Laplace DQN: Deep RL Algorithm for Temporally Flexible Planning

Abel Rassat
arassat@ethz.ch

Alpay Ozkan
aozkan@ethz.ch

Jose Lavariega
jlavariega@ethz.ch

Abstract

This paper explores a Distributional Reinforcement Learning (DRL) algorithm specifically the Laplace Code, previously introduced to learn the temporal evolution of immediate rewards through a biologically plausible algorithm. In order to further scale the use of this algorithm beyond tabular settings we implemented the Laplace Code with Deep Q-Networks (DQN) and compared its performance to popular DRL algorithms like C51 and Quantile Regression DQN (QR-DQN). Importantly, the distributions learnt by the Laplace Code enable to immediately adapt the agent's policy to be optimal for a smaller time horizon. To this end an Inverse Laplace approximation is applied to the learnt distribution. By experimenting with this transformation we uncovered the artifacts it generates and proposed methods to overcome these. With this work, we come closer to using the power of the Laplace representation in temporally dynamic real-world environments.

1 Introduction & Motivation

Despite successes in game playing, robotic control, and financial modeling, traditional reinforcement learning (RL) methods struggle to represent the inherent randomness of real-world environments. Classic RL focuses on learning the expected reward, which limits performance when facing complexities and uncertainties that challenge the robustness of the learned value function.

In practical scenarios, understanding the full distribution of possible outcomes provides a more informative perspective for decision-making. For example, healthcare treatment plans must account for various possible patient responses, not just the average outcome. In robotic control, the value function must be derived from a gigantic state-action space, which may be a multi-modal distribution.

Distributional Reinforcement Learning (DRL) aims to learn the distributions of the value function instead of their expectations. Notable advantages of this approach are more well-behaved optimization, reduced state aliasing, and providing an additional prediction the agent can rely on concerning the uncertainty of its returns [1]. Once the value distributions of state-action pairs have been learned, one can shift its mass towards higher or lower values leading to a stable method of incentivizing the agent to be more risk-seeking or risk-averse, respectively. Such successful applications using these techniques have been in robotics with locomotion tasks [15] or with discovering new algorithms for matrix multiplications [4]. Particularly in the former example, having access to a value distribution can be a valuable asset as an interpretable risk metric to avoid damaging costly equipment.

Learning an entire distribution is challenging due to the computation of an often intractable normalizing integral. To address this, methods by [1] and [2] respectively use categorical or quantile representations for the distribution. However, these approaches are limited as they ignore the temporal dynamics of rewards i.e., "Under a given policy what is the distribution of rewards at every future time step?". While the Laplace code representation [16] can capture the temporal evolution of immediate rewards, it has only been tested on simple discrete Markov Processes. This motivates the need for an

Foundations of Reinforcement Learning, FS2024. Department of Computer Science, ETH Zürich.
Code available at https://github.com/alpayozkan/Laplace$_{D}QN$

approximate RL approach using neural networks to represent the temporal reward evolution in more complex environments in which choice of actions leads to different state transitions.

In this paper, our contributions are:

1. The first implementation, to the best of our knowledge, of the Laplace code for discrete action spaces, with DQNs. We call our algorithm Laplace DQN.
2. Benchmark Laplace DQN, against C51[1] and QR-DQN[2] on simple gym environments.
3. Observe adaptation in time horizon on a gym environment and identifying artifacts of the Inverse Laplace approximation while proposing a method to prevent them

2 Background & Related Work

In a Markov decision process (S, A, P, γ, R) with finite state-action space (S, A), transition kernel $p : S \times A \rightarrow \mathcal{P}(S)$ discount factor γ and reward distributions $R(S, A)$, if an agent takes an action A_t at time t, then it undergoes a state transition and receives a reward according to: $S_{t+1} \approx p(\cdot|S_t, A_t)$; $R_t \sim \mathcal{R}(S_t, A_t)$ respectively. A Markov *policy* $\pi : S \rightarrow \mathcal{P}(A)$ is evaluated under the Q-function of that policy. $Q^{\pi}(x, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = x, A_0 = a]$. The optimal policy induces the maximum Q-function i.e., state-action values are larger or equal to any other Q-function induced by another policy.

The task of approximating the underlying Q-function from a prior initial distribution of values for each state-action pair is done in a deterministic scheme through the *Bellman Equation*:
$Q^{\pi}(x, a) = \mathbb{E}_{\pi}[R_0 + \gamma Q^{\pi}(S_1, A_1)|S_0 = x, A_0 = a]$. This function gives a single *expected* return. This forms the basis of all value-based RL [11]. Naturally, reward structures with stochastic nature are not captured by a single expected return, and so the *distributional Bellman equation* is introduced, which computes Q-function values at the level of probability distributions [1] [10].

$$\eta_{\pi}(x, a) = \mathbb{E}_{\pi}[(f_{R_o,\gamma})\#\eta_{\pi}(S_1, A_1)|S_0 = x, A_0 = a] \tag{1}$$

Where $\eta_{\pi}(x, a) \in \mathcal{P}(\mathbb{R})$ is the distribution of the returns and $g\#\mu$ is the push-forward of the measure μ through the function g as defined in [12]. Stating in terms of the return gives the familiar form of:

$$Z^{\pi}(x, a) \overset{D}{=} R(x, a) + \gamma Z^{\pi}(x', a). \tag{2}$$

Note that since the space of Q-function distributions $\mathcal{P}(\mathbb{R})$ is infinite-dimensional, the field relies on parametric approximations to make Distributional RL tractable. Three parametrization methods, with the two first serving as our benchmarks, are detailed below and visualized in figure 1. All three parametrizations have been successfully implemented in previous works as Distributional RL setups.

