Research Article

Yuling Max Chen*

Clustered Autoencoded Variational Inverse Reinforcement Learning

https://doi.org/10.1515/stat-2022-0109
Received Feb 28, 2022; accepted Jul 28, 2022

Abstract: **V**ariational **A**uto-**E**ncoder (VAE) is a handy and computationally friendly Bayesian tool for **I**nverse **R**einforcement **L**earning (IRL) problems, with the native setting of absent reward functions in a **M**arkov **D**ecision **P**rocess (MDP). However, recent works mainly deal with single reward, which turn out to be insufficient for complex dynamic environments with multiple demonstrators of various characteristics (hence multiple reward functions). This paper extends the dimensionality of reward (from $\mathbb{R}$ to $\mathbb{R}^K$) by incorporating a latent embedding and clustering step on top of a scalable Bayesian IRL model, which enhances her applicability to multi-reward scenarios. We introduce our method, **C**lustered **A**utoencoded **V**ariational **I**nverse **R**einforcement **L**earning (CAVIRL), which is able to approximate multiple posterior reward functions and learn the corresponding policies for experts of various characteristics and skills. As a by-product, the proposed model also thrives to determine the number of clusters K on her own, as opposed to the competing multi-reward imitation learning models that require K to be prespecified. We trained the proposed model within a grid world with multiple types of players, where we achieved 100% correctness in determining the number of players' types and 80%-83.9% match between the model-learned policies and the players' demonstrations from the data.

Keywords: Markov Decision Process, Variational Inference, Latent Variable Model, Clustering

2020 Mathematics Subject Classification: 60J20, 68T20, 62H30, 68W25, 93E35

1 Introduction

It is often a hard problem to understand how and why demonstrators react to a dynamic environment, especially when it takes expertise to make each analytical decision.

Apprenticeship learning, as it sounds like, trains an agent by letting it to watch and imitate the demonstration of demonstrators, as if it were an apprentice of the masters. Inverse Reinforcement Learning (Ng and Russell, 2000) is a common "first-step" of Apprenticeship Learning, as it aims to reveal the underlying benchmark a demonstrator bases off when making decisions. In a MDP, such benchmark is the *reward* function that compensates (or penalises if negative) the behaviours of the demonstrators, which is thereby the target of the so-called *Reward Imitation Learning*. According to Abbeel and Ng (2004), the reward function is deterministically represented by the coefficients of the optimal policy in form of an explicit linear combination. Consecutively, once we understand the underlying reward, we will be able to imitate the decision-making process of the demonstrators and inference on the policy via "forward" Reinforcement Learning, which is the "second-step" of Apprenticeship Learning.

While Abbeel and Ng (2004) learn the reward function via solving a **L**inear **P**roblem (LP), **B**ayesian **I**nverse **R**einforcement **L**earning (BIRL) (Ramachandran and Amir, 2007) is a more generative approach that makes it possible to: 1) learn from suboptimal demonstrations, 2) incorporate priori scientific knowledge to the model and hence generalize the reward that is of higher relevance to the specific task of interest, and 3) provide a stochastic realization of the learned policy from the generalized (policy) posterior distribution. Some

***Corresponding Author: Yuling Max Chen:** University of Oxford; E-mail: yuling.chen@eng.ox.ac.uk

efforts were made to achieve more concrete estimation of the reward, e.g., via Maximum-a-Posterior Inference (Choi and Kim, 2011), whereas they all suffer from the "scalability trap" — marginalization of likelihoods and evaluation of the Q-values (i.e., expected reward) can be intractable in a high dimensional environment (Chan and van der Schaar, 2021).

Scalable **B**ayesian **I**nverse **R**einforcement **L**earning (SBIRL) (Chan and van der Schaar, 2021) applies a variational approximator for the reward posterior, utilizing deep learning networks to cure intractability of functional evaluations in an MDP loop (i.e., state-value function $V^{\pi}(s)$ and the Q–function $Q^{\pi}(s, a)$ as defined by Abbeel and Ng (2004)). However, this approach only learns a single reward function among all demonstrators, which implicitly assumes that all demonstrators behave similarly under the same policy.

Built on SBIRL, this paper aims to fit a scalable IRL model in an extended environment where demonstrators have different characteristics and hence behave differently based on several different reward functions, i.e., the "multi-reward" case. Some previous works have researched on the "multi-reward" scenario, but they tended to suffer from various drawbacks. We highlight and comment on three main streams of previous works as follow:

1. Bayesian Multitask Inverse Reinforcement Learning (Dimitrakakis and Rothkopf, 2012) defined a **M**ultitask **P**olicy **O**ptimality (MPO) prior β as a function of the natural belief ϵ of the policy, i.e., $\beta([0, \epsilon])$ on $\mathbb{R}_+$, reflecting the priori belief of the ϵ–optimality of each policy. Joined with the MPO, the reward functions are sampled from a pre-defined arbitrary measure ψ conditioned on the demonstrations, which is computed by 2 steps of marginalization over the natural belief ϵ and the policy π (Equation 1).

$$\psi(R|D) = \int_{\mathcal{P}} \psi(R|\pi) d\xi(\pi|D) = \int_{\mathcal{P}} \int_{0}^{\infty} \psi(R|\epsilon, \pi) d\beta(\epsilon) d\xi(\pi|D) \tag{1}$$

   Unfortunately, the arbitrary measure ψ limits the applicability of this method to finite (discrete) reward function space $\mathcal{R}$ with a prespecified dimension K. It hence suffers from large memory complexity for the loss matrix if too many reward functions are taken into account, i.e., when K is large. Moreover, all demonstrations D are pooled together in absence of clustering effort, which makes it potentially harder to analyze the variation of behaviours among demonstrators with different characteristics, as you can barely tell them apart.
2. Some work applied the **D**irichlet **P**rocess **M**ixture (DPM) model to cluster the demonstrations, e.g., DPM-BIRL Choi and Kim (2012). The baseline hierarchy of DPM model contains: a) the probabilities of the latent cluster classes $\mathbf{p} = (p_1, \cdots, p_K)$ following a Dirichlet distribution (Equation 2) and b) the latent cluster labels c_m following a Multinomial distribution (Equation 3).

$$\mathbf{p}|\alpha \sim \text{Dirichlet}(\alpha/K, \cdots, \alpha/K) \quad \text{, for some constant } \alpha \tag{2}$$

$$c_m|\mathbf{p} \sim \text{Multinomial}(p_1, \cdots, p_K) \tag{3}$$

   DPM-BIRL assumes that the behaviour of each new demonstration trajectory is generated from the same distribution of behaviours in the observed demonstrations. As a result, a simply designed Metropolis-Hasting MCMC simulation suffers from a "selection bias" on the latent cluster labels c_m —— the successive demonstration trajectory is more likely to be assigned to the largest cluster that contains the most precedent demonstrations. The between-clusters correlations thereby fail to represent the clustering process in a realistic scenario. In fact, we would like to group a demonstration trajectory into a cluster because the demonstrator behaves similarly as those in the cluster, rather than because the cluster is large and contains more precedent trajectories. Besides, the number of clusters K is still required to be given.
3. Other work, e.g., Yu et al. (2019), utilized **G**enerative **A**dversarial **N**etwork (GAN) to approximate the reward function in an MDP with multiple demonstrators, where the generator induces demonstrations from the learned reward functions (via importance sampling) and the discriminator distinguishes the induced demonstrations from the truly-observed demonstrations. While this (to some extent) cures the scalability problem suffered by those aforementioned MCMC simulation models (1, 2), the adversarial reward estimator fail to generalize the characteristics among multiple demonstrators, as the model

implicitly assumes that each demonstrator has a unique reward function, i.e., number of clusters K equals number of observed demonstration trajectories N. However, in practice, demonstrators with similar behaviours might share the same reward function, i.e., $K \ll N$. Whether K equals N or not, K is still required to be fixed in order to let the GAN-based model work.

To the best of our knowledge, the past works on "multi-reward" IRL problem, including the aforementioned 3 streams above, suffer from mainly 3 drawbacks: (a) "scalability trap", (b) "selection bias" due to lack of clustering effort, and (c) inflexibility of K that has to be prespecified.

Therefore, in this paper we will introduce the **C**lustered **V**ariational **A**utoencoded **I**nverse **R**einforcement **L**earning (CVAIRL), which is designed to deal with the 3 drawbacks and achieve the following 3 goals:

1. to reveal the optimal number of clusters $K^\star$ among demonstrators and cluster the demonstrators accordingly,
2. to imitate the underlying reward functions for each cluster of demonstrators and thereby to induce the optimal policies for each cluster,
3. to be scalable for complex and high-dimensional dynamic environment by replacing inner loop Markov solve with neural networks.

