Nonlinear Inverse Reinforcement Learning with Gaussian Processes

Sergey Levine
Stanford University

Zoran Popović
University of Washington

Vladlen Koltun
Stanford University

1. Introduction

Inverse Reinforcement Learning (IRL): learning a **reward function** in a Markov decision process (MDP) from expert demonstrations. Used to generalize the expert's policy to unobserved situations. Applications include learning policies from examples, inferring goals, specifying tasks by demonstration.

Standard Method: Many rewards induce the expert's behavior, and we must select one that exhibits meaningful structure and generalizes effectively. A common approach is to learn rewards that are **linear** in some set of features (e.g. $R = \sum_i \theta_i f_i$), by assuming that the examples are **optimal** under the unknown reward.

Challenge: A good linear basis for the reward may not be known, and real-world demonstrations may not be optimal. A **probabilistic** method is required that can reason about uncertain and suboptimal demonstrations, **select** features that are **relevant**, and build the reward as a **nonlinear** function.

GPIRL: The Gaussian Process Inverse Reinforcement Learning (GPIRL) algorithm models the reward as the output of a Gaussian process (GP). The GP prior **prevents overfitting**, **selects** relevant features with an automatic relevance detection (ARD) kernel, and can represent **complex and nonlinear** rewards. Unique challenges associated with learning GP outputs are handled with a novel hyperparameter prior (see Box 4). A probabilistic IRL model allows the method to handle suboptimal real-world demonstrations.

2. Background

Markov Decision Process: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathbf{r}\}$
$\mathcal{S}$ – states $\quad \mathcal{A}$ – actions $\quad \gamma$ – discount $\quad \mathbf{r}$ – reward
$\mathcal{T}$ – state transition probabilities: $\mathcal{T}_{s'}^{sa} = P(s'|s,a)$
Optimal policy $\pi^\star$ maximizes $E\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{r}_{s_t} | \pi^\star\right]$

Example Traces: $\mathcal{D} = \{\zeta_1, ..., \zeta_N\}$, where $\zeta_i = \{(s_{i,0}, a_{i,0}), ..., (s_{i,T}, a_{i,T})\}$ is a path with states $s_{i,t}$ and observed actions $a_{i,t}$.

Maximum Entropy IRL: Probability of a path modeled as $P(\zeta_i) \propto e^{\sum_t \mathbf{r}_{s_{i,t}}}$. Corresponding value function is the solution for a "soft" Bellman operator [2]:

$$\mathbf{Q}^{\mathbf{r}} = \mathbf{r} + \gamma \mathcal{T} \mathbf{V}^{\mathbf{r}} \qquad \mathbf{V}_s^{\mathbf{r}} = \log \sum_a \exp \mathbf{Q}_{sa}^{\mathbf{r}}$$

The probability of taking action a in state s is then given by:

$$P(a|s) = \exp(\mathbf{Q}_{sa}^{\mathbf{r}} - \mathbf{V}_s^{\mathbf{r}})$$

Intuitively, when "stakes" are high, action is deterministic, when all options are equal, the action is random. The maximum likelihood objective is:

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

Gaussian processeses model distributions over functions in which all values are jointly normally distributed, with a covariance given by a kernel $k(\mathbf{x}_i, \mathbf{x}_j)$. The ARD RBF kernel, with hyperparameters $\boldsymbol{\theta} = \{\beta, \boldsymbol{\Lambda}\}$, is given by:

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

Learning $\boldsymbol{\Lambda}$ automatically selects the relevant dimensions of $\mathbf{x}$. For outputs $\mathbf{u}$ and covariance $\mathbf{K}_{\mathbf{u},\mathbf{u}}$, the marginal likelihood is:

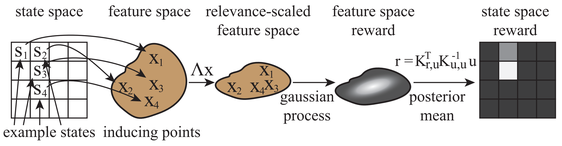
$$\mathcal{L}_{\mathcal{G}} = \underbrace{-\frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}}_{\text{penalizes poor fit}} - \underbrace{\frac{1}{2}\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}|}_{\text{penalizes complexity}} + \underbrace{\log P(\boldsymbol{\theta})}_{\text{hyperparameter prior}}$$

3. GPIRL Algorithm

GPIRL learns the hyperparameters $\boldsymbol{\theta}$ of the Gaussian process that represents the reward function, as well as its output $\mathbf{u}$ at a set of **inducing points** $\mathbf{X}_{\mathbf{u}}$ in feature space. Any set of feature values can be used, so long as it provides good coverage of the reward function. A simple choice that often works well is to choose the feature values of the states visited in the example trajectories.

The full reward can be recovered as the posterior mean of the GP, given by $\mathbf{r} = \mathbf{K}_{\mathbf{r},\mathbf{u}}^{\mathrm{T}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}$, where $\mathbf{K}_{\mathbf{r},\mathbf{u}}$ is the covariance between all states and the inducing points. The likelihood of $\mathbf{u}$ and $\boldsymbol{\theta}$ is therefore given by:

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



4. Gaussian Process with Learned Output

Unlike GP regression, GPIRL learns the **output** of a Gaussian process. This requires changes to the standard GP framework.

