Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Bayesian Nonparametric Feature Construction for Inverse Reinforcement Learning

Jaedeug Choi and **Kee-Eung Kim**
Department of Computer Science
Korea Advanced Institute of Science and Technology
Daejeon 305-701, Korea
jdchoi@ai.kaist.ac.kr, kekim@cs.kaist.ac.kr

Abstract

Most of the algorithms for inverse reinforcement learning (IRL) assume that the reward function is a linear function of the pre-defined state and action features. However, it is often difficult to manually specify the set of features that can make the true reward function representable as a linear function. We propose a Bayesian nonparametric approach to identifying useful composite features for learning the reward function. The composite features are assumed to be the logical conjunctions of the pre-defined atomic features so that we can represent the reward function as a linear function of the composite features. We empirically show that our approach is able to learn composite features that capture important aspects of the reward function on synthetic domains, and predict taxi drivers' behaviour with high accuracy on a real GPS trace dataset.

1 Introduction

Inverse reinforcement learning (IRL) aims to recover the expert's underlying reward function from her demonstrations and the environment model [Russell, 1998]. IRL is applied to increasingly various research areas, such as robotics [Argall *et al.*, 2009], computer animation [Lee and Popovi, 2010], preference learning [Erkin *et al.*, 2010; Ziebart *et al.*, 2008b] and cognitive science [Baker *et al.*, 2009].

In the last decade, a number of studies on IRL algorithms have appeared in the literature [Ng and Russell, 2000; Abbeel and Ng, 2004; Ratliff *et al.*, 2006; Syed *et al.*, 2008; Ziebart *et al.*, 2008a]. Most of them, especially when dealing with large state spaces, assume that the reward function is a linear function of some pre-defined features. The problem then reduces to learning the unknown weights of the linear function, but the result is highly dependent on the selection of features. Hence, it is desirable to automatically construct the features that compactly describe the structure of the reward function [Abbeel and Ng, 2004].

FIRL [Levine *et al.*, 2010] is one of the few IRL algorithms that construct the features for the reward function. It is assumed that the domain expert supplies the set of all potentially relevant features as *atomic features*, and the algorithm constructs the logical conjunctions of these features for the reward function. GPIRL [Levine *et al.*, 2011] is another IRL algorithm that represents the reward function as a nonlinear function of the atomic features using the Gaussian process (GP). The reward features are implicitly learned as the hyperparameters of the GP kernel.

In this paper, we propose a Bayesian nonparametric approach to constructing the reward function features in IRL. We extend Bayesian IRL [Ramachandran and Amir, 2007] by defining a prior on the *composite features*, which are defined to be the logical conjunctions of the atomic features. Since the number of composite features is not known a priori, we define the prior using the Indian buffet process (IBP) to infer the features and the number of features. Our approach has a number of advantages: First, it learns an explicit representation of the reward function features in the form of logical formulas, which are readily interpretable. Second, it can robustly learn from a noisy behaviour data since it uses a probabilistic model of the data. Third, it can incorporate the domain knowledge on the reward features and their weights into the prior distribution since it is a Bayesian framework.

2 Preliminaries

We use the following notations throughout the paper: $\boldsymbol{x}$ is a column vector with elements x_i and $\boldsymbol{x}_{-i} = \boldsymbol{x} \backslash x_i$. $\boldsymbol{X}$ is a matrix with elements $X_{i,j}$, $\boldsymbol{X}_{:,j}$ is the j-th column of $\boldsymbol{X}$, $\boldsymbol{X}_{-(i,j)} = \boldsymbol{X} \backslash X_{i,j}$, and $\boldsymbol{X}_{-i,j} = \boldsymbol{X}_{:,j} \backslash X_{i,j}$.

We denote a discrete-state Markov decision process (MDP) [Puterman, 1994] as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \{\boldsymbol{T}^a\}_{a \in \mathcal{A}}, \boldsymbol{r}, \gamma \rangle$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\boldsymbol{T}^a$ is the state transition probability such that $T^a_{s,s'} = P(s'|s,a)$, $\boldsymbol{r}$ is the (state-dependent) reward function, and $\gamma \in [0,1)$ is the discount factor.

A policy is defined as a mapping $\pi : \mathcal{S} \to \mathcal{A}$. The value of the policy π is defined as the expected discounted cumulative rewards of executing π, which satisfies the equation $\boldsymbol{v}^\pi = \boldsymbol{r} + \gamma \boldsymbol{T}^\pi \boldsymbol{v}^\pi$ where $T^\pi_{s,s'} = P(s'|s,\pi(s))$. Similarly, the Q-function is defined by the equation $\boldsymbol{Q}^\pi_{:,a} = \boldsymbol{r} + \gamma \boldsymbol{T}^a \boldsymbol{v}^\pi$. Given an MDP $\mathcal{M}$, an optimal policy π^* maximizes the value function for all the states. The value of an optimal policy should satisfy the Bellman optimality equation: $\boldsymbol{v}^* = \max_{a \in \mathcal{A}} (\boldsymbol{r} + \gamma \boldsymbol{T}^a \boldsymbol{v}^*)$.

The IRL refers to the problem of inferring the reward function of an expert given an MDP$\backslash \boldsymbol{r}$ and the expert's be-

haviour data $\mathcal{D}$. We assume that $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$ is generated by executing an optimal policy with respect to the unknown reward vector $\boldsymbol{r}$, where the n-th trajectory τ_n is an H-step sequence of state-action pairs, *i.e.*, $\tau_n = \{(s_{n,1}, a_{n,1}), \ldots, (s_{n,H}, a_{n,H})\}$.

