Reinforcement Learning with Possibility Theory

Tejas Gupta

Submitted as part of the honours requirements

Supervisor: Dr. Jeremie Houssineau

Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University

April 2025

Abstract

Deep Reinforcement Learning (RL) agents often struggle to balance exploration and exploitation due to their value estimates not accounting for epistemic uncertainty. Possibility theory, with its maxitive calculus and less restrictive normalization constraints, offers a method for tracking and using epistemic uncertainty. This thesis investigates whether possibilistic modeling of uncertainty can drive principled optimistic exploration. Three algorithmic variants are proposed: (i) **Possibilistic DQN**, which models Q-values via a Gaussian and acts via a parameter-free closed-form maximum expected value; (ii) **Possibilistic Q-Ensembles**, which maintain possibility weighting over multiple Q-networks and update them using a possibilistic Bayes rule; and (iii) **Possibilistic Model-Based Learning**, a pair of zero- and one-step planning algorithms that propagate optimistic targets from possibilistic models. The models are tested on benchmark environments along with sparse and stochastic variants. The study confirms the efficacy of maximum-expected-value optimism in the first method, particularly in deterministic and sparse environments. The ensemble possibilistic approach outperforms standard non-weighted ensembles and DQN. The third model underperformed relative to baselines; further work is needed to tune hyperparameters and improve model accuracy. Overall, we demonstrate the utility of possibility theory to capture uncertainty and make exploration more efficient.

Acknowledgements

I would like to express my sincerest gratitude to my supervisor, Prof. Dr. Jeremie Houssineau, for his insightful guidance and constant support. I am incredibly grateful to have had the opportunity to work on such a novel topic under his expertise.

I am thankful to my friends, who kept me motivated throughout the sometimes challenging journey of the project. In particular, I would like to thank Rudra and Ishika for reviewing my work and providing honest and constructive feedback on my presentation.

I extend my appreciation to my parents for their encouragement and unconditional support during the periods when progress was slow and difficult. I also want to thank my sister, Dhruti, whose exceptional work ethic has been an incredible source of inspiration.

Contents

Chapter 1

Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm within machine learning, enabling agents to learn optimal decisions in a broad range of environments. It has facilitated superhuman performance in complex games ([Silver et al., 2017]) and enabled control over robotic agents ([Liu et al., 2021]). The applicability of RL across diverse use cases stems from its core concept: an agent interacts with an environment by performing actions, which transition the agent from its current state to a subsequent state. Throughout this movement, the agent accumulates rewards. To make decisions, the agent employs a policy π, which maps the current state s to an action a. In order to learn effective policies, classic RL algorithms often estimate the expected return the agent anticipates from a state or a state-action pair; these are referred to as $V(s)$ and $Q(s, a)$ respectively.

A fundamental challenge in reinforcement learning is addressing uncertainty, which manifests in two primary forms: aleatoric and epistemic. Aleatoric uncertainty arises from the inherent randomness or stochasticity within an environment, such as probabilistic state transitions or random rewards. Epistemic uncertainty, conversely, stems from the agent's lack of knowledge. For instance, a reward might randomly be either 0 or 1 with probabilities $(p, 1 - p)$, representing aleatoric uncertainty. However, the agent might also be uncertain about the value of the probability p itself; this uncertainty, originating from the agent's incomplete knowledge about the reward probabilities, constitutes epistemic uncertainty. This work focuses specifically on epistemic uncertainty.

Epistemic uncertainty can be detrimental in reinforcement learning as it may lead to suboptimal policies. An agent might incorrectly conclude that a particular action is optimal due solely to its limited knowledge regarding the potential value of alternative actions. This frequently results in suboptimal outcomes, especially when the agent acts greedily and neglects to explore other options adequately. This exploration-exploitation trade-off represents a central problem in reinforcement learning. The two most prevalent methods for managing this trade-off are incorporating stochasticity in actions and employing optimism. With stochastic actions, the agent selects random actions with a certain probability ϵ, ensuring continuous exploration. Optimism involves assuming initially high values for unexplored actions, encouraging their exploration in the hope they might be optimal, and gradually refining these value estimates based on experience. While simple to implement, random action selection can be inefficient; for example, an agent should arguably explore less in frequently visited states and more in underexplored states. Optimism proves successful in simple tabular and smaller environments but is difficult to maintain in larger environments, particularly when value functions are parameterized by neural networks.

Possibility theory, introduced in [Zadeh, 1999], offers an effective approach to modeling uncertainty.

It serves as a less restrictive alternative to probability theory, allowing representation of belief in an event with a value between 0 and 1, where 1 signifies merely the absence of evidence contradicting the event's possibility. Possibility theory is less constrained as it lacks summative requirements, making the representation of complete ignorance straightforward. Its utility in modeling epistemic uncertainty, particularly in model-based reinforcement learning, has been demonstrated recently, for example in [Thomas and Houssineau, 2025], which explicitly models the environment possibilistically through the proposed algorithm Possibilistic Q-learning (PQL). Possibilistic modeling also introduces a natural notion of optimism via the maximum expected value, which balances exploration and exploitation more systematically than ϵ-greedy policies. Furthermore, possibility theory simplifies evidence-based updates (through Possibilistic Bayesian Update), a feature shown to be beneficial in PQL.

Although PQL was applied to model epistemic uncertainty in tabular environments, other research in Deep Reinforcement Learning has addressed uncertainty through Distributional Reinforcement Learning ([Bellemare et al., 2017a], [Dabney et al., 2017], [Zhang et al., 2022]). In these approaches, Q-values are represented not as single numbers but as entire probability distributions over possible Q-values, often modeled using atomic probabilities, quantiles, or Gaussian distributions. This distributional perspective is useful for predicting the risk associated with actions and can facilitate the development of risk-averse or risk-seeking policies. However, to the best of the author's knowledge, these distributions have not typically been employed to model the uncertainty in the Q-value estimates themselves, nor have they been explicitly used to guide exploration.

This thesis explores the utility of possibility theory for modeling the uncertainty inherent in an agent's value function estimates and investigates whether such uncertainty can be leveraged to drive exploration systematically within the environment.

To this end, three novel algorithms are proposed:

- **Possibilistic DQN:** Models Q-values using Gaussian possibility distributions. A corresponding maximum expected value measure is proposed, which demonstrates superior performance compared to tested benchmarks through improved targeted exploration induced by natural optimism.

- **Possibilistic Q-Ensembles:** Models the possibility of parameter sets θ being optimal, outperforming standard ensemble methods and DQN.

- **Possibilistic Model-Based Learning:** Models environment transitions possibilistically in continuous environments, incorporating optimistic model-based planning. Both zero-step and one-step roll-out methods are proposed.

The corresponding code is open-source and available at `https://github.com/tejaey/RL_possibility_theory`. The results are compared to standard baselines: DQN for discrete action spaces (Possibilistic DQN and Ensembles) and DDPG for continuous action spaces (Possibilistic Model-Based Learning). The thesis outline is as follows: Fundamental topics in reinforcement learning and possibility theory are introduced in 2. Subsequently, the proposed algorithms are detailed across three chapters: 3, 4, and 5. The experimental setup is described in 6, followed by the presentation and discussion of results in 7. The thesis concludes with a summary and directions for future work in 8.

Chapter 2

Background

2.1 Possibility Theory

Possibility theory, introduced in [Zadeh, 1999], is a counterpart to probability theory that provides an alternative, flexible method of measuring uncertainty. In this framework the uncertainty of an event is quantified by a possibility measure, which offers an alternative to model uncertainty due to incomplete knowledge. The possibility of an event can range from 0 to 1, where a value of 0 implies that the event is completely impossible and a value of 1 implies that the event is fully possible, i.e. we have no evidence against it being possible. In other words, possibility refers to the degree with which an event is possible given our current knowledge. In this regard, possibility is a purely ordinal measure, in that a possibility $\hat{p}(x)$ of even x being 0.8 only implies that the event is more possible than all events with less possibility and vice versa. This is in contrast to probability measures, where a probability of 1 implies that an event will certainly occur and a probability of 0.8 typically implies that the frequency of the event is 0.8.

2.1.1 Fuzzy Sets and Possibility Distributions

Possibility theory was introduced as an extension to fuzzy sets in [Zadeh, 1999]. A fuzzy set $\tilde{A}$ is defined as a set of ordered pairs:

$$\tilde{A} = \{(x, \hat{p}_{\tilde{A}}(x)) \mid x \in X\},$$

where $\mu_{\tilde{A}} : X \to [0, 1]$ is the membership function to the fuzzy set. The membership function over the set can also be understood as a *Possibility Distribution* $\hat{p}(x)$ over the set X. Analogous to probability theory, where the sum of the probabilities of all outcome states must be 1, a possibility distribution must ensure that at least one state is fully possible, i.e.,

$$\sup_{x \in X} \hat{p}(x) = 1.$$

The induced possibility measure for any subset $A \subseteq X$ is defined as the maximal value of $\hat{\pi}$ over the states:

$$\hat{P}(A) = \sup\{\hat{P}(x) \mid x \in A\}.$$

possibility measures also satisfy:

$$\hat{P}(\Omega) = 1, \quad \hat{P}(\varnothing) = 1.$$

Additionally, possibility measures, like fuzzy measures require monotonicity:

$$A \subseteq B \implies \hat{P}(A) \leq \hat{P}(B).$$

[Dubois and Prade, 2001] also introduced the notion of necessity, the dual of possibility, defined as

$$N(A) = \min\{1 - \hat{P}(x) \mid x \in A\} = 1 - \hat{P}(\neg A).$$

Necessity quantifies the lack of plausibility of the complement of an event, so that possibility and necessity together can be interpreted as upper and lower probability bounds of imprecise probabilities ([Dubois and Prade, 1992]).

2.1.2 Additivity and Maxitivity

Probability measures are *additive*. For any two disjoint events A and B (i.e., $A \cap B = \emptyset$), the probability of their union is given by

$$P(A \cup B) = P(A) + P(B).$$

In contrast, possibility measures are *maxitive* (or supremum-preserving) ([Dubois and Prade, 2007]). For any events A and B, the possibility measure of their union is given by

$$\hat{P}(A \cup B) = \max\{\hat{P}(A), \hat{P}(B)\}.$$

This property implies that if at least one event is highly possible, then their union is considered highly possible. This different choice of t-conorms for possibility measures results in a very different arithmetic of uncertainty, which can simplify the handling of incomplete information. In particular, maxitivity allows us to optimistically handle epistemic uncertainty using the notion of maximum expected value defined as:

$$\hat{\mathbb{E}}(x) = sup\{x \cdot \hat{p}(x)\}$$

2.1.3 Normalization

A probability distribution over an outcome space X requires that the probabilities of all states sum to 1:

$$\sum_{x \in X} P(x) = 1.$$

Even in situations of complete ignorance, a uniform distribution is imposed, which still assigns fractional probabilities to each outcome.

A possibility distribution, on the other hand, is normalized by requiring that at least one outcome has the maximal possibility:

$$\sup_{x \in X} \hat{p}(x) = 1.$$

This normalization permits complete ignorance to be represented trivially by assigning $\hat{p}(x) = 1$ for every x in X. Under such a distribution, each event has a necessity of 0, since

$$N(A) = 1 - \hat{P}(A^c) = 0,$$

when nothing is ruled out. This flexibility makes it easier to represent uncertainty qualitatively without imposing precise quantitative values.

2.1.4 Bayesian Update

One of the key advantages of possibility theory is that it makes it trivial to represent uninformed priors by setting possibility of all events to 1. Possibilistic bayesian update can be used to update the possibility of an event as more information is acquired.

In classical probability theory, Bayes theorem updates a prior distribution $P(\theta)$, in light of the evidence e with the likelihood $P(\theta \mid e)$ as

$$P(\theta \mid e) = \frac{P(e \mid \theta)P(\theta)}{\sum_{\theta_i} P(e \mid \theta_i)P(\theta_i)}$$

The denominator ensures that the posterior is normalised.

