algorithms

MDPI

Article

Distributional Reinforcement Learning with Ensembles

Björn Lindenberg *, Jonas Nordqvist and Karl-Olof Lindahl

Department of Mathematics, Linnæus University, 351 95 Växjö, Sweden; jonas.nordqvist@lnu.se (J.N.); karl-olof.lindahl@lnu.se (K.-O.L.)

* Correspondence: bjorn.lindenberg@lnu.se

Received: 8 April 2020; Accepted: 30 April 2020; Published: 7 May 2020

Abstract: It is well known that ensemble methods often provide enhanced performance in reinforcement learning. In this paper, we explore this concept further by using group-aided training within the distributional reinforcement learning paradigm. Specifically, we propose an extension to categorical reinforcement learning, where distributional learning targets are implicitly based on the total information gathered by an ensemble. We empirically show that this may lead to much more robust initial learning, a stronger individual performance level, and good efficiency on a per-sample basis.

Keywords: distributional reinforcement learning; multiagent learning; ensembles; categorical reinforcement learning

1. Introduction

The fact that ensemble methods may outperform single agent algorithms in reinforcement learning has been demonstrated numerous times [1–4]. These methods can involve combining several algorithms into one agent and then taking actions by a weighted aggregation scheme or rank voting. However, most conventional ensemble methods in reinforcement learning are often based on expected returns. Perhaps the simplest example is the average joint policy derived from an ensemble of independently trained agents, where the action of the ensemble is dictated by the average of the estimated Q-values of each agent.

An alternate view to that of Q-values, the distributional perspective on state-action returns, was discussed in [5]. This paradigm represents a shift of focus towards estimating or using underlying distributions of random return variables instead of learning expectations. This in turn paints a complex and more informationally dense picture, and there exists overwhelming empirical evidence that the distributional perspective is helpful in deep reinforcement learning. That is, apart from the possibility of overall stronger performance, algorithmic benefits may also involve the reduction of prediction variance, more robust learning with additional regularization effects, and a larger set of auxiliary goals such as learning risk-sensitive policies [5–9]. Moreover, there have recently been important theoretical works done on understanding the observed improvements and providing theoretical results on convergence [5,9–11].

In this paper, we propose a group-aided training scheme for distributional reinforcement learning, where we merge the distributional perspective with an ensemble method involving agents learning in separate environments. Our main contribution in this regard is the proposed Ensemble Categorical Control procedure (ECCprocedure). As an initial study, we also provide empirical results where an ECCalgorithm is tested on a subset of Atari 2600 games [12], which are standard environments for testing these types of algorithms.

Specifically, ECC is an extension of Categorical Distributional Reinforcement Learning (CDRL), which was introduced in [5] and made explicit in [10]. Similar to CDRL, we consider distributions defined on a fixed discrete support, with projections onto the support for all possible categorical

distributions arising internally in the algorithm. For each agent in ECC, we replace the target generation of CDRL by targets generated by the ensemble mean mixture distribution of the individual target distributions.

We argue that ECC implies an implicit sharing of information between agents during learning, where the distributional paradigm gives us more robust targets and an arguably more nuanced aggregated picture, which preserves multimodality. The experiments confirm the validity of the approach, where in all cases, the extension generates strong individual agents and good efficiency when regarded as an ensemble.

The paper is organized in the following way. In Section 2, we give a background to distributional reinforcement learning. In Section 3, we introduce the proposed ECCprocedure. At the end of Section 3, we give a reference to another contribution of the present work: the pseudocode and source code for an implementation of the ECCalgorithm. In Section 4, we present and evaluate the results of our implementation of the ECCalgorithm on five specific Atari 2600 environments. Finally, in Section 5, we zoom out and discuss the results in a broader context, as well as suggest future work.

2. Background

We considered agent-environment interactions. For each observed state, the agent selects an action, whereby the environment generates a reward and a next state. Following the framework of [10], we let $\mathcal{X}$ and $\mathcal{A}$ denote the sets of states and actions, respectively, and let $p\colon \mathcal{X} \times \mathcal{A} \to \mathscr{P}(\mathbb{R} \times \mathcal{X})$ be a transition kernel that maps state-action pairs to joint distributions of immediate rewards and next states. Then, we can model this interaction by a Markov Decision Process (MDP) $(\mathcal{X}, \mathcal{A}, p, \gamma)$, where $\gamma \in [0, 1)$ is a discount factor of future rewards. Moreover, an agent can sample its actions through a stationary policy $\pi\colon \mathcal{X} \to \mathscr{P}(\mathcal{A})$, which maps a current state to a distribution over available actions.

Throughout the rest of this paper, we consider MDPs where $\mathcal{X} \times \mathcal{A}$ is a countable state-action space. We denote by $\mathcal{D} = \mathscr{P}(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ the set of functions where $\eta \in \mathcal{D}$ maps each state-action pair (x, a) to a distribution $\eta^{(x,a)} \in \mathscr{P}(\mathbb{R})$. Similarly, we put $\mathcal{D}_n = \mathscr{P}_n(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$, where $\mathscr{P}_n(\mathbb{R})$ is the set of probability distributions with finite n^{th}-moments. For a given $\eta \in \mathcal{D}$, we let $Q_\eta\colon \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ denote the function that maps state-action pairs $\{(x, a)\}$ to the corresponding first moments of $\{\eta^{(x,a)}\}$, i.e.,

$$Q_\eta(x, a) \coloneqq \int_{\mathbb{R}} z \, \eta^{(x,a)}(dz).$$

To appreciate a subsequent summary of distributional reinforcement theory fully, we may also need to make the following definition explicit.

