Task Sheet 10

Universität Konstanz

Free energy principle

Deadline 10:00am July 4, 2023

Review on July 4 & 5, 2023

Lecture: *AI for autonomous robotics*, Summer Term 2024

Lecturer: Prof. Dr.-Ing. Heiko Hamann

Tutor: Jonas Kuckling & Paolo Leopardi

Task 10.1 Theoretical questions

a) Explain in your own words how the Kullback-Leibler divergence can be used to compare two probability distributions and how it relates to the Free Energy Principle. (*6 points*).

b) Explain the two steps a robot can take to minimize the free energy (*4 points*).

Task 10.2 Free energy principle control of a robot

In this exercise, you will implement a robot that is controlled by the free energy principle. The robot lives in a simple one dimensional world. It has a sensor to detect the local value of the ground s_c (a single floating point value, representing the grey scale color of the ground) and a sensor detecting its own motor output s_m. The only actuator output the robot is capable of is the motor output a which displaces the robot along the axis.

Following the formulas presented in the lectures, the robot maintains beliefs μ_c and μ_m about the causes of s_c and s_m. The robot assumes that it's sensor readings are caused by the underlying belief and some zero-mean Gaussian noise: $s_c = \mu_c + \xi_c$, $s_m = \mu_m + \xi_m$. Additionally, the robot knows that its speed will be dependent on the measured value and some zero-mean Gaussian noise: $\mu_m = \mu_c + \psi_m$.

The free energy is now defined as $F = \frac{1}{2} * \left(\pi_{z_c}(s_c - \mu_c)^2 + \pi_{z_m}(s_m - \mu_m)^2 + \pi_{w_m}(\mu_m - \mu_c)^2\right)$, where π_i is the inverse of the variance $1/(\sigma_i)^2$. The update rules for the beliefs and actions are:

- $\dot{\mu}_c = -k * (\pi_{z_c}(\mu_c - s_c) + \pi_{w_m}(\mu_c - \mu_m))$
- $\dot{\mu}_m = -k * (\pi_{z_m}(\mu_m - s_m) + \pi_{w_m}(\mu_m - \mu_c))$
- $\dot{a} = -k * (\pi_{z_m}(s_m - \mu_m))$

At every step, the robot updates its belief and actions according to the following formulas: $\mu_c = \mu_c + \dot{\mu}_c$, $\mu_m = \mu_m + \dot{\mu}_m$, $a = a + \dot{a}$. The robot, then moves x units in the world, with $x = speed * a$, where *speed* is a scaling factor (e.g., 0.01).

1. Write a simulation, in which the robot is operating according to the above principles for 6000 time steps. Set the sensing of the color to a constant value of $s_c(x) = 1$ (invariant of the actual position of the robot). For the four free parameters, choose $k = 0.3$, $\pi_{z_c} = \pi_{z_m} = \pi_{w_m} = 1$. At every time step, log the position, free energy, s_c, μ_c, s_m and μ_m. Then plot the development of position, free energy, s_c, μ_c, s_m and μ_m as a line plot.

2. Change your simulation to a gradient environment: $s_c(x) = 10 - x$. What do you observe? Also consider a periodic environment with $s_c(x) = cos(x) + 1.1$.

3. Compare the following cases across the three environments:

    - $k = 0.01$, $\pi_{z_c} = \pi_{z_m} = \pi_{w_m} = 1$: the standard case

- $k = 0.01$, $\pi_{z_c} = 0.0001$, $\pi_{z_m} = \pi_{w_m} = 1$: the robot does not update its belief for μ_c
- $k = 0.01$, $\pi_{z_m} = 0.0001$, $\pi_{z_c} = \pi_{w_m} = 1$: the robot does not update its belief for μ_m.

For your submission, please submit the code (and some short documentation on how to execute it), as well as some documentation of your results. The presentation of your results is up to you and can take many forms. For example, you could submit a short PDF file with plots or screenshots from your terminal. Alternatively, you could submit a video of the screen capture, where you explain what is happening. The spirit should be to not just complete the programming task and done - but to play with your little sim a bit. Explore for yourself. Let us know if you found something interesting. Have fun!