Imitation Learning with Dataset Aggregation Method and Convolutional Neural Networks

Charles Dognin
charles.dognin@ensae.fr

Jean-Baptiste Remy
jean.baptiste.remy@ensae.fr

Abstract

In this document we implement the paper A Reduction for Imitation Learning and Structured Prediction to No Regret Online Learning by Stephane Ross, Geoffrey J.Gordon and J.Andrew Bagnell to train a bot that plays the Enduro-V0 game, a car racing game based on the Open AI gym environment. We use the DAGGER algorithm to train the AI using Imitation Learning. Our detailed implementation can be found at https://github.com/JbRemy/Imitation_Learning.

1 INTRODUCTION

Imitation learning is a major sub-field of reinforcement learning with in particular many applications in robotics. Such systems cannot afford to fail or make too many errors because of the cost of replacement of their components. Contrary to the mainstream reinforcement learning, imitation learning uses the information of an expert that will guide the AI. The expert is generally a human but can also be an optimal/near-optimal planner. Typical approaches to imitation learning seek to train a classifier or regressor on a training set generated by the expert. The inputs are the observations/the states and the output the actions. Nonetheless the fact that the learner's predictions/outputs will be use later for inputs violate the crucial statistical learning hypothesis, that is the iid assumption. Ignoring this problem leads to poor performance. If the learner makes a mistake, it may encounter completely different observations than the expert's, leading to large and compounding errors. The SMILe algorithm proposed by Ross & Bagnell (2010) and the DAGGER algorithm proposed by the same authors in 2011 offer solutions to overcome this problem. Instead of mobilizing the experts only once to generate a training set, the expert guide the AI along several steps and progressively disappear to let the learned policy working.

2 MATHEMATICAL FRAMEWORK

In imitation learning, the goal is to mimic the expert policy π^*. The agent (learner) needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution. This policy is assumed to be deterministic, which means that $\pi^*(s)$ is the action chosen by the expert when he is in state s. For this purpose we want to minimize the 0-1-loss. For all state s the problem writes :

$$\min_{\pi} e_\pi(s) = E_{a\sim\pi_s}[e(s,a)] \tag{1}$$

Where, $e(s,a) = 1_{\pi^*(s)}(a)$ and π_s the distribution of all states conditionally to the current state s.

Our interest is to bound the regret $\mathcal{R}_\Pi(\pi) = J(\pi) - \min_{\pi'\in\Pi} J(\pi')$, where $J(\pi) = TE_{s\sim d_\pi}[e_\pi(s)]$ is the expected T-step regret and $d_\pi(s) = \frac{1}{T}\sum_{i=1}^{T} d_\pi^i$ [1] is the state visitation frequency.

[1] with d_π^i the distribution of states at time i for the policy π

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

2.1 The traditional approach

The traditional approach minimizes the $0-1$ loss under distribution d_{π^*} : $\min_\pi E_{s\sim d_{\pi^*}}[e_\pi(s)]$. The problem with this method is that the algorithm doesn't learn to recover from potential mistakes. The consequence is that, if under $\hat{\pi}$ a classifier makes a mistake at a rate ϵ, $J(\hat{\pi}) \leq J(\pi^*) + T^2\epsilon$. T being the number of steps for each trajectory.

To solve this problem we can use a Forward Training Algorithm in which the policy is iteratively learn on each time step, conditionally to the learned policy for the previous time steps. Ross and Bagnell proposed the SMILe Algorithm. This algorithm allows the agent to learn to recover from it's mistakes by repeating the training N times while mixing the expert policy with the learned policy at each step.

2.2 The Stochastic Mixing Iterative Learning Algorithm

In the SMILe algorithm the initial policy is set to be the expert policy $\pi^0 = \pi^*$. Then the current expert policy of iteration n, π^n is stochastically mixed with the learned policy $\hat{\pi}^{n+1}$ at each step. Hens, if the mixing rate is α $\pi^{n+1} = (1-\alpha)\pi^n + \alpha\hat{\pi}^{n+1}$. The final output policy must be cleared of the expert policy thus $\tilde{\pi}^N = \frac{1}{1-(1-\alpha)^N}[\pi^N - (1-\alpha)^N\pi^*]$.