2.1 Distributional RL Networks & Parametrizations

a) Categorical - C51
The categorical representation aims to learn the probabilities of each evenly spaced interval (delimited by so-called atoms) along the axis of possible values of a given state-action pair.

$$Q_{\theta}(x, a) \approx \sum_{k=1}^{\kappa} p_k \delta_k, \text{ s.t. } \sum_{k}^{\kappa} p_k = 1; p_k \geq 0 \forall k \tag{3}$$

Where δ_k refers to the Dirac distribution at location k. The Categorical update is a $\sqrt{\gamma}$ - contraction of the supremum-Cramer distance, and so repeated Categorical updates converge to a unique limit point, independent of initial prior distributions. For more details, consult [1], [12].

The common Categorical-DQN algorithm in use is called C51, for the usage of 51 atoms in the original implementation in [1].

b) Quantile - DQN
Quantile representation aims to learn the positions of the quantiles associated with the value distribution. This is a parametric form of the return distributions, where the $\theta(x, a)$ are learnable parameters.

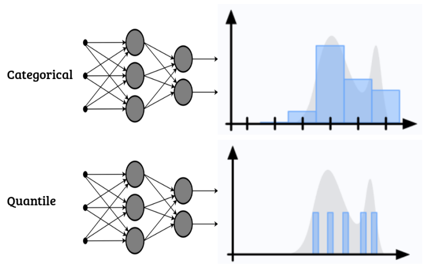


Figure 1: Parametrization benchmark types, visualized: Categorical and Quantile

$$Q_\theta(x,a) \approx \frac{1}{N}\sum_{i=1}^{N}\delta_{\theta_i(x,a)} \text{ s.t. } \delta_{\theta_i} \approx \hat{\tau}_i = \frac{\tau_{i-1}+\tau_i}{2} \tag{4}$$

Distributions are restricted to this parametric family. Each atom location at the current state-action pair is updated by following the gradient of the QR loss. For further details refer to [2] [3]. Along with Deep Quantile networks, this method is known as QR-DQN. In addition, Expectile Code is very similar to Quantile which we provide details in appendix.A.

2.2 Laplace Encoding

While all of the previous parametrizations of the distribution over the Q-function work effectively for reward distributions that remain static in time, these do not reflect reward distributions that change in time. As an example, in robotics and autonomous systems, the dynamically changing environment introduces changes and often penalizes state-action pairs that would be desirable under a static rewards structure. More concretely, during an earthquake, a stable configuration for the robot to keep standing up is different than under a stable day. Currently optimization works rely on adding time as an extension to the state space, and learning works rely on multiple aggregate policies for when the rewards structure changes. The Laplace representation captures such temporal changes in the rewards structure, which means it could be particularly suited to this type of temporally-dynamic problems.

The Laplace Code [16] is a DRL algorithm that has a particular representational power as it captures immediate temporal rewards starting from a given state. Akin to what is learned in categorical learning, sensitivities are defined at atom locations along the axis of possible reward of a given state. A value function is learned for each sensitivity and various discount factors γ equidistant in the space $\frac{1}{\log(\gamma)}$. Applying an approximation of the inverse Laplace transform (or Z transform for discrete-time), transfers the state values from the so-called γ-space to the τ-space. It is in this new space that the immediate temporal rewards can be read allowing to reduce the time horizon on which the state-values are based. We can then easily recompute the expected values for each state for this new time horizon and a fixed discount factor leading to temporal flexibility. This feature of the Laplace code may prove to be helpful in situations where a shorter time constraint must not lead to losing time learning a new optimal value function. Instead, the agent can immediately adapt to this new constraint and still act optimally.

To further clarify, the Laplace code can be seen in RL as enabling TD and Q-learning under varying temporal discounts. Under conditions of traditional Q-learning, the Q function approximation converges to equation 5

$$Q^\pi(s,a) = \mathbb{E}[\sum_{\tau=0}^{\infty}\gamma^\tau R_{t+\tau}|s_t,a_t] = \sum_{\tau=0}^{\infty}\gamma^\tau \mathbb{E}[R_{t+\tau}|s_t,a_t] \tag{5}$$

This shows the critical property: $Q^\pi(s,a)$ is a unilateral Z-transform of the expectation, with a real-valued parameter γ. The Z-transform is invertible, in the limit of infinite γ's, we can recover $\mathbb{E}[R_{t+\tau}|s_t,a_t]_{\tau=0}^{\infty}$ from $Q_\gamma(s_t,a_t)_{\gamma\in(0,1)}$.