2.1 BIRL: Bayesian Framework for IRL

Ramachandran and Amir [2007] proposed a Bayesian framework for IRL by modeling the compatibility of the reward with the behaviour data as the likelihood and the preference on the rewards as the prior. The likelihood is defined as an independent exponential, or softmax, distribution:

$$P(\mathcal{D}|\boldsymbol{r}, \eta) = \prod_{n=1}^{N} P(\tau_n|\boldsymbol{r}, \eta) = \prod_{n=1}^{N} \prod_{(s,a)\in\tau_n} \frac{\exp(\eta Q^*_{s,a}(\boldsymbol{r}))}{\sum_{a'\in\mathcal{A}} \exp(\eta Q^*_{s,a'}(\boldsymbol{r}))} \quad (1)$$

where η is the parameter representing the confidence of actions being optimal, and $\boldsymbol{Q}^*(\boldsymbol{r})$ is the optimal Q-function computed using $\boldsymbol{r}$. Assuming that the reward entries are independently distributed, the prior on the reward function is defined as $P(\boldsymbol{r}) = \prod_{s\in\mathcal{S}} P(r_s)$. Various distributions such as the uniform, normal, or Laplace distribution can be used as the prior depending on the domain knowledge on the reward function. The reward function is then inferred from the posterior distribution, using Markov Chain Monte Carlo (MCMC) for estimating posterior mean [Ramachandran and Amir, 2007] or gradient ascent for estimating the maximum-a-posteriori (MAP) [Choi and Kim, 2011].

2.2 Indian Buffet Process

The Indian buffet process (IBP) [Ghahramani *et al.*, 2007] defines a distribution over binary matrices with infinitely many columns. This process can be explained by a culinary metaphor: The customers enter an Indian buffet one after another and choose dishes from an infinite number of dishes. The first customer chooses dishes whose number is distributed by Poisson(α). Thereafter, the i-th customer chooses the dish j with a probability of ζ_j/i where ζ_j is the number of customers who have previously chosen the dish j. In addition, the i-th customer chooses new dishes whose number is distributed by Poisson(α/i). Let $\boldsymbol{Z}$ be an $M \times K$ binary matrix where M and K is the total number of customers and dishes, whose elements $Z_{ij} = 1$ indicates that the i-th customer chooses the dish j. By taking the limit $K \to \infty$, we obtain the probability distribution induced by the IBP:

$$P_{\text{IBP}}(\boldsymbol{Z}|\alpha) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(\zeta_k + \frac{\alpha}{K})\Gamma(M - \zeta_k + 1)}{\Gamma(M + 1 + \frac{\alpha}{K})} \quad (2)$$

where $\zeta_k = \sum_{m=1}^{M} Z_{mk}$. The parameter α controls the sparsity of the matrix.

3 BNP-FIRL: Bayesian Nonparametric Feature Construction for IRL

We assume that the reward vector is linearly parameterized so that $\boldsymbol{r} = \boldsymbol{\Phi}\boldsymbol{w}$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K]$ is a reward feature matrix whose k-th column $\boldsymbol{\phi}_k$ is an $|\mathcal{S}|$-dimensional binary vector representing the k-th reward feature, and $\boldsymbol{w}$ is an K-dimensional weight vector.

As we have mentioned in the introduction, most of the IRL algorithms assume that the reward features are already given, and just compute the weights [Ng and Russell, 2000; Abbeel and Ng, 2004; Ratliff *et al.*, 2006; Syed *et al.*, 2008; Ziebart *et al.*, 2008a]. In contrast, the goal of our work is to learn the reward features as well as to compute the weights. A small assumption made here is that a domain expert provides a set of (binary) atomic features, which includes all the relevant indicators that may possibly influence determining the rewards in some unknown way. Since the true reward function is hardly a linear function of the atomic features, we should learn the reward features as (non-linear but hopefully not too complex) functions of atomic features, so that we can still represent the reward function as a linear function of reward features. We refer to the reward features as *composite features* in order to distinguish them from atomic features.

In the following sections, we describe our Bayesian nonparametric approach to the feature construction in IRL, where the composite features are constructed by taking the logical conjunctions of the atomic features. We first define a nonparametric prior over the logical conjunctions of the atomic features using the IBP. We then derive the posterior over the composite features and the reward weights by extending BIRL. Finally, we present an MCMC algorithm for the posterior inference.

3.1 Prior for Feature Construction

Given a matrix of atomic features, defined as $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_M]$ where the m-th column $\boldsymbol{\psi}_m$ is an $|\mathcal{S}|$-dimensional binary vector representing the m-th atomic feature, we define the constructed composite feature matrix as $\boldsymbol{\Phi} = f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi})$ with:

$$\boldsymbol{\phi}_k = \bigwedge_{\substack{m\in\{1,\ldots,M\} \text{ s.t.} \\ x_m=1 \wedge Z_{m,k}=1}} \tilde{\boldsymbol{\psi}}^{(k)}_m$$

where $\boldsymbol{x}$ is an M-dimensional binary vector indicating the participation of the m-th atomic feature in any of the composite features, $\boldsymbol{Z}$ is an $M \times K$ binary matrix indicating the participation of the m-th atomic feature in the k-th composite feature, and $\boldsymbol{U}$ is also an $M \times K$ binary matrix indicating whether the m-th atomic feature is negated in the formula for the k-th composite feature so that $\tilde{\boldsymbol{\psi}}^{(k)}_m = \boldsymbol{\psi}_m$ if $U_{m,k} = 1$ and $\tilde{\boldsymbol{\psi}}^{(k)}_m = \neg\boldsymbol{\psi}_m$ otherwise.