Definition 1. *For a Borel measurable function $g\colon \mathbb{R} \to \mathbb{R}$ and $\nu \in \mathscr{P}(\mathbb{R})$, we let $g_\#\nu$ denote the push-forward measure defined by:*

$$g_\#\nu(A) \coloneqq \nu\left(g^{-1}(A)\right)$$

on all Borel sets $A \subseteq \mathbb{R}$. In particular, given $r, \gamma \in \mathbb{R}$, we let $(f_{r,\gamma})_\#\nu$ be the push-forward measure where $f_{r,\gamma}(x) \coloneqq r + \gamma x$.

Suppose further that we have a set $\mathcal{P}$ of categorical distributions supported on a fixed set $\mathbf{z} = \{\, z_1, z_2, \ldots, z_K \,\}$ of equally-spaced numbers. Then, the following projection operator minimizes the distance between any categorical distribution $\nu = \sum_{i=1}^{n} p_i \delta_{y_i}$ and elements in $\mathcal{P}$ with respect to the Cramér metric [9,13].

Definition 2. *The Cramér projection $\Pi_{\mathbf{z}}$ maps any Dirac measure δ_y to a distribution in $\mathcal{P}$ by:*

$$\Pi_{\mathbf{z}}(\delta_y) = \begin{cases} \delta_{z_1} & y \leq z_1, \\ \frac{z_{i+1}-y}{\Delta z}\delta_{z_i} + \frac{y-z_i}{\Delta z}\delta_{z_{i+1}} & z_i < y \leq z_{i+1}, \\ \delta_{z_K} & y > z_K. \end{cases}$$

Moreover, the projection is defined to be linear over mixture distributions such that:

$$\Pi_{\mathbf{z}}\left(\sum_i p_i \delta_{y_i}\right) = \sum p_i \Pi_{\mathbf{z}}\left(\delta_{y_i}\right).$$

2.1. Expected Reinforcement Learning

Before we go into the distributional perspective, let us first give a quick reminder about some value function fundamentals, here stated in operator form.

Let $(\mathcal{X}, \mathcal{A}, p, \gamma)$ be an MDP. Given $(x, a) \in \mathcal{X} \times \mathcal{A}$, we define the return of a policy π as the random variable:

$$Z_\pi(x,a) := \sum_{t=0}^{\infty} \gamma^t R_t \;\Big|\; X_0 = x, A_0 = a\,, \tag{1}$$

where $(R_t)_{t=0}^{\infty}$ is a random sequence of immediate rewards, indexed by time step t and dependent on random state-action pairs $(X_t, A_t)_{t=0}^{\infty}$ under p and π.

In an evaluation setting of some fixed policy π, let $Q_\pi \colon \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ be the expected return function, which by definition has values:

$$Q_\pi(x,a) = \mathbb{E}[Z_\pi(x,a)].$$

If we consider distributions dictated by p and π and let $R(x,a)$ and (X', A') denote the random reward and subsequent random state-action pair given $(x,a) \in \mathcal{X} \times \mathcal{A}$, then we recall the Bellman operator $\mathcal{T}^\pi$ defined by:

$$\forall (x,a)\ \ (\mathcal{T}^\pi g)(x,a) = \mathbb{E}_p[R(x,a)] + \gamma \mathbb{E}_{p,\pi}[g(X', A')] \tag{2}$$

on bounded real functions $g \in \mathcal{B}(\mathcal{X} \times \mathcal{A}, \mathbb{R})$. Moreover, in the search for values attained by optimal policies, we also recall the optimality operator $\mathcal{T}^*$ where:

$$\forall (x,a)\ \ (\mathcal{T}^* g)(x,a) = \mathbb{E}_p[R(x,a)] + \gamma \mathbb{E}_p[\max_{a'} g(X', a')]. \tag{3}$$

It is readily verified that both operators are contraction maps on the complete metric space $(\mathcal{B}(\mathcal{X} \times \mathcal{A}, \mathbb{R}), d_\infty)$. In addition, their unique fixed points are given by Q_π and Q^*, respectively, where Q^* is the optimal function defined by:

$$Q^*(x,a) = \max_\pi Q_\pi(x,a)$$

for all (x,a) [14].

2.2. Distributional Reinforcement Learning

We now proceed by presenting some of the main ideas of distributional reinforcement learning in a tabular setting. We will first look at the evaluation problem, where we are trying to find the state-action value of a fixed policy π. Second, we consider the control problem, where we try to find the optimal state-action value. Third, we consider the distributional approximation procedure CDRL used by agents in this paper.

