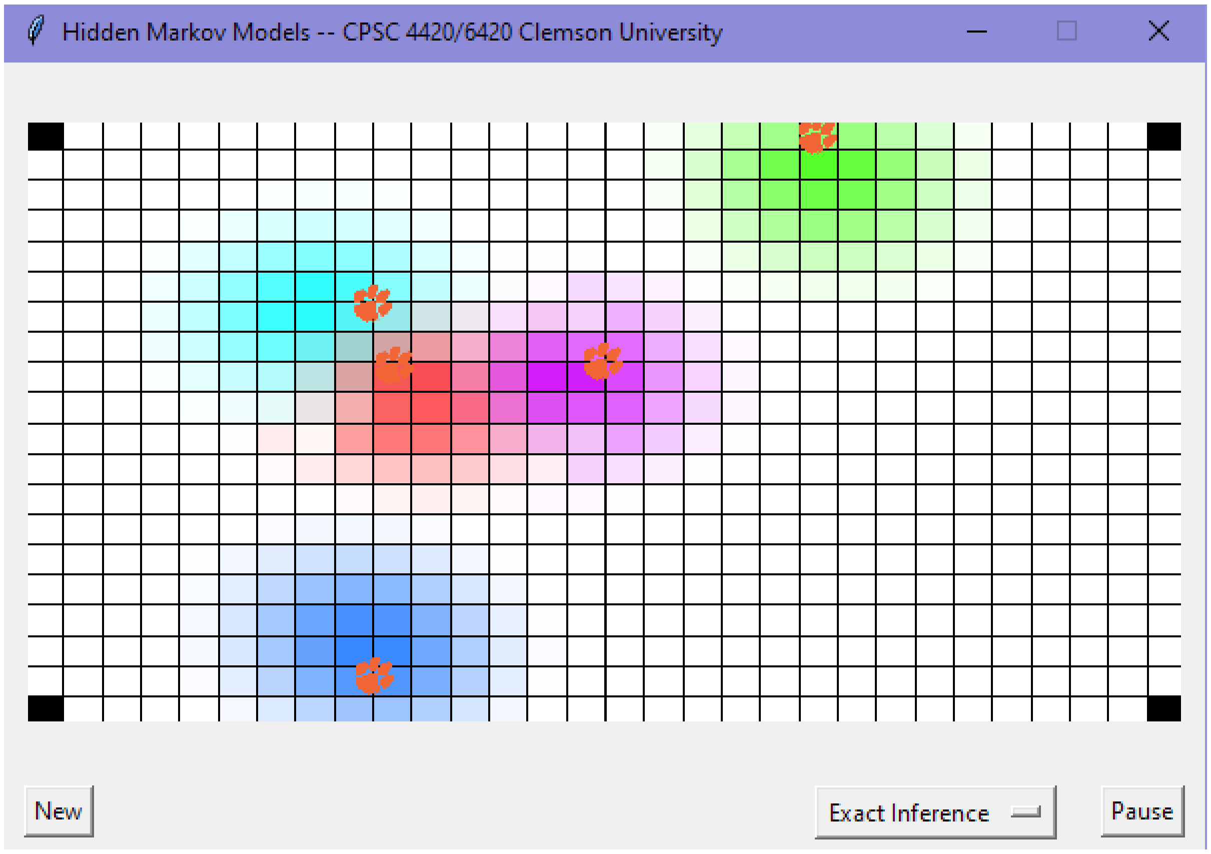
# CPSC 4420/6420: Artificial intelligence

## Assignment 4 [in pairs]

## Deadline: Sunday, November 21, 11:59pm



It's tiger season! To ensure the safety of its tigers, a zoo asked you to install a localization system that can track the activities of the tiger agents. You decided to utilize your knowledge about Hidden Markov Models and implement two inference algorithms: an exact algorithm that computes a full probability distribution over an agent's location, and particle filtering which approximates the same distribution using a set of samples.

