Robot Learning

Gaussian Process Inverse Reinforcement Learning

Prof. Songhwai Oh
ECE, SNU

Review: Gaussian Process Regression (GPR)

f and f_* are jointly Gaussian, hence, for any finite number of measurements at $x_1, \ldots, x_n$ and x_*, (again assuming $m(x) = 0$)

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{pmatrix}\right),$$

where $[K(X,X)]_{ij} = k(x_i, x_j)$.

Recall that the conditional distribution of a jointly Gaussian random vector $[\mathbf{x}^T \ \mathbf{y}^T]^T$ is such that $\mathbf{x}|\mathbf{y} \sim \mathcal{N}(\mathbb{E}(\mathbf{x}|\mathbf{y}), \Sigma_{x|y})$, where

$$\mathbb{E}(\mathbf{x}|\mathbf{y}) = \mathbb{E}(\mathbf{x}) + \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \mathbb{E}(\mathbf{y})) \quad (1)$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}. \quad (2)$$

By conditioning, we get

$$f_*|X_*, X, f \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}f, \\ K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*))$$

Review: Gaussian Process Regression (GPR)

Let $y(x) = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Then $\mathbf{cov}(y(x_p), y(x_q)) = K(x_p, x_q) + \sigma_n^2 \delta_{pq}$ or in a matrix form

$$\mathbf{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 \mathbb{I}$$

The joint distribution between y and f_* is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) + \sigma_n^2 \mathbb{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

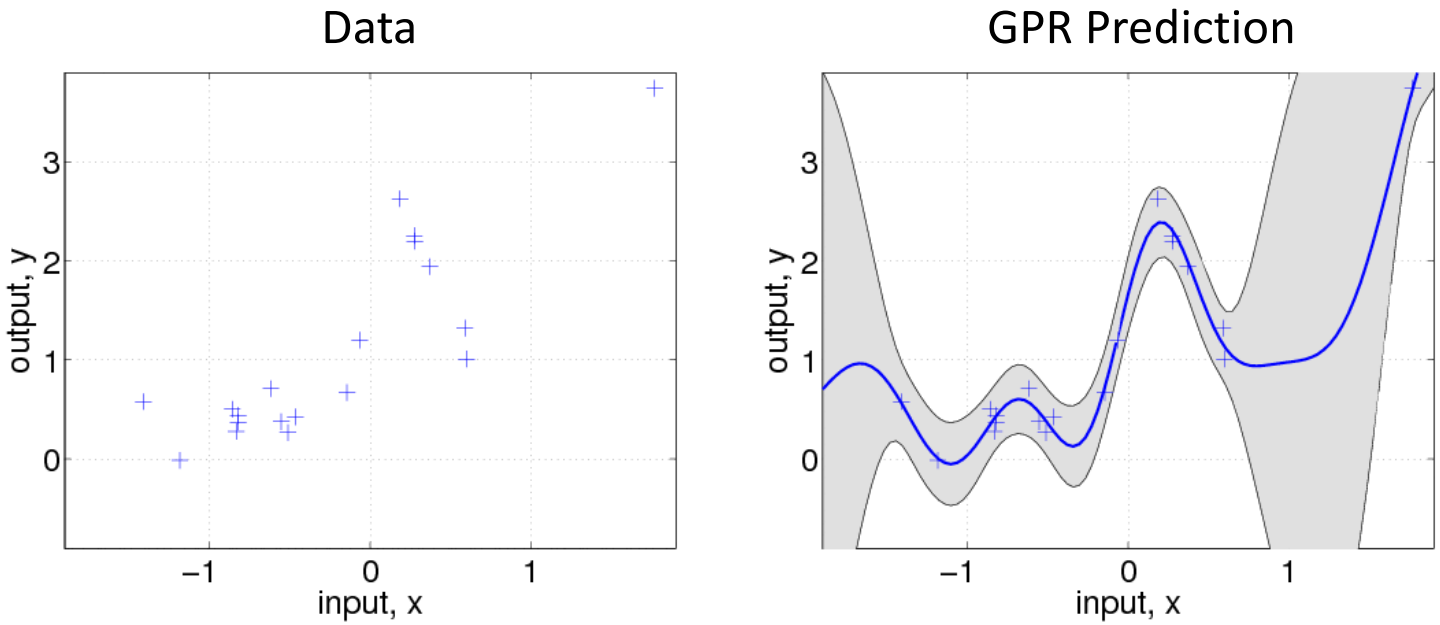
By conditioning, we get

$$f_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)),$$

$$\bar{f}_* = K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$$

$$\mathbf{cov}(f_*) = K(X_*, X_*) - K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} K(X, X_*))$$

Comments on GPR



- **Pros**: principled, probabilistic, **predictive uncertainty**
- **Cons:** computationally intensive (n x n matrix inversion)

- GPR uses all data: $f(x_*) = \sum_{i=1}^{n} \alpha_i k(x_i, x_*)$, where $\alpha = \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$
- cf. SVM is sparse: $f(x_*) = \sum_{i \in \mathcal{S}} a_i k(x_i, x_*) + b$

[Reference]

Joaquin Quiñonero-Candela, and Carl Edward Rasmussen. "**A unifying view of sparse approximate Gaussian process regression**." *Journal of Machine Learning Research* 6 (2005): 1939-1959.

SPARSE GAUSSIAN PROCESSES

Sparse Approximation of a Gaussian Process

- $\mathbf{u} = [u_1 \ \ldots \ u_m]^T$: **pseudo-inputs**, a.k.a. inducing variables, support points, and an active set.

