Inverse Reinforcement Learning via Deep Gaussian Process

Presenter: Ming Jin

UAI 2017

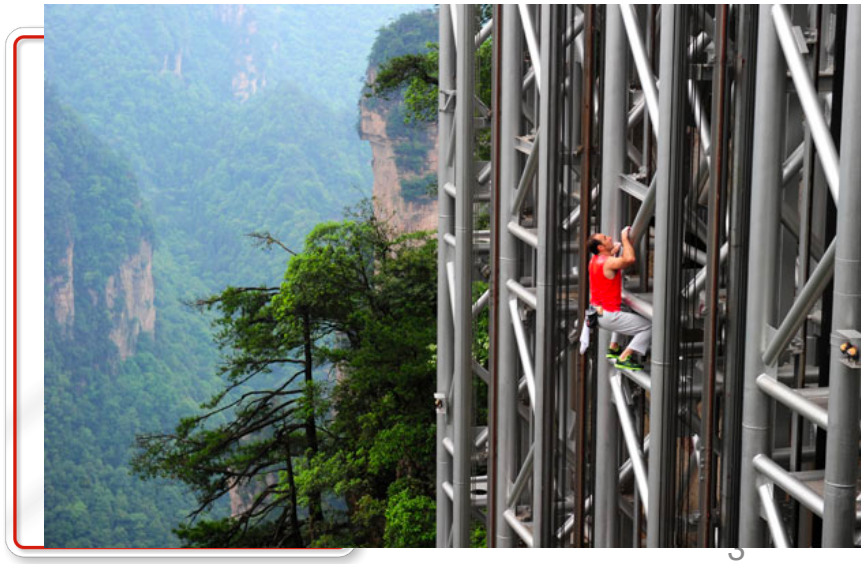
Ming Jin, Andreas Damianou,

Pieter Abbeel, and Costas J. Spanos

Understanding people's preference is key to bringing technology to our daily life

How AI can act like a social/ friendly/ intelligent/ real person?

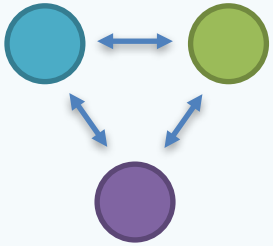
- People's decision making often involves…
  - Long-term planning vs. short-term gain
  - Risk seeking vs. risk aversion
  - Individual preferences over outcomes



How AI can act like a social/ friendly/ intelligent/ real person?

- Three learning paradigms..

Game theoretic approach

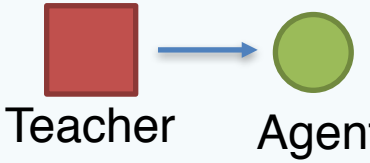


- Task as cooperative/ noncooperative game

- Agent as utility maximizer: Nash equilibrium

(Anand 1993, Stackelberg 2011)

Behavior cloning

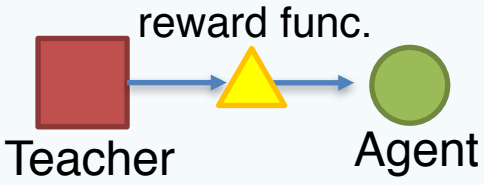


- Directly learn the teacher's policy using supervised learning

- Mapping from state to action

(Pomerleau, 1989; Sammut et al., 1992; Amit & Mataric, 2002)