Using the concept of the maxitive union and maximum expected value, a similar Bayesian Update can be formulated with respect to the possibility measure, which can be expressed as:

$$\hat{P}(\theta \mid e) = \frac{\hat{P}(e \mid \theta)\hat{P}(\theta)}{\sup_{\theta_i}\{\hat{P}(e \mid \theta_i)\hat{P}(\theta_i)\}}$$

Further details can be found in [Dubois and Prade, 2013].

2.2 Reinforcement Learning

Reinforcement Learning is a machine learning framework for an agent's sequential decision making in an environment. At each time step, the agent observes the state in which it currently is, takes an action which moves it to another state, and collects a reward (the reward collected can be zero).

The notion of Actions, States, Rewards, and the associated stochastic transitions is formally known as the Markov Decision Process (MDP). Here we will discuss some core Reinforcement Learning concepts along with previous work employing possibility theory.

2.2.1 Markov Decision Process

A MDP is defined by the mathematical tuple (S, A, P, R, γ) where

- **State Space S:** refers to all possible states in an environment.
- **Action Space A_s:** refers to all possible actions available to the agent in the state s. In some formulations, the action space A might be the same across states.
- **Transition Probabilities $P(s' \mid s, a)$:** refers to the probability of transitioning to state s' by taking the action a in state s. These transitions can be either stochastic or deterministic.
- **Reward function $R(s, a, s')$:** is the immediate reward received by taking the action a in state s and transitioning to state s'. $R(s, a)$ refers to the expected reward received by taking action a in state s.
- **Discount Factor γ:** is the discounting factor of future rewards to determine the current value of the current state. A reward of 1 obtained after K steps is worth γ^K at the current step. Trivially, if γ is 1 then there is no discounting of future rewards.

As the name suggests, the Markov decision process also satisfies the Markov Property, i.e., the next state s' and the reward r only depend on the current state-action pair (s, a); all prior history is irrelevant. It is important to note that it is possible to encode information about the history in the current state to convert environments whose transitions are path dependent to satisfy the Markov Property.

The agent's behaviour in a state is characterised by its policy π, where $\pi(a \mid s)$ refers to the probability of the agent enacting a at state s. The goal of reinforcement learning is to find an optimal policy π^* that maximises cumulative rewards in an MDP.

R_t refers to the random variable denoting the reward the agent receives at timestep t. We can further define the cumulative rewards from the time step t as

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Here, both the reward random variable and the cumulative reward random variable depend on the state at the current time t and the policy of the agent π. The expected cumulative reward under a given policy is represented by the state-value function $V^\pi(s)$.

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

Similarly, an action value function (Q-value) $Q^\pi(s, a)$ can be defined as the expected cumulative return from state s if the agent takes action a.

$$Q^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

These expectations quantify how good an action or state is in terms of its expected cumulative rewards. Correspondingly, two policies can be compared on a given state by comparing the value functions induced by that policy in that state. An optimal policy, hence, is the policy π^* that induces the optimal value function $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all s, a.

These expected values are also dependent on each other.

$$V^\pi(s) = \mathbb{E}_\pi[Q^\pi(s, A) \mid S = s]$$

$$Q^\pi(s, a) = \mathbb{E}^\pi[R + \gamma V^\pi(S') \mid S = s, A = a]$$

By substituting the values further, one can construct a recursive relationship; this is also known as the Bellman Equation:

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s].$$

The state value of the current state is just the same as the transition reward and the discounted state value of the next state. A similar relationship exists for the action value function as follows

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s]$$

The definition of the recursive expectations can be fully expanded as follows:

$$V^\pi(s) = \sum_{a \in A} \pi(a \mid s) \sum_{s' \in S} P(s' \mid s, a) \left[R(s, a, s') + \gamma V^\pi(s')\right]$$

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\big[R(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s'), Q^{\pi}(s',a')\big]$$

For a given policy, the Bellman Equations are linear. However, for an optimal policy, we have nonlinear maximisation operations as follows:

$$V^*(s) = \max_{a \in A} \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a]$$

This gives the intuitive result that the optimal value of a state is the same as the expected value of taking the best action from the state. The same result also applies to the Q-function:

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a]$$

In a finite state and action space, it is possible to solve the Bellman Optimality Equations to get the optimal values using value iteration or policy iteration, making it possible to calculate V^* from which it is trivial to deduce an optimal policy. However, in larger and continuous environments, this is no longer feasible. Thus, reinforcement learning algorithms attempt to either learn the value functions directly or learn a maximising policy by experiencing the MDP.

2.2.2 Deep-Q-Learning (DQN)

Q-Learning is a category of reinforcement learning algorithms that focus on on learning the optimal Q^* value for each state action pair by iterative updates. Then, the optimal policy involves taking the action that maximises the Q-value with respect to the current state. Q-learning is off-policy in that the agent does not explicitly learn a policy, but learns the Q-values from which a policy can be constructed. In its simple tabular form, the iteration happens as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Here α is the learning rate and (s,a,r,s') is one of the experienced transitions. This update is based on the Bellman Update discussed before. [Watkins and Dayan, 1992] showed that this tabular Q-learning converges to the optimal value if all states are visited infinitely during the learning process and the learning rates satisfy:

$$\sum_{t=0}^{\infty} \alpha_t(s,a) = \infty$$

$$\sum_{t=0}^{\infty} \alpha_t^2(s,a) < \infty$$

As mentioned before, the tabular approach to Q-learning is not feasible for continuous or large environments. Deep Q Learning provides an alternative method to functionally approximate the Q-values using deep neural networks. The function $Q(s, a \mid \theta)$ is parametrised by weights θ.

The method for doing Deep Q Learning was first introduced [Mnih et al., 2015] where the authors achieved human level performance in various ATARI games. In their paper, the network takes the state of the game (in the form of a raw image) and outputs the approximate Q-values for each of the set of discrete actions. Common to other Deep Learning Methods, Stochastic Gradient Descent is used to update the Q networks, with the loss function derived from the temporal difference (TD)

error. Particularly, the TD Error is the error between the temporal difference target y and the predicted state-action value $Q(s, a \mid \theta)$. The target y is:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

where θ^- is the parameters of the target network. A target network is typically a lagged copy of the main Q-Network and helps stabilise the learning process. The target network can either be continuously updated using Polyak Averaging $\theta^- \leftarrow \theta^- + \alpha(\theta - \theta^-)$ or it can be a copied from the main network periodically during the learning process.

The loss function can be constructed to minimise the mean squared TD Error.

$$L(\theta) = \mathbb{E}[(y - Q(s, a, ; \theta))^2]$$

The loss is calculated over a batch D which is a set of tuples (s, a, r, s') experienced by the agent. Taking the gradient of the loss:

$$\nabla_\theta L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[2 \cdot (Q(s, a; \theta) - y) \cdot \nabla_\theta Q(s, a; \theta)]$$

The network is updated by moving the parameters θ to minimise the loss:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta L(\theta)$$

In practice, the networks are not trained using sequential experiences as this can lead to divergence and instability in the learning process. Instead, [Mnih et al., 2015] introduced experiential learning. The transitions (s, a, r, s') experienced during training are stored in a replay buffer. During the training step, a mini-batch is randomly sampled from the replay buffer. The random sampling breaks temporal correlations in the data and generally results in smoother learning. Reusing past transitions also improves data efficiency as each transition can be used multiple times to improve the learning process.

2.2.3 Actor-Critic Methods

Actor Critic Methods are a type of On-Policy reinforcement learning algorithm, in that they explicitly learn the policy of the actor along with a critic (usually the value functions). In general, the actor decides what actions to take based on the state, while the critic estimates the advantage of the actions. The actor is updated to maximise the advantage estimate of the critic. Actor-Critic methods can handle continuous and larger action spaces better then Q-learning.

In policy gradient methods, the policy $\pi_\theta(s)$ is parametrised by θ, this is often a implemented using neural networks. An objective function $J(\theta)$ is defined as the expected return from the environment following policy parametrised by θ from some starting distribution of states. The policy gradient theorem provides a way to express the gradient of the objective function $J(\theta) = \mathbb{E}_{s_0\sim\rho_0, a_t\sim\pi_\theta}[R_t]$ with respect to some starting distribution of states ρ_0 as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}\left[Q^\pi(s, a)\nabla_\theta \log \pi_\theta(a \mid s)\right]$$

where s is distributed by d^π (the visitation frequency of the states under the policy π_θ) and a is distributed by the policy π_θ. The parameter can then updated using

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

In practice, we replace the unknown $Q_\pi(s, a \mid \theta)$ with the temporal difference:

$$\delta_t = r_{t+1} + \gamma V_{w'}(s_{t+1}) - V_w(s_t)$$

where we can minimise δ_t^2 to update w. Similar to the DQN learning discussed before, there is usually a target critic and online critic to stabilise learning. Importantly, $r_{t+1} + \gamma V_{w'}(s_{t+1})$ is an estimator of the Q-value $Q(s, a)$ where the action a is determined by the current policy π_θ; therefore δ_t itself is an estimator for the advantage function

$$A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t).$$

Hence, the actor's gradient update can be implemented as

$$\Delta\theta \propto \Delta_\theta log \pi_\theta(a_t \mid s_t).$$

This nudges the policy to increase the probability of action a_t if the observed reward is greater than the expected value estimated by $V_w(s)$.

Deterministic policies can also be used in place of the stochastic policy described above. In the Deterministic Policy Gradient (DPG) approach, the policy is represented by a deterministic function $a = \mu_\theta(s)$, where μ_θ is the actor network parameterized by θ. The policy update is based on the following gradient:

$$\nabla_\theta J \approx \mathbb{E}_{s\sim D}\left[\nabla_a Q_w(s, a)\big|_{a=\mu_\theta(s)} \cdot \nabla_\theta \mu_\theta(s)\right],$$

where D denotes the replay buffer containing past experiences.

In summary, actor critic methods, in its various forms (such as PPO, DDPG and A2C), involve a critic that predicts returns from a given state or state-action and the critic improves the policy to maximise the returns predicted by the critic.

2.2.4 Model-Based Reinforcement Learning

Model-Based RL algorithms build an internal simulator of the environment, by modeling the state transitions $P(s' \mid s, a)$ and reward functions $R(s, a, s')$. The agents are then able to train/plan using the simulator to improve it's policy without costly interactions with the 'real' environment.

In practice, learning the model can be treated as a supervised learning task as it is possible to treat each recorded transition (s, a, r, s') as a training sample for the models P' and R'. The model can help both in action selection and with value iteration. Dyna-Q Learning is common approach for value iteration, where the Q-values are updated by simulated planning steps that are intermixed with real experiences. The model can also utilized to improve in action selection, for example, it is possible to do Monte Carlo Tree Search to simulate action sequences and select the best performing action.

One of the essential challenges of model-based learning is learning an accurate model. A biased or imperfect model can result in non-optimal policies. There are many different approaches to account for this, such as shorter planning horizons and ensemble models. This thesis introduces possibilistic model-based learning as an alternative to these approaches.

2.3 Possibility Theory and Reinforcement Learning

Possibility theory provides a maxitive framework for modeling *epistemic* uncertainty—uncertainty arising from limited knowledge—complementing existing methods that primarily address *aleatoric*

randomness. In what follows, we review two approaches to account for uncertainty in value functions:

- **Distributional Reinforcement Learning** (e.g. C51 [Bellemare et al., 2017a]), which models the entire return distribution to account for variability in rewards.

- **Possibilistic Q-Learning** (PQL [Thomas and Houssineau, 2025]), which maintains possibility functions over transitions and rewards and applies supremum-based updates to drive principled optimism without additional hyperparameters.

2.3.1 Distributional Reinforcement Learning

One possible method of handling uncertainty in the environment is by maintaining distributions over $Q(s,a)$ or $V(s,a)$ instead of a single value. Along the same line, distributional Reinforcement Learning, introduced in [Bellemare et al., 2017a], provides a novel method to handle uncertainty in the reward distribution by modelling a distribution Z^π over Q Values. In particular, instead of learning only

$$Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$$

where Z^π is the random cumulative return, satisfying the above expectation and the recursive relationship

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

The above return also satisfies the Bellman Equation:

$$Z^*(s,a) \overset{D}{=} R_{t+1} + \gamma\, Z^*\Big(S_{t+1},\, \arg\max_{a'} \mathbb{E}\big[Z^*(S_{t+1},a')\big]\Big)$$

The algorithm C51 in [Bellemare et al., 2017a] works by maintaining a probability distribution over 51 possible values $\{z_1, z_2, ..., z_n\}$, which are placed uniformly over the range of returns. In particular, instead of the Q-Network outputting a single numeric value of state action pair, the network returns a probability tuple $\big(P(Z(s,a)=z_1), P(Z(s,a)=z_2), ..., P(Z(s,a)=z_{51})\big)$.

