University of Glasgow | School of Computing Science

Variational Inference for Inverse Reinforcement Learning with Gaussian Processes

Paulius Dilkas

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

Date of submission placed here

Contents

1 Introduction

Inverse reinforcement learning (IRL)—a problem proposed by Russell in 1998 [40]—asks us to find a reward function for a Markov decision process that best explains a set of given demonstrations. IRL is important because reward functions can be hard to define manually [1, 3], and rewards are not entirely specific to a given environment, allowing one to reuse the same reward structure in previously unseen environments [3, 18, 23]. Moreover, IRL has seen a wide array of applications in autonomous vehicle control [19, 20] and learning to predict another agent's behaviour [7, 49, 56, 57, 58]. Most approaches in the literature (see Section 3) make a convenient yet unjustified assumption that the reward function can be expressed as a linear combination of features. One proven way to abandon this assumption is by representing the reward function as a Gaussian process (GP) [18, 23, 30]. Our main goal is employ variational inference (VI) for recovering the reward function, which can prove useful in two ways:

1. As VI tends to be faster than Markov chain Monte Carlo sampling [6], we should be able to handle more data.
2. Modelling full posterior distributions for various parameters can result in more precise reward predictions, as the model simply holds more information.

2 Statement of the Problem

Definition 1. A *Markov decision process* (MDP) is a set $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, r\}$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of states and actions, respectively; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a function defined so that $\mathcal{T}(s, a, s')$ is the probability of moving to state s' after taking action a in state s; $\gamma \in [0, 1)$ is the discount factor (with higher γ values, it makes little difference whether a reward is received now or later, while with lower γ values the future becomes gradually less and less important); and $r : \mathcal{S} \to \mathbb{R}$ is the reward function.

In *inverse reinforcement learning*, one is presented with an MDP without a reward function $\mathcal{M} \setminus \{r\}$ and a set of expert demonstrations $\mathcal{D} = \{\zeta_i\}_{i=1}^N$, where each demonstration $\zeta_i = \{(s_{i,0}, a_{i,0}), \dots, (s_{i,T}, a_{i,T})\}$ is a multiset of state-action pairs representing the actions taken by the expert during a particular recorded session. Each state is also characterised by a number of features. The goal of IRL is then to find r such that the optimal policy under r

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| \pi\right]$$

matches the actions in $\mathcal{D}$.

Following previous work on GP IRL [23, 18], we use a maximum entropy IRL model [56], under which we have that

$$P(a|s) \propto \exp(Q_r(s, a)),$$

where

$$Q_r(s,a) = r(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')V_r(s'), \tag{1}$$

and $V_r(s)$ is a 'soft' version of the Bellman backup operator, which can be obtained by repeatedly applying the following equation until convergence: [23, 24]

$$V_r(s) = \log \sum_{a \in \mathcal{A}} \exp \left(r(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')V_r(s') \right).$$

The likelihood of the data can then be written down as [18, 23]

$$p(\mathcal{D}|r) = \prod_{i=1}^{N} \prod_{t=1}^{T} p(a_{i,t}|s_{i,t}) = \exp \left(\sum_{i=1}^{N} \sum_{t=1}^{T} Q_r(s_{i,t}, a_{i,t}) - V_r(s_{i,t}) \right). \tag{2}$$

However, a reward function learned by maximising this likelihood is not transferable to new situations [18, 23]. One needs to model the reward structure in a way that would allow reward predictions for previously unseen states.

One way to model rewards without assumptions of linearity is with a *Gaussian process* (GP). A GP is a collection of random variables, any finite combination of which has a joint Gaussian distribution [35]. We write $r \sim \mathcal{GP}(0, k_{\boldsymbol{\lambda}})$ to say that r is a GP with mean 0 and covariance function $k_{\boldsymbol{\lambda}}$, which uses a vector of hyperparameters $\boldsymbol{\lambda}$. Covariance functions take two state feature vectors as input and quantify how similar the two states are, in a sense that we would expect them to have similar rewards.

As training a GP with n data points has a time complexity of $\mathcal{O}(n^3)$ [35], numerous approximation methods have been suggested, many of which select a subset of data called *inducing points* and focus most of the training effort on them [25]. Let $\mathbf{X_u}$ be the matrix of features at inducing states, $\mathbf{u}$ the rewards at those states, and $\mathbf{r}$ a vector with $r(\mathcal{S})$ as elements. Then the full joint probability distribution can be factorised as

$$p(\mathcal{D}, \boldsymbol{\lambda}, \mathbf{X_u}, \mathbf{u}, \mathbf{r}) = p(\mathbf{X_u}) \times p(\boldsymbol{\lambda}|\mathbf{X_u}) \times p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X_u}) \times p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X_u}, \mathbf{u}) \times p(\mathcal{D}|r). \tag{3}$$

Here $p(\mathbf{X_u})$ and $p(\boldsymbol{\lambda}|\mathbf{X_u})$ are customisable priors,

$$\begin{aligned} p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X_u}) &= \mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{K_{u,u}}) \\ &= \frac{1}{(2\pi)^{m/2}|\mathbf{K_{u,u}}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{u}^\intercal \mathbf{K}_{\mathbf{u,u}}^{-1}\mathbf{u}\right) \\ &= \exp\left(-\frac{1}{2}\mathbf{u}^\intercal \mathbf{K}_{\mathbf{u,u}}^{-1}\mathbf{u} - \frac{1}{2}\log|\mathbf{K_{u,u}}| - \frac{m}{2}\log 2\pi\right) \end{aligned} \tag{4}$$

is the GP prior [35], where $m \in \mathbb{N}$ is the number of inducing points. The GP posterior is a multivariate Gaussian [23]

$$p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X_u}, \mathbf{u}) = \mathcal{N}(\mathbf{r}; \mathbf{K}_{\mathbf{r,u}}^\intercal \mathbf{K}_{\mathbf{u,u}}^{-1}\mathbf{u}, \mathbf{K_{r,r}} - \mathbf{K}_{\mathbf{r,u}}^\intercal \mathbf{K}_{\mathbf{u,u}}^{-1}\mathbf{K_{r,u}}), \tag{5}$$

and $p(\mathcal{D}|r)$ is as in (2). The matrices such as $\mathbf{K}_{\mathbf{r},\mathbf{u}}$ are called *covariance matrices* and are defined as $[\mathbf{K}_{\mathbf{r},\mathbf{u}}]_{i,j} = k_{\boldsymbol{\lambda}}(\mathbf{x}_{\mathbf{r},i}, \mathbf{x}_{\mathbf{u},j})$, where $\mathbf{x}_{\mathbf{r},i}$ and $\mathbf{x}_{\mathbf{u},j}$ denote feature vectors for the ith state in $\mathcal{S}$ and the jth state in $\mathbf{X}_{\mathbf{u}}$, respectively [18].