2.2.1. Evaluation

We consider a distributional variant of (2), the distributional Bellman operator given by $T^\pi : \mathcal{D} \to \mathcal{D}$,

$$\forall (x,a)\ \left(T^\pi \eta\right)^{(x,a)} := \int_{\mathbb{R}} \sum_{(x',a') \in \mathcal{X} \times \mathcal{A}} (f_{r,\gamma})_{\#} \eta^{(x',a')}\ \pi(a' \mid x') p(dr, x' \mid x, a). \tag{4}$$

Here, T^π is, for all $n \geq 1$, a γ-contraction in $\mathcal{D}_n$ with a unique fixed point when $\mathcal{D}_n$ is endowed with the supremum n^{th}-Wasserstein metric ([5], Lemma 3) (see [15] for more details on Wasserstein distances). Moreover by Proposition 2 of [9], T^π is expectation preserving when we have an initial coupling with the $\mathcal{T}^\pi$-iteration given in (2); that is, given an initial $\eta_0 \in \mathcal{D}$ and a function g, such that $g = Q_{\eta_0}$. Then, $\left(\mathcal{T}^\pi\right)^n g = Q_{(T^\pi)^n \eta_0}$ holds for all $n \geq 0$.

Thus, if we let $\eta_\pi \in \mathcal{D}$ be the function of distributions of Z_π in (1), then η_π is the unique fixed point satisfying the distributional Bellman equation:

$$\eta_\pi = T^\pi \eta_\pi.$$

It follows that iterating T^π on any starting collection η_0 with bounded moments eventually solves the evaluation task of π to an arbitrary degree.

2.2.2. Control

Recall the Bellman optimality operator $\mathcal{T}^*$ of (3). If we define a corresponding distributional optimality operator $T^* : \mathcal{D} \to \mathcal{D}$,

$$\forall (x,a)\ \left(T^* \eta\right)^{(x,a)} := \int_{\mathbb{R}} \sum_{(x',a') \in \mathcal{X} \times \mathcal{A}} (f_{r,\gamma})_{\#} \eta^{(x',a^*(x'))}\ p(dr, x' \mid x, a), \tag{5}$$

where $a^*(x') = \arg\max_{a' \in \mathcal{A}} Q_\eta(x', a')$, then expectation values generated by iterates under T^* will behave as expected. That is, if we put $Q_n := Q_{(T^*)^n \eta_0}$, then we have an exponentially fast uniform convergence $Q_n \to Q^*$ as $n \to \infty$. However, T^* is not a contraction in any metric over distributions and may lack fixed points altogether in $\mathcal{D}$ [5].

2.2.3. Categorical Evaluation and Control

In most real applications, the updates of (4) and (5) are either computationally infeasible or impossible to fully compute due to p being unknown. It follows that approximations are key to defining practical distributional algorithms. This could involve parametrization over some selected set of distributions along with projections onto these distributional subspaces. It could also involve stochastic approximations with sampled transitions and gradient updates with function approximation.

A structure for algorithms making use of such approximations is Categorical Distributional Reinforcement Learning (CDRL). In what follows is a short summary of the CDRL procedure fundamental to single agent implementations in this paper.

Let $\mathbf{z} = \{ z_1, z_2, \ldots, z_K \}$ be an ordered fixed set of equally-spaced real numbers such that $z_1 < z_2 < \cdots < z_K$ with $\Delta z := z_{i+1} - z_i$. Let:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{z_i} : p_1, \ldots, p_K \geq 0,\ \sum_{i=1}^{K} p_i = 1 \right\} \subset \mathscr{P}(\mathbb{R})$$

be the subset of categorical distributions in $\mathscr{P}(\mathbb{R})$ supported on $\mathbf{z}$. We consider parameterized distributions by using $\widehat{\mathcal{D}} = \mathcal{P}^{\mathcal{A}\times\mathcal{X}}$ as the collection of possible inputs and outputs of an algorithm. Moreover, for each $\eta \in \widehat{\mathcal{D}}$, we have:

$$Q_\eta(x,a) = \sum_{i=1}^{K} p_i(x,a) z_i.$$

as its Q-value function.

Given a subsequent treatment of our extension of CDRL, we first reproduce the steps of the general procedure in Algorithm 1 (see [10], Algorithm 1).

Algorithm 1: Categorical Distributional Reinforcement Learning (CDRL)

1. At each iteration step t and input $\eta_t \in \widehat{\mathcal{D}}$, sample a transition (x_t, a_t, r_t, x'_t).
2. Select a^* to be either sampled from $\pi(x_t)$ in the evaluation setting or taken as $a^* = \arg\max_a Q_{\eta_t}(x'_t, a)$ in the control setting.
3. Recall the Cramér projection $\Pi_{\mathbf{z}}$ given in Definition 2, and put:
$$\widehat{\eta}_t^{(x_t,a_t)} := \Pi_{\mathbf{z}} \left(f_{r_t}\right)_{\#} \eta_t^{(x'_t,a^*)}.$$
4. Take the next iterated function as some update η_{t+1} such that:
$$\mathrm{KL}\left(\widehat{\eta}_t^{(x_t,a_t)} \parallel \eta_{t+1}^{(x_t,a_t)}\right) < \mathrm{KL}\left(\widehat{\eta}_t^{(x_t,a_t)} \parallel \eta_t^{(x_t,a_t)}\right),$$
where:
$$\mathrm{KL}(\mathbf{p} \parallel \mathbf{q}) := \sum_{i=1}^{K} p_i \log\left(\frac{p_i}{q_i}\right)$$
denotes the Kullback–Leibler divergence.

