Epistemic Uncertainty modelling using Possibility Theory in Reinforcement Learning

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

Acknowledgements

Contents

Chapter 1

Introduction

1.1 Motivation and Context

1.2 Overview of Proposed Methods

1.3 Main Contributions

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. In other words, possibility refers to the degree with which an event is possible given our current knowledge. This is in contrast to probability measures, where a probability of 1 implies that an event is statistically certain (or highly frequent), while a probability of 0.8 typically implies that the event happens with a frequency of 80%.

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:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\},$$

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* $\hat{\pi}(x)$ 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.,

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

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

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

This further implies that the union of two events is maxitive:

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

Note that this holds even if A and B are not disjoint. In contrast to probability measures, where the probability of the union of disjoint events is the sum of the probabilities, possibility measures satisfy

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

Dubois and Prade (2001) also introduced the notion of necessity, the dual of possibility, defined as

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

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

Need to standardise notation here with the rest of the paper. 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

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

This additive property reflects the quantitative nature of probability, where the total weight is distributed among all outcomes.

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

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

This property implies that if at least one event is highly possible, then their union is considered highly possible. It allows possibility theory to express complete ignorance by simply assigning a possibility of 1 to all outcomes without forcing a partition of numerical weights.

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:

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

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

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

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

2.1.4 Intersections and Unions: Conjunctions and Disjunctions

For independent events A and B, the probability of the joint event (the intersection) is typically given by the product:

$$P(A \cap B) = P(A) \cdot P(B).$$

Similarly, disjoint events have probabilities that add:

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

In contrast, possibility theory uses triangular norms (t-norms) to model the logical AND (conjunction) of events (DUBOIS and and (1982)). A common t-norm is the minimum operator, so that for events A and B with possibility distributions $\pi_A(x)$ and $\pi_B(x)$ respectively, the possibility distribution for the intersection is given by:

$$\pi_{A \cap B}(x) = \min\{\pi_A(x), \pi_B(x)\}.$$

This indicates that the possibility of a state satisfying both A and B is determined by the lesser possibility of the two. Dually, t-conorms (such as the maximum operator) are used for logical OR (disjunction):

$$\pi_{A \cup B}(x) = \max\{\pi_A(x), \pi_B(x)\}.$$

Check if this is correct. Thus, while probability theory uses multiplication (for independent events) and addition (for disjoint events), possibility theory replaces these operations with the minimum and maximum operators, respectively. This results in a very different arithmetic of uncertainty, which can simplify the handling of incomplete information.

2.1.5 Fuzzy Measures and Integrals

Possibility measures are a subset of fuzzy measures, which generalize classical measures by relaxing the requirement of additivity and requiring only monotonicity:

$$A \subseteq B \implies m(A) \leq m(B).$$

In possibility theory, rather than using the expected value computed via the Lebesgue integral, it is possible to aggregate outcomes using the Sugeno integral—a nonlinear operator based on the max and min operations. The Sugeno integral serves as an analogue to the Lebesgue integral and is particularly useful in qualitative decision-making scenarios where precise numeric integration is neither possible nor desired (Dubois and Prade (2015)).

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.

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.

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

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

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:

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

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. 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 *alpha* 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 dis 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. In general, it is possible to to approximate the Q-Values using other methods (such as Linear Functions). The method for doing Depp 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 Method, Stochastic Gradient Descent is used to update the Q networks, with the loss function derived from the temporal difference (TD) error. Particularly, the Bellman Backup is used as the target for training, i.e the update target y for the transition (s,a,r,s') 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 error between y and its own outputs:

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

The loss is calculated over a batch D which is a set of tuples (s,a,r,s') expericnced 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. Q-Learning is a type of off-policy learning, as the method does not directly learn the policy, rather it only learns an estimate of optimal Q-Value Q^*; this means that the Q-Values can be trained from any observed transitions in a given environment.

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, while also learning the value estimates to perform stable policy gradient updates.

