University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

Bayesian Distributional Reinforcement Learning

Maxime Robeyns

May 2021

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

ABSTRACT

The main goal of this work is to provide an overview of the potentially fruitful use of Bayesian deep learning in distributional reinforcement learning.

Taking account of the various sources of uncertainty that may arise in the decision making process will form a central theme in this exposition. Distributional reinforcement learning stems from a relatively recent line of work proposing to take account of randomness in the environment, quantifying *aleatoric* uncertainty. A Bayesian treatment of these methods further allows us to account for uncertainty coming from lack of knowledge about the environment; in other words the *epistemic* uncertainty.

We will consider a new and flexible parametric approach to modelling arbitrary probability distributions, while quantifying the epistemic uncertainty in the parameters of the learned distribution.

Having quantified both the aleatoric and epistemic uncertainty in the returns gives occasion devise RL agents which not only reach the optimal policy, but do so while satisfying additional properties, such as sensitivity to risk, or performing exploration in a more directed and efficient manner.

We conclude with an overview of the additional properties which are afforeded by this framework, and employ our new modelling approach to demonstrate these properties in toy environments and computational studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

RL	Reinforcement Learning	
MDP	Markov Decision Process	5
KL	Kullback–Leibler	20
ELBO	evidence lower bound	20

NOTATION

$\mathcal{M}$	:	Markov Decision Process $\langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$
$\mathcal{X}$	:	State space of the MDP
$\mathcal{A}$	:	Action space of the MDP
R, R_t	:	Reward function, reward random variable
P	:	transition probabilities $P(x'\|x, a)$
$\gamma \in [0, 1)$	:	MDP discount factor
$x, x_t \in \mathcal{X}$	:	States, at time step t
$a, a^* \in \mathcal{A}$	:	Action, optimal action
π	:	Policy $\pi : \mathcal{X} \to \mathscr{P}(\mathcal{A})$
$\mathcal{T}^\pi, \tau$	:	Trajectory distribution, and sampled trajectory $\tau \sim \mathcal{T}^\pi$
$\mathcal{T}_\pi, \mathcal{T}$	:	Distributional Bellman operator and optimality operator
V^π, V	:	Value function, state value function $V : \mathcal{X} \to \mathbb{R}$
Q^π, Q	:	Action-value function, $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Z^π, Z	:	Random return, value distribution
Z^π_{MC}	:	Monte-Carlo value distribution under π
$\mathcal{Z}$	:	Space of value distributions
$\doteq$	:	an equality which is true by definition
$\mathbb{E}[X]$	:	expectation of the random variable X; i.e. $\mathbb{E}[X] \doteq \sum_x p(x)x$
$\mathscr{P}(\mathcal{X})$	:	a distribution over the set $\mathcal{X}$
$\mathbb{I}_p$	:	indicator random variable with value 1 if predicate p holds
δ_i	:	Dirac function at $i \in \mathbb{R}$
$(\Omega, \mathcal{F}, P)$	:	Probability space
$F_Y(y)$	:	CDF of a random variable Y, for some $y \in \mathbb{R}$
$F_Y^{-1}(\omega)$	:	Inverse CDF (quantile function), for some $\omega \in \Omega$

CONTEXTUAL BACKGROUND

When taking decisions, we are often just as concerned about what we don't know, as we are about what we do know. A chess player pays as much attention to what her opponent might do next, as she does to the current state of the board. The speed of a driver through a blind intersection will be informed not just from the cars he can see ahead of him, but also his uncertainty about hidden oncoming traffic. We constantly, perhaps subconciously, quantify our uncertainty and this forms a key piece of information which in combination with our appetite for taking risks informs our decision making process (Radford, 1989).

The field of Reinforcement Learning (RL) seeks to create *systems* capable of making decisions. If we are to create safe and rational decision making systems, then having the mechanisms to quantify the system's uncertainty in the task at hand, and rules to make use of this information seems incumbent. This theme of uncertainty in decision making is the main subject of this work.

1.1 REINFORCEMENT LEARNING AND UNCERTAINTY

Reinforcement learning originated from a combination of the fields of operations research (which provides the main algorithms and dynamic programming solution methods still used today) and trial-and-error learning (providing a connection to psychology, animal learning and the law of effect[1]). The latter is intimately linked to the concepts of uncertainty in decision making, and was first realised in the form of a machine in possibly the earliest work on neural networks by Minsky (1954)[2]. He made use of approximately 40 randomly connected vacuum tubes (representing Hebb synapses), each with a physical knob going between 0 and 1 showing the probability of a signal propagating. He called this machine a Stochastic Neural Analog Reinforcement Calculator (SNARC).

In a stark departure from the typically deterministic neural network function approximators in common use today, the SNARC perhaps more closely resembles a *Bayesian* neural network—where probability distributions are maintained over the weights of the networks. The vacuum tube capacitors would lose change over time in a relatively unpredictable manner, giving the SNARC its stochastic properties. One evaluation from the SNARC could be treated as a primitive way of sampling from the weights of a Bayesian network, and propagating a signal through the resulting network configuration.

Subsequent work on artificial neural networks was primarily developed in the context of supervised learning, where the link to animal learning was loosened, and computational constraints came to bear; making the interpretation of these methods as neural models increasingly questionable. They did however build an undeniable utility as general and effective function approximators. During this period, work in RL continued largely without neural networks, and it was only in 1989 with the introduction of Q-learning by Watkins (1989) that the use of neural networks for the purposes of learning control was made viable and was adopted widely.

By that point however, most connections to uncertainty in the decision making process were lost, although this did not seem to hinder progress. Reinforcement learning was used to great

1 Due to Edward Thorndike, the *law of effect* states that behaviours or '*responses*' that produce a pleasing effect in a particular situation are more likely to be repeated again when the situation next presents itself, and conversely behaviours leading to a negative effect are more likely to be avoided in that situation.

2 Concurrently, Farley and Clark (1954) worked a *simulation of self-organising systems by digital computer* along similar lines.

effect to build computer programs capable of playing backgammon (Tesauro, 1995), and steady innovations have culminated in very notable recent applications of RL to beat humans at Go, Chess and other board and computer games (Silver et al., 2016).

1.1.1 *Issues in Reinforcement Learning*

The results listed above risk overselling the current state of RL, which still lags far behind animal learning in many regards. One of these is the poor sample efficiency of many current RL algorithms, which may require on the order of millions of interactions with an environment before its learned behaviours match those of humans (Mnih et al., 2013). While this is acceptable for toy environments such as board games which lend themselves well to cheap and accurate simulation, many complicated real-world applications such as self-driving cars do not. Collecting experiences in these settings is expensive and sample inefficiency becomes prohibitive to the effective application of RL.

Most current RL methods also seem to lack a principled exploration strategy. These exploration strategies tackle the *exploration-exploitation* dilemma; which stipulates that a decision-making agent must choose whether to exploit its current understanding of the optimal behaviour, or to deviate from this, foregoing short-term rewards, and explore the environment which may lead to improved long-term performance. A commonly used heuristic, known as ϵ-greedy, will take a random action with probability ϵ, and keep to the known behaviour with probability $1 - \epsilon$. While this heuristic has been found sufficiently effective in a variety of contexts—and its simplicity ensures its continued widespread adoption—it fails to take into account the benefit that the agent stands to gain from an exploratory action, whether the exploratory action is being repeated, or what the risks of taking the action are.

This suggests another way in which current RL methods fall short of animal learning, namely that they generally don't estimate the risks posed by an exploratory action or behaviour before carrying it out. This can be important in contexts such as robotics where erratic exploratory actions pose a risk of physical damage, or in financial trade execution where the costs may not only be monetary but can have more widespread consequences.

1.1.2 *Methods*

Quantifying uncertainty may allow progress to be made in each of the three areas outlined above. We discern between two types of uncertainty. The first, *aleatoric* uncertainty, accounts for the irreducible uncertainty that is present in the environment. For instance in a game of poker, one cannot, even when counting cards, know for certain the contents of their opponents' hands—this uncertainty is aleatoric. The second, *epistemic* uncertainty, arises from a lack of knowledge about the system at hand and can be reduced by observing more data.

Mathematically, we articulate our uncertainty about a quantity of interest through the use of probability distributions. For example, in the context of RL the quantity of interest might be the output of the *value function*—which gives the expected sum of the numerical reward signal that an agent receives as it interacts with the environment.

Expressing our aleatoric uncertainty about the output of the value function amounts to defining a probability distribution over the returns, and this can be done straightforwardly using standard frequentist techniques. For instance, Morimura et al. (2010) demonstrate how fitting Gaussian, Laplace and asymmetric Laplace distributions over the returns using natural gradient descent can successfully represent the aleatoric uncertainty in the output of the value function. This uncertainty is well suited to quantifying risk, and the authors go on to apply risk metrics to the return distribution and demonstrate safer behaviours. In a similar line of work, Dabney

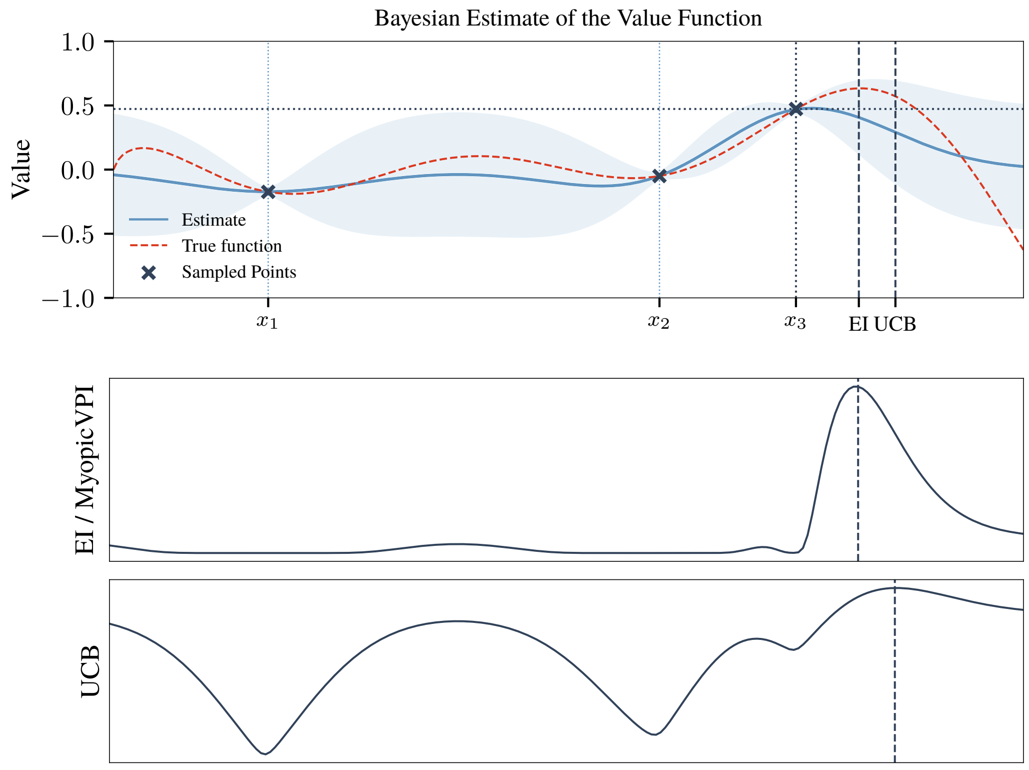


Figure 1.1: An example of acquisition functions—both mostly agree on the best next sampling point, with UCB notably remaining optimistic in the face of uncertainty, while EI places more credence on expected function value relative to the best sampled point.

et al. (2018) use quantile regression to estimate statistics of the return distribution, which again expresses the agent's aleatoric uncertainty. Interestingly, merely accounting for this uncertainty (and keeping the underlying algorithm fixed) already provide improvements over agents which eschew any uncertainty quantification, on common benchmark tasks.

Expressing the epistemic uncertainty can be done with a Bayesian estimate of the scalar-valued return, with the variance of this estimate tending to zero in the limit of infinite data. For example, Dearden et al. (1998) take a Bayesian approach to estimating the parameters of a Gaussian density over the returns, and apply this to the popular Q-learning algorithm. Capturing the epistemic (or *parametric*) uncertainty in the expected returns allows for many reasoned approaches to tackle the exploration-exploitation dilemma. These tend to share a common principle that by explicitly accounting for the agent's uncertainty in the expected rewards as probability distributions, we can control exploratory behaviour according to the probability that it is optimal or leads to an increase in useful information.

I have tried to illustrate this in Figure 1.1, where the top plot shows a hypothetical Bayesian estimate of a value function after making three observations (the x axis abstractly represents the agent's available actions). The bottom two plots show possible *acquisition* functions, which recommend where one should sample next in order to find the maximum of the dashed red value function, by trading off the expected value and uncertainty of the blue approximating function.

One of the methods explored in by Dearden et al. (1998) is the *myopic value of perfect information* (MyopicVPI), which approximates the expected utility of taking an exploratory action in terms of the expected improvement over the best known location (x_3 in Figure 1.1) that will result from the new information. The MyopicVPI closely resembles the *expected improvement* (EI) acquisition function used in Bayesian optimisation and bandit problems.

The exploration-exploitation problem also arises in these stochastic optimisation tasks, and we can appeal to the solution methods developed there for use in the RL setting. For instance, the *upper confidence bound* (UCB) is another simple acquisition function (shown on the bottom plot of Figure 1.1) which combines the current expected value (exploitation) with the current level of uncertainty about the actual value (exploration) to select a new input point according to the probability that it is optimal. This acquisition function is simpler yet remains more optimistic in the face of uncertainty; consider for instance the points between x_1 and x_2.

The epistemic and aleatoric uncertainties are not mutually exclusive however, and often co-exist in reinforcement learning problems. To this end we may try to model both simultaneously, which provides the motivation for a *Bayesian* approach to *distributional reinforcement learning*—the subject of this work.

The epistemic uncertainty in this context corresponds to uncertainty about what the correct return distribution is. If we use a sufficiently flexible parametric distribution for the return density, then a Bayesian estimate of the distribution's parameters would allow us to represent the epistemic uncertainty. The resulting Bayesian estimate may be thought of as a distribution over distributions.

These two uncertainties are therefore distinct and they give rise to different insights. Modelling the aleatoric uncertainty is well suited to evaluating measures of risk

// TODO finish introduction

2 TECHNICAL BACKGROUND

This chapter contains two main sections. In the first, we consider reinforcement learning, covering the necessary background before discussing more recent advances in *distributional* reinforcement learning. We then treat Bayesian modelling in the second half, discussing the key principles and giving a short comparison of approximation methods which are often required to make Bayesian inference tractable.

2.1 REINFORCEMENT LEARNING

Reinforcement learning is the subfield of machine learning concerned with learning to control a system, with the goal of maximising a long-term numerical reward signal. It distinguishes itself from supervised learning in that it falls upon the system to collect experiences (data) to learn from. The feedback signals are relatively weak in comparison to traditional supervised learning, which often makes *sample efficiency* an important objective in RL, beyond simply maximising the reward signal accrued over time through interactions with the environment.

Figure 2.1 shows the time-marching interaction of the learning system or *agent* with its environment. We will restrict our attention to settings with discrete timesteps and countable action sets. This need not be excessively restrictive: we can discretise continuous interactions between the agent and the environment (for instance in robotics or self-driving cars) to a fixed number of interactions every second, and continuous action spaces can be approximated with a finite set of actions without suffering too high an approximation cost.

At each time step $t = 1, 2, \ldots$, the agent observes a (potentially incomplete) representation of the current state of the environment x_t, and selects an action a_t to perform in the environment. The agent subsequently observes a new state x_{t+1}, along with a scalar-valued reward $r_t \in \mathbb{R}$, which we may think of as a measure of the instantaneous value of taking the previous action. The agent's goal is to maximise a discounted sum of these rewards over time.

This process is formalised as a Markov Decision Process (MDP).

Definition 2.1 (Markov Decision Process)**.** An MDP $\mathcal{M}$ is a tuple $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$, where

- $\mathcal{X}$ denotes a (possibly infinite) set of states which the agent can observe
- $\mathcal{A}$ is the set of actions which the agent can take in each state (here we only consider finite or *discrete* sets of actions),
- P denotes the state transition probability. If the agent is in state $x_t = x$, and takes action $a_t = a$, then the probability that it transitions to state $x_{t+1} = x'$ is given by $P(x'|x, a)$, where $x, x' \in \mathcal{X}$, and $a \in \mathcal{A}$.
- $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which gives the immediate reward received when action a is taken in state x, $R(x, a)$. We denote the random variable for rewards received at timestep t as $R_t \sim R(X_t, A_t)$.
- $\gamma \in [0, 1)$ is a discount factor; this is usually set close to 1 and its role will be explained later, finally

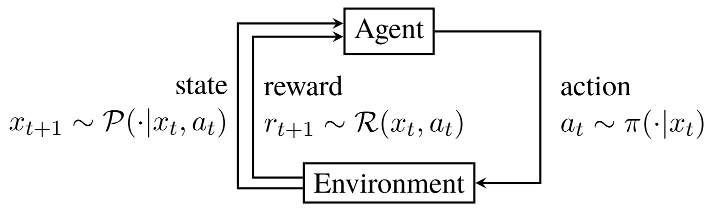

Agent
state
$x_{t+1} \sim \mathcal{P}(\cdot|x_t, a_t)$
reward
$r_{t+1} \sim \mathcal{R}(x_t, a_t)$
action
$a_t \sim \pi(\cdot|x_t)$
Environment

Figure 2.1: In one timestep, the agent takes an action in the environment, whereupon it observes a new state and reward, before beginning the next timestep.