We define the prior on $\boldsymbol{Z}$, $\boldsymbol{U}$, and $\boldsymbol{x}$ as follows: first, we start with $\boldsymbol{Z}$ that indicates which atomic features are used for each composite feature. Each composite feature generally consists of more than one exchangeable atomic features. In addition, we do not know a priori the number K of columns in $\boldsymbol{Z}$, which is the number of the composite features to be constructed. Thus, analogous to the latent feature modeling, we use the IBP as the prior on $\boldsymbol{Z}$:

$$P(\boldsymbol{Z}|\alpha) = P_{\text{IBP}}(\boldsymbol{Z}|\alpha). \quad (3)$$

Note that, by using the IBP, we inherently assume that K is unbounded even though it is at most 3^M. The rationale behind our assumption is that $\boldsymbol{Z}$ can have duplicated columns as well as infinitely many zero columns. Second, we use the Bernoulli distribution for every entry of the binary matrix $\boldsymbol{U}$,

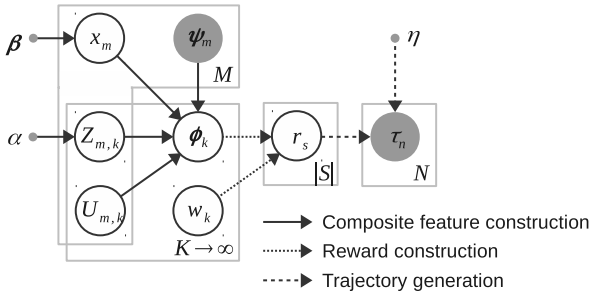

Figure 1: Graphical model of BNP-FIRL

so that the atomic features are negated with $p = 0.5$:

$$P(\boldsymbol{U}) = \prod_{m,k} P(U_{m,k}) = \prod_{m,k} P_{\text{Ber}}(U_{m,k}; 0.5). \quad (4)$$

Finally, we define the prior on the binary vector $\boldsymbol{x}$. Since $x_m = 0$ implies that the atomic feature $\boldsymbol{\psi}_m$ is not used at all for any composite feature, we use the prior that favors $\boldsymbol{x}$ being sparse. Specifically, we use the Bernoulli distribution with $p = \kappa$ where κ is Beta distributed with $\boldsymbol{\beta} = [\beta_1, \beta_2]$:

$$\begin{aligned} P(\kappa|\boldsymbol{\beta}) &= P_{\text{Beta}}(\kappa; \boldsymbol{\beta} = [\beta_1, \beta_2]) \\ P(x_m|\kappa) &= P_{\text{Ber}}(x_m; \kappa). \end{aligned} \quad (5)$$

This is analogous to controlling the row-wise sparsity in the IBP proposed by Rai and Daumé III [2008].

By combining Eqns (3), (4), and (5), the prior is defined as

$$P(\boldsymbol{\Phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\Psi}) = P(\boldsymbol{x}|\boldsymbol{\beta}) P(\boldsymbol{Z}|\alpha) P(\boldsymbol{U}) \quad (6)$$

where $P(\boldsymbol{x}|\boldsymbol{\beta}) = \int \prod_m P(x_m|\kappa) P(\kappa|\boldsymbol{\beta}) \mathrm{d}\kappa$.

3.2 Posterior Inference

BNP-FIRL extends BIRL by using the prior defined in the above and treating the behaviour $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$ as being drawn from the generative process as follows:

$$\begin{aligned} \kappa|\boldsymbol{\beta} &\sim \text{Beta}(\boldsymbol{\beta} = [\beta_1, \beta_2]) \\ x_m|\kappa &\sim \text{Bernoulli}(\kappa) \\ \boldsymbol{Z}|\alpha &\sim \text{IBP}(\alpha) \\ U_{m,k} &\sim \text{Bernoulli}(0.5) \\ w_k &\sim P(w_k) \\ \boldsymbol{\Phi} &:= f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}) \\ \boldsymbol{r} &:= \boldsymbol{\Phi}\boldsymbol{w} \\ \tau_n|\boldsymbol{r}, \eta &\sim \prod_{(s,a)\in\tau_n} \frac{\exp(\eta Q^*_{s,a}(\boldsymbol{r}))}{\sum_{a'\in\mathcal{A}} \exp(\eta Q^*_{s,a'}(\boldsymbol{r}))}. \end{aligned}$$

Fig 1 shows the graphical model of BNP-FIRL. Note that, as in BIRL, the reward weights w_k's are assumed to be independently distributed so that $P(\boldsymbol{w}) = \prod_{k=1}^{K} P(w_k)$.

