Studying the Applications of Probability Metrics and Divergence Measures in Solving Classic Control Tasks

by

Kartik Kaul Aima

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 12, 2020

Keywords: Machine Learning

Approved by

Saad Biaz, Chair, Professor of Computer Science and Software Engineering
Bo Liu, Assistant Professor of Computer Science and Software Engineering
Levent Yilmaz, Professor of Computer Science and Software Engineering

Abstract

Choosing the correct statistical distance for a machine learning problem is vital when estimating the degree of dissimilarity between two discrete distributions. In the distributional reinforcement learning problem, the distribution of returns that can be obtained by an agent are approximated across the entirety of the state space. To describe the expected behavior of the agent as it interacts with the environment in the distributional setting, the C51 algorithm initially proposed using the Wasserstein distance due to the convergence guarantees it offered for the policy evaluation problem. However due to the biased sample gradients produced by the Wasserstein distance, the KL divergence was ultimately used as the categorical loss function in the C51 algorithm.

In this thesis we studied two potential class of statistical distances and empirically observed their performance as viable categorical loss functions in the C51 algorithm as compared to the KL divergence. The first were probability metrics such as the Sinkhorn divergence and the Energy distance which attempt to alleviate the poor sample and computational complexity of the exact Wasserstein distance. The second were divergence measures that were instances of both the f divergence and α divergence. We studied the training time and testing time performance of these variations on the Acrobot and Cartpole environments.

We demonstrated that the statistical distances most suitable for approximating value distributions in these environments were divergence measures that possessed the zero-avoiding property or an amalgamation of zero-avoiding and zero-forcing properties. Strictly zero-forcing divergence measures were unsuitable for use as a categorical loss function in these environments. The Sinkhorn divergence was ill suited to serve as a categorical loss function whereas the Energy distance demonstrated evidence of learning in these environments, although its training performance paled in comparison to the more successful crop of divergence measures. This indicated that if an optimal transport based categorical loss function was to be used in the C51 algorithm, maximal entropic regularization would have to be applied.

Acknowledgments

First and foremost I would like to express my gratitude to my advisor Professor Saad Biaz for his guidance and mentorship through the course of my studies as a graduate student. I would also like to thank my committee members, Professor Yilmaz and Professor Liu.
I would also like to thank my friends, Vineet Nayak, Peijie Chen and Brodderick Rodriguez for their insight and advice which I sought while writing my thesis.

Finally, I would like to thank my parents for their support and encouragement, without which I would not have been able to complete my thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

Reinforcement learning is the study of the sequential decisions made by an agent, forced to optimize its behaviour through self-learning with the goal of amassing the largest possible reward whilst minimizing potential penalties incurred. It comes under the umbrella of machine learning along with supervised learning and unsupervised learning. In other branches of engineering, the same study of optimal sequential decision making is referred to as *optimal control*. The only difference between the two being that reinforcement learning looks to approach a problem with the perspective of maximization of long-term reward whereas optimal control approaches the problem with the perspective of minimization of long-term cost.

The reinforcement learning problem involves an agent interacting with a environment through trial and error, obtaining a reward or penalty for each action taken with no assisted supervision. There may be certain actions that the agent takes that provide instantaneous reward, however over the long-term horizon certain actions may negatively affect the agent's cumulative reward. In addition, the agent's realization of whether a particular action was helpful or consequential happens many steps into the future. Hence the agent should not greedily try to chase actions that will beget it greater rewards at the current time step at the cost of diminished long term rewards. Since the environment may not reward the agent in the short term for its actions, the agent must select actions in the here and now while planning ahead for greater cumulative reward for a longer time horizon.

As each decision is made sequentially, what decision an agent makes at the current time-step greatly determines what actions it shall make at a future time step. Actions that the agent

takes in future time steps will be predicated on the choice of actions that the agent made in previous iterations i.e. subsequent interactions in the environment are dependent on the choices made by the agent at previous iterations. Therefore the planning process must be meticulously designed to turn a blind eye to actions begetting short term rewards that may ultimately restrict the agent to an inferior selection of actions in the long time horizon eventually leading to diminished cumulative rewards.

A key breakthrough in trying to provide more information to the agent about its possible choice of actions was the work by Bellemare et al. (2017a) which rather than modelling the expected return obtained by the agent, instead approximated the distribution of the return that the agent can obtain for a given action. This approach, termed distributional reinforcement learning, has the benefit of allowing the agent to choose a particular action based on the risk that the user is comfortable with. This can be useful for instance in problem settings where a high priority is ascribed to safe exploration and taking risky and unknown actions that can lead the agent to hindrances are to be avoided at all costs. Thus the agent will be able to distinguish between riskier actions that may generate greater rewards but at a lower probability as well as safer actions that while generating a lower reward are more likely to produce it. This cannot be gleaned from the standard expectational perspective to reinforcement learning. In addition, forcing the agent to learn the underlying structure of the rewards forces the function approximation architecture to learn more about the environment. This may be beneficial to the agent and improve its data efficiency, assuming that it learns information that is of some benefit.

The C51 algorithm proposed by Bellemare et al. (2017a) originally proposed the Wasserstein to minimize the statistical distance between the sample value distribution and the target value distribution. The Wasserstein distance was so chosen due to its theoretical properties that could provide guarantees of the agent's behaviour for the policy evaluation problem similar to results present in the standard expectational reinforcement learning setting. However empirically, the sample Wasserstein distance did not provide a good approximation of the underlying distributional loss and hence the authors opted instead to use the KL divergence.

In this thesis we study entropic regularization based algorithms used in the optimal transport literature that attempt to mitigate the computational complexity and statistical complexity of the exact Wasserstein distance. We also study the family of f divergences that are used to quantify dissimilarity between two distributions using a convex function of their ratio. We specifically choose divergence measures that are also members of the class of α divergences so that we can interpret the desirable properties a divergence measure must possess in order to be successful in this domain.

Finally we test the candidacy of these probability metrics and divergence measures as the categorical loss function for the C51 algorithm in contrast to the baseline model which incorporated the KL divergence.

The organization of this thesis is as follows:

In chapter 2 we formulate the reinforcement learning problem as a Markov Decision Process (MDP) and introduce the value iteration and Q-learning algorithms. We then study reinforcement learning algorithms that utilize deep neural networks as function approximators such as the DQN algorithm and then introduce the C51 algorithm which incorporates the aforementioned distributional setting.

Having discussed the C51 algorithm, in chapter 3 we discuss viable categorical loss functions between discrete measures that can be used in place of the KL divergence. We first study Integral Probability Metrics (IPM) such as the entropy regularized Wasserstein distance and the Energy distance. We then study f divergences and provide insights into how the different instances of the f divergences differ on the method of quantifying dissimilarity by studying the properties of α divergences.

After discussing each probability metric and divergence measure, in chapter 4 we contrast the performance of variants of the C51 algorithm each of which utilize the aforementioned statistical distances as their categorical loss functions. We use the Acrobot and Cartpole environments for training and testing these variants. We close this chapter by interpreting the

performance of the loss functions through the less of zero forcing and zero avoiding properties of probability metrics and divergence measures.

In chapter 5 we finally conclude and discuss the future directions we would like to explore in the field of distributional reinforcement learning.

Chapter 2

Background

In this section we discuss the fundamental concepts in reinforcement learning. We begin by formulating the problem as a Markov Decision Process and then obtain the tabular based algorithm called Value Iteration (Bellman, 1957). We then study how it can applied to more complex settings through function approximation and sampling resulting in the Q-learning algorithm Watkins and Dayan (1992). Our discussion on reinforcement learning is based on Schulman et al. (2017).

Next we discuss deep reinforcement learning algorithms which integrate deep neural networks as value function approximators. The first such algorithm is DQN (Mnih et al., 2015) which overcame the deficiencies of the Q learning algorithm and was able to successfully play most games from the Arcade Learning Environment (Bellemare et al., 2013). Finally we discuss the C51 algorithm (Bellemare et al., 2017a) and its central idea of approximating value distributions. We also discuss the theoretical motivations for this approach as well as the method used in practice that resulted in superior performance than the DQN algorithm.

2.1 Reinforcement Learning

In this section the define the reinforcement learning problem in the context of a Markov Decision Process wherein an agent obtains rewards at each time step for interacting with an environment with the ultimate goal being to maximize the cumulative rewards obtained at the end of an episode. The primary means to maximize the cumulative reward is to find the optimal mapping at a given state to a particular action. With this goal in mind we study the Value Iteration

algorithm which tries to accomplish just that. Finally we study the Q-learning algorithm which is a sample based algorithm that estimates the transition dynamics of the model thereby bypassing prohibitive constraints of the Value Iteration algorithm making it suitable for a variety of applications.

2.1.1 Markov Decision Process (MDP)

The goal in reinforcement learning is to choose actions so as to maximize long term future rewards. The procedures that an agent must employ to successfully navigate the environment are generally described in the context of a *Markov Decision Process (MDP)*.

Given a discretized setting, the MDP outlines the following process: at each time step t, an agent interacts with the environment from a given state s by taking actions a_t and receiving rewards r_t thereby moving to a new state s'. The goal of a reinforcement learning agent is to maximize cumulative future reward over the entirety of time-steps the agent interacts with the environment. To generalize this concept of an MDP, we'll utilize the following set of definitions for its components:

Let S denote the *state space*, which consists of all possible states s that the agent can be in when interacting with the environment.

Let A denote the *action space*, which consists of the set of all possible actions a which the agent can take at a given state s at time t. Not all actions in the state space A will be available to the agent at a given time and state.

Let $P\left(r, s' | s, a\right)$ denote the *state transition probability distribution*, which given the current state and action taken, predicts the next state and the reward obtained. This formulation can be used for both stochastic and deterministic transition functions. For inherent stochasticity in the transitions, the return becomes a random variable and thus the objective becomes to maximize the expected return.

Let π denote the *policy*, which determines the action taken by the agent in the environment at any state. If the state transitions are deterministic then the policy is a *deterministic policy*, where the action chosen at a state is determined by, $a = \pi(s)$. Otherwise if the state transitions are stochastic then the policy is a *stochastic policy*, where the action chosen is conditioned on

the state, $a \sim \pi(a|s)$.

Let γ denote the *discount factor*, which serves as a coefficient to the reward obtained at each state. The discount factor, γ down-weights rewards in the more distant future thereby deprioritizing their importance to the agent and forcing it to focus on maximizing rewards over a relatively shorter horizon.

