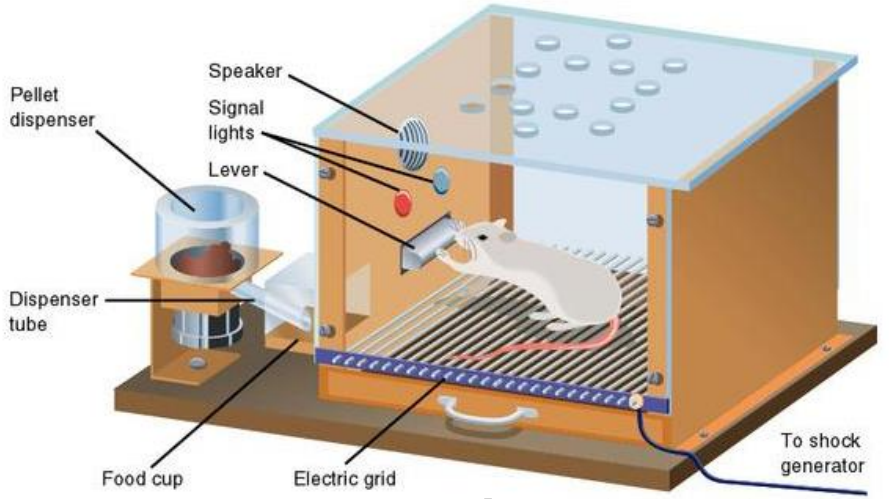
Introduction to inverse reinforcement learning

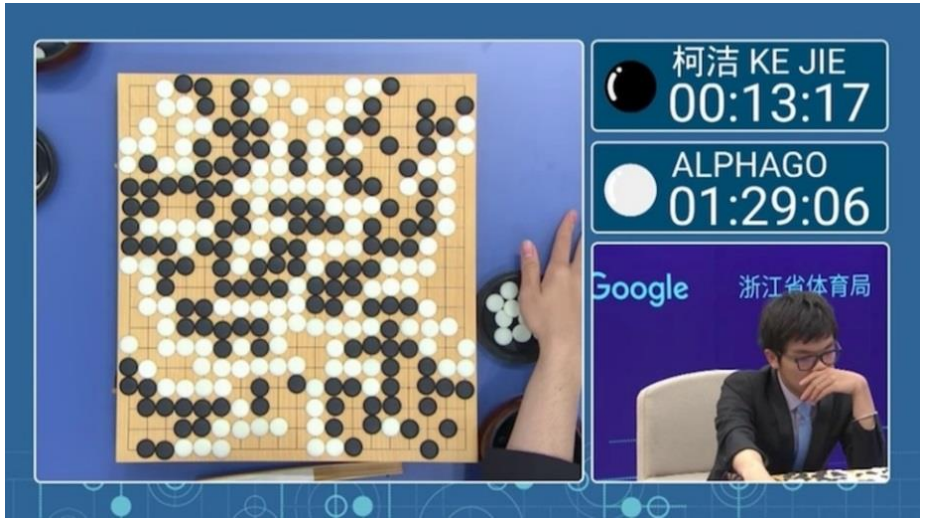
Eiji Uchibe
Dept. of Brain Robot Interface
ATR Computational Neuroscience Labs.

What is Reinforcement Learning (RL)?

- RL is a computational framework to find an optimal policy (controller) by trial and error
- Inspired from psychology
  - Thorndike's law of effect
  - Skinner's principle of reinforcement
- The most famous application is AlphaGo that has defeated top Go masters from across the world
- Learn behaviors from rewards designed by each trainer



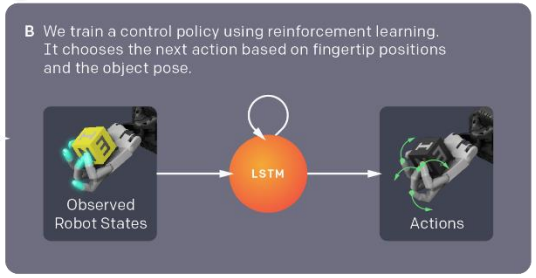
[Simply Psychology]

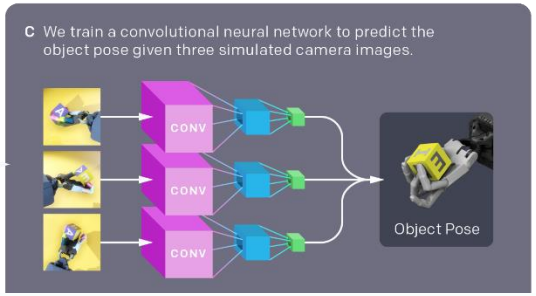


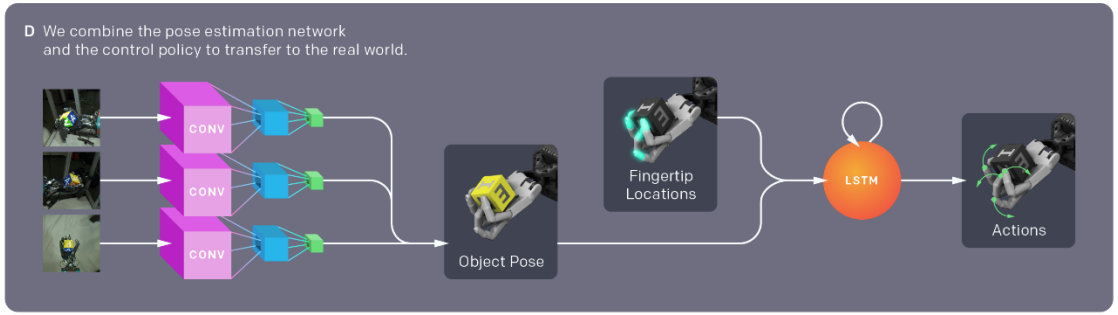
[gizmodo.jp]

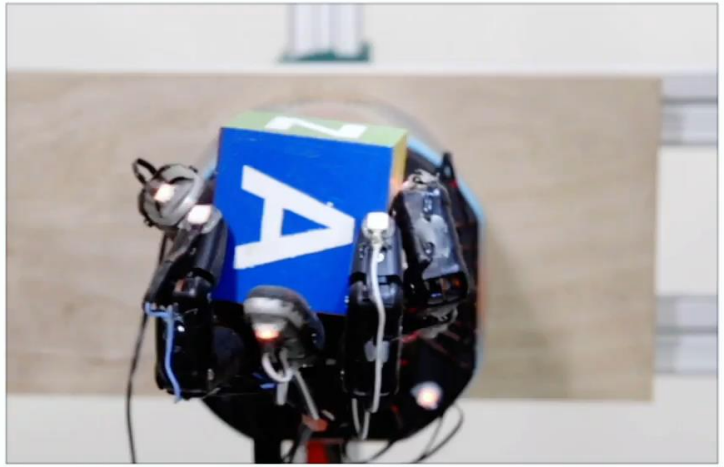
In-hand manipulation

[Andrychowicz et al., 2018]

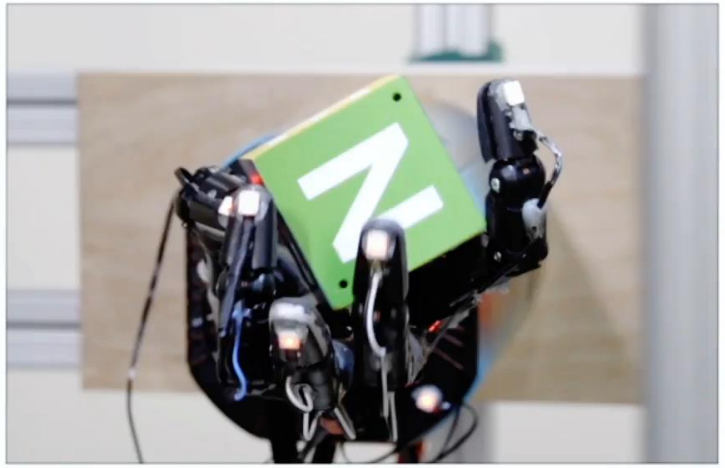




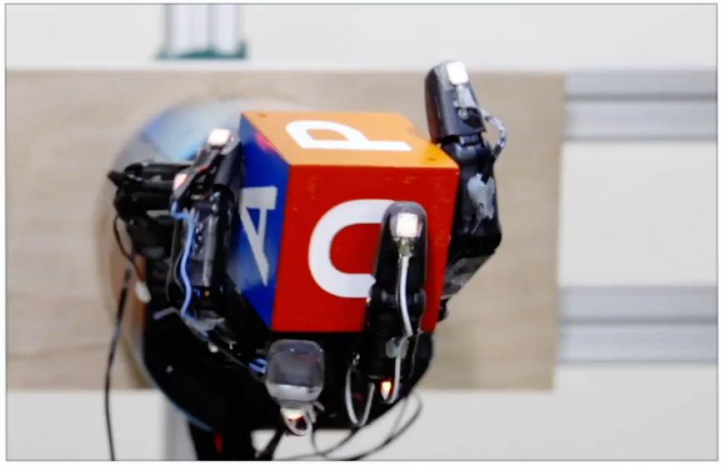




FINGER PIVOTING



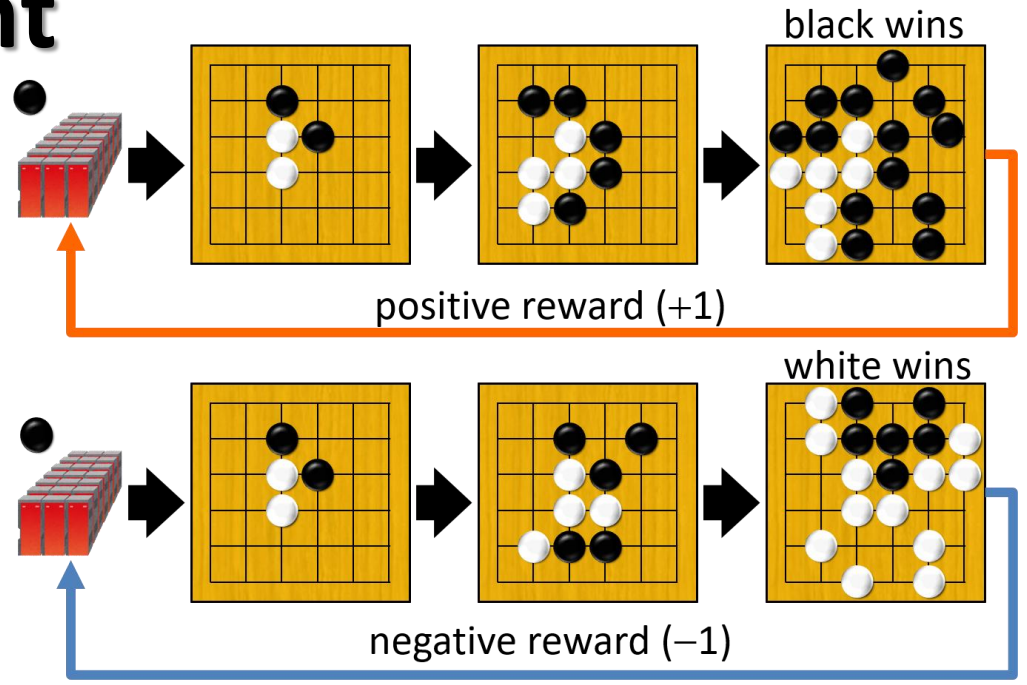
SLIDING



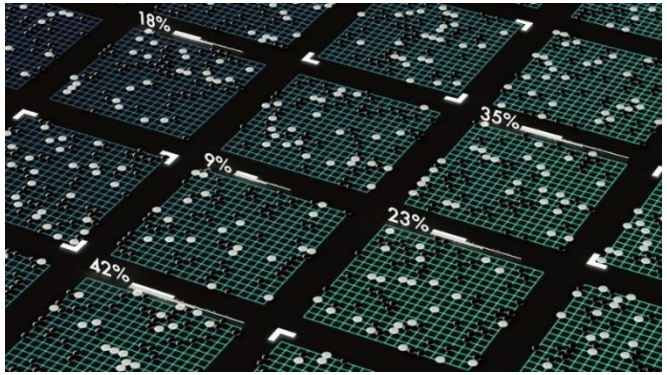
FINGER GAITING

Designing Reward is important

- In the case of Go
  - positive reward for winning
  - negative reward for losing
  - zero otherwise
- AlphaGo Zero, which does not use a record of a game of go, needs **4.9 million** games of self-play

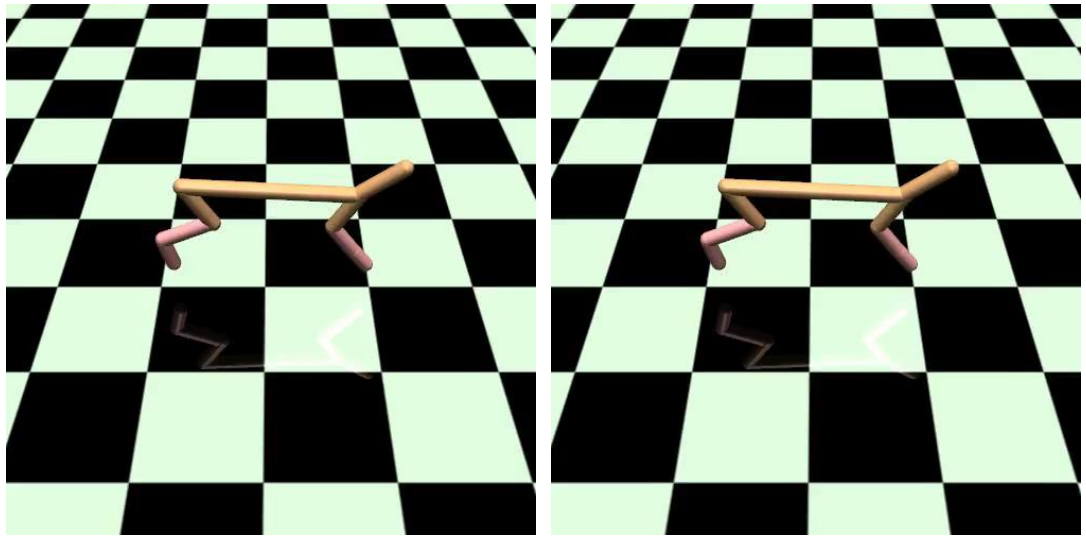


- Deep RL is applicable when we can collect samples by using multiple simulators
- What happens if we use a dense reward?