The posterior over the composite features for the reward function and the associated weights is then formulated as

$$P(\boldsymbol{\Phi}, \boldsymbol{w}|\mathcal{D}, \Theta) \propto P(\mathcal{D}|\boldsymbol{\Phi}, \boldsymbol{w}, \eta) P(\boldsymbol{w}) P(\boldsymbol{\Phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\Psi}) \quad (7)$$

where $\Theta = \{\eta, \alpha, \boldsymbol{\beta}, \boldsymbol{\Psi}\}$, $P(\mathcal{D}|\boldsymbol{\Phi}, \boldsymbol{w}, \eta)$ is the BIRL likelihood defined in Eqn (1) with $\boldsymbol{r} = \boldsymbol{\Phi}\boldsymbol{w}$, $P(\boldsymbol{w})$ is the prior on the weights, and $P(\boldsymbol{\Phi}|\alpha, \boldsymbol{\beta}, \boldsymbol{\Psi})$ is the prior on the composite features defined in Eqn (6). Note that α controls the number of composite features to be constructed and $\boldsymbol{\beta}$ controls the total number of the unique atomic features used in the construction. In other words, α and $\boldsymbol{\beta}$ are the parameters that control the column-wise and the row-wise sparsity of the composite feature matrix $\boldsymbol{\Phi}$, respectively.

Algorithm 1 MCMC algorithm for BNP-FIRL

```
Initialize x, Z, U, w.
for t = 1 to T do
  for m = 1 to M do
    Sample x_m according to Eqn (8).
  end for
  for m = 1 to M do
    for k = 1 to K do
      Sample U_{m,k} according to Eqn (9).
    end for
  end for
  for m = 1 to M do
    for k = 1 to K do
      if Σ_{i≠m} Z_{i,k} > 0 then
        Sample Z_{m,k} according to Eqn (10).
      end if
    end for
    Propose ξ = ⟨K+, Z+, U+, w+⟩: K+ ~ Poisson(α/M).
    Accept ξ with probability min{1, ρ_Z}.
  end for
  for k = 1 to K do
    Propose w' ~ N(w_k, λ).
    Accept w' with probability min{1, ρ_w}.
  end for
end for
```

We infer the composite features and the reward weights from the posterior using an MCMC algorithm described as follows (Algorithm 1): we first update x_m by sampling from the probability distribution conditioned on the rest of the random variables,

$$\begin{aligned} &P(x_m = 1|\mathcal{D}, \boldsymbol{x}_{-m}, \boldsymbol{Z}, \boldsymbol{U}, \boldsymbol{w}, \Theta) \\ &\quad \propto P(\mathcal{D}|f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta)(\beta_1 + \textstyle\sum_{i\neq m} x_i) \\ &P(x_m = 0|\mathcal{D}, \boldsymbol{x}_{-m}, \boldsymbol{Z}, \boldsymbol{U}, \boldsymbol{w}, \Theta) \\ &\quad \propto P(\mathcal{D}|f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta)(\beta_2 + M - \textstyle\sum_{i\neq m} x_i). \end{aligned} \quad (8)$$

Note in the above that, instead of drawing the Bernoulli distribution parameter κ from Beta($\boldsymbol{\beta} = [\beta_1, \beta_2]$) and then drawing x_m, we have collapsed κ for an efficient inference.

Next, in order to sample $U_{m,k}$ which indicates whether we should negate the atomic feature $\boldsymbol{\psi}_m$ in the composite feature $\boldsymbol{\phi}_k$, we sample it from the likelihood

$$\begin{aligned} &P(U_{m,k}|\mathcal{D}, \boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}_{-(m,k)}, \boldsymbol{w}, \Theta) \\ &\qquad \propto P(\mathcal{D}|f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta) \end{aligned} \quad (9)$$

since the prior on $U_{m,k}$ is uniformly random in $\{0, 1\}$.

We then sample $Z_{m,k}$, which indicates whether the atomic feature $\boldsymbol{\psi}_m$ appears in the composite feature $\boldsymbol{\phi}_k$. For the sake of exposition, we rephrase the IBP culinary metaphor into our context, the atomic features being the customers and the composite features being the dishes. The first atomic feature chooses Poisson(α) composite features. The m-th atomic feature chooses the k-th composite feature (already chosen by preceding atomic features) with probability $\sum_{i=1}^{m-1} Z_{i,k}/m$ and additionally chooses Poisson(α)$/m$ new composite features, where $Z_{i,k} = 1$ indicates that the i-th atomic feature $\boldsymbol{\psi}_i$ have chosen the k-th composite feature $\boldsymbol{\phi}_k$.

This metaphor and the exchangeability of the atomic features leads to the update method for $\boldsymbol{Z}_{m,:}$ [Rai and Daumé III, 2008] as follows: first, for the composite features that are already chosen (*i.e.*, $\sum_{i \neq m} Z_{i,k} > 0$), we update $Z_{m,k}$ according to the conditional distribution

$$\begin{aligned} &P(Z_{m,k} = 1 | \mathcal{D}, \boldsymbol{x}, \boldsymbol{Z}_{-m,k}, \boldsymbol{U}, \boldsymbol{w}, \Theta) \\ &\quad \propto P(\mathcal{D} | f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta) \textstyle\sum_{i \neq m} Z_{i,k} \\ &P(Z_{m,k} = 0 | \mathcal{D}, \boldsymbol{x}, \boldsymbol{Z}_{-m,k}, \boldsymbol{U}, \boldsymbol{w}, \Theta) \\ &\quad \propto P(\mathcal{D} | f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta)(M - \textstyle\sum_{i \neq m} Z_{i,k}). \end{aligned} \quad (10)$$