$Z^{-1}\{Q(s,a)\} = Z^{-1}\{\mathbb{E}[R_{t+\tau}|s,a]\}_{\tau=0}^{\infty}$. Note that the Z-transform is nothing more than the discrete-time equivalent of the Laplace transform. Henceforth the insight generalizes to the continuous time limit. This shows that $Q_\gamma(s_t, a_t)$ is the Laplace transform of $\mathbb{E}[R_{t+\tau}|a_t, r_t]$ with parameter $-\log\gamma$. Therefore:

$$\mathscr{L}^{-1}\{Q(s,a)\} = \mathscr{L}^{-1}\{\mathbb{E}[R_{t+\tau}|s,a]\}_{\tau>0} \tag{6}$$

Or, the Z-transform in continuous time. Hence, in Q-learning, the convergence not only encodes the expected sum of discounted rewards, as is done in traditional RL, but also the expected reward of all future timesteps, although this signal lies in a different space, very much analogous to the frequency and temporal spaces of the Fourier transform. In [17] and in this work we refer to these spaces as the γ-space and the τ-space, such that applying $mathcalL^{-1}$ or Z^{-1} to the γ-space yields the τ-space.

$$\mathbb{E}[\sum_{\tau=0}^{\infty}\bar{\gamma}^\tau r_{t+\tau}|s_t, a_t] = \sum_{\tau=0}^{\infty}\bar{\gamma}^\tau \sum_r rP(r_{t+\tau} = r|s_t, a_t) \tag{7}$$

[17] gives further detail on making this transformation tractable as neurally-plausible operations, and then applies the Laplace encoding in tabular experiments.

As a side note, the Laplace code is authored by computational neuroscientists who were interested in DRL because of its bio-plausibility [7] [18] and aimed to uncover a local algorithm that the authors contrast with the expectile code [13] that was championed as an algorithm found in the hippocampus. This DRL algorithm is similar to quantile regression but requires non-local computations for the value function to converge.

2.3 Multi-timescale and Temporal RL

Recent works have been conducted studying certain features of the Laplace Code. Learning the value function for different discount factors can be seen as learning for multiple timescales in biological reinforcement learning [8]. In this work, the authors showed that depending on a state's proximity to the end of a trajectory or episode, an accurate state-action value is more rapidly learned depending on the discount factor i.e., there is a clear advantage of having different time scales when the Q value has not converged yet i.e., might boost training speed but remains to be tested rigorously. This first benefit of learning over multiple timescales for the same neural network, named auxiliary learning, is thought to enhance the representation within the hidden layers leading to an improved value function associated with the highest discount factor. i.e., transfer the more accurate representation of the value function for lower discount factors to enhance the representation of the value function that will be used for acting. As done conventionally, the value function associated with the highest discount factor (0.99) is chosen for the behavioral policy. The second benefit of multi-timescale RL is the boosted ability to predict the timing of reward which is made obvious when applying the inverse Laplace or Z transform.

Elsewhere, the Laplace Code allows the recovery of a perhaps more phenomenologically and biologically plausible representation learned model of the environment for planning being the distribution of the temporal evolution of immediate rewards. This approach known as temporal reinforcement learning [6] no longer gives the primacy Temporal Difference algorithms aiming to learn state values for optimal planning.

3 Laplace DQN

Our implementation of the Laplace Code with DQN uses a multiplexer comprised of three neural networks (cf. section 3.1) that takes a state and a discount factor as input and outputs the values for each action (given the input state and discount factor) for each sensitivity. The optimal action is then selected by computing a proxy of the true state-action value function (equal in the limit of infinitely dense sensitivities) and selecting the action with the maximum value, i.e. for a Q-function learned with a fixed discount factor γ,

$$Q_\gamma(s_t, a_t) = \sum_{\tau=0}^{\infty}\gamma^\tau \mathbb{E}[r_{t+\tau}|s_t, a_t] \tag{8}$$

we approximate the expectation term as

$$\begin{aligned} &\mathbb{E}[r_{t+\tau}|s_t, a_t] \approx P(\theta_1 < r_{t+\tau} < \theta_2)\,\hat{\theta}_1 + \cdots + P(\theta_{H-1} < r_{t+\tau} < \theta_H)\,\hat{\theta}_H \\ &\Rightarrow Q_\gamma(s_t, a_t) \approx \sum_{h=1}^{H-1}\left(Q_{h,\gamma}(s_t, a_t) - Q_{h+1,\gamma}(s_t, a_t)\right)\hat{\theta}_h \end{aligned} \tag{9}$$

Where the Q-function for a given sensitivity h and a discount factor γ is defined as $Q_{h,\gamma}(s_t, a_t) = \sum_{\tau=0}^{\infty} \gamma^{\tau} P(r_{t+\tau} > \theta_h | s_t, a_t)$, assuming the reward sensitivity activation to be the Heaviside function. The atom locations are θ_i, for $i \in [H]$, with H the number of atoms and $\hat{\theta}_i = \frac{\theta_i + \theta_{i+1}}{2}$. We select the optimal action by then taking the action that maximizes the approximated Q-value.

In our implementation, the reward activations are kept as sigmoid functions with the slope as a hyperparameter and sensitivities are linearly spaced between a minimum and maximum reward value (r_{min}, r_{max}).