Consider first a finite MDP and a tabular setting. Define $\widehat{\eta}_t^{(x,a)} := \eta_t^{(x,a)}$ whenever $(x,a) \neq (x_t, a_t)$. Then, by the convexity of $-\log(z)$, it is readily verified that updates of the form:

$$\eta_{t+1} = (1-\alpha_t)\eta_t + \alpha_t \widehat{\eta}_t \qquad (\alpha_t \in (0,1))$$

satisfy Step 4. In fact, if there exists a unique policy π^* associated with the convergence of (3), then this update yields an almost sure convergence, with respect to the supremum-Cramér metric, to a distribution in $\widehat{\mathcal{D}}$ with π^* as the greedy policy (with some additional assumptions on the stepsizes α_t and sufficient support (see [10], Theorem 2, for details).

In practice, we are often forced to use function approximation of the form:

$$\eta^{(x,a)} = \phi(x,a;\boldsymbol{\theta}),$$

where ϕ is parameterized by some set of weights $\boldsymbol{\theta}$. Gradient updates with respect to $\boldsymbol{\theta}$ can then be made to minimize the loss:

$$\mathrm{KL}\left(\widehat{\eta}_t^{(x_t,a_t)} \parallel \phi\left(x_t, a_t; \boldsymbol{\theta}\right)\right), \tag{6}$$

where $\widehat{\eta}_t^{(x_t,a_t)} = \Pi_{\mathbf{z}} \left(f_{r_t}\right)_{\#} \phi(x'_t, a^*; \boldsymbol{\theta}_{\text{fixed}})$ is the computed learning target of the transition (x_t, a_t, r_t, x'_t). However convergence with the Kullback–Leibler loss and function approximation is still an open question. Theoretical progress has been made when considering other losses, although we may lose the stability benefits coming from the relative ease of minimizing (6) [9,11,16].

An algorithm implementing CDRL with function approximation is `C51`[5]. It essentially uses the same neural network architecture and training procedure as DQN[17]. To increase stability during

training, this also involves sampling transitions from an experience buffer and maintaining an older, periodically updated, copy of the weights for target computation. However, instead of estimating Q-values, `C51` uses a finite support $\mathbf{z}$ of 51 points and learns discrete probability distributions $\phi(x, a; \boldsymbol{\theta})$ over $\mathbf{z}$ via soft-max transfer. Training is done by using the KL-divergence as the loss function over batches with computed targets $\widehat{\eta}^{(x,a)}$ of CDRL.

3. Learning with Ensembles

3.1. Ensembles

Ensemble methods have been widely used in both supervised learning and reinforcement learning. In supervised learning, this can involve bootstrap aggregating predictors for better accuracy when given unstable processes such as neural networks or using "expert" opinion mixtures for better estimators [18,19]. A simple example that demonstrates the possible benefits of aggregation is the following average pool of k regression models: Given a sample to predict, assume that the models draw prediction errors ε_i, $i = 1, \ldots, k$ from a zero-mean multivariate normal distribution with $\mathbb{E}[\varepsilon_i^2] = \sigma^2$ and correlations $\rho_{ij} = \rho$. Then, the error made by averaging their predictions is $\varepsilon := (1/k) \sum_{i=1}^{k} \varepsilon_i$ with:

$$\mathbb{E}[\varepsilon^2] = (1 + \rho(k-1)) \, \frac{\sigma^2}{k}.$$

It follows that the mean squared error goes to σ^2/k as $\rho \to 0$, whereas we get σ^2 and no benefit when the errors are perfectly correlated.

Under the assumption of independently trained agents, we have a reinforcement learning variant of the average pool in the following definition.

Definition 3. *Given an ensemble of k agents, let $\widehat{Q}^{(i)}$ denote the Q-value function estimate of agent i, and let $\widehat{Q} := (1/k) \sum_{i=1}^{k} \widehat{Q}^{(i)}$ denote the mean function. Then, the average joint policy $\overline{\pi}$ selects actions according to:*

$$a^* = \arg\max_a \widehat{Q}(x, a) = \arg\max_a \frac{1}{k} \sum_{i=1}^{k} \widehat{Q}^{(i)}(x, a).$$

at every $x \in \mathcal{X}$.

Thus, $\overline{\pi}$ represents an aggregation strategy where we consider the information provided by each agent as equally important. Moreover, by the linearity of expectations and in view of (3), if we have initial functions $Q_0^{(i)}$ with n-step ensemble values $Q_n := (1/k) \sum_{i=1}^{k} Q_n^{(i)}$, then full updates $Q_n^{(i)} := \mathcal{T}^* Q_{n-1}^{(i)}$ of each agent will yield $Q_n = \mathcal{T}^* Q_{n-1}$ for the ensemble. Assume further that learning is done with a single algorithm in separate environments. If we take $\widehat{Q}^{(i)}(x, a)$ as estimates of $Q_n^{(i)}(x, a)$ for some step n, with errors ε_i distributed as multivariate Gaussian noise, then we should expect $\widehat{Q}(x, a)$ to have a smaller expected error variance in its estimation of $Q_n(x, a)$ similar to regression models. This implies more robust performance when given an unstable training process far from convergence, but it also implies diminishing improvements when the algorithm is close to converging to a unique policy.