Second, for choosing the new composite features, we sample $Z_{m,k}$ using the Metropolis-Hastings (MH) update: we first sample $\xi = \langle K^+, \boldsymbol{Z}^+, \boldsymbol{U}^+, \boldsymbol{w}^+ \rangle$ where $K^+ \sim \text{Poisson}(\alpha/m)$, $\boldsymbol{Z}^+$ is the $m \times K^+$ binary matrix whose entries in the m-th row are set to 1 and others are set to 0, and the $m \times K^+$ binary matrix $\boldsymbol{U}^+$ and K^+-dimensional vector $\boldsymbol{w}^+$ are drawn from the corresponding priors. We then accept ξ with probability $\min\{1, \rho_Z\}$ where

$$\rho_Z = \frac{P(\mathcal{D} | f(\boldsymbol{x}, [\boldsymbol{Z}, \boldsymbol{Z}^+], [\boldsymbol{U}, \boldsymbol{U}^+]; \boldsymbol{\Psi}), [\boldsymbol{w}; \boldsymbol{w}^+], \eta)}{P(\mathcal{D} | f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta)}. \quad (11)$$

In the above equation, $[\boldsymbol{X}, \boldsymbol{Y}]$ denotes horizontal concatenation, and $[\boldsymbol{x}; \boldsymbol{y}]$ denotes vertical concatenation.

Finally, we sample the weights w_k, again using the MH update. We first sample $w' \sim \mathcal{N}(w_k, \lambda)$ and then accept it with probability $\min\{1, \rho_w\}$ where

$$\rho_w = \frac{P(\mathcal{D} | f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}^{new}, \eta) P(w')}{P(\mathcal{D} | f(\boldsymbol{x}, \boldsymbol{Z}, \boldsymbol{U}; \boldsymbol{\Psi}), \boldsymbol{w}, \eta) P(w_k)}. \quad (12)$$

with $\boldsymbol{w}^{new}$ formed by taking $w_k^{new} = w'$ and $\boldsymbol{w}_{-k}^{new} = \boldsymbol{w}_{-k}$.

The posterior mean is commonly used for inferring the reward function since it is known to minimize the square error $L_{\text{SE}}(\boldsymbol{r}, \hat{\boldsymbol{r}}) = ||\boldsymbol{r} - \hat{\boldsymbol{r}}||_2$ [Ramachandran and Amir, 2007]. We thus estimate the posterior mean of the reward function. On the other hand, when we show the learned composite features, we choose the sample with the maximum posterior, *i.e.*, $\langle \hat{\boldsymbol{\Phi}}_{\text{MAP}}, \hat{\boldsymbol{w}}_{\text{MAP}} \rangle = \text{argmax}_{\langle \boldsymbol{\Phi}^{(t)}, \boldsymbol{w}^{(t)} \rangle} P(\boldsymbol{\Phi}^{(t)}, \boldsymbol{w}^{(t)} | \mathcal{D}, \Theta)$ and $\hat{\boldsymbol{r}}_{\text{MAP}} = \hat{\boldsymbol{\Phi}}_{\text{MAP}} \hat{\boldsymbol{w}}_{\text{MAP}}$. This is because the sample mean is ill-defined for $\hat{\boldsymbol{\Phi}}^{(t)}$'s with different dimensions.

4 Experimental Results

In this section, we show the performance of BNP-FIRL in a number of problem domains and compare it to FIRL [Levine *et al.*, 2010] and GPIRL [Levine *et al.*, 2011].[1] The performance was evaluated using the expected value difference (EVD) $\frac{1}{|\mathcal{S}|} ||\boldsymbol{v}^*(\boldsymbol{r}) - \boldsymbol{v}^{\pi(\boldsymbol{r}')}(\boldsymbol{r})||_1$ where $\boldsymbol{r}$ is the expert's reward function (*i.e.*, the ground truth), $\boldsymbol{r}'$ is the learned reward function, and $\pi(\boldsymbol{r}')$ is the optimal policy computed using $\boldsymbol{r}'$. The EVD thus can be seen as a measurement of the loss in the optimality incurred by using the policy from the learned reward function instead of the expert's reward function. In order to evaluate how well the composite features are learned, we computed the EVD on the same problem instance (*i.e.*, the training problem instance) where the expert's behaviour data was generated, and on additional random problem instances (*i.e.*, transfer problem instances) as well.

[1] code available at http://graphics.stanford.edu/projects/gpirl

Table 1: Learned features for 32×32 objectworld domain.

	Weights	Reward features
$\boldsymbol{\phi}_1$	8.61	$d(s, c_1) < 3 \wedge d(s, c_2) < 2$
$\boldsymbol{\phi}_2$	-11.53	$d(s, c_1) < 3 \wedge \neg(d(s, c_2) < 2)$
$\boldsymbol{\phi}_3$	-7.29	$d(s, c_1) < 3 \wedge \neg(d(s, c_2) < 2) \wedge d(s, c_3) < 9$
$\boldsymbol{\phi}_4$	0.51	$\neg(d(s, c_3) < 9)$

4.1 Objectworld Domain

The first set of experiments was performed on the objectworld domain [Levine *et al.*, 2011], where the agent can move north, south, east, west, or stay in the current location in an $N \times N$ grid. Each action has a failure probability of 0.3 which makes the agent move in a random direction. There are a number of colored objects randomly placed in the grid, each of them with one of the $C \geq 2$ colors.