For training, a target distribution is calculated as: $T(s,a) \overset{D}{=} r + \gamma Z(s',a^*)$ where a^* is the greedy action that maximises $\mathbb{E}[Z(s',a)]$.

One drawback of the approach is that the two different uncertainties cannot be easily decoupled. If information about the epistemic uncertainty of the distribution was available, it would be possible to incentive the agent to explore states with less certainty. The use of probability theory further enforces a frequentest belief on the Q-values. Later, we provide an extension of C51 with the use of possibility theory.

2.3.2 Possibilistic Q-Learning

Possibilistic Q-Learning (PQL) extends the traditional Q-learning algorithm by explicitly modeling and propagating epistemic uncertainty separately from aleatoric randomness ([Thomas and Houssineau, 2025]). It maintains possibility functions over the transition kernel $\hat{P}(\cdot \mid s,a)$ and reward distributions $\hat{R}(\cdot \mid s,a)$, updated via a supremum based Bayes rule that normalizes by the maximum plausibility rather than an integral.

At each update step, PQL computes an "optimistic" Bellman backup entirely within the learned

possibilistic model:

$$\overline{Q}(s,a) \;\leftarrow\; \mathbb{E}_{S'\sim\hat{P}(.|s,a)}\Big[\bar{r}(s,a,S') \;+\; \max_{a'}\overline{Q}(S',a')\Big],$$

where $\bar{r}(s,a,S')$ is the maximal-expected reward under the current possibility function. By initializing $\overline{Q}$ to a known upper bound (e.g. $r_{\max}/(1-\gamma)$), unvisited state–action pairs retain high values, naturally driving exploration without additional hyperparameters [Thomas and Houssineau, 2025].

As the agent observes new transitions (s,a,r,s'), both the transition and reward possibility functions are refined via the supremum-normalised update. This causes the optimistic Q–estimates to contract toward their true values, yielding a built-in, tuning-free optimism. Empirical results on tabular GridWorld tasks demonstrate that PQL achieves faster convergence and higher sample efficiency than standard Q-learning and its variants [Thomas and Houssineau, 2025].

Chapter 3

Possibilistic Q Values

Typically in Q-learning based algorithms, the Q-Network outputs a tuple of scalars representing the networks estimates of expected cumulative reward of the particular action a in state s. Exploration, then, driven by external stochasticity (e.g., with ϵ-greedy strategies). These strategies are often unable to take into account the models own uncertainties about Q-values; if the agent had information about the degree of epistemic uncertainty it has for $Q(s, a)$, then that information can be used to drive exploration in a more systemic and efficient manner. In particular, it can be possible to guide the agent to explore actions with higher uncertainty to gather more experience (in the form of transitions (s, a, r, s')) and increase certainty in its Q-values. In this chapter, we present one Deep Q Learning algorithm that parameterizes possibilities using a possibilistic Gaussian. First, we introduce tabular possibilistic Q-Learning with atomic Q-values.

3.1 Atomic Q Values

As discussed in [Bellemare et al., 2017a], Distributional RL parameterizes the return distribution $Z(s, a)$ over a discrete set of atoms $\{q_1, \ldots, q_N\}$ and learns probabilities $p_j(s, a) = \Pr\big(Z(s, a) = q_j\big)$. The atoms are equidistant between V_{max} and V_{min}. We now extend this framework with possibility theory by maintaining, for each state–action pair (s, a), a possibility function

$$\hat{p}\big(q_j \mid s, a\big) \;=\; \text{degree of possibility that } q_j \text{ is the true expected return}, \quad j = 1, \ldots, N.$$

This forms a fuzzy set over the Q-values for the state-action pair s, a as $\hat{Q}(s, a)$. In the beginning, the agent has no experience in the environment and is fully ignorant about the Q-values. Hence the agent has:

$$\hat{p}\big(q_j \mid s, a\big) = 1, \quad \forall\, j,\, s,\, a.$$

All Q-values are fully possible for all state action pairs.

3.1.1 Maximum Expected Q Values

We extend the notion of maximum expected values to Possibilistic Q Values as follows

$$\bar{Q}(s, a) = \sup_q \{q\hat{p}(q \mid s, a)\}.$$

In particular, for atomic Q Values we get:

$$\bar{Q}(s, a) = \sup_{q_j} \{q_j\hat{p}(q_j \mid s, a)\}.$$

There are two immediate issues. First, convergence can be highly sensitive to the Gaussian kernel's width σ. In Appendix B, we demonstrate that different initializations of σ on a toy example can yield suboptimal policies. In principle, σ could be scaled inversely with the visitation count of each state–action pair, but we do not explore that here. Second, updating per-atom possibility values within deep networks is nontrivial. To address these challenges, we propose below a parameter-free, mean–variance–based variant of possibilistic Q-values.

3.1.2 Possibilistic Bellman Update

Given a transition (s, a, r, s'), let $a^* = \arg\max_{a'} \bar{Q}(s', a')$ be the action that with the maximum expected value in the next state. The atomic targets are $\{T_i \mid i \in 1, \ldots, n\}$ where

$$T_k = r + \gamma\, q_k$$

We then compute a compatibility score for each source atom q_j against each target T_k via a Gaussian kernel:

$$L_{jk} = \exp\Big(-\tfrac{(q_j - T_k)^2}{2\sigma^2}\Big).$$

Aggregating over all target atoms weighted by their possibilities gives the total support for q_j:

$$\ell_j = \sum_{k=1}^{N} \hat{p}(q_k \mid s', a^*)\, L_{jk}.$$

Finally, apply a possibilistic normalisation (supremum) to update:

$$\hat{p}_{\text{new}}\big(q_j \mid s, a\big) = \frac{\ell_j\, \hat{p}\big(q_j \mid s, a\big)}{\max_i\{\ell_i\, \hat{p}(q_i \mid s, a)\}}.$$

By construction, $\max_j \hat{p}_{\text{new}}(q_j \mid s, a) = 1$, preserving the possibility normalisation constraints. In practice, the above mentioned update can be too strict and lead to drastic updates. To introduce stability, we exponentially move the possibilities as

$$\hat{p}_{\text{new}}(q_j \mid s, a) = \max(\alpha p_{\text{old}}, \hat{p}_{\text{new}}\big(q_j \mid s, a\big))$$

So that the possibilities never decrease by more than a factor of α.

Constraints

There are two immediate issues. First, convergence can be sensitive to the Gaussian kernel's width σ. In B, we show how different initialisations of σ on a toy example can yield suboptimal policies. In practice, σ can be scaled inversely to the number of visitations of a state-action pair - however we do not explore that here. Second, updating per-atom possibilities is nontrivial in deep networks. To address this, we propose a parameter-free, mean–variance–based possibilistic Q-values variant below.

3.2 Mean-Variance Possibilistic DQN

The aforementioned algorithm can be extended to a continuous space by modelling the uncertainty directly using a possibilistic Gaussian distribution over the Q-values, denoted by $\hat{p}(q \mid s, a)$. The Q-network outputs two parameters for each state–action pair:

- **Mean** $\mu(s, a \mid \theta)$ — the expected value of the return (equivalent to $Q(s, a)$).

- **Variance** $\sigma^2(s, a \mid \theta)$

where θ is the parameters of the neural network.

Together, these parameters define a Gaussian-like membership function for the state–action pair:

$$\hat{p}(q \mid s, a) = \exp\left(-\frac{(q - \mu(s, a))^2}{2\sigma^2(s, a)}\right).$$

By definition, this membership function satisfies the normalization condition

$$\sup_q \hat{p}(q \mid s, a) = 1,$$

which is achieved at $q = \mu(s, a)$; the most credible value of the return is the mean.

3.2.1 Possibilistic Bellman Equation

A possibilistic Bellman equation can be formulated for the above. Given an observed transition (s, a, r, s'), we select the action

$$a' = \arg\max_{a'} \mu(s', a')$$

and define the target parameters as:

$$\mu_{\text{target}} = r + \gamma\, \mu(s', a'),$$
$$\sigma^2_{\text{target}} = \gamma^2\, \sigma^2(s', a').$$

This follows directly from the conventional Bellman equation for Q-learning,

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a'),$$

and, in our case, we propagate both the mean and the uncertainty. Thus, the target distribution can be modeled as:

$$\mathcal{T}Q(s, a) \sim \mathcal{N}\Big(r + \gamma\, \mu(s', a'),\, \gamma^2\, \sigma^2(s', a')\Big).$$

Note that the γ^2 factor in the variance indicates that, if the mean estimate is accurate, then over time the variance should decrease, reflecting an increase in certainty regarding the μ estimates.

It is important to note that if $\gamma = 1$, the variance in the above description does not decrease over time. Although this has not been tested, one could mitigate the issue by scaling the target variance as

$$\sigma^2_{\text{target}} \;=\; \frac{\gamma^2\, \sigma^2(s', a')}{N(s, a)},$$

where $N(s, a)$ is the number of times the agent has observed (s, a). In continuous state–action spaces, $N(s, a)$ can be approximated via a Gaussian kernel:

$$\hat{N}(s, a) \;=\; \sum_{i=1}^{t} \exp\Big(-\tfrac{\|s - s_i\|^2 + \|a - a_i\|^2}{h^2}\Big),$$

so that the target variance shrinks naturally with experience in the environment at that state–action pair, causing σ to contract over time. However, this approach can be computationally expensive. We leave its practical implementation and development to future work.

3.2.2 Loss Function

To update the network, the discrepancy between the target distribution and the current estimated distribution must be minimized. Divergence metrics common in probability theory serve as proxies for a distance measure between our possibilistic distributions. In particular, we consider:

Kullback–Leibler Divergence

For two Gaussian distributions, the KL divergence is given by

$$D_{\mathrm{KL}}\Big(\mathcal{N}(\mu_1,\sigma_1^2)\,\|\,\mathcal{N}(\mu_2,\sigma_2^2)\Big) = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1-\mu_2)^2}{\sigma_2^2} - 1\right],$$

which measures how *surprised* the current estimate $\mathcal{N}(\mu_1,\sigma_1^2)$ would be if the target were $\mathcal{N}(\mu_2,\sigma_2^2)$. A note about D_{KL} is in the Appendix B.

Wasserstein-2 Metric

The Wasserstein-2 metric (also known as the 2nd-order Wasserstein distance) between two Gaussian distributions is

$$W_2^2\Big(\mathcal{N}(\mu_1,\sigma_1^2),\,\mathcal{N}(\mu_2,\sigma_2^2)\Big) = (\mu_1-\mu_2)^2 + (\sigma_1-\sigma_2)^2.$$

This metric provides a geometric measure of the distance between the two distributions, incorporating both the differences in means and differences in spread, and in our case is equivalent to a mean-squared error between the network parameters.

Why we prefer D_{KL} over W_2.

To minimize the Wasserstein-2 distance, one has to solve the optimal-transport problem for each mini-batch, inducting a systematic bias into the stochastic gradients, leading to unstable or vanishing updates. On the other hand, the Kullback-Leibler divergence mini-batch gradients are unbiased estimates of the true population gradients - making them a more robust and reliable in practice ([Bellemare et al., 2017b]). Similar to other works in distributional RL (such as [Bellemare et al., 2017a]), we only focus on loss via D_{KL} in the results.

3.2.3 Action Selection Methods

Information about uncertainty can be utilized to incentivize exploration. We consider two different methods of action selection.