Without loss of generality, we herein make the following 4 assumptions:

1. Demonstrators do not affect each other, namely, demonstrations are observed (and sampled) independently;
2. The characteristics of a demonstrator dominate (completely explain) his/her behaviours, hence it is sufficient to study the variation of behaviours by clustering the demonstrators according to their characteristics;
3. Demonstrators can change their characteristic over time, but they behave consistently along a demonstration trajectory. So, if a particular demonstrator A changes his/her characteristic between performing demonstration D_1 and D_2, then D_1, D_2 can be assigned to 2 different clusters, and A in D_1 is considered as a different person as A in D_2;
4. Demonstrators take the action that corresponds to the highest reward, without ϵ-greedy explorations (Sutton and Barto, 2018).

In the following sections, we will first go through some preliminaries in Section 2 and then introduce the model in Section 3. Section 4 shows some experimental results when we applied our model to the synthetic data simulated under the 4 assumptions mentioned above, and we conclude with Section 5 where we comment on our model and discuss the potential future works. Some Python source code of the model construction and experiments can be found in the github repo.

2 Preliminaries

2.1 Markov Decision Process (MDP) and Markov Properties

The sequential decision-making process in a dynamic environment is commonly modeled via a MDP. A MDP is a discretized-time stochastic control process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, involving the *states* ($s \in \mathcal{S}$), the *actions* ($a \in \mathcal{A}$), the *transition probability* ($P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$), and the *reward function* ($R : \mathcal{S} \times \mathcal{A} \to \mathcal{R} \subseteq \mathbb{R}$). At each time step $t \in \{1, \cdots, T\}$, an agent takes action a_t given state s_t, collects reward $R(s_t, a_t)$, and transits to the next state s_{t+1} with probability $P(s_t, a_t, s_{t+1})$.

In this paper, we introduce another discrete variable, *characteristics* ($c \in \mathcal{C} \subseteq \mathbb{N}$), representing the various characteristics of the demonstrators. Ideally demonstrators with the same characteristic c share the same reward function R_c and hence behave similarly. Therefore, a key step is to group the demonstrators into clusters with respect to characteristics. In our problem setup, the number of characteristic clusters $K = |\mathcal{C}|$ is

unknown and is needed to be learned from the data. In order to prevent over-stratification, we posit an upper bound for K — a predetermined *maximum number of clusters* $K_{\max}$.

Remark: *In the following subsections ([2.2, 2.3]), however, we will leave the characteristics label c for a moment, as the preliminary works do not concern the multi-reward scenario. We will pick c up again and embed it to our model formulation in Section 3.*

Modelling the dynamic environment with a MDP allows us to take the advantage of the following Markov Property (**Theorem** 1), by considering the state-action pair as random variable, i.e., $X = (s, a)$.

Theorem 1 (Factorization Property). *Suppose $\mathcal{G} = (V, E)$ is a* **D***irected* **A***cyclical* **G***raph (DAG) with node set V and edge set E. Let X_V be the set of variables over $\mathcal{G}$ following a probability distribution p. Then $p(x_V)$ factorizes $\mathcal{G}$ if and only if*

$$p(x_V) = \prod_{v \in V} p\left(x_v | x_{\mathrm{pa}_{\mathcal{G}}(v)}\right), \forall x_v \in X_V \tag{4}$$

where $\mathrm{pa}_{\mathcal{G}}(v)$ is the set of parents of the node $v \in V$.

2.2 Inverse Reinforcement Learning (IRL) and The Bayesian Approach

A typical IRL problem has no access to the transition probability and the rewards, leaving only the states and actions available in the form of demonstration trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^N$. Each trajectory is a sequence of states and corresponding actions taken by a demonstrator i: $\tau_i = \{s_1^{(i)}, a_1^{(i)}, \cdots, s_{T_i}^{(i)}, a_{T_i}^{(i)}\}$, where T_i is the length (in terms of time steps) of the i-th trajectory.

With a *discounting factor* γ, the Bellman equations define the *expected value of a policy π at state s* (Equation 5) and the *expected value of a state-action pair* (Equation 6), according to Sutton and Barto (2018).

$$V_R^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s, \pi(s), s') V_R^\pi(s') \tag{5}$$

$$Q_R^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s, a, s') V_R^\pi(s') \tag{6}$$

While the goal of a (forward) Reinforcement Learning is to learn an *optimal policy* ($\pi^\star$) that achieves the highest expected value for each certain action at each given state (Equation 7), the IRL aims to learn the reward function based on which demonstrators choose the optimal policy (Equation 8). Reward functions are linearly formulated as combinations of basis functions with penalty terms (e.g., L_p-penalty) to discourage boldly complicated formulations, according to Ng and Russell (2000).

$$\pi^\star(s) = \arg\max_{a \in \mathcal{A}} Q_R^\pi(s, a) \quad \text{Bellman Optimality (Sutton and Barto, 2018)} \tag{7}$$

$$R(s, a) = \arg\max_{a \in \mathcal{A}} Q_R^{\pi^\star}(s, a) \tag{8}$$

The Bayesian approach, namely **B**ayesian **I**nverse **R**einforcement **L**earning (BIRL) according to Ramachandran and Amir (2007), further assumes conditional independence among the conditional likelihoods of state-action pairs along the trajectory given the reward function (Equation 9), according to the *Factorization Property* (**Theorem 1**).

$$P(\tau_i | R) = \prod_{t=1}^{T_i} P((s_t, a_t) | R) \tag{9}$$

Each individual likelihood (in Equation 9) is modeled as an exponential family (Equation 10) in (Ramachandran and Amir, 2007) and the normalization constant Z_t is further parameterized by a Boltzmann distribution with energy Q and temperature β^{-1} marginalized over the action space $\mathcal{A}$ in (Ziebart et al., 2008), where β is the confidence parameter showing how optimal the demonstrator acts.

$$P((s_t, a_t) \mid R) = \frac{1}{Z_t} \exp(\beta Q_R^\pi(s_t, a_t)) = \frac{\exp(\beta Q_R^\pi(s_t, a_t))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q_R^\pi(s_t, a'))} \tag{10}$$

Following the independence assumption and the Bayes rule, the posterior of the reward function R given the demonstration trajectory τ is given by Equation 11, where P_R is the prior of the reward R.

$$P(R \mid \tau) = \frac{P(\tau \mid R) P_R(R)}{P_\tau(\tau)} \propto \exp\left(\beta \sum_t Q_R^\pi(s_t, a_t)\right) P_R(R) \tag{11}$$

2.3 Variational Inference and a Scalable Approach

Variational **I**nference (VI) (Blei et al., 2017) is a handy Bayesian approach to approximate the conditional distribution of the latent variable z given the observable variable x. In our problem setup, the reward R can be thought of as the latent variable along with the observed demonstrations D, hence making the application of VI particularly promising. VI approximates the target distribution $p(z|x)$ with a variational distribution $q(z)$, by minimizing the distance between them through an (usually constrained) optimization. A common practice is to use **K**ullback-**L**eibler (KL) divergence (Kullback and Leibler, 1951) as the distance measurement, and derive the minimization of KL-divergence to a maximization of the **E**vidence **L**ower **BO**und (ELBO) (Equation 12). The ELBO, as the lower bound of $\mathbb{E}(\log p(x))$ by definition, is considered as a good approximator for the marginal likelihood of the observed variable x, which forms a reliable benchmark for selecting the variational distribution q from a functional field $\mathcal{F}$.

$$\begin{aligned} \min_{q(z) \in \mathcal{F}} D_{KL}(q(z) \| p(z|x)) &\iff \max_{q(z) \in \mathcal{F}} \text{ELBO}(q) = \mathbb{E}(\log p(x|z)) - D_{KL}(q(z) \| p(z)) \\ &\iff \max_{q(R) \in \mathcal{F}} \text{ELBO}(q) = \mathbb{E}(\log p(\mathcal{D}|R)) - D_{KL}(q(R) \| p(R)) \end{aligned} \tag{12}$$

Coming back to BIRL, sampling reward from the posterior via a grid-walk styled process (as in (Ramachandran and Amir, 2007)) is computationally expensive, as either a single evaluation of Q_R^π or a parameterization of the variational reward posterior q_R require significant memory of tabularised reward data. **A**pproximate **V**ariational **R**eward **I**mitation **L**earning (AVRIL) (Chan and van der Schaar, 2021) resolves the scalability issue with 2 approximations under the VI framework:

1. Approximate the variational reward posterior $q_R(R)$ with a neural network q_ϕ parameterized by ϕ;
2. Approximate the Q-value function with another neural network Q_θ parameterized by θ.

