Exercise 8 solutions

July 6, 2023

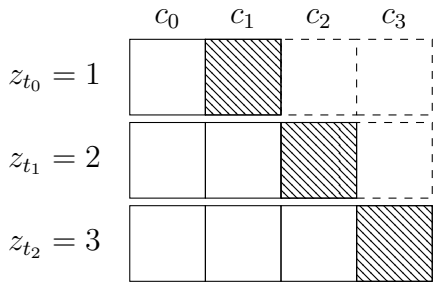
Exercise 1: Counting Model

A robot applies the so-called simple counting approach to build a grid map of a 1D environment consisting of the cells $c_0, \ldots, c_3$. While standing in cell c_0, the robot integrates four measurements $z_{t_0}, \ldots, z_{t_3}$. After integrating these measurements, the resulting belief of the robot with regards to the occupancy of the four cells is $b_0 = 0,\ b_1 = \frac{1}{4},\ b_2 = \frac{2}{3},\ b_3 = 1$. Given that the first three measurements are $z_{t_0} = 1,\ z_{t_1} = 2,\ z_{t_2} = 3$, compute the value of the last measurement z_{t_3}.

A counting model will compute the belief of a cell c_i as:

$$b_i = \frac{\text{hits}_i}{\text{hits}_i + \text{misses}_i}$$

For convenience, let us represent graphically what we know about the measurements. For each measurement we denote with a shaded rectangle a hit, with an empty rectangle a miss and with a dashed rectangle the area which is not subject to update:



This representation tells us that up until time t_2 the hit/miss scenario is the following:

	c_0	c_1	c_2	c_3
hits	0	1	1	1
misses	3	2	1	0

For each cell let us now consider the possible scenarios for a new measurement z_{t_3}:

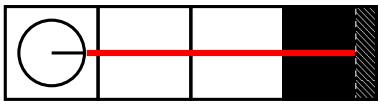
- For b_0 to be 0, either we get another miss or we should not be subject to update.

- For b_1 to be $\frac{1}{4}$ we necessarily need another miss, hence $z_{t_3} > 1$.
- For b_2 to be $\frac{2}{3}$ we necessarily need another hit, hence we have $z_{t_3} = 2$.
- For b_3 to be 1, either we get another hit or the cell is not subject to update. The latter is consistent with our observation of $z_{t_3} = 2$.

We thus conclude that $z_{t_3} = 2$.

Exercise 2: Occupancy Mapping

A robot has to build an occupancy grid map (cells $c_0, \ldots c_n$) of a simple one-dimensional environment using a sequence of measurements from a range sensor.



Assume a very simple sensor model: every grid cell with a distance (based on its coordinate) smaller than the measured distance is assumed to be occupied with $p = 0.3$. Every cell behind the measured distance is occupied with $p = 0.6$. Every cell located more than $20cm$ behind the measured distance should not be updated. Calculate the resulting occupancy grid map using the inverse sensor model (see mapping lecture slide 32) using Python.

Assign the cell coordinates, which span from 0 to 200 (both endpoints included) with increments of 10, to one array `c` *and the belief values to another array* `m`*. Use* `matplotlib.pyplot.plot(c,m)` *to visualize the belief. The measurements and the prior belief are given in the following table.*

Grid resolution	*10 cm*
Map length (1D only)	*2 m*
Robot position	c_0
Orientation (of the sensor)	*heading to c_n (see figure)*
Measurements (in cm)	*101, 82, 91, 112, 99, 151, 96, 85, 99, 105*
Prior	*0.5*

In order to solve this exercise we will use log-odds, as they provide a very simple update formula, with only one addition and one subtraction.

Recall the log-odds update formula on slide 29 of the grid mapping lecture:

$$\ell(m_i|z_{1:t}, x_{1:t}) = \ell(m_i|z_t, x_t) + \ell(m_i|z_{1:t-1}, x_{1:t-1}) - \ell(m_i)$$

From the exercise we know that the prior should be (formula given on slide 28):

$$p(m_i) = 0.5 \quad \Rightarrow \quad \ell(m_i) = \log \frac{p(m_i)}{1 - p(m_i)} = 0$$

We also know that the inverse sensor model is the following:

$$p(m_i|z_t, x_t) = \begin{cases} 0.3 & \text{if } \text{position}(m_i) \leq z_t \\ 0.6 & \text{if } \text{position}(m_i) > z_t \ \wedge \ \text{position}(m_i) \leq z_t + 20\,\text{cm} \\ 0.5(\text{unused}) & \text{if } \text{position}(m_i) > z_t + 20\,\text{cm} \end{cases}$$

The log-odds ratio should be applied to this function in order to obtain $\ell(m_i|z_t, x_t)$. Note that unused in this context means we should not update the corresponding m_i cells. Doing an update with $p(m_i|z_t, x_t) = 0.5$ would be equivalent, since $l(0.5) = 0$, but computationally more expensive.

