node x_k to a node x_{k+1} have cost $-\log(P_{x_\ell x_{\ell+1}} R_{y_{\ell+1} x_{\ell+1}})$ and the arrows from a node x_h to node t have cost 0. Finding the optimal path from node i to node t, i.e., $(i, x_0^*, x_1^*, \ldots, x_h^*, t)$ yields the optimal estimated state values.

References

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume 1.* Athena Scientific, 2017.

[2] M. Beckmann, *Dynamic programming and the secretary problem*, 1990, vol. 19.