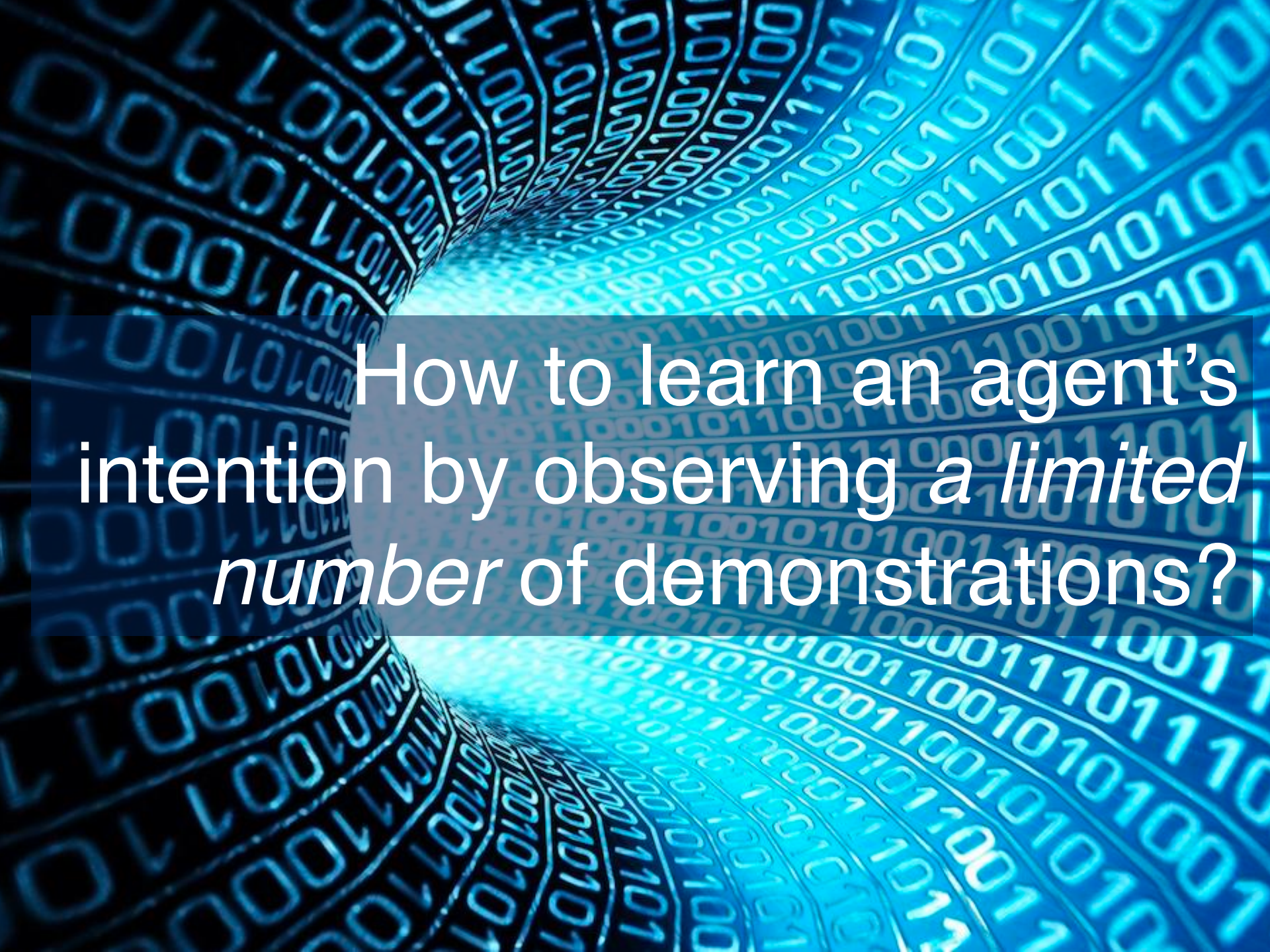
Inverse learning



- Learn the succinct reward function

- Derive the optimal policy from rewards

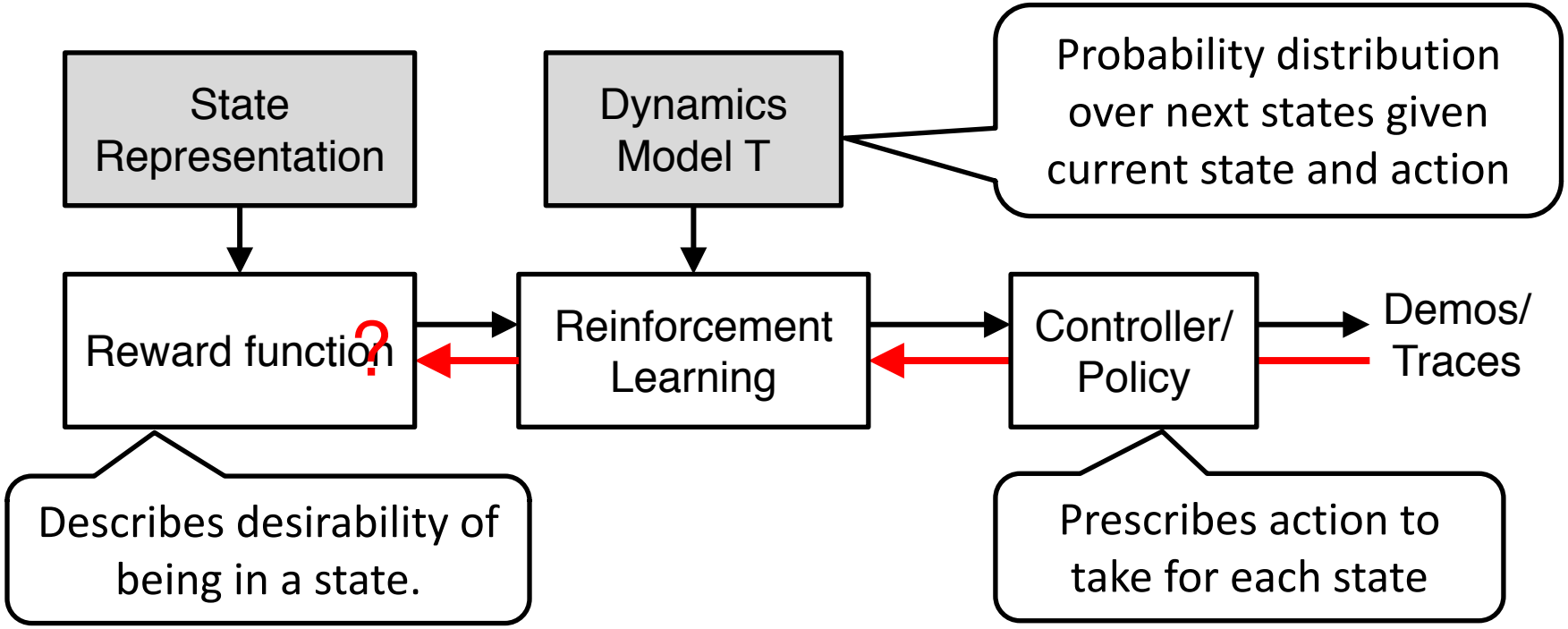
(Ng and Russell, 2000; Abbeel and Ng, 2004; Ratliff et al., 2006, Levine et al.)


How to learn an agent's intention by observing *a limited number* of demonstrations?

Outline

- Inverse reinforcement learning (IRL) problem formulation
- Gaussian process reward modeling
- Incorporating the representation learning
- Variational inference
- Experiments
- Conclusion

RL vs. IRL



Key challenges:

- Providing a formal specification of the control task.
- Building a good dynamics model.
- Finding closed-loop controllers.

(Adapted from Abbeel's slides on "Inverse reinforcement learning")

IRL in a nutshell: given demonstrations, infer reward

- Input:
  - Dynamics model $P_{sa}(s_{t+1}|s_t, a_t)$
  - *No* reward function $r^*(s)$
  - Demonstration $\mathcal{M}$: $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    (= trace of the teacher's policy π^*)
- Desired output:
  - Reward function $r(s)$
  - Policy $\hat{\pi}: S \rightarrow A$, which (ideally) has performance guarantees, e.g., expected reward difference (EVD)

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^*(x_t) \,|\pi^*\right] - \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^*(x_t) \,|\hat{\pi}\right]$$

Parametric vs. nonparametric representation of reward function

- Linear representation:

$$r(s) = \boldsymbol{w}^T \boldsymbol{\phi}(s)$$

Limited representative power

  - Maximum margin planning (MMP): (Ratliff et al, 2006)
  - Maximum Entropy IRL (MaxEnt): (Ziebart et al., 2008)

- Nonlinear representation:
  - Deep NN: (Wulfmeier et al., 2015)

Needs significant demonstrations to avoid overfitting

- Nonparametric representation:
  - Gaussian Process IRL: (Levine et al., 2011)

Deep Gaussian Process IRL: (Jin et al., 2017)

Two slides on Gaussian Process

- A Gaussian distribution depends on a mean and a covariance matrix.
- A Gaussian process depends on a mean and a covariance function.
- Let's start with a multivariate Gaussian Distr.:

$$p(\underbrace{f_1, f_2, \ldots, f_s}_{\boldsymbol{f_A}}, \underbrace{f_{s+1}, f_{s+2}, \ldots, f_N}_{\boldsymbol{f_B}}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix} \text{ and } \boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{AA} & \boldsymbol{K}_{AB} \\ \boldsymbol{K}_{BA} & \boldsymbol{K}_{BB} \end{bmatrix}$$

(Adapted from Damianou's slides on "System identification and control with (deep) Gaussian process")

Two slides on Gaussian Process

- In the GP context, we deal with an infinite dimensional Gaussian distribution:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_X \\ \dots \end{bmatrix} \text{ and } \boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{XX} & \cdots \\ \dots & \dots \end{bmatrix}$$

Conditional distribution:

Gaussian distribution

Given $p(\boldsymbol{f}_A, \boldsymbol{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$
Then the posterior distribution:
$p(\boldsymbol{f}_A|\boldsymbol{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}_{A|B}, \boldsymbol{K}_{A|B})$
with $\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + \boldsymbol{K}_{AB}\boldsymbol{K}_{BB}^{-1}(\boldsymbol{f}_B - \boldsymbol{\mu}_B)$
$\boldsymbol{K}_{A|B} = \boldsymbol{K}_{AA} - \boldsymbol{K}_{AB}\boldsymbol{K}_{BB}^{-1}\boldsymbol{K}_{BA}$

