A cascaded supervised learning approach to inverse reinforcement learning

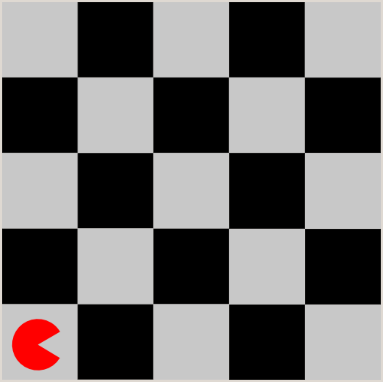
JFPDA2013

Edouard Klein[†‡], Bilal Piot[††], Matthieu Geist[†] and Olivier Pietquin[††]
firstname.lastname@supelec.fr

†Equipe IMS/MaLIS (Supélec), France
‡Equipe ABC UMR 7503 (Loria-CNRS), France
†UMI 2958 (GeorgiaTech-CNRS)

01-02 Juillet 2013

Imitation: Expert



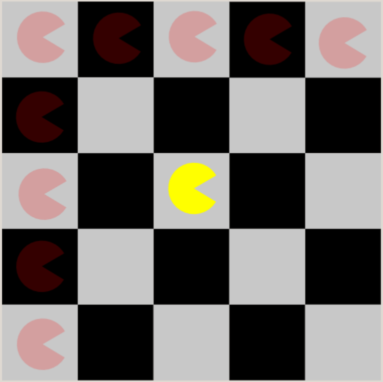
Expert

- The expert is an optimal agent in an MDP
- Its behavior is observed

Apprenticeship learning

Reward inference

Imitation: Expert



Expert

- The expert is an optimal agent in an MDP
- Its behavior is observed

Apprenticeship learning

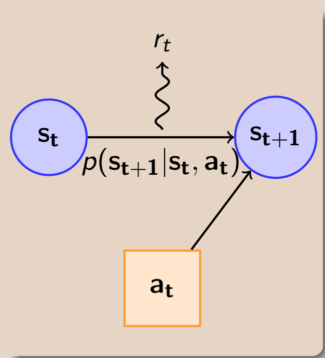
Reward inference

Contribution

CSI

- CSI Algorithm
  - Vanilla classification step. . .
  - . . . followed by a regression step that introduces the temporal structure of the MDP
  - Only needs data from the expert (if we use the heuristics)
  - Can use other data if available
  - Able to use off-the-shelf components
- Theoretical results
- Experimental results

Quick definitions



Notions

- State $s_t \in \mathcal{S}$
- Action $a_t \in \mathcal{A}$
- Reward $r_t = R(s_t) \in \mathbb{R}$
- Transition $(s_t, a_t, s_{t+1}, r_t) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R}$
- $\pi : \mathcal{S} \to \mathcal{A}$

Markovian criterion

Past states are irrelevant

RL problem and solution

Value function

$$V_R^\pi(s_t) = E\left[\sum_i \gamma^i R(s_{t+i}) \middle| \pi\right] \tag{1}$$

Goal

Optimal policy $\pi_R^* = \arg\max_\pi V_R^\pi$ $\qquad \pi_R^*(s) = \arg\max_a Q_R^{\pi^*}(s, a)$

IRL problem

Goal

Finding the reward R so that the observed behavior is optimal

Ill-posed

The null reward $\forall s, R(s) = 0$ is a solution

A certain class of classifiers

Score function based classifiers

- Classifier: map inputs $s \in \mathcal{S}$ to labels $a \in \mathcal{A}$
- Data: $D_{sa}^{\pi_E} = \{(s_i, a_i)_{1 \leq i \leq N}\}$
- Decision rule : $\pi^C \in \mathcal{A}^{\mathcal{S}}$
- Score function : $\pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s, a)$
- Very few exceptions (*e.g.* decision trees)

The idea behind CSI

Score function based classifiers

$$\pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s, a)$$

Expert policy

$$\pi_E(s) = \arg\max_a Q^{\pi_E}(s, a)$$

Bellman Equation for the expert

$$Q^{\pi_E}_{R^E}(s, a) = R^E(s, a) + \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s, a\right) Q^{\pi_E}_{R^E}(s', \pi_E(s')) \quad (3)$$

The idea behind CSI

Score function based classifiers

$$\pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s, a)$$

Expert policy

$$\pi_E(s) = \arg\max_{a} Q^{\pi_E}(s, a)$$

Bellman Equation for the expert

$$R^E(s, a) = Q^{\pi_E}_{R^E}(s, a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s, a\right) Q^{\pi_E}_{R^E}(s', \pi_E(s')) \tag{3}$$

The idea behind CSI

We view q as a quality function

$$R^C(s,a) = q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s,a\right) q(s', \pi^C(s')) \tag{2}$$

π^C is optimal for R^C and $\pi^C \approx \pi_E$, ergo we would be happy to find R^C.