Given this model, data $\mathcal{D}$, and inducing feature matrix $\mathbf{X}_{\mathbf{u}}$, our goal is then to find optimal values of hyperparameters $\boldsymbol{\lambda}$, inducing rewards $\mathbf{u}$, and the rewards for all relevant states $\mathbf{r}$. While the previous paper to consider this IRL model computed maximum likelihood estimates for $\boldsymbol{\lambda}$ and $\mathbf{u}$, and made an assumption that $\mathbf{r}$ in (5) has zero variance [23], we aim to avoid this assumption and use VI to approximate the full posterior distribution $p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}|\mathcal{D}, \mathbf{X}_{\mathbf{u}})$. *Variational inference* is an approximation technique for probability densities [6]. Let $q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})$ be our approximating family of probability distributions for $p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}|\mathcal{D}, \mathbf{X}_{\mathbf{u}})$ with its own hyperparameter vector $\boldsymbol{\nu}$. Then the job of VI algorithms is to optimise $\boldsymbol{\nu}$ in order to minimise the *Kullback-Leibler* (KL) divergence between the original probability distribution and our approximation. KL divergence (asymmetrically) measures how different the two distributions are, and in this case can be defined as [6]

$$\begin{aligned} D_{\mathrm{KL}}(q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})||p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}|\mathcal{D}, \mathbf{X}_{\mathbf{u}})) &= \mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})}[\log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}) - \log p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}|\mathcal{D}, \mathbf{X}_{\mathbf{u}})] \\ &= \mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})}[\log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}) - \log p(\mathcal{D}, \boldsymbol{\lambda}, \mathbf{X}_{\mathbf{u}}, \mathbf{u}, \mathbf{r})] \\ &+ \mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})}[\log p(\mathcal{D}, \mathbf{X}_{\mathbf{u}})]. \end{aligned}$$

The last term is both hard to compute and constant w.r.t. $q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})$ [6], so we can remove it from our optimisation objective. The negation of what remains is often called the *evidence lower bound* (ELBO) and is defined as[1] [5, 6]

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})}\left[\log \frac{p(\mathcal{D}, \boldsymbol{\lambda}, \mathbf{X}_{\mathbf{u}}, \mathbf{u}, \mathbf{r})}{q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})}\right] \\ &= \iiint q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}) \log \frac{p(\mathcal{D}, \boldsymbol{\lambda}, \mathbf{X}_{\mathbf{u}}, \mathbf{u}, \mathbf{r})}{q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})}\, d\boldsymbol{\lambda}\, d\mathbf{u}\, d\mathbf{r}. \end{aligned} \tag{6}$$

By considering full probability distributions instead of point estimates—as long as the approximations are able to capture important features of the posterior—our predictions are likely to be more accurate and rely on fewer assumptions. Moreover, we hope to make use of various recent advancements in VI for both time complexity and approximation distribution fit (see Section 3), making the resulting algorithm competitive both in terms of running time and model fit. Therefore, we raise the following research questions[2]:

1. Does VI allow us to make more accurate reward predictions when given the same amount of training time?

2. How does the reward prediction accuracy differ between VI and sampling approaches when given the same amount of training data?

[1] Throughout the proposal, all integrals should be interpreted as definite integrals over the entire sample space.

[2] For each question, we are primarily comparing the original GP IRL approach [23] with our variational version.

3. How well does our VI model approximate the real posterior distribution?
4. Is VI better at learning behaviour driven by multiple goals?
5. Is VI more robust at handling mistakes in demonstrations?

3 Literature Survey

As mentioned in the introduction, most IRL algorithms assume that the reward function can be represented as a linear combination of features. This assumption originated in one of the earliest papers on the topic by Ng and Russell [28], which introduced several linear programming approaches to the problem. The authors also noticed that often multiple reward functions can explain the same behaviour, and suggested heuristics for reward functions that are 'far away' from reward functions that do not fit the data.

A few years later, Abbeel and Ng [1] developed an algorithm for the same formulation of the problem with a guarantee to converge quickly. Neu and Szepesvári [27] identified a weakness in Abbeel and Ng's approach: the algorithm requires features to be 'appropriately' scaled, and optimal scaling may not be known. Instead, they suggest a way to combine IRL with *apprenticeship learning*, i.e., a supervised learning task for optimal *policy* recovery (whereas IRL focuses on recovering the reward function).

Ramachandran and Amir [32] were the first to formulate IRL in terms of Bayesian learning. While the model is easily interpretable and able to handle experts that make mistakes, the algorithm can only handle small state spaces and requires Monte Carlo Markov Chain (MCMC) sampling for inference.

Ziebart et al. [56] keep the linearity assumption, but introduce an influential idea: resolving the ambiguity when multiple reward functions explain the data by appealing to the maximum entropy principle.

Choi and Kim [9] extend the Bayesian model to learn good features as well as the reward function, trying to overcome the limitation of linearity. However, the approach is quite limiting: all features are assumed to have Boolean values, and the algorithm simply learns their conjunctions.

Levine et al. [23] are the first to suggest a way to learn nonlinear reward functions without harsh restrictions on the problem domain by using GPs. We base our work primarily on their paper, and the weaknesses we hope to address have already been covered in the previous section. A recent extension to their work by Jin et al. [18] aims to harness the power of deep learning by using several layers of GPs, making the model less dependent on being provided good features. They also use VI, but with a few simplifying assumptions: deterministic training conditional for the reward vector, and fully independent training conditional for the latent state (see Section 3.1 for details).