As a result, Chan and van der Schaar (2021) rewrite the maximization in Equation 12 as a constrained maximization with respect to (ϕ, θ) (Equation 13), which simultaneously updates the 2 approximators (q_ϕ, Q_θ) in a single step. The constraint makes sure the 2 neural networks are consistent.

$$\max_{\phi, \theta} \sum_{(s,a) \in \mathcal{D}} \log \frac{\exp(\beta Q_\theta(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q_\theta(s, a'))} - D_{KL}(q_\phi(R) \| p_R(R)) \text{ subject to } E_{\pi, P}\left[-\log q_\phi\left(\hat{R}\right)\right] < \epsilon \tag{13}$$

$$\text{where, in Equation 12, } p(\mathcal{D} \mid R) = \prod_{(s,a) \in \mathcal{D}} \frac{\exp(\beta Q_R^\pi(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q_R^\pi(s, a'))}, \text{ by Equation 9 and 10} \tag{14}$$

3 Clustered Autoencoded Variational Inverse Reinforcement Learning (CAVIRL)

In this section we will define the model structure, the training steps, as well as the derivations of the loss functions. In correspondence to the 3 goals aforementioned in Section 1, the model is designed to complete the following 3 tasks:

1. **Characteristic Clustering and Embedding**: determine the optimal number of characteristic clusters among the demonstrators, cluster the demonstration trajectories, and embed the corresponding characteristic information into each demonstration;
2. **Reward Imitation Learning**: Approximate the characteristic-embedded reward function for each clusters;
3. **Policy Reinforcement Learning**: Learn the optimal policy for each demonstrator based on the imitated reward.

3.1 Model Structure

One naive way of completing the 3 tasks could be to directly cluster the demonstration trajectories, and then learn the reward and the policy via apprenticeship learning (Abbeel and Ng, 2004) for each cluster separately. This is not desirable because of the following reasons:

1. *Poor Clustering Performance*. Common clustering techniques (e.g., K-Means) can perform poorly if the data is high-dimensional or has complicate structure. The distance measurement of K-Means (e.g., Euclidean distance by default in Python) is empirically not a good measurement for the variation of sequential data, as in our analysis. The cluster means defined by K-Means are hard to be identified in a high-dimensional space. Even if they were identified, they may not be the appropriate cluster "centers", as each dimension has different explanatory weight and can be cross-correlated.
2. *Irreproducibility*. Whenever the dataset is changed and K clusters are distinguished, we have to apply K apprenticeship learning models for each cluster, which winds up with K sets of trainable model parameters. Empirically, it is undesirable for a model to have unfixed number of parameters. Because any training progress on a training dataset is immediately useless if the testing dataset has different number of clusters.
3. *Ineffective Computation*. As said in 2, the total number of parameters is linearly proportional to the number of clusters K. As a result, the memory it takes to store the parameters is linearly proportional to K and the required computation is at least linearly (usually superlinearly) proportional to K. Hence, if K is large, we will suffer from large memory- and computational-complexity.
4. *Inefficient Usage of Data*. As said in 2, each apprenticeship learning model will no longer be exposed to the full scope of the dataset, which may easily cause over-fitting especially for the smaller clusters with fewer data points.

To avoid the 4 problems mentioned above, we formulate our model as an interactive machine with five components as in Sections (3.1.1, 3.1.2, 3.1.3, 3.1.4). Figure 4 shows how a CAVIRL model is structured.

3.1.1 Latent Autoencoder $\xi_\omega : \mathcal{D} \xrightarrow{\mathcal{Z}} \mathcal{D}_{\text{Recon}}$

The *Latent Autoncoder* ξ_ω acts as an identity function that takes the demonstration trajectories $\mathcal{D}$ and outputs the reconstructed demonstration trajectories $\mathcal{D}_{\text{Recon}}$ (Figure 1). The role of ξ_ω is to project the demonstrations onto a lower-dimensional latent space $\mathcal{Z}$ that extracts sufficient information for characteristic clustering (Task 1). Ideally, the closer $\mathcal{D}_{\text{Recon}}$ and $\mathcal{D}$ are, the better the latent demonstration z is as a lower-dimensional surrogate of $\mathcal{D}$.

Remark: *The design of a latent autoencoder is flexible and dependent on the problem scenario. In general Inverse Reinforcement Learning setting,* **R***ecurrent* **N***eural* **N***etworks (RNN) is often utilized to capture the sequential stochastic relations along a demonstration trajectory.*

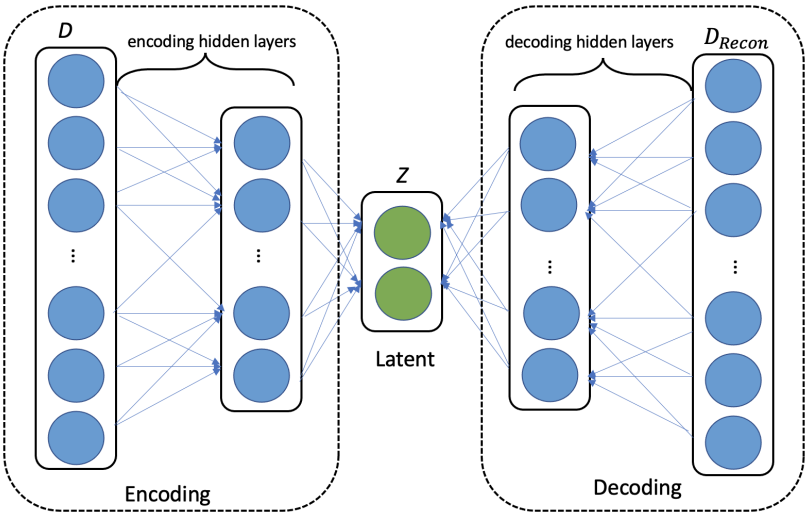


Figure 1: Graphical Illustration of the Latent Autoencoder

3.1.2 Selector $h_\psi : \mathcal{Z} \to [0,1]^{|C|}$ and Sampler $s : [0,1]^{|C|} \to \mathcal{C}$

The *Selector* h_ψ is a stacked feedforward neural network (parameterized by ψ) that takes the latent demonstrations z, and computes the probability of z being in each characteristic cluster by putting a Softmax layer as the output gate. In conjunction with the *Samper* s, this step is able to assign characteristic clustering labels c to each latent demonstration z under 2 modes:

1. *"argmax"*: choose c with the largest probability;
2. *"categorical"*: sample c from a categorical distribution with the probabilities given by the Selector.

Remark: *The output dimension of the Selector, i.e., K, is determined by the characteristic clustering step (Section 3.2.1) on the latent demonstrations z.*

3.1.3 Reward Encoder $q_\phi : \mathcal{S} \times \mathcal{C} \to [0,1]^{|\mathcal{A}|}$

The *Reward Encoder* q_ϕ is a stacked **R**ecurrent **N**eural **N**etworks (RNN) in conjunction to a sequence of hidden feedforward layers, parametrized by ϕ. It encodes the "c-embeded" states (characteristic-embedded states, see Section 3.2.2 for details) and computes the variational posterior distribution of the reward, i.e., $q_\phi(R_c)$ in Equation 13. In this paper, we adapted a common Normal assumption on the variational approximator, as did in (Chan and van der Schaar, 2021),

$$q_\phi(R_c) = N(\mu, \sigma) \propto \exp\left(-\frac{1}{2}\frac{(R-\mu)^2}{\sigma^2}\right) \tag{15}$$

Hence, the encoded hidden space is the space of mean μ and standard deviation σ of q_ϕ. The encoder is designed with 2 modes representing 2 assumptions on the variational reward distribution:

1. *"state-only"* mode: assume the reward of each action at a given state is generalized by a single posterior distribution, thereby the latent space is 2-dimensional, i.e., $(\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_+$;
2. *"state-action"* mode: assume the reward of each action at a given state is generalized by a different posterior distribution, thereby the latent space is $2|\mathcal{A}|$-dimensional, i.e., $(\mu, \sigma) \in \mathbb{R}^{|\mathcal{A}|} \times \mathbb{R}_+^{|\mathcal{A}|}$, where $|\mathcal{A}|$ is the size of the action space.

Remark: *For modelling simplicity, we encode the hidden space as a space of mean μ and logged standard deviation* $\log\sigma$*, as* $(\mu, \log\sigma) \in \mathbb{R}^2$.