In contrast to traditional machine learning which involves optimizing a single objective function, by utilizing an MDP the reinforcement learning problem is broken down into two stages:

The **Policy Evaluation** problem deals with obtaining the expected sum of rewards (also known as the *return*) for a given policy π. The return is also a random variable. The undiscounted return of an agent interacting in an environment for T time-steps, following a policy π is expressed as:

$$\eta(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} r_t\right] = r_t + r_{t+1} + \cdots + r_{T-1} + r_T$$

To ensure that the sum of rewards results in a finite quantity, a discount factor γ is utilized which serves as a coefficient to each individual rewards through a geometric series of weights $(1, \gamma, \gamma^2, \ldots)$. Since $\gamma \in [0, 1)$ the discounted expected return for T time-steps, following a policy π will be a finite scalar sum expressed as:

$$\eta(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right] = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots + r_{T-1} + r_T$$

The **Policy Optimization** problem deals with maximizing the expected sum of rewards obtained by the agent over the space of possible policies, whilst following a given policy for the entire length of an episode. For a given stochastic environment that an agent interacts with for T time steps, this involves maximizing the following expression:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} r_t\right]$$

We restrict our analysis to maximizing the expected sum of returns over a fixed horizon of T time steps. This is referred to as the *fixed horizon episodic* setting.

Our episodic setting is formulated as a stochastic process where the first step involved is sampling an initial state from a probability distribution: $s_0 \sim \mu(s_0)$. For a given policy π, the agent samples the first action: $a_0 \sim \pi(a_0|s_0)$. The reward obtained from taking that action and the resultant next state the agent transitions to is sampled from the state transition matrix: $s_1, r_0 \sim P(s_1, r_0|s_0, a_0)$. At this point the process is repeated by sampling the next action.

This entire process, is referred to as *sampling a trajectory*. The trajectory terminates when the agent choose its final action $a_{T-1} \sim \pi(a_{T-1} \mid s_{T-1})$ and the state transition matrix determines the final reward and state: $s_T, r_{T-1} \sim P(s_T|s_{T-1}, a_{T-1})$.

We next define the **state-value function** as the expected sum of future rewards for the length of an episode given that it is present at a specific state. It is expressed as:

$$\begin{aligned} V^{\pi,\gamma}(s) &= \mathbb{E}_\pi \left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \mid s_0 = s \right] \\ &= \mathbb{E}_{a\sim\pi} \left[Q^{\pi,\gamma}(s,a) \right] \end{aligned}$$

We define the **state-action-value function** as the expected sum of future rewards that can be obtained by the agent for the length of an episode given that it is present at a specific state and takes a specific action. It is also referred to as the **Q-function** and expressed as:

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}_\pi \left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \mid s_0 = s, a_0 = a \right]$$

2.1.2 Value Iteration

The **Value Iteration (VI)** algorithm (Bellman, 1957) attempts to maximize an agent's expected return whilst interacting in an environment for an episode of length T. We'll denote the reward obtained at the final time step, s_T as the terminal reward $V_T(s_T)$.

The Value Iteration algorithm maximizes the expected return for each intermediate policy chosen by the agent at each time step:

$$\max_{\pi_0} \max_{\pi_1} \ldots \max_{\pi_T - 1} \mathbb{E} \left[r_0 + r_1 + \cdots + r_{T-1} + V_T(s_T) \right]$$

As the immediate rewards obtained at transition at earlier time steps are not dependent on the choice of policies at subsequent time steps they can be separated from the rest of the expression:

$$\max_{\pi_0} \mathbb{E}\left[r_0 + \max_{\pi_1} \mathbb{E}\left[r_1 + \cdots + \max_{\pi_{T-1}} \mathbb{E}\left[r_{T-1} + V_T\left(s_T\right)\right]\right]\right]$$

The nested problem is simply the tail sub-problem where a policy, $\pi_{T-1}(s)$ has to be chosen at the penultimate time step, $T-1$ such that the agent can obtain the value function at the penultimate time step which is given by: $V_{T-1}^{\pi-1}(s) = \max_a \mathbb{E}_{s_T}\left[r_{T-1} + V_T\left(s_T\right)\right]$

We can substitute this in our original VI expression to finally obtain the following expression:

$$\max_{\pi_0} \mathbb{E}\left[r_0 + \max_{\pi_1} \mathbb{E}\left[r_1 + \cdots + \max_{\pi_{T-2}} \mathbb{E}\left[r_{T-2} + V_{T-1}\left(s_{T-1}\right)\right]\right]\right]$$

Algorithm 1 Value Iteration (Bellman, 1957)

for $t = T-1, T-2, \ldots, 0$ **do**

for $s \in S$ **do**

$\pi_t(s), V_t(s) = \max_a \mathbb{E}\left[r_t + V_{t+1}\left(s_{t+1}\right)\right]$

end for

end for

Thus the VI algorithm involves solving tail sub-problems of longer time lengths using the solution obtained previously for tail sub-problems of shorter lengths. Specifically, the agent would begin by solving the pen-ultimate transition and use its solution to solve the antepenultimate tail sub-problem. In such a manner it would solve tail sub-problems recursively one by one which would take it all the way to the initial state s_0. At the point of convergence, the agent would be able to obtain the optimal value function V^* (Bertsekas, 2019).

We can therefore consider the successive solving of the tail sub-problems of the VI algorithm as repeated applying an operation at the current time step of the state-value function so as to obtain the state-value function at the previous time step. We'll denote these successive operations using a function known as the **Bellman backup operator** and will express the operation as:

$$[\mathcal{T}V](s) = \max_a \mathbb{E}_{s'|s,a} \left[r + \gamma V\left(s'\right) \right]$$

Theoretical results have shown that the Bellman backup operation is a γ contraction in the ∞-norm. Using the contraction mapping principle, it's been shown that through repeated applications of the Bellman backup operator, the VI algorithm converges to the optimal state-value function V^* (Bertsekas and Tsitsiklis, 1996).

2.1.3 Q-Learning

An issue with using the Value Iteration algorithm is that the transition dynamics and the reward function of the agent as it moves from one state to the next must be known by it i.e. the *model* of the system must be known. Only after this strict constraint was satisfied could a dynamic programming based algorithm be used to obtain the optimal state-value function. To avoid this strict constraint, we will use a *model-free* algorithm, which utilize the past interactions of the agent in the environment to obtain a proxy for the transition dynamics and the reward to solve both the policy evaluation and policy optimization problems.

We'll begin by formulating the Bellman backup for the state-action value function using the expression for the Bellman backup for state-value function that we previously defined for a one-step lookahead:

$$\begin{aligned} Q^\pi\left(s_0, a_0\right) &= \mathbb{E}_{s_1 \sim P(s_1|s_0,a_0)} \left[r_0 + \gamma V^\pi\left(s_1\right) \right] \\ &= \mathbb{E}_{s_1 \sim P(s_1|s_0,a_0)} \left[r_0 + \gamma \mathbb{E}_{a_1 \sim \pi} \left[Q^\pi\left(s_1, a_1\right) \right] \right] \end{aligned}$$

We can also write the above expression in terms of a Bellman operator that is successively applied to evaluate policy π by solving the *Bellman expectation* equation:

$$\left[\mathcal{T}^\pi Q\right]\left(s_0, a_0\right) = \mathbb{E}_{s_1 \sim P(s_1|s_0,a_0)} \left[r_0 + \gamma \mathbb{E}_{a_1 \sim \pi} \left[Q\left(s_1, a_1\right) \right] \right]$$

As we had explained previously for the state-value function, due to the contraction mapping principle, repeated application of the Bellman backup operator on the state-action-value function will lead to the fixed point (Bertsekas and Tsitsiklis, 1996): $Q, \mathcal{T}^{\pi}Q, \left(\mathcal{T}^{\pi}\right)^2 Q, \cdots \rightarrow Q^{\pi}$

Analogous to the optimal state-value function, V^* we can denote Q^* as the optimal state-action-value function. We'll define the optimal state-action-value-function as the Q function when the agent follows the the optimal policy π^*.
It can be expressed as a maximization of state-action-value function over the space of all possible policies π:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

From the Bellman expectation equation in place of the state-action-value function following a policy π, we can substitute the optimal state-action-value function following the optimal policy π^*:

$$Q^*\left(s_0, a_0\right) = \mathbb{E}_{s_1 \sim P(s_1|s_0,a_0)}\left[r_0 + \gamma \max_{a_1} Q^*\left(s_1, a_1\right)\right]$$

We can also write the above expression in terms of a Bellman operator that is successively applied to obtain the optimal policy π^* by solving the *Bellman optimality* equation:

$$\left[\mathcal{T}Q\right]\left(s_0, a_0\right) = \mathbb{E}_{s_1 \sim P(s_1|s_0,a_0)}\left[r_0 + \gamma \max_{a_1} Q\left(s_1, a_1\right)\right]$$

Again, due to the contraction mapping principle, repeated application of the Bellman backup operator will lead to the fixed point (Bertsekas and Tsitsiklis, 1996): $Q, \mathcal{T}Q, \mathcal{T}^2Q, \cdots \rightarrow Q^*$.

To make these reinforcement learning algorithms work in model free settings, we have to remove the constraint of requiring the transition dynamics i.e. requiring an underlying model of the system. To do this we'll utilize sample updates from the current state to the next state with the agent collecting immediate rewards. These samples will provide an unbiased estimator of both the Bellman backup equation and the Bellman optimality equation (Bertsekas et al., 1995) (Jaakkola et al., 1994):

$$[\widehat{\mathcal{T}Q}]\left(s_0, a_0\right) = r_0 + \gamma \max_{a_1} Q\left(s_1, a_1\right)$$

Therefore to obtain a sampled version of the value iteration algorithm utilizing the state-action-value function instead of the state-value function, we'll simply utilize the individual samples as unbiased estimators of the Q-function for the entirety of a trajectory:

$$[\mathcal{T}Q]\left(s_t, a_t\right) \rightarrow \widehat{\mathcal{T}Q}_t = r_t + \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right)$$

In addition, instead of using the average of the unbiased estimates as our update to the state-action-value function, we can use that particular state-action-value function (over the space of all possible state-action-value function) as the update that minimizes the ℓ^2 norm between the unbiased sample estimate and the Q-function for all T time steps:

$$Q^{(n+1)}(s, a) = \arg\min_{Q} \sum_{t=1}^{T} \left\| \widehat{\mathcal{T}\mathcal{Q}}_t - Q\left(s_t, a_t\right) \right\|^2$$

Its important to note that, while using a sample based version of the VI has permitted bypassing the constraint of having perfect knowledge of the transition dynamics, the issue that arises with using an unbiased estimate is that, the inherent noise added in every transition can pull the update away from the desired contraction mapping provided by the Bellman operator. This weakens the convergence guarantees that the original VI algorithm provided.

To generalize the applicability of the Value Iteration algorithm to environments that have high dimensionality we forgo the setting where the state-action values for all state-action pairs could be both mapped and looked up using a table. We'll instead parameterize the state-action-value function using a function approximator: Q_θ.