But designing reward is difficult …

- Task: move forward as fast as possible (continuous state-action problem)



immediate reward

$$r(\boldsymbol{s}, \boldsymbol{a}) = \underbrace{v_x}_{\text{forward velocity}} - 0.05 \underbrace{\|\boldsymbol{a}\|_2^2}_{\text{squared norm of applied torque}}$$

- Even if the reward function is well-shaped, it is not enough to find an optimal policy when learning time is limited
- Inverse RL provides the method to design the reward from behaviors of experts

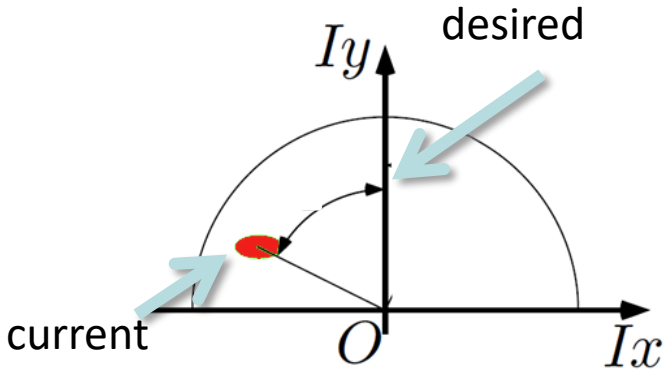
[Deep Reinforcement Learning Doesn't Work Yet]

Designing Reward is Difficult

- Task: catch a battery pack
- Two reward functions: r_{orig} and r_{aug}

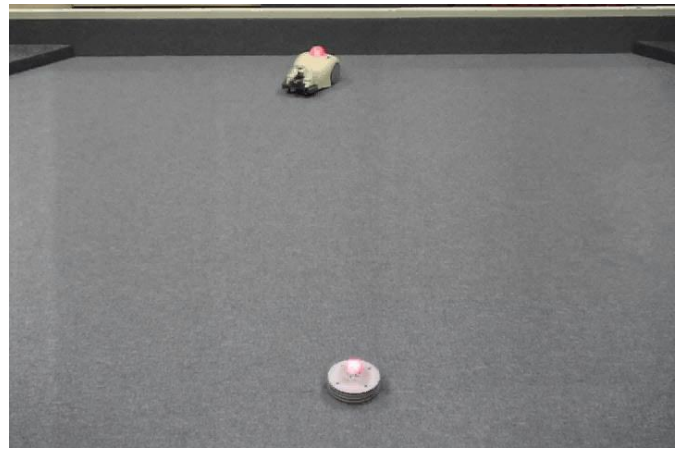
$$r_{\text{orig}} = \begin{cases} +1 & \text{if catching a battery pack} \\ -0.05 & \text{if moving} \\ 0 & \text{otherwise} \end{cases}$$

$$r_{\text{aux}} = \exp\left(-\frac{(\theta - \theta_d)^2}{2\sigma^2}\right)$$

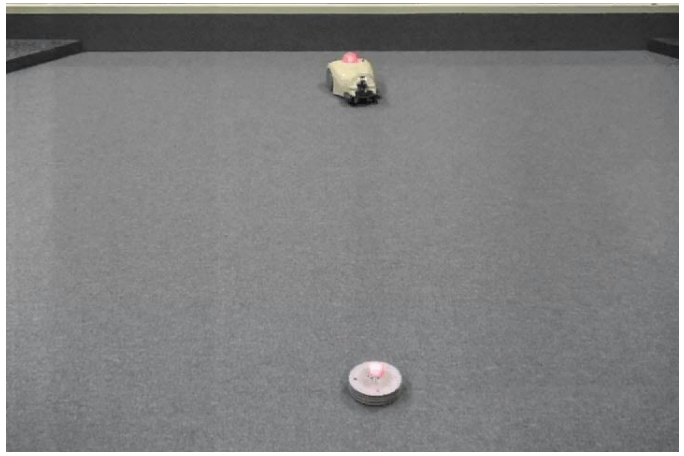


- Watching a battery pack was obtained according to the choice of w although it learned faster

Trained with r_{orig}

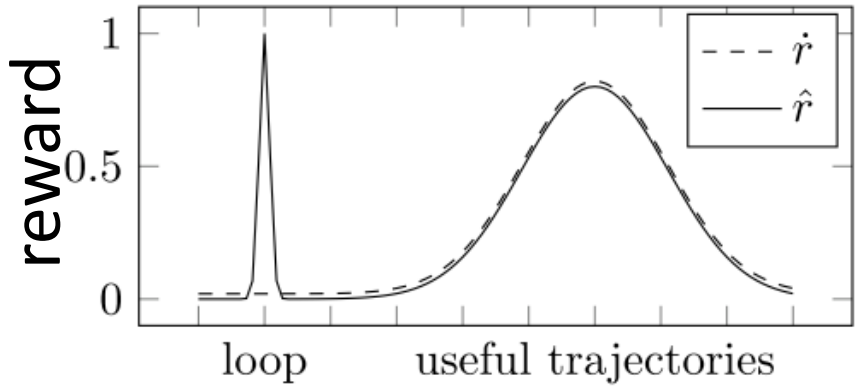


Trained with $r_{\text{orig}} + wr_{\text{aux}}$



Designing Reward is Difficult

- Task: finish the boat race quickly
  - not directly reward the player's progression around the course
  - get rewards by hitting targets laid out along the route



[Everitt et al., 2017]

  - $\dot{r}$: true reward
  - $\hat{r}$: corrupt, observed reward

https://www.youtube.com/watch?v=tlOIHko8ySg

Reward function for flipping a handkerchief

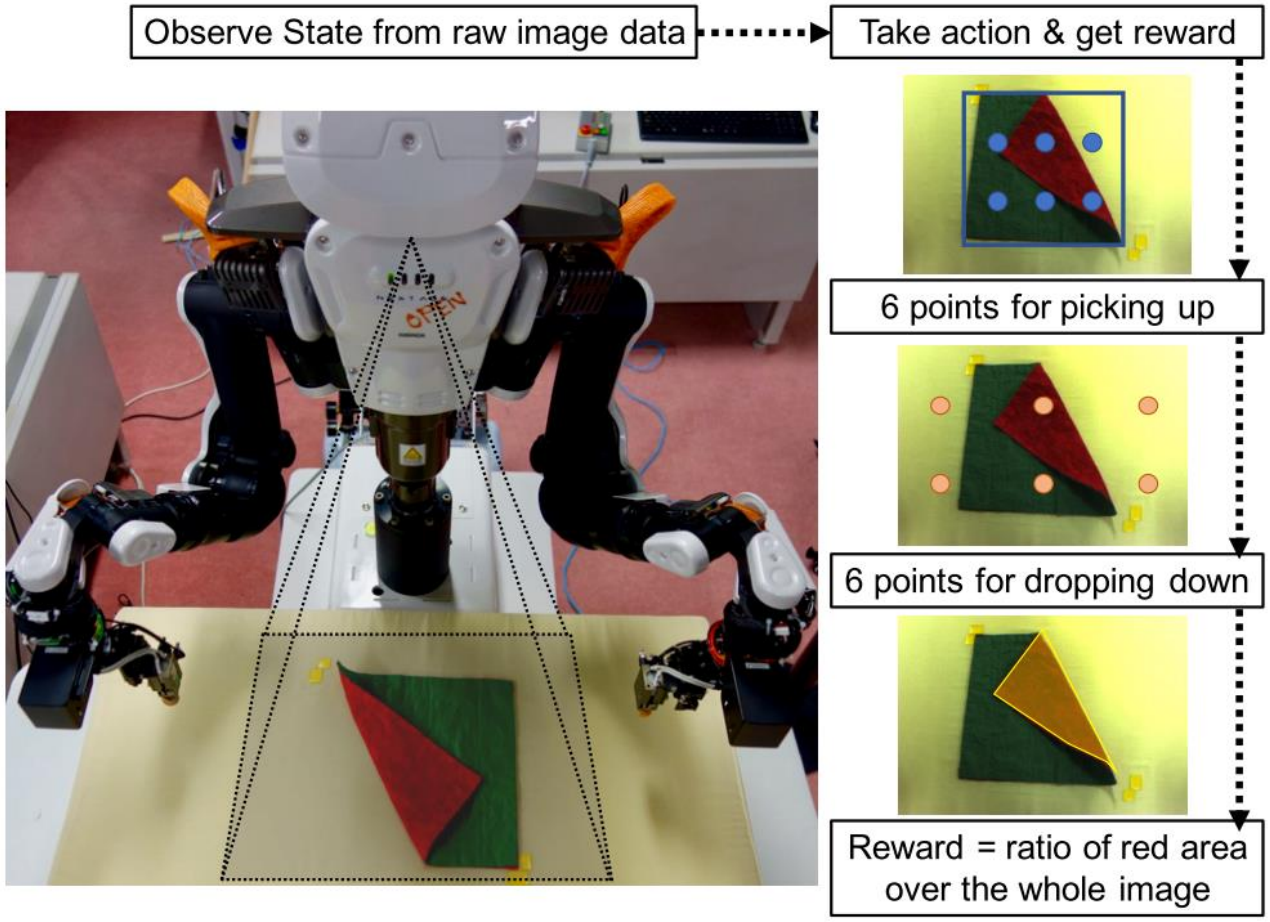
- Entropy-regularized reinforcement learning (Deep Dynamic Policy Programming)
- No simulator

Deep Dynamic Policy Programming for Robot Control with Raw Images

Yoshihisa Tsurumine, Yunduan Cui, Takamitsu Matsubara and Eiji Uchibe

Intelligent System Control Laboratory
Graduate School of Information Science
Nara Institute of Science and Technology

Department of Robot Brain Interface
Advanced Telecommunications
Research Institute



Tsurumine, Y., Cui, Y., Uchibe, E., and Matsubara, T. (2017). Deep dynamic policy programming for robot control with raw images. In *Proc. of IROS*.

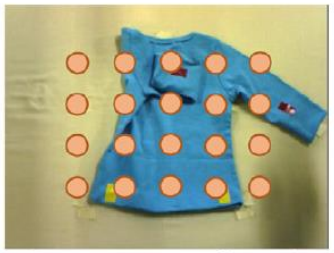
Folding a T-shirt

Table 3
Settings and learning parameters of folding t-shirt task.

(a) Parameter setting of folding t-shirt task

MDP setting	Description
State	The input state is a 84 × 84 px RGB image from the NEXTAGE's integrated camera.
Action	10 × 20 = 200 gripper actions are defined as picking up the t-shirt from 10 points over its current area and dropping it down to 5 × 4 points over the table.

Picking up



Dropping down

Re

Difficult to design a good reward in practice

Sleeve distance from center

+

Hem distance from target point

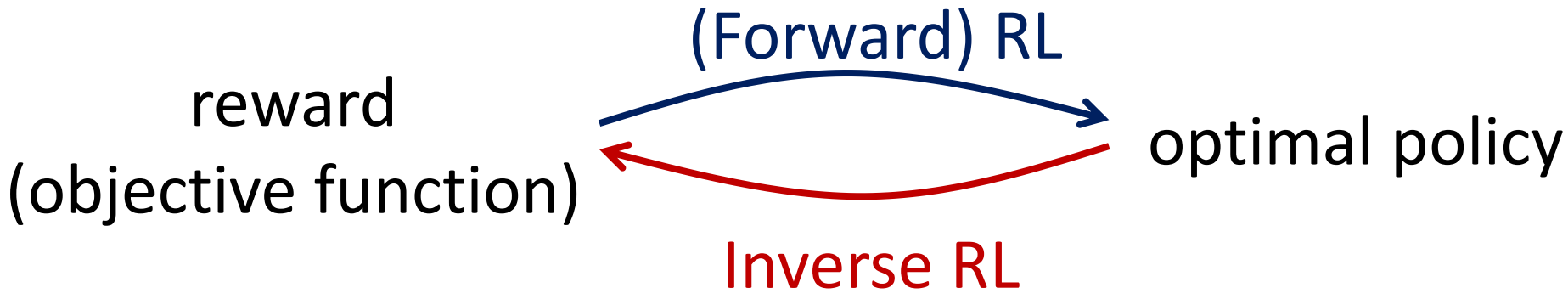
Algorithm 3: Reward function of t-shirt folding task

Initialize $InitHemR = [0.675, 0.8]$, $InitHemL = [0.325, 0.8]$
Initialize $TargetHemR = [0.675, 0.208]$,
$TargetHemL = [0.325, 0.208]$
Function `HemReward(`*SleevePoint, CenterHem*`)`**:**
  Initialize reward = 0
  reward = -Sum(|*SleevePoint* - *CenterHem*|)
  return reward
Function `SleeveReward(`*HemPoint, InitHem, TargetHem*`)`**:**
  Initialize reward = 0
  Initalize *Distance* = |*InitHem* − *TargetHem*|
  reward = Sum(*Distance* − |*HemPoint* − *TargetHem*|)
  return reward
Function `ShirtReward()`**:**
  Initialize reward = 0
  Update color marker
  Get *HemPointR*, *HemPointL*, *SleevePointR*, *SleevePointL*
  if *Detect hem marker* **then**
    CenterHem = (*HemPointR* + *HemPointL*)/2
    reward = `SleeveReward(`*SleevePointR, CenterHem*`)` +
    `SleeveReward(`*SleevePointL, CenterHem*`)`
  else
    reward = 1
  if *Detect sleeve marker* **then**
    reward = reward +
    `HemReward(`*HemPointR, InitHemR, TargetHemR*`)` +
    `HemReward(`*HemPointL, InitHemL, TargetHemL*`)`
  return reward

Tsurumine, Y., Cui, Y., Uchibe, E., and Matsubara, T. (2019). Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. *Robotics and Autonomous Systems*, 112: 72-83.

Inverse Reinforcement Learning (RL)

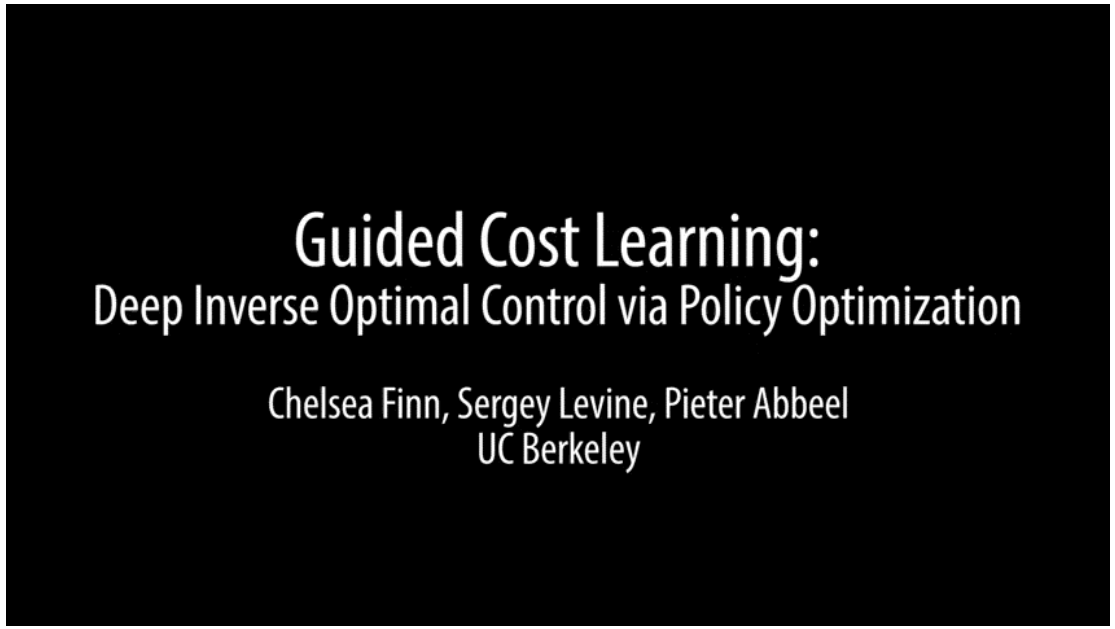
- Estimate a reward function from observed behaviors generated by an optimal policy

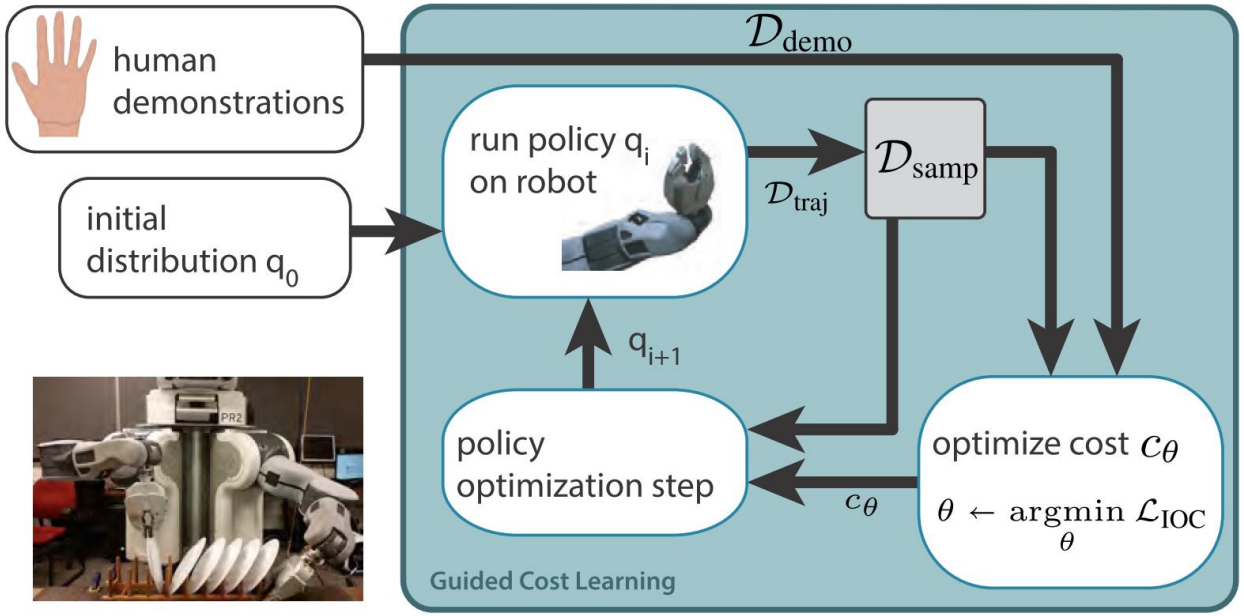


- It is often easy to demonstrate some good behaviors
- Ill-posed problem. That is, the solution is not uniquely determined

