ASEN 6519 Probabilistic Algorithms for Aerospace Autonomy
Spring 2019
Exercise 1: HMMs and Basic Inference Algorithms

Out: Tuesday 02/12/2019 (posted on Canvas)

Due: Tuesday 02/26/2019, 5 pm (via Canvas)

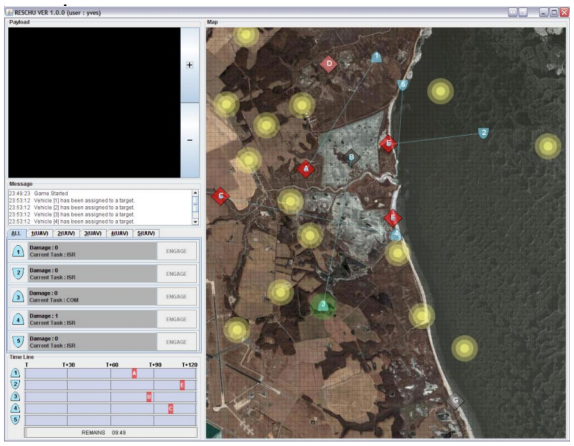
Use whatever programming environment you wish, i.e. Matlab, Python, R, Julia, etc. – but do not simply use toolboxes or built-in functions in place of writing core algorithms for yourself. You need not hand in an overly formal report, but what you turn in should be neatly prepared, and address the questions provided below for each part in a clearly organized manner. Be sure to explain the logic of your solution, and appropriately comment your code (code should be in an appendix). ***You may work in groups of up to 2 students to turn in a single assignment (include both names on the assignment; only one person needs to turn it in). You may discuss your work with other students/groups, but work must be your own.***

Application Background Recall the HMM developed by Boussesmart and Cummings [1] that was presented in lecture, which models the behaviors of human operators supervising a large group of autonomous medium altitude long endurance (MALE) UAVs, high altitude long endurance (HALE) UAVs, and underwater autonomous vehicles (UUVs). The HMM is, for instance, useful for recognizing deviation from expected behaviors. This allows for detection and prediction of the occurrence of potentially critical events. The HMM was learned from a corpus of user data collected in a simulation environment called the Research Environment for Supervisory Control of Heterogeneous Unmanned Vehicles (RESCHU) Simulator. In RESCHU, the vehicles autonomously perform complementary surveillance tasks with the ultimate goal of locating specific objects of interest in urban coastal and inland settings. UAVs can be of two types: one that provides high-level sensor coverage (HALEs, akin to Global Hawk UAVs), while the other provides more low-level target surveillance and video gathering (MALEs, similar to Predator UAVs). In contrast, UUVs are all of the same type and perform low-level surveillance tasks similar to the MALEs. Thus, the single operator controls a heterogeneous team of vehicles which may consist of up to three different types of platforms, each with different characteristics. The interface used to conduct the data collection study is shown in Figure 1.

The key task characteristics for the RESCHU simulation were:

- HALEs processed emergent non-identified targets, i.e. new possible target pickups;
- UUVs and MALEs performed automatic visual target acquisition and identification of HALE target pickups;
- once UUVs and MALEs processed targets, an automated planner assigned each vehicle a new target at random; since these assignments could be sub-optimal, the operator could intervene to correct the assignments;

[1] Y. Boussesmart and M. Cummings, 'Behavioral Recognition and Prediction of an Operator Supervising Multiple Heterogeneous Unmanned Vehicles', AIAA InfoTech, 2009



```
23:49:23  Game Started
23:53:12  Vehicle [1] has been assigned to a target.
23:53:12  Vehicle [2] has been assigned to a target.
23:53:12  Vehicle [3] has been assigned to a target.
23:53:12  Vehicle [4] has been assigned to a target.
```

```
1  Damage : 0  Current Task : ISR
2  Damage : 0  Current Task : ISR
3  Damage : 0  Current Task : COM
4  Damage : 1  Current Task : ISR
5  Damage : 0  Current Task : ISR
REMAINS  09:49
```

Figure 1: RESCHU operator interface (image taken from original paper).