To enforce that for sensitivities that are located for higher reward values have smaller associated Q_h than for sensitivities located for lower reward values for any given state-action pair (by definition) we apply an accumulation to values before they output by our model. A comparison of performance on Cartpole with another method to impose this property and without imposing it can be found in Fig 7. Section B.3 of the appendix.

Note that the C51, QR-DQN and Laplace DQN implementations used in our experiments rely on the standard DQN architecture [9] but it is likely that using dueling architectures [19] and prioritized replay [14] would lead to better performance [5].

3.1 Inverse Laplace Approximation

We calculated the inverse Laplace transformation of the gamma space $Q_{h,\gamma_i}(s, a)$ to obtain tau space which gave us information about the probability distribution of the rewards. For that we followed the SVD based approximation as outlined in the appendix of [16]. So we calculate the inverse of F matrix with SVD decomposition as illustrated in (10), to get to the tau-space. This operation is repeated for different actions and sensitivities, we parallelized the implementation accordingly to get an efficient solution as number of sensitivities is a high number. The iterative approach was so slow and not practical for our experiments.

$$\underbrace{\begin{bmatrix} Q_{h,\gamma_1}(s,a) \\ Q_{h,\gamma_2}(s,a) \\ \vdots \\ Q_{h,\gamma_N}(s,a) \end{bmatrix}}_{\gamma \text{ space}} = \underbrace{\begin{bmatrix} \gamma_1^0 & \gamma_1^1 & \gamma_1^2 & \cdots & \gamma_1^T \\ \gamma_2^0 & \gamma_2^1 & \gamma_2^2 & \cdots & \gamma_2^T \\ \gamma_3^0 & \gamma_3^1 & \gamma_3^2 & \cdots & \gamma_3^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_N^0 & \gamma_N^1 & \gamma_N^2 & \cdots & \gamma_N^T \end{bmatrix}}_{\text{F}} \underbrace{\begin{bmatrix} P(r_0 > \theta_h \mid s,a) \\ P(r_1 > \theta_h \mid s,a) \\ \vdots \\ P(r_T > \theta_h \mid s,a) \end{bmatrix}}_{\tau \text{ space}} \tag{10}$$

3.2 Multiplexer Extension

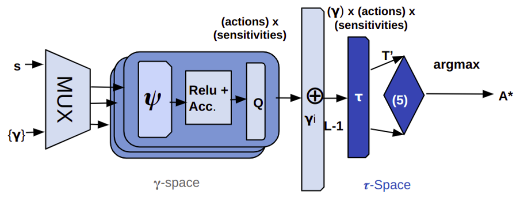

s
{γ}
MUX
ψ
Relu + Acc.
Q
(actions) x (sensitivities)
γ-space
⊕
γi
L-1
(γ) x (actions) x (sensitivities)
τ
T'
(5)
argmax
A*
τ-Space

Figure 2: Laplace DQN Multiplexer

We had problems training Laplace DQN agent for most of the tasks. We had a single network mapping inputs state and a discount factor or gamma to output Q values. However, we had very unstable results and could not reach the optimal best reward even with high gamma values. The reason we hypothesized was the conflict between low and high gamma values being trained with a common network. Low gamma and high gamma values lead to very different Q values and this misalignment created instability during training. As a result, as illustrated in Figure-2, we proposed a simple solution to mitigate this conflict by introducing a multiplexer to separate the networks to learn different Q values given gammas in different ranges. We empirically chose 3 intervals for gamma ranges: low: [0, 0.5], medium:[0.5, 0.75], high:[0.75, 1]. Hence, given an input with gamma, the multiplexer forwards to the model with the right gamma range from which we obtain the Q values.

This stabilized training and we managed to get performance comparable to quantile and categorical agents. In order to still benefit from auxiliary learning [8] we choose not use too many modules for different gamma ranges to have have reasonably sized intervals.

3.3 Training

We train Laplace DQN for 5 linearly spaced discount factors over each gamma range of the multiplexer. The goal is to learn the optimal Q-function for these discount factors and have the model interpolate for the Q-function of discount factors it wasn't trained with. This led to training our model with 15 discount factors each with its environment and replay buffer as different discount factors lead to different optimal policies. We loop over this list of discount factors during training and the loss from for Laplace TD-learning (cf. appendix equation 13 and 14) is then backpropagated for each discount factor through its corresponding network.

4 Experiments

The goal of our experiments was to show that we can train a Laplace agent combined with DQN and we have comparable scores to other distributional RL algorithms. Also, we aimed to show the effect on Laplace DQN when time-horizon changes to validate the theory empirically.

We compared our Laplace DQN with multiplexer to C51 and QR-DQN in different Reinforcement Learning environments in gymnasium: Cartpole-v1, LunarLander-v2, and Acrobot-v1 as illustrated in Figure-3. We have obtained comparable results in these environments for Laplace DQN with the policy of the highest discount factor (0.99), the same discount factor as for the two other methods. We applied convolution to smoothen the curves as they were noisy but were improving with episodes. We fixed the seeds for all of the experiments and for Laplace DQN the seed was even more critical as we trained agents with significantly varying average rewards.