Application: Robot control

- Adversarial imitation learning
  - Expert data is provided by kinesthetic demonstration
  - Iteration of reward estimation and policy improvement

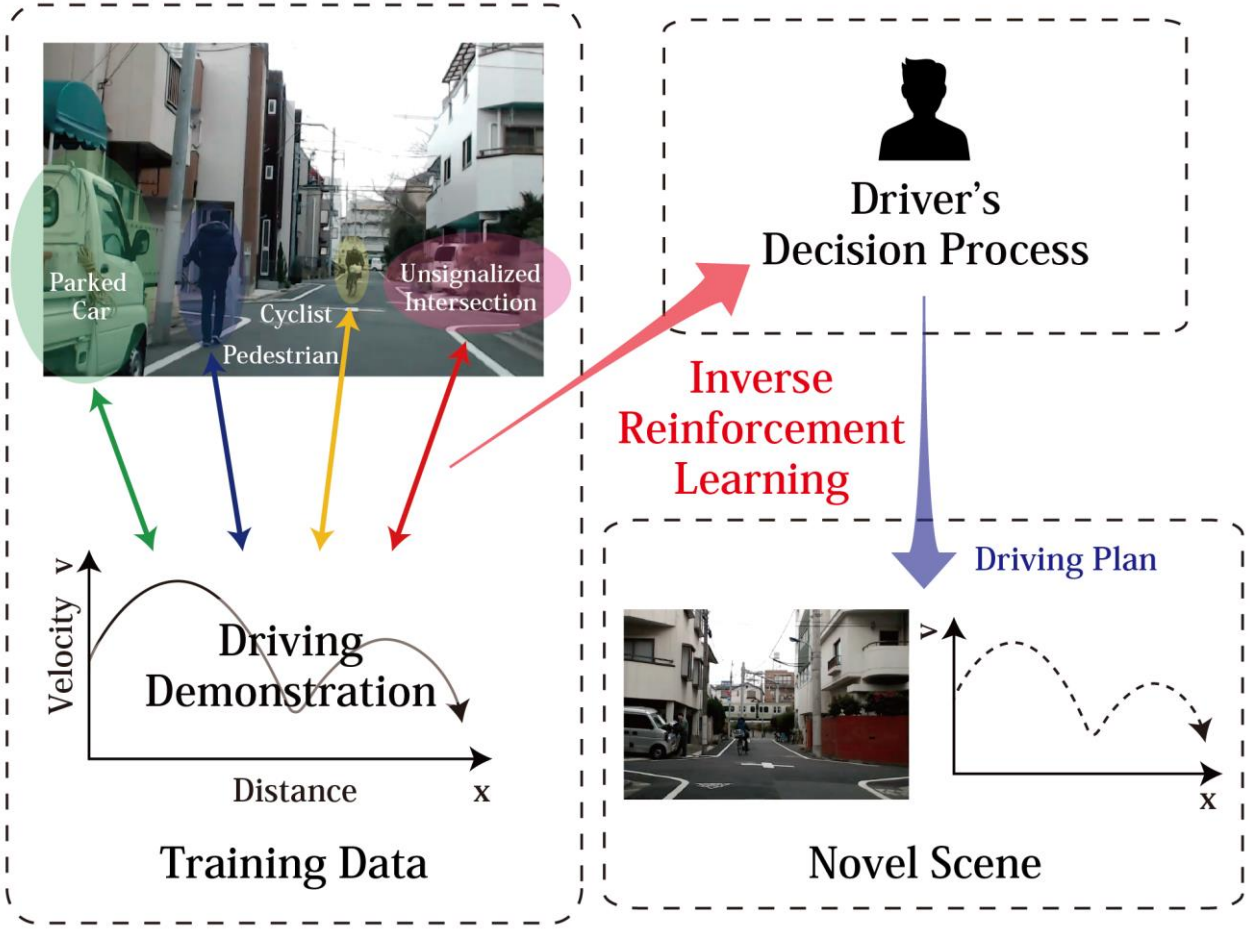




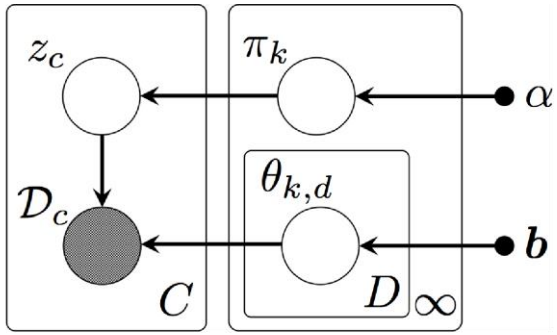
Finn, C., Levine, S., & Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *Proc. of ICML*, 49–58. ICML.

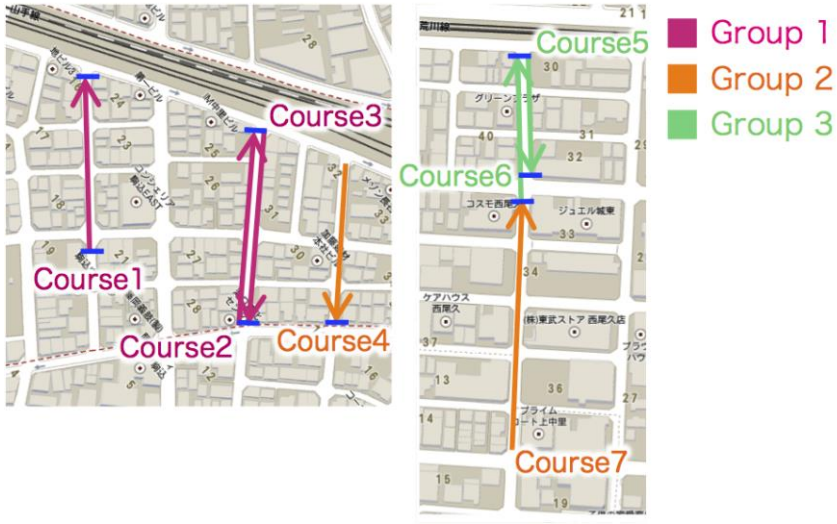
Modeling risk anticipation behaviors

- Estimate a speed control behavior



- Classification

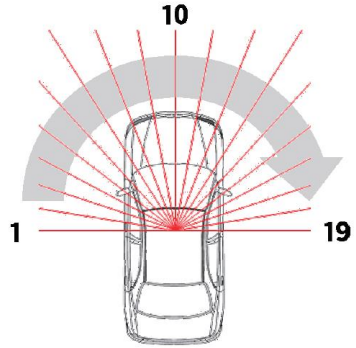




Shimosaka, M., Kaneko, T., & Nishi, K. (2014). Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning. *Proc. of the 17th International IEEE Conference on Intelligent Transportation Systems*, 1694–1700.

Safe autonomous driving

- Estimate the reward to learn the policy that achieves lateral acceleration of 0.3G or less.
- Important reward features
  - track10: distance to the edge of the corner
  - speedZ: Speed of the z-axis direction (to avoid sliding off the track)



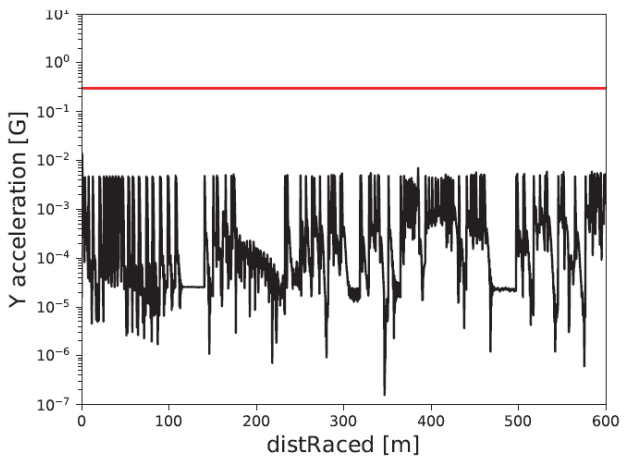


図 7: 推定された報酬によるモデルの横加速度推移

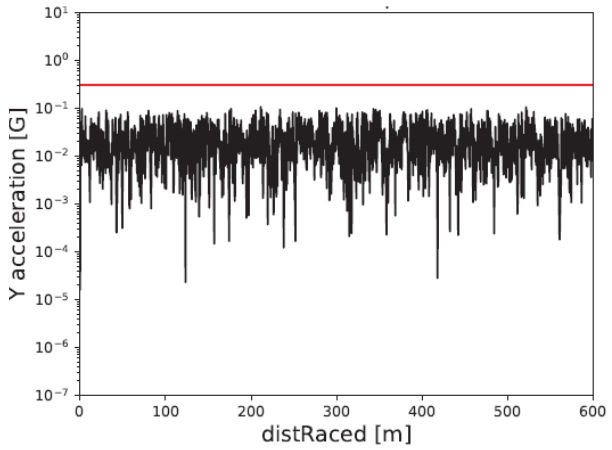
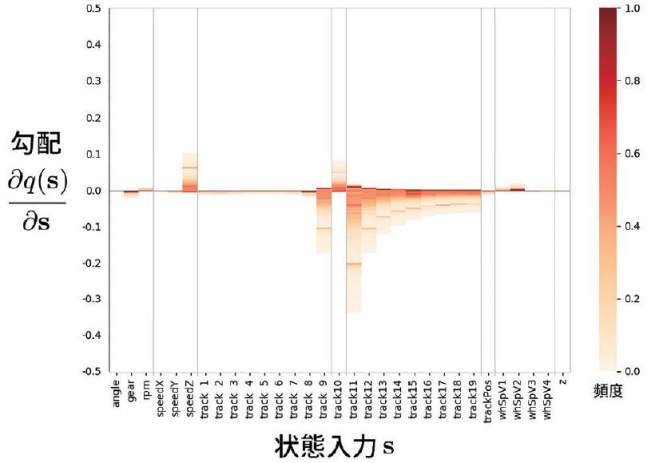


図 5: エキスパート軌跡の横加速度推移



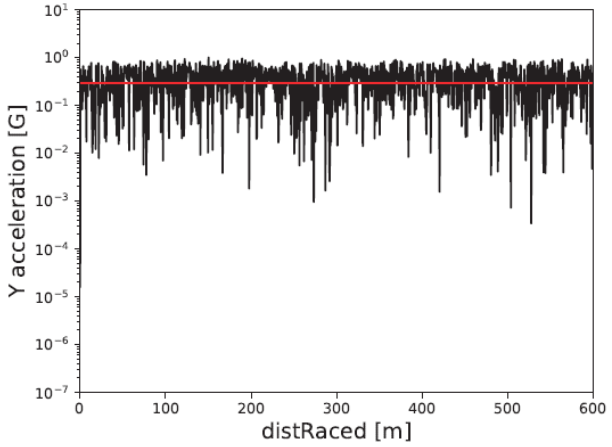
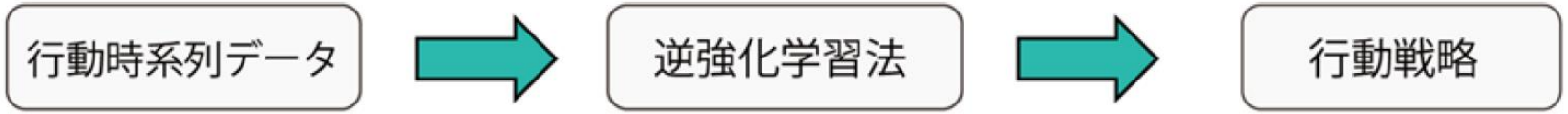


図 6: ベースライン軌跡の横加速度推移

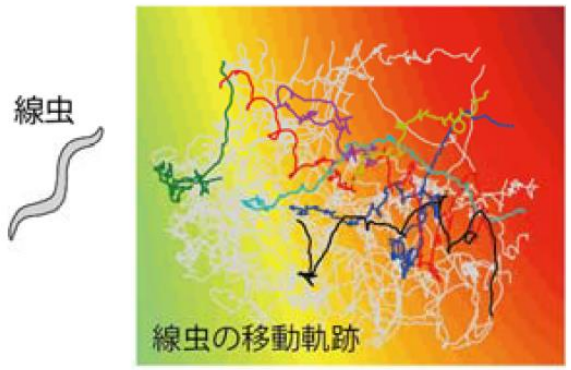
岸川, 荒井. (2019). 深層逆強化学習による自動運転の安心走行実現. 第33回人工知能学会全国大会予稿集.

Investigation of *C. elegans* thermotactic behavior

- Two basic strategies are found
  - Directed Migration (DM): Worms efficiently reached specific temperatures, which explains their thermotactic behavior when fed.
  - Isothermal Migration (IM). Worms moved along a constant temperature, which reflects isothermal tracking, well-observed in previous studies.



温度勾配

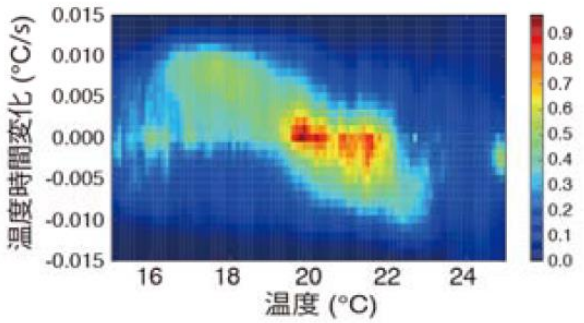


線虫は育成された温度を好むように、また
飢餓を経験した温度を避けるように移動する。

行動戦略を示す数式

$$\pi(s'|s) = \frac{p(s'|s)\exp(v(s'))}{\sum_x p(x|s)\exp(v(x))}$$

推定された価値関数：$v(s)$

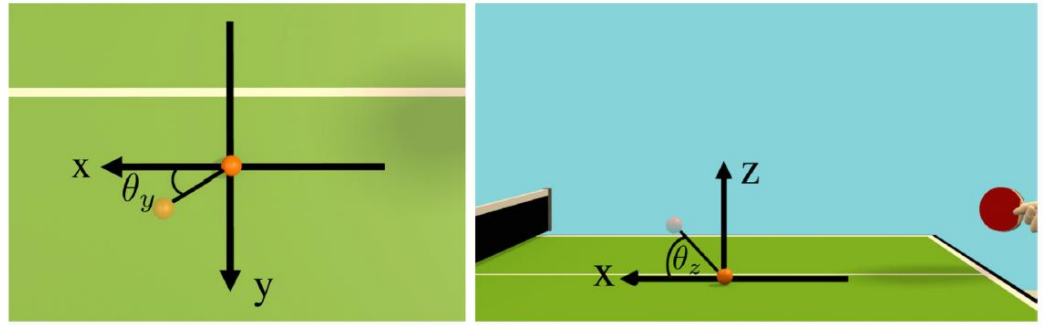


動物が行動していて遭遇する各状況が、戦略上
どれくらいの価値があるのかを示している。

Yamaguchi, S., Honda, N., Ikeda, M., Tsukada, Y., Nakano, S., Mori, I., and Ishii, S. (2018). Identification of animal behavioral strategies by inverse reinforcement learning. PLoS Computational Biology.