However, in real applications, and in particular with function approximation, there may be instances where the improved performance by $\overline{\pi}$ does not vanish due to agents converging to distinct sub-optimal policies. An illustration of this phenomenon can be seen in Figure 1. It shows evaluations during learning in the `LunarLander-v2` environment [20]. The single agents used CDRL on a 29 atom support. To approximate distributions, the agents used small neural networks with three encoding layers consisting of 16 units each. The architecture was purposely chosen to make it harder for the optimizer to converge to an optimal policy, possibly due to lack of capacity. At each evaluation point, the models were tested with $\varepsilon = 0.001$. The figure also includes evaluations of average joint policies of

five agents having the same evaluation ε. However, we can see that the joint information provided by an ensemble of five agents transcends individual capacity, indicating that some agents settle on distinct sub-optimal solutions.

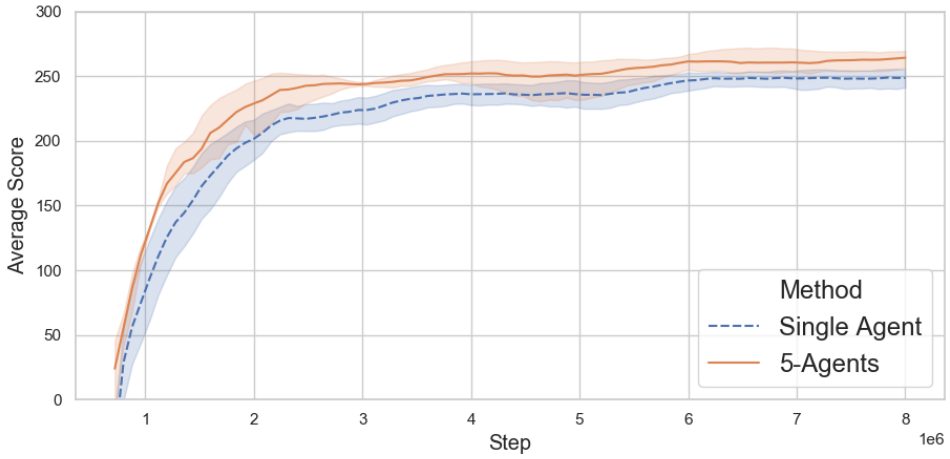

Figure 1. Low capacity CDRL implementations in the LunarLander-v2 environment. We can see that the enhanced performance of an average joint policy of five agents may not vanish due to agents settling on distinct sub-optimal policies.

3.2. Ensemble Categorical Control

We consider an ensemble of k agents, each independently trained with the same distributional algorithm, where η_i, $i = 1, \ldots, k$ are their respective distributional collections. There are several ways to aggregate distributional information provided by the ensemble with respect to forecasts and risk-sensitivity [21,22]. Perhaps the simplest is a distributional variant of the average joint policy, where we consider the mean function $\overline{\eta}$ of mixture distributions:

$$\forall (x, a)\ \overline{\eta}^{(x,a)} := \frac{1}{k} \sum_{i=1}^{k} \eta_i^{(x,a)}. \tag{7}$$

Since $\overline{\eta}^{(x,a)}$ is a linear pool, it preserves multimodality during aggregation. Hence, it maintains an arguably more nuanced picture of estimated future rewards compared to methods that generate unimodal aggregations around unrealizable expected values. In addition, expectations under $\overline{\eta}$ yield the Q-function used by the average joint policy in Definition 3 with all the performance benefits that this entails during learning.

The finite support of the CDRL procedure may provide another reason to aggregate by $\overline{\eta}$: Under the assumption that $\eta_i^{(x,a)}$, $i = 1, \ldots, k$ are drawn as random vectors from some multivariate normal population with mean $\boldsymbol{\mu}(x, a)$ and covariance $\boldsymbol{\Sigma}(x, a)$, then $\overline{\eta}$ is a maximum likelihood estimate of the mean categorical distribution $\boldsymbol{\mu}(x, a)$ induced by the algorithm over all possible training runs [23]. It follows that $\overline{\eta}$ may provide more robust estimates in reflecting mean t-step capabilities of the procedure in terms of distributions found by sending $k \to \infty$.

It then stands to reason that (7) should help accelerate learning by providing better and more robust targets in the control setting of CDRL. This implies implicitly sharing information gained between agents and following accelerated learning trajectories closer to the true expected capability of an algorithm. We can summarize this as an extension of the CDRL control procedure.

For a fixed support $\mathbf{z}$, we parameterize individual distribution functions $\eta_{i,t}$, $i = 1, \ldots, k$, at time step t by using $\widehat{\mathcal{D}} = \mathcal{P}^{\mathcal{A} \times \mathcal{X}}$ as possible inputs and outputs of the algorithm. Let $\overline{\eta}_t$ be the mean function of $\{\eta_{i,t}\}_{i=1}^{k}$ according to (7). The extension is then given by Algorithm 2.

Algorithm 2: Ensemble Categorical Control (ECC)

1. At each iteration step t and for each agent input $\eta_{i,t}$, sample a transition (x, a, r, x').
2. Let $a^* = \arg\max_{a'} Q_{\overline{\eta}_t}(x', a')$.
3. Recall the Cramér projection $\Pi_{\mathbf{z}}$ given in Definition 2, and put:

$$\widehat{\eta}_{i,t}^{(x,a)} := \Pi_{\mathbf{z}} \, (f_r)_{\#} \, \overline{\eta}_t^{(x',a^*)}.$$

4. For each agent, follow Step 4 of CDRL with target $\widehat{\eta}_{i,t}^{(x,a)}$.

We note that if updates are done in full or on the same transitions, then the algorithm trivially reduces to CDRL by the linearity of $(f_r)_{\#}$; hence, we lose the benefits of the ensemble.