Finally, instead of using GPs to model nonlinear reward functions, one can use a neural network (NN), as demonstrated by Wulfmeier et al. [53]. Their approach benefits from constant time inference and the ability to learn complex features either from already-given features or even from raw data. The only disadvantage (as demonstrated in the paper) is that NNs take longer to learn compared to GPs.

3.1 Variational Inference

Variational inference has seen a recent increase in interest among academics, with different approaches focusing on different goals: better time complexity, handling a wider variety of models, making approximations more accurate, and using more complex function approximation techniques (such as NNs) to infer local latent variable values without having to calculate them individually for each data point [55]. As our IRL model is based on a GP, we will begin by reviewing some of the VI approaches applied specifically to GP regression. Based on a recent review of scalable GPs [55], we will concentrate on *stochastic variational sparse approximations*, as they have achieved modelling accuracy close to that of the full GP with no approximations, with many methods providing a time complexity of $\mathcal{O}(m^3)$. Below we provide a short overview of various assumptions that have been used in approximating *sparse* GPs (i.e., GPs that use inducing points), following on a paper by Quiñonero-Candela and Rasmussen [31].

Subset of data ($\mathcal{O}(m^3)$) is a baseline method of simply using a subset of data points.

Subset of regressors ($\mathcal{O}(nm^2)$) [43, 44, 50] is a degenerate approximation that uses a weight for each inducing point. A GP is called *degenerate* if the covariance function has a finite number of non-zero eigenvalues, restricting the prior distribution to only a finite number of linearly independent functions [31].

Deterministic training conditional ($\mathcal{O}(nm^2)$) [42] approximation imposes a zero-variance normal distribution for $\mathbf{r}|\mathbf{u}$, resulting in the same mean but different variance predictions compared to the subset of regressors.

Fully independent training conditional ($\mathcal{O}(nm^2)$) [45] has the assumption that the GP values are independent of each other when conditioned on the inducing values:

$$q(\mathbf{r}|\mathbf{u}) = \prod_{i=1}^{n} p(r_i|\mathbf{u}).$$

Partially independent training conditional ($\mathcal{O}(nm^2)$) [48, 41] approximates the same distribution as the fully independent training conditional, but considers a block diagonal rather than a diagonal covariance matrix.

Transduction tailors the predictive distribution to specific test inputs [31]. As we are not too concerned about a specific set of test inputs in the IRL setting, transduction is of limited interest to us.

Authors, year	Inducing points	Hyperparameters	Complexity
Titsias, 2009 [47]	variational	variational	$\mathcal{O}(nm^2)$
Hensman et al., 2013 [14]	fixed	variational	$\mathcal{O}(m^3)$
Gal et al., 2014 [11]	variational	variational	$\mathcal{O}(nm^2)$
Cheng and Boots, 2017 [8]	variational	variational	$\mathcal{O}(nm_\alpha + nm_\beta^2)$
Hensman et al., 2017 [13]	fixed	variational	$\mathcal{O}(nm)$
Peng et al., 2017 [29]	variational	variational	$\mathcal{O}(m^3)$

Table 1: Summary of relevant VI approximations to GPs. For both inducing points and hyperparameters, 'fixed' means 'chosen before the algorithm starts' and 'variational' means 'included amongst the variational parameters'. Hyperparameters m_α and m_β refer to the number of bases used to represent the GP's mean and covariance, respectively.

Augmentation [34] aims to improve predictive accuracy by adding each test input to the inducing points.

Nyström approximation ($\mathcal{O}(nm^2)$) [51] approximates the prior covariance of $\mathbf{r}$, but can lead to negative predictive variances.

Relevance vector machine ($\mathcal{O}(m^3)$) [46] is a degenerative approximation supporting a limited range of covariance functions.

Table 1 further summarises some of the recent and/or influential GP approximation approaches that might be relevant to our situation. For many of the variables in our model (including the inducing points and the hyperparameters of the covariance function), we have three possible ways of handling them:

- Provide a full Bayesian treatment by defining a prior probability distribution. While this option can easily prevent overfitting, the choice of prior can be difficult to justify.
- Fix the value of the variable. We can then optimise the value outside of the main algorithm, e.g., by running the algorithm with a set of possible values and keeping the value that produces the best performance. This option can be slow if we are interested in finding the optimal value, but could be appropriate if the variable has little impact on the algorithm's performance.
- Treat the variable as a variational parameter. This method will efficiently optimise the value of the variable, but is vulnerable to overfitting. Furthermore, this requires us to find the derivative of the ELBO w.r.t. the variable, which could be difficult.

Titsias [47] is the first to suggest a variational approximation where inducing points are treated as variational parameters. The paper has influenced many of the later works covered in this section, and has been reformulated by Gal et al. [11] into a distributed algorithm. We will fix the inducing points for the baseline version of our algorithm, but keep the idea in mind for possible future work.

Hensman et al. [14] reduce the complexity from $\mathcal{O}(nm^2)$ to $\mathcal{O}(m^3)$ using a stochastic variational inference (SVI) approach. While results such as efficiently computable natural gradients and analytically tractable optimal solutions for variational parameters are unlikely to transfer to a different problem domain, the overall SVI framework and an approximating Gaussian distribution for $\mathbf{u}$ will play an important role in our proposal.

Cheng and Boots [8] suggest using different bases in the reproducing kernel Hilbert space for the mean and covariance functions. The paper makes few assumptions and mostly relies on the work of Titsias [47], making it seem transferable to a new domain. However, applying the idea to IRL is likely to be beyond the scope of the project.

The variational Fourier features (VFF) algorithm by Hensman et al. [13] is only defined for Matérn kernels, which is likely to be too restrictive for our situation (the original paper on using GPs for IRL [23] used the automatic relevance detection kernel that has weights controlling how important each feature is). While extending VFF to support a flexible class of kernels defined by Wilson and Adams [52] is an interesting and promising avenue of work, it is likely to be beyond the scope of this project as well.