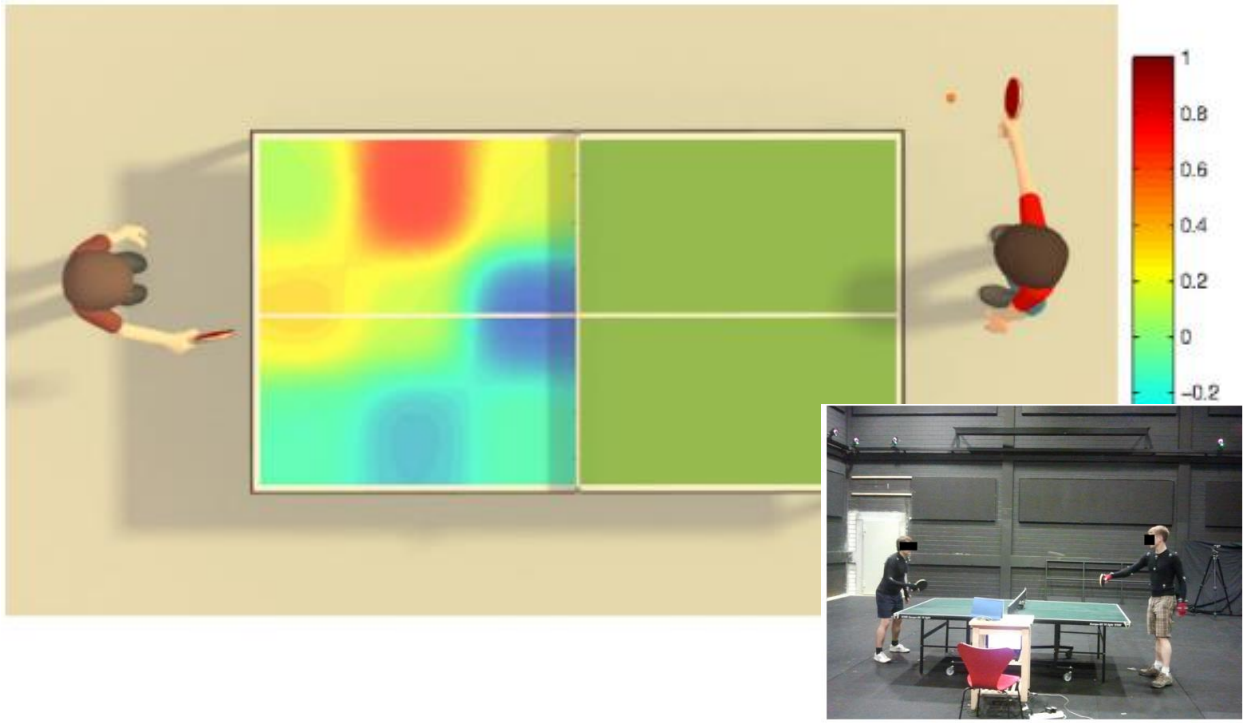
Table tennis

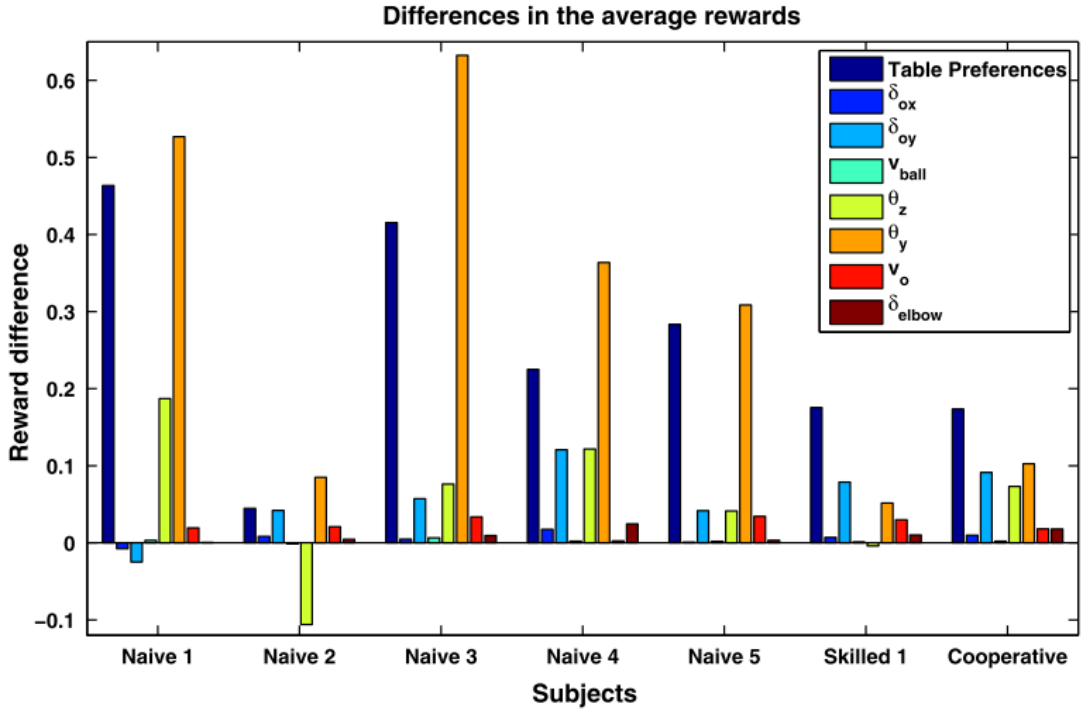
- Reward function for table preferences
- Individual player preferences



(a) θ_y (b) θ_z

Fig. 5 The bouncing angles θ_y and θ_z in the xy- and xz-surface define the orientation of the ball. While θ_z corresponds to the horizontal bouncing angle, θ_y corresponds to the direction of the ball and thereby defines if the ball is played cross to the *left*, cross to the *right* or straight

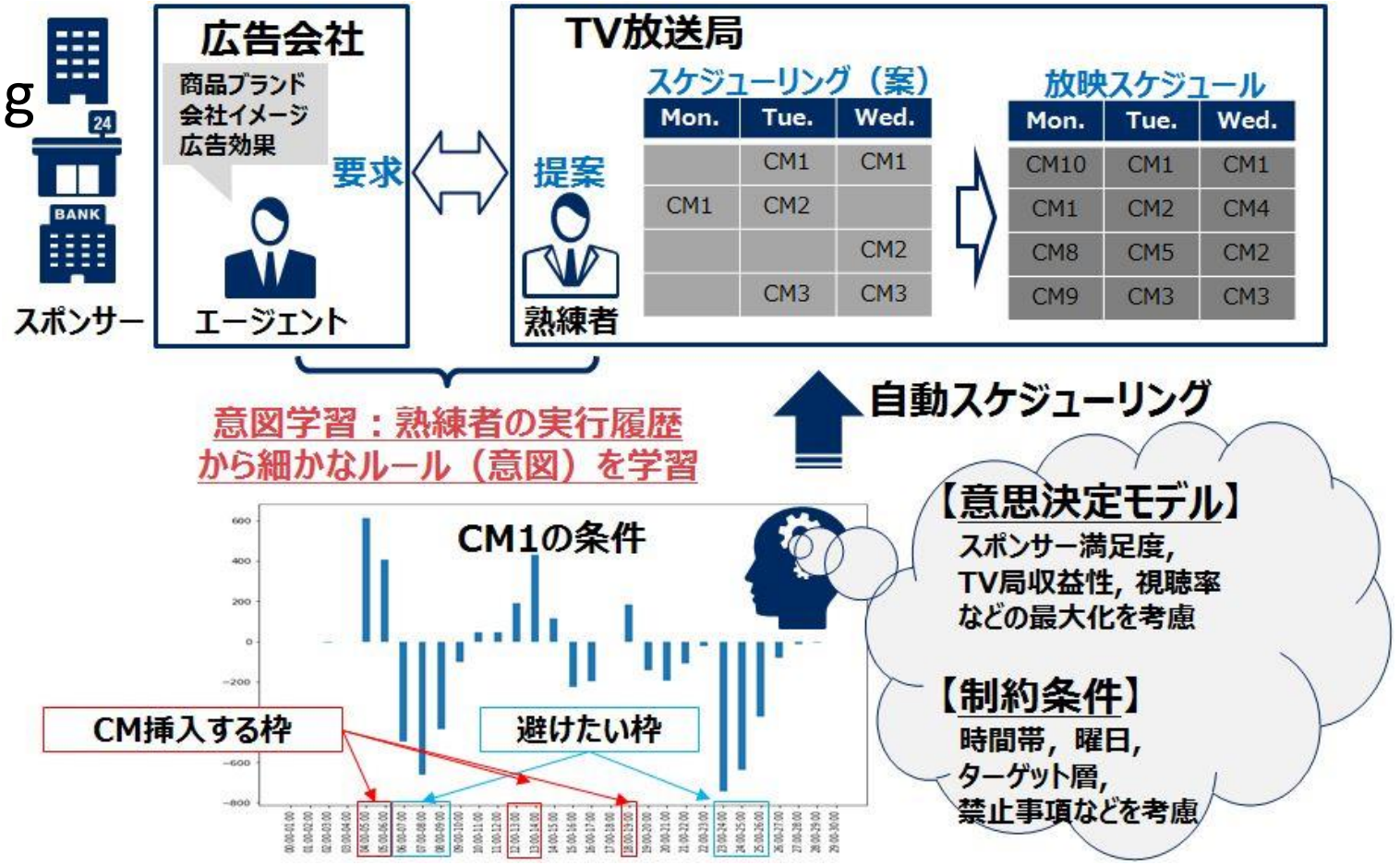




Muelling, K., Boularias, A., Mohler, B., Schölkopf, B., and Peters, J. (2014). Learning strategies in table tennis using inverse reinforcement learning. Biological Cybernetics, 108(5): 603-619.

TV Advertisement Scheduling by Learning Expert Intentions

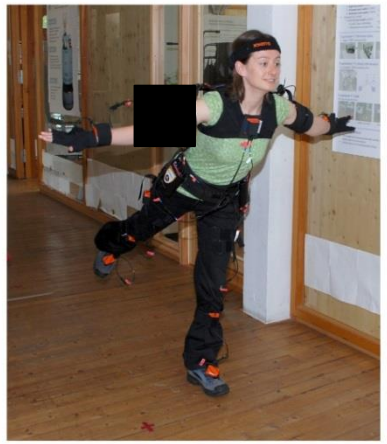
- Imitate the decision-making process of scheduling experts



NECプレスリリース(2019/07/17)

Suzuki, Y., Wee, W.M., & Nishioka, I. (2019). TV Advertisement Scheduling by Learning Expert Intentions. In Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3071–81.

What is Imitation Learning?



- Learn how to act given a set of demonstrations provided by a human expert
- Formulated as Maximum A Posteriori problem

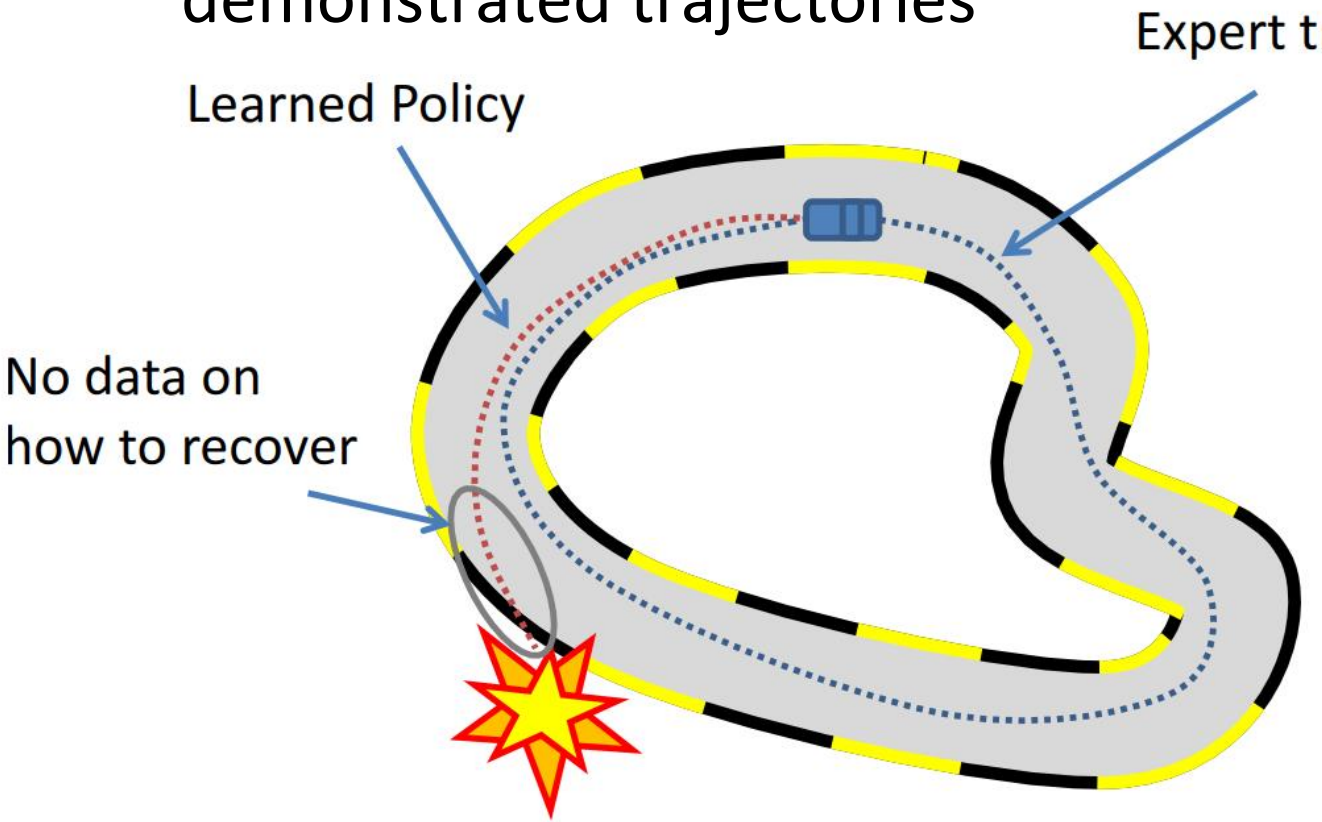
$$\arg\max_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta} \mid \mathcal{D}^E) = \arg\max_{\boldsymbol{\theta}} \Big[\ln p(\boldsymbol{\theta}) + \sum_{(s,a)\in\mathcal{D}^E} \ln \pi(a \mid s, \boldsymbol{\theta}) + \sum_{s\in\mathcal{D}^E} \ln p(s \mid \boldsymbol{\theta})\Big]$$

  - $\pi(a \mid s, \boldsymbol{\theta})$: stochastic policy
  - $p(s)$: state distribution generated by π
- Because the third term is difficult to evaluate, it is often ignored

[Ross and Bagnell, 2010]

Naïve Imitation Learning Works Poorly

- The robot's errors compound when drifting away from the expert's demonstrated trajectories



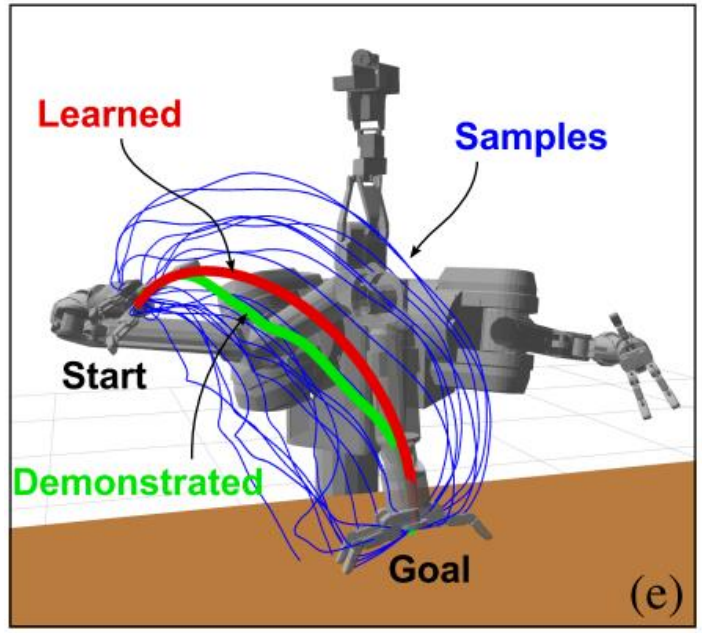
- No mechanism to recover when the generated trajectory deviates from the demonstrated ones

[Ross and Bagnell, 2010]