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}, \mathbf{u}) d\mathbf{u} = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u},$$

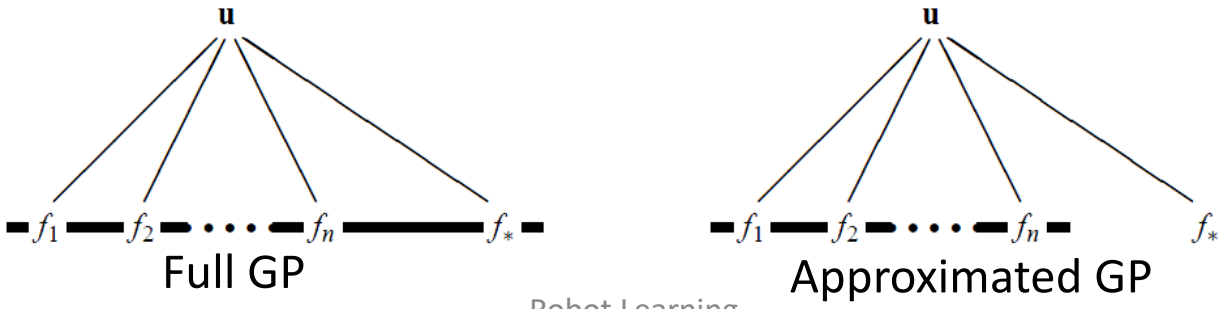
  where $p(\mathbf{u}) = \mathcal{N}(0, \mathbf{K}_{uu})$.

- Approximation by introducing a conditional independence assumption:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) = \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}.$$

$$\begin{aligned} p(\mathbf{f}|\mathbf{u}) &= \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}) , \\ p(\mathbf{f}_*|\mathbf{u}) &= \mathcal{N}(K_{*,\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, K_{*,*} - Q_{*,*}) \\ Q_{\mathbf{a},\mathbf{b}} &\triangleq K_{\mathbf{a},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} K_{\mathbf{u},\mathbf{b}} \end{aligned}$$

- Many sparse approximation methods for GPs can be formulated under this framework. $\mathbf{u}$ can be either fixed or learned from data.
- Computational complexity is reduced to $O(m^2 n)$ from $O(n^3)$.



Subset of Regressors (SoR)

- Approximation (deterministic):

$$q_{SoR}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{fu}K_{uu}^{-1}\mathbf{u}, 0) \qquad q_{SoR}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(K_{*u}K_{uu}^{-1}\mathbf{u}, 0)$$

- Joint prior between $\mathbf{f}$ and $\mathbf{f}_*$ (recall that $p(\mathbf{u}) = \mathcal{N}(0, \mathbf{K}_{uu})$):

$$q_{SoR}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(0, \begin{bmatrix} K_{fu}K_{uu}^{-1}K_{uf} & K_{fu}K_{uu}^{-1}K_{u*} \\ K_{*u}K_{uu}^{-1}K_{uf} & K_{*u}K_{uu}^{-1}K_{u*} \end{bmatrix}\right) = \mathcal{N}\left(0, \begin{bmatrix} Q_{ff} & Q_{f*} \\ Q_{*f} & Q_{**} \end{bmatrix}\right),$$

where $Q_{ab} = K_{au}K_{uu}^{-1}K_{ub}$.

- Predictive distribution:

$$\begin{aligned} q_{SoR}(\mathbf{f}_*|\mathbf{y}) &= \mathcal{N}(Q_{*f}(Q_{ff} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, Q_{**} - Q_{*f}(Q_{ff} + \sigma_n^2\mathbf{I})^{-1}Q_{f*}) \\ &= \mathcal{N}(\sigma_n^{-2}K_{*u}\Sigma K_{uf}\mathbf{y}, K_{*u}\Sigma K_{u*}), \end{aligned}$$

where $\Sigma = (\sigma_n^{-2}K_{uf}K_{fu} + K_{uu})^{-1}$.

Bernhard W. Silverman. **Some aspects of the spline smoothing approach to non-parametric regression curve fitting**. *J. Roy. Stat. Soc. B*, 47(1):1–52, 1985.

Deterministic Training Conditional (DTC)

- Approximation (deterministic in training):

$$q_{DTC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{fu}K_{uu}^{-1}\mathbf{u}, 0) \qquad q_{DTC}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Joint prior between $\mathbf{f}$ and $\mathbf{f}_*$:

$$q_{DTC}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{ff} & Q_{f*} \\ Q_{*f} & K_{**} \end{bmatrix}\right),$$

where $Q_{ab} = K_{au}K_{uu}^{-1}K_{ub}$.

- Predictive distribution:

$$\begin{aligned} q_{DTC}(\mathbf{f}_*|\mathbf{y}) &= \mathcal{N}(Q_{*f}(Q_{ff} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, K_{**} - Q_{*f}(Q_{ff} + \sigma_n^2\mathbf{I})^{-1}Q_{f*}) \\ &= \mathcal{N}(\sigma_n^{-2}K_{*u}\Sigma K_{uf}\mathbf{y}, K_{**} - Q_{**} + K_{*u}\Sigma K_{u*}), \end{aligned}$$

where $\Sigma = (\sigma_n^{-2}K_{uf}K_{fu} + K_{uu})^{-1}$.

Matthias Seeger, Christopher K. I.Williams, and Neil Lawrence. **Fast forward selection to speed up sparse Gaussian process regression**. *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.

Fully Independent Training Conditional (FITC)

- Approximation:

$$q_{FITC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{fu}K_{uu}^{-1}\mathbf{u}, \mathbf{diag}[K_{ff} - Q_{ff}]) \qquad q_{FITC}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Joint prior between $\mathbf{f}$ and $\mathbf{f}_*$:

$$q_{FITC}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{ff} + \mathbf{diag}[K_{ff} - Q_{ff}] & Q_{f*} \\ Q_{*f} & K_{**} \end{bmatrix}\right),$$

where $Q_{ab} = K_{au}K_{uu}^{-1}K_{ub}$.