Log-Variance-Weighted Selection

In this method, the action is chosen by combining the mean and the log-variance of the Q-value:

$$a^* = \arg\max_a \left\{\mu(s,a) + \beta\cdot\log\sigma^2(s,a)\right\}.$$

Here, β is a hyperparameter that adjusts the influence of the uncertainty (measured by $\log\sigma^2(s,a)$) on the action selection. β can be interpreted as inducing optimism and providing an exploration bonus to the algorithm (similar to other algorithms such as UCB ([Auer, 2003]))

Maximum Expected Value

Alternatively, we can utilize the notion of maximum expected value introduced in [Thomas and Houssineau, 2025]. The optimistic estimate of the Q-value is defined as

$$\bar{Q}(s,a) = \sup_{q\in\mathbb{Q}}\{q\, f(q\mid s,a)\}.$$

Under the assumption that $f(q \mid s, a)$ is a Gaussian possibility function, this maximum has a closed-form solution (see Lemma 1 in Appendix):

$$\bar{Q}(s,a) = \frac{\mu(s,a) + \sqrt{\mu(s,a)^2 + 4\sigma^2(s,a)}}{2}.$$

This yields parameter free principled explorative strategy for action selection.

3.2.4 Aleatoric and Epistemic Uncertainty

The mean–variance network conflates aleatoric and epistemic uncertainty. For example, consider a two-state terminal MDP (S_0, S_1) with actions a_0 and a_1 that deterministically transition from S_0 to S_1. Action a_0 yields a fixed reward of 1, so $\mu(s_0, a_0) = 1$ and $\sigma(s_0, a_0) = 0$, while a_1 yields $\mathcal{N}(0, 1000)$, so $\mu(s_0, a_1) = 0$ and $\sigma^2(s_0, a_1) = 1000$.

A standard mean–variance decision rule (such as $\arg\max_a \bar{Q}(s,a)$) will repeatedly select a_1 for its high variance, even though that variance is purely aleatoric and the model may be fully certain about the reward distribution. In other words, the mean–variance criterion cannot distinguish epistemic ignorance from aleatoric noise - leading to unnecessary over exploration. The next chapter presents an ensemble approach that isolates epistemic uncertainty using ensemble networks.

Chapter 4

Possibilistic Q-Ensembles

In reinforcement learning, accurately quantifying epistemic uncertainty in action-value estimates is essential for balancing exploration and exploitation. In this chapter, we introduce *possibilistic Q-ensembles*, a framework that represents uncertainty by maintaining a possibility distribution over a set of Q-value functions.

Epistemic uncertainty is modeled by an ensemble of N Q-networks $\{Q_i\}_{i=1}^N$, each of which is a hypothesis about the true Q-function. Sampling from this ensemble approximates a posterior over Q-values, thereby capturing the agent's uncertainty about different state–action pairs. In particular, if the networks disagree on an underexplored pair (s, a), the uncertainty is high; if they largely agree, the region is deemed well explored. Empirically, ensembles tend to yield more accurate Q-value estimates and more stable learning compared to single-network approaches ([Hans and Udluft, 2010]).

We formalize this as follows. Let

$$\mathbb{Q} = \{Q_i\}_{i=1}^N, \quad p_i \;=\; \text{possibility weight for } Q_i, \quad p_i \in [0, 1].$$

Initially, $p_i = 1$ for all i, reflecting an uninformed prior. As training proceeds, each p_i is updated in proportion to the network's likelihood on observed transitions.

More generally, let Θ be the parameter space and let

$$\hat{p}_\Theta : \Theta \to [0, 1]$$

be a possibility distribution over parameters θ, indicating the possibility of θ being the optimal value. We induce a possibility distribution over Q-values by

$$\hat{p}_q(x \mid s, a) = \sup_{\theta \in \Theta} \big\{ \hat{p}_\Theta(\theta) \;\big|\; Q(s, a \mid \theta) = x \big\}.$$

In practice, we sample parameters $\theta_1, \ldots, \theta_N \in \Theta$, train the corresponding networks Q_{θ_i}, and let $p_i = \hat{p}_\Theta(\theta_i)$ to approximate this supremum.

When selecting actions, one naturally considers the maximum expected Q-value under the possibility distribution:

$$\bar{Q}(s, a) = \sup_{\theta \in \Theta} \big\{ \hat{p}_\Theta(\theta)\, Q(s, a \mid \theta) \big\}.$$

Using our finite ensemble, this reduces to

$$\bar{Q}(s, a) = \max_{i=1,\ldots,N} \big\{ p_i\, Q_i(s, a) \big\}.$$

4.1 Bellman Update

There are several choices of Bellman update that can be employed to train an ensemble of Q-networks. Here, we explore two different methods: Independent Update and Conservative Ensemble Update. The choice between the two trades off stability and exploration.

For all networks, a common minibatch $\mathbb{B}$ can be sampled from the set of experienced transitions $\mathbb{D}$. Sampling from the same minibatch allows the losses calculated for the networks to be compared; this is crucial as the loss is used to perform Bayesian updates to the possibility of each network.

4.1.1 Independent Update

Each Q-network is updated independently using its own Q-value estimates and action choices. Using the same minibatch for each network, the target y_i for network i can be calculated as

$$y_i = r + \gamma \max_{a' \in \mathbb{A}} Q_i(s', a')$$

where r is the reward and γ is the discount factor. Separate targets mean that we are not forcing the networks to agree on the Q-values, thereby preserving network diversity. Even though the exploration strategy and experience are shared, not sharing the target information ensures each network remains an independent, valid Q-learning process. This approach does introduce the risk of overestimation, which is common in Q-learning.

4.1.2 Conservative Ensemble Update

In this method, all Q-networks share a common target that takes the maximum over the most pessimistic estimate of the state–action values from the ensemble. The target is computed as

$$y_{\min} = r + \gamma \max_{a' \in \mathbb{A}} \left(\min_{j \in \{1,\dots,N\}} Q_j(s', a') \right)$$

The maxmin update has been shown to reduce the overestimation bias inherent in Q-learning and is a popular strategy when dealing with ensemble Q-networks ([Lan et al., 2021]); it is also a common motivation behind Double DQN. In particular, this method introduces a downward bias as

$$y_{\min} \leq r + \gamma Q_i(s', a')$$

for any network i. With larger N, this could lead to an underestimation bias. Moreover, using the minimum means that all Q-networks are being pushed towards a single estimate of $Q(s, a)$, which can hamper diversity—especially in under-explored states.

B demonstrates empirically that conservative update can lead to quicker convergence within the ensemble. Faster agreement within in the networks can be beneficial, but it could bias the possibilities towards the most pessimistic Q network, decreasing exploration in the environment and the Θ space.

4.2 Bayesian Possibility Update

Using either of the aforementioned updates, the loss L_i for network i can be computed as

$$L_i = \frac{1}{|\mathbb{B}|} \sum_{(s,a,r,s') \in \mathbb{B}} \big(y_i - Q_i(s, a)\big)^2.$$

The loss provides a likelihood estimate given by

$$\mathcal{L}_i = e^{-L_i}.$$

The new possibility $\hat{p}_i$ for network i is estimated by

$$\hat{p}_i = \frac{\hat{p}_i \, \mathcal{L}_i}{\sup_{j \in \{1,\ldots,N\}} \{\hat{p}_j \, \mathcal{L}_j\}}.$$

However, updating the possibility solely using this rule can result in instability, as the p_i may change drastically over choices of mini-batches. In practice, one network's possibilities remains 1 while the possibility of the rest goes to zero. To ensure stability, we use possibilistic exponential moving average:

$$\hat{p}_i = \max \left\{ \alpha \, \hat{p}'_i, \; \frac{\hat{p}_i \, \mathcal{L}_i}{\sup_{j \in \{1,\ldots,N\}} \{\hat{p}_j \, \mathcal{L}_j\}} \right\}.$$

This procedure ensures that the possibility does not ever decrease by more than a factor α. Both the approaches maintain that at least one θ_i in the ensemble remains fully possible - the results for both are compared in the results.

B demonstrates how the average possibility of a network evolves given the above two methods (Bayes Update and Bayes with EMA). The slower reduction in the average possibility for Bayes with EMA means that the ensemble has more opportunity to explore the Θ space.

4.3 Action Selection

Given an ensemble with possibility weights, we consider two different methods for action selection.

4.3.1 Maximum Possible Value

In this scheme, we compute an ensemble Q-value as a weighted average:

$$\bar{Q}_{\text{ens}}(s, a) = \sup_i \{p_i \, Q_i(s, a)\}$$

and select the next action as

$$a^*(s) = \arg\max_{a \in \mathcal{A}_s} \bar{Q}_{\text{ens}}(s, a).$$

Here $\bar{Q}_{\text{ens}}(s, a)$ acts a proxy for the *true* maximum expected value from

$$\bar{Q}(s, a) = \sup_{\theta \in \Theta} \big\{ \hat{p}_\Theta(\theta) \, Q(s, a \mid \theta) \big\}.$$

4.3.2 Majority Voting

Majority voting is another common technique in ensemble approaches. In this method, each network votes for an action based on

$$a_i^* = \arg\max_{a \in \mathcal{A}} Q_i(s, a),$$

and the value of each vote is weighted by p_i. The final action is chosen as the one with the highest total weighted vote. This approach is resilient to outlier Q-value estimates because each vote is clipped at p_i and does not directly depend on the magnitude of $Q_i(s, a)$. Majority voting has been shown to lead to more robust policies ([Hans and Udluft, 2010]), partially because it mitigates the effects of overestimation from any single model.

Chapter 5

Possibilistic Model Based Learning

MaxMax Q-Learning ([Zhu et al., 2024]) is a model-based actor–critic algorithm that leverages imagined future transitions to maintain optimism under uncertainty. At each step, the agent samples $s'_1, \ldots, s'_k$ from its learned dynamics model $P(s' \mid s, a)$ and applies the Bellman backup

$$Q(s,a) \;\leftarrow\; R(s,a) \;+\; \gamma \max_{1 \le i \le k} \Big[\max_{j \in \{1,2\}} Q_j\big(s'_i, a'\big) \Big],$$

thereby updating toward the most promising imagined outcome. This double maximization drives exploration by focusing on the highest-value transitions. Here, Q_1 and Q_2 denote the Q-value functions for two cooperative agents in the environment.

Here, we extend the above approach to single-agent model-based reinforcement learning by performing possibilistic planning steps. To perform the planning step, a possibilistic distribution $\hat{P}_s(s' \mid s, a)$ over next states along with the reward model $R(s, a, s')$ is trained. The uncertainty in these possibilistic transitions is then handled optimistically by taking the maximum over the imagined TD targets. We provide two different approaches:

- **Zero-Step Update:** We sample s^*_k uniformly from $N_\epsilon(s)$ and update the Q_θ network using TD-errors from those imagined targets.
- **One-Step Update:** We use each sampled $S' \sim \hat{P}_s(. \mid s', \mu(s'))$ to do one-step roll-out under the current policy, then update the value of $Q_\theta(s, a)$.

5.1 Zero–Step Update

Figure 5.1 illustrates our *zero-step model-based update*, which utilizes each real transitions to create imagined zero-step roll-outs generated by a learned dynamics model $\hat{p}(. \mid s^*, \mu(a)$. Intuitively, the roll-outs propagate optimistic value information to Q_θ for the local neighbourhood of each replay-buffer state.

