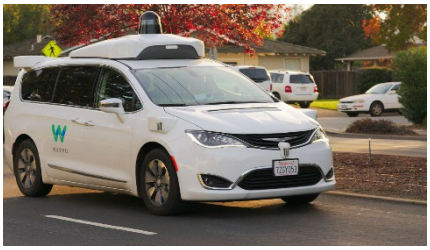
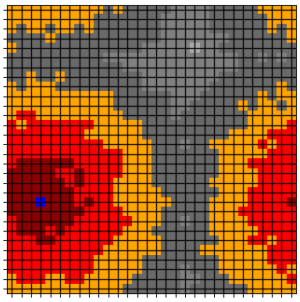
3460:460/560 AI, Project 3 – HMM for car tracking



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 round on a *n*x*n* grid. All cars reside on the grid points. The round world means periodic boundary condition i.e. the position of the car is described as (x%n, y%n) if $x \notin [0,n)$ or $y \notin [0,n)$. For example, if n=10, a car is at (9, 3) at time t and moving east the position of the car at time t+1 will be ((9+1)%10,3) = (0,3). For example, an estimated proximity of the blue hidden car is shown here.
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 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 other 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. (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 3.4 or 2.8.
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 hidden car so you (your robo-car) can drive safely and automatically. To facilitate the grading, you are required to use Python to complete your project.

Part1. Locating a stationary car (emission probabilities).

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. Assume σ is <u>two-third</u> of the length of the car. 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 and collect microphone signals (given). More specifically, for your project, given all the readings, your function will:

(1) output 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, $p(C_t \mid E_1=e_1,\ldots, E_t=e_t)$. Save the map to a file. See the sample output.
(2) Output to the console the most likely location of the hidden car.

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 (transition probabilities).

Now, let's consider the case where the hidden car is moving according to a set of given transition probabilities $p(C_t \mid C_{t-1})$ = p(direction | location). A sample transition probabilities file is attached. In this part, you will implement a function that updates the posterior probability about the location of the hidden car at 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 (given). More specifically, for your project, given all the readings with σ being <u>one-third</u> of the length of the car., your function will:

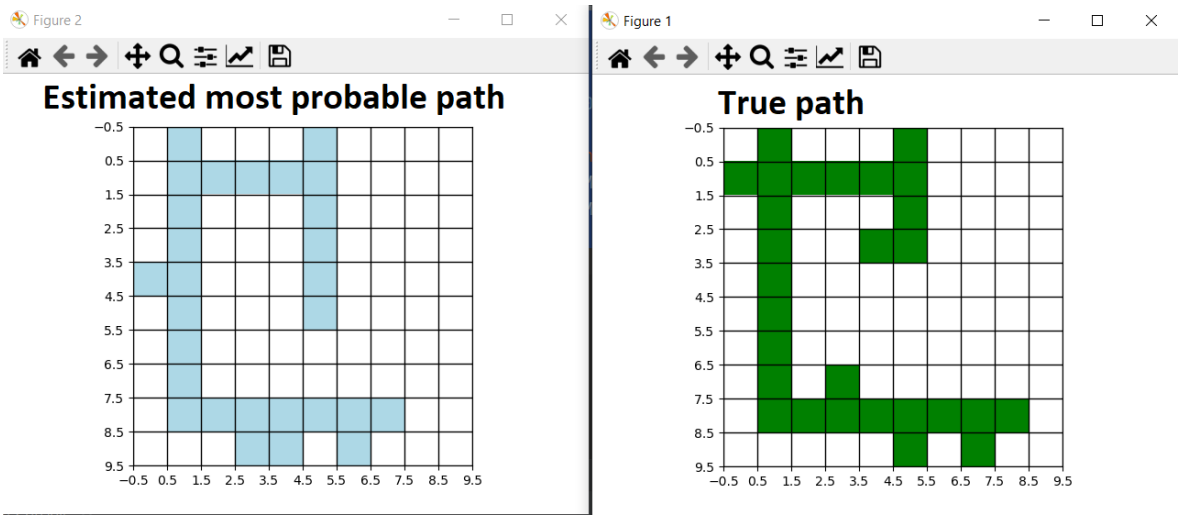
(1) Output to a file the top two most likely locations of the hidden car at any given time from time 1 to time t.
(2) Output to the console the top two most likely locations of the hidden car at time t.

Note: Microphone signal is 1D and noisy. The accuracy of the inference varies depending on the quality of your microphone as well as the transition probability.

Part III. (Graduate students) Do either one of the following two.

(1) Read one of the original papers on HMM. Submit a short (1 page) report summarizing the approach and the findings of the paper.
Thad Starner, Alex Pentland. Real-Time American Sign Language Visual Recognition from Video Using Hidden Markov Models. Master's Thesis, MIT, Feb 1995, Program in Media Arts. https://www.cc.gatech.edu/~thad/p/031_10_SL/real-time-asl-recognition-from%20video-using-hmm-ISCV95.pdf

(2) Inferencing the most probable path the car took (Viterbi algorithm). The car moved from C_0 to C_t, you have the transition probabilities and all the readings till time t. In this part, you'll implement a function that outputs the <u>most probable path the car took</u>. Save the path to a file.



Note: Make sure you understand HMM before coding. Your code should not be very long.

Submission instructions:

1. Submit an electronic copy of the program using project3 dropbox at Brightplace.
2. I believe one file per part should be sufficient for this project. For the convenience of grading, make sure you use the template.
3. Be sure to submit your working solution before the due date! Do not submit non-working programs. The submission time will be used to assess late penalties.