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 graient of the objective function $J(\theta)$ 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 distrbuted by the policy π_θ. This means that the gradient of expected reward can be calculated as the sum of action values weighted by

$$\frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}$$

The parameter can then updated using

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

In practice, one uses the advantage function $A(s, a) = Q(s, a) - V(s)$ instead of the $Q(s, a)$ value directly as that increases stability in learning. It is shown that replacing the Q-Value with the advantage function does not introduce a bias.

The actor network is then updated with:

$$\Delta\theta_{\text{actor}} \propto \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t$$

where $\hat{a}$ is the empirical advantage.

The critic in actor critic environments is commonly the optimal state-value function approximate $V_w(s)$ or the optimal action-value approximation $Q_w(s, a)$. The critic can be trained on similar to Q-Learning using temporal difference (TD) learning. The state-value critic, for example, can be updated by minimising the error δ_t:

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

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 actor a is determined by the current policy π_θ; therefore δ_t itself is an estimator for the advantage function. Hence, the actor's gradient update can be implemented as

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

which nudges the policy to increase the probability of action a_t if the observed reward is greater than the expected value estimated by $V_w(s)$. In other words, the critic criticises the actions and the actor uses this to improve its policy.

It is also possible to have deterministic policies instead of stochastic policy described above. In the deterministic policy gradient approach (DPG) $a = \mu_\theta$ represnts the determinestic policy produced by the actor. The update to the actors policy is implemented as:

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

Here, the action-value function itself is the critic, and μ_θ is updated to maximise the Q-Value. The Q value is trained using the mini-batches, similar to in DQN.

2.2.4 Model-Based Reinforcement Learning

The algorithms discussed earlier are all model-free, in that, none of the methods maintain an internal ”model” of the environment. In contrast, there exists a category of algorithms that use and/or learn a model of the environment. This can include learning the state transitions $\hat{P}(s' \mid s, a)$ and the reward function $\hat{R}(s, a)$. The agent is able to plan with the environment model to evaluate its actions without enacting them in the real environment.

In MDP, if the true transition and reward functions are known, it is possible to infer the optimal policies by dynamic programming, this is also referred to as planning. However, in most applications, the exact P, R are unknown and the various Model Based approaches attempt to learn a model of environment through repeated interaction. The learned P, R can then be used for planning in the environment. A major benefit of model based approaches is the improved sample efficiency during training as the agent is able to simulate experience using the environment model instead of taking real actions. Model Based learning can be particularly beneficial if taking samples in the real environment is costly or limited.

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 $\hat{P}$ and $\hat{R}$. 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 help 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 create result in non-optimal policies. There are many different approaches to account for this including: shorter planning horizons, uncertainty bounds on the model and ensemble models.

2.3 Possibility Theory and Reinforcement Learning

Reinforcement Learning algorithm primarily deal with uncertainty in two different forms: aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty refers to the uncertainty in the environment because of the inherent randomness; this could include randomness in state transitions and stochastic rewards. Epistemic uncertainty, on the other hand, refers to the uncertainty in the environment because of our lack of knowledge/ information. Possibility Theory can be helpful to evaluate and make use of the latter.

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. (2017), 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 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. (2017) works by maintaining a probability distribution over 51 possible values $\{z_1, z_2, ..., Z_51\}$, which are placed uniformly over the range of returns. In particular, instead of the Q-Network outputting a single float value of state action pair, the network returns a probability tuple $(P(Z(s, a) = z_1), P(Z(s, a) = z_2), ..., P(Z(s, a) = z_51))$. 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.

2.3.2 Possibilistic Q Learning

Possibilistic Q Learning, as introduced by Thomas and Houssineau (2025), extends traditional Q-Learning by explicitly accounting for both aleatoric and epistemic uncertainty through the use of possibility theory. In this framework, epistemic uncertainty is modeled with possibility functions, allowing for a clear separation between uncertainty due to inherent randomness (aleatoric) and uncertainty due to lack of knowledge (epistemic).

