Supervised Learning of Behaviors

Felipe Glicério Gomes Marcelino

18 April 2020

1 Notes

Slide 3

1. Definition of sequential decision problems
2. Imitation Learning: supervised learning for decision making
    - Does direct imitation work?
    - How can we make it work more often
3. A little bit of theory
4. Case studies of recent work in **deep imitation learning**
5. Goals:
    - Understand definitions & notation
    - Understand basic limitation learning algorithms
    - Understand tools for theoretical analysis

Slide 4 - 5

Terminology & Notation

- $\mathbf{a}_t$ or $\mathbf{u}_t$ - action - Output of the classifier
- $\mathbf{s}_t$ or $\mathbf{x}_t$ - state - *Explained below*
- $\mathbf{o}_t$ - observation - Image
- $\pi_\theta(a|b)$ - policy - Distributions over actions given the observation
- θ is the parameters of the model
- t - time step

The action can be discrete like:

1. Run Away
2. Ignore
3. Pet

Or can be continues: The direction do you want to run when you see the pictures. It can be achieve using multivariate normal distributions.

Deterministic policy - As simply a conditional distribution that outputs a Dirac Delta over just a single action - So one action has probability one and everything else has probability zero.

Some policies is dependent of states and not observation. The difference between states and observations is: An observation that is possible to see. For instance, image pixels. On the other hand, the state is a representation of this image. For example, the image of Cheetah can be represented as physical state: velocity, position, orientation, composition of Cheetah body and etc. The states fully describe everything that is going on in the world whereas the observations can be indistinguishable to someone but has different states between them.

Or can be continuous: The direction do you want to run when you see the pictures. The action can be moduled using multivariate normal distributions.

Or can be continuous: The direction do you want to run when you see the pictures. The action uses multivariate normal distributions as a model, for example.

Deterministic policy - As simple a conditional distribution that outputs a Dirac Delta over just a single action - So one action has probability one, and everything else has probability zero.

- Some policies are dependent on states and not observation. The difference between states and observations is: An observation that is possible to see. For instance, image pixels. On the other hand, the state is a representation of this image. For example, the image of a cheetah represented as a physical state: velocity, position, orientation, the composition of the cheetah body, etc. The states fully describe everything that is going on in the world, whereas the observations can be indistinguishable to someone but have different states between them.

Figure [1] shows the state representation of the observation. As we can see, the image shows a car in front of the cheetah, but the state can represent cheetah using position and vectors. In that case, observation may be insufficient for the model infers about the cheetah.

Slide 5

Figure [2] has a Markov chain representing each element of the decision-making. $p(s_{t+1}|s_t, a_t)$ represents the probability of entering the next state (s_{t+1}) given the current state and the current action.

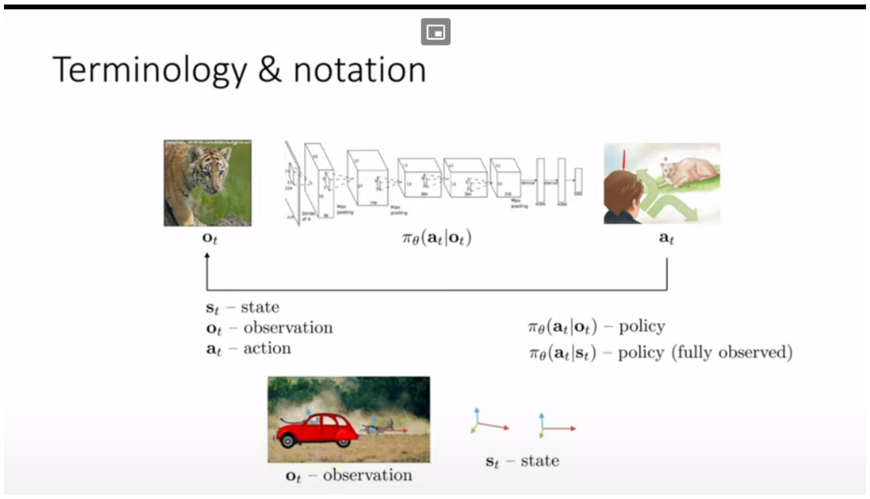

Terminology & notation
$\mathbf{o}_t$
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$
$\mathbf{a}_t$
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)
$\mathbf{o}_t$ – observation
$\mathbf{s}_t$ – state