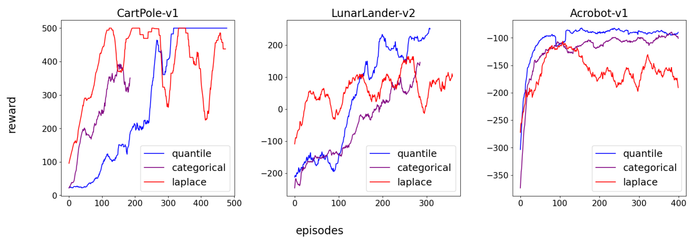


Figure 3: Performance comparison of Quantile (QR-DQN), Categorical (C51), and Laplace DQN across various episodes on three tasks: CartPole-v1, LunarLander-v2, and Acrobot-v1. The y-axis represents the reward obtained, and the x-axis represents the number of episodes.

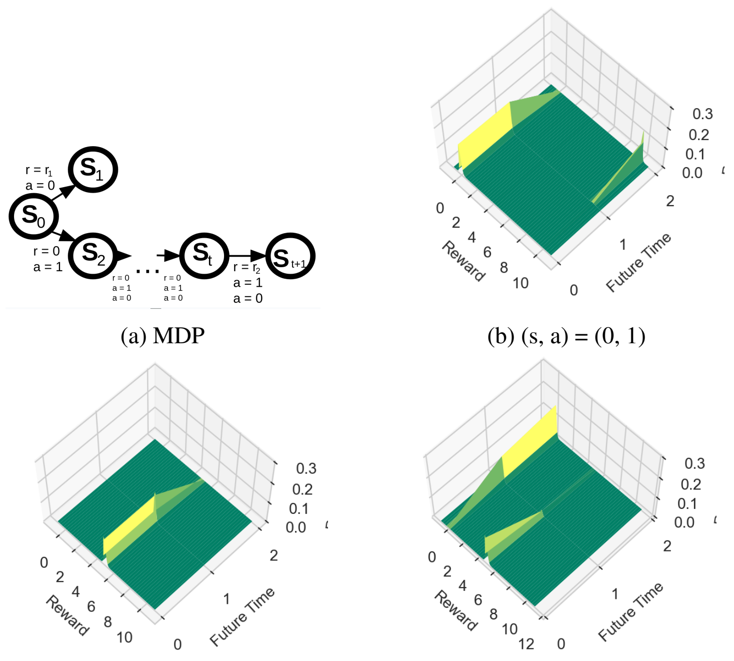
Once our agent was trained we conducted a change in time-horizon experiment where we explicitly stopped the environment after the time-horizon defined was reached. We chose the LunarLander environment for these tests as the optimal policy changes with different time-horizons. For LunarLander, given a shorter time constraint for an episode, we anticipate the agent to take more direct and risky landings to obtain rewards more rapidly whereas it would act with greater precautions if it disposed of a larger number of steps. In contrast, for other environments like Cartpole and Acrobot, the optimal policy does not change with time-horizon since the same reward is perceived at each time-step until the pole passes over a certain angle or until the end of the pole passes above a horizontal line respectively. In Table-1, we compared average reward among different models when time-horizon is changed. For Laplace DQN's different time-horizon changes, we tried different regularisation factor values alpha that are used for the calculation inverse Laplace SVD-based approximation as mentioned in [16].

No significant benefit from the change in time horizon for the LunarLander experiment led us to attempt better understanding this feature of the Laplace Code on a simpler MDP environment presented in panel a) of Figure 4. The MDP can lead to two possible state trajectory for the agent. Either the agent takes action 0 at state 0 reaches state 1 while perceiving a reward $r1$ and the episode terminates, or action 1 is taken in state 0 and the agent moves from state 2 to state $t + 1$ where a reward $r2$ is perceived only for the last transition before the episode terminates.

Method	T=50	T=100	T=200	T=T$_{max}$
Categorical [c51]	26.51	67.28	121.47	161.63
Quantile	16.53	48.58	65.65	224.71
Laplace DQN	2.14	52.32	57.42	188.45

Table 1: Comparison of Average Rewards for Categorical (C51), Quantile (QR-DQN), and Laplace on the LunarLander-v2 Environment across Various Time Horizons.

Figure 4 (b-c) show the tau-space for the state-action pairs $(s, a) = (0, 1)$ and $(s, a) = (0, 0)$ respectively. Note that a tabular learning of the Q-function was used instead of our Laplace DQN in this example to obtain a gamma-space with fewer noise leading to cleaner representation of the tau-space as was tested. For a visualisation of the gamma-space in a simple Markov Process computed by our Laplace DQN and its corresponding tau-space, please refer to Fig. 5 in section 1 of the Appendix. In panel (d) is represented the tau space for the $(s, a) = (0, 0)$ for the same MDP but with added state transitions for the shorter branch all perceiving reward 0 after $r1$ is perceived in state 1. In our experiment $T = 3$, $r1 = 5$ and $r2 = 10$ which yields for a discount factor of 0.99 an optimal policy that chooses action 1 in state 0.