A key concept in this approach is the *maximum expected value*, defined as

$$\bar{\mathbb{E}}\big(\Phi(\psi)\big) = \sup_{\psi \in \Psi} \{\Phi(\psi)\, f_\psi(\psi)\},$$

where f_ψ denotes the possibility function over the parameter ψ. This operator selects the highest weighted value of $\Phi(\psi)$, thereby capturing an optimistic estimate based on the current knowledge. Correspondingly, the most credible value of ψ is given by

$$\mathbb{E}^*(\psi) = \arg\max_{\psi \in \Psi} f_\psi(\psi).$$

Using these definitions, one can derive a recursive formulation for the maximum expected Q-value:

$$\bar{Q}(s,a) = \bar{\mathbb{E}}_{S'}\Big[\bar{r}(s,a,S') + \max_{a'} \bar{Q}(s',a' \mid S')\Big].$$

Here, $\bar{r}(s,a,S')$ represents the expected reward computed with the possibility-based model, and $\bar{Q}(s',a' \mid S')$ denotes the optimistic Q-value at the next state s'.

In this model, a possibility function $P(\cdot \mid s,a)$ is maintained over the state transitions. As training progresses, this possibility function is updated along with the maximum expected value, adapting the model's uncertainty estimates based on new observations. Since the initial Q-values in the tabular setting are set to an upper bound, the algorithm naturally favors exploration in states with high uncertainty. Over time, as more data is gathered, the optimistic estimates are refined toward the true Q-values.

Chapter 3

Possibilistic Q Values

Typically in Q-Learning based algorithm, 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). This strategy is unable to account for the aleatoric uncertainties in the environment and the models own epistemic uncertainty in the estimated Q values. By representing uncertainty, 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. Here we will present two different algorithms using possibility distributions over $Q(s, a)$.

3.1 Mean-Variance Networks

In this proposed possibilistic approach, instead of focusing on a single scalar for $Q(s, a)$, we evaluate a possibility distribution over the Q-values, denoted by $f(q \mid s, a)$. The distribution is parameterized by a mean and a variance; that is, the Q-network outputs two parameters for each state–action pair:

- **Mean** $\mu(s, a)$ — the expected value of the return (equivalent to $Q(s, a)$).
- **Variance** $\sigma^2(s, a)$ — which quantifies the uncertainty in the value of $\mu(s, a)$.

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

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

By definition, this membership function satisfies the normalization condition

$$\sup_q f(q \mid s, a) = 1,$$

which is achieved at $q = \mu(s, a)$. This aligns with the intuitive belief that the most credible value of the return is the mean.

3.1.1 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:

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

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.

3.1.2 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

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

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 A.

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.

3.1.3 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. Why is this a good way to choice, why not without the log - could be related to information theory (?). comment: This related to entropy regulirazation.

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):

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

We can try an ensemble of mean variance networks, where the $\sigma_i(s,a)$ can be viewed as being proportional to the certainity of that meanvar network for that (s,a) pair.

3.2 Atomic Q Values

TODO: implement the code for this. Discuss with prof

As discussed before, Bellemare et al. (2017) introduced distributional RL, which parameterizes a distribution over Q-values using fixed atomic values and assigns probabilities to these fixed atoms. This approach can potentially be enhanced using possibility theory—that is, by generating possibility values for the canonical atomic returns.

Let $f(q \mid s,a)$ be the mapping that assigns to each $q \in \mathbb{Q}$ (where $\mathbb{Q}$ is a finite ordered tuple of possible q values, e.g., $\{q_1, q_2, \ldots, q_N\}$) the possibility that q is the true Q-value for the state–action pair (s,a). An informed prior over the Q-values can be formed by setting

$$f(q \mid s,a) = 1 \quad \forall\, q \in \mathbb{Q},\ s,\ a,$$

i.e., initially all values of q are considered fully possible.

