Reinforcement Learning in Autonomous Driving

Apprenticeship and Interactive Reinforcement Learning in Autonomous Driving

Supervisor: M.Sc. Sina Shfaei, Presenters: Alexander Prams, Mirko Gschwindt
Agenda

- Motivation
- Interactive Reinforcement Learning
- Apprenticeship Learning via Supervised Learning
- Apprenticeship Learning via IRL
- State-of-the-Art
- Proposals for future Applications
Human-like behavior
Traditional Reinforcement Learning

\[
\max \mathbb{E}[R(s_0) + \ldots + R(s_T) | \pi]
\]

- Environment Model (MDP)
- Reward Function \( R(s) \)
- Reinforcement Learning
- Optimal policy \( \pi \)
Traditional Reinforcement Learning

Environment Model (MDP)

Reward Function \( R(s) \)

Reinforcement Learning

Optimal policy \( \pi \)
Traditional Reinforcement Learning

Environment Model (MDP)

\[ \max E[\sum_{t=0}^{\infty} R(s_t) | \pi] \]

Optimal policy \( \pi \)
Problem Summary

• Sometimes we want human-like behavior
• Reward function hard to formulate for certain tasks
• Long training times
• Unstable training runs for high dimensional environments
Agenda

- Motivation
- Interactive Reinforcement Learning
- Apprenticeship Learning via Supervised Learning
- Apprenticeship Learning via IRL
- State-of-the-Art
- Proposals for future Applications
Interactive Reinforcement Learning
Interactive Reinforcement Learning

Abstract idea: have a human supervisor that influences the training process by helping the agent

- Give action-based advices
- Adjust/give rewards
Approach: Action Advisor

- Agent in state $S(t)$
- Action $A(t)$
- External Trainer
- Environment
- Reward $R(t+1)$
- State $S(t+1)$
- Confirms action $A(t)$ or selects an alternative action
Approach: Reward Correction

Confirms reward $R(t+1)$ or gives a different reward

External Trainer

Agent in state $S(t)$

State $S(t+1)$

Reward $R(t+1)$

Environment

Action $A(t)$
Agenda

Motivation
Interactive Reinforcement Learning
Apprenticeship Learning via Supervised Learning
Apprenticeship Learning via IRL
State-of-the-Art
Proposals for future Applications
Supervised Action Learning
Supervised Action Learning

Learn a direct mapping from certain states to actions

Problem: New situations need new data labelled by humans
DAGGER (Dataset Aggregation): An Extension of Supervised Learning

1. Use the expert’s policy to gather a dataset of trajectories $D$

2. Train policy $\pi_2$ that best mimics the expert on these trajectories

3. Use $\pi_n$ to collect more trajectories and add them to dataset $D$

4. Policy $\pi_{n+1}$ is the policy that best mimics the expert on the whole dataset $D$

Source: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning (Ross, Gordon, Bagnell 2011)
DAGGER (Dataset Aggregation): An Extension of Supervised Learning

Iterative Learning Algorithm

Intuitive Approach: Over the iterations DAGGER builds up a set of inputs that the learned policy is likely to encounter during execution based on previous experience.
DAGGER (Dataset Aggregation): An Extension of Supervised Learning

Super Tux Kart

![Graph showing average falls per lap vs. number of training data points for different methods: DAgger (β = 1, α = 1), SMILE (α = 0.1), and Supervised. The graph illustrates the performance improvement with increasing training data.]
DAGGER (Dataset Aggregation): An Extension of Supervised Learning
Agenda

- Motivation
- Interactive Reinforcement Learning
- Apprenticeship Learning via Supervised Learning
- Apprenticeship Learning via IRL
- State-of-the-Art
- Proposals for future Applications
Markov Decision Process

- 5-tuple: \((S, A, P_a, R_a, \gamma)\)
  - \(S\): Set of possible states
  - \(A\): Set of possible actions (per state: \(A_s\))
  - \(P_a\): State transition probabilities
  - \(R_a\): Reward function for a given action (and possibly a state)
  - \(\gamma\): Discount factor for rewards in the future
Inverse Reinforcement Learning

![Diagram of Inverse Reinforcement Learning](image)
Inverse Reinforcement Learning - Reward

\[ \mathbb{E}_{s_0 \sim D} [V^\pi(s_0)] = \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi] \]
\[ = \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t) | \pi] \]
\[ = w \cdot \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi] \]

This gives us the feature expectations \( \mu(\pi) \) for a policy:
\[ \mu(\pi) = \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi] \]

\( V \) is the value of the policy
\( \phi(s_t) \) is the vector of different features we’d like to trade off in the reward function
Inverse Reinforcement Learning – Expert trajectories

- We have $m$ expert trajectories (samples from an expert): $\{s_0^{(i)}, s_1^{(i)}, \ldots\}_{i=1}^m$
- Empirical estimate for the expert’s feature expectations $\mu_E = \mu(\pi_E)$:

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{\infty} \gamma^t \phi(s_t)$$
Inverse Reinforcement Learning – Reward function approximation

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.

2. Compute $t^{(i)} = \max_{w: \|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T (\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of $w$ that attains this maximum.

3. If $t^{(i)} \leq \epsilon$, then terminate.

4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.

5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.

