A PROBABILISTIC PERSPECTIVE ON REINFORCEMENT LEARNING VIA SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning via Supervised Learning (RvS) only uses supervised techniques to learn desirable behaviors from large datasets. RvS has attracted much attention lately due to its simplicity and ability to leverage diverse trajectories. We introduce Density to Decision (D2D), a new framework, to unify a myriad of RvS algorithms. The Density to Decision framework formulates RvS as a two-step process: i) density estimation via supervised learning and ii) decision making via exponential tilting of the density. Using our framework, we categorise popular RvS algorithms and show how they are different by the design choices in their implementation. We then introduce a novel algorithm, Implicit RvS, leveraging powerful density estimation techniques that can easily be tilted to produce desirable behaviors. We compare the performance of a suite of RvS algorithms on the D4RL benchmark. Finally, we highlight the limitations of current RvS algorithms in comparison with traditional RL ones.

1 INTRODUCTION

Leveraging large amounts of off-policy data is essential for using Reinforcement Learning (RL) in real-world settings (Levine et al., 2020). Recently, a straightforward and efficient approach to use large datasets has been to treat RL as a Supervised Learning problem, namely RL via Supervised Learning (RvS) (Emmons et al., 2021). The simplicity of RvS comes from bypassing the difficulties of Temporal Difference (TD) learning (Sutton, 1988), to instead use the observed return to learn good policies. Furthermore, the efficiency of RvS comes from the ability to learn from sub-optimal trajectories present in the dataset.

The RvS algorithms usually convert the RL problem into behavior cloning where the samples are weighted by, filtered by, or conditioned on the observed Monte Carlo (MC) return. It is however unclear what objective these RvS algorithms are maximising. We introduce the Density Estimation to Decision Making (D2D) framework to unify the RvS algorithms. This novel framework allows us to formulate RvS as a two steps process. The first step consists of performing density estimation over the dataset, which can be done via supervised learning. The second step consists of sampling from the exponential tilt of the learned density to obtain samples that are both likely and have high observed MC returns.

Using the D2D framework, we can derive recent algorithms such as Return Conditioned Policy (Kumar et al., 2019), Upside-Down RL (Srivastava et al., 2019), Trajectory Transformer (Janner et al., 2021), Decision Transformer (Chen et al., 2021), and others. We also introduce implicit RL via Supervised learning, a new implicit algorithm leveraging advances in density modeling. We benchmark these different algorithms using the D4RL framework (Fu et al., 2020). Finally, we highlight the limitations of current RvS algorithms in comparison with traditional RL algorithms.

2 A PROBABILISTIC INFERENCE PERSPECTIVE ON RL VIA SUPERVISED LEARNING

Reinforcement Learning via Supervised Learning algorithms are trained on large datasets collected by agents of diverse expertise. The promise of RvS algorithms is in leveraging large amounts of sub-optimal and near-optimal data to achieve performance beyond what would be attainable by only

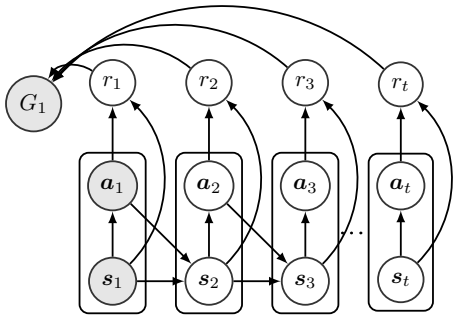


Figure 1: Having access to the Monte Carlo return $G_1 = \sum_{t'=1}^{T} \gamma^{(t'-1)} r_{t'}$ provides information about the future trajectory and, since it is observed (shaded), it can easily be used with supervised learning and density estimation algorithms.

using less but near-optimal data. Current RvS algorithms use the dataset to learn a good policy by weighting the transitions as a function of the return, by conditioning the action distribution on the return, or by learning a forward model of the environment to estimate the return of an action. These different approaches do not look related at first glance, but consolidating them into an unified framework would improve our understanding of the connections between these methods and allows us to design better algorithms.

We introduce the Density to Decision (D2D) framework to unify the RvS methods listed above. This framework consists of two steps: i) learning a density model using the dataset, and ii) using the learned density for decision making. Using this framework, we demonstrate that these RvS methods approximate the same density but make different design choices in their approximation. Finally, we propose an algorithm using a different design choice better suited for density estimation and efficient decision making.