Specifically, Riedmiller (2005) proposed using the parameterized formulation of the state-action-value function in the ℓ^2 norm minimization expression.

However another alternative is, instead of obtaining the exact solution to this minimization problem, *stochastic gradient descent* based optimization can instead be employed on a loss

function.

This will involve calculating the gradient of the function in the direction that points to the steepest descent. This is achieved by taking the partial derivative of our minimization objective with respect to the parameter θ. We'll update our parameter using the obtained gradient weighted by a step size. The resulting algorithm is the **Q-learning** algorithm (Watkins and Dayan, 1992).

Algorithm 2 Q-learning (Watkins and Dayan, 1992)

Initialize $\theta^{(0)}$
for $n = 0, 1, 2, \ldots$ **do**
  Run policy $\pi^{(n)}$ for T time steps
  $g^{(n)} = \nabla_\theta \sum_t \left(\widehat{\mathcal{T}\mathcal{Q}}_t - Q_\theta \left(s_t, a_t \right) \right)^2$
  $\theta^{(n+1)} = \theta^{(n)} - \alpha g^{(n)}$
end for

2.2 Deep Reinforcement Learning

Deep reinforcement learning combines the use of deep neural networks and reinforcement learning to combat high dimensionality problems where the use of a lookup table to store state-action values becomes infeasible. In particular convolutional neural network based architectures learn to compactly approximate state-action values for high dimensional input data. After first discussing their use in the classical reinforcement learning setting we then study how to approximate a distribution over returns instead of simply approximating the expectation, thus leading to more information available to the agent when making a decision on the next course of action.

2.2.1 Deep Q Networks

There are a few issues that arise when using the Q-learning algorithm due to the function approximation setting. As mentioned previously the algorithm loses the contraction mapping guarantees that were present when it was utilizing a table lookup representation. In fact the Q-learning algorithm may actually diverge in more complex settings/environments failing to optimize our objective.

One of the causes for the failure of Q-learning is due to the *high correlation* of the experience sampled by our agent while its interacting with the environment. Since practically batches of samples are fed to the agent, if there is a high correlation between the samples then the gradient update will not lead to a generalized performance update for the agent and instead will degrade the performance of the agent in parts of the state-space where it will encounter dissimilar tuples of experience. To counter this, a procedure was devised that ensures more stable updates to the state-action-value function.

Another reason for the algorithm's instability is the fact that the agent has a *non-stationary target*. At each gradient update, the state-action value function is also updated. This leads to the target for the state-action value function also to be updated. Thus we have an optimization problem over a moving target which can drastically change the data distribution which is being used to provide feedback to the agent.

Deep Q Networks (DQN) (Mnih et al., 2015) attempted to rectify these underlying issues in the Q-learning algorithm, leveraging deep neural networks as function approximators as well as introducing the following novel strategies:

1. **Experience Replay** - To mitigate issues arising from correlation, a dataset comprising of the agent's prior experiences while interacting with the environment was stored into a *replay buffer*. These previous training examples could be sampled uniformly using mini-batches and used for the gradient update step for the state-action-value function. Thus by incorporating previous experience tuples at each step of the Q-learning update, we're able to de-correlate the sample experience. Since the experiences stored in our replay buffer are known to be optimal for the rest of the state space we ensure that the current update step doesn't cause instability by displacing the correct weights at other parts of the state space whilst also improving the weights locally with the current update.
In addition to reducing the potential of divergence, by utilizing experience replays the agent avoids having to re-experience optimal training examples to ensure convergence to the objective. Thus the addition of the replay buffer leads to an improvement in data efficiency of the algorithm.

2. **Fixed Q-targets** - To ensure the stability of the gradient updates to the state-action-value function, we fix the target weights of the state-action-value function over which the Bellman backup operation is performed for a fixed interval of time-steps. This freezing of the state-action-value functions is used as a target approximation to an optimal state-action-value function. Meanwhile we update the weights of another state-action-value function $Q \Rightarrow \mathcal{T}Q^{(n)}$ that serves as our estimate. Thus we have to maintain two sets of weight vectors: one for our estimate and one for our target, the former being updated immediately while the latter is updated after periodic intervals. Thus the loss to optimize is the mean square difference between the estimate state-action-value function and the target state-action-value function and is used to perform the gradient update to the estimate state-action-value function. After the conclusion of

the fixed interval the target state-action-value function is updated from the accumulated gradient updates from our estimate state-action-value function.

Algorithm 3 Deep Q-learning with experience replay (DQN) (Mnih et al., 2015)

Initialize replay memory D to capacity N.

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

for episode $= 1, M$ **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

for $t = 1, T$ **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action a_t in emulator and observe reward r_t and image x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j + 1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$

end for

end for

2.2.2 Distributional Reinforcement Learning

The methods we have described so far, try to predict the expected cumulative rewards at the end of a trajectory. However another approach that has been suggested involves trying to learn the underlying distribution of returns (Bellemare et al., 2017a). Instead of taking the expectation of the the immediate reward and discounted next state-action-value function as in the regular Bellman equation, Bellemare et al. (2017a) instead approximated the entire distribution over returns to better assess the impact of selecting a particular action as well as discern the variance associated with the corresponding return for that action.

Thus the *distributional* form of the Bellman equation utilizes a random variable $Z(s, a)$ in place of the state-action-value function and a random variable $R(s, a)$ in place of the reward obtained on transitioning to the next state, r and is expressed as:

$$Z^{\pi}(s, a) \stackrel{D}{=} R(s, a) + \gamma Z^{\pi}\left(s', a'\right) \quad s' \sim P^{\pi}(\cdot|s, a) \quad a' \sim \pi(\cdot|s')$$

The value distribution Z provides the agent with an estimate of all the returns that can be obtained by it at every point in the state space. The equivalent distributional Bellman operator that can be applied on a value distribution is the following:

$$\mathcal{T}^{\pi} Z(s, a) \stackrel{D}{:=} R(s, a) + \gamma Z\left(s', a'\right)$$

As we had described previously, in the classical expectational setting the application of the Bellman backup operator provided a γ contraction in the ∞ norm leading to a fixed point. In the distributional setting, the ∞ norm is inapplicable and therefore a probability metric must be relied on to provide similar contraction guarantees for the distributional Bellman operator $\mathcal{T}^{\pi} Z$.

The most common statistical distance that's used to minimize the dissimilarity between two discrete measures is the **KL divergence** (Kullback and Leibler, 1951) which for two discrete measures p and q and points x_i is denoted by:

$$\mathrm{D}_{KL}(p\|q) = \sum p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

Caticha (2004) showed that the KL divergence was the only divergence measure that possessed *locality, coordinate invariance* and *subsystem independence* properties.

However a constraint that has to be satisfied in order to use the KL divergence is that the support of distribution p must be contained in the support of the distribution q. In the case that the distributions p and q have a disjoint support then $\mathrm{D}_{KL}(p\|q) = \infty$. Thus it can't be used as a viable metric over which the behaviour of $\mathcal{T}^{\pi}Z$ can be described due to the contracting effect on the value distribution upon applying the operator.

Ultimately, Bellemare et al. (2017a) used the **Wasserstein metric** which is an integral over the difference between the inverse of the cumulative distribution functions of the measures P, Q. The generalized form of the Wasserstein metric is denoted by:

$$W_p(P,Q) = \left(\int_0^1 \left| F_P^{-1}(x) - F_Q^{-1}(x) \right|^p dx \right)^{\frac{1}{p}}$$

We're primarily concerned with the case where $p = 1$ referred to as **Wasserstein-1** (W_1):

$$W_1(P,Q) = \int_0^1 \left| F_P^{-1}(x) - F_Q^{-1}(x) \right| dx$$

The benefit of using the Wasserstein metric is that the metric is *sum invariant, scale sensitive* and *"Jensen-like"* (Bellemare et al., 2017a). These three properties respectively are:

$$W_p(A+P, A+Q) \leq W_p(P,Q)$$
$$W_p(aP, aQ) \leq |a| W_p(P,Q)$$
$$W_p(AP, AQ) \leq \|A\|_p W_p(P,Q)$$

By defining a *maximal Wasserstein metric* as the largest Wasserstein-1 difference between value distributions Z_1 and Z_2:

$$\sup_{s,a} \bar{W}_p \left(\mathcal{T}^\pi Z_1(s,a), \mathcal{T}^\pi Z_2(s,a)\right) \leq \gamma \sup_{s,a} \bar{W}_p \left(Z_1(s,a), Z_2(s,a)\right)$$

and utilizing these three properties of the Wasserstein metric, Bellemare et al. (2017a) showed that for a given policy π, the distributional Bellman operator, $\mathcal{T}^\pi Z$ was a γ contraction in the Wasserstein-1 metric.

While this result was true for the policy evaluation problem, it did not hold for the policy optimization problem. Thus using the Wasserstein-1 distance as a probability metric would not guarantee that the optimal value distribution Z^* would be obtained.

The novel algorithm that Bellemare et al. (2017a) proposed was **C51**. The C51 algorithm uses a softmax function to generates a discrete probability measure with 51 possible outputs over the parameterized value distribution, Z_θ. At each update step Z_θ is updated towards a target value distribution using the distributional Bellman backup operator. However since Z and R are random variables to be sampled from, Z_θ is actually being optimized towards *samples* of the target value distribution (from experience replay) and not the actual target distribution. The key steps involved in the algorithm are as follows:

1. The agent samples a new experience tuple at a given state-action pair by taking a new action a' and transitioning to a new state s' and obtaining an immediate reward r:
$r, s', a' \sim R(x,a), P(\cdot \mid s,a), \pi\left(\cdot \mid s'\right)$

2. The distributional Bellman backup operator is applied on this sample, to obtain a sample backup of the value distribution: $\hat{\mathcal{T}}^\pi Z_{\theta^-}(s,a) := r + \gamma Z_\theta\left(s', a'\right)$

3. Setting this sample backup as the fixed target value distribution, the parameterized estimate of the value distribution would ideally be optimized using gradient descent on the Wasserstein-1 loss function: $l_\theta(s,a) := W_1\left(\hat{\mathcal{T}} Z_{\theta^-}(s,a), Z_\theta(s,a)\right)$.

However, in practice using the Wasserstein distance leads to *biased estimates* of the sample gradient which fail to converge to the minima (Bellemare et al., 2017b). In particular, Bellemare et al. (2017b) concluded that "minimizing the sample Wasserstein loss by stochastic gradient descent may in general fail to converge to the minimum of the true (Wasserstein) loss".