With this algorithm, we can show that for $k \in \{1, .., T-1\}$:

$$J(\pi^n) \leq J(\pi^0) + n\sum_{i=1}^{k} \alpha^i T i (1-\alpha)^{T-i}\bar{A}_i + n\alpha^{k+1}TTk + 1 \tag{2}$$

Where $\bar{A}_i = \frac{1}{n}\sum_{j=1}^{n} A_i(\pi^{j-1}, \hat{\pi}^j)$ is the average regret of executing i times the learned policy over the current expert policy[2]. This allows to control regret under $\tilde{\pi}^n$ since we can prove that :

$$J(\tilde{\pi}^n) \leq J(\pi^n) + p_n T^2 \tag{3}$$

with $p_n = (1-\alpha)^n$.

Using the previous bound 2, regret could be reduce to 0 if $A_k(\pi^{n-1}, \hat{\pi}^n) = 0$. This can be approximate by choosing $\hat{\pi}^n$ to mimic π^n. The update is then :

$$\hat{\pi}^n = p_{n-1}\hat{\pi}^{*n} + (1-p_{n-1})\tilde{\pi}^{n-1}, \text{ with } \hat{\pi}^{*n} = \underset{\pi}{\operatorname{argmin}}\, E_{s\sim d_{\pi^{n-1}}}(e_\pi(s)) \tag{4}$$

The regret for such a choice of update can be bounded as follows for $\alpha = \frac{\sqrt{3}}{T^2\sqrt{\log(T)}}$, and $N = 2T^2(\log(T))^{3/2}$:

$$J(\tilde{\pi}^N) \leq J(\pi^*) + O(T(\tilde{A}_1 + \tilde{\epsilon}) + 1) \tag{5}$$

2.3 The Dataset Aggregation Method

2.3.1 Description

DAGGER brings learners and experts trajectory distributions closer by labelling additional data points resulting from applying the current policy.

The DAGGER algorithm is an iterative algorithm that trains a deterministic policy. We start by collecting several expert's trajectories to build the initial $\mathcal{D}$ dataset. Then at each step i we use three actors. There is π_i, a combination of the expert and the learned policy, upgraded at each step. π_i^* is the expert policy and $\hat{\pi}_i$ is the learned policy. At each step i, we:

1. Upgrade the transitory policy : $\pi_i = \beta_i\pi^* + (1-\beta_i)\hat{\pi}_i$
2. Sample T-steps trajectories with π_i

[2] $A_i(\pi, \pi') = \bar{J}_i^\pi(\pi') - J(\pi)$, with $\bar{J}_i^\pi$ the average expected T-step cost of executing i times the policy π' instead of policy π

3. Get dataset $\mathcal{D}_i$ of visited states by π_i and actions given by the expert for those states
4. Aggregate datasets $\mathcal{D}$ and $\mathcal{D}_i$
5. Train classifier $\hat{\pi}_{i+1}$ on the new richer dataset $\mathcal{D}$.
   After N step, we choose the $\hat{\pi_i}$ that performed best on validation

It is as if at each step, we were asking the expert his opinon about our current trajectory. Then gathering this opinion (his response to the states we encountered) and the previous datasets of trajectories, we can train a new policy more accurate because taking into account more expert's opinion. Regarding the mixing transitory policy, we initially choose $\beta_1 = 1$ and $\beta_i = p^{i-1}$ to have probability of using the expert that decays exponentially.

2.3.2 Main Theoretical results

Infinite sample case We suppose l strongly convex, bounded over Π, $\beta_i \leq (1-\alpha)^{i-1}$ for all i for some constant α independent of T and $\epsilon_N = min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} E_{s \sim d_{\pi_i}}[l(s, \pi)]$

1. If N is $\tilde{O}(T)$, there exists a policy $\hat{\pi} \in \hat{\pi_{1:N}}$ such that $E_{s \sim d_{\hat{\pi}}}[l(s, \hat{\pi})] \leq \epsilon_N + O(\frac{1}{T})$.
2. If N is $\tilde{O}(uT)$ there exists a policy $\hat{\pi} \in \hat{\pi_{1:N}}$ such that $J(\hat{\pi}) \leq J(\pi^*) + uT\epsilon_N + O(1)$

Finite sample case We suppose sampling m trajectories with π_i at each iteration i and denote this dataset D_i. Let $\hat{\epsilon}_N = min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} E_{s \sim D_i}[l(s, \pi)]$