Figure 1: Representation of state and observation, notations and policy

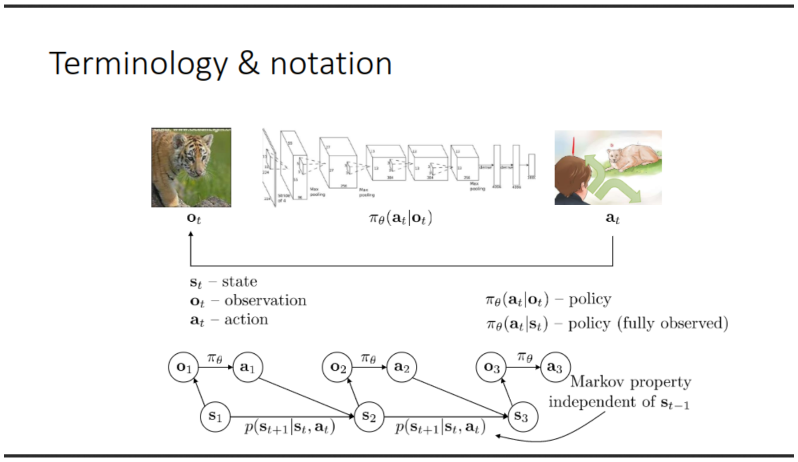

Terminology & notation
$\mathbf{o}_t$
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$
$\mathbf{a}_t$
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)
$\mathbf{o}_1$ π_θ $\mathbf{a}_1$
$\mathbf{o}_2$ π_θ $\mathbf{a}_2$
$\mathbf{o}_3$ π_θ $\mathbf{a}_3$
$\mathbf{s}_1$ $\mathbf{s}_2$ $\mathbf{s}_3$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$
$p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$
Markov property
independent of $\mathbf{s}_{t-1}$

Figure 2: Markov Chain of the given observation, state and action

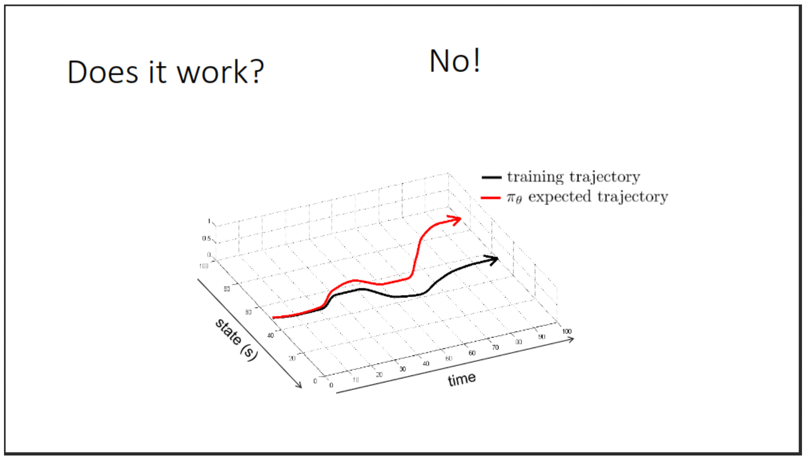


Figure 3: Sequences of states over time.

State s_3 is independent of s_1 given s_2 . It uses the **d-separation** procedure as prof independence. If you know the present, then the future is conditionally independent of the past.

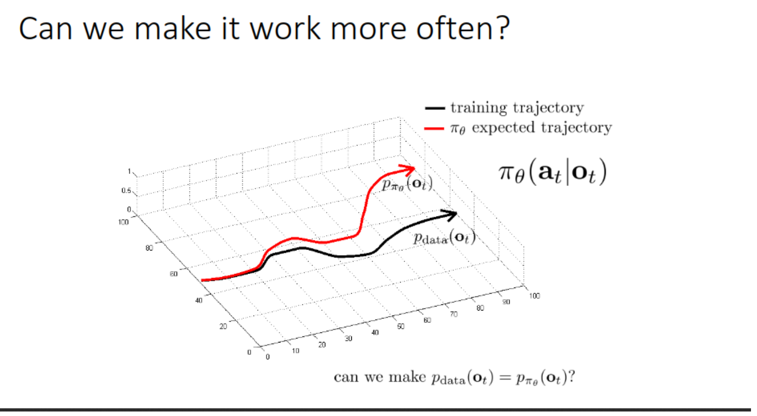
If the model only uses the observations and not the states, they do not form a Markov chain. Further, the observations are not independent among themselves, in such a way that $o_3$is not independent of o_1 given o_2.