Given a real transition (s, a), we draw K neighbour states $s^*_1, \ldots, s^*_K$ uniformly from an ε-ball $\mathcal{N}_\varepsilon(s)$. For each $k = 1, \ldots, K$, the policy μ_ϕ prescribes an action $a^*_k = \mu_\phi(s^*_k)$, and we sample n possible successors

$$s'_{k,i} \sim \hat{p}(\cdot \mid s^*_k, a^*_k), \quad r_{k,i} = R(s^*_k, a^*_k, s'_{k,i}).$$

Each successor yields a return

$$y_{k,i} = r_{k,i} \;+\; \gamma\, Q_\theta\big(s'_{k,i}, \mu_\phi(s'_{k,i})\big),$$

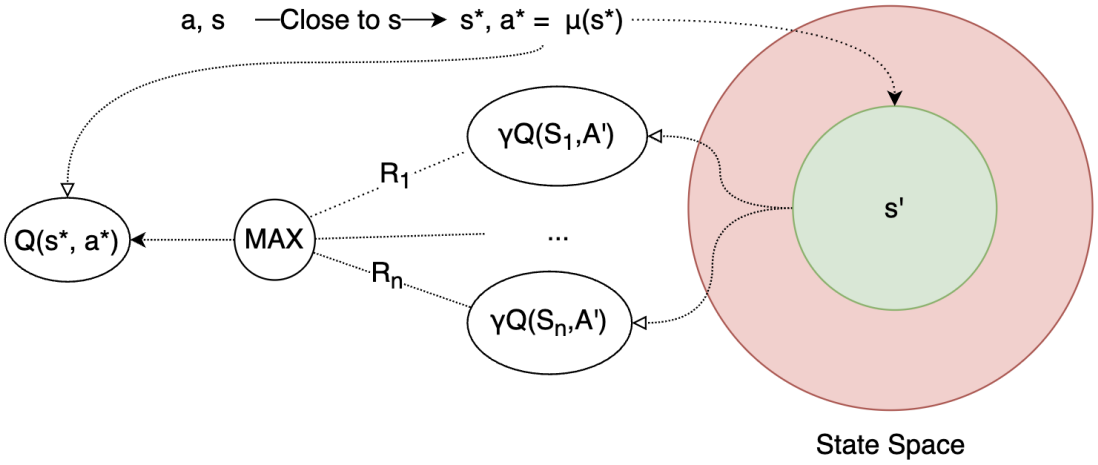


Figure 5.1: Zero-step roll-out: from each real state s we sample a neighbouring state $s^* \in \mathcal{N}_\varepsilon(s)$, draw n next-state candidates $s'_i \sim \hat{p}(s' \mid s^*, a^*)$, and form a one-step target by taking the maximum Bellman return.

from which we form a neighbour-target by taking the maximum over imagined outcomes:

$$y_k = \max_{1 \le i \le n} y_{k,i}.$$

This produces K distinct targets $y_1, \ldots, y_K$ and hence K TD-errors. We minimise the average squared error across these neighbours:

$$L_Q(\theta) = \frac{1}{K} \sum_{k=1}^{K} \big(Q_\theta(s^*_k, a^*_k) - y_k\big)^2,$$

followed by a standard policy gradient update to maximize $\mathbb{E}_{s \sim \mathbb{B}}[\, Q_\theta(s, \mu_\phi(s))]$ with respect to ϕ.

5.2 One–Step Update

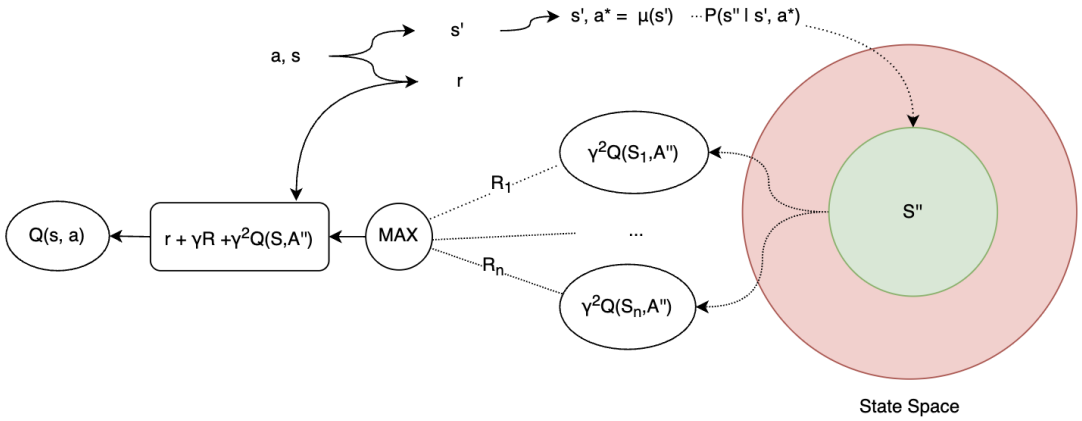


Figure 5.2: One-step roll-out: from each real transition (s, a, r, s') we draw n imagined successors $S_i \sim \hat{P}(\cdot \mid s', \mu(s'))$, roll each out one more step under the policy, and form a two-step Bellman return by taking the maximum over samples.

Figure 5.2 illustrates our simplified *one-step model-based update*, which does not sample neighbours of s but directly imagines multiple successors from the observed next state s'.

Given a real transition (s, a, r, s'), we sample n candidate successors

$$S_i \sim \hat{P}\big(\cdot \mid s',\, \mu_\phi(s')\big), \quad R_i = R\big(s',\, \mu_\phi(s'),\, S_i\big) \quad (i = 1, \ldots, n).$$

For each sample, let

$$A_i'' = \mu_\phi(S_i),$$

and compute the two-step return

$$y_i = r \;+\; \gamma\, R_i \;+\; \gamma^2\, Q_\theta\big(S_i, A_i''\big).$$

We then form a single optimistic target by taking the maximum over these n returns:

$$y = \max_{1 \le i \le n} y_i.$$

Training proceeds by minimising the squared error

$$L_Q(\theta) = \big(Q_\theta(s, a) - y\big)^2,$$

and then updating the policy via $\nabla_\phi \mathbb{E}_{s \sim \mathbb{B}}\big[Q_\theta(s, \mu_\phi(s))\big]$.

5.3 State Sampling Methods

For both the aforementioned methods, the next states are possibilistic imagined using either Quantile and Gaussian sampling methods as described here.

5.3.1 Quantile Networks

We train two networks, $q_\tau(s, a)$ and $q_{1-\tau}(s, a)$, to predict the τ and $1-\tau$ quantiles of the next-state distribution. Samples are drawn as

$$\hat{s}_i' = q_\tau(s, a) + \big(q_{1-\tau}(s, a) - q_\tau(s, a)\big)\, u_i, \quad u_i \sim \mathcal{U}(0, I).$$

Each quantile network minimises the standard pinball loss

$$L_\tau = \max\big((\tau - 1)(s' - q_\tau(s, a)),\; \tau(s' - q_\tau(s, a))\big),$$

via standard backpropagation. Here, all $\hat{s}'$ within the quantiles are treated as being fully possible.

5.3.2 Mean–Variance Network

A single network outputs $\mu(s, a)$ and $\log \sigma(s, a)$ for a Gaussian $\mathcal{N}(\mu, \sigma^2)$. We minimise the negative log-likelihood

$$L = \log \sigma(s, a) + \frac{\|s' - \mu(s, a)\|^2}{2\sigma(s, a)^2},$$

assuming diagonal covariance. Samples are generated by

$$\hat{s}' = \mu(s, a) + \sigma(s, a)\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Chapter 6

Experimental Setup

6.1 Algorithms

We evaluate four algorithms:

- **Mean–Variance Possibilistic DQN** (Alg. 1): a discrete-action Q-learning method, benchmarked against DQN.
- **Possibilistic Q-Ensembles** (Alg. 2): an ensemble-based discrete Q-learning approach, benchmarked against DQN.
- **Zero–Step Possibilistic Model-Based** (Alg. 3): a continuous-action actor–critic variant, benchmarked against DDPG.
- **One–Step Possibilistic Model-Based** (Alg. 4): a continuous-action actor–critic variant, benchmarked against DDPG.

6.2 Environments

We test each algorithm on both standard and custom sparse variants in discrete and continuous control.

6.2.1 Discrete Action Spaces

- **CartPole-v1**: reward of 1 for each timestep the pole remains upright.
- **StochasticCartPole**: adds Gaussian noise to transitions,

  $$s_{t+1} \sim \mathcal{N}\big(s'_{t+1}, 0.3^2 I\big),$$

  where s'_{t+1} is the standard next state.
- **LunarLander-v3**: dense shaping rewards for orientation, velocity, landing, and crashing.
- **SparseLunarLander**: only retains the horizontal/vertical orientation bonus and the terminal landing/crash reward; all other dense terms are removed.

6.2.2 Continuous Action Spaces

- **Walker2d-v5**: The agent is a bipedal robot, applying torques to various joints to keep the robot upright and moving.
- **SparseWalker2DEnv**: removes the dense per-step alive bonus of 1.0.
- **Hopper-v5**: A one-legged hopper that moves itself forward by changing the torques. The rewards are dense with some control bonuses and costs.
- **SparseHopper**: only forward-velocity reward is retained; the survival bonus and control cost are zeroed.

6.3 Implementation Details

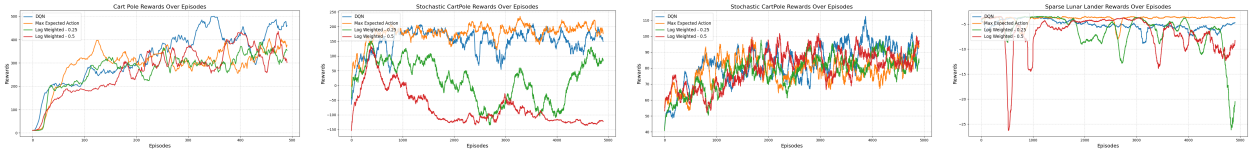
All experiments use Gymnasium environments and PyTorch implementations. We train each agent for 5000 episodes, with results averaged over 3 random seeds. CartPole-v1 is only tested for 1000 steps as almost all agents were able to converge to an optimum policy in that time.

- **General:** Batch size: 256; Hidden dimension: 128; Learning rate: 10^{-3}.
- **Discrete agents:** ϵ-greedy probability: 0.1; Training frequency: every 5 steps; Ensemble size: 5.
- **Continuous agents:** Roll-out planning frequency: every 50 steps; Training frequency: every 10 steps; Quantile bounds for uniform transition models: 0.1 and 0.9.
- **Hardware:** Apple M1 MacBook Pro (8-core CPU, 16 GB RAM; no GPU).

Chapter 7

Results and Discussion

7.1 Mean–Variance Possibilistic DQN



(a) CartPole-v1 (b) LunarLander-v3 (c) Stochastic Cart Pole (d) Sparse Lunar Lander

Figure 7.1: Mean-Variance Q-Network Performance

7.1 compares the performance of 4 different algorithms on the test environments.

- Classic Deep-Q-Learning algorithm
- Possibilistic Mean-Variance with Max Expected Action Selection
- Possibilistic Mean-Variance with Log-Variance-Weighted $\beta = 0.25$
- Possibilistic Mean-Variance with Log-Variance-Weighted $\beta = 0.5$

In the three deterministic environments (CartPole-v1, LunarLander-v3, Sparse Lunar Lander), the max-expected action performs the best—except in CartPole-v1. The mean-var network is able to model epistemic uncertainty and drive exploration in a systematic manner. In particular, the log-weighted exploration performs worse than max-expected and requires additional tuning of the parameter β, which can be an expensive process. Since there is no aleatoric uncertainty in these environments, the variance only captures the model's lack of knowledge and not any inherent randomness. The underperformance in CartPole is likely because the environment is too simple, and the emphasis on exploration wastes time, whereas a greedy algorithm performs better.

In Stochastic CartPole, all 4-algorithms have comparable performance, but DQN clearly performs better than the possibilistic variants. This suggests that the possibilistic networks conflate epistemic and aleatoric uncertainty. The possibilistic approaches over-explore actions with high-variance that are caused by noise in the environment rather than uncertainty caused to under-exploration of the state-action pair, leading to worse results. By greedily focusing on the expected return from the state-action pair, the DQN network is able to yield a better policy. Possibilistic Q-Ensembles is

introduced to better isolate epistemic uncertainty from aleatoric uncertainty.