Note that there may be slight variations on the MDP as defined above which add elements to the 5-tuple. For instance, a game usually has a fixed starting configuration, and we can express this as an initial state distribution P_0 (i.e. $x_0 \sim P_0$). In *episodic* MDPs, the episode might terminate upon reaching a terminating state $x \in \mathcal{X}^+$, whereupon the agent would receive a final reward and go back to a start state.

A *policy* is used to describe the way in which an agent behaves in an environment.

Definition 2.2 (Policy)**.** A (stochastic stationary) *policy*, $\pi(a|x) \doteq \mathrm{P}(a_t = a|x_t = x)$ maps each state $x \in \mathcal{X}$ to a distribution over $\mathcal{A}$. Intuitively this gives us the probability that the agent selects action a in state x. We denote by Π the set of all such stationary stochastic policies.

The states that the agent visits, and consequently the data that the learning system has access to, are dictated not just by the transition dynamics of the environment $P(\cdot|x, a)$, which lie outside the agent's control, but also the actions that the agent chooses to take, $\pi(\cdot|x)$. From this perspective, the agent's policy induces a new transition probability P^π for the MDP, which we can define as:

$$P^\pi(\cdot|x) \doteq \sum_{a \in \mathcal{A}} \pi(a|x) P(\cdot|x, a). \tag{2.1}$$

Going forward we will use this superscript notation to denote other objects which are affected by the agent's policy.

Definition 2.3 (Trajectory)**.** The tuple $(x_t, a_t, r_{t+1}, x_{t+1})$ is called a *transition* and summarises a single timestep (a single loop in Figure 2.1). Multiple transitions stringed together are called a *roll-out*, and the full list of transitions from the start of an episode to termination at timestep T is called a *trajectory*

$$\tau = (x_0, a_0, r_1, x_1, a_1, r_2, \ldots). \tag{2.2}$$

Fixing a policy induces a *trajectory distribution*, and we denote samples of trajectories from this distribution as $\tau \sim \mathcal{T}^\pi$. The likelihood of a trajectory is:

$$\mathrm{P}(\tau|\pi) = \prod_{t=0}^{T} P(x_{t+1}|x_t, a_t) \pi(a_t|x_t). \tag{2.3}$$

Note that for simplicity, we make the common independence assumption for all the terms in the above.

The agent's policy also affects the sum of rewards that it accrues during a trajectory as it interacts with the environment. Note that these are additionally subject to the randomness of the environment transition dynamics. The sum over these rewards forms the feedback signal which the agent seeks to maximise, and we refer to this sum as the *return*.

Definition 2.4 (Returns)**.** For some fixed policy π, the *return* Z_t is a random variable defined as the sum of discounted future rewards R_t observed along a trajectory. The following definitions are used interchangeably

$$Z_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{2.4}$$

$$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.5}$$

$$= R_{t+1} + \gamma Z_{t+1}. \tag{2.6}$$

We will also find it convenient to write $Z^\pi(x)$ for the returns induced by policy π starting at state x, and more generally $Z^\pi(x, a)$ when the initial action is further specified:

$$Z^\pi(x, a) \doteq \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t), \tag{2.7}$$

The discount factor above, $\gamma \in [0, 1)$ first serves the pragmatic role of bounding the sum above to ensure that the returns are a mapping from states (and actions) to the reals—having infinite returns would make it difficult to evaluate different policies, and would void the *contraction assumption* which is the subject of the next section.

Since the goal of an agent is to find the policy π which leads to the largest returns, we can also understand γ as regulating a trade-off between obtaining myopic policies, which aim to maximise short-term gain (potentially at the expense of long-term performance) and policies which target large predicted future rewards (potentially foregoing short term rewards, or even incurring short-term losses). A small value of γ discounts future rewards very strongly, which will encourage more short-term policies, while and a larger γ would encourage longer-term policies. Note that in all cases the discounting is exponential.

The expected value of the returns gives rise to the *value functions*, which are traditionally used to evaluate the 'quality' of a policy.

Definition 2.5 (State value function)**.** A *state value function* $v^\pi : \mathcal{X} \to \mathbb{R}$ is a mapping from a state $x \in \mathcal{X}$ to the expected returns that the agent will receive by starting in state x and following policy π thereafter:

$$v^\pi(x) \doteq \mathbb{E}_{\tau \sim \mathcal{T}^\pi} \big[Z^\pi(x)\big] = \mathbb{E}_{\tau \sim \mathcal{T}^\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \mid x_0 = x \right]. \tag{2.8}$$

The expectation above is taken over the trajectory distribution induced by the policy—we could equivalently have taken the expectation over $x_t \sim P(\cdot|x_{t-1}, a_{t-1})$ and $a_t \sim \pi(\cdot|x_{t-1})$. Intuitively, the value function expresses a notion of how good it is to be in state x while following policy π. If both the state and action are specified, we obtain the closely related action-value function, giving the value of taking action a in state x, and subsequently following policy π.

Definition 2.6 (Action value function)**.** The *action value function* (also called the *Q-function*), $q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a function taking a state-action pair and returning the expected returns:

$$q^\pi(x, a) \doteq \mathbb{E}_{\tau \sim \mathcal{T}^\pi} \big[Z^\pi(x, a)\big] = \mathbb{E}_{\tau \sim \mathcal{T}^\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) | x_0 = x, a_0 = a \right]. \tag{2.9}$$

This will become the basis for the *Q-learning* algorithm, discussed later. The random variables V, Q will be used to denote the expected returns.

2.1.1 *Learning Control in MDPs*

With the general goal of maximising expected returns, the agent is intent on finding the *optimal policy* for the MDP $\mathcal{M}$ at hand; that is, an optimal decision making rule such that the returns generated by any other policy $\pi' \neq \pi^*$ are no greater than the returns generated by the optimal policy $Z^{\pi'} \leq Z^{\pi^*}$. More formally, the set of optimal policies is defined as

$$\Pi^* \doteq \max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \mathcal{T}^\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \right]. \tag{2.10}$$

By expressing $\pi^* \in \Pi^*$ as a member of a set, we emphasise that there may not be one single optimal policy, but rather a set of equally good policies—finding any one of these will suffice for solving the RL problem. We may similarly define an optimal value function $v^*(x) \doteq \sup_{\pi \in \Pi} v^\pi(x)$ and optimal action-value function $q^*(x,a) \doteq \sup_{\pi \in \Pi} q^\pi(x,a)$.

There are many approaches to learning an optimal policy. Here, we will restrict our focus to the class of *model-free*, value-based methods; that is, methods where the agent does not have a model of the transition dynamics $P(\cdot|x,a)$, and where the agent learns an approximation of the value function from which it derives a policy. This is done by iteratively updating the agent's approximation of the value function using experience gathered from the environment. This modern approach using function approximation methods is derived from the classical dynamic programming methods, originating in operations research.

2.1.1.1 *Dynamic Programming*

Dynamic programming can be used as a way of planning in an MDP; both for predicting the value function v^π and finding an optimal policy π^*, which we can use for control. In this section we assume that the transition dynamics of the environment $P(\cdot|x,a)$ are somehow known to the agent—in the next section we will remove this assumption.

The reader may be familiar with dynamic programming as a technique for algorithm design, however in this context it refers to a method for mathematical optimisation. The term carries broadly the same intuitions in the two settings—exploiting the optimal sub-structure of the problem to derive an optimal solution, and relying on the fact that the subproblems overlap (a state may be visited multiple times). An MDP satisfies both the optimal sub-structure and overlapping subproblems properties.

In 1952, Richard Bellman observed that for sequential decision making tasks, "*the problem of determining an optimal sequence of operations may be reduced to that of determining an optimal first operation*" (Bellman, 1952). This is formalised in the following theorem.

Theorem 2.1 (Principle of Optimality)**.** A policy $\pi(a|x)$ achieves the optimal value from some state x; $v^\pi(x) = v^*(x)$ iff for any state x' reachable from x, π achieves the optimal value from state x' too: $v^\pi(x') = v^*(x')$.

We will first consider how dynamic programming is used to evaluate the value function induced by the current policy π. The principle of optimality can then be applied to this DP solution, requiring only a slight modification, and gives us a way of finding the optimal policy.

Definition 2.7 (Bellman evaluation operator)**.** Let V denote the set of all value functions. The *Bellman evaluation operator* is the mapping $\mathcal{T}_\pi : (\mathcal{X} \to \mathbb{R}) \to (\mathcal{X} \to \mathbb{R}) \times V$, defined as

$$(\mathcal{T}_\pi v)(x) \doteq R(x, \pi(x)) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x, \pi(x)) \cdot v(x'). \tag{2.11}$$

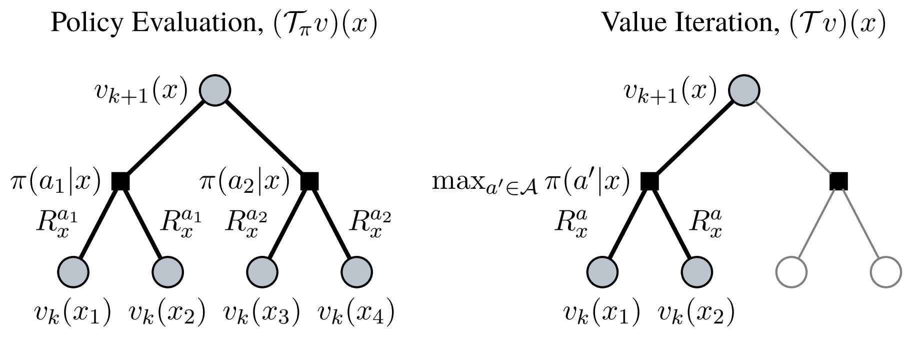

Policy Evaluation, $(\mathcal{T}_\pi v)(x)$
Value Iteration, $(\mathcal{T} v)(x)$
$v_{k+1}(x)$
$\pi(a_1|x)$
$\pi(a_2|x)$
$R_x^{a_1}$
$R_x^{a_1}$
$R_x^{a_2}$
$R_x^{a_2}$
$v_k(x_1)$ $v_k(x_2)$ $v_k(x_3)$ $v_k(x_4)$
$v_{k+1}(x)$
$\max_{a' \in \mathcal{A}} \pi(a'|x)$
R_x^a
R_x^a
$v_k(x_1)$ $v_k(x_2)$

Figure 2.2: A single application of the Bellman evaluation operator (left) and optimality operator (right). For compactness, the notation R_x^a is used to denote $R(x, a)$. Round vertices are states, and square vertices denote actions.

Consider a complete metric space (V, d) over value functions, where the distance metric is the ∞-norm, defined as the largest difference between state values:

$$d(u, v) \doteq \|u - v\|_\infty = \sup_{x \in \mathcal{X}} |u(x) - v(x)\|. \tag{2.12}$$

Lemma 2.1 (Contraction of $\mathcal{T}_\pi$)**.** The evaluation operator $\mathcal{T}_\pi$ is a γ-contraction in the metric space (V, d), whose unique fixed point is v^π.

A proof of the above is given in the appendix. This means that repeated applications of $\mathcal{T}_\pi$ to an arbitrary initial value function v will converge to v^π at a linear rate of γ:

$$\lim_{n \to \infty} (\mathcal{T}_\pi^n v)(x) = v^\pi(x). \tag{2.13}$$

In practice, we use the *Bellman expectation equation* to perform these updates, which can be seen as a rearrangement of the value function given in Equation 2.8:

$$\begin{aligned} v^\pi(x) &= \mathbb{E}_{\substack{a_t \sim \pi \\ x_t \sim P^\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \mid x_0 = x \right] \\ &= \mathbb{E}_{\substack{a_t \sim \pi \\ x_t \sim P^\pi}} \left[R(x, a_0) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} R(x_t, a_t) \mid x_0 = x \right] \\ &= \sum_{a \in \mathcal{A}} \pi(a|x) \left[R(x, a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x, a) \cdot v^\pi(x') \right] \end{aligned} \tag{2.14}$$

Therefore in this dynamic programming setting, where we assume the environment dynamics P are known and we have an explicit policy π, we would iteratively update the value function as

$$v_{k+1}(x) = \sum_{a \in \mathcal{A}} \pi(a|x) \left[R(x, a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x, a) v_k(x') \right], \tag{2.15}$$

and this will eventually converge to v^π.

This process is called *policy evaluation*, and the diagram on the left of Figure 2.2 illustrates a single application of Equation 2.15. Starting in a state x, we compute the sum of the immediate reward received by taking an action in $\{a_1, a_2\}$ and the discounted value of transitioning to feasible subsequent state $\{x_1, \ldots, x_4\}$, as found by evaluating the old value function, $v_k(x_i)$.

We then average over all these values, weighting each by the probability that action a_i was taken under π.

Policy evaluation allows us to obtain a value function for the current policy π, however we are interested in obtaining the *optimal* policy π^*, by finding the optimal value function. The following modification of the Bellman evaluation operator allows us to acheive this.

Definition 2.8 (Bellman optimality operator)**.** The *Bellman optimality operator* is also a mapping between value functions, $\mathcal{T} : (\mathcal{X} \to \mathbb{R}) \to (\mathcal{X} \to \mathbb{R})$, and is defined as

$$(\mathcal{T}v)(x) \doteq \max_{a\in\mathcal{A}} \left[R(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) \cdot v(x') \right]. \tag{2.16}$$

Lemma 2.2. If the Bellman evaluation operator $\mathcal{T}_\pi$ is a contraction mapping, then $\mathcal{T}$ is also a contraction mapping, with v^* as its unique fixed point.

Once again, a proof of this is provided in the appendix. We may use this result to find the optimal value function v^* in the dynamic programming setting, in a procedure is called *value iteration*. This consists of iteratively applying the Bellman optimality operator to an arbitrary initial value function. If the conditions under which the Bellman optimality operator is a contraction mapping are satisfied, then we are guaranteed to converge at a rate of γ, to v^* given sufficient iterations.

Explicitly, the update rule, called the *Bellman optimality equation*, is as follows:

$$v^*_{k+1}(x) = \max_{a\in\mathcal{A}} \left[R(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) \cdot v^*_k(x') \right], \tag{2.17}$$

and an example of its application is shown on the right side of Figure 2.2.

In the dynamic programming setting, we can recover the optimal policy π^* by taking the action which leads to the subsequent state x' with the highest value, as determined by $v^*(x')$. Since we are only focusing on *model free* RL methods in this work, (where we don't have access to $P(x'|x,a)$), we will now remove the assumption that the environment dynamics are known, with a method for model free control called Q-learning.

2.1.1.2 *Model Free Control and TD*

An optimal policy can be found by maximising over the optimal action-value function,

$$\pi^*(a|x) = \begin{cases} 1 & \text{if } a = \arg\max_{a\in\mathcal{A}} q^*(x,a) \\ 0 & \text{otherwise.} \end{cases} \tag{2.18}$$

Recall the relationship between the value function and action-value function:

$$q^\pi(x,a) = R(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) v^\pi(x'). \tag{2.19}$$

All the contraction-mapping results and Bellman equations from the previous section applied to the state-value function have direct analogues when applied to the Q-function. In particular, the Bellman optimality equation applied to the Q-function is

$$\begin{aligned} q^*_{k+1}(x,a) &= R(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) v^*(x') \quad (2.20) \\ &= R(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) \max_{a'\in\mathcal{A}} q^*(x',a'). \quad (2.21) \end{aligned}$$

However, this update rule still requires that the agent has knowledge of the environment dynamics in order to calculate the expectation over subsequent states. Instead, we may use a sampling-based method called *temporal difference learning* (occasionally *asynchronous value iteration*), where we only update the approximation of the Q-function for a single visited state-action pair.

Definition 2.9 (Temporal difference learning)**.** *Temporal Difference* TD learning is a model-free update rule which bootstraps the current value-function estimate and a sampled transition $(x_t, a_t, r_{t+1}, x_{t+1})$ to perform an update:

$$\begin{aligned} q^*_{k+1}(x,a) &= \begin{cases} (1-\alpha_t)q^*_k(x,a) + \alpha_t\big[r_{t+1} + \gamma \max_{a'\in\mathcal{A}} q^*_k(x_{t+1},a')\big] & \text{if } x = x_t, a = a_t \\ q^*_k(x,a) & \text{otherwise} \end{cases} \\ &= \begin{cases} q^*_k(x,a) + \alpha_t\big[r_{t+1} + \gamma \max_{a'\in\mathcal{A}} q^*_k(x_{t+1},a') - q^*_k(x,a)\big] & \text{if } x = x_t, a = a_t \\ q^*_k(x,a) & \text{otherwise} \end{cases} \end{aligned} \tag{2.22}$$

where α_t is an annealed learning rate or step size. The *temporal difference* δ_{TD} is the difference between the previous value estimate for the sampled state-action pair $q^*_k(x_t, a_t)$, and the marginally more accurate updated value estimate $r_{t+1} + \gamma \max_{a'\in\mathcal{A}} q^*_k(x_{t+1}, a')$. By bootstrapping the current estimate of the Q-function in this way, the agent can learn from incomplete episodes of experience—intuitively updating a guess towards a slightly improved guess.

2.1.2 *Deep Q Learning*

TODO

2.1.3 *Distributional Reinforcement Learning*

The key quantity in our exposition of model-free RL so far has been the value functions, which summarise the discounted sum of all the future rewards as a single scalar value: the expected returns $\mathbb{E}[Z_t]$. Distributional RL (Bellemare et al., 2017) argues for modelling the entire distribution over the random returns received by the agent, Z.

