EM Algorithm for Labeled IRL

James MacGlashan

1 Introduction

In this document, I explored the EM algorithm approach to solving our labeled IRL problem. As a review, I first include the description of our model that we've previously defined and then provide the EM algorithm approach.

2 Probability Model

The motivation for this model is that we consider the the final label (L) that a user gives a trajectory of size N to be some random function of what they thought about each of the action selections (A) exhibited in the trajectory. However these step-wise evaluations (X) are unobserved in the data.

We model the probability that an action is evaluated as good or not as proportional to its selection probability according a softmax policy computed for the reward function with parameters $\boldsymbol{\theta}$. Specifically:

$$\Pr(x_i = +1|s, a, \theta) = \pi(s, a|\boldsymbol{\theta}) \tag{1}$$

$$\Pr(x_i = -1|s, a, \theta) = 1 - \pi(s, a|\boldsymbol{\theta}), \tag{2}$$

where $\pi(s, a|\boldsymbol{\theta})$ is the softmax policy over Q-values computed for the reward function parameterized by $\boldsymbol{\theta}$:

$$\pi(s, a|\boldsymbol{\theta}) = \frac{e^{\beta Q(s,a|\boldsymbol{\theta})}}{\sum_{a'} e^{\beta Q(s,a'|\boldsymbol{\theta})}}, \tag{3}$$

β is a selectable parameter, and $Q(s, a|\boldsymbol{\theta})$ is the Q-value computed for the reward function parameterized by $\boldsymbol{\theta}$.

For the probability distribution of L, given the sequence of N step-wise labels, we would like a distribution that has the property that as more step-wise labels are positive, the probability of a positive trajectory label increases (and vice versa). Although there are many possible distributions that satisfy this property, for concreteness, we choose the sigmoid function. That is,

$$\Pr(L = +1|X_1, ..., X_n) = \frac{1}{1 + e^{-\phi \sum_i^N X_i}} \tag{4}$$

$$\Pr(L = -1|X_1, ..., X_n) = 1 - \Pr(L = +1|X_1, ..., X_n), \tag{5}$$

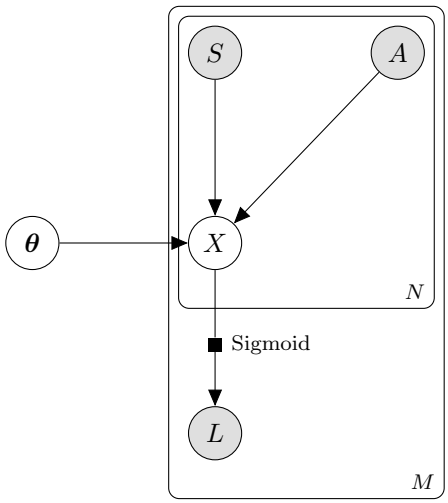

Figure 1: Plate Diagram of our Probability Model

where ϕ is a selectable parameter that tunes how quickly of a majority of step-wise labels increases/decreases the probability of trajectory assignment. For example, when $\phi = 0$, trajectory labels are assigned uniformly randomly independently of step-wise labels. As $\phi \to \infty$, the sigmoid converges to a step function in which a trajectory containing even one more positive step-wise label than negative step-wise labels will deterministically cause a positive trajectory label (and vice versa).

3 EM Algorithm

The problem with estimating the $\boldsymbol{\theta}$ parameters of our reward function is that we have a latent variable vector X (or rather, some of the elements of the X vector are latent, and some may be observed), which prevents us from easily computing the likelihood of the model and maximizing parameters for it. The EM approach to solving this problem is to first choose values for $\boldsymbol{\theta}$; then choose a new $\boldsymbol{\theta}$ that maximizes the expected value of the log likelihood function where the distribution of the expectation is the probability distribution of latent variables (the Xs in our case) given the observations available and previous $\boldsymbol{\theta}$ choice; and then repeating this process. The maximization process can be performed using gradient ascent.