Since using the Wasserstein-1 loss was not feasible, the C51 algorithm instead minimizes the KL divergence which despite not being scale sensitive, generates *unbiased estimates* of the sample gradient (Bellemare et al., 2017b). To use the KL divergence, the C51 algorithm ensures that the supports of the target value distribution and the estimate value distribution are not disjoint.

To facilitate this, the target value distribution is *projected* to the space of the estimate value distribution. Thus, $\Phi\mathcal{T}^{\pi}Z_{\theta^-}$ will have the same support as Z_{θ}. These steps are illustrated in Figure 2.1.

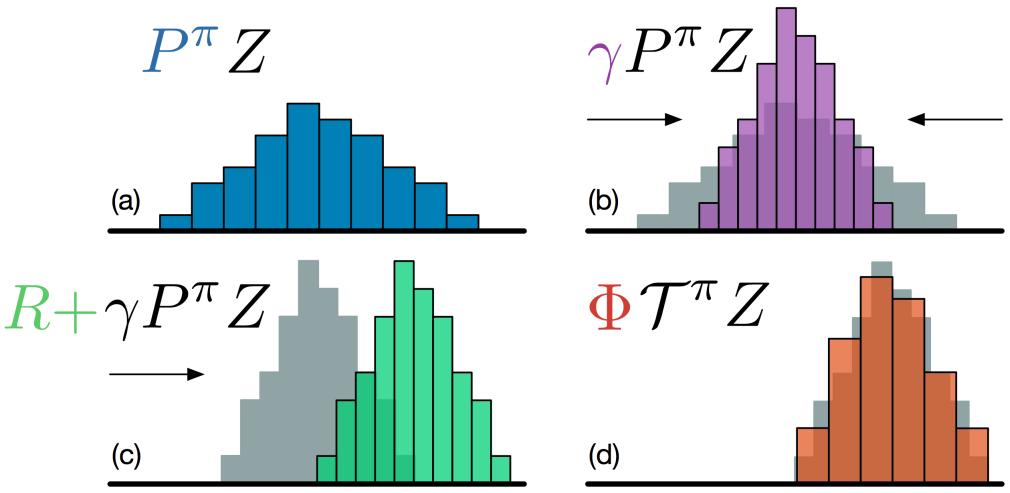

Figure 2.1: Steps involved in projecting the target value distribution to the space of the estimate value distribution (Source: Bellemare et al. (2017a))

4. The final step involves minimization of the KL divergence bringing the estimate value distribution, Z_{θ} closer to the projected sample backup (target) value distribution, $\Phi\mathcal{T}^{\pi}Z_{\theta^-}$.

Algorithm 4 Categorical Algorithm (C51) (Bellemare et al., 2017a)

input A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$

$Q\left(x_{t+1}, a\right) := \sum_i z_i p_i\left(x_{t+1}, a\right)$

$a^* \leftarrow \arg\max_a Q\left(x_{t+1}, a\right)$

$m_i = 0, \quad i \in 0, \ldots, N-1$

for $j \in 0, \ldots, N-1$ **do**

$\#$ Compute the projection of $\mathcal{T} z_j$ onto the support $\{z_i\}$

$\hat{\mathcal{T}} z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$

$b_j \leftarrow \left(\hat{\mathcal{T}} z_j - V_{\text{MIN}}\right) / \Delta z \quad \# b_j \in [0, N-1]$

$l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$

$\#$ Distribute probability of $\hat{\mathcal{T}} z_j$

$m_l \leftarrow m_l + p_j\left(x_{t+1}, a^*\right)(u - b_j)$

$m_u \leftarrow m_u + p_j\left(x_{t+1}, a^*\right)(b_j - l)$

end for

output $-\sum_i m_i \log p_i\left(x_t, a_t\right)$ $\#$ Cross-entropy loss

Chapter 3

Theoretical Analysis

In this section we'll first discuss a general family of probability metrics called Integral Probability Metrics (IPM), the Wasserstein distance being one such member. We then discuss the theoretical limitations on using the Wasserstein distance and study innovations from the field of computational optimal transport (Peyré et al., 2019) in mitigating its prohibitive computational complexity and sample complexity. Our discussion on computational optimal transport is based on Cuturi and Solomon (2017).

We then study a more general family of divergence measures called f divergences which have seen successful applications in deep generative modelling. Our discussion on f divergences is based on Nowozin et al. (2016). Of the possible instances of f divergences we restrict our focus to a special subset which are also members of the class of α divergences. By studying the properties of α divergences we can better interpret the results obtained in the subsequent section. Our discussion on α divergences is based on Cevher et al. (2008).

A thing to note is that Integral Probability Metrics cannot be formulated as f divergences and vice-versa. However the **total variation distance** is the only statistical distance that is both a member of the class of f divergences and the class of Integral Probability Metrics (Sriperumbudur et al., 2012). It is expressed as the difference of the two discrete measures at all possible points of dissimilarity, x_i:

$$D_{\mathrm{TVD}}(\mu, \nu) = \sum_{i=1}^{N} |\mu(x_i) - \nu(x_i)|$$

Using the total variation distance to approximate value distributions had been studied rigorously by Morimura et al. (2010a,b). Bellemare et al. (2017a) also studied its potential use and ultimately concluded that on applying operations that can cause contractions (such as the discount factor) the absolute difference between the estimate value distribution and the projected target value distribution will remain unchanged. Therefore it was not considered as a viable distance metric for the C51 categorical loss function.

3.1 Integral Probability Metrics

Integral Probability Metrics (IPM) are a class of methods used to minimize the statistical distance between two discrete measures. The distance metric generated from IPM involve taking the supremum of the difference in expectations when sampling from two discrete measures over a suitable set of functions (Sriperumbudur et al., 2010) (Müller, 1997).
The general form of an IPM between two measures μ and ν for the class of *witness functions*, $\mathcal{F}$ is the following:

$$\mathcal{D}(\mu, \nu; \mathcal{F}) = \sup_{f \in \mathcal{F}} [\mathbb{E}_\mu f(x) - \mathbb{E}_\nu f(y)]$$

The choice of the witness function determines which particular IPM is being used to minimize the distance between the measures. In this section, from an optimal transport perspective, we'll analyze the relationship between three IPM: the Wasserstein distance, the entropy regularized Wasserstein distance and the Energy distance (or MMD).

3.1.1 Computational Optimal Transport

The field of *computational optimal transport* deals with the geometry underpinning both discrete and continuous measures. Since the field of machine learning often utilizes discrete measures, optimal transport has found many useful applications in different sub-fields of machine learning. Primarily its often used when a task at hand involves comparing, contrasting or minimizing the statistical distance between two discrete measures.

The original problem that initiated this field was **Monge's Problem** (Monge, 1781). The problem abstractly dealt with obtaining the optimal mapping T from a source measure μ to a

target measure ν across a distance D such that the work done in moving probability mass at all points in the source measure to all points in the target measure is minimized.

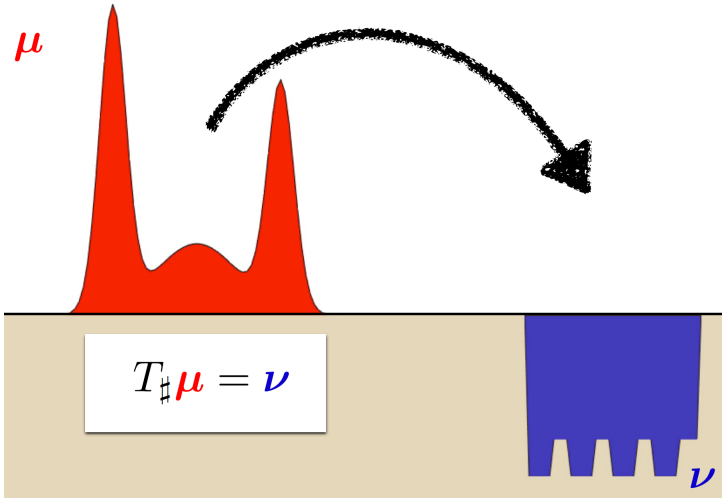

Figure 3.1: Monge's Problem (Source: Cuturi and Solomon (2017))

Let the operator $\sharp$ denotes the fact that the mapping T *pushes forward* the discrete measure μ onto the discrete measure ν.

Then Monge's problem can be formulated as determining the optimal mapping, $T_\sharp \mu = \nu$ that minimizes the product of the mass transported from the source distribution and the cost c of transporting the probability mass for all points x in the source measure and the target measure on the geometric space Ω:

$$\inf_{T_\sharp \mu = \nu} \int_\Omega c(x, T(x)) \mu(dx)$$

Unfortunately due to the fact that the problem setting involved an integral over non-convex constraints, significant breakthroughs in this problem stalled and instead further development in the field shifted to a *relaxed* formulation of the problem known as the **Kantorovich problem** (Kantorovich, 1960). In particular the issue that arose was that the Monge formulation did not permit splitting up of a Dirac measure from the source into a multiplicity of Dirac measures in the target. In contrast, the Kantorovich relaxation of the problem permitted the splitting up of a Dirac measure from the source measure into separate Dirac measures in the target measure. This implies that unlike in the Monge problem where we use a deterministic mapping

between the source and destination measure, in the Kantorovich problem we utilize *measure coupling* which associates measures using a joint probability distribution across the product of the geometric space: $P \in \mathcal{P}(\Omega \times \Omega)$

Therefore for source measure μ and target measure ν we consider all measure couplings (joint probability distributions) such that the marginal distributions of the joint distribution will result in μ and ν. We'll denote this as:

$$\begin{aligned}\Pi(\mu, \nu) \stackrel{\text{def}}{=} \{P \in \mathcal{P}(\Omega \times \Omega) \mid \forall A, B \subset \Omega, \\ P(A \times \Omega) = \mu(A) \\ P(\Omega \times B) = \nu(B)\}\end{aligned}$$

The Kantorovich problem involves optimizing over the space of possible measure couplings, $\Pi(\mu, \nu)$ between source measure μ and target measure ν that minimizes the cost $c(x, y)$ of transporting probability mass $P(dx, dy)$ from point x in the source measure to point y in the target measure:

$$\inf_{P \in \Pi(\mu,\nu)} \iint c(x, y) P(dx, dy)$$

This formulation is known as the *primal* form of the Kantorovich problem. For the given Kantorovich formulation if we set the cost function as $c(x, y) = D^p(x, y)$ then we obtain a family of p distance metrics where $p \geq 1$. Specifically the p**-Wasserstein distance** minimizing over the space of possible measure couplings $\Pi(\mu, \nu)$ between source measure μ and target measure ν is expressed as:

$$W_p(\mu, \nu) \stackrel{\text{def}}{=} \left(\inf_{P \in \Pi(\mu,\nu)} \iint D(x, y)^p P(dx, dy) \right)^{1/p}$$

The geometry induced by applying the Wasserstein distance between two measures differs from that obtained by applying divergence measures such as the KL divergence. McCann (1997) showed that the Wasserstein distance provided a set of measure couplings on the shortest path between the two measures referred to as the *displacement interpolation*.