1. If N is $O(T^2 log(\frac{1}{\delta}))$ and m is $O(1)$ then with probability at least $1-\delta$ there exists a policy $\hat{\pi} \in \hat{\pi_{1:N}}$ such that $Es \sim d_{\hat{\pi}}[l(s, \hat{\pi})] \leq \epsilon_N + O(\frac{1}{T})$
2. If N is $O(u^2 T^2 log(\frac{1}{\delta}))$ and m is $O(1)$ then with probability at least $1-\delta$ there exists a policy $\hat{\pi} \in \hat{\pi_{1:N}}$ such that $J(\hat{\pi}) \leq J(\pi^*) + uT\hat{\epsilon_N} + O(1)$

No Regret Algorithms Guarantees for Infinite and Finite sample case

1. For DAGGER, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ such that $E_{s \sim d_{\hat{\pi}}}[l(s, \hat{\pi})] \leq \epsilon_N + \gamma_N + \frac{2l \max}{N}[n_\beta + T \sum_{i=n_\beta+1}^{N} \beta_i]$, for γ_N the average regret of $\hat{\pi_{1:N}}$
2. For DAGGER, with probability at least $1-\delta$, there exists a policy $\hat{\pi} \in \hat{\pi}_{1:N}$ such that $E_{s \sim d_{\hat{\pi}}}[l(s, \hat{\pi})] \leq \hat{\epsilon}_N + \gamma_N + \frac{2l \max}{N}[n_\beta + T \sum_{i=n_\beta+1}^{N} \beta_i] + l_{max} \sqrt{\frac{2log(1/\delta)}{mN}}$ for γ_N the average regret of $\hat{\pi}_{1:N}$

3 EXPERIMENTS

We initially wanted to implement both algorithms. After discussing with Stephane Ross, one of the author, who told us that he personally always found the DAGGER working better than SMiLE, we decided to focus on the more efficient algorithm. In order to compensate for that, we decided to try a novel approach using convolutionnal neural networks to train the classifier at each step.

3.1 The environment: Open AI gym and Enduro-V0

The Open AI gym platform is a mainstream reinforcement learning environment which allows people to test the accuracy and effectiveness of their algorithm. We chose a game that was not too simple nor too complex for us to show the expert policy: Enduro is a racing game where the goal is to beat other AI adversaries. There are 9 actions in the game which correspond to different combinations of acceleration and steering wheel orientations. We play the roles of the expert, showing the way to our AI at each step.

Figure 1: Enduro

3.2 The inputs and the model

At each step, learning $\hat{\pi}$ is achieved by training a classifier to choose the appropriate action with respect to the state.

The inputs of the classifier are: the current image displayed by the game, and the last three images. The past images allow the agent to understand the dynamic of the instant and make a proper decision. If it were to be fed only with the current image it would not be able to assess whereas something is moving right or left for example.

For the classification task we built a neural network with a convolutional layer followed by a fully connected module. Since we feed for images to the network, the dimension of the input is very large[3]. To reduce it, the convolutional layer is followed by a max pooling layer of kernel size 2×2 and stride 2, allowing the dimension to drop to 257088 inputs units for the fully connected layer. At first we build a network with only a fully connected hidden layer as it was implemented as such in the articles, but with this game, performing direct max pooling to reduce the dimension looses to much information, mainly because the road is delimited by a thin line, that gets lost. Adding a convolutional layer extracts important feature way more efficiently. The agent went from, always going left to being able to drive himself toward the center.

To simulate the expert policy we are going to play by ourselves. To simulate the transitory-mix policy we are going to let the algorithm play at random times corresponding to probability $\beta_i = p^i$.

Results

We do three iterations of the DAGGER algorithm. Each time, we compute the accuracy of the trained neural network-based classifier. The first iteration gives us an accuracy of 31%, the second iteration an accuracy of 33% and the third one is 35%. The consistency of this accuracy proves that improvements of the agent are only due to the data set aggregation, and not to a better classification

[3] $4 \times 160 \times 210 \times 3 = 403200$

algorithm.

To measure the effectiveness of our learned algorithm, we count the score the algorithm gets without our intervention on a specified portion of the game. Namely, we wait for the level to change from a green environment to a white one.

Iteration	**1**	**2**	**3**
Score	66	80	82

Table 1: Performance of the agent

References

[1] Ross, Stephane & J.Gordon Geoffrey (2011) A Reduction of Imitation Learning and Structured Prediction to No-Reret Online Learning *Appearing in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics 2011*

[2] Ross, Stephane & Bagnell, J.Andrew (2010) Efficient Reduction for Imitation Learning *Appearing in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics 2010*