To formalize this process for our problem, first note that the likelihood our parameters (and state-action sequence) given an $\boldsymbol{x}$ vector and label l is

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta} | l, \boldsymbol{x}) = \Pr(l|\boldsymbol{x}) \prod_i \Pr(x_i | s_i, a_i, \boldsymbol{\theta}) \tag{6}$$

and the log likelihood is

$$\log \mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta} | l, \boldsymbol{x}) = \log \Pr(l|\boldsymbol{x}) + \sum_i \log \Pr(x_i | s_i, a_i, \boldsymbol{\theta}). \tag{7}$$

Additionally, the gradient of the log likelihood is,

$$\nabla_{\boldsymbol{\theta}} \log \mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta} | l, \boldsymbol{x}) = \sum_i \frac{\nabla_{\boldsymbol{\theta}} \Pr(x_i | s_i, a_i, \boldsymbol{\theta})}{\Pr(x_i | s_i, a_i, \boldsymbol{\theta})}. \tag{8}$$

To simplify the EM algorithm description, I will introduce the notation $\boldsymbol{x}_k$ to indicate the subset of observed elements in an $\boldsymbol{x}$ vector, and $\boldsymbol{x}_u$ to represent a possible assignment to the subset of the unobserved values of an $\boldsymbol{x}$ vector. Using this notation, the expected value of the log likelihood under some candidate parameter $\boldsymbol{\theta}'$ for missing X elements distributed according to $\boldsymbol{\theta}$ is

$$\begin{aligned} &E_{\boldsymbol{x}_u \sim \Pr(\boldsymbol{x}_u | l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \left[\log \mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}' | l, \boldsymbol{x}) \right] \\ &\qquad = \sum_{\boldsymbol{x}_u} \Pr(\boldsymbol{x}_u | l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) \log \mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}' | l, \boldsymbol{x}) \end{aligned}$$

Given that, the corresponding EM algorithm operating on a single trajectory is as follows (it is generalized to many trajectories by simply summing over each trajectory).

Algorithm 1 Labeled-IRL EM Algorithm

Require: initial $\boldsymbol{\theta}_0$, and data $\boldsymbol{s}$, $\boldsymbol{a}$, $\boldsymbol{x}_k$, l
  for $t = 0$ to K **do**
    $\boldsymbol{\theta}_{t+1} \leftarrow \arg\max_{\boldsymbol{\theta}'} \sum_{\boldsymbol{x}_u} \Pr(\boldsymbol{x}_u | l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_t) \log \mathcal{L}(\boldsymbol{\theta}', \boldsymbol{s}, \boldsymbol{a} \mid l, \boldsymbol{x}_k, \boldsymbol{x}_u)$
  end for

To compute the expected value, we need to enumerate each of the possible assignments to the unknown $\boldsymbol{x}$ elements and compute the probability of them given the observed data and model parameters $\boldsymbol{\theta}$. This probability is computed as

$$\begin{aligned} \Pr(\boldsymbol{x}_u | l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) &= \frac{\Pr(l | \boldsymbol{x}_k, \boldsymbol{x}_u) \Pr(\boldsymbol{x}_u | \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) \Pr(\boldsymbol{x}_k | \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) \Pr(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(l | \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) \Pr(\boldsymbol{x}_k | \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) \Pr(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \\ &= \frac{\Pr(l | \boldsymbol{x}_k, \boldsymbol{x}_u) \Pr(\boldsymbol{x}_u | \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(l | \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \\ &= \frac{\Pr(l | \boldsymbol{x}_k, \boldsymbol{x}_u) \prod_i \Pr(\boldsymbol{x}_{u,i} | s_i, a_i, \boldsymbol{\theta})}{\Pr(l | \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}. \end{aligned}$$

A straight forward computation of $\Pr(l | \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$ requires marginalizing over all possible assignment to the unknown X elements; however, it can be computed efficiently using dynamic programming because the probability of a label is a sigmoid function that operates on the sum of the x elements. I will not cover the dynamic programming computation of it here.

Unfortunately, even with an efficient means to compute $\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$, when the number of unknown X variables is large, the number of $\boldsymbol{x}_u$ assignments enumerated in the expectation's outer sum grows exponentially, and the product series over each of unknown element probabilities in the above equation ($\prod_i \Pr(\boldsymbol{x}_{u,i}|s_i, a_i, \boldsymbol{\theta})$) can have underflow issues. A resolution to this problem is to estimate the expectation with sampling. Monte Carlo sampling is unfortunately intractable because it is not easy to sample from $\Pr(\boldsymbol{x}_u|l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$; moreover, it would not address the underflow issue in the product series. However, it is easy to sample from $\Pr(\boldsymbol{x}_u|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$ (removing the conditioning on the label), which we can use in importance sampling. With importance sampling, we can replace the expectation computation with the sample-estimate

$$\frac{1}{C}\sum_{j}^{C} \frac{\Pr(\boldsymbol{x}_u^j|l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(\boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \log \mathcal{L}(l, \boldsymbol{x}_k, \boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}). \tag{9}$$

where $\boldsymbol{x}_u^j$ is sample from the distribution $\Pr(\boldsymbol{x}_u|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$.