While Peng et al. [29] provide a highly efficient distributed implementation, the derivation of the ELBO relies primarily on the fact that the evidence for a GP is a Gaussian, while in our case the evidence is anything but. Hence, it is unlikely that the ideas from this paper could be applicable to our IRL model.

Finally, since we expect $p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r} | \mathcal{D}, \mathbf{X}_\mathbf{u})$ to be highly irregular, we would like our approximation to be capable of representing a wide range of possible probability distributions. The primary way of representing complex posteriors in VI is by using *normalising flows*, i.e., a collection of invertible functions—parametrised by additional variational parameters—that are applied to latent variables [38]. Normalising flows, along with some of the ideas mentioned previously, could provide great benefits to the GP IRL VI model. However, we are forced to keep any non-essential features as possible future work in order to make the project feasible in a given time frame. Furthermore, note that due to various differences between regression and IRL, we cannot simply apply a GP VI model as a whole. Instead, smaller tricks and ideas from various papers can (and will) be used throughout the project.

4 Proposed Approach

In this section we show a feasible way to apply VI to the GP IRL model. Section 4.1 defines several previously-missing parts of the model and provides intuition for how the approximating distribution could be structured. Section 4.2 then defines the approximating distribution, where some choices are justified by a body of literature, and some are novel. Next, Section 4.3 derives a simplified expression for the ELBO. Finally, in Section 4.4 we describe our strategy for optimising the ELBO.

4.1 Preliminaries

In order to properly investigate the difference between variational inference and maximum likelihood estimation, we keep other parts of the model the same. Namely, we set the covariance function to a version of the automatic relevance detection kernel [23, 26]

$$k_{\boldsymbol{\lambda}}(\mathbf{x}_i, \mathbf{x}_j) = \lambda_0 \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\intercal \boldsymbol{\Lambda}(\mathbf{x}_i - \mathbf{x}_j) - \mathbb{1}[i \neq j]\sigma^2 \operatorname{Tr}(\boldsymbol{\Lambda})\right),$$

where λ_0 is the overall 'scale' factor for how similar or distant the states are, $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$ is a diagonal matrix that determines the relevance of each feature (where d denotes the number of features), $\mathbb{1}$ is defined as

$$\mathbb{1}[b] = \begin{cases} 1 & \text{if } b \text{ is true} \\ 0 & \text{otherwise,} \end{cases}$$

and σ^2 is set to $10^{-2}/2$ (as the original paper noted that the value makes little difference to the performance of the algorithm [23]). Our vector of hyperparameters for the covariance function is then $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_d)^\intercal$. Similarly, we keep the expression for the prior of $\boldsymbol{\lambda}$:

$$p(\boldsymbol{\lambda}|\mathbf{X}_\mathbf{u}) = \exp\left(-\frac{1}{2}\operatorname{Tr}(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2}) - \sum_{i=1}^{d}\log(\lambda_i + 1)\right). \tag{7}$$

Defining the prior for $\boldsymbol{\lambda}$ means that we commit ourselves to a Bayesian treatment of the variable as opposed to, e.g., treating it as a variational parameter. While the remainder of the proposal will stick with this assumption, the alternative is certainly worth exploring in its own right, and will be investigated after the main formulation is implemented.

Next, we can rewrite the posterior by using the chain rule and Bayes' theorem in order to get a better sense of what we are trying to approximate:

$$\begin{aligned} p(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}|\mathcal{D}, \mathbf{X}_\mathbf{u}) &= p(\boldsymbol{\lambda}|\mathbf{X}_\mathbf{u}, \mathcal{D})p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathcal{D})p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u}, \mathcal{D}) \\ &\propto p(\boldsymbol{\lambda}|\mathbf{X}_\mathbf{u}, \mathcal{D})p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathcal{D})p(\mathcal{D}|r)p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u}) \\ &\propto p(\boldsymbol{\lambda}|\mathbf{X}_\mathbf{u}, \mathcal{D})p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u})p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})p(\mathcal{D}|r)p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u}) \\ &\propto p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})p(\boldsymbol{\lambda}|\mathbf{X}_\mathbf{u})p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u})p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})p(\mathcal{D}|r)p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u}) \end{aligned}$$

Note that now there are only two unknown probability distributions: $p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})$ and $p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u})$, which can be expressed as follows:

$$\begin{aligned} p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u}) &= \int p(\mathcal{D}|r)p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u})\, d\mathbf{r}, \\ p(\mathcal{D}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}) &= \iint p(\mathcal{D}|r)p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u}, \mathbf{u})p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})\, d\mathbf{u}\, d\mathbf{r}. \end{aligned}$$

This suggests the following form for the approximation:

$$q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r}) = q(\boldsymbol{\lambda}) \times q(\mathbf{u}|\boldsymbol{\lambda}) \times q(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{u}). \tag{8}$$

4.2 Structure of the Approximating Distribution

At this point, we are forced to make assumptions about the approximate posterior in order to arrive at an implementable solution. Time permitting, ways to relax the assumptions may be investigated towards the end of the project.

First, as is common in the literature for applying VI to GPs [8, 14, 16, 47], we simply set

$$q(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{u}) = p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X}_{\mathbf{u}}, \mathbf{u}). \tag{9}$$

We can make a similarly justified choice for $q(\mathbf{u}|\boldsymbol{\lambda})$:

$$q(\mathbf{u}|\boldsymbol{\lambda}) = q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S}), \tag{10}$$

where $\mathbf{m} \in \mathbb{R}^m$ is the mean vector and the $m \times m$ positive semi-definite matrix $\mathbf{S}$ is the covariance matrix [8, 13, 15].

