Learning Techniques for HRI

Scribes: *Isabel Rayas and Setareh Nasihati Gilani*

11 October 2018

1 Introduction

So far we have have used models in our approaches for robot planning. But how do we derive the parameters for our model? We should learn these from somewhere. We categorize learning into two categories:

1. **Learning from demonstration (Model free learning)**
   In these methods, robot learns a policy - a mapping from states to actions. The robot doesn't care about the system dynamics.

2. **Learning from experience (Model based learning)**
   In these methods, robot tries to learn the domain/reward functions or in other words the dynamics of the system.

Table 1 shows what should we learn in each approach, how we learn them and when we learn them along with advantages/disadvantages of each approach.

	Model free	Model Based
What?	State: Discrete/ Continues Action: Discrete/ Continues	Transition: Discrete/ Continues Reward: Continues[1]
How?	Demonstration/Teleoperation	Transition: Supervised/ Unsupervised Reward: Inverse Optimal Control[2]
When?	Offline/ Online Learning	Offline/Online Learning
Advantages	No design bias Fast execution Doesn't require domain knowledge	Reward function generalization Easier to learn human model Safety Guarantees Understanding of the world
Disadvantages	Lots of data	

Table 1: Model-Free vs Model Based Learning

We will focus on deriving the Transition function in Model based approaches. We will first discuss this in the case of fully observable systems in which the transition function can be derived using Supervised learning and then we will move onto partially observable systems in which we will use unsupervised learning approaches to gain the transition function.

2 Supervised Learning Approach

This method is used when the system is fully observable. Our task is learning the human model based on the observations. We use the Maximum likelihood method to derive the parameters. Consider the example below:

- θ is a binary variable showing whether humans trust the robot or not.
- h_θ is our hypothesis of θ
- N: number of users, t of them trust the system, $nt = N - t$ do not trust the robot

We have the following for the $P(h_\theta/O)$ according to the bayes rule:

$$P(h_\theta/O) \propto P(O/h_\theta)P(h_\theta) \tag{1}$$

We want to find the θ that maximizes $P(O/h_\theta)$:

$$P(O/h_\theta) = \prod_{j=1}^{N} P(O_j/h_\theta) = \theta^t(1-\theta)^{N-t}$$

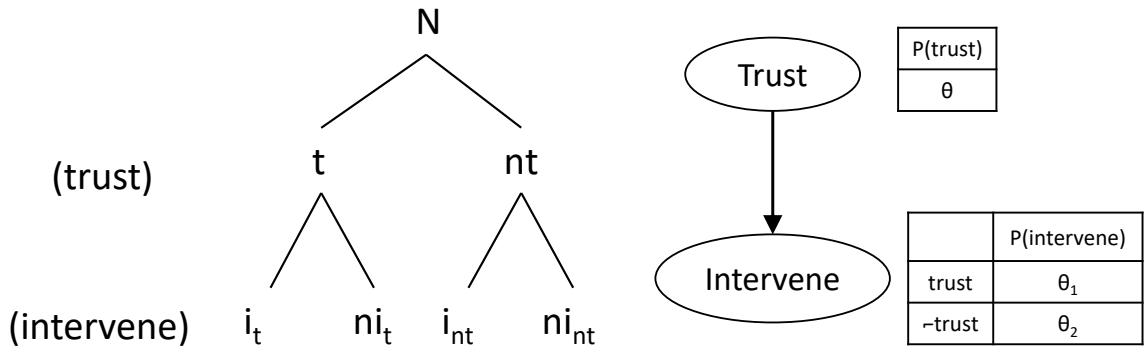
Maximizing the above quantity is the same as maximizing its log function:

$$L = \log P(O/h_\theta) = t\log\theta + (N-t)\log(1-\theta)$$

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = \frac{t}{\theta} - \frac{N-t}{1-\theta} = 0 \Rightarrow \boxed{\theta = \frac{t}{N}}$$

Now imagine our usual example of "human having/not having trust in robot and possible intervening during the robot's action". Figure 1(a) shows the number of observations for each condition from the users and Figure 1(b) shows the dynamics and dependencies of the system.

