Exercise 4: Markov Processes, Cellular Automata and Fuzzy Logic

Formal Methods II, Fall Semester 2013

Distributed: 8.11.2013
Due Date: 29.11.2013
Send your solutions to: tobias.klauser@uzh.ch or deliver them in class.

Markov Processes – Theoretical Exercise

1. A Markov model can be used for a simplified description of the behavior of cows.[1] Assume that a single cow's behavior can always be characterized by one of three states: it is either standing around, lying on the meadow, or eating grass.

   If it is standing, there is a 0.5 probability it starts eating; otherwise it will lie down with probability 0.3 or remain standing in all other cases. If the cow is lying on the meadow, it will stay that way in 70% of the cases; it will stand up again in 25% of the cases and will start eating in the remaining cases. If the cow is eating, it will change its behavior to just standing around with a probability of 0.2; otherwise it will either continue eating or lie down with equal probabilities.

   (a) (1 point) Draw a process diagram that shows the states and transition arcs between them labeled with their respective transition probabilities.

   (b) (2 points) Give the transition matrix and calculate the stationary distribution, i.e. the limiting probabilities of a cow being (standing, lying, eating).

Hidden Markov Models – Practical exercise

2. (5 points) Extract the T9-like predictive text input system (`t9.py`) together with the text file (`PitPendulum.txt`) from `FM13_ex4.zip`.

   This program prompts the user to input a sequence of key strokes (e.g. `43556`) and prints out the most probable sequence of hidden states corresponding to the observed sequence (in our case, `hello`) – as demonstrated during the lecture.

   Modify the code in order to obtain an iPhone-like predictive text input system: In other words, we would like to input a sequence of key strokes (e.g. `whst`), and see what the system suggests us (in our case, something like `what` which sounds more English – correcting the `s` into the letter `a` which lies next to it on the iPhone keyboard).

[1] This example is inspired by the winner of the 2013 Ig Nobel Prize in probability, "Are Cows More Likely to Lie Down the Longer They Stand?", B. J. Tolkamp, et al., Applied Animal Behaviour Science, vol. 124, nos. 1-2, 2010, pp. 110. - See `http://www.improbable.com/ig/winners/#ig2013` for more information.

Obviously, only using a Hidden Markov Model will never provide a perfect solution. Convince us however about the potential of your implementation by showing how a particular word (other than `what`) can still be correctly recognized even though several keys are mistyped!

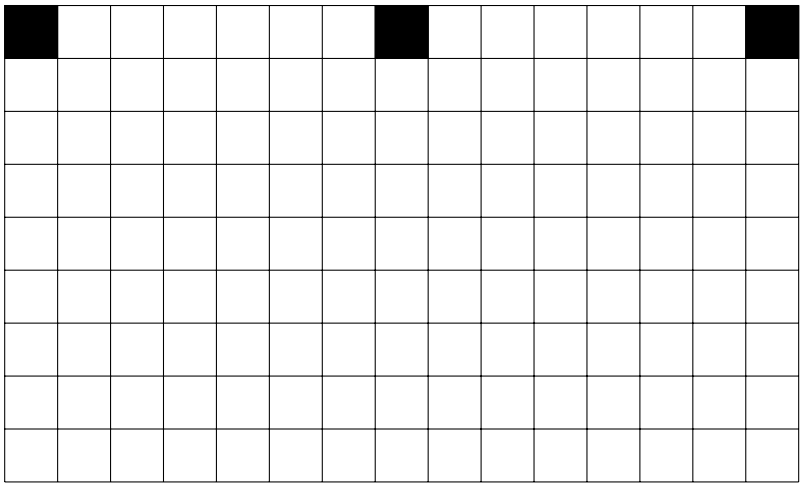
Cellular Automata

3. Given is a one-dimensional binary CA with a neighborhood of $r = 1$ with Wolfram's rule number 210. Apply periodic boundary condition (roll-over).

   (a) (1 point) Fill in the rule table (attention to the row order):

a_{n-1}^t	a_n^t	a_{n+1}^t	a_n^{t+1}
1	1	1	
1	1	0	
1	0	1	
1	0	0	
0	1	1	
0	1	0	
0	0	1	
0	0	0	

(b) (2 points) Compute the CA (using cyclic boundary conditions) over 8 generations by hand (fill the grid).



(c) (2 bonus points) Which rule number would lead to the inverted version[2] of the pattern above, assuming the CA starts with the inverted initial configuration? Shortly explain your reasoning.