Next, we need to choose an approximating distribution for $\boldsymbol{\lambda}$, but, unfortunately, all papers in Table 1 either fix it or treat it as a variational parameter. Hence, we make our first assumption without justification from previous literature:

$$q(\boldsymbol{\lambda}) = \prod_{i=0}^{d} q(\lambda_i). \tag{11}$$

We want to restrict λ_0 to be positive so that $k_{\boldsymbol{\lambda}}$ would produce non-negative values and not become trivial. Similarly, we want that $\lambda_i \geq 0$ for $i = 1, \dots, d$ so that $\boldsymbol{\Lambda}$ is a positive-definite matrix. Considering the possible distributions for all $d + 1$ variables, we would like the mean to be flexible (i.e., not tied to zero, like in the exponential distribution), and the tails to converge to zero as the value of the random variable moves away from the mean. We might want to support some right skew, but the distribution should be close to symmetric with at least some parameter values. This limits our choice of distributions quite significantly, and we decide to go with the gamma distribution as it is fairly flexible and commonly used [17]. We then define the probability density functions as

$$q(\lambda_i) = \Gamma(\lambda_i; \alpha_i, \beta_i) = \frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)} \lambda_i^{\alpha_i - 1} e^{-\beta_i \lambda_i}, \quad i = 0, \dots, d, \tag{12}$$

where $\alpha_i > 0$ and $\beta_i > 0$ are parameters of the distribution, and $\Gamma(\cdot)$ is the gamma function. This gives us our vector of variational parameters $\boldsymbol{\nu} = (\mathbf{m}, \mathbf{S}, \alpha_0, \beta_0, \dots, \alpha_d, \beta_d)^\intercal$.

4.3 Evidence Lower Bound

In this section we derive and simplify the ELBO for this (now fully specified) model. In order to derive the ELBO, let us go back to (6) and write[3]

$$\mathcal{L} = \mathbb{E}[\log p(\mathcal{D}, \boldsymbol{\lambda}, \mathbf{X}_{\mathbf{u}}, \mathbf{u}, \mathbf{r})] - \mathbb{E}[\log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}, \mathbf{u}, \mathbf{r})].$$

[3] At this point, we will drop the subscript denoting which variables the expectation is taken over. Also note that throughout the derivation equality is taken to mean 'equality up to an additive constant'.

By plugging in (3) and (8), we get

$$\begin{aligned}\mathcal{L} &= \mathbb{E}[\log p(\mathbf{X_u}) + \log p(\boldsymbol{\lambda}|\mathbf{X_u}) + \log p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X_u}) + \log p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X_u}, \mathbf{u}) + \log p(\mathcal{D}|r)] \\ &\quad - \mathbb{E}[\log q(\boldsymbol{\lambda}) + \log q(\mathbf{u}) + \log q(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{u})].\end{aligned}$$

Note that $\mathbb{E}[\log p(\mathbf{X_u})]$ is just a constant, so we can simply drop it from the expression. Furthermore, since $q(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{u}) = p(\mathbf{r}|\boldsymbol{\lambda}, \mathbf{X_u}, \mathbf{u})$, they cancel each other out. Then we can substitute various terms with their definitions to get

$$\begin{aligned}\mathcal{L} &= \mathbb{E}\left[-\frac{1}{2}\operatorname{Tr}(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2}) - \sum_{i=1}^{d}\log(\lambda_i + 1)\right] + \mathbb{E}[\log \mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}})] \\ &\quad + \mathbb{E}\left[\sum_{i=1}^{N}\sum_{t=1}^{T} Q_r(s_{i,t}, a_{i,t}) - V_r(s_{i,t})\right] \quad - \sum_{i=0}^{d}\mathbb{E}\left[\log\left(\frac{\beta_i^{\alpha_i}}{\Gamma(\alpha_i)}\lambda_i^{\alpha_i - 1}e^{-\beta_i\lambda_i}\right)\right] \\ &\quad - \mathbb{E}[\log \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})].\end{aligned}$$

We can simplify the last term by noting that $-\mathbb{E}[\log \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})] = \frac{1}{2}\log|\mathbf{S}|$ up to an additive constant that depends on the number of dimensions of the distribution [2]. Also note that we cannot use this fact for $\mathbb{E}[\log \mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}})]$ because $\mathbf{K}_{\mathbf{u},\mathbf{u}}$ depends on $\boldsymbol{\lambda}$. We also plug in the definitions of $\mathcal{N}(\mathbf{u}; \mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}})$ and Q_r, and get:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\log|\mathbf{S}| + \mathbb{E}\left[-\frac{1}{2}\operatorname{Tr}(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2}) - \sum_{i=1}^{d}\log(\lambda_i + 1)\right] \\ &\quad + \mathbb{E}\left[-\frac{1}{2}\mathbf{u}^{\intercal}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}| - \frac{m}{2}\log 2\pi\right] \\ &\quad + \mathbb{E}\left[\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t}) - V_r(s_{i,t}) + \gamma\sum_{s' \in \mathcal{S}}\mathcal{T}(s_{i,t}, a_{i,t}, s')V_r(s')\right] \\ &\quad - \sum_{i=0}^{d}\mathbb{E}[\alpha_i\log\beta_i - \log\Gamma(\alpha_i) + (\alpha_i - 1)\log\lambda_i - \beta_i\lambda_i]\end{aligned}$$

Now we can remove $\mathbb{E}\left[-\frac{m}{2}\log 2\pi\right]$ since it is constant w.r.t. both the variational parameters and the variables the expectation is over, and move constants (or variational parameters) independent of the approximated variables outside of the expectations. Also note that $\mathbb{E}[\lambda_i] = \alpha_i/\beta_i$ and $\mathbb{E}[\log\lambda_i] = \psi(\alpha_i) - \log\beta_i$, where ψ is the digamma function defined as $\psi(x) = \frac{d}{dx}\log\Gamma(x)$ [5]. Moreover, we can simplify $\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t})$ by choosing to represent all visited states in $\mathbf{r} = (r_1, \dots, r_k)^{\intercal}$, and defining a new vector $\mathbf{t} = (t_1, \dots, t_k)^{\intercal}$, where t_i is the number of times the state associated with the reward r_i has been visited across all demonstrations. Then