Figure 3.2 demonstrates how using a euclidean distance as an interpolant between two Gaussians would simply obtain the mean of the two Gaussians leading to a point of discontinuity between the two whereas using the Wasserstein distance, the interpolant demonstrates the different set of measure couplings lying on the shortest path between the two Gaussians providing a more intuitive interpretation.

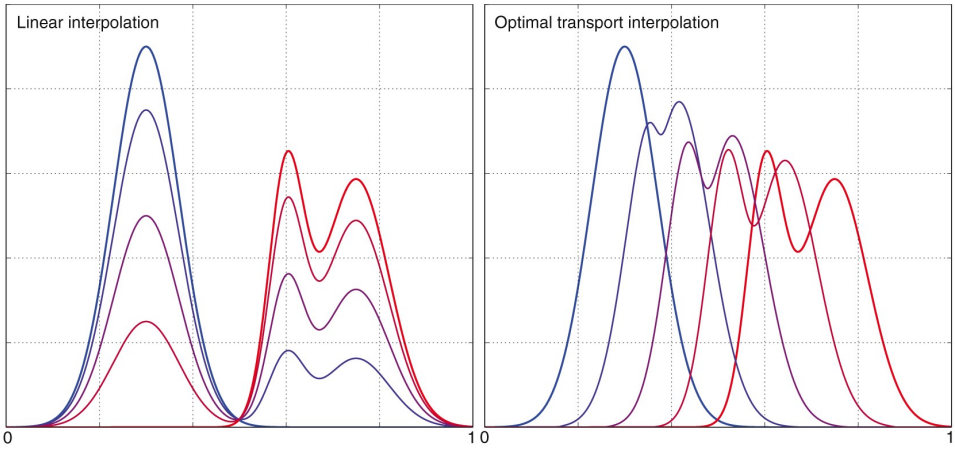

Figure 3.2: Displacement interpolation induced by optimal transport as compared to linear interpolation (Source: Cuturi and Solomon (2017))

However an issue that we can observe is that computing the exact solution to the Wasserstein distance will require linear programming, since the factor p is present on the cost term and not on the optimal measure coupling representing the probability mass being transported from point x to point y.

We'll next analyze the computational complexity for obtaining the exact solution to the optimal transport problem for two discrete empirical measures.

Given two discrete empirical measures μ and ν on the geometric space Ω consisting of n and m points respectively. Let δ_{x_i} denote the Dirac measure at point x_i, having corresponding length a_i. Similarly, let δ_{y_j} denotes the Dirac measure at point y_j, having corresponding length b_j. Figure 3.3 depicts the respective measures.

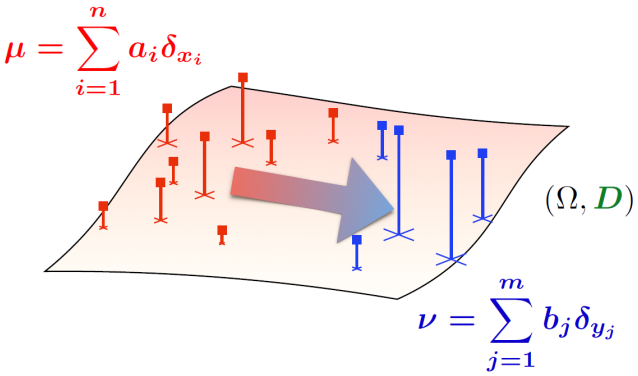

Figure 3.3: Optimal transport problem between two discrete measure (Source: Cuturi and Solomon (2017))

To obtain the optimal mapping from μ to ν we first define the cost function as the pairwise p-Wasserstein distance between all points in the source measure and the target measure:

$$M_{XY} \overset{\text{def}}{=} \left[D\left(x_i, y_j\right)^p\right]_{ij}$$

Next we define the space of measure couplings for consideration in our optimal mapping. If we consider these measure couplings as matrices, we desire only those matrices which have the appropriate marginal distributions, ensuring that all probability mass from μ is mapped to ν. Therefore if we compute the product of matrix with an identity matrix we obtain a and if we compute the product of the transpose of the matrix with the identity matrix we obtain b:

$$U(a, b) \overset{\text{def}}{=} \left\{P \in \mathbb{R}_+^{n \times m} | P\mathbf{1}_m = a, P^T\mathbf{1}_n = b\right\}$$

Finally, since the objective is to obtain that measure coupling which is the infimum of the product of the cost function and the probability mass distribution, the formulation of the p-Wasserstein involves computing their Frobenius inner product:

$$W_p^p(\mu, \nu) = \min_{P \in U(a,b)} \langle P, M_{XY} \rangle$$

Thus the p-Wasserstein distance indeed requires linear programming to obtain an exact solution. This results in a prohibitive computational complexity: $O((n+m)nm\log(n+m))$.

Thus not only is using the p-Wasserstein distance impractical for high dimensionality problems, but the resultant loss function is also *non-differentiable* thus making it unsuitable for gradient based optimization.

In addition to the computational complexity we're also interested in the *sample complexity* since in practical machine learning settings, i.i.d samples drawn from the source and target measures: $\hat{\mu}_n, \hat{\nu}_m$ serve as proxies for the true underlying measures μ and ν. The sample complexity quantifies whether the p-Wasserstein distance between the samples of the measures, $W_p\left(\hat{\mu}_n, \hat{\nu}_m\right)$ is truly representative of the actual underlying Wasserstein distance $W_p(\mu, \nu)$ that we're trying to approximate.

Unfortunately, Dudley (1969) showed that for dimensionality $d > 3$, the sample complexity of using the p-Wasserstein distance is also prohibitive: $O\left(1/n^{1/d}\right)$. This implies that an exponential number of samples are required to obtain a suitable mapping for high dimensionality problems. Thus not only is the exact solution computationally prohibitively, the resultant sample approximation is a poor representation of the underlying "true" distance metric between the two measures.

Thus to make an optimal transport inspired distance metric feasible for high dimensionality machine learning problem settings, *regularization* needs to be employed to not only optimize both the computational and sample complexity but also to engineer a loss function that is both stable and differentiable, with the later property enabling gradient based optimization.

Wilson (1969) suggested **entropic regularization** as a possible solution wherein a regularization term, $E(P)$ was added to the primal form of the p-Wasserstein distance parameterized by a *blurring coefficient* $\gamma \geq 0$:

$$W_\gamma(\mu, \nu) \stackrel{\text{def}}{=} \min_{P \in U(a,b)} \langle P, M_{XY} \rangle - \gamma E(P)$$

where the entropy term is the following strongly concave function:

$$E(P) \stackrel{\text{def}}{=} -\sum_{i,j=1}^{nm} P_{ij} \left(\log P_{ij} - 1\right)$$

The resultant **entropy regularized Wasserstein distance** adds a degree of blurring to the optimal solution sacrificing the accuracy of reaching the optimal measure coupling, $P^\star$ as a trade-off for reducing the computational and sample complexity. This results in a more scalable and differentiable loss function.

Figure 3.4 illustrates the effect of varying the blurring parameter γ on the optimal transport mapping. The left most figure corresponds to the exact optimal transport solution between the two measures. On moving to the right we increase γ, and so the mapping becomes less sharp and more diffused.

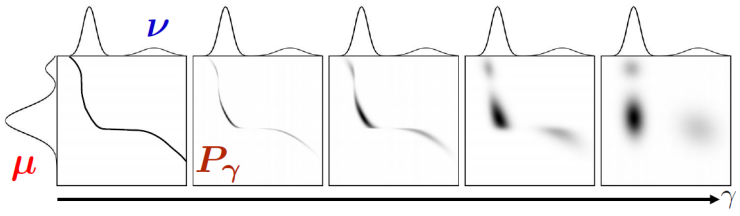

Figure 3.4: Increasing the γ parameter leads to a more regularized mapping between measures μ and ν (Source: Cuturi and Solomon (2017))

Cuturi (2013) devised a scalable technique utilizing GPUs to efficiently compute the optimal solution to the regularized Wasserstein distance called **Sinkhorn's algorithm**. Cuturi (2013) showed that for the following entropy regularized Wasserstein formulation:

$$P_\gamma \stackrel{\text{def}}{=} \underset{P \in U(a,b)}{\operatorname{argmin}} \langle P, M_{XY} \rangle - \gamma E(P)$$

the optimal solution, P_γ is a product of the following matrices: $P_\gamma = \operatorname{diag}(u)\mathbf{K}\operatorname{diag}(v)$

Wherein, vectors, $u \in \mathbb{R}_+^n, v \in \mathbb{R}_+^m$ are variables to be approximated via repetitive operations. The *kernel matrix*, $\mathbf{K}$ is the exponential of the negative pairwise p-Wasserstein distances between all points in the source measure and the target measure divided by the blurring parameter: $\mathbf{K} \stackrel{\text{def}}{=} e^{-M_{XY}/\gamma}$

The vectors u and v are determined using the constraints on the probability mass that can be transferred between the source and target measure. Specifically, the space of feasible measure couplings, must fulfill the marginal constraints for the source measure and target measures:

$$P_\gamma \in U(a,b) \Leftrightarrow \begin{cases} \operatorname{diag}(u)\mathbf{K}\operatorname{diag}(v)\mathbf{1}_m & = a \\ \operatorname{diag}(v)\mathbf{K}^T\operatorname{diag}(u)\mathbf{1}_n & = b \end{cases}$$

Since the product of the diagonal of a vector with the identity matrix results in the original matrix we can use the following substitutions: $v = \operatorname{diag}(v)\mathbf{1}_m$ and $u = \operatorname{diag}(u)\mathbf{1}_n$. In addition the product of the diagonal of a vector with another matrix results in the Hadamard product, $\odot$ between the two. Thus the diagonalization of the former vector in the above formulation is removed. Using these two properties we can re-express the marginal constraints to be satisfied by the optimal measure coupling of the entropy regularized Wasserstein distance:

$$P_\gamma \in U(a,b) \Leftrightarrow \begin{cases} u \odot \mathbf{K}v & = a \\ v \odot \mathbf{K}^T u & = b \end{cases}$$

By moving variables between the LHS and the RHS, we can thus determine the form that the variables u and v must take in order to satisfy the constraints of the optimal measure coupling:

$$P_\gamma \in U(a,b) \Leftrightarrow \begin{cases} u = a/\mathbf{K}v \\ v = b/\mathbf{K}^T u \end{cases}$$

Sinkhorn (1964) proved that the algorithm converges to an optimal solution. The implementation by Cuturi (2013) leveraging GPUs for hardware acceleration thus resulted in a computationally tractable method to apply optimal transport based distance metrics for high dimensionality problems.