Slide 7:12 - Imitation Learning

Behavioral cloning: Uses the observation state and action to create a training dataset. Using this training dataset to train the policy is going to result in a bad generalization. The distribution of the training dataset can't cover all situations, so maybe the policy can't deal with surprise elements.Also, it does not fit into a markovian model because of the dependence of observation spaces. For example, missing images of the car on another lane can struggle with the performance of the model.

The trajectories demonstrate in Figure [3] show that the training trajectory is different from the expected trajectory. It is because when policy commits a mistake, there is a chance that it is going to a state that it hasn't seen in the training dataset. Consequently, the model commits another mistake, more significant than the last one. Now the policy is in state that highly different from the training distribution. Another mistake is going to happen. The mistakes gradually accumulate through time. AS you can see, the expected trajectory ends up been abruptly different from the training trajectory.

Using different trajectories, with some noise, helps the model to recover from smaller mistakes. These "erroneous" trajectories create a more robust policy. However, blind spots problems persist here.



Can we make it work more often?

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Figure 4

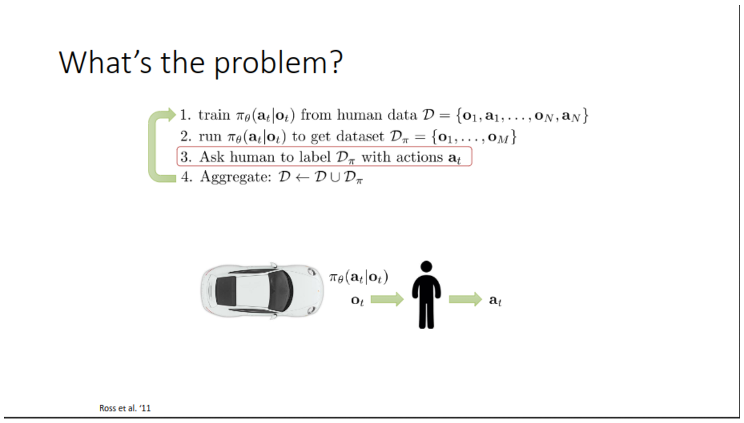
Mathematical problem formalization: The explanation for the difficulty of training the behavioral cloning model is the drift. Once the mode makes the first mistake, the policy starts making more significant mistakes. The observation sampled from training data is called $p_{data}(o_t)$.The observations have dependencies among them. But, supervised learning methods disregard these dependencies. The policy trained with sampled observations is going to generate actions with mistakes. These actions result in observation data also. But, this observation has a different distribution from $p_{data}(o_t)$. This new distribution is called $p_{\pi_\theta}(o_t)$. These two distributions made explicit here are different because the policy takes different actions. So, how can we make these two distributions, $p_{data}$and p_{π_θ} , equals and make the imitation learning achieves a reasonable performance?

The questions: Can we make $p_{data}(o_t) = p_{\pi_\theta}(o_t)$? See Figure [4] to understand.

Slides 13:16 - DAgger: Dataset Aggregation

Goal: Collect training data from $p_{\pi_\theta}(o_t)$ instead of $p_{data}(o_t)$. How to achieve this? Just run the policy.

1. Train $\pi_\theta(a_t|o_t)$ from human data $D = \{\ o_1, a_1, \cdots, o_n, a_n\ \}$
2. run $\pi_\theta(a_t|o_t)$ to get dataset $D_\pi = \{o_1, \cdots, o_M\ \}$
3. Ask human to label D_π with actions a_t
4. Aggregate: $D \leftarrow D \cup D_\pi$



What's the problem?