- Predictive distribution:

$$\begin{aligned} q_{FITC}(\mathbf{f}_*|\mathbf{y}) &= \mathcal{N}(Q_{*f}(Q_{ff} + \Lambda)^{-1}\mathbf{y}, K_{**} - Q_{*f}(Q_{ff} + \Lambda)^{-1}Q_{f*}) \\ &= \mathcal{N}(K_{*u}\Sigma K_{uf}\Lambda^{-1}\mathbf{y}, K_{**} - Q_{**} + K_{*u}\Sigma K_{u*}), \end{aligned}$$

where $\Sigma = (K_{uu} + K_{uf}\Sigma^{-1}K_{fu})^{-1}$ and $\Lambda = \mathbf{diag}[K_{ff} - Q_{ff} + \sigma_n^2\mathbf{I}]$.

Edward Snelson and Zoubin Ghahramani. **Sparse Gaussian processes using pseudo-inputs**. *Advances in Neural Information Processing Systems 18*, Cambridge, Massachussetts, 2006.

Partially Independent Training Conditional (PITC)

- Approximation:

$$q_{PITC}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{fu}K_{uu}^{-1}\mathbf{u}, \mathbf{blockdiag}[K_{ff} - Q_{ff}]) \qquad q_{PITC}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u})$$

- Joint prior between $\mathbf{f}$ and $\mathbf{f}_*$:

$$q_{PITC}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(0, \begin{bmatrix} Q_{ff} + \mathbf{blockdiag}[K_{ff} - Q_{ff}] & Q_{f*} \\ Q_{*f} & K_{**} \end{bmatrix}\right),$$

where $Q_{ab} = K_{au}K_{uu}^{-1}K_{ub}$.

- Predictive distribution:

$$\begin{aligned} q_{FITC}(\mathbf{f}_*|\mathbf{y}) &= \mathcal{N}(Q_{*f}(Q_{ff} + \Lambda)^{-1}\mathbf{y}, K_{**} - Q_{*f}(Q_{ff} + \Lambda)^{-1}Q_{f*}) \\ &= \mathcal{N}(K_{*u}\Sigma K_{uf}\Lambda^{-1}\mathbf{y}, K_{**} - Q_{**} + K_{*u}\Sigma K_{u*}), \end{aligned}$$

where $\Sigma = (K_{uu} + K_{uf}\Sigma^{-1}K_{fu})^{-1}$ and $\Lambda = \mathbf{blockdiag}[K_{ff} - Q_{ff} + \sigma_n^2\mathbf{I}]$.

S. Levine, Z. Popovic, and V. Koltun, "**Nonlinear inverse reinforcement learning with Gaussian processes**," Advances in Neural Information Processing Systems (NIPS), Dec, 2011.

GAUSSIAN PROCESS INVERSE REINFORCEMENT LEARNING (GPIRL)

Setting

- Markov Decision Processes and a Stochastic Policy
  - Markov decision process (MDP): $\boldsymbol{M} = \{\boldsymbol{S}, \boldsymbol{F}, \boldsymbol{A}, \boldsymbol{T}, \gamma, \boldsymbol{r}\}$
    - Goal: find the optimal policy $\boldsymbol{\pi}$ which maximizes the sum of $\boldsymbol{r}$
  - Inverse Reinforcement Learning (IRL): $\boldsymbol{M}/\boldsymbol{r}$ with $\boldsymbol{\mathcal{D}} = \{\varsigma_i\}$
    - Goal: find the underlying reward $\boldsymbol{r}$, which best explains $\boldsymbol{\mathcal{D}}$
    - Assumption: $\boldsymbol{\mathcal{D}}$ comes from experts (optimal policy)
  - Stochastic Policy
    - Motivation: In practice, demonstrations are inconsistent and noisy
      - $\pi(a|s) = \frac{\exp Q(s,a)}{\sum_{a'} \exp Q(s,a')}$
    - Stochastic policy can be computed by the soft Bellman equation
      - $Q(s,a) = \boldsymbol{r}(s,a) + \gamma \sum_{s'} \boldsymbol{T}(s'|s,a) V(s')$
      - $V(s) = \log \sum_{a'} \exp Q(s,a')$

GPIRL

- Sparse Gaussian process regression method is used to estimate a reward function
  - Reward function is represented by the mean of posterior distribution of sparse Gaussian process
  - $\boldsymbol{r} = K_{\boldsymbol{F}u} K_{uu}^{-1} u$
- GPIRL is capable of modeling the non-linear reward structure
- Probabilistic model using stochastic policy
- Objective function:

$$\log P(\boldsymbol{\mathcal{D}}|\boldsymbol{r}) + \log P(u|X_u, \theta) + \log P(\theta|X_u)$$

  - $\log P(\boldsymbol{\mathcal{D}}|\boldsymbol{r})$: data fitting
  - $\log P(u|X_u, \theta)$: regularization of inducing output u
  - $\log P(\theta|X_u)$: hyperparameter regularization

Data Log Likelihood

- Softmax policy function:

$$\pi(a|s) = \frac{\exp(\mathbf{Q}^{\mathbf{r}}_{sa})}{\sum_{a'} \exp(\mathbf{Q}^{r}_{sa'})}$$

- Value functions are:

$$\begin{aligned} \mathbf{Q}^{\mathbf{r}} &= \mathbf{R} + \gamma \mathcal{T} \mathbf{V}^{\mathbf{r}} \\ \mathbf{V}^{\mathbf{r}}_{s} &= \log \sum_{a} \exp \mathbf{Q}^{\mathbf{r}}_{sa} \end{aligned}$$