(a) MDP

(b) (s, a) = (0, 1)

(c) (s, a) = (0, 0)

(d) (s, a) = (0, 0) with subsequent 0 reward states

Figure 4: Tau-space for tabularly learned Q-function of a simple MDP

An extensive hyperparameter search was conducted to find the values for all three experiments. For further consultation, consult the appendix C.

5 Discussion

Although the results for the change in time horizon experiment in LunarLander did not yield the anticipated higher performance for Laplace DQN that should have immediately adapted to the optimal policy for the new time horizon (Table 1), Figure 4 furnishes a valuable hint as to what may be going wrong. In panel (c), the desired distribution for the tau-space would be a signal at transition 0 and no signal and transition 1 as the episode must have terminated. Instead, we observe a tau-space where the last reward perceived by the agent is repeated which tricks the agent towards choosing action 0 after no change in time horizon i.e., simply going to the tau-space and back. Naturally, the value of action 1 is higher for a discount factor close enough to 1 and should have been chosen instead.

To solve this unintended effect of erroneous tau-space representations for trajectories terminating before the maximum time limit of the episode, we created an MDP with auxiliary states (cf. panel d) such that regardless of action 1 or 0 being chosen at state 0, the episode would finish after 3 steps. Where the episode should have terminated in the original MDP, the agent received 0 rewards for subsequent steps. Performing action selection from the tau-space by applying equation 10, we only obtained the expected behavior of sticking to action 1 for no change in time horizon for the MDP with auxiliary states.

This toy example suggest that a similar nuisance is taking place for LunarLander. Precisely, if states led to the episode finishing before the time horizon Laplace DQN was trained with, then in the tau space, the reward perceived before that terminal state might be sustained. These artifacts mess with the tau-space of state encountered earlier which ultimately creates false signals and poor decision making when applying equation 10 in the tau space (regardless of change in time horizon). To test this idea we could then have trained Laplace DQN on LunarLander where we would send the agent to a valid point of the state space i.e., an auxiliary state (but that is never visited in practice) and have it stay there perceiving reward 0 at every time step until the time limit is reached. We could do this anytime the episode terminates before the time limit. We would then repeat the change in time horizon experiments for Laplace DQN and check if we obtain better results this way.

6 Conclusion

One promise of the Laplace Code is to hedge against state aliasing to an even greater extent than other DRL methods. Having a distribution for the value of the state already allows to distinguish many states that share the same expected value. Laplace DQN takes this a step further by using the same networks to learn the Q-function over a range of discount factors which effectively informs on the temporal evolution of immediate rewards. This representational power of the Laplace Code could prove to be particularly useful in partially observable environments where the problem of state aliasing is prevalent.

To this end, future works could explore filtering methods for the gamma-space in a noisy setting to ensure a reasonable representation in the tau-space as highlighted in our experiments. Researching methods to leverage the information in the tau-space also appears to be a fruitful path e.g., applying a transformation to the reward distributions to bias the behavior of agent akin to what was done with quadrupedal robots with their learned value distributions to make them more risk-averse or risk-seeking in real-time [15].

As the SVD-based Inverse Laplace approximation is very noise-sensitive, a neural network approximation of the Inverse Laplace may be a direction worth exploring. So the inverse Laplace transformation can be modeled with a neural network to map gamma space to tau space which we can use to marginalize with sensitivities for action selection as outlined in [16]. We can use this framework for action selection during training with time-horizon changes.

Lastly, the Laplace Code paper introduces a memory extension that allows to recover the gamma-space from the tau-space i.e., the value distributions after having manipulated the tau-space such as a change in time horizon. Without the memory extension, this feature is only possible for the restricted class environments with reward non-zero perceived only at absorbing states. As a result, even after a change in time horizon, the agent can still benefit from the strengths of the DRL approach instead of applying equation 10 for action selection.

7 Acknowledgements

We would like to thank Prof. Niao He and the course staff for providing an engaging, challenging and mentally rewarding course. We would further like to extend our thanks to Dr. Yardim for aiding in running an efficient ship, and we would lastly like to thank Leo Widmer for being our advisor during this project, and for providing the encouragement to continue.

References

[1] Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning, July 2017.

[2] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional Reinforcement Learning With Quantile Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v32i1.11791.

[3] Patricia Daxbacher. Deep distributional reinforcement learning: From dqn to c51 and qr-dqn.

[4] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, October 2022. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-022-05172-4.

[5] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning, October 2017.

[6] Marc W. Howard, Zahra G. Esfahani, Bao Le, and Per B. Sederberg. Foundations of a temporal RL, February 2023.

[7] Adam S. Lowet, Qiao Zheng, Sara Matias, Jan Drugowitsch, and Naoshige Uchida. Distributional Reinforcement Learning in the Brain. *Trends in Neurosciences*, 43(12):980–997, December 2020. ISSN 01662236. doi: 10.1016/j.tins.2020.09.004.

[8] Paul Masset, Pablo Tano, HyungGoo R. Kim, Athar N. Malik, Alexandre Pouget, and Naoshige Uchida. Multi-timescale reinforcement learning in the brain, November 2023.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013.