1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$
$\mathbf{o}_t$
$\mathbf{a}_t$

Ross et al. '11

Figure 5

Slide 15

What is the problem? Using a human to label the observations is expensive. Also, it is necessary to do it iteratively as the model train. Another problem is that people might not provide good labels; in that way, the policy(model) can't achieve reasonable performance.

Slide 16

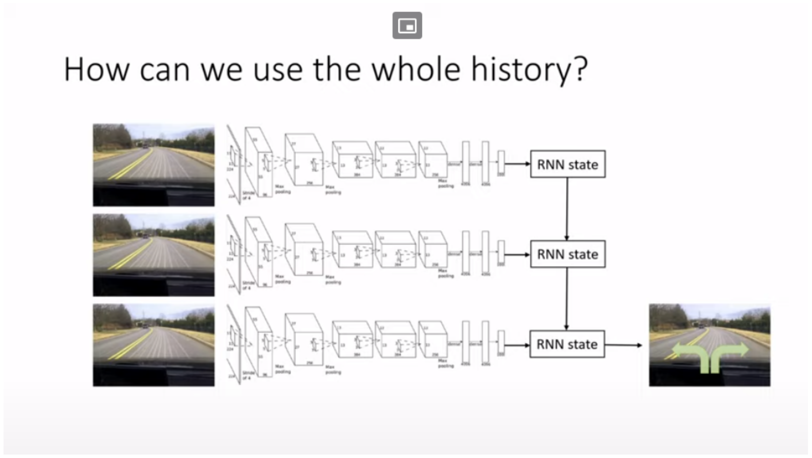
Can we make Dagger work without more data?

- Dagger addresses the problem of distributional "drift". Drift occurs because model makes mistakes, creating a different distribution for observations.
- Theoretical solution: What if the model is so good that it doesn't make mistakes(drift)?
- So, the model must mimic expert behavior very accurately. Is it possible?
- Model has to avoid overfitting.

Slide 17:24 - Problems encountered with DAgger

Why might we fail to fit the expert? Why might the model not mimic the expert's behavior very well?

1. Non-Markovian behavior: A policy is a distribution of our actions conditional on the current observation. $\pi_\theta(a_t|o_t)$ Behavior depends only on current observation. If we see the same thing twice, we do the same thing



How can we use the whole history?

Figure 6: Recurrent Model

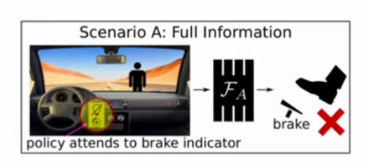

Scenario A: Full Information
$\mathcal{F}_A$
brake
policy attends to brake indicator

Figure 7: With dashboard

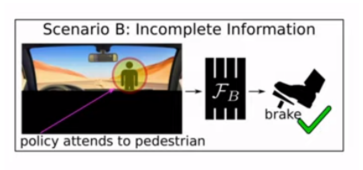
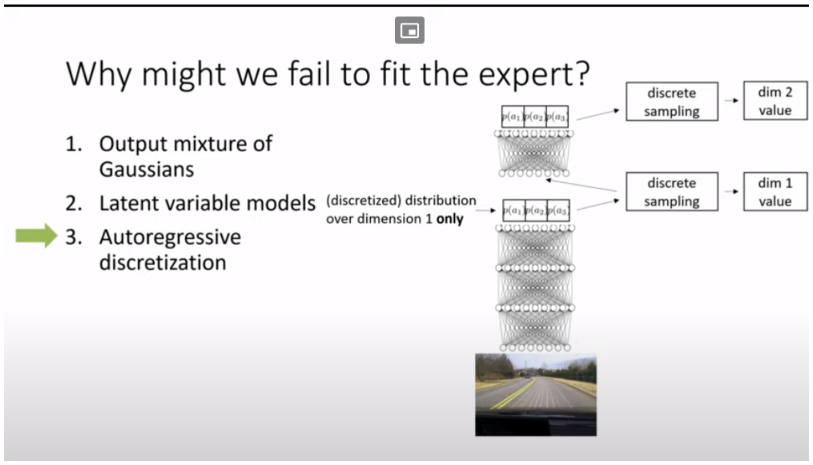

Scenario B: Incomplete Information
$\mathcal{F}_B$
brake
policy attends to pedestrian

Figure 8: Without dashboard

twice, regardless of what happened before. But, it is very unnatural for human demonstrators. It is very kind of unnatural for people to act in a truly markovian way because we can react in slightly or considerably different ways in the same situation. So a better model might be something that produces a distribution over the action condition on a history of past observations $\rightarrow \pi_\theta(a_t|o_1, \cdots, o_t)$. The best model to deal with Non-Markovian observations is recurrent models showed in Figure [6].