The motivation for doing so lies in the many sources of randomness that the agent must contend with. Stochasticity in the received reward R_t, the transition dynamics P and hence the next state-action pair (x_{t+1}, a_{t+1}), as well as the random return $Z(x_{t+1}, a_{t+1})$ all influence the distribution in returns at the current state. By averaging over these quantities and working with only the expectation of the returns, the agent discards valuable information gained from potentially costly interactions with the environment. Retaining this information in the return distribution can be seen as bringing model-free RL slightly closer to model-based RL.

Just as the expected return was defined recursively by the Bellman equation (2.11), an analagous *distributional* Bellman equation can be defined for distributions over returns (**?**Bellemare et al., 2017):

$$Z(x,a) \overset{D}{=} R(x,a) + \gamma Z(x', a'), \tag{2.23}$$

where the $A \overset{D}{=} B$ notation indicates that the random variable A is distributed according to the same probability law as B.

We cannot simply assume that the TD methods derived in the previous section from the appliation of the Bellman optimality equation will work in the distributional setting. In particular, we must first verify that the *distributional Bellman operator* (and indeed, the *distributional Bellman optimality operator*) remains a contraction mapping, whose fixed point is Z^π (respectively, Z^*).

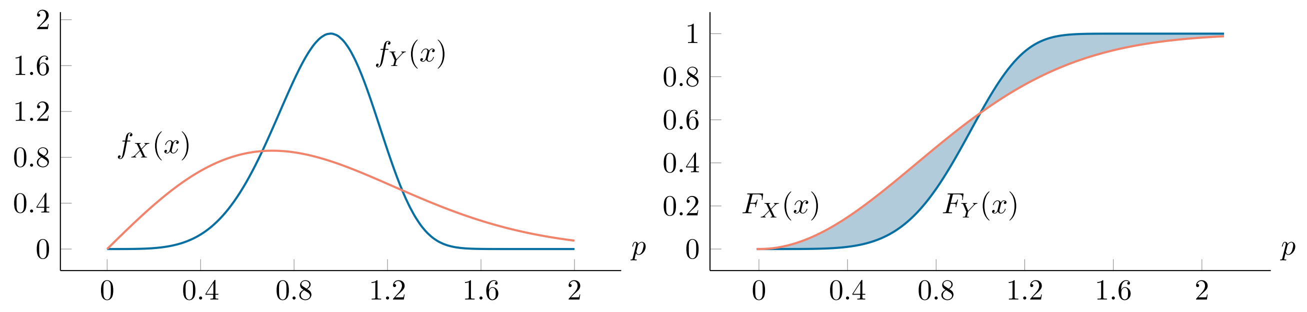


Figure 2.3: An illustration of the 1-Wasserstein distance $d_1(X,Y)$, found as the area between the two random variables' distribution functions.

We will work in the complete metric space $(\mathcal{Z}, \bar{d}_p)$. Here $\mathcal{Z}$ denotes the set of all return distributions with bounded moments (i.e. for all x, a pairs, $\mathbb{E}[Z^\pi(x,a)]^p \leq \infty$).

The distance metric must be defined in the space of return distributions $\mathcal{Z}$; that is, mappings from state-action pairs to distributions over the reals $Z^\pi(x,a) \in \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathbb{R})$. The popular KL divergence is unsuitable in this case due to its asymmetry—it is not a distance metric. Instead, Bellemare et al. (2017) propose the use of the *Wasserstein* distance between CDFs.

Definition 2.10 (Wasserstein distance)**.** For $1 \leq p \leq \infty$ and two random variables X, Y on a continuous domain, with finite moments and distribution functions F_X and F_Y, respectively, the *Wasserstein distance* $d_p(X,Y)$ is defined as

$$d_p(X,Y) \doteq \left(\int_0^1 |F_X^{-1}(\omega) - F_Y^{-1}(\omega)|^p d\omega \right)^{1/p}. \tag{2.24}$$

$F^{-1}(\cdot)$ denotes the inverse CDF or *quantile function*. In particular,

$$d_\infty(X,Y) \doteq \sup_{\omega \in [0,1]} |F_X^{-1}(\omega) - F_Y^{-1}(\omega)|, \tag{2.25}$$

which simply corresponds to the maximal distance between a point $\omega \in [0,1]$ on the quantile function of X and Y. The Wasserstein distance between two distributions is a metric.

To build some intuition for the properties of this distance metric, the 1-Wasserstein $d_1(X,Y)$ is simply the area between the two CDFs (as illustrated in Figure 2.3). In this case, the Wasserstein distance is equivalent to the *earth mover's distance* (EMD): if the two density functions $f_X(x)$ and $f_Y(x)$ are thought of as piles of earth, the EMD is the least amount of 'earth' that must be displaced to transform the first into the second (this is always possible; both distributions have unit mass).

The maximal form of the Wasserstein metric between two return distributions is:

$$\bar{d}_p(Z_1, Z_2) \doteq \sup_{\substack{x \in \mathcal{X} \\ a \in \mathcal{A}}} d_p\big(Z_1(x,a), Z_2(x,a)\big), \tag{2.26}$$

and this is the metric used to define the complete metric space of return distributions $(\mathcal{Z}, \bar{d}_p)$.

To devise a similar point iteration method to that used in the previous section for policy evaluation and control, we must first consider the convergence properties of the distributional Bellman operators.

Definition 2.11 (Distributional Bellman operator)**.** The *distributional Bellman operator*[1] is a mapping between value distributions $\mathcal{T}_\pi : (\mathcal{X} \times \mathcal{A} \to \mathbb{R}) \to (\mathcal{X} \times \mathcal{A} \to \mathbb{R})$ defined as

$$(\mathcal{T}_\pi Z)(x,a) \overset{\text{D}}{=} R(x,a) + \gamma Z(X', A'), \tag{2.27}$$

where $X' \sim P(\cdot|x,a)$ and $A' \sim \pi(\cdot|X')$.

Lemma 2.3 (Bellemare et al. (2017))**.** $\mathcal{T}_\pi : \mathcal{Z} \to \mathcal{Z}$ is a γ-contraction in $\bar{d}_p$.

By Banach's fixed point theorem, $\mathcal{T}_\pi$ has a unique fixed point in $(\mathcal{Z}, \bar{d}_p)$, and further this fixed point is the return distribution Z^π; that is, for any initial Z, $\lim_{n\to\infty} \mathcal{T}_\pi^n Z_0 = Z^\pi$. We may therefore use the distributional bellman operator to perform policy evaluation through a point iteration method in the distributional setting, just as in the expected value setting.

We now consider whether the same is true for control in the distributional setting, where we seek a method to find a policy π^* which maximises returns, and also achieves the distributional equivalent by reaching the *optimal value distribution*.

Definition 2.12 (Optimal value distribution; OVD)**.** This is the value distribution of an optimal policy. Recall that Π^* was defined in Equation 2.10 as the set of optimal policies; the set of optimal value distributions is defined as

$$\mathcal{Z}^* \doteq \{Z^{\pi^*} : \pi^* \in \Pi^*\}. \tag{2.28}$$

While there are potentially many optimal policies $|\Pi^*| > 1$, in the expected value setting this is inconsequential since they all reach the same expected value V^* and we may treat them as being equivalent. However, in the distributional setting, there may be many *distinct* OVDs—each pertaining to a different optimal policy and each having different moments. Crucially, not all value distributions with expectation V^* are optimal; they must match the full distribution of the return under some optimal policy.

Definition 2.13 (Distributional Bellman optimality equation)**.** The optimality operator in the distributional setting is one implementing a greedy action selection rule:

$$(\mathcal{T} Z)(x,a) \doteq R(x,a) + \gamma Z\left(X', \arg\max_{a' \in \mathcal{A}} \mathbb{E}[Z(X', a')]\right). \tag{2.29}$$

We first note the somewhat reassuring fact that this remains a contraction mapping for the expected returns.

Lemma 2.4 (Bellemare et al. (2017))**.** For any two $Z_1, Z_2 \in \mathcal{Z}$, then

$$||\mathbb{E}[\mathcal{T} Z_1] - \mathbb{E}[\mathcal{T} Z_2]||_\infty \leq \gamma ||\mathbb{E}[Z_1] - \mathbb{E}[Z_2]||_\infty, \tag{2.30}$$

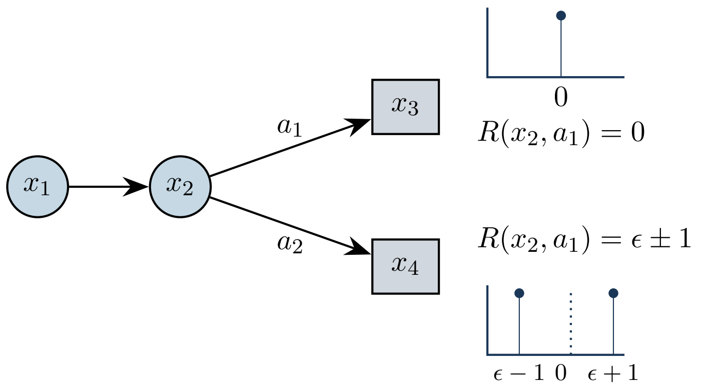
where $\mathbb{E}[\lim_{n\to\infty} \mathcal{T}^n Z] = V^*$.

Therefore point-iteration methods from the expected value setting should perform no worse when applied in the distributional setting. This is a slightly underwhelming result however, which is not improved when we consider that while the mean of the return distribution converges to V^*, the rest of the distribution need not converge to any specific distribution under $\mathcal{T}$.

In general, the operator $\mathcal{T}$ is not a contraction. To see this, consider the example in Figure 2.4.

There is a unique transition from state x_1 to state x_2. Taking action a_1 leads to the terminal state x_3 where no reward is received. Taking action a_2 from x_2 leads to terminal state x_4 and a

1 We re-use the notation $\mathcal{T}_\pi$ here, hoping that it is clear from the context whether it refers to the distributional Bellman operator or the Bellman operator in the expected value setting.


x_1
x_2
a_1
a_2
x_3
x_4
0
$R(x_2, a_1) = 0$
$R(x_2, a_1) = \epsilon \pm 1$
$\epsilon - 1$ 0 $\epsilon + 1$

	x_1	x_2, a_2	x_2, a_2
Z^*	$\epsilon \pm 1$	0	$\epsilon \pm 1$
Z	$\epsilon \pm 1$	0	$-\epsilon \pm 1$
$\mathcal{T}Z$	0	0	$\epsilon \pm 1$

Figure 2.4: Counter example: the operator $\mathcal{T}$ is not a contraction (adapted from Bellemare et al. (2017)).

reward of $\epsilon \pm 1$ is received with equal probability. This action is optimal since, in expectation, $\mathbb{E}[R(x_2, a_1)] > 0$; hence there is a unique optimal policy

$$\pi^*(a_1|x_2) = 0$$
$$\pi^*(a_2|x_2) = 1,$$

and $\mathcal{T}$ has a unique fixed point in this case. (Note that the optimality operator $\mathcal{T}$ need not have a fixed point—indeed, this would be the case if $\epsilon = 0$.)

Suppose the current estimate of the return distribution, Z was as given in the second row of the table in Figure 2.4. In particular, $\mathbb{E}[Z(x_2, a_1)] > \mathbb{E}[Z(x_2, a_2)]$. When we apply $\mathcal{T}$ to Z, the greedy action will be selected, giving $\mathcal{T}Z(x_1) = Z(x_2, a_1)$, and by calculating the maximum 1-Wasserstein distance for all the entries in the table, we find that

$$\bar{d}_1(\mathcal{T}Z, \mathcal{T}Z^*) > \bar{d}_1(Z, Z^*). \tag{2.31}$$

The distributional Bellman optimality operator is therefore not a contraction.

The implications of this are unclear: for algorithms merely utilising the expected values of the return distributions, the result from Lemma 2.4 should provide sufficient convergence guarantees. The empirical performance of distributional RL algorithms often outperforms that of expected-value methods by a wide margin (**?**Yang et al. (2020)), suggesting that there may be other benefits to modelling the return distributions.

However for approaches looking to utilise the full distribution over the returns (for instance the location of the modes, or some other moments of the distribution) this negative result may suggest that this is ill advised.

We will consider some common approaches to modelling the value distribution Z^π in Chapter 3, before discussing the implementation and performance of distribution RL algorithms in Chapter 4.

2.2 BAYESIAN MODELLING

We now shift the focus of this background chapter from reinforcement learning, to the second theme of this work; probabilistic machine learning and Bayesian inference.

Making inferences is arguably not just about applying the single most likely explanation that we can think of, but also taking into consideration alternative feasible explanations. To entirely discard these competing explanations on the count that they are less likely than the current most

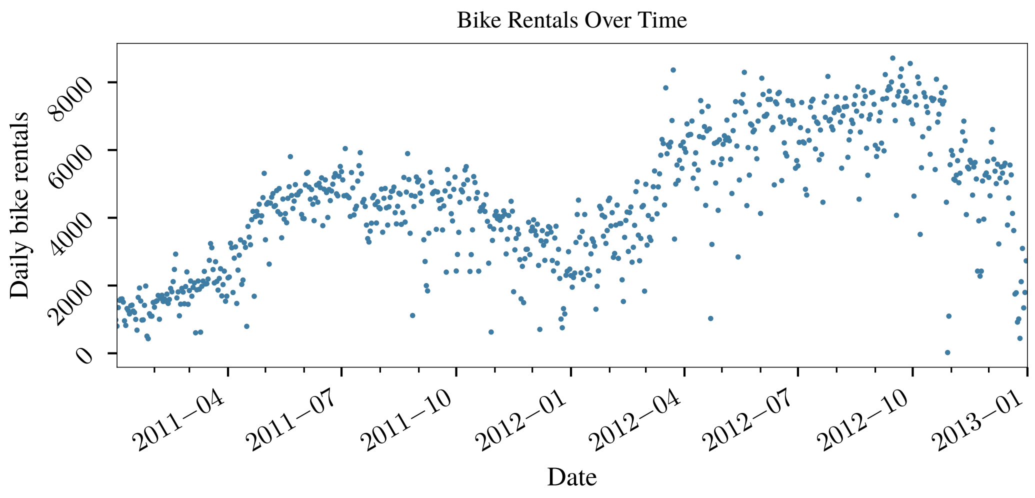


Figure 2.5: Should a simple, or complex model be used to predict the number of bike rentals?

firmly held belief—perhaps only marginally so—seems brash. We may only have observed a small handful of data points before we are asked to make an inference, and it may be likely that we change our mind in the future. In light of this we should be qualifying our predictions, by taking the alternatives into account, rather than claiming one with full certainty.

Accounting for these competing explanations is not just an academic excercise that *might be nice to do*, but becomes important in the low data setting where there are only a few examples for all scenarios, and sometimes crucial in the tails of the data distribution $\mathrm{P}(\mathcal{X})$, irrespective of the size of the original dataset. Here, understanding our uncertainty about predictions can help us respect safety guarantees, among other uses.

There are also good reasons why the Bayesian attack might yield better predictive accuracy than a frequentist approach. TODO pick up from here.

As an example, consider the data in Figure 2.5, showing the number of daily rentals for a bike sharing scheme. Should we prefer a simple model such as $y = \theta_0 + \theta_1 x$, or as complex a model as we can computationally manage, such as $y = \sum_{i=1}^{10^8} \theta_i x^i$? A common viewpoint is that a Bayesian should prefer the latter, Rasmussen and Ghahramani (2001); Wilson et al. (2020) restricting the functional complexity of the solutions through the use of priors, as opposed to restricting the flexibility of the model itself. Indeed, the trend in Figure 2.5 is very complicated and captures the myriad reasons why somebody might choose to rend a bike on a given day; due to the weather and time of year, weekends and national holidays, events, the popularity of the bike sharing scheme so forth.

While modelling is valuable precisely because it offers a simplifying summary of the full data, using models with high bias (such as low-order polynomials) to avoid overfitting is arguably the wrong approach. It seems far more elegant to have a method for setting the parameters of a potentially highly expressive model, which avoids blindly interpolating the data when given the opportunity. This is what Bayesian inference achieves.

* * *

Bayesian modelling begins by defining a *probabilistic model*, which is a joint description of how known data $\mathcal{D}$ and some unknown data θ (sometimes thought of as a *hypothesis*) interact[2];

2 The notation θ is used here for consistency with later sections, however this term is by no means limited to model parameters. More generally, we could treat it as representing any *latent* or unobserved variables, where the notation z is more common.

$p(\theta, \mathcal{D})$. In the bike example above, the θ would be the model parameters, and $\mathcal{D}$ would be the observed variables including the daily bike rental count, and other covariates such as the weather, time of year and so on.

If we were to draw samples from $p(\theta, \mathcal{D})$ before observing any data, then these should reflect our prior intuitions about the variables that we might observe. For instance we could set *weight-space* priors, expressing a preference for the model parameters to be small and centered around zero *a priori*. If we expected more rentals during the summer than during the winter, then we could specify $p(\theta, \mathcal{D})$ such that draws of $\mathcal{D}$ (obtained by integrating out θ, and sampling from $p(\mathcal{D})$) would reflect this. Some models might allow for *function-space* priors, which allow us to choose the 'wiggliness' or periodicity of any function of the covariates that the model approximates—for instance we could express a preference for cycles in demand to be on the scale of months and not days.