Score function based classifiers

$$\pi^C(s) \in \arg\max_{a \in \mathcal{A}} q(s,a)$$

Expert policy

$$\pi_E(s) = \arg\max_a Q^{\pi_E}(s,a)$$

Bellman Equation for the expert

$$R^E(s,a) = Q^{\pi_E}_{R^E}(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s,a\right) Q^{\pi_E}_{R^E}(s', \pi_E(s')) \tag{3}$$

The idea behind CSI

After a classifier has learned a score function q

$$R^C(s,a) = q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s,a\right) q(s', \pi^C(s')) \tag{4}$$

Non expert data

$$D_{sas}^{\sim} = \{s_i, a_i, s_i'\}_{0 \le i \le N}. \tag{5}$$

Sampled version of Eq. 7

$$\hat{r}_i = q(s_i, a_i) - \gamma q(s_i', \pi^C(s_i')). \tag{6}$$

CSI Pseudo-code

Algorithm 1: CSI algorithm

Given a training set $D_{sa}^{\pi_E} = \{(s_i, a_i = \pi_E(s_i))\}_{1 \leq i \leq N}$
and another training set $D_{sas}^{\sim} = \{(s_j, a_j, s'_j)\}_{1 \leq j \leq N'}$;
Train a score function-based classifier on $D_{sa}^{\pi_E}$, obtaining decision rule π^C
and score function $q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$;
Learn a reward function $\hat{R}^C$ from the dataset $\{((s_j, a_j), \hat{r}_j)\}_{1 \leq j \leq N'}$,
$\forall (s_j, a_j, s'_j) \in D_{sas}^{\sim}, \hat{r}_j = q(s_j, a_j) - \gamma q(s'_j, \pi_C(s'_j))$;
Output the reward function $\hat{R}^C$;

Heuristics

After a classifier has learned a score function q

$$R^C(s,a) = q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s,a\right) q(s', \pi^C(s')) \tag{7}$$

Non expert data

$$\cancel{D_{sas}^{\sim} = \{s_i, a_i, s'_i\}_{0 \leq i \leq N}} \tag{8}$$

Expert data

$$D_{sas}^{\pi_E} = \{(s_i, a_i, s'_i)_{1 \leq i \leq N}\}$$

Sampled version of Eq. 7

$$\left(s_i, \pi_E(s_i)\right), \hat{r}_i = q(s_i, \pi_E(s_i)) - \gamma q(s'_i, \pi^C(s'_i)). \tag{9}$$

Heuristics

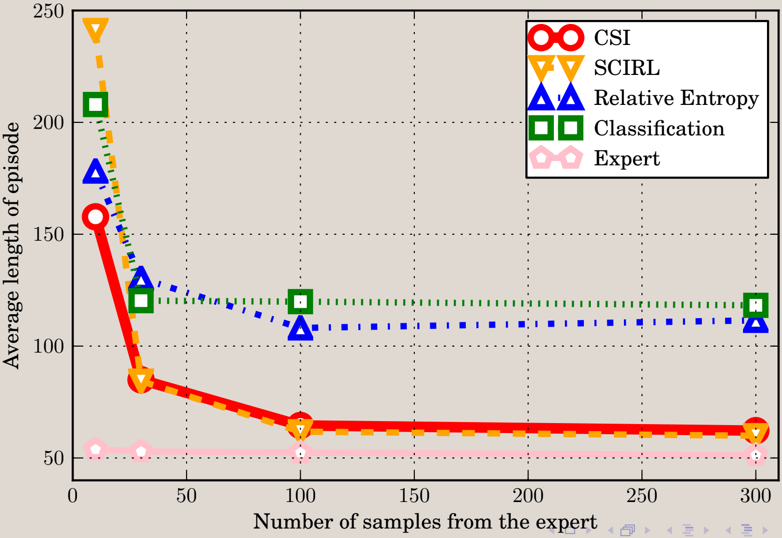
$$\left(s_i, \forall a \neq \pi_E(s_i)\right), \hat{r}_{min} = \min_{i \in [\![1;N]\!]} \hat{r}_i - 1. \tag{10}$$

Error bound

Theorem

$$0 \leq \mathbf{E}\left[V_{\hat{R}^C}^{\pi^*_{\hat{R}^C}}(s) - V_{\hat{R}^C}^{\pi_E}(s) \middle| s \sim \rho_E \right] \leq \frac{1}{1-\gamma}\left(\epsilon_C \Delta q + \epsilon_R (1 + C_{\pi^*_{\hat{R}^C}})\right). \tag{11}$$

Results on the mountain car

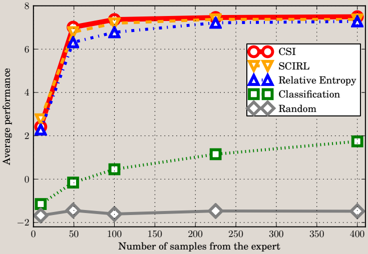


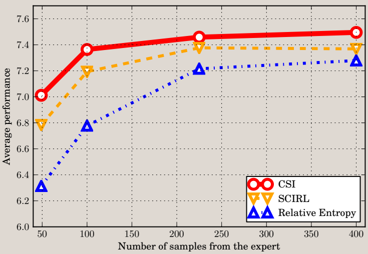
Results on the driving problem

Description

- Widespread benchmark
- Goal of the expert : avoid other cars, do not go off-road, go fast
- Using only data from the expert and natural features

Results on the driving problem





Description

- Widespread benchmark
- Goal of the expert : avoid other cars, do not go off-road, go fast
- Using only data from the expert and natural features

Possible future work

Real world problems

The difficult part is solving the MDP once the reward has been found by CSI

Thank you. . .

. . . for your attention