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N}\sum_{t=1}^{T} r(s_{i,t})\right] &= \mathbb{E}[\mathbf{t}^{\intercal}\mathbf{r}] = \mathbf{t}^{\intercal}\mathbb{E}[\mathbf{r}] = \mathbf{t}^{\intercal}\mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u}) \sim q(\boldsymbol{\lambda})q(\mathbf{u})}[\mathbf{K}_{\mathbf{r},\mathbf{u}}^{\intercal}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}] \\ &= \mathbf{t}^{\intercal}\mathbb{E}_{\boldsymbol{\lambda} \sim q(\boldsymbol{\lambda})}[\mathbf{K}_{\mathbf{r},\mathbf{u}}^{\intercal}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}] = \mathbf{t}^{\intercal}\mathbb{E}[\mathbf{K}_{\mathbf{r},\mathbf{u}}^{\intercal}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m}.\end{aligned}$$

Finally, as $\mathbf{u}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}$ is a function of $\mathbf{u}$ and $\boldsymbol{\lambda}$, we can take the expectation of $\mathbf{u}$, leaving the expectation of $\boldsymbol{\lambda}$:

$$\mathbb{E}[\mathbf{u}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}] = \mathbb{E}[\mathrm{Tr}(\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{S}) + \mathbf{m}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}] = \mathrm{Tr}(\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{S}) + \mathbf{m}^\intercal\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m}.$$

This allows us to simplify $\mathcal{L}(\boldsymbol{\nu})$ to the following:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\nu}) = {} & \frac{1}{2}\log|\mathbf{S}| - \frac{1}{2}\mathrm{Tr}(\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2}]) - \sum_{i=1}^{d}\mathbb{E}[\log(\lambda_i+1)] - \frac{1}{2}\mathrm{Tr}(\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{S}) \\ & - \frac{1}{2}\mathbf{m}^\intercal\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m} - \frac{1}{2}\mathbb{E}[\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}|] + \sum_{i=0}^{d}\alpha_i - \log\beta_i + \log\Gamma(\alpha_i) + (1-\alpha_i)\psi(\alpha_i) \\ & + \mathbf{t}^\intercal\mathbb{E}[\mathbf{K}_{\mathbf{r},\mathbf{u}}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m} - \sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}[V_r(s_{i,t})] - \gamma\sum_{s'\in\mathcal{S}}\mathcal{T}(s_{i,t},a_{i,t},s')\mathbb{E}[V_r(s')].\end{aligned}$$

4.4 Variational Inference

The typical way to optimise a quantity (the ELBO, in this case) involves computing its gradient [6]. Unfortunately, some of the terms in $\mathcal{L}$ are still left as expected values. *Black box variational inference* (BBVI) [33] suggests a way to express the gradient as an expectation without having to take the gradient of the posterior:

$$\nabla_{\boldsymbol{\nu}}\mathcal{L} = \mathbb{E}_{(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})}[\nabla_{\boldsymbol{\nu}}\log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})(\log p(\mathcal{D},\boldsymbol{\lambda},\mathbf{X}_{\mathbf{u}},\mathbf{u},\mathbf{r}) - \log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r}))].$$

The gradient of the ELBO then has a unbiased estimate

$$\nabla_{\boldsymbol{\nu}}\mathcal{L} \approx \frac{1}{S}\sum_{s=1}^{S}\nabla_{\boldsymbol{\nu}}\log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}_s,\mathbf{u}_s,\mathbf{r}_s)(\log p(\mathcal{D},\boldsymbol{\lambda}_s,\mathbf{X}_{\mathbf{u}},\mathbf{u}_s,\mathbf{r}_s) - \log q_{\boldsymbol{\nu}}(\boldsymbol{\lambda}_s,\mathbf{u}_s,\mathbf{r}_s)),$$

computed by drawing S Monte Carlo samples $(\boldsymbol{\lambda}_s,\mathbf{u}_s,\mathbf{r}_s)\sim q_{\boldsymbol{\nu}}(\boldsymbol{\lambda},\mathbf{u},\mathbf{r})$.

The main theorem behind the BBVI trick is the Lebesgue's dominated convergence theorem [39], which—under some conditions—allows one to claim that:

$$\lim_{n\to\infty}\int_E f_n\,d\mu = \int_E \lim_{n\to\infty} f_n\,d\mu.$$

We can use the same idea on parts of $\mathcal{L}$, derivatives of which cannot be taken otherwise. In our case, we want a gradient instead of a limit, and the integral represents an expected value.

Let us briefly examine the options available in case the theorem is not applicable to some of our expected values. First,

$$\frac{1}{2}\mathrm{Tr}(\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-2}]) \quad \text{and} \quad \sum_{i=1}^{d}\mathbb{E}[\log(\lambda_i+1)]$$

come from a chosen prior distribution for $\boldsymbol{\lambda} | \mathbf{X}_\mathbf{u}$. There are two ways to deal with them: replacing the prior with an easier-to-handle alternative, and treating $\boldsymbol{\lambda}$ as a variational parameter.

Then,

$$\frac{1}{2}\operatorname{Tr}(\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{S}), \quad \frac{1}{2}\mathbf{m}^\intercal\mathbb{E}[\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m}, \quad \text{and} \quad \frac{1}{2}\mathbb{E}[\log|\mathbf{K}_{\mathbf{u},\mathbf{u}}|]$$

all derive from $p(\mathbf{u}|\boldsymbol{\lambda}, \mathbf{X}_\mathbf{u})$, which cannot be eliminated or replaced without changing the idea of the project. However, notice that the expressions inside $\mathbb{E}[\cdot]$ are all functions of $\mathbf{K}_{\mathbf{u},\mathbf{u}}$, which depends only on $\boldsymbol{\lambda}$ and $\mathbf{X}_\mathbf{u}$. Thus, if we make $\boldsymbol{\lambda}$ a variational parameter, the expected values are no longer taken w.r.t. $\boldsymbol{\lambda}$, and $\mathbb{E}[f(\mathbf{K}_{\mathbf{u},\mathbf{u}})] = f(\mathbf{K}_{\mathbf{u},\mathbf{u}})$ for any function f that does not involve $\mathbf{u}$ or $\mathbf{r}$. As a result, the three terms simplify to expressions without expected values.