Ramdas et al. (2017) and Genevay et al. (2018) showed that there are two edge cases of the entropy regularized optimal transport problem: the *minimal entropy* form and the *maximal entropy* form.

The minimal entropy form corresponds to the exact Wasserstein distance, for which the optimal fixed point solution requires a network flow solver with high computational complexity. By applying some degree of entropic regularization we'll obtain the entropy regularized Wasserstein distance which has a corresponding set of sub-optimal solutions P_γ. If we set $\gamma \to \infty$ this results in the maximal entropy form, where the resultant measure coupling is simply the product of the marginal distributions of the source measure and target measures.

However an issue that is endemic with the entropy regularized Wasserstein distance and the maximal entropy form of the optimal transport problem is that the distances do not equal zero for the case where the target measure and source measure are identical. Ramdas et al. (2017) proposed a solution to **de-bias** both the entropy regularized Wasserstein distance and the maximal entropy formulation by subtracting from both solutions the distances when the source measure is contrasted to itself and the target measure is contrasted to itself.

Thus we can express the sub-optimal solution of the *biased* entropy regularized Wasserstein distance using the following formulation:

$$W_\gamma(\mu, \nu) = \langle P_\gamma, M_{XY} \rangle$$

The sub-optimal solution of the *de-biased* entropy regularized Wasserstein distance, referred to as the **Sinkhorn divergence** can be obtained by normalizing the biased formulation:

$$\bar{W}_\gamma(\mu, \nu) = W_\gamma(\mu, \nu) - \frac{1}{2}\left(W_\gamma(\mu, \mu) + W_\gamma(\nu, \nu)\right)$$

For the minimal entropy form of the regularized optimal transport problem, the blurring parameter is set to: $\gamma \to 0$ to obtain the following formulation of the exact optimal solution to the Wasserstein distance:

$$W^p(\mu, \nu) = \langle P^\star, M_{XY} \rangle$$

For the *biased* maximal entropy form of the regularized optimal transport problem, the blurring parameter is set to: $\gamma \to \infty$ to obtain the following formulation:

$$\mathcal{E}(\mu, \nu) = \left\langle ab^T, M_{XY} \right\rangle$$

Similarly, we can obtain the *de-biased* maximal entropy form of the regularized optimal transport problem, referred to as the **Energy distance** or **MMD** (Gretton et al., 2012) by normalizing the biased formulation:

$$\mathcal{MMD}(\mu, \nu) = \mathcal{E}(\mu, \nu) - \frac{1}{2}(\mathcal{E}(\mu, \mu) + \mathcal{E}(\nu, \nu))$$

To conclude, a comparison can be drawn between the sample complexity and computational complexity of each of these algorithms fulfilling the original motivation for this analysis. In comparison to the previously prohibitive, $O((n+m)nm\log(n+m)$ computational complexity and $O\left(1/n^{1/d}\right)$ sample complexity of the exact Wasserstein distance, the MMD has a computational complexity of $O(1/\sqrt{n})$ and a sample complexity of $(n+m)^2$ (Gretton et al., 2012) while the Sinkhorn Divergence has a computational complexity of $O\left((n+m)^2\right)$ and a sample complexity of $O\left(\frac{1}{\gamma^{d/2}\sqrt{n}}\right)$ (Feydy et al., 2019) (Genevay et al., 2019).

3.2 f Divergences

f divergences are the second class of methods used to quantify the degree of dissimilarity between two discrete measures. Csiszár (1967) introduced the general form of the f divergence measure. Let μ and ν be two discrete measures on the measurable space Ω. For any convex function $f : (0, \infty) \to \mathbb{R}$ with $f(1) = 0$, and given two measures ν and μ where $\mu \ll \nu$ and the respective probability mass functions are denoted as $\nu(x)$ and $\mu(x)$, then the general form of the f divergence of ν from μ can be expressed as:

$$\mathcal{D}_f(\mu \| \nu) = \sum_{x \in \Omega} \nu(x) f\left(\frac{\mu(x)}{\nu(x)}\right)$$

Csiszár and Körner (1981) showed that the f divergences satisfied both *joint convexity* and *Jensen's inequality*. Other favorable properties of f divergences include being *non-negative*, *scale sensitive*, and *invariant* (Cichocki and Amari, 2010). Each member of the class of f divergences quantifies the aforementioned dissimilarity between the measures using a different criterion.

The following select set of divergence measures are instances of f divergences across all possible points of dissimilarity, x_i between the two measures and for a particular choice of the convex function f:

On choosing $f(x) = x \log x$ we obtain the **KL divergence**:

$$D_{\mathrm{KL}}(\mu \| \nu) = \sum_{i=1}^{N} \mu(x_i) \log\left(\frac{\mu(x_i)}{\nu(x_i)}\right)$$

On choosing $f(x) = -\log x$ we obtain the **reverse KL divergence**:

$$D_{\text{Reverse KL}}(\mu \| \nu) = \sum_{i=1}^{N} \nu(x_i) \log\left(\frac{\nu(x_i)}{\mu(x_i)}\right)$$

On choosing $f(x) = 2(\sqrt{x} - 1)^2$ we obtain the **squared Hellinger distance**:

$$D_{\mathrm{H}^2}(\mu\|\nu) = 2\sum_{i=1}^{N}\left(\sqrt{\mu(x_i)} - \sqrt{\nu(x_i)}\right)^2$$

On choosing $f(x) = \frac{1}{2}(x-1)^2$ we obtain the **Pearson χ^2 divergence**:

$$D_{\text{Pearson }\chi^2}(\mu\|\nu) = \frac{1}{2}\sum_{i=1}^{N}\frac{\left(\nu(x_i) - \mu(x_i)\right)^2}{\mu(x_i)}$$

On choosing $f(x) = \frac{(1-x)^2}{2x}$ we obtain the **Neyman χ^2 divergence**:

$$D_{\text{Neyman }\chi^2}(\mu\|\nu) = \frac{1}{2}\sum_{i=1}^{N}\frac{\left(\mu(x_i) - \nu(x_i)\right)^2}{\nu(x_i)}$$

On choosing $f(x) = x\log\frac{2x}{x+1} + \log\frac{2}{x+1}$ we obtain the **Jensen-Shannon divergence**:

$$D_{\text{JSD}}(\mu\|\nu) = D_{\text{KL}}\left(\mu(x_i)\|\frac{\mu(x_i)+\nu(x_i)}{2}\right) + D_{\text{KL}}\left(\nu(x_i)\|\frac{\mu(x_i)+\nu(x_i)}{2}\right)$$

As mentioned previously, the potential convex function, f used to formulate the f divergence should satisfy $f(1) = 0$. This is the desired theoretical target where the target measure perfectly matches the source measure implying that there is no dissimilarity between the two measures.

3.2.1 α Divergences

The class of α divergences are themselves a set of divergence measures of the parent family of f divergences and hence enjoy the same properties as their parent divergence measure. Chernoff et al. (1952) proposed the α divergence while Minka et al. (2005) provided the general formulation of the α divergence for $\alpha \in [-\infty, +\infty]$ which adapted for discrete measures μ and ν is as follows:

$$D_\alpha(\mu\|\nu) = \frac{\sum_i^N \alpha\mu(x_i) + (1-\alpha)\nu(x_i) - [\mu(x_i)]^\alpha[\nu(x_i)]^{1-\alpha}}{\alpha(1-\alpha)}$$

For values of α in the range $\alpha \in [-1, 2]$, the following special cases arise (Minka et al., 2005):

$$D_{\alpha=-1}(\mu\|\nu) = \frac{1}{2}\sum_{i=1}^{N}\frac{(\nu(x_i)-\mu(x_i))^2}{\mu(x_i)} = D_{\text{Pearson }\chi^2}(\mu,\nu)$$

$$\lim_{\alpha\to 0} D_\alpha(\mu\|\nu) = \sum_{i=1}^{N}\nu(x_i)\log\left(\frac{\nu(x_i)}{\mu(x_i)}\right) = D_{\text{Reverse KL}}(\mu,\nu)$$

$$D_{\alpha=\frac{1}{2}}(\mu\|\nu) = 2\sum_{i=1}^{N}\left(\sqrt{\mu(x_i)}-\sqrt{\nu(x_i)}\right)^2 = D_{\text{H}^2}(\mu,\nu)$$

$$\lim_{\alpha\to 1} D_\alpha(\mu\|\nu) = \sum_{i=1}^{N}\mu(x_i)\log\left(\frac{\mu(x_i)}{\nu(x_i)}\right) = D_{\text{KL}}(\mu,\nu)$$

$$D_{\alpha=2}(\mu\|\nu) = \frac{1}{2}\sum_{i=1}^{N}\frac{(\mu(x_i)-\nu(x_i))^2}{\nu(x_i)} = D_{\text{Neyman }\chi^2}(\mu,\nu)$$

Minka et al. (2005) analyzed the applications of α divergences in Bayesian inference and provided the following conclusions:

For negative values of α, the target measure ν concentrates on a single mode in the source measure μ which covers the largest probability mass. In particular for $\alpha \leq 0$ if the source measure μ assigns zero probability mass at any point x_i then the target measure ν will also force probability mass concentrated at the point x_i to be zero. Thus the divergence measure prioritises minimizing dissimilarity at the single largest mode in the source measure and disregards the rest of the points in the measure. This property of select α divergence measures is known as **zero-forcing** and is exhibited by $D_{\text{Reverse KL}}(\mu\|\nu)$ and $D_{\text{Pearson }\chi^2}(\mu\|\nu)$.

On the other hand for positive values of α, the target measure ν tries to distribute probability mass evenly across all points where the source measure μ has assigned some probability mass. In particular for $\alpha \geq 1$, the target measure distributes probability mass at all points of

non-zero probability in the source measure, not singularly prioritizing a specific mode in the source measure. Thus the bulk of the points in the source measure with non-zero probability will also be approximated in the target measure even if there is little valuable information at those particular point. This property of select α divergence measures is known as **zero-avoiding** and is exhibited by $D_{\mathrm{KL}}(\mu\|\nu)$ and $D_{\text{Neyman }\chi^2}(\mu\|\nu)$.

Finally for values of α in the range $0 < \alpha < 1$, the target measure ν will not concentrate simply on the largest mode in the source measure μ but will also assign probability mass to other modes proximal to the largest mode in the source measure. However all points x_i that are distant to the largest mode will not be assigned any probability mass (even if they contain a smaller mode). This property of select α divergence measures is an *amalgamation* of zero-forcing and zero-avoiding and is exhibited by $D_{\mathrm{H}^2}(\mu\|\nu)$.