To avoid premature convergence to correlated errors, we would ideally want the agents to have the freedom to explore different trajectories during learning. In the case of function approximation, this can involve maintaining a separate experience buffer for each agent. It can also involve periodical updates of ensemble target networks in the hope of generating sufficiently diverse policies until convergence. The latter is in practical terms the only way to minimize overhead costs induced by inter-thread ensemble queries in simulations. Too short periods here imply fast initial learning; but with correlated errors, high overhead costs, and instability [17]. Long periods would imply the possibility of more diverse policies, but with slower learning. The pseudocode for an algorithm using function approximation with ECC can be found in Algorithm A1. The source code for an implementation of ECC can be found at [24].

4. Empirical Results on a Subset of Atari 2600 Games

As a first step in understanding the properties of the extension ECC discussed in Section 3.2, we now evaluate an implementation of the procedure on five Atari 2600 environments found in the Arcade Learning Environment [12,20,25].

Specifically, we looked at ensembles of $k = 5$ agents. To get a proper comparison of the algorithms, we employed for all agents the well-tested architecture, hyperparameters, and training procedure as `C51` in [5]; except for a slightly smaller individual replay buffer size at 900 K. This yielded an implicit buffer size of 4.5 M for the entire ECCensemble. In addition, we employed for each ECC agent a larger ensemble target network. The network consisted of copied weights from all ECCnetworks and was updated periodically at every 10K steps with negligible overhead.

We trained k agents on the first 40 M frames (roughly 185 h of Atari-time at 60 Hz). Agent models were saved every 400 K frames. For each save, we evaluated the performance of the individual agents (ECCagent) and the ensemble with an average joint policy (ECCensemble). Moreover, we took an ensemble of $k = 5$ independently trained agents using $\overline{\pi}$ as our baseline (CDRL joint). For comparison, we also evaluated each such single agent (CDRL agent). In all performance protocols, we started an episode under the 30 no-op regime [17] with an exploration epsilon set to $\varepsilon = 0.001$. The evaluation period was 500 K frames with episodes truncated at 108 K frames (30 min).

In our particular implementation in [24], each algorithm required roughly two days of compute time per environment for training and evaluation combined. Single replay buffers used ~35 GB of optimized RAM (~47 GB raw); hence, we used ~175 GB of RAM for concurrently training the ECCensemble.

4.1. Online Performance

To get a sense of the algorithmic robustness and speed of learning, we report the online performance of agents and ensembles [7]. Under this protocol, we recorded the average return for each evaluation point during learning. We also stored the best average return score for all points of each seed.

We can see in Table 1 and Figure 2 that the extension ensemble was on par or outperformed the baseline in online performance over all five environments. Moreover, in four out of five games, single ECC agents had similar performance to the joint policy of k independently trained agents, which was the main training objective of the extension algorithm. We also note that in all environments, except possibly `Breakout` and `KungFuMaster`, ensemble agents seemed to be uncorrelated enough to generate a boost in performance by their joint information, while ECC agents had a better individual performance than single CDRL agents in four out of five games.

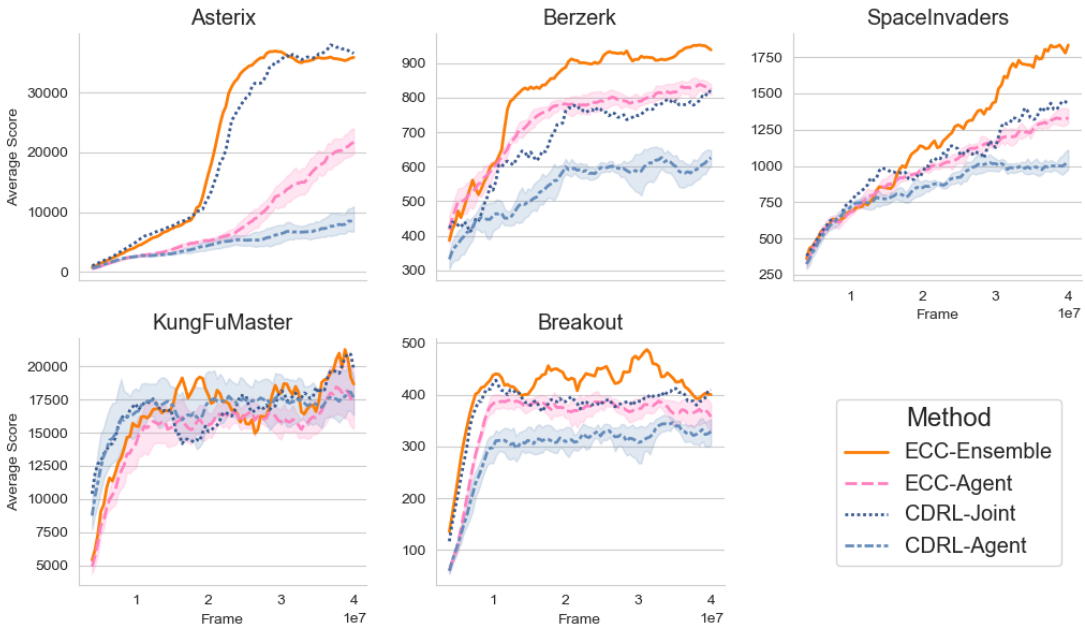

Figure 2. Online performance over the first 40 M frames. The evaluation scores shown are moving averages over 4 M frames. The data are available at [24].