2.1 From Density Estimation to Decision Making

Density Estimation. The dataset is composed of the triplets: states, action and MC return $(\mathbf{s}_t, \mathbf{a}_t, G_t)$. Their graphical structure and dependencies are depicted in Figure 1. The MC return (Sutton & Barto, 2018) can be understood as providing hindsight information about the outcome of the trajectory (Furuta et al., 2021). The first step consists of learning a density over the triplets which can easily be estimated via supervised learning methods. Specifically, the objective can be defined as the negative log-likelihood of the data:

$$\mathbb{E}_{\{\mathbf{s}_t, \mathbf{a}_t, G_t\} \sim \mathcal{D}} \Big[-\log p_{\boldsymbol{\theta}}(\mathbf{a}_t, G_t \mid \mathbf{s}_t) \Big]. \tag{1}$$

Decision Making. We are interested in making decisions better or as good as the best ones observed in the dataset. Thus, we cannot simply sample from the learned density as this would result in decisions present in the dataset ignoring the quality of these decisions. To improve decision making we have to direct the sampling towards the best decisions in the dataset. A natural approach is to use the Exponential Tilt (Asmussen & Glynn, 2007; O'Donoghue et al., 2020) of the MC return, which is defined as:

$$p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta) = C \, p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t) \exp(\eta^{-1} G_t), \tag{2}$$

where $\mathbf{s}_t$ is observed, and C is a normalizing constant. As we decrease $\eta \to 0$, the density becomes concentrated around regions that are potentially less likely in the dataset but where the MC return is large. We recover the dataset density by increasing $\eta \to \infty$ independently of the MC return. Making η an important hyper-parameter as we want samples that are both likely in the dataset and have high MC return.

Connection to Control-as-Inference. We note the similarity between D2D decision-making objective and the Control-as-Inference target density. In both cases, the exponential function is used to turn a trajectory return into a positive number which can then be treated as a probability density.

	Density Estimation ($\mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, G_t) \sim \mathcal{D}}$)	Decision Making ($\mathbf{a}, G \sim p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta)$)
RWR	$\exp(\eta^{-1} G_t) p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t)$	$p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t)$
RCP	$p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t, G_t) p_{\boldsymbol{\theta}}(G_t \mid \mathbf{s}_t)$	$p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t, G) p_{\boldsymbol{\theta}}(G \mid \mathbf{s}_t) \exp(\eta^{-1} G)$
RBC	$p_{\boldsymbol{\theta}}(G_t \mid \mathbf{s}_t, \mathbf{a}_t) p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t)$	$p(G \mid \mathbf{s}_t, \mathbf{a}) p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t) \exp(\eta^{-1} G)$
IRvS (Ours)	$p_{\boldsymbol{\theta}}(\mathbf{a}_t, G_t \mid \mathbf{s}_t)$	$p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t) \exp(\eta^{-1} G)$

Table 1: Comparison of different policy training and action selection.

Thus, trajectories with larger return have higher density making them more likely to be sampled. The main difference with Control-as-Inference is in the training process. Control-as-Inference introduces observed "optimality variables" (Levine, 2018). These optimality variables are used to define a backward message. The trajectory density is decomposed by time step using the backward message which must be approximated using (soft) TD-learning. Thus making training more challenging than D2D which can be performed by simple supervised learning.

Algorithm 1 Density to Decision Framework

Input: Dataset $\mathcal{D} = \{\mathbf{s}_t, \mathbf{a}_t, G_t\}_{t=1}^{T}$
Input: Learning rate κ
Input: Temperature η
 # Density Estimation
while not converged **do**
  Sample minibatch $\mathcal{B} \sim \mathcal{D}$
  # Compute loss
  $\mathcal{L}(\boldsymbol{\theta}) \leftarrow \sum_{(\mathbf{s}_t, \mathbf{a}_t, G_t) \in \mathcal{B}} - \log p_{\boldsymbol{\theta}}(\mathbf{a}_t, G_t \mid \mathbf{s}_t)$
  Update parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \kappa \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
end while
 # Decision Making
$\mathbf{s}_0 \sim \mu(\mathbf{s}_0)$
$t \leftarrow 0$
while episode not done **do**
  $\mathbf{a}_t, G_t \sim p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t) \exp(\eta^{-1} G)$
  $\mathbf{s}_{t+1}, r \sim p_{\text{env}}(\mathbf{s}, r \mid \mathbf{s}_t, \mathbf{a}_t)$
  $t \leftarrow t + 1$
end while

2.2 A Probabilistic Inference Perspective on RvS Algorithms

Using our D2D framework, we can categorize multiple RvS algorithms as different design choices to approximate the same density. An overview of the different methods an design choices are provided in Table 1.

