DAgger: Interactive Experts and No-Regret Learning

Sanjiban Choudhury
Easy

Expert is realizable
\[ \pi^E \in \Pi \]

As \( N \to \infty \), drive down
\[ \epsilon = 0 \text{ (or Bayes error)} \]

Nothing special.
Collect lots of data and
do Behavior Cloning

Hard

Non-realizable expert + limited expert support

Even as \( N \to \infty \),
behavior cloning \( O(\epsilon T^2) \)
Why can’t we just collect data \((s, a^*)\) on the distribution of states the learner visits?
Introducing an *interactive* expert!
To know the distribution, you need a learner
To train a learner, you need a distribution
Activity!
Think (30 sec): How can we solve the chicken and egg problem, i.e. train the learner on a distribution of states it visits?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
An *embarrassingly* simple algorithm: FORWARD

Idea: Train a different learner policy at every timestep by interactively querying expert

Get start state samples $s_0 \sim d^0(.)$

for $t = 0 \ldots T-1$

Query interactive expert to get

$a_t^* = \pi^*(s_t)$

Train a learner policy at time $t$

$\pi^t = \text{Train}(s_t, a_t^*)$

Execute learner policy $\pi^t$ to get

next state samples $s_{t+1} \sim d^{t+1}_{\pi}(.)$
But what if we want ONE policy?
DAGGER

Episode IV

A NEW HOPE
DAGGER: A meta-algorithm for imitation learning

A Reduction of Imitation Learning and Structured Prediction
to No-Regret Online Learning

Stéphane Ross
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu
DAGGER: Iteration 0

Data

Human drives

Policy $\pi_0$

[Ross et al’11]
DAgger: Iteration 1

Robot $\pi_0$ drives

Human corrects!

Policy $\pi_1$
DAgger: Iteration 2

Robot $\pi_1$ drives

[Ross et al'11]

AGGREGATE DATA
DAgger: Iteration 1

Robot $\pi_N$ drives

After many iterations ....
we are able to drive like a human!
Dagger (Dataset Aggregation)

Aggregated Training Distribution

Test Distribution

Human Distribution
But why does *aggregating* data work?
Imitation learning is just a game

Be stable

Slowly change predictions
The Imitation Game

Learner

- Initialize policy

Adversary

- Chooses loss

\[ \pi_1 \] [policy]

\[ l_1(\cdot) \] [loss]

\[ \pi_2 \]

\[ l_2(\cdot) \]

Update policy

...(continue)
Let’s prove!
How can I customize DAGGER to be more practical?
Customizing your DAGGER

Q1. The policy iteration at step 1 is crappy and visits irrelevant states. What do I do?

   Blend the expert and learner policy \( \pi_i = (1 - \beta_i) \hat{\pi}_i + \beta_i \pi^* \)

Q2. What if I can’t afford to store all the aggregated data?

   Online gradient descent!
Original results from DAGGER!
DAgger here reacts dynamically to an untrained obstacle
A brief history of DAGGER

NLP folks (Hal Daume III in particular) were first to identify feedback effects in sequential prediction tasks

- Feedback: Previous word predictions feed back in as inputs for future inputs. Data is non-IID.
- Search-based Structured Prediction (SEARN, Daume 2009) looks at this problem for part-of-speech (POS) tagging and handwriting recognition
- DAGGER made the connected the fields on sequential prediction, imitation learning and online learning
DAGGER is a foundation

Imitation under uncertainty

SAIL
ExPLORE
STROLL

Counterfactual Teaching

Agnostic
SysID
DaaD

Model learning

DAEQUIL
AGGREVATE(D)

Imitation learning

DPI
LOLS
NRPI

Reinforcement Learning

EIL
HG-DAGGER
SHIV

Query efficient imitation learning
Many cool applications of DAGGER in robotics

Lee et al, Learning quadrupedal locomotion over challenging terrain (2020)

Choudhury et al, Data Driven Planning via Imitation Learning (2018)

Chen et al Learning by Cheating (2020)

Pan et al Imitation learning for agile autonomous driving (2019)
DAGGER is not just for imitation learning!

Agnostic System Identification
for Model-Based Reinforcement Learning

Stéphane Ross  
Robotics Institute, Carnegie Mellon University, PA, USA  
stephaneross@cmu.edu

J. Andrew Bagnell  
Robotics Institute, Carnegie Mellon University, PA, USA  
dragnell@ri.cmu.edu
Hidden charges from DAGGER
Hidden charge #1: Not all mistakes are equal

Consider the example shown in Fig. 1. The expert stays on the track and never visits bad states $S_{bad}$. The learner, on the other hand, immediately drifts off into $S_{bad}$. Moreover, for all $s \in S_{bad}$, the learner can perfectly imitate the expert. In other words, $(s, \hat{\pi}) = 0$ for these states. In fact, it is likely that for certain policy classes, this is the optimal solution! At the very least, DAGGER is susceptible to learn such policies as is shown in the counter example in Laskey et al. [8].

Various alternate formulations for imitation learning The phenomenon discussed in the previous section leads to a host of interesting questions and problems. We will simply list them here and then tackle them one by one.

4.1 Constrained policy search It seems we can overcome the ‘cheating in bad states’ phenomenon by constraining the space of policies to not enter $S_{bad}$ (or have state visitation probabilities to be low under the induced distribution). How do we specify such constraints and how do we solve IL under such constraints?

4.2 Distribution matching If we dig a little deeper, we realize that instead of minimizing imitation loss as we do in (1), what we really care about is making sure our policy distribution is the same as the experts. This is exactly the premise of GAIL [5].

4.3 Expert in the loop Say we have an ‘expert-in-the-loop’ that overrides the learner during roll-out. This would cause a train test distribution mismatch? How should we handle this?

4.4 Actively minimizing calls to the expert It seems wasteful for the expert to have to label corrective actions that are not relevant to the task, e.g. $s \in S_{bad}$. That being said, it seems wasteful for the expert to have to label states where the policy has

2 Do we need stationarity?
3 Which we will not yet directly use
4 For stochastic policies, we sample actions $a_t \sim \pi(s_t)$

We put in $\mathbb{1}$ to allow some flexibility in case we want non-uniform loss or we want to bring back the cost function into the loss function.

Dagger minimizes 0-1 loss, but what we really want to optimize are advantages! (More next lecture)
Hidden charge #2: Dagger asks the expert for queries *everywhere*

We’ll see how to learn from limited human feedback (interventions)
Hidden charge #3: Dagger expects at least one policy to be good *everywhere*

Learner simply can’t cross the bridge! …

… but can take the long way round.