3.1.4 Expected Reward Decoder $Q_\theta : \mathcal{S} \times \mathcal{C} \to \mathcal{R}$

The *Expected Reward Decoder* Q_θ is a sequence of feedforward neural networks parameterized by θ, that takes each "c-embedded" state pairs $((s_t^c, s_{t+1}^c)$ (see Section 3.2.2 for details) and evaluates the expected reward for each available action at state s_t^c, i.e., $\{Q(s_t^c, a)\}_{a \in \mathcal{A}}$.

Remark: *Both the Reward Encoder q_ϕ and Expected Reward Decoder Q_θ are consistently designed as in (Chan and van der Schaar, 2021).*

3.2 Characteristic Clustering and Embedding

Characteristic Clustering and Embedding are 2 major steps we designed to determine the number of characteristic clusters and incorporate the characteristic information to the next-step of reward imitation learning.

3.2.1 Clustering: Determining the Optimal Number of Clusters

Once the *Latent Autoencoder* encodes a set of lower-dimensional latent representation z of the demonstration trajectories $\mathcal{D}$, we perform a search of the optimal number of clusters $K^\star$ that best stratifies the data. In conjunction with K-Means clustering, this involves:

- three separability scores: Silhouette's score (Rousseeuw, 1987), Calinski-Harabasz score (Caliński and Harabasz, 1974), Davies-Bouldin index (Davies and Bouldin, 1979);
- one statistic: Gap Statistic (Tibshirani et al., 2001);
- one inspection method: Elbow Method (Thorndike, 1953).

We choose $K^\star$ that is indicated as optimal by the majority (if not all) of the aforementioned metrics. Then, we cluster and assign a characteristic label $c \in \{1, \cdots, K^\star\}$ to each demonstration trajectory.

3.2.2 Embedding: Dimensionality Extension and Trajectories-to-Pairs

Consider a demonstration trajectory $\tau_i = \{s_0, a_0, s_1, a_1, \cdots, s_{T_i}, a_{T_i}\}$ that is labeled with characteristic c_i. It can be decomposed as a *state trajectory* $\tau_i^s \equiv \{s_0, \cdots, s_{T_i}\}$ and an *action trajectory* $\tau_i^a \equiv \{a_0, \cdots, a_{T_i}\}$.

We incorporate the characteristic information into the state trajectory, via extending the state dimension by 1 at each time step, as shown in Figure 2. This procedure is termed in this paper as *"c-embedding"*. As a result, a *c-embedded* state trajectory thereby reflects the unique characteristic of the demonstrator.

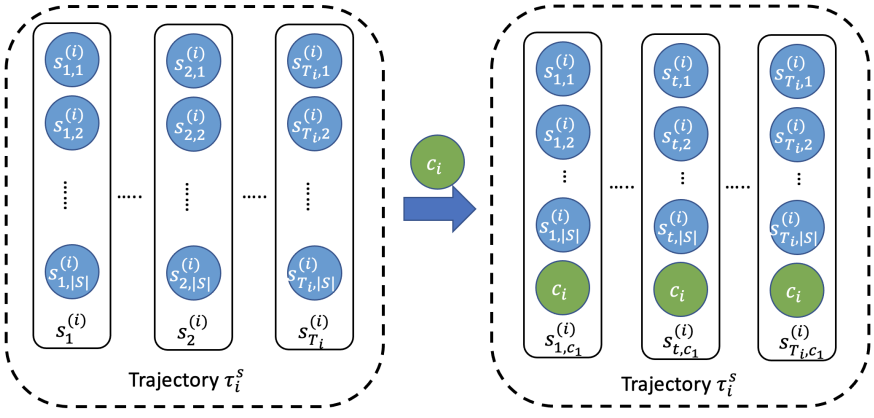


Figure 2: A Graphical Demonstration of *c-Embedding* the Characteristic Label into a State Trajectory

In preparation of the inputs of the reward encoder q_ϕ and the expected reward decoder Q_θ, we break down the trajectories into self-overlapping pairs. As in Figure 3, a state trajectory $\tau_i^s = \{s_0, \cdots, s_{T_i}\}$ following this procedure will be broken down into overlapping pairwise tuples $\{(s_0, s_1), (s_1, s_2), \cdots, (s_{T_i-1}, s_{T_i})\}$. This is a direct application of the Factorization property (Theorem 1), where each state-action pair is conditionally independent to each other given the directly precedent state-action pair along a MDP (Equation 16).

$$p(\tau_i) = p(\{s_0, a_0, s_1, a_1, \cdots, s_{T_i}, a_{T_i}\}) = p(s_0, a_0) \prod_{t=0}^{T_i-1} p((s_{t+1}, a_{t+1})|(s_t, a_t)) \tag{16}$$

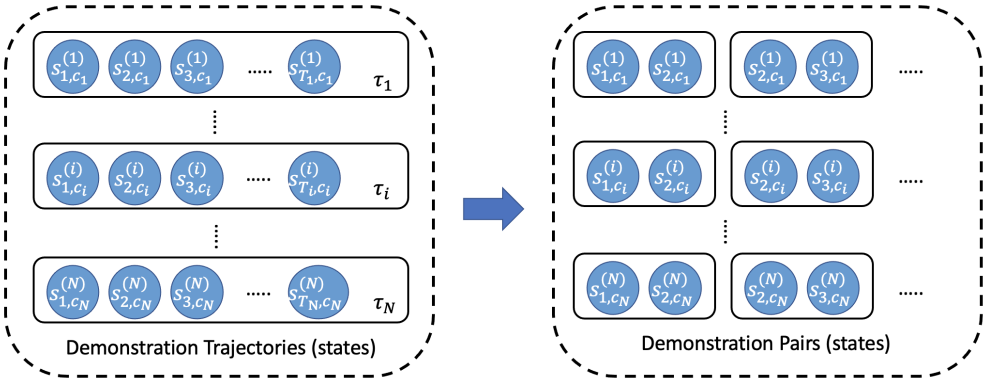


Figure 3: A Graphical Demonstration of Decomposing a State Trajectory into Pairs.

Remark: *The decomposition of the trajectories shortens the time range of each single input and relieves the model from having to bother with long sequential data, taking the advantage from the Markov property.*

3.3 Integrating the Machine

The model is integrated as 1+4 steps, with the pre-training step 0 and the rest model training steps 1-4.

0. *(Black dashed arrows in Figure 4)* Pretrain the *Latent Autoencoder* ξ_ω and determine the optimal number of characteristics $K^\star$, which is used to define the output dimension of the *Selector* h_ψ;
1. *(Green arrows in Figure 4)* Encode the demonstration trajectories $\mathcal{D} = \{\tau_i\}$ as latent demos z through the *Latent Autoencoder* ξ_ω. Compute the probability distribution of characteristics $h_\varphi(c)$ for each latent demo z through the *Selector* h_ψ, and assign a characteristic label c to each z through the *Sampler* s;
2. *(Grey dashed arrows in Figure 4)* "C-embed" the characteristic labels c into the trajectories and then decompose them into pairs $\Delta_t^{(i)}$;
3. *(Red arrows in Figure 4)* Encode the "c-embedded" state pairs and output the hyperparameters (μ, σ) of the variational reward posterior distribution $q(R_c|\mathcal{D})$, through the *Reward Encoder* q_ϕ;
4. *(Blue arrows in Figure 4)* Decode the "c-embedded" state pairs and output the expected value of the reward for each action at a given state $Q_R(s, a)$, through the *Expected Reward Decoder* Q_θ. Then Deduce the optimal policy $\pi^\star$.