Return Conditioned Policy (RCP). One can model the return conditional policy $p(\mathbf{a} \mid \mathbf{s}, G)$ and a state conditional distribution over return $p(G \mid \mathbf{s})$ using supervised learning. Conditioning on the return G allows the model to leverage trajectories with a sub-optimal observed return, potentially resulting in more efficient learning. An additional advantage of this approach is that a simple Gaussian approximation to $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}, G)$ can capture a multimodal action distribution $p(\mathbf{a} \mid \mathbf{s})$, which can be obtained by marginalising the return G, i.e., $p(\mathbf{a} \mid \mathbf{s}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(G \mid \mathbf{s})}[p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}, G)]$. To recover the richness of the training distribution, one would need to learn a multimodal distribution over MC return $p_{\boldsymbol{\theta}}(G \mid \mathbf{s})$, but in practice only a subset of the training behaviors is desired: the best behavior. The learning is done over the following factorisation:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\{\mathbf{s}_t, \mathbf{a}_t, G_t\} \sim \mathcal{D}} \Big[- p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t, G_t) p_{\boldsymbol{\theta}}(G_t \mid \mathbf{s}_t) \Big], \tag{3}$$

where $\boldsymbol{\theta}$ denotes the density that is learned, and the decision making distribution is obtained:

$$p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta) = p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t, G) p(G \mid \mathbf{s}_t) \exp(\eta^{-1} G). \tag{4}$$

Kumar et al. (2019) and Srivastava et al. (2019) estimate the maximum return and then condition on it at the beginning of the episode. For simplicity, it is also possible to ignore η and simply condition the policy on the maximum observed return in the dataset: $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t, G_{\text{max}})$ as is done in Decision Transformer (DT) (Chen et al., 2021).

Reweighted Behavior Cloning (RBC). One can model a state-conditional action distribution $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s})$ and a state-action conditional return distribution $p_{\boldsymbol{\theta}}(G \mid \mathbf{s}, \mathbf{a})$. Contrary to RCP which can capture a multimodal action distribution $p(\mathbf{a} \mid \mathbf{s})$ by combining simpler return conditional distributions, in turn RBC must directly learn a multimodal action distribution $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s})$. The decision making distribution can be approximated via reweighting the action samples. Density estimation and decision making can be respectively done as:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\{\mathbf{s}_t, \mathbf{a}_t, G_t\} \sim \mathcal{D}} \Big[- p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t) p_{\boldsymbol{\theta}}(G_t \mid \mathbf{s}_t, \mathbf{a}_t) \Big] \tag{5}$$

$$p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta) \approx C \sum_i \exp(\eta^{-1} G_i) \delta(a_i, G_i), \tag{6}$$

where C is a normalising constant $G \sim p(G \mid \mathbf{a}, \mathbf{s}_t)$ and $\mathbf{a} \sim p(\mathbf{a} \mid \mathbf{s}_t)$. For example, Trajectory Transformer (TT) (Janner et al., 2021) performs decision making by sampling multiple actions from $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}_t)$ and evaluates $p_{\boldsymbol{\theta}}(G \mid \mathbf{s}_t, \mathbf{a})$ for each action using model roll-outs sampled from $p_{\boldsymbol{\theta}}(\mathbf{s}_{t+1}, r_t \mid \mathbf{s}_t, \mathbf{a}_t)$. Again, η can be ignored in this case by taking the action with the highest observed return. The ancestral sampling procedure used to select action in TT does not scale well for large action spaces and long planning horizons.