Two key rules of probability theory allow us to work with this probabilistic model. We use the sum rule to obtain a marginal distribution from any joint distribution by integrating out variables:

$$p(\mathcal{D}) = \int p(\mathcal{D}, \theta) d\theta, \tag{2.32}$$

and we use the product rule to express any joint distribution as the product of conditional distributions:

$$p(\mathcal{D}, \theta) = p(\mathcal{D}|\theta)p(\theta) = p(\theta|\mathcal{D})p(\mathcal{D}). \tag{2.33}$$

As a corollary of the above, we get the definition of a conditional probability distribution:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}, \theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{2.34}$$

If we assigned a type of `Belief` to distributions over the hypothesis θ, then we can see that two of the terms in Equation 2.34 are `Belief`-things: the posterior $p(\theta|\mathcal{D})$ and the prior $p(\theta)$. The difference between these is mostly semantic, with one being the updated version of the other: the prior represents what we believe about θ before observing data, and the posterior represents our updated view of θ after observing data. This also emphasises that Equation 2.34 may be applied recursively as new data arrives, in a process called *Bayesian belief updating*.

The other terms are distributions over the data, and their similarity is less pronounced. We might assign a 'type' of `Accuracy` to the likelihood term $p(\mathcal{D}|\theta)$, which gives us the probability of observing the data $\mathcal{D}$ for some fixed setting of the parameters θ. Frequentists often treat maximising this likelihood as a self-contained framework for learning model parameters—usually by minimising some loss function $\ell(y, \hat{y})$ which is a close analogue to the likelihood, and a measure of the user's unhappiness with a prediction.

The choice of likelihood function determines what ends up being predicted by the model. If we want to predict the average number of bikes rented per day, then we ought to maximise a Gaussian likelihood (equivalently minimise the *squared error loss*), however if the median is of interest instead, then the best guess is to maximise a Laplace likelihood (or minimise the *absolute error loss*).

The slightly unassuming *marginal likelihood* term $p(\mathcal{D})$ is more than just a normalising constant. Also called the *model evidence*, we can write it out more fully as

$$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\theta, \mathcal{M})p(\theta|\mathcal{M}) d\theta, \tag{2.35}$$

where we have introduced the explicit condition on the model choice $\mathcal{M}$. (All the terms in Equation 2.34 are actually conditioned on $\mathcal{M}$, which we omitted for simplicity.)

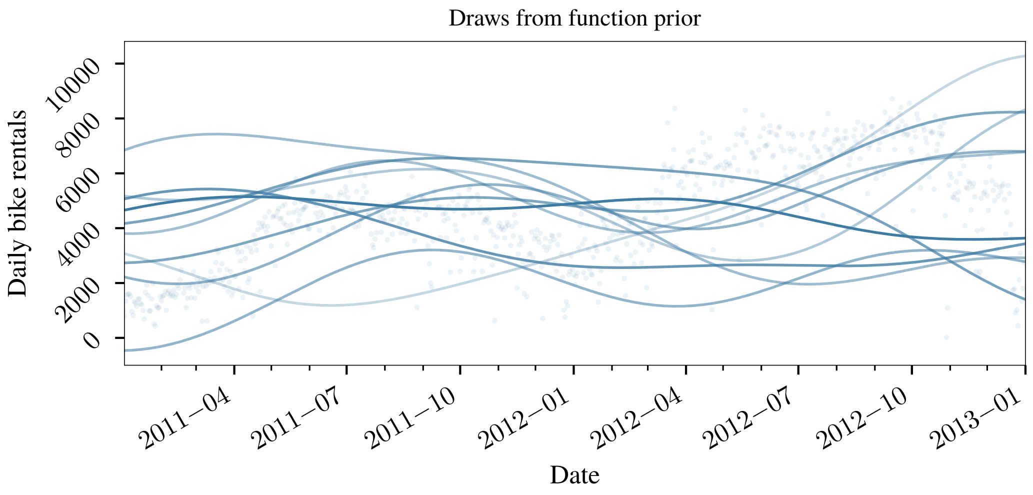


Figure 2.6: Random draws from the prior over functions $p(f(x)|\mathcal{M})$ resulting from sampling $\hat{\theta} \sim p(\theta|\mathcal{M})$ (for some arbitrary $\mathcal{M}$ and prior, which I have left unspecified). When overlaid on the example dataset, these random functions seem flexible enough that one of them (perhaps not plotted here) might have plausibly generated the data, assuming corrupting observation noise. Hence we might consider the bike dataset $\mathcal{D}$ to be *in the support* of this hypothetical model $\mathcal{M}$: $p(\mathcal{D}|\mathcal{M}) > 0$.

Suppose that we sample some parameters from the prior $\hat{\theta} \sim p(\theta|\mathcal{M})$, and apply these to the model $\mathcal{M}$. This would induce a probability distribution over the functions that we may end up modelling *a priori*, $p(f(x)|\mathcal{M})$. We may treat the model evidence in Equation 2.35 as the probability that we would generate any given dataset $\mathcal{D}$ by sampling from this prior over functions $p(f(x)|\mathcal{M})$. A cartoon example of this is provided in Figure 2.6, where the opacity of each function plot corresponds to its prior probability $p(f(x)|\mathcal{M})$.

The model evidence provides the Bayesian with an elegant approach to model selection. The common approaches to avoiding overfitting in a frequentist context involve somewhat ad-hoc regularisation terms and adjusting the complexity of the model (e.g. the number of parameters) in proportion to the number of observed data points. However the Bayesian methodology makes no accommodation for adjusting the model complexity based on the amount of data available: our beliefs about the (potentially very sophisticated) process which generated the data ought to be independent of the number of observations we have made. For instance we should not use a linear model for the bike dataset having seen only 5 observations, since we understand that the myriad factors that link a covariate to the number of bike rentals cannot be accurately described by this model.

To this end, it is commonly argued (Neal, 1996; MacKay, 1992; Rasmussen and Ghahramani, 2001) that we should always use as complex a model as we have occasion for (limited by our computational budget), and instead rely on our beliefs, expressed through the model evidence, to limit the complexity of the solutions.

Wilson and Izmailov (2020) emphasise this point, stressing that we shouldn't conflate the model flexibility with the complexity of the model class. The flexibility of the model $\mathcal{M}$ should be such that the diversity of datasets which we may plausibly encounter should be supported by the model: $p(\mathcal{D}|\mathcal{M}) > 0$. The model class should be the simplest which is consistent with the available data and the model evidence (as stated in 2.35) provides this implicit 'regularisation', and works to discourage complicated models. An extremely complicated model which interpolates each point exactly may obtain very high likelihood $p(\mathcal{D}|\theta)$, however if the parameters which lead to this are improbable (small $p(\theta)$) then it will be discouraged.

The term on the left of Equation 2.34, $p(\theta|\mathcal{D})$, is the *posterior* distribution which summarises our beliefs about the values that the parameters θ ought to take after we have observed the data $\mathcal{D}$. When θ is taken to represent model parameters, this posterior distribution represents an updated distribution over the model parameters which has been 'calibrated' after observing the data. While we would like to evaluate the posterior exactly and efficiently, this proves to be a somewhat elusive goal for complicated models, where computing the marginal likelihood becomes intractable. The approximations that we fall back to are the subject of the next section.

With the posterior in hand, we finally come to the task of making predictions with our model. For a given test input x, the distribution over outputs y is found by marginalising out the model parameters:

$$p(y|x, \mathcal{D}) = \int p(y|x, \theta)p(\theta|\mathcal{D})d\theta. \tag{2.36}$$

The output of such a model is a combination of the likelihood of observing the output point for some setting of the parameters, weighted by the posterior, which captures our current beliefs about those parameters having observed the training data $\mathcal{D}$. This marginalisation is perhaps the defining characteristic of a *Bayesian* approach to learning, yet unfortunately for all its clean theoretical motivations, this marginalisation is also what makes exact Bayesian inference intractable for models in which we cannot calculate this integral analytically. We will review some approximation methods in Section 2.2.2.

2.2.1 *Bayesian Neural Networks*

A Bayesian neural network will be the model used in later chapters to estimate the parameters of the value distribution, hence we briefly review this model class here.

We begin with a simple linear model, and the basic framework for learning the parameters of such a model. Consider a set of data $\mathcal{D} = \{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^N$ consisting of input-target pairs ($\boldsymbol{x}_i \in \mathbb{R}^Q, \boldsymbol{y}_i \in \mathbb{R}^D$). Further suppose that we define a mapping $y(\boldsymbol{x}; \boldsymbol{W})$ from the input activities $\boldsymbol{x}$ to the target values $\boldsymbol{y}$ of the form

$$f(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{b}) = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}, \tag{2.37}$$

where $\boldsymbol{W} \in \mathbb{R}^{Q \times D}$ is a parameter matrix, and $\boldsymbol{b} \in \mathbb{R}^D$ is a vector of 'bias' or intercept terms. All linear models of this form can be obtained from different settings of $\boldsymbol{W}$ and $\boldsymbol{b}$.

If we make the common modelling assumption of i.i.d. observations $(\boldsymbol{y}_i, \boldsymbol{x}_i)$, and that the observed target values $\boldsymbol{y}$ are corrupted by Gaussian noise,

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) + \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(\boldsymbol{0}, \sigma^{-1}\boldsymbol{I}),$$

for a precision term σ^{-1}, then finding the parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ which minimise the sum of squared errors

$$\arg\min_{\hat{\boldsymbol{\theta}} \in \Theta} E_D(\mathcal{D}|\boldsymbol{\theta}) = \arg\min_{\hat{\boldsymbol{\theta}} \in \Theta} \sum_{i=1}^N \frac{1}{2}[\boldsymbol{y}_i - y(\boldsymbol{x}_i; \boldsymbol{\theta})]^2, \tag{2.38}$$

gives us an estimate of the expected value of $\boldsymbol{y}$ under the maximum likelihood framework (the likelihood in this case being a Gaussian).

The task of 'learning' is therefore to find the parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$ which define a linear mapping of the form given in Equation 2.37 which fits the data well; that is, gives low E_D on test data.

This simple linear model does not get us very far however and we would like to apply tme same ideas to come up with a more flexible model. Linear basis function regression offers a simple extension which allows us to model non-linear functions. We first pass the inputs $\boldsymbol{x}$

through non-linear basis functions, giving a feature vector $\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \dots, \phi_K(\boldsymbol{x})]$, and subsequently perform linear regression as above with this new vector. Traditionally, these basis functions $\phi(\boldsymbol{x})_k$ are usually selected manually by the practitioner to be polynomials of different degrees, or sinusoids of different frequencies.

The key difference between a linear basis function regression model and a *neural network* is that the basis functions in a neural network are parametrised, with these parameters learned from data. In particular, the basis functions now take the form

$$\phi_k^{\boldsymbol{w}_k, b_k} \doteq \phi_k(\boldsymbol{x}\boldsymbol{w}_k + b_k),$$

where ϕ_k is now a scalar-valued (non-linear) function, $\boldsymbol{w}_k$ is a vector of weights, and b_k a real-valued bias term. For example, if we selected $\phi_k(\cdot) = \tanh(\cdot)$, then $\phi_k^{\boldsymbol{w}_k, b_k} = \tanh(\boldsymbol{x}\boldsymbol{w}_k + b_k)$. Generally all ϕ_k are taken to be the same function, and we call this ϕ_k an *activation function*.

Summarising the parameters of this model as $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2\}$, we can express this new non-linear model of $\boldsymbol{y}$ more compactly by writing it as follows:

$$\begin{aligned} y(\boldsymbol{x}; \boldsymbol{\theta}) &= \Phi^{\boldsymbol{W}_1, \boldsymbol{b}_1}(\boldsymbol{x})\boldsymbol{W}_2 + \boldsymbol{b}_2, \text{ where} \\ \Phi^{\boldsymbol{W}_1, \boldsymbol{b}_1}(\boldsymbol{x}) &= \phi(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1), \end{aligned}$$

with $\phi(\cdot)$ applied element-wise to the resulting vector. For the basis function expansion the parameters are $\boldsymbol{W}_1 \in \mathbb{R}^{Q \times K}$, $\boldsymbol{b}_1 \in \mathbb{R}^K$. The parameters for the old linear model are now $\boldsymbol{W}_2 \in \mathbb{R}^{K \times D}$, $\boldsymbol{b}_2 \in \mathbb{R}^D$. These parameters $\boldsymbol{\theta}$ *can* be selected using backpropagation to minimise $E_D(\mathcal{D}|\boldsymbol{\theta})$, and this is the prevalent approach among practitioners.

This model contains a single hidden layer, and for a sufficiently wide hidden layer using a sigmoid activation function, this is sufficient to approximate arbitrary functions (Cybenko, 1989). However with few modifications to the underlying methods, we can repeatedly compose these layers; giving a *deep* neural network. One benefit of this is that these deep networks achieve greater *representational* efficiency than their shallow counterparts (Bengio and Lecun, 2007), creating a very flexible class of parametric function approximators.

TODO cover Bayesian neural networks here.

// Louizos and Welling (2016) give a good motivation for the need for uncertainty estimates in deep neural networks in their paper.

2.2.2 *Approximate Inference for Bayesian Models*

"Exact marginalisation" quips David MacKay (2003) "is a macho activity enjoyed by those who are fluent in definite integration". Indeed, the cases in which the integrals in Equations 2.34 and 2.36 may be computed in closed form are usually limited to the case of *conjugate distributions*.

A prior distribution $p(y)$ and likelihood $p(x|y)$ are said to be conjugate iff the posterior $p(y|x)$ belongs to the same family of probability distributions as the prior $p(y)$. While the set of probability distributions which have a conjugate prior are relatively numerous (this includes most of the exponential family), these can only be used to define relatively simple likelihood functions. For more complicated likelihoods (for instance, if θ parametrises a Bayesian neural network), then there exists no conjugate prior distribution, and we must seek recourse to approximate methods for performing Bayesian marginalisation.

We will focus on two such approximation methods. The first, variational inference (VI) casts Bayesian inference as an optimisation problem. The second, a more recent line of work, proposes to use ensembles of deep networks to estimate predictive uncertainty (Lakshminarayanan et al., 2017). While it is unclear as to whether these ensemble methods are really *Bayesian*, there are arguments for their interpretation as an approximation to Bayesian marginalisation (Wilson and Izmailov, 2020), and certain modifications can make this association clearer Pearce et al. (2020).

There exist many other, potentially more accurate approximations to the intractable posterior, however the two methods above have been selected for their speed, which is particularly important in the RL setting where inference must be repeated many times per seconds.

2.2.3 *Variational Inference*

Faced with an intractable posterior distribution $p(\theta|x)$, for which we must evaluate the expensive integral in the marginal likelihood, we might resolve to instead approximate the posterior with a simpler class of distributions which we can more easily work with, $q(\theta)$.

The idea of variational inference is to minimise the divergence between our approximate distribution for the posterior $q(\theta)$, and the intractable posterior distribution itself, $p(\theta|x)$. One suitable distance measure is the Kullback–Leibler (KL) divergence,

$$KL(q(\theta)\|p(\theta|x)) \doteq \int q(\theta) \log \frac{q(\theta)}{p(\theta|x)} d\theta, \tag{2.39}$$

which is a measure of the 'mismatch' between two distributions with the same support. The KL divergence is always positive, and if two distributions are the same then it will be equal to 0. It is however asymmetric, $KL(q\|p) \neq KL(p\|q)$, and this leads to two possible situations. The first, $KL(q\|p)$ is called *forward* KL. If the approximating distribution q is simpler than the true posterior p, which is often the case, then forward KL will result in mode-seeking behaviour, where the approximate distribution will model one (or more) of the modes of the true posterior. Minimising the alternative, *reverse* KL will result in mean-seeking behaviour, where the approximating distribution will model the mean of the true posterior. The former is generally preferred to the latter, hence we minimise $KL(q\|p)$.

Initially, equation 2.39 may seem problematic: how do we sample from the posterior in the denominator if it is intractable, and how can we optimise the KL with respect to the distribution $q(\theta)$? The following is one derivation of the variational lower bound.

The following results from applying a series of identities:

$$\begin{aligned}
\log p(x) &= \int q(\theta) \log p(x) d\theta && \text{(def. marginalisation)} \\
&= \int q(\theta) \log \frac{p(x,\theta)}{p(\theta|x)} d\theta && \text{(def. conditional probability)} \\
&= \int q(\theta) \log \frac{p(x,\theta)q(\theta)}{p(\theta|x)q(\theta)} d\theta \\
&= \int q(\theta) \log \frac{p(x,\theta)}{q(\theta)} d\theta + \int q(\theta) \log \frac{q(\theta)}{p(\theta|x)} d\theta \\
&\doteq \mathcal{L}\big(q(\theta)\big) + KL\big(q(\theta)\|p(\theta|x)\big), && \text{(def. KL)}
\end{aligned}$$

where in the last line, we have defined $\mathcal{L}\big(q(\theta)\big)$, which we call the evidence lower bound (ELBO). To see why this is, consider $\log p(x) = \mathcal{L}\big(q(\theta)\big) + KL\big(q(\theta)\|p(\theta|x)\big)$—since the logarithm is monotonic, and the KL is always non-negative, we have that $\mathcal{L}\big(q(\theta)\big) \leq \log p(x)$; that is, the log evidence is indeed lower-bounded by the ELBO.