Lastly, $\mathbf{t}^\intercal\mathbb{E}[\mathbf{K}_{\mathbf{r},\mathbf{u}}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}]\mathbf{m}$ and both instances of $\mathbb{E}[V_r(s)]$ come from $p(\mathcal{D}|r)$. The first expected value can be resolved by making $\boldsymbol{\lambda}$ a variational parameter. The MDP value function requires more effort. We can start by making the deterministic training conditional assumption (as is commonly done in previous work [18, 23]), which sets the posterior covariance matrix of $\mathbf{r}$ to the zero matrix. Then the posterior $\mathbf{r} = \mathbf{K}_{\mathbf{r},\mathbf{u}}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}$. Hence we still have a (complicated) function of $\boldsymbol{\lambda}$ and $\mathbf{u}$ in the integral w.r.t. both variables. The integral (or expected value) w.r.t. $\boldsymbol{\lambda}$ can be eliminated the usual way, i.e., by making $\boldsymbol{\lambda}$ variational. As for $\mathbf{u}$, we can borrow a trick from the deep GP IRL paper [18] and set $q(\mathbf{u}) = \delta(\mathbf{u} - \mathbf{m})$, where δ is the Dirac delta function. The expected value then becomes $\mathbb{E}[V_r(s)] = V_{\mathbf{K}_{\mathbf{r},\mathbf{u}}^\intercal\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}}(s)$.

We can then take the BBVI algorithm as the basis for ours, keeping in mind the numerous improvement possibilities provided both by the BBVI paper [33] and many others (see Section 3).

5 Work Plan

The work plan can be summarised as follows:

1. Derive expressions for the derivatives of the ELBO w.r.t. the variational parameters. *Deliverables*: full derivations that can be included as supplementary material of the final paper, including formal proofs for applications of the dominated convergence theorem. *Date*: 15 December.

2. Implement the algorithm as a more specialised version of Algorithm 2 in the BBVI paper [33]. As most of the relevant IRL algorithms have been implemented in MATLAB[4,5], it makes sense to do the same in order to have accurate time-sensitive comparisons between the algorithms. *Deliverables*: a working implementation that can

[4] `http://graphics.stanford.edu/projects/gpirl/irl_toolkit.zip`

[5] `https://github.com/jinming99/DGP-IRL`

be integrated into the IRL toolkit by Levine et al. [23], and the same algorithm written in pseudo code for the paper. *Date*: 1 February.

3. Evaluate the algorithm by comparing it to multiple alternatives, as detailed in Section 5.1. *Deliverables*: multiple plots in an SVG or PDF format, along with code that produces them. *Date*: 1 April.

4. Write/finalise the paper. *Deliverable*: a 14-page paper. *Date*: around 23 April, the deadline has not been updated yet.

5.1 Evaluation

We would like to compare our approach to alternatives that support nonlinear reward functions. According to a recent survey [3], this leaves us with five algorithms: LEARCH [37], MmpBoost [36], the original GP IRL paper [23], its deep GP extension [18], and an NN-based reward function approximation [53]. As the first two have already been shown to underperform [23], we will focus on the remaining three.

The algorithms will be compared on variations of two commonly used fictional scenarios: object world [23] and highway driving behaviour [1, 22]. While more recent papers (especially those using deep learning) often use the binary world benchmark as well [18, 53], our approach is not able to construct new features, and thus cannot perform well on such a task.

Our main evaluation metric is *expected value difference* (EVD) [18], calculated as

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

where π^* is the optimal policy, and $\hat{\pi}$ is the policy generated using our reward function. We can measure how EVD changes both over time and as the number of demonstrations increase during training as well as use EVD to quantify how well the learned GP reward functions transfer to previously unseen yet similar environments. The learned reward functions can also be visualised as shades of grey on maps, just like in previous papers [18, 23]. Lastly, to evaluate how well the variational approximation models the real posterior, we can use two metrics recently proposed by Yao et al. [54]: Pareto-smoothed importance sampling that measures overall goodness of fit, and variational simulation-based calibration that can detect bias in point estimates.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*,

volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. Briefly mentioned in the literature review.

[2] Nabil A. Ahmed and Digambar V. Gokhale. Entropy expressions and their estimators for multivariate distributions. *IEEE Trans. Information Theory*, 35(3):688–692, 1989. The entropy of a multivariate Gaussian distribution.

[3] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *CoRR*, abs/1806.06877, 2018. Exactly what it says on the tin. Only cited for the list of algorithms I'm supposed to be competing against.

[4] Francis R. Bach and David M. Blei, editors. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[5] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. Describes VI, KL divergence, and properties of the gamma distribution.

[6] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. A recent review of VI.

[7] Kenneth D. Bogert and Prashant Doshi. Multi-robot inverse reinforcement learning under occlusion with estimation of state transitions. *Artif. Intell.*, 263:46–73, 2018. Using IRL to predict positions of patrolling robots.

[8] Ching-An Cheng and Byron Boots. Variational inference for Gaussian process models with linear complexity. In Guyon et al. [12], pages 5190–5200. One of the papers on applying VI to GPs.

[9] Jaedeug Choi and Kee-Eung Kim. Bayesian nonparametric feature construction for inverse reinforcement learning. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 1287–1293. IJCAI/AAAI, 2013. First attempt at learning rewards that are not linear combinations of features by constructing new features.

[10] Jennifer G. Dy and Andreas Krause, editors. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2018.

[11] Yarin Gal, Mark van der Wilk, and Carl E. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3257–3265, 2014. One of the algorithms for applying VI to GPs.

[12] Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017.

[13] James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18:151:1–151:52, 2017. One of the algorithms for applying VI to GPs.

[14] James Hensman, Nicoló Fusi, and Neil D. Lawrence. Gaussian processes for big data. In Ann Nicholson and Padhraic Smyth, editors, *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press, 2013. One of the algorithms for applying VI to GPs.

[15] Quang Minh Hoang, Trong Nghia Hoang, and Kian Hsiang Low. A generalized stochastic variational Bayesian hyperparameter learning framework for sparse spectrum Gaussian process regression. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2007–2014. AAAI Press, 2017. One of the papers for applying VI to GPs. Not cool enough to be described separately, but still comes up.

[16] Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data. In Bach and Blei [4], pages 569–578. One of the papers on applying VI to GPs.

[17] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to Mathematical Statistics*. What's New in Statistics Series. Pearson, 2018. Used as a reference for the gamma distribution.