When $\boldsymbol{\Lambda}\mathbf{x}_i = \boldsymbol{\Lambda}\mathbf{x}_j$ due to zero entries in $\boldsymbol{\Lambda}$:

- $\mathbf{u}_i$ and $\mathbf{u}_j$ must be equal, so one of the two points is redundant
- $-\frac{1}{2}\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}|$ goes to infinity
- Does not happen in GP regression due to noise and fitting term, but in GPIRL, fitting term goes to zero as $\mathbf{u}$ goes to zero

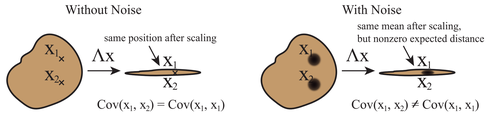
Noise: Although $\mathbf{u}$ is noiseless, we can have noise in $\mathbf{X}_{\mathbf{u}}$. This reflects uncertainty about the location of nonredundant inducing points.

Assume Gaussian noise with variance σ^2.

Expected distance in k^{th} feature is $(x_{ik} - x_{jk})^2 + 2\sigma^2$, and the kernel is:

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

No two points are deterministically related so long as $\mathrm{tr}(\boldsymbol{\Lambda}) > 0$.



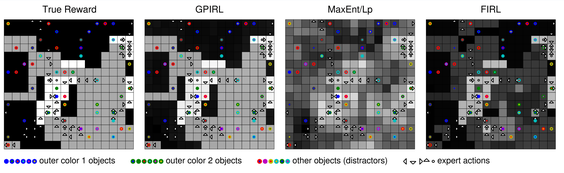
Hyperparameter prior: Degeneracies can also occur as $\boldsymbol{\Lambda} \to 0$ or $\beta \to 0$.

This is avoided with a hyperparameter prior that captures the belief that no inducing points are deterministically related.

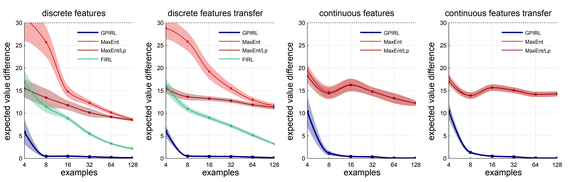
Deterministic relationship implies infinite partial correlation, so we penalize partial correlation $[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]_{ij}$:

$$\log P(\boldsymbol{\theta}) = -\frac{1}{2}\sum_{ij}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]_{ij}^2 = -\frac{1}{2}\mathrm{tr}(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2})$$

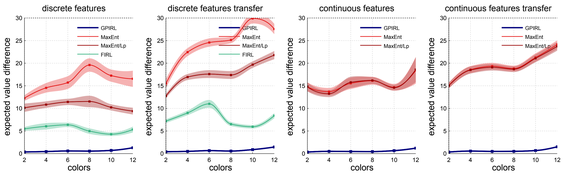
5. Results



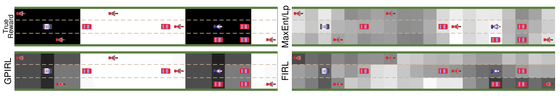
Example rewards learned on gridworld with randomly placed objects. Outer colors 1 and 2 are relevant, all other colors are irrelevant distractors. Features are distances to objects of each color. The reward learned by GPIRL closely resembles the true one, rewards learned with FIRL [1] and MaxEnt [2] do not.



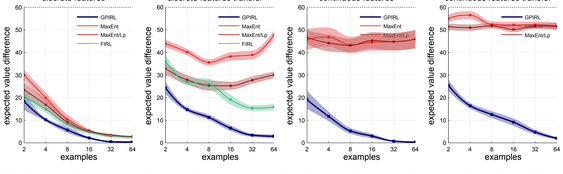
Transfer experiments. Features are either indicators for discrete distance bins or continuous values. Transfer results are obtained by learning on one gridworld and testing on 10 other random object placements.



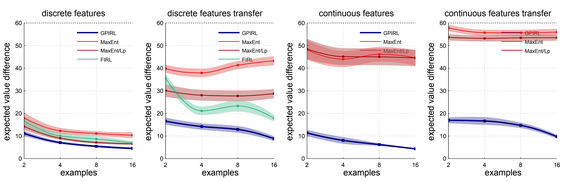
Transfer with increasing numbers of irrelevant object colors. Additional colors act as distractors. GPIRL suffers less from distractors than prior algorithms.



Example rewards learned from human demonstration on a highway driving task. The task requires driving fast when possible, slowing down near police vehicles. The color of the road shows the reward for traveling there at the highest speed. Both the true reward and GPIRL correctly penalize speeding near police, while prior methods do not accurately capture this relationship.



Highway driving with synthetic examples drawn from suboptimal policy. Features are distances to nearest car of each type in front, behind, and to the sides, as discrete bins or continuous values. Prior methods suffer on transfer tests, as their learned rewards are not represented in terms of the correct features.



Driving with human demonstrations. The expected value difference is computed against the stochastic policy under the true reward. GPIRL more accurately learned the policy the human expert was attempting to demonstrate.

This work was supported by NSF Graduate Research Fellowship DGE-0645962.

[1] S. Levine, Z. Popović, and V. Koltun. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems 23*. 2010.

[2] B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.