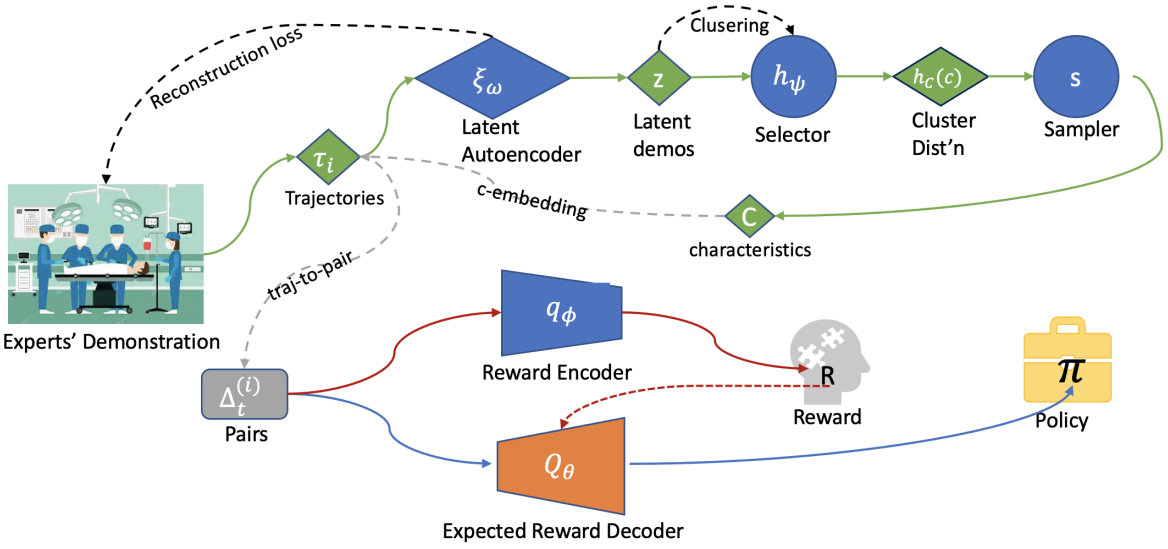


Figure 4: Workflow of CAVIRL

3.4 The Hierarchical Stochastic Model Behind the Machine

The CAVIRL model in this paper is based on 3 layers of hierarchy:

$$\begin{aligned} \left(s_t^{(i)}, a_t^{(i)}\right) \mid R, c &\overset{\perp}{\sim} G \\ R \mid c &\sim \text{Normal}(\mu, \sigma) \\ c &\sim h_{\mathcal{C}}(c) \end{aligned} \tag{17}$$

where G is some distribution from which the demonstrations are generated.

While the ultimate goal is to target the posterior distribution of the optimal action $a^\star_{R_c}$ at each state s given the induced reward R_c (i.e., the optimal policy $\pi^\star_{R_c}(s)$), we can prove by Bayes Rule that it is sufficient to target the joint posterior distribution of the optimal state-action pairs conditioned on the reward (Equation 18). Further, as the induced reward functions R_c are solely differed by the characteristics $c \in \mathcal{C}$, we have the last 2 hierarchies in Equation 17 by the Normality assumption of $R|c$ and the Categorical assumption of c, which gives Equation 19.

$$p(\pi^\star_{R_c}(s)) = p(a^\star_{R_c}|s) \overset{\text{Bayes Rule}}{\propto} p(a^\star, s|R_c) \tag{18}$$

$$\propto p(a^\star, s|R, c)p(R|c)h_{\mathcal{C}}(c) \tag{19}$$

Proof. Proof of Equation 18:

$$p(a_{R_c} \mid s) = p(a|s, R_c) = \frac{p(a, s, R_c)}{p(s, R_c)} = \frac{p(a, s, R_c)}{p(s)p(R_c)} = \frac{p(a, s|R_c)}{p(s)} \propto p(a, s|R_c)$$

Proof of Equation 19:

$$p(a, s|R_c) = \frac{p(a, s, R_c)}{p(R_c)} = \frac{p(a, s, R|c)h_{\mathcal{C}}(c)}{p(R|c)h_{\mathcal{C}}(c)} \propto p(a^\star, s|R, c)p(R|c)h_{\mathcal{C}}(c)$$

□

The graphical structure of the model (Figure 5) illustrates the cross-dependent relationship among all the variables, where β (the confidence parameter defined in Equation 10) is fixed as a known constant in this paper for simplicity.

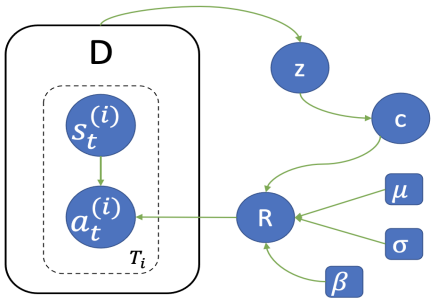


Figure 5: Graphical Structure of the CAVIRL Model

3.5 The Objective (Loss) Functions

We introduce 3 loss functions, one for pretraining and the other two for model training.

3.5.1 Pretraining: Reconstruction Loss for AutoEncoder

We utilized the **M**eans **S**quared **E**rror (MSE) between the reconstructed demonstrations $\mathcal{D}_{\text{Recon}}$ and the observed demonstrations $\mathcal{D}$ as a reconstruction loss to train the *Latent Autoencoder* ξ_ω (as in Step 0 in Section 3.3). The pretraining algorithm in given in the Appendix (Algorithm 1).

$$\mathcal{L}_{\text{Recon}}(\omega) = \text{MSE}(\mathcal{D}, \mathcal{D}_{\text{Recon}}) = \|\mathcal{D} - \mathcal{D}_{\text{Recon}}\|^2 = \sum_{i=1}^{N} \left\| \tau_i - \tau_i^{(\text{Recon})} \right\|^2 \tag{20}$$

3.5.2 Training: 2 Evidence Lower BOunds (ELBO)

The model training steps (step 1-4 in Section 3.3) involve 3 neural networks — the *Selector* h_ψ, the *Reward Encoder* q_ϕ, and the *Expected Reward Decoder* Q_θ. We separately train them in 2 phases:

> **Phase 1** Train the *Reward Encoder-Decoder*(q_ϕ, Q_θ) with the *Selector* h_ψ fixed;
>
> **Phase 2** Train the *Selector* h_ψ with the *Reward Encoder-Decoder* (q_ϕ, Q_θ) fixed.

The information we would like to extract from the demonstration data are the underlying reward R and the characteristics c of the demonstrators. So, the target distribution is the joint probability of the reward and the characteristics conditioned on the demonstration, i.e., $p(R, c \mid D)$. Therefore, according to the hierarchical model (Equation 17), the Variational Inference (Blei et al., 2017) minimizes the KL-Divergence in Equation 21, which can be derived to a maximization of the full ELBO function in Equation 22.

$$\min_{\phi,\psi} D_{KL}(h_\psi(c)q_\phi(R \mid c) \| p(R, c \mid D)) \tag{21}$$

$$\mathcal{L} = \mathbb{E}_{h_\psi \times q_\phi}[\log p(D|R, c)] - D_{KL}(h_\psi(c) \| p(c)) - \mathbb{E}_{h_\psi}[D_{KL}(q_\phi(R|c) \| p(R|c))] \tag{22}$$

Proof. The derivation applies the similar idea as did in (Blei et al., 2017).

$$\begin{aligned}
&D_{KL}\left(h_\psi(c)q_\phi(R|c) \| p(R, c|D)\right) \\
&= \mathbb{E}_{h_\psi \times q_\phi}[\log h_\psi(c) + \log q_\phi(R|c)] - \mathbb{E}_{h_\psi \times q_\phi}[\log p(R, c|D)] \\
&= \mathbb{E}_{h_\psi \times q_\phi}[\log h_\psi(c) + \log q_\phi(R|c)] - \mathbb{E}_{h_\psi \times q_\phi}[\log p(D \mid R, c) + \log p(R|c) + \log p(c) - \log p(D)]
\end{aligned}$$

$$= \mathbb{E}_{h_\psi}[\log h_\psi(c) - \log p(c)] + \mathbb{E}_{h_\psi \times q_\phi}[\log q_\phi(R|c) - \log p(R|c)] - \mathbb{E}_{h_\psi \times q_\phi}[\log p(D|R, c)] + \log p(D)$$
$$= D_{KL}(h_\psi(c)\|p(c)) + \mathbb{E}_{h_\psi}[D_{KL}(q_\phi(R|c)\|p(R|c))] - \mathbb{E}_{h_\psi \times q_\phi}[\log p(D|R, c)] + \text{Constant}$$

where the constant term does not depend on h_ψ and q_ϕ. □

3.5.2.1 ELBO1 (for Phase 1)

ELBO1 omits the second term in Equation 22 as it only involves h_ψ, which is fixed in **Phase 1**. Phase 1 implicitly assumes characteristic labels as known from the *Selector* and *Sampler*, hence $h_\psi(c) = 1, \forall c$. Then, By Equation (9, 10, 14) and the neural network approximator Q_θ (Approximator 2) in Section 2.3, we derived ELBO1 as a function of (ϕ, θ) in Equation 23.

$$\mathcal{L}_v(\phi, \theta) = \mathbb{E}_{q_\phi} \underbrace{\left[\sum_{c\in C}\sum_{\tau\in\mathcal{D}_c}\sum_{(s_c,a)\in\tau} \log\left(\frac{\exp(Q_\theta(s_c, a))}{\sum_{a'\in\mathcal{A}}\exp(Q_\theta(s_c, a'))}\right)\right]}_{(\star)} - D_{KL}\left(q_\phi(R|c)\|p(R|c)\right) \tag{23}$$

where $\mathcal{D}_c$ is the subset of the demonstrations that is labeled with characteristic c and s_c is the "c-embedded" state.