We prepared a total of CN atomic features of the form "$d(s, c_i) < j$" indicating that the distance between the agent's location s and the nearest color i object is less than j, where $i \in \{1, \ldots, C\}$ and $j \in \{1, \ldots, N\}$. The true reward function was set to

$$r_s = \begin{cases} 1 & \text{if } d(s, c_1) < 3 \wedge d(s, c_2) < 2, \\ -2 & \text{if } d(s, c_1) < 3 \wedge \neg(d(s, c_2) < 2), \\ 0 & \text{otherwise.} \end{cases}$$

We generated 10 random training problem instances by sampling the locations of objects and their colors, and gathered trajectories of length 20. In order to measure how well the learned reward function generalizes to novel yet similar problem instances, we measured EVD on additional 10 transfer problem instances for each training problem instance, generated in the same way we prepared training problem instances.

The top row in Fig 2 shows the EVD performances when the trajectories are generated by the optimal policy. BNP-FIRL outperformed FIRL in the experiments. This is mainly due to the fact that BNP-FIRL is allowed to have multiple composite features for each state, whereas FIRL can only have one because of the way the algorithm constructs the composite features. Tbl 1 shows the 4 reward features and the corresponding weights found by BNP-FIRL. In comparison, FIRL produced a total of 21 features. On the other hand, BNP-FIRL was on par in performance with GPIRL. Nonetheless, one of the advantages of using BNP-FIRL is that the learned features are explicitly represented and thus readily interpretable.

The bottom row in Fig 2 shows the EVD performances when the trajectories are generated from an ϵ-greedy policy. It is interesting to observe that BNP-FIRL outperforms GPIRL and FIRL. We conjecture that this is due to the likelihood in BNP-FIRL being particularly robust to noisy behaviour data.

4.2 Simulated-highway Domain

The second set of experiments was done on the simulated-highway domain [Levine *et al.*, 2011], where the agent drives a vehicle by moving one lane left or right at speeds $\{1, 2, 3, 4\}$

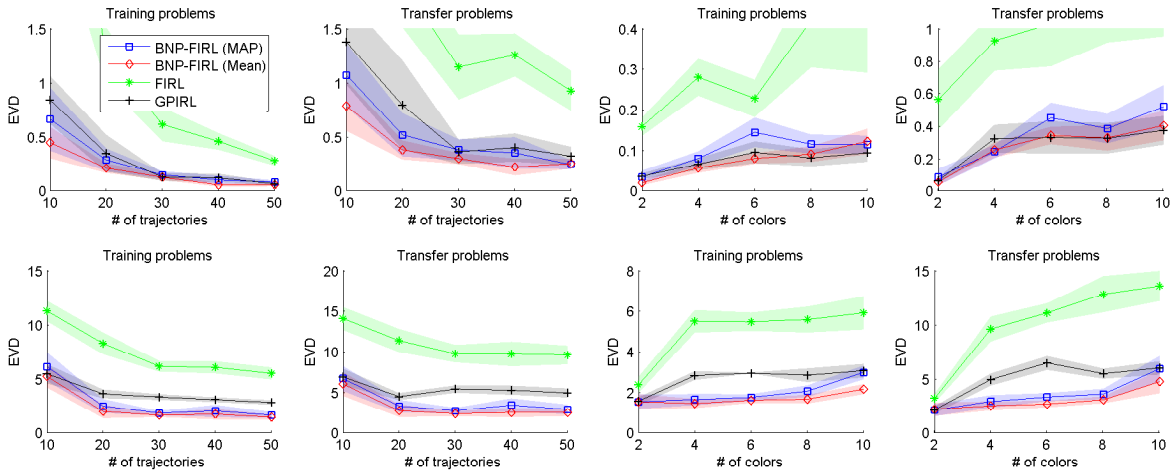

Figure 2: Averages and standard errors of the EVD on 10 random instances of the 32×32 objectworld domain, with *Top row*: trajectories generated by the optimal policy, and *Bottom row*: trajectories generated by ϵ-greedy policy ($\epsilon = 0.2$). *Left two columns*: fix $C = 4$ (the number of colors) and vary $|\mathcal{D}|$ (the number of trajectories). *Right two columns*: vary C and fix $|\mathcal{D}| = 50$.

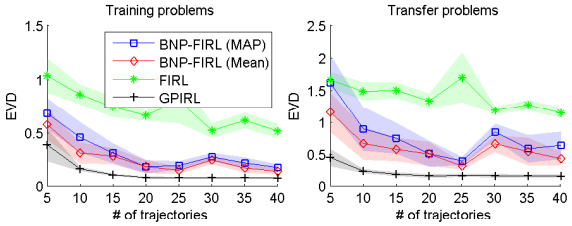

Figure 3: Averages and standard errors of the performance results over 10 instances of the highway domain.

on a three-lane highway. The actions that change the lane or speed fail with probability 0.3. There are other vehicles driving at speed 1, each of them being either a civilian or a police vehicle and either a car or a motorcycle (a total combination of 4 categories). We prepared an expert that prefers to drive as fast as possible but avoids driving at speeds 3 or 4 within a distance of 2 from a police vehicle. We also prepared 3 types of atomic features: The first indicate the current speed (4 features) and the second indicate the current lane (3 features). The third indicate the distance to the nearest car from each category, of the form "$d(s, c) \leq j$" where $c \in \{\text{police, civilian}\} \times \{\text{car, motorcycle}\}$ and $j \in \{0, \ldots, 5\}$ (24 features). We generated the trajectory data by executing the optimal policy for 200 time steps.