Return Weighted Regression (RWR). It is possible to approximate the target density by bypassing the estimation of the return altogether. A policy can be learned using importance weights to bias the policy toward actions with a higher observed return. This has the advantage of being simple and computationally efficient. Specifically, one can directly amortize the distribution using importance weights proportional to the exponential of the observed MC return:

$$\mathcal{L}(\boldsymbol{\theta}) = D_{\text{KL}} \Big(\exp(\eta^{-1} G) p(G, \mathbf{s}, \mathbf{a}) \tag{7}$$

$$midp_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}) p(\cdot) \Big)$$

$$= \mathbb{E}_{\{\mathbf{s}_t, \mathbf{a}_t, G_t\} \sim \mathcal{D}} \Big[- \exp(\eta^{-1} G_t) \log p_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \mathbf{s}_t) + c \Big]. \tag{8}$$

where c does not depend on $\boldsymbol{\theta}$. Action selection can be performed efficiently using a forward pass $p_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s})$. We note the similarity with reward weighted regression (Peters & Schaal, 2007). Alternatively, the weights could also be an indicator variable $I_{\{G > g\}}$ which is 1 when the observed MC return is higher than a return threshold g and 0 otherwise.

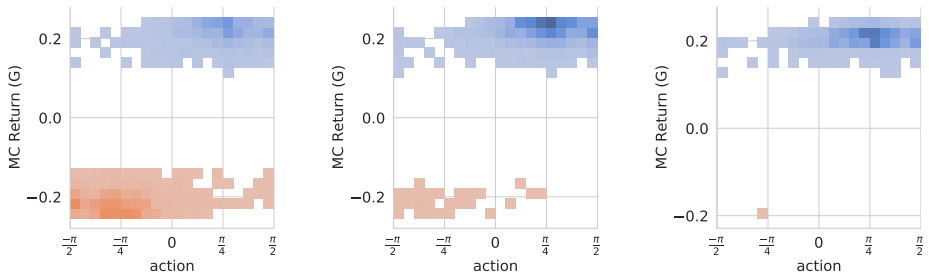
2.3 Implicit RvS

Using the D2D framework to investigate current RvS algorithms highlights the design choices that are currently made for the density estimation and decision making phases. We notice that no method directly model the joint distribution over the action and the return. Modeling the joint presents certain difficulties, but could potentially provides certain advantages. We now introduce a novel implicit method to model the joint distribution and that can be use to perform decision making in an efficient manner.

Implicit RL via Supervised Learning (IRvS). Using the D2D framework, we introduce a novel RvS algorithm. As opposed to the previous methods that factorize the joint distribution into different conditional distributions, it is possible to model the joint distribution directly. Modeling the joint has the advantages of using both the action distribution and the future return distribution signals to train a better model. It also does not require ancestral sampling to produce and evaluate an action as sampling can be done at the same time. The training loss function and the decision making distribution are respectively given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\{\mathbf{s}_t, \mathbf{a}_t, G_t\} \sim \mathcal{D}} \Big[- p_{\boldsymbol{\theta}}(\mathbf{a}_t, G_t \mid \mathbf{s}_t) \Big], \tag{9}$$

$$p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta) = p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t) \exp(\eta^{-1} G). \tag{10}$$



(a) $p(\mathbf{a}, G \mid \mathbf{s}_0)$ (b) $p(\mathbf{a}, G \mid \mathbf{s}_0)\exp(\frac{G}{0.1})$ (c) $p(\mathbf{a}, G \mid \mathbf{s}_0)\exp(\frac{G}{0.05})$

Figure 2: In a), we examine the dataset joint distribution over action (x-axis) and MC return (y-axis) given $\mathbf{s}_0$, where the action is an angle $\in [-\pi/2, \pi/2]$. In b) and c), we observe the exponential tilt in G of the initial density for different temperatures η. As we decrease the temperature, the density concentrate on actions with a more significant MC return.

Implementation In order to model the joint distribution over action and return we leverages advances in Energy Based Models (EBMs) (LeCun et al., 2006). EBMs are a promising class of generative models that can easily model discontinuity and their effectiveness for behavior cloning has been demonstrated in Florence et al. (2021). Specifically, the energy function is defined as:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))}{Z_{\boldsymbol{\theta}}}, \tag{11}$$

where data $\mathbf{x} \in \mathcal{X}$, $E_{\boldsymbol{\theta}}(\mathbf{x})$ is the parametrized energy function and $Z_{\boldsymbol{\theta}} = \int_{\mathbf{x}} \exp(E_{\boldsymbol{\theta}}(\mathbf{x}))$ is the normalising constant. Training the density model can be performed using the InfoNCE loss function defined as:

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{i=1}^{N} -\log(\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}_i \mid \mathbf{x}, \{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg.}}})) \text{ and } \tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}_i \mid \mathbf{x}, \{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg.}}}) = \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i)}}{e^{-E_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i)} + \sum_{j=1}^{N_{\text{neg.}}} e^{-E_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{y}_i^j)}}.$$

where $\{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg.}}}$ is a set of negative counter-examples. Negative sampling can be performed efficiently by minimizing $\mathbf{a}, G = \arg\min_{\mathbf{a},G} p_{\boldsymbol{\theta}}(\mathbf{a}, G \mid \mathbf{s}_t; \eta)$ using stochastic gradient Langevin dynamics (Welling & Teh, 2011).