Table 1. Best achieved evaluation scores in online performance over the first 40 M samples, here with 95% confidence when there is more than one seed. The data are available at [24].

Game	CDRL Agent	ECCAgent	CDRL Joint	ECCEnsemble
Asterix	12,998 ± 3042	28,196 ± 903	39,413	38,938
Berzerk	795 ± 47	958 ± 12	890	1034
SpaceInvaders	1429 ± 91	1812 ± 87	1850	2395
Breakout	444 ± 44	546 ± 27	515	665
KungFuMaster	27,984 ± 1767	27,302 ± 2213	25,826	29,629

4.2. Relative Ensemble Sample Performance

Although ensembles will digest frames at nearly k times the rate of a single CDRL algorithm, we considered here the relative sample performance, where we looked at performance versus the total information accumulated by an algorithm. Under this protocol, we measured the relative ratio of mean evaluation scores as a function of the total amount of frames seen by each learning system. This would give us an idea of how efficiently an ensemble algorithm could translate experience into performance on a per-sample basis compared to single CDRL. Note that if single CDRL agents all converged to correlated errors, then the joint policy should eventually converge to $1/k$-efficiency in relative sample performance. Thus, in general, we should expect the relative performance to degrade as training progresses with diminishing ensemble benefits.

Table 2 shows the measured relative performance of the two ensemble methods, averaged over the first 40 M samples. We note that initial learning with ensembles may generate performance much higher than $1/k$-efficiency. We also note that the extension ensemble came close to full efficiency in

`Berzerk` and `Breakout`, i.e., it displayed a near k-factor increase in learning rate. However, depending on the environment, the actual speed-up may vary wildly during learning, as shown in Figure 2.

Table 2. Rough estimates of relative sample performance, here expressed as percentages of CDRL agent performance and averaged over the first 40 M samples. The data are available at [24].

Method	Asterix	Berzerk	Breakout	SpaceInvaders	KungFuMaster
ECCEnsemble	47.7 %	93.7 %	93.7 %	63.4 %	66.9 %
CDRL Joint	56.3 %	86.7 %	86.1 %	67.2 %	87.0 %

5. Discussion

In this paper, we proposed and studied an extension of categorical distributional reinforcement learning, where we employed averaged learning targets over an ensemble. This extension implied an implicit sharing of information between agents during learning, where under the distributional paradigm, we should expect a richer and more robust set of predictions while preserving multimodality during aggregation. To test these assumptions, we did an initial empirical study on a subset of Atari 2600 games, where we employed essentially the same architecture and hyperparameter set as the `C51` algorithm in [5]. In all cases, we saw that the single agent performance objective of the extension was accomplished. We also studied the effects of keeping extension amplified agents in an ensemble, where in some cases, the performance benefits were present and stronger than an averaged ensemble of independent agents.

We note that unlike massively distributed approaches such as Ape-X [26], the extension represents a decentralized distributed learning system with minimal overhead. As such, it naturally comes with poor scalability, but with greater efficiency on a per-sample basis. An interesting idea here would be to somewhat counteract the poor scalability by choosing agents with successively lower capacity as the ensemble size increases. We should then expect to see better performance with increasing size until a cutoff point is reached, hinting at the minimum capacity needed to find and represent strong solutions effectively.

We leave as future work the matter of convergence analysis and hyperparameter tuning, in particular the update period for a target ensemble network. It is quite possible that the update frequency of `C51` was too aggressive when using ensemble targets. This may lead to premature convergence to correlated agents upon reaching difficult environmental plateaus with rarely seen transitions to more abundant rewards. Some interesting ideas here would be scheduled update periods or eventually switching to CDRL from a much stronger and robust level of individual performance. However, to gauge these matters fully, we would need a more comprehensive empirical study.

Author Contributions: Conceptualization, B.L., J.N., and K.-O.L.; methodology, B.L. and J.N.; software, B.L.; validation, B.L.; formal analysis, B.L., J.N., and K.-O.L.; investigation, B.L.; data curation, B.L.; writing, original draft preparation, B.L.; writing, review and editing, B.L., J.N., and K.-O.L.; visualization, B.L.; supervision, K.-O.L.; project administration, K.-O.L. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the referees for comments that helped improve the presentation. The authors would also like to thank Morgan Ericsson, Department of Computer Science and Media Technology, Linnæus University, for productive discussions and technical assistance with the LNU-DISAHigh Performance Computing Platform.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CDRL	Categorical Distributional Reinforcement Learning
MDP	Markov Decision Process
ECC	Ensemble Categorical Control

Appendix A

Algorithm A1: Ensemble categorical control.