[1]Output of the reward function is continues; however, input can be discrete or continues
[2]Inverse Reinforcement learning



(a) Observations from humans (b) System Dynamics

Figure 1: Summarized decision tree based on system variables

$$L = P(O/h_\theta, \theta_1, \theta_2) = \prod_{j=1} NP(O_j/h_\theta, \theta_1, \theta_2)$$

$$= \prod^{i_t} P(intervene, trust)/h_\theta, \theta_1, \theta_2) * \prod^{ni_t} P(\neg intervene, trust)/h_\theta, \theta_1, \theta_2) *$$

$$\prod^{i_{nt}} P(intervene, \neg trust)/h_\theta, \theta_1, \theta_2) * \prod^{ni_{nt}} P(\neg intervene, \neg trust)/h_\theta, \theta_1, \theta_2)$$

$$= \theta_1^{it}\theta^{it} * (1-\theta_1)^{ni_t}\theta^{ni_t} * \theta_2^{i_{nt}}(1-\theta)^{i_{nt}} * (1-\theta_2)^{ni_{nt}}(1-\theta)^{ni_{nt}}$$

$$= \theta_1^{it}\theta^{t}(1-\theta_1)^{ni_t}\theta_2^{i_{nt}}(1-\theta)^{nt}(1-\theta_2)^{ni_{nt}}$$

Now we will set the derivative of log likelihood to zero to derive the best values (which maximizes L) of θ, θ_1 and θ_2:

$$l = \log L = i_t \log\theta_1 + t\log\theta + ni_t \log(1-\theta_1) + i_{nt}\log\theta_2 + nt\log(1-\theta) + ni_{nt}\log(1-\theta_2)$$

$$\frac{\mathrm{d}l}{\mathrm{d}\theta} = \frac{t}{\theta} - \frac{nt}{1-\theta} = 0 \Rightarrow \boxed{\theta = \frac{t}{N}}$$

$$\frac{\mathrm{d}l}{\mathrm{d}\theta_1} = \frac{i_t}{\theta_1} - \frac{ni_t}{1-\theta_1} = 0 \Rightarrow \boxed{\theta_1 = \frac{i_t}{t}}$$

$$\frac{\mathrm{d}l}{\mathrm{d}\theta_2} = \frac{i_{nt}}{\theta_2} - \frac{ni_{nt}}{1-\theta_2} = 0 \Rightarrow \boxed{\theta_2 = \frac{i_{nt}}{nt}}$$

Question: What is wrong with this method (Maximum likelihood estimation)?
Answer: It cannot handle insufficient data. Imagine we only have one human subject who trusts the robot. In this case θ will be 1. A lot of variables would be 0 in this case but we know that this is not realistic.
Solution: One way to account for insufficient data is to assume the prior on θ is not uniform in Equation 1 and not disregard it (which we did here). In this way, we will account for the fact that we may not have enough observations to estimate all the parameters.
The prior distribution will have some parameters which are called hyper-parameters. (The reason is that they parameterize the distribution of θ where θ is already a parameter). For certain prior distributions, this hyper-paramters can be treated as virtual counts. They represent the confidence we have or our initial estimate of the parameters which will be later updated when we observe the actual data.

3 Unsupervised Learning Approach

Now we will use an Expectation Maximization algorithm as an unsupervised learning approach for an HMM. Here, θ represents the parameters of the model, in this case, the transition (T) and observation (M) matrices of the HMM. x is the hidden state, o is the sequence of observations, and we want to infer the sequence of states that corresponds to the observations that we have seen in order to learn the model.

$$P(o|O) = \sum_x P(o, x|\theta)$$

We take the log of both sides to get the log likelihood:

$$\log P(o|O) = \log \sum_x P(o, x|\theta)$$

We can assume that x is distributed based on some probability distribution Q.

$$\log P = \log \sum_x \frac{Q(x)P(o, x|\theta)}{Q(x)} \tag{2}$$

Above we have just multiplied by 1. However, it turns out that this is greater than or equal to putting the log inside the sum; this is called **Jensen's Inequality** and is true because the logarithm is a concave function. This gives us a lower bound on our observations.

$$\log P \geq \sum_x Q(x) \log \frac{P(o, x|\theta)}{Q(x)} \tag{3}$$

Within Eq 2, the summation is the expectation over x where x is drawn from $Q(x)$. Recall that since we assume $Q(x)$ is a distribution, the sum over all values of x, weighted by $Q(x)$ is the expectated value if x is discrete.