3 Experiments

3.1 Didactic Example

We present a simple didactic experiment to communicate better the advantages of modeling the joint distribution over the state, action, and MC return $p(\mathbf{a}, G \mid \mathbf{s})$. The agent starts the episode at the point $(0, 0)$ with a randomly selected target and must choose an angle $\alpha \in [-\pi/2, \pi/2]$ (the half-circle) as an action. Reaching one of the targets results in a positive reward while the other one results in a negative reward. The angle is then converted to a 2d action as $\mathbf{a} = (\Delta_x, \Delta_y) = (\cos(\alpha), \sin(\alpha))$. The agent deterministically choose the angle to minimize the distance with the selected target. A small amount of Gaussian noise is then added to the action $\mathbf{a}$ to make the policy stochastic. There is no learning in this experiment. Graphical depiction of the environment and of random trajectories can be seen in Figure 3 in Appendix A.1.

In Figure 2a, we examine the joint distribution of the angle α (x-axis) and the MC return (y-axis) conditioned on the initial state $\mathbf{s}_0 = (0, 0)$. The distribution colors represent the (unobserved) target on which the behavior policy was conditioned. We can observe that the distribution is bimodal in both the MC return and the angle distribution. The stochasticity of the discounted MC return comes only from the random noise added to the policy as some policies take longer to reach the target. For each target conditioned policy, we can observe a mode for the action at around $\pi/4$ ($-\pi/4$) for the blue (orange) policy representing an angle of $45°$ ($-45°$)

Given the observed angle and return for $\mathbf{s}_0$ in the dataset in Figure 2a, we are interested in generating a distribution over the best action. Since the environment dynamics are fixed and the stochasticity comes from the policy alone, we can use the exponentially tilted density $p(\mathbf{a}, \mathbf{s}, G)\exp(\eta^{-1}G)$ with respect to G to generate a distribution over best actions. In Figures 2b and 2c, we can observe that as we decrease the temperature, the density over the maximum return increases and so does the density over the angle $\alpha = \pi/4$.

We observe that even without access to the target on which the policy was conditioned, modeling the joint density including the MC return G allows us to recover the distribution over the desired policy. Similarly, in offline RL, we do not observe the expertise level of the policies that collected the data. We can nevertheless use the observed return as guidance to learn desirable behaviors.

3.2 Continuous Control Dataset

To better understand the importance of the different design choices in RvS, we compare the algorithms on the D4RL benchmark (Fu et al., 2020). The D4RL benchmark provides a suite of tasks and datasets to evaluate offline RL agents (Levine et al., 2020). We compare the different algorithms on three continuous locomotion tasks with diverse datasets.

Algorithm 1 outlines the training and evaluation procedure. We train the density estimation algorithms for 100k gradient steps and then evaluate them on 100 episodes (of 1000 steps each). We repeat this procedure for three seeds. As in Emmons et al. (2021) we define G as the average future reward. Additionally, we normalize the states and G to lie in $[-1, 1]$.

We focus on the Gym Locomotion v2 tasks, consisting of the HalfCheetah, Hopper, and Walker2D datasets from the D4RL offline RL benchmark (Fu et al., 2020). These datasets are collected by agents with medium and expert levels of expertise in solving the respective tasks. The difficulty lies in leveraging sub-optimal and near-optimal trajectories to learn a policy. In Table 2 we show the baseline comparisons for the medium-expert dataset for the previously mentioned tasks. The values reported are the average return and standard deviation for 3 training initialisation seeds over 100 episodes each. We can see that IRvS performs on par with the best among the other methods for the Hopper and Walker tasks.

Environment Method	HalfCheetah	Hopper	Walker2d
RBC	7467 +/- 1054	1508 +/- 27	4156 +/- 226
RCP	7169 +/- 1380	1328 +/- 111	3984 +/- 107
RWR	11006 +/- 167	1593 +/- 519	4994 +/- 17
IRvS (Ours)	5593 +/- 944	1537 +/- 27	4768 +/- 149

Table 2: Baseline comparisons on D4RL MuJoCo gym based tasks using the Medium-Expert dataset. The results displayed are the return average over 100 episodes after 100k gradient steps.