Input: Number of iteration steps N, ensemble size k, support $\mathbf{z}$
Initialize starting states $x_1, \ldots, x_k$ in independent environments
Initialize agent networks $\eta_{\theta_1}, \ldots, \eta_{\theta_k}$ with random parameters $\theta_1, \ldots, \theta_k$
Initialize target network $\overline{\eta} = \frac{1}{k} \sum_i \eta_{\theta_i^-}$ with $\theta_i^- \leftarrow \theta_i$
Initialize replay buffers $\mathcal{B}_1, \ldots, \mathcal{B}_k$ with the same size S
for $t = 1$ **to** N **do**
  for all $i \in \{1, \ldots, k\}$ **do**
    Set a_i to be a uniform random action with probability ε_t
    Otherwise, set $a_i \leftarrow \arg\max_{a'} Q_{\eta_{\theta_i}}(x_i, a')$
    Execute a_i, and store the transition (x_i, a_i, r_i, x_i') in $\mathcal{B}_i$
    Set $x_i \leftarrow x_i'$
  end for
  if $t \equiv 0 \mod P_{\text{update}}$ **then**
    for all $i \in \{1, \ldots, k\}$ **do**
      Initialize loss $L \leftarrow 0$
      Sample uniformly a minibatch $B \subset \mathcal{B}_i$
      for all $(x, a, r, x') \in B$ **do**
        Set $a^* \leftarrow \arg\max_{a'} Q_{\overline{\eta}}(x', a')$
        Set $L \leftarrow L + \mathrm{KL}\left(\Pi_{\mathbf{z}} (f_r)_{\#} \overline{\eta}^{(x', a^*)} \parallel \eta_{\theta_i}^{(x,a)}\right)$
      end for
      Update θ_i by a gradient descent step on L
    end for
  end if
  if $t \equiv 0 \mod P_{\text{clone}}$ **then**
    for all $i \in \{1, \ldots, k\}$ **do**
      Update target network with $\theta_i^- \leftarrow \theta_i$
    end for
  end if
end for

References

1. Singh, S.P. The efficient learning of multiple task sequences. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1992; pp. 251–258.
2. Sun, R.; Peterson, T. Multi-agent reinforcement learning: weighting and partitioning. *Neural Netw.* **1999**, *12*, 727–753. [CrossRef]
3. Wiering, M.A.; Van Hasselt, H. Ensemble algorithms in reinforcement learning. *IEEE Trans. Syst. Man, Cybern. Part B (Cybernetics)* **2008**, *38*, 930–936. [CrossRef] [PubMed]
4. Faußer, S.; Schwenker, F. Selective neural network ensembles in reinforcement learning: Taking the advantage of many agents. *Neurocomputing* **2015**, *169*, 350–357. [CrossRef]

5. Bellemare, M.G.; Dabney, W.; Munos, R. A distributional perspective on reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 449–458.
6. Morimura, T.; Sugiyama, M.; Kashima, H.; Hachiya, H.; Tanaka, T. Parametric return density estimation for reinforcement learning. In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, 8–11 July 2010; pp. 368–375.
7. Dabney, W.; Rowland, M.; Bellemare, M.G.; Munos, R. Distributional reinforcement learning with quantile regression. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
8. Dabney, W.; Ostrovski, G.; Silver, D.; Munos, R. Implicit Quantile Networks for Distributional Reinforcement Learning. *Int. Conf. Mach. Learn.* **2018**, *80*, 1096–1105.
9. Lyle, C.; Bellemare, M.G.; Castro, P.S. A comparative analysis of expected and distributional reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4504–4511.
10. Rowland, M.; Bellemare, M.; Dabney, W.; Munos, R.; Teh, Y.W. An Analysis of Categorical Distributional Reinforcement Learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Canary Islands, Spain, 9–11 April 2018; pp. 29–37.
11. Bellemare, M.G.; Le Roux, N.; Castro, P.S.; Moitra, S. Distributional reinforcement learning with linear function approximation. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, Okinawa, Japan, 16–18 April 2019; pp. 2203–2211.
12. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
13. Rizzo, M.L.; Székely, G.J. Energy distance. *Wiley Interdiscip. Rev. Comput. Stat.* **2016**, *8*, 27–38. [CrossRef]
14. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-Dynamic Programming*; Athena Scientific: Belmont, MA, USA, 1996.
15. Villani, C. *Optimal Transport: Old and New*; Springer Science & Business Media: Berlin, Germany, 2008; Volume 338.
16. Bellemare, M.G.; Danihelka, I.; Dabney, W.; Mohamed, S.; Lakshminarayanan, B.; Hoyer, S.; Munos, R. The Cramer Distance as a Solution to Biased Wasserstein Gradients. *arXiv* **2017**, arXiv:1705.10743.
17. 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. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
18. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
19. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
20. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.
21. Clemen, R.T.; Winkler, R.L. Combining probability distributions from experts in risk analysis. *Risk Anal.* **1999**, *19*, 187–203. [CrossRef]
22. Casarin, R.; Mantoan, G.; Ravazzolo, F. Bayesian calibration of generalized pools of predictive distributions. *Econometrics* **2016**, *4*, 17. [CrossRef]
23. Johnson, R.A.; Wichern, D.V. *Applied Multivariate Statistical Analysis*; Pearson: Harlow, UK, 2014.
24. Lindenberg, B.; Nordqvist, J.; Lindahl, K.O. bjliaa/ecc: ecc; (Version v0.3-alpha). Zenodo: 2020. Available online: https://zenodo.org/record/3760246#.XrP1Oi4za1U (accessed on 22 April 2020).
25. Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; et al. Stable Baselines. 2018. Available online: https://github.com/hill-a/stable-baselines (accessed on 22 April 2020).
26. Horgan, D.; Quan, J.; Budden, D.; Barth-Maron, G.; Hessel, M.; van Hasselt, H.; Silver, D. Distributed Prioritized Experience Replay. *arXiv* **2018**, arXiv:1803.00933.