Gaussian process

Given $p(\boldsymbol{f}_X, \boldsymbol{f}_*) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$
Then the posterior distribution:
$p(\boldsymbol{f}_*|\boldsymbol{f}_X) \sim \mathcal{N}(\boldsymbol{\mu}_{*|X}, \boldsymbol{K}_{*|X})$
with $\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + \boldsymbol{K}_{AB}\boldsymbol{K}_{BB}^{-1}(\boldsymbol{f}_B - \boldsymbol{\mu}_B)$
$\boldsymbol{K}_{A|B} = \boldsymbol{K}_{AA} - \boldsymbol{K}_{AB}\boldsymbol{K}_{BB}^{-1}\boldsymbol{K}_{BA}$

(Adapted from Damianou's slides on "System identification and control with (deep) Gaussian process")

Reward function modeled by a Gaussian process

- Reward is a function of states: $r(\boldsymbol{x}) \in \mathcal{R}$
- We discretize the world into n states, each described by a feature vector $\boldsymbol{x}_i \in \mathcal{R}^m$:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} \in \mathcal{R}^{n \times m}, \boldsymbol{r} = \begin{bmatrix} r(\boldsymbol{x}_1) \\ \vdots \\ r(\boldsymbol{x}_n) \end{bmatrix} \in \mathcal{R}^n$$

- It can be modeled with a zero-mean GP prior:

$$\boldsymbol{r}|\boldsymbol{X}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{XX}})$$

Parameter of the covarince function

How to find out the reward given demonstrations?

- GPIRL trains the parameters by maximum likelihood estimation (Levine et al., 2011):

$$p(\mathcal{M}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\mathcal{M}|\boldsymbol{r})p(\boldsymbol{r}|\boldsymbol{X},\boldsymbol{\theta})d\boldsymbol{r}$$

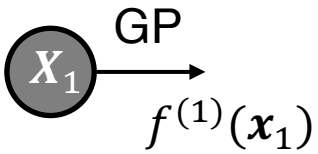
RL term (Ziebart et al., 2008) | GP prior

- Prediction of the reward at any test input is found through the conditional:

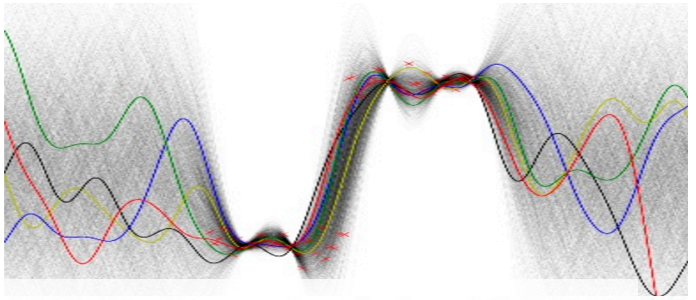
$$r^*|\boldsymbol{r},\boldsymbol{X},\boldsymbol{x}^* \sim \mathcal{N}(\boldsymbol{K}_{\boldsymbol{x}^*}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-\mathbf{1}}\boldsymbol{r}, k_{\boldsymbol{x}^*\boldsymbol{x}^*} - \boldsymbol{K}_{\boldsymbol{x}^*\boldsymbol{X}}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}^{-\mathbf{1}}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{x}^*})$$

Can we improve the *complexity* of reward function w/o *overfitting*?

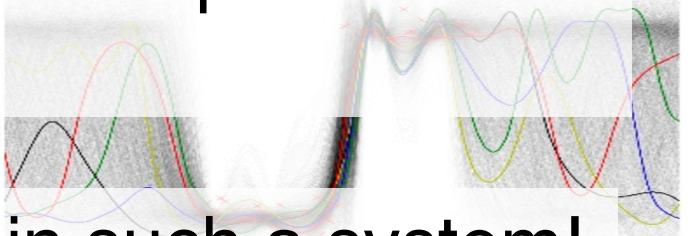
- Step function example:



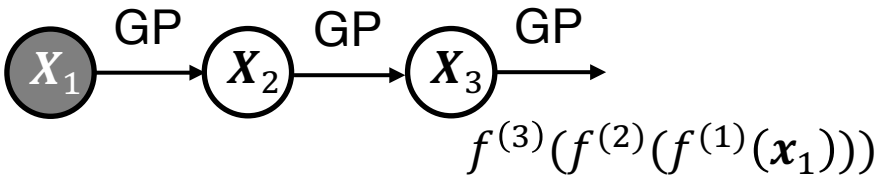
Predictive draws of step function

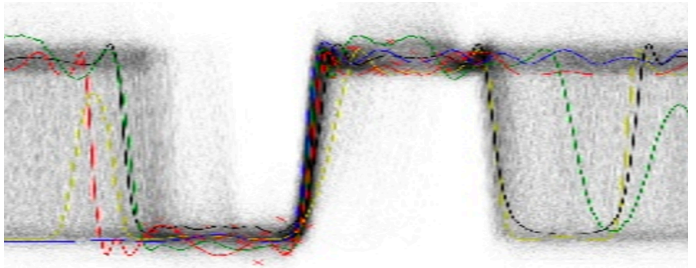


Overfitting does not appear to be a problem for a deeper architecture…



…the main challenge is to train such a system!

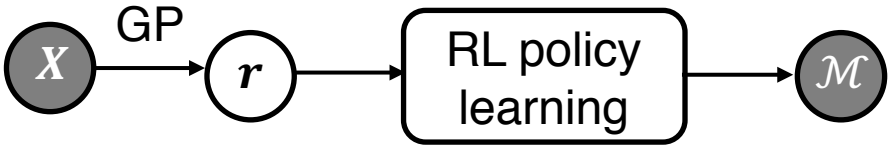




Figures adapted from http://keyonvafa.com/deep-gaussian-processes/

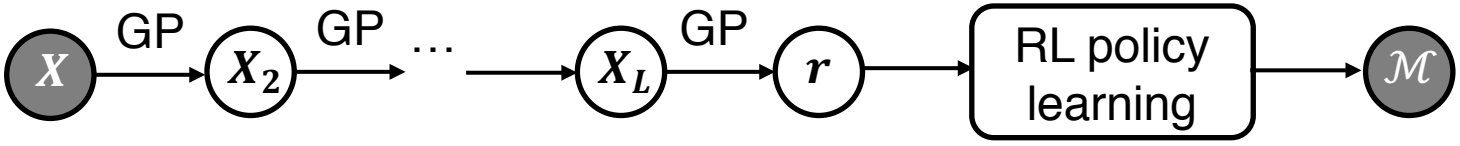
Can we improve the *complexity* of reward function w/o *overfitting*?