- Hence, we have

$$\pi(a|s) = \frac{\exp(\mathbf{Q}^{\mathbf{r}}_{sa})}{\sum_{a'} \exp(\mathbf{Q}^{r}_{sa'})} = \frac{\exp(\mathbf{Q}^{\mathbf{r}}_{sa})}{\exp(\mathbf{V}^{\mathbf{r}}_{s})} = \exp(\mathbf{Q}^{\mathbf{r}}_{sa} - \mathbf{V}^{\mathbf{r}}_{s})$$

- Data log likelihood under $\mathbf{r}$:

$$\begin{aligned} \log P(\mathcal{D}|\mathbf{r}) &= \sum_{i} \sum_{t} \log P(a_{i,t}|s_{i,t}) = \sum_{i} \sum_{t} \log \pi(a_{i,t}|s_{i,t}) \\ &= \sum_{i} \sum_{t} \left(\mathbf{Q}^{\mathbf{r}}_{s_{i,t} a_{i,t}} - \mathbf{V}^{\mathbf{r}}_{s_{i,t}} \right) \end{aligned}$$

Objective Function

$$P(\mathbf{u}, \boldsymbol{\theta}|\mathcal{D}, \mathbf{X_u}) \propto P(\mathcal{D}, \mathbf{u}, \boldsymbol{\theta}|\mathbf{X_u}) = \left[\int_{\mathbf{r}} \underbrace{P(\mathcal{D}|\mathbf{r})}_{\text{IRL term}} \underbrace{P(\mathbf{r}|\mathbf{u}, \boldsymbol{\theta}, \mathbf{X_u})}_{\text{GP posterior}} d\mathbf{r} \right] \underbrace{P(\mathbf{u}, \boldsymbol{\theta}|\mathbf{X_u})}_{\text{GP probability}}$$

$$\log P(\mathcal{D}|\mathbf{r}) = \sum_i \sum_t \log P(a_{i,t}|s_{i,t}) = \sum_i \sum_t \left(\mathbf{Q}^{\mathbf{r}}_{s_{i,t}a_{i,t}} - \mathbf{V}^{\mathbf{r}}_{s_{i,t}} \right)$$

$$\log P(\mathbf{u}, \boldsymbol{\theta}|\mathbf{X_u}) = -\frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{K}^{-1}_{\mathbf{u},\mathbf{u}}\mathbf{u} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}| - \frac{n}{2}\log 2\pi + \log P(\boldsymbol{\theta})$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \beta \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \boldsymbol{\Lambda} (\mathbf{x}_i - \mathbf{x}_j) \right)$$

Assumption:

- Deterministic training conditional: $P(\mathbf{r}|\mathbf{u}, \boldsymbol{\theta}, \mathbf{X_u}) = N(\mathbf{K_{ru}}\mathbf{K_{uu}^{-1}}\mathbf{u}, 0)$

$$\log P(\mathcal{D}, \mathbf{u}, \boldsymbol{\theta}|\mathbf{X_u}) = \underbrace{\log P(\mathcal{D}|\mathbf{r} = \mathbf{K}^{\mathrm{T}}_{\mathbf{r},\mathbf{u}}\mathbf{K}^{-1}_{\mathbf{u},\mathbf{u}}\mathbf{u})}_{\text{IRL log likelihood}} + \underbrace{\log P(\mathbf{u}, \boldsymbol{\theta}|\mathbf{X_u})}_{\text{GP log likelihood}}$$

Kyungjae Lee, Sungjoon Choi, and Songhwai Oh, "**Inverse Reinforcement Learning with Leveraged Gaussian Processes**," in Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2016.

LEVERAGED GAUSSIAN PROCESS INVERSE REINFORCEMENT LEARNING (LGPIRL)