[10] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki TANAKA. Nonparametric return distribution approximation for reinforcement learning. In *International Conference on Machine Learning*, 2010. URL `https://api.semanticscholar.org/CorpusID:976202`.

[11] Andrew G. Barto Richard S. Sutton. *Reinforcement Learning: An Introduction*. The MIT Press, 2015.

[12] Mark Rowland, Marc G. Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning, 2018. URL `https://arxiv.org/abs/1802.08163`.

[13] Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G. Bellemare, and Will Dabney. Statistics and Samples in Distributional Reinforcement Learning, February 2019.

[14] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay, February 2016.

[15] Lukas Schneider, Jonas Frey, Takahiro Miki, and Marco Hutter. Learning risk-aware quadrupedal locomotion using distributional reinforcement learning, 2023.

[16] Pablo Tano, Peter Dayan, and Alexandre Pouget. A Local Temporal Difference Code for Distributional Reinforcement Learning.

[17] Pablo Tano, Peter Dayan, and Alexandre Pouget. A local temporal difference code for distributional reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages

13662–13673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9dd16e049becf4d5087c90a83fea403b-Paper.pdf.

[18] Timothy H Muller, James L. Butler, S. Veselic, Bruno Miranda, J. Wallis, Peter Dayan, Timothy E Behrens, Z. Kurth-Nelson, and S. Kennerley. Distributional reinforcement learning in prefrontal cortex. *Nature Neuroscience*, 2024. doi: 10.1038/s41593-023-01535-w.

[19] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning, April 2016.

A Expectile Code

The expectile representation displays the expectiles. Just like the quantile regression loss is an asymmetric version of the absolute value loss, so is the expectile regression loss also an asymmetric representation of the squared loss. However expectile networks learn from a *statistic*, not a sample.

$$Q_\theta(x,a) = \frac{1}{K}\sum_{i=1}^{N} \delta_{\theta(x)} \tag{11}$$

The distribution is imputed from target expectiles by solving a first-order optimality conditions on eq. 12 or minimizing a convex optimization problem (eq. 13).

$$\nabla_q ER(q;\mu,\tau_i)|_{q=\epsilon_i} = 0, \forall_i \in [K] \tag{12}$$

$$\sum_{i=1}^{K}(\nabla_q ER(q;\mu,\tau_i)|_{q=\epsilon_i})^2 \tag{13}$$

For further details, [13].

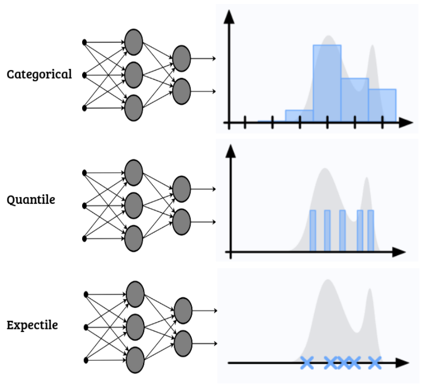


Figure 5: Comparison of Categorical, Quantile and Expectile codes.

B Laplace DQN

B.1 Laplace Encoding

Figure 6 shows an intermediary step in our method as a sanity check, replicating the same rewards structure in the toy Markov Process (MP) of [17]. The bottom half of the image is generated on our code. The caption is adapted from the referenced paper.

Figure 6 shows the Laplace code applied to a specific MP. In the **top**, starting from s at timesteps $\tau = 0$ and $\tau = 1$ the agent obtains with equal probabilities r = 2 and -2. At $\tau = 2$ and $\tau = 3$ it obtains with equal probabilities r = 1 or -1.

In the **bottom**, starting from the left. Applying the Q-learning backups to the MP coverges on the set of $Q_{h,\gamma}(s,a)$ shown in the γ-space. The plot shows $Q_{h-1,\gamma}(s,a) - Q_{h,\gamma}(s,a)$ as a function of the parameter and γ of the unit.

The arrow symbolizes the transition under the inverse laplace operation into the τ space. Using the sigmoid function, the final τ space at time τ is a smooth approximation to the probability that the immediate reward in τ timesteps from the present will converge to θ_h, given that the current state is s.

As illustrated, the Laplace code recovers a temporal map of the environmnent, inidicating all possible rewards at all future times.

B.2 Methods to enforce monotonicity comparison

For Laplace DQN, we required the ouput of dimension (#gamm, #sens, #acts) to be non-decreasing in the axis of number of sensitivities due to the mathematical theory. We tried without any monotonicy constraint but could not train laplace agent optimally. As a result, we enforced this constraint with different approaches. The simplest one was to do accumulation since the output Q values are positive due to Heaviside and Expectation as shown in (15).

Other alternative we tried was keeping the largest to the left by based on the maximum of the differences and zero, ensuring no increase between consecutive elements.