- IRL with reward modeled as GP: (Levine et al., 2011)



$$p(\mathcal{M}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\mathcal{M}|\boldsymbol{r})p(\boldsymbol{r}|\boldsymbol{X},\boldsymbol{\theta})d\boldsymbol{r}$$

- IRL with reward modeled as deep GP:



$$p(\mathcal{M}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\mathcal{M}|\boldsymbol{r})p(\boldsymbol{X_2},\ldots,\boldsymbol{X_L},\boldsymbol{r}|\boldsymbol{X},\boldsymbol{\theta})d(\boldsymbol{X_2},\ldots,\boldsymbol{X_L},\boldsymbol{r})$$

We need a tractable form of the likelihood function for training

- Let's illustrate with a 2-stack GP:

The integral is still intractable…

$$p(\mathcal{M}|\boldsymbol{X})$$

$$= \int p(\mathcal{M}, \boldsymbol{r}, \boldsymbol{D}, \boldsymbol{B}|\boldsymbol{X}) d(\boldsymbol{D}, \boldsymbol{B}, \boldsymbol{r})$$

$$p(\mathcal{M}|\boldsymbol{r}) \quad * \quad p(\boldsymbol{r}|\boldsymbol{D}) * p(\boldsymbol{D}|\boldsymbol{B}) * p(\boldsymbol{B}|\boldsymbol{X})$$

GP prior: $\mathcal{N}(\mathbf{0}, \boldsymbol{K_{DD}})$

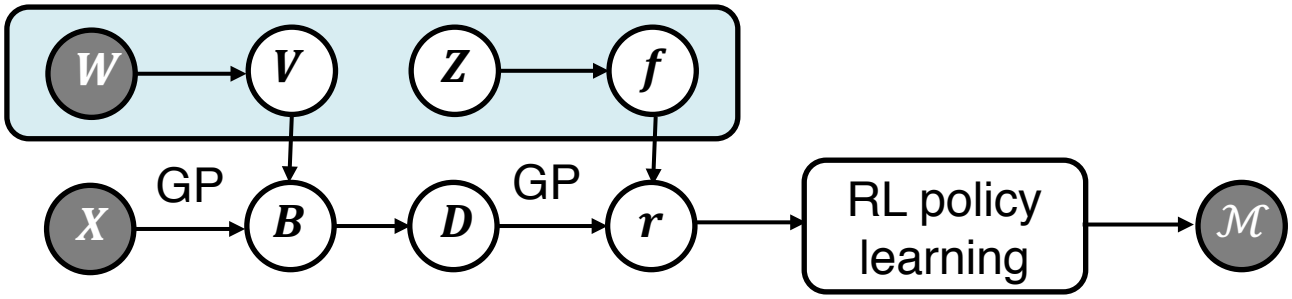
GP prior: $\mathcal{N}(\mathbf{0}, \boldsymbol{K_{XX}})$

RL probability given by MaxEnt (Ziebart et al., 2008)

Gaussian noise: $D_{ij} \sim \mathcal{N}(B_{ij}, \lambda^{-1})$

Introduce inducing points

- Add to each latent layer..



$$p(\mathcal{M}, \boldsymbol{r}, \boldsymbol{f}, \boldsymbol{D}, \boldsymbol{B}, \boldsymbol{V} | \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W}) =$$

GP prior: $\mathcal{N}(\mathbf{0}, \boldsymbol{K_{ZZ}})$

$$p(\mathcal{M}|\boldsymbol{r}) * p(\boldsymbol{r}|\boldsymbol{f}, \boldsymbol{D}, \boldsymbol{Z}) * p(\boldsymbol{f}|\boldsymbol{Z}) * p(\boldsymbol{D}|\boldsymbol{B}) * p(\boldsymbol{B}|\boldsymbol{V}, \boldsymbol{X}, \boldsymbol{W})$$

Conditional Gaussian: $\mathcal{N}(\boldsymbol{K_{DZ}K_{ZZ}^{-1}f}, \boldsymbol{\Sigma_r})$

Conditional Gaussian: $\boldsymbol{b}^m \sim \mathcal{N}(\boldsymbol{K_{XW}K_{WW}^{-1}v^m}, \boldsymbol{\Sigma_B})$

Use *variational lower bound* to make training tractable

- Introduce variational distribution:

Model distributions $\mathcal{P}$

: cond. Gaussian

$p(\boldsymbol{D}|\boldsymbol{B})$: Gaussian noise

$p(\boldsymbol{V}|\boldsymbol{W})$: $GP(\boldsymbol{0}, \boldsymbol{K_{WW}})$

Variational distributions $\mathcal{Q}$

$q(\boldsymbol{B}|\boldsymbol{V}, \boldsymbol{X}, \boldsymbol{W}) = p(\boldsymbol{B}|\boldsymbol{V}, \boldsymbol{X}, \boldsymbol{W})q(\boldsymbol{f}$

$q(\boldsymbol{D})$: $\prod \delta(\boldsymbol{d}^m - \boldsymbol{K_{XW}}\boldsymbol{K_{WW}^{-1}}\tilde{\boldsymbol{v}}^m)$

$q(\boldsymbol{V})$: $\prod \mathcal{N}(\tilde{\boldsymbol{v}}^m, \boldsymbol{G}^m)$

- Jensen's inequality to derive lower bound:

$$\log \int \mathcal{P} \;\; \geq \int \mathcal{Q} \;\; \log \frac{\mathcal{P}}{\mathcal{Q}}$$

Use *variational lower bound* to make training tractable

- Derive a tractable lower bound:

$$\log p(\mathcal{M}|\boldsymbol{X})$$

$$= \log \int p(\mathcal{M}|\boldsymbol{r}) p(\boldsymbol{r}|\boldsymbol{f}, \boldsymbol{D}) p(\boldsymbol{f}) p(\boldsymbol{D}|\boldsymbol{B}) p(\boldsymbol{B}|\boldsymbol{V}, \boldsymbol{X}) p(\boldsymbol{V})$$

$$\geq \int q(\boldsymbol{f}) q(\boldsymbol{D}) p(\boldsymbol{B}|\boldsymbol{V}, \boldsymbol{X}) q(\boldsymbol{V}) \log \frac{p(\mathcal{M}|\boldsymbol{r}) p(\boldsymbol{r}|\boldsymbol{f}, \boldsymbol{D}) p(\boldsymbol{f}) p(\boldsymbol{D}|\boldsymbol{B}) p(\boldsymbol{V})}{q(\boldsymbol{f}) q(\boldsymbol{D}) q(\boldsymbol{V})}$$

$$= \mathcal{L}_{\mathcal{M}} + \mathcal{L}_{\mathcal{G}} - \mathcal{L}_{KL} + \mathcal{L}_{\mathcal{B}} - \frac{nm}{2} \log(2\pi\lambda^{-1})$$

The **value can be computed**

We can also **compute the derivative** with respect to the parameters:

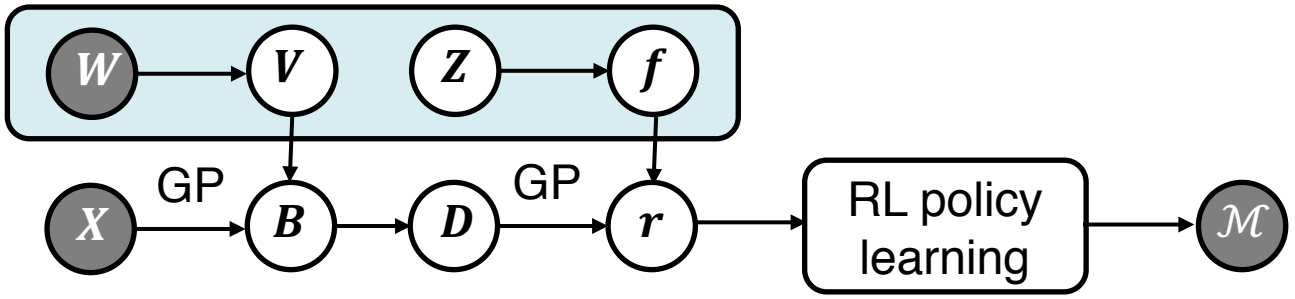
- Kernel function parameters
- Inducing points

Optimizing the objective turns the variational distribution Q into the true model posterior

Inducing points provide a succinct summary of data

- Given a new state $\boldsymbol{x}^*$, the predicted reward is a function of the *latent representation*:

$$r^* = \boldsymbol{K}_{\boldsymbol{D}^*\boldsymbol{Z}}\boldsymbol{K}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1}\tilde{\boldsymbol{f}}$$



- This can be used for knowledge transfer in a new situation

The original likelihood for the deep GP IRL is intractable..

Our method introduces inducing points combined with variational inequality to derive a lower bound for inference.

Outline

- Inverse reinforcement learning (IRL) problem formulation
- Gaussian process reward modeling
- Incorporating the representation learning
- Variational inference
- **Experiments**
- Conclusion

We use the *expected value difference (EVD)* as the metric

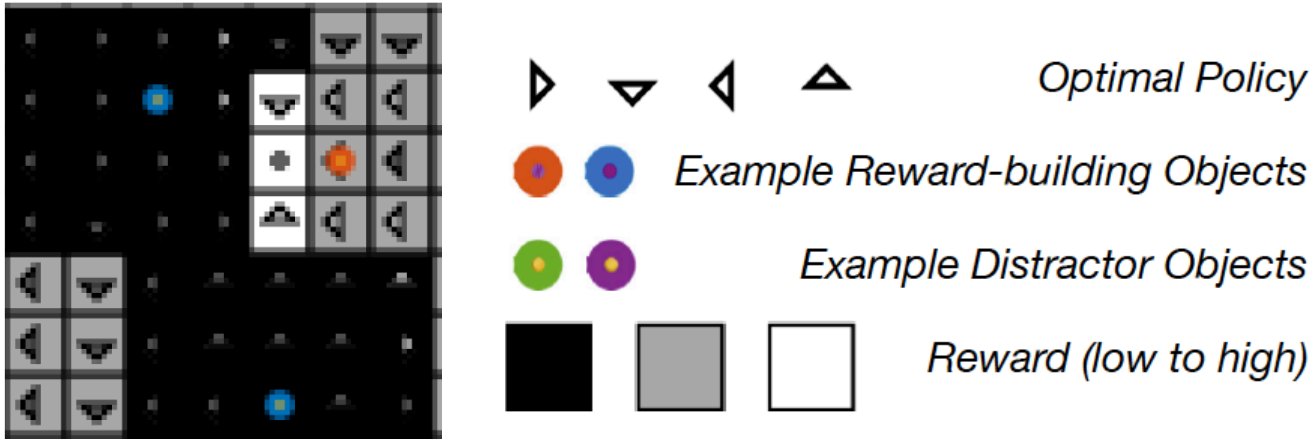
- EVD measures the expected reward earned under the optimal policy π^* w/ true reward and the policy derived by IRL $\hat{\pi}$

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t) \,|\pi^*\right] - \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t) \,|\hat{\pi}\right]$$

- We also visually compare the true reward with that learned by IRL

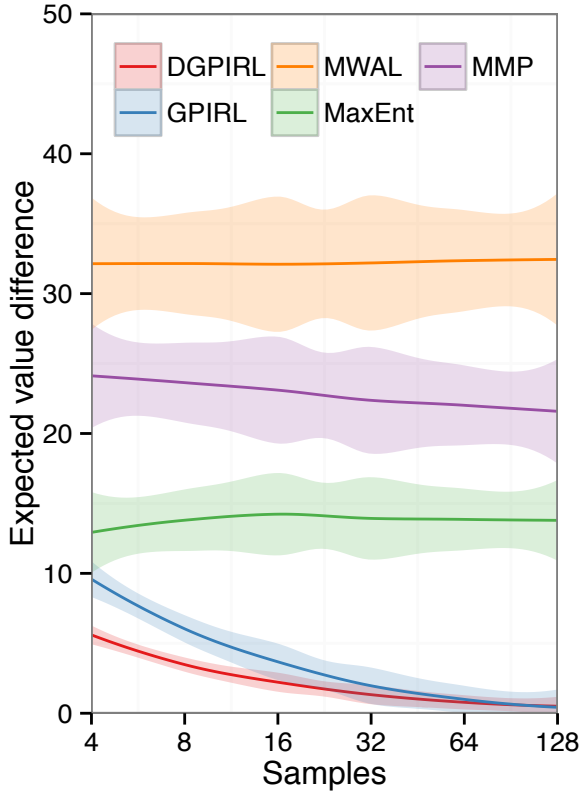
ObjectWorld has a nonlinear reward structure

- **Reward:** +1: 1step within & 3 step within
  -1 if only 3 steps withi , 0 otherwise
- Features: min. dist. to an object of each type
- Nonlinear, but still preserves local distance