Note that this simplifies a bit further too:

$$\begin{aligned}\frac{\Pr(\boldsymbol{x}_u^j|l, \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(\boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} &= \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u)\Pr(\boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \frac{1}{\Pr(\boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \\ &= \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u)}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}\end{aligned}$$

Consequently, we have removed the product series from the expectation weight, thereby avoiding underflow issues. Also, as noted previously, the $\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$ term can be computed efficiently with dynamic programming.

Now we can write a tractable EM algorithm where we can compute the maximization using gradient ascent.

Algorithm 2 Labeled-IRL Approximate EM Gradient Ascent Algorithm

Require: initial $\boldsymbol{\theta}_0$; data $\boldsymbol{s}$, $\boldsymbol{a}$, $\boldsymbol{x}_k$, l; and learning rate α
  for $t = 0$ to K **do**
    draw $j = 1$ to C samples of $\boldsymbol{x}_u^j \sim \Pr(\boldsymbol{x}_u|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_t)$
    for $j = 1$ to C **do**
      $w_j \leftarrow \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u)}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_t)}$ ▷ Expectation step
    end for
    $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}_t$
    for 1 to M **do** ▷ Gradient ascent maximization loop
      $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}' + \alpha \frac{1}{C} \sum_j^C w_j \sum_{x_i \in \boldsymbol{x}_k \cup \boldsymbol{x}_u^j} \frac{\nabla_{\boldsymbol{\theta}'} \Pr(x_i|s_i, a_i, \boldsymbol{\theta}')}{\Pr(x_i|s_i, a_i, \boldsymbol{\theta}')}$
    end for
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}'$
  end for

4 DP Algorithm For Label Probability

One of the terms that must be computed for the EM algorithm weight is $\Pr(l \mid \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$. A straightforward computation of this value would involve marginalizing over $\boldsymbol{x}_u$, which grows exponentially large as the number of unknown feedbacks increases. However, by exploiting the fact that $\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$ is a function of the sum of the feedback values, the marginalization can be reduced to a summation that iterates over a number of values that is a linear function of the number of unobserved feedbacks. To demonstrate, first note that $\Pr(l \mid \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$, where all feedback values are known, is defined as

$$\Pr(l \mid \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) = S\left(\sum_{x \in \boldsymbol{x}} x\right), \tag{10}$$

where S is the sigmoid function. Therefore, $\Pr(l \mid \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$, where some of the feedbacks are unobserved, can expressed by marginalizing over the possible *sums* of the unknown feedbacks, rather than marginalizing over all possible feedback assignments:

$$\Pr(l \mid \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) = \sum_{\tau=-|\boldsymbol{x}_u|}^{|\boldsymbol{x}_u|} S\left(\tau + \sum_{x \in \boldsymbol{x}_k} x\right) \Pr(\tau \mid \boldsymbol{s}_u, \boldsymbol{a}_u, \boldsymbol{\theta}), \tag{11}$$

where $\Pr(\tau \mid \boldsymbol{s}_u, \boldsymbol{a}_u, \boldsymbol{\theta})$ is the probability that the unknown feedbacks will sum to τ, given the states and actions for the unknown feedbacks, and the model parameters $\boldsymbol{\theta}$.

Computing this simpler marginal distribution requires computing $\Pr(\tau \mid \boldsymbol{s}_u, \boldsymbol{a}_u, \boldsymbol{\theta})$: the probability distribution of the sum of possible feedback values. Fortunately, this probability can expressed recursively and computed efficiently with a dynamic programming algorithm.

To describe this recursive relationship, note that if we knew the probability distribution of the sum of feedback values for all unknown feedbacks except the last of them, then we could compute the probability that *all* of the feedbacks would some to some value τ as the probability that the remaining feedbacks sum to $\tau - 1$ and the last feedback is a $+1$ *or* the remaining feedbacks sum to $\tau + 1$ and the last feedback is a -1. That is,

$$\begin{aligned}\Pr(\tau_n = \tau \mid \boldsymbol{s}_n, \boldsymbol{a}_n, \boldsymbol{\theta}) = &\Pr(x_n = +1 \mid s_n, a_n) \Pr(\tau_{n-1} = \tau_n - 1 \mid \boldsymbol{s}_{n-1}, \boldsymbol{a}_{n-1}, \boldsymbol{\theta}) \\ &+ \Pr(x_n = -1 \mid s_n, a_n) \Pr(\tau_{n-1} = \tau_n + 1 \mid \boldsymbol{s}_{n-1}, \boldsymbol{a}_{n-1}, \boldsymbol{\theta}),\end{aligned} \tag{12}$$

where n is the number unobserved feedbacks, τ_i is the random variable specifying the sum of the first i unobserved feedbacks; s_i and a_i represents the ith state and action that are associated with unobserved feedback x_i; $\boldsymbol{s}_i$, $\boldsymbol{a}_i$ represents the set of the first i state and actions associated with the unobserved feedbacks; and finally, where the probability of the first unobserved feedback summing to