- RNN is expensive to train and evaluation.
- Aside: why might this work poorly? - Causal Confusion - In Figure [7] shows a dashboard with the light on the car's panel. The light indicates the usage of the brake. In that case, the model is going to learn that if the light is on then, press the brake, not the other way around.
- Aside: why might this work poorly? - Causal Confusion - In Figure [8] shows a scenario without a dashboard. However, the model can create correlation with things that happens in the past. Unexpected actions are going to happen, breaking far away from the traffic light.


Why might we fail to fit the expert?

1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization

(discretized) distribution over dimension 1 **only**
$p(a_1)$ $p(a_2)$ $p(a_3)$
discrete sampling
dim 1 value
discrete sampling
dim 2 value

Figure 9: Autoregressive discretization model demonstrations

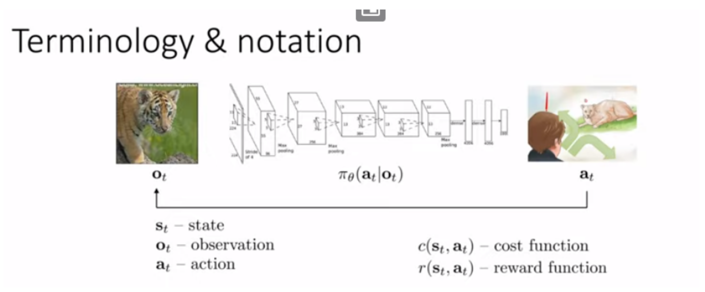
- see: Hann et al,. ”Causal Confusion in Imitation Learning”
- Empirical DAgger can mitigate causal confusion.

2. Multimodal behavior: Use the average of two good actions. However, the average of good actions can be a lousy action instead of a good one. Continuous action is most affected by this problem. Solutions:

   - Output mixture of Gaussians: $\pi(a|o) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$ - Instead of output the mean and variance of a single normal distribution, output N’s different normal distributions(mean, variance) and outputs weights on each of the N distributions. This model might not work so well if the action space is humongous. But it is straightforward to implement
   - Latent variable models: Can express complex things, but they are hard to train. Ex: Conditional Variational Autoencoder(Explained in the second part of the course), Normalizing flow/realNVP or Stein Variational Gradient Descent.
   - Autoregressive discretization: Works in low dimension action space. Using bins to discretize continuous actions. It discretizes one dimension at a time and uses softmax into discrete samples. Consequently, it has a dim 1 value. Feed it as input into another neural net with the images. The output is another discrete dim 1 value, repeat the process. Look at Figure [9]

Slide 29 - Imitation learning: What is the problem?

- Humans need to provide data, which is typically finite


Terminology & notation

$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$c(\mathbf{s}_t, \mathbf{a}_t)$ – cost function
$r(\mathbf{s}_t, \mathbf{a}_t)$ – reward function

Figure 10: Terminology & Notation

  - Deep learning works best when data is plentiful

- Humans are not good at providing some kinds of actions
- Humans can learn autonomously; Can your machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

Slide 30 - Terminology & Notation

Using a new notation and modify the last terminology a little bit to start thinking about learning without imitation. Figure [10]

$$\underset{\theta}{min} E_{a\sim\pi_\theta(a|s),s'\sim p(s'|s,a)}[\delta(s' = \text{eaten by tiger})] \tag{1}$$

$$\underset{\theta}{min} E_{s_{1:T},a_{1:T}}[\sum_t \delta(s_t = \text{eaten by tiger})] \tag{2}$$