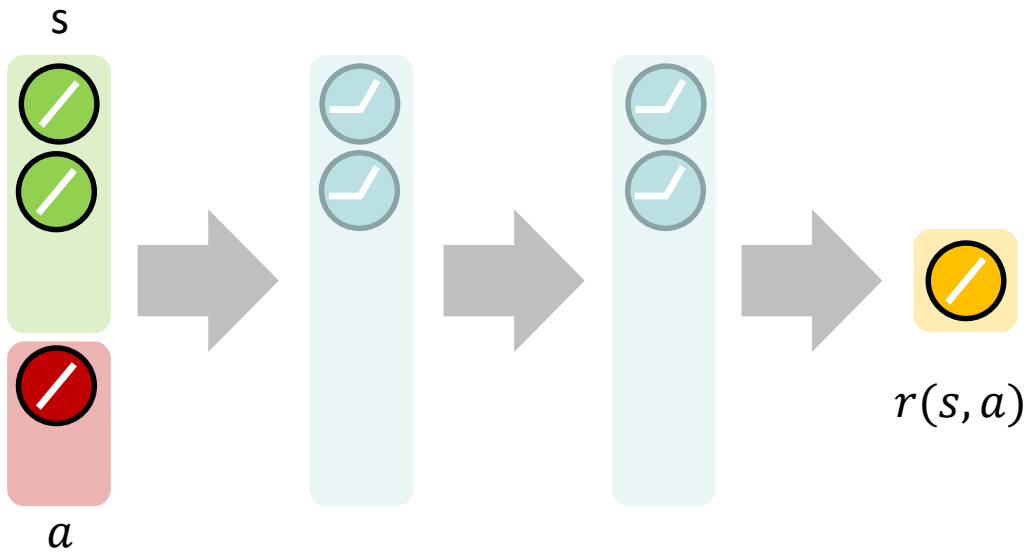
Preparation

- $\tau = \{s_1, a_1, \ldots, s_T, a_T\}$: trajectory of state s and action a
- $r(s, a; w)$: immediate reward at state s and action a parameterized by w
  - For a linear function approximator, $r(s, a; w) = w^\top \phi(s, a)$, where $\phi(s, a)$ is a feature vector
- Total reward along a trajectory τ

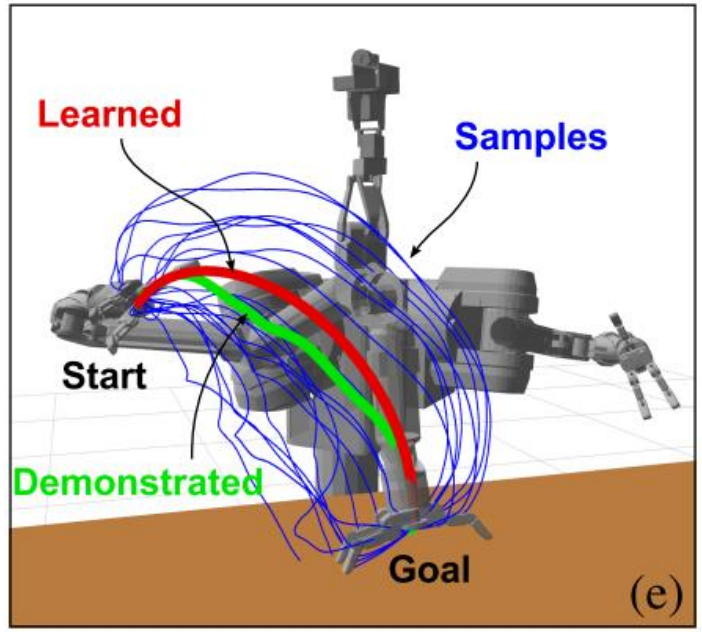
$$R(\tau) = \sum_t r(s_t, a_t; w)$$
$$= w^\top \sum_t \phi(s_t, a_t) = w^\top \phi(\tau)$$



[Kalakrishnan et al., 2013]



Maximum Entropy Inverse Reinforcement Learning



[Kalakrishnan et al., 2013]

- $P(\tau|w)$: probability of τ parameterized by w (blue)
- The expectation of features under $P(\tau|w)$ is equal to the expected empirical feature count

$$\underbrace{\sum_{\tau} P(\tau \mid w)\phi(\tau)}_{\text{expected feature under } P(\tau \mid w)} = \underbrace{\frac{1}{m}\sum_{j} \phi(\tau_j)}_{\text{expected empirical feature count of the expert}} \triangleq \tilde{\phi}$$

- Since many reward weights satisfy the above constraint, the principle of maximum entropy is employed to resolve ambiguities in choosing distributions

Ziebart, B.D., Maas, A., Bagnell, J.A., and Dey, A. (2008). Maximum entropy inverse reinforcement learning. *Proc. of AAAI*, 1433-1438.

Principle of maximum entropy (1/2)

- Choose a 'best' from a number of different probability distributions that all express the current state of knowledge
- Constrained optimization problem

$$\max_p -\sum_\tau p(\tau) \ln p(\tau)$$ ➡ Maximize the entropy of $p(\tau)$

$$\sum_\tau p(\tau)\phi_i(\tau) = \tilde{\phi}_i, \forall i$$ ➡ constraints on the expected feature

$$\sum_\tau p(\tau) = 1,\ p(\tau) \geq 0$$ ➡ $p(\tau)$ is a probability distribution

Ziebart, B.D., Maas, A., Bagnell, J.A., and Dey, A. (2008). Maximum entropy inverse reinforcement learning. *Proc. of AAAI*, 1433-1438.

Principle of maximum entropy (2/2)

- The Lagrangian is given by

$$\mathcal{L} = -\sum_{\tau} p(\tau) \ln p(\tau) + \lambda \left(\sum_{\tau} p(\tau) - 1 \right) + \sum_{i} w_i \left(\sum_{\tau} p(\tau) \phi_i(\tau) - \tilde{\phi}_i \right)$$

  - λ, w_i: Lagrange multipliers

- Calculus of variation: $\partial \mathcal{L} / \partial p = 0$ yields

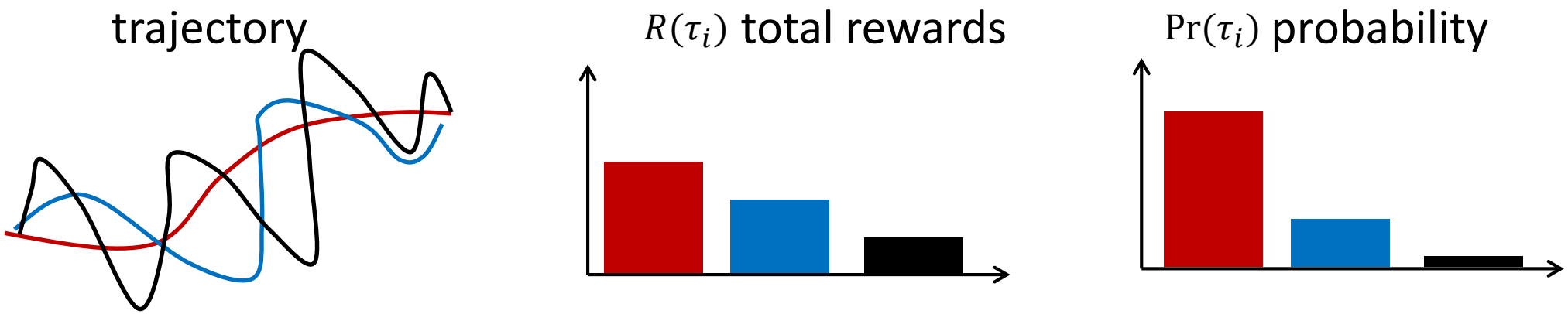
$$\ln p(\tau) = \sum_{i} w_i \phi_i(\tau) + \lambda - 1$$

$$\Rightarrow \quad p(\tau|w) = \frac{1}{Z(w)} \exp\left(w^{\top} \phi(\tau)\right) = \frac{1}{Z(w)} \exp\left(R(\tau)\right)$$

Probability under the principle of maximum entropy

- Trajectories with equivalent returns have equal probabilities
- Trajectories with higher returns are exponentially more preferred

$$p(\tau|w) = \frac{1}{Z(w)} \exp\big(R(\tau; w)\big)$$



Ziebart, B.D., Maas, A., Bagnell, J.A., and Dey, A. (2008). Maximum entropy inverse reinforcement learning. *Proc. of AAAI*, 1433-1438.

Parameter optimization

- w is estimated by maximum likelihood from the expert's dataset $\mathcal{D}^{\pi} = \{\tau_j^{\pi}\}_{j=1}^{N}$
- Log-likelihood is given by

$$L(w) = \frac{1}{N}\sum_{\tau_j^{\pi} \in \mathcal{D}^{\pi}} \ln p\left(\tau_j^{\pi} \mid w\right) = \frac{1}{N}\sum_{\tau_j^{\pi} \in \mathcal{D}^{\pi}} R(\tau_j^{\pi}; w) - \ln Z(w)$$

- w is updated by gradient ascent

$$w \leftarrow w + \alpha \nabla L(w), \qquad \nabla L(w) = \frac{1}{N}\sum_{\tau_j^{\pi} \in \mathcal{D}^{\pi}} \nabla R\left(\tau_j^{\pi}; w\right) - \nabla \ln Z(w)$$

- How to evaluate the gradient of $\ln Z(\boldsymbol{w})$?

MaxEnt-IRL: evaluate $Z(w)$ by solving model-based reinforcement learning

- $$\nabla \ln Z(w) = \frac{1}{Z(w)} \frac{\partial Z(w)}{\partial w} = \frac{1}{Z(w)} \sum_{\tau} \exp\left(w^{\top} \phi(\tau)\right) \phi(\tau)$$

$$= \underbrace{\sum_{\tau} P(\tau; w) \phi(\tau)}_{\text{expected feature under } P(\tau; w)} = \mathbb{E}_{P(\tau; w)}[\phi(\tau)]$$

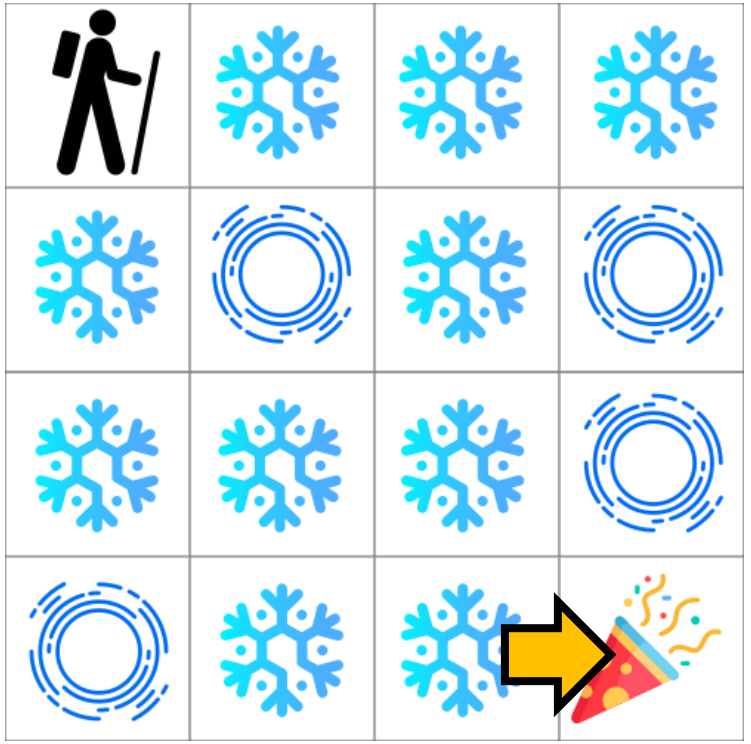
- In RL, the following equation holds

$$\sum_{\tau} P(\tau; w) \phi(\tau) = \underbrace{\sum_{s,a} P(s, a; w) \phi(s, a)}_{\text{expected feature under the joint state-action distribution}}$$

Samples from $P(s, a; w)$ can be generated by the optimal policy $\pi^*(a \mid s; w)$ trained with the current estimated reward $r(s, a; w)$

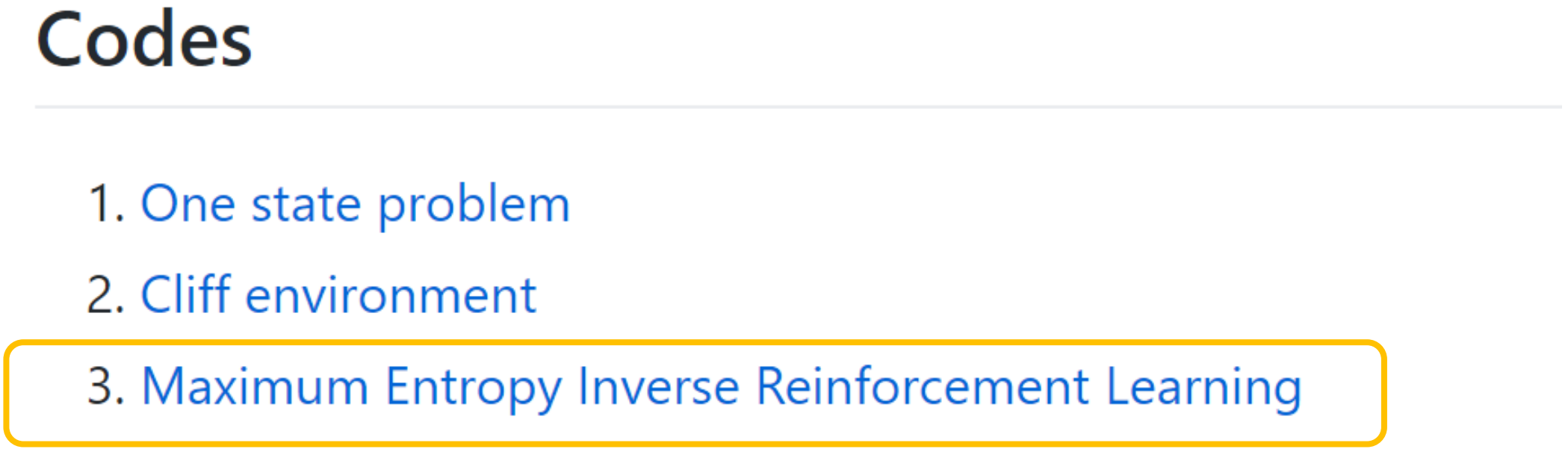
Simulation: FrozenLake task

- Discrete-state and discrete-action MDP task provided by OpenAI gym
  - Simplify the task by considering deterministic MDP
- Positive reward (+1) for entering the goal from left
- 0 otherwise
- Episode ends when entering a hole
  - 0 reward is given to the agent



Source code

- https://github.com/uchibe/kyutech-lectures



Driver Route Modeling

- Modeling route selection strategies of 25 taxi drivers
- Fomulated as a deterministic MDP
  - 300,000 states (e.g. road segments)
  - 900,000 actions (e.g. transitions at intersection)
- Different trips have different destinations and slightly different corresponding MDPs.
- We assume that the reward weight is independent of the goal state and therefore a single reward weight can be learned from many MDPs that differ only in goal state

road network (Pittsburgh)

Ziebart, B.D., Maas, A., Bagnell, J.A., & Dey, A.K. (2009). Human Behavior Modeling with Maximum Entropy Inverse Optimal Control. *Proc. of AAAI Spring Symposium on Human Behavior Modeling*, 3931–3936.