Figure 3.5 demonstrates the effect of varying α on the probability mass assignment by the target measure. On increasing the value of α, the target measure curtails the zero-forcing property and amplifies the zero-avoiding property.

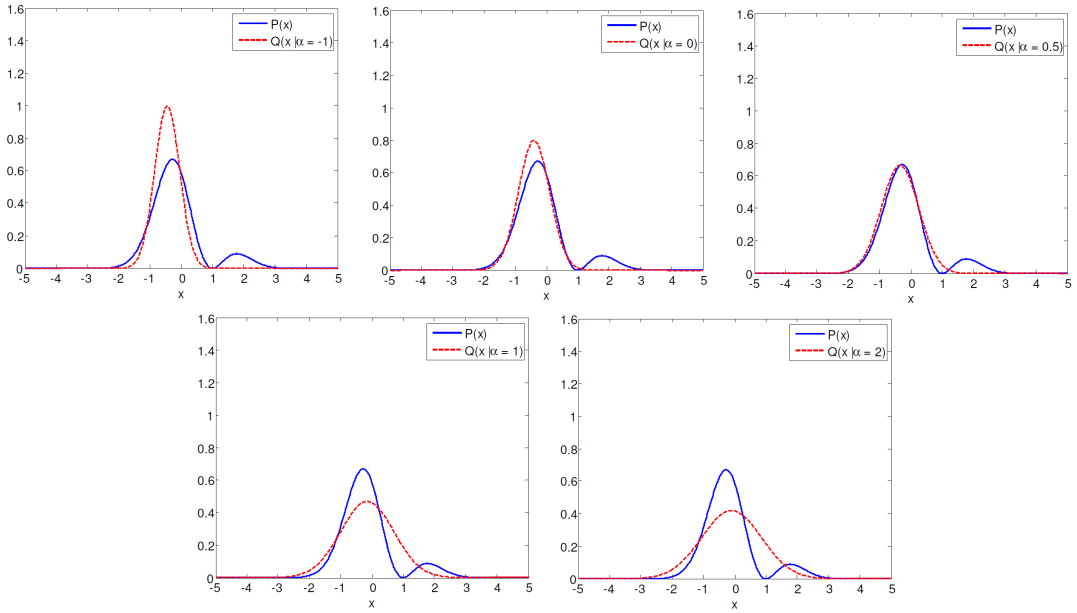

Figure 3.5: Resultant target measure, Q on applying the Pearson χ^2 divergence ($\alpha = -1$), Reverse KL divergence ($\alpha = 0$), Squared Hellinger distance ($\alpha = 0.5$), KL divergence ($\alpha = 1$) and Neyman χ^2 divergence ($\alpha = 2$) to approximate the source measure, P (Adapted from: Cevher et al. (2008))

Chapter 4

Experimental Results

After having gained insights into the theoretical characteristics of the probability metrics and divergence measures we now proceed to determine empirically, the efficacy of using them in place of the KL divergence in minimizing the statistical distance between the projected target value distribution and the estimate value distribution in the C51 categorical loss function. We modified the implementations by Park et al. (2019) to benchmark the performance of the DQN (Mnih et al., 2015) algorithm and contrast the performance of the Sinkhorn divergence, the Energy distance (or MMD), the reverse KL divergence, Pearson χ^2 divergence, squared Hellinger distance, Neyman χ^2 divergence and Jensen-Shannon divergence as feasible categorical loss functions for the C51 algorithm.

To facilitate efficient computation of the Energy distance and Sinkhorn divergence we utilized the GeomLoss library (Feydy et al., 2019). For the Sinkhorn divergence we use a blurring coefficient, corresponding to $\gamma = 0.01$, as suggested by Cuturi (2013) for applied machine learning problems. Each algorithm utilizes similar hyperparameter settings. Specifically, the replay buffer size is 1,000 with the target value distribution being updated after 200 steps and the agent sampling a mini-batch of 32 transitions from the replay buffer.
After training each algorithm variant for 10,000,000 frames we test the trained agent for 1000 episodes.

We utilize the following two classic control environments implemented in the OpenAI Gym (Brockman et al., 2016) as test beds for gauging the training time and testing time performance:

Cartpole (Barto et al., 1983) - The goal of the agent while navigating the CartPole environment is to keep a pole balanced whilst moving the cart on which the pole is place in the east or west direction.

Figure 4.1 depicts the average training time performance of each categorical loss function as well as the baseline DQN and C51 algorithm performance. Smoothing utilizing exponential moving average was applied on the training curves.

Figure 4.2 depicts the best test time performance of each categorical loss function as well as the baseline DQN and C51 algorithm performance.

Acrobot (Sutton, 1996) - The goal of the agent while navigating the Acrobot environment is to swing a pendulum with double links and two actuated joints up to a particular threshold by applying rotational force on one of its links.

Figure 4.3 depicts the average training time performance of each loss function as well as the baseline DQN and C51 algorithm performance. Smoothing utilizing exponential moving average was applied on the training curves.

Figure 4.4 depicts the best test time performance of each categorical loss function as well as the baseline DQN and C51 algorithm performance.