$$\underset{\theta}{min} E_{s_{1:T},a_{1:T}}[\sum_t c(s_t, a_t)] \tag{3}$$

Equation (1) means: Minimize the expected number of times the tiger eats someone. Delta function is one if the tiger eats it or zero otherwise. While equation (2) is minimization using the temporal aspect. It minimizes the expectation for a sequence of states and actions of the delta function for being eaten by the tiger and it is closer to what a proper reinforcement problem seems. Replacing delta function with cost and then equation (2) transform to equation (3). Now, we want to minimize these costs over a time sequence. The distributions of states are according to the dynamics, and the actions distribute according to the policy. The cost gives the model a numerical quantity of how bad things are, so getting eaten by a tiger might have a cost of a million and get bitten by a mosquito might have a cost of one. The other option used is the transformation of the cost function into a reward function but as a maximization objective.

Slide 32 - A cost function for imitation?

It is possible to formalize reward and cost function to an imitation model. (4):

$$r(s,a) = logp(a = \pi*(s)|s) \qquad c(s,a) = \begin{cases} 0 \text{ if a } = \pi*(s) \\ 1 \text{ otherwise} \end{cases} \tag{4}$$

- Reward function: Maximize the log probability of the policy's action matches the action from deterministic optimal policy $\pi*$
- Cost function: Cost is zero if action matches the optimal action and one otherwise. Zero one loss. Just counting the number of mistakes the model makes. Don't care about where the model is in a good state or bad state; only care about is whether the model did the same thing as what the expert would have done.

Analysis

Using the cost function from equations (4). Assume:

$$\begin{gathered} \pi_\theta(a \neq \pi*(s)|s) \leq \epsilon \\ \text{for all s} \in D_{train} \end{gathered} \tag{5}$$

This assumption in equations (5) says: Probability of taking the wrong action would be less than or equal to ϵ. In other words, it says the model can memorize the data, but no perfectl.The value of ϵ depends on how good the model is. Better models have small ϵ .

$$\mathbb{E}\left[\sum_t c(s_t,a_t)\right] \leq \underbrace{\epsilon t + (1-\epsilon)(\epsilon(t-1) + (1-\epsilon)(\cdots))}_{\text{T terms, each O}(\epsilon\text{t})} \tag{6}$$

Equation (6) states: T is the total steps to achieve the objective. $\epsilon\mathbb{T}$ is the probability off falling off. So once the model makes one mistake, the model just makes mistakes forever. If the model doesn't fall off, then it has $(1-\epsilon)$ (that is the probability of not falling off) probability of taking the right action and go to the next state, which is in its training set and then basically the same thing happens. It gets the probability $\epsilon(T-1)$ of falling off to the next T - 1 steps and then repeats. Assuming the $(1-\epsilon)$ is close to 1, each of these terms in the sum is on the order of ϵT. So, it has T terms, each $O(\epsilon T)$, hence the bound going to be $O(\epsilon T^2)$. It means that as the length of the time steps grows, the number of mistakes increases quadratically. Briefly, this assumption is not good because the generalization of models isn't taking account of this assumption.

More General Analysis

Using cost function from (4), let's do a less naive analysis. Assume:

$$\begin{aligned} \pi_\theta(a \neq \pi*(s)|s) \leq \epsilon \\ \text{for } \mathbf{s} \sim p_{train}(s) \end{aligned} \tag{7}$$

States sampled from p_{train} distribution. For states that were not sampled from this distribution, all bets are off (It is not possible to say if some sampled is from p_{data} or other distribution). The objective of the proof is: Even if the samples come from another distribution, it is possible to get samples that look like samples from p_{train}.

Dagger

With DAgger, $p_{train}(s) \rightarrow p_\theta(s)$. The p_{train} distribution converges to p_θ distribution whenever the model uses the policy and aggregation operation explained above is used. If p_{train}becomes equal to p_θ then bound of the expectation of the total cost is ϵT. It is assume that the model never end up seeing out of distribution states because the distributions is equal. Every single step ϵ probability to fall off, then there are T steps.

$$\mathbb{E}\left[\sum_t c(s_t, a_t)\right] \leq \epsilon T \tag{8}$$

Behavioral Cloning

If $p_{train}(s) \neq p_\theta(s)$:

$$p_\theta(s_t) = \underbrace{(1-\epsilon)^t}_{\text{probability we made no mistakes}} p_{train}(s_t) + (1-(1-\epsilon)^t) \underbrace{p_{mistake}(s_t)}_{\text{some other distribution}} \tag{9}$$