The prior of reward R conditioned on characteristics c is given by a Normal distribution with zero mean (reflecting neutral belief on the reward), and some standard deviation as a function of characteristics (reflecting some priori knowledge of the characteristic influence on the demonstrators' behaviours).

$$p(R|c) \sim \text{Normal}(0, g(c)) \tag{24}$$

Proof. Derivation of Equation 23 is sufficient to show $\log p(\mathcal{D}|R, c) \simeq (\star)$.

$$\begin{aligned}
p(\mathcal{D}|R, c) &= \prod_{i=1}^{N} p(\tau_i|R, c), \text{ under assumption 1 in Section 1;} \\
&= \prod_{i=1}^{N}\prod_{t=1}^{T_i} p((s_t^{(i)}, a_t^{(i)})|R, c), \text{ by Factorization Property (Equation 9) and Equation 14;} \\
&= \prod_{i=1}^{N}\prod_{t=1}^{T_i} \frac{\exp(Q_{R,c}^{\pi_{\tau_i}}(s_t^{(i)}, a_t^{(i)}))}{\sum_{a'\in\mathcal{A}}\exp(Q_{R,c}^{\pi_{\tau_i}}(s_t^{(i)}, a'))}, \text{ by Boltzmann parameterization (Equation (10));} \\
&\simeq \prod_{i=1}^{N}\prod_{t=1}^{T_i} \frac{\exp(Q_\theta(s_t^{(i)}, a_t^{(i)}))}{\sum_{a'\in\mathcal{A}}\exp(Q_\theta(s_t^{(i)}, a'))}, \text{ by neural network Approximator 2;} \\
&= \prod_{c\in C}\prod_{\tau\in\mathcal{D}_c}\prod_{(s_c,a)\in\tau} \frac{\exp(Q_\theta(s_c, a))}{\sum_{a'\in\mathcal{A}}\exp(Q_\theta(s_c, a'))}
\end{aligned}$$

□

3.5.2.2 ELBO2 (for Phase 2)

ELBO2 involves all terms in Equation 22 and can be derived into Equation 25.

$$\begin{aligned}
\mathcal{L}_c(\psi) = &\sum_{\tau\in\mathcal{D}}\sum_{c\in C_\tau} h_\psi(c) \sum_{(s_c,a)\in\tau} \log\left(\frac{\exp(Q_\theta(s_c, a))}{\sum_{a'\in\mathcal{A}}\exp(Q_\theta(s_c, a'))}\right) \\
&- D_{KL}(h_\psi(c)\|p(c)) - \sum_{c\in C} h_\psi(c)[D_{KL}(q_\phi(R|c)\|p(R|c))]
\end{aligned} \tag{25}$$

where C_τ is the set of possible characteristic labels that can be assigned to a demonstration trajectory τ.

The prior of the characteristic labels $p(c)$ is set to be a flat distribution (e.g., discrete Uniform distribution as Equation 26), reflecting our neutral belief that a demonstrator is equally likely to be gifted with any characteristic.

$$p(c) \sim \text{Uniform}\left\{\frac{1}{|C|}, \cdots, \frac{1}{|C|}\right\} \tag{26}$$

3.5.2.3 Regularization of ELBO1

Recall that q_ϕ and Q_θ are defined separately. To make them consistent, we ideally want the *imitated reward* computed from Q_θ to be statistically significant under the reward posterior distribution computed from q_ϕ. By Equation (5, 6), the *imitated reward* of an action at a given state is given by the difference between the Q-value of the current state-action pair and the discounted Q-value of the next state-action pair, with a discounting factor γ.

$$\hat{R}(s,a) = Q_R^\tau(s,a) - \gamma Q_R^\tau(s',a') \tag{27}$$

Proof. Along a given trajectory τ, the expected reward of an action a at a given state s following the reward function R is given by

$$\begin{aligned} Q_R^\tau(s_t,a_t) &= R(s_t,a_t) + \gamma V_R^\tau(s_{t+1}), \text{ by Equation (6)} \\ &= R(s_t,a_t) + \gamma[R(s_{t+1},a_{t+1}) + \gamma V_R^\tau(s_{t+2})], \text{ by Equation (5)} \\ &= R(s_t,a_t) + \gamma Q_R^\tau(s_{t+1},a_{t+1}), \text{ by the induction hypothesis} \end{aligned}$$

for any time step t along the trajectory, □

Hence, we regularize ELBO 1 by constraining the expected negative log-likelihood of the imitated reward to be small.

$$\mathcal{L}_r = \mathbb{E}_{\pi,T}\left[-\log(q_\phi\left(Q_\theta(s,a) - \gamma Q_\theta(s',a')\right))\right] < \epsilon \quad \text{for some positive small number } \epsilon \tag{28}$$

3.5.2.4 Regularization of ELBO2

To promote sparse clustering assignments, we regularize ELBO 2 by constraining an Entropy loss to be small.

$$\mathcal{L}_e = \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}\left[-\sum_c h_\psi(c)\log h_\psi(c)\right] < \epsilon \quad \text{for some positive small number } \epsilon \tag{29}$$

3.5.2.5 Objective Functions for the 2 Training Phases

With 2 positive regularization coefficients ($\alpha 1, \alpha 2$), the objective functions are the ELBOs in conjunction with the corresponding regularization terms. The model training algorithm in given in the Appendix (Algorithm 2).

Phase 1 : maximize $\mathcal{L}_1 = \mathcal{L}_v - \alpha_1\mathcal{L}_r$, by Equation 23 and 28

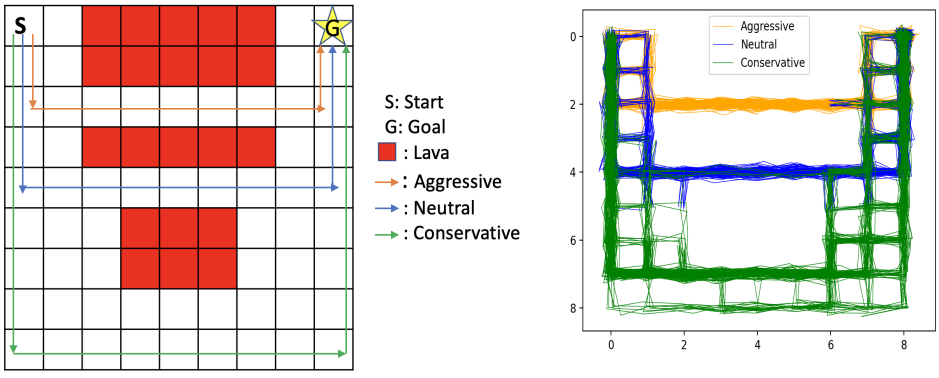
Phase 2 : maximize $\mathcal{L}_2 = \mathcal{L}_c - \alpha_2\mathcal{L}_e$, by Equation 25 and 29

4 Experiments

4.1 A Grid-World Environment and the Simulated Dataset

We designed a 9×9 grid-world environment as in Figure 6a. Players will learn a path from the starting point **S** to the goal **G** in the grid-world, where they receive a positive reward (+1). They should also learn to avoid the lava areas (red squares), as whenever they fall into the lava they receive a huge penalty (−10000) and game-over. Players are also penalized by the time they wasted on zigzagging in the grid-world, although the

penalization of each time step differs with respect to 3 characteristics — aggressive (−5), neutral (−1), and conservative (−0.1). As a result, players with the 3 characteristics follow 3 corresponding policies and ideally learn 3 different paths — orange, blue and green (as in Figure 6a), respectively.



(a) A Grid World Environment with 3 Types of Players **(b)** Simulated Dataset

Figure 6: Environment (a) and Dataset (b)

We simulated 300 demonstrations — 100 for each characteristic. Each trajectory contains 30 time steps, and is zero-padded if the demonstration terminates earlier, e.g., player reached **G** before 30 time steps. Each state along a trajectory contains the row and column of the player's position in the grid-world (hence 2-dimensional), and each action is 1-dimensional contain one of the available actions — "up", "down", "left", "right", or "stay" (i.e., no move). Hence the training demonstration data is [300, 30, 3], where the states data is [300, 30, 2] and the actions data is [300, 30, 1]. We also incorporated noise in the training dataset, by assigning a small weight to the non-optimal actions at each state for each policy. The simulated state data is shown in Figure 6b, which is roughly consistent to the 3 ideal paths in 6a.