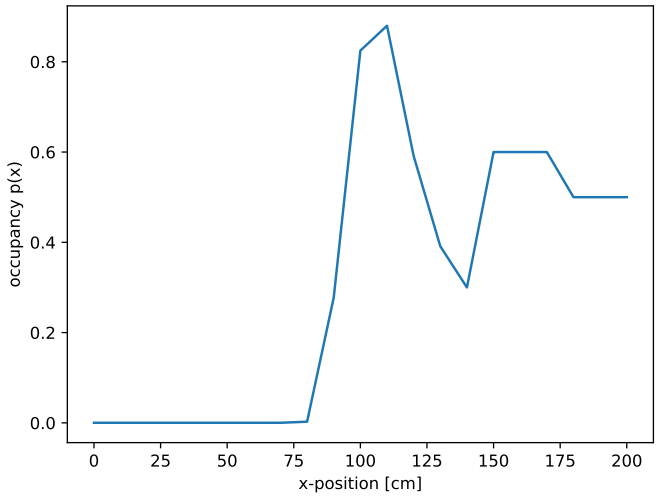
The solution of this exercise will thus involve applying for each measurement and for each cell the log-odds update formula. Once we are done with that we convert from log-odds to probability and display the output. Note that the inverse transformation is the unique solution w.r.t. p of the log-odds definition:

$$\begin{gathered} \ell = \log \frac{p}{1-p} \ \Rightarrow \ \exp \ell = \frac{p}{1-p} \\ \Rightarrow \ (1-p)\exp \ell = p \ \Rightarrow \ \exp \ell = p(1+\exp \ell) \\ \Rightarrow \ p = \frac{\exp \ell}{1+\exp \ell} = \frac{\exp \ell + 1 - 1}{1+\exp \ell} = 1 - \frac{1}{1+\exp \ell} \end{gathered}$$

```
import numpy as np
import matplotlib.pyplot as plt

prob_to_logodds = lambda p: np.log(p / (1 - p))
logodds_to_prob = lambda l: 1 - 1. / (1 + np.exp(l))


# Inverse sensor model in log-odds form
def log_inv_sensor_model(z, c):
    if c > z - 10:
        # The sensor detects a wall for this cell
        return prob_to_logodds(0.6)
    # The sensor detects free space for this cell
    return prob_to_logodds(0.3)

if __name__ ==  '__main__':

    # Cell position (in cm) for each cell
    c = range(0, 201, 10)

    # Map belief in log-odds form
    logodds = np.zeros(len(c))

    # Set of measurements (in cm)
    meas = [101, 82, 91, 112, 99, 151, 96, 85, 99, 105]
```

```
    # Initial prior of the cells, the non informative prior of p = 0.5 will
    # yield 0 in log-odds form, so technically we could leave it out in the
    # the log-odds update formula
    prior = prob_to_logodds(0.5)
    print("prior:", prior)

    # Integrate each measurement
    for i in range(len(meas)):
        # Update every cell of the map
        for j in range(len(c)):
            # Anything beyond 20cm of the measurement should not be updated
            if c[j] > meas[i] + 20:
                continue
            # Log-odds update formula for the cells
            logodds[j] = logodds[j] - prior + log_inv_sensor_model(meas[i], c[j])

    # Apply the inverse transformation from log-odds to probability
    m = logodds_to_prob(logodds)
    # Plot the resulting estimate
    plt.plot(c, m)
    plt.xlabel("x-position [cm]")
    plt.ylabel("occupancy p(x)")
    plt.savefig("graph.pdf")
    plt.show()
```

The resulting belief is the following:



Note that the resulting graph is not a distribution, i.e., it does not normalize to 1. Rather, each vertex of the curve represents the parameter of a Bernoulli distribution associated to the corresponding cell.

Exercise 3: Occupancy Mapping

Prove that in the occupancy grid mapping framework the occupancy value of a grid cell $P(m_j|x_{1:t}; z_{1:t})$ is independent of the order in which the measurements are integrated.

Let us consider the log-odds update formula and let us recursively unwrap it:

$$\begin{aligned}\ell(m_i|z_{1:t}, x_{1:t}) &= \ell(m_i|z_t, x_t) + \ell(m_i|z_{1:t-1}, x_{1:t-1}) - \ell(m_i) = \\ &= \ell(m_i|z_t, x_t) + \ell(m_i|z_{t-1}, x_{t-1}) + \ell(m_i|z_{1:t-2}, x_{1:t-2}) - 2 \cdot \ell(m_i) = \\ &= \ldots = \sum_{k=1}^{t} \ell(m_i|z_k, x_k) - t \cdot \ell(m_i)\end{aligned}$$

It is clear that $\ell(m_i|z_{1:t}, x_{1:t})$ is simply a sum of the log-odds form of the inverse measurements. Suppose that we exchange a measurement at time i with the measurement at a different time j. Since the sum is a commutative operator we will still obtain the same value of $\ell(m_i|z_{1:t}, x_{1:t})$. Hence, we can repeat exchanging measurements to obtain any order permutation of the measurements without changing the result. Finally, since the log-odds value does not change, neither will the corresponding probability.