Driver Route Modeling

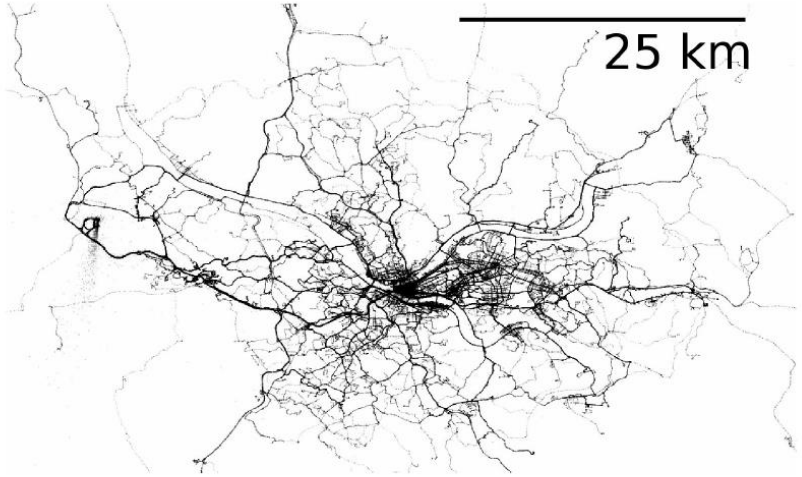
- 

Figure 2: The collected GPS datapoints

	Matching	90% Match	Log Prob
Time-based	72.38%	43.12%	N/A
Max Margin	75.29%	46.56%	N/A
Action	77.30%	50.37%	-7.91
Action (costs)	77.74%	50.75%	N/A
MaxEnt paths	**78.79%**	**52.98%**	**-6.85**

Table 1: Evaluation results for optimal estimated travel time route, max margin route, Boltzmann Q-value distributions (Action) and Maximum Entropy

Model	Dist. Match	90% Match
Markov (1x1)	62.4%	30.1%
Markov (3x3)	62.5%	30.1%
Markov (5x5)	62.5%	29.9%
Markov (10x10)	62.4%	29.6%
Markov (30x30)	62.2%	29.4%
Travel Time	72.5%	44.0%
Our Approach	**82.6%**	**61.0%**

Table 2: Evaluation results for Markov Model with various grid sizes, time-based model, and our umodel

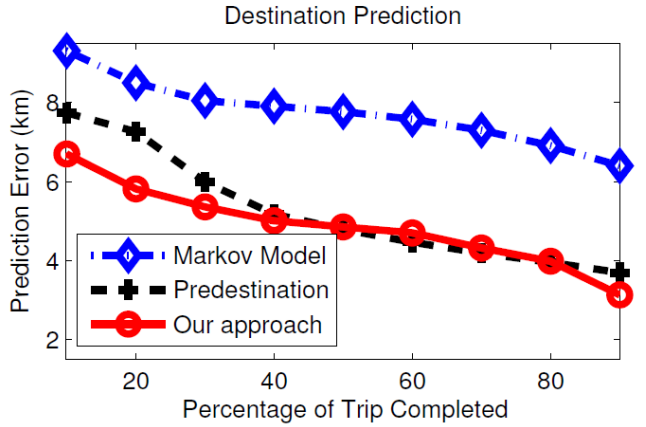
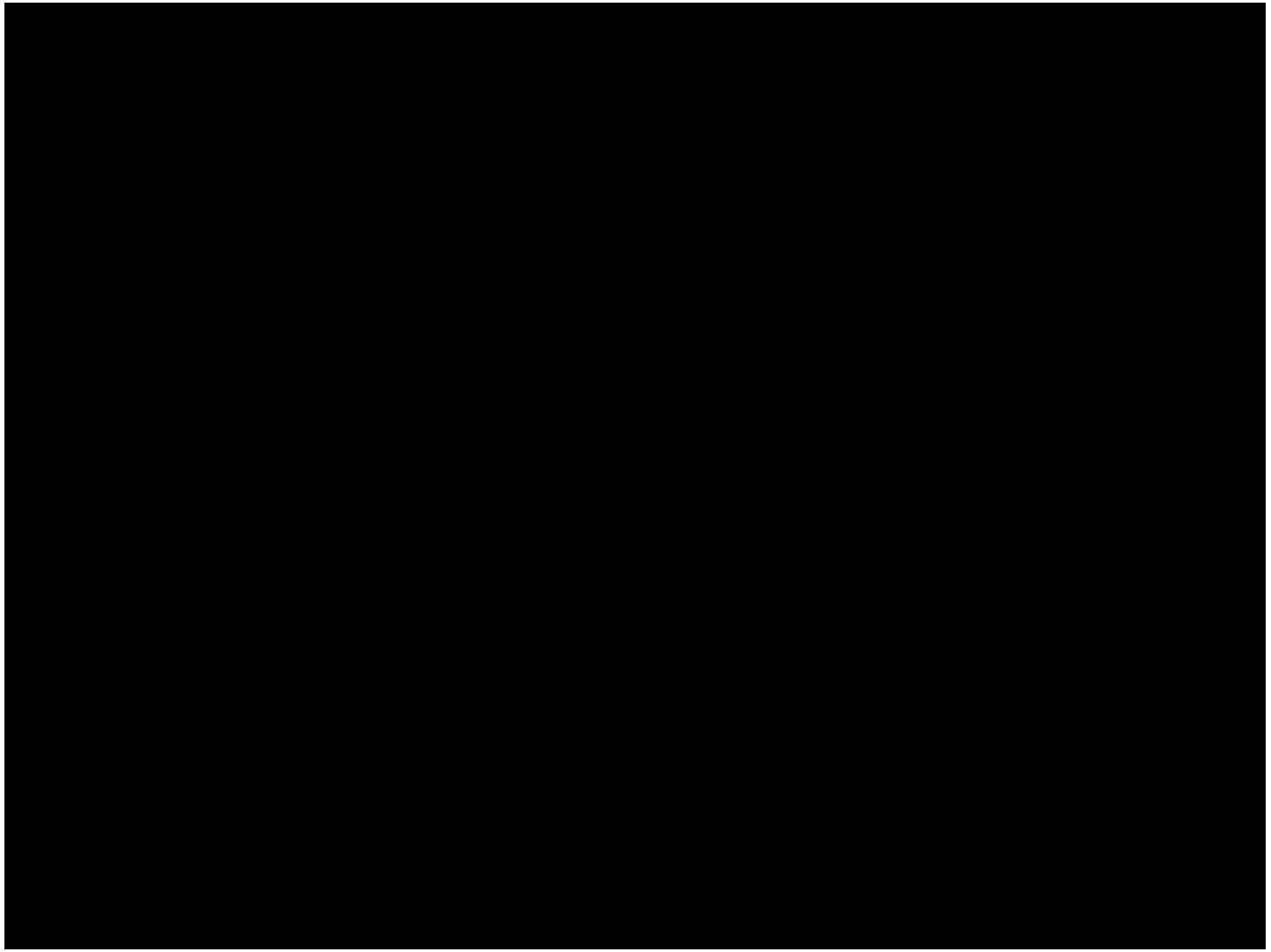


Figure 3: The best Markov Model, Predestination, and our approach's prediction errors

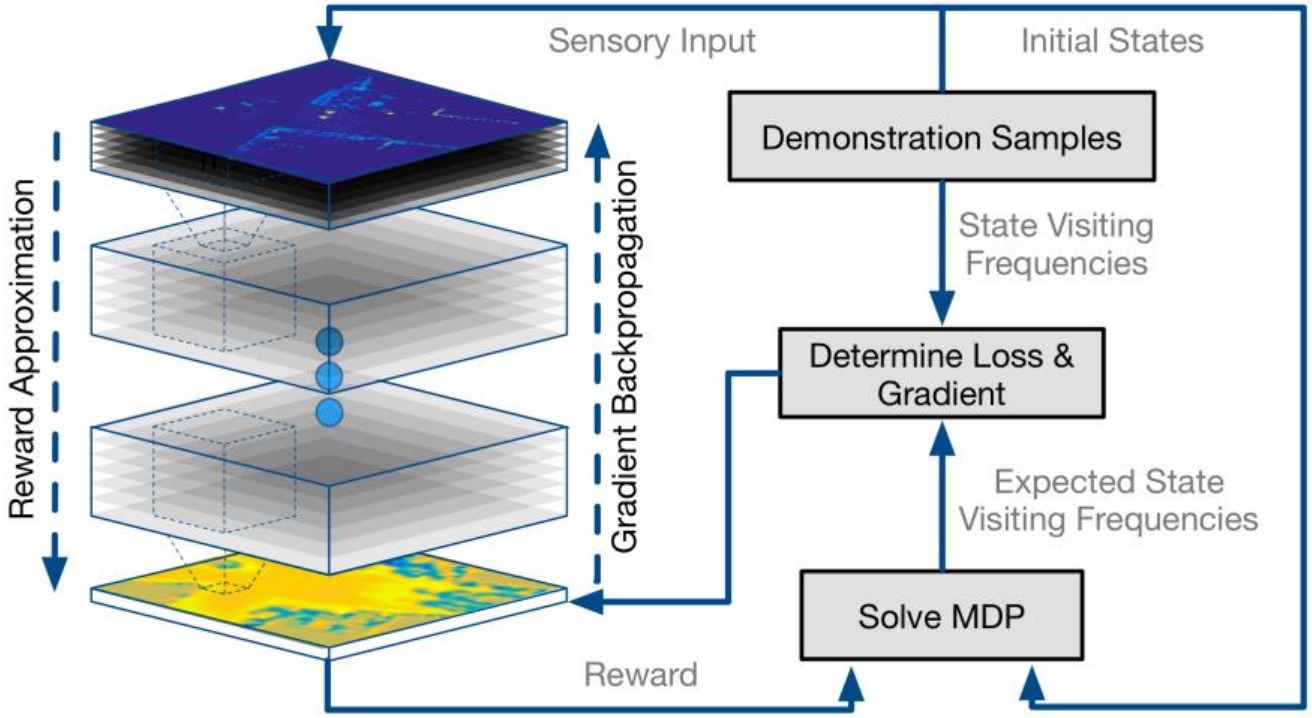
Ziebart, B.D., Maas, A., Bagnell, J.A., & Dey, A.K. (2009). Human Behavior Modeling with Maximum Entropy Inverse Optimal Control. *Proc. of AAAI Spring Symposium on Human Behavior Modeling*, 3931–3936.

Prediction of pedestrians's behavior

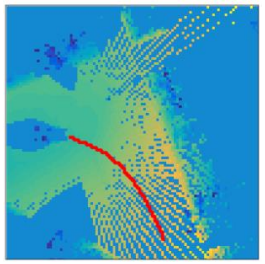


Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., & Srinivasa, S. (2009). Planning-based predictions for pedestrians. *Proc. of IEEE/RSJ IROS.*

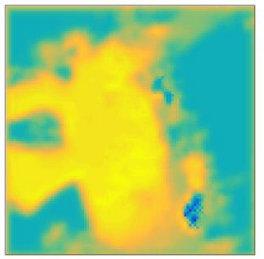
Maximum Entropy Deep Inverse Reinforcement Learning (MEDIRL): Nonlinear MaxEnt-IRL



エキスパートデータ



推定された報酬

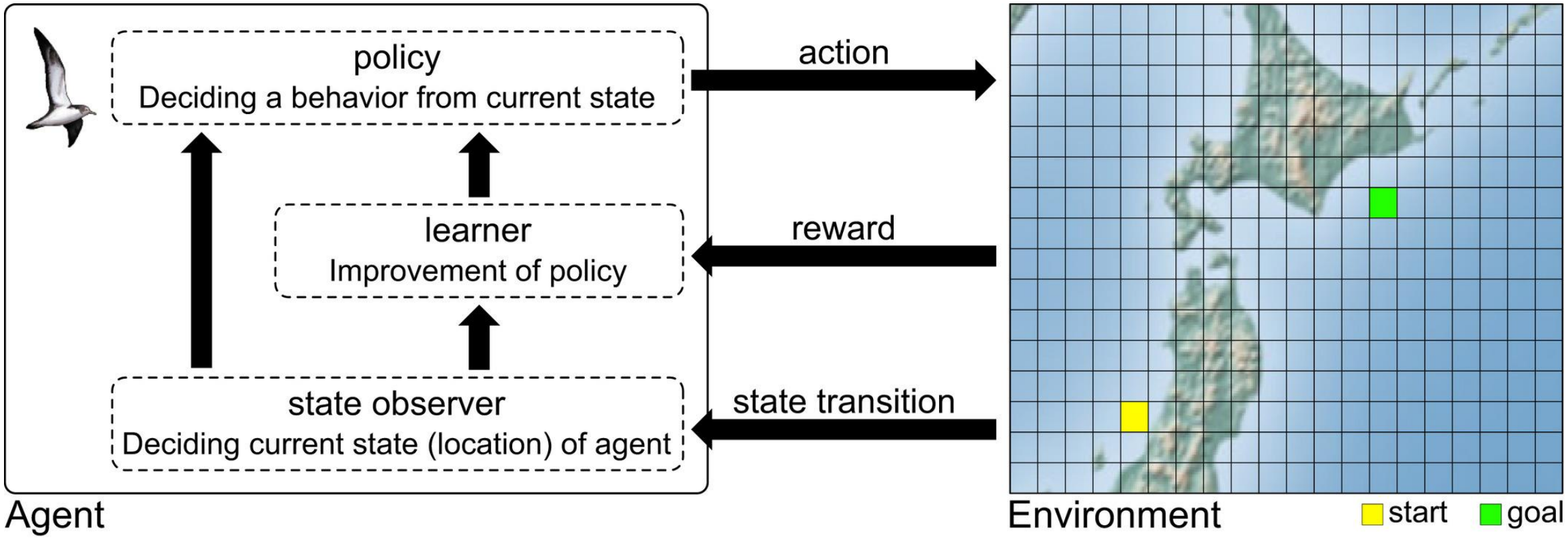


Wulfmeier, M., Wang, D.Z., & Posner, I. (2016). Watch This : Scalable Cost-Function Learning for Path Planning in Urban Environments. *Proc. of IEEE/RSJ IROS*.

Wulfmeier, M., Rao, D., Wang, D.Z., Ondruska, P., & Posner, I. (2017). Large-scale cost function learning for path planning using deep inverse reinforcement learning. *International Journal of Robotics Research*, vol. 36, no. 10: 1073–1087.

Prediction of the most likely route

- Formulate as discrete-state and discrete-action MDP
  - The original state is given by the position (x_t, y_t) and the elapsed time z_t

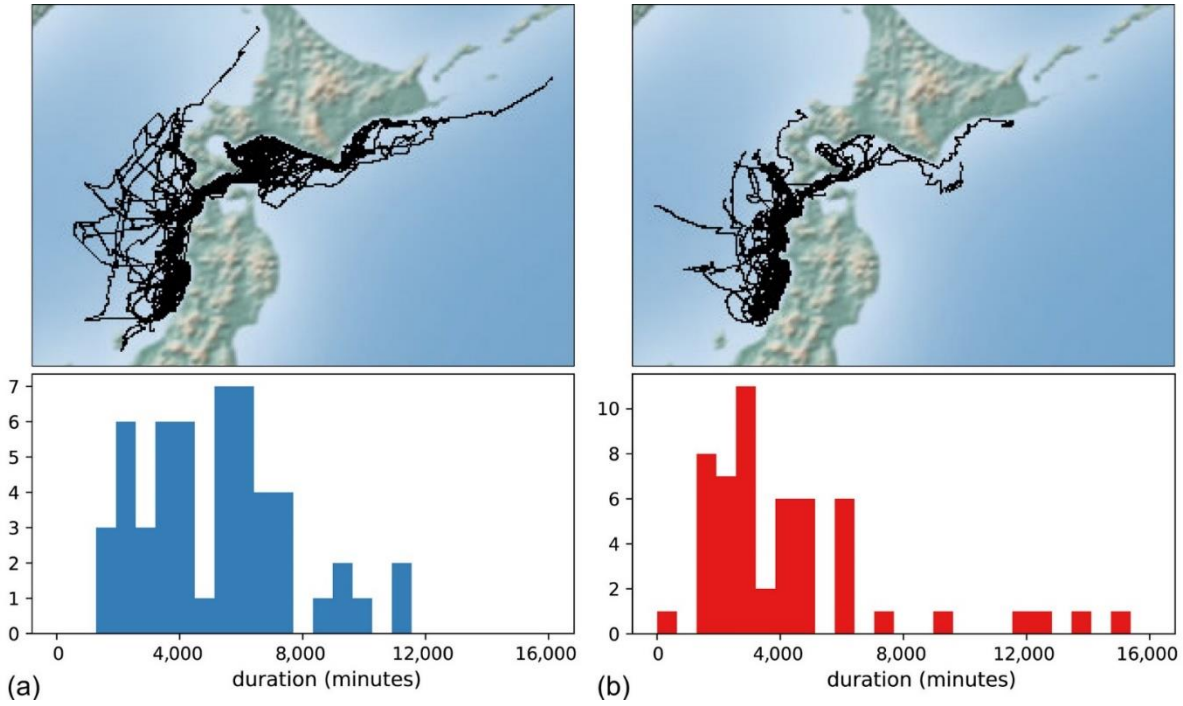


Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., Matsumoto, S., and Yoda, K. (2018). Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. Ecosphere.