- 'pop up' threats appeared dynamically on the surveillance map; to avoid damaging the UAVs and UUVs by entering these areas, the operator could optimize the vehicle path by assigning a different goal or adding waypoints to its existing path to avoid threat areas;

- operators had to maximize points in each mission by avoiding damage to the vehicles and localizing/ID'ing all hostile targets.

The Markov chain portion of the model identified in the study defines discrete event stochastic transition process among the following four discrete operator command 'modes', which are the realizations of discrete random variable x_k at tasking event k:

1. Planning-Engaging UUVs & MALEs

2. Planning-Engaging HALEs

3. Monitoring HALEs

4. Monitoring UUVs & MALEs

The observation sequence for this model consists of observed tasking inputs at each time step, which are recorded from the operator's user interface (UI) and post-processed into a feature vector y_k. The observation feature space consists of a 'tasking grammar' that describes combined vehicle selections and UI manipulation events at any given event k. Table 1 shows the mutually exclusive elements of the tasking grammar and y_k observation indices assigned to each element.

Vehicle Selected	UI Operation				
	select sidebar	select map	set waypoint	set goal	visual task
UUV	$y_k = 1$	$y_k = 2$	$y_k = 3$	$y_k = 4$	$y_k = 5$
MALE	$y_k = 6$	$y_k = 7$	$y_k = 8$	$y_k = 9$	$y_k = 10$
HALE	$y_k = 11$	$y_k = 12$	$y_k = 13$	$y_k = 14$	$y_k = 15$

Table 1: Tasking grammar and y_k element indices.

Questions *(complete all 3)*

The learned HMM CPTs $P(x_k|x_{k-1})$, $P(y_k|x_k)$ and initial state distribution $P(x_0)$ are in the posted file 'nominal_hmm_params.txt'.

1. Implement the forward-backward algorithm for the posted 'nominal_hmm_short_log.txt' observation sequence. Report the posterior probabilities $P(x_k|y_{1:T})$ for each event step k using a suitably sized table, and also report the data log-likelihood $\log p(y_{1:T})$ for all T observations. Use the resulting posteriors to classify the most likely x_k values for each $k = 1 : T$ (plot these as a time trace).

2. Repeat problem 1 using likelihood weighted approximate inference at sample sizes 100, 1000 and 10,000. Explain precisely how you constructed your posterior probability estimates from the Monte Carlo samples, and comment on the accuracy of the results compared to the forward-backward algorithm.

3. Implement the forward-backward algorithm for the posted 'nominal_hmm_long_log.txt' sequence; report the posterior probabilities for only the first 5 and last 5 steps in the sequence, along with the data log-likelihood $p(y_{1:T})$ (where the first time step refers to the point where observations y_k start). Use the resulting posteriors for all steps to classify the most likely x_k values for each $k = 1 : T$ (plot these as a single time trace). **Hint:** you may have to modify the forward-backward algorithm to account for very small probabilities and avoid numerical instabilities that result from using long data sequences. See the posted report by Tobias Mann for guidance on how to do this (this report uses the notation in the tutorial by Rabiner, also posted).

Extra Credit Challenge Questions

Attempt as many of these as you like, in any order you wish – **partial/full credit given only if you turn in a *complete* solution for any particular problem (i.e. no half-finished/incomplete attempts).**

C1. Implement and apply the Viterbi algorithm described in the HMM tutorial by Rabiner to arrive at maximum a posteriori (MAP) state trajectory estimates for the short and long data streams in part A. Compare the results to maximizing the marginal posterior of each $p(x_k|y_{1:T})$ produced by the full forward-backward algorithm.

C2. Do problem 2 again, this time using the bootstrap particle filter (see the Arulumpalam tutorial posted for more details). For the resampling step, use an effective sampling size of $0.5N_s$, where N_s is the number of particles. Compare your results to the exact 'forward step only' results, the likelihood weighting results, and the full forward-backward posteriors (be sure to explain why the particle filter will not produce the same results as the full forward-backward algorithm no matter how large N_s gets). What seems to be the smallest sample size N_s that produces acceptable results?