Fig 3 shows that BNP-FIRL again performs better than FIRL. On the other hand, BNP-FIRL performs slightly worse than GPIRL although, as shown in Fig 4, the learned reward functions from the two algorithms were very similar to the true one. On the other hand, FIRL was not able to capture good features for the reward function and as a result, the learned reward function is very different from the true one.

4.3 Taxi Driver Behaviour Prediction

The final set of experiments was done on learning the taxi drivers' preference using the GPS trace data collected in San Francisco [Piorkowski *et al.*, 2009]. We modeled the road network as a graph, which consists of 20531 road segments obtained from the OPENSTREETMAP.[2] As in Ziebart *et al.* [2008b], we assumed that the taxi drivers try to reach a destination in a trip by taking the road segments according to their preferences. Note that each trip typically has a different destination. We used the goal-oriented MDP $\mathcal{M}_g$ to define the environment model, where the states are road segments, and the state transitions correspond to taking one of the road segments at intersections. The rewards were assumed to be negative everywhere except at the destination, which was represented as an absorbing goal state g with a zero reward.

The composite features $\phi_{(s,a),k}$ and the weights w_k were assumed to be independent of the destination. In other words, given the destination g, the reward was calculated as $\boldsymbol{r}_g = \boldsymbol{F}(g) \cdot \boldsymbol{w}$ where $w_k < 0$, $F_{(g,a),k}(g) = 0$ and $F_{(s,a),k}(g) = \phi_{(s,a),k}$ for all $s \in \mathcal{S} \backslash \{g\}$. This change leads to a slight modification to the likelihood so that $P(\mathcal{D}|\boldsymbol{\Phi}, \boldsymbol{w}, \eta) = \prod_n P(\tau_n|\boldsymbol{r}_{g_n}, \eta; \mathcal{M}_{g_n})$ where g_n is the destination in the trajectory τ_n. Note that, although we have different MDPs $\mathcal{M}_{g_n}$ for each trajectory due to different destination, they all share the same reward features $\boldsymbol{\Phi}$ and weights $\boldsymbol{w}$.

We prepared state-dependent atomic features representing the properties of the road segments, such as the type, the speed limit, the number of lanes, and one-way.[3] We also prepared action-dependent atomic features representing the angle of the turn.[4]

Among the traces of 500 taxis in the original data, we selected 500 trips from 10 taxis, visualized in Fig 5. The

[2] http://www.openstreetmap.org

[3] type $\in$ {highway, primary street, secondary street, living street}, speed limit $\in$ {below 20 mph, 20-30 mph, 30-40 mph, above 40 mph}, # lanes $\in$ {1, 2, 3+}. These features were obtained using the OPENSTREETMAP

[4] turn angle $\in$ {hard left, soft left, straight, soft right, hard right, u-turn}.

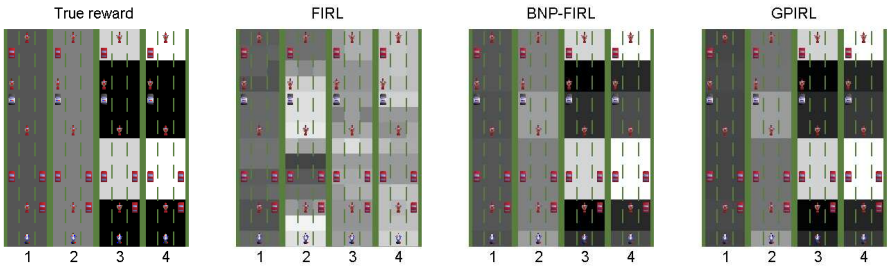

Figure 4: Rewards for the states at each speed for the highway domain. The numbers below the figures present the speed and the brightness of each state represents the reward (brighter=higher).

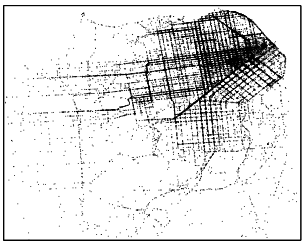

Figure 5: GPS traces of 500 trips from 10 taxis collected in San Francisco.

accumulated distance of the trips was 1137 miles. We then segmented the GPS traces and mapped them to the road segments using a preprocessing algorithm described in Lou *et al.* [2009].

We compared BNP-FIRL to two baseline methods and two IRL algorithms in the literature. As for the first baseline, we used the shortest path to the destination. As for the second baseline, we used the optimal policy from an MDP with heuristically set reward weight $\boldsymbol{w}'$. A simple heuristic would be counting the feature visitations in the trace data, and set the reward weights accordingly. Since we enforce the reward weights to be negative, we used the normalized counts of feature visitations offset by the maximum count value. As for the two IRL algorithms, we chose the maximum entropy IRL (MaxEntIRL) [Ziebart *et al.*, 2008a] and the gradient ascent BIRL (MAP-BIRL) [Choi and Kim, 2011]. Since both algorithms depend on a pre-defined set of features, we supplied the set of all atomic features. We do not report the results from FIRL and GPIRL since it was not straightforward to modify them to appropriately handle multiple MDPs sharing the same reward features and weights.