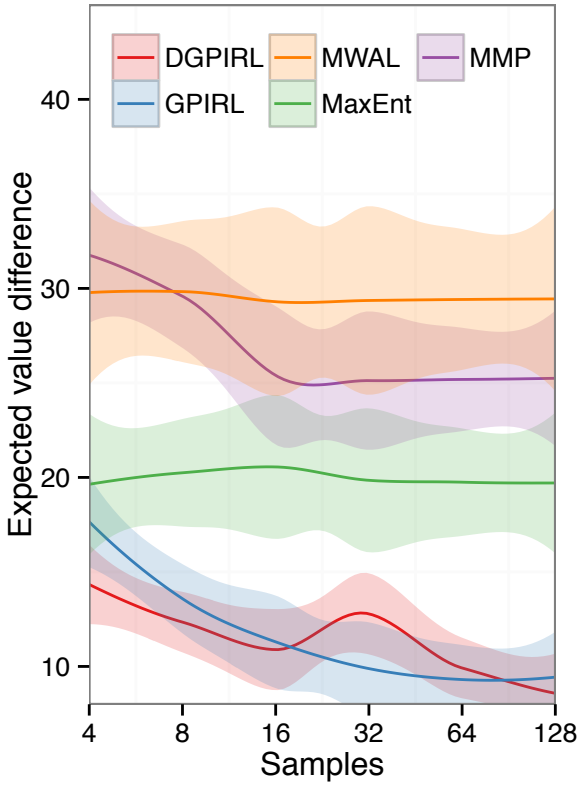


Both DGP-IRL and GPIRL captures the correct reward

Plot of EVD for the world where demos are given

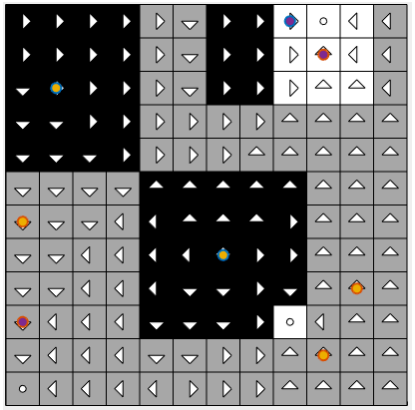


Plot of EVD for a new world where demos are not available

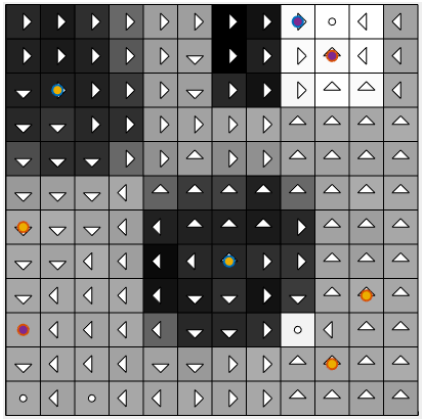


Both DGP-IRL and GPIRL captures the correct reward

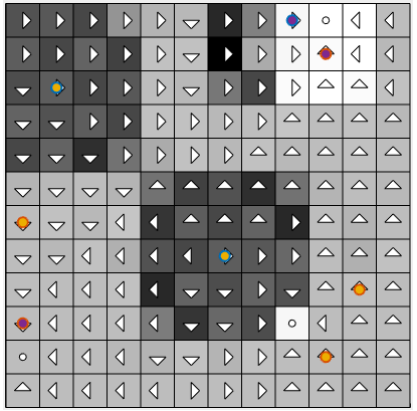
Ground truth



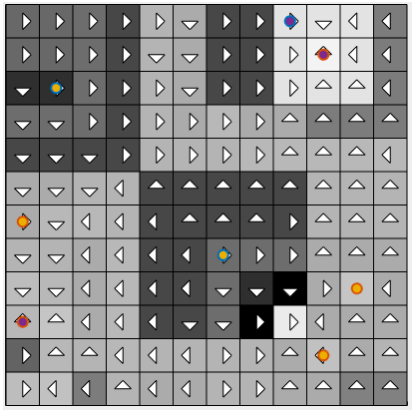
DGP-IRL



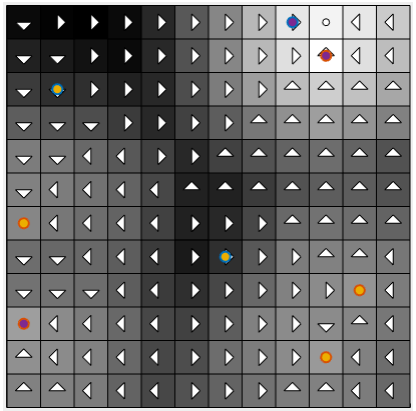
GPIRL



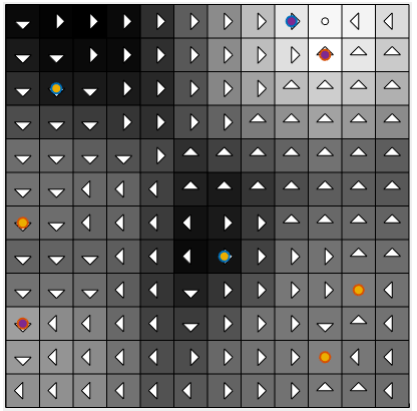
FIRL



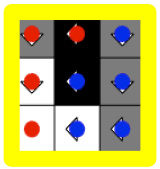
MaxEnt

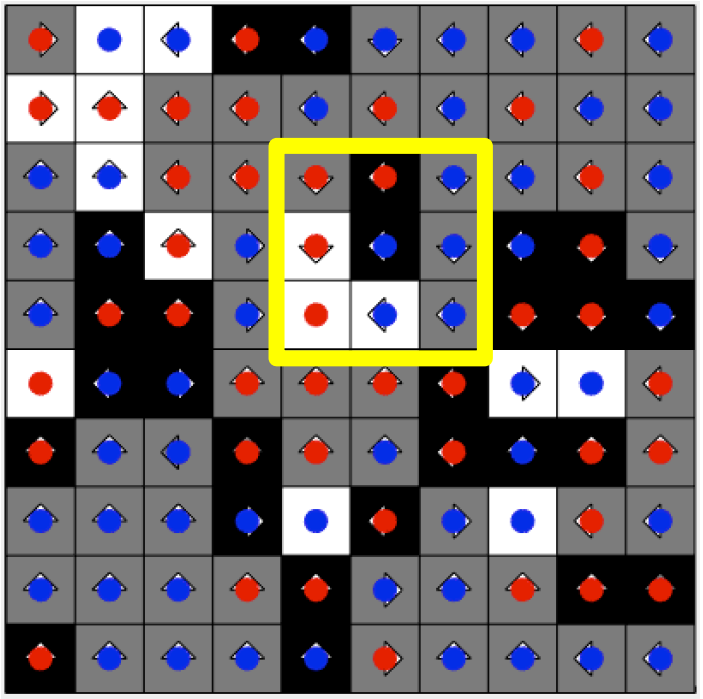


MMP



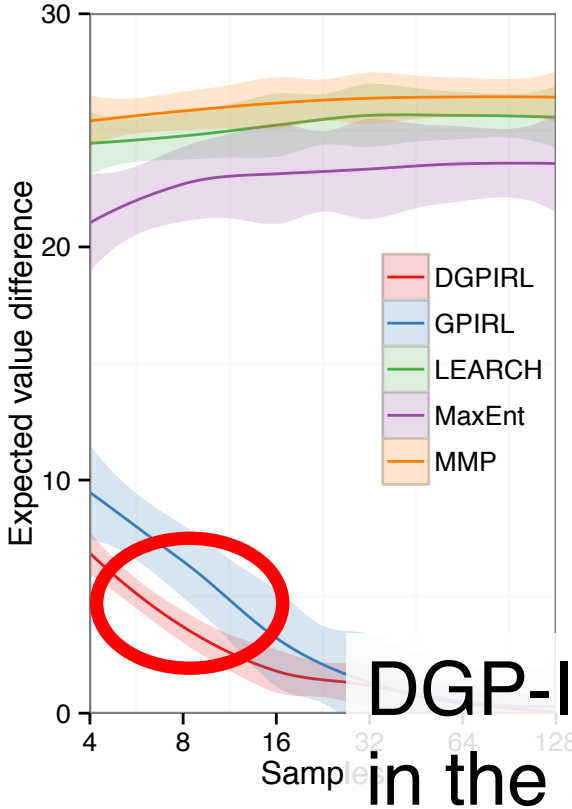
BinaryWorld has even more nonlinear reward structure