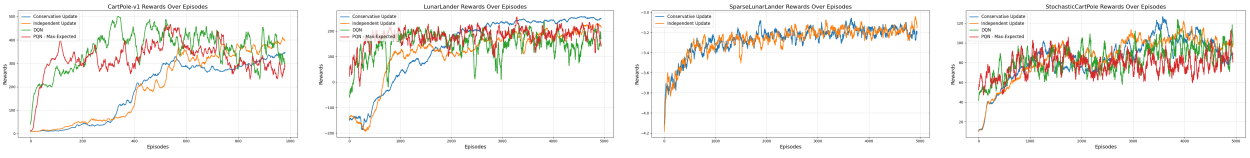
7.2 Possibilistic Q-Ensembles

We evaluated twelve variants of the Possibilistic Q-Ensemble across each environment. These variants differ in:

- **Bellman targets:** Independent vs. conservative (max–min) updates.
- **Action selection:** Maximum-expected-value vs. majority-voting.
- **Possibility updates:** Bayesian update vs. exponential-moving-average (EMA) Bayesian. No update baseline is also included where all networks remain fully possible

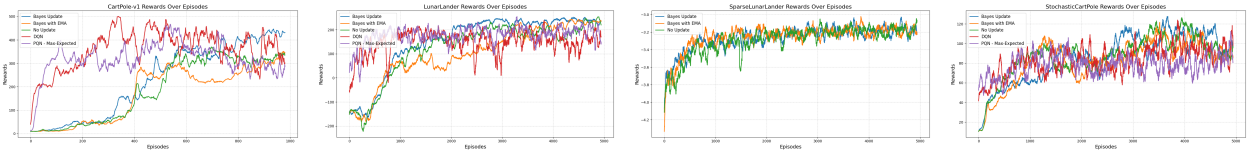
In total, each environment was run with twelve Possibilistic Q-Ensemble configurations (plus DQN). Raw results appear in Appendix B. DQN results are omitted for Sparse LunarLander, as its performance was substantially worse and obscures the comparison (see Appendix B).

Bellman Target Method



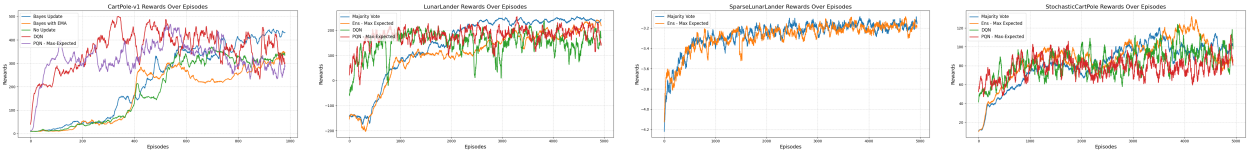
(a) CartPole-v1 (b) LunarLander-v3 (c) Sparse LunarLander (d) Stochastic CartPole

Possibility Update Method



(e) CartPole-v1 (f) LunarLander-v3 (g) Sparse LunarLander (h) Stochastic CartPole

Action Selection Method



(i) CartPole-v1 (j) LunarLander-v3 (k) Sparse LunarLander (l) Stochastic CartPole

Figure 7.2: Performance of across methodologies for Possibilistic Ensembles

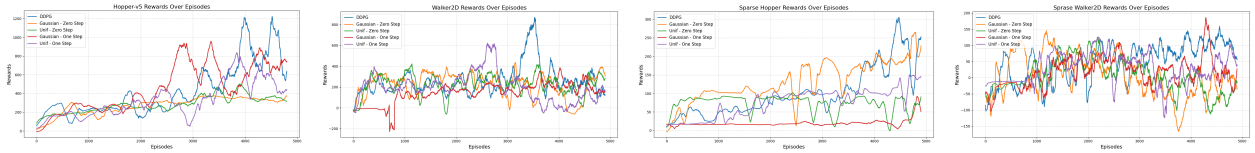
As a whole, the ensemble approach has less variance in cumulative rewards compared to the single-network methods in all environments. At the same time, they exhibit slower learning curves, as all networks in the ensemble must agree before the ensemble as a whole can converge on a policy — this can drive exploration. This is particularly beneficial in the sparse environment of Sparse Lunar Lander, where the ensemble drastically outperforms the single-network methods by enabling more effective exploration. Overall, the ensemble methods demonstrate slightly better asymptotic performance; even after 5000 episodes, the curves continue to trend upward. This trend is especially clear in Lunar Lander and Stochastic CartPole, where ensemble methods outperform single-network approaches and show continued improvement, while the single-network curves plateau.

Possibility updates seem to have some effect on performance. In particular, it can be observed that in CartPole and Lunar Lander, any update method performs better than no update, where the networks just have possibility 1 always. Updating possibilities acts like a filter: in CartPole it quickly isolates the strongest network, reducing ensemble disagreement, while in LunarLander the EMA's smoother weighting gives a slight performance edge. In the other tasks, possibility updates have less pronounced effects. The quick filtering of the possibilities (as evidenced in the average possibility over time plot in B) implies that only one single network remains possible whereas the possibilities of the remaining networks quickly tend to 0.

The choice of Bellman target (conservative vs. Independent) methods does not seem to have a massive consistent difference in performance. In CartPole, because the environment is simple, the independent, unstable yet greedy optimistic method seems to perform slightly better, whereas in more challenging environments the robustness offered by the conservative update seems to perform slightly better. In the other two environments, there is no substantial difference.

Both action selection methods seem to perform very similarly. This was expected because, in the case of the updates to the possibilities, when only one network remained with a high possibility of 1 while the rest went down to zero, both voting methods would select the same action, so there is no noticeable difference in performance here.

7.3 Possibilistic Model Based Learning



(a) Hopper-v5 (b) Walker2d-v5 (c) Sparse Hopper (d) Sparse Walker2d

Figure 7.3: Possibilistic Model-Based Algorithm Performance

In general, the possibilistic model based approaches underperform the DDPG baselines. We attribute the underperformance to two different factors: sensitivity to hyperparameters and inaccuracy in learned model.

The zero-step approach is particularly sensitive to the choice for neighbourhood sampling parameter ϵ. In particular, environment dynamics (such as stability of the hopper) and value functions for Hopper and Sparse Hopper are both sensitive to minor perturbations in the state. If ϵ is too large, then the predicted $Q(s^*, .)$ is likely to be overestimated; for small η, Q network values $Q(s + \eta, .)$ and $Q(s, .)$ similarly, however this might not be the case for Hopper due to non-linearity - so it is possible for s to be stable and $s + \eta$ to be catastrophic. Additionally, it is entirely possible for the agent to never experience s_0, where $||s_0 - (s + \eta)|| < ||\eta||$ and improve its value estimate. Empirically, in B , we compare two choices for ϵ: (0.01, 0.05), and $\epsilon = 0.01$ significantly overperforms $\epsilon = 0.05$, indicating a high sensitivity to this parameter. The figure 7.3 uses $\epsilon = 0.01$, and it performs well for Sparse Hopper. We expect that further tuning of the parameters could improve performance in all the environments.

We also think that the inaccuracy in the learned transition and reward model could be one of the reasons for the higher variance in the performance of model based approaches above. This could both be because of lack experience in the environment (and limited recorded transitions) and because of non-linearity in the environment leading to incorrect estimates by the neural-network. The network used in the above experiments are relatively small and it is possible that because of

that the model was able to properly capture the dynamics of the environment. The uncertainty in the model drives exploration but it also makes convergence to a greedy optimal policy slower, increasing the variance in the rewards. This is a challenge for both zero-step and one-step model based approach proposed before.

In 7.3 , we also observe that the performance of Gaussian and Uniform do not have consistent difference in performance. In Hopper, the Gaussian sampling seems to outperform Uniform sampling, this is likely because the Gaussian sampler more accurately samples the next state close to the actual next state and reduces the error caused by incorrect estimation of the values in underobserved states as explained before. However, there is no difference for performance in Walker2d as that environment is less sensitive to small perturbations in the state.

Chapter 8

Conclusion

This thesis introduces four novel algorithm types: Mean–Variance Possibilistic DQN, Possibilistic Q-Ensembles, and Possibilistic Model-Based Learning (incorporating both zero-step and one-step updates). These algorithms integrate possibility theory into various deep reinforcement learning frameworks to explicitly model epistemic uncertainty in value functions and systematically drive exploration using this uncertainty information. The overarching objective was to investigate whether the unique algebraic structures and measures of possibility theory could be leveraged to better model uncertainty and guide exploration effectively.

The first proposed approach was Mean-Variance based Possibilistic DQN, where uncertainty is modeled using a Gaussian possibility distribution. Experiments demonstrated that the tuning-free notion of maximum expected value outperformed other parameterized measures, such as log-variance weighting, as well as the classic DQN approach, particularly in environments lacking aleatoric uncertainty. However, because this method conflates aleatoric and epistemic uncertainty, the proposed algorithm underperformed DQN in stochastic environments, as it was unable to distinguish between variance arising from environmental randomness (aleatoric) and variance due to lack of knowledge (epistemic). This highlights a key limitation in its ability to differentiate between the sources of uncertainty.

To address this limitation, Possibilistic Q-Ensembles were proposed. This method models the possibility of a network being optimal based on its loss, which is then used to weight the network and guide action selection in a more uncertainty-aware manner. The ensemble approaches exhibited lower variance and better asymptotic performance compared to standard DQN and the previously proposed Possibilistic DQN. Due to the possibilistic weighting, this method effectively functions as a filter for various network initializations; in this context, possibilistic weighting outperformed common ensemble methods (where all networks retain a possibility of 1). The specific choices for action selection and update rules did not appear to significantly shape performance. It should be noted, however, that ensembles are more computationally demanding than single-network approaches.

Lastly, Possibilistic Model-Based Learning was explored, where state transitions are explicitly modeled possibilistically, and Q-values are updated optimistically. Both zero-step (neighbor sampling) and one-step (sampling from the next state) update methods were proposed and tested; both underperformed the DDPG baseline. Two primary challenges were identified: the performance of the model-based approach appears highly sensitive to hyper-parameters, and potential inaccuracies in the learned transition model can cause instability, particularly in highly nonlinear environments. While modeling uncertainty using the proposed maxitive optimistic method can drive exploration,

it may also hinder convergence to an optimal policy.

In summary, this work demonstrates that possibility theory can serve as a valuable tool for modeling uncertainty in reinforcement learning and can be effective in driving exploration and exploitation. Both Possibilistic Q-Ensembles and Possibilistic DQN show promise; however, Possibilistic Q-Ensembles appear to perform better, especially due to their ability to differentiate between epistemic and aleatoric uncertainty. Nevertheless, it was also noted that these approaches can be sensitive to hyper-parameters and may suffer from instability due to inaccuracies in possibility modeling, particularly in the model-based approaches, potentially introducing an additional layer of complexity.

Future work could explore several directions. Given the strong performance of Mean–Variance Possibilistic DQN and Possibilistic Q-Ensembles, maintaining an ensemble of mean–variance networks could be beneficial: the mean–variance network would model aleatoric uncertainty, while epistemic uncertainty could be captured by the ensemble. In the current ensemble approach, since many networks become impossible, dynamically resizing the ensemble could improve computational performance. Moreover, in the present mean–variance network the variance only moderated by the discount factor; future work could improve this by scaling the variance according to the visitation count of each state–action pair (for example, via a kernel-density estimate), thereby producing tighter and more realistic uncertainty bounds. The model-based approaches require better tuning, and additional hyperparameter optimization is necessary to fully evaluate these methods. Finally, these approaches should be tested in more complex environments to validate their utility. Overall, possibility theory offers a promising framework for modeling epistemic uncertainty in reinforcement learning, enabling more robust decision-making under uncertainty.

Bibliography

[Auer, 2003] Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3(null):397–422.

[Bellemare et al., 2017a] Bellemare, M. G., Dabney, W., and Munos, R. (2017a). A distributional perspective on reinforcement learning.

[Bellemare et al., 2017b] Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017b). The cramer distance as a solution to biased wasserstein gradients.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley & Sons.

[Dabney et al., 2017] Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2017). Distributional reinforcement learning with quantile regression.

[Dubois and Prade, 1992] Dubois, D. and Prade, H. (1992). When upper probabilities are possibility measures. *Fuzzy Sets and Systems*, 49(1):65–74.

[Dubois and Prade, 2001] Dubois, D. and Prade, H. (2001). Possibility theory, probability theory and multiple-valued logics: A clarification. *Ann. Math. Artif. Intell.*, 32:35–66.

[Dubois and Prade, 2007] Dubois, D. and Prade, H. (2007). Possibility theory. *Scholarpedia*, 2(10):2074. revision #137677.

[Dubois and Prade, 2013] Dubois, D. and Prade, H. (2013). Updating with belief functions, ordinal conditioning functions and possibility measures.