4. (4 points) Mark each of the following statements about Cellular Automata (CA) as either **t**rue or **f**alse:

	t	**f**
Each cell of a CA can be looked upon as a Finite State Machine.	☐	☐
In 1D CAs, the complexity of the emerging patterns depends mostly on the neighborhood radius r.	☐	☐
Given an arbitrary elementary CA (with rule table) and the cell states at iteration i, the initial configuration can always be computed.	☐	☐
Given the cell states of a 1D CA at arbitrary iterations i and $i+1$ one can in some cases reproduce the rule table.	☐	☐
In 2D CAs, the Moore neighborhood consists of a central cell and 4 neighbors whereas the von Neumann neighborhood consists of the central cell and 8 neighbors.	☐	☐
The rule table of a 1D CA with fixed boundary conditions contains fewer rules than the rule table of a 1D CA with cyclic boundary conditions.	☐	☐
CAs can be used to simulate processes from e.g. physics or biology.	☐	☐
A majority of CA rules are capable of universal computation.	☐	☐

[2] all black cells replaced by white cells and vice versa

Fuzzy Logic

5. In the lecture, you will be introduced to the inverted pendulum problem in which a pole mounted on top of a cart has to be balanced at the unstable equilibrium position.

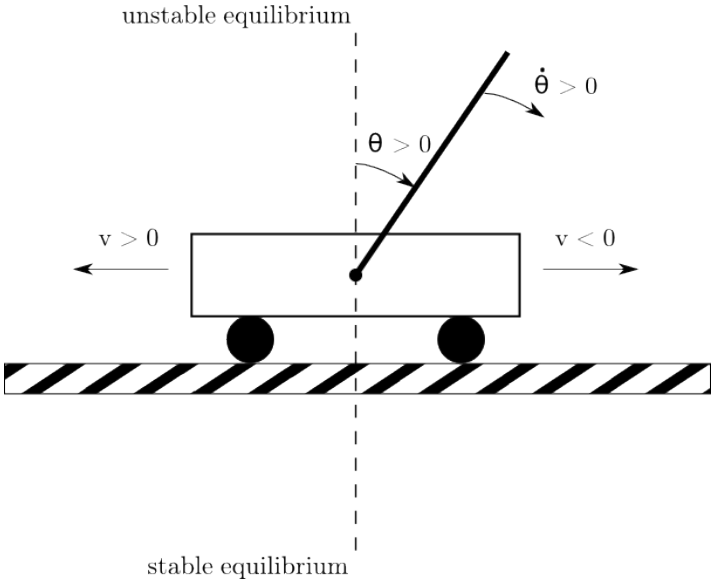


Figure 1: The inverted pendulum on the cart.

Note: You may want to have a look at section 7.4 of the additional Fuzzy Logic course notes which describe this topic. Also the rather counter-intuitive choice of axes (see Figure 1) is explained there.

Another problem with the same plant is the swing-up problem. In this case, the controller has to move the pole from the stable equilibrium to the unstable equilibrium instead of starting in the latter equilibrium and merely keeping the pole there. In physical terms, this is done by pumping energy into the system, leading to an increasingly high amplitude until the pole resides at the desired position. The original pole balancing controller algorithm kicks in at that point.

Your task is to design a fuzzy logic controller for the swing-up problem using the variables defined in the lecture notes (in order to get a feeling of what the swing-up controller is supposed to do have a look at any relevant videos like the one shown in class). You do not have to care about the transition to the original balancing algorithm. In particular, you shall:

(a) (1 point) Define the membership functions.

(b) (2 points) Construct the set of fuzzy rules. Shortly explain your particular choice of rules.

(c) (2 points) Using the respective geometric output functions, show how to defuzzyfy the output when, during one cycle, the kinetic energy is maximal and when it is minimal (these are two distinct cases). You can assume $\theta = 180\,^\circ$, $\dot{\theta} = \frac{36\,^\circ}{s}$ for the case where kinetic energy is maximal and $\theta = 90\,^\circ$, $\dot{\theta} = \frac{0\,^\circ}{s}$ where kinetic energy is minimal.

Hints / Tips

- Initially the pendulum is at the stable equilibrium position ($\theta = 180\,^\circ$) with zero angular velocity.
- Assume that the boundaries of the input variables (angle and angular velocity) are: $\theta \in (-180\,^\circ, 180\,^\circ]$, $\dot{\theta} \in [-90\,^\circ/s, 90\,^\circ/s]$ and the boundaries of the output variable (motor voltage) is $v \in [-10V, 10V]$.
- The goal of the stabilizing controller that was discussed during the lecture is to make the pendulum maintain its position at the unstable equilibrium point ($\theta = 0\,^\circ$). The goal of the swing-up controller that you are supposed to design in this exercise is to pump energy into the system so that the pendulum starts swinging and eventually reaches a position near the equilibrium point (thats when the stabilizing controller is supposed to kick in).
- Pay attention to the definition of the positive and/or negative values of each variable that is shown in Figure 1 and in the supplemental lecture notes on fuzzy logic.