The comparison of different monotonicity methods are illustrated in Figure 7. What

$$Q_{h,\gamma}(s_t,a_t) \to \mathbb{E}\left[\sum_{\tau=0}^{\infty}\gamma^{\tau} f_h(r_{t+\tau})\Big| s_t,a_t\right] = \sum_{\tau=0}^{\infty}\gamma^{\tau}\mathbb{E}\left[f_h(r_{t+\tau})\Big| s_t,a_t\right] \tag{14}$$

$$= \sum_{\tau=0}^{\infty}\gamma^{\tau}\mathbb{E}\left[H(r_{t+\tau}-\theta_h)\Big| s_t,a_t\right] = \sum_{\tau=0}^{\infty}\gamma^{\tau} P(r_{t+\tau}>\theta_h|s_t,a_t) \tag{15}$$

B.3 Different Reward Sensitivity Resolution

In our default Laplace-DQN setting, we distributed sensitivities uniformly across an interval [r_min, r_max]. However, we tested alternative sensitivity distributions as uniform distribution seemed suboptimal because of the way certain environments (like LunarLander) have rewards distributed. As we received rewards more frequently around the center of the interval, that is how we chose the interval. Hence, we placed a Gaussian distribution for the sensitivities placed at (r_min + r_max)/2. Although we expected this approach to work better, we had better results with simple uniformly distributed sensitivities. The comparison with different methods are illustrated in table 2.

Method	Score
Uniform	188.45
Gaussian Std-20	-90.22
Gaussian Std-50	-203.77

Table 2: Laplace DQN Average Reward for different sensitivity resolution in LunarLander

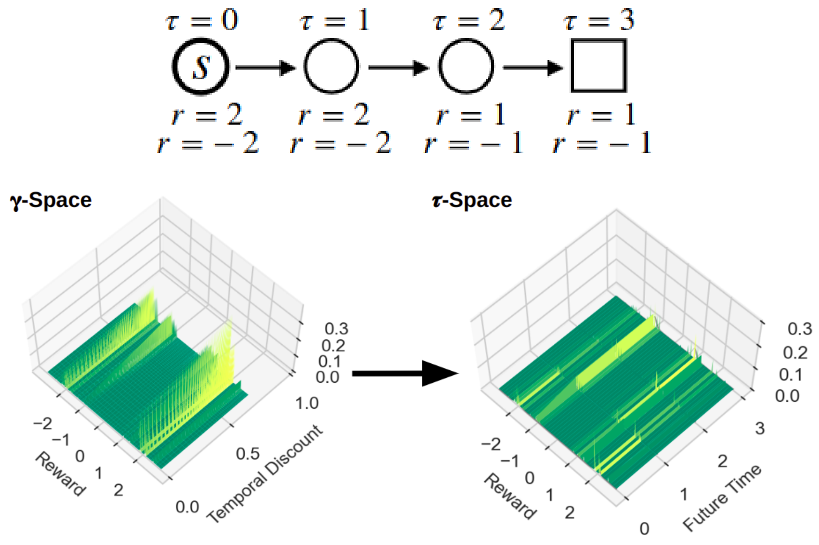


Figure 6: A Toy MP and a visualization of the evolution of rewards.

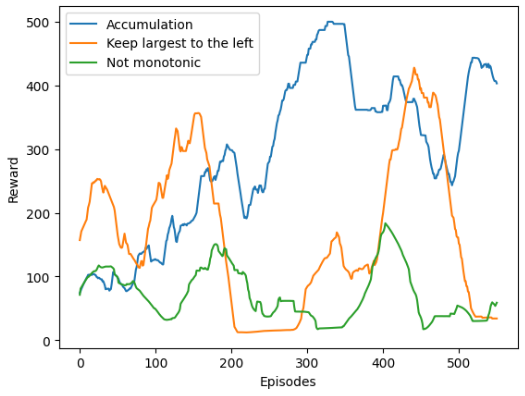


Figure 7: Comparing the performance of different methods to enforce non-increasing values over increasing sensitivities

C Hyperparameters

C.1 Optuna and Hyperparameter Search

An extensive part of our method was an effort in finding the best hyperparameters for each of the architectures we tested. The hyperparameter search was conducted through the optuna library, and experiments running on optuna encompassed about 2 full weeks of the allotted time for the project. Tests were done on the following parametrizations, searching over the following hyperparameters:

Parametrization	Hyperparameters	Values
Quantile Parametrization	batch size	128
	buffer size	100000
	exploration final eps	0.04
	exploration fraction	0.16
Categorical Parametrization	batch size	128
	number of atoms	62
	ranges: v_min and v_max	*environment-dependant*
	start train at	32
	update net every n steps	4
	learning rate	.006
Laplace Parametrization	ranges: r_min and r_max	*environment-dependant*
	sensitivities	500
	learning rate	0.001
Laplace Parametrization with multiple gammas	sensitivities	100
	sharpnesses	25
	number of gammas	5
	learning rate	0.005

The optimizaztion objective function was to increase the average of the rewards over the last 100 episodes. When using the Laplace parametrization, the aveerage of the rewards accross multiple discount factors,γ's, was maximized. The optuna trials were conducted over the following environments, for all of the parametrizations:

1. gym Cartpole
2. gym Acrobot
3. gym LunarLander

Due to the added increased complexity of LunarLander, we observed an increase of experiment runtime from about 10 hours to conduct all 50 experiments in a trial, to 3 days, an increase in 600%. Each experiment fully encompasses one training run with a selection of hyperparameters.

Records from all hyperparameter search trials can be provided upon request.