**Basic Requirements**

To get started, download the code for this project from Canvas, under Files > Assignments > hmm.zip. You should try to take advantage of Python's math and random libraries. After downloading the zip file, you should be able to display the GUI for the homework by typing the following at the command line:

```
python hmm.py
```

You'll need to modify `hmm.py` in order to implement two HMM inference algorithms. To simplify things, we assume that the zoo is a discretized 2D rectangular grid equipped with a number of landmarks that facilitate the tracking of the agent. At each time step *t*, let $X_t$ denote the actual location of an agent (which is unobserved). We assume there is a local conditional distribution $p(x_t|x_{(t-1)})$that governs the agent's movement. In addition, we receive as measurement$E_t$, a vector of four values that denote the agent’s distance to each of the four landmarks that are installed in the four corners of the environment. However, the distance observations are noisy. For each landmark *i*, the distance-based sensor provides a measurement$E_t^i$, which which is a Gaussian random variable with mean equal to the true distance between the landmark positioned at $L_i$and the agent, and variance $σ^2$, i.e.

$$E_t^i \sim N\left(\left\|L_i - X_t\right\|, \sigma^2\right)$$

(equivalently, the observed distance equals to the true distance plus a zero-mean Gaussian having variance $σ^2$). Assuming each observation is independent, the sensor model follows:

$$p(e_t|x_t) = \prod_i p(e_t^i|x_t),$$

where $p(e_t^i|x_t)$ is the probability density function (PDF) of the Gaussian evaluated at $e_t^i$ assuming that the agent is at $x_t$. Note that the PDF does not return a probability (can exceed 1), but for this assignment, we can get away with treating it like a probability.

Given the above, your task is to implement two tracking algorithms that compute the posterior distribution $P(X_t|E_1=e_1,\ldots,E_t=e_t)$ (your beliefs of where an agent is) and uupdate it for each t=1,2,...

**Question 1 (5 points) Exact Inference**

Modify the `ExactInference` class to compute a full probability distribution of an agent’s location over the discretized 2D world.

First, you should implement the `observe()` function to update the current belief $P(X_t$ | E1=e1, ...,$E_{t-1}=e_{t-1}$) upon observing new distances $E_t=e_t$ as:

$$P(X_t|E_1=e_1,\ldots,E_t=e_t) \propto P(X_t|E_q=e_1,\ldots,E_{t-1}=e_{t-1})\,p(e_t|x_t),$$

where the sensor probabilities $p(e_t|x_t)$ are computed as described above. You should store the current posterior probability as `self.belief` in `ExactInference`.

Next, you should account for passage of time and implement the `timeUpdate()` function to update the belief at current time t: $P(X_t | E_1=e_1,\ldots,E_t=e_t)$ to the next time step t+1 and track the agent's location as:

$$P(X_{t+1}=x_{t+1}|E_1=e_1,\ldots,E_t=e_t) \propto \sum_{x_t} P(X_t=x_t|E_1=e_1,\ldots,E_t=e_t)\,p(x_{t+1}|x_t),$$

where the agent is moving according to transition probabilities $p(x_t | x_{t-1})$. The transition model can be accessed via `self.transition_model(row,col)` along with the grid cell that the agent may reach from the given cell (row, col). As before, the posterior probability should be stored as `self.belief`.

After implementing both functions, you should be able to accurately track any agent using `ExactInference`.

**Question 2 (5 points) Particle Filtering**

Even though exact inference works well, it wastes a lot of time computing probabilities for every available grid cell, even for cells that is unlikely to have a tiger agent. We can address this problem using particle filtering that has complexity linear in the number of particles rather than linear in the number of discrete states.

Modify the `ParticleFilter` class to calculate an approximate belief distribution of an agent's location over the discrete 2D world. The particles have already been initialized randomly. Your task is to implement the `observe()` and `timeUpdate()` functions. Both functions should modify `self.particles` which is a list of 2D cells (row, col). Your code should be able to track agents nearly as effectively as it does with exact inference. Keep in mind though that the resulting belief distribution will look noisier compared to the one obtained by exact inference.

**Important notes:**

1. You can add more tiger agents to the environment by changing the #agents parameters while calling "App(#agents, algs, root)" at the end of the file. If your code works with a single agent, it should be able to scale to any number of agents since they are treated independently.
2. Please read carefully *all* the comments in the `hmm.py` file. It's unlikely you can start implementing the two algorithms without getting a better idea of how the code works and what is expected from each function.
3. We strongly recommend that you watch/review the lectures on HMMs and Particle Filtering before getting started. The questions are short and straightforward—no more than about 70 lines of code in total—but only if your understanding of the probability and inference concepts is clear!

## Submission

Please submit your modified `hmm.py` file to Homework 4 on Canvas. You may work in pairs if you want to. If you do so, please add a comment at the top of `hmm.py` with both of your names. You do not need to submit a copy of this document.

## Getting Help

If you get stuck, please do not hesitate to contact us for help, and stop by during office hours. We also encourage you to post questions and initiate discussions on Canvas. Your colleagues are also there to help you.