Returning to Equation 2.39, in order to minimise the KL divergence between the approximate posterior $q(\theta)$ and the true posterior, it suffices to maximise this lower bound:

$$KL\big(q(\theta)\|p(\theta|x)\big) \to \min_{q(\theta)} \quad \Leftrightarrow \quad \mathcal{L}\big(q(\theta)\big) \to \max_{q(\theta)}.$$

Thus the optimisation problem is now that of maximising the ELBO, which crucially no longer relies on the intractable posterior, however it still requires integrating over θ. To proceed, we perform the following rearrangement

$$\begin{aligned}
\mathcal{L}\big(q(\theta)\big) &= \int q(\theta) \log \frac{p(x,\theta)}{q(\theta)} d\theta \\
&= \int q(\theta) \log \frac{p(x|\theta)p(\theta)}{q(\theta)} d\theta && \text{(def. joint distribution)} \\
&= \int q(\theta) \log p(x|\theta) d\theta + \int q(\theta) \log \frac{p(\theta)}{q(\theta)} d\theta \\
&= \mathbb{E}_{q(\theta)}\big[\log p(x|\theta)\big] - KL\big(q(\theta)\|p(\theta)\big). && (2.40)
\end{aligned}$$

The last line emphasises that maximising the ELBO represents a trade-off between maximising the expected log likelihood, and minimising the KL divergence between the approximate posterior and prior, which has a regularising effect.

TODO discuss stochastic variational inference using automatic differentiation. Reference: Graves

2.2.4 *Deep Ensembles*

TODO

Mention work by Lakshminarayanan et al. (2017), **?**, Izmailov et al. (2021), Pearce et al. (2020) and potentially others.

Plot expected value through bike dataset, and provide comparisons to other approximation methods.

PIECEWISE-LINEAR LOG LIKELIHOOD ESTIMATION

In this chapter we will consider methods for obtaining a Bayesian estimate of the return distribution. That is, a method of representing the return distribution conditioned on some state-action pair $\mathrm{P}(R_t|\theta, s_t, a_t)$ with uncertainty estimates in the parameters of the distribution θ. We will begin by reviewing the approach taken by most previous work on distributional RL, namely quantile regression, as well as considering its Bayesian interpretation. We will then take inspiration from this, and address some of the shortcomings of the quantile regression approach, to devise a new approach to representing arbitrary, multi-modal probability distributions by learning approximating the log-likelihood with a piecewise linear function.

3.1 QUANTILE REGRESSION

We first begin by briefly reviewing quantile regression Koenker and Bassett (1978), which was first applied to the distributional RL setting by Dabney et al. (2017).

Consider a real-valued random variable Y with distribution function $F_Y(y) = P(Y \leq y)$. The τth *quantile* of Y is the smallest point $y \in Y$ in the support of the distribution for which the CDF is at least as great as τ. For instance, the median corresponds to $\tau = 0.5$, the first quartile is at $\tau = 0.25$, and the first percentile is given by $\tau = 0.01$. More generally, for some chosen quantile $\tau \in (0, 1)$ we can define the quantile function as

$$Q_Y(\tau) = F_Y^{-1}(\tau) = \inf\{y : F_Y(y) \geq \tau\},$$

Working with univariate random variables is not very interesting however—predicting quantiles for jointly-distributed random variables (or a multivariate random variable) leads to the notion of the conditional quantile function. Taking the jointly-distributed view, given a vector of covariates $X = \boldsymbol{x}$, the τ-th conditional quantile function is

$$Q_Y(\tau|\boldsymbol{x}) = \inf\{y \ : \mathrm{P}(Y \leq y|X = \boldsymbol{x}) \geq \tau\}.$$

Attempting to learn or approximate this function from data gives us *quantile regression*.

Definition 3.1 (Quantile Regression; Koenker and Bassett (1978))**.** For some selected quantile $\tau \in (0, 1)$, and some model of the conditional quantile function $Q_Y(\tau|\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta}, \tau) + \epsilon$, parametrised by $\boldsymbol{\theta} \in \mathbb{R}^d$ (we leave the exact form of f undefined for now), the task of *quantile regression* is to estimate these parameters $\boldsymbol{\theta}$ from observed data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ by minimising the expected *quantile regression loss*, $\rho_\tau(u)$,

$$\begin{aligned}\rho_\tau(u) &= \begin{cases} u(\tau - 1) & \text{if } u < 0 \\ u\tau & \text{otherwise} \end{cases} \\ &= u\big(\tau - \mathbb{I}(u < 0)\big). \end{aligned} \tag{3.1}$$

That is, assuming i.i.d. data, we are interested in finding

$$\boldsymbol{\theta} = \operatorname*{arg\,min}_{\hat{\boldsymbol{\theta}} \in \Theta} \sum_{i=1}^{N} \rho_\tau\big(y_i - f(\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}, \tau)\big). \tag{3.2}$$

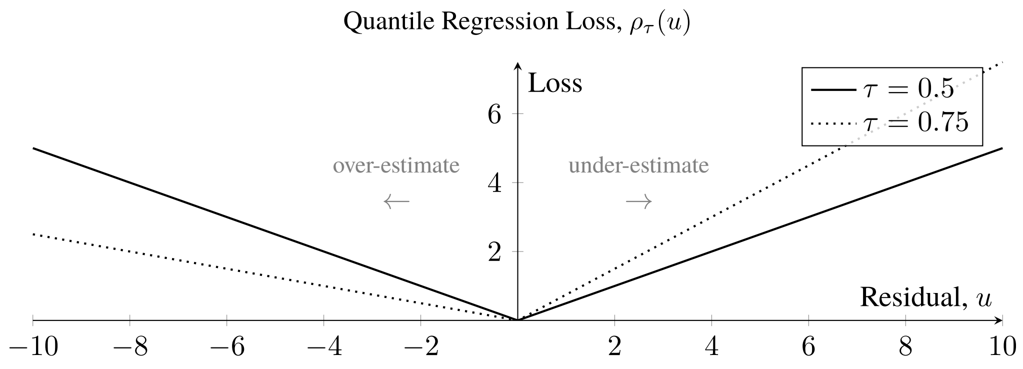


Figure 3.1: Visualisation of the quantile regression loss function.

Note that for now we are only concerned with estimating the quantile function for single settings of τ. We will treat the more interesting (and useful) case of learning multiple quantiles simultaneously later.

Why does this loss function work? Suppose that we have an estimate for the τth quantile value which we denote $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\theta}, \tau)$. We can write the expected quantile regression loss in full as:

$$\begin{aligned}\mathbb{E}[\rho_\tau(Y - \hat{y})] &= \int_{-\infty}^{\infty} \rho_\tau(y - \hat{y}) dF_Y(y) \\ &= (\tau - 1) \int_{-\infty}^{\hat{y}} (y - \hat{y}) dF_Y(y) + \tau \int_{\hat{y}}^{\infty} (y - \hat{y}) dF_Y(y). \qquad (3.3)\end{aligned}$$

Since the expected loss is convex, differentiating, equating to 0 and solving will give us the value of $\hat{y}$ for which the quantile regression loss is minimized. We can differentiate Equation 3.3 (using Leibniz's rule) to get

$$\begin{aligned}\frac{d}{d\hat{y}} \mathbb{E}[\rho_\tau(Y - \hat{y})] &= (\tau - 1) \left((\hat{y} - \hat{y}) + \int_{-\infty}^{\hat{y}} \frac{\delta}{\delta \hat{y}} (y - \hat{y}) dF_Y)y \right) \\ &\quad + \tau \left(-(\hat{y} - \hat{y}) + \int_{\hat{y}}^{\infty} \frac{\delta}{\delta \hat{y}} (y - \hat{y}) dF_Y(y) \right) \\ &= (1 - \tau) F_Y(\hat{y}) - \tau(1 - F_Y(\hat{y})) \\ &= F_Y(\hat{y}) - \tau,\end{aligned}$$

hence when the derivative is equal to 0, we get $\hat{y} = F^{-1}(\tau)$, i.e. the τ-th quantile, as required.

To motivate the quantile regression loss at a more intuitive level, we can consider the effect of minimising different loss functions parametrised by τ. In the frequentist setting, we can change the statistic of Y being predicted by $f(\boldsymbol{x}, \boldsymbol{\theta})$ by changing the loss function. For instance minimising the sum of squared errors gives us an estimator for the mean: $\arg\min_{\hat{\boldsymbol{\theta}} \in \Theta} \sum \big(y - f(\boldsymbol{x}, \hat{\boldsymbol{\theta}})\big)^2$, however the median can instead be predicted by minimising the sum of absolute deviations: $\arg\min_{\hat{\boldsymbol{\theta}} \in \Theta} \sum |y - f(\boldsymbol{x}, \hat{\boldsymbol{\theta}})|$, called the *least absolute deviations* (LAD) estimator.

This sum of absolute deviations is merely the quantile regression loss for $\tau = 0.5$. Figure 3.1 visualises the effect of introducing asymmetry in the QR loss function to predict other quantiles. When the loss is asymmetric, we prefer points which will leave us on the flatter of the two branches. For instance, when predicting the $\tau = 0.75$th quantile, under-estimates (i.e. positive residuals, where $y - f(\boldsymbol{x}, \boldsymbol{\theta}, 0.75) > 0$) are three times as costly as an over-estimate. To compensate for this, the parameters giving the predicted value $\hat{y} = f(\boldsymbol{x}, \boldsymbol{\theta}, 0.75)$ are chosen such over-estimates are three times as likely as under-estimates: that is,

$$\mathrm{P}(Y \leq \hat{y}) = 3\mathrm{P}(Y > \hat{y}).$$

When samples from Y are smaller than $\hat{y}$ three quarters of the time, and larger than $\hat{y}$ one quarter of the time, then $\hat{y}$ must be the 75th percentile, as required.

* * *

Quantile regression can also be given a probabilistic interpretation. For familiarity, first consider methods for estimating the conditional mean of Y given some observations $\boldsymbol{x} \sim X$. Here the standard approach in many frameworks (linear, generalised-linear, nonparametric etc.) is to assume a model of the form $y = f(\boldsymbol{x}, \boldsymbol{\theta}) + \epsilon$, where ϵ follows a zero-mean, symmetric distribution. This ϵ term is often interpreted as explaining away any un-modelled determinants and corrupting observation noise, and is often taken to be Gaussian-distributed, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, where we would model y as $\mathcal{N}(y; f(\boldsymbol{x}, \boldsymbol{\theta}), \sigma^2)$. As the exponential of a negative quadratic, finding the parameters $\boldsymbol{\theta}$ which maximise the likelihood of observations under this Gaussian model is equivalent to minimising the sum of squared errors, and we recover the OLS estimate.

Quantile regression, as presented above, lends itself to a similar interpretation. For the $\tau = 0.5$th quantile, the analysis is exactly as above with the exception that the error term is now Laplace distributed, $\epsilon \sim \text{Laplace}(0, \sigma)$, for some scale term σ. Our model for the *median* of y is now of the form $\text{Laplace}(y; f(\boldsymbol{x}, \boldsymbol{\theta}), \sigma)$, where the Laplace density function is

$$f_L(x; \mu, \sigma) = \frac{1}{2\sigma} \exp\left(-\frac{|x - \mu|}{\sigma}\right). \tag{3.4}$$

We can see that when the location parameter is set to $\mu = f(\boldsymbol{x}, \boldsymbol{\theta})$, maximising the joint probability of the observed data points (i.e. a product of Laplace densities, assuming i.i.d. data) is equivalent to minimising the sum of absolute deviations $\arg\min_{\hat{\boldsymbol{\theta}} \in \Theta} \sum |y - f(\boldsymbol{x}, \hat{\boldsymbol{\theta}})|$, hence we recover the LAD predictor for the median.

To generalise this to further quantiles, it suffices to change the asymmetry of the distribution of the error term ϵ. The class of asymmetric Laplace distributions (Kozubowski, 1999) are parametrised by a location, scale and asymmetry parameter. Many forms have been suggested, however for fluidity of exposition, here we write the density function as

$$f_{\text{ALD}}(x; \mu, \sigma, \tau) = \frac{\tau(1-\tau)}{\sigma} \exp\left\{-\frac{x-\mu}{\sigma}\big(\tau - \mathbb{I}(x - \mu < 0)\big)\right\} \tag{3.5}$$

$$\propto \exp\left\{-\rho_\tau(x - \mu)\right\} \tag{3.6}$$

where $\tau \in (0, 1)$. This allows us to interpret the target quantile as the asymmetry parameter in the ALD, and furthermore we can see that the ALD is merely the negative exponent of the quantile regression loss. Hence maximising the ALD in Equation 3.5 is equivalent to minimising the quantile regression loss. The asymmetric Laplace distribution is visualised in Figure 3.2—is has heavy tails (more so than, for instance, a Gaussian), finite moments and is 'nice' to work with due to its simple form and robustness to outliers.

> As an aside, we can generalise the statistics of the distribution for Y predicted by other estimators by changing the asymmetry of their loss function—or equivalently the likelihood that they maximise. For example, if we made the Gaussian likelihood maximised by an estimator for the conditional mean asymmetric (by weighting the density above and below the location μ differently), then we

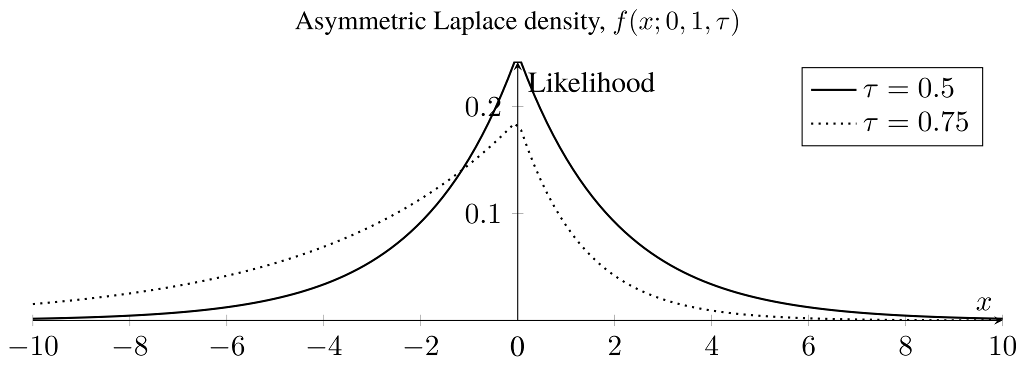


Figure 3.2: Asymmetric Laplace likelihood.

would obtain a predictor for the *expectiles* of the distribution, which generalise the mean in the same way that quantiles generalise the median.

Recall that the reason for performing quantile regression in the first place was to represent the full distribution over the returns. If one has the inverse distribution function $Q_Y(\cdot)$, then this is sufficient to fully describe the distribution. Therefore the full potential of this framework lies not in single-τ fits, but simultaneously finding $Q_Y(\tau|\boldsymbol{x}), \forall\tau \in (0,1)$.

We might begin by pooling individual estimates of $Q_Y(\tau|\boldsymbol{x})$, for a fixed set of τ. This gives the following likelihood

$$p(\mathcal{D}|\boldsymbol{\theta},\tau) = \prod_{j=1}^{M} \frac{\tau(1-\tau)}{\sigma} \exp\left\{-\frac{\rho_\tau(y_j - f(\boldsymbol{x}_j,\boldsymbol{\theta}))}{\sigma}\right\}. \tag{3.7}$$

While this might be valid in the asymptotic case with a large number of data points, in the more realistic, low data setting this approach fails due to a poor sharing of information between the individual estimates. This is demonstrated in Figure 3.3, where the expected quantile value is plotted as a different line, for each of the $N = 20$ models. The underlying synthetic data is a sinusoid corrupted with asymmetric, Weibull-distributed noise. In particular, this post-estimation pooling of individual estimates gives cause for concern from the Bayesian standpoint; where the distinct posterior beliefs $p(\boldsymbol{\theta}_i|\mathcal{D})$ for each of the N independent models are naïvely being combined.

A straightforward solution is to jointly predict the N quantiles using a single model. The model is now of the form $\boldsymbol{y} = f(\boldsymbol{x},\boldsymbol{\theta}) + \epsilon$, where each output y_i represents the value of quantile τ_i: $y_i = Q_Y(\tau_i|\boldsymbol{x})$. Put otherwise, the likelihood is now

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{j=1}^{M} \frac{\tau_i(1-\tau_i)}{\sigma} \left\{-\frac{\rho_{\tau_i}(\boldsymbol{y}_j - f(\boldsymbol{x}_j,\boldsymbol{\theta}))}{\sigma}\right\}. \tag{3.8}$$

Figure 3.4 shows the improved performance of this model; where each quantile extimate is more coherent with the others, and the slightly skewed Weibull distribution can be seen along the y-dimension.

This approach of performing quantile regression for N fixed quantiles is taken for distributional RL by Dabney et al. (2017) (using the frequentist equivalent of minimising the expected quantile regression loss). Note that in their implementation (and also in the one used to generate Figure 3.4), there is no mechanism to ensure the monotonicity of the quantile outputs of the network. While the output is close to being monotonic even without such a mechanism, for

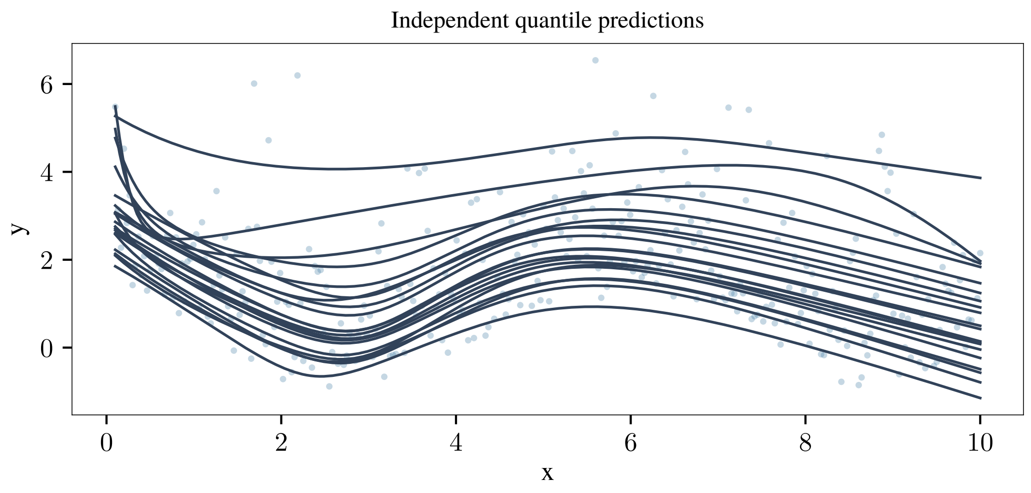


Figure 3.3: Individual (independent) quantile predictions for $N = 20$ quantiles. There is a poor sharing of information and quantiles overlap.