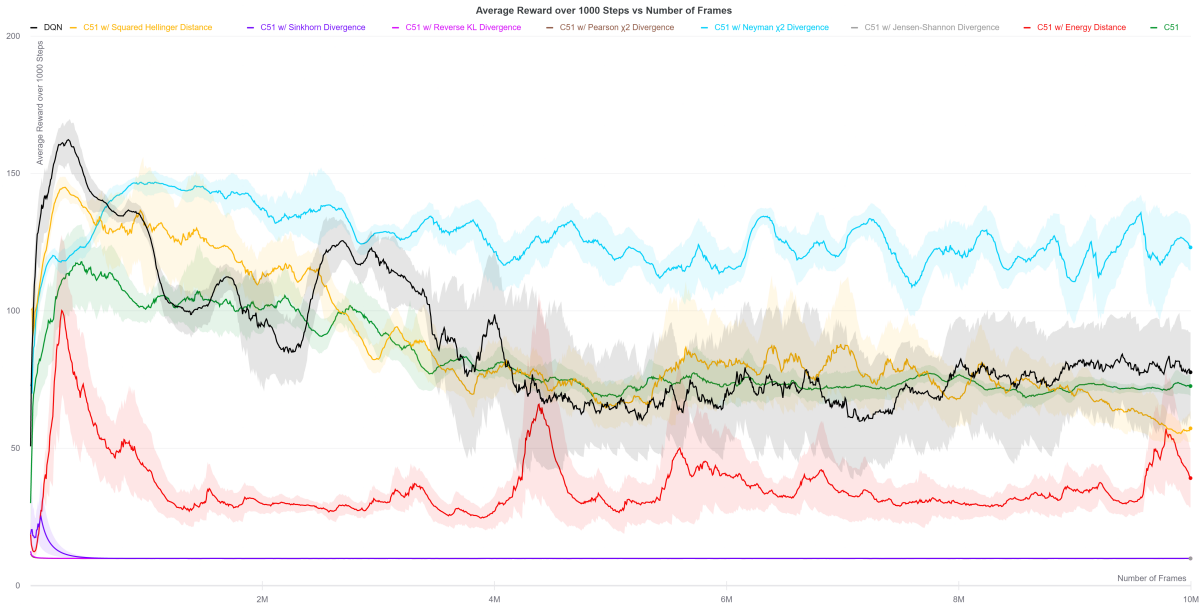

Figure 4.1: Comparison of the average training time performance on Cartpole

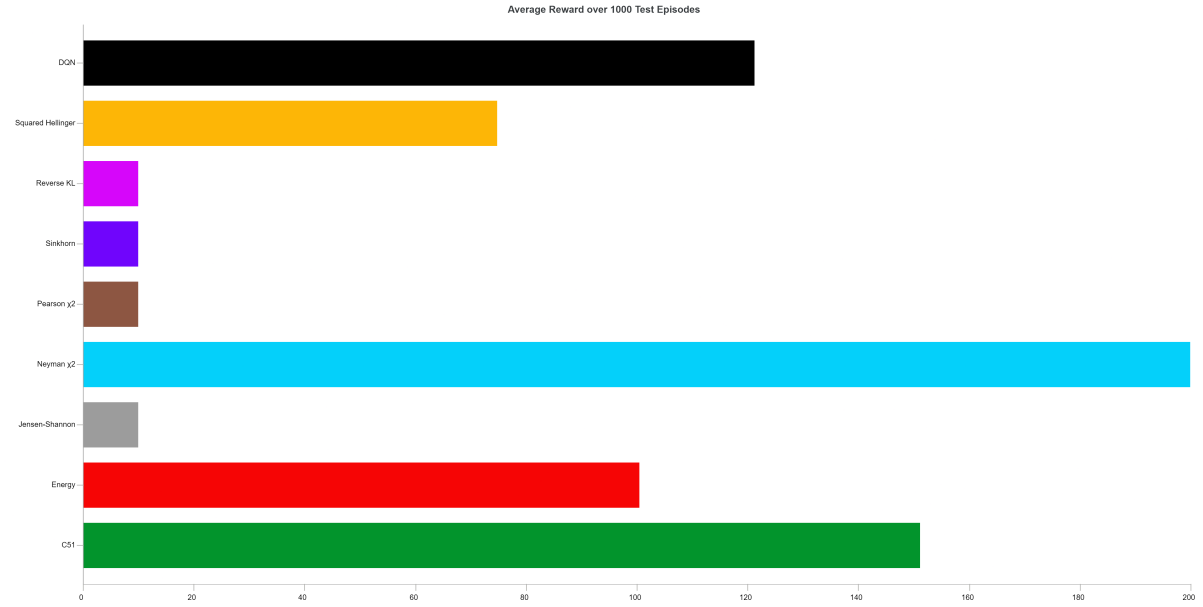

Figure 4.2: Comparison of the best test time performance on Cartpole

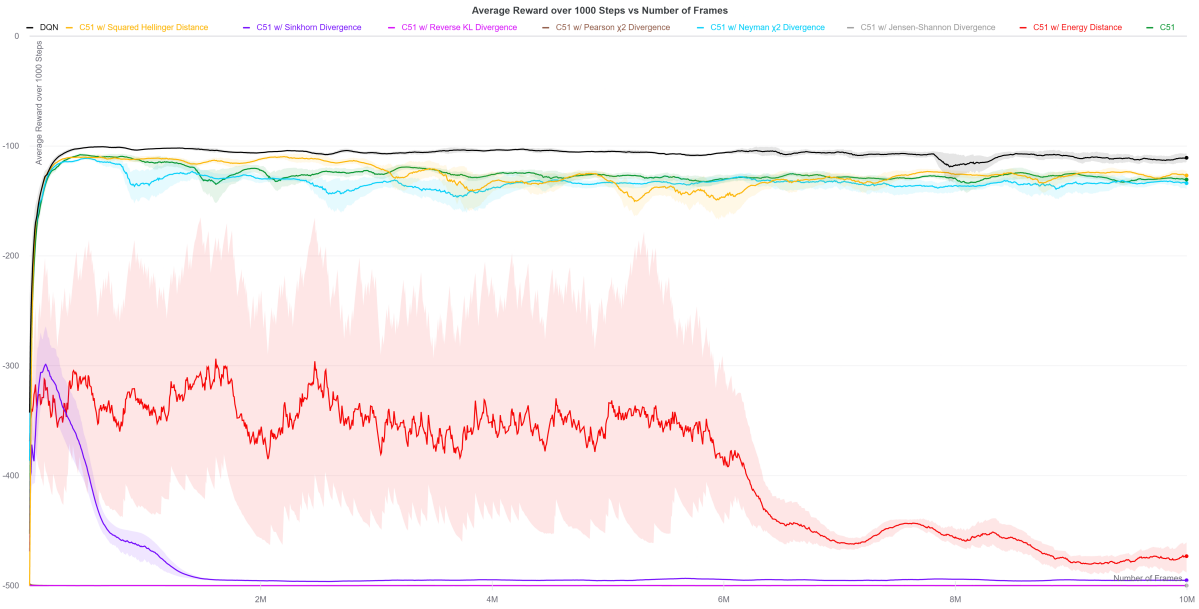

Figure 4.3: Comparison of the average training time performance on Acrobot

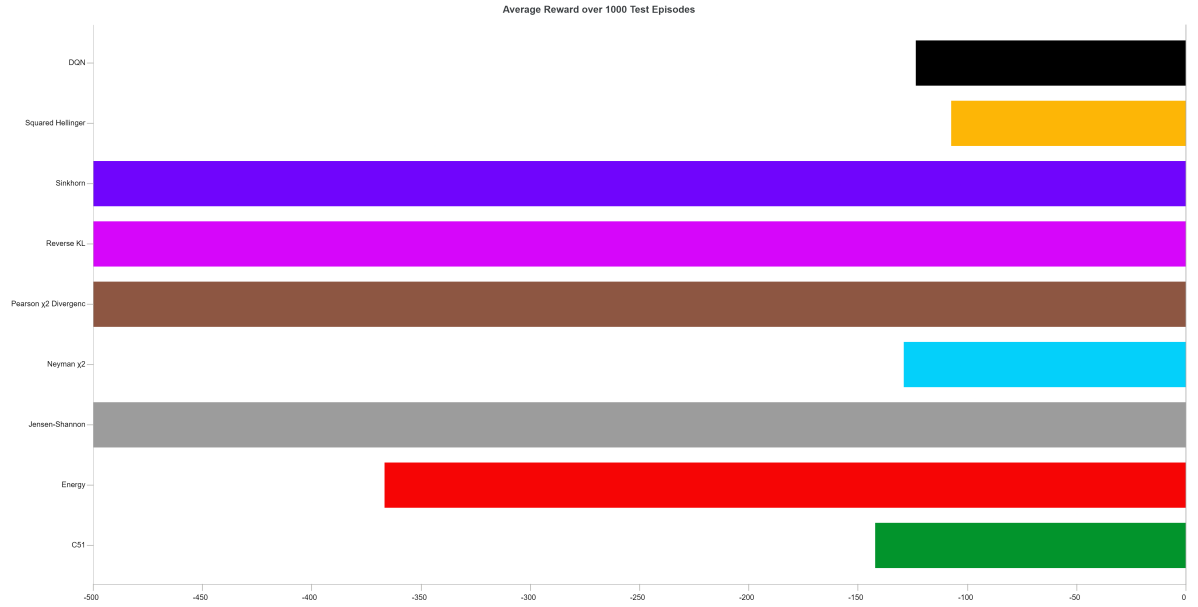

Figure 4.4: Comparison of the best test time performance on Acrobot

4.1 Discussion

Based on the above observations we can infer the following:

Both the baseline C51 algorithm which utilizes the KL divergence and the variation utilizing the Neyman χ^2 divergence as the categorical loss function performed well in both environments with the Neyman χ^2 divergence outperforming the baseline C51 algorithm during training. The Neyman χ^2 divergence came out with the best performance in the Cartpole environment during testing. The commonality between both successful methods is that they are both instances of *zero-avoiding* α divergence measures.

The variations utilizing the reverse KL divergence, Pearson χ^2 divergence and Jensen-Shannon divergence did not show any evidence of learning in both the Cartpole and Acrobot environments. The commonality between the first two methods is that they are both instances of *zero-forcing* α divergence measures. Theis et al. (2016) also showed that the Jensen-Shannon Divergence is also an instance of a zero-forcing divergence measure.

The variation utilizing the squared Hellinger distance performed adequately in the Cartpole though worse than the zero-avoiding α divergence measures. Surprisingly the Hellinger squared divergence provided the best performance in the Acrobot environment during testing. Since the Hellinger distance assigned probability mass in a fashion that was an amalgamation of both the zero-forcing and zero-avoiding property it would seem that for some environments a moderate influence of both properties might be most suitable.

The other class of methods we tested belonged to the family of integral probability metrics. Of the two the Energy distance fared better than the Sinkhorn divergence. The Energy distance performed better for the Cartpole environment than it did for the more complex Acrobot environment though the training time performance in general was sub-standard in contrast to the more successful crop of f divergence methods. Theis et al. (2016) had showed that the Energy distance was also an instance of a zero-forcing distance. However in contrast to the zero-forcing f divergence which failed to serve as viable categorical loss functions, the Energy

distance showed better evidence of learning. A possible reason for this could be the more informative nature of optimal transport based metrics which we had discussed in the theoretical analysis.

The Sinkhorn divergence on the other hand was not able to serve as a viable categorical loss function. The training time performance of the Sinkhorn divergence was the most unstable, with it showing promising performance for a multiplicity of training iterations on the Acrobot environment but ultimately collapsing leading to inferior training performance similar to the zero-forcing divergence measures. This would seem to indicate that while entropic regularization made computing an optimal transport based distance more scalable and tractable, the Sinkhorn divergence seems to also be affected by the underlying issue of *biased sample gradients* described by Bellemare et al. (2017b) that made the exact Wasserstein distance unsuitable for use in the distributional reinforcement learning setting.

Thus we can conclude that for the C51 algorithm, if an optimal transport based distance metric is to be used as the categorical loss function, then maximal entropic regularization would have to be applied.

Chapter 5

Conclusion

In this thesis we analyzed the performance of statistical distances from the class of Integral Probability Metrics and f divergences as viable categorical loss functions in the C51 algorithm. We observed that f divergences that possessed zero-avoiding properties or an amalgamation of zero-avoiding and zero-forcing properties were best suited to the task of minimizing the geometrical distance between the projected target value distribution and the estimate value distribution for solving classic control tasks. We also observed that with the addition of entropic regularization, the maximal entropy form of the optimal transport problem was more suited than the minimal entropy form as a categorical loss function in the C51 algorithm.

In the future we would like to test other family of divergence measures such as the Bregman divergence (Bregman, 1967), Stein divergence (James and Stein, 1992) (Sra, 2016) and Rényi divergence (Van Erven and Harremos, 2014) as feasible categorical loss functions in the C51 algorithm. Given the results we obtained for the classic control environments we would also like to study if similar or contrasting results will be obtained on testing these variations on the Arcade Learning Environment (Bellemare et al., 2013) as well as comparing training and test time performance to the QR-DQN (Dabney et al., 2017) and IQN (Dabney et al., 2018) algorithms. Finally we would like to study the behaviour of these categorical loss functions in the C51-lite algorithm which utilized linear function approximation and whose performance was contrasted with that of the DQN-lite algorithm by Lyle et al. (2019) that analyzed the advantages of using the distributional setting over the expectational setting.

References

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846.

Bellemare, M. G., Dabney, W., and Munos, R. (2017a). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017b). The cramer distance as a solution to biased wasserstein gradients. *CoRR*, abs/1705.10743.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Bertsekas, D. P. (2019). *Reinforcement learning and optimal control*. Athena Scientific Belmont, MA.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Caticha, A. (2004). Relative entropy and inductive inference. In *AIP conference proceedings*, volume 707, pages 75–96. American Institute of Physics.

Cevher, V., Waters, A., Beirami, A., and Kahle, D. (2008). ELEC 633 Graphical Models: Alpha divergence.

Chernoff, H. et al. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507.

Cichocki, A. and Amari, S.-i. (2010). Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568.

Csiszár, I. (1967). Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318.

Csiszár, I. and Körner, J. (1981). Information theory: Theorems for discrete memoryless systems. *Budapest, Hungary: Hungarian Acad. Sci.*

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300.

Cuturi, M. and Solomon, J. (2017). A primer on optimal transport. In *Tutorial of 31st Conference on Neural Information Processing Systems*.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2017). Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*.

Dudley, R. M. (1969). The speed of mean glivenko-cantelli convergence. *The Annals of Mathematical Statistics*, 40(1):40–50.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690.

Genevay, A., Chizat, L., Bach, F., Cuturi, M., and Peyré, G. (2019). Sample complexity of sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1574–1583. PMLR.

Genevay, A., Peyré, G., and Cuturi, M. (2018). Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710.

James, W. and Stein, C. (1992). Estimation with quadratic loss. In *Breakthroughs in statistics*, pages 443–460. Springer.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

Lyle, C., Bellemare, M. G., and Castro, P. S. (2019). A comparative analysis of expected and distributional reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4504–4511.

McCann, R. J. (1997). A convexity principle for interacting gases. *Advances in mathematics*, 128(1):153–179.

Minka, T. et al. (2005). Divergence measures and message passing. Technical report, Technical report, Microsoft Research.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010a). Nonparametric return distribution approximation for reinforcement learning. In *ICML*.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010b). Parametric return density estimation for reinforcement learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 368–375.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279.

Park, J., Kim, K., Chen, W., and Lei, W. (2019). Pytorch implementations of deep value function approximation architectures. Available at `https://git.io/JUhtB`.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Ramdas, A., Trillos, N. G., and Cuturi, M. (2017). On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Schulman, J., Levine, S., and Finn, C. (2017). CS 294: Deep Reinforcement Learning.

Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879.

Sra, S. (2016). Positive definite matrices and the s-divergence. *Proceedings of the American Mathematical Society*, 144(7):2787–2797.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., Lanckriet, G. R., et al. (2012). On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. (2010). Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Van Erven, T. and Harremos, P. (2014). Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Wilson, A. G. (1969). The use of entropy maximising models, in the theory of trip distribution, mode split and route split. *Journal of transport economics and policy*, pages 108–126.