Data

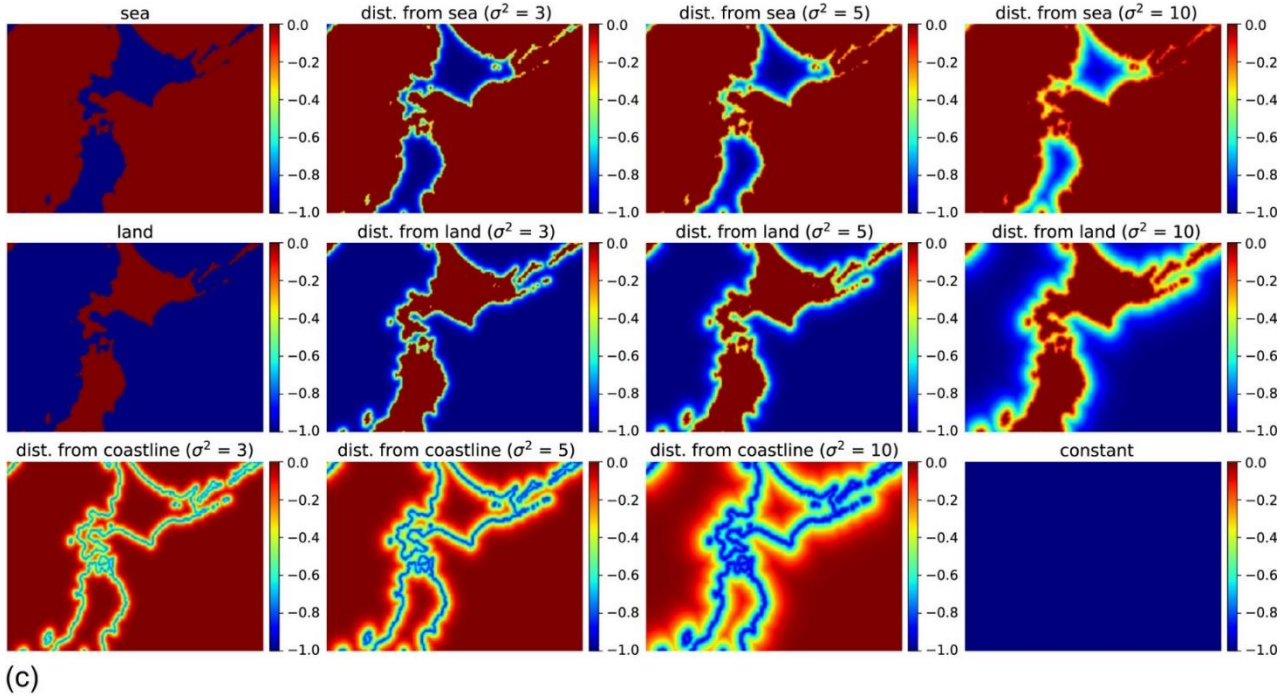
- 106 trajectories (53 males and 53 females)

Male shearwater trajectories and the histogram of trajectory duration



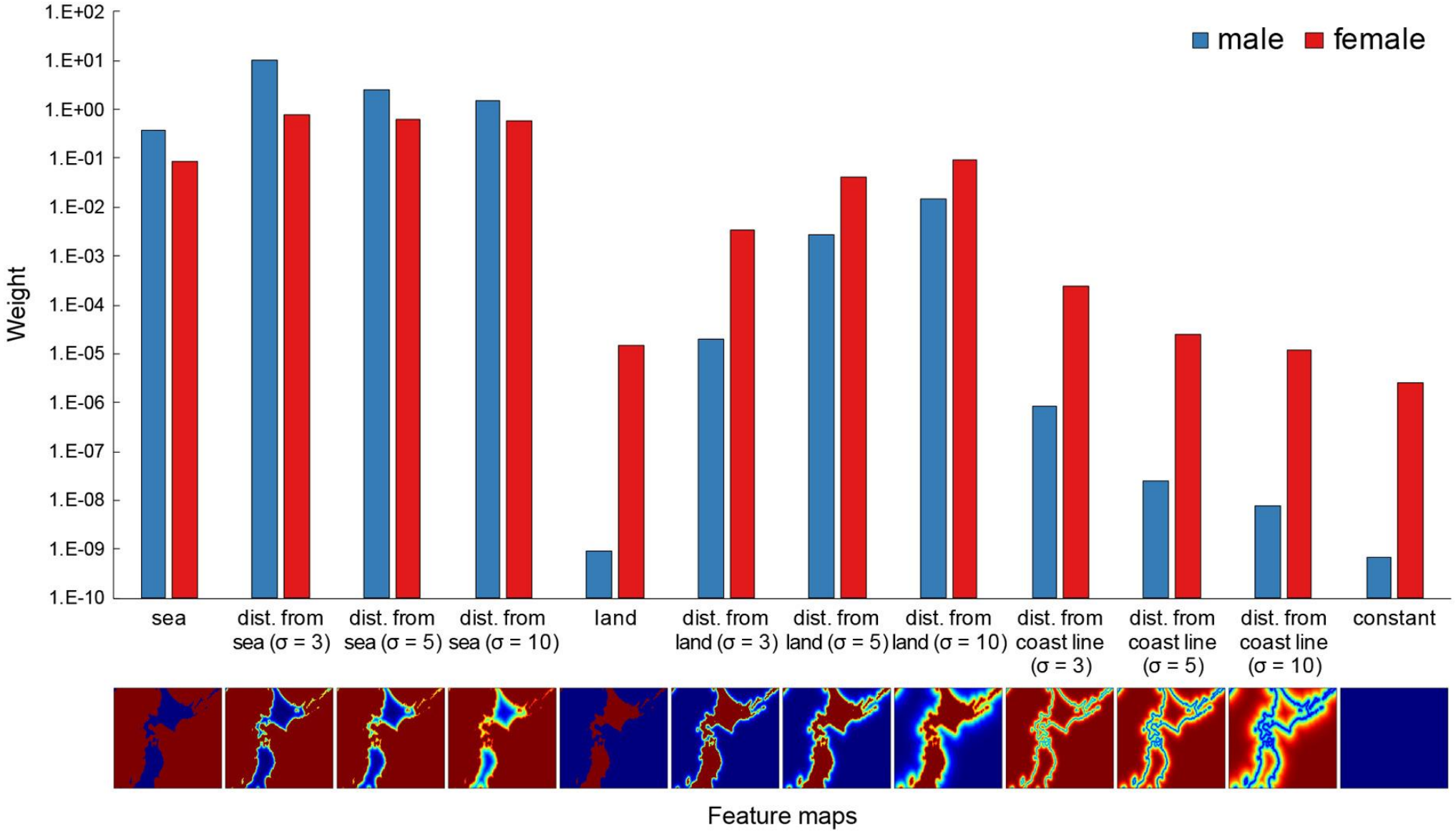
Female shearwater trajectories and the histogram of trajectory duration

reward features



Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., Matsumoto, S., and Yoda, K. (2018). Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. Ecosphere.

Estimated reward weights

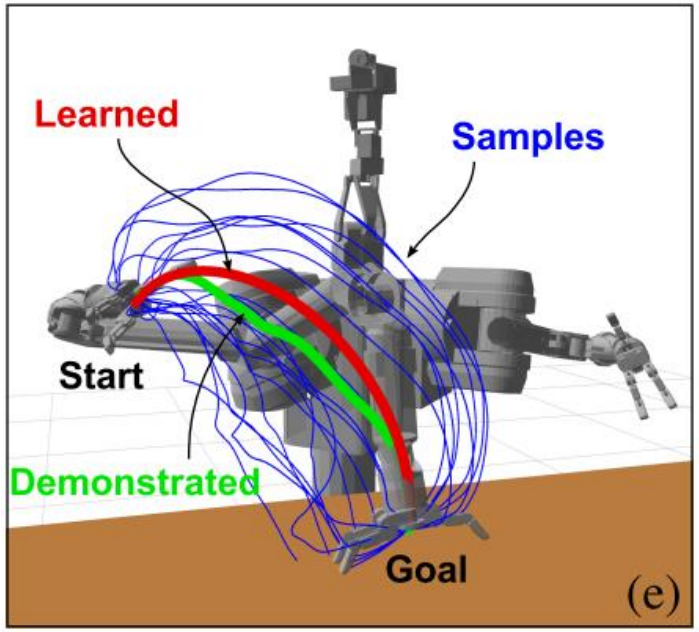


PI_LOC: Sample-based evaluation of Z

- Assumption: Two datasets are available
  - Expert's dataset $\mathcal{D}^{\pi}$
  - Baseline dataset $\mathcal{D}_s$, where control noise is introduced to the experts' dataset independently at each time-step
  - $\mathcal{D}^{\pi} \cup \mathcal{D}_s$ is considered the total dataset
- Approximated distribution

$$P(\tau; w) = \frac{\exp(R(\tau; w))}{\sum_{\tau \in \mathcal{D}^{\pi} \cup \mathcal{D}_s} \exp\big(R(\tau; w)\big)}$$

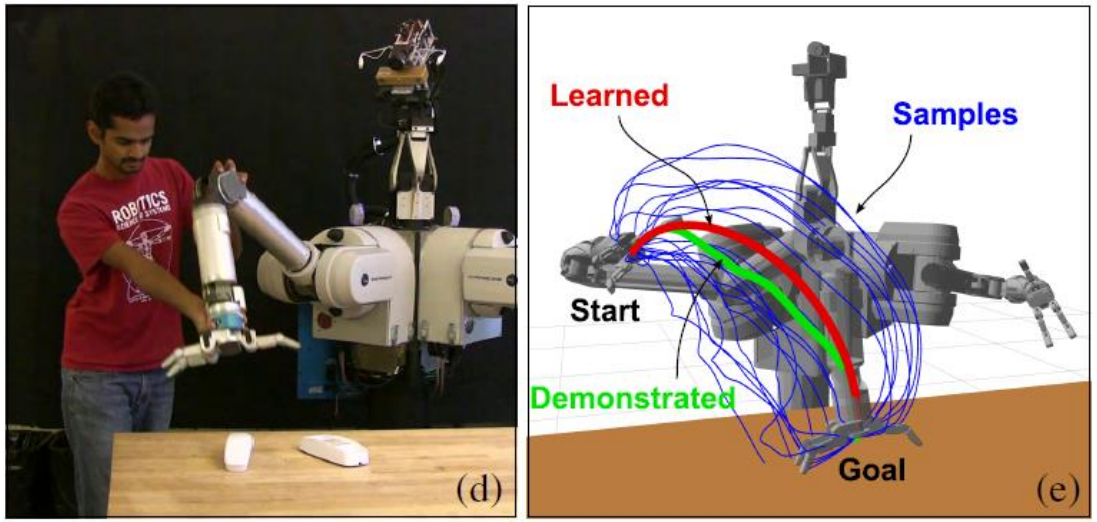
  - w is obtained by maximum likelihood without solving forward RL
- It is assumed that we can ignore trajectories that are far from those of expert to compute $R(\tau; w)$

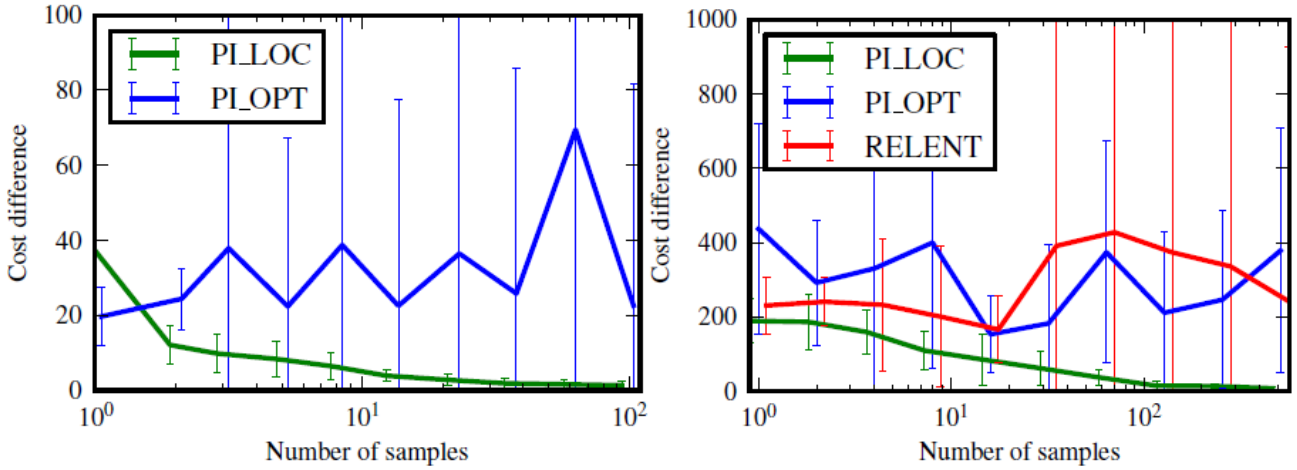


Kalakrishnan, M., Pastor, P., Righetti, L., & Schaal, S. (2013). Learning objective functions for manipulation. *Proc. of ICRA*, 1331–1336.

Comparison of sample generators

- PI_OPT: Generate samples from the optimal policy trained with the estimated reward (Aghasadeghi & Bretl, 2011).
- RELENT: Generate samples from uniform distribution and correct the estimator by importance sampling (Boularias et al., 2011)





(a) Learning inverse kinematics (b) Learning optimal motion planning

Fig. 3. Our algorithm (PI_LOC) was compared with algorithms PI_OPT and RELENT (see Sec. V for details), on two domains: (a) inverse kinematics, and (b) optimal motion planning. The charts show the cost difference $\mathbf{w}^*\tilde{\mathbf{\Phi}} - \mathbf{w}^*\mathbf{\Phi}^*$, with error bars depicting ± 1 standard deviation. PI_LOC shows fast learning with lower variance than the other algorithms on both test domains.