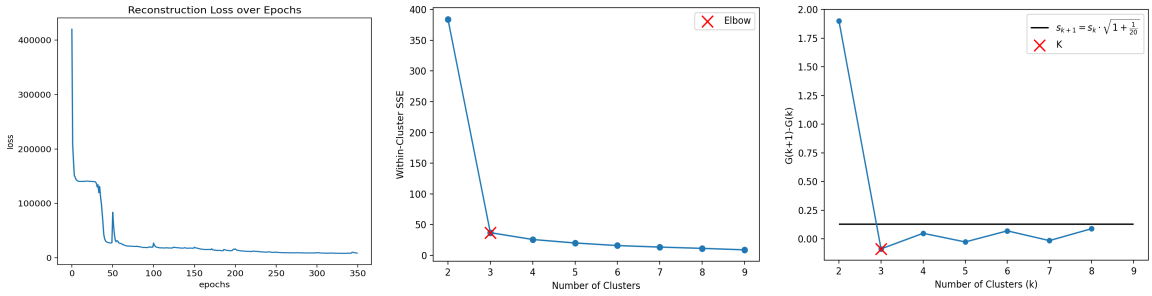
4.2 Pretraining

The *Latent Autoencoder* ξ_ω utilizes a stacked RNN with 5 hidden layers and tanh activation functions as the latent encoder, and it utilizes 4 feedforward layers with **E**xponential **L**inear **U**nit (ELU) activation functions (Equation 30 with $\alpha = 0.5$) as the latent decoder.

$$ELU(x) = \begin{cases} x, x \geq 0 \\ \alpha(e^x - 1), x < 0 \end{cases} \tag{30}$$

While each demonstration has 90(= 30 × 3) dimensions, we mapped them to a latent space with 32 dimensions, i.e., $z \in \mathcal{Z} \subseteq \mathbb{R}^{32}$. The MSE reconstruction loss (Equation 20) was minimized by an Adam optimizer and showed a clear convergence after 350 epochs (Figure 7a). We performed a search for the optimal number of clusters $K^\star$ from a set of candidates $\{2, 3, \cdots, 10\}$. The Elbow method (Figure 7b), Gap Statistics (Figure 7c), Silhouette's score and Davies-Bouldin index (Table 1) are all pointing toward $K^\star = 3$ as optimal, which is 100% consistent to the 3 characteristics we defined in the dataset. For illustration purpose, this was done on the first 2 principal components (PC) of the latent demonstrations z (which explain 75% of the total variation), rather than directly on z. In the space of the first 2 PCs, data points moderately assemble into 3 groups with respect to characteristics (Figure 8a). With $K^\star = 3$, the K-Means clustering labels (in Figure 8b) are moderately (though not perfectly) aligned with the true characteristics (in Figure 8a). Herein, the

model training steps (in Section 4.3) will take the K-Means clustering labels as the surrogate of the unknown characteristics.

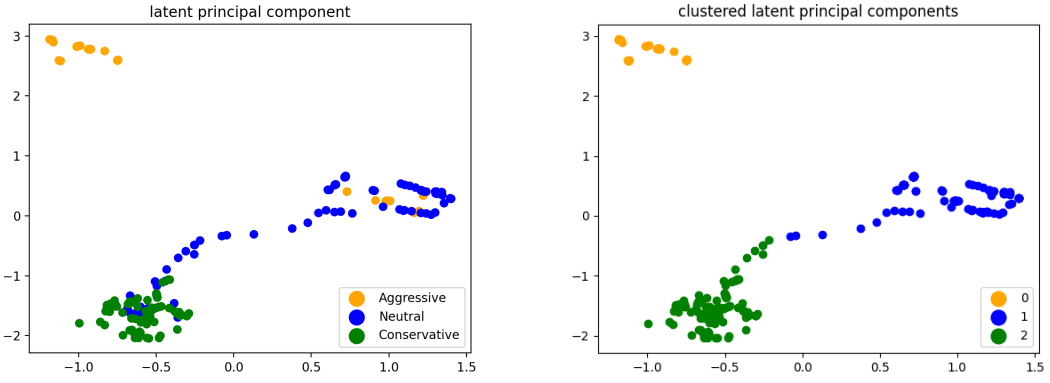


(a) Reconstruction Loss **(b)** Elbow Method **(c)** Gap Statistics

Figure 7: Pretraining: Convergence of Reconstruction Loss and the Identification of the Optimal Number of Clusters

Table 1: Separability Scores for K-Means over the Latent Autoencoder with Different Number of Clusters

k	Silhouette score	Calinski Harabasz	Davies Bouldin
2	0.66	425.77	0.40
3	0.82	3465.01	0.23
4	0.73	3498.21	0.44
5	0.62	3691.58	0.61
6	0.57	3773.75	0.63
7	0.58	3901.63	0.55
8	0.60	4113.85	0.54
9	0.56	4413.63	0.56



(a) by Characteristics **(b)** by K-Means Clustering Labels

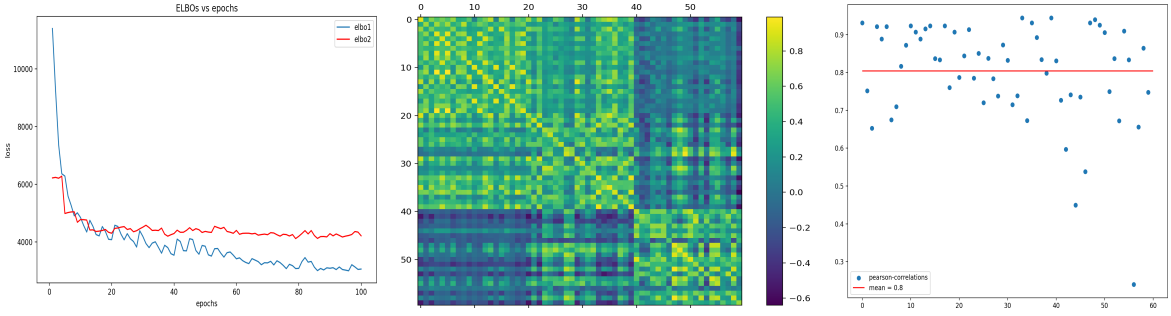
Figure 8: The First 2 Principal Components of the Latent Demonstrations z by Characteristics and by K-Means Clustering Labels

4.3 Model Training and Performance

The *Reward Encoder* q_ϕ was designed as a RNN with 2 hidden layers and tanh activation functions, connected to 3 feedforward layers with ELU($\alpha = 1$), under the *"state-only"* mode (Section 3.1.3). The *Expected Reward Decoder* Q_θ was designed as 5 feedforward layers with ReLU activation functions and a ELU($\alpha = 1$) output gate layer. The *Selector* h_ψ was designed as 5 feedforward layers with ELU($\alpha = 0.5$) activation functions and a Softmax output gate layer.

The model was trained in 2 phases (Section 3.5.2) alternately for 20 times, where each phase is trained with 4 epochs and 64 mini-batches in a single alternation. For simplicity, the regularization coefficients (α_1, α_2) and the discounting factor γ are all set to 1. Using 2 Adam optimizers with learning rates decay over epochs, the 2 loss functions converged after round 100 epochs (Figure 9a), giving roughly 80%-83.9% match in induced actions when we test the model on several independently and evenly simulated test datasets of 60 demonstrations (20 for each characteristics).

We computed the Row-wise Pearson Correlation Coefficients (de Vries, 1993) between the induced actions and the actions in a test dataset, which is a 60×60 matrix as the test set contains 60 demonstrations. The heat map (Figure 9b) shows high correlation coefficients around the diagonal, where the average correlation achieves as high as roughly 80% in 9c.



(a) Convergence of Loss Functions **(b)** Heatmap **(c)** Diagonal Correlation Coefficients

Figure 9: Model Training (a) and the Row-wise Pearson Correlation Coefficients Between the Induced Policy and the Test Dataset (b,c)

5 Future Work

At this stage, we have completely introduced the presented model with a toy example. We believe there are 3 potential improvements awaiting future exploitation.

1. *Robustness against dataset*. Determining the optimal number of clusters $K^\star$ requires extra amount of effort during the model pretraining step. However, this effort is not reproducible — whenever we change the dataset, we have to train a new *Latent Autoencoder* for identification of $K^\star$. On the other hand, the presented model heavily relies on the solidness of $K^\star$, as it cannot be fully defined without $K^\star$. Future effort should aim to either relief the model dependency of $K^\star$ by giving robust predictions even with a poor $K^\star$, or incorporate the identification of $K^\star$ to the training step of the model so that everything is learned "at one batch" without pretraining.
2. *Application of Attention structure*. In the presented model, we assumed conditional independence by the Local Markov Property (Equation 9), whereas long- and short-term auto-correlations might exist along a demonstration trajectory. The Attention structure (e.g., Transformer Model (Vaswani et al., 2017)) has been proved to be a handy technique to capture long- and short-term correlations and dependencies in sequential data, which can potentially be applied to the encoders of our model.