For a transition (s, a, r, s'), we can form a likelihood function by using a Gaussian kernel. Specifically, the likelihood of observing a reward r given that the true return is q_j is defined as:

$$p_R(r \mid q_j) = \exp\left(-\frac{(q_j - r)^2}{2\sigma^2}\right).$$

This likelihood is used to update the possibility function $f(q \mid s, a)$ via a possibilistic version of Bayes' rule. The updated possibility of the atom q_j is given by:

$$f(q_j \mid s, a) \leftarrow \frac{p_R(r \mid q_j)\, f(q_j \mid s, a)}{\sup\limits_{k \in \{1,2,\ldots,N\}} \big\{p_R(r \mid q_k)\, f(q_k \mid s, a)\big\}}.$$

Since $f(q_j \mid s, a)$ is initialized to 1 for all q_j, the first update simplifies to the ratio of likelihoods.

Note that by construction, for every s, a there exists at least one j such that

$$f(q_j \mid s, a) = 1,$$

satisfying the normalization condition in possibility theory.

In the discrete setting, the function $f(q_j \mid s, a)$ can be parameterized with a table. The optimistic Q-value can then be computed using the maximum expected value method:

$$\bar{Q}(s, a) = \sup_{j \in \{1,\ldots,N\}} \{q_j\, f(q_j \mid s, a)\}.$$

This means that initially, when the possibility scores are all 1, the maximum Q-value is chosen (i.e., all actions appear highly promising). As learning proceeds, the credibility of overly optimistic q_j values will decrease and $\bar{Q}(s, a)$ will favor the Q-values with high credibility.

This framework can be generalized to the deep learning case by using a deep neural network as a function approximate for $f(q_j \mid s, a)$.

A Bellman target distribution can be formulated as

$$\mathcal{T}Q = r + \gamma\, f(s', a'),$$

where $a' = \arg\max_a \bar{Q}(s', a)$. How to account for the shift here? For the deep learning update, the possibility distribution must be renormalized as follows:

$$f'(q_j \mid s, a) = \frac{f(q_j \mid s, a)}{\sup\limits_{q \in \mathbb{Q}} f(q \mid s, a)}.$$

A loss function such as the L_2 divergence loss can be used for gradient descent. For example, if $\hat{f}(q_j \mid s, a)$ denotes the target possibility distribution obtained via the Bellman backup, we can define the loss as:

$$L(s, a) = \sum_{j=1}^{N} \Big(f(q_j \mid s, a) - \hat{f}(q_j \mid s, a)\Big)^2.$$

Minimizing this loss updates the network parameters so that the predicted possibility distribution aligns with the target distribution.

Chapter 4

Possibilistic Q Ensembles

Epistemic uncertainty can be modeled by having a possibility distribution over Q-value functions. Practically, this can be achieved by training an ensemble of Q-networks, where each network represents a hypothesis of possible Q-functions. This approach can be interpreted as drawing samples from a posterior distribution of possible Q-values, thereby approximating the agent's uncertainty over the state–action values. In particular, the different networks should disagree on underexplored (s, a) pairs, indicating uncertainty about that region of the environment; conversely, if the networks generally agree, then the region is likely well explored. Ensembles have been shown to perform better than a single network because they can provide improved Q-value estimates and lead to more stable learning cite. In this work, we explore three different avenues:

- **Possibility Distribution:** A possibility distribution p_i is maintained for each network $Q_i \in \mathbb{QN}$, where $\mathbb{QN}$ denotes the set of Q-networks. The value p_i quantifies the model's belief in Q_i given the evidence so far. Initially, each network is assigned a possibility of 1 to represent an uninformed state. As learning progresses, we adjust the possibility of each network using its respective likelihood.

- **Mean-Var:** Mean Variance networks themselves provide a representation of uncertainty for each network in the ensemble. A high $\mu_i(s, a)$ represents a lower possibility/certainty in that network. This can allow us to train the network on independent data and still obtain valid uncertainty/belief values that are s, a-specific rather than global. We still need to explore this.

- **Just the Ensemble:** The ensemble, through its disagreement or agreement over Q-values, represents certainty or uncertainty. Still need to explore this. Probably should not go into this.

4.1 Possibility Distribution

4.1.1 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).