[18] Ming Jin, Andreas C. Damianou, Pieter Abbeel, and Costas J. Spanos. Inverse reinforcement learning via deep Gaussian process. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. This paper builds on top of the original GPIRL paper to add an extra layer of latent variables.

[19] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *I. J. Social Robotics*, 8(1):51–66, 2016. An example of how IRL can be used in socially adaptive robot path planning.

[20] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *I. J. Robotics Res.*, 35(11):1289–1307, 2016. An example of how IRL can be used in socially adaptive robot path planning.

[21] Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors. *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*. MIT Press, 2001.

[22] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Feature construction for inverse reinforcement learning. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 1342–1350. Curran Associates, Inc., 2010. The highway driving scenario is based on this paper.

[23] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with Gaussian processes. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 19–27, 2011. First paper to tackle rewards that cannot be expressed as a linear combination of features.

[24] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Supplementary material: Nonlinear inverse reinforcement learning with Gaussian processes. `http://graphics.stanford.edu/projects/gpirl/gpirl_supplement.pdf`, December 2011. Contains derivations of likelihood partial derivatives and additional details about the implementation.

[25] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *CoRR*, abs/1807.01065, 2018. A recent review of scalable GPs.

[26] R.M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer New York, 2012. The first mention of automatic relevance detection/determination kernels.

[27] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In Ronald Parr and Linda C. van der Gaag, editors, *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 295–302. AUAI Press, 2007. One of the early papers about IRL.

[28] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000. An influential early approach to the IRL problem.

[29] Hao Peng, Shandian Zhe, Xiao Zhang, and Yuan Qi. Asynchronous distributed variational Gaussian process for regression. In Doina Precup and Yee Whye Teh, editors,

Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, pages 2788–2797. PMLR, 2017. One of the VI for GP papers.

[30] Qifeng Qiao and Peter A. Beling. Inverse reinforcement learning with Gaussian process. *CoRR*, abs/1208.2112, 2012. An alternative formulation of IRL with GPs.

[31] Joaquin Quiñonero-Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005. A review of sparse approximations for GP regression.

[32] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2586–2591, 2007. Mentioned in the literature survey.

[33] Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 814–822. JMLR.org, 2014. A basis for any algorithm I can come up with.

[34] Carl E. Rasmussen. Reduced rank Gaussian process learning. *Unpublished manuscript*, 2002. The first time augmentation was suggested.

[35] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. The main book on GPs.

[36] Nathan D. Ratliff, David M. Bradley, J. Andrew Bagnell, and Joel E. Chestnutt. Boosting structured prediction for imitation learning. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1153–1160. MIT Press, 2006. Another early attempt at nonlinear IRL.

[37] Nathan D. Ratliff, David Silver, and J. Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Auton. Robots*, 27(1):25–53, 2009. One of the first attempts at nonlinear IRL.

[38] Danilo J. Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Bach and Blei [4], pages 1530–1538. How to use VI to approximate distorted posterior distributions.

[39] Halsey Lawrence Royden and Patrick Fitzpatrick. *Real analysis*, volume 32. Macmillan New York, 1988. Cited for the dominated convergence theorem.

[40] Stuart J. Russell. Learning agents for uncertain environments (extended abstract). In Peter L. Bartlett and Yishay Mansour, editors, *Proceedings of the Eleventh Annual*

Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998., pages 101–103. ACM, 1998. The first paper (talk) that defines inverse reinforcement learning.

[41] Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 953–960. MIT Press, 2002. The second paper on the partially independent training conditional approximation.

[42] Matthias W. Seeger, Christopher K. I. Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003.* Society for Artificial Intelligence and Statistics, 2003. The paper on the deterministic training conditional GP approximation.

[43] Bernhard W Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–52, 1985. First paper about the subset of regressors approximation.

[44] Alexander J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In Leen et al. [21], pages 619–625. The third paper about the subset of regressors approximation.

[45] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 1257–1264, 2005. The paper on the fully independent training conditional GP approximation.

[46] Michael E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001. The relevance vector machine as an approximation for a GP.

[47] Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David A. Van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, volume 5 of *JMLR Proceedings*, pages 567–574. JMLR.org, 2009. One of the earlier and more influential papers on applying VI to GPs.

[48] Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000. The first paper on the partially independent training conditional approximation.

[49] Adam Vogel, Deepak Ramachandran, Rakesh Gupta, and Antoine Raux. Improving hybrid vehicle fuel efficiency using inverse reinforcement learning. In Jörg Hoffmann

and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press, 2012. Using IRL to predict where the driver is going.

[50] Grace Wahba, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein, and Barbara E. Klein. The bias-variance tradeoff and the randomized GACV. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems 11, [NIPS Conference, Denver, Colorado, USA, November 30 - December 5, 1998]*, pages 620–626. The MIT Press, 1998. The second paper about the subset of regressors approximation.

[51] Christopher K. I. Williams and Matthias W. Seeger. Using the Nyström method to speed up kernel machines. In Leen et al. [21], pages 682–688. The paper on Nyström approximation.

[52] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1067–1075, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. A class of kernels generated from a mixture of Gaussian spectral densities.

[53] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015. An IRL algorithm that uses a neural network.

[54] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. In Dy and Krause [10], pages 5577–5586. A recent paper presenting two ways to evaluate VI.

[55] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *CoRR*, abs/1711.05597, 2017. Another recent review, focusing more on comparing algorithms.

[56] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. An influential idea: maximum entropy IRL.

[57] 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 Hee Yong Youn and We-Duke Cho, editors, *UbiComp 2008: Ubiquitous Computing, 10th International Conference, UbiComp 2008, Seoul, Korea, September 21-24, 2008, Proceedings*, volume 344 of *ACM International Conference Proceeding Series*, pages 322–331. ACM, 2008. Using IRL to predict the behaviour of taxi drivers.

[58] Brian D. Ziebart, Nathan D. Ratliff, Garratt Gallagher, Christoph Mertz, Kevin M. Peterson, James A. Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha S. Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, pages 3931–3936. IEEE, 2009. Using IRL to predict the movement of pedestrians.