3. *Outperforming the demonstrators*. Our model implicitly assumes the observed demonstrations are done by "experts", hence learning their behaviours with "faith". But if the model observes both good and bad demonstrations, can it think out of the box? Future research could be done to outperform the demonstrators, via extrapolation (e.g., (Brown et al., 2019)) and exploration (e.g., ϵ–greedy policy and (Balakrishnan et al., 2020)).

While there are still steps before being perfect, the CAVIRL model is a good improvement of work in (Chan and van der Schaar, 2021), as it enhances the applicability from "single-reward" to "multi-reward" scenario. It has also proved to the readers that it can potentially capture the variations in behaviours among the demonstrators. Most importantly, it brings a new stream of methodologies for the problem of Multi-Reward Imitation Learning via Bayesian Inverse Reinforcement Learning, enlightened by Variational Bayesian Methods.

Acknowledgement: I would like to express a sincere appreciation to Professor Mihaela van der Schaar and Alex J. Chan, the authors of the paper, *Scalable Bayesian Inverse Reinforcement Learning*, for their kind and patient clarifications on their paper throughout my work towards this paper.

Conflict of Interests: As an independent and self-funded researcher, the author declares no conflict of interest.

References

Abbeel, P., and Ng, A. Y. (2004), Apprenticeship learning via inverse reinforcement learning. *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, ACM, New York, NY, USA, p. 1.

Balakrishnan, S., Nguyen, Q. P., Low, B. K. H., and Soh, H. (2020), "Efficient Exploration of Reward Functions in Inverse Reinforcement Learning via Bayesian Optimization,".

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017), Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518), 859–877. http://dx.doi.org/10.1080/01621459.2017.1285773

Brown, D. S., Goo, W., Nagarajan, P., and Niekum, S. (2019), Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations.

Caliński, T., and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1), 1–27. https://www.tandfonline.com/doi/abs/10.1080/03610927408827101

Chan, A. J., and van der Schaar, M. (2021). Scalable Bayesian Inverse Reinforcement Learning. *CoRR*, abs/2102.06483. https://arxiv.org/abs/2102.06483

Choi, J., and Kim, K.-e. (2011). MAP Inference for Bayesian Inverse Reinforcement Learning. *Advances in Neural Information Processing Systems*, eds. J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Vol. 24, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2011/file/3a15c7d0bbe60300a39f76f8a5ba6896-Paper.pdf

Choi, J., and Kim, K.-e. (2012), Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions. *Advances in Neural Information Processing Systems*, eds. F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Vol. 25, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/140f6969d5213fd0ece03148e62e461e-Paper.pdf

Davies, D. L., and Bouldin, D. W. (1979). A Cluster Separation Measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 224–227.

de Vries, H. (1993). The rowwise correlation between two proximity matrices and the partial rowwise correlation. *Psychometrika*, 58(1), 53–69.

Dimitrakakis, C., and Rothkopf, C. A. (2012). Bayesian Multitask Inverse Reinforcement Learning. *Recent Advances in Reinforcement Learning*, p. 273–284. http://dx.doi.org/10.1007/978-3-642-29946-9_27

Kullback, S., and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79–86. http://www.jstor.org/stable/2236703

Ng, A. Y., and Russell, S. (2000). Algorithms for Inverse Reinforcement Learning„ in *in Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, pp. 663–670.

Ramachandran, D., and Amir, E. (2007), Bayesian Inverse Reinforcement Learning. in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 2586–2591.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65. https://www.sciencedirect.com/science/article/pii/0377042787901257

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (Second). The MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, 18(4), 267–276.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2), 411–423. https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00293

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need.

Yu, L., Song, J., and Ermon, S. (2019). Multi-Agent Adversarial Inverse Reinforcement Learning.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. *Proc. AAAI*, pp. 1433–1438.

6 Appendix

6.1 Algorithms for Pretraining and Model Training

Algorithm 1: Pre-Training Algorithm of CAVIRL

Data Input: 3-dimensional tensors,
- State trajectories: $[N, T, |\mathcal{S}|]$, where T is the maximum trajectory length.
- Action trajectories: $[N, T, 1]$

Initialization:
(i) learning rates (η_0), (ii) mini-batch size (b_0), (iii) latent dimension ($|\mathcal{Z}|$), (iv) action dimension ($|\mathcal{A}|$), (v) Latent Autoencoder: ξ_ω, (vi) maximum number of clusters $K_{\max}$

Training the Latent Autoencoder:
while *not converged (with decaying learning rate starting from η_0)* **do**
  Sample mini-batched demonstrations $\mathcal{D}_{\text{mb}}$ of size b_0 from the data.
  Compute the reconstructed demonstration: $\mathcal{D}_{\text{mb,Recon}} = \xi_\omega(\mathcal{D}_{\text{mb}})$.
  Compute the reconstruction loss: $\mathcal{L}_{\text{Recon}}(\omega, \mathcal{D}) = E\left[\frac{N}{b_0}\|\mathcal{D}_{\text{mb}} - \mathcal{D}_{\text{mb,Recon}}\|^2\right]$ by Eq. (20)
  Compute and update the model parameters of ξ_ω: $\omega \leftarrow \omega + \eta_0 \nabla_\omega \mathcal{L}_{\text{Recon}}(\omega, \mathcal{D})$
end
Identification of the Optimal Number of Cluster $K^\star$:
Get latent demonstrations z from the latent autoencoder.
for $k \in \{2, \cdots, K_{\max}\}$ **do**
  K-Means(k) clustering over z.
  Test the quality of clustering via separability scores, gap statistic and elbow method.
end
Return: Optimal number of characteristics $K^\star$ and the predicted characteristic labels c via K-Means.

Algorithm 2: Training Algorithm of CAVIRL

Data Input: via "c-embedding" and trajectories-to-pairs (Section 3.2.2)

- "c-embedded" State pairs: $[N \times (T-1), 2, |\mathcal{S}|+1]$;
- Action trajectories: $[N \times (T-1), 2, 1]$
- Number of characteristics ($|\mathcal{C}| = K^{\star}$) and the characteristics labels (c) from Algorithm 1.

Initialization of Hyper Parameters:

(i) learning rates (η_1, η_2), (ii) mini-batch sizes (b_1, b_2), (iii) number of epochs (e_1, e_2), (iv) regularization coefficients (α_1, α_2), (v) reward encoder q_ϕ, (vi) expected reward decoder Q_θ, (vii) selector h_ψ

Model Training: (Alternating between Phase 1 and Phase 2)

while *not converged (with decaying learning rates starting from* (η_1, η_2)*)* **do**

  Phase 1: fix h_ψ

  for $i = 1 \ldots e_1$ **do**

    Sample mini-batched demonstrations $\mathcal{D}_{\text{mb1}}$ of size b_1 from the data.

    Compute objective 1: $\mathcal{L}_1'(\phi, \theta, \mathcal{D}) = E\left[\frac{N}{b_1}\left(\mathcal{L}_v(\phi, \theta, \mathcal{D}_{\text{mb1}}) - \alpha_1 \mathcal{L}_r\right)\right]$ by Section 3.5.2.5

    Compute and update the new parameters of (q_ϕ, Q_θ): $(\phi, \theta) \leftarrow (\phi, \theta) + \eta_1 \nabla_{\phi', \theta'} \mathcal{L}_1'(\phi, \theta, \mathcal{D})$

  end

  Phase 2: fix (q_ϕ, Q_θ)

  for $i = 1 \ldots e_2$ **do**

    Sample mini-batched demonstrations $\mathcal{D}_{\text{mb2}}$ of size b_2 from the data.

    Compute objective function 2: $\mathcal{L}_2'(\psi, \mathcal{D}) = E\left[\frac{N}{b_2}\left(\mathcal{L}_c(\psi, \mathcal{D}_{\text{mb2}}) - \alpha_2 \mathcal{L}_e\right)\right]$ by Section 3.5.2.5

    Compute and update the new parameters of h_ψ: $\psi \leftarrow \psi + \eta_2 \nabla_{\psi'} \mathcal{L}_2'(\psi, \mathcal{D})$

  end

end

Return: The distribution of the imitated reward $\hat{q}(\hat{R})$, the expected reward $\hat{Q}_{\hat{R}}(s, a)$ learned for each state-action pair, and the implied policy $\pi^{\star}$.