Query-Efficient Imitation Learning for End-to-End Simulated Driving

Jiakai Zhang, Kyunghyun Cho
New York University
Overview

- Introduction
  - End-to-end learning for self-driving
  - Related work

- Learning method
  - Convolutional neural network
  - Imitation learning using SafeDAgger

- Experiment
  - Setup
  - Results

- Conclusion and future work
Introduction

➤ End-to-end learning for self-driving

• Sensory input from front-facing camera

• Control signal

Steering + Brake
Introduction

Related work

• Supervised learning
  • ALVINN net [Pomerleau 1989]
  • DeepDriving [Chen et al. 2015]
  • End-to-end learning for self-driving cars [Bojarski et al. 2016]

• Imitation learning
  • DAgger [Ross, Gordon, and Bagnell 2010]
  • SafeDAgger [Zhang and Cho 2017]
DAgger algorithm

Initialize

Dataset $D_0$ + Policy $\hat{\pi}_1$

Policy $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$ → Dataset $D'$

Iteration

Policy $\hat{\pi}_i$ ← Dataset $D_i = D' \cup D_{i-1}$

Return

Best policy $\hat{\pi}_i$

Disadvantage:
- Query a reference policy constantly
- Safe issue to environment
SafeDAgger algorithm

Initialize

- Dataset $D_0$
- Policy $\hat{\pi}_1$
- Safety classifier $c_1$

Iteration

- Policy $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$
- Safety classifier $c_i$

- Policy $\hat{\pi}_i$
- Safety classifier $c_1$

Return

- Best policy $\hat{\pi}_i$
- Safety classifier $c_i$

Dataset $D'$ not safe

Dataset $D_i = D' \cup D_{i-1}$

Advantage:
- Query-efficient
- Safety feature
Safety classifier

- Deviation of a primary policy from a reference policy defined
\[
\epsilon(\pi, \pi^*, \phi(s)) = \|\pi(\phi(s)) - \pi^*(\phi(s))\|^2
\]

- Optimal safety classifier defined as
\[
c_{\text{safe}}^*(\pi, \phi(s)) = \begin{cases} 
0, & \text{if } \epsilon(\pi, \pi^*, \phi(s)) > \tau \\
1, & \text{otherwise}
\end{cases}
\]

Learning safety classifier

- Minimize a binary cross-entropy loss
\[
l_{\text{safe}}(c_{\text{safe}}, \pi, \pi^*, D') = -\frac{1}{N} \sum_{n=1}^{N} c_{\text{safe}}^*(\phi(s)_n') \log c_{\text{safe}}(\phi(s)_n', \pi) + (1 - c_{\text{safe}}^*(\phi(s)_n')) \log(1 - c_{\text{safe}}(\phi(s)_n', \pi))
\]
Experiment – Setup

➢ TORCS – Open source racing game

Training tracks

Test tracks
### Experiment – Model

| Input image – 3x160x72 |  
|------------------------|---|
| Convolutional layer – 64x3x3 | x 4 |
| Max Pooling – 2x2 |  
| Convolutional layer – 128x5x5 |  
| Fully connected layer | x 2 |
| **Control signals** | **Environment variables** |  
|  

![Diagram](image.png)

- **Primary policy**
- **Safety classifier**

Optimization algorithm: stochastic gradient descent
Results

Safe Frames

Unsafe Frames
Results

- Evaluation on test tracks
  1. Mean squared error of steering angle
  2. Damage per lap
  3. Number of laps
  4. Portion of time driven by a reference policy
Results

Mean squared error of steering angle

- Dashed curve – with traffic
- Solid curve – without traffic

MSE (Steering Angle)

# of Dagger Iterations

DAgger, SafeDAgger, Supervised
Results

Damage per Lap

# of Dagger Iterations

Damage per Lap

Dashed curve – with traffic
Solid curve – without traffic
Results

Number of Laps

Dashed curve – with traffic
Solid curve – without traffic
Results

Portion of time driven by a reference policy

- Dashed curve – with traffic
- Solid curve – without traffic

# of Dagger Iterations

% of $c_{safe} = 0$

DAGger
SafeDAGger
Demo

By model
Conclusion

➢ Proposed SafeDAgger algorithm
  • Query efficient
  • Safety feature

➢ End-to-end simulated driving
  • Trained a convolutional neural network to drive in TORCS with traffic

Future work

➢ Evaluate SafeDAgger in the real world
➢ Learn to use temporal information