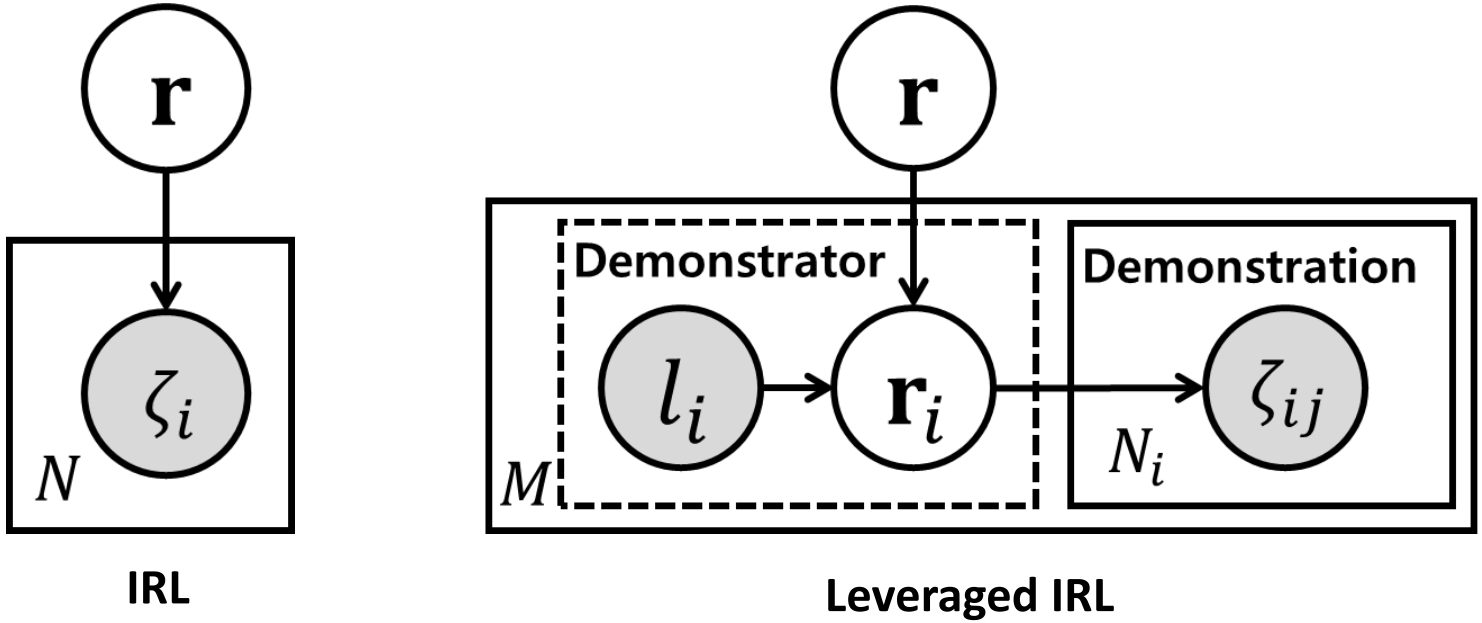
Proposed Demonstrator Model

Motivation

- General assumption of standard IRL
  - Demonstrations $\mathcal{D}$ come from experts (optimal policy)
  - Expert always obeys the optimal policy π which is induced by the underlying reward r
- Drawbacks of the assumption
  - Demonstrations of experts are often distributed near high reward regions
  - The resulting reward function learned from experts' demonstrations cannot approximate low reward regions accurately
- Main Idea
  - Proposed model can incorporate demonstrations about both what to do (positive) and what not to do (negative)
  - Demonstrations about what not to do can provide information near low reward regions

Proposed Demonstrator Model

- A reward function is modeled as a leveraged Gaussian process
- Leverage parameter indicates proficiency of demonstrator
  - Positive proficiency indicates positive demonstrator
  - Negative proficiency indicates negative demonstrator



Leveraged Inverse Reinforcement Learning

Problem Formulation

- Objective Function

$$\max_{\boldsymbol{u},\boldsymbol{\theta}=\{\boldsymbol{\beta},\boldsymbol{\Lambda}\}} \sum_{\boldsymbol{i}} \log \boldsymbol{P}(\boldsymbol{\mathcal{D}}|\boldsymbol{r}=\boldsymbol{K_{ru}K_{uu}^{-1}u},\boldsymbol{l_i}) + \log \boldsymbol{P}(\boldsymbol{u}|\boldsymbol{X_u},\boldsymbol{\theta}) + \log \boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{X_u})$$

IRL Likelihood LGP Prior Prior

- $\log \boldsymbol{P}(\boldsymbol{\mathcal{D}}|\boldsymbol{r}=\boldsymbol{K_{ru}K_{uu}^{-1}u},\boldsymbol{l}) = \sum_i \sum_j \sum_t \boldsymbol{Q}_i(\boldsymbol{s_{jt}},\boldsymbol{a_{jt}}) - \boldsymbol{V}_i(\boldsymbol{s_{jt}})$
- $\log \boldsymbol{P}(\boldsymbol{u}|\boldsymbol{X_u},\boldsymbol{\theta}) = -\frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{K_{uu}^{-1}u} - \frac{\boldsymbol{1}}{\boldsymbol{2}}\log|\boldsymbol{K_{uu}}| - \frac{\boldsymbol{n}}{\boldsymbol{2}}\log \boldsymbol{2\pi}$
- $\log \boldsymbol{P}(\boldsymbol{\theta}|\boldsymbol{X_u}) = -\frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{tr}(\boldsymbol{K_{uu}^{-2}}) - \sum_k(\boldsymbol{\lambda_k}+\boldsymbol{1})$
- Objective function is maximized by a gradient ascent method

Leveraged Kernel Function

A (mean-zero) Gaussian process is fully specified by its covariance (kernel) function.

Loosely speaking, a kernel function is usually a decreasing function of some distance measure between two inputs.

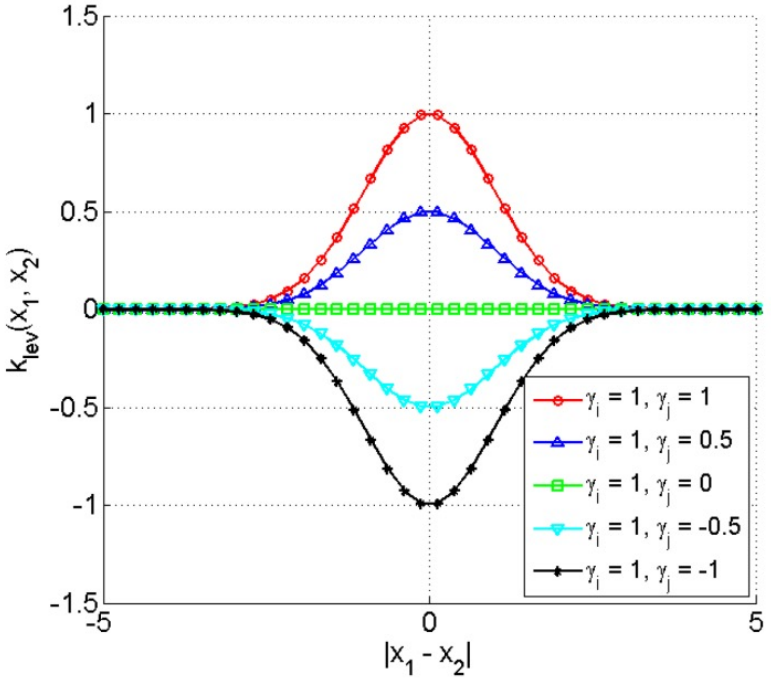
We propose a leveraged kernel function which incorporates not only the inputs of the training data but also the leverage γ of each training data.