Tbl 2 shows the average prediction accuracy and their standard errors. The prediction accuracy was measured in terms of the turn and the route predictions. For each trip, we computed the path from the origin to the destination using the algorithms. The correct actions taken at the intersections were counted in the turn prediction accuracy, and the ratio of the total distance taking the correct road segments was calculated for the route prediction accuracy. All the results were obtained by the 5-fold cross validation. Tbl 3 presents the weights and composite features learned by BNP-FIRL, which are fairly intuitive. For example, making hard left turns (ϕ_1) is avoided with the highest penalty. Making turns on a highway is also highly undesirable except making a soft right turn to take an exit ramp (ϕ_2). It was also found that taxi drivers prefer to take the primary street (ϕ_9) or the road with 20-30 mph limit (ϕ_8), rather than the highway (ϕ_5) or the road with speed limit higher than 40 mph (ϕ_3). This was because the trips were generally short, the average distance being 2.27 miles.

Table 2: Driver behaviour prediction results.

	Turn prediction (%)	Route prediction (%)
Shortest path	77.06 ($\pm$0.14)	32.91 ($\pm$0.21)
MDP ($\boldsymbol{w}'$)	83.79 ($\pm$0.21)	43.42 ($\pm$0.23)
MaxEntIRL	80.27 ($\pm$0.67)	42.80 ($\pm$1.07)
MAP-BIRL	84.97 ($\pm$0.58)	46.87 ($\pm$0.92)
BNP-FIRL (MAP)	**86.22 ($\pm$0.24)**	**48.42 ($\pm$0.54)**
BNP-FIRL (Mean)	**86.28 ($\pm$0.18)**	**48.70 ($\pm$0.70)**

Table 3: Learned features for driver behaviour prediction.

	Weights	Reward features
ϕ_1	-2.40	hard left turn
ϕ_2	-1.98	highway $\wedge\neg$(secondary street) $\wedge\neg$(below 20 mph) $\wedge\neg$(soft right turn) $\wedge\neg$(straight)
ϕ_3	-1.38	above 40 mph
ϕ_4	-0.41	$\neg$(highway) $\wedge\neg$(oneway)
ϕ_5	-0.27	highway
ϕ_6	-0.24	$\neg$(straight)
ϕ_7	-0.21	$\neg$(below 20 mph) $\wedge\neg$(2 lanes) $\wedge\neg$(oneway)
ϕ_8	-0.12	20-30 mph
ϕ_9	-0.11	primary street

5 Conclusion

We presented BNP-FIRL, a Bayesian nonparametric approach to constructing reward features in IRL. We defined the reward features as the logical conjunctions of the atomic features provided by the domain expert, and formulated a nonparametric prior over the reward features using the IBP. We derived an MCMC algorithm to carry out the posterior inference on the reward features and the corresponding reward weights.

We showed that BNP-FIRL outperforms or performs on par with prior feature-learning IRL algorithms through experiments on synthetic domains. In comparison to FIRL, BNP-FIRL produces more succinct sets of features with richer representations and learns better reward functions. In comparison to GPIRL, BNP-FIRL produces the set of features that are explicitly represented and readily interpretable. We also presented reward feature learning results on a real GPS trace data collected from taxi drivers, predicting their behaviour with higher accuracy than prior IRL algorithms.

Acknowledgments

This work was supported by National Research Foundation of Korea (Grant# 2012-007881), the Defense Acquisition Program Administration and Agency for Defense Development of Korea (Contract# UD080042AD), and the IT R&D program of MKE/KEIT (Contract# 10041678).

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[Argall *et al.*, 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[Baker *et al.*, 2009] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[Choi and Kim, 2011] Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems 24*, 2011.

[Erkin *et al.*, 2010] Zeynep Erkin, Matthew D. Bailey, Lisa M. Maillart, Andrew J. Schaefer, and Mark S. Roberts. Eliciting patients' revealed preferences: An inverse markov decision process approach. *Decision Analysis*, 7(4):358–365, 2010.

[Ghahramani *et al.*, 2007] Zoubin Ghahramani, Thomas L. Griffiths, and Peter Sollich. Bayesian nonparametric latent feature models. *Bayesian Statistics*, 8:1–25, 2007.

[Lee and Popovi, 2010] Seong Jae Lee and Zoran Popovi. Learning behavior styles with inverse reinforcement learning. *ACM Transaction on Graphics*, 29:1–7, 2010.

[Levine *et al.*, 2010] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems 23*, 2010.

[Levine *et al.*, 2011] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems 24*, 2011.

[Lou *et al.*, 2009] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.

[Ng and Russell, 2000] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[Piorkowski *et al.*, 2009] Michal Piorkowski, Natasa Sarafijanovoc-Djukic, and Matthias Grossglauser. A parsimonious model of mobile partitioned networks with clustering. In *Proceedings of the 1st International Conference on Communication Systems and Networks and Workshops*, 2009.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Rai and Daumé III, 2008] Piyush Rai and Hal Daumé III. The infinite hierarchical factor regression model. In *Advances in Neural Information Processing Systems 21*, 2008.

[Ramachandran and Amir, 2007] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[Ratliff *et al.*, 2006] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[Russell, 1998] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, 1998.

[Syed *et al.*, 2008] Umar Syed, Michael Bowling, and Robert E. Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[Ziebart *et al.*, 2008a] Brian D. Ziebart, Andrew Maas, James Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

[Ziebart *et al.*, 2008b] Brian D. Ziebart, Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell. Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, 2008.