3460:460/560 AI, Project 2 – HMM for car tracking

Problem Description: A self-driving car or robo-car is a vehicle that can sense its environment and moving safely with little or no human input [wiki].

For this project, you will design a car agent that uses a simple sensor to locate other cars so your robo-car can drive safely. (Note: the idea was borrowed from Chris Piech at Stanford. https://stanford.edu/~cpiech/cs221/homework/prog/driverlessCar/driverlessCar.html#emission)

Assumptions:

1) The world is a 2D NxN *periodic* grid. All cars reside on the grid points.
2) Your agent (robo-car) senses the environment through a microphone. Assume each of the K other cars moves independently and that the noise in sensor readings for each car is also independent. For simplicity, in this project, we assume there is only one other car (hidden) besides your car agent. (K=1)
3) At each time step t, you know where your agent is, a pair of coordinates representing the actual location (for example $agent_t = (4, 5)$), and a noise estimate of the distance to the hidden car (for example $e_t = 3.5$).
4) Where the car resides at time t ($carX_t$, $carY_t$) is unobserved ("hidden"), but the car moves according to a local conditional distribution *p*(direction|location), for example at (1, 5), the probability the car moves to (2, 5) is 0.6, to (1, 4) is 0.1, to (1,6) is 0.2, to (0, 5) is 0.1. Assuming the transition probability does not change with time, and it varies only with locations.
5) The signal e_t (from the microphone) your robo-car receives at time t is a value of a Gaussian random variable with mean equal to the true distance between your agent and the other car and variance σ^2 (let's assume σ is one-third of the length of the car). (for example, if your agent is at (4, 5) and the car is at (1, 5), the actual distance is 3, but e_t might be 3.2 or 2.9.
6) Initial belief: you will update your belief based on new evidence perceived. But, before any readings, you believe the car could be anywhere: a uniform prior.

Your job is to track the car so you (your robo-car) can drive safely and automatically. To facilitate the grading, you are required to use Jupiter notebook to complete your project.

Part1. Locating a stationary car.

In this part, we assume that the other car is **stationary,** true location of the car ($car^{(T)}X_t$, $car^{(T)}Y_t$) = ($car^{(T)}X_0$, $car^{(T)}Y_0$) for all t. You will implement a function that, upon observing a new distance measurement e_t, updates your current belief (posterior probability) the car is at ($carX_t$, $carY_t$), i.e. update

$$p(C_{t-1} = (carX_{t-1}, carY_{t-1}) \mid E_1=e_1,\ldots, E_{t-1}=e_{t-1})$$

to

$$p(C_t \mid E_1=e_1,\ldots, E_t=e_t) \propto (\sum p(C_{t-1} \mid E_1=e_1,\ldots, E_{t-1}=e_{t-1})\, p(C_t \mid C_{t-1}))\, p(e_t \mid C_t).$$

You are expected to find where the stationary car is as you drive around the car. More specifically, for your project, you will be given all the readings (an example *stationaryCarReading10.csv*), your *output* will be the probability map at time t after seeing the readings from time 1 to time t. The probability map indicates the posterior probabilities of the car at a location, $p(C_t \mid E_1=e_1,\ldots, E_t=e_t)$.

Note. Since the observation is Gaussian, consider using the normal probability density function (pdf). Import norm from scipy.stats and use norm.pdf(eDist, mean, std).

Part II. Inferencing where a moving car is.

Now, let's consider the case where the other car is moving according to transition probabilities $p(C_t \mid C_{t-1}) = p(\text{direction} \mid \text{location})$. A sample transition probabilities file can be found here (*transitionProb10.csv*). And a sample of readings is also given (*movingCarReading10.csv*). In this part, you will implement a function that updates the posterior probability about the location of the hidden car at a current time t: $p(C_t \mid E_1=e_1,\ldots, E_t=e_t)$ based on the readings from time 1 to time t. Your *output* will be a sequence of most possible (carX1, carY1), second most possible (carX2, carY2) and 3rd most possible (carX3, carY3) locations of the hidden car.

What to submit.

You are provided with 2 *.ipynb* templates for Part I and II. Complete the templates. Submit the completed templates.