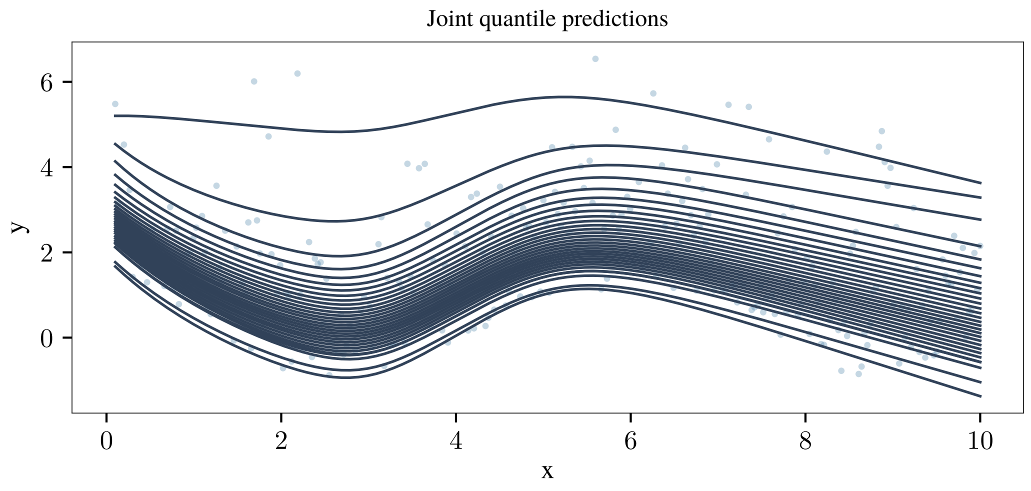


Figure 3.4: Jointly predicting quantile outputs are more cohesive; yet there is no constraint ensuring monotonicity.

the purposes of comparing the quantiles of a predicted distribution to a target distribution, (an approximation of the Wasserstein distance between the two), monotonicity has little importance.

Further performance improvements in distributional RL have subsequently been brought about with more expressive variants of this joint quantile regression. For instance Dabney et al. (2018) use *implicit quantile networks* to vary the τ_i locations of the N quantile predictions—for instance allowing the quantile predictions to be more concentrated around areas of interest such as the modes of the distribution. Clements et al. (2020) are the first to use this Bayesian interpretation of quantile regression for distributional RL, allowing them for instance to implement risk-averse, 'uncertainty aware' policies.

While the approaches listed above have been empirically successful, they only estimate *statistics* of the return distribution as opposed to being a full description. For instance in the quantile regression approaches, only the values of a finite set of quantiles are retained, and all other information is lost. This necessitates what Rowland et al. (2019) refer to as an *imputation strategy*, which maps the vector of statistics (e.g. the vector of N quantile estimates) to a distribution which has those statistics.

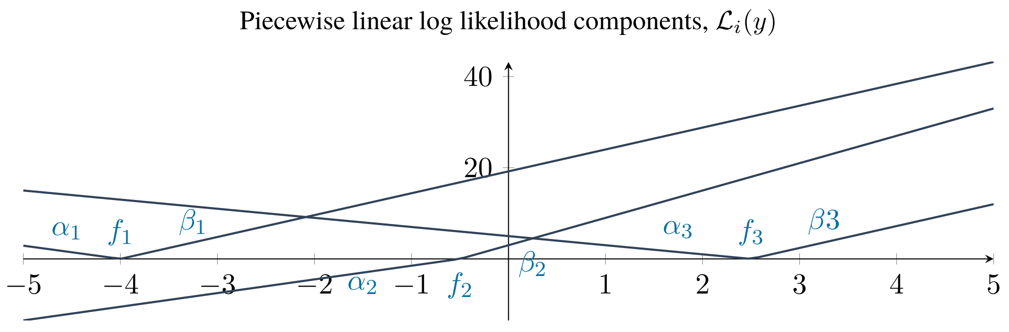


Figure 3.5: Visualisation of $N = 3$ piecewise-linear log likelihood components; the similarity to the quantile regression loss.

In general, existing distributional RL algorithms can be decomposed as the combination of some statistical estimator and an imputation strategy. If an estimate of the full return distribution was available for each state-action pair, then this imputation step would be unnecessary. Barth-Maron et al. (2018) attempted to use mixtures of Gaussians for this purpose, and reported underwhelming initial results. It is with this motivation that we consider the following approach to estimating the return distribution, taking the log-likelihoods to be piecewise linear.

3.2 PIECEWISE LINEAR LOG LIKELIHOOD ESTIMATION

The following is an approach to modelling univariate distributions $\mathrm{P}(y)$, which we will refer to as *piecewise linear log-likelihood estimation*, PL3E. The distribution $\mathrm{P}(y)$ is taken to be the following factorised density:

$$\mathrm{P}(y) = \frac{1}{Z} \prod_{i=1}^{N} \mathrm{P}_i(y), \tag{3.9}$$

for some normalising term Z. Taking logarithms, the log-probability $\mathcal{L}(y) = \log \mathrm{P}(y)$, can be written as

$$\mathcal{L}(y) = \sum_{i=1}^{N} \mathcal{L}_i(y) - \log Z. \tag{3.10}$$

Drawing inspiration from the quantile regression loss above, where positive and negative residuals are weighted differently, we define each term to be of the form:

$$\mathcal{L}_i(y) = -(y - f_i)\big(\alpha_i \Theta(y - f_i) - \beta_i \Theta(f_i - y)\big), \tag{3.11}$$

for Θ a Heaviside step function, with α_i, β_i and f_i some trainable parameters. In the following, we make the assumption that the f_i are ordered, such that $f_i \leq f_{i+1}$; we will review this assumption when discussing the implementation later.

We can see Equation 3.11 as defining a piecewise linear *loss* function not too dissimilar to the quantile regression loss in Equation 3.3. For $N = 3$, Figure 3.5 illustrates the individual piecewise-linear components, for some arbitrary example f, α and β values.

In order to calculate the value of the normalising term, we must find the integral of the PDF in Equation 3.9. For simplicity we proceed using the log density, and consider each segment between consecutive f_i separately. The log likelihood of the jth segment, where $y \in (f_j, f_{j+1}]$ is found by summing the individual piecewise-linear components in this range

$$\begin{aligned}\mathcal{L}(f_j < y \leq f_{j+1}) &= -\sum_{i=1}^{j} \alpha_i (y - f_i) + \sum_{i=j+1}^{N} \beta_i (y - f_i) - \log Z_j \\ &= \left(\sum_{i=j+1}^{N} \beta_i - \sum_{i=1}^{j} \alpha_i \right) y + \left(\sum_{i=1}^{j} \alpha_i f_i - \sum_{i=j+1}^{N} \beta_i f_i \right) - \log Z_j \\ &\doteq a_j y + b_j - \log Z_j,\end{aligned}$$

where for notational simplicity in subsequent equations we have defined the coefficients for the jth interval (the terms in parentheses) as a_j and b_j.

We must also consider the leftmost segment where $y \in (-\infty, f_1]$:

$$\begin{aligned}\mathcal{L}(-\infty < y \leq f_1) &= \sum_{i=1}^{N} \beta_i (y - f_i) - \log Z_0 \\ &= \underbrace{\sum_{i=1}^{N} \beta_i}_{a_0} y - \underbrace{\sum_{j=1}^{N} \beta_i f_i}_{b_0} - \log Z_0,\end{aligned}$$

as well as the rightmost segment for $y \in (f_N, \infty)$:

$$\begin{aligned}\mathcal{L}(f_N < y < \infty) &= -\left(\sum_{i=1}^{N} \alpha_i (y - fi) \right) - \log Z_N \\ &= \underbrace{-\sum_{i=1}^{N} \alpha_i}_{a_N} y + \underbrace{\sum_{i=1}^{N} \alpha_i f_i}_{b_N} - \log Z_N,\end{aligned}$$

where we can see that a_0, b_0, a_N and b_N are just special cases of the definition of a_j and b_j given above.

We may now find the normalising term for the distribution parametrised by $\boldsymbol{\theta} = \{\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ by summing the integrals of the PDF between each f_i, giving special consideration to the integrals at the endpoints from $-\infty$ to f_1 as well as f_N to ∞.

Beginning with the case where $f_j < y \leq f_{j+1}$, we have

$$\begin{aligned}\int_{f_j}^{f_{j+1}} dy\, e^{a_j y + b_j} &= \left[\frac{1}{a_j} e^{a_j y + b_j} \right]_{f_j}^{f_{j+1}} \\ &= \frac{1}{a_j} e^{b_j} \left(e^{a_j f_{j+1}} - e^{a_j f_j} \right).\end{aligned}$$

It is possible, although hopefully unlikely, that some $a_j = 0$. We treat this as a pathological scenario, and resolve to re-sample network weights from the posterior distribution in these cases, giving us new distribution parameters $\boldsymbol{\theta}$ and hence a new a_j—which we hope will be non-zero. For non-probabilistic approaches, some small $\epsilon \sim \mathcal{N}(0, 1^{-n})$ can also be added to a_j, for $n > 0$.

When $y \in (-\infty, f_1)$, we have

$$\begin{aligned}\lim_{\ell \to -\infty} \int_{\ell}^{f_1} dy\, e^{a_0 y + b_0} &= \lim_{\ell \to -\infty} \left[\frac{1}{a_0} e^{a_0 y + b_0} \right]_{\ell}^{f_1} \\ &= \lim_{\ell \to -\infty} \frac{1}{a_0} e^{b_0} \left(e^{a_0 f_1} - e^{a_0 \ell} \right) \\ &= \frac{1}{a_0} e^{a_0 f_1 + b_0} \qquad \text{(A1)}\end{aligned}$$

where the final equality above (A1) holds so long as $a_0 > 0$. To satisfy this condition, recall that in the first special case above we had that $a_0 = \sum_{i=1}^{N} \beta_i$, and in particular observe that β_1 only features in the calculations for a_0 and b_0. Hence we may treat β_1 as a 'free parameter' which we use to ensure that $a_0 > 0$. Writing $\beta_1 + \sum_{i=2}^{N} \beta_i > 0$, we can therefore set $\beta_1 = -\sum_{i=2}^{N} \beta_i + \epsilon^+$ for some small positive ϵ^+; ensuring numerical stability.

Similarly for the segment where $y \in (f_N, \infty)$, we have

$$\begin{aligned}\lim_{\ell \to \infty} \int_{f_N}^{\ell} dy\, e^{a_N y + b_N} &= \lim_{\ell \to \infty} \left[\frac{1}{a_N} e^{a_N y + b_N} \right]_{f_N}^{\ell} \\ &= \lim_{\ell \to \infty} \frac{1}{a_N} e^{b_N} \left(e^{a_N \ell} - e^{a_N f_N} \right) \\ &= -\frac{1}{a_N} e^{a_N f_N + b_N} \qquad \text{(A2)}\end{aligned}$$

with the assumption in (A2) being that $a_N < 0$. We observe that $a_N = -\sum_{i=1}^{N} \alpha_i$, with α_N being the 'free parameter' in this case, appearing in the calculations for a_N and b_N. Setting $\alpha_N = -\sum_{i=1}^{N-1} \alpha_i + \epsilon^+$ makes the assumption hold.

We can now explicitly write down the normalising term Z as the sum of these integrals:

$$\begin{aligned}Z &= \int_{-\infty}^{f_1} dy\, e^{a_0 y + b_0} + \sum_{j=1}^{N} \int_{f_j}^{f_{j+1}} dy\, e^{a_j y + b_j} + \int_{f_N}^{\infty} dy\, e^{a_N y + b_N} \\ &= \frac{1}{a_0} e^{a_0 f_1 + b_0} + \sum_{j=1}^{N} \frac{1}{a_j} e^{b_j} \left(e^{a_j f_{j+1}} - e^{a_j f_j} \right) - \frac{1}{a_N} e^{a_N f_N + b_N}.\end{aligned}$$

To implement this efficiently, observe that each of the coefficients a_j and b_j are made up of the same underlying sum of terms, hence a simple dynamic programming algorithm, computing the partial sums of coefficients, takes the time to compute the normalising term from $O(N^2)$ for a naive implementation to $O(N)$. Note that this improvement is nice to have, it is mostly inconsequential when neural networks are used to estimate the parameters (where the cost is dominated by the quadratic complexity at each layer).

To consolidate the above, we can plot the N individual piecewise-linear components as estimated by a Bayesian neural network on some bimodal synthetic data, and compare this to the resulting piecewise linear log likelihood. Figure 3.6 shows the negative log likelihood components in the left pane, and the resulting sum of (negative) log likelihood components on the right. The shaded uncertainty estimates represent the parametric uncertainty in the gradient of the components of the log likelihood, while the plotted blue lines show the expected values $-\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})} \left[\mathcal{L}_i(y) \right]$, where variance in the estimates for the f_i locations results in slightly curved lines. Also notice the two pronounced upward and downward lines in the left plot; which result from the constraints placed on β_1 and α_N.

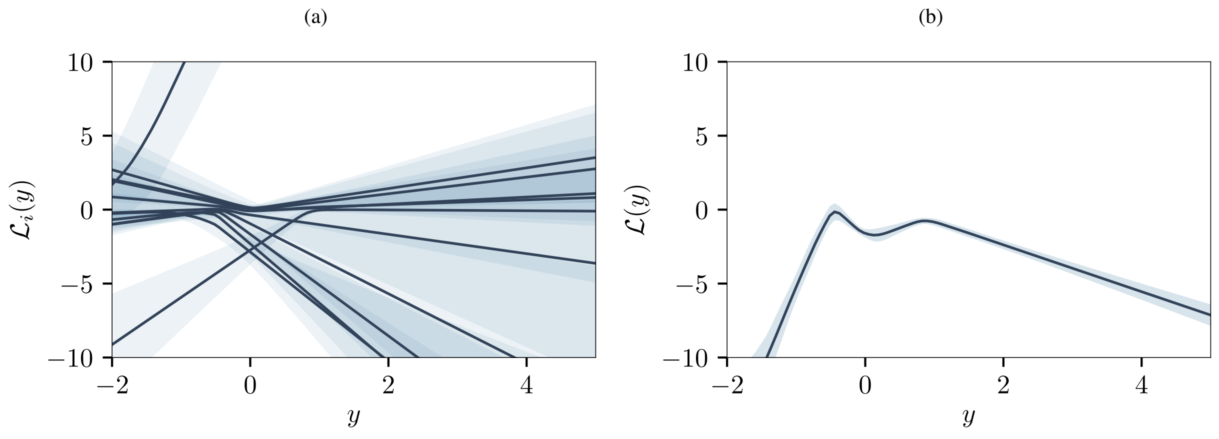


Figure 3.6: a) The (expected) piecewise linear components with uncertainty estimates in $\boldsymbol{\theta} = \{\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ and b) the resulting (expected) negative log likelihood. In the above, $N = 11$ components were used.

In theory, should we want to model more complicated multimodal distributions, we can increase the number of components N in the log likelihood. In practice however, increasing N requires that the function approximator used can scale to the additional parameters $\boldsymbol{\theta}$. Beyond increasing the computational cost, this increase in piecewise linear components (whose parameters are randomly initialised according to the specified prior) is likely to initially have a strong regularising effect on the complexity of the resulting distributions; giving generic unimodal distributions as opposed to more expressive multimodal distributions.

3.2.1 *Properties of the Distribution*

One of the benefits of this simple piecewise linear log-likelihood is that the integrals under the resulting distribution are relatively easy to compute. One quantity of interst for instance is the mean; which in the context of distributional RL would give us an approximation for the value function from the return distribution.

The integral required to compute $\mathbb{E}_Y[y] = \int_{-\infty}^{\infty} y f_Y(y) dy$ is relatively long and unenlightening so I have put it in the appendix (see Appendix B.1). The expected value of this distribution comes to

$$\begin{aligned}\mathbb{E}_Y[y] &= \int_{-\infty}^{\infty} y f_Y(y)\, dy \\ &= \frac{e^{a_0 f_1 + b_0}(a_0 f_1 - 1)}{a_0^2} \\ &+ \sum_{j=1}^{N-1} \frac{e^{b_j}\left(e^{a_j f_{j+1}}\big(a_j f_{j+1} - 1\big) - e^{a_j f_j}\big(a_j f_j - 1\big)\right)}{a_j^2} \\ &- \frac{e^{a_N f_N + b_N}(a_N f_N - 1)}{a_N^2}. \end{aligned} \tag{3.12}$$

It is useful to evaluate the merits of this expectation as a method for mean regression, since the distributional Q-learning algorithm will make extensive use of the mean estimate. For exposition, we can fit a PL3E distribution to the example bike sharing dataset to predict the expected number of daily rentals. This is shown in Figure 3.7.