4 Discussion

In this section, we highlight the differences between D2D and traditional Offline RL algorithms and discuss the advantages of both approaches. We first compare the supervised density estimation procedure used by D2D algorithms as opposed to the TD learning approach used by Offline RL algorithms to evaluate actions. We then compare the Exponential Tilt approach of D2D algorithms as opposed to the maximization of a learned Q value by Offline RL algorithms to perform action selection.

The first difference to note is the action evaluation procedure. RvS uses the observed Monte Carlo return rather than a parametrized Q value to learn a policy. On one hand, Q values can be challenging to learn when function approximation, off-policy, and bootstrapping are used as it can diverge (Van Hasselt et al., 2018), while the observed MC return provides an easy signal to use. On the other hand, Q-values learned via TD learning (Sutton, 1988) perform *trajectory stitching* thus a good action in a bad trajectory can be evaluated independently of the trajectory outcome and be

recognised as a good action. While in D2D, a good action in a bad trajectory will only be evaluated based on the trajectory outcome. A potential middle ground would be to use TD learning to perform on-policy evaluation as it is stable and provides good performances (Goo & Niekum, 2022; Brandfonbrener et al., 2021).

The second main difference between D2D and Offline RL algorithms is the action selection strategy. For a given state, D2D algorithms limit action selection candidates to the actions that have been observed for that state in the dataset. The agent might be optimistic for state-action pair that have not been observed in the dataset, and selecting unobserved actions would potentially result in worst performance than limiting action selection to observed actions. Yet, limiting action selection could potentially limit the asymptotic performance of the agent. D2D algorithms can explicitly control the risk-reward tradeoff via the temperature η. For small η, the agent might select action with high, but less likely, MC return. As we increase the temperature, the agent will seek less risk. A similar strategy is used by popular offline RL algorithms. For example, by using explicit density estimation to constraint the action selection to the actions observed in the dataset (Wu et al., 2022), training the Q-value network to be conservative for unobserved actions (Kumar et al., 2020) or by constraining the policy using behavior cloning regularization (Fujimoto & Gu, 2021).

5 Conclusion

In this work, we introduce the Density Estimation to Decision Making framework which unifies commonly used RvS Algorithms. The 2 steps process brings clarity on the training and action selection of these algorithms, and also provides guidance on designing novel algorithms. Using our framework we introduce IRvS which directly targets the joint density distribution over the action and return instead of the commonly used arbitrary density factorization. We show that IRvS is competitive with the state-of-the-art RvS algorithms on the D4RL Mujoco locomotion tasks. Finally, we discuss the trade-off of using RvS over traditional Offline RL algorithms.

Reinforcement Learning via Supervised Learning algorithms have been shown to be competitive with Offline RL algorithms for trajectory optimization. An interesting next step would be to compare their performance, generalisation capabilities and robustness on more difficult tasks.

References

Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

Wonjoon Goo and Scott Niekum. You only evaluate once: a simple baseline algorithm for offline rl. In *Conference on Robot Learning*, pp. 1543–1553. PMLR, 2022.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Brendan O'Donoghue, Ian Osband, and Catalin Ionescu. Making sense of reinforcement learning and probabilistic inference. *arXiv preprint arXiv:2001.00805*, 2020.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning*, 2007.

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, 2011.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *arXiv preprint arXiv:2202.06239*, 2022.

A APPENDIX

A.1 TOY EXPERIMENT

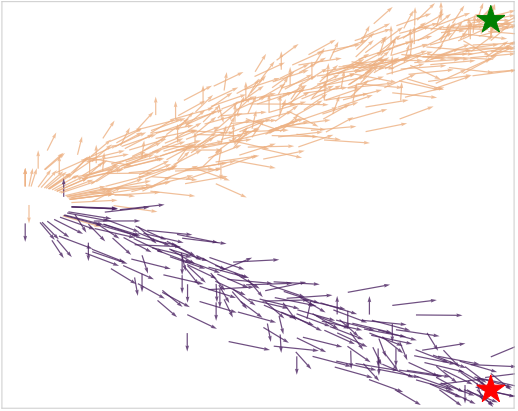


Figure 3: Example of trajectories in the toy task.