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, γ is the discount factor, and $\mathbb{A}$ is the set of actions. 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.

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

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

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 less underexplored states.

4.1.2 Bayesian Possibility Update

Loss-Based Likelihood

Using either of the updates above, the loss L_i for network i is computed as the mean squared error between the target y_i and the estimate $Q_i(s, a)$. The loss provides a likelihood estimate given by

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

GradNorm-Based Update

Check if this makes sense? The gradient norm

$$G_i = |\nabla_{\theta_i} L_i|$$

with respect to the training data indicates how "surprising" the data is for model i. Recent work Lee and AlRegib (2020) demonstrates the efficacy of gradient norms as a measure of a network's uncertainty. In particular, if a new observed transition results in a large change in the model's parameters θ_i, it may suggest that the model's prior knowledge of the state–action space was inadequate. We can incorporate the grad-norm by adding a normalized value to the loss:

$$L_i' = L_i + \beta \frac{G_i}{\max_{j \in \{1,\ldots,N\}} G_j},$$

Entroy can be mean surprisal. Surprisal - log. where the parameter β controls the impact of the gradient norm. We will provide comparisons over the different β values.

Possibility Update

Let L_i be the loss of network Q_i for a minibatch $\mathbb{B}$. The new possibility p_i for network i is estimated by

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

However, updating the possibility solely using this rule can result in instability, as the p_i may change drastically over updates. To ensure stability, we perform a possibilistic exponential moving average:

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

This procedure ensures that the possibility does not drastically decrease because of a single minibatch and that at least one p_i retains a possibility of 1.

4.1.3 Action Selection

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

Possibilistic Averaging

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

$$Q_{\text{ens}}(s, a) = \sum_{i=1}^{N} p_i\, Q_i(s, a)$$

and select the next action as

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

The value p_i represents the vote weight for the ensemble member Q_i. Averaging ensemble predictions is a common technique in reinforcement learning to improve Q-value estimates. Note that this action selection is equivalent to that which would be obtained if the possibility values were normalized to sum to 1. In general, we can consider this choice as a sample from the set of defuzzified ensemble Q-values. A more general formulation can be expressed as:

$$Q^{\alpha}_{\text{ens}}(s,a) = \frac{\sum_{i=1}^{N} p_i^{\alpha} Q_i(s,a)}{\sum_{i=1}^{N} p_i^{\alpha}},$$

where the parameter α can be used to control the “sharpness” with respect to the possibility values. This method can be sensitive to the individual Q-values; if one model has a very high Q-value, then that can skew the action selection. This is related to centriods maybe?, this needs to be motivated.

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 effectively ”clipped” at p_i and does not directly depend on the magnitude of $Q(s,a)$. 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.

4.2 Mean-Var Network

The estimated variance by each Mean-Var network in the ensemble can be used to infer the possibility of each network - with the idea that the uncertainity in $\mu_i(s,a)$ is proportional to $\sigma_i(s,a)$. p_i can be formulated as

$$p_i(s) = \exp(-\lambda \sup_k \{\sigma_i^2(s,a_j)\})$$

which can be normalised as

$$p_i(s) = \frac{p_i}{sup_j\{p_j\}}.$$

The same methods for action selection can be used, where the possibility value for each can be estimated directly from the $p_i(s)$ calculated for that observed state. The update of the Q-values can be done in the same way as done for the single Q Mean Var network in the previous chapter.

Chapter 5

MaxMax Q-Learning

MaxMax Q-Learning, introduced in Zhu et al. (2024), is a model-based actor critic algorithm designed to overcome relative over-generalization in cooperative multi-agent reinforcement learning. In multi-agent settings, agents learning together can undervalue joint optimal state-action pairs as they might rarely experience them, causing them to converge to less optimal actions. MMQ works by imagining many possible next states and considering the state transition that might lead to the best result; this helps the agents maintain optimism when uncertain.