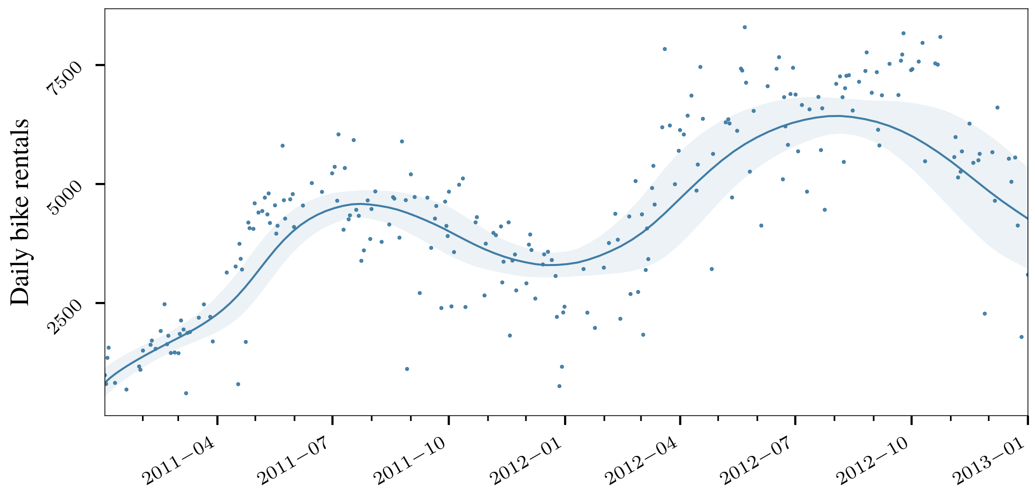


Figure 3.7: Expected value of the PL3E distribution overlaid on the held-out test data.

While we can't directly compare the performance (in terms of RMSE or negative log likelihood) of PL3E to the quantile regression approaches discussed previously, since these only estimate quantiles of the distribution, we can use a mixture of Gaussians (MoG) as a baseline. To draw comparisons to more conventional regression methods, I have also use a Gaussian process for comparison. The results are shown in Table 3.1. TODO discuss results

Having a closed form expression for the return distribution at each state-value pair allows us to sample from the resulting distribution; which the methods employed by previous distributional RL algorithms generally could not do. We will make use of this ability to draw samples from the estimated distributions in the next chapter when we come to put this method to use in the reinforcement learning setting.

Inverse transform sampling makes use of the inverse distribution function (i.e. the quantile function) to map uniform samples drawn from the unit interval $\mathcal{U}[0,1]$ onto the support of the distribution. We must therefore find an expression for the distribution function $F_Y(y)$ which we can then invert to find the quantile function $F_Y^{-1}(y)$.

For any PL3E distribution specified by the parameters $\boldsymbol{\theta} = \{\boldsymbol{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$, the CDF evaluated at some point y is found by integrating the density function from $-\infty$ to y:

$$F_Y(y) = \int_{-\infty}^{y} dt\, f_Y(t). \tag{3.13}$$

Since we have already evaluated this integral when computing the expression for the normalising constant Z, we may re-use the integrals under segments (f_j, f_{j+1}) where $f_{j+1} < y$. We must then add the integral for the last (partial) segment—noting that this may in fact be the first segment if $y < f_1$.

More formally, let us define the integer K for some evaluation point y as

$$K = \max\{0\} \cup \left\{\hat{K} \in \{1, \ldots, N\} \ : \ f_{\hat{K}} \leq y\right\}. \tag{3.14}$$

Table 3.1: RMSE and NLL results for predictions on UCI datasets. TODO copy results from various script outputs

	Variational Inference			Deep Ensembles		
Dataset	PL3E	MoG	GP	PL3E	MoG	GP
Bike						
Boston						
Concrete						
Energy						
Kin8nm						
Naval						
Power						
Protein						
Wine						
Yacht						

We now proceed as follows:

$$\begin{aligned}
F_Y(y) &= \int_{-\infty}^{y} dt\ f_Y(t) \\
&= \frac{1}{Z}\begin{cases} \int_{-\infty}^{y} dt\ e^{a_0 t + b_0} & \text{if } y < f_1 \\ \int_{-\infty}^{f_1} dt\ e^{a_0 t + b_0} + \sum_{j=1}^{K-1} \int_{f_j}^{f_{j+1}} dt\ e^{a_j t + b_j} + \int_{f_K}^{y} dt\ e^{a_K t + b_K} & \text{otherwise} \end{cases} \\
&= \frac{1}{Z}\begin{cases} \frac{1}{a_0} e^{a_0 y + b_0} & \text{if } y < f_1 \\ \left(\frac{1}{a_0} e^{a_0 f_1 + b_0} + \sum_{j=1}^{K-1} \frac{1}{a_j} e^{b_j}\left(e^{a_j f_{j+1}} - e^{a_j f_j}\right) + \frac{1}{a_K} e^{b_K}\left(e^{a_K y} - e^{a_K f_K}\right)\right) & \text{otherwise} \end{cases} \\
&= \frac{1}{Z}\left(A + \sum_{j=1}^{K-1} \frac{1}{a_j} e^{b_j}\left(e^{a_j f_{j+1}} - e^{a_j f_j}\right) + \frac{1}{a_K} e^{b_K}\left(e^{a_K y} - e^{a_K f_{K+\mathbb{I}(K=0)}}\right)\right).
\end{aligned} \tag{3.15}$$

where we have defined $A \doteq \frac{1}{a_0} e^{a_0 f_1 + b_0}$ for notational brevity. The terms a_j, b_j are defined for $j \in [0, N]$ as they were in the previous section which, for completeness, was:

$$\begin{aligned}
a_j &= \sum_{i=j+1}^{N} \beta_i - \sum_{i=1}^{j} \alpha_i \\
b_j &= \sum_{i=1}^{j} \alpha_i f_i - \sum_{i=j+1}^{N} \beta_i f_i.
\end{aligned}$$

Inverting Equation 3.15 above gives us an analytical expression for the quantile function. The new limit for the summand (previously K, as defined in Equation 3.14), for some input point $0 \le x \le y$, let the integer L be defined as

$$L \doteq \max\{0\} \cup \left\{\hat{L} \in \{1, \dots, N\}\ :\ F_Y(f_{\hat{L}}) \le x\right\}, \tag{3.16}$$

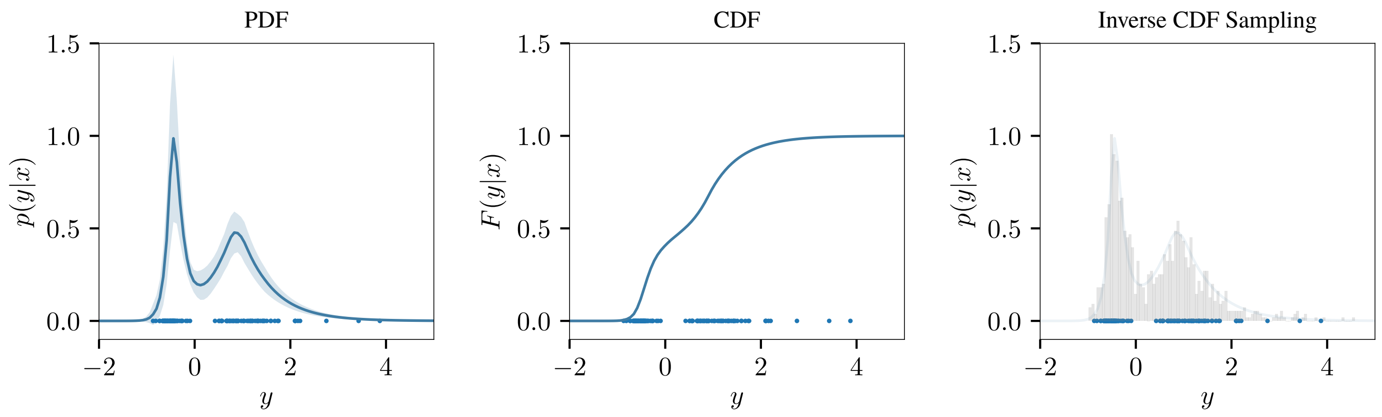


Figure 3.8: Density, distribution function, and quantile function. The (synthetic) observations are plotted along the bottom of each plot.

We can now invert the expression for the CDF, giving

$$F_Y^{-1}(x) = \frac{\log\left(\left(xZ - A - \sum_{j=1}^{L-1} \frac{1}{a_j} e^{b_j}\left(e^{a_j f_{j+1}} - e^{a_j f_j}\right)\right) a_L e^{-b_L} + e^{a_L f_{L+\mathbb{I}(L=0)}}\right)}{a_L}. \tag{3.17}$$

As usual, the normal constraints on invertible functions persist—if the distribution function is not monotonically increasing at some point (i.e. the distribution has density 0 at some point), then it is not invertible. For points evaluated within the support of the distribution this should not be problematic. Issues arising from a zero-valued coefficient a_j, for $j \leq L$ may arise if the number of components N is large compared to the complexity of the data being modelled, however these are relatively uncommon, and can be treated at runtime by adding a small $\epsilon \sim \mathcal{N}(0, 1^{-n})$.

Nonetheless, it is convenient to have a relatively simple closed form expression for the quantile function, which is not the case for many probability distributions. Figure 3.8 shows a fitted density on multi-modal test data, alongside the distribution function and a histogram of samples obtained by passing uniform random samples through the corresponding quantile function.

Finally it is convenient to find the mode(s) of the resulting distribution, as these will always appear at one of the f_j locations in the distribution so long as the density at the neighbouring points f_{j-1} and f_{j+1} are lower than at f_j.

In other words, f_j is a mode if

$$\begin{aligned} f_Y(f_j) &> f_Y(f_{j-1}) \\ a_j f_j + b_j &> a_{j-1} f_{j-1} + b_{j-1} \end{aligned}$$

and

$$\begin{aligned} f_Y(f_j) &> f_Y(f_{j+1}) \\ a_j f_j + b_j &> a_{j+1} f_{j+1} + b_{j+1}, \end{aligned}$$

where we have taken logarithms to arrive at each expression above. Only the first and second comparisons are required for the endpoints f_1 and f_N, respectively. Figure 3.9 plots the locations of the modes, with uncertainty estimates, for the same synthetic test data.

TODO briefly discuss PyTorch implementation and link to source code.

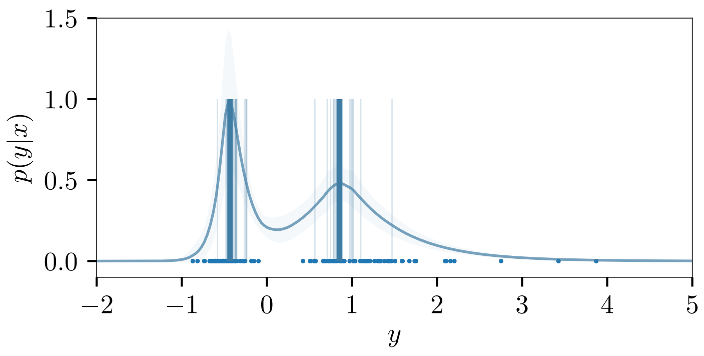


Figure 3.9: The mode locations can be determined by considering the density at each f_j.

4 EXPERIMENTS

4.1 RELATED WORK

TODO discuss prior work and trends in distributional RL

4.2 METHODS

TODO method for using pl3e in DRL algorithm Mention sampling, distributional Bellman update etc.

4.3 EXPERIMENTS

Copy and discuss results from classic control experiments

Include comparisons to QR-DQN, DQN, IQN, UADQN

4.3.1 *Directed exploration*

Include Deep Sea results, and discussion of directed exploration.

4.3.2 *Risk-Averse control*

Include computational experiments / gridworlds such as cliff walk

A

CONTRACTION MAPPINGS

In this appendix, we deal with the contraction mapping theorems which underlie RL in some more detail. The following exposition is largely guided by the work of Bellman (1958) and Denardo (1967).

In order to learn an optimal policy, we must consider the distance between functions (and later probability distributions), which requires a formal notion of distance between two objects in a set.

Definition A.1 (Metric Space)**.** Consider an ordered pair (X, d), for some set X, and a function defining a concept of *distance* between any two members of the set $d : M \times M \to \mathbb{R}$. This pair is called a *metric space* if the following conditions hold for any two elements $x, y \in X$:

- $d(x, y) = 0$ iff $x = y$,
- $d(x, y) = d(y, x)$,
- $d(x, z) \leq d(x, y) + d(y, z)$, for any other point $z \in X$.

Clearly the set of real numbers paired with the Euclidean distance is an example of a metric space.

We will want the metric spaces we work with to be *complete* to facilitate convergence proofs. We say that a metric space is complete when every Cauchy sequence is a convergent sequence in this space.

Definition A.2 (Cauchy sequence)**.** A sequence $x_1, x_2, \ldots = \{x_n\}$ in a metric space (X, d) is a *Cauchy sequence* if for every positive real number $\epsilon > 0$, there is a positive integer N_ϵ such that for all positive integers $m, n \geq N_\epsilon$,

$$d(x_m, x_n) < \epsilon.$$

Since the terms in a Cauchy sequence become arbitrarily close to each other, we can intuitively understand this property as ensuring that there are no points 'missing' from the set X.

As a simple example, we can use this property to show that the open interval $X = (0, 5) \in \mathbb{R}$, paired with the euclidean distance $d = \ell_2$ is not a complete metric space. The sequence $x_n = 1/n$ is Cauchy—for arbitrary $\epsilon > 0$, all terms x_n for $n > 1/5$ fit in the $(0, 5)$ interval—however the limit of this sequence, $\lim_{n\to\infty} x_n = 0$, does not belong to X, so it cannot be a complete metric space.

If we are working in a complete metric space, in order to prove that a sequence is convergent, it suffices to show that it is a Cauchy sequence. Before seeing how to learn control in an MDP, we also require the contraction mapping theorem.

Definition A.3 (Contraction)**.** Let (X, d) be a metric space, and f a mapping between any two elements in this space, $f : X \to X$. We call f a *contraction* (or *contraction mapping*) if there exists a real number $k \in [0, 1)$ such that for any $x, y \in X$

$$d(f(x), f(y)) \leq kd(x, y). \tag{A.1}$$

Intuitively, the contraction mapping reduces the distance between these two points in the metric space. This is a particularly useful concept when x, y are functions or probability distributions.

Definition A.4 (Banach Fixed Point Theorem)**.** For (X, d) a complete metric space, and $f : X \to X$ a contraction, there exists only one unique fixed point x^* such that

$$f(x^*) = x^*. \tag{A.2}$$

Moreover, if we begin with an arbitrary point x in the set X, and define $f^n(x)$ as the repeated composition of f with itself n times (that is, $f^n(x) = f(f^{n-1}(x))$, $f^1 = f(x)$), then

$$f^n(x) \to x^*, \text{ as } n \to \infty. \tag{A.3}$$

Proof. See Appendix **??**.

A.0.1 *Contractive Dynamic Programming*

We will now consider the classical approach to learning a value function from experience. For simplicity, in this section we will assume that the transition dynamics of the environment $\mathcal{P}(\cdot|x, a)$ are somehow known to the agent—in the next section we will remove this assumption.

The reader may be familiar with dynamic programming as a technique for algorithm design, however we use the term in this context to refer to a method for mathematical optimisation. The term carries broadly the same intuitions in the two settings—exploiting the sub-structure of the problem to derive an optimal solution.

Let V denote the set of all value functions. Recall that value functions are bounded mappings from states to the reals; that is, $v \in V$ iff $v : \mathcal{X} \to \mathbb{R}$ and $\sup_{x\in\mathcal{X}} |v(x)| < \infty$. We will define the distance metric between value functions as the ∞-norm:

$$d(u, v) = \sup_{x\in\mathcal{X}} |u(x) - v(x)|. \tag{A.4}$$

The space of value functions V is complete in this metric.

We will also introduce the new notation for the returns which is decoupled from the policy, $h(x, a, v)$. It retains the same intuitive meaning, here h is a mapping $h : \mathcal{X} \times \mathcal{A} \times V \to \mathbb{R}$ representing he total returns or '*payoff*' for starting in state x and selecting action a with the prospect of receiving $v(y)$, if taking action a in state x causes the transition to state y.

We use this notation to define the following operator, whose iterative application to an initial guess for the value function will, providing that it is a contraction mapping, converge to the actual value function v^π induced by the policy, as its unique fixed point.

Definition A.5 (Evaluation Operator)**.** Let $\mathcal{T}_\pi$ be an operator $\mathcal{T}_\pi : V \to V$, which maps a value function to another value function. It is defined as

$$[\mathcal{T}_\pi v](x) \doteq h(x, \pi(x), v), \tag{A.5}$$

for all $x \in \mathcal{X}$ and $v \in \mathcal{V}$, and we refer to this as the *evaluation operator*. (Note, this should not be confused with $\mathcal{T}^\pi$, previously defined as the trajectory distribution induced by π.)

We must ensure that the evaluation operator is a contraction mapping at the point of applying dynamic programming to the specific RL setting—for now we proceed by assumption.

Assumption A.1 (Contraction of $\mathcal{T}_\pi$)**.** For some constant γ satisfying $0 \le \gamma < 1$, then for all $u, v \in V$ let us assume that $\mathcal{T}_\pi$ is a contraction mapping. That is,

$$d(\mathcal{T}_\pi u, \mathcal{T}_\pi v) \le \gamma d(u, v). \tag{A.6}$$

Recall that the goal of value-based RL is to find the optimal value function, v^*, which the agent can use to make optimal decisions. We therefore require an operator whose fixed point is not v^π, but the optimal value function v^*.