$$k(\mathbf{x}_i, \mathbf{x}_j) = (1 - |\gamma_i - \gamma_j|)\, k_{SE}(\mathbf{x}_i, \mathbf{x}_j)$$

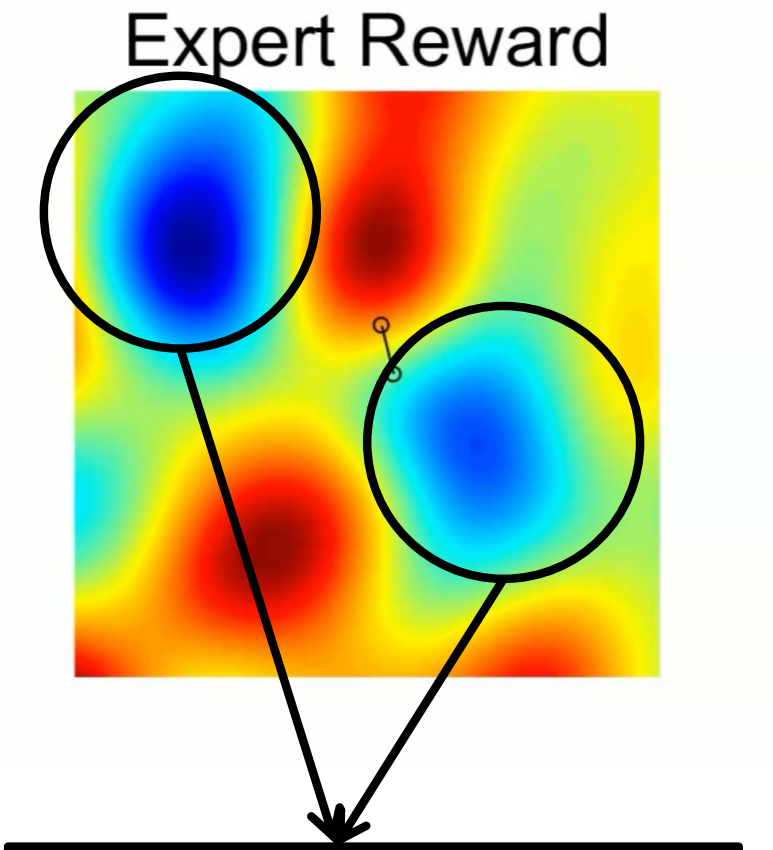
for $\gamma \in [-1, 1]$

Continuous version:

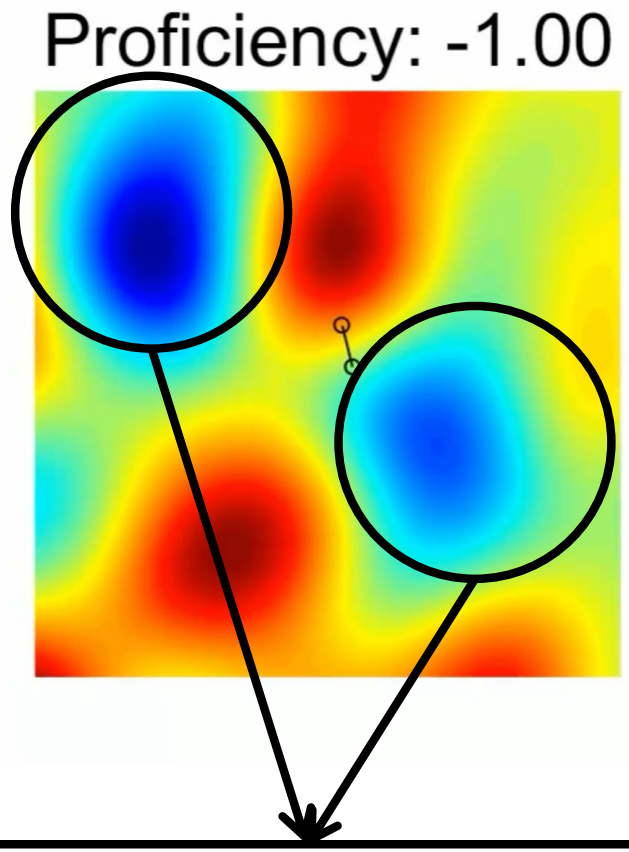
$$k_{SL}(\mathbf{x}_i, \mathbf{x}_j) = \cos\left(\frac{\pi}{2}(\gamma_i - \gamma_j)\right) k_{\text{PSD}}(\mathbf{x}_i, \mathbf{x}_j)$$



Leveraged Inverse Reinforcement Learning



Lack of demonstrations near the low reward region

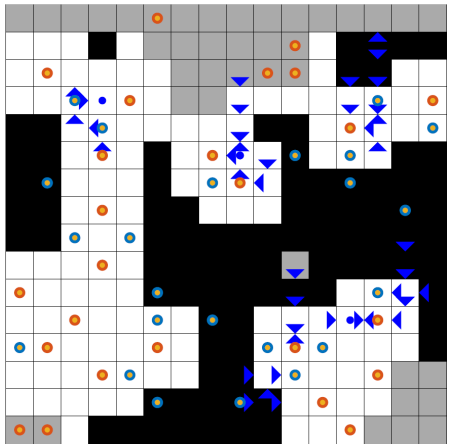


Negative demonstrations effectively provide information near the low reward region