Evaluation of the normalizing constant

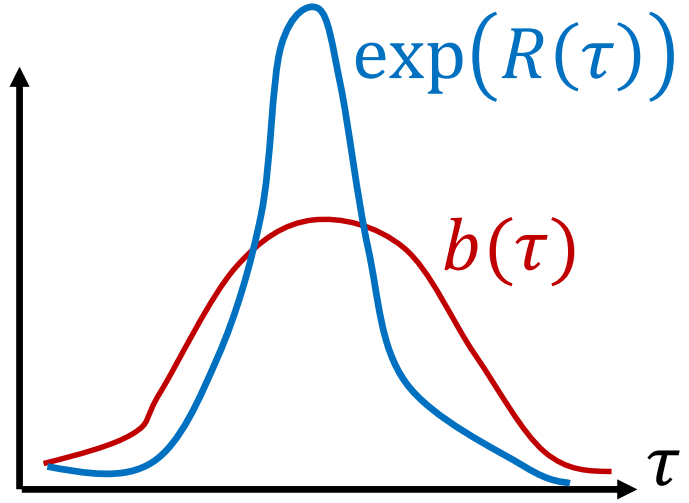
- Importance sampling estimator from samples of $b(\tau)$

$$\ln Z(w) = \ln \sum_{\tau} \exp\big(R(\tau; w)\big) = \ln \sum_{\tau} \frac{\exp\big(R(\tau; w)\big)}{b(\tau)} b(\tau)$$

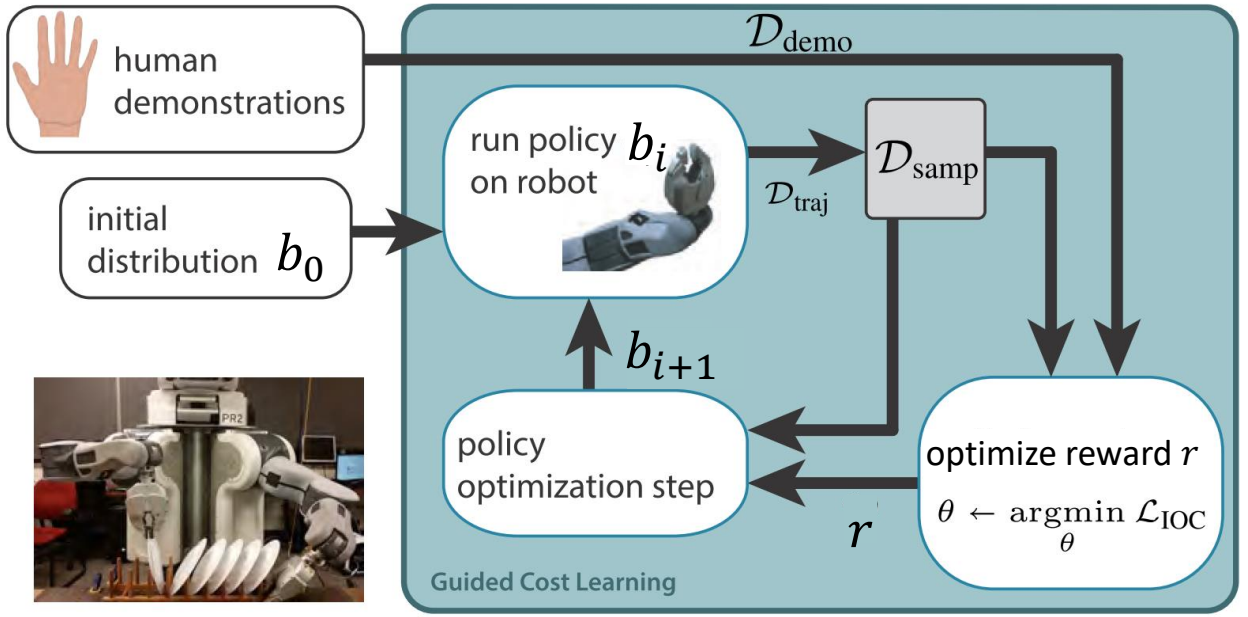
$$= \ln \mathbb{E}_b \left[\frac{\exp\big(R(\tau; w)\big)}{b(\tau)} \right] \approx \ln \frac{1}{M} \sum_{\tau_i^b \in \mathcal{D}^b} \frac{\exp\big(R(\tau_i; \boldsymbol{w})\big)}{b(\tau_i)}$$

  - RELENT assumes that $b(\tau)$ is a uniform distribution. Since it is far from the expert distribution, the estimator has a large variance.
  - PI_LOC uses samples collected by perturbated expert policy, but the importance sampling is not used. Therefore, it gives a biased estimate.
  - PI_IOC improves the sampling distribution to collect samples, but the importance sampling technique is not used. Therefore, it gives a biased estimate.

Guided Cost Learning



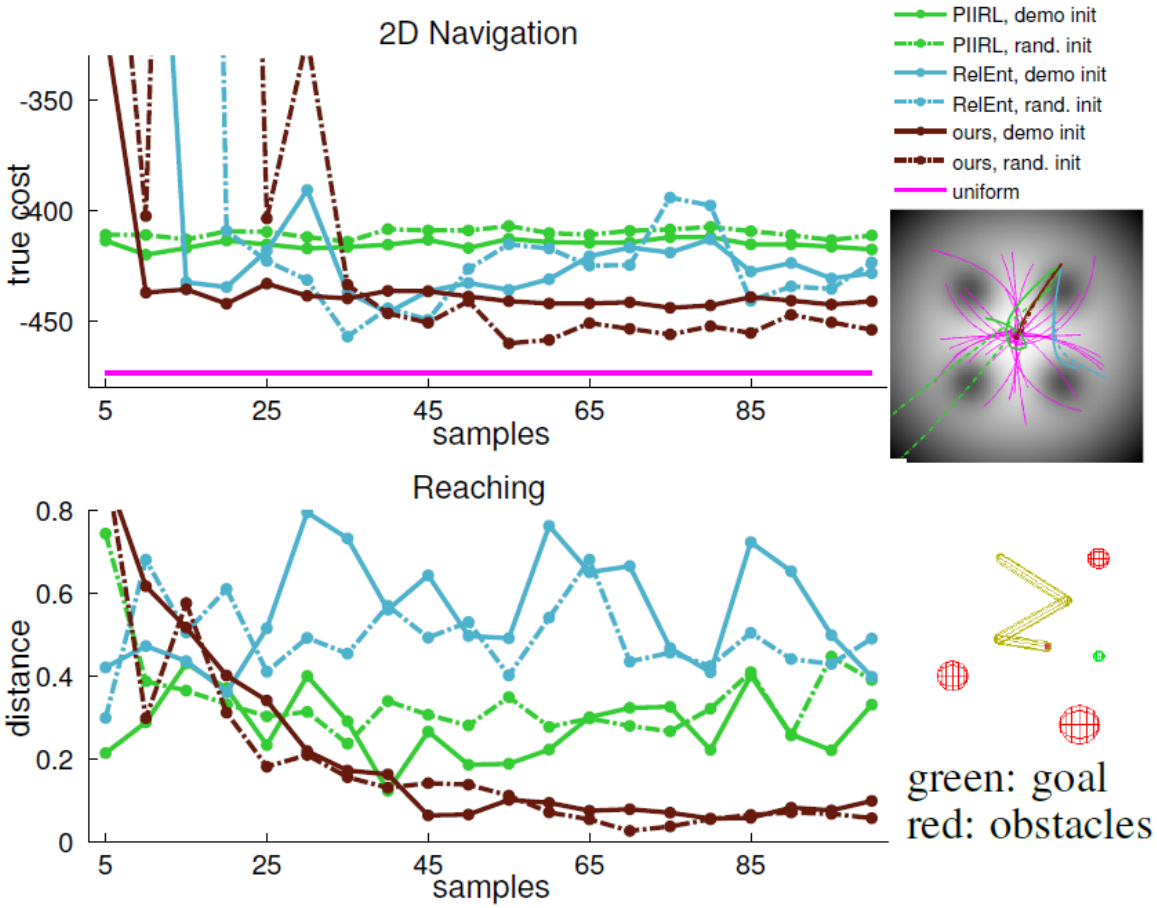
- What is the best $b(\tau)$? ➡ $b(\tau) \propto \exp(R(\tau))$
- Designing $b(\tau)$ is quite difficult because $R(\tau)$ is unknown
- Instead, we can adaptively refine $b(\tau)$ to generate more samples in those regions of the trajectory space that are good according to the current reward function
- Iteration of reward estimation and policy improvement



Finn, C., Levine, S., & Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *Proc. of ICML*, 49–58. ICML.

Experimental results

- Updating the sampling distribution is important
  - Note: the vertical axis represents the total cost (smaller is better)



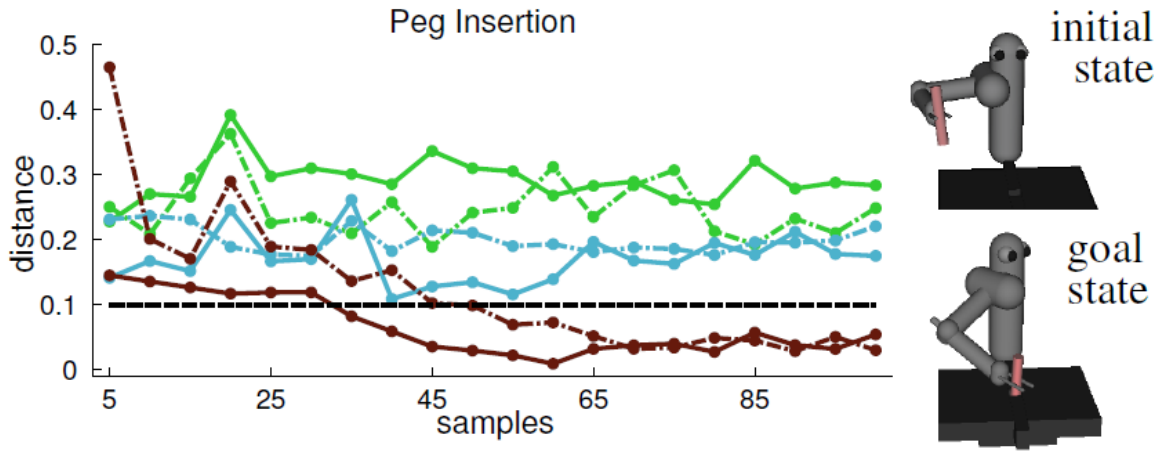
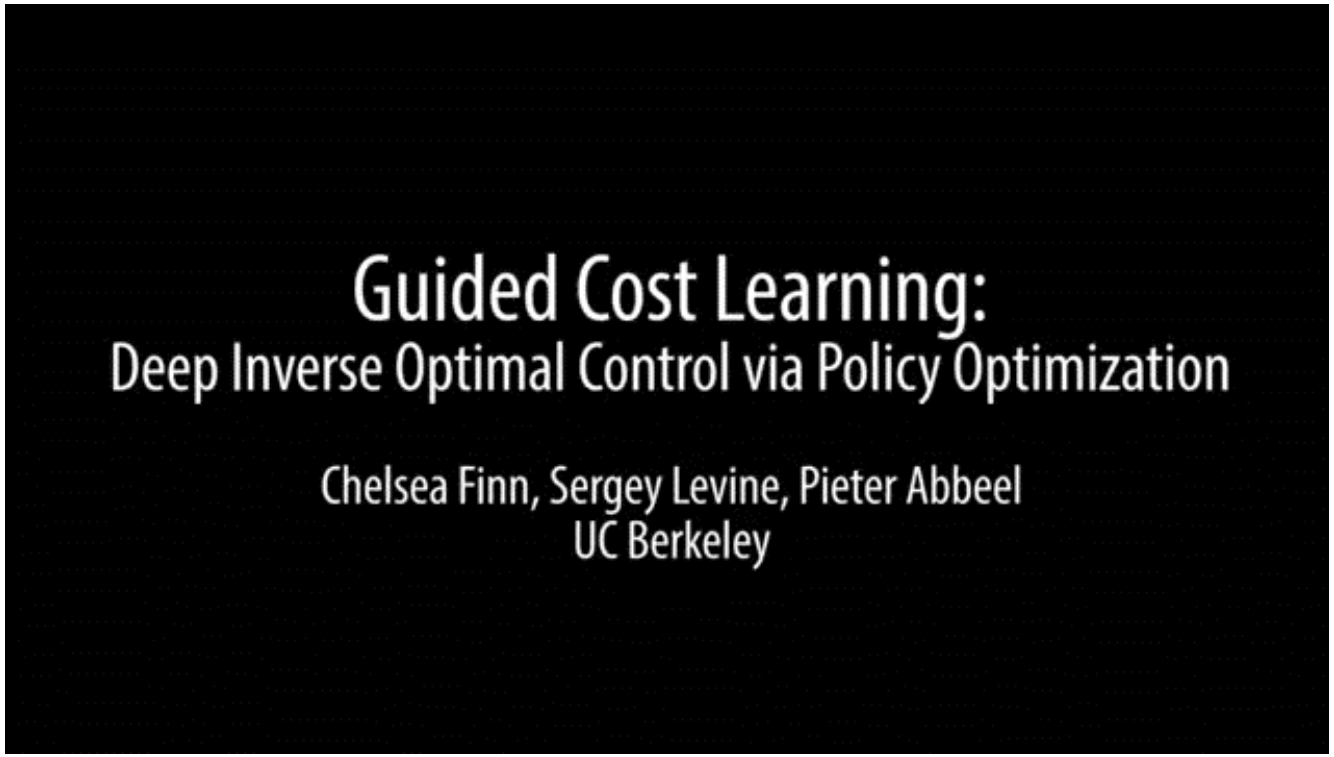


Figure 2. Comparison to prior work on simulated 2D navigation, reaching, and peg insertion tasks. Reported performance is averaged over 4 runs of IOC on 4 different initial conditions . For peg insertion, the depth of the hole is 0.1m, marked as a dashed line. Distances larger than this amount failed to insert the peg.

Real robot experiments



dish (NN)	RelEnt IRL	GCL $q(\mathbf{u}_t|\mathbf{x}_t)$	GCL reopt.
success rate	0%	**100%**	**100%**
# samples	100	90	90

pouring (NN)	RelEnt IRL	GCL $q(\mathbf{u}_t|\mathbf{x}_t)$	GCL reopt.
success rate	10%	**84.7%**	34%
# samples	150,150	75,130	75,130

pouring (affine)	RelEnt IRL	GCL $q(\mathbf{u}_t|\mathbf{x}_t)$	GCL reopt.
success rate	0%	0%	–
# samples	150	120	–

Table 1. Performance of guided cost learning (GCL) and relative entropy (RelEnt) IRL on placing a dish into a rack and pouring almonds into a cup. Sample counts are for IOC, omitting those for optimizing the learned cost. An affine cost is insufficient for representing the pouring task, thus motivating using a neural network cost (NN). The pouring task with a neural network cost is evaluated for two positions of the target cup; average performance is reported.

Finn, C., Levine, S., & Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *Proc. of ICML*, 49–58. ICML.

References

- Aghasadeghi, N., & Bretl, T. (2011). Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals. In *Proc. of IROS,* 1561–1566.
- Boularias, A., Kober, J. & Peters, J. (2011). Relative Entropy Inverse Reinforcement Learning. In *Proc. of AISTATS*, 182-189.
- Dvijotham, K., & Todorov, E. (2010). Inverse Optimal Control with Linearly Solvable MDPs. In *Proc. of ICML*, 335–342.
- Finn, C., Levine, S., & Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. In *Proc. of ICML*, 49–58. ICML.

References

- Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., Matsumoto, S., and Yoda, K. (2018). Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. Ecosphere.
- Huszár, F. (2016). An Alternative Update Rule for Generative Adversarial Networks.
- Kalakrishnan, M., Pastor, P., Righetti, L., & Schaal, S. (2013). Learning objective functions for manipulation. *Proc. of ICRA*, 1331–1336.
- 岸川, 荒井. (2019). 深層逆強化学習による自動運転の安心走行実現. 第33回人工知能学会全国大会予稿集.
- Muelling, K., Boularias, A., Mohler, B., Schölkopf, B., and Peters, J. (2014). Learning strategies in table tennis using inverse reinforcement learning. Biological Cybernetics, 108(5): 603-619.
- Ross, S., & Bagnell, J.A. (2010). Efficient Reductions for Imitation Learning. *Proc. of AISTATS*, 9:661–668.
- Shimosaka, M., Kaneko, T., & Nishi, K. (2014). Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning. *Proc. of the 17th International IEEE Conference on Intelligent Transportation Systems*, 1694–1700.

References

- Shimosaka, M., Nishi, K., Sato, J., & Kataoka, H. (2015). Predicting driving behavior using inverse reinforcement learning with multiple reward functions towards environmental diversity. In *Proc. of IEEE Intelligent Vehicles Symposium (IV 2015)*, pp. 567–572, 2015.
- Tsurumine, Y., Cui, Y., Uchibe, E., and Matsubara, T. (2017). Deep dynamic policy programming for robot control with raw images. In *Proc. of IEEE/RSJ IROS*.
- Tsurumine, Y., Cui, Y., Uchibe, E., and Matsubara, T. (2019). Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. *Robotics and Autonomous Systems*, 112: 72-83.
- Wulfmeier, M., Wang, D.Z., & Posner, I. (2016). Watch This : Scalable Cost-Function Learning for Path Planning in Urban Environments. *Proc. of IEEE/RSJ IROS*.
- Wulfmeier, M., Rao, D., Wang, D.Z., Ondruska, P., & Posner, I. (2017). Large-scale cost function learning for path planning using deep inverse reinforcement learning. *International Journal of Robotics Research*, vol. 36, no. 10: 1073–1087.

References

- Yamaguchi, S., Honda, N., Ikeda, M., Tsukada, Y., Nakano, S., Mori, I., and Ishii, S. (2018). Identification of animal behavioral strategies by inverse reinforcement learning. PLoS Computational Biology.
- Ziebart, B.D., Maas, A., Bagnell, J.A., and Dey, A. (2008). Maximum entropy inverse reinforcement learning. *Proc. of AAAI*, 1433-1438.
- Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., & Srinivasa, S. (2009). Planning-based predictions for pedestrians. *Proc. of IEEE/RSJ IROS.*
- Ziebart, B.D., Maas, A., Bagnell, J.A., & Dey, A.K. (2009). Human Behavior Modeling with Maximum Entropy Inverse Optimal Control. *Proc. of AAAI Spring Symposium on Human Behavior Modeling*, 3931–3936.