- **Reward:** neighborhood  with 4 blues (+1) and 5 blues (-1)
  - Nonlinear, combinatorial
- **Features:** ordered list of colors (top left to bottom right)

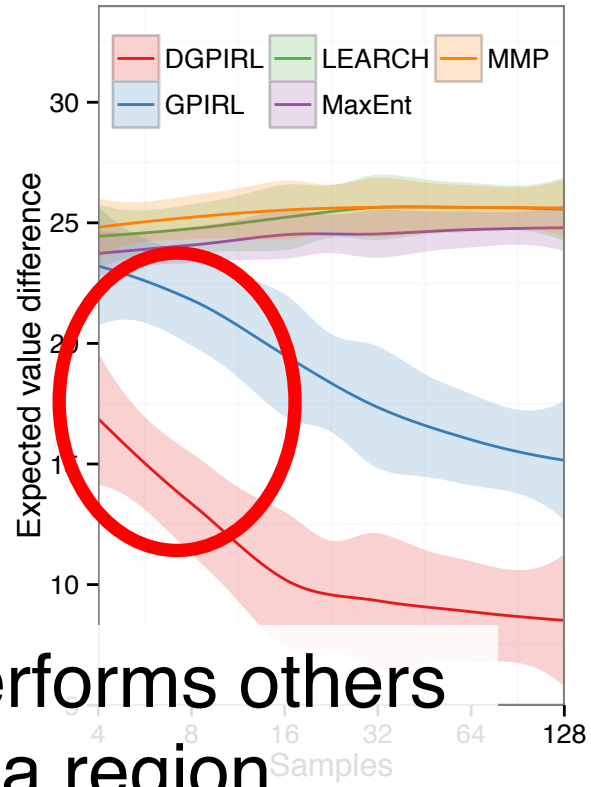


DGP-IRL outperforms GPIRL in this more complex case

Plot of EVD for the world where demos are given



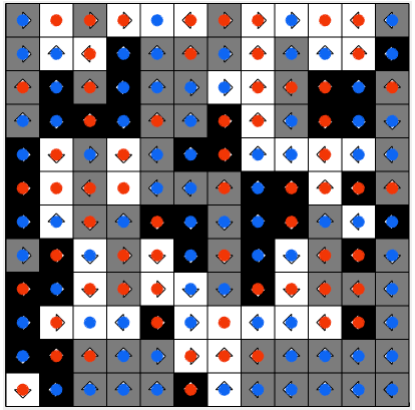
Plot of EVD for a new world where demos are not available



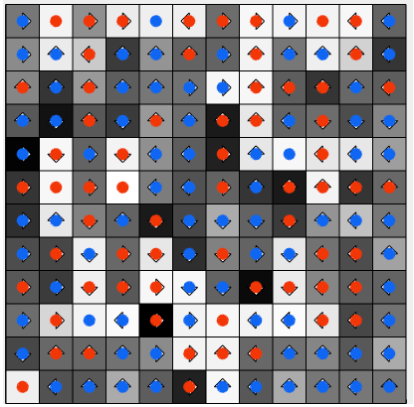
DGP-IRL outperforms others in the samll data region