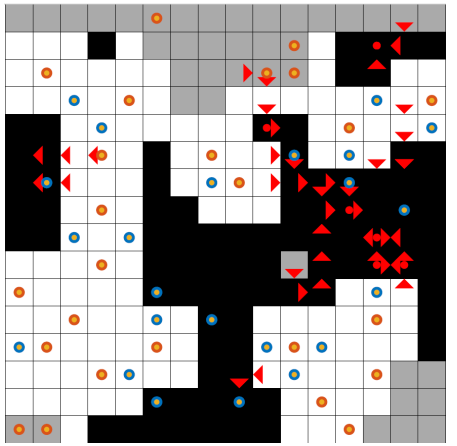
Benefits of Negative Demonstrations

Objectworld

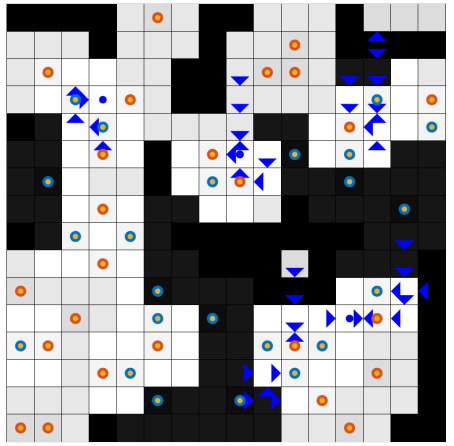
- First row: positive and negative demonstrations
- Second row: reconstruction results of GPRIL and LIRL (proposed)
- The result from LIRL is more accurate than GPIRL
- Negative demonstrations provide information about the low reward region



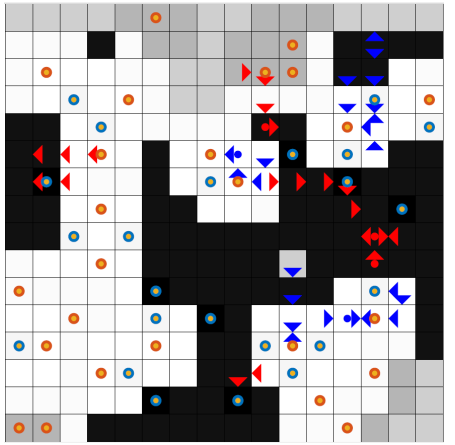
Positive demonstrations



Negative demonstrations



GPIRL result



LIRL result

Experimental Results

- Compared methods
  - Maximum margin planning (MMP)
  - Gaussian process IRL (GPIRL)
  - Nonparametric Bayesian feature constructing method (NPBFIRL)
  - Apprenticeship Learning (AL)
  - Maximum entropy IRL (MaxEntIRL)
  - Learning to search (Learch)
  - OptV
  - Feature construction for IRL (FIRL)
  - Bayesian IRL (BIRL)
- Performance measure
  - Expected Value Difference $\left|v^{\pi^*} - v^{\pi}\right|$
    - Value difference between the optimal policy and the policy induced by learned reward function

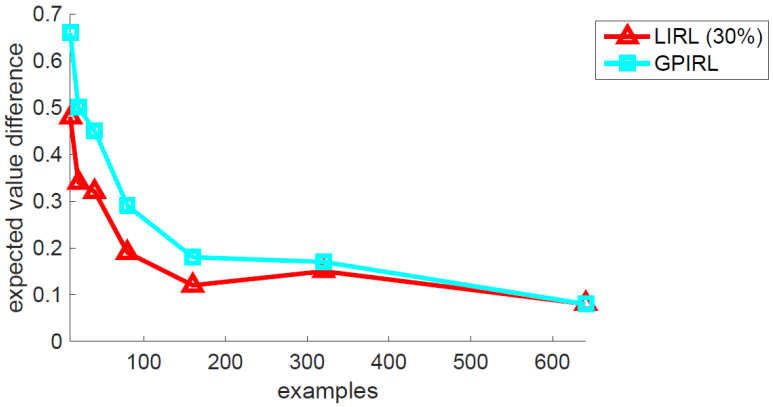
Experimental Results

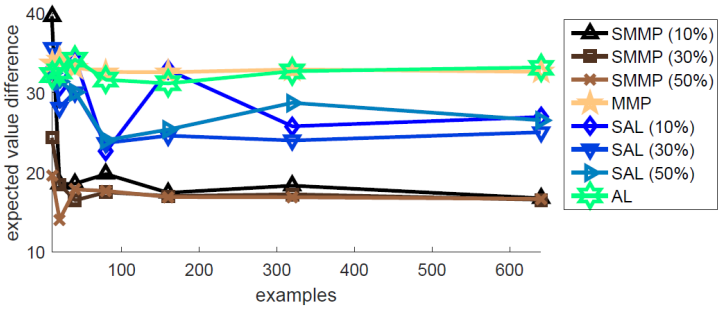
- Objectworld
  - 32×32 grid world
  - Five actions (up, down, right, left, stay)
  - Objects with random colors are randomly deployed
    - Colors are selected from 2 distinct colors
  - Reward function
    - Two specific colors c_1 and c_2
    - $d_i(s)$ is the Euclidean distance from state s to the nearest object whose outer color is c_i
    - $$r(s) \begin{cases} 1, & d_1(s) < 3 \wedge d_2(s) < 2 \\ -2, & d_1(s) < 3 \wedge d_2(s) \geq 2 \\ 0, & otherwise \end{cases}$$