τ is defined as the probability that the first feedback takes that value:

$$\Pr(\tau_1 = \tau \mid \boldsymbol{s}_n, \boldsymbol{a}_n, \boldsymbol{\theta}) = \begin{cases} \Pr(x_n = +1 \mid s_n, a_n) & \text{if } \tau_1 = 1 \\ \Pr(x_n = -1 \mid s_n, a_n) & \text{if } \tau_1 = -1 \\ 0 & \text{otherwise} \end{cases}. \tag{13}$$

We can compute this recursive probability distribution using dynamic programming in which we build a $2n + 1 \times n$ matrix, where cell i, j specifies the value for $\Pr(\tau_j = i \mid \boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta})$. The values for the matrix are then filled out column by column, from $j = 1$ to n. Consequently, using this DP algorithm, we can compute the $\Pr(l \mid \boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})$ term required for our EM weight in only $O(|\boldsymbol{x_u}|^2)$ time, rather than $O(2^{|\boldsymbol{x_u}|})$.

5 Other Importance Sampling Distributions

One of the potential problems with the importance sampling distributions being drawn from the policy under the theta values that is as the agent optimizes for theta, it will produce more detemrinistic action selection, leading to progressively more biased samples that may not adequately cover the space. Indeed, initial experiments with the current distribution show that in some cases, the log likelihood starts getting worse, whereas when using the full marginalization, it consistently improves.

To comabt this issue, we should explore other sampling distributions that better span the space. One potential issue is that with a different sampling distribution, we must include the product series of the individual feedback probabilities, which could produce underflow. However, if we simply choose a slightly nosier sampling feedback distribution than the feedback distribution under the current parameters, then we might avoid underflow problems.

That is, let a vector for unknown feedbacks $\boldsymbol{x}_u$ be drawn from some sampling distribution f that indepdently draws feedbacks as a function of each state-action pair and the current parameters θ:

$$f(\boldsymbol{x}_u \mid \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}) = \prod_i f_i(x_{u,i} \mid s_i, a_i, \boldsymbol{\theta}). \tag{14}$$

Then using f as the importance sampling distribution, we have the following importance sampling weights for each sample x_u^j:

$$\begin{aligned} w_j &= \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u^j) \Pr(\boldsymbol{x}_u^j|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \frac{1}{f(\boldsymbol{x}_u^j \mid \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \\ &= \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u^j) \prod_i \Pr(x_{u,i}^j|s_i, a_i, \boldsymbol{\theta})}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \frac{1}{\prod_i f_i(x_{u,i}^j \mid s_i, a_i, \boldsymbol{\theta})} \\ &= \frac{\Pr(l|\boldsymbol{x}_k, \boldsymbol{x}_u^j)}{\Pr(l|\boldsymbol{x}_k, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta})} \prod_i \frac{\Pr(x_{u,i}^j|s_i, a_i, \boldsymbol{\theta})}{f_i(x_{u,i}^j \mid s_i, a_i, \boldsymbol{\theta})} \end{aligned} \tag{15}$$

Therefore, as long as $\frac{\Pr(x^j_{u,i}|s_i,a_i,\boldsymbol{\theta})}{f_i(x^j_{u,i}|s_i,a_i,\boldsymbol{\theta})}$ is reasonably near 1, there shouldn't be underflow issues. This condition may be held if the single feedback distribution f_i, is also based on a softmax policy distribution of the current weights theta, but is more uniform (e.g., a small β value.) That is,

$$f_i(x_i \mid s_i, a_i, \boldsymbol{\theta}) = \begin{cases} \frac{e^{\hat{\beta}Q(s,a|\boldsymbol{\theta})}}{\sum_{a'} e^{\hat{\beta}Q(s,a'|\boldsymbol{\theta})}} & \text{if } x_i = 1 \\ 1 - \frac{e^{\hat{\beta}Q(s,a|\boldsymbol{\theta})}}{\sum_{a'} e^{\hat{\beta}Q(s,a'|\boldsymbol{\theta})}} & \text{if } x_i = -1 \end{cases}, \tag{16}$$

where $\hat{\beta} < \beta$ and β is the normal softmax policy parameter used by our probability model (Equation 3), such that f_i is a more nosiy, uniform response.