Distribution of states at time step T. The first term of equation (9) means the probability of not making a mistake. The model is at a time step T is indistinguishable from where the expert would have been. The other part of equation (9) is some other distribution we don't know.

$$|p_\theta(s_t) - p_{train}(s_t)| = (1-(1-\epsilon)^t)|p_{mistake}(s_t) - p_{train}(s_t)| \tag{10}$$

Equation (10) states the total variation divergence between two distributions. The maximum value possible is 2, when two distributions are totally different.

$$|p_\theta(s_t) - p_{train}(s_t)| = \sum_{s_t} |p_\theta(s_t) - p_{train}(s_t)| \tag{11}$$

Using the expression after the equal signal from equation (9) to replace p_θ and p_{train} results in equation (11):

$$\begin{aligned}(1-\epsilon)^t p_{train}(s_t) + (1-(1-\epsilon)^t)p_{mistake}(s_t)\\ -(1-\epsilon)^t p_{train}(s_t) - (1-(1-\epsilon)^t)p_{train}(s_t) =\\ \sum_{s_t} |p_\theta(s_t) - p_{train}(s_t)|\end{aligned} \tag{12}$$

Cancels $(1-\epsilon)^t p_{train}(s_t)$ above in equation (12). $(1-(1-\epsilon))$ is positive, putting it in evidence, hence take outside of absolute value. Because of this:

$$|p_\theta(s_t) - p_{train}(s_t)| = (1-(1-\epsilon)^t))|p_{mistake}(s_t) - p_{train}(s_t)| \tag{13}$$

As mentioned before, the worst possible total variation divergence value is 2. Area of PDF(probability density function) is 1, as a consequence, when two distributions are different, and the mass doesn't overlap, the sum of the area is 2. With this in mind:

$$|p_\theta(s_t)-p_{train}(s_t)| = (1-(1-\epsilon)^t))|p_{mistake}(s_t)-p_{train}(s_t)| \leq 2(1-(1-\epsilon)^t) \tag{14}$$

Useful identity: $(1-\epsilon)^t \geq 1-\epsilon t$ for $\epsilon \in [0,1]$ $=(1-(1-\epsilon)^t) \leq \epsilon t$
As result, (14) is:

$$|p_\theta(s_t) - p_{train}(s_t)| = (1-(1-\epsilon)^t))|p_{mistake}(s_t) - p_{train}(s_t)| \leq 2\epsilon t \tag{15}$$

Using equation (15) and calculate the bound of the cost. Pushing expectation inside the sum by the linearity of expectation:

$$\sum_t \mathbb{E}_{p\theta(s_t)}[c_t] = \sum_t \sum_{s_t} p_\theta(s_t)c_t(s_t) \leq \sum_t \sum_{s_t} p_{train}(s_t)c_t(s_t)+|p_\theta(s_t)-p_{train}(s_t)|c_{max} \tag{16}$$

Explaining the right part of inequality (16) above:

$$\begin{aligned}p_\theta = p_{train} + p_\theta - p_{train}\\ p_\theta c = p_{train}c + (p_\theta - p_{train})c\\ \leq p_{train}c + |p_\theta - p_{train}|c_{max}\\ \text{Adding cost on all terms}\\ \text{When the equation has a difference between two terms}\\ \text{we know that is less than or equal to the absolute value of difference}\\ c_{max} \text{ is one, because it is a zero-one loss}\end{aligned}$$

Returning to equation (16), now we have all elements for deriving its bound.The single-step of the first term of the right part of inequality is bounded by ϵ . Next, the second term, the absolute value, is bounded by ϵt calculated in equation (15).

$$\sum_t \mathbb{E}_{p\theta(s_t)}[c_t] = \sum_t \sum_{s_t} p_\theta(s_t)c_t(s_t) \leq \sum_t \epsilon + 2\epsilon t \leq eT + 2\epsilon T^2 \tag{17}$$

Using O notation on the bound: $O(\epsilon T^2)$. It shows why behavioral cloning can be problematic.