Concretely, MMQ works by sampling possible next states $s'_1, \ldots, s'_k$ from a distribution over the state transitions $P(s' \mid s, a)$. This sampling is done using an internal model of the environment. In addition, the model learns a reward function $R(s, a)$ which maps state-action pairs to their expected reward. The agent then evaluates each candidate next state for the Bellman backup and update, and chooses the best next state for the update. In effect, the Bellman backup returns

$$Q(s,a) \leftarrow R(s,a) + \gamma \max_{i \in \{1,\ldots,k\}} \left[\max_{a'} Q(s'_i, a') \right].$$

The double maximization means that the agent imagines the next best possible state and assumes that outcome for the update. This update ensures that the estimate of the Q-value is high if any possible state transition leads to a positive result.

The Q-Values acts like the critic in the environment and the agent learns a policy π_{θ_i} by maximising on $Q(s, \pi_{\theta_i}(s))$. Rewards can also be learning as a supervised learning problem where the following mean-squared error for the transition (s, a, r, s') is minimised

$$(R(s, s') - r)^2.$$

Not Sure how this relates to possibility theory. Discuss with prof

5.1 State Sampling Methods

5.1.1 Quantile Networks

Here, the model learns to predict specific quantiles τ and $1-\tau$ over the next-state distribution from the state–action pair s, a; this is represented by the lower and upper bound networks $q_\tau(s, a)$ and $q_{1-\tau}(s, a)$ — forming a distribution over next possible states. The next state can be sampled as

$$\hat{s}'_i = q_\tau(s, a) + \Big(q_{1-\tau}(s, a) - q_\tau(s, a)\Big) \cdot u_i,$$

where u_i is sampled from the uniform distribution $u_i \sim \mathcal{U}(0, 1)$. This method samples a range of plausible next states from the learned quantiles.

The quantile networks can be trained using the quantile loss as done in Zhu et al. (2024). For quantile τ, the loss can be calculated as

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

The loss can be backpropagated using gradient descent to update the parameters of the quantile model q_τ.

5.1.2 Mean-Var Network

In this method, we use a neural network to model the Gaussian distribution

$$\mathcal{N}(\mu(s, a), \sigma^2(s, a))$$

from which the next possible state $\hat{s}'$ can be drawn. Practically, the network produces estimates for the mean of the next state $\mu(s, a)$ and the logarithm of the standard deviation, $\log(\sigma(s, a))$. As is common in the literature, the log of the standard deviation is used to improve numerical stability and ensure the variance remains positive.

The network can be trained to maximize the likelihood of observing the state s' from the state–action pair (s, a). This is achieved by minimizing the negative log likelihood (NLL) of the observation under the predicted distribution. The loss is

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

Here, we also assume a diagonal covariance matrix for the distribution of next states.

The imagined state can be sampled as

$$\hat{s}' = \mu(s, a) + \sigma(s, a) \cdot \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I)$.

Using the Mean-Variance method of sampling next possible states means that the agent is fully learning the environment by modeling both the state transitions and reward functions. As a result, we can also perform model-based roll-outs to obtain better reward estimates and improve sample efficiency. This possibilistic approach over next possible states allows us to account for the epistemic uncertainty in the next state transitions with this model-based approach. Is it possible to weigh the next sample using their possibility gotten from the Normal distribution, should we do an importance sampling approach (if we are sampling 5 we can $\mu - 1\sigma$, $\mu - 0.5\sigma$, μ, ...,). Maybe importance sampling can be done above as well.

Chapter 6

Experimental Setup

Chapter 7

Results and Discussion

Chapter 8

Conclusion

Bibliography

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning.

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

DUBOIS, D. and and, H. P. (1982). A class of fuzzy measures based on triangular norms a general framework for the combination of uncertain information. *International Journal of General Systems*, 8(1):43–61.

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

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

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

Dubois, D. and Prade, H. (2015). Possibility theory and its applications: Where do we stand? *Mathware and Soft Computing Magazine*, 18.

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

Lee, J. and AlRegib, G. (2020). Gradients as a measure of uncertainty in neural networks.

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.

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

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

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

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

Appendix A

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. □

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

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