In other words, in Eq 2, we are taking the logarithm of the expected value:

$$\sum_x \frac{Q(x)P(o,x|\theta)}{Q(x)} = \mathbb{E}_{x-Q(x)} \frac{P(o,x|\theta)}{Q(x)}$$

and in Eq 3, we are taking the expected value of the logarithm:

$$\sum_x Q(x) \log \frac{P(o,x|\theta)}{Q(x)} = \mathbb{E}_{x-Q(x)} \log \frac{P(o,x|\theta)}{Q(x)}$$

By maximizing the expectation of the logarithm, which is our lower bound, we can maximize the left side of the equation as well.

First we look at how we pick $Q(x)$. We want to pick $Q(x)$ such that the lower bound is as tight as possible; that is, we want to decrease the difference in the inequality. When does this difference become zero? When $Q(x)$ is proportional to $P(o,x|\theta)$. But $Q(x)$ is some arbitrary probability distribution over x, so it must sum to 1:

$$\sum_x Q = 1$$

and we can write it as:

$$Q(x) = \frac{P(o,x|\theta)}{\sum_x P(o,x|\theta)} = \frac{P(o,x|\theta)}{P(o|\theta)} \tag{4}$$

This is called the E-step (estimation step).

Now we also have to pick the parameters θ to maximize this quantity. This is called the M-step (maximizing step).

$$\theta = argmax_\theta \sum_x Q(x) \log \frac{P(o,x|\theta)}{Q(x)}$$

Essentially, we're doing gradient ascent to make the lower bound as tight as possible in two steps: first we pick Q, then we pick θ.

Now, does this actually converge? We denote the log likelihood as l. Then, we have:

$$l(\theta) = \sum_x Q(x) \log \frac{P(o,x|\theta)}{Q(x)}$$

And for iteration $i+1$, we have:

$$l(\theta_{i+1}) = \sum_x Q(x) \log \frac{P(o,x|\theta_{i+1})}{Q(x)}$$

$$\geq \sum_x Q(x) \log \frac{P(o, x|\theta_i)}{Q(x)}$$
$$= l(\theta_i)$$

For the Q that we have chosen, we know that Eq 4 holds because we specifically chose Q such that we make the inequality tight. For the next iteration, we know that l cannot be smaller than the previous iteration because we have chosen θ to maximize the lower bound. This holds generally for any θ. Therefore, log likelihood l is always monotonically increasing.

This is, however, also a shortcoming to this method. We will not necessarily reach the global maximum.

The HMM case.

In the HMM case, the theta we optimized are transition and observation matrices. What is interesting here is that we can compute the joint probability of our observations and hidden state sequence using these two matrices.

Here, our two steps become:

1. E-step. Given the transition and observation matrices we have, compute expectations over observation and state sequences:

$$Q(x) \ \propto \ P(o, x|T, M)$$

2. M-step. Given the expectations over observation and state sequences, update the transition and observation matrices:

$$T, M = argmax_{T,M} \sum_x Q(x) \log \frac{\prod_{t=1}^{T} M(O_t|x_t) \prod_{t=1}^{T} T(x_t|x_{t-1}) \prod(x_0)}{Q(x)}$$

within which

$$P(o, x|M, T) = \prod_{t=1}^{T} M(O_t|x_t) \prod_{t=1}^{T} T(x_t|x_{t-1}).$$

Since this is for an HMM, the expectations can be computed using the forward and backward pass algorithms from previous lectures. Once we do that, we see that the transition matrix from state q to state s is

$$T(s|q) = \frac{\mathbb{E}[\#\ transitions\ from\ q\ to\ s]}{\mathbb{E}[\#\ transitions\ from\ q]}$$

and the observation matrix for state s is

$$M(o_i|s) = \frac{\mathbb{E}[\#\ times\ in\ s\ and\ observed\ o_i]}{\mathbb{E}[\#\ times\ in\ s]}$$

One final note is that this all assumes we know the structure of the problem (in this case, an HMM) and we just want to find out the parameters, in this case the T and M matrices. In other problems, the structure itself may be unknown and must also be inferred.