[Hans and Udluft, 2010] Hans, A. and Udluft, S. (2010). Ensembles of neural networks for robust reinforcement learning. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 401–406.

[Lan et al., 2021] Lan, Q., Pan, Y., Fyshe, A., and White, M. (2021). Maxmin q-learning: Controlling the estimation bias of q-learning.

[Liu et al., 2021] Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., and Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics*, 10(1):22.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare,

M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Silver et al., 2017] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm.

[Thomas and Houssineau, 2025] Thomas, J. and Houssineau, J. (2025). Possibilistic q-learning: Uncertainty modelling for tuning-free optimism. Unpublished manuscript.

[Watkins and Dayan, 1992] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[Zadeh, 1999] Zadeh, L. (1999). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34.

[Zhang et al., 2022] Zhang, S., Liu, B., and Whiteson, S. (2022). Mean-variance policy iteration for risk-averse reinforcement learning.

[Zhu et al., 2024] Zhu, T., Jin, Y., Houssineau, J., and Montana, G. (2024). Mitigating relative over-generalization in multi-agent reinforcement learning.

Appendix A

Algorithms

Algorithm 1 Mean–Variance Possibilistic DQN

Require: replay buffer $\mathcal{B}$, play–train interval l, target update interval C, batch size M, learning rate α, discount γ, update rate τ

Ensure: online network parameters θ, target network parameters $\theta^- \leftarrow \theta$

Initialize empty buffer $\mathcal{B}$
for $t = 1, 2, \ldots$ **do**
  Observe state s_t
  Select action $a_t = \arg\max_a \bar{Q}(s_t, a; \theta)$
  Execute a_t, observe (r_t, s_{t+1})
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
  if $t \bmod l = 0$ **then**
    Sample minibatch $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    for $j = 1$ to M **do**
      $a'_j \leftarrow \arg\max_{a'} \mu(s'_j, a'; \theta^-)$
      $\mu_j^{\text{tgt}} \leftarrow r_j \; + \; \gamma\, \mu(s'_j, a'_j; \theta^-)$
      $\sigma_j^{2,\text{tgt}} \leftarrow \gamma^2\, \sigma^2(s'_j, a'_j; \theta^-)$
      $\mathcal{T}_j \leftarrow \mathcal{N}(\, \mu_j^{\text{tgt}},\ \sigma_j^{2,\text{tgt}}\,)$
      $(\mu_j, \sigma_j^2) \leftarrow$ online_network$(s_j, a_j; \theta)$
      $L_j \leftarrow D_{\text{KL}}\big(\mathcal{N}(\mu_j, \sigma_j^2) \,\|\, \mathcal{T}_j\big)$
    end for
    $L \leftarrow \frac{1}{M} \sum_{j=1}^{M} L_j$
    $\theta \leftarrow \theta - \alpha\, \nabla_\theta L$
    $\theta^- \leftarrow \tau\, \theta + (1 - \tau)\, \theta^-$
  end if
end for

Algorithm 2 Possibilistic Q-Ensembles

Require: replay buffer $\mathcal{B}$, play–train interval l, batch size M, learning rate α, discount γ, smoothing factor α_p, ensemble size N

Ensure: online network params $\{\theta_i\}_{i=1}^{N}$, target params $\{\theta_i^-\}_{i=1}^{N}$, possibility weights $\{p_i\}_{i=1}^{N}$

Initialize θ_i, set $\theta_i^- \leftarrow \theta_i$, $p_i \leftarrow 1$, $\mathcal{B} \leftarrow \emptyset$
for $t = 1, 2, \ldots$ **do**
  Observe state s_t
  $\bar{Q}(a) \leftarrow \max_i \{\, p_i \, Q_i(s_t, a; \theta_i)\}$
  $a_t \leftarrow \arg\max_a \bar{Q}(a)$
  Execute a_t, observe (r_t, s_{t+1})
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
  if $t \bmod l = 0$ **then**
    Sample minibatch $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    for $i = 1$ to N **do**
      $y_{i,j} \leftarrow r_j + \gamma \, \max_{a'} Q_i(s'_j, a'; \theta_i^-) \quad \forall j$
      $L_i \leftarrow \frac{1}{M} \sum_{j=1}^{M} \big(Q_i(s_j, a_j; \theta_i) - y_{i,j}\big)^2$
      $\ell_i \leftarrow \exp(-L_i)$
    end for
    Update possibilities: $p_i \leftarrow \frac{p_i \, \ell_i}{\max_j \{p_j \, \ell_j\}}$, $p_i \leftarrow \max(\alpha_p \, p_i, \; p_i)$
    for $i = 1$ to N **do**
      $\theta_i \leftarrow \theta_i - \alpha \, \nabla_{\theta_i} L_i$
      $\theta_i^- \leftarrow \tau \, \theta_i + (1 - \tau) \, \theta_i^-$
    end for
  end if
end for

Algorithm 3 Zero–Step using Gaussian State distributions

Require: replay buffer $\mathcal{B}$, play–train interval l_{train}, planning interval l_{plan}, batch size M, neighbour count K, samples per neighbour n, discount γ, Polyak factor τ, critic LR α_Q, actor LR α_ϕ, state-model LR α_P, reward-model LR α_R, neighbourhood radius ε

Ensure: critic params θ, target critic $\theta^- \leftarrow \theta$, actor ϕ, state-model ψ, reward-model ω

Initialize $\mathcal{B} \leftarrow \emptyset$
for $t = 1, 2, \ldots$ **do**
  Observe s_t, select $a_t = \mu_\phi(s_t)$
  Execute a_t, observe (r_t, s_{t+1})
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
  if $t \bmod l_{\text{train}} = 0$ **then** ▷ Real-transition training
    Sample $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    for $j = 1$ to M **do**
      $(\mu_j^s, \sigma_j^s) \leftarrow f_\psi(s_j, a_j)$
      $L_j^P \leftarrow \log \sigma_j^s + \frac{\|s'_j - \mu_j^s\|^2}{2(\sigma_j^s)^2}$
      $\hat{r}_j \leftarrow g_\omega(s_j, a_j, s'_j)$
      $L_j^R \leftarrow (r_j - \hat{r}_j)^2$
      $y_j^{\text{real}} \leftarrow r_j + \gamma\, Q_{\theta^-}(s'_j, \mu_\phi(s'_j))$
      $L_j^Q \leftarrow \big(Q_\theta(s_j, a_j) - y_j^{\text{real}}\big)^2$
    end for
    $\psi \leftarrow \psi - \alpha_P \nabla_\psi \big(\frac{1}{M} \sum_j L_j^P\big)$
    $\omega \leftarrow \omega - \alpha_R \nabla_\omega \big(\frac{1}{M} \sum_j L_j^R\big)$
    $\theta \leftarrow \theta - \alpha_Q \nabla_\theta \big(\frac{1}{M} \sum_j L_j^Q\big)$
    $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \big(\frac{1}{M} \sum_j Q_\theta(s_j, \mu_\phi(s_j))\big)$
    $\theta^- \leftarrow \tau\, \theta + (1 - \tau)\, \theta^-$
  end if
  if $t \bmod l_{\text{plan}} = 0$ **then** ▷ Planning (imagined) step
    Sample $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    $J_{\text{plan}} \leftarrow 0$
    for $j = 1$ to M **do**
      for $k = 1$ to K **do**
        $s^*_{j,k} \sim \mathcal{N}_\varepsilon(s_j), \quad a^*_{j,k} \leftarrow \mu_\phi(s^*_{j,k})$
        $(\mu^s_{j,k}, \sigma^s_{j,k}) \leftarrow f_\psi(s^*_{j,k}, a^*_{j,k})$
        for $i = 1$ to n **do**
          $s'_{j,k,i} \sim \mathcal{N}(\mu^s_{j,k}, (\sigma^s_{j,k})^2)$
          $r_{j,k,i} \leftarrow g_\omega(s^*_{j,k}, a^*_{j,k}, s'_{j,k,i})$
          $y_{j,k,i} \leftarrow r_{j,k,i} + \gamma\, Q_{\theta^-}(s'_{j,k,i}, \mu_\phi(s'_{j,k,i}))$
        end for
        $y_{j,k} \leftarrow \max_{1 \le i \le n} y_{j,k,i}$
      end for
      $L_j^{\text{plan}} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \big(Q_\theta(s_j, a_j) - y_{j,k}\big)^2$
      $J_{\text{plan}} \leftarrow J_{\text{plan}} + L_j^{\text{plan}}$
    end for
    $J_{\text{plan}} \leftarrow \frac{1}{M} J_{\text{plan}}$
    $\theta \leftarrow \theta - \alpha_Q \nabla_\theta J_{\text{plan}}$
    $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \big(\frac{1}{M} \sum_j Q_\theta(s_j, \mu_\phi(s_j))\big)$
    $\theta^- \leftarrow \tau\, \theta + (1 - \tau)\, \theta^-$
  end if
end for

Algorithm 4 One–Step using Gaussian State distributions

Require: replay buffer $\mathcal{B}$, train interval l_{train}, planning interval l_{plan}, batch size M, samples n, discount γ, Polyak factor τ, critic LR α_Q, actor LR α_ϕ, state-model LR α_P, reward-model LR α_R

Ensure: critic params θ, target critic params $\theta^- \leftarrow \theta$, actor params ϕ, state-model params ψ, reward-model params ω

Initialize $\mathcal{B} \leftarrow \emptyset$
for $t = 1, 2, \ldots$ **do**
  Observe s_t, select $a_t = \mu_\phi(s_t)$
  Execute a_t, observe (r_t, s_{t+1})
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
  if $t \bmod l_{\text{train}} = 0$ **then** ▷ Real-transition training
    Sample $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    for $j = 1$ to M **do**
      $(\mu_j^s, \sigma_j^s) \leftarrow f_\psi(s_j, a_j)$
      $L_j^P \leftarrow \log \sigma_j^s + \frac{\|s'_j - \mu_j^s\|^2}{2(\sigma_j^s)^2}$
      $\hat{r}_j \leftarrow g_\omega(s_j, a_j, s'_j)$
      $L_j^R \leftarrow (r_j - \hat{r}_j)^2$
      $y_j^{\text{real}} \leftarrow r_j + \gamma\, Q_{\theta^-}(s'_j, \mu_\phi(s'_j))$
      $L_j^Q \leftarrow \big(Q_\theta(s_j, a_j) - y_j^{\text{real}}\big)^2$
    end for
    $\psi \leftarrow \psi - \alpha_P \nabla_\psi \big(\frac{1}{M} \sum_j L_j^P\big)$
    $\omega \leftarrow \omega - \alpha_R \nabla_\omega \big(\frac{1}{M} \sum_j L_j^R\big)$
    $\theta \leftarrow \theta - \alpha_Q \nabla_\theta \big(\frac{1}{M} \sum_j L_j^Q\big)$
    $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \big(\frac{1}{M} \sum_j Q_\theta(s_j, \mu_\phi(s_j))\big)$
    $\theta^- \leftarrow \tau\, \theta + (1 - \tau)\, \theta^-$
  end if
  if $t \bmod l_{\text{plan}} = 0$ **then** ▷ Planning step
    Sample $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^{M}$ from $\mathcal{B}$
    $J_{\text{plan}} \leftarrow 0$
    for $j = 1$ to M **do**
      $(\mu_j^s, \sigma_j^s) \leftarrow f_\psi(s'_j, \mu_\phi(s'_j))$
      for $i = 1$ to n **do**
        $S_{j,i} \sim \mathcal{N}(\mu_j^s, (\sigma_j^s)^2)$
        $R_{j,i} \leftarrow g_\omega(s'_j, \mu_\phi(s'_j), S_{j,i})$
        $A''_{j,i} \leftarrow \mu_\phi(S_{j,i})$
        $y_{j,i} \leftarrow r_j + \gamma\, R_{j,i} + \gamma^2\, Q_{\theta^-}(S_{j,i}, A''_{j,i})$
      end for
      $y_j \leftarrow \max_{1 \le i \le n} y_{j,i}$
      $L_j^{\text{plan}} \leftarrow \big(Q_\theta(s_j, a_j) - y_j\big)^2$
      $J_{\text{plan}} \leftarrow J_{\text{plan}} + L_j^{\text{plan}}$
    end for
    $J_{\text{plan}} \leftarrow \frac{1}{M} J_{\text{plan}}$
    $\theta \leftarrow \theta - \alpha_Q \nabla_\theta J_{\text{plan}}$
    $\phi \leftarrow \phi + \alpha_\phi \nabla_\phi \big(\frac{1}{M} \sum_j Q_\theta(s_j, \mu_\phi(s_j))\big)$
    $\theta^- \leftarrow \tau\, \theta + (1 - \tau)\, \theta^-$
  end if
end for