6. Set $i = i + 1$, and go back to step 2.
Gridworld

![Graph showing performance of expert vs. log10 (number of sample trajectories)]

- *iri only non-zero weight features*
- *iri all features*
- *parameterized policy stochastic*
- *parameterized policy majority vote*
- *mimic the expert*
Simple Driving Simulator

Image of a driving simulator interface with controls for auto-pilot, manual, off-road, left, middle, right, and center settings.
Simple Driving Simulator
Simple Driving Simulator
Simple Driving Simulator

Expert

Learned
Agenda

- Motivation
- Interactive Reinforcement Learning
- Apprenticeship Learning via Supervised Learning
- Apprenticeship Learning via IRL
- State-of-the-Art
- Proposals for future Applications
Apprenticeship Learning in AD

Safety

Reliability

Comfortable User Experience
Different Driving Styles

Comfort is subjective
How it works

- Velocity
- Acceleration
- Normal Acceleration
- Distance to other agents
- Distance to desired lane
- Jerk
- Normal Jerk
- Desired Speed

Maximum Entropy Inverse Reinforcement Learning
Maximum Entropy Inverse Reinforcement Learning

Mapping through reward weights $\theta$

$$\text{reward}(f_\zeta) = \theta^T f_\zeta = \sum_{s_j \in \zeta} \theta^T f_{s_j}$$

Ziebart et. al. (2008)
Maximum Entropy Inverse Reinforcement Learning

Problem: calculating reward weights is ambiguous:
There can be many different reward weights that lead to optimal trajectories

\[
\text{reward}(f_{\zeta}) = \theta^T f_{\zeta} = \sum_{s_j \in \zeta} \theta^T f_{s_j}
\]

Ziebart et. al. (2008)
Maximum Entropy Inverse Reinforcement Learning

Constraint: match feature expectations

\[ \sum_{\text{Path } \zeta_i} P(\zeta_i) f_{\zeta_i} = \tilde{f} \]

Many different distributions of paths match feature counts, when any demonstrated behaviour is suboptimal.
Maximum Entropy Inverse Reinforcement Learning

Principle of Maximum Entropy:

Choose the distribution that does not match any additional preferences beyond matching feature expectations.
Maximum Entropy Inverse Reinforcement Learning

For deterministic MDPs:

\[
P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^T f_{s_j}}
\]

Plans with equivalent rewards have equal probabilities

For non-deterministic MDPs:

\[
P(\zeta|\theta, T) = \sum_{o \in T} P_T(o) \frac{e^{\theta^T f_{\zeta}}}{Z(\theta, o)} I_{\zeta \in o}
\]
Maximum Entropy Inverse Reinforcement Learning

Maximize the entropy of the distribution subject to feature constraint from observed data

\[ \theta^* = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \sum \log P(\tilde{z}|\theta, T) \]

Maximize Likelihood of observed data unter maximum entropy distribution
The results

Average acceleration (left) and jerk (right)

Demonstrated Trajectory

Initial Guess

Learned Trajectory
Helicopter Acrobatics through Apprenticeship Learning
Helicopter Acrobatics through Apprenticeship Learning

Helicopter Model

\[
\begin{align*}
\dot{u} &= v r - w q + A_x u + g_x + w_u, \\
\dot{v} &= w p - u r + A_y v + g_y + D_0 + w_v, \\
\dot{w} &= u q - v p + A_z w + g_z + C_4 u_4 + D_4 + w_w, \\
\dot{p} &= q r (I_{yy} - I_{zz}) / I_{xx} + B_x p + C_1 u_1 + D_1 + w_p, \\
\dot{q} &= p r (I_{zz} - I_{xx}) / I_{yy} + B_y q + C_2 u_2 + D_2 + w_q, \\
\dot{r} &= p q (I_{xx} - I_{yy}) / I_{zz} + B_z r + C_3 u_3 + D_3 + w_r.
\end{align*}
\]
Helicopter Acrobatics through Apprenticeship Learning

The best 5 out of 10 examples of the human expert were chosen for each maneuver.

Inputs for learning the trajectory.
Helicopter Acrobatics through Apprenticeship Learning

Coloured line: demonstration

Black line: learned trajectory
Helicopter Acrobatics through Apprenticeship Learning
Agenda

Motivation

Interactive Reinforcement Learning

Apprenticeship Learning via Supervised Learning

Apprenticeship Learning via IRL

State-of-the-Art

Proposals for future Applications
Apprenticeship Learning for Driving Behaviour

- Poisslbe interface for passengers to select their preferred driving behavior
- Allows tradeoffs of ecologic factors, speed and maybe even security without explicitly stating them
- Would require to have reinforcement learning as a primary tool for driving autonomous cars
... as a use case for emergency situation behavior

- Treatment of edge cases that are not learnable with traditional methods (without significant problems):
  - Collision Avoidance
  - Ambulance on the road
Apprenticeship Learning for Comfort Systems

Learning of air conditioning behaviours from observation of human driver

- Outside temperature
- Inside temperature
- Humidity
- Ventilation strength
- Air temperature
- Ventilation type
... as a use case for non-driving safety systems

Hazard warning lights

Feature inputs: speed, acceleration, camera input, ...

Binary output: flash / don’t flash
... as a use case for non-driving safety systems

Automatic high beam options could be improved through apprenticeship learning.

Better recognition of when to turn on the high beam by observing the human driver.

Camera Input, possibly other sensors (e.g. LIDAR)
... as a use case for non-driving safety systems

Collect data from many drivers for many driven kilometers

Let the car learn from human drivers when it should use the horn to warn others
Conclusion

Inverse Reinforcement Learning is a great opportunity for training human-like agents

Interactive Reinforcement Learning can help with problems in current RL approaches

Overall: human input can prove beneficial for training and deployment of RL agents