Experimental Results

- Objectworld E.V.D result

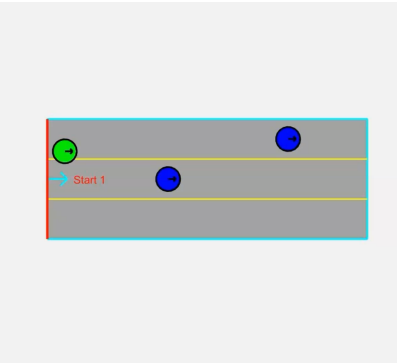
Algorithms	Sample Size						
	10	20	40	80	160	320	640
LIRL (10%)	**0.42**	**0.3**	0.20	0.24	**0.12**	0.16	0.11
LIRL (30%)	0.48	0.34	0.32	**0.19**	**0.12**	0.15	**0.08**
LIRL (50%)	0.79	0.36	**0.17**	0.43	0.23	**0.08**	0.13
SMMP (10%)	39.62	18.53	18.60	19.79	17.45	18.34	16.76
SMMP (30%)	24.35	18.51	16.50	17.51	17.04	17.25	16.57
SMMP (50%)	19.59	14.09	17.86	17.68	16.91	16.88	16.68
SAL (10%)	33.00	30.69	34.28	22.62	32.86	25.79	26.95
SAL (30%)	35.61	28.17	30.10	23.70	24.61	24.01	25.04
SAL (50%)	32.49	31.40	30.08	24.04	25.38	28.73	26.54
BIRL	14.72	15.37	15.25	13.26	12.52	12.48	12.91
GPIRL	0.66	0.50	0.45	0.29	0.18	0.17	**0.08**
NPBFIRL	10.76	10.43	9.99	9.74	10.19	10.17	10.27
MEIRL	18.33	15.76	15.68	14.29	13.79	13.50	13.70
OptV	40.77	36.73	29.78	22.61	14.78	6.77	2.08
MMP	33.72	34.20	33.15	32.61	32.57	32.92	32.63
AL	32.20	32.69	34.16	31.63	31.17	32.70	33.18
LEARCH	36.59	36.20	35.17	34.27	32.30	32.06	31.66
FIRL	42.07	42.36	41.82	42.30	42.15	42.20	42.31



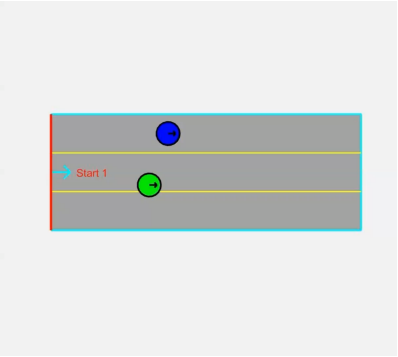


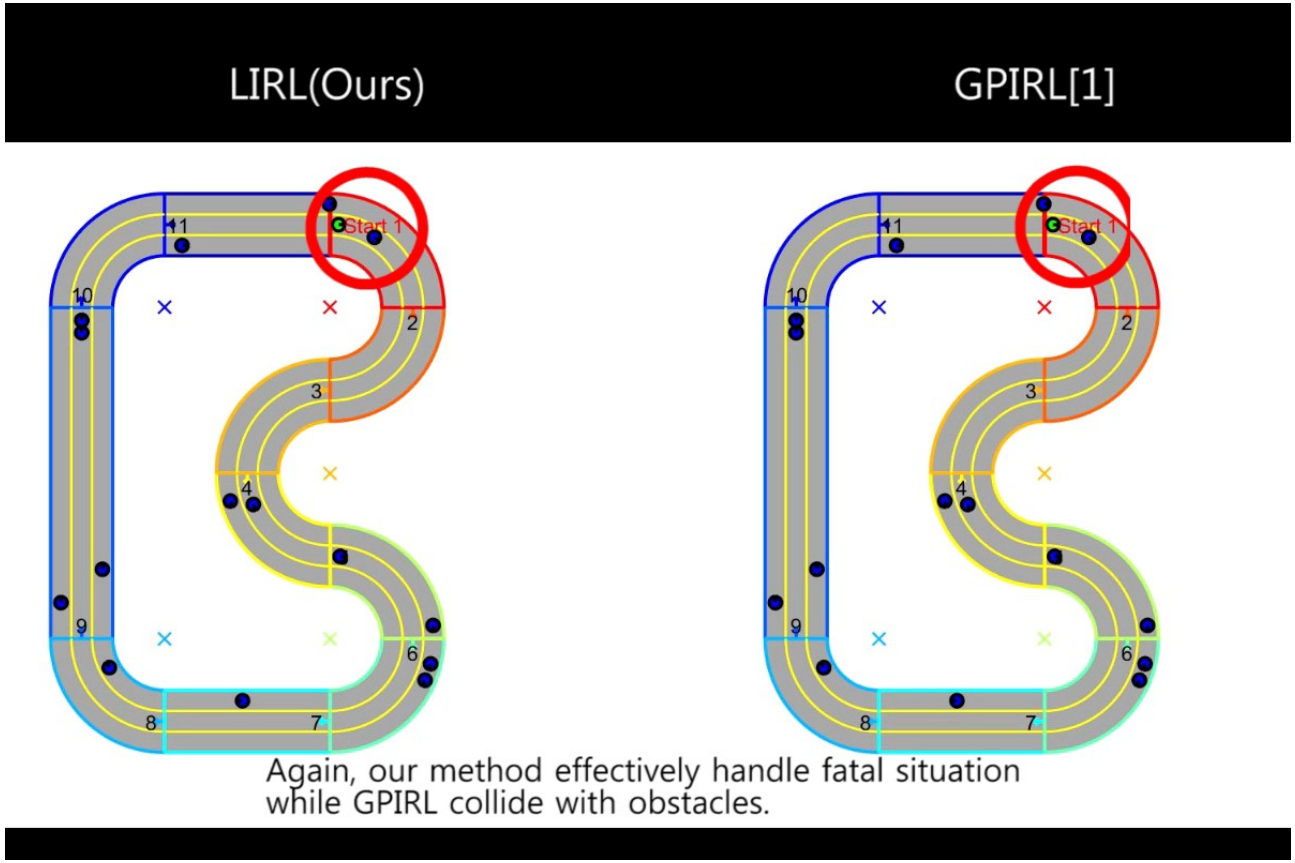
Track Driving Learning from Do's and Don'ts

Do's



Don'ts





State: position and heading
Action: linear velocity and angular velocity
Dynamic model: unicycle dynamics