DGP-IRL outperforms GPIRL in this more complex case

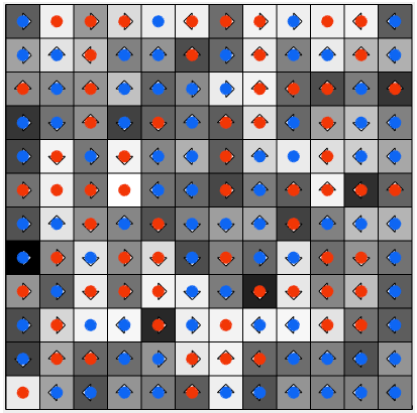
Ground truth



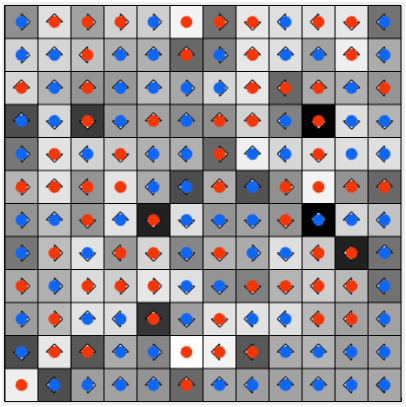
DGP-IRL



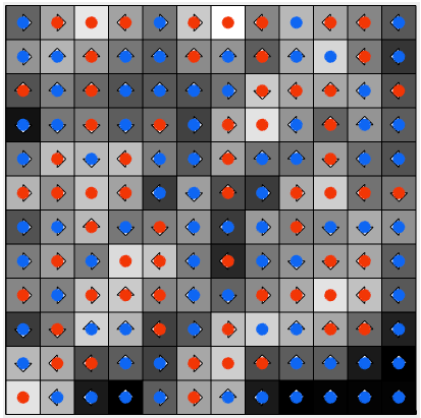
GPIRL



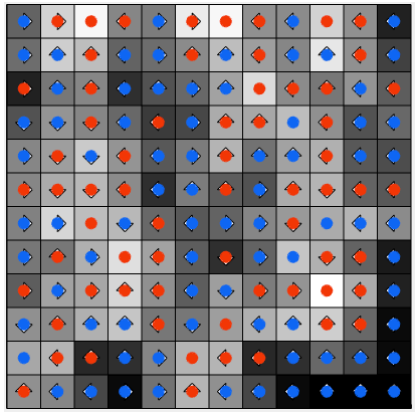
FIRL



MaxEnt

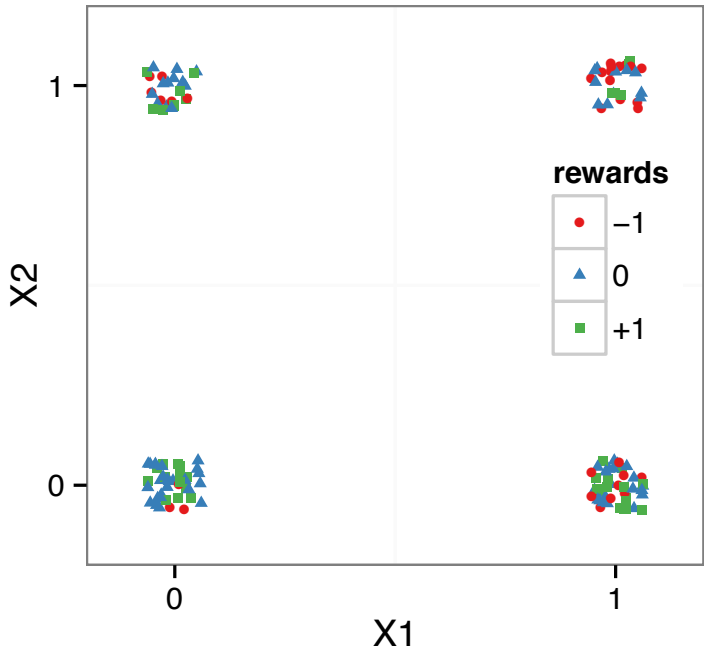


MMP

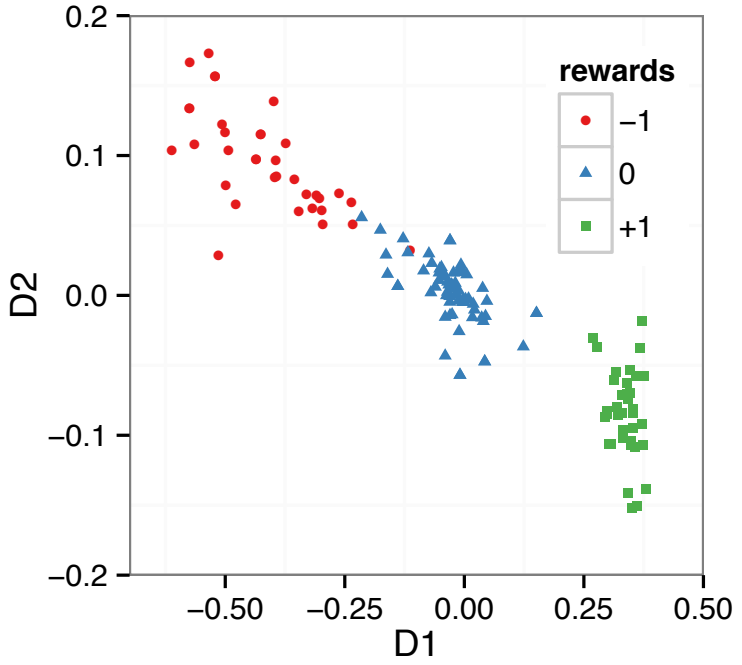


DGP-IRL learns the most succinct feature for the reward

The reward is not separable in the original feature space…



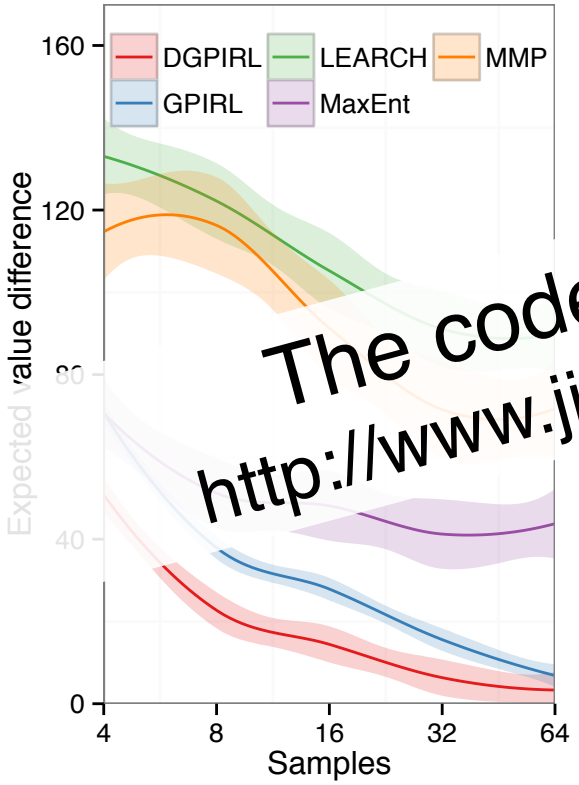
…but separable in the latent space



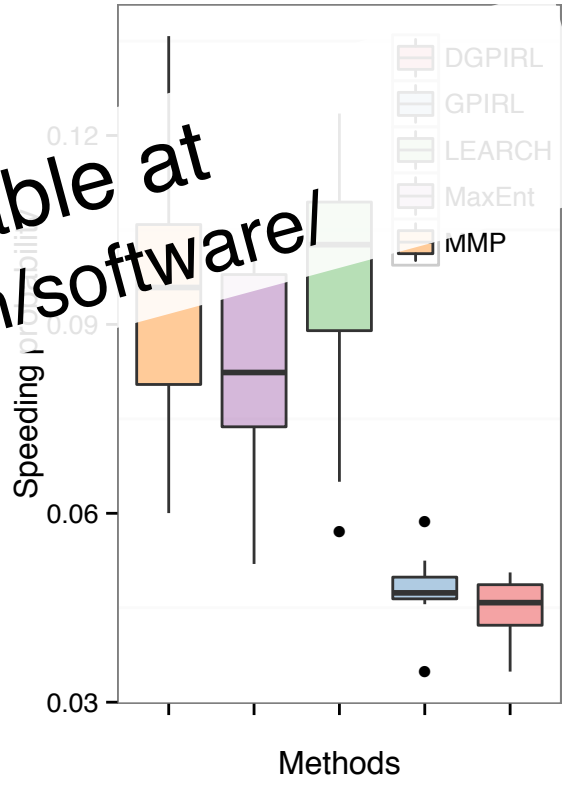
DGP-IRL also outperforms in learning the driving behavior

The goal is to navigate the robot car as fast as possible, but avoid speeding when the police car is nearby.

DGP-IRL outperforms in EVD



…and can avoid tickets



The code is available at http://www.jinming.tech/software/

Outline

- Inverse reinforcement learning (IRL) problem formulation
- Gaussian process reward modeling
- Incorporating the representation learning
- Variational inference
- Experiments
- **Conclusion**

How to learn an agent's intention by observing *a limited number* of demonstrations?

- Model the reward function as a deep GP to handle complex characteristics.
- This enables simultaneous feature state representation learning and IRL reward estimation.
- Train the model through variational inference to guard against overfitting, thus work efficiently with limited demonstrations.

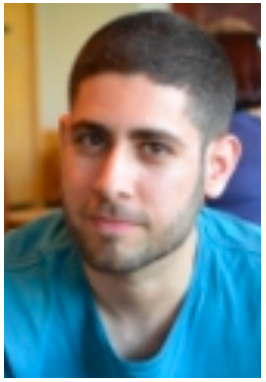
Future works

- DGP-IRL enable easy incorporation of side knowledge (through priors on the latent space) to IRL.
- Combine deep GPs with other complicated inference engines, e.g., selective attention models (Gregor et al., 2015).
- Application domains: mobile sensing for health, building and grid controls, multiobjective control, and human-in-the-loop gamification.

Incomplete List of Collaborators

Amazon.com, UK

- Andreas Damianou



UC Berkeley

- Pieter Abbeel
- Costas Spanos

SinBerBEST

Additional slides

DGP-IRL also outperforms in learning the driving behavior

- Three-lane highway, vehicles of specific class (civilian or police) and category (car or motorcycle) are positioned at random, driving at the same constant speed.
- The robot car can switch lanes and navigate at up to three times the traffic speed.
- The state is described by a continuous feature which consists of the closest distances to vehicles of each class and category in the same lane, together with the left, right, and any lane, both in the front and back of the robot car, in addition to the current speed and position.