Definition A.6 (Maximisation Operator)**.** Let $\mathcal{T}$ be a mapping with domain and range V. It is defined as

$$[\mathcal{T}v](x) \doteq \sup_{a \in \mathcal{A}} h(x, a, v), \tag{A.7}$$

for all $v \in V$ and $x \in \mathcal{X}$, and we refer to this as the *maximisation operator*. We must now show that the repeated application of the maximisation operator to an initial guess v converges to a fixed point $v^{\mathtt{max}}$, and further that this fixed point is equal to the optimal value function v^*.

Theorem A.1. If Assumption A.1 is satisfied; that is, $\mathcal{T}_\pi$ is a contraction mapping with constant γ, then $\mathcal{T}$ is also a contraction mapping with the same constant γ:

$$d(\mathcal{T}u, \mathcal{T}v) \leq \gamma d(u, v). \tag{A.8}$$

Proof. Consider an arbitrary policy $\pi \in \Pi$ and any two value functions $u, v \in V$. Since $\mathcal{T}_\pi$ is a contraction mapping (by assumption), then $[\mathcal{T}_\pi u](x) \leq [\mathcal{T}_\pi v](x) + \gamma d(u, v)$. Fixing some state $x \in \mathcal{X}$, we have

$$\begin{aligned}\sup_{\pi \in \Pi} [\mathcal{T}_\pi u](s) &= \sup_{\pi(x) \in \mathcal{A}} h(x, \pi(x), u) \\ &= [\mathcal{T}u](x) \\ &\leq [\mathcal{T}v](x) + \gamma d(u, v)\end{aligned}$$

By the symmetry of d, we also have that $[\mathcal{T}v](x) \leq [\mathcal{T}u](x) + \gamma d(v, u)$, which implies

$$|[\mathcal{T}v](x) - [\mathcal{T}u](x)| \leq \gamma d(u, v). \tag{A.9}$$

Finally, by Equation A.4 we get

$$d(\mathcal{T}u, \mathcal{T}v) \leq \gamma d(u, v). \tag{A.10}$$

$\square$

Hence by the Banach fixed point theorem, repeatedly updating $v \leftarrow \mathcal{T}u$ will converge to some unique fixed point, $v^{\mathtt{max}}$. The question now becomes whether this fixed point is the optimal value function.

We will show this in two parts, beginning with the somewhat more intuitive case for $v^* \geq v^{\mathtt{max}}$, followed by the less obvious case for $v^{\mathtt{max}} \geq v^*$ which will require an additional monotonicity condition. We will then conclude that $v^* = v^{\mathtt{max}}$.

Corollary A.1 (Denardo (1967), Corollary 1)**.** For any small positive constant $\epsilon > 0$, and any state $x \in \mathcal{X}$, there exists a policy π_ϵ such that

$$h(x, \pi_\epsilon(x), v^{\mathtt{max}})) \geq \sup_{a \in \mathcal{A}} h(x, a, v^{\mathtt{max}}) - \epsilon(1 - \gamma), \tag{A.11}$$

and thus any such π_ϵ satisfies $v^{\pi_\epsilon} \geq v^{\mathtt{max}} - \epsilon$, and by extension

$$v^* \geq v^{\mathtt{max}} - \epsilon. \tag{A.12}$$

The proof of the above (omitted for brevity) relies on the triangle inequality and the result that $\mathcal{T}$ is a contraction mapping. For the reverse inequality, we require the following assumption.

Assumption A.2 (Monotonicity)**.** For any two $u, v \in V$, and state $x \in \mathcal{X}$, we write $u \geq v$ if $u(x) \geq v(x)$. The monotonicity assumption requires that if if $u \geq v$, then $\mathcal{T}_\pi u \geq \mathcal{T}_\pi v$.

With this additional assumption in place, we can finally show the equality of the fixed point of the maximisation operator, and the optimal value function.

Theorem A.2 (Denardo (1967), Theorem 3)**.** If the contraction assumption (A.1) and monotonicity assumptions (A.2) hold, then $v^{\mathtt{max}} = v^*$.

Proof. For any $x \in \mathcal{X}$ and policy $\pi \in \Pi$, we have that $h(x, \pi(x), v^{\mathtt{max}}) \leq \sup_{a \in \mathcal{A}} h(s, a, v^{\mathtt{max}})$. Therefore recursive applications of the monotonicity assumption n times yields

$$\mathcal{T}_\pi^n v^{\mathtt{max}} \leq \mathcal{T}_\pi^{n-1} v^{\mathtt{max}} \ldots \leq \mathcal{T}_\pi^1 v^{\mathtt{max}} \leq \mathcal{T} v^{\mathtt{max}} = v^{\mathtt{max}}. \tag{A.13}$$

Appealing to the contraction property, $\lim_{n \to \infty} \mathcal{T}_\pi^n v^{\mathtt{max}} = v^\pi$, therefore $v^\pi \leq v^{\mathtt{max}}$ for any policy π, which implies that $v^* \leq v^{\mathtt{max}}$. Using the result from Corollary A.1, we have $v^{\mathtt{max}} - \epsilon \leq v^*$ for any positive ϵ. Combining these results gives

$$v^{\mathtt{max}} - \epsilon \leq v^* \leq v^{\mathtt{max}}. \tag{A.14}$$

We conclude that $v^{\mathtt{max}} = v^*$. □

CALCULATIONS FOR THE DISTRIBUTION PROPERTIES

This appendix holds the calculations for some of the properties of the distributions defined by a piecewise linear log-likelihood.

B.1 MEAN CALCULATION

In this section we evaluate the integral $\mathbb{E}[Y] = \int_{-\infty}^{\infty} y f_Y(y)\ dy$, where recall the density function is of the form

$$f_Y(y) = e^{a_j y + b_y},$$

with $j \in \{0, \dots, N\}$ such that $f_j < y \leq f_{j+1}$.

We begin by evaluating the indefinite integral $\int xe^{ax+b}dx$. Letting $u = ax + b$, we have

$$\begin{aligned}
\int xe^{ax+b}dx &= \frac{1}{a}\int \frac{u-b}{a} e^u du \\
&= \frac{1}{a^2}\int e^u(u-b)du \\
&= \frac{1}{a^2}\int e^u u - be^u du \\
&= \frac{1}{a^2}\int e^u u\ du - \frac{b}{a^2}\int e^u du
\end{aligned}$$

We can integrate the first term above by parts, setting $f = u$ and $dg = e^u$. Now,

$$\begin{aligned}
\frac{1}{a^2}\int e^u u\ du - \frac{b}{a^2}\int e^u du &= \frac{1}{a^2}\left(e^u u - \int e^u\ du\right) - \frac{b}{a^2}\int e^u du \\
&= \frac{e^u u}{a^2} + \left(-\frac{b}{a^2} - \frac{1}{a^2}\right)\int e^u\ du \\
&= \frac{e^u u}{a^2} - \frac{(b+1)e^u}{a^2} + c \\
&= \frac{e^{ax+b}(ax+b)}{a^2} - \frac{(b+1)e^{ax+b}}{a^2} + c \\
&= \frac{e^{ax+b}(ax-1)}{a^2} + c.
\end{aligned}$$

Having found this indefinite integral for the expectation of individual segments of the distribution, we can find the integral of each segment from $-\infty$ to ∞. As previously, we consider three cases: for the first when $-\infty < y < f_1$ we have

$$\begin{aligned}
\lim_{\ell\to-\infty}\int_{\ell}^{f_1} y f_Y(y)\,dy &= \lim_{\ell\to-\infty}\int_{\ell}^{f_1} y e^{a_0 y+b_0} dy \\
&= \lim_{\ell\to-\infty}\left[\frac{e^{a_0 y+b_0}(a_0 y-1)}{a_0^2}\right]_{\ell}^{f_1} \\
&= \lim_{\ell\to-\infty}\frac{e^{a_0 f_1+b_0}(a_0 f_1-1)}{a_0^2} - \frac{e^{a_0\ell+b_0}(a_0\ell-1)}{a_0^2} \\
&= \frac{e^{a_0 f_1+b_0}(a_0 f_1-1)}{a_0^2} - \lim_{\ell\to-\infty}\left(\frac{e^{a_0\ell+b_0}a_0\ell}{a_0^2} - \frac{e^{a_0\ell+b_0}}{a_0^2}\right) \\
&= \frac{e^{a_0 f_1+b_0}(a_0 f_1-1)}{a_0^2} - \lim_{\ell\to-\infty}\frac{\ell}{e^{-(a_0\ell+b_0)}a_0},
\end{aligned}$$

where the final step above follows from the fact that $a_0 > 0$. Now, by L'Hôpital's rule,

$$\lim_{\ell\to-\infty}\frac{\frac{d}{d\ell}\ell}{\frac{d}{d\ell}e^{-(a_0\ell+b_0)}a_0} = \lim_{\ell\to-\infty}\frac{1}{-a_0^2 e^{-a_0\ell}-b_0} = 0,$$

once again, since $a_0 > 0$. Hence for the first integral,

$$\lim_{\ell\to-\infty}\int_{\ell}^{f_1} y f_Y(y)\,dy = \frac{e^{a_0 f_1+b_0}(a_0 f_1-1)}{a_0^2}. \tag{B.1}$$

For the second case where $f_j \le y < f_{j+1}$ for some $j \in \{1,\ldots,N\}$, we have

$$\begin{aligned}
\int_{f_j}^{f_{j+1}} y f_Y(y) dy &= \left[\frac{e^{a_j y+b_j}(a_j y-1)}{a_j^2}\right]_{f_j}^{f_{j+1}} \\
&= \frac{e^{a_j f_{j+1}+b_j}(a_j f_{j+1}-1)}{a_j^2} - \frac{e^{a_j f_j+b_j}(a_j f_j-1)}{a_j^2} \\
&= \frac{e^{b_j}\left(e^{a_j f_{j+1}}\left(a_j f_{j+1}-1\right) - e^{a_j f_j}\left(a_j f_j-1\right)\right)}{a_j^2}.
\end{aligned}$$

Finally for the last case where $f_N \le y < \infty$, we have

$$\begin{aligned}
\lim_{\ell\to\infty}\int_{f_N}^{\ell} y f_Y(y)\,dy &= \lim_{\ell\to\infty}\left[\frac{e^{a_N y+b_N}(a_N y-1)}{a_N^2}\right]_{f_N}^{\ell} \\
&= \lim_{\ell\to\infty}\frac{e^{a_N\ell+b_N}(a_N\ell-1)}{a_N^2} - \frac{e^{a_N f_N+b_N}(a_N f_N-1)}{a_N^2} \\
&= \lim_{\ell\to\infty}\frac{e^{a_N\ell+b_N}a_N\ell}{a_N^2} - \frac{e^{a_N\ell+b_N}}{a_N^2} - \frac{e^{a_N f_N+b_N}(a_N f_N-1)}{a_N^2} \\
&= \lim_{\ell\to\infty}\frac{e^{a_N\ell+b_N}a_N\ell}{a_N^2} - \frac{e^{a_N f_N+b_N}(a_N f_N-1)}{a_N^2} && \text{(since } a_N < 0\text{)} \\
&= -\frac{e^{a_N f_N+b_N}(a_N f_N-1)}{a_N^2}. && \text{(by L'Hôpital's rule, as above)}
\end{aligned}$$

Therefore the expected value of y is:

$$\begin{aligned}
\mathbb{E}_Y[y] &= \int_{-\infty}^{\infty} y f_Y(y)\, dy \\
&= \frac{e^{a_0 f_1 + b_0}(a_0 f_1 - 1)}{a_0^2} \\
&+ \sum_{j=1}^{N-1} \frac{e^{b_j}\left(e^{a_j f_{j+1}}\left(a_j f_{j+1} - 1\right) - e^{a_j f_j}\left(a_j f_j - 1\right)\right)}{a_j^2} \\
&- \frac{e^{a_N f_N + b_N}(a_N f_N - 1)}{a_N^2}
\end{aligned}$$

B.2 VARIANCE CALCULATION

To find the variance of the distribution, we must evaluate the following integral:

$$\mathrm{Var}(Y) = \int_{-\infty}^{\infty} y^2 f_Y(y)\, dy - \mu^2, \tag{B.2}$$

for μ the mean of the distribution, as calculated above.

BIBLIOGRAPHY

G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap. Distributed Distributional Deterministic Policy Gradients. *arXiv:1804.08617 [cs, stat]*, Apr. 2018. URL http://arxiv.org/abs/1804.08617.

M. G. Bellemare, W. Dabney, and R. Munos. A Distributional Perspective on Reinforcement Learning. *arXiv:1707.06887 [cs, stat]*, July 2017. URL http://arxiv.org/abs/1707.06887.

R. Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, Aug. 1952. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.38.8.716. URL https://www.pnas.org/content/38/8/716.

R. Bellman. Dynamic programming and stochastic control processes. *Information and Control*, 1 (3):228–239, Sept. 1958. ISSN 0019-9958. doi: 10.1016/S0019-9958(58)80003-0. URL https://www.sciencedirect.com/science/article/pii/S0019995858800030.

Y. Bengio and Y. Lecun. Scaling learning algorithms towards AI. *Large-scale kernel machines*, 2007. URL https://nyuscholars.nyu.edu/en/publications/scaling-learning-algorithms-towards-ai.

W. R. Clements, B. Van Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth. Estimating Risk and Uncertainty in Deep Reinforcement Learning. *arXiv:1905.09638 [cs, stat]*, Sept. 2020. URL http://arxiv.org/abs/1905.09638.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec. 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional Reinforcement Learning with Quantile Regression. *arXiv:1710.10044 [cs, stat]*, Oct. 2017. URL http://arxiv.org/abs/1710.10044.

W. Dabney, G. Ostrovski, D. Silver, and R. Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. In *International Conference on Machine Learning*, pages 1096–1105. PMLR, July 2018. URL http://proceedings.mlr.press/v80/dabney18a.html.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-Learning. *AAAI*, AAAI-98:8, 1998. URL www.aaai.org.

E. V. Denardo. Contraction Mappings in the Theory Underlying Dynamic Programming. *SIAM Review*, 9(2):165–177, 1967. ISSN 0036-1445. URL https://www.jstor.org/stable/2027440.

B. Farley and W. Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, Sept. 1954. ISSN 2168-2704. doi: 10.1109/TIT.1954.1057468.

A. Graves. Practical Variational Inference for Neural Networks. page 9.

P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. Wilson. What Are Bayesian Neural Network Posteriors Really Like? *arXiv:2104.14421 [cs, stat]*, Apr. 2021. URL http://arxiv.org/abs/2104.14421.

R. Koenker and G. Bassett. Regression Quantiles. *Econometrica*, 46(1):33, Jan. 1978. ISSN 00129682. doi: 10.2307/1913643. URL https://www.jstor.org/stable/1913643?origin=crossref.

T. J. Kozubowski. A Class of Asymmetric Distributions. page 22, 1999.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6405–6416, Red Hook, NY, USA, Dec. 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

C. Louizos and M. Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *arXiv:1603.04733 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1603.04733.

D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992. URL https://resolver.caltech.edu/CaltechETD:etd-01042007-131447.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

M. Minsky. Theory of neural-analog reinforcement systems and its application to the brain-model problem. 1954.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, Dec. 2013. URL http://arxiv.org/abs/1312.5602.

T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka. Parametric return density estimation for reinforcement learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 368–375, Arlington, Virginia, USA, July 2010. AUAI Press. ISBN 978-0-9749039-6-5.

R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996. ISBN 978-0-387-94724-2 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL http://link.springer.com/10.1007/978-1-4612-0745-0.

T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely. Uncertainty in Neural Networks: Approximately Bayesian Ensembling. *arXiv:1810.05546 [cs, stat]*, Feb. 2020. URL http://arxiv.org/abs/1810.05546.

K. J. Radford. Decision-making Under Conditions of Uncertainty. In K. J. Radford, editor, *Individual and Small Group Decisions*, pages 41–64. Springer, New York, NY, 1989. ISBN 978-1-4757-2068-6. doi: 10.1007/978-1-4757-2068-6_3. URL https://doi.org/10.1007/978-1-4757-2068-6_3.

C. Rasmussen and Z. Ghahramani. Occam's Razor. Cambridge, Mass., 2001. MIT Press. ISBN 978-0-262-12241-2.

M. Rowland, R. Dadashi, S. Kumar, R. Munos, M. G. Bellemare, and W. Dabney. Statistics and Samples in Distributional Reinforcement Learning. *arXiv:1902.08102 [cs, stat]*, Feb. 2019. URL http://arxiv.org/abs/1902.08102.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://www.nature.com/articles/nature16961.

G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, Mar. 1995. ISSN 0001-0782. doi: 10.1145/203330.203343. URL https://doi.org/10.1145/203330.203343.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. Ph.D., University of Cambridge, 1989. URL https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.330022.

A. G. Wilson and P. Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *arXiv:2002.08791 [cs, stat]*, Apr. 2020. URL http://arxiv.org/abs/2002.08791.

J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. Efficiently Sampling Functions from Gaussian Process Posteriors. *arXiv:2002.09309 [cs, stat]*, July 2020. URL http://arxiv.org/abs/2002.09309.

D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T. Liu. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. *arXiv:1911.02140 [cs, stat]*, Aug. 2020. URL http://arxiv.org/abs/1911.02140.