Appendix B

Extra Details

Lemma 1 (Closed Form Maximum of $q \cdot f(q)$ for Gaussian Possibility)**.** *Let $f(q)$ be a Gaussian-shaped possibility function:*

$$f(q) = \exp\left(-\frac{(q-\mu)^2}{2\sigma^2}\right)$$

Define

$$g(q) = q \cdot f(q) = q \cdot \exp\left(-\frac{(q-\mu)^2}{2\sigma^2}\right).$$

Then the supremum of $g(q)$ over $q \in \mathbb{R}$ is achieved at

$$q^* = \frac{\mu + \sqrt{\mu^2 + 4\sigma^2}}{2}.$$

Proof. We differentiate $g(q)$ with respect to q:

$$g(q) = q \cdot \exp\left(-\frac{(q-\mu)^2}{2\sigma^2}\right),$$

$$g'(q) = \exp\left(-\frac{(q-\mu)^2}{2\sigma^2}\right)\left(1 - \frac{q(q-\mu)}{\sigma^2}\right).$$

Setting $g'(q) = 0$, we solve

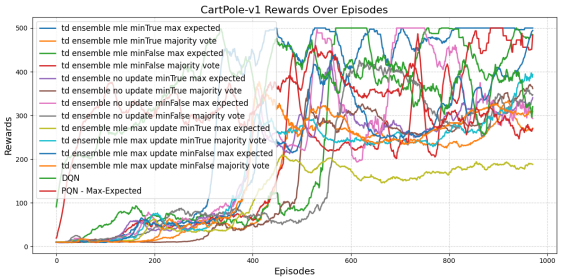
$$1 - \frac{q(q-\mu)}{\sigma^2} = 0 \quad \Rightarrow \quad q^2 - \mu q - \sigma^2 = 0.$$

Solving this quadratic equation yields

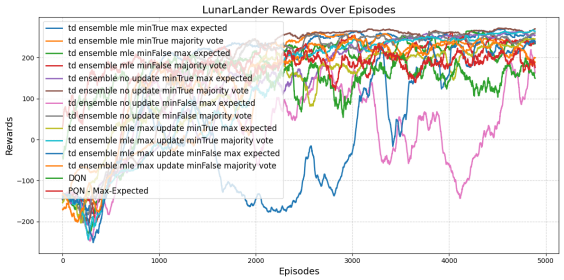
$$q^* = \frac{\mu + \sqrt{\mu^2 + 4\sigma^2}}{2}.$$

This is the unique maximizer of $g(q)$ over $\mathbb{R}$, as $g''(q) < 0$ at this point. □

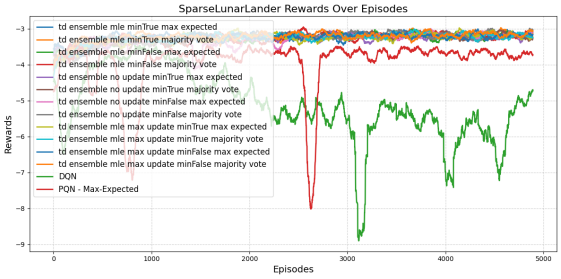
Raw Rewards for Possibilistic Ensembles



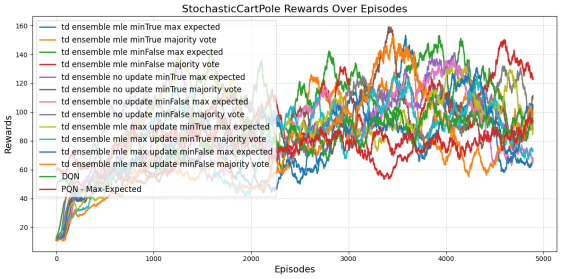
(a) CartPole-v1



(b) LunarLander-v3



(c) Sparse LunarLander



(d) Stochastic CartPole

Figure B.1: Raw reward curves for all four environments under the possibilistic Q-ensemble.

Average Ensemble Possibility: Bayesian Update vs. EMA Bayesian Update

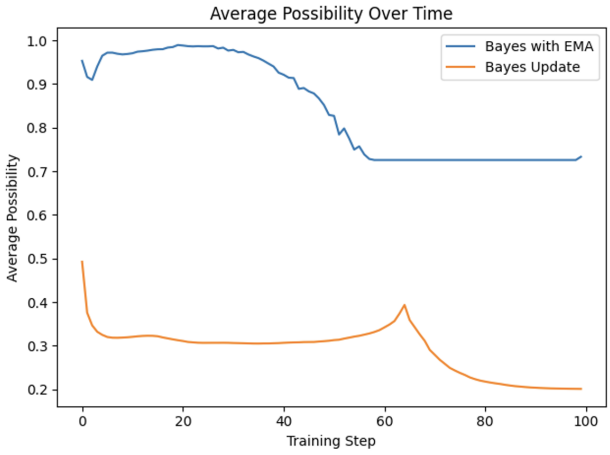


Figure B.2: Average possibility across ensemble networks under Bayesian update versus EMA Bayesian update. Environment: CartPole. Ensemble size: 5 networks. Action selection: maximum-expected value.

Kullback-Leibler Divergence

In [Cover and Thomas, 1991], the relative entropy (or Kullback-Leibler divergence) is introduced as the expected logarithm of the likelihood ratio. It measures the inefficiency of assuming that the distribution is q when the true distribution is p.

Discrete Case

For discrete probability mass functions $p(x)$ and $q(x)$, the KL divergence is defined as

$$D(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

Continuous Case

For continuous probability density functions $p(x)$ and $q(x)$, the KL divergence is defined as

$$D(p\|q) = \int p(x) \log \frac{p(x)}{q(x)}\, dx.$$

KL Divergence Between Two Gaussian Distributions

Consider two Gaussian distributions:

$$p(x) = \mathcal{N}(\mu_1, \sigma_1^2), \quad q(x) = \mathcal{N}(\mu_2, \sigma_2^2).$$

The KL divergence is given by

$$\begin{aligned} KL(p\|q) &= -\int p(x) \log q(x)\, dx + \int p(x) \log p(x)\, dx \\ &= \frac{1}{2} \log(2\pi\sigma_2^2) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}\Big(1 + \log(2\pi\sigma_1^2)\Big) \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \end{aligned}$$

Ensemble Disagreement: Conservative vs Independent Target

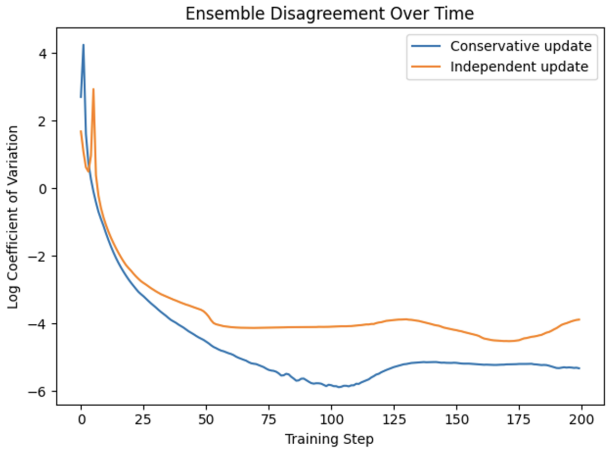


Figure B.3: Ensemble disagreement over time: conservative vs. independent update, without possibility updates, in the CartPole environment.

Possibilistic Atomic Q Learning

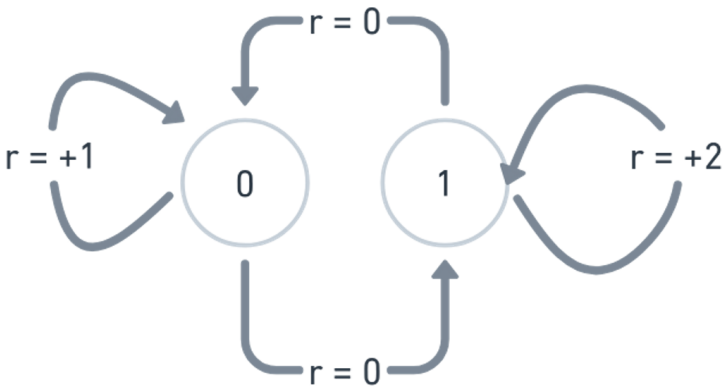
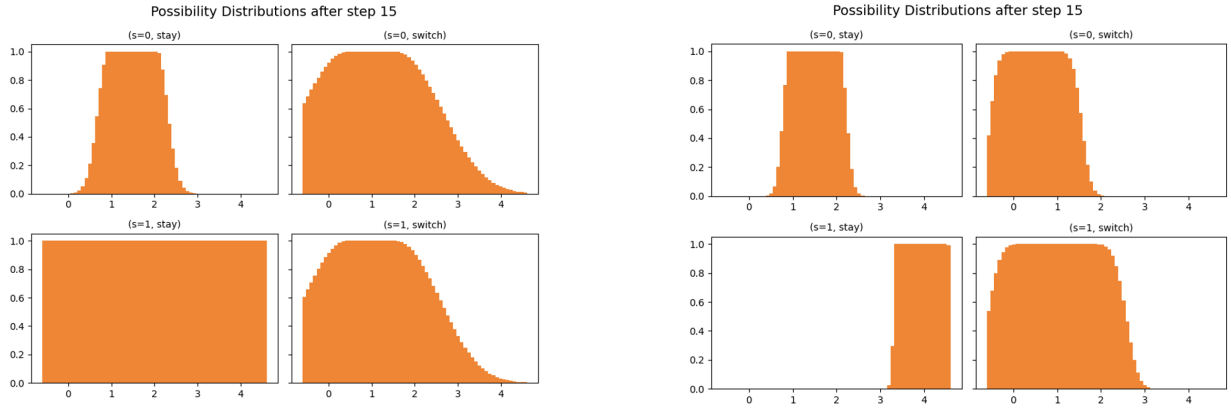


Figure B.4: Two-state test environment with discount rate $\gamma = 0.5$.

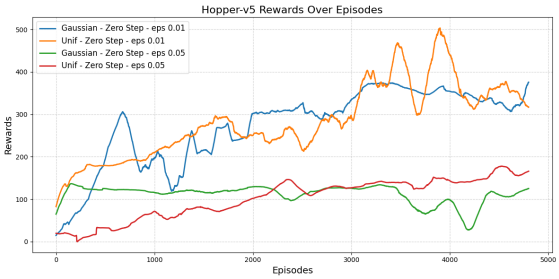
Here, we include performance of Possibilistic Atomic Q Learning presented in 3.1 in the environment as visualised in B.4. In particular, we highlight the sensitivity of the converges policy to the Gaussian-kernel similarity parameter σ. In particular, we look at the greedy policy where the agent always chooses the policy with the maximum expected value.



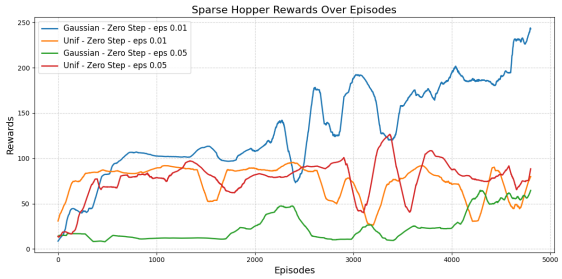
(a) $\sigma = 1$ (b) $\sigma = 0.2$

Figure B.5: Comparison for choice of σ in Possibilistic Atomic Q Learning.

Sample Radius Comparison for Zero Step Possibilistic Model Based Learning



(a) Hopper-v5



(b) Sparse Hopper

Figure B.6: Comparison for choice of ϵ in Zero Step Possibilistic Model Based Learning